The Metropolis Hastings algorithm, Part 2

Matthew Stephens
University of Chicago
January 23, 2026

See here for a PDF version of this vignette.

Prerequisites

You should be familiar with the basics of the Metropolis-Hastings algorithm.

Introduction

In this vignette, we follow up on the original algorithm with a couple of points: (i) numerical issues that were glossed over, and (ii) a useful plot.

Avoiding numerical issues in the acceptance probability

The key to the M-H algorithm is computing the acceptance probability, which recall is given by

$$A = \min\left\{1, \frac{\pi(y)Q(x_t \mid y)}{\pi(x_t)Q(y \mid x_t)}\right\}.$$

In practice both terms in the fraction may be very close to 0, so on a computer you should do this computation on the logarithmic scale before exponentiating; something like this:

$$\frac{\pi(y)Q(x_t \mid y)}{\pi(x_t)Q(y \mid x_t)} = \exp\left[\log \pi(y) + \log Q(x_t \mid y) - \log \pi(x_t) - \log Q(y \mid x_t)\right]$$

Furthermore, the log values in this expression should be computed directly, and not by computing them and then taking the log. For example, in our previous example we sampled from a target distribution that was the exponential:

$$\pi(x) = \exp(-x) \qquad (x > 0)$$

we can directly compute $\log \pi(x) = -x$. The code we had in that previous example can therefore be written as follows:

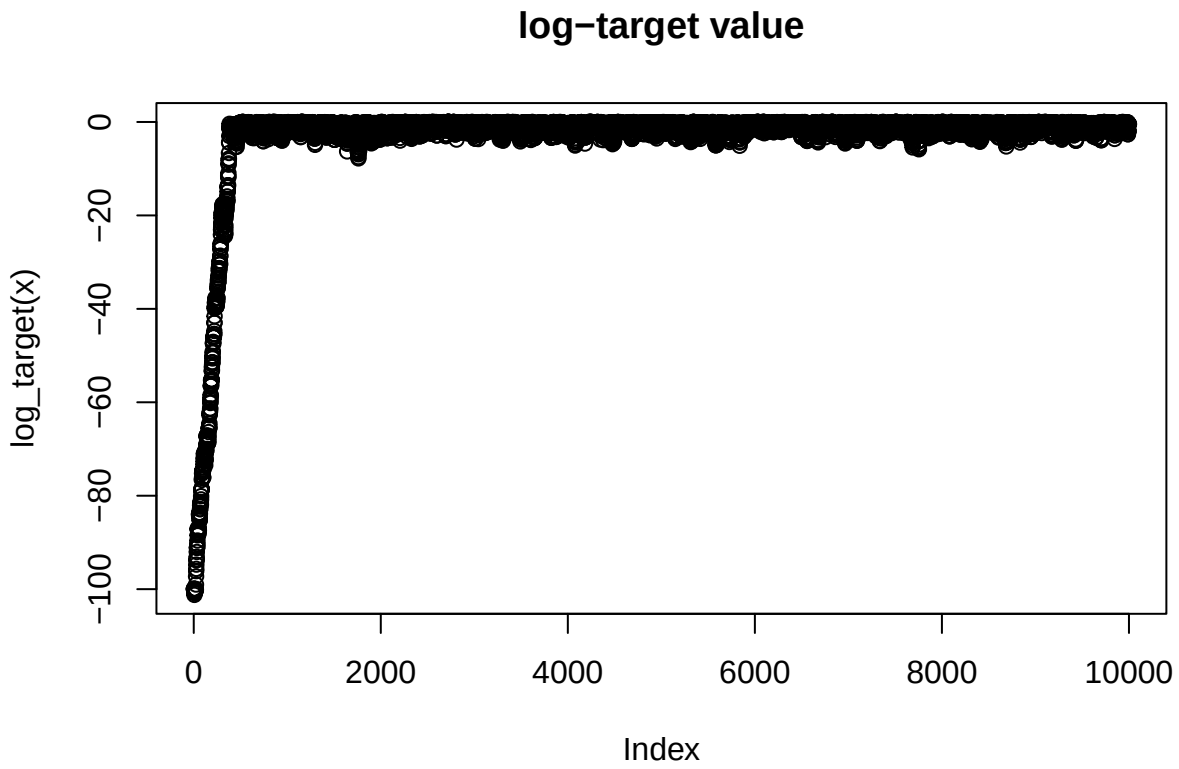
```
log_target <- function(x){
  return(ifelse(x<0,-Inf,-x))
}
x = rep(0,10000)
x[1] = 100
```

```
for(i in 2:10000){
  current_x = x[i-1]
  proposed_x = current_x + rnorm(1,mean=0,sd=1)
  A = exp(log_target(proposed_x) - log_target(current_x) )
  if(runif(1)<A){
    x[i] = proposed_x       # accept move with probabily min(1,A)
  } else {
    x[i] = current_x        # otherwise "reject" move, and stay where we are
  }
}
```

