Finite Markov Chains

Thomas J. Sargent and John Stachurski

October 5, 2020

1 Contents

In addition to what's in Anaconda, this lecture will need the following libraries:

```
In [1]: !conda install -y quantecon
```

2 Overview

Markov chains are one of the most useful classes of stochastic processes, being

- simple, flexible and supported by many elegant theoretical results
- valuable for building intuition about random dynamic models
- central to quantitative modeling in their own right

You will find them in many of the workhorse models of economics and finance.

In this lecture, we review some of the theory of Markov chains.

We will also introduce some of the high-quality routines for working with Markov chains available in QuantEcon.py.

Prerequisite knowledge is basic probability and linear algebra.

Let's start with some standard imports:

```
In [2]: import quantecon as qe
        import numpy as np
        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt
        %matplotlib inline
```

3 Definitions

The following concepts are fundamental.

3.1 Stochastic Matrices

A **stochastic matrix** (or **Markov matrix**) is an $n \times n$ square matrix P such that

1. each element of P is nonnegative, and
2. each row of P sums to one

Each row of P can be regarded as a probability mass function over n possible outcomes.

It is too not difficult to check Section **??** that if P is a stochastic matrix, then so is the k-th power P^k for all $k \in \mathbb{N}$.

3.2 Markov Chains

There is a close connection between stochastic matrices and Markov chains.

To begin, let S be a finite set with n elements $\{x_1, \dots, x_n\}$.

The set S is called the **state space** and $x_1, \dots, x_n$ are the **state values**.

A **Markov chain** $\{X_t\}$ on S is a sequence of random variables on S that have the **Markov property**.

This means that, for any date t and any state $y \in S$,

$$\mathbb{P}\{X_{t+1} = y \,|\, X_t\} = \mathbb{P}\{X_{t+1} = y \,|\, X_t, X_{t-1}, \ldots\} \tag{1}$$

In other words, knowing the current state is enough to know probabilities for future states.

In particular, the dynamics of a Markov chain are fully determined by the set of values

$$P(x, y) := \mathbb{P}\{X_{t+1} = y \,|\, X_t = x\} \qquad (x, y \in S) \tag{2}$$

By construction,

- $P(x, y)$ is the probability of going from x to y in one unit of time (one step)
- $P(x, \cdot)$ is the conditional distribution of X_{t+1} given $X_t = x$

We can view P as a stochastic matrix where

$$P_{ij} = P(x_i, x_j) \qquad 1 \le i, j \le n$$

Going the other way, if we take a stochastic matrix P, we can generate a Markov chain $\{X_t\}$ as follows:

- draw X_0 from some specified distribution
- for each $t = 0, 1, \ldots$, draw X_{t+1} from $P(X_t, \cdot)$

By construction, the resulting process satisfies (2).

3.3 Example 1

Consider a worker who, at any given time t, is either unemployed (state 0) or employed (state 1).

Suppose that, over a one month period,

1. An unemployed worker finds a job with probability $\alpha \in (0, 1)$.
2. An employed worker loses her job and becomes unemployed with probability $\beta \in (0, 1)$.

In terms of a Markov model, we have

- $S = \{0, 1\}$
- $P(0, 1) = \alpha$ and $P(1, 0) = \beta$

We can write out the transition probabilities in matrix form as

$$P = \begin{pmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{pmatrix} \tag{3}$$

Once we have the values α and β, we can address a range of questions, such as

- What is the average duration of unemployment?
- Over the long-run, what fraction of time does a worker find herself unemployed?
- Conditional on employment, what is the probability of becoming unemployed at least once over the next 12 months?

We'll cover such applications below.

3.4 Example 2

Using US unemployment data, Hamilton [3] estimated the stochastic matrix

$$P = \begin{pmatrix} 0.971 & 0.029 & 0 \\ 0.145 & 0.778 & 0.077 \\ 0 & 0.508 & 0.492 \end{pmatrix}$$

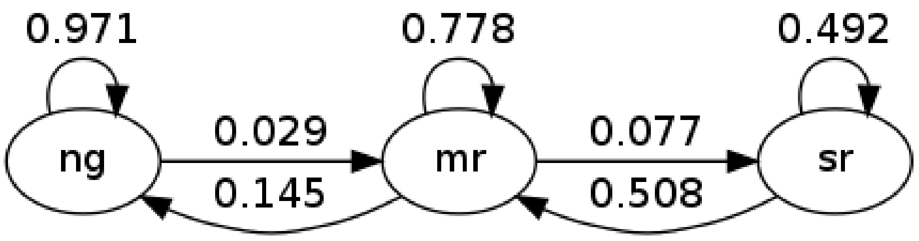
where

- the frequency is monthly
- the first state represents "normal growth"
- the second state represents "mild recession"
- the third state represents "severe recession"

For example, the matrix tells us that when the state is normal growth, the state will again be normal growth next month with probability 0.97.

In general, large values on the main diagonal indicate persistence in the process $\{X_t\}$.

This Markov process can also be represented as a directed graph, with edges labeled by transition probabilities



Here “ng” is normal growth, “mr” is mild recession, etc.

4 Simulation

One natural way to answer questions about Markov chains is to simulate them.

(To approximate the probability of event E, we can simulate many times and count the fraction of times that E occurs).

Nice functionality for simulating Markov chains exists in QuantEcon.py.

- Efficient, bundled with lots of other useful routines for handling Markov chains.

However, it’s also a good exercise to roll our own routines — let’s do that first and then come back to the methods in QuantEcon.py.

In these exercises, we’ll take the state space to be $S = 0, \ldots, n-1$.

4.1 Rolling Our Own

To simulate a Markov chain, we need its stochastic matrix P and a probability distribution ψ for the initial state to be drawn from.

The Markov chain is then constructed as discussed above. To repeat:

1. At time $t = 0$, the X_0 is chosen from ψ.
2. At each subsequent time t, the new state X_{t+1} is drawn from $P(X_t, \cdot)$.

