Unit 1: Introduction

READINGS AND RESOURCES

- Smaldino PE (2017) Models are stupid, and we need more of them.
- Kauffman, S. A. (1971). Articulation of parts explanation in biology and the rational search for them. In R. C. Buck & R. S. Cohen (Eds.), PSA 1970 (pp. 257–72). Irvine, CA.
- Weisberg M (2007) Who is a modeler? *British Journal for the Philosophy of Science* 58: 207-233.
- NetLogo: https://ccl.northwestern.edu/netlogo/
- NetLogo User's Manual: https://ccl.northwestern.edu/netlogo/docs/

Welcome

Welcome to this short course on models of social dynamics. I'm Paul Smaldino, I'm a Professor of Cognitive and Information Sciences at the University of California Merced.

We're here because we're interested in understanding the complex dynamics of social, ecological, and evolutionary systems. This is challenging. One reason that it's challenging is because the tools used to describe those systems, especially for those trained in the social sciences, tend to have severe limitations. Verbal models can paint vivid pictures in our minds, but they are often ambiguous. Statistical models are important for establishing empirical relations, but they rarely establish causal origins for phenomena of interest.

This course is about formal models: mathematical and computational models of complex processes. More specifically, in this module we'll focus on relatively simple computational models that even those without heavy math or programming backgrounds should be able to get off the ground relatively painlessly. Formal models can help us communicate ideas unambiguously, and they provide us with a toolkit with which we are better able to form intuitions and develop generative theories about the behaviors of complex social systems.

The material for this course is inherently interdisciplinary. We will cover models on topics that should be of interest to almost anyone who studies social behavior. After this one, each of the next 5 units will focus on a broad topic and some simple but illuminating models. We'll deal with:

- Contagion
- Opinions and polarization
- Cooperation
- Coordination and norms

- Sociopolitical cycles

Before we get started, I want to go into a little more detail about what models are and how they are useful. After that, before this unit is over I'll give a brief introduction to the NetLogo programming language and how we will use it to explore agent-based models.

Philosophy of modeling

What are models? Why should we use them?
Lots of researchers use models. How do you know if you're using a model? Ask yourself this question: am I directly studying the thing I'm interested in? Something the answer is yes. If I want to know the mass of a rock, I can weigh the rock. Very often, the answer is no. Instead, I represent my system of interest with something else. In this case you may be using a model, which is any structure (real or abstract) that can represent some real-world phenomenon.

- Engineers may make physical scale models to assess design principles.
- Biomedical researchers may use animal models to make general inferences about genetics, hormones, etc.
- Behavioral researchers may design experiments that model the larger class of phenomena of interest. Consider the marshmallow test. Unless you work in the marshmallow industry, you are probably not that interested in how long children can wait to get a second marshmallow before eating the first. Instead, researchers use this to model situations where issues like willpower and trust in authority come into play.
- And finally, we can build formal mathematical or computational models. Sometimes these models are extremely specific, and are basically simulations of physical models used by engineers. Sometimes, the models are more abstract, and instead are designed to capture core elements of a theoretical idea.

Formal models are special, because the contain nothing more or less than what we put into them. Sometimes, critics of models say "well, you baked some result into the model, so it had to happen." This is literally true of every formal model, because the model is merely a series of computations specified by the modeler. To put it another way, a formal model is a logical engine that turns assumptions into conclusions. This is useful because our intuitions about complex systems are often terrible. Good models can often show us how our assumptions lead to unexpected conclusions.

We use models for a lot of reasons. Here are some.

- Prediction. Sometimes models let us make empirical predictions. This is obviously useful. But models have lots of use other than prediction.
- Precision. The word is messy. Words are ambiguous. We can speak past each other. Multiple can use the same words in different ways, leading

to several interpretations that are consistent with the same set of words. In contrast, a formal model is specified in math or code and it is very clear what it does or does not do.

- Tractability. The world is complicated. Data may be hard to acquire to do constraints of time, space, or cash. And if we're interested in big picture stuff like history or evolution, we can only run the clock once. Models, on the other hand, are only constrained by our ability to imagine how a formal system might be constructed, and to some extent by the limitations imposed by what is computationally feasible.
- Mental models. How we interpret what we see in the world is shaped by our ability to draw analogies to things we have experienced or imagined in the past. Everything we learn is filtered through the narratives we hold in our memory. By learning about formal models, we arm ourselves with new analogies for the behavior of complex systems. Of all the uses for formal models, this may be the most powerful.

Hypothesis formation and the articulation of parts

In order to form a hypothesis or theory about some system, we have to articulate the stem in terms of a set of parts and the relationships between those parts. This is sometimes called decomposition.

How should a system be decomposed? There's no one right all-purpose answer. Instead, it should be shaped by what is useful for constructing a meaningful explanation.

