STA 601/360 Homework 8

Yifei Wang

07 November, 2019

1. Hoff problem 8.3 Hierarchical modeling: The files `school1.dat` through `school8.dat` give weekly hours spent on homework for students sampled from eight different schools. Obtain posterior distributions for the true means for the eight different schools using a hierarchical normal model with the following prior parameters:

$$\mu_0 = 7,\ \gamma_0^2 = 5,\ \tau_0^2 = 10,\ \eta_0 = 2,\ \sigma_0^2 = 15,\ \nu_0 = 2$$

```
school1 = scan("school1.dat")
school2 = scan("school2.dat")
school3 = scan("school3.dat")
school4 = scan("school4.dat")
school5 = scan("school5.dat")
school6 = scan("school6.dat")
school7 = scan("school7.dat")
school8 = scan("school8.dat")
school = list(school1, school2, school3, school4, school5, school6, school7, school8)
```

Part (a) Run a Gibbs sampling algorithm to approximate the posterior distribution of $\{\theta, \sigma^2, \mu, \tau^2\}$. Assess the convergence of the Markov chain, and find the effective sample size for $\{\sigma^2, \mu, \tau^2\}$. Run the chain long enough so that the effective sample sizes are all above 1,000.

```
library(coda)
### priors and data
mu0 = 7
gamma20 = 5
tau20 = 10
eta0 = 2
sigma20 = 15
nu0 = 2

m = length(school)
n = unlist(map(school, length))
ybar = unlist(map(school, mean))
S = 5000

### initiation
set.seed(10)
mu.mcmc = array(dim=c(S))
tau2.mcmc = array(dim=c(S))
sigma2.mcmc = array(dim=c(S))
theta.mcmc = array(dim=c(S, m))
THETA = ybar
MU = mean(THETA)
TAU2 = var(THETA)
SIGMA2 = mean(unlist(map(school, var)))

### gibbs sampler
```

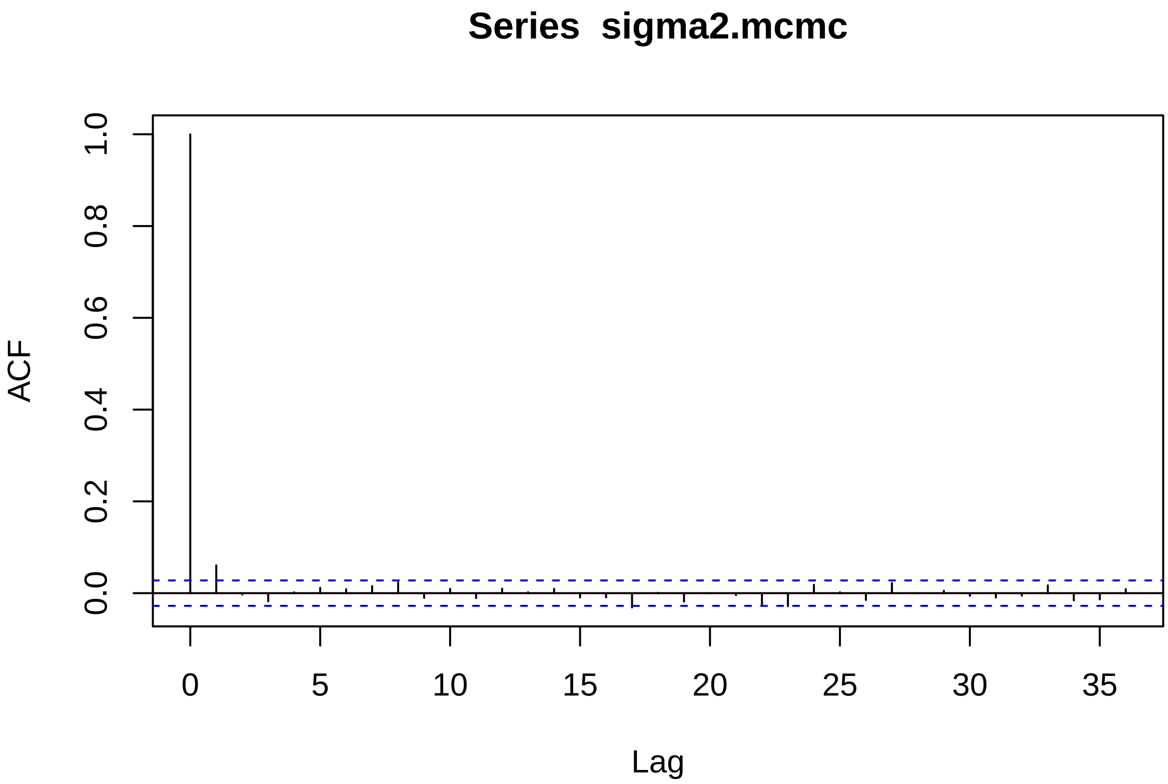
```
for (s in 1:S) {
  # THETA
  for (j in 1:m) {
    vTHETA = 1/(n[j]/SIGMA2 + 1/TAU2)
    eTHETA = vTHETA * (ybar[j]*n[j]/SIGMA2 + MU/TAU2)
    THETA[j] = rnorm(1, eTHETA, sqrt(vTHETA))
  }

  # SIGMA2
  nun = nu0 + sum(n)
  sn = nu0 * sigma20
  for (j in 1:m) {sn = sn + sum((school[[j]] - THETA[j])^2) }
  SIGMA2 = 1 / rgamma(1, nun/2, sn/2)

  # MU
  vMU = 1/(m/TAU2 + 1/gamma20)
  eMU = vMU * (m*mean(THETA)/TAU2 + mu0/gamma20)
  MU = rnorm(1, eMU, sqrt(vMU))

  # TAU2
  etan = eta0 + m
  tn = eta0*tau20 + sum((THETA - MU)^2)
  TAU2 = 1 / rgamma(1, etan/2, tn/2)

  # store results
  theta.mcmc[s, ] = THETA
  sigma2.mcmc[s] = SIGMA2
  mu.mcmc[s] = MU
  tau2.mcmc[s] = TAU2
}
```

```
library(grid)
library(gridExtra)
```

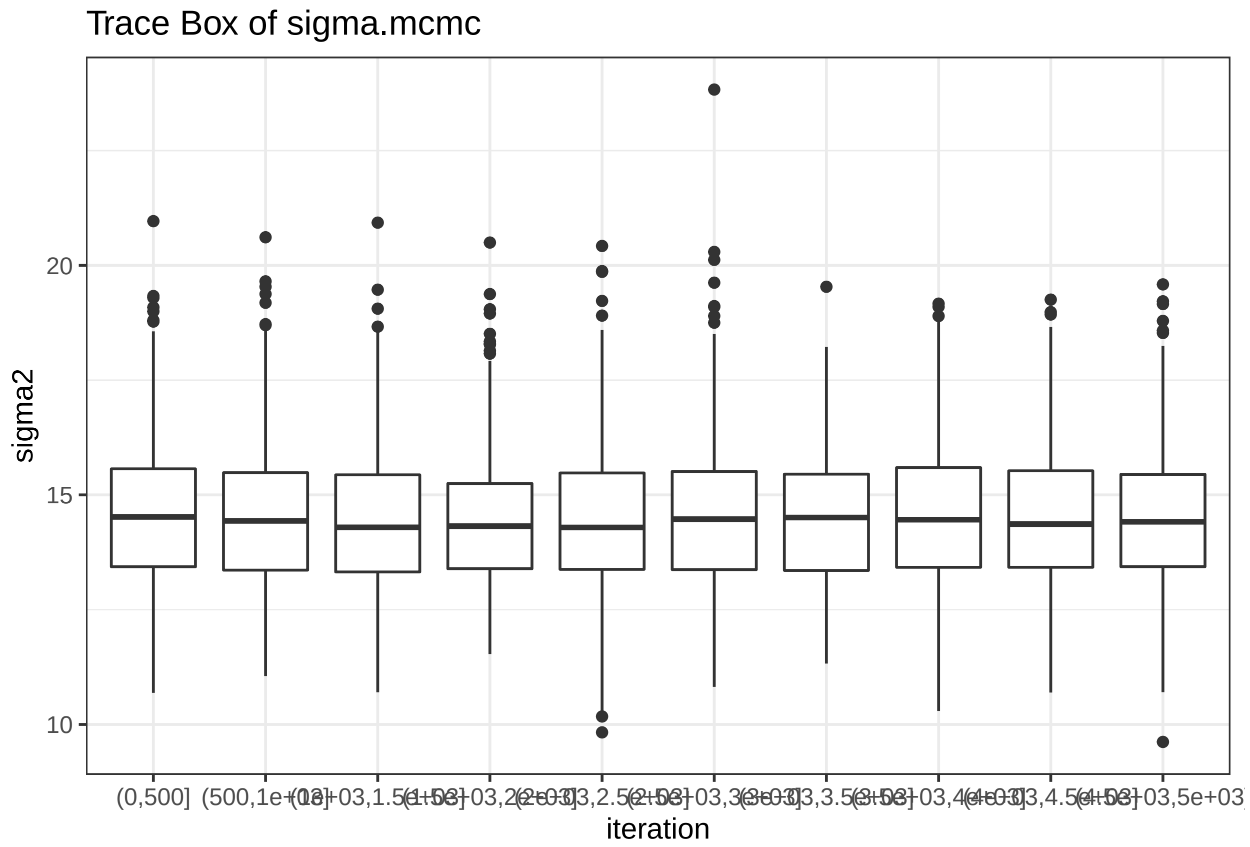
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

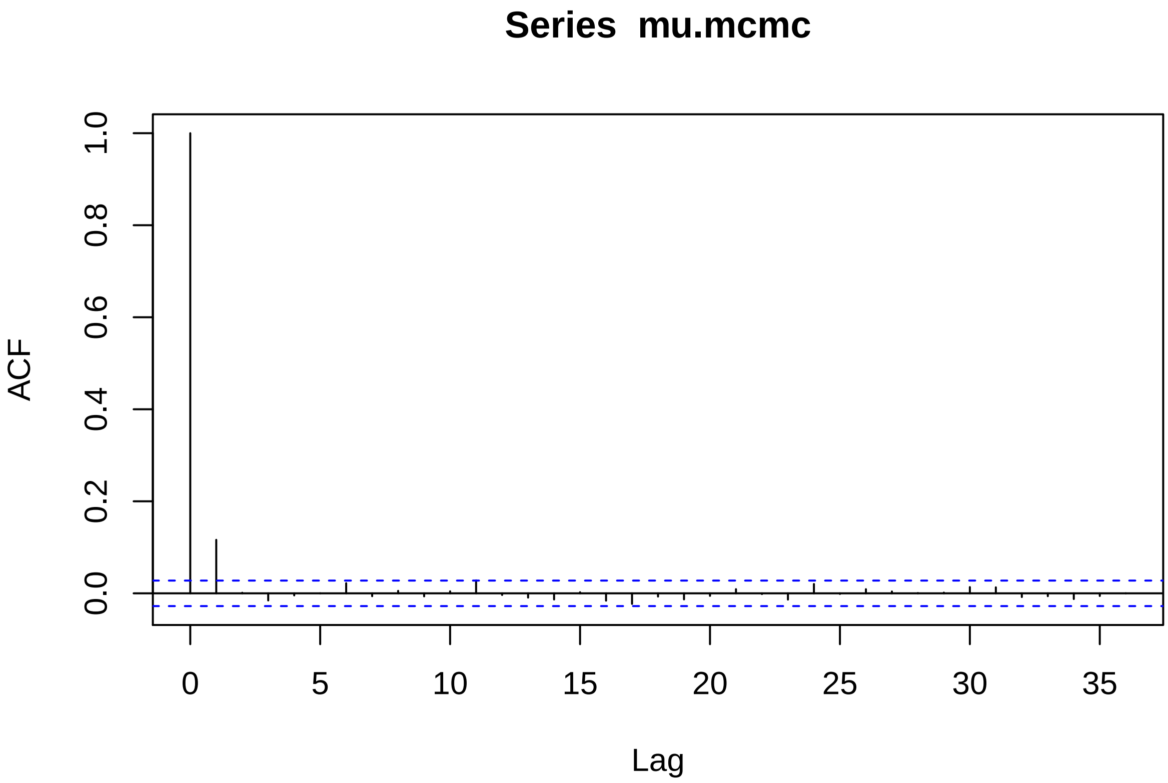
```
### convergence of Markov chains

# sigma2
acf(sigma2.mcmc)
```



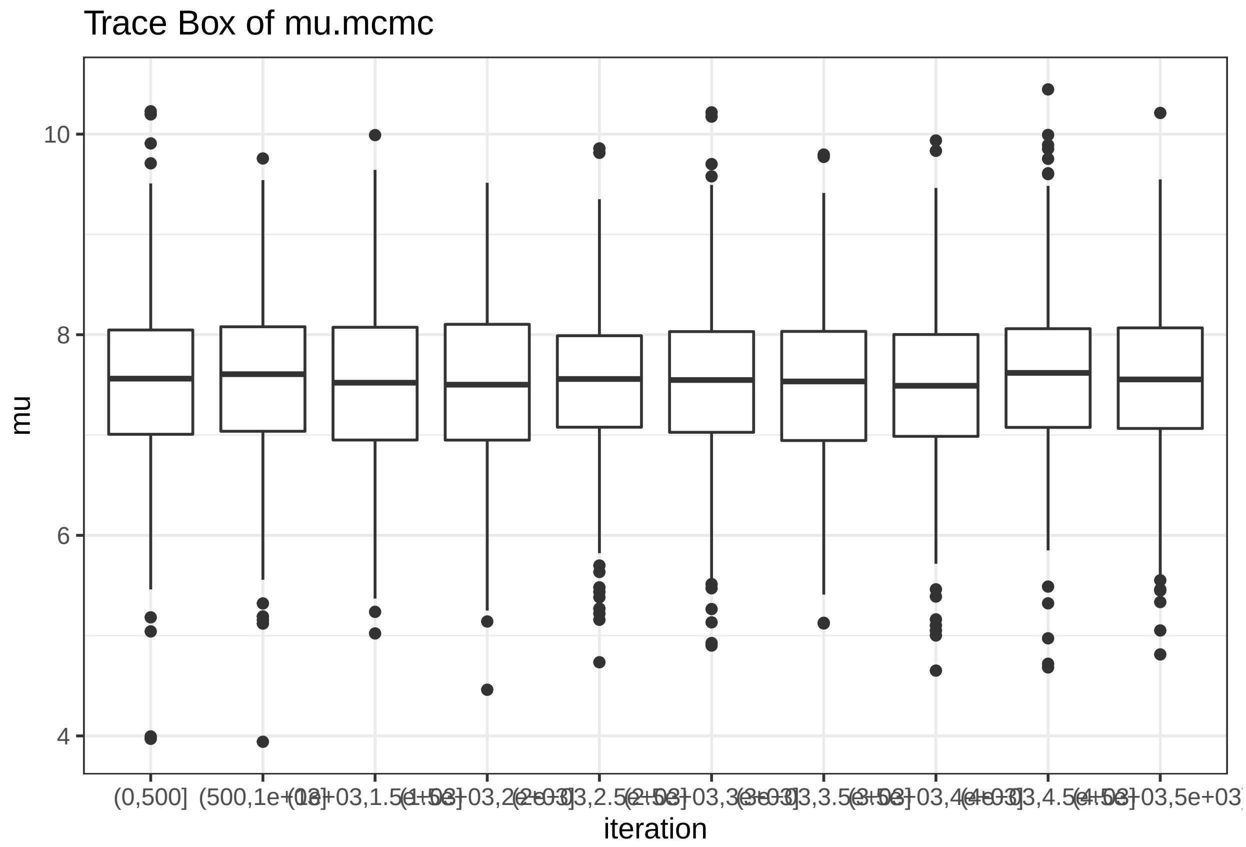
```
sigma2_df = data.frame(sigma2 = sigma2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(sigma2_df, aes(y = sigma2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigma.mcmc") +
  xlab("iteration")
```



