UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Dipartimento di scienze economiche,
aziendali, matematiche e statistiche
“Bruno de Finetti”

Bayesian Statistics

Markov Chain Monte Carlo methods

Leonardo Egidi

A.A. 2022/23

A partial change of paradigm

Up to now we have typically generated *iid* variables directly from the density of interest π or indirectly in the case of importance sampling.

The Metropolis-Hastings algorithm and the Gibbs sampling generate instead *correlated* variables from a stochastic process called Markov chain. Markov chains carry different convergence properties that can be exploited to provide easier proposals in cases where generic importance sampling does not readily apply.

We will briefly review these stochastic process to fully understand how an MCMC algorithm works and to program it in a proper way.

The basic problem

Suppose we want to draw from our posterior distribution $\pi(\theta|y)$, but we cannot sample independent draws from it. For example, we often do not know the normalizing constant.

However, we may be able to sample draws from $\pi(\theta|y)$ that are slightly dependent. If we can sample slightly dependent draws using a Markov chain, then we can still find quantities of interests from those draws.

This process is called Monte Carlo Integration. Basically a fancy way of saying we can take quantities of interest of a distribution from simulated draws from the distribution.

Indice

Stochastic process

Def (Stochastic process)

Given a state space Ω and a sequence of consecutive time instants $\mathcal{T} = [0, T]$, a stochastic process $\{X^{(t)}\}_{t \in \mathcal{T}}$ is defined as a collection of random variables, i.e., each $X^{(t)} : \Omega \to E$ is a measurable function from the set of possible outcomes Ω to a measurable space E.

Informally, a stochastic process is then a consecutive set of random quantities defined on some known state space.

Notation In the Bayesian framework, we use the symbol θ to refer to the random parameter(s), and we will consider a draw of $\theta^{(s)}$ to be the state of the chain at iteration $s, s = 1, \ldots, S$, where S is the total number of desired simulations.

Markov chains

A Markov chain is a stochastic process in which future states are independent of past states given the present state.

Def (Markov chain)

A Markov chain $\{X^{(t)}\}$ *is a sequence of dependent random variables*

$$X^{(0)}, X^{(1)}, \ldots, X^{(t)}, \ldots$$

such that the probability distribution of $X^{(t)}$ *given the past variables depends only on the last value,* $X^{(t-1)}$*. This conditional probability distribution is called a transition kernel* K*, that is:*

$$X^{(t)}|X^{(0)}, X^{(1)}, \ldots, X^{(t-1)} \sim K(X^{(t-1)}, X^{(t)}).$$

In other words, $p(X^{(t)}|X^{(0)}, X^{(1)}, \ldots, X^{(t-1)}) \equiv p(X^{(t)}|X^{(t-1)})$.

Examples of Markov chains

Examples:

1. a simple *random walk* stochastic process satisfies:

$$X^{(t+1)} = X^{(t)} + \epsilon_t, \tag{1}$$

where $\epsilon_t \sim \mathcal{N}(0, 1)$, independently of $X^{(t)}$; therefore, the Markov kernel $K(X^{(t)}, X^{(t+1)})$ corresponds to a $\mathcal{N}(X^{(t)}, 1)$ density, depending on $X^{(t)}$ only.

2. a stochastic sequence of binary random variables $\{X^{(t)}\}$, where $X^{(t)} = 1$ with probability p_t, and p_t is generated from a $\text{Beta}(3 + X^{(t-1)}, 3 - X^{(t-1)})$.

Properties of a Markov chain

For the most part, the Markov chains encountered in Markov Chain Monte Carlo (MCMC) settings enjoy some fundamental properties.

1. Irreducibility The kernel K allows for free moves all over the state-space: no matter the starting value $X^{(0)}$, the sequence $\{X^{(t)}\}$ has a positive probability of eventually reaching any region of the state-space.
2. Positive Recurrency The chain will return to any arbitrary nonnegible set an infinite number of times, with a finite expected return time.
3. Aperiodicity The only length of time for which the chain repeats some cycle of values is the trivial case with cycle length equal to one.
4. Stationarity There exists a probability distribution f such that if $X^{(t)} \sim f$, then $X^{(t+1)} \sim f$. Therefore, formally the kernel and the stationary distribution satisfy the equation:

$$\int_{\mathcal{X}} K(x, y) f(x) dx = f(y)$$

Stationarity and limiting distribution

In Bayesian terms, our Markov chain is a bunch of draws $\{\theta^{(s)}\}$ of θ that are each slightly dependent on the previous one, and the posterior π is our target density f. The chain wanders around the parameter space, remembering only where it has been in the last period.

The fourth property, **stationarity**, is fundamental in MCMC setting. In case of recurrent chains, the stationary distribution is also a *limiting* distribution, in the sense that the limiting distribution of $\theta^{(s)}$ as $s \to \infty$ is π for almost any initial value $\theta^{(0)}$.

This fifth property is also called ergodicity, and it obviously has major consequences from a simulation point of view in that, if a given kernel K produces an ergodic Markov chain with stationary distribution π, generating a chain from this kernel K will eventually produce simulations from π.

Monte Carlo Integration on the Markov Chain

Once we have a Markov chain that has converged to the stationary distribution, then the draws in our chain appear to be like draws from $\pi(\theta|y)$.

So it seems like we should be able to use Monte Carlo Integration methods to find our quantities of interest. One problem: our draws are not independent, and this assumption is required for Monte Carlo Integration to work. Remember the Strong Law of Large Numbers: if $\theta^{(1)}, \ldots, \theta^{(S)}$ are *iid* with mean $\mu = E(\theta^{(s)})$, then with probability 1:

$$\frac{1}{S}\sum_{s=1}^{S}\theta^{(s)} \to \mu \text{ a.s., as } S \to \infty.$$

Luckily, we have the Ergodic Theorem for correlated samples.

The Ergodic theorem

The following theorem means suggests that the Strong Law of Large Numbers that lies at the basis of Monte Carlo methods can also be applied in MCMC settings.

Theorem (The Ergodic Theorem)

Given S values $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(S)}$ from a Markov chain that is aperiodic, irreducible, and positive recurrent (then the chain is called ergodic), and let $h : \Theta \to \mathbb{R}$ a real and integrable function such that $E(h(\theta)) < \infty$. Then with probability 1, as $S \to \infty$

$$\frac{1}{S}\sum_{s=1}^{S} h(\theta^{(s)}) \to \int_{\Theta} h(\theta) f(\theta) d\theta, \tag{2}$$

where the target distribution $f(\cdot)$ is the stationary distribution of the chain.

Comments to the Ergodic theorem

- This is the Markov chain analogue to the SLLN, and it allows us to ignore the dependence between draws of the Markov chain when we calculate quantities of interest from the draws. The stationary distribution is also a limiting distribution.
- The asymptotic variance of a MCMC estimator is approximately:

$$S^{-1}\sigma^2(1 + 2\sum_s \rho_s),$$

  where ρ_s is the s-th lag autocorrelation of the sequence (it behaves like a penalty).
- We will rely on MCMC algorithms—such as Metropolis-Hasting and Gibbs sampling—which are almost always theoretically convergent, meaning that the algorithm always converges to the stationary distribution f.

How Does a Markov Chain Work - Discrete example

For discrete state space with k possible states, the transition kernel K is defined as a $k \times k$ matrix of transition probabilities:

$$P = \begin{bmatrix} Pr(A|A) & Pr(B|A) & Pr(C|A) & \dots & Pr(k|A) \\ Pr(A|B) & Pr(B|B) & & \dots & Pr(k|B) \\ \dots & & & & \\ Pr(A|k) & Pr(B|k) & & \dots & Pr(k|k) \end{bmatrix}$$

where the first row expresses the probabilities of reaching any other state from state A in **one step**, and so on for the other rows. Thus, we may define $\Pr(\theta^{(s+1)} = A|\theta^{(s)} = B), \forall s$. Notice that the transition kernel does not depend on s. We say that the chain is homogeneous.

The **goal** is to generate $\theta^{(1)}, \dots, \theta^{(S)}$ from the stationary distribution underlying the chain.

How Does a Markov Chain Work - Discrete example

Suppose that $k = 6$ and that P is:

$$P = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

where

$$\Pr(\theta^{(s+1)} = B | \theta^{(s)} = A) = 0.5$$

$$\Pr(\theta^{(s+1)} = C | \theta^{(s)} = B) = 0.25$$

How Does a Markov Chain Work - Discrete example

(1) Define an initial vector of probabilities of length k, $\pi^{(0)}$.

(2) At iteration 1, the distribution $\pi^{(1)}$ - from which $\theta^{(1)}$ will be generated - is given by:

$$\pi^{(1)} = \pi^{(0)} P$$

(3) At iteration 2, the distribution $\pi^{(2)}$ - from which $\theta^{(2)}$ will be generated - is given by:

$$\pi^{(2)} = \pi^{(1)} P = \pi^{(0)} P^2$$

(4) ...

(s) At iteration s, the distribution $\pi^{(s)}$ from which $\theta^{(s)}$ will be generated is given by:

$$\pi^{(s)} = \pi^{(s-1)} P = \pi^{(0)} P^s$$

How Does a Markov Chain Work - Discrete example

We may find the **unique**, invariant distribution, i.e. a probability vector π such that

$$\pi = \pi P \tag{3}$$

In our example this vector is $\pi = (0.1, 0.2, 0.2, 0.2, 0.2, 0.1)$.

For all the MCMC algorithms we use in Bayesian statistics, the Markov chain will typically converge to π, regardless of our starting points.

So if we can devise a Markov chain whose stationary distribution π is our desired posterior distribution—$\pi(\theta|y)$—then we can run this chain to get draws that can be considered as approximately drawn from $\pi(\theta|y)$, once the chain has converged.

Thinning

Once we have a Markov chain that has converged to the stationary distribution, then the draws in our chain appear to be like draws from $\pi(\theta|y)$.

In order to break the dependence between draws in the Markov chain, some have suggested only keeping every d-th draw of the chain. This is known as thinning.

- **Pros**: Perhaps gets you a little closer to *iid* draws. Saves memory since you only store a fraction of the draws.
- **Cons**: Unnecessary with ergodic theorem. Shown to increase the variance of your Monte Carlo estimates.

Indice

MCMC methods

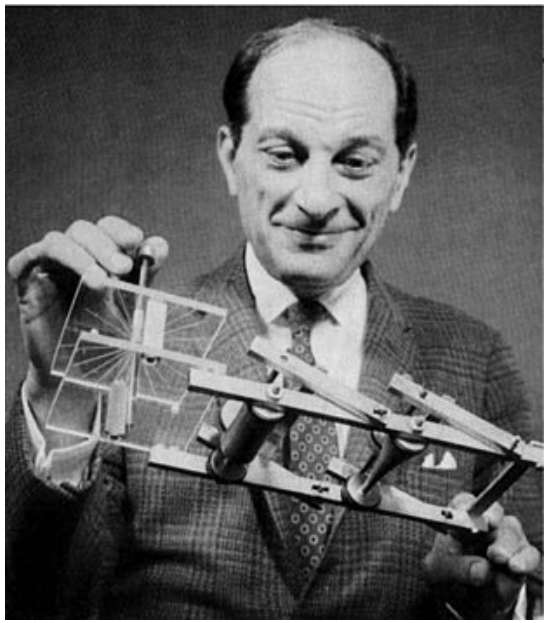
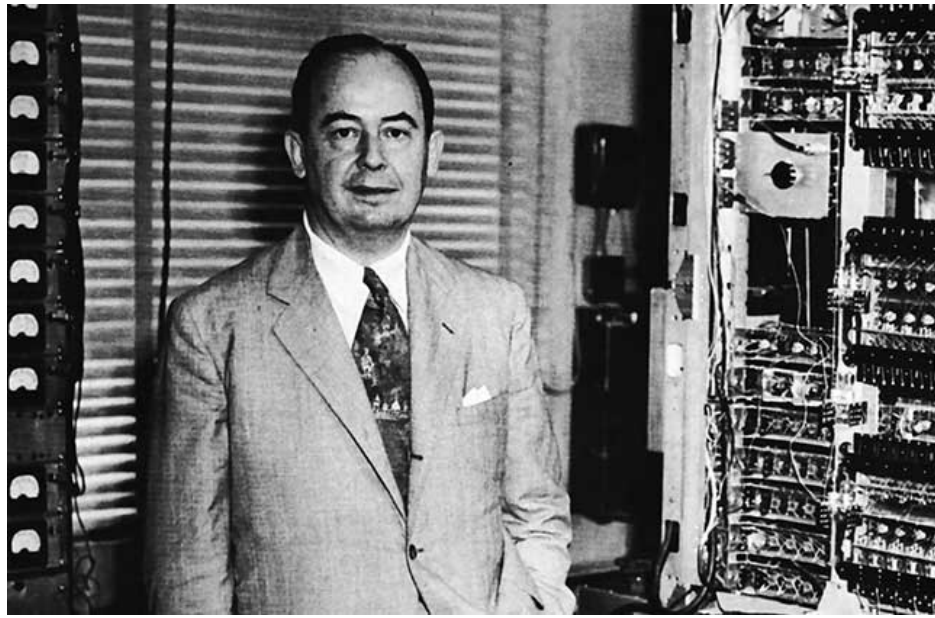
MCMC is a class of methods in which we can simulate draws that are slightly dependent and are approximately from a (posterior) distribution.

We then take those draws and calculate quantities of interest for the (posterior) distribution.
In Bayesian statistics, there are generally two MCMC algorithms that we use:

- Metropolis-Hastings algorithm
- Gibbs sampling

History

- **Monte Carlo**: used by Enrico Fermi, 1930s, while studying neutron diffusion.
- **MCMC** The modern version was invented in the late 1940s by Stanislaw Ulam, while he was working on nuclear weapons projects at the Los Alamos National Laboratory. John Von Neumann programmed the ENIAC computer to carry out Monte Carlo calculations.

Metropolis-Hastings algorithm

Suppose we have a posterior $\pi(\theta|y)$ that we want to sample from, but

- the posterior does not look like any distribution we know (no conjugacy...)
- the posterior consists of more than 2 parameters (grid approximations intractable)
- some (or all) of the full conditionals do not look like any distributions we know (thus, Gibbs sampling is unfeasible)

If all else fails, we can use the Metropolis-Hastings algorithm, which will always work, at least in theory.

