MCMC success example

Arnab Hazra

2/27/2022

We first simulate two sets of observations $X_1, \ldots, X_m \sim \text{Normal}(\mu_X, 1)$ and $Y_1, \ldots, Y_n \sim \text{Normal}(\mu_Y, 1)$. We want to estimate μ_X and μ_Y. We choose priors $\mu_X, \mu_Y \sim \text{Normal}(0, 100^2)$.

```r
rm(list = ls())

m <- 100
n <- 100

mu.X <- 2
mu.Y <- 4

set.seed(100)

X <- rnorm(m, mean = mu.X, sd = 1)
Y <- rnorm(n, mean = mu.Y, sd = 1)

data <- list(X = X, Y = Y, m = m, n = n)
```

Next, we write the script for JAGS.

(1) Define the model as a string.

```r
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.0.5
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```r
model_string <- textConnection("model{

  # Likelihood (dnorm uses a precision, not variance)
  for(i in 1:m){
  X[i] ~ dnorm(mu.X, 1)
  }
  for(j in 1:n){
  Y[j] ~ dnorm(mu.Y, 1)
  }

  # Priors

  mu.X ~ dnorm(0, 0.0001)
```

```
  mu.Y ~ dnorm(0, 0.0001)
  }")
```

(2) Load the data and compile the MCMC code

```
inits <- list(mu.X = mean(X), mu.Y = mean(Y))
model <- jags.model(model_string, data = data, inits = inits, quiet = TRUE, n.chains = 2)
```
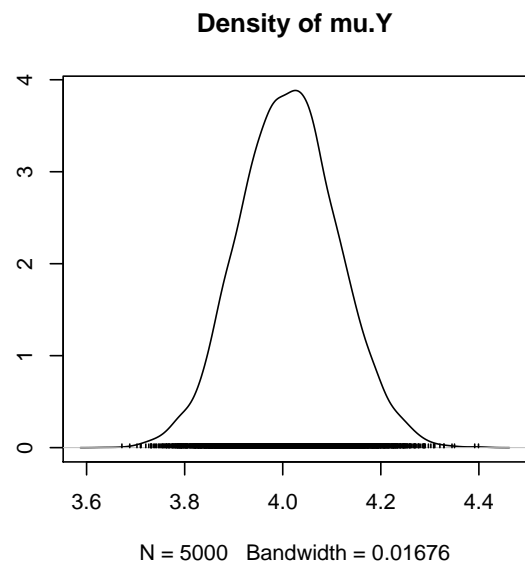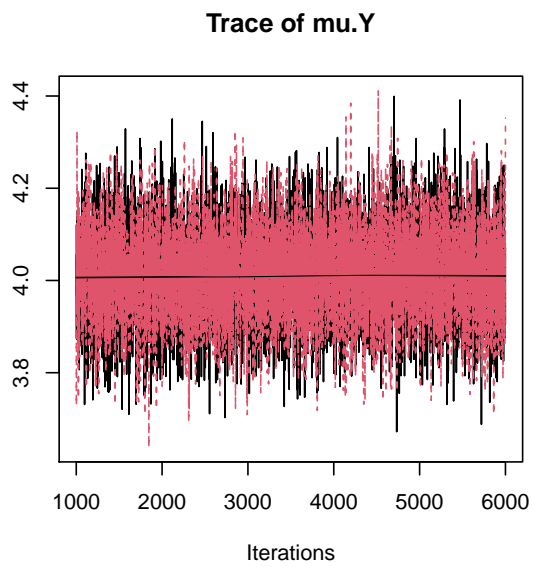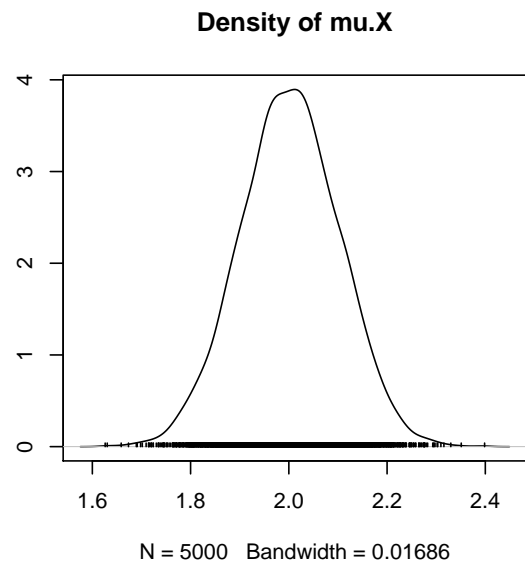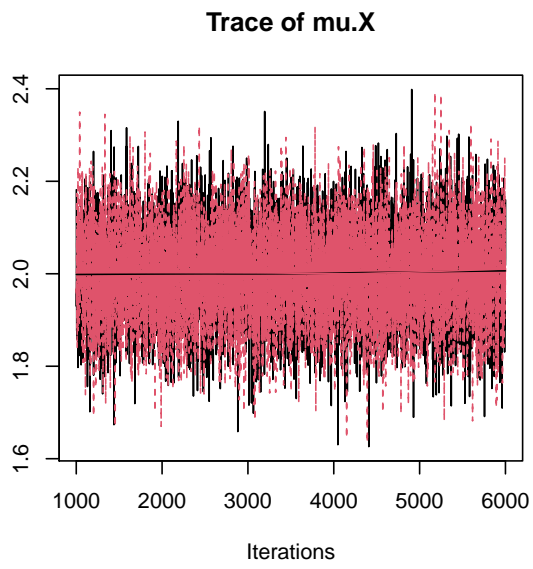
(3) Burn-in for 1000 samples

```
update(model, 1000, progress.bar = "none")
```

