STA 602 - Intro to Bayesian Statistics

Lecture 11

Li Ma

Duke University

Monte Carlo Markov Chain Diagnostics

- We have so far seen our first example of an MCMC sampler—the Gibbs sampler.
- Generally, an MCMC sampler generates a sequence of draws that are correlated

$$\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$$

- We want to apply the "law of large number" on these correlated samples to get an estimate of the integral

$$\mathrm{E}_p g = \int g(u)p(u)du$$

by

$$\frac{1}{S}\sum_{i=1}^{S} g(\boldsymbol{\theta}^{(i)}) \to \mathrm{E}_p g.$$

- But when is this Monte Carlo approximation good?
- This is a harder question to answer than for standard Monte Carlo where the samples are independent. Why?

Convergence and mixing

- In order to have small error in the approximation, we need to have two things happen
  - Convergence—the chain has moved into a high probability region of the target distribution after a burn-in period, which should be discarded.
  - Good mixing—the chain should be able to move across high probability regions of the target distribution with ease, rather than getting stuck in one high probability region for a long time, and only jump to another very infrequently.

Convergence and mixing

- Convergence reduces the bias in our integral evaluation—we are in fact applying the LLN to the correct distribuiton p.
- Good mixing means that we give "deep" coverage on all important regions of the target distribution. Poor mixing can lead to
  - large Monte Carlo errors (assuming the chain is run long enough)
  - completely wrong estimate if the chain had barely visited some high probability regions
- How do we tell?
  - Strictly speaking, there is no rigorous way to tell with certainty if convergence and/or good mixing have been achieved. (The simple scenario like in the bivariate normal example doesn't occur in practice.)
  - We can only tell if there is clear deviation from convergence and proper mixing using heuristic strategies.
  - In particular, the chain could well be trapped around a local mode and looks as if it had converged.

Some heuristic strategies to check clear deviation

- Assessing convergence and mixing:
  - Traceplot (unbinned or binned)
  - Auto-correlation.
  - Common summary/test statistics applied to the MCMC chain.

Trace plots

- For each parameter, plot their value drawn in each iteration versus the index of iterations.
- A "good-looking" trace plot should not have
  - Drifting patterns (indicating non-convergence).
  - "S" patterns (indicating poor mixing). This can often occur as the MCMC moves are very small.
  - Infrequent jumps across regions (indicating very poor mixing).

Some examples of trace plots

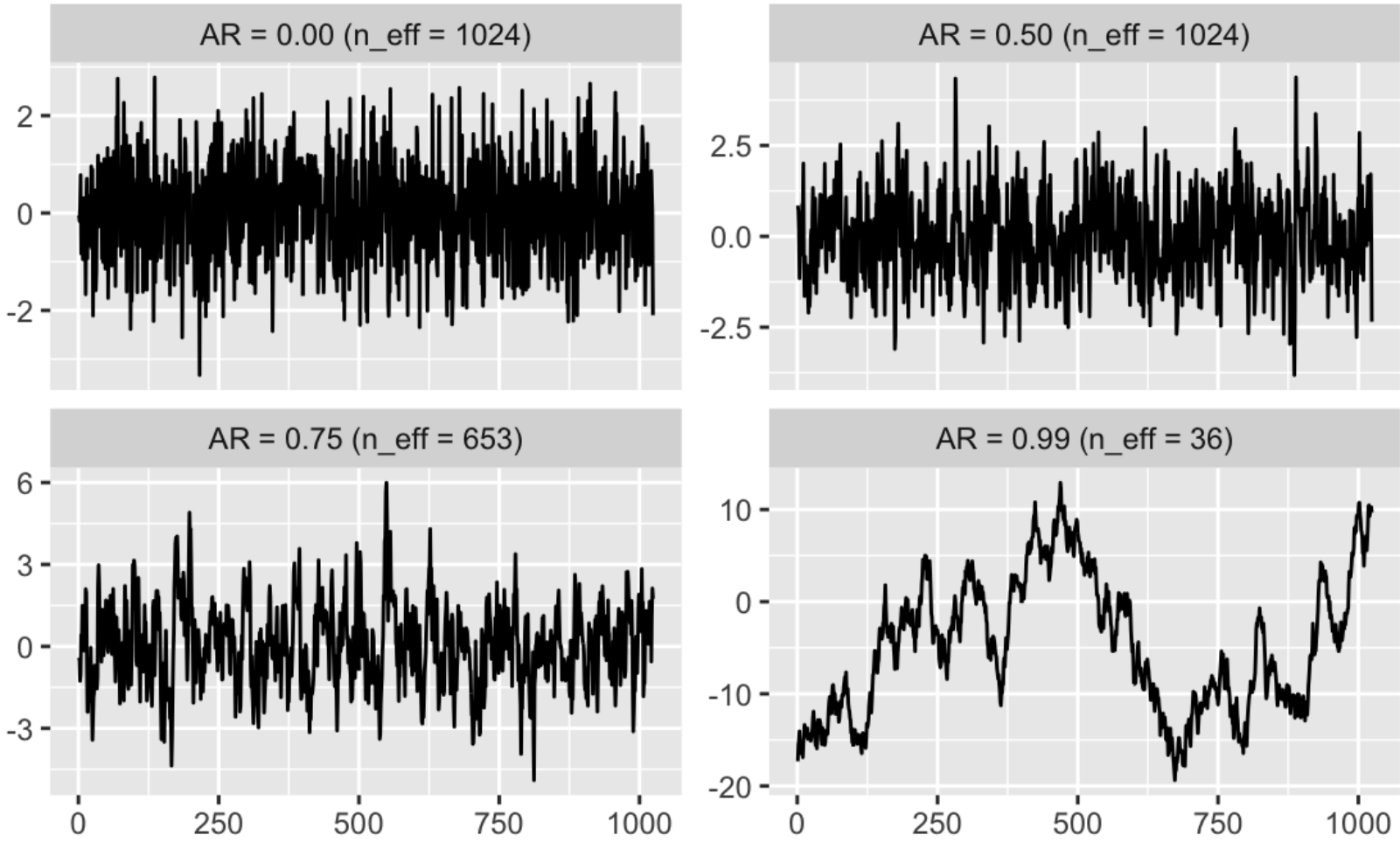


Figure 1: Source: https://jrnold.github.io

Autocorrelation of the MCMC chain

- Autocorrelation quantifies how similar (sticky) adjacent values in the MCMC chains are.
  - Note that variables in a Markov Chain depends on the past only through the present.
  - There will still be "marginal" correlation across draws (without conditioning on the present).
  - Such auto-correlation decays with the lag.
- Lag-k autocorrelation for evaluating $\mathrm{E}_p g = \int g(\theta) p(\theta) d\theta$:

$$\begin{aligned}
\mathrm{acf}_k(g) &= \rho_k(g) \\
&= \widehat{\mathrm{corr}}(g^{(t)}, g^{(t+k)}) \\
&= \frac{\frac{1}{S-k}\sum_{t=1}^{S-k}(g^{(t)} - \bar{g})(g^{(t+k)} - \bar{g})}{\frac{1}{S-1}\sum_{t=1}^{S}(g^{(t)} - \bar{g})^2}.
\end{aligned}$$

where $g^{(t)} = g(\theta^{(t)})$ and $\bar{g} = \frac{1}{S}\sum_{t=1}^{S} g(\theta^{(t)})$.

