

1 Notes on Markov Chains

A Markov chain is a finite-state process that can be characterized completely by two objects – the finite set of states $Z = \{z_1, \dots, z_n\}$ and the transition matrix $\Pi = [\pi_{ij}]_{i,j=1,n}$ that describes the probability of leaving state i and entering state j . Π does not depend on time and must have rows that sum to 1 and are not negative:

$$\begin{aligned}\pi_{ij} &\geq 0 \\ \sum_j \pi_{ij} &= 1 \quad \forall i.\end{aligned}$$

That is, with probability 1 you must move from state i to some state j and transition probabilities are never negative. Π gives the probability of transitioning between states in t and $t + 1$; correspondingly, Π^n gives the probability of transitioning between states in t and $t + n$.

Markov chains that have two properties possess unique invariant distributions. First, the matrix Π must possess one and only one eigenvalue of modulus 1. Second, it cannot possess any absorbing states (a state i^* such that $\pi_{i^*i^*} = 1$). A sufficient condition for the absence of absorbing states is that the matrix Π^n have only non-zero elements for some large enough $n < \infty$; note that this does not require Π to have only non-zero elements; rather, it says that it is always possible to move from state i to state j in a finite number of steps. This condition also rules out the presence of transient states (a state i^* such that $p_{i^*j} = 0 \forall j$) which can never be entered once it is left (transient states are also problematic and we'll just get rid of them too). If these two conditions are satisfied, then there exists a set of probabilities $[\pi^*]_{i=1,n}$ that define the unconditional probability of being in state i ; this vector solves the eigenvalue problem

$$\Pi\pi^* = \pi^*,$$

so that the invariant distribution is the eigenvector associated with the unitary eigenvalue after the eigenvectors are orthonormalized. An alternative method is to compute

$$\Pi^* = \lim_{t \rightarrow \infty} \Pi^t$$

which produces a matrix with identical columns, each of which is the invariant distribution.

Some Markov chains do not have unique invariant distributions. Consider

$$\Pi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix};$$

using this in the iterative approach yields

$$\begin{aligned}\Pi^1 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \Pi^2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \Pi^3 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\end{aligned}$$

and so on. Clearly no limit exists; this is an example of a Markov chain possessing cyclically-moving subsets (sets of realizations that generate a deterministic "path" from one to another). This matrix has two unitary eigenvalues and therefore fails our first condition. Failing the second condition (absorbing states) implies that the invariant distribution will not be unique – any linear combination of absorbing states is itself an invariant distribution.

We will be interested in representing general Markov processes as Markov chains, since they are computationally convenient (they have bounded support). For example, take an AR(1) process

$$z_t = \rho z_{t-1} + \epsilon_t$$

where ϵ is white noise with variance σ^2 . Let $|\rho| < 1$. We need to choose Z and Π in such a way as to best approximate the continuous process for a given n . The unconditional variance of this process is

$$\sigma_z^2 = \frac{\sigma^2}{1 - \rho^2};$$

we therefore choose $z_1 = -m\sigma_z$ and $z_n = m\sigma_z$ in order to cover m standard deviations in both directions. Choosing an evenly-spaced grid is then straightforward:

$$z_i = z_1 + \frac{z_n - z_1}{n - 1} (i - 1).$$

Based on standard probability results, $m = 1$ would cover 67 percent of the support of the unconditional distribution, $m = 2$ would cover 96 percent, $m = 3$ would cover 99 percent, and so on. Keep in mind that there is a tradeoff – for a given number of nodes, accuracy is lower when the range is larger (in the sense that conditional moments may be poorly approximated).

The elements of Π are chosen to match the probability of moving from z_i into the interval

$$\left(z_j - \frac{1}{2} \left(\frac{z_n - z_1}{n - 1} \right), z_j + \frac{1}{2} \left(\frac{z_n - z_1}{n - 1} \right) \right].$$

For the first interval we use

$$\left(-\infty, z_1 + \frac{1}{2} \left(\frac{z_n - z_1}{n - 1} \right) \right]$$

and the for the last we use

$$\left(z_n - \frac{1}{2} \left(\frac{z_n - z_1}{n-1} \right), \infty \right).$$

That is,

$$\begin{aligned} \pi_{ij} &= \Pr \left(\epsilon_{t+1} \leq z_j + \frac{1}{2} \left(\frac{z_n - z_1}{n-1} \right) - \rho z_i \right) - \Pr \left(\epsilon_{t+1} \leq z_j - \frac{1}{2} \left(\frac{z_n - z_1}{n-1} \right) - \rho z_i \right) \\ &= \Phi \left(\frac{z_j + \frac{1}{2} \left(\frac{z_n - z_1}{n-1} \right) - \rho z_i}{\sigma} \right) - \Phi \left(\frac{z_j - \frac{1}{2} \left(\frac{z_n - z_1}{n-1} \right) - \rho z_i}{\sigma} \right) \end{aligned}$$

where Φ is the cdf of the standard normal. Most programming languages provide a built-in function that evaluates Φ , and others are available on the web. Obviously we can accomodate other density functions as well.

