On the Importance of Frailty in Social Science Theory (and other lessons of agent-based modeling)

Paul E. Johnson
Associate Director, Center for Research Methods and Data Analysis
Professor, Political Science, University of Kansas

2012

This paper was prepared for presentation at the 4th World Congress on Social Simulation, National Chengchi University, Taipei, Taiwan, September 4-7, 2012.

Abstract

This paper is about the theoretical implications of agent-based modeling exercises. Construction of an agent-based model challenges a social scientist to formalize many concepts and relationships that would have remained implicit or unrecognized. While formalizing these "unimportant" assumptions can be a nuisance, it can also have substantial theoretical payoffs. In order to fill the gaps of the model, the researcher is forced to confront the gaps in the theory that motivated the model in the first place. Using examples drawn from several large political science simulation models, the paper argues that frailty, defined as unpredictability in the behavior of agents, is often required in order to bring closure to the modeling exercise. It is difficult (or impossible) to square the dynamic or aggregate implications of the agent-based model with observations without placing a substantial amount of emphasis on frailty. Hence, the component in behavior that we often treat as "error" in empirical analysis is actually a vital part of the glue that makes the many different moving parts of a social system interact in coherent ways. The example models were developed with the Swarm simulation system (http://www.swarm.org) during the last decade.

1 Introduction

This paper is about the process of agent-based modeling. The "process" is one of translation, from a researcher's ideas about the parts of an interactive process into a working computer model. I'm trying to summarize my experience in a number of modeling projects and outline some ideas about when the translation from idea to model is most likely to be useful. Another purpose is to make it easier for researchers and program developers to understand each other. To facilitate that communication process, I have some suggestions about the process that occurs between the posing of the question, “what would happen if we had an agent-based model of (whatever)?” and the development and analysis of the computer model.

I’m writing about this from the perspective of a professor who frequently serves the role of a programming consultant. Researchers, professors and students alike, often approach me with ideas that they would like to translate into an agent-based model. After working on quite a few projects, I’ve accumulated a mental list of “danger signs” and “opportunity indicators” about the agent-based modeling projects.

2 Terminology and Purposes

First, the easy part: terminology. For the sake of discussion, here are some definitions.

agent. An isolated collection of data and routines for gathering, adjusting, and (selectively) revealing data.

agent-based model. A collection of agents along with a computer framework required to manage their interaction and to collect information.

Second, the difficult part: what are the purposes of agent-based modeling?

In my opinion, the creation of an agent-based model is best understood as a “hypothesis-generating exercise” (Johnson, 1996). We want to find out what might happen if a collection of separate things are thrown together into a system. The agent-based modeling exercise is on the same epistemological level as game theory or mathematical models of physical systems. These models take simplified characteristics of objects and formalize them so as to use tools of reasoning with which they can be better understood and formalized. These models are not conducted with the purpose of representing reality in a one-to-one translation. Rather, the purpose is to force ourselves (or our clients) to think through a theory thoroughly and understand all of the implications of our newly-made-explicit ideas.

Many of the earliest, and still most exciting, agent-based models were framed on the effort to produce “something” from “nothing.” They seemed to be realistic because they generated patterns that appeared to match empirically observed patterns. The macro-level “something” emerged (Holland, 1992, 1996, 1998; Strogatz, 2003; Miller and Page, 2007; Mitchell, 2009). The separate parts seemed to organize themselves (Bak, 1999). Patterns that we notice in our empirical reality, patterns that we may have taken for granted or that we may not have noticed, are given a deeper foundation, a bottom-up explanation (Epstein and Axtell, 1996). For examples, consider the abstract work in artificial life (Langton, 1986, 1995) and evolution (Kauffman, 1993, 1995).

Within this school of thought, it is *very* important to design agents that are *very* simple, perhaps even unrealistically simple. Consider the models of bee-hives or ant food searches, for example. The bees are very simple creatures, they do not intentionally regulate the temperature of the hive. And yet, as they go about their simple “lives,” the hive’s temperature ends up “just right”. The ants don’t understand that they are following or emitting pheromones as they follow the trail between their home and a pile of food. The models of bees and ants (or ’boids’, or whatnot) don’t imbue the individuals with much, if any, intelligence, and yet the aggregate pattern is produced.

When I meet a client who has been reading the complexity literature, I generally know what they want from an agent-based model. The mission, which is nearly ubiquitous in the literature of complexity and artificial life, is to produce an emergent property. I choose to define emergent property as an aggregate regularity that is not obviously expected on the basis of the individual components in a model. Note the wiggle room created in this definition by the phrase, "not obviously expected." I've often wondered if emergent properties must be "unexpected properties" (and for whom must they be unexpected).

