Discrete Markov chains: finding the stationary distribution

Matt Bonakdarpour, with contributions by Matthew Stephens and Jitao David Zhang
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 and linear algebra.

Overview

In this note, we illustrate one way of analytically obtaining the stationary distribution for a finite discrete Markov chain.

3x3 example

Assume our probability transition matrix is:

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

Since every state is *accessible* from every other state, this Markov chain is irreducible. Every irreducible finite state space Markov chain has a unique stationary distribution. Recall that the stationary distribution π is the row vector such that

$$\pi = \pi P$$

.

Therefore, we can find our stationary distribution by solving the following linear system:

$$\begin{aligned} 0.7\pi_1 + 0.4\pi_2 &= \pi_1 \\ 0.2\pi_1 + 0.6\pi_2 + \pi_3 &= \pi_2 \\ 0.1\pi_1 &= \pi_3 \end{aligned}$$

subject to $\pi_1 + \pi_2 + \pi_3 = 1$. Putting these four equations together and moving all of the variables

to the left hand side, we get the following linear system:

$$\begin{aligned} -0.3\pi_1 + 0.4\pi_2 &= 0 \\ 0.2\pi_1 + -0.4\pi_2 + \pi_3 &= 0 \\ 0.1\pi_1 - \pi_3 &= 0 \\ \pi_1 + \pi_2 + \pi_3 &= 1 \end{aligned}$$

We will define the linear system in matrix notation:

$$\underbrace{\begin{bmatrix} -0.3 & 0.4 & 0 \\ 0.2 & -0.4 & 1 \\ 0.1 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{b} \quad A\pi^T = b$$

The stationary distribution, which is usually represented by a row vector, is transposed with π^T.

Since this linear system has more equations than unknowns, it is an overdetermined system. Overdetermined systems can be solved using a QR decomposition, so we use that here. (In brief, `qr.solve` works by finding the QR decomposition of A, $A = QR$ with $Q'Q = I$ and R an upper triangular matrix. Then if $A\pi^T = b$ it must be the case that $QR\pi^T = b$ which implies $R\pi^T = Q'b$, and this can be solved easily because R is triangular.)

```
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        <- qr.solve(A,b)
names(pi) <- c('state.1', 'state.2', 'state.3')
pi
# state.1 state.2 state.3
# 0.54054 0.40541 0.05405
```

We find that:

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

Therefore, under proper conditions, we expect the Markov chain to spend more time in states 1 and 2 as the chain evolves.

The General Approach

Recall that we are attempting to find a solution to

$$\pi = \pi P$$

such that $\sum_i \pi_i = 1$. First we rearrange the expression above to get:

$$\begin{aligned} \pi - \pi P &= 0 \\ \pi(I - P) &= 0 \\ (I - P)^T \pi^T &= 0 \end{aligned}$$

One challenge though is that we need the constrained solution which respects that π describes a probability distribution (i.e. $\sum \pi_i = 1$). Luckily this is a linear constraint that is easily represented by adding another equation to the system. So as a small trick, we need to add a row of all 1's to our $(I - P)^T$ (call this new matrix A) and a 1 to the last element of the zero vector on the right hand side (call this new vector b). Now we want to solve $A\pi = b$ which is over-determined so we solve it as above using `qr.solve`.

The function `stationary` below implements the general approach, and we test it with the worked example above.

```
stationary <- function(transition) {
  stopifnot(is.matrix(transition) &&
        nrow(transition)==ncol(transition) &&
            all(transition>=0 & transition<=1))
  p <- diag(nrow(transition)) - transition
  A <- rbind(t(p),
         rep(1, ncol(transition)))
  b <- c(rep(0, nrow(transition)),
         1)
  res <- qr.solve(A, b)
  names(res) <- paste0("state.", 1:nrow(transition))
  return(res)
}
stationary(matrix(c(0.7, 0.2, 0.1, 0.4, 0.6, 0, 0, 1, 0),
                  nrow=3, byrow=TRUE))
# state.1 state.2 state.3
# 0.54054 0.40541 0.05405
```