Effective sample size (ESS)

- The ESS is the sample size under standard Monte Carlo samples that would give the same variance as the MCMC estimate:

$$\mathrm{Var}\bar{g} = \frac{\mathrm{Var}_p g}{S_{eff}}.$$

- The variance of an MCMC estimate $\bar{g}$ for $\mathrm{E}_p g$ is given by

$$\begin{aligned}
\mathrm{Var}\bar{g} &= \mathrm{E}\left[(\bar{g} - \mathrm{E}_p g)^2\right] \\
&= \mathrm{E}\left[\left(\frac{1}{S}\sum_t g^{(t)} - \mathrm{E}_p g\right)^2\right] = \frac{1}{S^2}\mathrm{E}\left[\sum_{s,t}(g^{(t)} - \mathrm{E}_p g)(g^{(s)} - \mathrm{E}_p g)\right] \\
&= \frac{1}{S^2}\mathrm{E}\left[\sum_{t=1}^{S}(g^{(t)} - \mathrm{E}_p g)^2 + \sum_{t\neq s}(g^{(t)} - \mathrm{E}_p g)(g^{(s)} - \mathrm{E}_p g)\right] \\
&= \frac{1}{S^2}\sum_{t=1}^{S}\mathrm{E}\left[(g^{(t)} - \mathrm{E}_p g)^2\right] + \frac{1}{S^2}\sum_{t\neq s}\mathrm{E}\left[(g^{(t)} - \mathrm{E}_p g)(g^{(s)} - \mathrm{E}_p g)\right] \\
&= \frac{1}{S}\mathrm{E}\left[(g^{(t)} - \mathrm{E}_p g)^2\right] + \frac{1}{S^2}\sum_{t\neq s}\mathrm{E}\left[(g^{(t)} - \mathrm{E}_p g)(g^{(s)} - \mathrm{E}_p g)\right]
\end{aligned}$$

Effective sample size (ESS)

- Note that in the first term

$$\frac{1}{S} \underbrace{\mathrm{E}\left[(g^{(t)} - \mathrm{E}_p g)^2\right]}_{\text{expectation under } p \text{ when MCMC has converged}} = \frac{\mathrm{Var}_p g}{S}.$$

  because under convergence, the marginal distribution of $\theta^{(t)}$ is exactly p.
- Thus this term is exactly the variance of the sample mean (of sample size S) from a standard Monte Carlo (i.e., i.i.d. sampling).

Effective sample size (ESS)

- The second term,

$$\frac{1}{S^2}\sum_{t\neq s}\mathrm{E}\left[(g^{(t)}-\mathrm{E}_p g)(g^{(s)}-\mathrm{E}_p g)\right]$$

$$=\frac{1}{S^2}\cdot 2\sum_{k=1}^{S-1}\sum_{t=1}^{S-k}\mathrm{E}\left[(g^{(t)}-\mathrm{E}_p g)(g^{(t+k)}-\mathrm{E}_p g)\right]$$

$$=\frac{2}{S^2}\cdot\sum_{k=1}^{S-1}(S-k)\cdot\mathrm{Cov}(g^{(t)},g^{(t+k)})$$

$$=\frac{2}{S^2}\cdot\sum_{k=1}^{S-1}(S-k)\cdot\mathrm{corr}(g^{(t)},g^{(t+k)})\cdot\mathrm{Var}_p g$$

$$\approx\frac{2}{S}\sum_{k=1}^{S-1}\mathrm{corr}(g^{(t)},g^{(t+k)})\cdot\mathrm{Var}_p g.$$

- The last approximation is based on the assumption that the Lag-k autocorrelation decays sufficiently fast with k and so only for small k, the contribution is non-negligible.

Effective sample size (ESS)

- The lag-k autocorrelation $\text{corr}(g^{(t)}, g^{(t+k)})$ can be estimated by $\text{acf}_k(g)$.
- The equivalent sample size (ESS), S_{eff}, is one such that
- Thus

$$\text{Var}\bar{g} = \frac{\text{Var}_p g}{S_{eff}(g)} \quad \text{and so} \quad S_{eff}(g) = \frac{\text{Var}_p g}{\text{Var}\bar{g}}.$$

- Based on the previous calculation,

$$\begin{aligned} S_{eff}(g) &\approx \frac{\text{Var}_p g}{\frac{\text{Var}_p g}{S} + \frac{2}{S}\text{Var}_p g \cdot \sum_k \text{acf}_k(g)} \\ &\approx \frac{S}{1 + 2\sum_k \text{acf}_k(g)}. \end{aligned}$$

- The larger the autocorrelations, the larger the variance of the MCMC estimate, and the smaller the ESS.

Back to the bivariate normal example

```
mu <- c(2,2) # mean
rho <- 0.5 # correlation - change this to different values
Sigma <- matrix(c(1,rho,rho,1),ncol=2); # covariance

S <- 1000
theta.mc <- matrix(0,nrow=S,ncol=2)
colnames(theta.mc) <- c("theta1","theta2")
theta <- c(0,0) # initial value
theta.prev <- theta

# Gibbs sampling
for (t in 1:S) {
  theta[1] <- rnorm(1,mean=mu[1]+rho*(theta[2]-mu[2]),sd=sqrt(1-rho^2))
  theta[2] <- rnorm(1,mean=mu[2]+rho*(theta[1]-mu[1]),sd=sqrt(1-rho^2))
  theta.mc[t,] <- theta
  theta.prev <- theta
}
```

For $\rho = 0.5$

```
library(coda)
theta.coda <- mcmc(theta.mc, start = 1) # no burn-in steps
options(digits=3)
summary(theta.coda)
```

```
##
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 1
## 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
## theta1 1.98 1.01    0.032         0.0516
## theta2 1.97 1.04    0.033         0.0437
##
## 2. Quantiles for each variable:
##
##             2.5%  25%  50%  75% 97.5%
## theta1  0.05224 1.27 2.00 2.66  3.85
## theta2 -0.00985 1.26 1.99 2.66  4.06
```

For $\rho = 0.5$

```
autocorr(theta.coda)
```

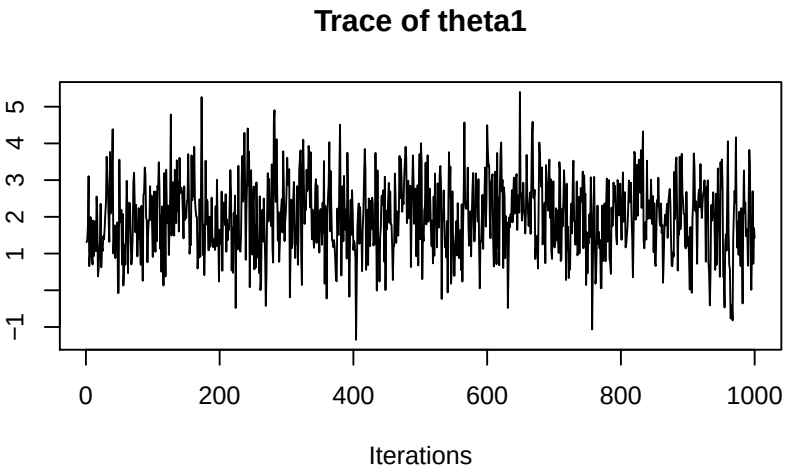
```
## , , theta1
##
##         theta1  theta2
## Lag 0   1.0000  0.5187
## Lag 1   0.2679  0.1362
## Lag 5  -0.0204 -0.0316
## Lag 10  0.0746  0.0200
## Lag 50 -0.0429 -0.0302
##
## , , theta2
##
##          theta1   theta2
## Lag 0   0.51868  1.00000
## Lag 1   0.51994  0.27462
## Lag 5  -0.00384 -0.01858
## Lag 10  0.08484  0.02438
## Lag 50 -0.01612  0.00618
```

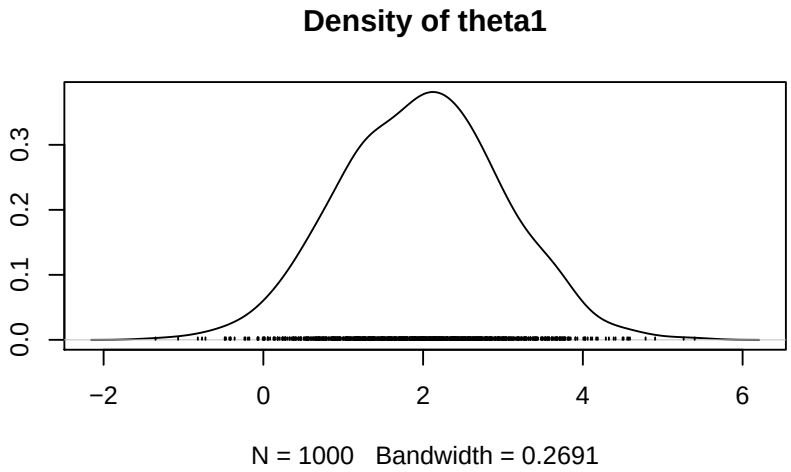
```
effectiveSize(theta.coda)
```

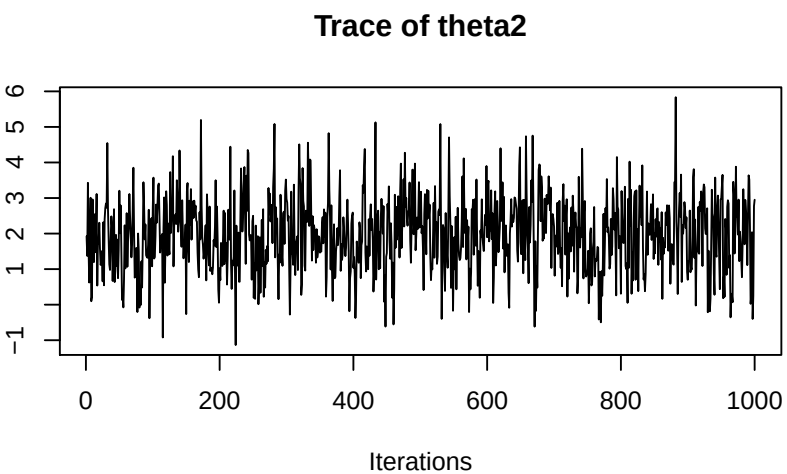
```
## theta1 theta2
##    383    569
```