Here are two prominent examples:

Newton's model of planetary motion.

In the early 17th century, Johannes Kepler used the best available astronomic data to show that not only did the planets orbit the sun, and not the earth, but that their orbits were elliptical, not circular. Why? No one knew. In these early days of Western science, several explanations were proposed, but it was Isaac Newton whose explanation persisted, because he developed it using a formal model. He proposed a universal law of gravitation in which objects are attracted to one another with a force proportional to their mass and the inverse square of their distance. It's noteworthy that others beside Newton had also proposed that the force of gravity might be approximated by an inverse square law, including luminaries like Robert Hooke, Christopher Wren, and Edmund Halley. Newton, however, was the first to test the consequences of this claim. By modeling the trajectory of one celestial object around another, he showed that his inverse square law could rise to the observed elliptical orbits, and enabling his theory of gravity to remain the gold standard for over two hundred years until it was replaced by general relativity.

Lotka-Volterra model of predator-prey interactions

Let’s take another example. For decades, fur traders like the Hudson’s Bay Company in Canada noticed cyclical patterns in the populations of the animals they hunted, like the Canada lynx and the snowshoe hare. Not only did both populations rise and fall, but their cycles seems to be locked in phase, and unexplained by external factors like climate. An explanation came from two Italian mathematicians, working separately, who applied ideas of autocatalytic chemical reactions to the problem. Assume that, in the absence of lynx, hare will reproduce and increase in number. In the absence of hare, lynx will starve and decrease in number. When both are present, lynx will eat hare, increasing their numbers while diminishing the hare – their food source. When these assumptions are formalized, which can be done using coupled differential equations, the cyclical patterns observed by the fur traders emerge.

Both these examples concern models being used to explain existing patterns in data. As we will see, models can also be used to explore more abstract theoretical ideas, though in such cases more background is usually needed.

Both of these examples are also expressed as mathematical equations. In the Lotka-Volterra model, the populations of predators and prey animals are each expressed as a single number. This has the nice advantage of tractability – we can often find exact solutions – but comes with several costs.

- Limited realism. The models assume homogeneity: that every member of a species is identical to every other, and that interactions occur at random, almost as if the populations were swirling in a giant gas cloud. In reality, most social populations have important sources of heterogeneity. Individuals differ in traits all kinds of ways, both in terms of inherent traits and in terms of social and environmental affordances. Populations are structured non-randomly.
- Limited intuition. Unless you are very familiar with functional forms, it can be difficult to get intuitions about a system just by looking at some equations. It would be nice if the population dynamics could be visualized in an intuitive way.
- Higher bar to entry. Reliance on mathematics sets a high bar to entry for using these models. To use the two models discussed, you must understand differential equations. I think differential equations are worth learning, but a lack of mathematical background shouldn’t impeded you from many of the great insights of modeling. There is another way.

Agent-based modeling

An alternative approach is agent-based modeling. An agent-based model is a class of formal model in which individuals (aka “agents”) are simulated as explicit computational entities. So, for example, instead of using a single number to represent all of the predators in a population, we would represent every single predator as an individual agent.

There are some costs to using ABMs. Instead of clear analytic results, we must run simulations and often rely on abductive reasoning to draw conclusions.

However, there are also many benefits. ABMs can account for more heterogeneity, complexity, and population structure.

To be clear, I think it’s counterproductive to debate whether mathematical or agent-based models are better. They are both useful. Which one is appropriate for you depends on the complexity of the problem and the skillset of the researcher. In many cases, we can try to get as far as possible with mathematical analyses, and then tackle the more complicated questions with agent-based modeling. On the other hand, agent-based models may be more intuitive and easier to implement than mathematical models for many scholars. In this introductory course, we will focus on agent-based modeling.

Agent-based models are inherently computational. You will therefore need to write your model using a computer language. Most programming languages are Turing complete, which means that, in theory, anything you can do in one language you can do in any other.

I have personally worked on agent-based models in: Java, R, MatLab, Python, and NetLogo. I have colleagues who regularly use C++, Javascript, or Julia. In general, you should use whichever language you’re most comfortable with. For this course, we need to pick a language. We’re going to use NetLogo.

NetLogo first appeared in 1999, and is a software package expressly designed to build and analyze agent-based models with the philosophy of “low threshold, no limit.” “Low threshold” means that NetLogo is very easy to learn even for those with programming coding experience. It has a bunch of drag-and-drop features, and many built-in functions common to models of social behavior. “No limit” means that, in theory, NetLogo can be used for high-level scientific modeling. And many peer-reviewed papers are published each year describing models built and analyzed with NetLogo.

For me, the primary reason to use NetLogo for this course is that it is designed to be easy to learn, with syntax that reads like pseudocode. And it is extremely easy to get dynamic visualization and plotting. In general, I think it’s a really fun program to play with, and I hope you agree.