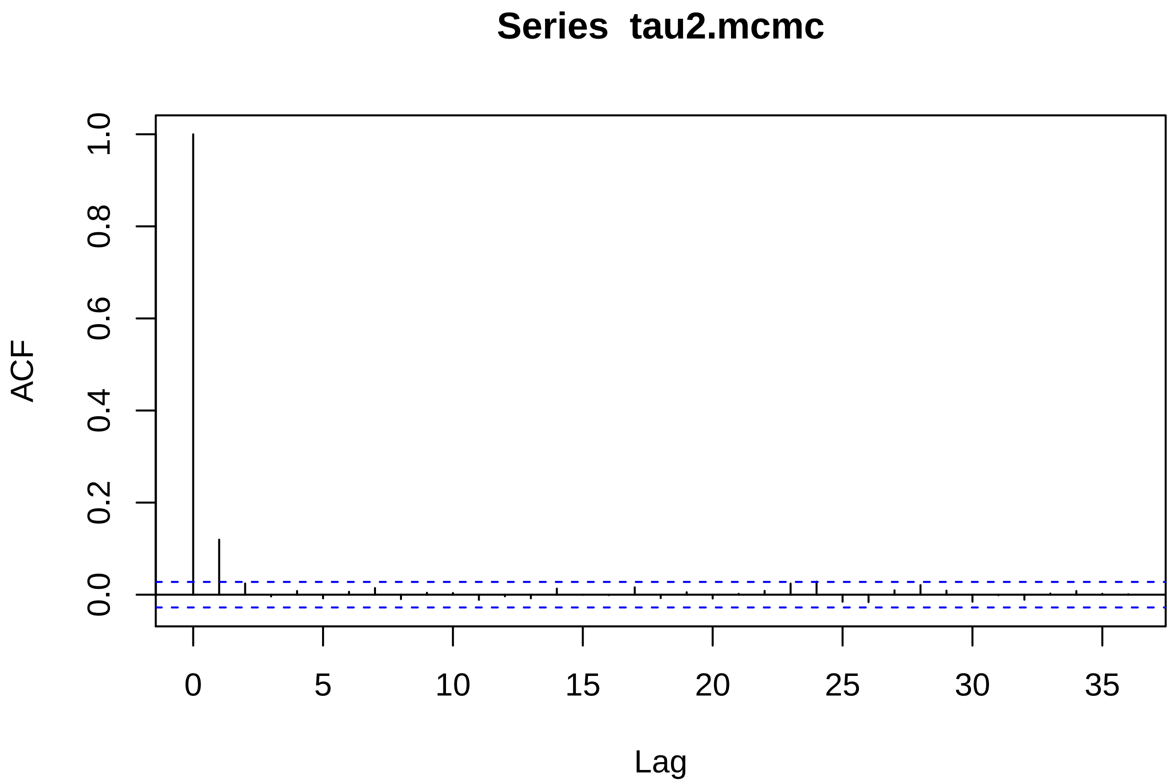
```
# mu
acf(mu.mcmc)
```



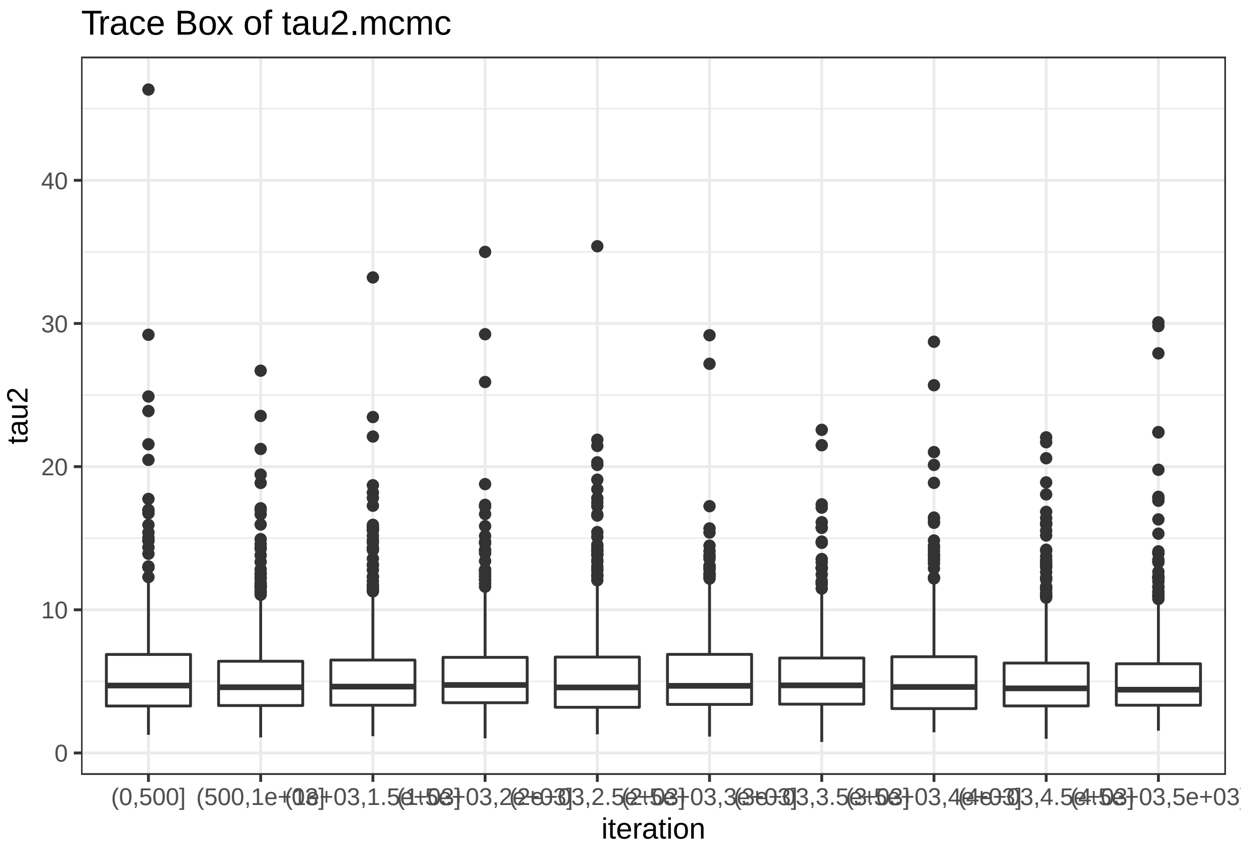
```r
mu_df = data.frame(mu = mu.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(mu_df, aes(y = mu, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of mu.mcmc") +
  xlab("iteration")
```



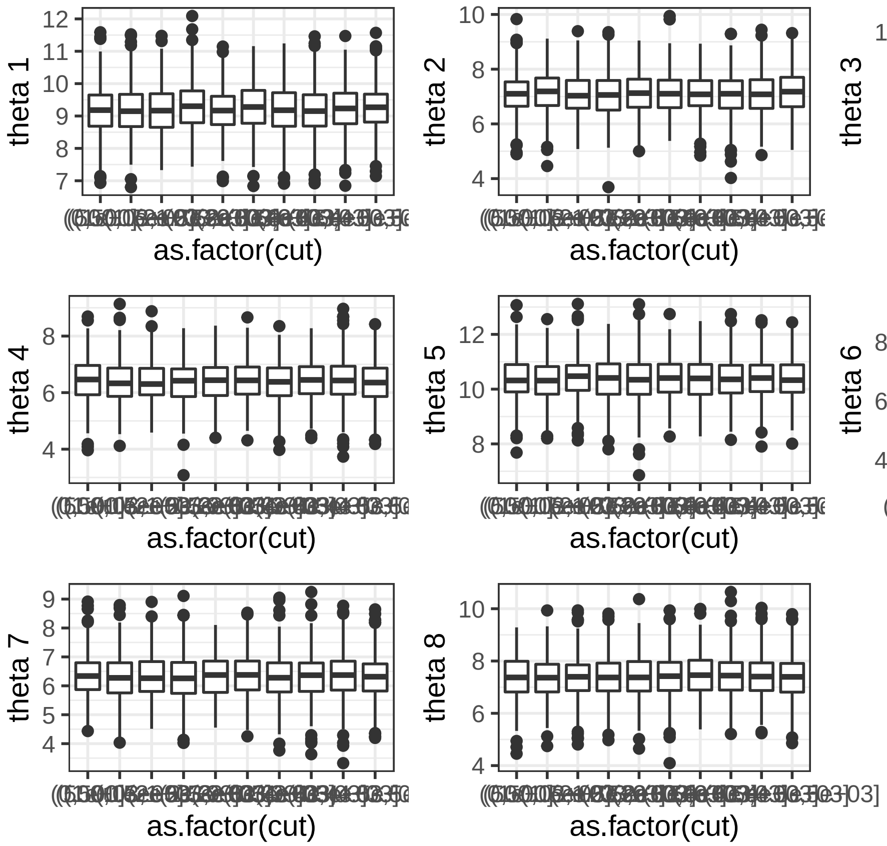
```r
# tau2
acf(tau2.mcmc)
```



```
tau2_df = data.frame(tau2 = tau2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(tau2_df, aes(y = tau2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of tau2.mcmc") +
  xlab("iteration")
```



```
# theta
par(mfrow = c(3, 3), mar=c(1,1,1,1))
for (j in 1:m) {
  acf(theta.mcmc[, j])
}

p <- list()
for (j in 1:m) {
  theta_df = data.frame(theta = theta.mcmc[, j],
                        cut = cut(1:S, breaks = seq(0, S, S/10)))
  p[[j]] = ggplot(theta_df, aes(y = theta, x = as.factor(cut))) +
    geom_boxplot() +
    ylab(paste("theta", j))
}
do.call(grid.arrange, p)
```



```r
### effective sample sizes
print(paste("Effective size of sigma2:", effectiveSize(sigma2.mcmc)))
```

```
## [1] "Effective size of sigma2: 4427.49885912669"
```

```r
print(paste("Effective size of mu:", effectiveSize(mu.mcmc)))
```

```
## [1] "Effective size of mu: 3957.4504563521"
```

```r
print(paste("Effective size of tau2:", effectiveSize(tau2.mcmc)))
```

```
## [1] "Effective size of tau2: 3930.98814694888"
```

```r
for (j in 1:m) {
  print(paste("Effective size of theta", j, ":", effectiveSize(theta.mcmc[, j])))
}
```

```
## [1] "Effective size of theta 1 : 4568.42273313246"
## [1] "Effective size of theta 2 : 4999.99999999999"
## [1] "Effective size of theta 3 : 5241.70162834763"
## [1] "Effective size of theta 4 : 5000.00000000001"
## [1] "Effective size of theta 5 : 4310.67151246706"
## [1] "Effective size of theta 6 : 4712.15814147604"
## [1] "Effective size of theta 7 : 4942.70635768778"
## [1] "Effective size of theta 8 : 5000.00000000001"
```

The acf plots indicate that all the parameters have a small number of autocorrelation. The trace plots indicate that all the parameters have reached the convergence. Also, the effective sample sizes of all parameters are greater than 1000, when the lenght of chain is 5000.

Part (b) Compute posterior means and 95% confidence regions for $\{\sigma^2, \mu, \tau^2\}$. Also, compare the posterior densities to the prior densities, and discuss what was learned from the data.

```
### posterior mean

# sigma2
quantile(sigma2.mcmc, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 11.72729 17.91366
```

```
mean(sigma2.mcmc)
```

```
## [1] 14.50893
```

```
# mu
quantile(mu.mcmc, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5% 
## 5.853528 9.060198
```

```
mean(mu.mcmc)
```

```
## [1] 7.524882
```

```
# tau2
quantile(tau2.mcmc, probs = c(0.025, 0.975))
```

```
##      2.5%     97.5% 
##  1.926337 14.188374
```

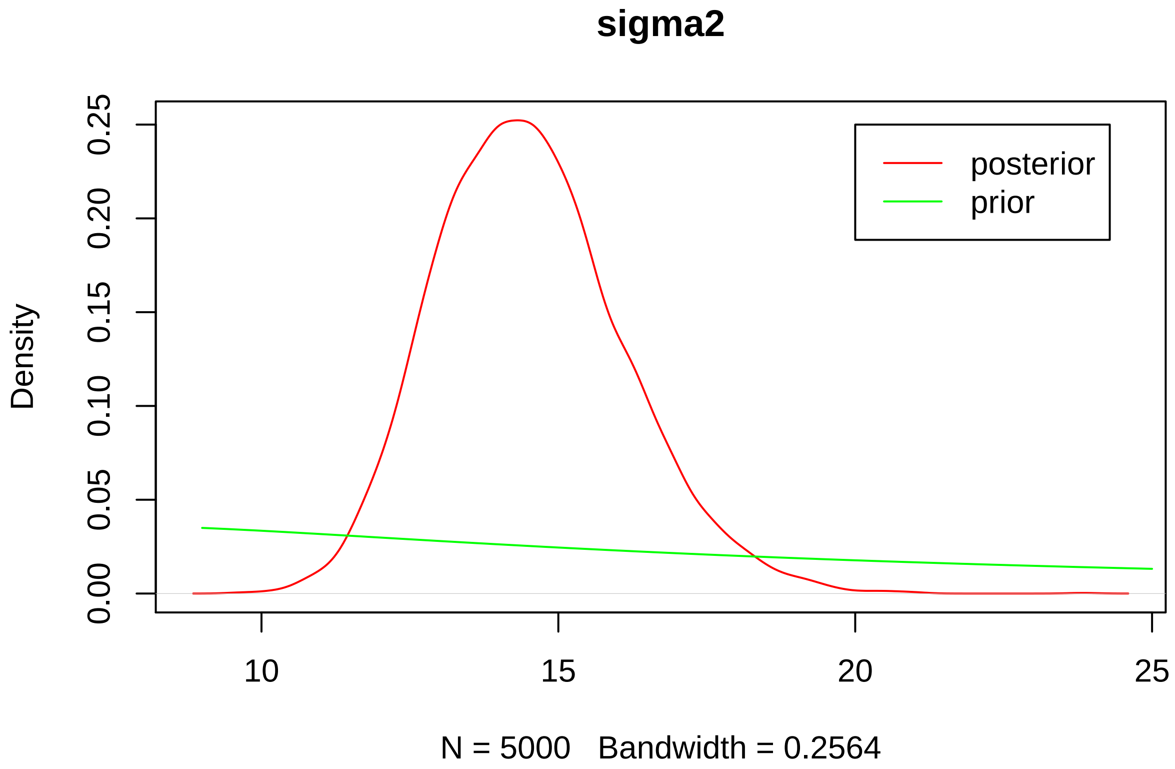
```
mean(tau2.mcmc)
```

```
## [1] 5.50623
```

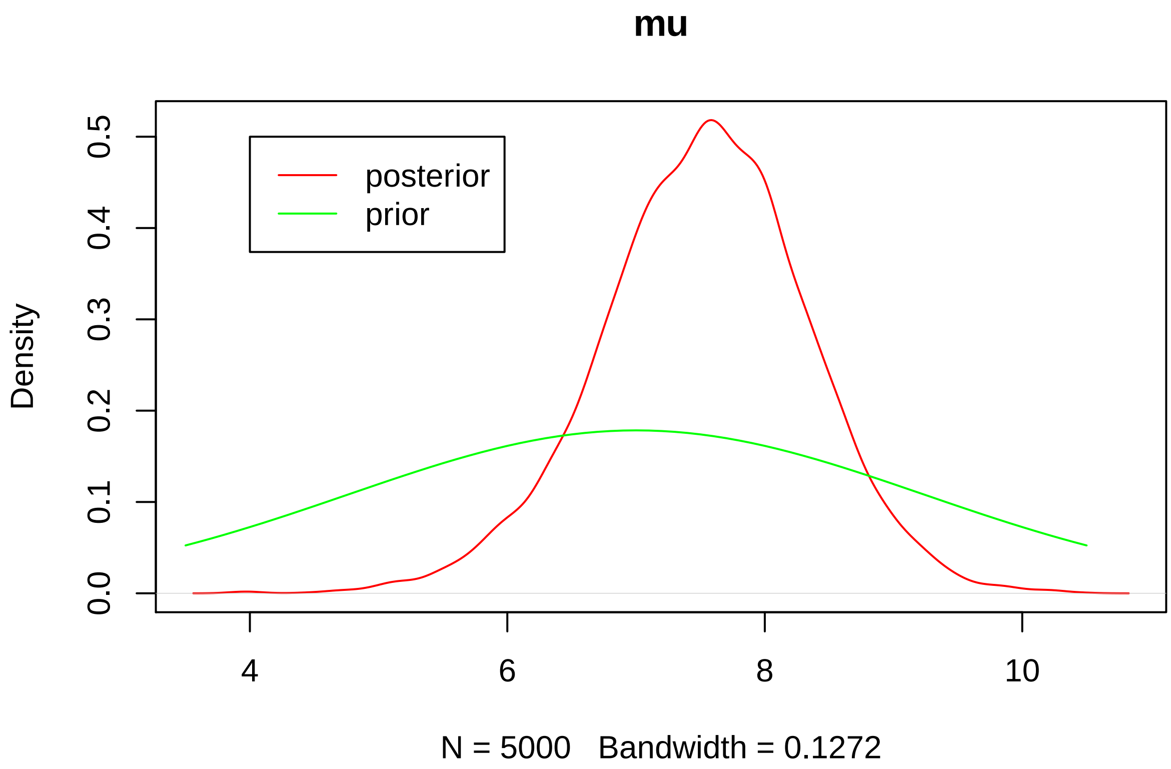
```
library(MCMCpack)
```

```
## Loading required package: MASS

## 
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
## 
##     select

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##
```