To implement this simulation procedure, we need a method for generating draws from a discrete distribution.

For this task, we’ll use `random.draw` from QuantEcon, which works as follows:

```
In [3]: ψ = (0.3, 0.7)           # probabilities over {0, 1}
        cdf = np.cumsum(ψ)       # convert into cummulative distribution
        qe.random.draw(cdf, 5)   # generate 5 independent draws from ψ
```

```
Out[3]: array([1, 1, 1, 1, 0])
```

We’ll write our code as a function that takes the following three arguments

- A stochastic matrix `P`

- An initial state `init`
- A positive integer `sample_size` representing the length of the time series the function should return

```
In [4]: def mc_sample_path(P, ψ_0=None, sample_size=1_000):

            # set up
            P = np.asarray(P)
            X = np.empty(sample_size, dtype=int)

            # Convert each row of P into a cdf
            n = len(P)
            P_dist = [np.cumsum(P[i, :]) for i in range(n)]

            # draw initial state, defaulting to 0
            if ψ_0 is not None:
                X_0 = qe.random.draw(np.cumsum(ψ_0))
            else:
                X_0 = 0

            # simulate
            X[0] = X_0
            for t in range(sample_size - 1):
                X[t+1] = qe.random.draw(P_dist[X[t]])

            return X
```

Let's see how it works using the small matrix

```
In [5]: P = [[0.4, 0.6],
             [0.2, 0.8]]
```

As we'll see later, for a long series drawn from `P`, the fraction of the sample that takes value 0 will be about 0.25.

Moreover, this is true, regardless of the initial distribution from with X_0 is drawn.

The following code illustrates this

```
In [6]: X = mc_sample_path(P, ψ_0=[0.1, 0.9], sample_size=100_000)
        np.mean(X == 0)
```

```
Out[6]: 0.25051
```

You can try changing the initial distribution to confirm that the output is always close to 0.25.

4.2 Using QuantEcon's Routines

As discussed above, QuantEcon.py has routines for handling Markov chains, including simulation.

Here's an illustration using the same P as the preceding example

```
In [7]: from quantecon import MarkovChain

        mc = qe.MarkovChain(P)
        X = mc.simulate(ts_length=1_000_000)
        np.mean(X == 0)
```

```
Out[7]: 0.249567
```

The QuantEcon.py routine is JIT compiled and much faster.

```
In [8]: %time mc_sample_path(P, sample_size=1_000_000) # Our version
```

```
    CPU times: user 1.51 s, sys: 1.44 ms, total: 1.51 s
Wall time: 1.51 s
```

```
Out[8]: array([0, 1, 1, …, 1, 1, 1])
```

```
In [9]: %time mc.simulate(ts_length=1_000_000) # qe version
```

```
    CPU times: user 42.3 ms, sys: 0 ns, total: 42.3 ms
Wall time: 42 ms
```

```
Out[9]: array([0, 0, 0, …, 0, 1, 1])
```

4.2.1 Adding State Values and Initial Conditions

If we wish to, we can provide a specification of state values to `MarkovChain`.

These state values can be integers, floats, or even strings.

The following code illustrates

```
In [10]: mc = qe.MarkovChain(P, state_values=('unemployed', 'employed'))
         mc.simulate(ts_length=4, init='employed')
```

```
Out[10]: array(['employed', 'employed', 'employed', 'employed'], dtype='<U10')
```

```
In [11]: mc.simulate(ts_length=4, init='unemployed')
```

```
Out[11]: array(['unemployed', 'employed', 'unemployed', 'unemployed'],
  ↪dtype='<U10')
```

```
In [12]: mc.simulate(ts_length=4)  # Start at randomly chosen initial state
```

```
Out[12]: array(['employed', 'employed', 'unemployed', 'unemployed'], dtype='<U10')
```

If we want to simulate with output as indices rather than state values we can use

```
In [13]: mc.simulate_indices(ts_length=4)
```

```
Out[13]: array([1, 1, 1, 1])
```

5 Marginal Distributions

Suppose that

1. $\{X_t\}$ is a Markov chain with stochastic matrix P
2. the distribution of X_t is known to be ψ_t

What then is the distribution of X_{t+1}, or, more generally, of X_{t+m}?

To answer this, we let ψ_t be the distribution of X_t for $t = 0, 1, 2, \ldots$.

Our first aim is to find ψ_{t+1} given ψ_t and P.

To begin, pick any $y \in S$.

Using the law of total probability, we can decompose the probability that $X_{t+1} = y$ as follows:

$$\mathbb{P}\{X_{t+1} = y\} = \sum_{x \in S} \mathbb{P}\{X_{t+1} = y \,|\, X_t = x\} \cdot \mathbb{P}\{X_t = x\}$$

In words, to get the probability of being at y tomorrow, we account for all ways this can happen and sum their probabilities.

Rewriting this statement in terms of marginal and conditional probabilities gives

$$\psi_{t+1}(y) = \sum_{x \in S} P(x, y) \psi_t(x)$$

There are n such equations, one for each $y \in S$.

If we think of ψ_{t+1} and ψ_t as *row vectors* (as is traditional in this literature), these n equations are summarized by the matrix expression

$$\psi_{t+1} = \psi_t P \tag{4}$$

In other words, to move the distribution forward one unit of time, we postmultiply by P.

By repeating this m times we move forward m steps into the future.

Hence, iterating on (4), the expression $\psi_{t+m} = \psi_t P^m$ is also valid — here P^m is the m-th power of P.

As a special case, we see that if ψ_0 is the initial distribution from which X_0 is drawn, then $\psi_0 P^m$ is the distribution of X_m.

This is very important, so let's repeat it

$$X_0 \sim \psi_0 \quad \implies \quad X_m \sim \psi_0 P^m \tag{5}$$

and, more generally,

$$X_t \sim \psi_t \quad \implies \quad X_{t+m} \sim \psi_t P^m \tag{6}$$

5.1 Multiple Step Transition Probabilities

We know that the probability of transitioning from x to y in one step is $P(x, y)$.

