Markov Matrix

Tony Starfield

recorded: 2005

I'm now going to go into perhaps the most difficult section conceptually of the modeling class, and that is the idea of **ecosystem modeling**. I say it's difficult, because up to now I have been pushing very vigorously the idea of keep it simple, focus on your objectives, and use rapid prototyping. And you may recall the story I told right at the beginning of this class where I said people said ecological modeling and biological modeling is different from everything else. Details always matter. Well, with population modeling and with decision analysis, we figured out how to focus on the main points and use the objective to drive what we put into the model. With ecosystem modeling, one seems to be back to square one, because this is a situation where again people come back and say 'Well, that might work for population modeling, it might work for decision analysis, but when you try to model an ecosystem, every detail matters.' So the discussion today is going to be looking for a way of cutting through ecosystem modeling and trying to use the ideas of rapid prototyping and concentrating on your objectives to develop simple models in a situation where most of the existing models are horribly complicated.

I'm going to kick off by drawing some distinctions in the ecosystem modeling. This figure represents a landscape. Each square or rectangle on the landscape represents some kind of relatively homogeneous patch. If you work in GIS, you can think of it as a pixel. Now, there are two ways of thinking about landscape and ecosystem models. The first is considering a situation where the processes between squares predominate. Those are the processes that are going to determine pattern on the landscape. If one were trying to develop a model where processes between squares were important, then one would begin with a spatial model that looks at interaction between the pixels.

The alternative is to say, let's pick a particular square - suppose we pick that one - and we are mainly interested in the processes that describe how the system inside that square change over time. And in that case, we would concentrate on that particular square, although the processes might be affected by what's happening in neighboring squares. Ultimately, of course, one wants to take both approaches. You want to be able to take the spatial interactions and the temporal story inside each square and combine them.

Where does one start? Should one start with the spatial interactions or should one start with the single square? Sometimes it's six of one and half a dozen of the other. But again, if spatial interactions predominate, start with them. If temporal interactions within a square predominant, start with the single square.

We're going to talk about the single square today. And the way models of a single patch or square have been developed in the past looks something like this. This is a very simplified version. Here G stands, for example, for grass. S for shrubs. T for trees. The little circles below refer to the below ground biomass, the root systems and so on. Gr stands for grazers, Br for browsers, Pr for predators. And then we have clouds producing rainfall, sun producing sunshine. The arrows represent interactions between the circles. So, for example, a tree might send material down to the root system or draw material up from the root system. The browser is going to affect the trees and the trees are going to affect the browser. Now, why do I say this is oversimplified? Well, one could break grasses up into palatable and unpalatable grasses. One could look at small trees and tall trees. One could look at small grazers and large grazers. One could break the predators up into those that eat small animals and those that eat larger animals. I think you can see the opportunities for expanding the model, and expanding it, and expanding it are horrific.

The way in which people have approached the construction of ecosystem models like this has been to first come up with a diagram, such as this, for the system. And then identify the experts in each of the components. Each of these circles represent a component of the system. And the arrows are obtained by the experts discussing what they would need from somebody else's components. So, for example, the arrow going from grass to grazers comes from the grazer expert saying, ‘I need to know something about the amount of grass available.’ The arrow going from grazer to grass is the grass expert saying, ‘I need to know how many grazers there are and what they're eating.’ And having sorted out this kind of diagram, the usual procedure is to then say to the experts, ‘Go away and develop the model for your circle - being aware of what you are going to get from other circles and what you need to give to other circles.

The idea here conceptually in terms of computer programming is rather like a subroutine. Each circle is a subroutine or subprogram that that expert is free to go out and develop. And eventually what happens, maybe a year later or two years later, is everybody comes back with their individual programs and one tries to knit them all together.

So, if you think about this, this is the exact opposite of rapid prototyping. This is a situation where you don't even see how your computer model is working until everybody has developed his or her own piece and you have put them all together. It's a huge investment in time and effort before you get any feedback at all or any output from the model.

So the first approach one takes is one of saying, ‘How can one simplify this? How can one control this?’ Remember the concept of you need to control your model. Your model mustn't control you.

So one starts by saying in the diagram, let's make sure we only put in everything that is absolutely necessary. The problem is, if you think back to the ping-pong ball example, that there are lots of clock experts out there, and so you're going to have people who want to put into the model components that you think are unimportant, but because you are not going to test the model until way down when everybody's developed their components, it's very difficult to say to them ‘Leave them out, we'll come back to you in a year or two years to see if we need to introduce your component.’