Basic Metropolis Hastings

1. Choose a starting value $\theta^{(0)}$.
2. At iteration s, given $\theta^{(s-1)}$, draw a candidate θ^* from a proposal (instrumental) distribution $g(\theta^*|\theta^{(s-1)})$.
3. Compute the acceptance ratio:

$$R = \frac{\pi(\theta^*|y)g(\theta^{(s-1)}|\theta^*)}{\pi(\theta^{(s-1)}|y)g(\theta^*|\theta^{(s-1)})}. \tag{4}$$

4. Compute the accept probability as $\rho = \min\{R, 1\}$.
5. The value at iteration s is:

$$\theta^{(s)} = \begin{cases} \theta^* & \text{with probability } \rho \\ \theta^{(s-1)} & \text{wth probability } 1-\rho \end{cases}$$

6. Repeat steps 2-5 and get S samples $\theta^{(1)}, \ldots, \theta^{(S)}$.

Basic M-H: steps review

Step 1: *choosing a starting value*

- Under some easily satisfied regularity conditions on the proposal density $g(\theta^*|\theta^{(s-1)})$, the sequence $\theta^{(1)}, \ldots, \theta^{(S)}$ will converge to a random variable (limiting distribution) that is distributed according to the posterior distribution $\pi(\theta|y)$.
- The important thing to remember is that $\theta^{(0)}$ must have positive probability, that is $\pi(\theta^{(0)}|y) > 0$.

Basic M-H: steps review

Step 2: *draw θ^* from the proposal distribution*

- The proposal $g(\theta^*|\theta^{(s-1)})$ determines where we move to in the next iteration of the Markov chain (analogous to the transition kernel).
- The support of g must contain the support of the posterior.
- Different M-H algorithms are constructed depending on the choice of proposal density:
  - Independent M-H: if the proposal density g is independent of the current value:
$$g(\theta^*|\theta^{(s-1)}) = g(\theta^*).$$
  - Random Walk M-H: if the proposal density g is symmetric around zero, that is satisfying $g(\theta^*|\theta^{(s-1)}) \equiv h(\theta^* - \theta^{(s-1)})$. It yields that $g(\theta^{(s-1)}|\theta^*) \equiv h(\theta^{(s-1)} - \theta^*)$ and the acceptance ratio (4) has the simple form:
$$R = \frac{\pi(\theta^*|y)}{\pi(\theta^{(s-1)}|y)}.$$

Basic M-H: steps review

Step 2: *draw θ^* from the proposal distribution*

- In the Independent M-H all our candidate draws θ^* are drawn from the same distribution, regardless of where the previous draw was. This can be extremely efficient or extremely inefficient, depending on how close the proposal distribution is to the posterior. In general, chain will behave well only if the proposal distribution has heavier tails than the posterior.
- In the Random-Walk M-H the acceptance probability does not depend on g. This means that, for a given pair $(\theta^{(s-1)}, \theta^*)$, the probability of acceptance is the same whether θ^* is generated from a Normal or a Cauchy distribution.

Basic M-H: steps review

Step 3: *compute acceptance probability* ρ

- The acceptance probability is:

$$\rho = \min\left\{1, \frac{\pi(\theta^*|y)g(\theta^{(s-1)}|\theta^*)}{\pi(\theta^{(s-1)}|y)g(\theta^*|\theta^{(s-1)})}\right\}.$$

  In the case where proposal distribution is symmetric, we have that $g(\theta^*|\theta^{(s-1)}) = g(\theta^{(s-1)}|\theta^*)$, thus:

$$\rho = \min\left\{1, \frac{\pi(\theta^*|y)}{\pi(\theta^{(s-1)}|y)}\right\}.$$

- If our candidate draw has higher probability than our current draw, then our candidate is better $\Rightarrow$ so we definitely accept it. Otherwise, our candidate is accepted according to the probability ratio (4).
- Since ρ is a ratio, we only need $\pi(\theta|y)$ up to a constant of proportionality, since $p(y)$ cancels out in both the numerator and denominator.

Basic M-H: steps review

Step 3: *compute acceptance probability* ρ

- In cases where our proposal distribution is not symmetric, we need to weight our evaluations of the draws at the posterior densities by how likely we are to draw each draw.
- For example, if we are very likely to jump to some θ^*, then $g(\theta^*|\theta^{(s-1)})$ is likely to be high, so we should accept less of them than some other θ^* that we are less likely to jump to.

Basic M-H: steps review

Step 4: *decide whether to accept* θ^*

- Accept θ^* as your $\theta^{(s)}$ with probability ρ. If θ^* is not accepted, then $\theta^{(s)} = \theta^{(s-1)}$. Practically, how we do this?
  1. For each θ^*, draw a value $U \sim \text{Unif}(0, 1)$.
  2. If $U \leq \rho$, accept θ^*; otherwise, use $\theta^{(s-1)} = \theta^{(s)}$. Candidate draws with higher density than the current draw are always accepted.
  3. Unlike in rejection sampling (A-R method), each iteration always produces a draw, either $\theta^{(s-1)}$ or θ^*.
- Now, we have to analyse the acceptance rate of our M-H algorithm. It is very important to monitor the acceptance rate, that is the fraction of candidate draws that are accepted, of our M-H algorithm.

M-H acceptance rate

- If your acceptance rate is too high, the chain is probably not mixing well (not moving around the parameter space quickly enough).
- If your acceptance rate is too low, your algorithm is too inefficient (rejecting too many candidate draws).
- There is a **trade-off**: we would like *large* jumps (updates), so that the chain explores the state space, but large jumps usually have low acceptance probability as the posterior density can be highly peaked (and you jump off the mountain side)
- What is too high and too low depends on your specific algorithm, but generally:
  - random walk: somewhere between 0.25 and 0.50 is recommended
  - independent chain: something close to 1 is preferred

A simple example: random-walk M-H for a Gamma(4.4, 1.7)

Gaussian proposal distribution with standard deviation 2.

```
# Define a function for the entire M-H with fixed arguments
    # n.sims: number of simulations (samples)
    # start = starting value for theta_0
    # burnin = throw out a certain number of the first draws
    # cand.sd = sd of the normal proposal
    # shape, rate = gamma arguments

mh.gamma <-
  function(n.sims=10^5, start=1, burnin=0.3*n.sims, cand.sd=2,
           shape=1, rate=1) {

    # (1) initialization for theta_0 and definition of the draws

    theta.cur <- start
    draws <- c()
```

Ex 1: random-walk M-H for a Gamma(4.4, 1.7)

R code continuation:

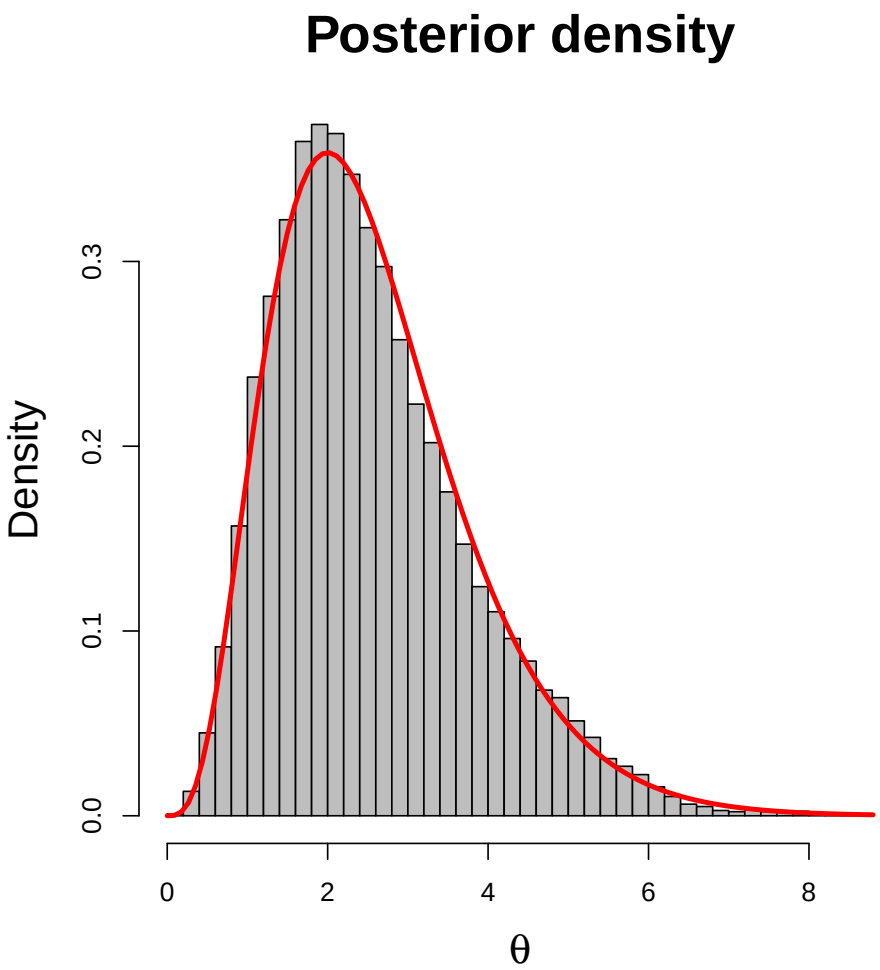
```
# (2)  sampling from a normal with mean
#      equal to the last value and sd = sd

theta.update <- function(theta.cur, shape, rate) {
  theta.can <- rnorm(1, mean = theta.cur, sd = cand.sd)

#  (3)  accept probability
  accept.prob <- dgamma(theta.can, shape = shape, rate = rate)/
    dgamma(theta.cur, shape = shape, rate = rate)

# (4)  accept step
  if (runif(1) <= accept.prob)
    theta.can
  else theta.cur
}
```

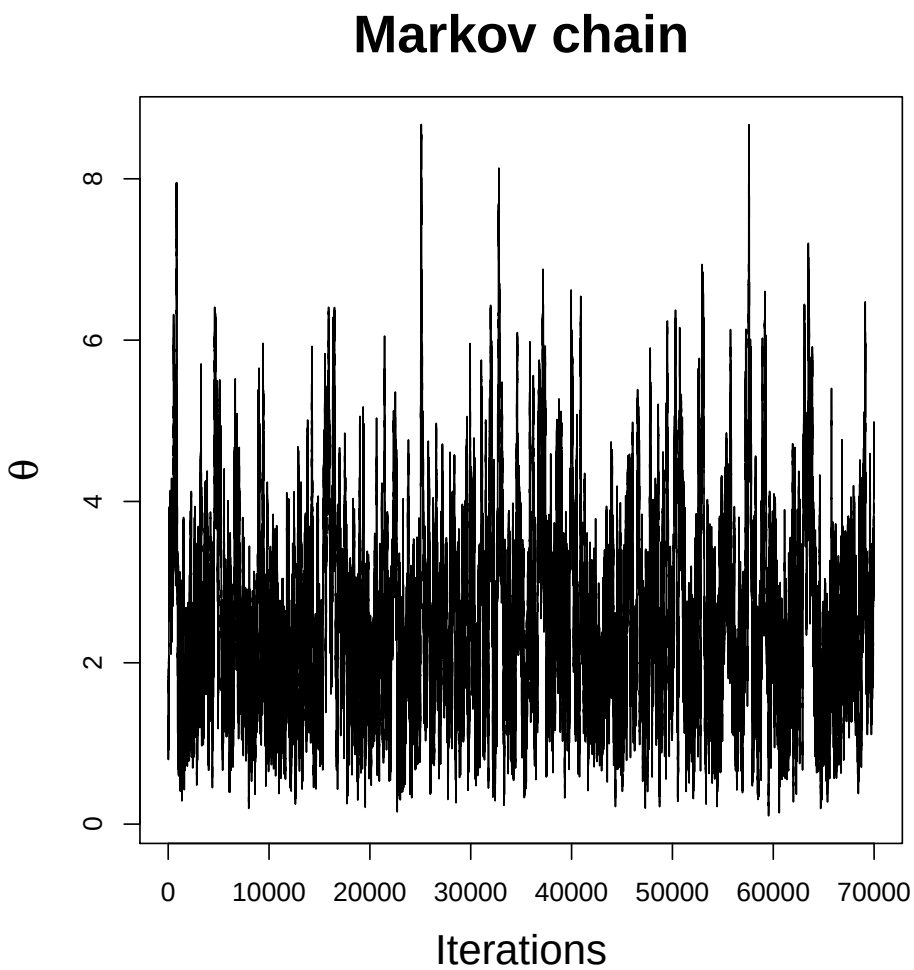
Ex 1: random-walk M-H for a Gamma(4.4, 1.7)

R code continuation:

```
# store the final draws
for (i in 1:n.sims) {
    draws[i] <- theta.cur <- theta.update(theta.cur,
                          shape = shape, rate = rate)}


# plots
samp<-draws[(burnin+1):n.sims]
par(mfrow=c(1,2))
hist(samp, nclass=50, prob=T,
      main="Posterior density",
      col="grey", xlab=expression(theta), cex.lab=2, cex.main=2)
curve(dgamma(x, shape=shape, rate=rate), add=T, lwd =3,
      col ="red")
plot(samp, type="l", ylab=expression(theta), xlab="Iterations",
  cex.lab=2, cex.main=2, main="Markov chain")
return(samp)}
```

Ex 1: random-walk M-H for a Gamma(4.4, 1.7)





Ex 2: independent M-H for a Beta(2.7, 6.3)