It turns out that the probability of transitioning from x to y in m steps is $P^m(x, y)$, the (x, y)-th element of the m-th power of P.

To see why, consider again (6), but now with ψ_t putting all probability on state x

- 1 in the x-th position and zero elsewhere

Inserting this into (6), we see that, conditional on $X_t = x$, the distribution of X_{t+m} is the x-th row of P^m.

In particular

$$\mathbb{P}\{X_{t+m} = y \,|\, X_t = x\} = P^m(x, y) = (x, y)\text{-th element of } P^m$$

5.2 Example: Probability of Recession

Recall the stochastic matrix P for recession and growth considered above.

Suppose that the current state is unknown — perhaps statistics are available only at the *end* of the current month.

We estimate the probability that the economy is in state x to be $\psi(x)$.

The probability of being in recession (either mild or severe) in 6 months time is given by the inner product

$$\psi P^6 \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

5.3 Example 2: Cross-Sectional Distributions

The marginal distributions we have been studying can be viewed either as probabilities or as cross-sectional frequencies in large samples.

To illustrate, recall our model of employment/unemployment dynamics for a given worker discussed above.

Consider a large population of workers, each of whose lifetime experience is described by the specified dynamics, independent of one another.

Let ψ be the current *cross-sectional* distribution over $\{0, 1\}$.

The cross-sectional distribution records the fractions of workers employed and unemployed at a given moment.

- For example, $\psi(0)$ is the unemployment rate.

What will the cross-sectional distribution be in 10 periods hence?

The answer is ψP^{10}, where P is the stochastic matrix in (3).

This is because each worker is updated according to P, so ψP^{10} represents probabilities for a single randomly selected worker.

But when the sample is large, outcomes and probabilities are roughly equal (by the Law of Large Numbers).

So for a very large (tending to infinite) population, ψP^{10} also represents the fraction of workers in each state.

This is exactly the cross-sectional distribution.

6 Irreducibility and Aperiodicity

Irreducibility and aperiodicity are central concepts of modern Markov chain theory.

Let's see what they're about.

6.1 Irreducibility

Let P be a fixed stochastic matrix.

Two states x and y are said to **communicate** with each other if there exist positive integers j and k such that

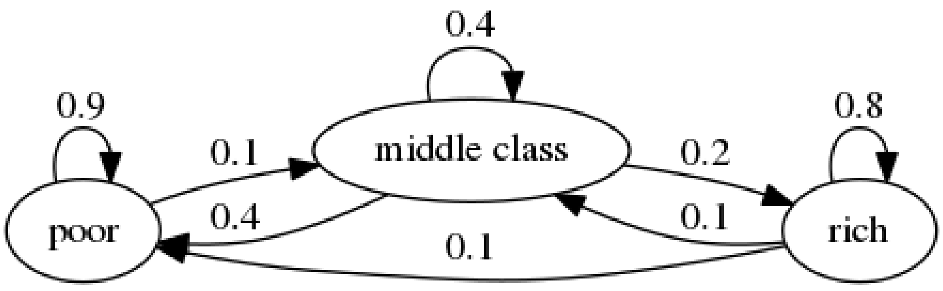
$$P^j(x,y) > 0 \quad \text{and} \quad P^k(y,x) > 0$$

In view of our discussion above, this means precisely that

- state x can be reached eventually from state y, and
- state y can be reached eventually from state x

The stochastic matrix P is called **irreducible** if all states communicate; that is, if x and y communicate for all (x, y) in $S \times S$.

For example, consider the following transition probabilities for wealth of a fictitious set of households



We can translate this into a stochastic matrix, putting zeros where there's no edge between nodes

$$P := \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0.4 & 0.4 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

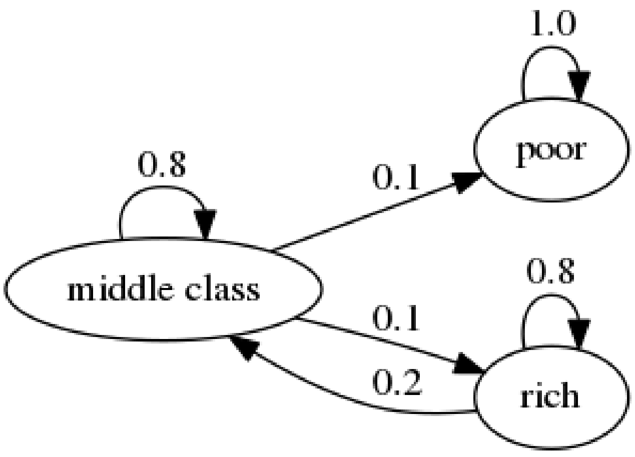
It's clear from the graph that this stochastic matrix is irreducible: we can reach any state from any other state eventually.

We can also test this using QuantEcon.py's MarkovChain class

```
In [14]: P = [[0.9, 0.1, 0.0],
              [0.4, 0.4, 0.2],
              [0.1, 0.1, 0.8]]

         mc = qe.MarkovChain(P, ('poor', 'middle', 'rich'))
         mc.is_irreducible
```

```
Out[14]: True
```

Here's a more pessimistic scenario, where the poor are poor forever



This stochastic matrix is not irreducible, since, for example, rich is not accessible from poor.

Let's confirm this

```
In [15]: P = [[1.0, 0.0, 0.0],
              [0.1, 0.8, 0.1],
              [0.0, 0.2, 0.8]]

         mc = qe.MarkovChain(P, ('poor', 'middle', 'rich'))
         mc.is_irreducible
```

```
Out[15]: False
```

We can also determine the "communication classes"

```
In [16]: mc.communication_classes
```

```
Out[16]: [array(['poor'], dtype='<U6'), array(['middle', 'rich'], dtype='<U6')]
```

It might be clear to you already that irreducibility is going to be important in terms of long run outcomes.