An important and useful plot

In practice MCMC is usually performed in high dimensional space. It can therefore be really hard to visualize directly the values of the chain. A simple 1-d summary that is always available is the log-target density, $\log \pi(x)$. So you should usually plot a trace of this whenever you run an MCMC scheme.

```
plot(log_target(x), main = "log-target value")
```



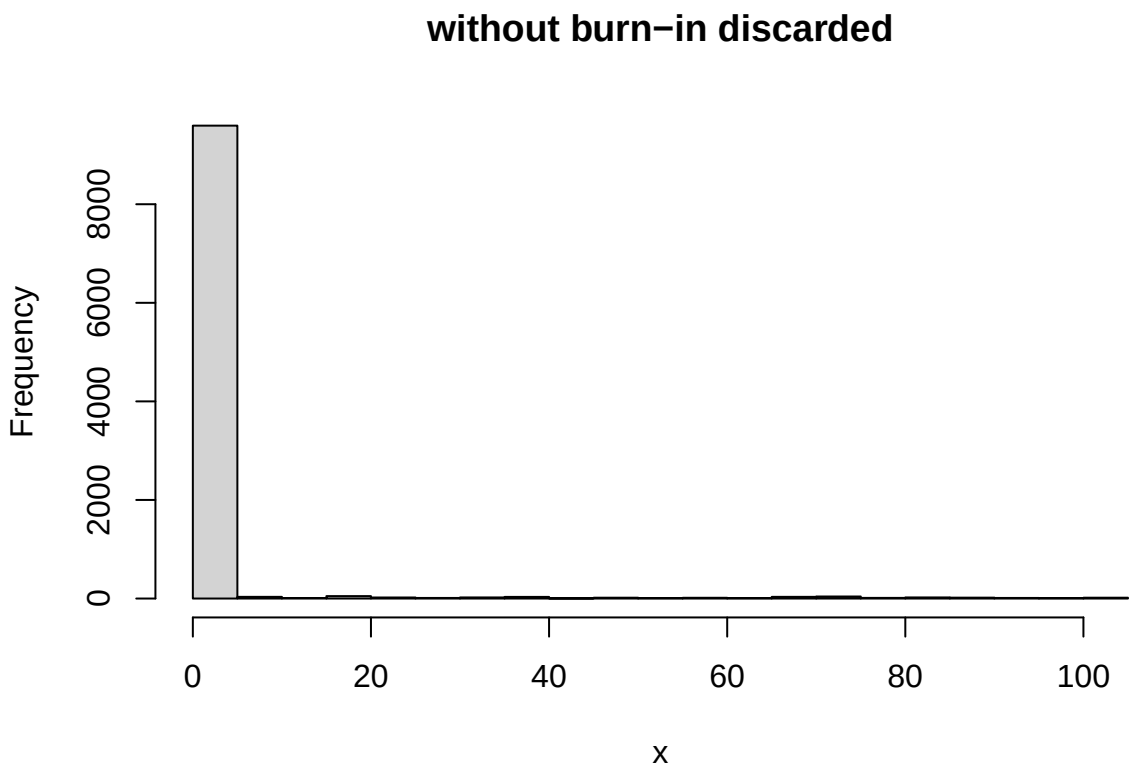
Here the plot shows "typical" behaviour of MCMC scheme: because the starting point is chosen not to be close to the optimal of π, the chain initially takes some iterations to find a part of the space where $\pi(x)$ is "large". Once it finds that part of the space it starts to explore around the region where $\pi(x)$ is large.

