Simulating discrete Markov chains: an introduction

Matt Bonakdarpour
University of Chicago
January 15, 2026

See here for a PDF version of this vignette.

Pre-requisites

This document assumes basic familiarity with Markov chains.

Illustrative Example

In this note, we will describe a simple algorithm for simulating Markov chains. We first settle on notation and describe the algorithm in words.

Let P_{ij} denote the one-step transition probability. That is,

$$P_{ij} = P(X_{t+1} = j | X_t = i)$$

In what follows, we will assume that the transition probabilities do not depend on time t. These are called *time homogenous* Markov chains.

The general idea of simulating discrete Markov chains can be illustrated through a simple example with 2 states. Assume our state space is $\{1, 2\}$ and the transition matrix is:

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

We denote the (i, j)-th entry of the matrix P as P_{ij}.

Now assume that our Markov chain starts in state 1 so that $X_0 = 1$. Since we are starting in state 1, our transition probabilities are defined by the first row of P. Our chain can either remain in state 1 with probability P_{11} or transition to state 2 with probability P_{12}. Therefore, to simulate X_1, we must generate a random variable according to the probabilities $P_{11} = P(X_1 = 1 | X_0 = 1) = 0.7$ and $P_{12} = P(X_1 = 2 | X_0 = 1) = 0.3$.

In general, we can generate any discrete random variable according to a set of probabilities $p = \{p_1, \ldots, p_K\}$ with Inverse Transform Sampling. Also note that this is equivalent to taking a single draw from a multinomial distribution with probability vector p – we use this method in the algorithm below.

General Algorithm

Here we present a general algorithm for simulating a discrete Markov chain assuming we have S possible states.

1. Obtain the $S \times S$ probability transition matrix P
2. Set $t = 0$
3. Pick an initial state $X_t = i$.
4. For t = 1...T:
    1. Obtain the row of P corresponding to the current state X_t.
    2. Generate X_{t+1} from a multinomial distribution with probability vector equal to the row we obtained above.

We implement this in the following function, initializing the first state to 1:

```
# simulate discrete Markov chains according to transition matrix P
run.mc.sim <- function( P, num.iters = 50 ) {

  # number of possible states
  num.states <- nrow(P)

  # stores the states X_t through time
  states     <- numeric(num.iters)

  # initialize variable for first state
  states[1]    <- 1

  for(t in 2:num.iters) {

    # probability vector to simulate next state X_{t+1}
    p  <- P[states[t-1], ]

    ## draw from multinomial and determine state
    states[t] <-  which(rmultinom(1, 1, p) == 1)
  }
  return(states)
}
```

Simulation 1: 3x3 example

Assume our probability transition matrix is:

$$P = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.6 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

We initialize this matrix in R below:

```r
# setup transition matrix
P <- t(matrix(c( 0.7, 0.2, 0.1,
                 0.4, 0.6,   0,
                   0,   1,   0  ), nrow=3, ncol=3))
```

Now we will use the function we wrote in the previous section to run several chains and plot the results:

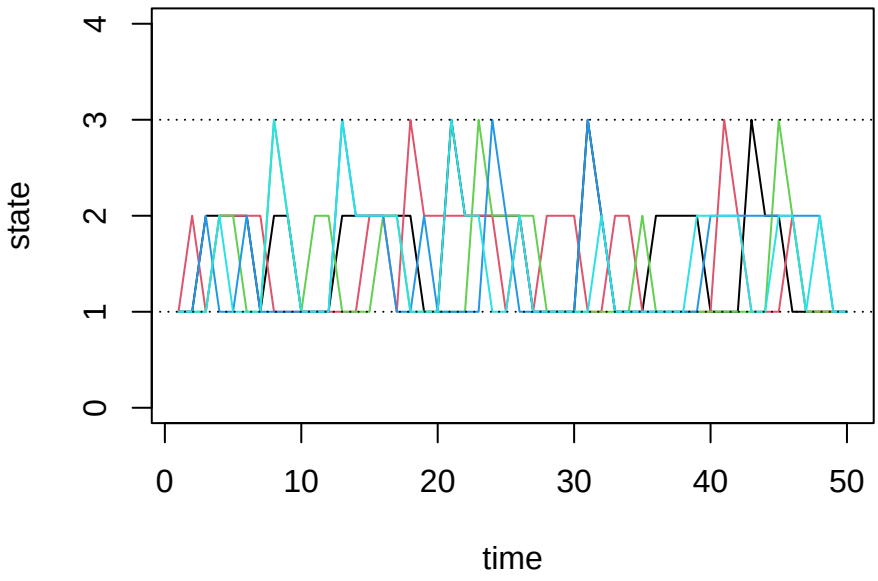
```r
num.chains     <- 5
num.iterations <- 50

# each column stores the sequence of states for a single chains
chain.states  <- matrix(NA, ncol=num.chains, nrow=num.iterations)

# simulate chains
for(c in seq_len(num.chains)){
  chain.states[,c] <- run.mc.sim(P)
}
```

