MCMC failure example

Arnab Hazra

2/27/2022

We first simulate two sets of observations $X_1, \ldots, X_n \sim \text{Normal}(\mu_X, 1)$ and $Y_1, \ldots, Y_n \sim \text{Normal}(\mu_Y, 1)$. However, suppose these data are not observable and we only know $Z_i = X_i + Y_i, i = 1, \ldots, n$. Thus, $Z_i \sim \text{Normal}(\mu_X + \mu_Y, 2)$. We want to estimate μ_X and μ_Y. We choose priors $\mu_X, \mu_Y \sim \text{Normal}(0, 100^2)$.

```
rm(list = ls())

n <- 100

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

set.seed(100)

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

data <- list(Z = Z, n = n)
```

Next, we write the script for JAGS.

(1) Define the model as a string.

```
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
```

```
model_string <- textConnection("model{

  # Likelihood (dnorm uses a precision, not variance)
  for(i in 1:n){
  Z[i] ~ dnorm(mu.X + mu.Y, 1 / 2)
  }

  # 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(Z)/2, mu.Y = mean(Z)/2)
model <- jags.model(model_string, data = data, inits = inits, quiet = TRUE, n.chains = 2)
```

(3) Burn-in for 1000 samples

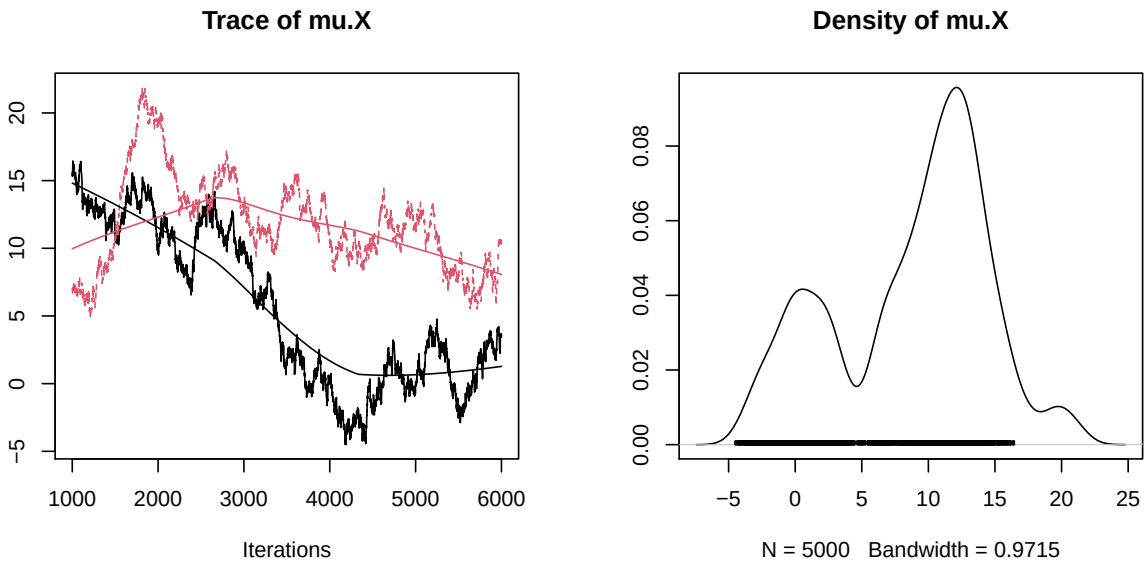
```
update(model, 1000, progress.bar = "none")
```