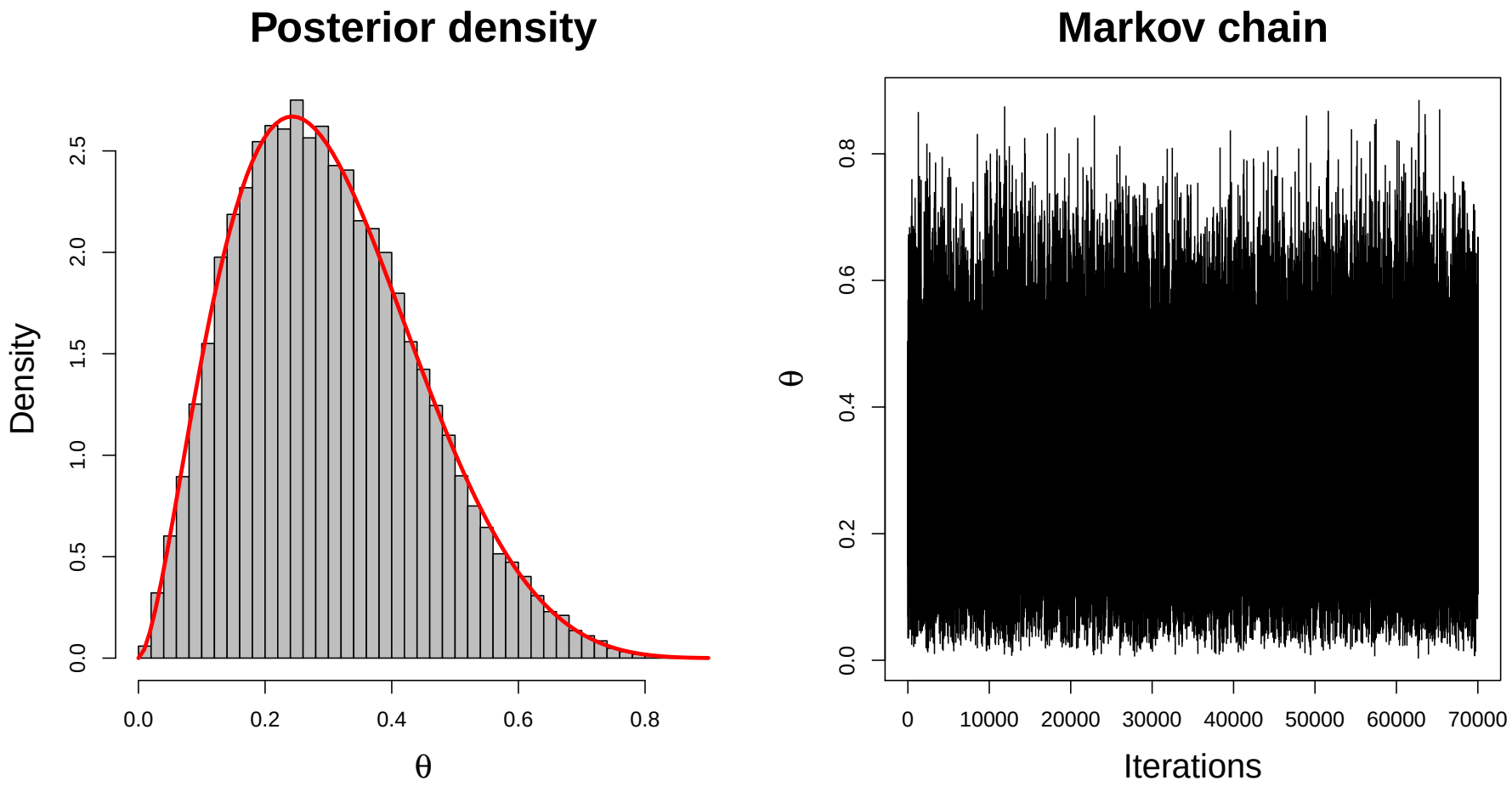


Figure: $g(\theta) = \text{Unif}(0, 1)$. Acceptance rate $= 0.503$

Ex 2: independent M-H for a Beta(2.7, 6.3)

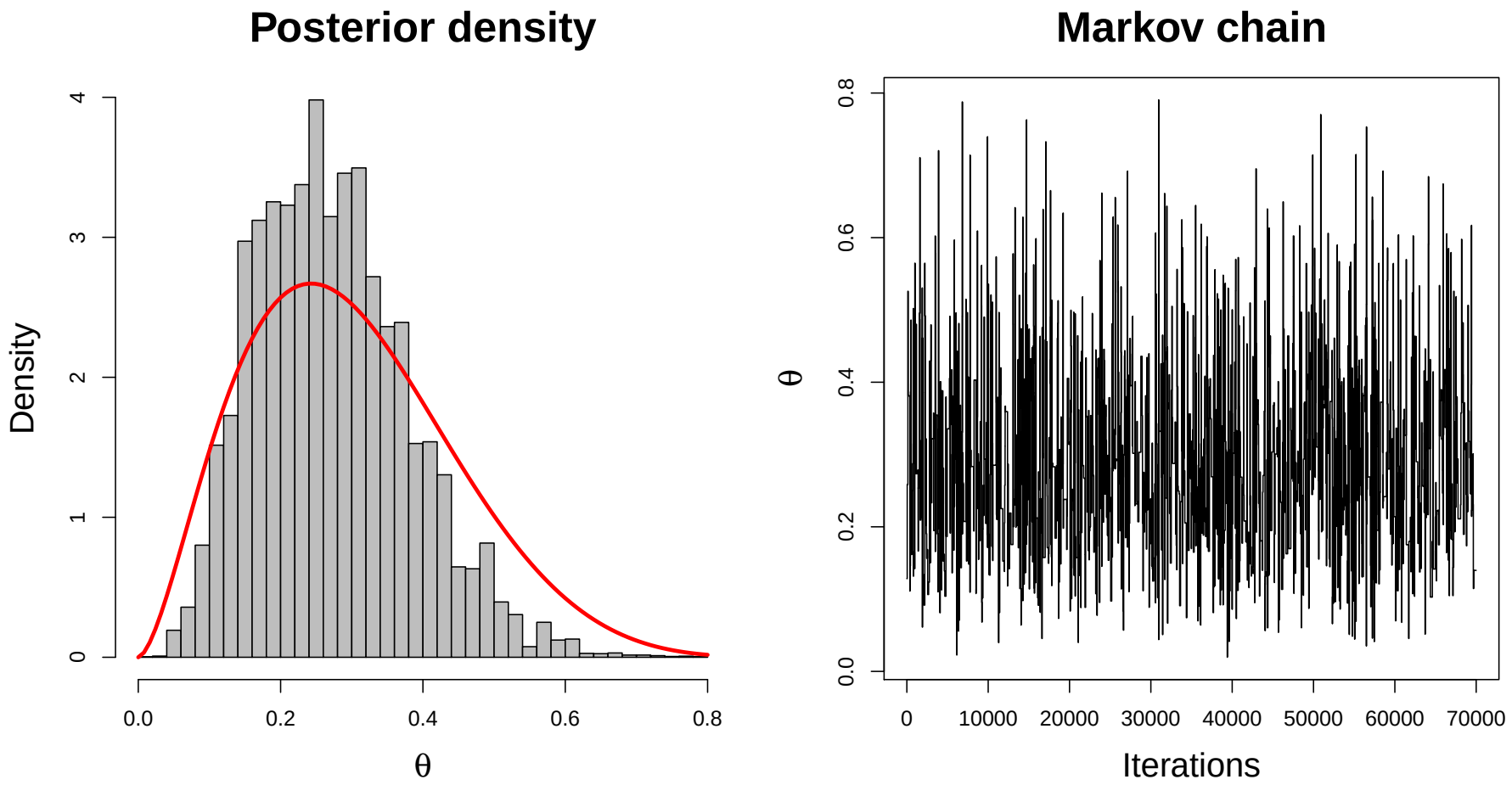


Figure: $g(\theta) = \text{Unif}(0, 3)$. Acceptance rate $= 0.12$

Metropolis-Hastings for Shuttle data

Challenger data

In 1986, the space shuttle Challenger exploded during takeoff, killing the seven astronauts aboard. The explosion was the result of an *O-ring* failure, a splitting of a ring of rubber that seals the parts of the ship together. The accident was believed to have been caused by the unusually cold weather (31 Fahrenheit or 0 Celsius) at the time of launch, as there is reason to believe that the O-ring failure probabilities increase as temperature decreases. Data on previous space shuttle launches and O-ring failures is given in the dataset `challenger` provided with the `mcsm` package.

Metropolis-Hastings for Shuttle data

Suppose to have the following data coming from the previous shuttle flights:

```
x <- c(66, 70, 69, 68, 67, 72, 73, 70, 57, 63, 70, 78, 67,
       53, 67, 75, 70, 81, 76, 79, 75, 76, 58)
Y <- c(0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1)
```

where we registered the number of failures Y for each of the $n = 23$ flights, at a given temperature (Fahrenheit) x.

Metropolis-Hastings for Shuttle data

We suppose a simple binomial model for Y:

$$\Pr(Y_i = y) = \binom{7}{y} \pi_i^y (1 - \pi_i)^{7-y},$$

where π_i is the failure probability for the i-th flight:

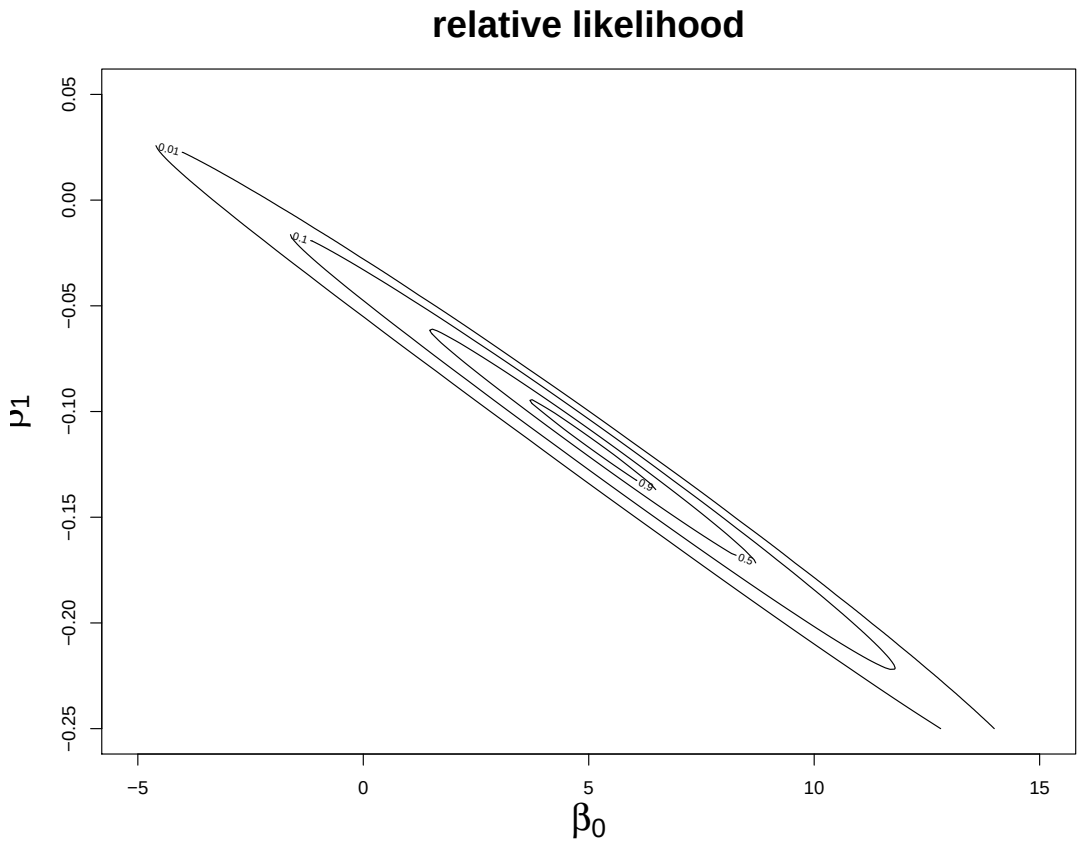
$$\pi_i = \frac{\exp\{\beta_0 + \beta_1 x_i\}}{1 + \exp\{\beta_0 + \beta_1 x_i\}}$$

and x_i is the temperature value. The log-likelihood is then:

$$l(\beta_0, \beta_1) = \sum_{i=1}^{n} y_i(\beta_0 + \beta_1 x_i) - \sum_{i=1}^{n} 7 \log(1 + \exp\{\beta_0 + \beta_1 x_i\}).$$

Metropolis-Hastings for Shuttle data

```
loglik <- function(beta) {
  sum(y * (beta[1] + beta[2] * x)) -
    sum(7 * log(1 + exp(beta[1] + beta[2]*x)))}
```



Metropolis-Hastings for Shuttle data

We elicit a uniform prior distribution for β_0 and β_1, $\pi(\beta_0, \beta_1) \propto 1$. The posterior is then proportional to:

$$\pi(\beta_0, \beta_1|y) \propto p(y|\beta_0, \beta_1)\pi(\beta_0, \beta_1) = \frac{\exp\{\sum_{i=1}^n y_i(\beta_0 + \beta_1 x_i)\}}{\prod_{i=1}^n [1 + \exp\{\beta_0 + \beta_1 x_i\}]}.$$

The posterior above is not known...We need MCMC simulation. So, we set up a random-walk Metropolis with the following proposal distributions at iteration s:

$$\beta_0^* \sim \mathcal{N}(\beta_0^{(s-1)}, s_1)$$
$$\beta_1^* \sim \mathcal{N}(\beta_1^{(s-1)}, s_2)$$

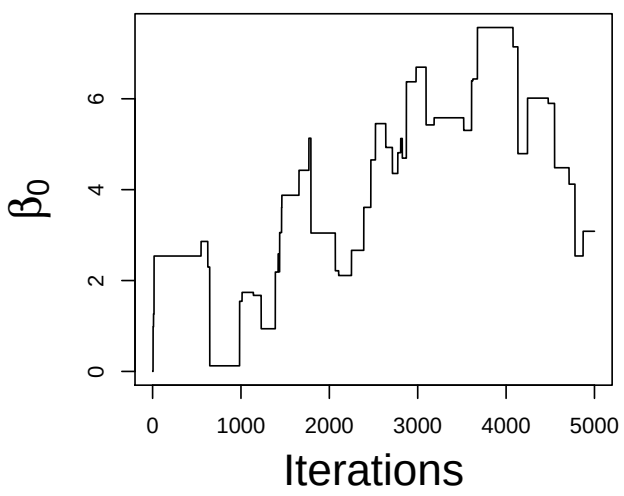
where s_1, s_2 are the instrumental variances.

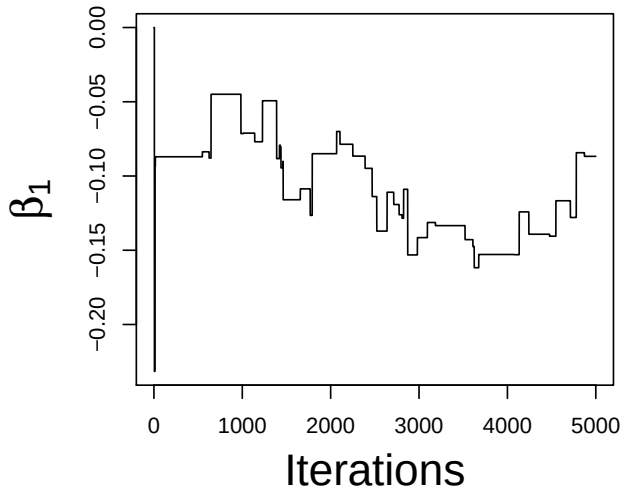
Metropolis-Hastings for Shuttle data

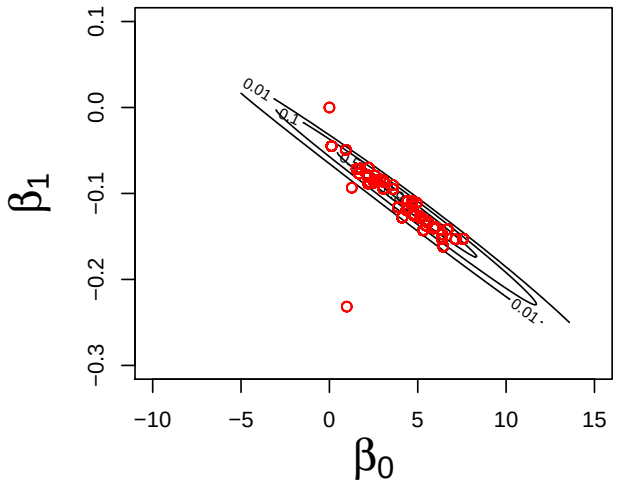
```
mh <- function(nsim, s1, s2, b.init) {
  mh.out <- matrix(ncol = 2, nrow = nsim)
   b <- b.init
   for (i in 1:nsim) {
     b.p <- c(rnorm(1, b[1], s1), rnorm(1,b[2], s2))
    if (runif(1) < exp(loglik(b.p) - loglik(b)))
      b <- b.p
      mh.out[i, ] <- b
       }
  mh.out}
```

Metropolis-Hastings for Shuttle data

Scenario 1: $s_1 = s_2 = 1$. Poor mixing!