This class assumes no prior knowledge of programming, but some exposure with the basics – things like variables, conditional statements, and loops, will be helpful. Even better if you know something about object-oriented programming, but it’s OK if you don’t.

A NetLogo Primer

I am not going to give a detailed course on how to use NetLogo. For this, I strongly urge you to download NetLogo and go through the tutorials in the included user manual. Play around with the code in the included Models Library. You’ll get the hang of it. You can find all the necessary materials at http://ccl.northwestern.edu/netlogo/

Patches and turtles

The most basic unit of an agent-based model in NetLogo is the turtle. Most times, the turtles are your agents. Why are they called turtles? In the 1950s, a scientist named Gray Walter made a simple robot that could be easily programmed to walk around the room by accepting commands for distance to walk forward and turning angles. He called this robot the tortoise. Then, in the 1960s, Feurzeig, Papert, and Solomon (1967) introduced the Logo programming language, in which a little green triangle could move around similarly, allowing users to draw names. They called the triangle the turtle, in part after Gray Walter, and in part because it sort of looks like a turtle if you squint. NetLogo follows in this tradition: it is uses networked turtles.

Turtles are located in space, can move, die and reproduce, can be networked, and can have any number of properties you want to give them. What is this space their on? Patches. The default space in NetLogo is a two-dimensional grid of square patches. Patches are defined by their location and do not move, but they can also have arbitrary properties. They can interact with the turtles, and some models will use patches rather than turtles as the principal agents.

Netlogo also has a third category of object called Links, which can be used to model arbitrary network structures. We won’t deal with these in this course, but it’s a nice feature.

Asking

One of the most important things needed for agent-based modeling is the ability to command each agent to execute the same set of commands in a random order on each tick of the model’s clock. In NetLogo, however, we don’t command, we ask. Of all the procedures used, the most important one is ASK, which generates a list of all the agents that meet some criteria and, in a random order, has each of them run some block of code.

BehaviorSpace

I won’t go into this in too much detail, but NetLogo has this useful tool for doing batch runs to get the distribution of model outcomes over a range of parameters. The purpose of this tutorial is to introduce insights from modeling, not model analysis, but any data I show aggregating across many simulations, I used BehaviorSpace. Batch runs are essential for analyzing agent-based models. If you're going to use NetLogo for agent-based modeling, BehaviorSpace is an essential tool.

A simple simulation

CODE: movement.nlogo

To get familiar with coding up a model in NetLogo, we'll make a simple simulation that simple creates some agents and has them move around. This will allow us to get familiar with some of the methods and conventions of NetLogo and agent-based modeling. In this unit, and every unit in this course, I've provided the basic model code, so that we can focus on understanding the model's behavior. At your own pace, examine the model code to understand how it works and how modeling assumptions were translated into code.

In this first simulation (it's not really a model of anything), we'll simply create a bunch of turtles and have them execute a simple correlated random walk. For fun, we'll also enable them to leave colored trails of their movement paths. Our model will have several parameters that we can manipulate on the fly.

SETTING UP THE MODEL

- Our agents will move in a two-dimension space with toroidal boundaries, also called period boundaries. This just means that we avoid the complications of dealing with boundaries and corners in our world, and allow it to "wrap around," so that an agent going off the left edge of the screen appears at the right edge, and so on.
- We'll create buttons for our SETUP and GO procedures, which correspond to how the model is initialized and how the dynamics play out at each time step.
- Our model will have several parameters for us to adjust on the fly:
    - *num-turtles*: the number of agents in our simulation
    - *speed*: the size of the forward step agents take each time step
    - *turning-angle*: the maximum turn the agents make.
- We'll also add a Boolean switch called pen-down? A Boolean variable is one that takes only two possible values: true or false. This one will allow us to toggle whether around agents leave trails when they move.