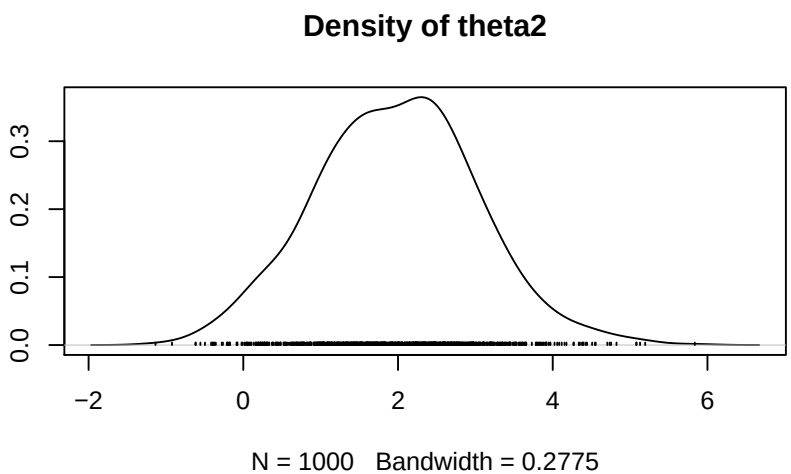
For $\rho = 0.5$

```
plot(theta.coda)
```



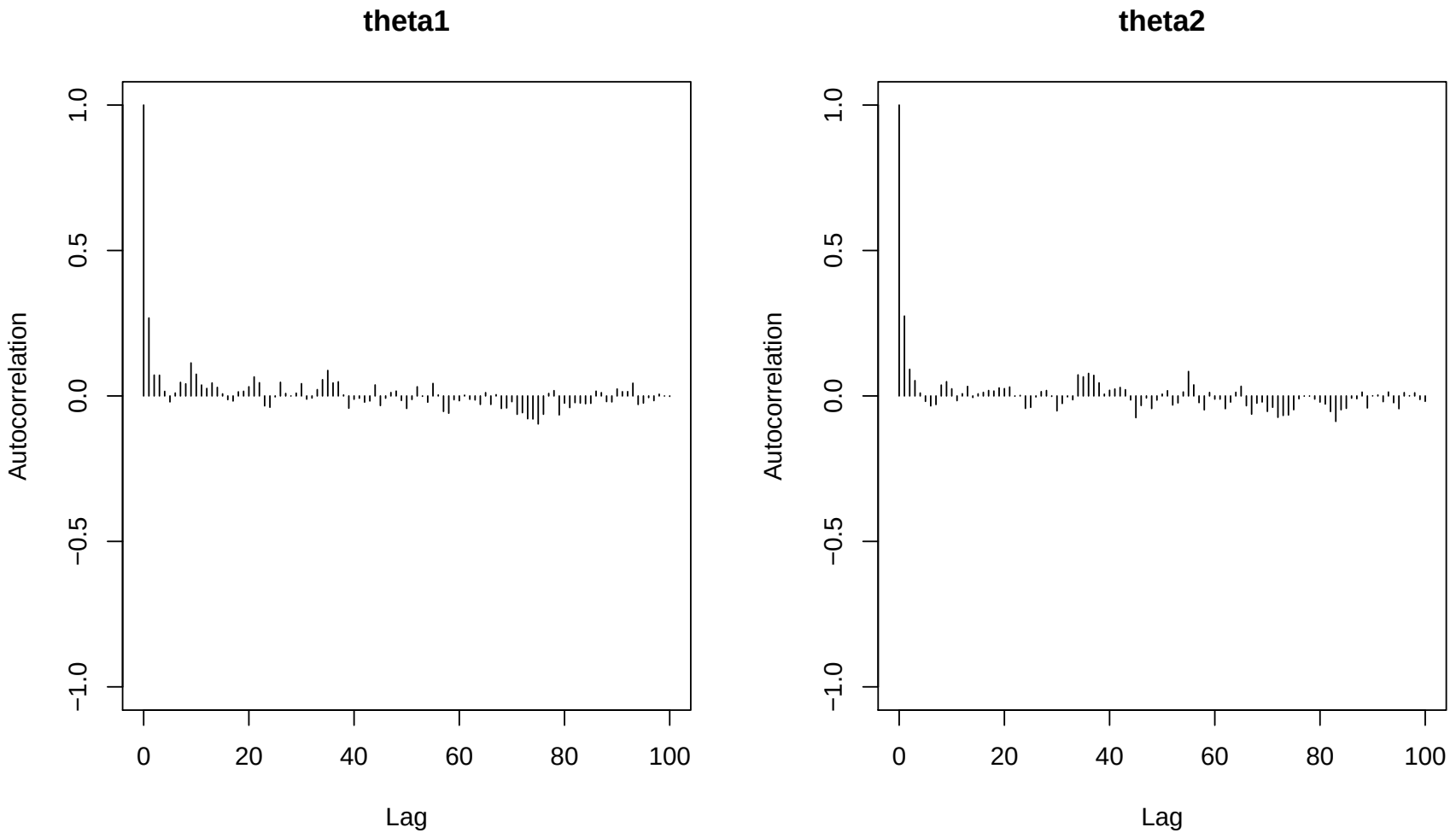






For $\rho = 0.5$

```
autocorr.plot(theta.coda,lag.max=100)
```



For $\rho = 0.99$

```r
library(coda)
theta.coda <- mcmc(theta.mc, start = 1) # no burn-in steps
options(digits=3)
summary(theta.coda)
```

```
##
## Iterations = 1:2000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean    SD Naive SE Time-series SE
## theta1 1.73 0.932   0.0208          0.214
## theta2 1.73 0.930   0.0208          0.179
##
## 2. Quantiles for each variable:
##
##           2.5%  25%  50%  75% 97.5%
## theta1 -0.0518 1.12 1.75 2.37  3.36
## theta2 -0.1001 1.11 1.74 2.37  3.34
```

For $\rho = 0.99$

```
autocorr(theta.coda)
```

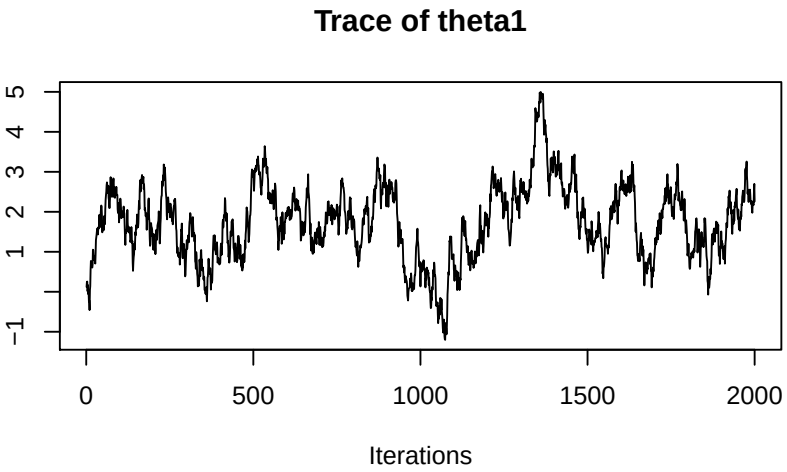
```
## , , theta1
##
##        theta1 theta2
## Lag 0   1.000  0.989
## Lag 1   0.977  0.965
## Lag 5   0.885  0.874
## Lag 10  0.770  0.762
## Lag 50  0.313  0.309
##
## , , theta2
##
##        theta1 theta2
## Lag 0   0.989  1.000
## Lag 1   0.988  0.977
## Lag 5   0.896  0.884
## Lag 10  0.781  0.772
## Lag 50  0.318  0.315
```