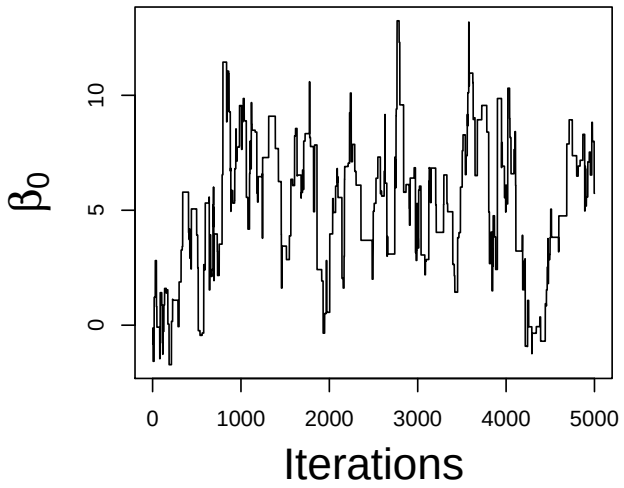


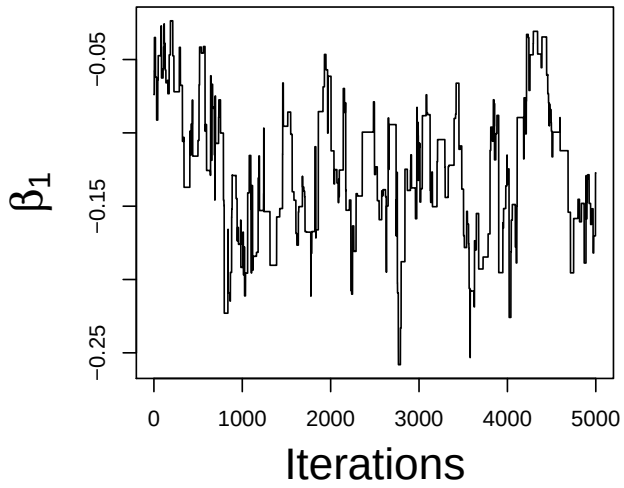


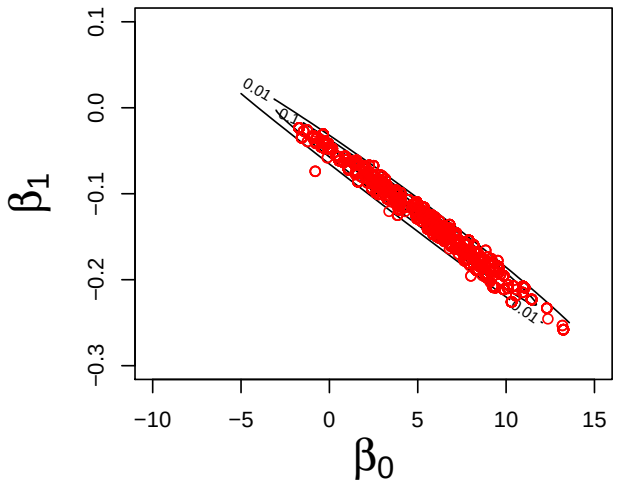


Metropolis-Hastings for Shuttle data

Scenario 2: $s_1 = 2$, $s_2 = 0.1$. Better, still not adequate...

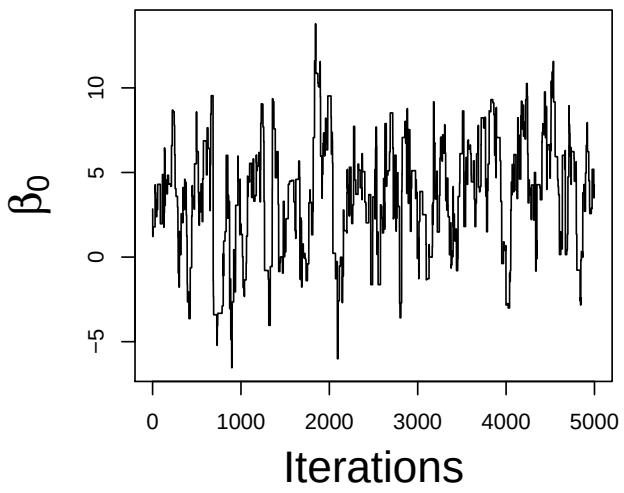


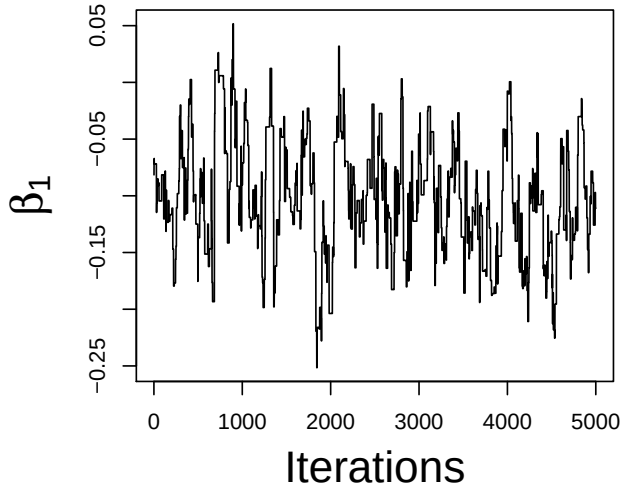


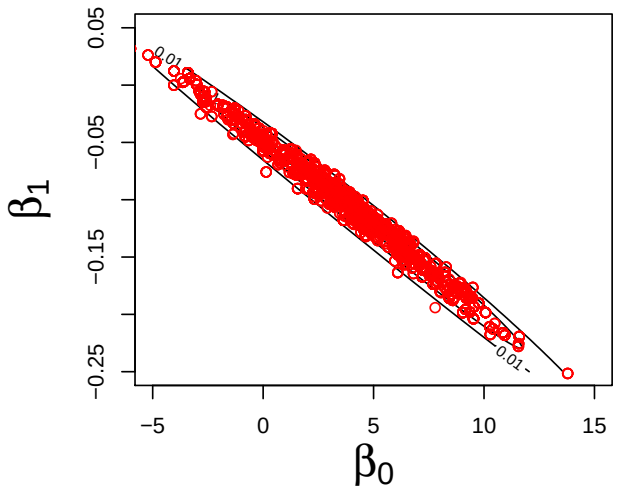


Metropolis-Hastings for Shuttle data

Scenario 3: s_1, s_2 estimated from Scenario 2. Better, still not good...







Metropolis-Hastings for Shuttle data

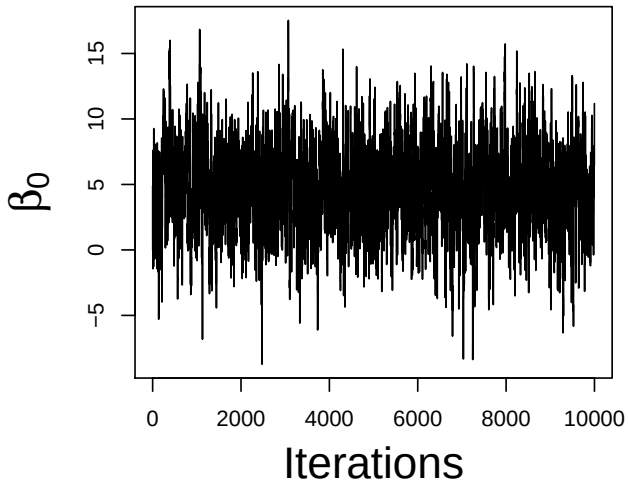
Consider now a correlated bivariate normal for the proposal distribution:

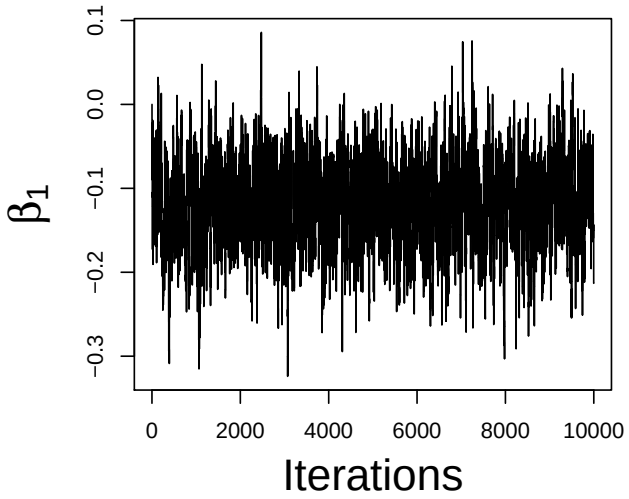
$$\begin{pmatrix} \beta_0^* \\ \beta_1^* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \beta_0^{(s-1)} \\ \beta_1^{(s-1)} \end{pmatrix}, \begin{pmatrix} \sigma_0^2 & \text{Cov}(\beta_0, \beta_1) \\ \text{Cov}(\beta_1, \beta_0) & \sigma_1^2 \end{pmatrix} \right).$$

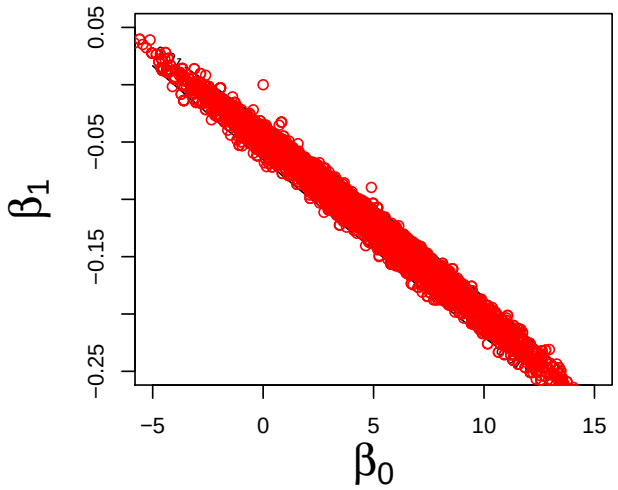
```
library(mvtnorm)
 mhd <- function(nsim, V, b.init) {
   mh.out <- matrix(ncol = 2, nrow = nsim)
   b <- b.init
   for (i in 1:nsim) {
     b.p <- rmvnorm(n = 1, mean = b, sigma = V)
     if (runif(1) < exp(loglik(b.p) - loglik(b)))
       b <- b.p
       mh.out[i, ] <- b
       }
   mh.out}
```

Metropolis-Hastings for Shuttle data

Scenario 4: cov.matrix V estimated from Scenario 3. Better!



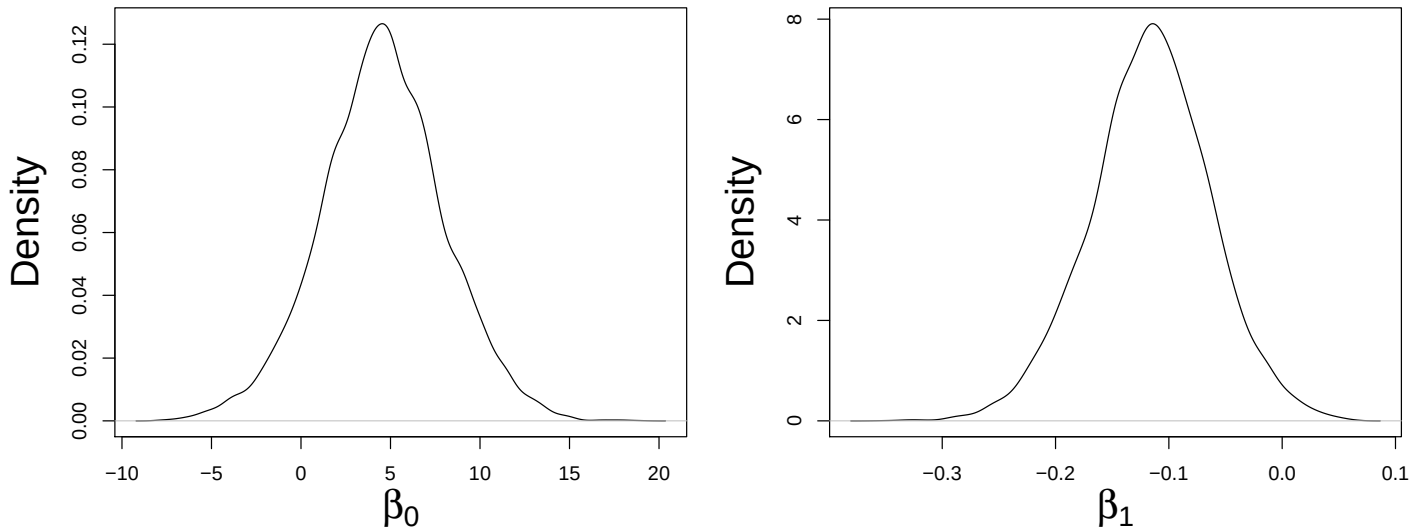




Metropolis-Hastings for Shuttle data

- When there is a not negligible parameters' correlation, a suited proposal should capture this fact.
- We can use the last simulation to report posterior summaries for β_0 and β_1.

```
apply(mh.out4,2, mean)
[1] 4.5555695 -0.1163703
```



Indice

Gibbs sampling

Suppose that the parameter of interest is $\theta = (\theta_1, \ldots, \theta_k)$ and we want to sample from its posterior distribution, $\pi(\theta_1, \theta_2, \ldots, \theta_k | y)$.

We can use the Gibbs sampler to sample from the joint distribution if we knew the full conditional distributions for each parameter.

For each parameter, the full conditional distribution is the distribution of each single component of the parameter vector, conditional on the known information and all the other parameters:

$$\pi(\theta_j | \theta_{\sim[j]}, y),$$

with $\theta_{\sim[j]} = (\theta_1, \theta_2, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_k)$.

Question: How can we know the joint distribution simply by knowing the full conditional distributions?

