Topics in Bayesian Statistics

HW#1

학번	182STG18
이름	이하경
제출일	2019.04.09

Exercise

예제 4.3 에 대한 M-H 알고리즘에서 후보표본을 다변량 정규분포에서 생성하여 원소 모두에 같은 보폭을 사용하였다. 그러나 파라미터마다 분산이 다를 수 있고, 즉 적절한 보폭이 다를 수 있으므로 μ와 $\log(\sigma^2)$가 서로 다른 분산 δ_1^2과 δ_2^2을 가진다고 가정한 후 다음의 알고리즘에 따라 적절한 값을 찾아보려고 한다.

$$(\mu, \log(\sigma^2)) \sim N_2\left((\mu_0, \log(\sigma_0^2)),\ diag(\delta_1^2, \delta_2^2)\right), \qquad \mu_0 = 10, \sigma_0^2 = 25$$

1) Preliminary Run: 4.2절의 알고리즘을 짧게 실행하여 $\theta = (\mu, \log(\sigma^2))$의 대략적인 추정치와 분산을 추정한다.

짧은 실행을 위해서 nchain=3, nsim=2000, nburn=0으로 설정하였다.
$\delta = 1$으로 동일하게 설정하여 MH 알고리즘을 실행한 결과 구성된 sample로부터의 추정치와 분산은 다음과 같다.

	μ	$\log(\sigma^2)$
추정치	15.1426	3.2289
분산	2.5366	0.2271

2) 추정된 분산에 2.4를 곱하여 δ_1^2과 δ_2^2의 값으로 정한다.

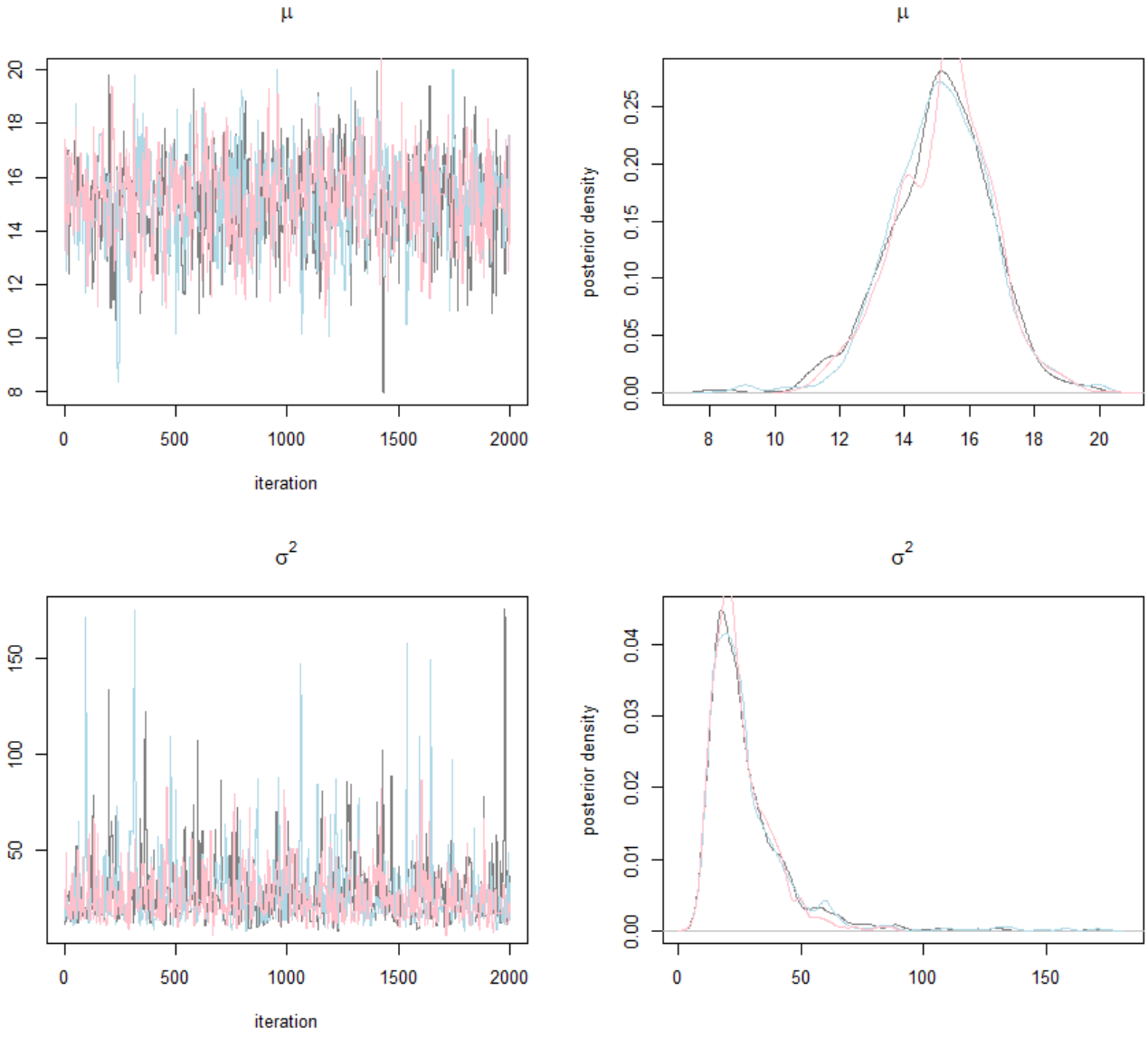
δ_1^2	δ_2^2
5.4817	0.4978

즉 μ와 $\log(\sigma^2)$의 보폭(표준편차)을 각각 $\delta_1 = 2.3413, \delta_2 = 0.7055$로 정하여 알고리즘을 실행하였다.