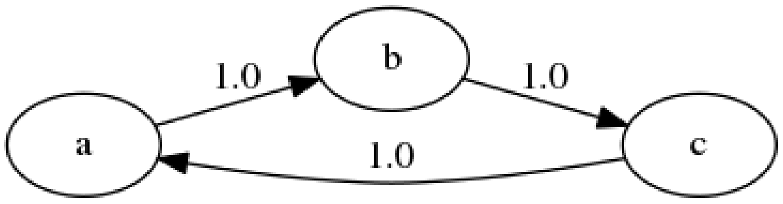
For example, poverty is a life sentence in the second graph but not the first.

We'll come back to this a bit later.

6.2 Aperiodicity

Loosely speaking, a Markov chain is called periodic if it cycles in a predictible way, and aperiodic otherwise.

Here's a trivial example with three states



The chain cycles with period 3:

```
In [17]: P = [[0, 1, 0],
               [0, 0, 1],
               [1, 0, 0]]

          mc = qe.MarkovChain(P)
          mc.period
```

```
Out[17]: 3
```

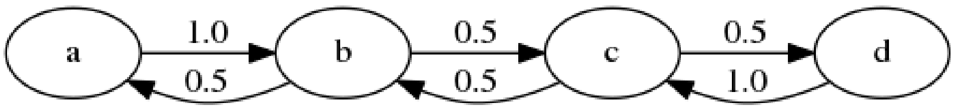
More formally, the **period** of a state x is the greatest common divisor of the set of integers

$$D(x) := \{j \geq 1 : P^j(x, x) > 0\}$$

In the last example, $D(x) = \{3, 6, 9, ...\}$ for every state x, so the period is 3.

A stochastic matrix is called **aperiodic** if the period of every state is 1, and **periodic** otherwise.

For example, the stochastic matrix associated with the transition probabilities below is periodic because, for example, state a has period 2



We can confirm that the stochastic matrix is periodic as follows

```
In [18]: P = [[0.0, 1.0, 0.0, 0.0],
               [0.5, 0.0, 0.5, 0.0],
               [0.0, 0.5, 0.0, 0.5],
               [0.0, 0.0, 1.0, 0.0]]

          mc = qe.MarkovChain(P)
          mc.period
```

```
Out[18]: 2
```

```
In [19]: mc.is_aperiodic
```

```
Out[19]: False
```

7 Stationary Distributions

As seen in (4), we can shift probabilities forward one unit of time via postmultiplication by P.

Some distributions are invariant under this updating process — for example,

```
In [20]: P = np.array([[0.4, 0.6],
                       [0.2, 0.8]])
         ψ = (0.25, 0.75)
         ψ @ P
```

```
Out[20]: array([0.25, 0.75])
```

Such distributions are called **stationary**, or **invariant**.

Formally, a distribution ψ^* on S is called **stationary** for P if $\psi^* = \psi^* P$.

(This is the same notion of stationarity that we learned about in the lecture on AR(1) processes applied to a different setting.)

From this equality, we immediately get $\psi^* = \psi^* P^t$ for all t.

This tells us an important fact: If the distribution of X_0 is a stationary distribution, then X_t will have this same distribution for all t.

Hence stationary distributions have a natural interpretation as stochastic steady states — we'll discuss this more in just a moment.

Mathematically, a stationary distribution is a fixed point of P when P is thought of as the map $\psi \mapsto \psi P$ from (row) vectors to (row) vectors.

Theorem. Every stochastic matrix P has at least one stationary distribution.

(We are assuming here that the state space S is finite; if not more assumptions are required)

For proof of this result, you can apply Brouwer's fixed point theorem, or see EDTC, theorem 4.3.5.

There may in fact be many stationary distributions corresponding to a given stochastic matrix P.

- For example, if P is the identity matrix, then all distributions are stationary.

Since stationary distributions are long run equilibria, to get uniqueness we require that initial conditions are not infinitely persistent.

Infinite persistence of initial conditions occurs if certain regions of the state space cannot be accessed from other regions, which is the opposite of irreducibility.

This gives some intuition for the following fundamental theorem.

Theorem. If P is both aperiodic and irreducible, then

1. P has exactly one stationary distribution ψ^*.
2. For any initial distribution ψ_0, we have $\|\psi_0 P^t - \psi^*\| \to 0$ as $t \to \infty$.

For a proof, see, for example, theorem 5.2 of [2].

(Note that part 1 of the theorem requires only irreducibility, whereas part 2 requires both irreducibility and aperiodicity)

A stochastic matrix satisfying the conditions of the theorem is sometimes called **uniformly ergodic**.

One easy sufficient condition for aperiodicity and irreducibility is that every element of P is strictly positive.

- Try to convince yourself of this.

7.1 Example

Recall our model of employment/unemployment dynamics for a given worker discussed above.

Assuming $\alpha \in (0, 1)$ and $\beta \in (0, 1)$, the uniform ergodicity condition is satisfied.

Let $\psi^* = (p, 1 - p)$ be the stationary distribution, so that p corresponds to unemployment (state 0).

Using $\psi^* = \psi^* P$ and a bit of algebra yields

$$p = \frac{\beta}{\alpha + \beta}$$

This is, in some sense, a steady state probability of unemployment — more on interpretation below.

Not surprisingly it tends to zero as $\beta \to 0$, and to one as $\alpha \to 0$.

7.2 Calculating Stationary Distributions

As discussed above, a given Markov matrix P can have many stationary distributions.

That is, there can be many row vectors ψ such that $\psi = \psi P$.

In fact if P has two distinct stationary distributions ψ_1, ψ_2 then it has infinitely many, since in this case, as you can verify,

$$\psi_3 := \lambda \psi_1 + (1 - \lambda) \psi_2$$

is a stationary distribution for P for any $\lambda \in [0, 1]$.