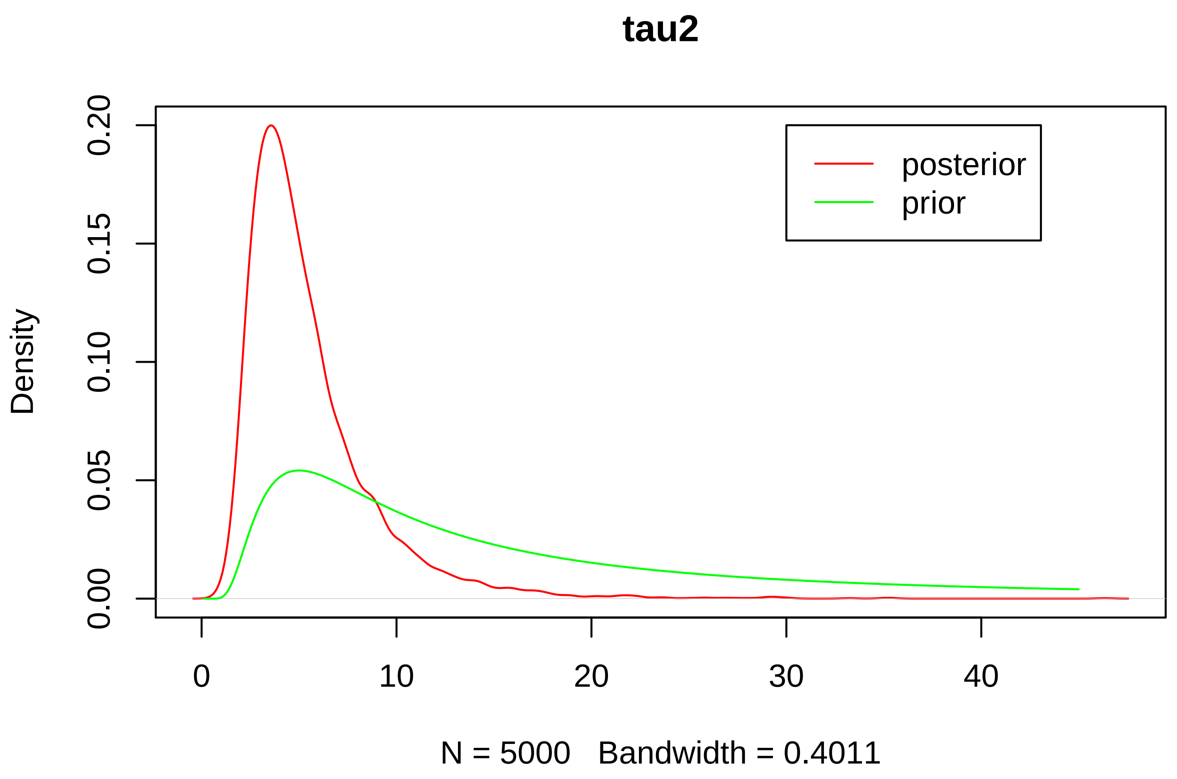
```
# sigma2
sigma2.prior = dinvgamma(seq(9, 25, by = 0.1), nu0/2, nu0*sigma20/2)
plot(density(sigma2.mcmc), col="red", main="sigma2")
lines(seq(9, 25, by = 0.1), sigma2.prior, col="green")
legend(x=20, y = 0.25,
       legend = c("posterior", "prior"),
       lty=c(1,1),
       col=c("red", "green"))
```



```
# mu
mu.prior = dnorm(seq(3.5, 10.5, by = 0.1), mu0, sqrt(gamma20))
plot(density(mu.mcmc), col="red", main="mu")
lines(seq(3.5, 10.5, by = 0.1), mu.prior, col="green")
legend(x=4, y = 0.5,
       legend = c("posterior", "prior"),
       lty=c(1,1),
       col=c("red", "green"))
```



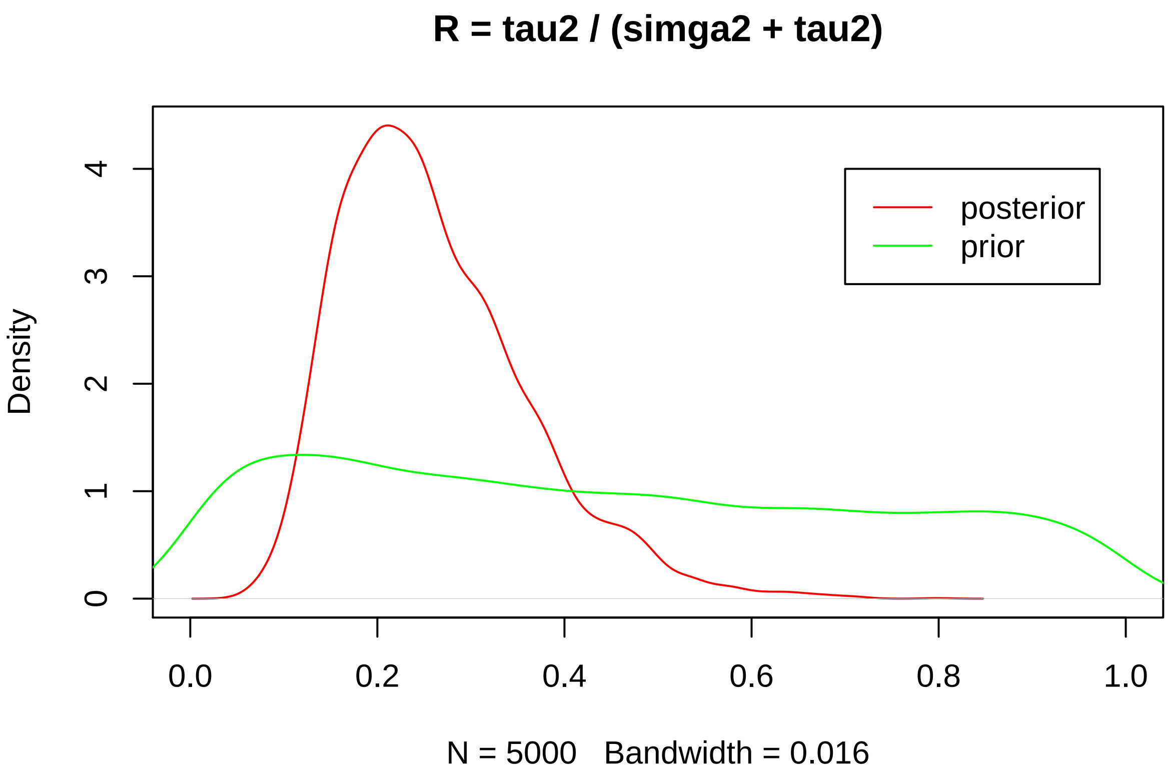
```
# tau2
tau2.prior = dinvgamma(seq(0, 45, by = 0.1), eta0/2, eta0*tau20/2)
plot(density(tau2.mcmc), col="red", main="tau2")
lines(seq(0, 45, by = 0.1), tau2.prior, col="green")
legend(x=30, y = 0.2,
       legend = c("posterior", "prior"),
       lty=c(1,1),
       col=c("red", "green"))
```



For all the parameters, their prior are more spread than the posterior. The posterior of τ^2 and μ have similar shape with their priors, but more centered. The posterior of σ^2 is way more centered than its prior. The model learn from the information in data to strengthen the beliefs of the posterior parameters, making their posterior distribution more centered at the posterior means (less variance).

Part (c) Plot the posterior density of $R = \frac{\tau^2}{\sigma^2+\tau^2}$ and compare it to a plot of the prior density of R. Describe the evidence for between-school variation.

```
tau2.prior = 1/rgamma(S, eta0/2, eta0*tau20/2)
sigma2.prior = 1/rgamma(S, nu0/2, nu0*sigma20/2)
R.prior = tau2.prior / (sigma2.prior + tau2.prior)
R.posterior = tau2.mcmc / (sigma2.mcmc + tau2.mcmc)

plot(density(R.posterior), xlim = c(0, 1), col="red", main="R = tau2 / (simga2 + tau2)")
lines(density(R.prior), col="green")
legend(x=0.7, y = 4,
       legend = c("posterior", "prior"),
       lty=c(1,1),
       col=c("red", "green"))
```



The statistcs R represent the ratio between between-school variance and the overall variance, which thus shows the evidence of the between-school variance and the within-school variance. The prior distribution of R shows that the prior between-school variance and within-school variance are roughly the same. Posterior density of R, however, shows that between-school variance takes up about 20% of the overall variance, which further indicates that within-school variance is a greater source of the overall variance.

Part (d) Obtain the posterior probability that θ_7 is smaller than θ_6, as well as the posterior probability that θ_7 is the smallest of all the θ's.

```
### Pr(theta7 < theta6)
mean(theta.mcmc[, 7] < theta.mcmc[, 6])
```

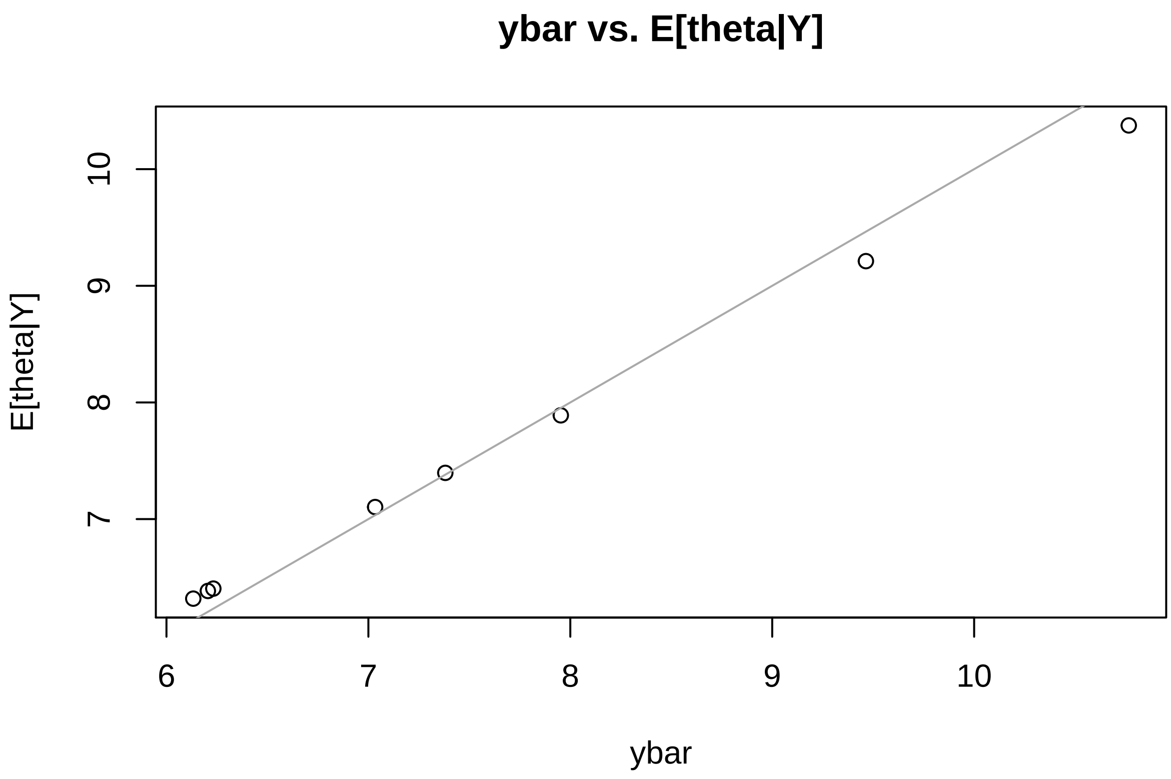
```
## [1] 0.521
```

```
### Pr(min(theta) = theta7)
runningsum = 0
for (i in 1:S) {
  runningsum = runningsum + all(theta.mcmc[i, 7] <= theta.mcmc[i, ])
}
runningsum / S
```

```
## [1] 0.3262
```

Part (e) Plot the sample averages $\bar{y}_1, \ldots, \bar{y}_8$ against the posterior expectations of $\theta_1, \ldots, \theta_8$, and describe the relationship. Also compute the sample mean of all observations and compare it to the posterior mean of μ.

```
plot(ybar, apply(theta.mcmc, 2, mean), main = "ybar vs. E[theta|Y]", ylab = "E[theta|Y]")
lines(seq(6, 11), seq(6, 11), col = "darkgray")
```



Sample averages and the posterior expectations have a linear relationship. Also we can observe a shrinkage effect there: groups with bigger averages are below the grey identical line and groups with smaller averages are above the grey identical line. The posterior expectations of groups with large or small extreme averages would be pulled towards the overall mean $\mu = 7.52$, and the larger or smaller the group averages are, the bigger the pulling (shrinkage) effects.

```
# sample mean
sum(unlist(map(school, sum))) / sum(n)
```

```
## [1] 7.691278
```

```
# posterior mean of mu
mean(mu.mcmc)
```

```
## [1] 7.524882
```

The sample mean of all observations is really close to the posterior mean of μ.

2. Hoff problem 9.1 Extrapolation: The file `swim.dat` contains data on the amount of time, in seconds, it takes each of four high school swimmers to swim 50 yards. Each swimmer has six times, taken on a biweekly basis.

```
swim = read.table("swim.dat")
```

Part (a) Perform the following data analysis for each swimmer separately:

i. Fit a linear regression model of swimming time as the response and week as the explanatory variable. To formulate your prior, use the information that competitive times for this age group generally range from 22 to 24 seconds.

Suppose we have $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ and we let $x \in \{0, 2, 4, 6, 8, 10\}$ so that the "week-effect" has no effect on the first week. Since the population generally range from 22 to 24, we can assume the population mean is 23 and population standard deviance is 0.5. Then the changes of time should be in range $(-2, 2)$ for all x in $[0, 10]$. Thus we have $-2 < \beta_0 < 2$ and $-0.4 < \beta_1 < 0.4$ with high probability. Thus we have the variance of β_0 and β_1 to be 1ˆ2 and 0.2ˆ2 respectively.

```r
library(mvtnorm)
### data
Y = swim
X = cbind(rep(1, 6), seq(0, 10, by = 2))

### priors
b0 = c(23, 0)
Sigma0 = matrix(data = c(1, 0, 0, 0.2^2), nrow = 2, ncol = 2)
nu0 = 1
s20 = 1
m = 4
n = 6
p = 2
S = 5000

### initiation
set.seed(10)
beta.mcmc = array(dim = c(m, S, p))
sigma2.mcmc = array(dim = c(m, S))

### gibbs sampler
for (i in 1:m) {
  BETA = b0
  y = matrix(unlist(Y[i, ]))
  for (s in 1:S) {
    # SIGMA2
    nun = nu0 + n
    s2n = nu0 * s20 + t(y - X %*% BETA) %*% (y - X %*% BETA)
    SIGMA2 = 1/rgamma(1, nun/2, s2n/2)

    # BETA
    vBETA = solve(solve(Sigma0) + t(X) %*% X / SIGMA2)
    eBETA = vBETA %*% (solve(Sigma0) %*% b0 + t(X) %*% y / SIGMA2)
    BETA = t(rmvnorm(1, eBETA, vBETA))

    # update
    sigma2.mcmc[i, s] = SIGMA2
    beta.mcmc[i, s, ] = BETA
  }
}
```

ii. For each swimmer j, obtain a posterior predictive distribution for Y_j^*, their time if they were to swim two weeks from the last recorded time.

```r
### posterior predictive distribution
Y.pred.mcmc = array(dim = c(m, S))
for (i in 1:m) {
  for (s in 1:S) {
    beta.posterior = beta.mcmc[i, s, ]
    sigma2.posterior = sigma2.mcmc[i, s]
    Y.pred.mcmc[i, s] = rnorm(1, t(beta.posterior) %*% c(1, 12), sqrt(sigma2.posterior))
  }
}
```