3) 4.2절의 알고리즘을 변형하여 $N\left(\theta^{curr}, diag(\delta_1^2, \delta_2^2)\right)$로부터 후보표본을 추출하는 MH 알고리즘을 코딩한다.
4) MH를 짧게 수행하여 후보표본의 채택확률 및 경로그림을 확인해본다.

1)에서 $\delta = 1$으로 동일한 보폭을 설정했던 것과 달리 2에서 정한 값으로 보폭을 다르게 설정하여 기존 함수에서 delta에 2*1의 벡터를 입력하여 알고리즘을 수행하였다. 체인 수(nchains), 준비단계 반복 수(nburn), 준비단계 이후의 반복 수(nsim)를 1)과 동일하게 하여 짧게 실행하였다.

▷ 경로그림과 사후 밀도함수



▷ 채택확률(Acceptance Rate) = **36.72%**

(δ_1^2, δ_2^2)을 각각 (5.4817, 0.4978)으로 설정했을 때 채택확률은 약 37%로 이상적인 채택확률 24%에 비해 높은 편이다.

5) 채택확률이 대략 24%가 되도록 δ_1^2과 δ_2^2의 값을 조정한다.

δ_1^2	δ_2^2
10.89	1.21
$(\delta_1 = 3.3)$	$(\delta_2 = 1.1)$

δ_1^2과 δ_2^2의 값을 바꿔가며 채택확률이 24%가 되도록 하였다. 보폭(표준편차)가 대략 $\underline{\delta_1 = 3.3,\ \delta_2 = 1.1}$일 때 채택확률이 **24.06%**로 계산되었다. 따라서 이 값을 최종 값으로 선택하였다.

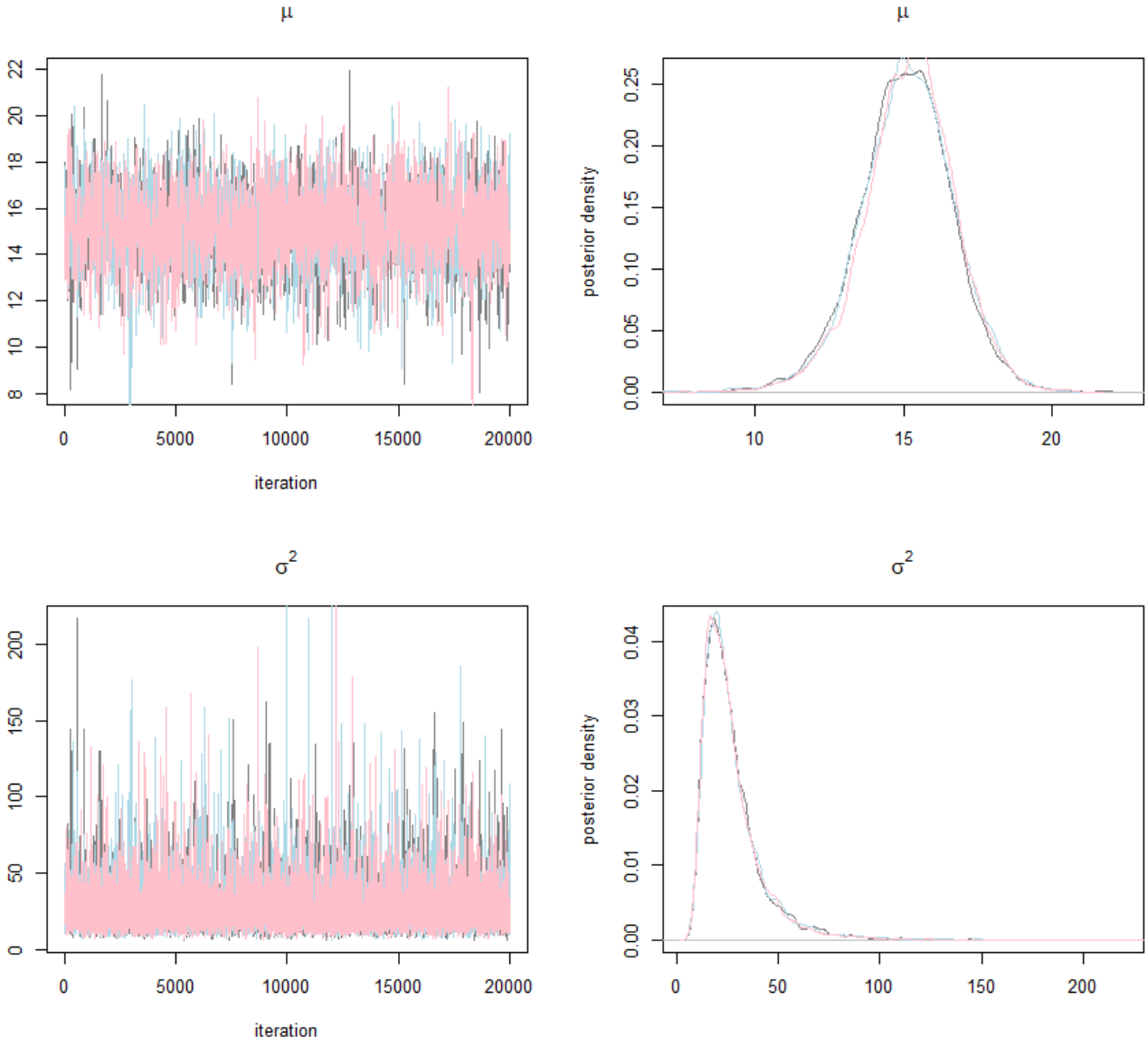
6) 최종 선택된 δ_1^2과 δ_2^2을 사용하여 MH를 충분히 길게 수행한 후 수렴속도, 효율, 사후추론 결과를 4.2절의 공통 분산을 사용하였을 때와 비교하라.

다중 체인의 수는 3, 준비단계 이후의 반복수는 20000, 준비단계의 반복수는 5000으로 설정하고 $\delta = 1$으로 보폭을 동일하게 설정하였을 때와 $(\delta_1, \delta_2) = (3.3, 1.1)$으로 서로 다르게 설정하였을 때의 결과를 비교하였다.