(4) Generate 5000 post-burn-in samples

```
params <-c("mu.X", "mu.Y")
samples <- coda.samples(model,
                        variable.names = params,
                        n.iter = 5000, progress.bar = "none")
```
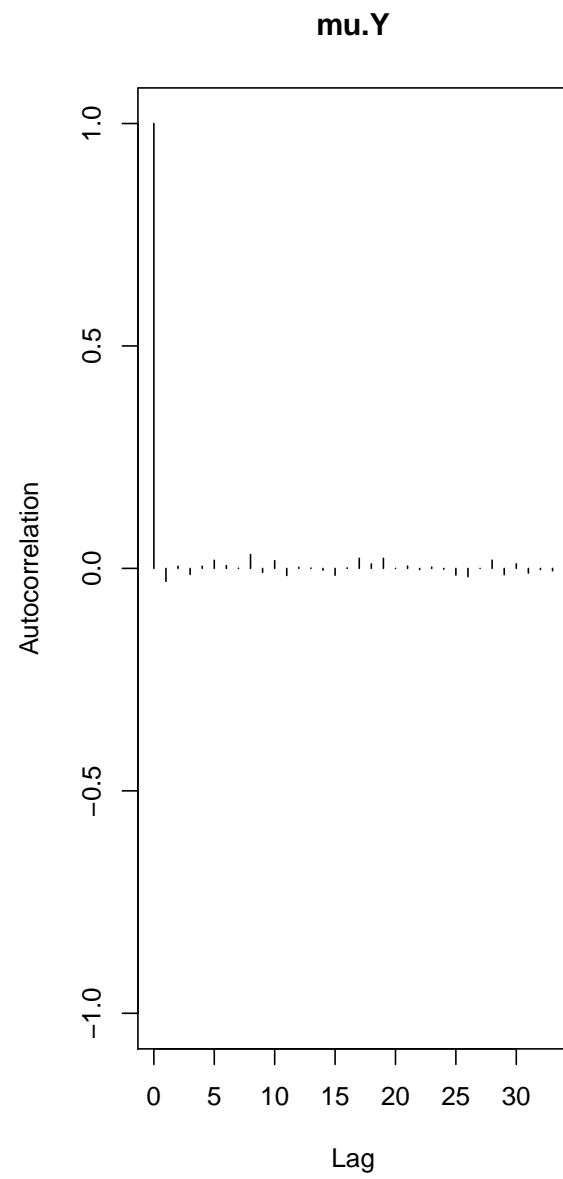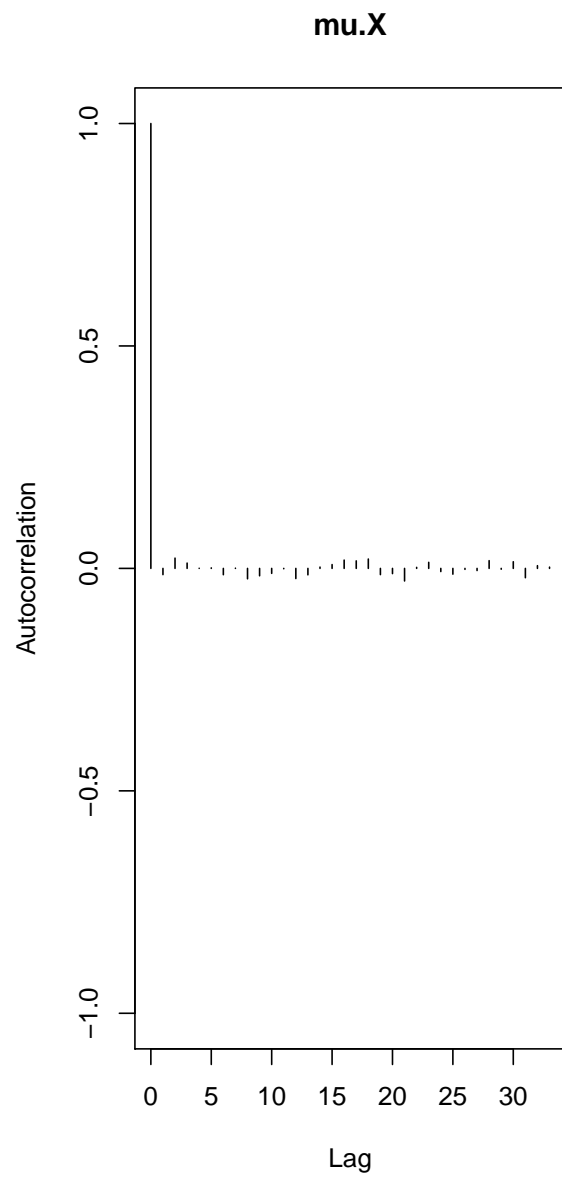
(5) Graphical diagnostics

```
plot(samples)
```