Part (b) The coach of the team has to decide which of the four swimmers will compete in a swimming meet in two weeks. Using your predictive distributions, compute Pr $\left(Y_j^* = \max\{Y_1^*, \ldots, Y_4^*\} \mid Y\right)$ for each swimmer j, and based on this make a recommendation to the coach.

```
### standard variance
apply(Y.pred.mcmc, 1, sd)
```

```
## [1] 0.7490079 0.7446779 0.7437557 0.7451182
```

```
### Pr(Yj* = max{Y1*,...,Y4*} | Y)
Y.pred.max = apply(Y.pred.mcmc, 2, max)

# Y1*
mean(Y.pred.mcmc[1, ] == Y.pred.max)
```

```
## [1] 0.0706
```

```
# Y2*
mean(Y.pred.mcmc[2, ] == Y.pred.max)
```

```
## [1] 0.4566
```

```
# Y3*
mean(Y.pred.mcmc[3, ] == Y.pred.max)
```

```
## [1] 0.1168
```

```
# Y4*
mean(Y.pred.mcmc[4, ] == Y.pred.max)
```

```
## [1] 0.356
```

In this scenario, all swimmers have similar posterior predictive standard deviation. I recommend the coach to pick the first swimmer because he/she has the smallest posterior predictive probablity to become the slowest one among all of these 4 swimmers. Also, we could see that swimmer 1 has the greatest posterior predictive probability (and it almost reach 0.5) to become the fastest one among all of these 4 swimmers.

```
### Pr(Yj* = max{Y1*,...,Y4*} | Y)
Y.pred.min = apply(Y.pred.mcmc, 2, min)

# Y1*
mean(Y.pred.mcmc[1, ] == Y.pred.min)
```

```
## [1] 0.4978
```

```
# Y2*
mean(Y.pred.mcmc[2, ] == Y.pred.min)
```

```
## [1] 0.075
```

```
# Y3*
mean(Y.pred.mcmc[3, ] == Y.pred.min)
```

```
## [1] 0.3222
```

```
# Y4*
mean(Y.pred.mcmc[4, ] == Y.pred.min)
```

```
## [1] 0.105
```

3. Math Problem: Derive the Gibbs sampler for the "Standard Bayes" ANOVA:

$$Y_{ijk} \sim N(\theta_{ij}, \sigma^2)$$

where

$$\theta_{ij} = \mu + a_i + b_i + (ab)_{ij}$$

and

$$\begin{aligned} \mu &\sim N(0, \sigma_\mu^2) \\ a_1, \ldots, a_{m_1} &\sim N(0, \sigma_a^2) \\ b_1, \ldots, b_{m_2} &\sim N(0, \sigma_b^2) \\ (ab)_1, \ldots, (ab)_{m_1 m_2} &\sim N(0, \sigma_{ab}^2) \end{aligned}$$

Find appropreate priors for the variance components. Analyse the dataset ToothGrowth in R using this approach, treating "Supplement type" and "Dose" as factors (note that "Dose" is coded as numeric, make sure you do not use it as such). Provide justification for prior parameter choice and report convergence information as well information about shrinkage of the θ_{ij} means. What makes the teeth of guinea pigs grow?

(The response is the length of odontoblasts (cells responsible for tooth growth) in 60 guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, orange juice or ascorbic acid (a form of vitamin C and coded as VC). https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/ToothGrowth.html)

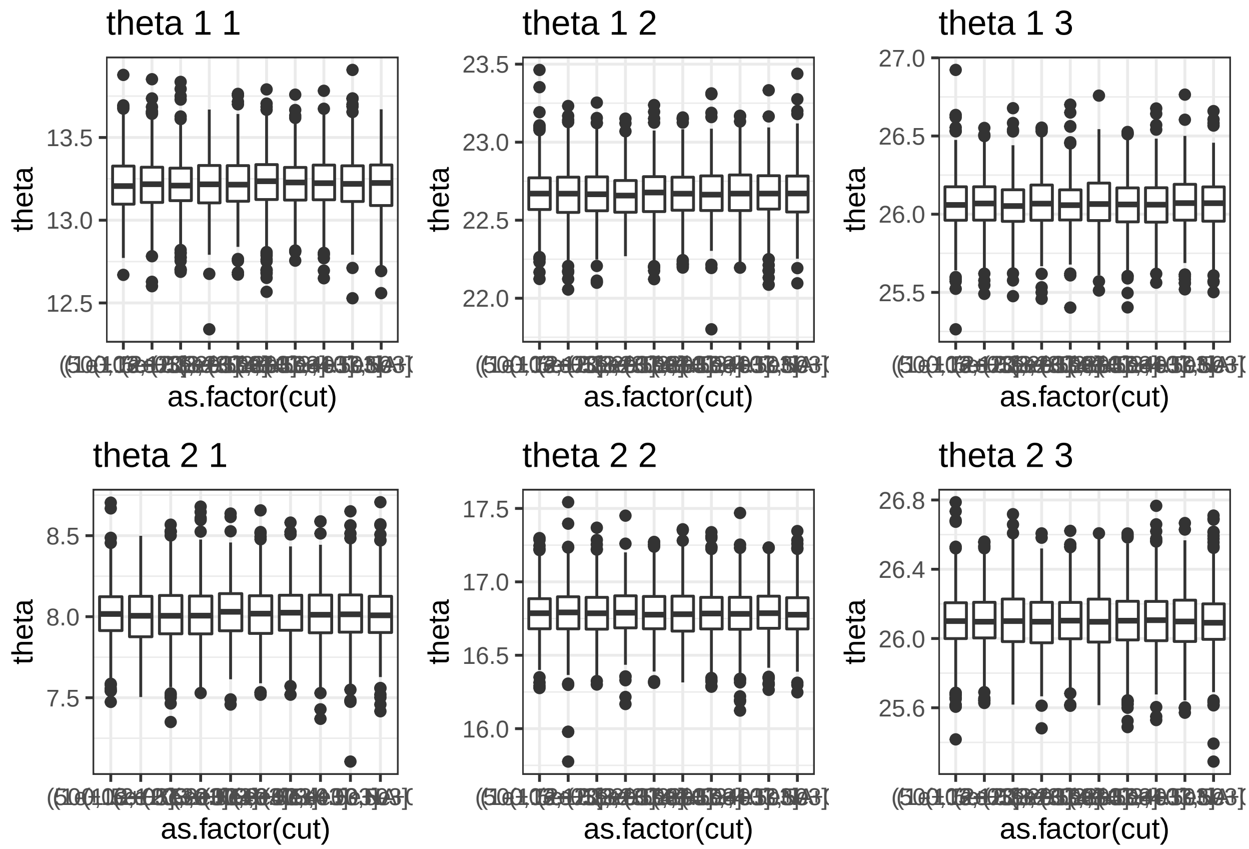
The math proof of all the full conditionals see the other scanned PDF file.

The prior parameters we need to determine are $\nu_0, \sigma_0^2, \nu_\mu, \sigma_{\mu_0}^2, \nu_{a_0}, \sigma_{a_0}^2, \nu_{b_0}, \sigma_{b_0}^2, \nu_{ab_0}, \sigma_{ab_0}^2$. We have little prior information for this experiment, so we set $\nu_0 = \nu_\mu = \nu_{a_0} = \nu_{b_0} = \nu_{ab_0} = 1$. The overall mean μ should not move around too much, so we let $\sigma_{\mu_0}^2 = 10$. Group means should move a little bit more than the overall mean, and the dose have more effect than supp, so we let $\sigma_{a_0}^2 = 20$, $\sigma_{b_0}^2 = 40$ and $\sigma_{ab_0}^2 = 50$. Finally the variance of all observations should the the biggest one, so we set $\sigma_0^2 = 100$, greater than the samples variance.

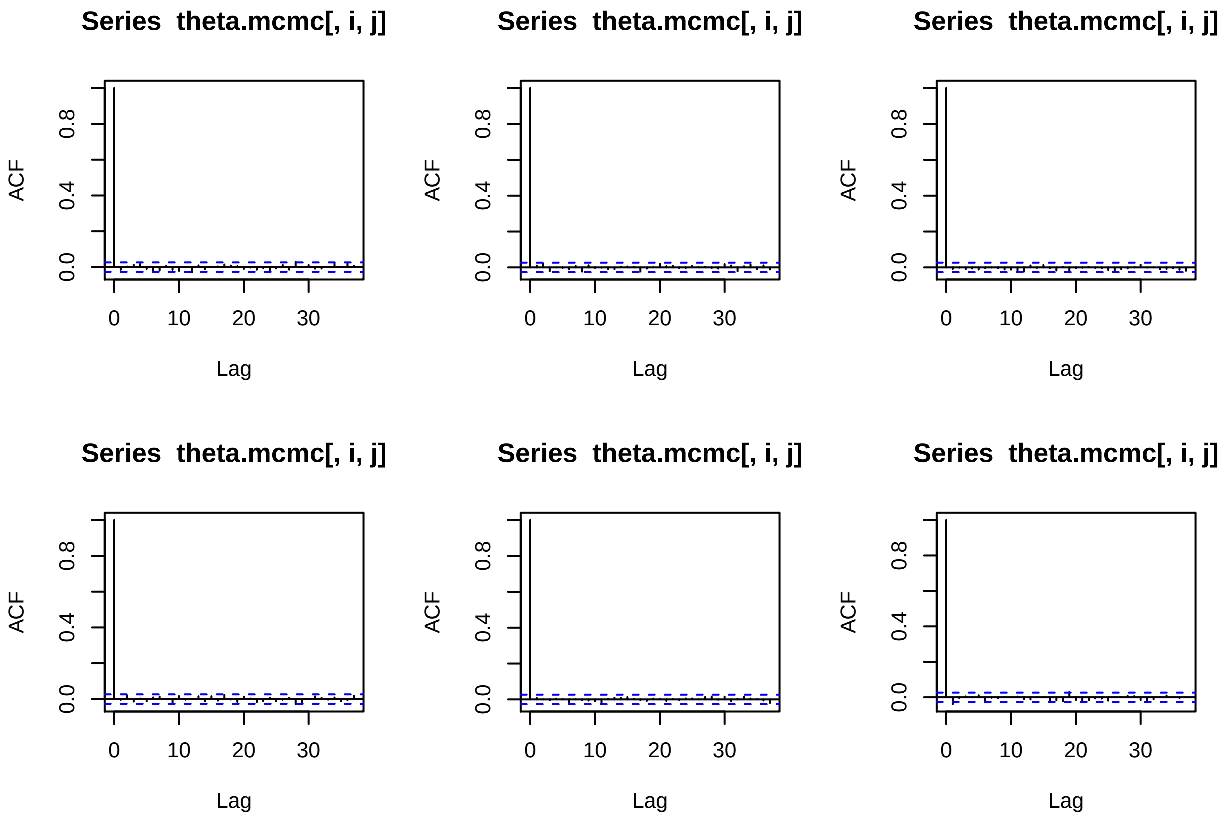
```
### data
data("ToothGrowth")
data = ToothGrowth
data$dose = as.factor(ToothGrowth$dose)
# n_group = table(data$supp, data$dose)
l1 = levels(data$supp)
l2 = levels(data$dose)
m1 = length(l1)
m2 = length(l2)
n = nrow(data)
na = c(sum(data$supp == l1[1]), sum(data$supp == l1[2]))
nb = c(sum(data$dose == l2[1]), sum(data$dose == l2[2]), sum(data$dose == l2[3]))
nab = matrix(c(sum(data$supp == l1[1] & data$dose == l2[1]),
               sum(data$supp == l1[2] & data$dose == l2[1]),
               sum(data$supp == l1[1] & data$dose == l2[2]),
               sum(data$supp == l1[2] & data$dose == l2[2]),
               sum(data$supp == l1[1] & data$dose == l2[3]),
               sum(data$supp == l1[2] & data$dose == l2[3])), nrow = m1, ncol = m2)
S = 5500

### priors
nu0 = numu0 = nua0 = nub0 = nuab0 = 1
s20 = 100
smu20 = 10
sa20 = 20
sb20 = 40
sab20 = 50

### setup
sigma2.mcmc = array(dim = c(S))
sigmamu2.mcmc = array(dim = c(S))
sigmaa2.mcmc = array(dim = c(S))
sigmab2.mcmc = array(dim = c(S))
sigmaab2.mcmc = array(dim = c(S))
mu.mcmc = array(dim = c(S))
a.mcmc = array(dim = c(S, m1))
b.mcmc = array(dim = c(S, m2))
ab.mcmc = array(dim = c(S, m1, m2))
update_data = function(data, MU, A, B, AB) {
  data$MU = rep(MU, n)
  data$A[data$supp == "OJ"] = A[1]
  data$A[data$supp == "VC"] = A[2]
  data$B[data$dose == 0.5] = B[1]
  data$B[data$dose == 1] = B[2]
  data$B[data$dose == 2] = B[3]
  data$AB[data$supp == "OJ" & data$dose == 0.5] = AB[1, 1]
  data$AB[data$supp == "OJ" & data$dose == 1] = AB[1, 2]
  data$AB[data$supp == "OJ" & data$dose == 2] = AB[1, 3]
  data$AB[data$supp == "VC" & data$dose == 0.5] = AB[2, 1]
  data$AB[data$supp == "VC" & data$dose == 1] = AB[2, 2]
  data$AB[data$supp == "VC" & data$dose == 2] = AB[2, 3]
  return(data)
}

### initiation
set.seed(5)
SIGMA2 = s20
SIGMAMU2 = smu20
SIGMAA2 = sa20
SIGMAB2 = sb20
SIGMAAB2 = sab20
MU = mean(data$len)
A = rep(0.1, m1)
B = rep(0.1, m2)
AB = matrix(0.1, nrow = m1, ncol = m2)
data = update_data(data, MU, A, B, AB)
```

```r
### gibbs sampler
for (s in 1:S) {
  # MU
  vMU = 1/(1/SIGMAMU2 + n/SIGMA2)
  eMU = vMU * sum(data$len - data$A - data$B - data$AB) / SIGMA2
  MU = rnorm(1, eMU, vMU)
  data = update_data(data, MU, A, B, AB)

  # A
  for (i in 1:m1) {
    cnd = (data$supp == l1[i])
    vA = 1/(1/SIGMAA2 + na[i]/SIGMA2)
    eA = vA * sum((data$len - data$B - data$AB - data$MU)[cnd]) / SIGMA2
    A[i] = rnorm(1, eA, vA)
  }
  data = update_data(data, MU, A, B, AB)

  # B
  for (j in 1:m2) {
    cnd = (data$dose == l2[j])
    vB = 1/(1/SIGMAB2 + nb[j]/SIGMA2)
    eB = vB * sum((data$len - data$A - data$AB - data$MU)[cnd]) / SIGMA2
    B[j] = rnorm(1, eB, vB)
  }
  data = update_data(data, MU, A, B, AB)

  # AB
  for (i in 1:m1) {
    for (j in 1:m2) {
      cnd = (data$supp == l1[i] & data$dose == l2[j])
      vAB = 1/(1/SIGMAAB2 + nab[i, j]/SIGMA2)
      eAB = vAB * sum((data$len - data$A - data$B - data$MU)[cnd]) / SIGMA2
      AB[i, j] = rnorm(1, eAB, vAB)
    }
  }
  data = update_data(data, MU, A, B, AB)

  # SIMGA2
  nun = nu0 + n
  s2n = nu0*s20 + sum(data$len - data$MU - data$A - data$B - data$AB)
  SIGMA2 = 1/rgamma(1, nun/2, s2n/2)

  # SIMGAMU2
  numun = numu0 + 1
  smu2n = numu0*smu20 + MU^2
  SIGMAMU2 = 1/rgamma(1, numun, smu2n)

  # SIMGAA2
  nuan = nua0 + m1
  sa2n = nua0*sa20 + sum(A^2)
  SIGMAA2 = 1/rgamma(1, nuan, sa2n)

  # SIMGAB2
  nubn = nub0 + m2
  sb2n = nub0*sb20 + sum(B^2)
  SIGMAB2 = 1/rgamma(1, nubn, sb2n)

  # SIMGAAB2
  nuabn = nuab0 + m1 * m2
  sab2n = nuab0*sab20 + sum(AB^2)
  SIGMAAB2 = 1/rgamma(1, nuabn, sab2n)

  # update
  sigma2.mcmc[s] = SIGMA2
  sigmamu2.mcmc[s] = SIGMAMU2
  sigmaa2.mcmc[s] = SIGMAA2
  sigmab2.mcmc[s] = SIGMAB2
  sigmaab2.mcmc[s] = SIGMAAB2
  mu.mcmc[s] = MU
  a.mcmc[s, ] = A
  b.mcmc[s, ] = B
  ab.mcmc[s, ,] = AB
}

# compute theta
theta.mcmc = array(dim = c(S, m1, m2))
for (i in 1:m1) {
  for (j in 1:m2) {
    theta.mcmc[, i, j] = mu.mcmc + a.mcmc[, i] + b.mcmc[, j] + ab.mcmc[, i, j]
  }
}
```