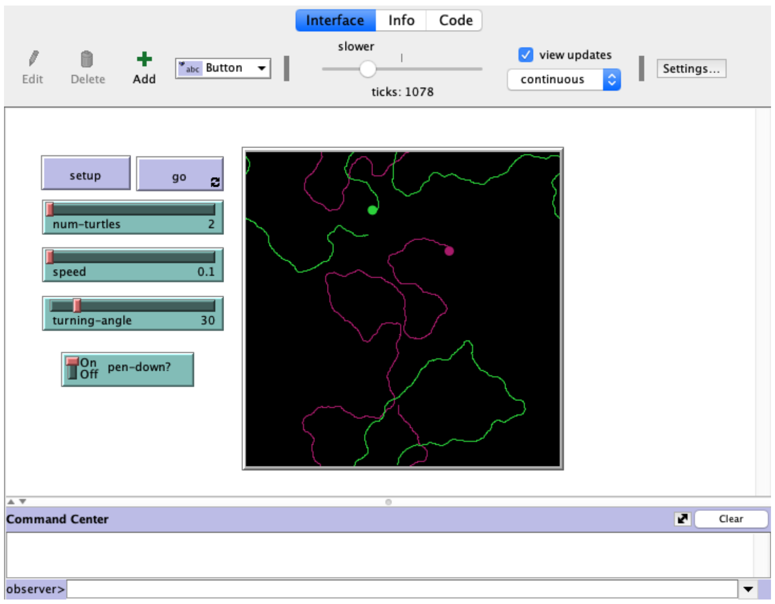
INITIALIZATION

- Our setup procedure always begins with clear-all (clearing all variables and starting fresh) and ends with reset-ticks (which resets the model's clock).

- We then create our agents (turtles), with the number determined by our parameter num-turtles. Each newly-created turtle immediately executes the code in brackets, which sets its shape to a circle and places it in a random spatial location.

DYNAMICS

- First, we use a conditional statement to determine whether the turtles will be asked to put their pen up or down, using our Boolean switch. This allows us to toggle the pens throughout the model runs.
- Then, each turtled is asked to execute a few commands. This means that, in random order, each turtle does the following:
    - o Turn a random amount. Each turtle has a directional heading, which is randomly assigned when it is created. Here, the turtle draws a random integer from a uniform distribution between 0 and *turning-angle*, and turns that many degrees to the left. It then does the same to turn to the right.
    - o Note that the resulting distribution of turning angles is not uniformly distributed between plus and minus turning-angle. Instead, smaller turns will be more common. The reason that that the sum of two uniform distributions is not uniform but binomial – it's the same reason that when you roll two dice, a seven is more common than a two.
    - o The turtle then takes a step forward some amount, where a step size of 1 corresponds to the width of a patch.



ticks: 1078

num-turtles 2

speed 0.1

turning-angle 30

That's it. You can slow down the model using the speed slider in the Interface tab to get a sense of the agent movement, and use the pen to see how different parameter values lead to different movement trajectories.

For the rest of the course, I will focus on the formal assumptions of each model, and less so on the specifics of how those assumptions are executed in NetLogo. It is left up to you to examine the code to see how the code is constructed.

Outline of the course

Including this one, the course is divided into seven units, with units 2-6 each focusing on a particular class of models, and the short unit 7 summarizing the lessons learned and suggesting directions for further study.

- In unit 2, we'll consider models of contagion: how diseases, behaviors, or information can spread from person to person.
- In unit 3, we'll look at opinions, modeling information in a slightly more nuanced way – as a continuous rather than binary variable that can be influenced by social forces and individual predilections.
- In unit 4, we'll consider a key factor in social organization: cooperation. We'll look at this through the lens of evolutionary game theory, and see how cooperative strategies might be able to overcome those who would exploit their goodwill.
- In unit 5, we'll consider that cooperation works best when individuals can coordinate on norms and goals, and we'll investigate how new norms might spread.
- In unit 6, we'll consider dynamics that are cyclical rather than those that converge to equilibria. We'll first tackle a simple predator-prey model, before turning to a more complex model about the rise and fall of empires.
- Finally, in unit 7, we'll consider what we've learned and how we can apply it moving forward.

I've chosen these topics because I think they are foundational to understanding social change and cultural evolution. For me, a useful mnemonic for these foundations is the "3 Cs": Communication, Cooperation, and Coordination. Understand these, and you understand a lot about human nature. At the end of each unit, I'll also talk about how the ideas and models I've presented have been extended and taken in other directions. Let's get started. Thanks for watching.

Exercises

- *What models are*. In each of the following cases (a-d) answer the following: (i) What might be seen as a model, and how might it represent a real-world system? (ii) Give an example of a question about that real-world system that could be at least partly answered by analyzing the model. What are some limitations to how well that question can be answered with the model?
  a) A mathematician writes down equations describing two quantities, X and Y. X increases when Y is small and decreases when Y is large. In

contrast, Y increases when X is large, but decreases when X is small. She graphs changes to X and Y over time, and experiments with the exact rates at which X and Y grow and shrink in relation to each other.

b) A biopsychologist wants to understand the role of hormones in human cognition. He breeds rats that lack receptors for the hormone vasopressin, thought to be involved in social cognition and emotion. He studies the formation of partner preferences with these rats, as well as with other control rates that do not lack the receptors.

c) An engineer builds a scale replica of the San Francisco Bay watershed in her garage, including the water and all major geological and civic structures (inlets, bridges, etc.). She can modify the replica in any way she likes.

d) A behavioral economist recruits college students into his lab for an experimental game, in groups of four. He gives each individual $10, and allows each to either keep the money or donate some proportion to a central fund. All donated money is then doubled and distributed evenly among all four players.