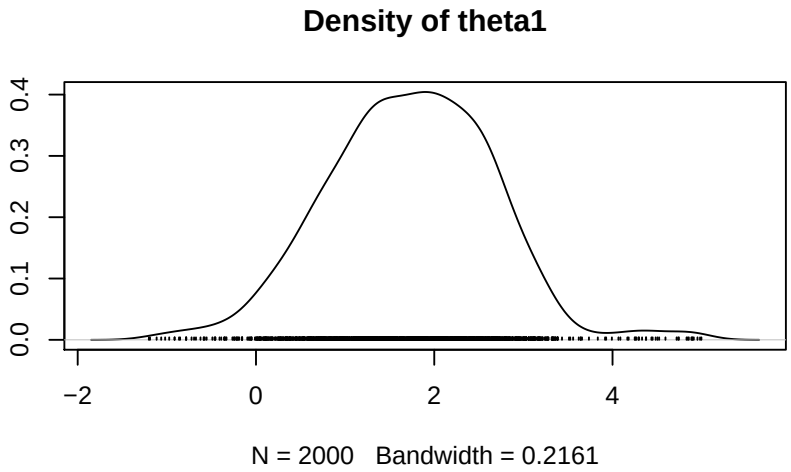
```
effectiveSize(theta.coda)
```

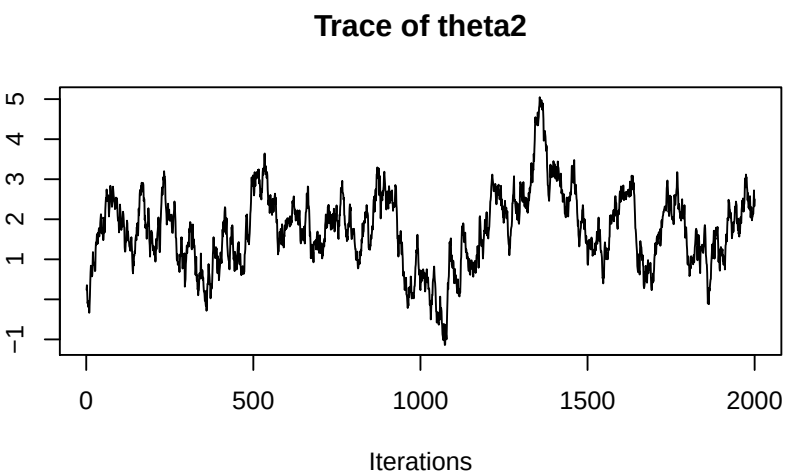
```
## theta1 theta2
##   19.0   27.1
```

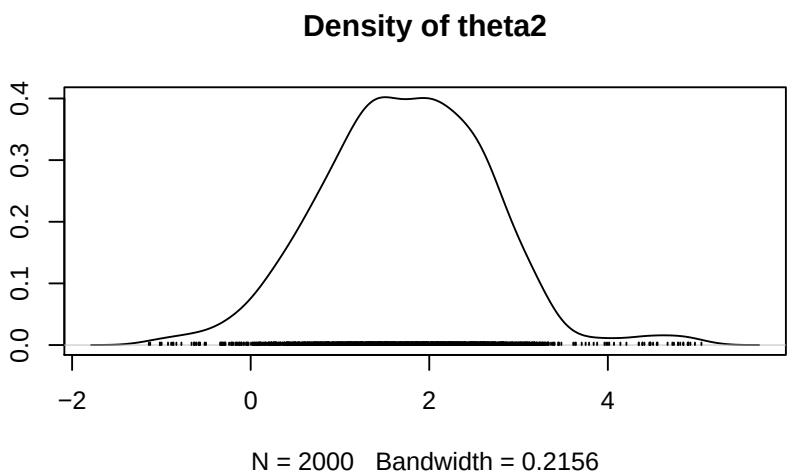
For $\rho = 0.99$

```
plot(theta.coda)
```