If we restrict attention to the case where only one stationary distribution exists, one option for finding it is to try to solve the linear system $\psi(I_n - P) = 0$ for ψ, where I_n is the $n \times n$ identity.

But the zero vector solves this equation, so we need to proceed carefully.

In essence, we need to impose the restriction that the solution must be a probability distribution.

There are various ways to do this.

One option is to regard this as an eigenvector problem: a vector ψ such that $\psi = \psi P$ is a left eigenvector associated with the unit eigenvalue $\lambda = 1$.

A more stable and sophisticated algorithm is implemented in QuantEcon.py.

This is the one we recommend you use:

```
In [21]: P = [[0.4, 0.6],
              [0.2, 0.8]]

         mc = qe.MarkovChain(P)
         mc.stationary_distributions  # Show all stationary distributions
```

```
Out[21]: array([[0.25, 0.75]])
```

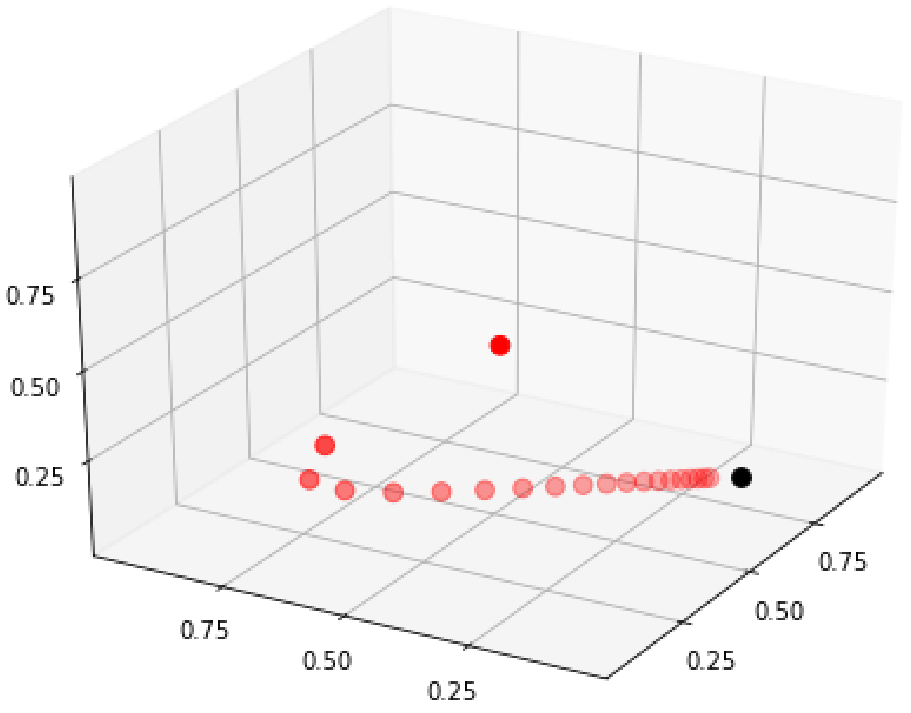
7.3 Convergence to Stationarity

Part 2 of the Markov chain convergence theorem stated above tells us that the distribution of X_t converges to the stationary distribution regardless of where we start off.

This adds considerable weight to our interpretation of ψ^* as a stochastic steady state.

The convergence in the theorem is illustrated in the next figure

```
In [22]: P = ((0.971, 0.029, 0.000),
              (0.145, 0.778, 0.077),
              (0.000, 0.508, 0.492))
         P = np.array(P)

         ψ = (0.0, 0.2, 0.8)        # Initial condition

         fig = plt.figure(figsize=(8, 6))
         ax = fig.add_subplot(111, projection='3d')

         ax.set(xlim=(0, 1), ylim=(0, 1), zlim=(0, 1),
                xticks=(0.25, 0.5, 0.75),
                yticks=(0.25, 0.5, 0.75),
                zticks=(0.25, 0.5, 0.75))

         x_vals, y_vals, z_vals = [], [], []
         for t in range(20):
             x_vals.append(ψ[0])
             y_vals.append(ψ[1])
             z_vals.append(ψ[2])
             ψ = ψ @ P

         ax.scatter(x_vals, y_vals, z_vals, c='r', s=60)
         ax.view_init(30, 210)

         mc = qe.MarkovChain(P)
         ψ_star = mc.stationary_distributions[0]
         ax.scatter(ψ_star[0], ψ_star[1], ψ_star[2], c='k', s=60)

         plt.show()
```



Here

- P is the stochastic matrix for recession and growth considered above.
- The highest red dot is an arbitrarily chosen initial probability distribution ψ, represented as a vector in $\mathbb{R}^3$.
- The other red dots are the distributions ψP^t for $t = 1, 2, \ldots$.
- The black dot is ψ^*.

You might like to try experimenting with different initial conditions.

8 Ergodicity

Under irreducibility, yet another important result obtains: For all $x \in S$,

$$\frac{1}{m} \sum_{t=1}^{m} \mathbf{1}\{X_t = x\} \to \psi^*(x) \quad \text{as } m \to \infty \tag{7}$$

Here

- $\mathbf{1}\{X_t = x\} = 1$ if $X_t = x$ and zero otherwise
- convergence is with probability one
- the result does not depend on the distribution (or value) of X_0

The result tells us that the fraction of time the chain spends at state x converges to $\psi^*(x)$ as time goes to infinity.

This gives us another way to interpret the stationary distribution — provided that the convergence result in (7) is valid.

The convergence in (7) is a special case of a law of large numbers result for Markov chains — see EDTC, section 4.3.4 for some additional information.

8.1 Example

Recall our cross-sectional interpretation of the employment/unemployment model discussed above.

Assume that $\alpha \in (0, 1)$ and $\beta \in (0, 1)$, so that irreducibility and aperiodicity both hold.

We saw that the stationary distribution is $(p, 1 - p)$, where

$$p = \frac{\beta}{\alpha + \beta}$$

In the cross-sectional interpretation, this is the fraction of people unemployed.