Our function returns a vector that contains the states of our simulated chain through time. Recall that our state space is $\{1, 2, 3\}$. Below, we visualize how these chains evolve through time:

```r
matplot(chain.states, type='l', lty=1, col=1:5, ylim=c(0,4), ylab='state', xlab='time')
abline(h=1, lty=3)
abline(h=3, lty=3)
```



Simulation 2: 8x8 example

Next we will do a larger experiment with the size of our state space equal to 8. Assume our probability transition matrix is:

$$P = \begin{bmatrix} 0.33 & 0.66 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & 0.33 & 0.33 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.33 & 0.33 & 0.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0.33 & 0.33 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.33 & 0.33 & 0.33 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.66 & 0.33 \end{bmatrix}$$

We first initialize our transition matrix in R:

```
P <- t(matrix(c( 1/3, 2/3,   0,   0,  0,   0,   0,   0,
                 1/3, 1/3, 1/3,   0,  0,   0,   0,   0,
                   0, 1/3, 1/3, 1/3,  0,   0,   0,   0,
                   0,   0, 1/3, 1/3, 1/3,  0,   0,   0,
                   0,   0,   0, 1/3, 1/3, 1/3,  0,   0,
                   0,   0,   0,   0, 1/3, 1/3, 1/3,  0,
                   0,   0,   0,   0,   0, 1/3, 1/3, 1/3,
                   0,   0,   0,   0,   0,   0, 2/3, 1/3), nrow=8, ncol=8))
```

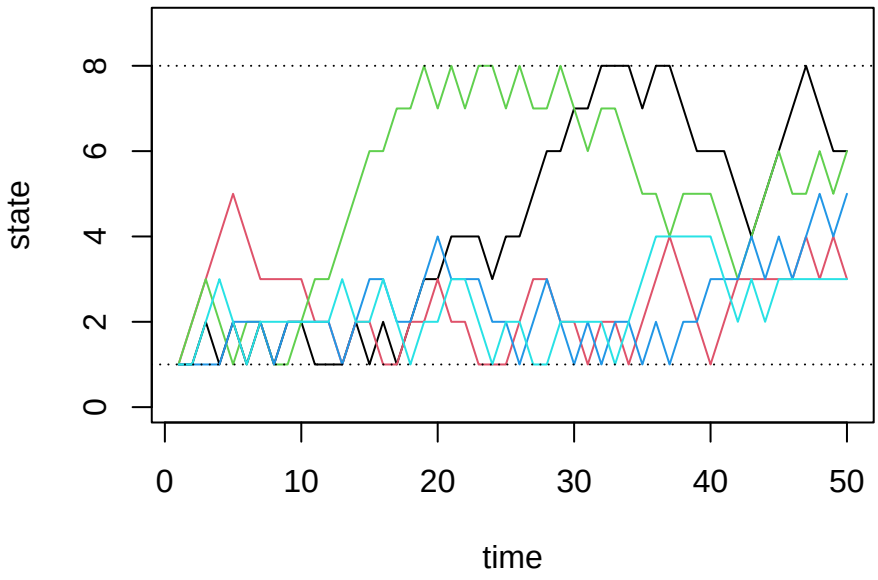
After briefly studying this matrix, we can see that for states 2 through 7, this transition matrix forces the chain to either stay in the current state or move one state up or down, all with equal probability. For the edge cases, states 1 and 8, the chain can either stay or reflect towards the middle states.

Now we run our simulations with the transition matrix above:

```
num.chains     <- 5
num.iterations <- 50
chain.states <- matrix(NA, ncol=num.chains, nrow=num.iterations)
for(c in seq_len(num.chains)){
  chain.states[,c] <- run.mc.sim(P)
}
```

And finally we plot the chains through time below:

```
matplot(chain.states, type='l', lty=1, col=1:5, ylim=c(0,9), ylab='state', xlab='time')
abline(h=1, lty=3)
abline(h=8, lty=3)
```


state
0
2
4
6
8
0
10
20
30
40
50
time