Simulating discrete Markov chains: limiting disributions

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 simulating Markov chains, as seen here: Simulating Discrete Markov Chains: An Introduction.

Overview

In this note, we show the empirical relationship between the stationary distribution, limiting probabilities, and empirical probabilities for discrete Markov chains.

Limiting Probabilities

Let $\pi^{(0)}$ be our initial probability vector. For example, if we had a 3 state Markov chain with $\pi^{(0)} = [0.5, 0.1, 0.4]$, this would tell us that our chain has a 50% probability of starting in state 1, a 10% probability of starting in state 2, and a 40% probability of starting in state 3. If we wanted to have our initial state equal to 1, we would set $\pi^{(0)} = [1, 0, 0]$.

Let P be our probability transition matrix. Recall that the probability vector after n steps is equal to:

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

where P^n is the matrix P raised to the n-th power.

With the facts above, we could keep track of our probability vector $\pi^{(n)}$ as we simulate the Markov chain as follows:

1. Obtain probability transition matrix P
2. Set $P_n = P$
3. For t = 1…T:
    1. Simulate next step of Markov chain.
    2. Set $\pi^{(n)} = \pi^{(0)} P_n$
    3. Set $P_n = P_n P$

We augment our function for simulating Markov chains from this note with the following changes:
1. We keep track of $\pi^{(n)}$.

2. We add the functionality of simulating multiple chains at once – as a result, the vectors we dealt with in the previous note have turned into matrices.
3. We use the `rmultinom()` function instead of our `inv.transform.sample()` method.

```
# simulate discrete Markov chains
run.mc.sim <- function( P,   # probability transition matrix
                        num.iters=50,
                        num.chains=150 )
  {

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

  # states X_t for all chains
  states     <- matrix(NA, ncol=num.chains, nrow=num.iters)

  # probability vectors pi^n through time
  all_probs  <- matrix(NA, nrow=num.iters, ncol=num.states)

  # forces chains to start in state 1
  pi_0      <- c(1, rep(0, num.states-1))

  # initialize variables for first state
  P_n           <- P
  all_probs[1,] <- pi_0
  states[1,]    <- 1

  for(t in 2:num.iters) {

    # pi^n for this iteration
    pi_n           <- pi_0 %*% P_n
    all_probs[t,]  <- pi_n

    for(chain_num in seq_len(num.chains)) {
      # probability vector to simulating next state
      p                     <- P[ states[t-1,chain_num], ]
      states[t,chain_num]   <- which(rmultinom(1, 1, p) == 1)
    }

    # update probability transition matrix
    P_n           <- P_n %*% P
  }
  return(list(all.probs=all_probs, states=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))
```

We will inspect the limiting probabilities and compare them to the stationary distribution. In this note, we derived the stationary distribution for this transition matrix. Recall that the stationary distribution π is the vector such that

$$\pi = \pi P$$

We found that:

$$\pi_1 \approx 0.54, \pi_2 \approx 0.41, \pi_3 \approx 0.05$$

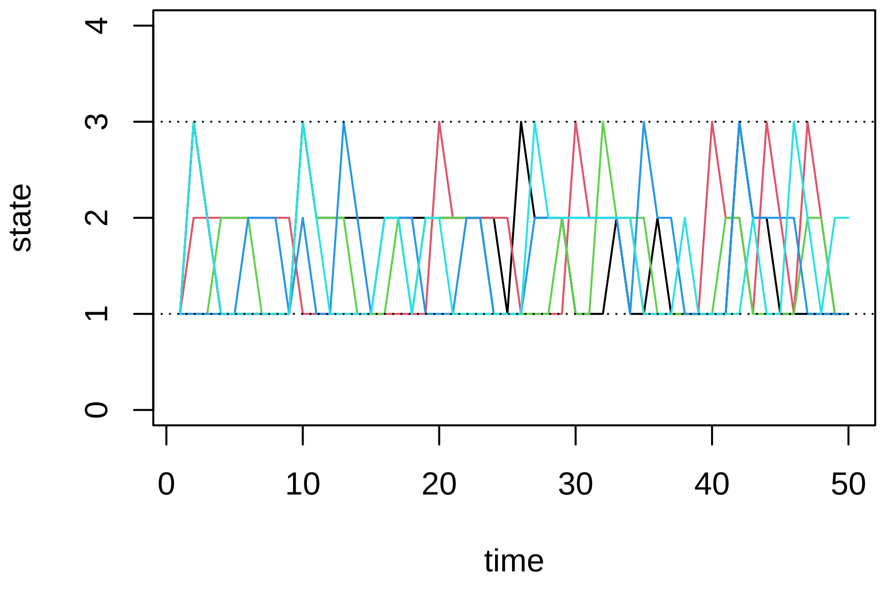
Therefore, with proper conditions (see below), we expect the Markov chain to spend more time in states 1 and 2 as the chain evolves.