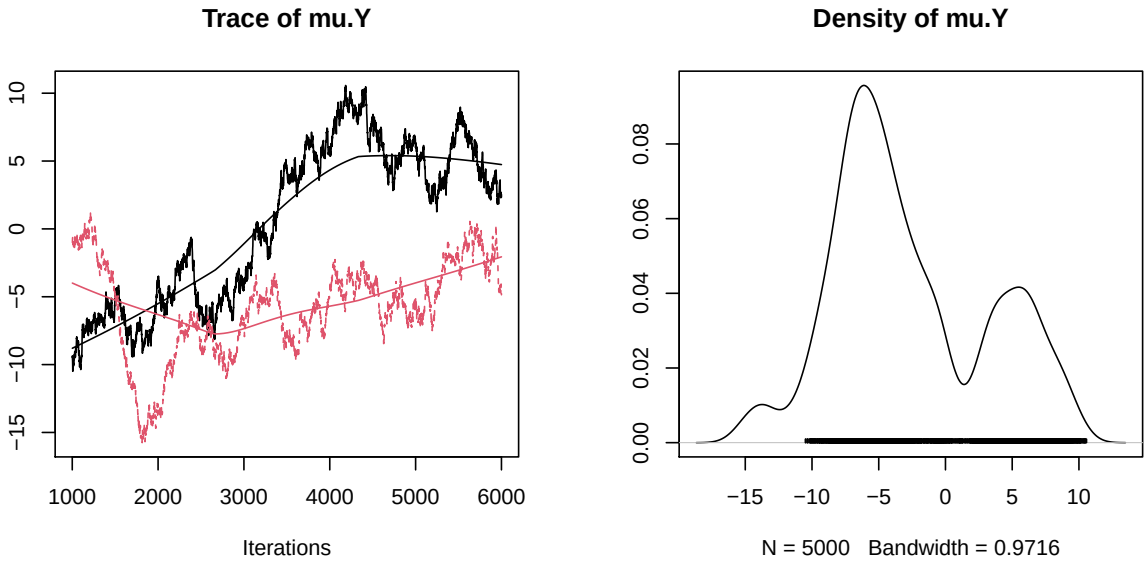
(4) Generate 5000 post-burn-in samples

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

(5) Graphical diagnostics

```
plot(samples)
```

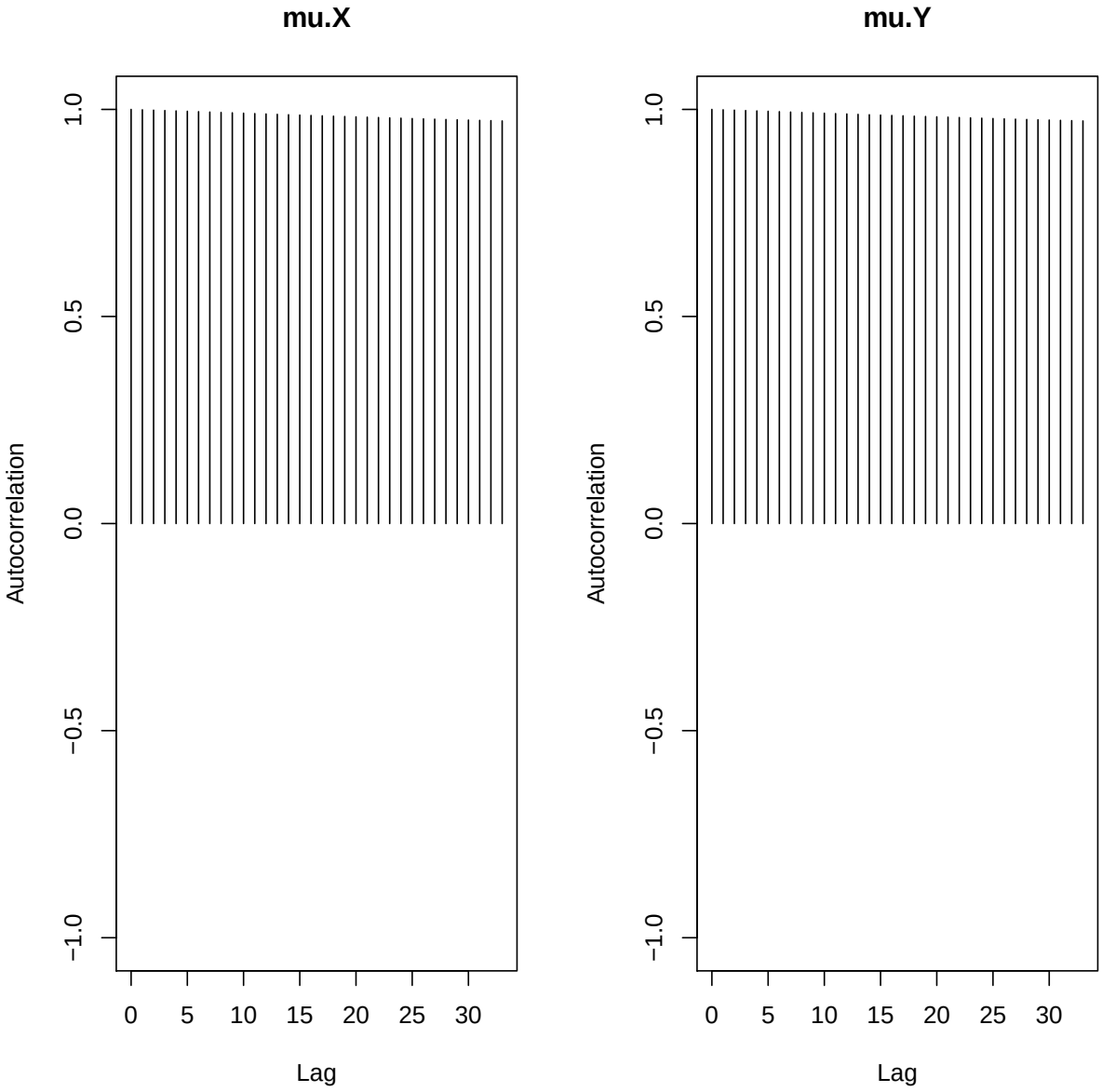


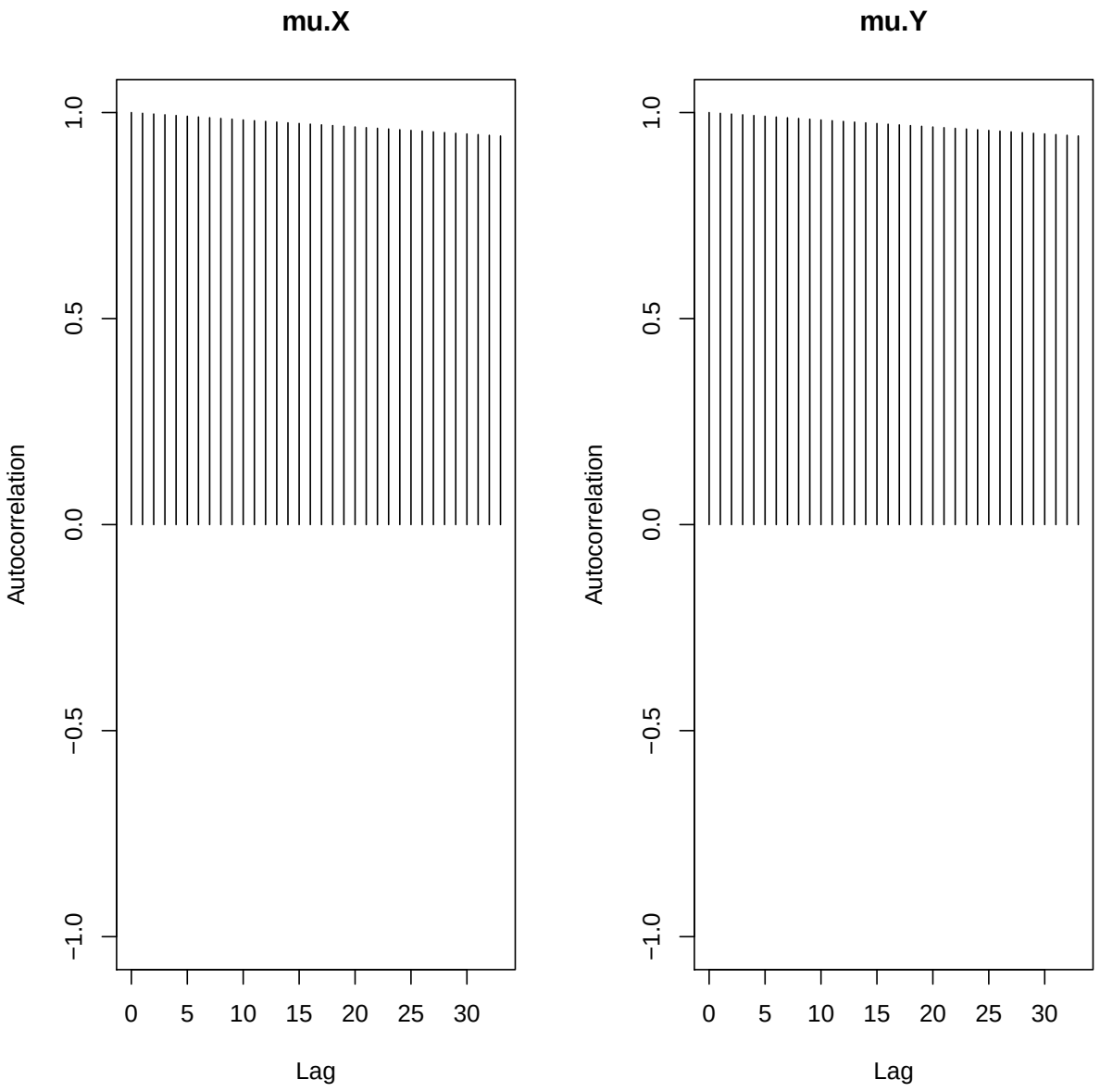


The trace plots do not look like caterpillars, the chains (one in black and one in gray) do not give similar estimates.

```
library(coda)

autocorr.plot(samples)
```


mu.X
mu.Y
Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30



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

Numerical diagnostics

```
# Low autocorrelation indicates convergence

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

```
## , , mu.X
##
##            mu.X      mu.Y
## Lag 1 0.9990959 -0.998801
##
## , , mu.Y
##
```

```
##            mu.X      mu.Y
## Lag 1 -0.999394 0.9991072
```

```
# high ESS indicates convergence

effectiveSize(samples)
```

```
##     mu.X     mu.Y
## 6.684242 6.872405
```

```
# R less than 1.1 indicates convergence

gelman.diag(samples)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## mu.X       4.55       10.7
## mu.Y       4.55       10.7
##
## Multivariate psrf
##
## 3.3
```

```
# |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
##  13.15 -13.17
```