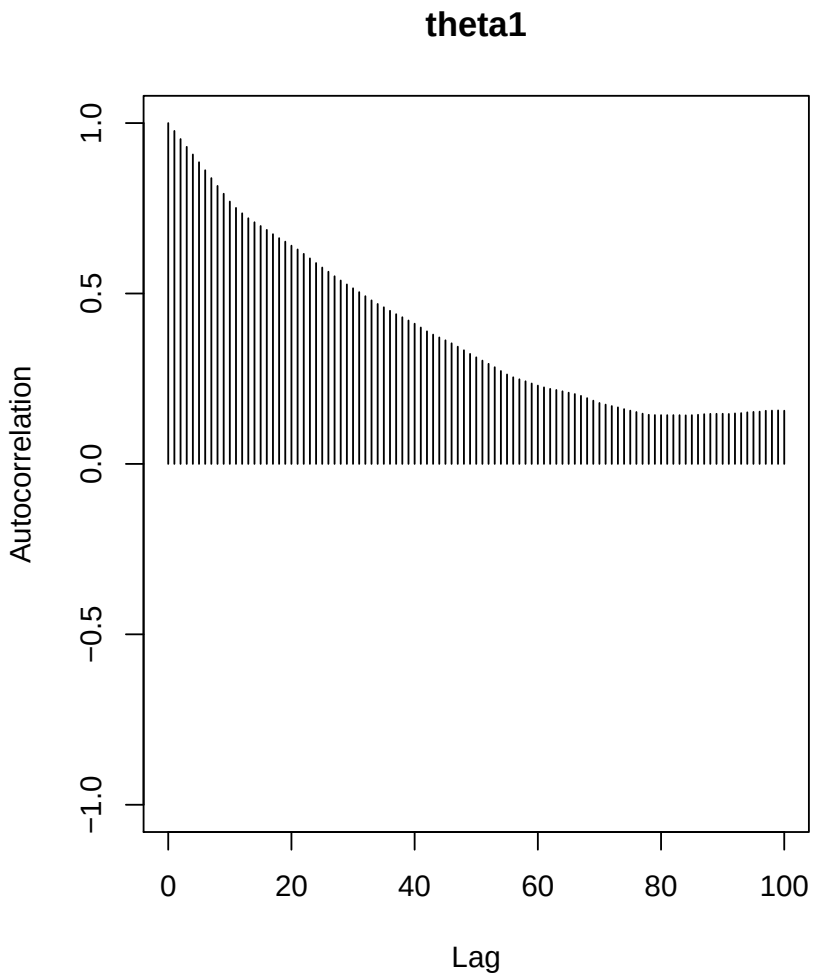


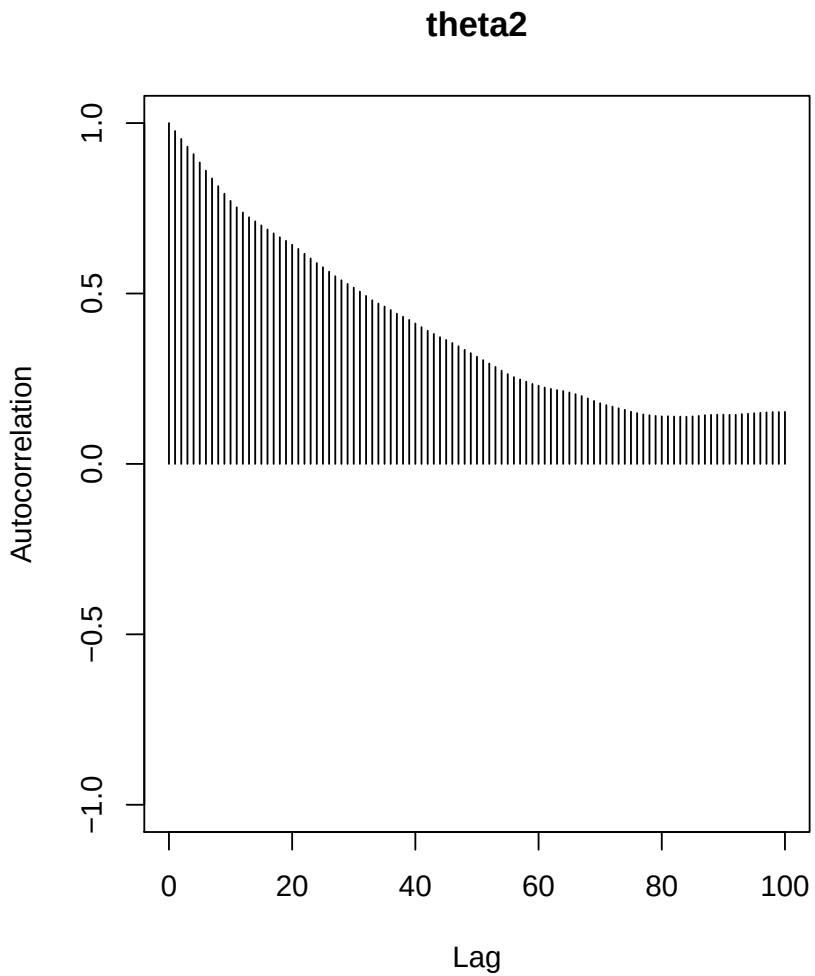




For $\rho = 0.99$

```
autocorr.plot(theta.coda,lag.max=100)
```





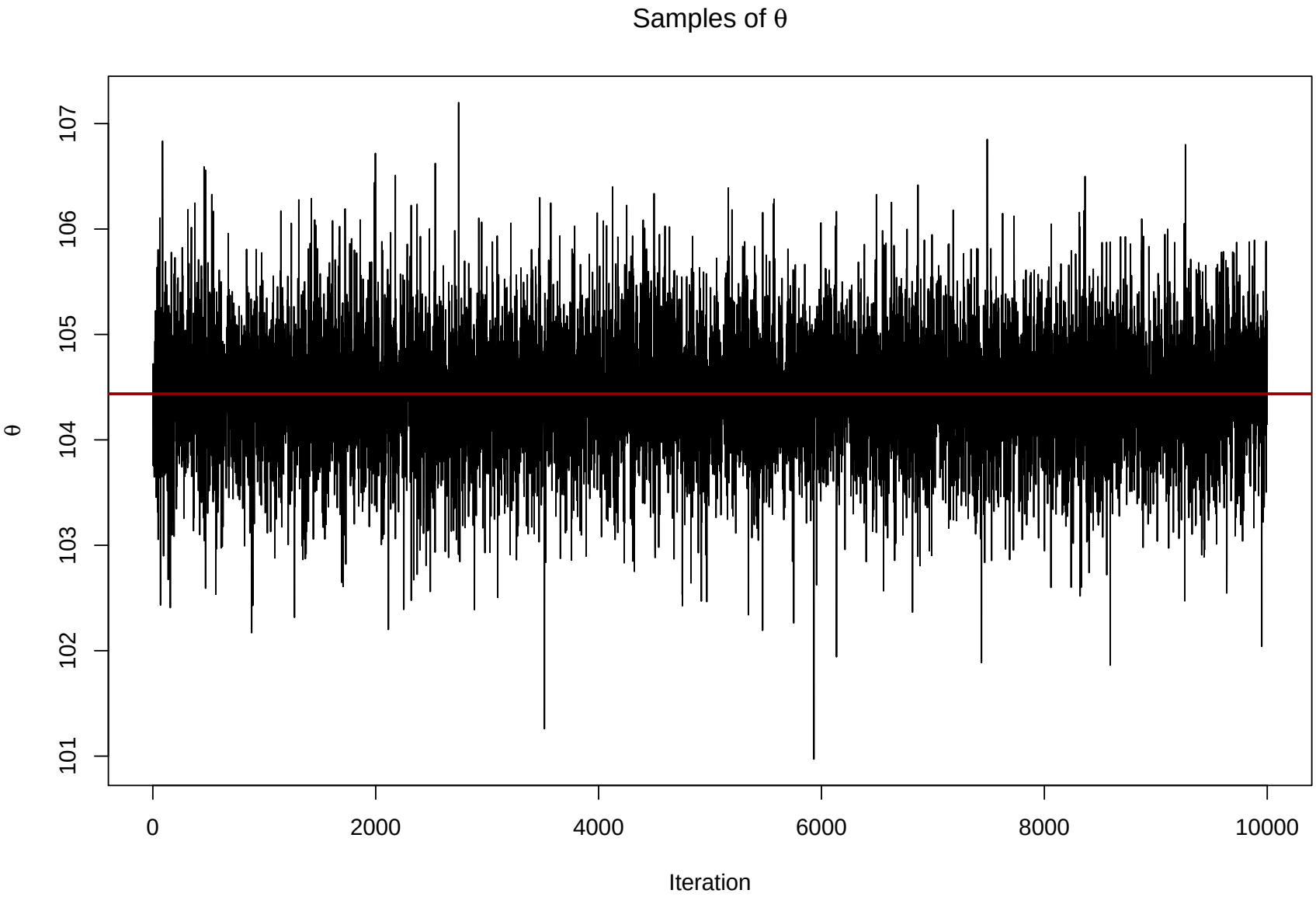
Thinning the chain

- Sometimes when we run the MCMC very long, there might be too many iterations that need to be saved.
- Can we save just a subset of the chain with the least impact on the resulting Monte Carlo error?
- Save every few iterations rather than every iteration.
- The stronger the auto-correlation of the chain, the more steps we can skip to maintain an efficient sample size.

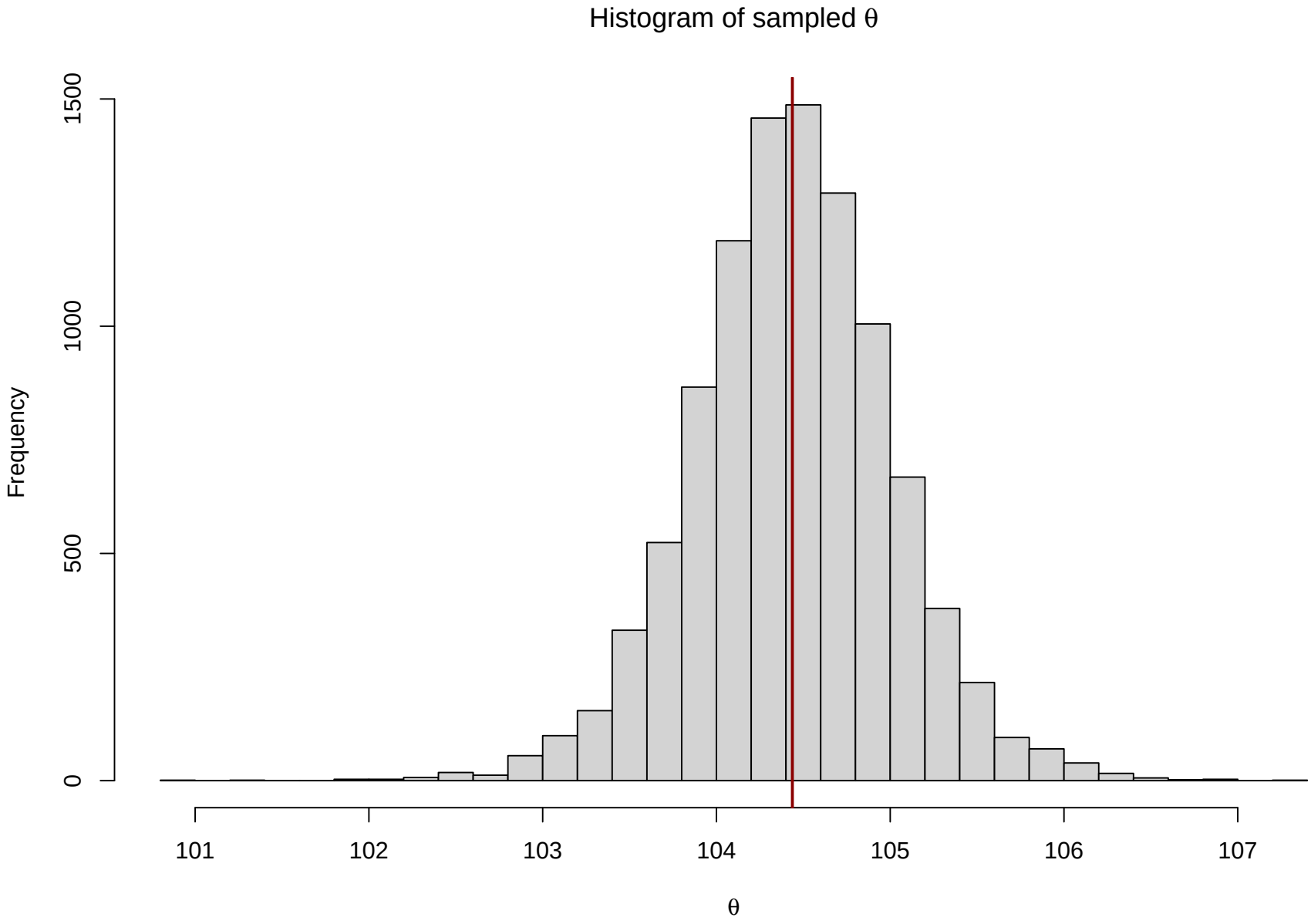
Example: Air pollutant measurements

```
x <- c(104,105,103,102,105,107,106,104,103,106) # the data
n <- length(x) # sample size

S <- 10000
xbar <- mean(x)
s2 <- var(x)
n <- length(x)

THETA <- matrix(NA,nrow=S,ncol=2,dimnames=list(1:S,c("theta","sigma2")))
THETA.init <- c(xbar,s2) # Initial values set to the MLE
THETA.curr <- THETA.init # the parameter values at the current iteration
mu.0 <- 100; nu.0 <- 1; tau2.0 <- 25; sigma2.0 <- 4

### Start Gibbs sampling
for (t in 1:S) {
  tau2.n <- 1/(1/tau2.0 + n/THETA.curr[2])
  mu.n <- (mu.0/tau2.0 + xbar*n/THETA.curr[2])/(1/tau2.0 + n/THETA.curr[2])

  ## Update theta
  THETA.curr[1] <- rnorm(1,mean=mu.n,sd=sqrt(tau2.n))

  ## Update sigma2
  THETA.curr[2] <- 1/rgamma(1,shape=(nu.0+n)/2,
                            rate=1/2*(nu.0*sigma2.0+sum((x-THETA.curr[1])^2)))

  ## Save the current iteration
  THETA[t,] <- THETA.curr
}
```