(1) $\delta = 1$ (보폭 동일)

	μ	$\log(\sigma^2)$
추정치	15.2457	3.1642
분산	2.5224	0.2189

▷ 경로그림과 사후밀도함수



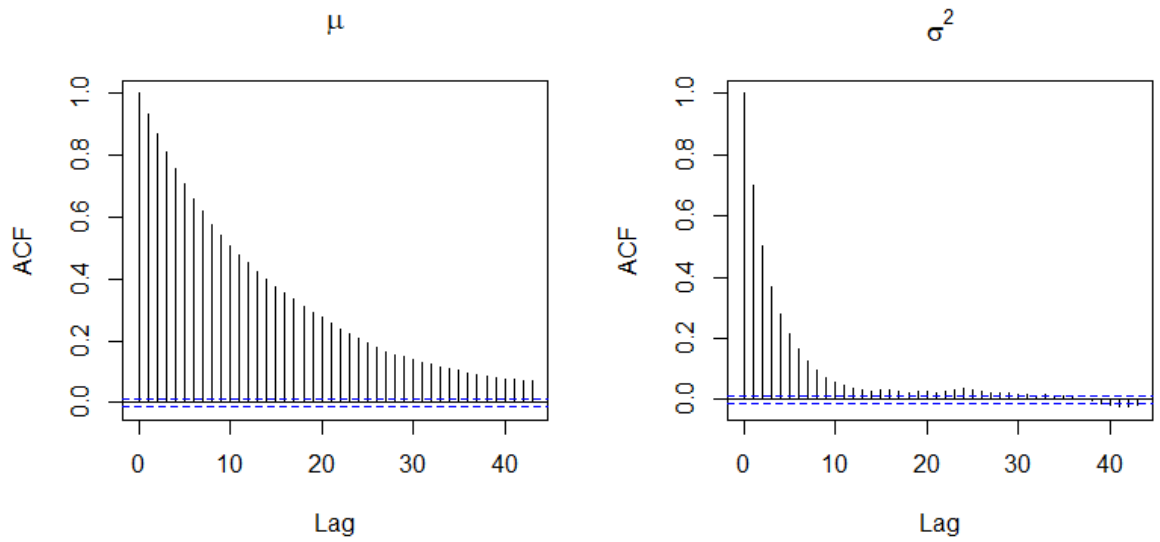
- 서로 다른 체인의 경로그림 및 사후밀도함수의 형태가 거의 유사하고, 경로그림을 통해 mixing이 잘 이루어졌다고 판단하여 수렴이 이루어졌다고 보았다.

▷ 채택확률 및 Gelman 상수의 상한

	μ	σ^2
Gelman Upper CI	1.02	1.00
채택확률		39.22%

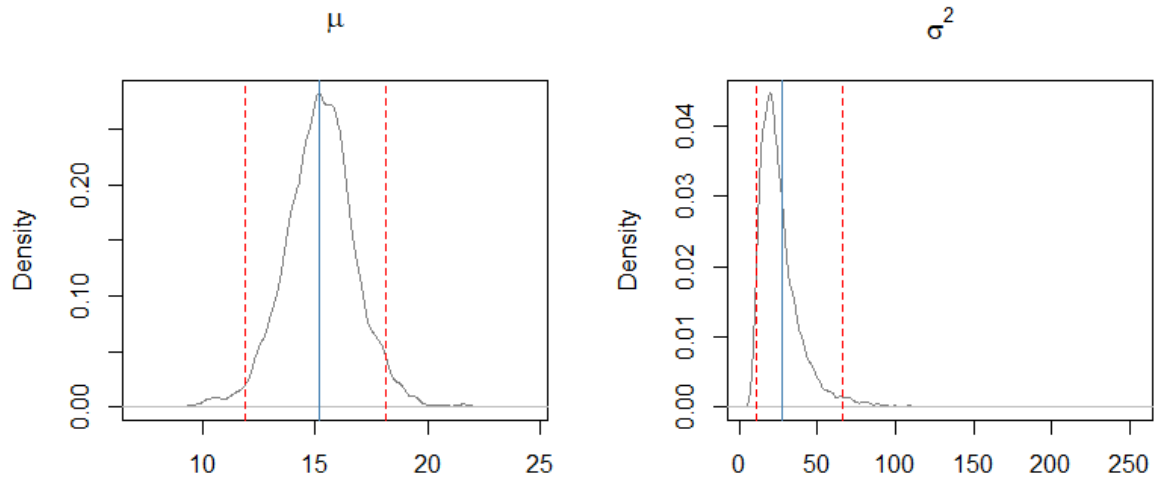
- Gelman 상수의 상한이 모두 1.1 이하로 1에 가까워 수렴이 이루어졌다고 보았다.
- 채택확률은 24%에 비해 높은 것으로 나타났다.

▷ 자기상관그림



- $\delta = 1$으로 동일한 보폭인 경우 σ^2는 문제가 없으나 μ의 자기상관함수 그림에서 준비단계 5000번 이후임에도 불구하고 자기상관성이 어느 정도 높은 것으로 보인다.