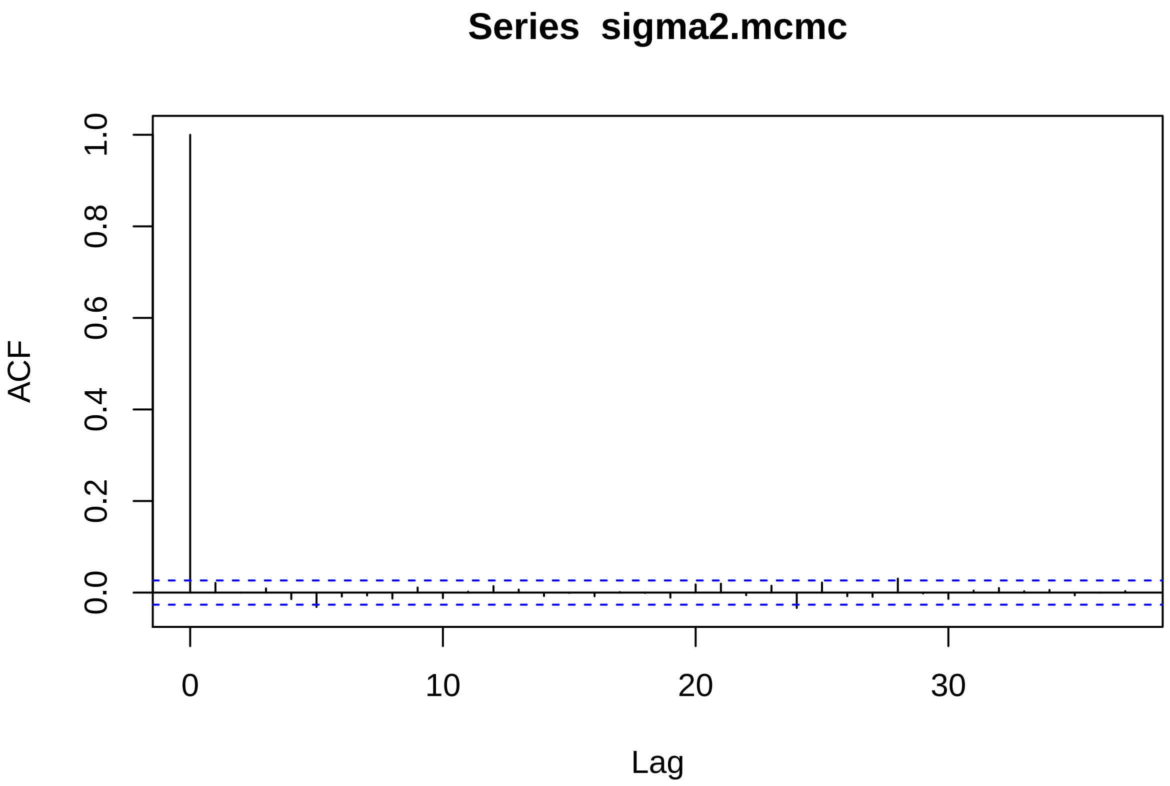
```
### convergence check

# theta
burnin = 500  # drop the few values to avoid bad visual of trace plots
p <- list()
for (i in 1:m1) {
  for (j in 1:m2) {
    index = (i-1)*m2 + j
    theta_df = data.frame(theta = theta.mcmc[(burnin+1):S, i,j],
                          cut = cut((burnin+1):S, breaks = seq(0, S-burnin, (S-burnin)/10)))
    p[[index]] = ggplot(theta_df, aes(y = theta, x = as.factor(cut))) +
      geom_boxplot() +
      ggtitle(paste("theta", i, j))
  }
}
do.call(grid.arrange, c(p, nrow = 2, ncol = 3))
```



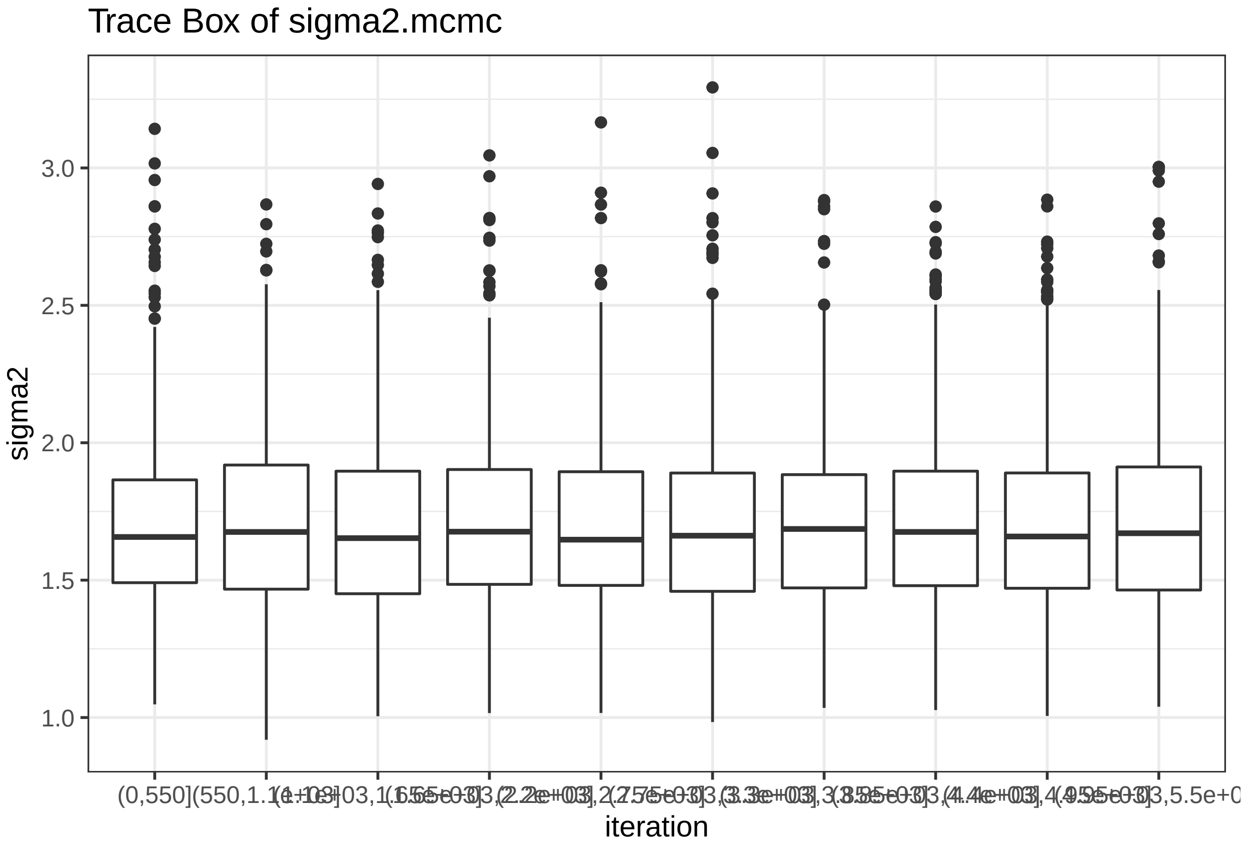
```r
par(mfrow = c(2, 3))
for (i in 1:m1) {
  for (j in 1:m2) {
    acf(theta.mcmc[, i, j])
  }
}
```



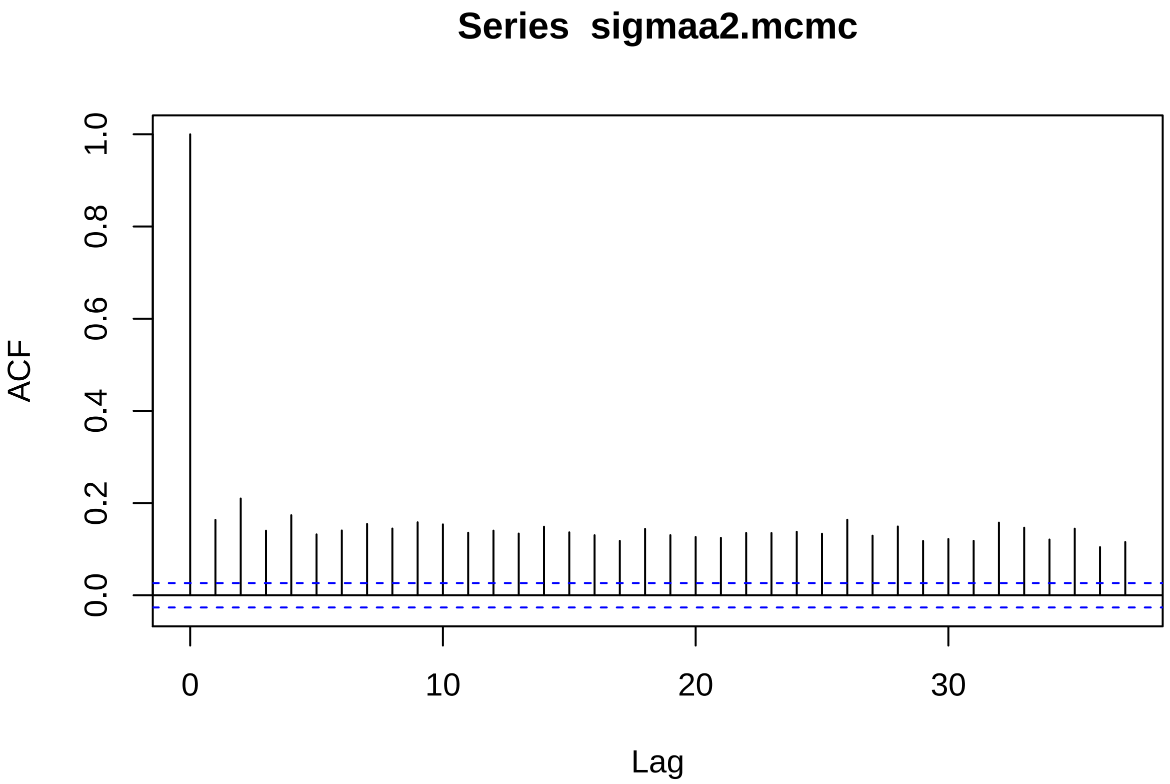
```r
# sigma2
acf(sigma2.mcmc)
```



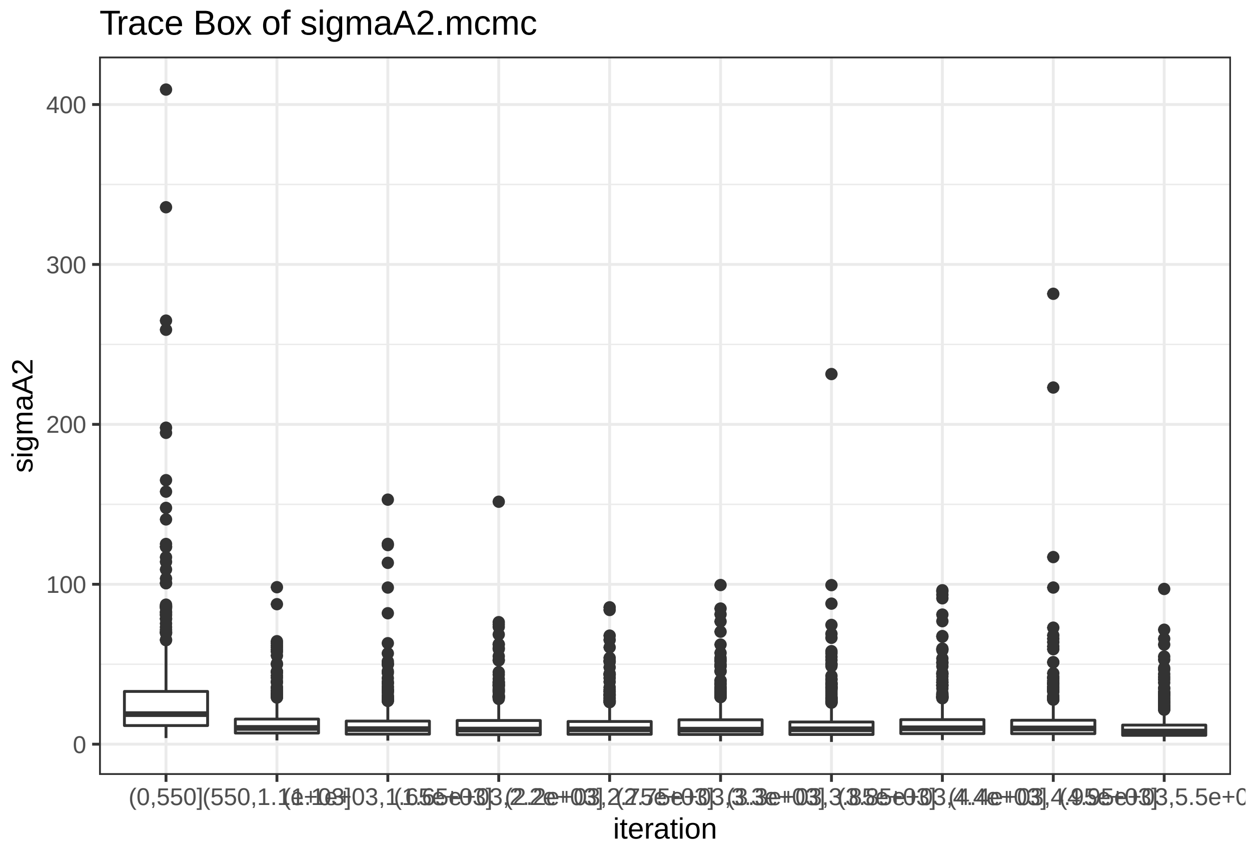
```r
df = data.frame(sigma2 = sigma2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = sigma2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigma2.mcmc") +
  xlab("iteration")
```