Now we will use the function we wrote in the previous section to check this result numerically.

```r
sim1 <- run.mc.sim(P)
```

Our function returns a list containing two matrices. The second matrix called "states" contains the states of each of our simulated chains through time. Recall that our state space is $\{1, 2, 3\}$. Below, we first visualize how 5 of these chains evolve:

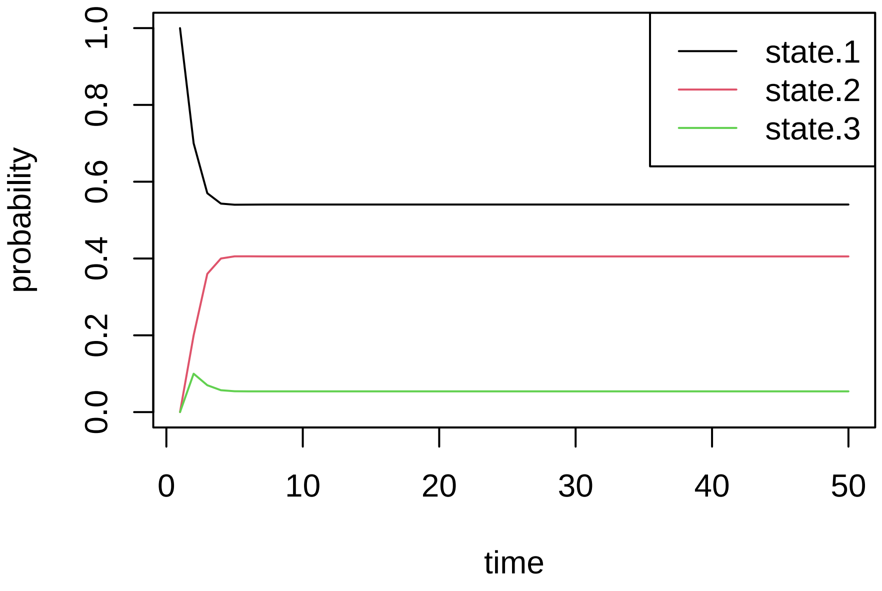
```r
states <- sim1[[2]]
matplot(states[,1:5], type='l', lty=1, col=1:5, ylim=c(0,4), ylab='state', xlab='time')
abline(h=1, lty=3)
abline(h=3, lty=3)
```



The first matrix we get from our function contains $\pi^{(n)}$ through time. We can see how $\pi^{(n)}$ evolves as n grows, and we can check if it converges to the stationary distribution we found above. For irreducible finite state Markov chains, note that $\pi^{(n)}$ converges if and only if the Markov chain is aperiodic. In this note, we only consider finite, irreducible, and aperiodic Markov chains.

Using $\pi^{(n)}$, we plot the time evolution of the probability of being in each state:

```
all.probs <- sim1[[1]]
matplot(all.probs, type='l', col=1:3, lty=1, ylab='probability', xlab='time')
legend('topright', c('state.1', 'state.2', 'state.3'), lty=1, col=1:3)
```



Indeed, we see that these probabilities quickly converge. Just by looking at the plot, we can see that the final probabilities are about equal to the stationary distribution π we found above.