The Hammersley-Clifford Theorem (for two blocks only)

Theorem (Hammersley-Clifford)

Suppose we have a joint density $f(x, y)$. We can write out the joint density in terms of the conditional densities $f(x|y)$ and $f(y|x)$, that is:

$$f(x, y) = \frac{f(y|x)}{\int \frac{f(y|x)}{f(x|y)} dy}.$$

Proof.

$$f(x, y) = f(y|x)f(x) = f(y|x) \int \frac{f(x)}{f(y)} dy = f(y|x) \int \frac{\frac{f(x,y)}{f(y)}}{\frac{f(x,y)}{f(x)}} dy = f(y|x) \int \frac{f(x|y)}{f(y|x)} dy = \frac{f(y|x)}{\int \frac{f(y|x)}{f(x|y)} dy}$$

□

The Hammersley-Clifford Theorem (for two blocks only)

The theorem shows that knowledge of the conditional distribution is sufficient to get the joint density. This approach works for more than two blocks of parameters also.

Studying the full conditionals rather than the full posterior may be easier in many cases and able to break a big problem in smaller sub-problems.

But how do we figure out the full conditionals?

Steps for Calculating Full Conditional Distributions

Suppose we have a posterior distribution $\pi(\theta|y)$. To calculate the full conditionals for each component of θ, do the following:

1. Write out the full posterior ignoring constants of proportionality.
2. Pick a block of parameters (for example, θ_1) and drop everything that does not depend on θ_1.
3. Use your knowledge about distributions to figure out what the normalizing constant is (and thus what is the full conditional distribution $\pi(\theta_1|\theta_{\sim[1]}, y)$.
4. Repeat steps 2 and 3 for all parameter blocks.

Gibbs sampling

Suppose we wish to to sample from $\pi(\theta|y)$, where the parameter vector is $\theta = (\theta_1, \ldots, \theta_k)$.

1. Pick a vector of starting values $\theta^{(0)}$
2. At step s, given $\theta^{(s-1)} = (\theta_1^{(s-1)}, \ldots, \theta_k^{(s-1)})$ we can draw:
   - $\theta_1^{(s)} \sim \pi_1(\theta_1|\theta_2^{(s-1)}, \ldots, \theta_k^{(s-1)}, y)$
   - $\theta_2^{(s)} \sim \pi_2(\theta_2|\theta_1^{(s)}, \theta_3^{(s-1)}, \ldots, \theta_k^{(s-1)}, y)$
   - $\ldots$
   - $\theta_k^{(s)} \sim \pi_k(\theta_k|\theta_1^{(s)}, \theta_2^{(s)}, \ldots, \theta_{k-1}^{(s)}, y)$
3. Repeat step 2 until until we get S draws, with each draw being a vector $\theta^{(s)}$, $s = 1, \ldots, S$.

$\pi_1, \ldots, \pi_k$ are the *full conditional distributions* for the components $\theta_1, \ldots, \theta_k$, respectively. The final result is a Markov chain with a bunch of draws θ that can be considered approximately drawn from our posterior.

Example: bivariate normal

As a toy example, suppose we want to sample from:

$$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right)$$

The full conditionals for the two vector components are univariate normals, respectively:

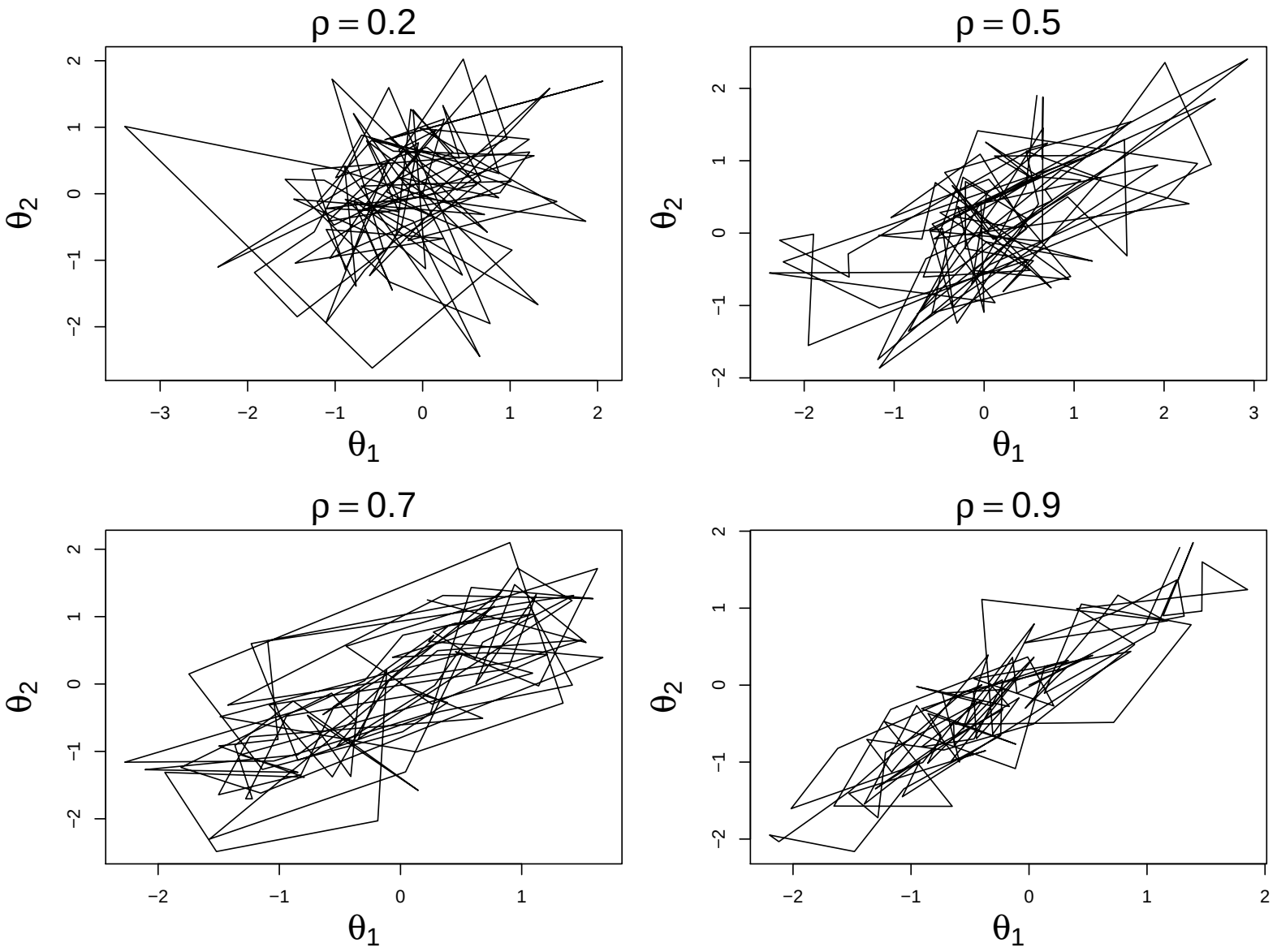
$$\theta_1 | \theta_2 \sim \mathcal{N}(\mu_1 + \rho(\theta_2 - \mu_2), 1 - \rho^2)$$
$$\theta_2 | \theta_1 \sim \mathcal{N}(\mu_2 + \rho(\theta_1 - \mu_1), 1 - \rho^2)$$

We can generate several independent chains, starting from different corners of the two-dimensional parameter space.

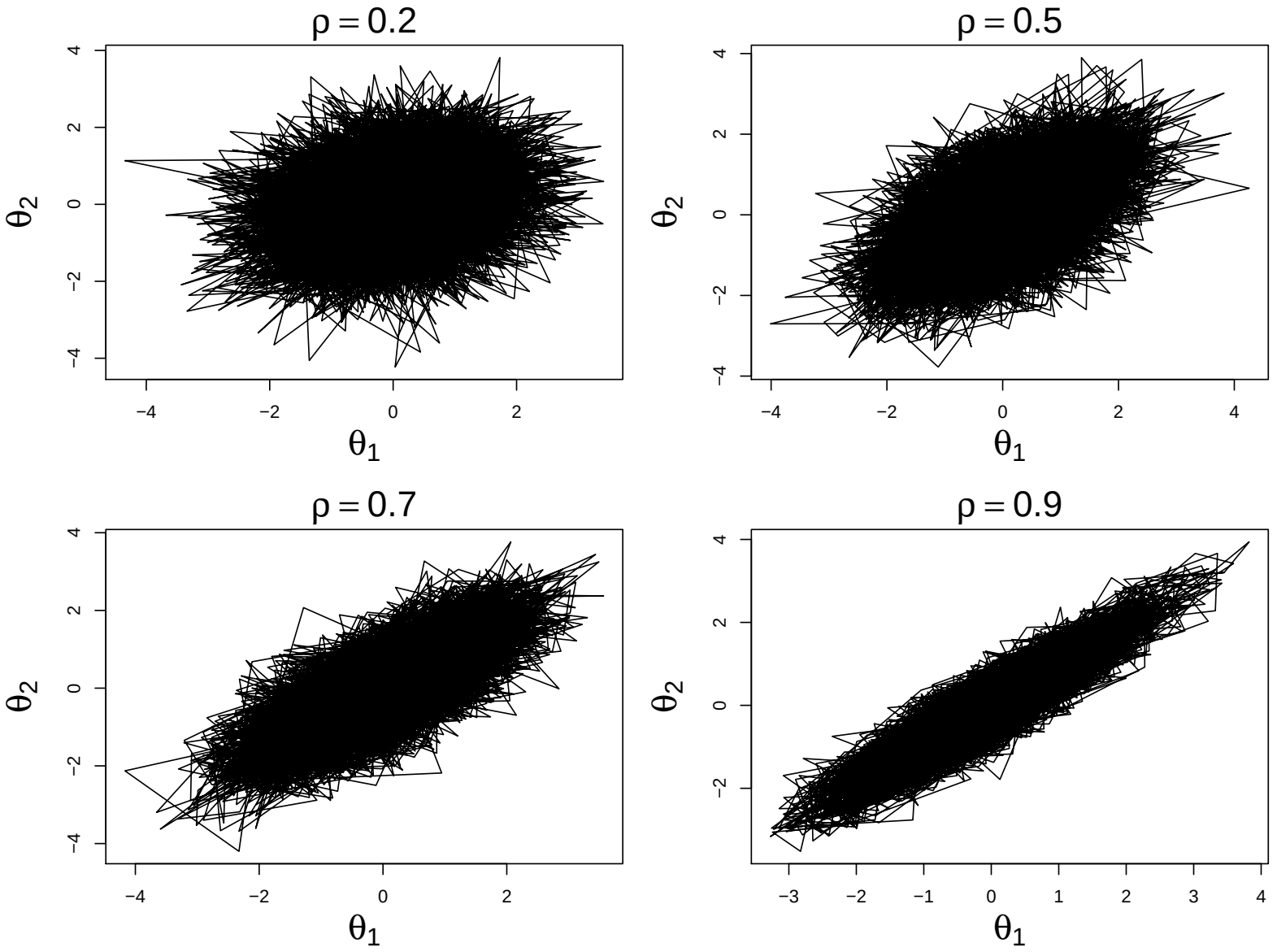
Example: bivariate normal

```
bn<-function(init=c(0,0),mu=c(0,0),
              rho=0, nsim=100){
  theta=matrix(nrow=nsim, ncol=2)
  theta[1,]=init
  for (i in 2:nsim){
    theta[i,1]<-rnorm(1,mean= mu[1]+ rho*(theta[i-1,2]-mu[2]),
     sd=sqrt(1-rho^2))
    theta[i,2]<-rnorm(1,mean= mu[2]+ rho*(theta[i,1]- mu[1]),
     sd=sqrt(1-rho^2))
  }
  plot(theta, type="l", xlab=expression(theta[1]),
       ylab=expression(theta[2]), cex.lab =2, cex.main=2,
       main= bquote(rho==.(rho)))
 }
```

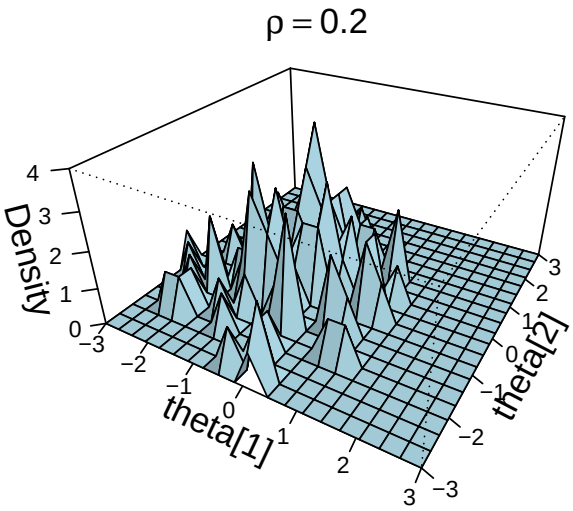
Example: bivariate normal (100 draws)

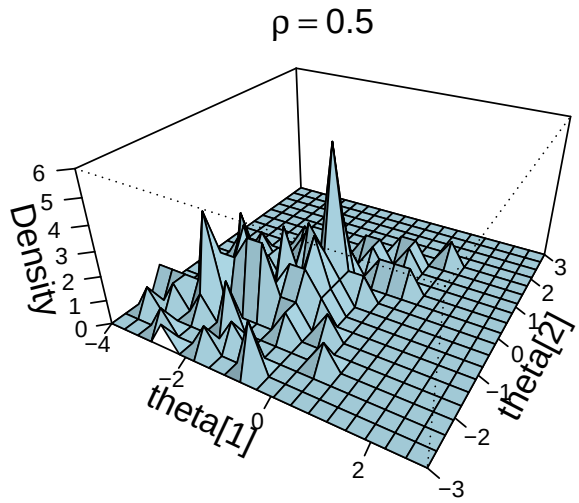


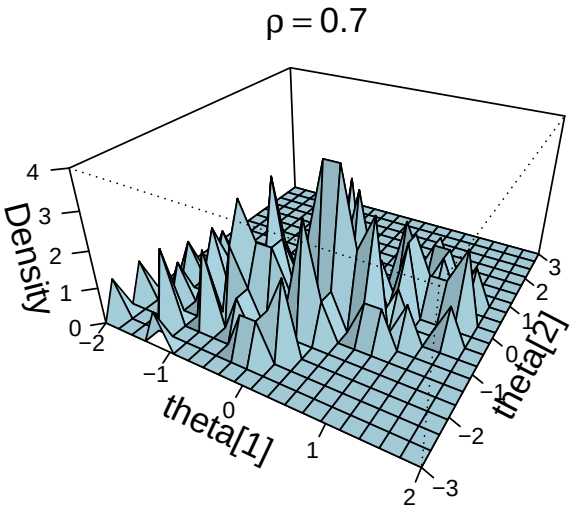
Example: bivariate normal (10000 draws)