▷ 95% HPD(사후 신뢰구간)

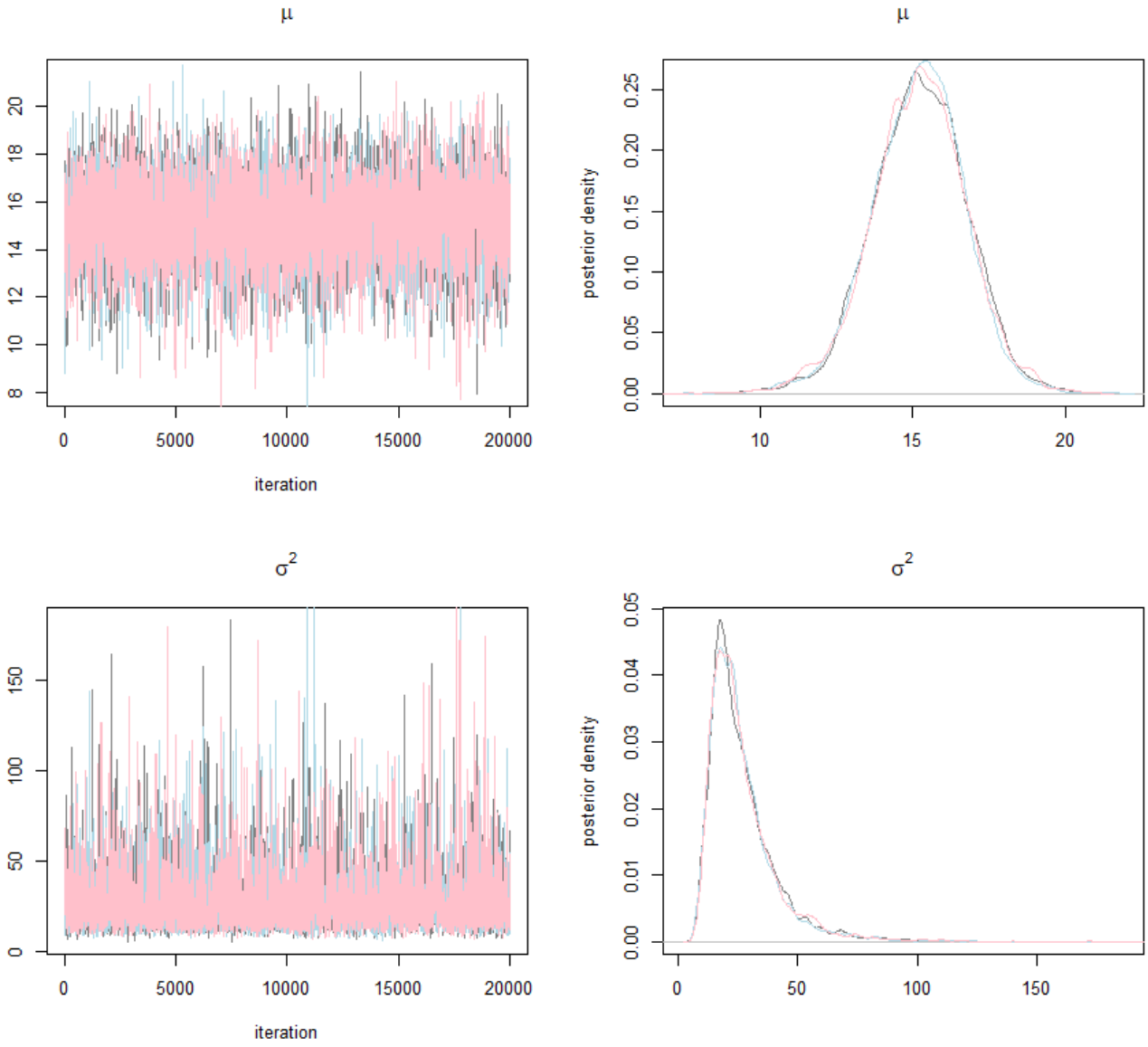


	estimate	2.5%	97.5%
μ	15.1576	11.9255	18.1295
σ^2	27.0284	10.6176	66.6478

(2) $(\delta_1, \delta_2) = (3.3, 1.1)$

	μ	$\log(\sigma^2)$
추정치	15.1281	3.1741
분산	2.4026	0.2260

▷ 경로그림과 사후밀도함수



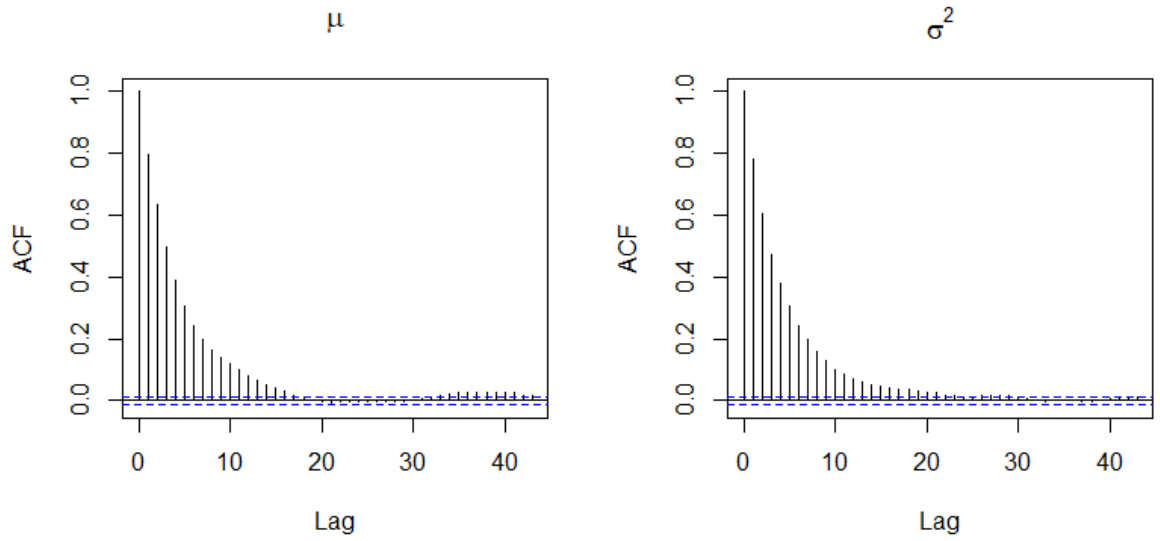
- 경로그림 및 사후밀도함수가 체인에 상관없이 충분히 안정적으로 보이며 실제 분포와 유사하게 그려졌다. 수렴이 적절히 이루어졌다고 볼 수 있다.