So the pressure is to put in more detail rather than to simplify. However, one could try to develop a model like this where one asks the question, ‘Is every component absolutely necessary?’ And where one tries to develop the component models as simply as possible. That's the closest you could get to rapid prototyping.

So for years I have grappled with this idea of how to take the process that we've been looking at in this particular diagram - how to take this process and streamline it and control it. And eventually I came to the conclusion one couldn't. If you look at this process, I would describe it as a bottom-up process. Now, bottom up and top down are used in different contexts in ecology. But the idea here is that you build the model up from the components. And eventually I started asking myself the question, ‘If bottom up is so hard to control, is it possible that one ought to be taking a different approach?’ And a different approach would be top-down.

So, how might a top-down approach work? Well, remember we're talking about a single patch or pixel. Now, in the modeling I've been talking about, usually that pixel is fairly small. One actually has a few shrubs, a few trees, some grass and a few animals walking in and out. But suppose one has a larger pixel. Perhaps a square kilometer or a square mile even. And let's imagine a satellite that passes over that patch, say once a day or once a week. And suppose we could collect 100 years of information, or 200 years of data from that satellite. Every single time it passes it takes a picture and we create a video of 100 or 200 years of those pictures.

And suppose we take that video and we play it awfully fast. We review it in the space of about 10 or 15 or 20 minutes. And then we say to whoever is watching, ‘Okay, what did you see?’ What they are going to say is something very simple. For example, if they were looking at a patch in northern Minnesota, in a forest patch, they might say, ‘Well, for some time we saw forests dominated by jack pine. And then it seemed to be dominated by spruce. And then it was dominated by jack pine again. And then it looked like a white and red pine dominated forest.’ In other words, they would be identifying ‘states’ which you could apply to that pixel in exactly the same way in which a GIS expert might take remote sensing data and categorize a pixel.

Well, the simplest thing one could do in a top-down approach is to try to figure out ‘How could I represent the data collected from that satellite?’ And the way one might be able to do it would be in a diagram like this.

Here we have time along one axis and we have the states - jack pine, white pine, spruce - along the vertical axis. And if we are running that model and we say jack pine changed to spruce for a little while, changed back to jack pine, stayed there for a while, went into white pine, stayed there for a while, went into spruce, stayed there, and then came back to jack pine. That would be the simplest way of representing the data from this remote satellite.

Well, if that's the simplest way to present it, then the first step in a rapid prototyping modeling approach would be to say, how could I model that? Can I develop a model that will produce output that looks like that, that will produce a graph of state versus time? If one follows that line of reasoning, you start going to the literature and saying, are there any models out there in the literature that could possibly do it? And it turns out that the first kind of model you find in the literature is something that is called a **Markov Matrix Model**.

And a Markov Matrix Model works something like this, one chooses a time step - so this is salami tactics. And one has time T and time T + 1. If you were working in the forests of northern Minnesota, things change rather slowly. So maybe the difference between T and T + 1 would be 10 years. Now, at time T you know that your patch is either a jack pine patch, a white pine patch or a spruce patch. So you know which state you are in. You want to predict what state you are going to be 10 years later. Is it going to be jack pine, white pine or spruce?

So suppose we start in jack pine. What this matrix represents is the probability that 10 years later you will find it still in jack pine. And these data are obtained from either lots of data on one patch over a long period or by looking at a large number of patches over a relatively shorter period. So the best estimate is that if you are in jack pine there's a 70% chance, a .7 probability that you will still be in jack pine 10 years later. A .1 probability that you will be in white pine. And a .2 probability that you'll be in spruce. Notice that these numbers add up to 1. If they didn't, there would have to be some other state.

If we look at the spruce column, we see that there's a .5 chance that if you are in spruce now you will be in jack pine 10 years later and a .5 probability that you will still be in spruce. And that there is no way you can switch in 10 years from spruce to white pine.

Now, suppose one has these numbers. How could one build a model to try and produce the time line we were talking about? Well, I think you could see that you could use your random number generator to do that. Suppose you were in jack pine. Suppose you generated a random number. If the random number were less than .7, you would say I'm still in jack pine. If it were between .7 and .8, you would say, I'm in white pine. And if it was between .8 and 1, you would say I'm in spruce.

And if you did that a large number of times, 70% of the time you would switch to jack pine, 10% to white pine and 20% to spruce, which is precisely what you want. And something like this could be done very easily on a spreadsheet. Let's take a look at how this might be done.

< 00:16:45 END >