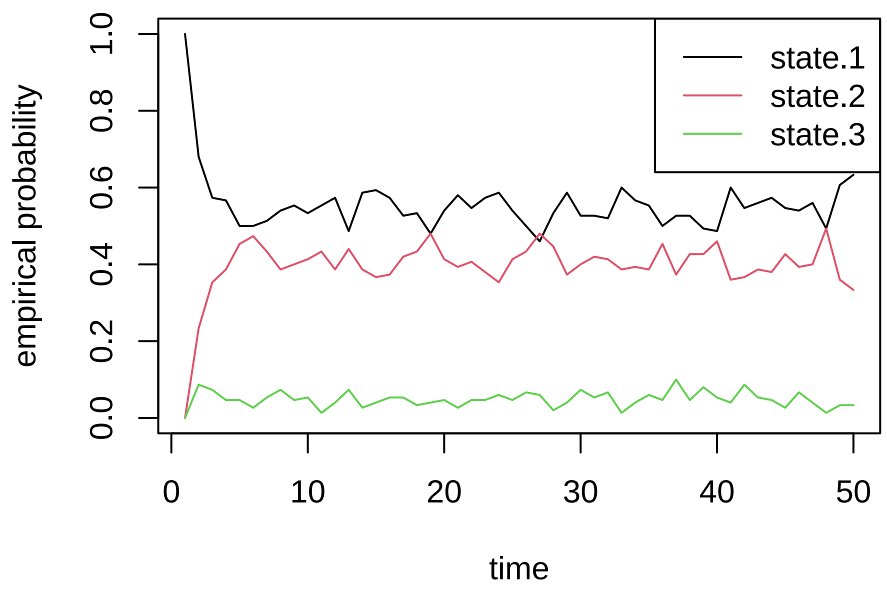
By inspecting the actual values, we confirm that the values of $\pi^{(n)}$ converge to the vector π exactly. The first row in the matrix below is from the simulation, and the second row is the quantity we obtained by solving for the stationary distribution:

```
# solve for stationary distribution
A         <- matrix(c(-0.3, 0.2, 0.1, 1, 0.4, -0.4, 0, 1, 0, 1, -1, 1 ), ncol=3,nrow=4)
b         <- c(0,0,0, 1)
pi        <- drop(solve(t(A) %*% A, t(A) %*% b))

# show comparison
results1           <- t(data.frame(pi_n = all.probs[50,], pi = pi))
colnames(results1) <- c('state.1', 'state.2', 'state.3')
results1
#      state.1 state.2 state.3
# pi_n  0.5405  0.4054 0.05405
# pi    0.5405  0.4054 0.05405
```

Finally, we can also plot the proportion of chains that are in each state through time. These should roughly equal the probability vectors above, with some noise due to random chance:

```
state.probs <- t(apply(apply(sim1[[2]], 1, function(x) table(factor(x, levels=1:3))), 2, functio
matplot(state.probs[1:50,], col=1:3, lty=1, type='l', ylab='empirical probability', xlab='time')
legend('topright', c('state.1', 'state.2', 'state.3'), lty=1, col=1:3)
```



Simulation 2: 8x8 example

Next we will quickly do two larger experiments 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))
```

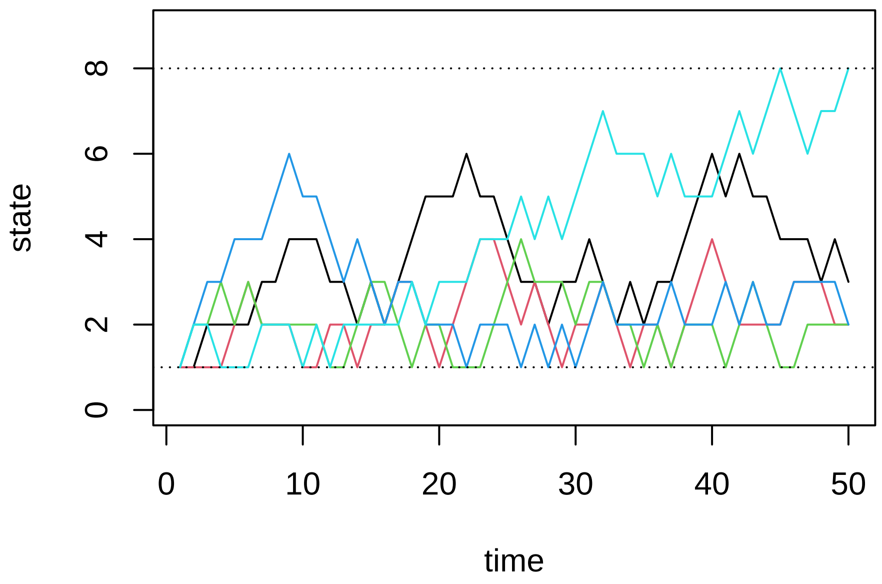
After briefly studying this matrix, we can see that for states 2 through 7, this transition matrix forces the chain to either stay in the current state or move one state up or down, all with equal probability. For the edge cases, states 1 and 8, the chain can either stay or reflect towards the middle states. Since it's "easier" to get to one of the middle states (either from above or below), we should see that the probabilities for these states converge to a higher number than the states on the boundaries.

Now we run our simulation with the transition matrix above:

```
sim2a <- run.mc.sim(P)
```

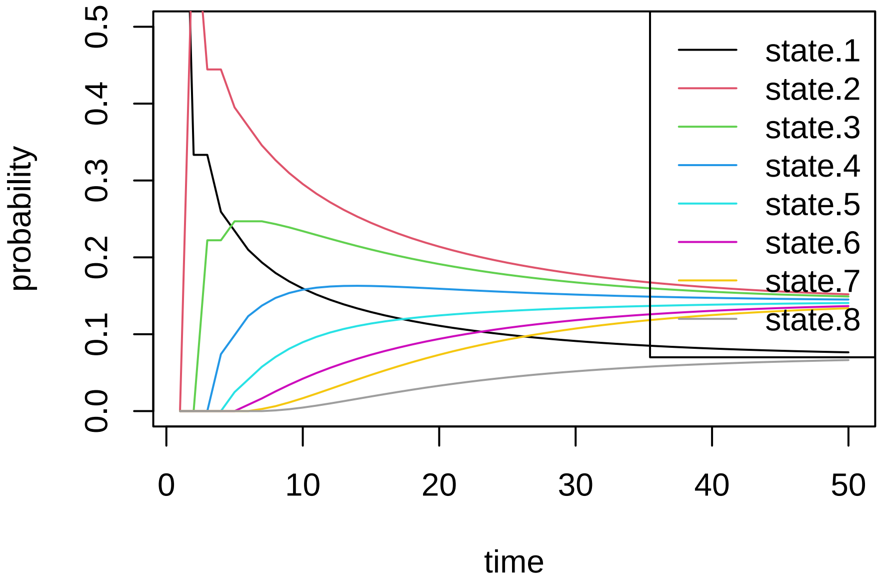
and now plot 5 of the chains through time below:

```
states <- sim2a[[2]]
matplot(states[,1:5], 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)
```



Next we inspect $\pi^{(n)}$ through time:

```
all.probs <- sim2a[[1]]
matplot(all.probs, type='l', col=1:8, lty=1, ylab='probability',
        xlab='time', ylim=c(0, 0.5))
legend('topright', paste('state.', 1:8, sep=''), lty=1, col=1:8)
```



These results match our intuition above. The probability of being in states 1 and 8 converge to smaller values than the others.

Now we alter the transition matrix above to encourage the chain to stay in states 4 and 5:

$$P = \begin{bmatrix} 0.33 & 0.66 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.08 & 0.08 & 0.84 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.08 & 0.84 & 0.08 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.08 & 0.84 & 0.08 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.84 & 0.08 & 0.08 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.33 & 0.33 & 0.33 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.66 & 0.33 \end{bmatrix}$$

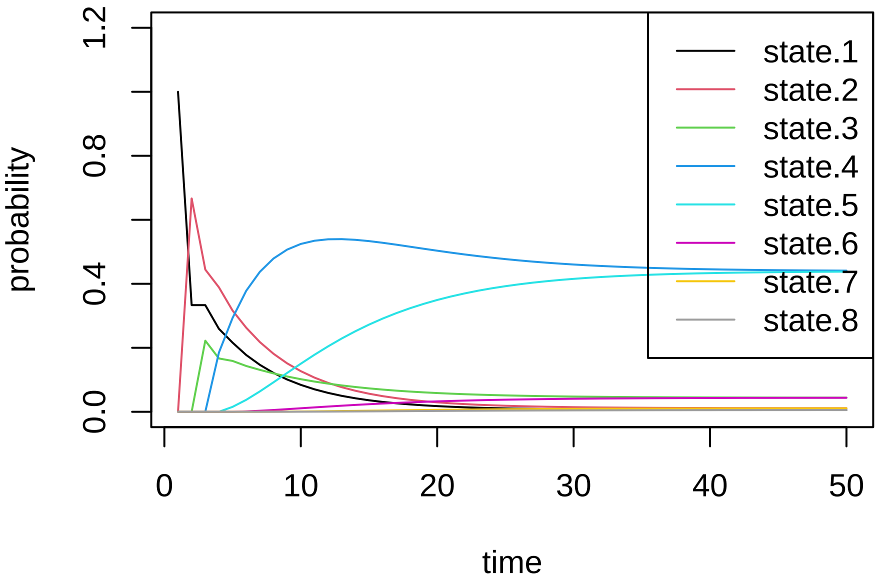
and initialize the transition matrix in R:

```
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,  .5/6, .5/6,   5/6,     0,     0,     0,    0,
                   0,     0, .5/6,   5/6,  .5/6,     0,     0,    0,
                   0,     0,    0,  .5/6,   5/6,  .5/6,     0,    0,
                   0,     0,    0,     0,   5/6,  .5/6,  .5/6,    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))

sim2b <- run.mc.sim(P)
```

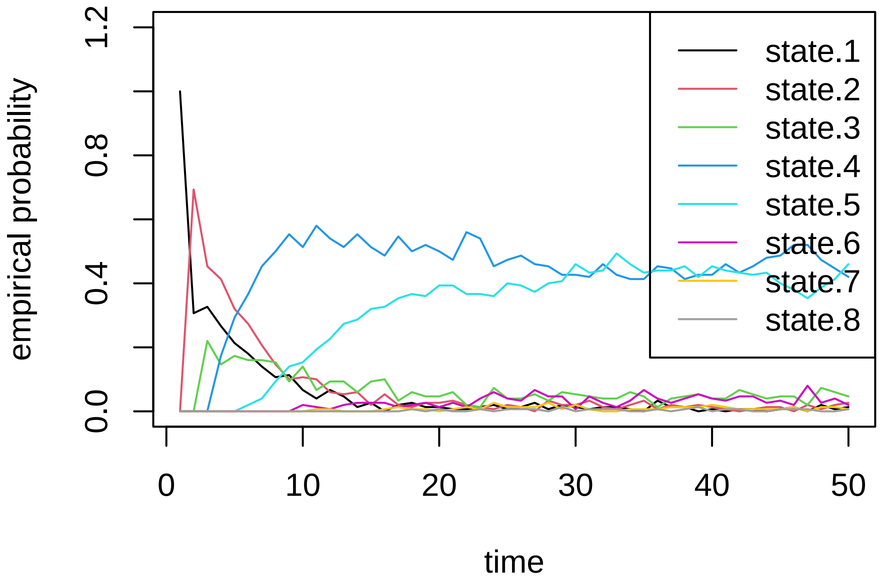
Below we inspect $\pi^{(n)}$ through time and see that the probability vector converges to a vector placing most of the probability mass on states 4 and 5.

```
all.probs <- sim2b[[1]]
matplot(all.probs, type='l', col=1:8, lty=1, ylab='probability',
        xlab='time', ylim=c(0,1.2))
legend('topright', paste('state.', 1:8, sep=''), lty=1, col=1:8)
```



Finally we confirm that our empirical probabilities also exhibit similar behavior:

```
state.probs <- t(apply(apply(sim2b[[2]], 1, function(x) table(factor(x, levels=1:8))), 2, functi
matplot(state.probs[1:50,], col=1:8, lty=1, type='l', ylab='empirical probability', xlab='time',
legend('topright', paste('state.', 1:8, sep=''), lty=1, col=1:8)
```


empirical probability
time
state.1
state.2
state.3
state.4
state.5
state.6
state.7
state.8