Analyzing a Tennis Game with Markov Chains

What is a Markov Chain?

A Markov chain is a way to model a system in which:

1) The system itself consists of a number of states, and the system can only be in one state at any time. This is often viewed as the system moving in discrete steps from one state to another.

2) The probability that the system will move between any two given states is known. These probabilities are usually given as a matrix, where row *i* of the matrix is the list of probabilities of moving from state *i* to any other state.

Once the matrix for a Markov chain is known it can be used to find all sort of probabilities, including the probability that the system will be in any given state after a given number of steps, the number of steps it will take on average to move from one given state to another, and the probability that the system will end up in a given state in the long run.

A simple example of a Markov chain is a coin flipping game. Let's say we have a coin which has a 45% chance of coming up Heads and a 55% chance of coming up tails. I win the game if the coin comes up Heads twice in a row and you will win if it comes up Tails twice in a row. In terms of playing the game since we are only interested in getting two heads/tails we only need to track what the coin was on the last flip. This means the game has 5 states:

State 1: The start (we haven't flipped the coin yet)
State 2: Heads was the previous flip.
State 3: Tails was the previous flip.
State 4: I won the game.
State 5: You won the game.

The probability table for this process looks like this:

	Start	Heads	Tails	I win	You win
Start	0	0.45	0.55	0	0
Heads	0	0	0.55	0.45	0
Tails	0	0.45	0	0	0.55
I win	0	0	0	1	0
You win	0	0	0	0	1

Entering this as a matrix we have:

$$\text{flipmatrix} = \begin{pmatrix} 0 & 0.45 & 0.55 & 0 & 0 \\ 0 & 0 & 0.55 & 0.45 & 0 \\ 0 & 0.45 & 0 & 0 & 0.55 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix};$$

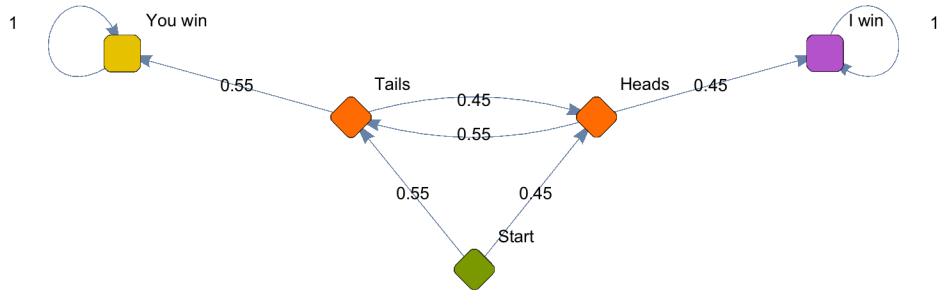
In *Mathematica* we can now describe the game using DiscreteMarkovProcess, letting the computer know the game starts in state 1:

```
coinprocess = DiscreteMarkovProcess[1, flipmatrix]
```

```
DiscreteMarkovProcess[1, {{0, 0.45, 0.55, 0, 0},
  {0, 0, 0.55, 0.45, 0}, {0, 0.45, 0, 0, 0.55}, {0, 0, 0, 1, 0}, {0, 0, 0, 0, 1}}]
```

Before we do any calculations we can have *Mathematica* show us the graph of how the system moves from state to state:

```
Graph[ coinprocess, VertexLabels → Thread[Range[5] → coinstates],
 ImageSize → 500, EdgeLabels → Getedges[flipmatrix], VertexSize → Medium]
```



Now we can see the chance that I will win in 4 flips or less:

```
PDF[coinprocess[4], 4]
```

```
0.363994
```

The chance that I will win in any number of flips:

```
PDF[coinprocess[∞], 4]
```

```
0.41711
```

How long on average it takes me to win

```
Mean[ FirstPassageTimeDistribution[ coinprocess, 4]]
```

```
3.01265
```

How long on average it takes you to win:

```
Mean[ FirstPassageTimeDistribution[ coinprocess, 5]]
```

```
2.96815
```