▷ 채택확률 및 Gelman 상수의 상한

	μ	σ^2
Gelman Upper CI	1.00	1.01
채택확률		23.92%

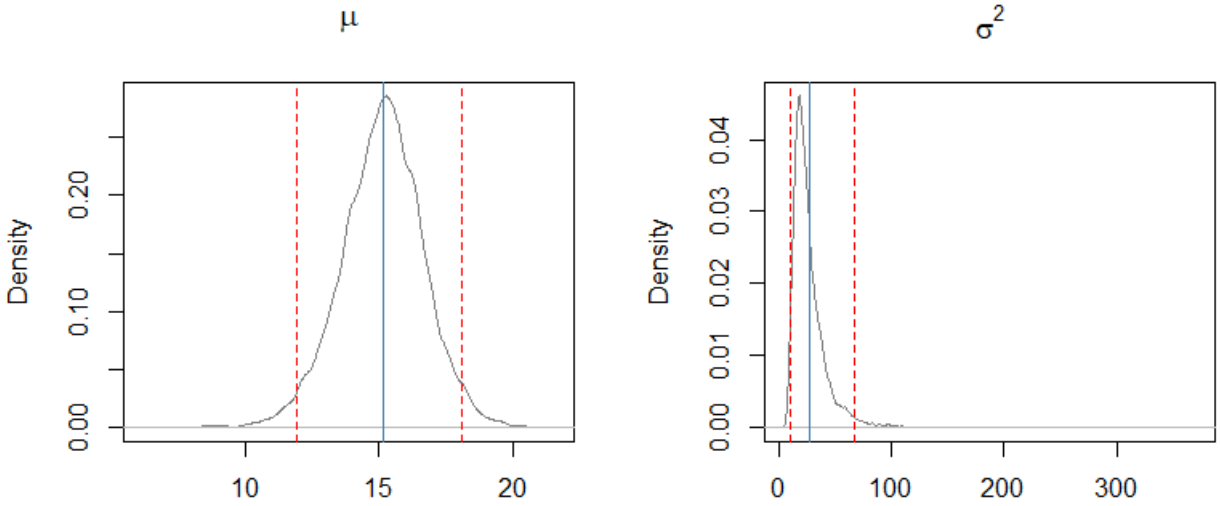
- Gelman 상수가 모두 1에 가까워 수렴이 이루어졌다고 할 수 있으며, 채택확률이 약 24%으로 적절하였다.

▷ 자기상관그림



- 보폭을 다르게 설정한 경우 두 모수 모두 자기상관함수 그래프에서 값이 빠르게 감소하는 것을 확인하였다.

▷ 95% HPD(사후 신뢰구간)



	posterior mean	2.5%	97.5%
μ	15.1618	11.9637	18.1294
σ^2	26.8281	10.5009	66.5773
$\log(\sigma^2)$	3.2895	2.3515	4.1984

두 결과의 비교

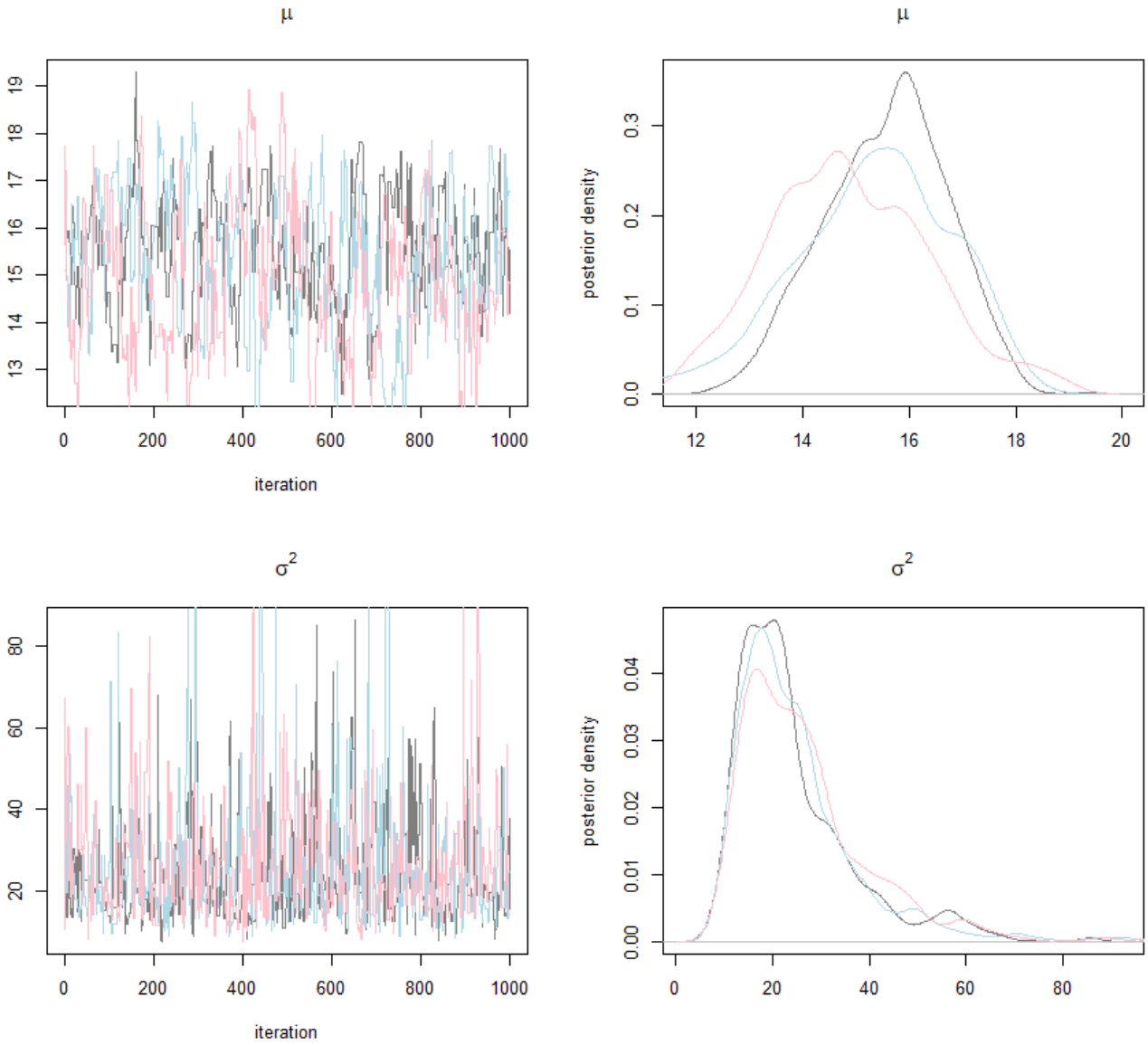
두 경우 모두 경로그림이 안정적이게 보이며, 사후밀도함수가 실제 분포와 유사한 형태로 그려졌다. Gelman 상수의 상한 역시 모두 1.1 이하로 나타났다. 준비단계 반복 수를 5000으로 충분히 설정하여 이후 20000번의 시뮬레이션 동안 두 가지 경우 모두 수렴에 이르렀다고 할 수 있다. 동일 분산을 1로 설정했을 때 채택확률은 약 40%로 높게 나타났지만 분산을 다르게 하였을 때 24% 정도로 이상적이라고 할 수 있다.