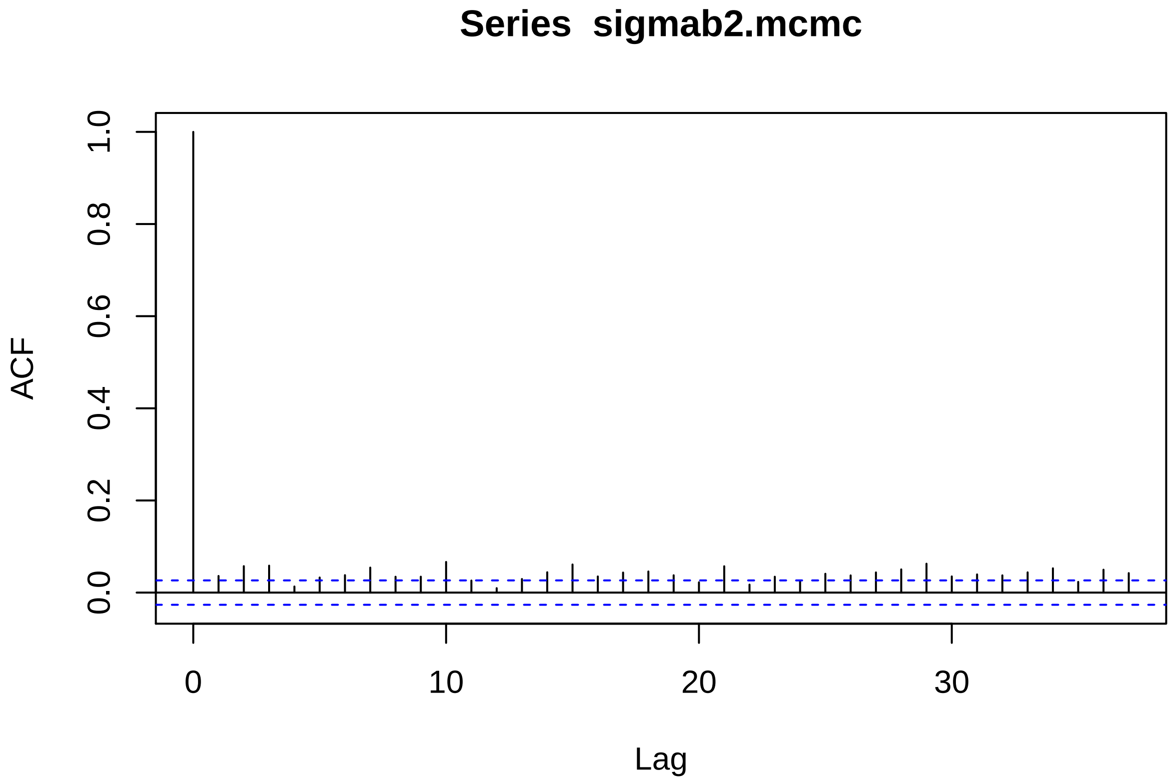
```r
# sigmaa2
acf(sigmaa2.mcmc)
```



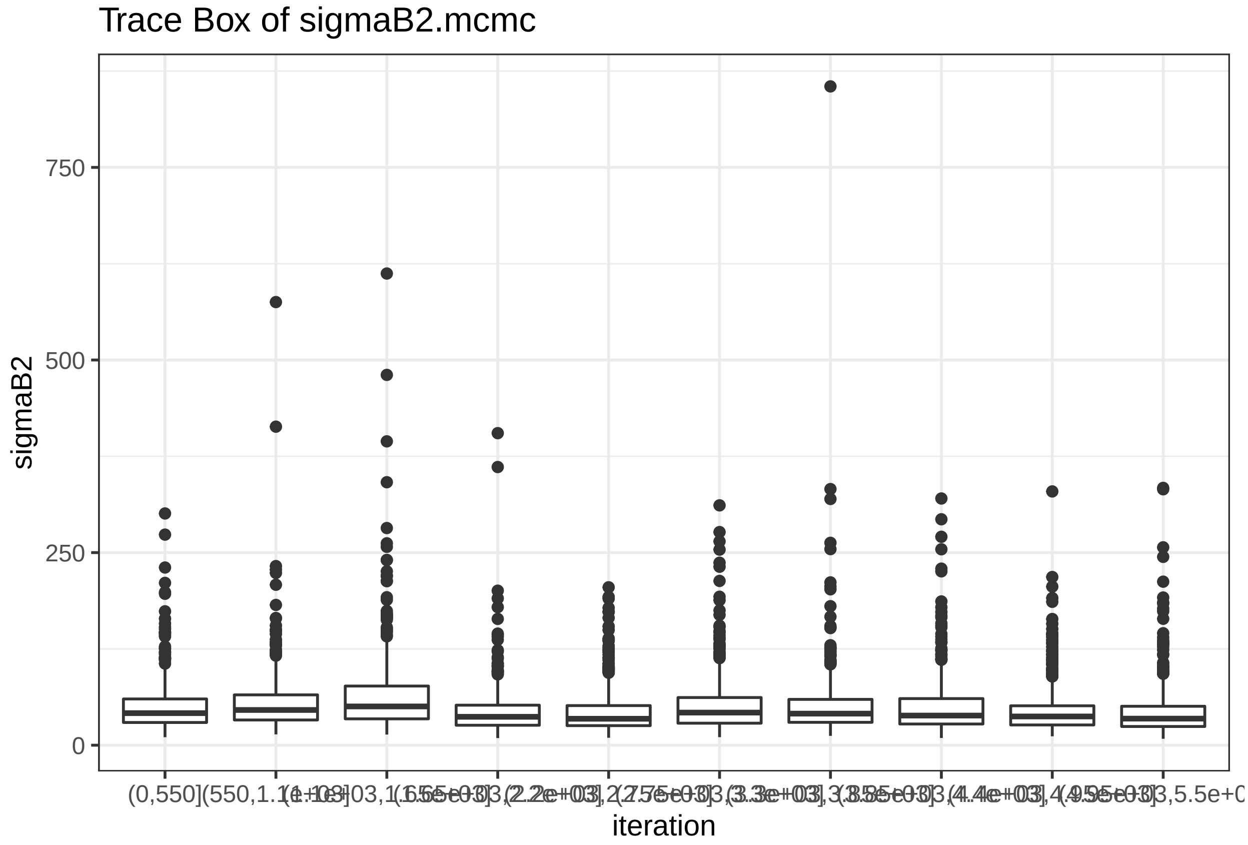
```r
df = data.frame(sigmaA2 = sigmaa2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = sigmaA2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigmaA2.mcmc") +
  xlab("iteration")
```



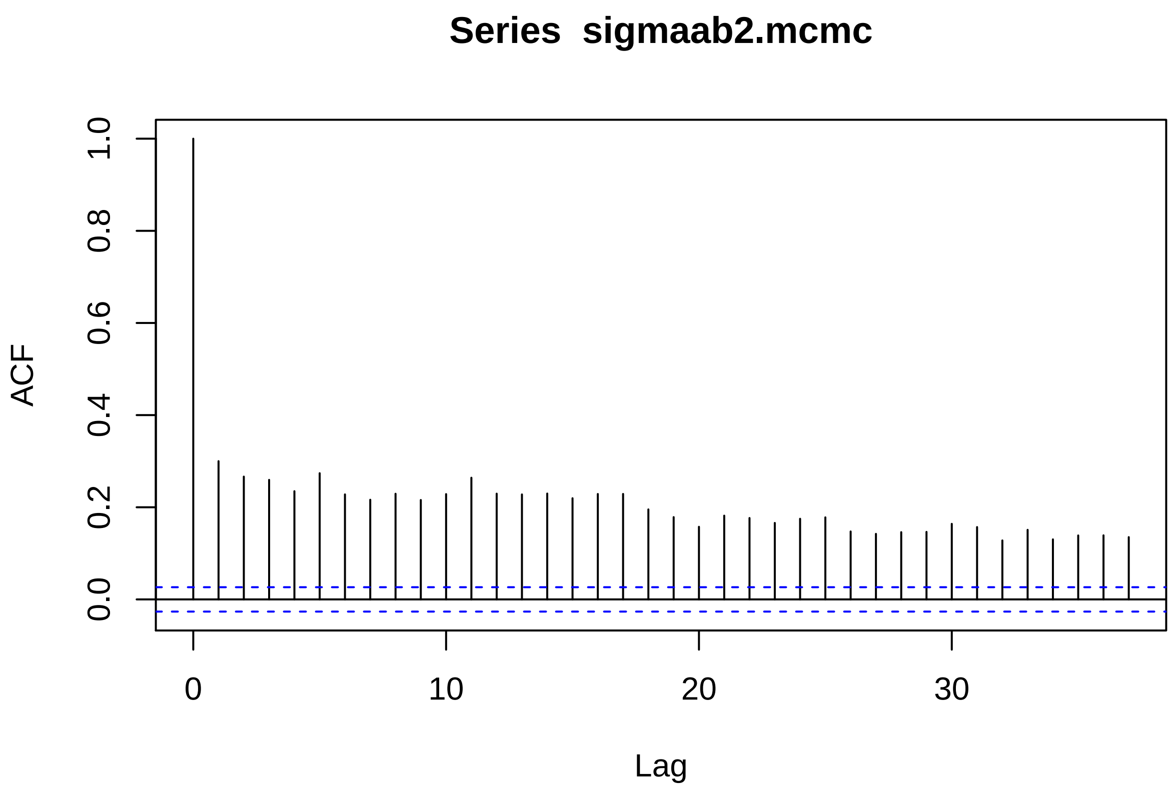
```r
# sigmab2
acf(sigmab2.mcmc)
```



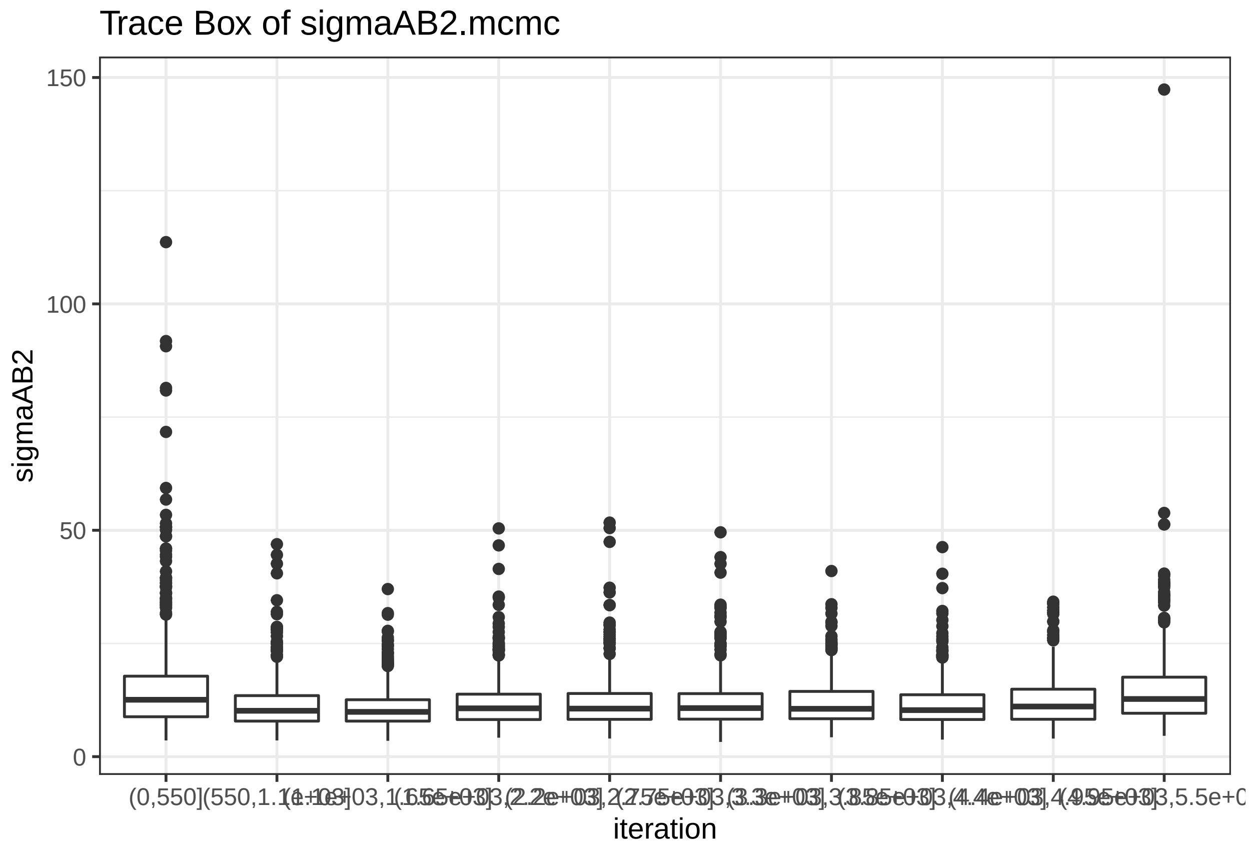
```r
df = data.frame(sigmaB2 = sigmab2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = sigmaB2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigmaB2.mcmc") +
  xlab("iteration")
```



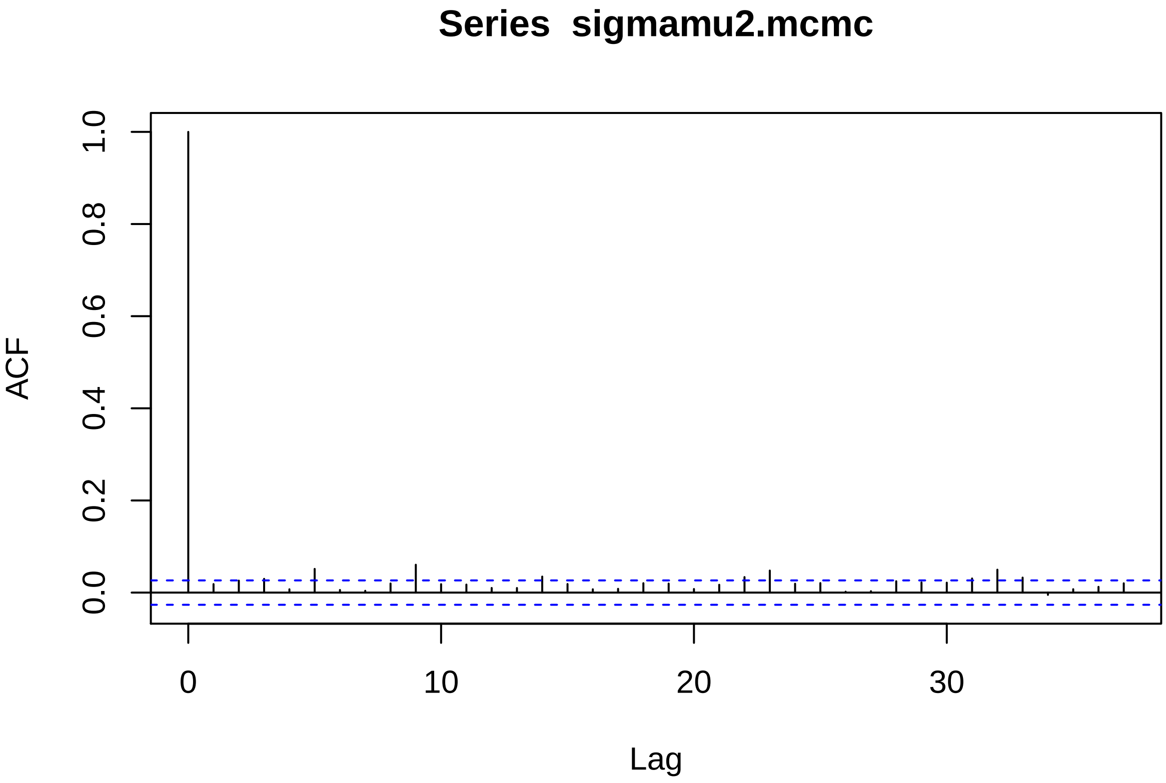
```r
# sigmaab2
acf(sigmaab2.mcmc)
```



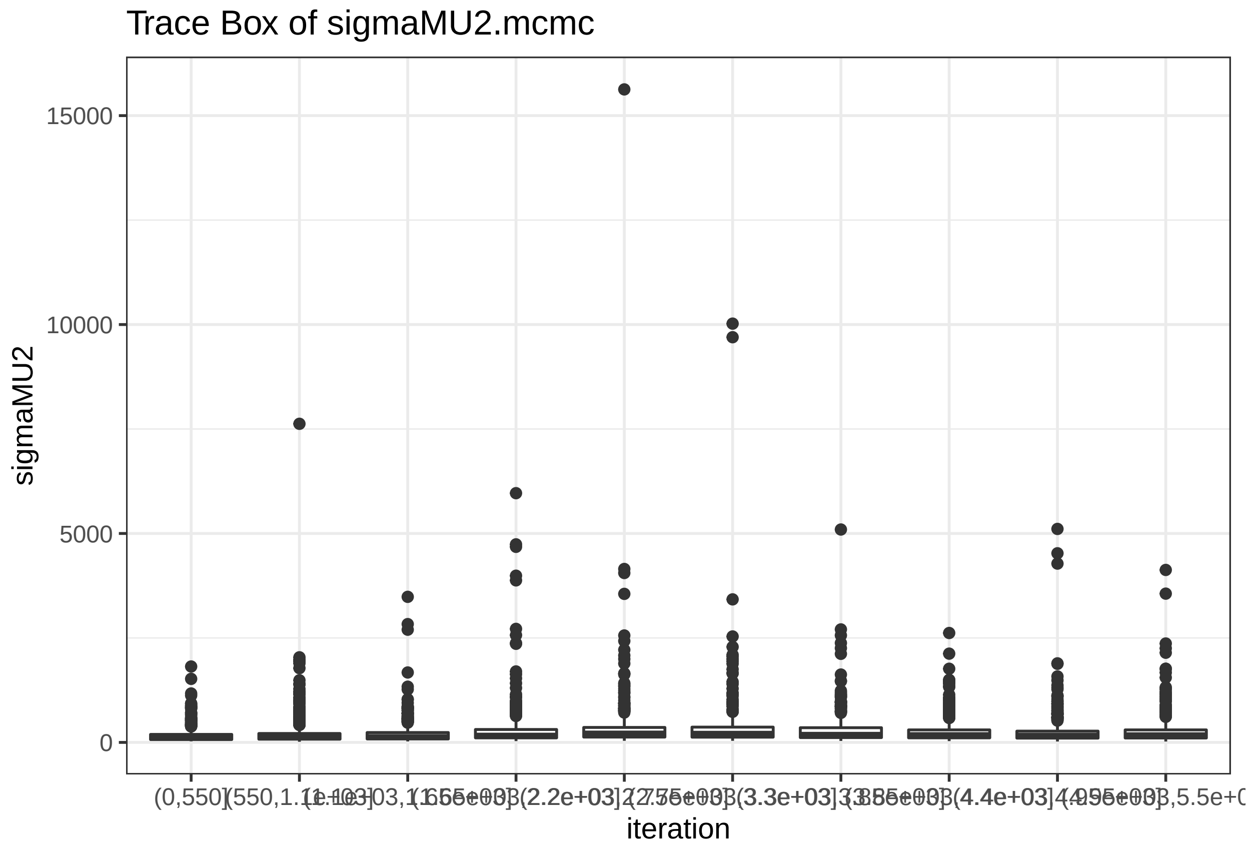
```
df = data.frame(sigmaAB2 = sigmaab2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = sigmaAB2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigmaAB2.mcmc") +
  xlab("iteration")
```