We could even do this for an arbitrary “Heads” probability. In this case the probability matrix looks like this:

```
Clear[p]
```

$$\text{flipmatrix2} = \begin{pmatrix} 0 & p & 1-p & 0 & 0 \\ 0 & 0 & 1-p & p & 0 \\ 0 & p & 0 & 0 & 1-p \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix};$$

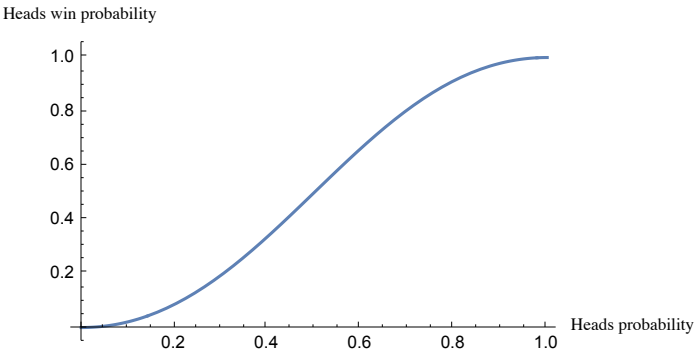
We can now get an exact formula for what my chances of winning are:

```
newcoinprocess = DiscreteMarkovProcess[1, flipmatrix2];
headswins = Simplify[PDF[ newcoinprocess[∞], 4]]
```

$$-\frac{(-2+p)\,p^2}{1-p+p^2}$$

Graphing this as the probability p goes from 0 to 1 looks like:

```
Plot[ headswins, {p, 0, 1},
 AxesLabel → Text /@ {"Heads probability", "Heads win probability"}]
```



If I want to figure out how I need to bias the coin to win 3 times out of 4 I can use Reduce:

```
Reduce[ headswins == 3 / 4 ⋀ 0 ≤ p ≤ 1, p, Reals]
```

```
p == Root[3 - 3 #1 - 5 #1^2 + 4 #1^3 &, 2]
```

```
N[%, 10]
```

```
p == 0.6581690825
```

So if I can bias the coin to come up heads 65.82% of the time I'll have more than 75% chance of winning the game.

A Tennis Game as a Markov Chain

To describe a tennis game as a Markov chain we need to define both the states and the probability to move between states.

There are two states for each score (one for no faults and one for a single fault) together with "Win" states for players A (the server) and B - for a total of 32 states:

```
states = {"{0, {0, 0}}", "{0, {0, 15}}", "{0, {0, 30}}", "{0, {0, 40}}",
   "{0, {15, 0}}", "{0, {15, 15}}", "{0, {15, 30}}", "{0, {15, 40}}",
   "{0, {30, 0}}", "{0, {30, 15}}", "{0, {40, 0}}", "{0, {40, 15}}",
   "{0, Deuce}", "{1, {0, 0}}", "{1, {0, 15}}", "{1, {0, 30}}", "{1, {0, 40}}",
   "{1, {15, 0}}", "{1, {15, 15}}", "{1, {15, 30}}", "{1, {15, 40}}",
   "{1, {30, 0}}", "{1, {30, 15}}", "{1, {40, 0}}", "{1, {40, 15}}", "{1, Deuce}",
   "{0, A Adv.}", "{1, A Adv.}", "{0, B Adv.}", "{1, B Adv.}", "A win", "B win"};
```