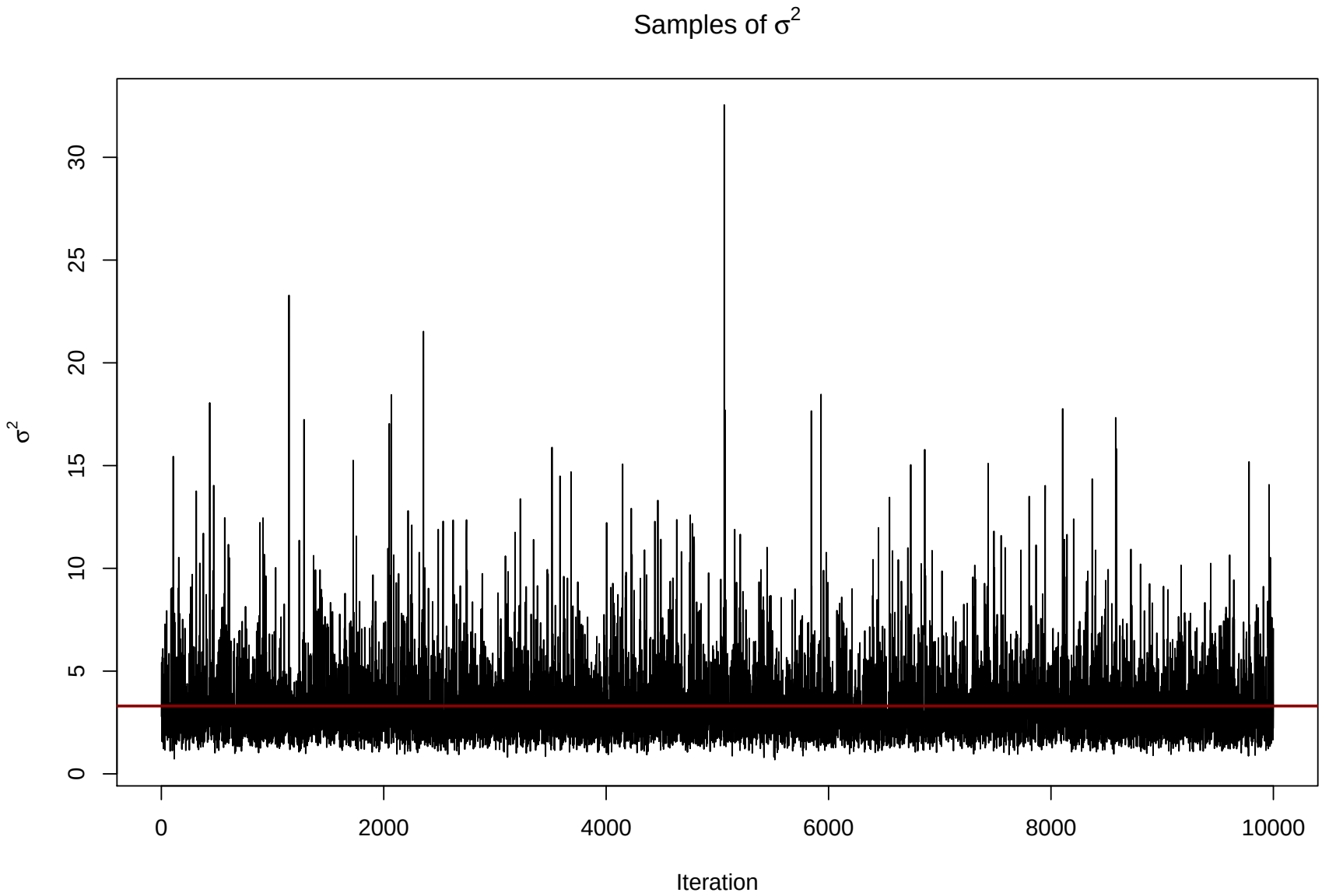
Trace plot for θ



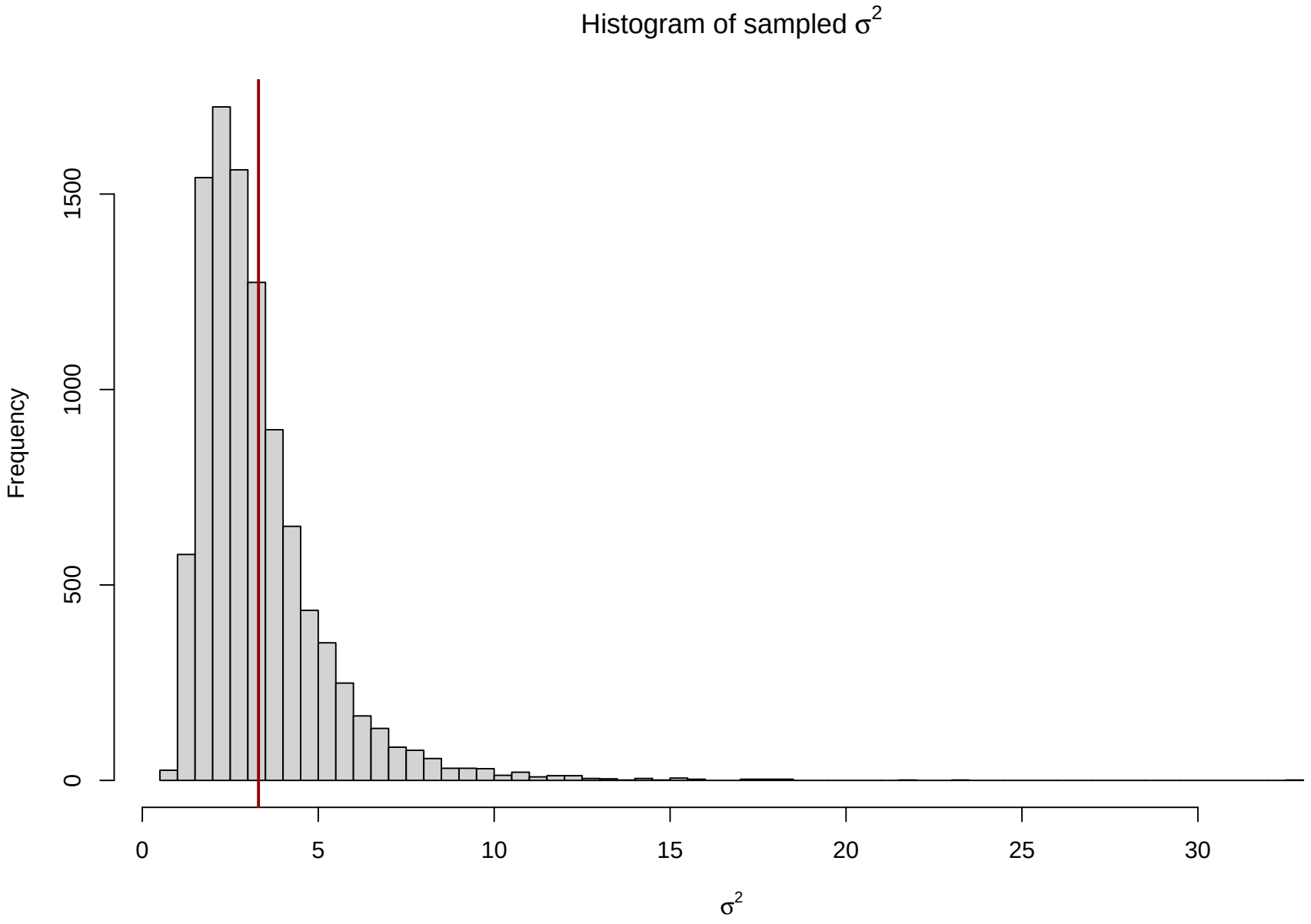
Histogram for θ



Trace plot for σ^2



Histogram for σ^2



MCMC diagnostics

```
library(coda)
THETA.coda <- mcmc(THETA, start = 1) # no burn-in steps
options(digits=3)
summary(THETA.coda)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean    SD Naive SE Time-series SE
## theta  104.4 0.575  0.00575        0.00575
## sigma2   3.3 1.848  0.01848        0.02020
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75%  97.5%
## theta  103.28 104.08 104.44 104.80 105.57
## sigma2   1.31   2.11   2.85   3.94   8.02
```

Autocorrelation and ESS

```
autocorr(THETA.coda)
```

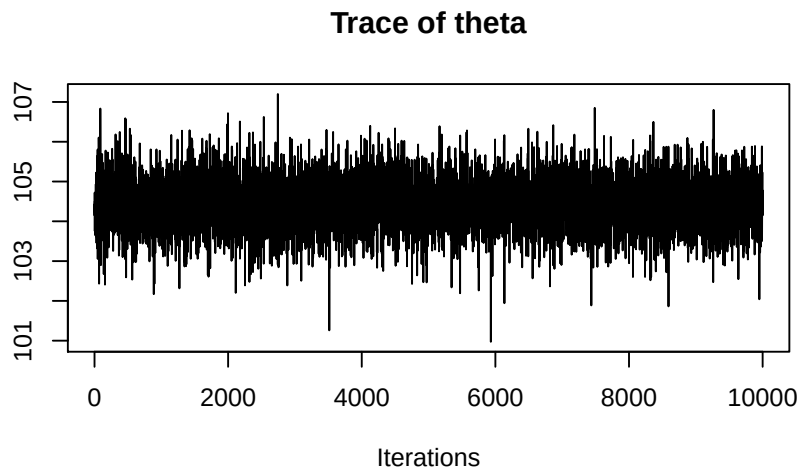
```
## , , theta
##
##            theta   sigma2
## Lag 0    1.00000 -0.05283
## Lag 1   -0.00154  0.01328
## Lag 5    0.00377  0.00721
## Lag 10   0.00825 -0.00519
## Lag 50  -0.01892 -0.01137
##
## , , sigma2
##
##            theta    sigma2
## Lag 0   -0.05283  1.00e+00
## Lag 1   -0.05606  1.11e-01
## Lag 5    0.01465  3.28e-05
## Lag 10  -0.02335 -8.76e-03
## Lag 50  -0.00646 -1.14e-02
```

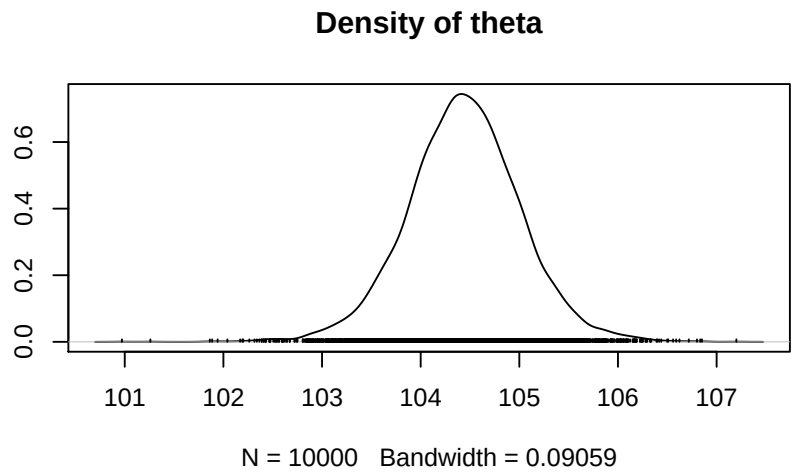
```
effectiveSize(THETA.coda)
```

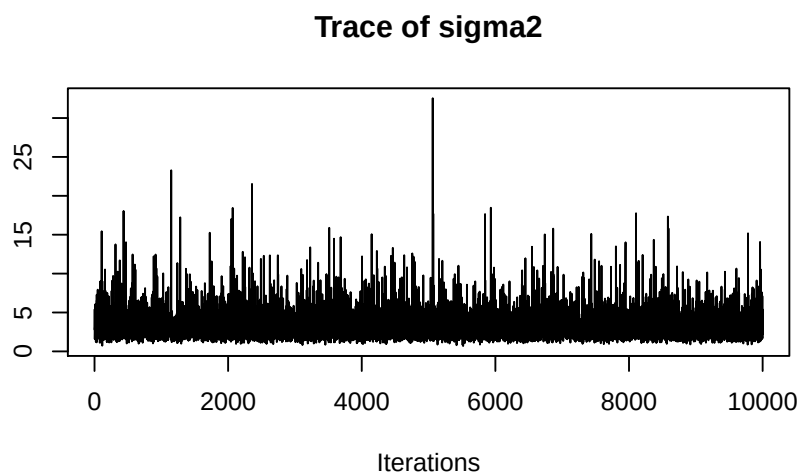
```
##  theta sigma2
##  10000   8372
```