In view of our latest (ergodicity) result, it is also the fraction of time that a worker can expect to spend unemployed.

Thus, in the long-run, cross-sectional averages for a population and time-series averages for a given person coincide.

This is one interpretation of the notion of ergodicity.

9 Computing Expectations

We are interested in computing expectations of the form

$$\mathbb{E}[h(X_t)] \tag{8}$$

and conditional expectations such as

$$\mathbb{E}[h(X_{t+k}) \mid X_t = x] \tag{9}$$

where

- $\{X_t\}$ is a Markov chain generated by $n \times n$ stochastic matrix P
- h is a given function, which, in expressions involving matrix algebra, we'll think of as the column vector

$$h = \begin{pmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{pmatrix}$$

The unconditional expectation (8) is easy: We just sum over the distribution of X_t to get

$$\mathbb{E}[h(X_t)] = \sum_{x \in S} (\psi P^t)(x) h(x)$$

Here ψ is the distribution of X_0.

Since ψ and hence ψP^t are row vectors, we can also write this as

$$\mathbb{E}[h(X_t)] = \psi P^t h$$

For the conditional expectation (9), we need to sum over the conditional distribution of X_{t+k} given $X_t = x$.

We already know that this is $P^k(x, \cdot)$, so

$$\mathbb{E}[h(X_{t+k}) \mid X_t = x] = (P^k h)(x) \tag{10}$$

The vector $P^k h$ stores the conditional expectation $\mathbb{E}[h(X_{t+k}) \mid X_t = x]$ over all x.

9.1 Expectations of Geometric Sums

Sometimes we also want to compute expectations of a geometric sum, such as $\sum_t \beta^t h(X_t)$.

In view of the preceding discussion, this is

$$\mathbb{E}\left[\sum_{j=0}^{\infty} \beta^j h(X_{t+j}) \mid X_t = x\right] = [(I - \beta P)^{-1} h](x)$$

where

$$(I - \beta P)^{-1} = I + \beta P + \beta^2 P^2 + \cdots$$

Premultiplication by $(I - \beta P)^{-1}$ amounts to "applying the **resolvent operator**".

10 Exercises

10.1 Exercise 1

According to the discussion above, if a worker's employment dynamics obey the stochastic matrix

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

with $\alpha \in (0,1)$ and $\beta \in (0,1)$, then, in the long-run, the fraction of time spent unemployed will be

$$p := \frac{\beta}{\alpha + \beta}$$

In other words, if $\{X_t\}$ represents the Markov chain for employment, then $\bar{X}_m \to p$ as $m \to \infty$, where

$$\bar{X}_m := \frac{1}{m} \sum_{t=1}^{m} \mathbf{1}\{X_t = 0\}$$

The exercise is to illustrate this convergence by computing $\bar{X}_m$ for large m and checking that it is close to p.

You will see that this statement is true regardless of the choice of initial condition or the values of α, β, provided both lie in $(0, 1)$.

10.2 Exercise 2

A topic of interest for economics and many other disciplines is *ranking*.

Let's now consider one of the most practical and important ranking problems — the rank assigned to web pages by search engines.

(Although the problem is motivated from outside of economics, there is in fact a deep connection between search ranking systems and prices in certain competitive equilibria — see [1])

To understand the issue, consider the set of results returned by a query to a web search engine.

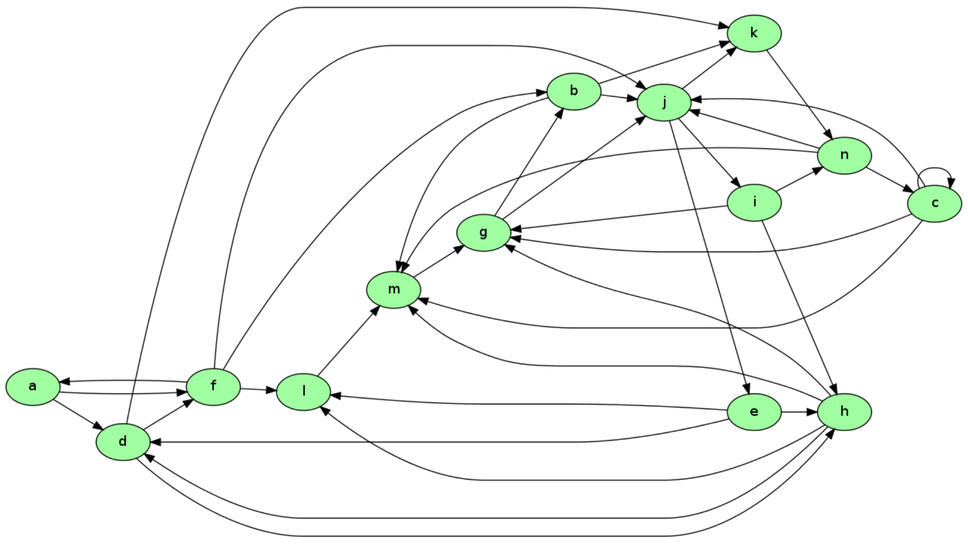
For the user, it is desirable to

1. receive a large set of accurate matches
2. have the matches returned in order, where the order corresponds to some measure of "importance"

Ranking according to a measure of importance is the problem we now consider.

The methodology developed to solve this problem by Google founders Larry Page and Sergey Brin is known as PageRank.

To illustrate the idea, consider the following diagram



Imagine that this is a miniature version of the WWW, with

- each node representing a web page
- each arrow representing the existence of a link from one page to another

Now let's think about which pages are likely to be important, in the sense of being valuable to a search engine user.

One possible criterion for the importance of a page is the number of inbound links — an indication of popularity.

By this measure, `m` and `j` are the most important pages, with 5 inbound links each.

However, what if the pages linking to `m`, say, are not themselves important?

Thinking this way, it seems appropriate to weight the inbound nodes by relative importance.