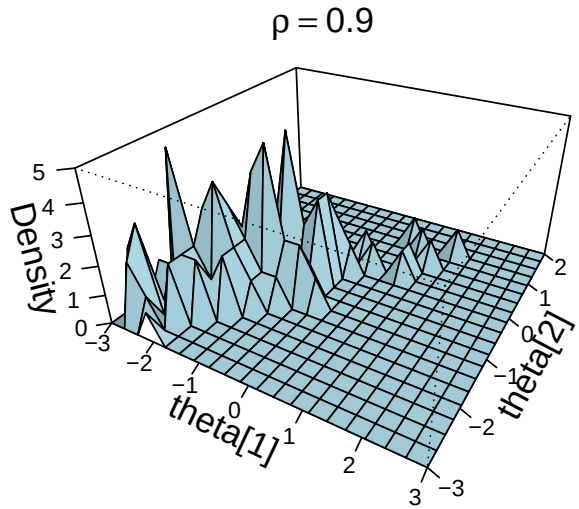


Example: bivariate normal (100 draws)

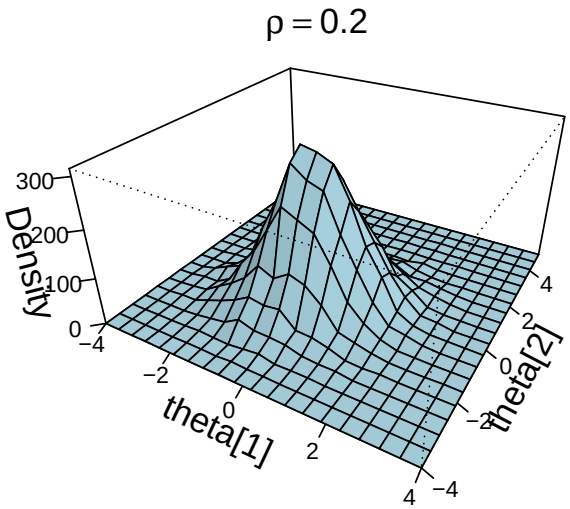


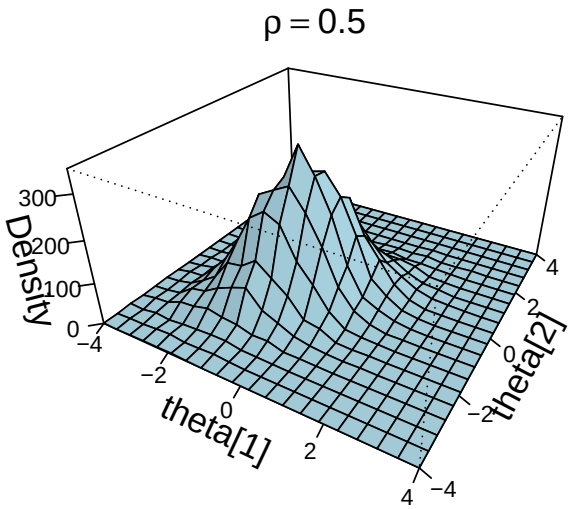


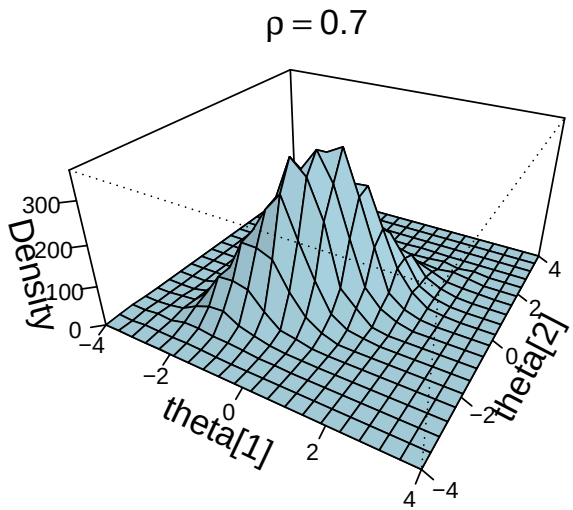


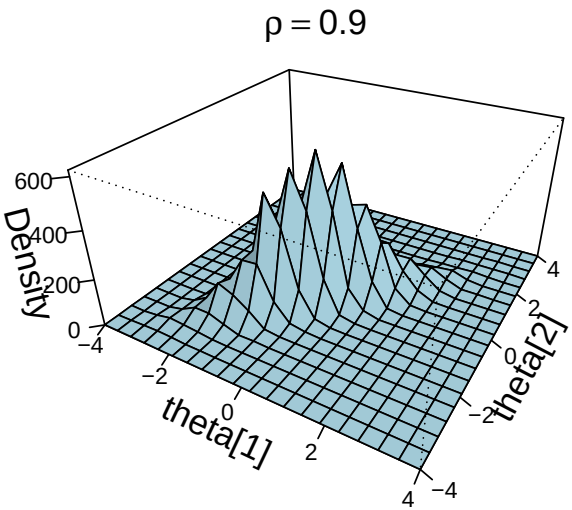


Example: bivariate normal (10000 draws)









Example: nuclear pumps

Suppose we have data on the number of failures (Y_i) for each of 10 pumps in a nuclear plant. We also have the total times (t_i) of observation for each pump.

```
 y <- c(5, 1, 5, 14, 3, 19, 1, 1, 4, 22)
 t <- c(94, 16, 63, 126, 5, 31, 1, 1, 2, 10)
 rbind(y, t)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
y    5    1    5   14    3   19    1    1    4    22
t   94   16   63  126    5   31    1    1    2    10
```

We want to model the number of failures with a Poisson model, where the expected number of failure λ_i's differs for each pump. Since the time for which we observed each pump is different, we need to scale each λ_i by its observed time t_i (offset!).

Example: nuclear pumps

Thus, we assume $Y_i \sim \text{Pois}(\lambda_i t_i)$, which means that our likelihood is

$$p(y|\theta) \propto \prod_{i=1}^{n} \text{Pois}(\lambda_i t_i)$$

Let us put some Gamma priors on $\lambda_1, \ldots, \lambda_{10}$ and β, respectively:

$$\lambda_1, \ldots, \lambda_{10} \sim \text{Gamma}(1.8, \beta), \quad \beta \sim \text{Gamma}(0.01, 1)$$

so we have a total of 11 parameters. The joint posterior is then:

$$\begin{aligned}\pi(\lambda, \beta|y, t) &\propto \prod_{i=1}^{10} \text{Pois}(\lambda_i t_i)\text{Ga}(1.8, \beta)\text{Gamma}(0.01, 1) \\ &= \prod_{i=1}^{10} \left(\frac{e^{-\lambda_i t_i}(\lambda_i t_i)^{y_i}}{y_i!} \frac{\beta^{1.8}}{\Gamma(1.8)} \lambda_i^{1.8-1} e^{-\lambda_i \beta} \right) \frac{1}{\Gamma(0.01)} \beta^{0.01-1} e^{-\beta}\end{aligned}$$

Example: nuclear pumps

This formula may be simplified as:

$$\pi(\lambda, \beta | y, t) \propto \prod_{i=1}^{10} \left(e^{-\lambda_i(\beta+t_i)} \lambda^{y_i+1.8-1} \right) \beta^{10\times 1.8+0.01-1} e^{-\beta}$$

Now we may find the full conditional distributions $\pi(\lambda_i | \lambda_{\sim[i]}, \beta, y, t)$ and $\pi(\beta | \lambda, y, t)$, where $\lambda_{\sim[i]}$ denotes the vector λ without the i-th component.

$$\pi(\lambda_i | \lambda_{\sim[i]}, \beta) \propto e^{-\lambda_i(\beta+t_i)} \lambda^{y_i+1.8-1}$$
$$\pi(\beta | \lambda) \propto e^{-(\sum_i \lambda+1)\beta} \beta^{10\times 1.8+0.01-1}$$

Then:

$$\lambda_i | \lambda_{\sim[i]}, \beta, y, t \sim \text{Gamma}(y_i + 1.8, \beta + t_i)$$
$$\beta | \lambda, y, t \sim \text{Gamma}(10 \times 1.8 + 0.01, \sum_i \lambda_i + 1)$$

Example: nuclear pumps

Gibbs sampling:

1. Pick an initial value for β, $\beta^{(0)}$.
2. At iteration $s, s = 1, 2, \ldots, S$: given $\lambda^{(s-1)} = (\lambda_1^{(s-1)}, \ldots, \lambda_{10}^{(s-1)})$ we draw:
   - $\lambda_1^{(s)} \sim \text{Gamma}(y_1 + 1.8, \beta^{(s-1)} + t_1)$
   - $\lambda_2^{(s)} \sim \text{Gamma}(y_2 + 1.8, \beta^{(s-1)} + t_2)$
   - $\ldots$
   - $\lambda_{10}^{(s)} \sim \text{Gamma}(y_{10} + 1.8, \beta^{(s-1)} + t_{10})$
3. We draw $\beta^{(s)}$ from $\text{Gamma}(10 \times 1.8 + 0.01, \sum_i \lambda_i^{(s)} + 1)$, using the vector $\lambda^{(s)}$ found in step 2.
4. Repeat using most updated values until we get S draws.

Comments

- The Gibbs sampler is automatic (no user set parameters) which is good for software. There is not an accept/reject step!
- But, M-H is more general and if dependence in the full conditionals, $\pi(\theta_j|\theta_{\sim[j]}, y)$ is strong, then the Gibbs sampler can be very slow to move around the space, and—in some instances—a joint M-H proposal may be more efficient.
- One can combine the two in a hybrid sampler, updating some components using Gibbs and others using M-H.

Indice

MCMC output analysis

In an ideal world, our simulation algorithm would return *iid* samples from the target (posterior) distribution.

However, MCMC simulation has two short-comings

- The distribution of the samples, $\tilde{\pi}(\theta^{(s)})$ only converges to the target distribution as $s \to \infty$.
- The samples are dependent.

Now we shall consider how we deal with these issues. In typical practice, one monitors the performance of an MCMC algorithm by:

- inspecting the value of the acceptance rate (in M-H only)
- constructing graphs
- computing diagnostic statistics on the stream of simulated values

Convergence

Recall that MCMC is an iterative procedure, such that

- Given the current state of the chain, $\theta^{(s)}$, the algorithm makes a probabilistic update to $\theta^{(s+1)}$.
- The update is made in such a way that the distribution $\tilde{\pi}(\theta^{(s)}) \to \pi(\theta|y)$, the target distribution, as $s \to \infty$, for any starting value $\theta^{(0)}$.

Hence, the early samples are strongly influenced by the distribution of $\theta^{(0)}$, which presumably is not drawn from $\pi(\theta|y)$.

Convergence

Some practical issues for the users:

- The accepted practice is to discard an initial set of samples as being unrepresentative of the steady-state distribution of the Markov chain (the target distribution)
- That is, the first $B+1$ samples $\{\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(B)}\}$ are discarded.
- This *user-defined* initial portion of the chain to discard is known as a burn-in phase for the chain.
- The value of B, the length of burn-in, is determined by the user using various convergence diagnostics which provide evidence that $\tilde{\pi}(\theta^{(B)})$ and $\pi(\theta|y)$ are in some sense close.

Convergence

We may recognize three different kinds of convergence:

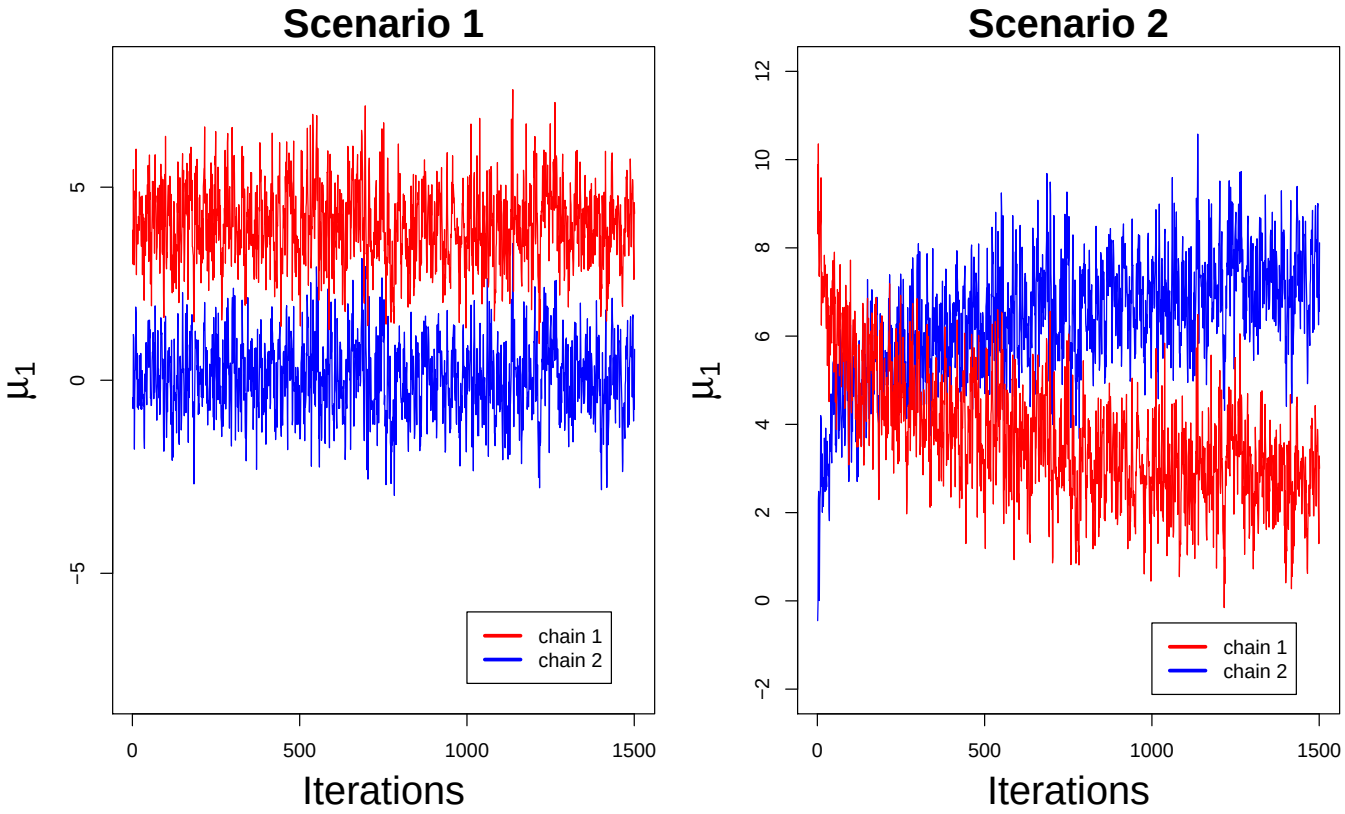
1. to the stationary distribution of the chain
2. of the empirical mean to the integral we wish to approximate
3. to an *iid* sample

In order to check for the convergences above, the MCMC users has several choices, the most popular among them are:

- Running multiple chains from different starting values that are over-dispersed relative to the posterior distribution (and checking the chains mixing).
- Checking the autocorrelation in the draws: a strong correlation between successive iterates may prevent the algorithm from exploring the entire region of the parameters' space.
- Running diagnostic tests: Gelman-Rubin statistics $\hat{R}$, Geweke diagnostics,...