초기의 5000개의 준비단계를 설정하고 충분히 길게 실행했을 경우 동일한 보폭을 지정했을 때와 서로 다른 보폭을 지

정했을 때의 결과는 큰 차이 없이 모두 안정적으로 수렴하는 모습을 보였다. 사후 추론에서 추정치와 분산 역시 거의 동일하게 추정되었으며, 95% HPD 역시 거의 비슷하다. 그러나 자기상관 그림에서 μ의 경우 서로 다른 보폭을 사용한 경우가 동일한 보폭을 사용한 경우에 비해 자기상관성이 훨씬 더 빠르게 감소하였다.

다음은 수렴 속도를 비교하기 위해 준비단계의 초기 1000개의 sample만으로 경로그림과 사후밀도함수를 그려본 것이다.

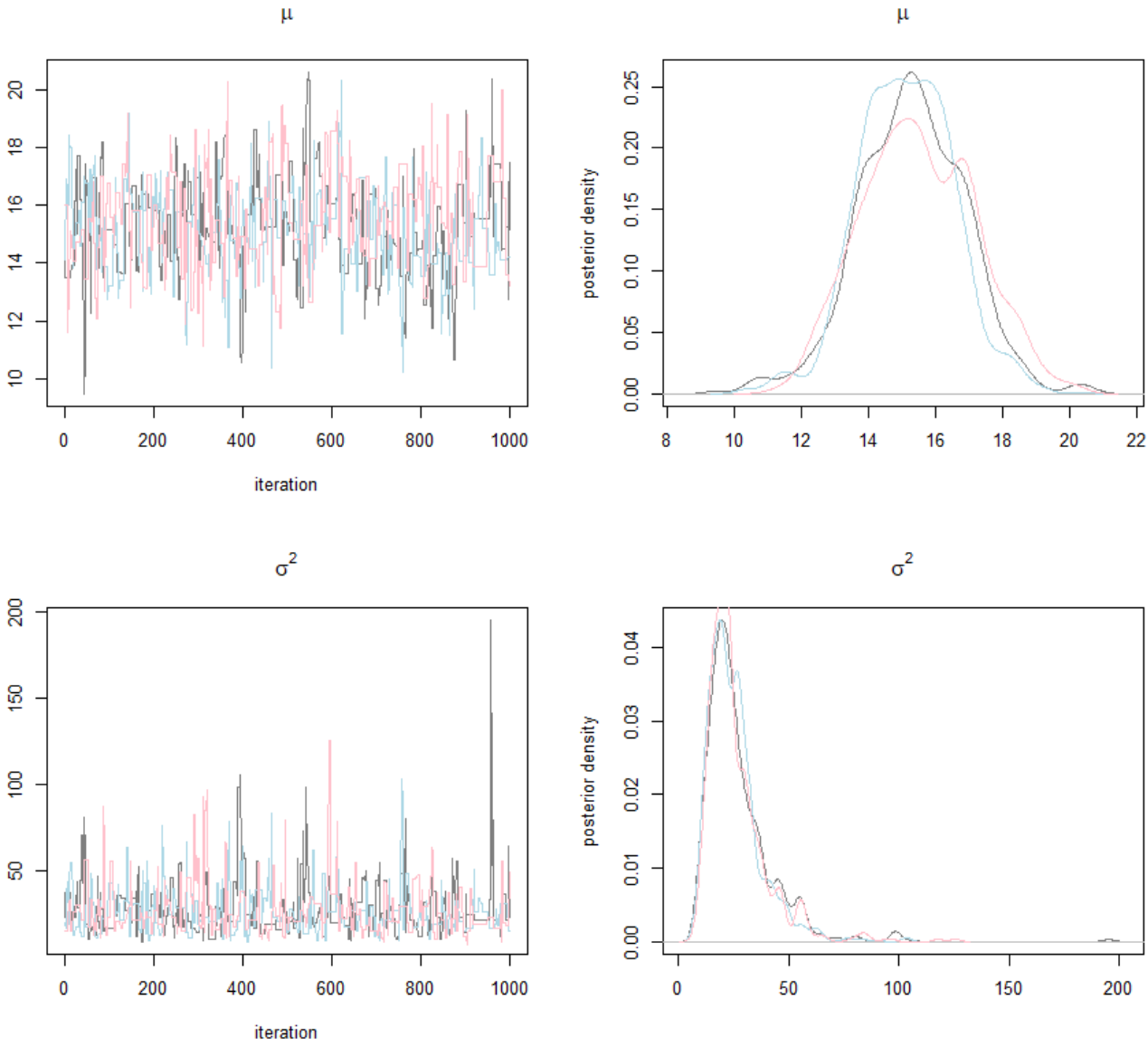
(1) $\delta = 1$일 때



- 초기 1000개의 sample들만으로 경로그림과 사후밀도함수를 그렸을 때 체인 별로 사후밀도함수가 유사하지 않은 것으로 보아 충분히 수렴하지 않았다고 생각되었다. 초기 1000개로 Gelman 상수를 구하였을 때 아래와 같이 μ의 상한이 1.24로 수렴에 이르렀다고 할 수 없다.