The PageRank algorithm does precisely this.

A slightly simplified presentation that captures the basic idea is as follows.

Letting j be (the integer index of) a typical page and r_j be its ranking, we set

$$r_j = \sum_{i \in L_j} \frac{r_i}{\ell_i}$$

where

- ℓ_i is the total number of outbound links from i
- L_j is the set of all pages i such that i has a link to j

This is a measure of the number of inbound links, weighted by their own ranking (and normalized by $1/\ell_i$).

There is, however, another interpretation, and it brings us back to Markov chains.

Let P be the matrix given by $P(i,j) = \mathbf{1}\{i \to j\}/\ell_i$ where $\mathbf{1}\{i \to j\} = 1$ if i has a link to j and zero otherwise.

The matrix P is a stochastic matrix provided that each page has at least one link.

With this definition of P we have

$$r_j = \sum_{i \in L_j} \frac{r_i}{\ell_i} = \sum_{\text{all } i} \mathbf{1}\{i \to j\} \frac{r_i}{\ell_i} = \sum_{\text{all } i} P(i,j) r_i$$

Writing r for the row vector of rankings, this becomes $r = rP$.

Hence r is the stationary distribution of the stochastic matrix P.

Let's think of $P(i,j)$ as the probability of "moving" from page i to page j.

The value $P(i,j)$ has the interpretation

- $P(i,j) = 1/k$ if i has k outbound links and j is one of them
- $P(i,j) = 0$ if i has no direct link to j

Thus, motion from page to page is that of a web surfer who moves from one page to another by randomly clicking on one of the links on that page.

Here "random" means that each link is selected with equal probability.

Since r is the stationary distribution of P, assuming that the uniform ergodicity condition is valid, we can interpret r_j as the fraction of time that a (very persistent) random surfer spends at page j.

Your exercise is to apply this ranking algorithm to the graph pictured above and return the list of pages ordered by rank.

There is a total of 14 nodes (i.e., web pages), the first named `a` and the last named `n`.

A typical line from the file has the form

```
d -> h;
```

This should be interpreted as meaning that there exists a link from `d` to `h`.

The data for this graph is shown below, and read into a file called `web_graph_data.txt` when the cell is executed.

```
In [23]: %%file web_graph_data.txt
         a -> d;
         a -> f;
         b -> j;
         b -> k;
         b -> m;
         c -> c;
         c -> g;
         c -> j;
         c -> m;
         d -> f;
         d -> h;
         d -> k;
         e -> d;
         e -> h;
         e -> l;
         f -> a;
         f -> b;
         f -> j;
         f -> l;
         g -> b;
         g -> j;
         h -> d;
         h -> g;
         h -> l;
         h -> m;
         i -> g;
         i -> h;
         i -> n;
         j -> e;
         j -> i;
         j -> k;
         k -> n;
         l -> m;
         m -> g;
         n -> c;
         n -> j;
         n -> m;
```

```
Overwriting web_graph_data.txt
```

To parse this file and extract the relevant information, you can use regular expressions.

The following code snippet provides a hint as to how you can go about this

```
In [24]: import re
         re.findall('\w', 'x +++ y ****** z')  # \w matches alphanumerics
```

```
Out[24]: ['x', 'y', 'z']
```

```
In [25]: re.findall('\w', 'a ^^ b &&& $$ c')
```

```
Out[25]: ['a', 'b', 'c']
```

When you solve for the ranking, you will find that the highest ranked node is in fact `g`, while the lowest is `a`.

10.3 Exercise 3

In numerical work, it is sometimes convenient to replace a continuous model with a discrete one.

In particular, Markov chains are routinely generated as discrete approximations to AR(1) processes of the form

$$y_{t+1} = \rho y_t + u_{t+1}$$

Here u_t is assumed to be IID and $N(0, \sigma_u^2)$.

The variance of the stationary probability distribution of $\{y_t\}$ is

$$\sigma_y^2 := \frac{\sigma_u^2}{1 - \rho^2}$$

Tauchen's method [4] is the most common method for approximating this continuous state process with a finite state Markov chain.

A routine for this already exists in QuantEcon.py but let's write our own version as an exercise.

As a first step, we choose

- n, the number of states for the discrete approximation
- m, an integer that parameterizes the width of the state space

Next, we create a state space $\{x_0, \ldots, x_{n-1}\} \subset \mathbb{R}$ and a stochastic $n \times n$ matrix P such that

- $x_0 = -m \, \sigma_y$
- $x_{n-1} = m \, \sigma_y$
- $x_{i+1} = x_i + s$ where $s = (x_{n-1} - x_0)/(n - 1)$

Let F be the cumulative distribution function of the normal distribution $N(0, \sigma_u^2)$.

The values $P(x_i, x_j)$ are computed to approximate the AR(1) process — omitting the derivation, the rules are as follows:

1. If $j = 0$, then set

$$P(x_i, x_j) = P(x_i, x_0) = F(x_0 - \rho x_i + s/2)$$

1. If $j = n - 1$, then set

$$P(x_i, x_j) = P(x_i, x_{n-1}) = 1 - F(x_{n-1} - \rho x_i - s/2)$$

1. Otherwise, set

$$P(x_i, x_j) = F(x_j - \rho x_i + s/2) - F(x_j - \rho x_i - s/2)$$

The exercise is to write a function `approx_markov(rho, sigma_u, m=3, n=7)` that returns $\{x_0, \dots, x_{n-1}\} \subset \mathbb{R}$ and $n \times n$ matrix P as described above.

- Even better, write a function that returns an instance of QuantEcon.py's MarkovChain class.