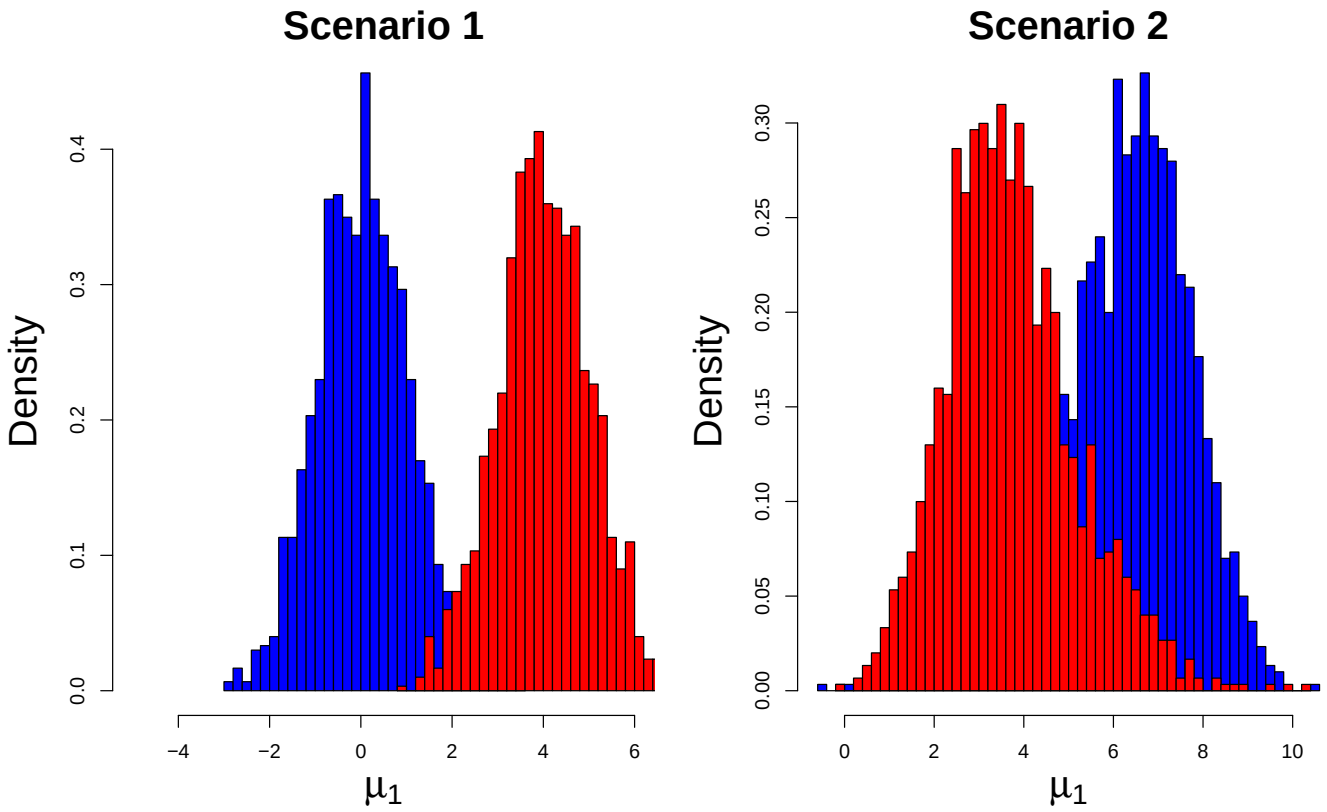
Graphical analysis of convergence

Consider a fake example: we run two chains for a given parameter μ_1, and we find these two distinct situations: what can we say about convergence?



Graphical analysis of convergence

Let's take a look to the estimated densities coming from the two chains:



Bimodal distributions! We cannot use them to make inference on μ_1...

Graphical analysis of convergence

- Scenario 1: Either sequence alone looks stable, but the juxtaposition makes it clear that they have not converged to a common distribution.
- Scenario 2: The two sequences happen to cover a common distribution but neither sequence appears stationary.

These graphs demonstrate the need to use between-sequence and also within-sequence information when assessing convergence.

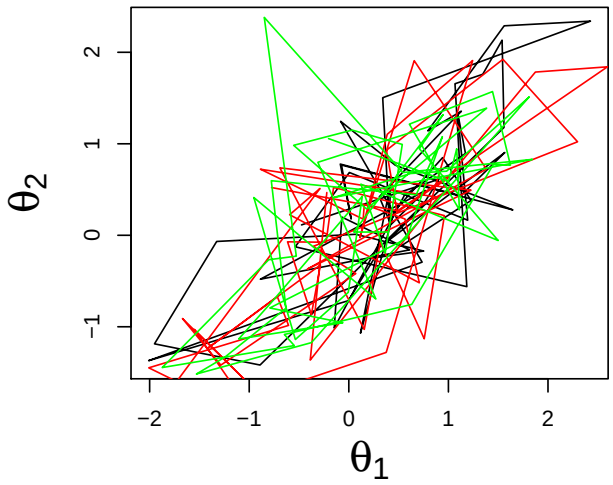
Running multiple chains

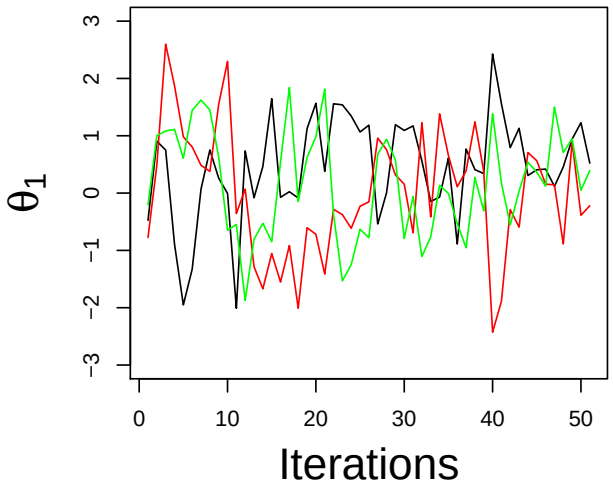
Consider the case of the bivariate normal with the Gibbs sampling. Let's run 100 simulations from three distinct starting points:

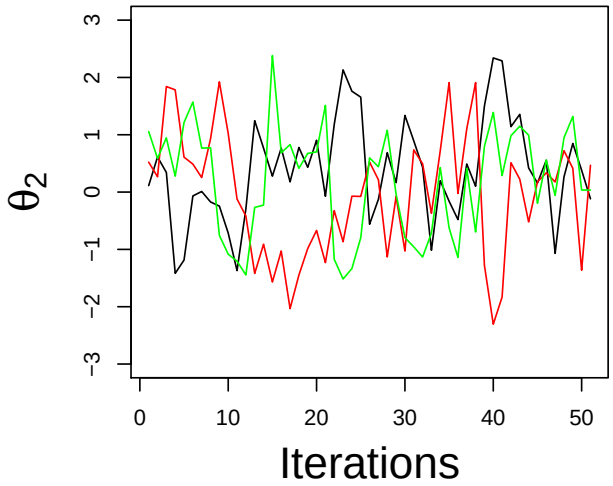
```
my.draws<-bm(init=c(0,0),rho=0.7)
my.draws2<-bm(init=c(5,5), rho=0.7)
my.draws3<-bm(init=c(10,10), rho=0.7)
```

Running multiple chains: 100 samples

1) Convergence: poor, stationarity is not reached.

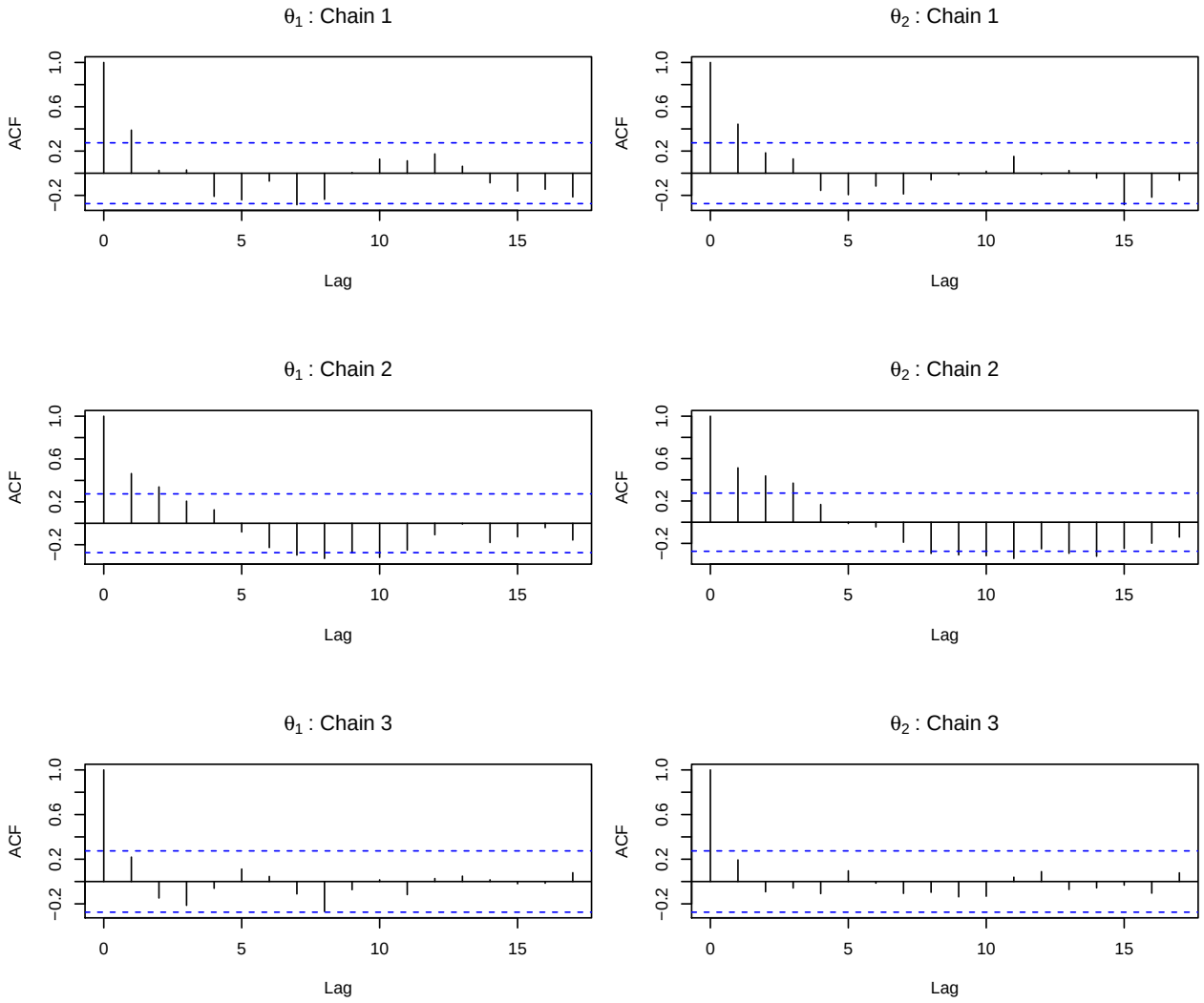






Running multiple chains: 100 samples

2) Autocorrelation: they do not seem iid.



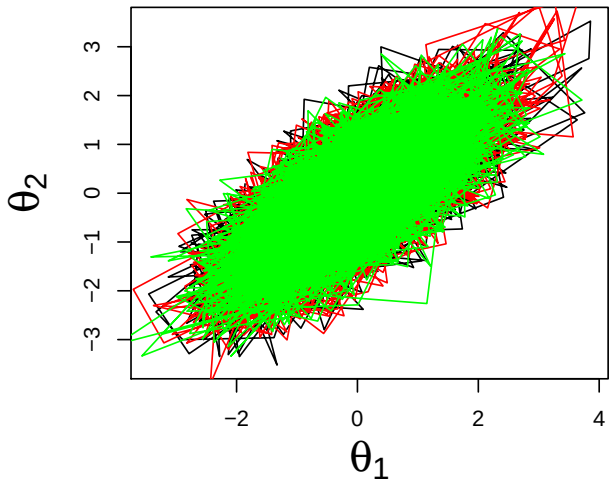
Running multiple chains: 10000 samples

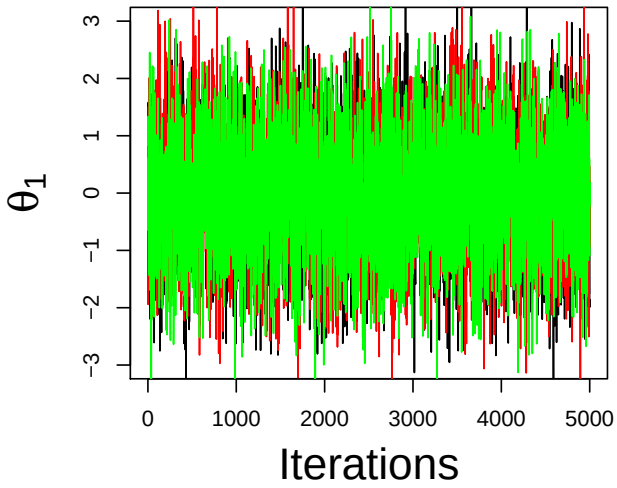
We repeat the same experiments running 10000 iterations:

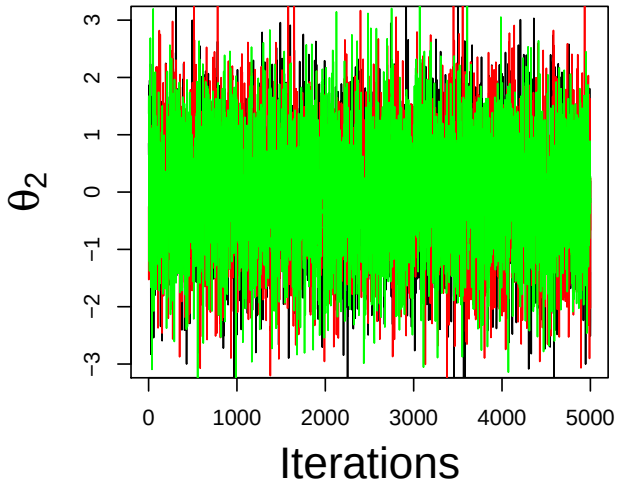
```
my.draws<-bm(init=c(0,0),rho=0.7, nsim=10000)
my.draws2<-bm(init=c(5,5), rho=0.7, nsim=10000)
my.draws3<-bm(init=c(10,10), rho=0.7, nsim=10000)
```

Running multiple chains: 10000 samples

1) Convergence: Ok! Each chain is not discernible from the others.