The chance to move from any fault/score combination to any other can be described by 4 probabilities (all from 0 to 1):

f_1: The chance Player A faults on the first serve

p_1: The chance Player A will win the point if the first serve is in

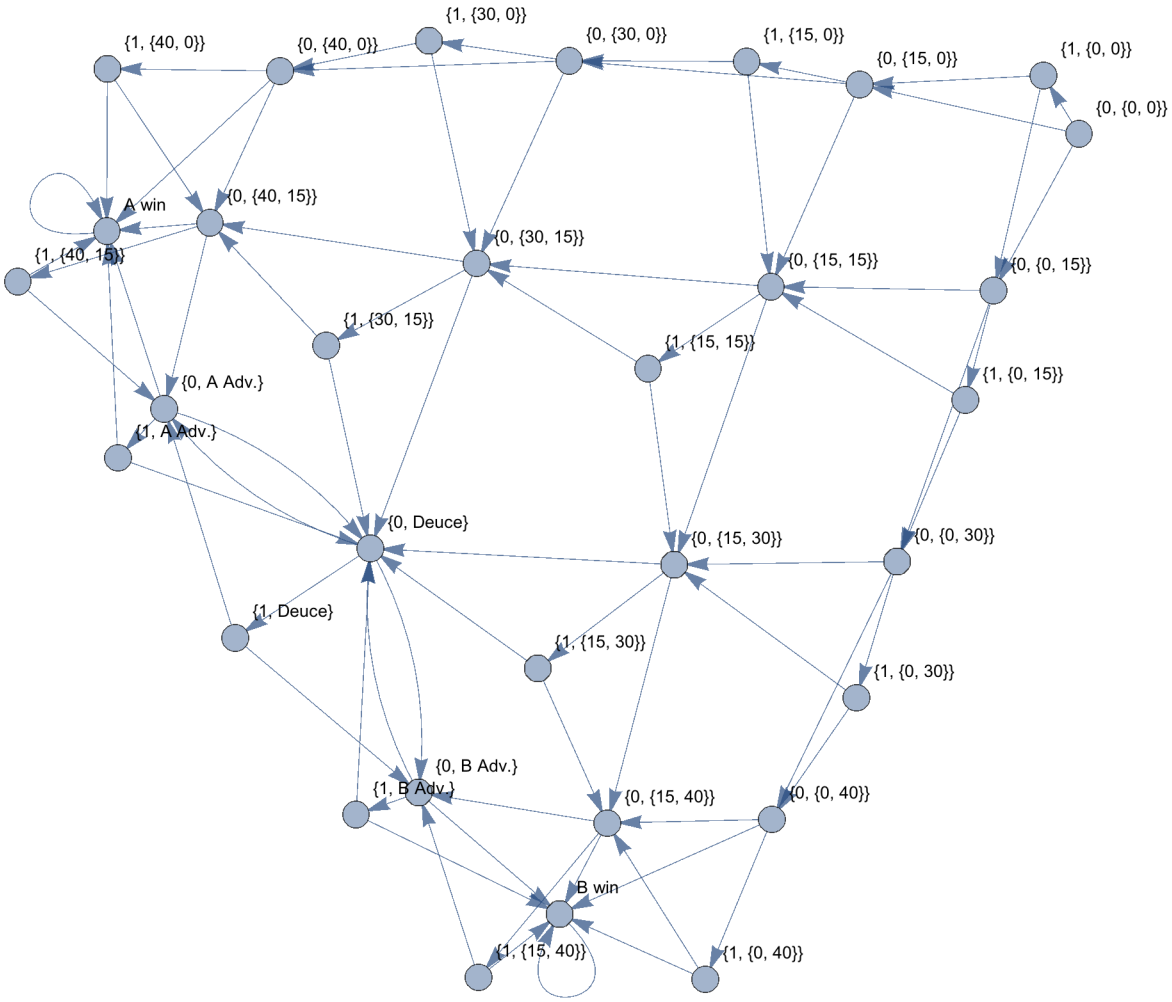
f_2: The chance Player A will fault on a second serve