Gelman	Point est.	Upper CI
μ	1.08	1.24
σ^2	1.02	1.02

(2) $(\delta_1, \delta_2) = (3.3, 1.1)$일 때



Gelman	Point est.	Upper CI
μ	1.05	1.18
σ^2	1.04	1.07

서로 다른 보폭(1)을 사용한 경우 역시 초기 1000개의 sample로는 충분히 수렴하였다고 할 수 그러나 동일 보폭을 사용하였을 경우보다는 조금 더 완화된 경향을 보인다. μ의 Gelman 상수 역시 (1)에 비해 줄어들어 수렴속도 면에서 μ의 보폭을 3.3으로 지정하였을 때가 조금 더 빠르다고 할 수 있다. 반면 σ^2은 Gelman 상수가 약간 더 크게 계산되었지만 두 경우 모두 1.1 이하로 초기 단계에서도 수렴에 가까워졌다고 할 수 있다.

[Appendix] R code

```
library(coda)
library(tidyverse)

# dataList
mu0 = 10 ; sigsq0 = 25 ; a = 0.5 ; b = 1
x = c(10, 13, 15, 11, 9, 18, 20, 17, 23, 21)
dataList <- list(x = x, mu0 = mu0, sigsq0 = sigsq0, a = a, b = b)

# posterior kernel
post.norm <- function(theta, dataList) {
  # retrieve data from dataList
  x = dataList$x
  mu0 = dataList$mu0
  sigsq0 = dataList$sigsq0
  a = dataList$a
  b = dataList$b

  mu <- theta[1] ; sigsq <- theta[2]
  f <- exp( -0.5*length(x)*log(sigsq) -0.5*sum((x-mu)^2)/sigsq -
              0.5*(mu-mu0)^2/sigsq0-(a+1)*log(sigsq) -b/sigsq )
  return(f)
}

# metropolis algorithm
metro.norm <- function(nsim, nburn, delta, dataList, initList) {
  # initial values of mu and log.sigsq
  mu = initList$mu
  log.sigsq = log(initList$sigsq)

  theta.curr <- c(mu, log.sigsq)
  p <- length(theta.curr)

  # --- start iterations
  par.samples <- matrix(0, nsim, p)
  for (iter in 1:(nsim+nburn)) {
    z <- rnorm(p)
    theta.prop <- z*delta + theta.curr
    mu.curr <- theta.curr[1] ; sigsq.curr <- exp(theta.curr[2])
    mu.prop <- theta.prop[1] ; sigsq.prop <- exp(theta.prop[2])
    alpha <- (post.norm(c(mu.prop, sigsq.prop), dataList) / post.norm(c(mu.curr, sigsq.curr), dataList)) *
              (sigsq.prop / sigsq.curr)
    if (runif(1) < alpha) { theta.next <- theta.prop } else { theta.next <- theta.curr }
    theta.curr <- theta.next

    if (iter > nburn) par.samples[iter-nburn,] <- c(theta.next[1], exp(theta.next[2]))
  }
  # --- end iterations

  return(par.samples)
}
```

```
# draw random initial values
init.random <- function(x) {
  resampledX <- sample(x, replace = T)
  mu.init <- mean(resampledX)
  sigsq.init <- var(resampledX)
  return(list(mu = mu.init, sigsq = sigsq.init))
}

# check convergence
# 경로그림과 사후밀도함수
check.plots <- function(mcmc.samples) {
  mu.samples <- mcmc.samples[,1,]
  sigsq.samples <- mcmc.samples[,2,]

  cols <- c('gray50', 'lightblue', 'pink')
  par(mfrow = c(2, 2))
  plot(mu.samples[,1], col = cols[1], type = "l", xlab = "iteration", ylab = "", main = quote(mu))
  lines(mu.samples[,2], col = cols[2])
  lines(mu.samples[,3], col = cols[3])
  plot(density(mu.samples[,1]), col = cols[1], xlab = "", ylab = "posterior density", main = quote(mu))
  lines(density(mu.samples[,2]), col = cols[2])
  lines(density(mu.samples[,3]), col = cols[3])
  plot(sigsq.samples[,1], col = cols[1], type = "l", xlab = "iteration", ylab = "", main = quote(sigma^2))
  lines(sigsq.samples[,2], col = cols[2])
  lines(sigsq.samples[,3], col = cols[3])
  plot(density(sigsq.samples[,1]), col = cols[1], xlab = "", ylab = "posterior density", main = quote(sigma^2))
  lines(density(sigsq.samples[,2]), col = cols[2])
  lines(density(sigsq.samples[,3]), col = cols[3])
}

# gelman 상수
check.gelman <- function(mcmc.samples) {
  samples.1 <- mcmc(mcmc.samples[,,1])
  samples.2 <- mcmc(mcmc.samples[,,2])
  samples.3 <- mcmc(mcmc.samples[,,3])
  codaSamples <- mcmc.list(list(samples.1, samples.2, samples.3))
  gelman <- gelman.diag(codaSamples)
  return(gelman)
}

# 채택확률
check.accept <- function(mcmc.samples) {
  Metro.draws <- mcmc(mcmc.samples[,,1])
  accept.rate <- 1 - rejectionRate(Metro.draws)
  return(accept.rate)
}

# 자기상관함수
check.acf <- function(mcmc.samples) {
  mu.samples <- mcmc.samples[,1,]
  sigsq.samples <- mcmc.samples[,2,]
```

```
  par(mfrow = c(1, 2))
  acf(mu.samples[,1], main = quote(mu))