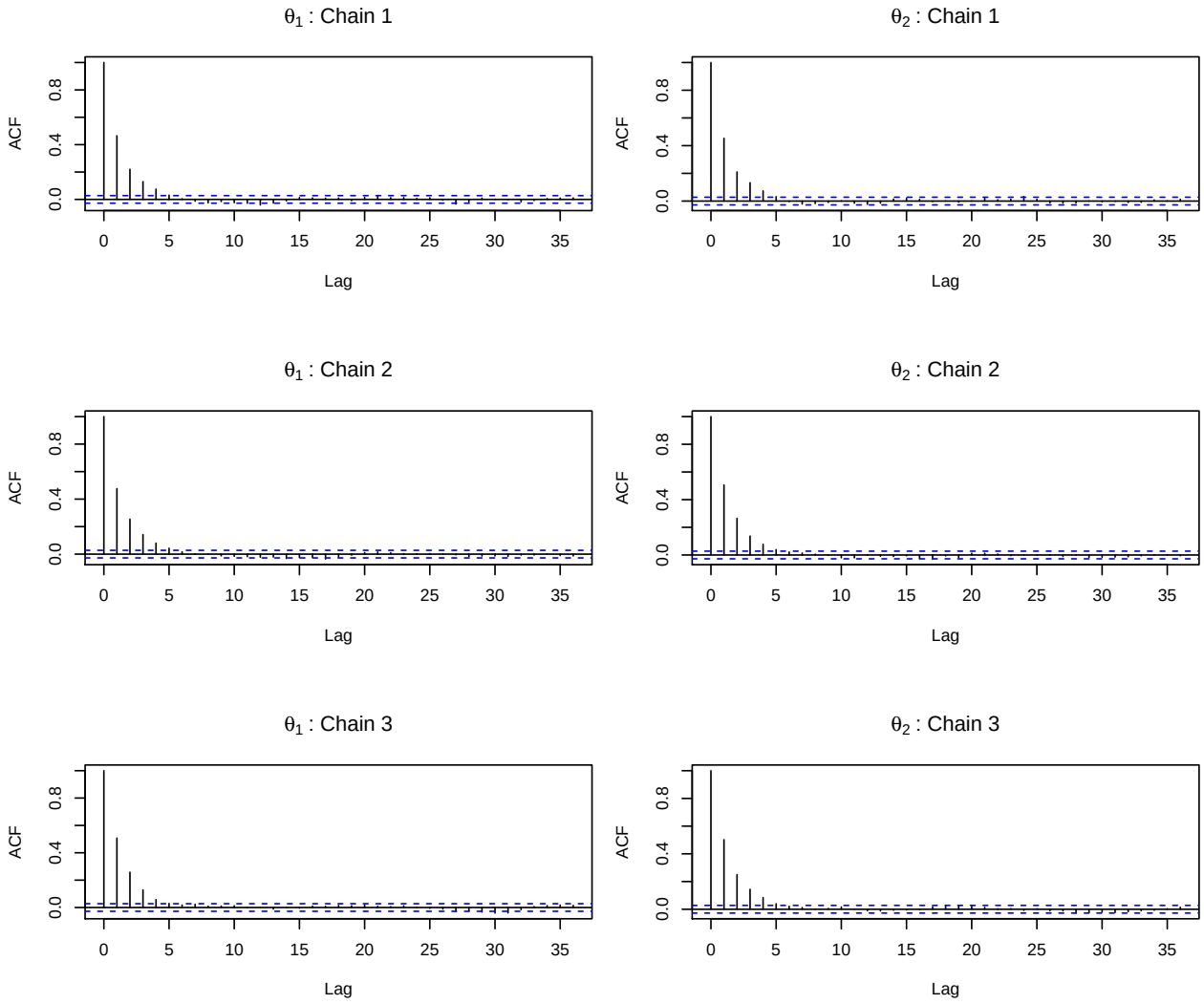






Running multiple chains: 10000 samples

2) Autocorrelation: Ok! Autocorrelation vanishes after a few lags.



Graphical analysis

- Chains should be *mixed*, we should not be able to distinguish one from the others.
- Regardless of the starting points, the chains should converge to the same distribution.
- The chain will often be seen to *migrate* away from $\theta^{(0)}$ toward a region of high posterior probability centred around a mode of $\pi(\theta|y)$.
- Graphical analysis of the chains trace is the first step. Of course, we need more formal tests.

Formal convergence test: Gelman-Rubin statistic

Consider a scalar parameter θ and let $\theta_m^{(s)}$ be the s-th draw in the m-th chain, with $s = 1, \ldots, S$ and $m = 1, \ldots, M$. We may then compute:

- the *between-chains* variance

$$B = \frac{S}{M-1} \sum_{m=1}^{M} (\bar{\theta}_m - \bar{\theta})^2$$

where

$$\bar{\theta}_m = \frac{1}{S} \sum_s \theta_m^{(s)}; \quad \bar{\theta} = \frac{1}{M} \sum_m \bar{\theta}_m.$$

- the *within-chain* variance:

$$W = \frac{1}{M(S-1)} \sum_m \sum_s (\theta_m^{(s)} - \bar{\theta}_m)^2$$

Gelman-Rubin statistic

An unbiased estimator for the posterior variance $\mathrm{Var}(\theta|y)$ is the weighted average:

$$\widehat{\mathrm{Var}(\theta|y)} = \frac{S-1}{S}W + \frac{1}{S}B$$

Early in iterations, $\widehat{\mathrm{Var}(\theta|y)}$ overestimates the true posterior variance:

- Because of overdispersion of the starting values, this overestimates the true variance, but it is unbiased if the starting distribution equals the stationary distribution (if starting values were not overdispersed).
- The *potential scale reduction factor* (or Gelman-Rubin statistic) is:

$$\hat{R} = \sqrt{\frac{\widehat{\mathrm{Var}(\theta|y)}}{W}} \approx \sqrt{\left(1 + \frac{B}{SW}\right)}, \qquad (5)$$

Gelman-Rubin statistic

- For any finite S, the within variance W should be an underestimate of $\mathrm{Var}(\theta|y)$ because the individual sequences have not had time to range over all of the target distribution and, as a result, will have less variability.
- $\hat{R}$ declines to 1 as $S \to \infty$.
- If the potential scale reduction is high, then we have reason to believe that proceeding with further simulations may improve our inference about the target distribution.
- When the chains have mixed (converged) the variance within each sequence and the variance between sequences for each variable will be roughly equal.
- As a golden rule, convergence and mixing of the chains is reached when $\hat{R} \leq 1.1$.

Geweke diagnostic

The Geweke diagnostic takes two nonoverlapping parts (usually the first 0.1 and last 0.5 proportions) of the Markov chain and compares the means of both parts, using a difference of means test to see if the two parts of the chain are from the same distribution (null hypothesis).

The test statistic is a standard Z-score with the standard errors adjusted for autocorrelation.

Remarks

- If $\hat{R}$ is high, try increasing the number of iterations.
- If we have more than one parameter, then we need to calculate the potential scale reduction factor for each parameter. We should run our chains out long enough so that all the potential scale reduction factors are small enough.
- Formal tests for convergence should not be taken without question as evidence for convergence.
- Graphical plots and examining posterior distributions for stability should always be employed for key (functions of) variables of interest
- Running multiple chains from dispersed starting points to check for stability in the estimates (and use the G&R test)

Convergence diagnostics in R

The coda package covers four formal tests for convergence, perhaps the two most popular ones being those reported by Geweke and those of Gelman and Rubin.

Before we use the diagnostics, we should turn our chains into mcmc objects.

```
library(coda)
  chain1 <- mcmc(my.draws)
  chain2 <- mcmc(my.draws2)
  chain3 <- mcmc(my.draws3)
```

- We can tell the `mcmc()` function to burn-in or drop draws with the start and end arguments.
- `mcmc()` also has a thin argument, which only tells it the thinning interval that was used (it does not actually thin for us).

Convergence diagnostics in R

To implement the G&R test in the bivariate Gaussian example:

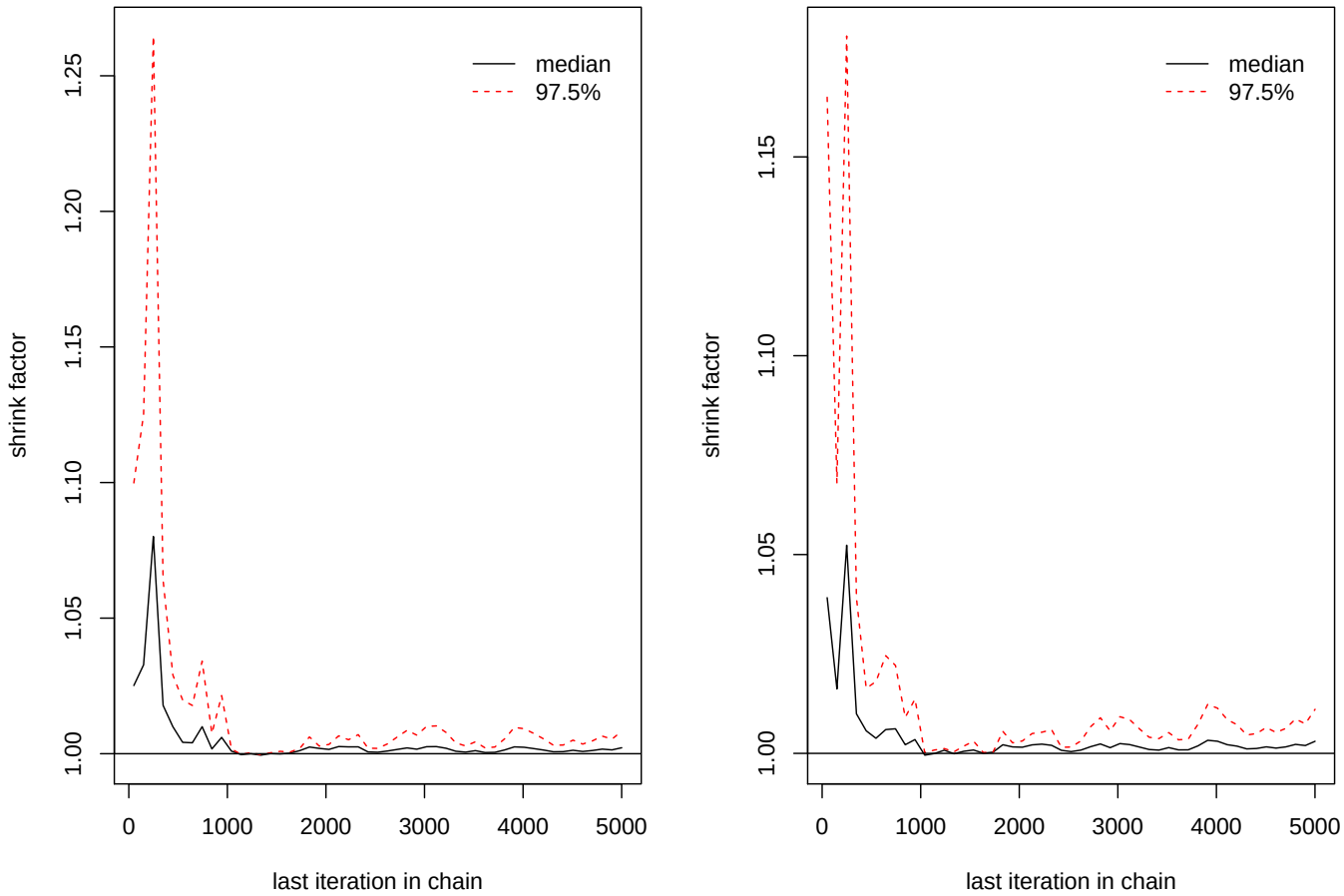
```
gelman.diag(combinechains)
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1       1.01
[2,]          1       1.01

Multivariate psrf

1
```

For both the parameters, $\hat{R} \leq 1$.

Convergence diagnostics in R

We may also produce a plot for the two parameters:

```
gelman.plot(combinechains)
```



Convergence diagnostics in R

To implement the Geweke test for the bivariate Gaussian case:

```
geweke.diag(chain1)

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5
var1  var2
1.604 1.189

geweke.diag(chain2)

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5
var1   var2
0.3123 0.4908
```

We accept the null hypothesis of means equality for both the parameters and both the chains.

Formal autocorrelation test

Once the simulated sequence have mixed, we can compute an approximate *effective number of independent simulation draws* for any estimand of interest θ.

If the S simulations draws within each sequence were truly independent, then the between-variance B would be an unbiased estimate of the posterior variance, $\mathrm{Var}(\theta|y)$, and we would have a total of $S \times M$ independent simulations from the M chains.

One way to define the effective sample size for correlated simulations is to consider the statistical efficiency of the average of the simulations, $\bar{\theta}$, as an estimate for the posterior mean $E(\theta|y)$. Then, we define the limit:

$$\lim_{S\to\infty} SM\mathrm{Var}(\bar{\theta}) = \left(1 + 2\sum_{t=1}^{\infty}\rho_t\right)\mathrm{Var}(\theta|y). \tag{6}$$

Formal autocorrelation test

In equation (6), ρ_t is the autocorrelation of the sequence θ at lag t. If the S simulation draws from each of the M chains were independent, then $\mathrm{Var}(\bar{\theta})$ would simply be $\frac{1}{SM}\mathrm{Var}(\theta|y)$ and the sample size would be SM. In the presence of correlation we then define the *effective sample size* as

$$n_{\text{eff}} = \frac{SM}{1 + 2\sum_{t=1}^{\infty} \rho_t}, \tag{7}$$

and we can estimate this quantity by:

$$\hat{n}_{\text{eff}} = \frac{SM}{1 + 2\sum_{t=1}^{T} \hat{\rho}_t}. \tag{8}$$

- The ESS estimates the reduction in the true number of samples, compared to iid samples, due to the autocorrelation in the chain. The closer to $S \times M$ the better for ESS.

Further reading

Further reading:

- Chapter 6 from *Bayesian Computation with R*, J. Albert.
- Chapter 6 and 7 from *Introducing Monte Carlo Methods with R*, C. Robert and G. Casella.
- Chapter 11 from *Bayesian Data Analysis* (3rd ed.), A. Gelman et al.

Strong and deep reading:

- Chapters 4,6,7 and 8 from *Monte Carlo Statistical Methods*, C. Robert and G. Casella.