p_2: The chance Player A will win the point on the second serve

This leads to a 32-by-32 matrix whose entries involve f_1, p_1, f_2, and p_2 and are often 0:

```
tennismatrix =
{{0, 1 - f1 - (1 - f1) p1, 0, 0, (1 - f1) p1, 0, 0, 0},
 {0, 0, 1 - f1 - (1 - f1) p1, 0, 0, (1 - f1) p1, 0, 0},
 {0, 0, 0, 1 - f1 - (1 - f1) p1, 0, 0, (1 - f1) p1, 0},
 {0, 0, 0, 0, 0, 0, 0, (1 - f1) p1},
 {0, 0, 0, 0, 0, 1 - f1 - (1 - f1) p1, 0, 0},
 {0, 0, 0, 0, 0, 0, 1 - f1 - (1 - f1) p1, 0},
 {0, 0, 0, 0, 0, 0, 0, 1 - f1 - (1 - f1) p1},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 1 - (1 - f2) p2, 0, 0, (1 - f2) p2, 0, 0, 0},
 {0, 0, 1 - (1 - f2) p2, 0, 0, (1 - f2) p2, 0, 0},
 {0, 0, 0, 1 - (1 - f2) p2, 0, 0, (1 - f2) p2, 0},
 {0, 0, 0, 0, 0, 0, 0, (1 - f2) p2},
 {0, 0, 0, 0, 0, 1 - (1 - f2) p2, 0, 0},
 {0, 0, 0, 0, 0, 0, 1 - (1 - f2) p2, 0},
 {0, 0, 0, 0, 0, 0, 0, 1 - (1 - f2) p2},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0}}
```

Here is the graph of the states in the chain:



Just as before we can set up the chain in *Mathematica* using DiscreteMarkovProcess (starting with state 1) and find the probability that Player A will win the game (state 31):

```
tennisprocess = DiscreteMarkovProcess[1, tennismatrix];

chanceforservertowin = Simplify[PDF[tennisprocess[∞], 31]]
```

$$\Big(((-1+f1)\,p1+f1\,(-1+f2)\,p2)^4\,\big(15+8\,(-1+f1)^3\,p1^3+34\,f1\,(-1+f2)\,p2+28\,f1^2\,(-1+f2)^2\,p2^2+8\,f1^3\,(-1+f2)^3\,p2^3+4\,(-1+f1)^2\,p1^2\,(7+6\,f1\,(-1+f2)\,p2)+2\,(-1+f1)\,p1\,\big(17+28\,f1\,(-1+f2)\,p2+12\,f1^2\,(-1+f2)^2\,p2^2\big)\big)\Big/\Big(1+2\,(-1+f1)^2\,p1^2+2\,f1\,(-1+f2)\,p2+2\,f1^2\,(-1+f2)^2\,p2^2+2\,(-1+f1)\,p1\,(1+2\,f1\,(-1+f2)\,p2)\Big)$$

Looking at a specific example, suppose one player has fault chances of 39% and 9%, with chances of winning the points of 68% and 58%. Then the chances that player will win when serving is:

```
chanceforservertowin /. {f1 → .39, f2 → .09, p1 → .68, p2 → .58}
```

```
0.777097
```

Suppose the other player has fault chances of 25% and 7% with winning chances of 59% and 37%. The chance that the second player will win when serving will then be:

```
chanceforservertowin /. {f1 → .25, f2 → .07, p1 → .59, p2 → .37}
```

```
0.570966
```

The chance that the first player will win when the second player was serving is then

```
1 - %
```

```
0.429034
```

Over the summer both had a series of games where each served 18 times. The model predicts how many games the first player will win:

```
0.777097 * 18 + .429034 * 18
```

```
21.7104
```

And in fact the first player won 22 times.

We can also model how long each game would be. When our server had fault chances of 39% and 9% and win chances of winning the points of 68% and 58%. When the server wins the average number of serves in the game would be:

```
Mean[
  FirstPassageTimeDistribution[ DiscreteMarkovProcess[1, Evaluate[tennismatrix /.
      {f1 → 39 / 100, f2 → 9 / 100, p1 → 68 / 100, p2 → 58 / 100}]], 31]] // N
```

```
8.60193
```