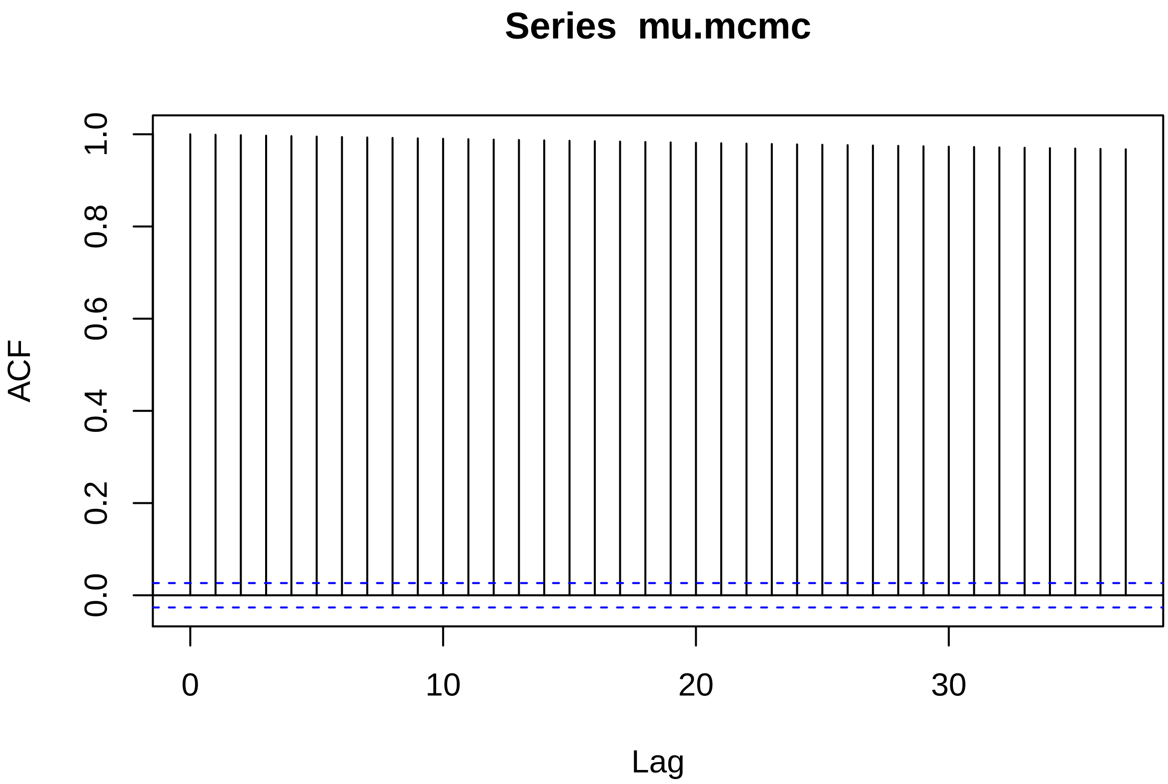
```
# sigmamu2
acf(sigmamu2.mcmc)
```



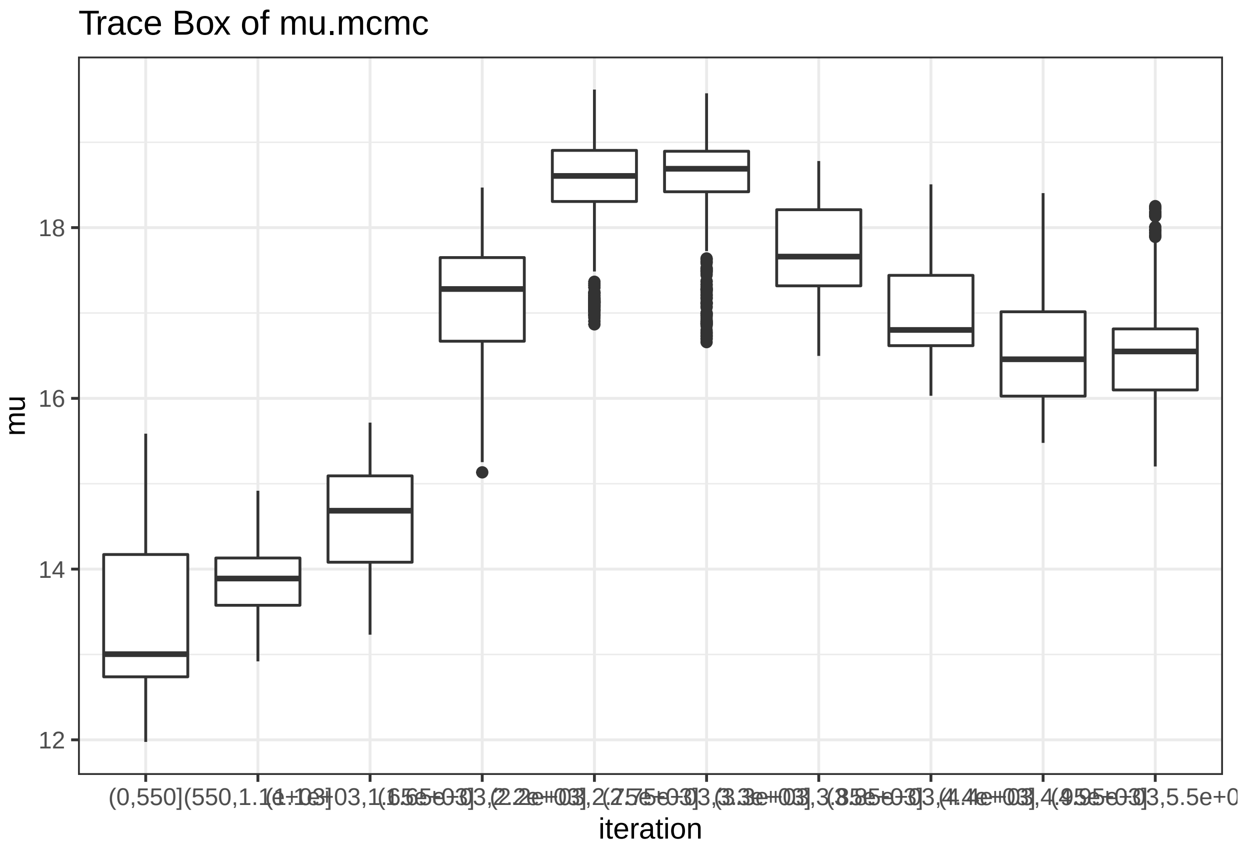
```
df = data.frame(sigmaMU2 = sigmamu2.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = sigmaMU2, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of sigmaMU2.mcmc") +
  xlab("iteration")
```



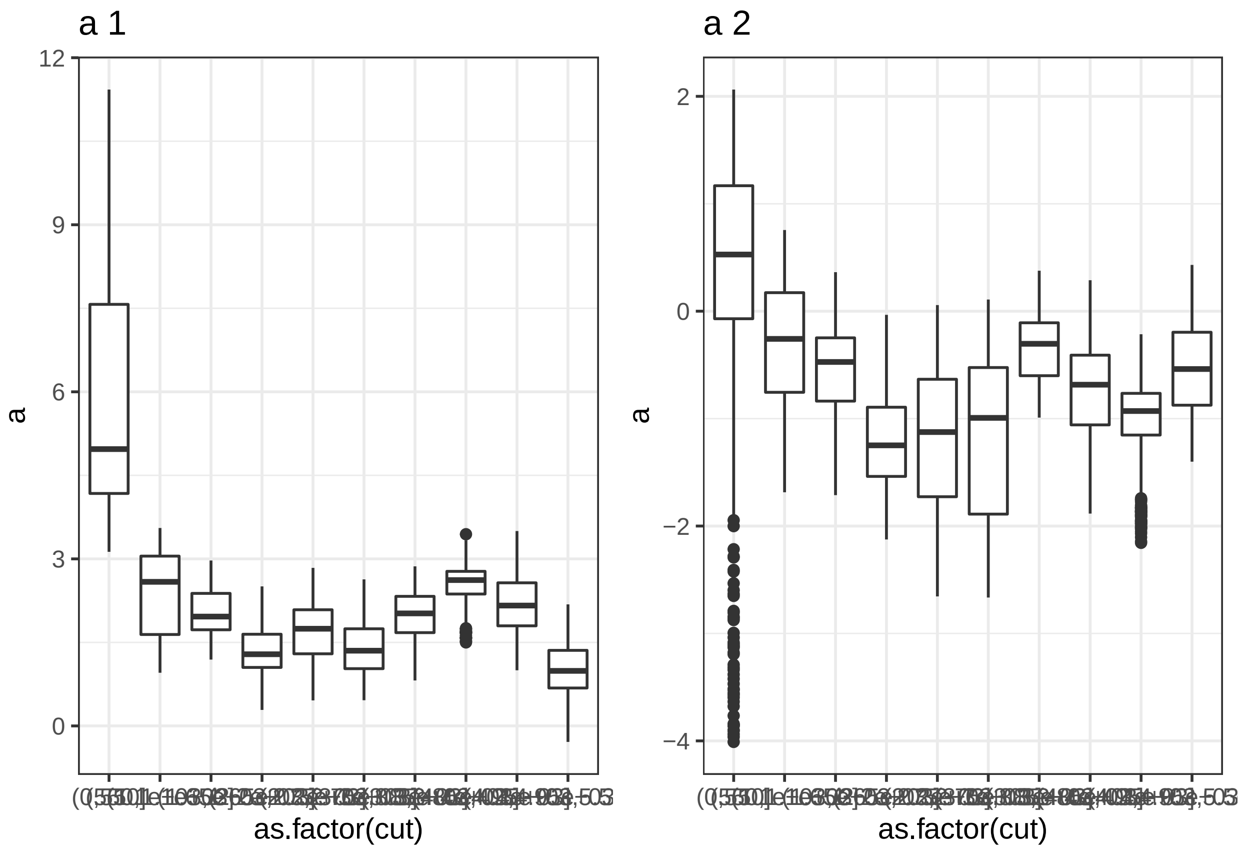
```
# mu
acf(mu.mcmc)
```



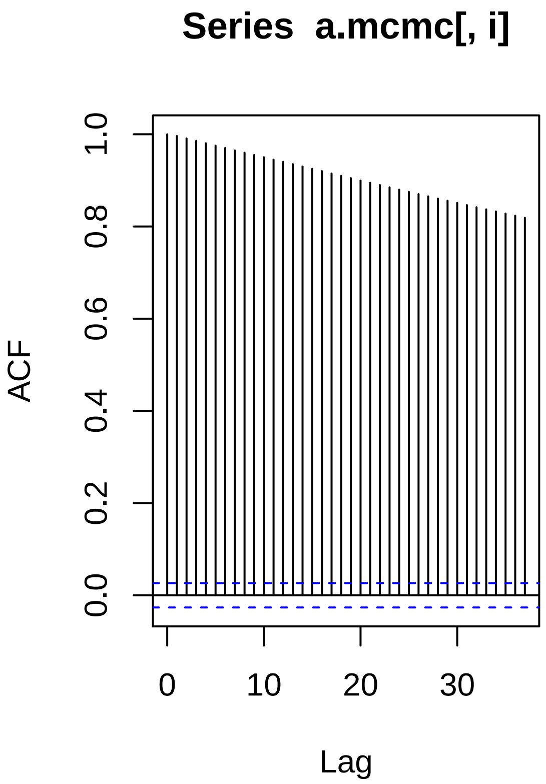
```
df = data.frame(mu = mu.mcmc, cut = cut(1:S, breaks = seq(0, S, S/10)))
ggplot(df, aes(y = mu, x = as.factor(cut))) +
  geom_boxplot() +
  ggtitle("Trace Box of mu.mcmc") +
  xlab("iteration")
```

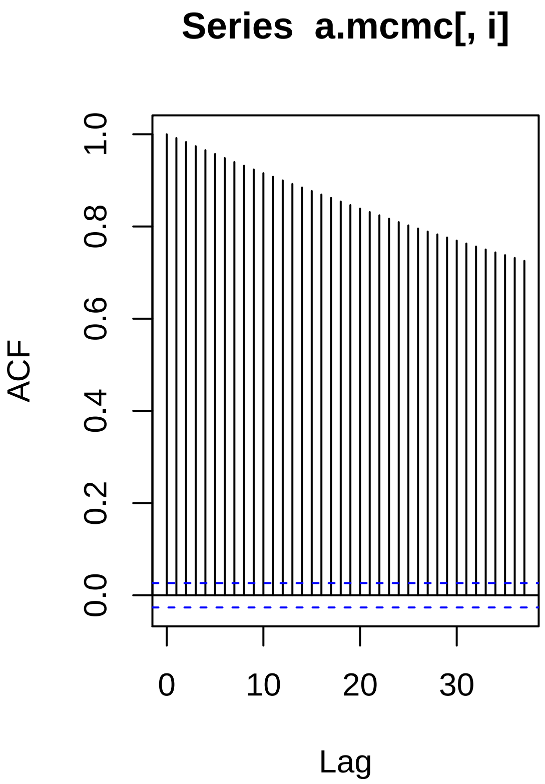


```
# a
p <- list()
for (i in 1:m1) {
  a_df = data.frame(a = a.mcmc[, i], cut = cut(1:S, breaks = seq(0, S, S/10)))
  p[[i]] = ggplot(a_df, aes(y = a, x = as.factor(cut))) +
    geom_boxplot() +
    ggtitle(paste("a", i))
}
do.call(grid.arrange,c(p, ncol = 2))
```

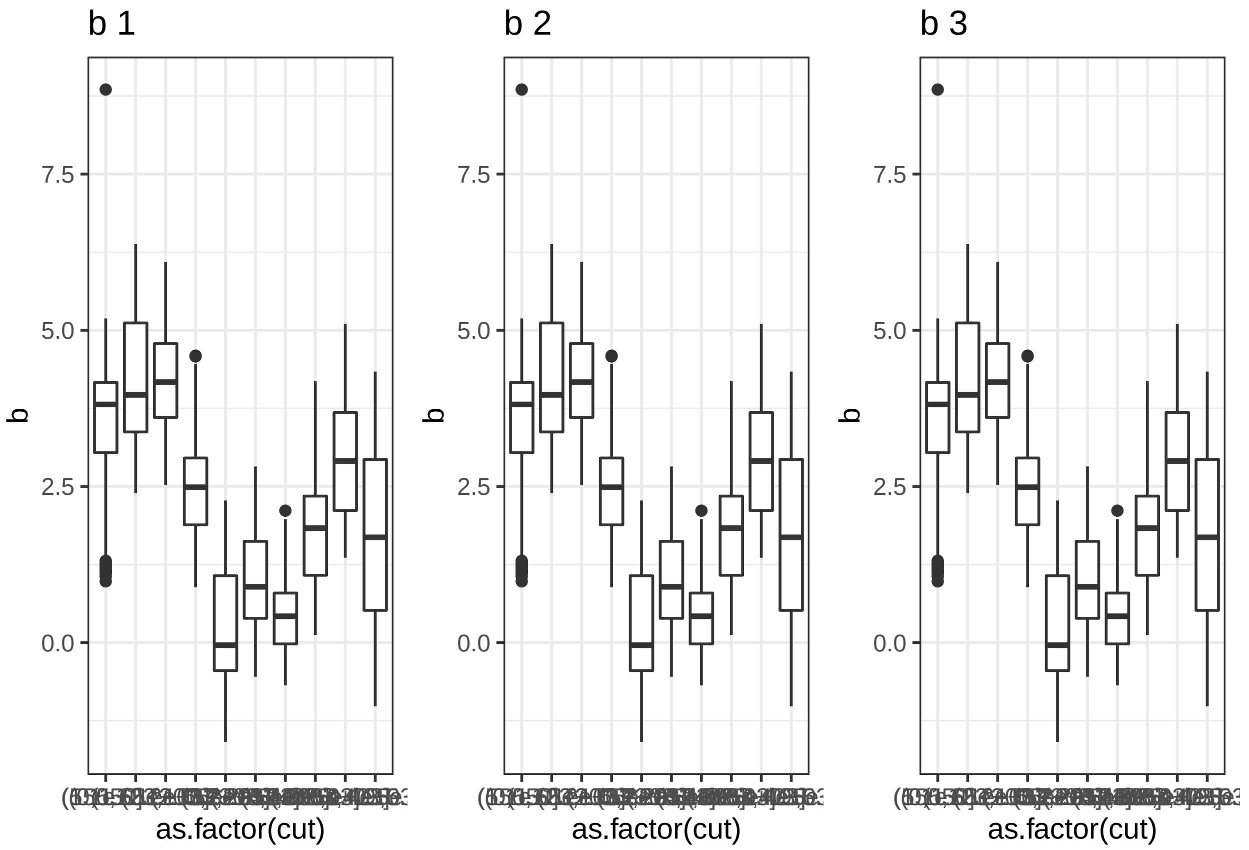


```r
par(mfrow = c(1, 2))
for (i in 1:m1) {
  acf(a.mcmc[, i])
}
```