A better choice for the grid would be to use the zeros from an n th-order Hermite polynomial, since these polynomials are used to numerically integrate normal random variables. That is, the approximation becomes a version of **Gauss-Hermite quadrature**, which is particularly useful for computing expectations of normal random variables because the weighting function is proportional to the density of the standard normal. The quadrature rule is given by

$$\int_{-\infty}^{\infty} f(x) e^{-x^2} dx = \sum_{i=1}^N \omega_i f(x_i) + \frac{n! \sqrt{\pi}}{2^n} \frac{f^{(2n)}(\xi)}{(2n)!};$$

the weights and nodes for some values of N are given in the following table (the nodes are $\pm x_i$ with the same weights on both) and the rest can be obtained using any number of publicly-available routines:

Table 1
Gauss-Hermite Quadrature

n	x_i	ω_i	n	x_i	ω_i
2	0.7071067811	0.8862269254	7	2.651961356	0.000971781245
				1.673551628	0.05451558281
3	1.224744871	0.2954089751	10	0.8162878828	0.4256072526
				0.0000000000	0.8102646175
				1.1816359	0.0000076404
4	1.650680123	0.08131283544	10	2.532731674	0.001343645746
				0.5246476232	0.03387439445
5	2.020182870	0.01995324205	10	1.036610829	0.2401386110
				0.9585724646	0.3429013272
				0.0000000000	0.6108626337
5	0.9585724646	0.3936193231	10	0.3429013272	0.6108626337
				0.0000000000	0.9453087204

To apply Gauss-Hermite quadrature to expectations of functions of nonstandard normal random variables, we simply use the linear change of variables

$$x = \frac{y - \mu}{\sigma \sqrt{2}}$$

and compute

$$\begin{aligned} E[f(Y)] &= \int_{-\infty}^{\infty} f(y) e^{-(y-\mu)^2/(2\sigma^2)} dy \\ &= \frac{1}{\sqrt{\pi}} \sum_{i=1}^n \omega_i f(\sigma\sqrt{2}x_i + \mu). \end{aligned}$$

Unfortunately, we cannot blindly use Gauss-Hermite quadrature, because the nodes used would be conditional on the current value and this dependence would propagate forward and make the number of states very large (in fact, they would nearly fill the real line). To fix this we renormalize by conditioning on the current nodes. The expectation becomes

$$E[v(z') | z = z_i] = \sum_{j=1}^n \bar{\lambda}_{i,j} v(z'_j)$$

where

$$\begin{aligned} z'_j &= \sqrt{2}\sigma^2 x_j + (1 - \rho)\bar{z} + \rho z_i \\ \lambda_{i,j} &= \frac{1}{\sqrt{\pi}} \omega_i \frac{\phi(z'_j | z_i)}{\phi(z'_j | \bar{z})} \\ \bar{\lambda}_{i,j} &= \frac{\lambda_{i,j}}{\sum_j \lambda_{i,j}} \end{aligned}$$

and x_i and ω_i are the nodes and weights for the n th order Hermite polynomial and ϕ is the standard normal pdf. Essentially we create the transition matrix as

$$\Pi = [\bar{\lambda}_{i,j}]_{i,j=1}^n$$

and the realizations are the z_i . The benefit of this approach is that it provides a better approximation in the tails of the distribution, since the nodes are concentrated there. The Tauchen-Hussey programs, available on the web in Fortran and Matlab, implement this procedure for vector processes.

For extremely persistent processes, Floden recommends using

$$\begin{aligned} \sigma_b &= \omega\sigma^2 + (1 - \omega)\sigma_z^2 \\ \omega &= 0.5 + 0.25\rho \end{aligned}$$

as the "variance" that goes into the Markov chain approximation. A Matlab program to implement this approximation procedure is available at Martin's website.

Taking expectations of Markov chains is trivial:

$$E[v(z') | z = z_i] = \sum_j \pi_{ij} v(z_j).$$

Other moments are straightforward to compute as well, such as the variance:

$$\text{var}(v(z') | z = z_i) = \sum_j \pi_{ij} v(z_j)^2 - \left(\sum_j \pi_{ij} v(z_j) \right)^2.$$

One point to note is that a symmetric Markov chain with an even number of nodes will not have its mean as a node – I suggest you use n odd in these cases. Another point to remember is that normals are negative half the time; to use a log-normal, approximate the normal and take $\exp(z)$ for each node. Remember that the mean of a log-normal is $\mu + \frac{1}{2}\sigma^2$, so adjust the mean of the underlying normal appropriately.