11 Solutions

11.1 Exercise 1

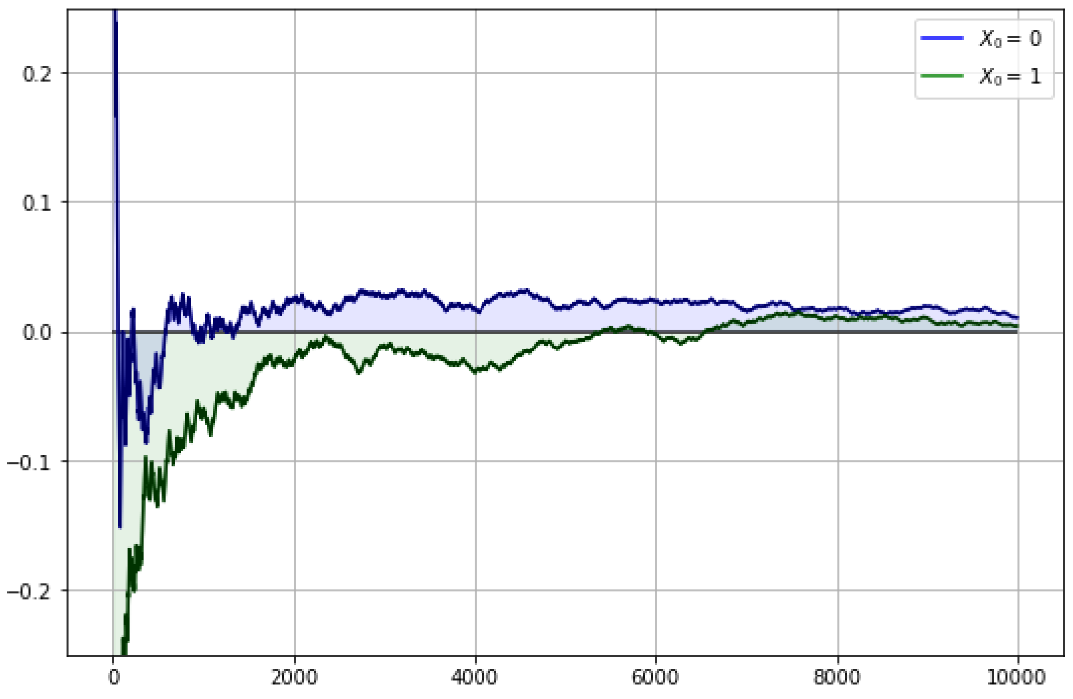
We will address this exercise graphically.

The plots show the time series of $\bar{X}_m - p$ for two initial conditions.

As m gets large, both series converge to zero.

```
In [26]: α = β = 0.1
         N = 10000
         p = β / (α + β)

         P = ((1 - α,       α),               # Careful: P and p are distinct
              (    β,   1 - β))
         P = np.array(P)
         mc = MarkovChain(P)

         fig, ax = plt.subplots(figsize=(9, 6))
         ax.set_ylim(-0.25, 0.25)
         ax.grid()
         ax.hlines(0, 0, N, lw=2, alpha=0.6)   # Horizonal line at zero

         for x0, col in ((0, 'blue'), (1, 'green')):
             # Generate time series for worker that starts at x0
```

```
    X = mc.simulate(N, init=x0)
    # Compute fraction of time spent unemployed, for each n
    X_bar = (X == 0).cumsum() / (1 + np.arange(N, dtype=float))
    # Plot
   ax.fill_between(range(N), np.zeros(N), X_bar - p, color=col, alpha=0.1)
    ax.plot(X_bar - p, color=col, label=f'$X_0 = \, {x0} $')
    # Overlay in black--make lines clearer
    ax.plot(X_bar - p, 'k-', alpha=0.6)

ax.legend(loc='upper right')
plt.show()
```



11.2 Exercise 2

```
In [27]: """
         Return list of pages, ordered by rank
         """
         import re
         from operator import itemgetter

         infile = 'web_graph_data.txt'
         alphabet = 'abcdefghijklmnopqrstuvwxyz'

         n = 14 # Total number of web pages (nodes)

         # Create a matrix Q indicating existence of links
         #  * Q[i, j] = 1 if there is a link from i to j
         #  * Q[i, j] = 0 otherwise
         Q = np.zeros((n, n), dtype=int)
         f = open(infile, 'r')
```

```
    edges = f.readlines()
    f.close()
    for edge in edges:
        from_node, to_node = re.findall('\w', edge)
        i, j = alphabet.index(from_node), alphabet.index(to_node)
        Q[i, j] = 1
    # Create the corresponding Markov matrix P
    P = np.empty((n, n))
    for i in range(n):
        P[i, :] = Q[i, :] / Q[i, :].sum()
    mc = MarkovChain(P)
    # Compute the stationary distribution r
    r = mc.stationary_distributions[0]
    ranked_pages = {alphabet[i] : r[i] for i in range(n)}
    # Print solution, sorted from highest to lowest rank
    print('Rankings\n ***')
    for name, rank in sorted(ranked_pages.items(), key=itemgetter(1),
↪reverse=1):
        print(f'{name}: {rank:.4}')
```

```
  Rankings
 ***
g: 0.1607
j: 0.1594
m: 0.1195
n: 0.1088
k: 0.09106
b: 0.08326
e: 0.05312
i: 0.05312
c: 0.04834
h: 0.0456
l: 0.03202
d: 0.03056
f: 0.01164
a: 0.002911
```

11.3 Exercise 3

A solution from the QuantEcon.py library can be found here.

Footnotes

[**1**] Hint: First show that if P and Q are stochastic matrices then so is their product — to check the row sums, try post multiplying by a column vector of ones. Finally, argue that P^n is a stochastic matrix using induction.

References

[1] Y E Du, Ehud Lehrer, and A D Y Pauzner. Competitive economy as a ranking device over networks. submitted, 2013.

[2] Olle Häggström. *Finite Markov chains and algorithmic applications*, volume 52. Cambridge University Press, 2002.

[3] James D Hamilton. What's real about the business cycle? *Federal Reserve Bank of St. Louis Review*, (July-August):435–452, 2005.

[4] George Tauchen. Finite state markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20(2):177–181, 1986.