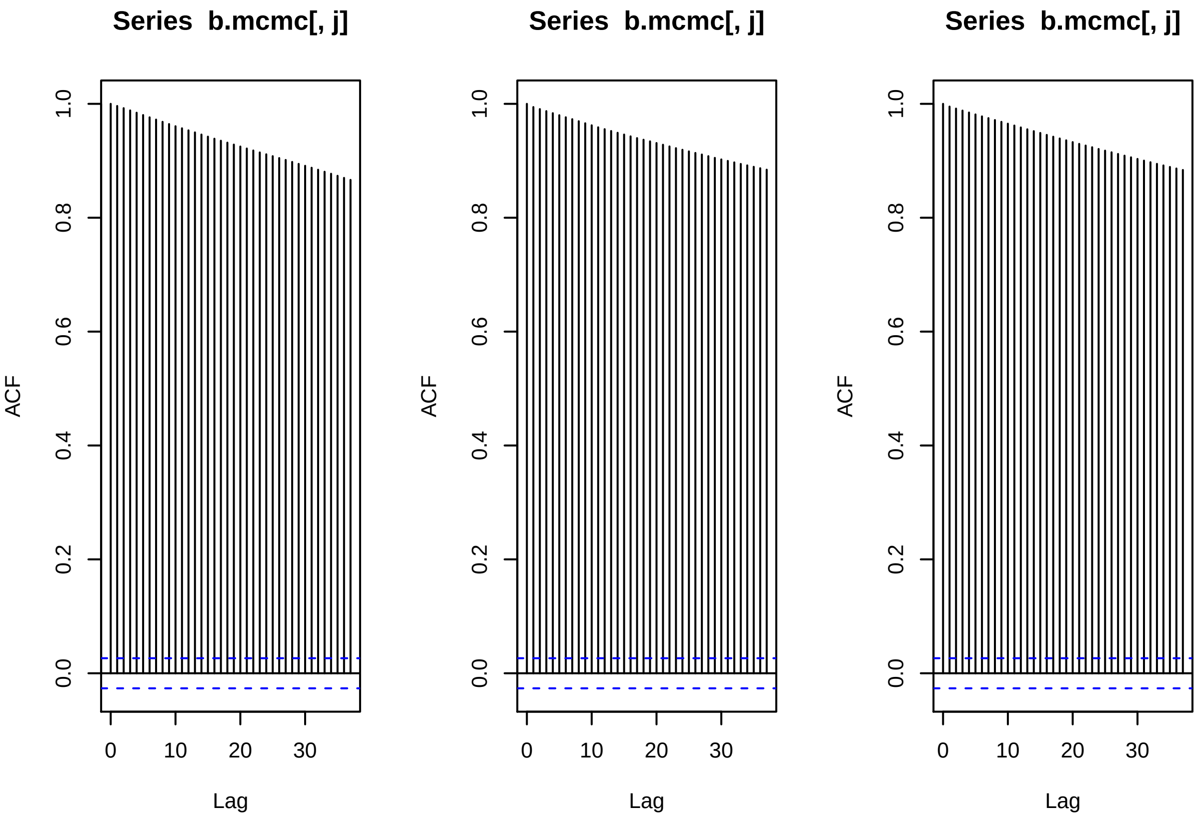




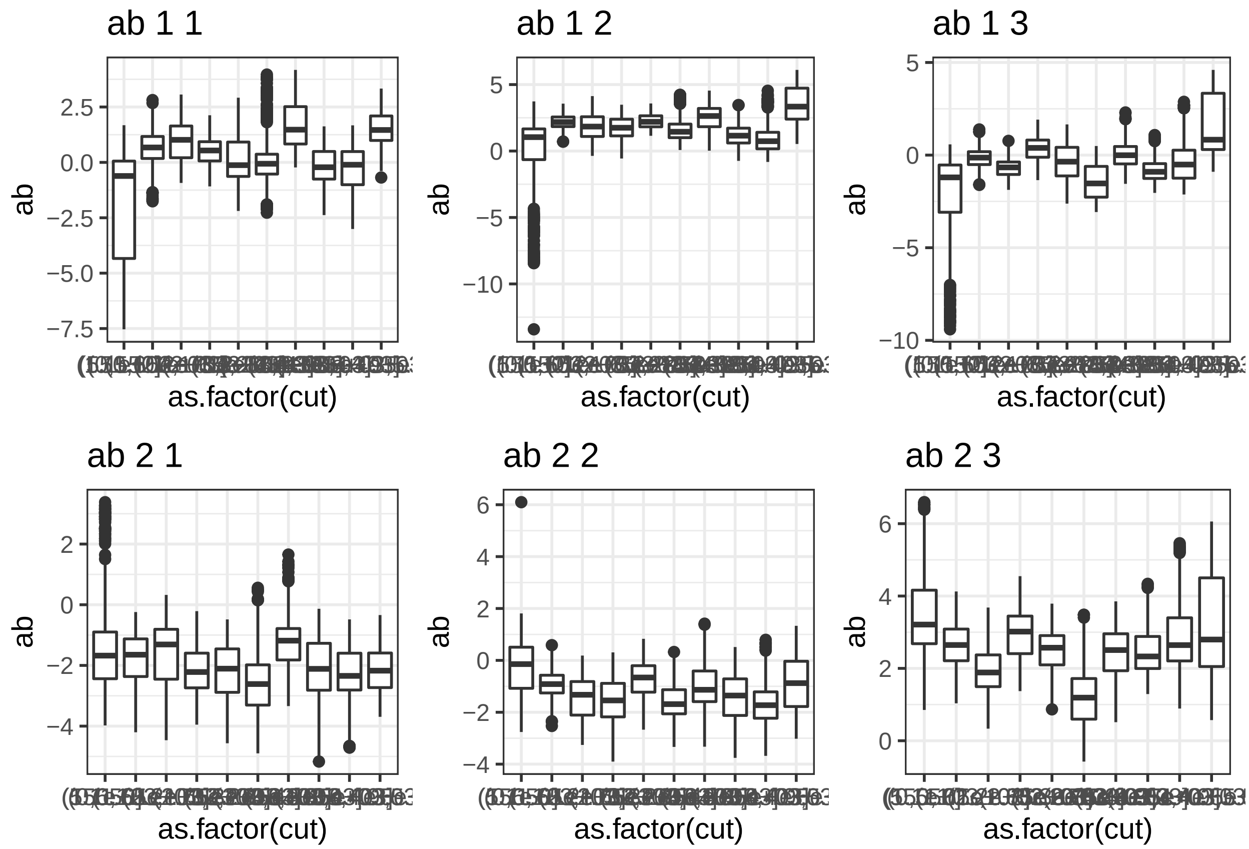
```r
# b
p <- list()
for (j in 1:m2) {
  b_df = data.frame(b = b.mcmc[, i], cut = cut(1:S, breaks = seq(0, S, S/10)))
  p[[j]] = ggplot(b_df, aes(y = b, x = as.factor(cut))) +
    geom_boxplot() +
    ggtitle(paste("b", j))
}
do.call(grid.arrange, c(p, ncol = 3))
```



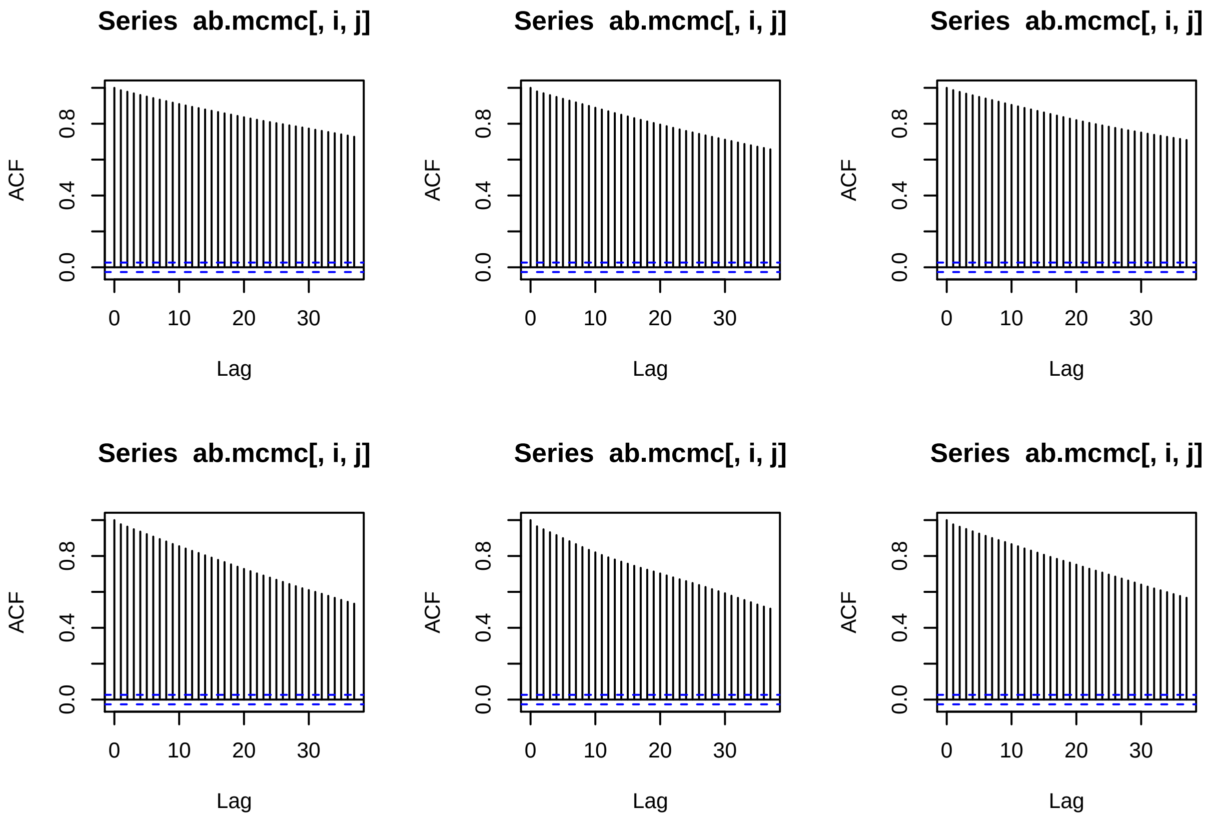
```r
par(mfrow = c(1, 3))
for (j in 1:m2) {
  acf(b.mcmc[, j])
}
```



```
# ab
p <- list()
for (i in 1:m1) {
  for (j in 1:m2) {
    index = (i-1)*m2 + j
    ab_df = data.frame(ab = ab.mcmc[, i,j], cut = cut(1:S, breaks = seq(0,S,S/10)))
    p[[index]] = ggplot(ab_df, aes(y = ab, x = as.factor(cut))) +
      geom_boxplot() +
      ggtitle(paste("ab", i, j))
  }
}
do.call(grid.arrange,c(p, nrow = 2, ncol = 3))
```



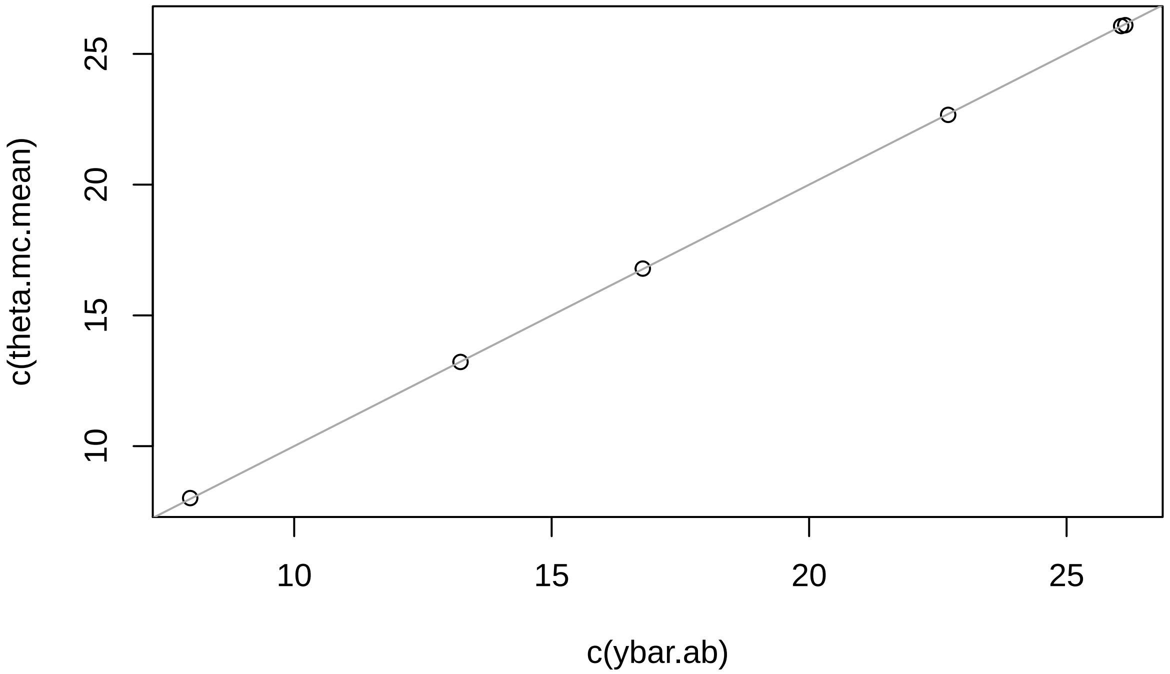
```
par(mfrow = c(2, 3))
for (i in 1:m1) {
  for (j in 1:m2) {
    acf(ab.mcmc[, i, j])
  }
}
```



For the means, from the above plots we can find that $\mu, a_i, b_j, (ab)_{ij}$ have really bad convergence problems. There is a obvious trend in the trace plots and also huge autocorrelation. However, all of these are fine because the trance plots and acf plots of all the θ's are fine. Inferencing the averages for each sub cell is relatively easy compared to differentiating the effects of each components (factors). Also, we only cares about the convergence and posterior of θ_{ij}, because it contains all the information we need in this scenario.

For the variance, we find that σ_a^2 perform least in traceplots and autocorrelation, but it's still acceptable. This results from the fact that there is less differences across group a than the group b.

```r
### shrinkage of means
ybar.ab = matrix(nrow = m1, ncol = m2)
for (i in 1:m1) {
  for (j in 1:m2) {
    ybar.ab[i, j] = mean(data$len[(data$supp == l1[i]) & (data$dose == l2[j])])
  }
}

theta.mc.mean = apply(theta.mcmc, c(2, 3), mean)

plot(c(ybar.ab), c(theta.mc.mean))
lines(1:30, 1:30, col = "darkgrey")
```



Yes, there is a little shrinkage towards the overall means. The groups with large sample averages are slightly below the grey line and the groups with small sample averages are slightly above the grey line. Due to the facts that each groups have exactly the same sample sizes, the shrinkage effects are pretty similar and small.

```
for (i in 1:m1) {
  for (j in 1:m2) {
    print(paste("theta", i, j))
    print(quantile(theta.mcmc[, i, j], probs = c(0.025, 0.975)))
  }
}
```

```
## [1] "theta 1 1"
##     2.5%    97.5%
## 12.87245 13.56473
## [1] "theta 1 2"
##     2.5%    97.5%
## 22.33109 23.00140
## [1] "theta 1 3"
##     2.5%    97.5%
## 25.71637 26.40974
## [1] "theta 2 1"
##     2.5%    97.5%
## 7.674053 8.361809
## [1] "theta 2 2"
##     2.5%    97.5%
## 16.45443 17.13689
## [1] "theta 2 3"
##     2.5%    97.5%
## 25.74381 26.44098
```

Accroding to these posterior confidence interval, we know that both increasing dose (to 2) and using orange juice could help guinea pigs grow teeth. Also, the dominating effects here is dose. The increase of using orange juice when dose is small is greater than the increase of using orange juice when dose is high.