  acf(sigsq.samples[,1], main = quote(sigma^2))
}

# 95% HPD
infer.HPD <- function(mcmc.samples) {
  mcmc.samples.combined <- rbind(mcmc.samples[,,1], mcmc.samples[,,2], mcmc.samples[,,3])
  estimate <- apply(mcmc.samples.combined, 2, mean)
  HPD <- apply(mcmc.samples.combined, 2, function(x) quantile(x, c(0.025, 0.975)))
  out <- rbind(estimate, HPD)
  return(out)
}
```

```
# 1) preliminary run
nchains = 3
nsim = 2000 ; nburn = 0
p = 2
mcmc.samples.one <- array(0, c(nsim, p, nchains))

delta.one <- 1
for (ich in 1:nchains) {
  initList <- init.random(x)
  mcmc.samples.one[,,ich] <- metro.norm(nsim, nburn, delta.one, dataList, initList)
}

# mu와 sigsq의 추정치/분산
c(mean(mcmc.samples.one[,1,1]), mean(log(mcmc.samples.one[,2,1])))
c(var(mcmc.samples.one[,1,1]), var(log(mcmc.samples.one[,2,1])))

(delta.mu <- sd(mcmc.samples.one[,1,]))
(delta.sigsq <- sd(log(mcmc.samples.one[,2,])))

# 2)
(delta.est <- c(delta.mu, delta.sigsq) * sqrt(2.4))

# 3-4) quick MH & acceptance rate
mcmc.samples.est <- array(0, c(nsim, p, nchains))
for (ich in 1:nchains) {
  initList <- init.random(x)
  mcmc.samples.est[,,ich] <- metro.norm(nsim, nburn, delta.est, dataList, initList)
}
check.accept(mcmc.samples.est)
check.plots(mcmc.samples.est)

# 5)
delta.change <- c(3.3, 1.1)
mcmc.samples.change <- array(0, c(nsim, p, nchains))
for (ich in 1:nchains) {
  initList <- init.random(x)
  mcmc.samples.change[,,ich] <- metro.norm(nsim, nburn, delta.change, dataList, initList)
}
```

```
check.accept(mcmc.samples.change)
check.plots(mcmc.samples.change)

# 6)
nchains = 3
nsim = 25000 ; nburn = 0

#####
mcmc.samples.change <- array(0, c(nsim, p, nchains))
for (ich in 1:nchains) {
  initList <- init.random(x)
  mcmc.samples.change[,,ich] <- metro.norm(nsim, nburn, delta.change, dataList, initList)
}

c(mean(mcmc.samples.change[-(1:5000),1,1]), mean(log(mcmc.samples.change[-(1:5000),2,1])))
c(var(mcmc.samples.change[-(1:5000),1,1]), var(log(mcmc.samples.change[-(1:5000),2,1])))

check.plots(mcmc.samples.change[-(1:5000),,])
check.gelman(mcmc.samples.change[-(1:5000),,])
check.accept(mcmc.samples.change[-(1:5000),,])
check.acf(mcmc.samples.change[-(1:5000),,])
(HPD.optim <- infer.HPD(mcmc.samples.change[-(1:5000),,]))

par(mfrow = c(1, 2))
plot(density(mcmc.samples.change[-(1:5000),1,1]), col = 'gray50', main = quote(mu), xlab = '')
abline(v = HPD.optim[1,1], col = 'steelblue')
abline(v = HPD.optim[-1,1], col = 'red', lty = 2)
plot(density((mcmc.samples.change[-(1:5000),2,1])), col = 'gray50', main = quote(sigma^2), xlab = '')
abline(v = (HPD.optim[1,2]), col = 'steelblue')
abline(v = (HPD.optim[-1,2]), col = 'red', lty = 2)

check.plots(mcmc.samples.change[(1:1000),,])
check.gelman(mcmc.samples.change[(1:1000),,])

#####
mcmc.samples.one <- array(0, c(nsim, p, nchains))
for (ich in 1:nchains) {
  initList <- init.random(x)
  mcmc.samples.one[,,ich] <- metro.norm(nsim, nburn, delta = delta.one, dataList, initList)
}
c(mean(mcmc.samples.one[-(1:5000),1,1]), mean(log(mcmc.samples.one[-(1:5000),2,1])))
c(var(mcmc.samples.one[-(1:5000),1,1]), var(log(mcmc.samples.one[-(1:5000),2,1])))

check.plots(mcmc.samples.one[-(1:5000),,])
check.gelman(mcmc.samples.one[-(1:5000),,])
check.accept(mcmc.samples.one[-(1:5000),,])
check.acf(mcmc.samples.one[-(1:5000),,])
(HPD.one <- infer.HPD(mcmc.samples.one[-(1:5000),,]))

par(mfrow = c(1, 2))
plot(density(mcmc.samples.one[-(1:5000),1,1]), col = 'gray50', main = quote(mu), xlab = '')
abline(v = HPD.one[1,1], col = 'steelblue')
abline(v = HPD.one[-1,1], col = 'red', lty = 2)
```

```
plot(density(mcmc.samples.one[-(1:5000),2,1]), col = 'gray50', main = quote(sigma^2), xlab = '')
abline(v = HPD.one[1,2], col = 'steelblue')
abline(v = HPD.one[-1,2], col = 'red', lty = 2)

check.plots(mcmc.samples.one[(1:1000),,])
check.gelman(mcmc.samples.one[(1:1000),,])
```