If the same server were to lose the game, the average game time would be

```
Mean[
  FirstPassageTimeDistribution[ DiscreteMarkovProcess[1, Evaluate[tennismatrix /.
      {f1 → 39 / 100, f2 → 9 / 100, p1 → 68 / 100, p2 → 58 / 100}]], 32]] // N
```

```
9.74636
```

Some analysis

Now that we have a formula we can ask some general questions about how to improve a player's chance to win a game. Suppose you can practice to improve your scores on any of the four values - can you tell how to practice to get the most benefit? This involves taking various derivatives of the win formula and seeing under what conditions the derivative for one variable is larger than the derivative for another. These inequalities get very complex, so it's useful to have the computer simplify them under the assumotion that the range of values for the probabilities is (0,1):

```
range = 0 < f1 < 1 ⋀ 0 < f2 < 1 ⋀ 0 < p1 < 1 ⋀ 0 < p2 < 1;
```

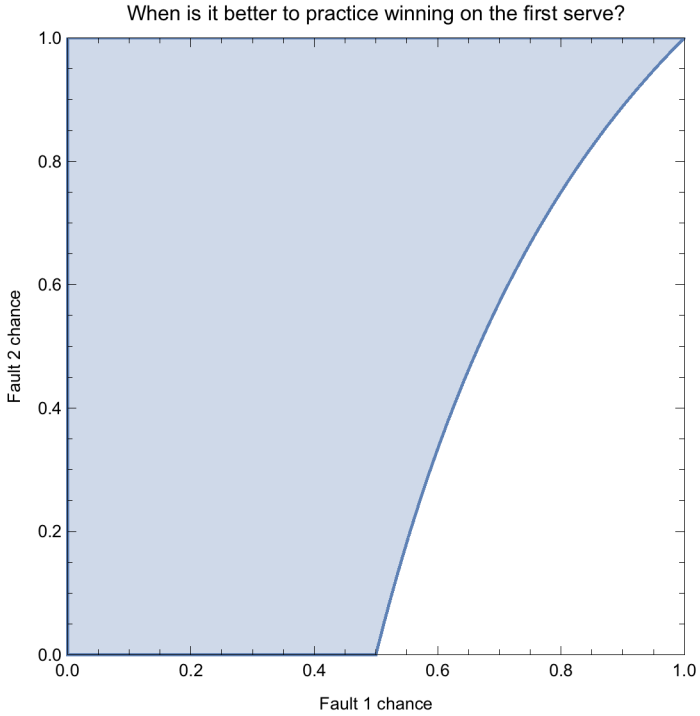
Comparing the benefit gained from improving your chances of winning on the first serve as opposed to winning on the second:

```
Simplify[ Reduce[ D[ chanceforservertowin, p1] > D[chanceforservertowin, p2],
  {f1, f2, p1, p2}, Reals], range]
```

$$\frac{1}{f1} + f2 > 2$$

So it's better to practice winning on the first serve if $\frac{1}{f_1} + f_2 > 2$. We can graph this to see what ranges of values make practicing the first point better:

```
RegionPlot[ 1/f1 + f2 > 2, {f1, 0, 1}, {f2, 0, 1}, PlotRangePadding → 0,
 FrameLabel → {"Fault 1 chance", "Fault 2 chance"},
 PlotLabel → "When is it better to practice winning on the first serve?"]
```



Notice that it's always better to practice winning on the first serve over the second if the fault value $f_1 \le \frac{1}{2}$; for values of f_1 larger than $\frac{1}{2}$ it can still be better provided the fault value f_2 is large enough.

Is it better to practice not faulting on the second serve or winning on that second serve? Note because we are decreasing the chance to fault we need a − on that derivative

```
Simplify[ Reduce[ -D[ chanceforservertowin, f2] > D[chanceforservertowin, p2],
  {f1, f2, p1, p2}, Reals], range]
```

$$f2 + p2 > 1$$

It turns out it's better to practice not faulting on the second serve only when the sum of the chances to fault on serve 2 and win on serve 2 is greater than one.

Author: Dr. Christopher Moretti
Revision Date: August 3, 2015