Burn-in

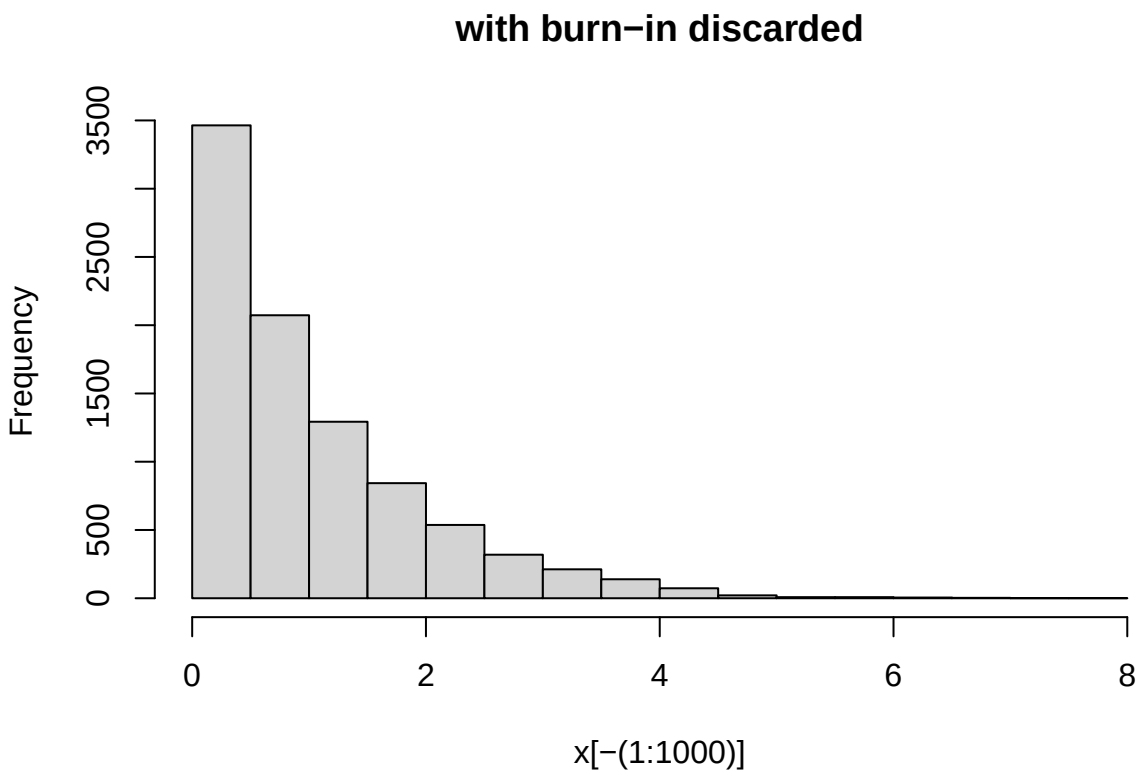
The plot in the previous section immediately shows that there is an initial period of time where the Markov Chain is unduly influenced by its starting position. In other words, during those iterations the Markov Chain has not "converged" and those samples should not be considered to be samples from π. To address this it is common to discard the first set of iterations of any chain; the iterations that are discarded are often called "burn-in".

Here, based on the plot we might discard the first 1000 iterations or so as burn-in. Here are comparisons of the samples with and without burnin discarded:

```
hist(x, main="without burn-in discarded")
```




```
hist(x[-(1:1000)], main="with burn-in discarded")
```


with burn−in discarded
Frequency
0
500
1500
2500
3500
0
2
4
6
8
x[−(1:1000)]