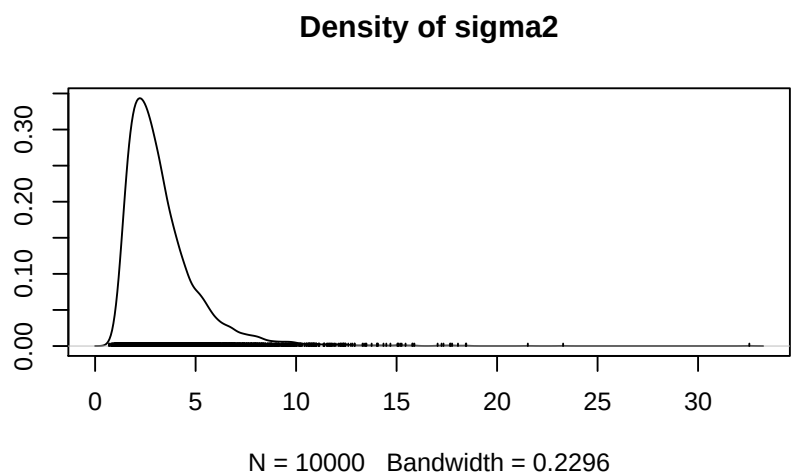
Trace plots

```
plot(THETA.coda)
```



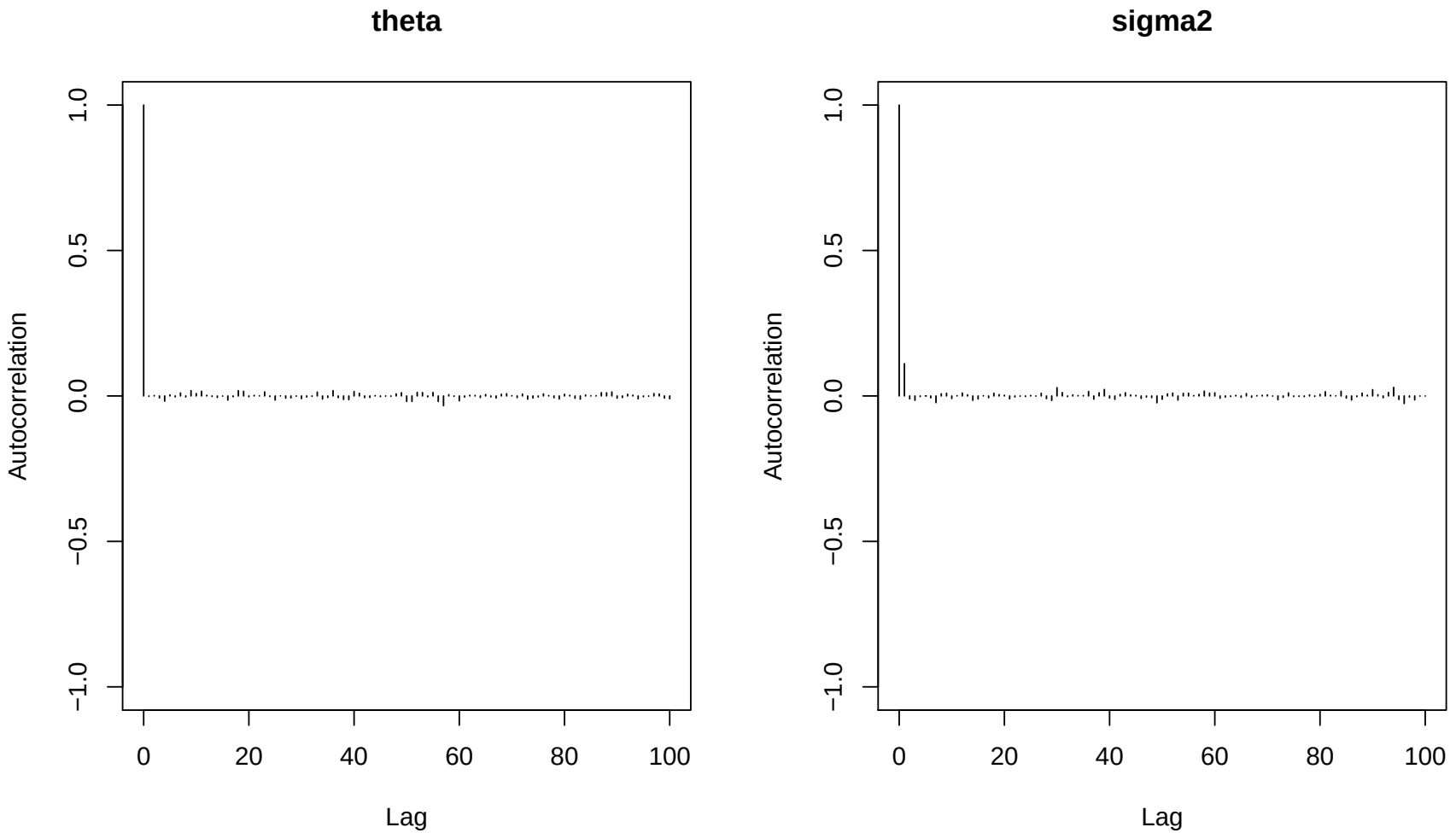






Autocorrelation plots

```
autocorr.plot(THETA.coda,lag.max=100)
```



Using multiple chains for MCMC diagnostics

- Using multiple chains to examine convergence and mixing is a common technique.
- If an MCMC chain has been run long enough after iteration S (i.e., converged and mixed well), then if we run M separate chains with *overdispersed* initial values, then after iteration S, they should all
  - have converged (i.e., moved to a high-probability region);
  - be able to cover all high-probability regions sufficiently deep.
- These imply that the distributions of draws from the M chains should all be similar.
- So we can construct various statistics to examine their differences.

Gelman-Rubin statistic

- Start off from running M separate chains with overdispersed starting points (with respect to the target p) for $2S$ iterations.
- Discard the first S draws in each as burn-in.
- Suppose the remaining S iterations are

$$
\begin{array}{c}
g_1^{(1)}, g_1^{(2)}, \cdots, g_1^{(S)} \\
g_2^{(1)}, g_2^{(2)}, \cdots, g_2^{(S)} \\
\vdots \\
g_M^{(1)}, g_M^{(2)}, \cdots, g_M^{(S)}
\end{array}
$$

- Let's use these $S \cdot M$ draws to estimate $\mathrm{Var}_p g$.

Main idea

- If convergence is achieved and mixing is fine, then an unbiased estimate for Var$\bar{g}$ should be similar whether or not we use draws from within each chain, and those from across chains.
- Otherwise, we will overestimate Var$_p g$ if we use draws across chains and underestimate if we only use draws from within each chain.
- Strategy:
  - Compute the ratio $\hat{R}$ of these two estimate (and take a square root).
  - Examine how close $\hat{R}$ is to 1.

The Gelman-Rubin statistics

- Specifically, the Gelman-Rubin statistic is

$$\hat{R} = \sqrt{\frac{\frac{S-1}{S}W + \frac{1}{S}B}{W}}$$

where

$$B = \frac{S}{M-1}\sum_{m=1}^{M}(\bar{g}_m - \bar{g}_\cdot)^2 \quad \text{and} \quad W = \frac{1}{M}\sum_{m=1}^{M} s_m(g)^2$$

where $\bar{g}_m = \frac{1}{S}\sum_t g_m^{(t)}$, $\bar{g}_\cdot = \frac{1}{M}\sum_{m=1}^{M}\bar{g}_m$, and $s_m(g)^2 = \frac{1}{S-1}\sum_{t=1}^{S}(g_m^{(t)} - \bar{g}_m)^2$.

- As $S \to \infty$, $\hat{R} \to 1$.
- Gelman (2004) recommends running the chains until $\hat{R} < 1.1$, though this is just a rule-of-thumb based on experience. There is not much theoretical support for that.

Using different portions of a single chain

- Another strategy for examining convergence and mixing is by comparing disjoint portions of a single chain.
  - Geweke (1992) proposes to compare the MCMC estimates based on two disjoint portions of a single chain, and quantify the difference using a z-score.
  - Raftery and Lewis diagnostic focuses on variability in the quantiles.
- Strategies using a single chain is less robust as it can never tell if a chain is simply stuck in a high-probability region!
- Using multiple chains based on overdispersed starting points has a chance of detecting that.
- Better yet: A variant of Gelman-Rubin combines the two strategies, by using M separate chains of length S and divide each into d bins, resulting in a total of dM chains of length S/d (often $d = 2$).

Bivariate example

```
M <- 10; S <- 1000;
mu=c(2,2)
rho <- 0.99
theta.mc <- matrix(NA,nrow=S,ncol=2)
theta.mcmc.multiple <- mcmc.list()
theta.init.multiple <- mvtnorm::rmvnorm(M,mean=mu, sigma=10*matrix(c(1,0,0,1),ncol=2))
for (m in 1:M) {
# Begin Gibbs sampling
  theta <- theta.init.multiple[m,] # initial value
  theta.prev <- theta

  for (t in 1:S) {
    theta[1] <- rnorm(1,mean=mu[1]+rho*(theta[2]-mu[2]),sd=sqrt(1-rho^2))
    theta[2] <- rnorm(1,mean=mu[2]+rho*(theta[1]-mu[1]),sd=sqrt(1-rho^2))
    theta.mc[t,] <- theta
    theta.prev <- theta
  }
  theta.coda = mcmc(theta.mc,start=1) # no burnin steps
  theta.mcmc.multiple[[m]] = theta.coda
}
gelman.diag(theta.mcmc.multiple)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.07       1.14
## [2,]       1.07       1.14
##
## Multivariate psrf
##
## 1.07
```

Bivariate example

```
gelman.plot(theta.mcmc.multiple)
```