Trace of mu.X

Iterations

Density of mu.X

N = 5000   Bandwidth = 0.01686

Trace of mu.Y

Iterations

Density of mu.Y

N = 5000   Bandwidth = 0.01676
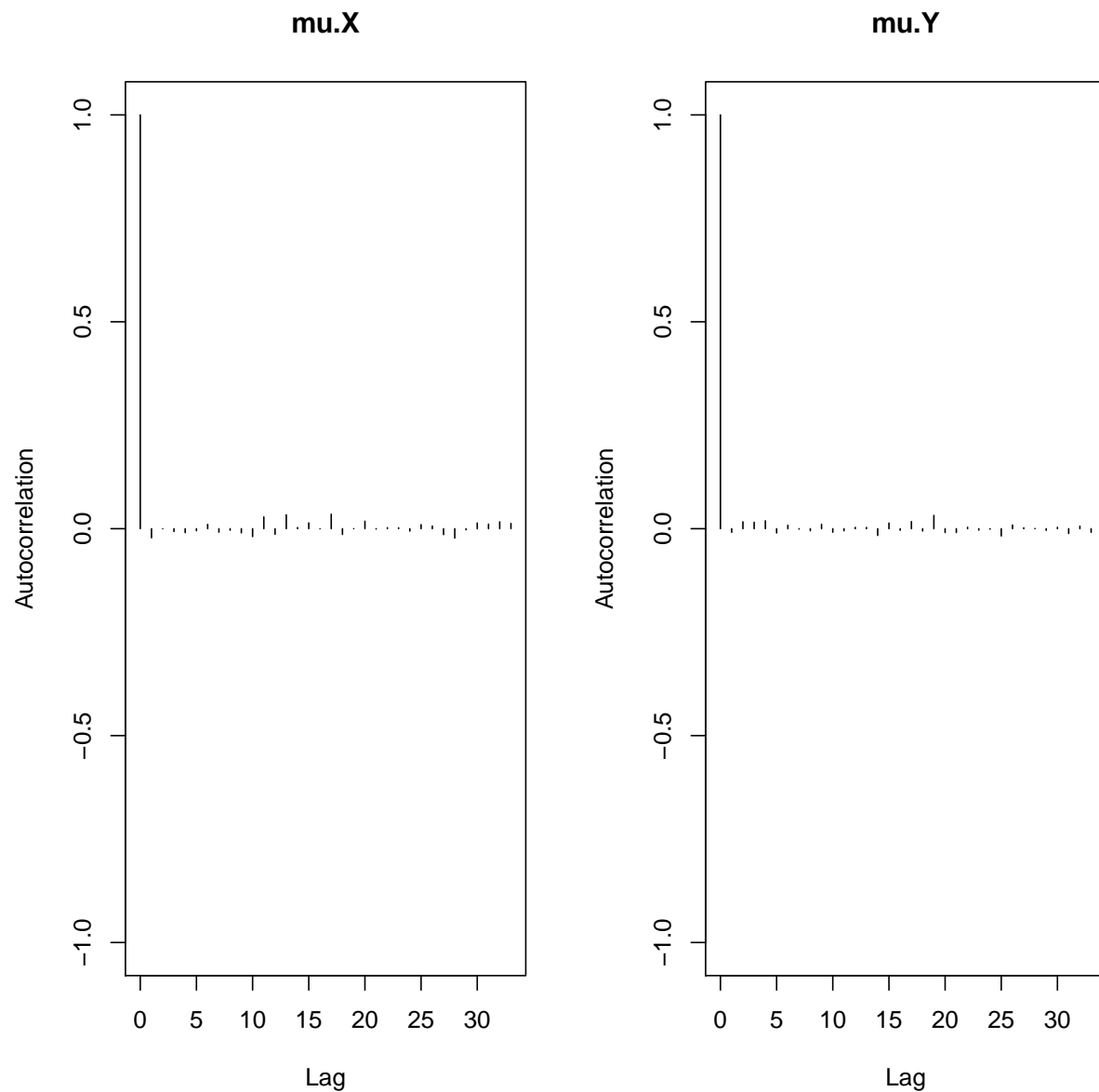
The trace plots look like caterpillars, the chains (one in black and one in gray) give similar estimates.

```
library(coda)

autocorr.plot(samples)
```

mu.X

mu.Y

4

mu.X   **mu.Y**

Autocorrelation is near zero for all lags. Together, these indicate convergence.

Numerical diagnostics

```r
# Low autocorrelation indicates convergence

autocorr(samples[[1]], lag = 1)
```

```
## , , mu.X
##
##           mu.X        mu.Y
## Lag 1 -0.0140107 0.005160406
##
## , , mu.Y
##
```

```
##              mu.X          mu.Y
## Lag 1 0.02106957 -0.02915034
```

```
# high ESS indicates convergence

effectiveSize(samples)
```

```
##     mu.X     mu.Y
## 10223.69 10299.20
```

```
# R less than 1.1 indicates convergence

gelman.diag(samples)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## mu.X          1          1
## mu.Y          1          1
##
## Multivariate psrf
##
## 1
```

```
# |z| less than 2 indicates convergence

geweke.diag(samples[[1]])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   mu.X   mu.Y
## -1.662 -1.146
```