The main idea here is that most people who are presented with an agent-based model will be interested if its aggregate properties are consistent with their understanding of reality. But is even better if the property evidenced in the model can then be "discovered" in reality. Empirical regularities seem to come in three flavors. One is a "factoid" that is widely understood as a pattern in reality. At the opposite end of the spectrum, we have a "revelation," something that exists, but has not been recognized (probably because theory did not tell us where to look). In between the two, we have a "hunch," something a researcher suspects, or wishes to demonstrate, but other scientists are not aware of it. An agent-based model that generates a emergent property that amounts to the revelation of an empirical pattern will have traction with the scientific audience. In the literature of complexity, the best example I have heard of is the models of "landslides" in sand piles and self-organized criticality (Bak, 1999). As far as I can tell, these models generate claims about the relevance of "power laws" in observational data that we might not have detected otherwise. I have the same impression about mathematical models based on fractals (Mandelbrot, 1983; Barnsley, 1993).

Of course, any aggregates that emerge from a model can be interesting, and here we come to the curious role of factoids. I had not realized that the the "hive stabilizing" behaviors of bees had been recognized for several decades (Heinrich, 1981) when I started reading about Santa Fe Institute projects that generated an emergent property of stable temperatures in simulated hives (Hiebeler, 1994). The fascinating thing about the bee models is not only that we understand problem management among bees (although that is very interesting, if you happen to be a scholar of bees!), but that we might gain insight into other domains of communal activity. Swarm Intelligence is the moniker for the general proposition that communal problems of humans might be understood from a bee hive's point of view (Theraulaz and Deneubourg, 1992; Bonabeau et al., 1999; Bonabeau and Theraulaz, 2000; Jacob et al., 2007). The motivating idea in this context is that the bee hive model is well understood and consistent with our understanding of reality, and so we hope for a "spill over" from that domain to others. If a model produces a pattern that seems obviously wrong–it is not consistent with our factoids–we may not be interested. As such, the patterns that we believe are real help us to sort though proposed models. Of course, if a model produces an emergent property that seems wrong, but more careful empirical analysis rejects the factoid, rather than the model, then we think the model is even more useful.

Finally, consider the importance of hunches and the "reverse engineering" approach to agent-based models. I doubt that very many models have emergent properties that were completely unexpected to their authors. The most usual path (again, I believe) to an agent-based model begins with a supposed aggregate pattern that the researcher wants to "reverse engineer." A researcher observes (or guesses about, or wishes for) an aggregate pattern, and then sets out to develop an individual level model that produces that aggregate pattern. Some researchers present results indicating (or pretending) they have found an unexpected aggregate pattern from the agent-based model. If they are truthful in making that claim, we are more likely to take the model seriously, because it appears "more true," or "more like reality." That model is not a magician's card trick; it is not designed primarily to produce the "unexpected" emergent property.

On the other hand, if we suspect that the model is engineered to produce that aggregate pattern, we are less interested. Since we cannot see into their minds at the moment of a model's creation, we are unable to know if we are being deceived when the authors of agent-based models claim that their model produces some previously unrecognized empirical reality. If we suspect that a model is reverse engineered to produce a emergent property, we are less interested mainly because, rightly or wrongly, it seems as if the author has not made an imaginative contribution. It is not difficult, or so it seems, to code up some agents to produce any particular aggregate pattern. Strictly speaking, I don't think that is correct, since I've tried to write simple models to reproduce some aggregate patterns and found it surprisingly difficult to do. Nevertheless, many suspect that a model that is engineered to produce an emergent property is both easy to create and intellectually dubious. Whether an emergent property mirrors some empirical reality is really quite beside the point here. The point is that it is probably easy to design a model to produce one pattern, and so we suppose there's not much worth learning from that model.

How can we reclaim credibility for a model that has, to some extent, been reverse engineered? This did not occur to me while working on a series of projects in the early 2000s, but now the answer seems perfectly obvious. One must reverse engineer some similar models to produce different emergent properties! This allows us to conduct comparative statics and dynamics, to find out if one model's design is somehow more realistic or interesting than another. From this point of view, then, a modeling project faces a heavy burden. Not only must one develop the "author's" model, the one the author is interested in, the one that has the right emergent properties. One must also develop the "other guy's" model, so that the author's model has something firm to lean against.

Anybody can throw together a model that produces some aggregate regularity. That's not the objective for most researchers in the complexity tradition. Instead, the objective is to produce an aggregate reality that others would not expect from an inspection of the individual pieces.

3 The Agent-Based Modeling Checklist

I have been developing a "checklist" for creation of agent-based models. This has helped me to teach about agent-based modeling and it has been especially helpful in interacting with people who want me to write agent-based models for them.

In a perfect world, the following would occur. A scientist visits my office with an exhaustive mathematical characterization of the agents that are being studied. The scientist also has a clear model for the over-time process of community development. All I need to do is take those formulas and translate them into Objective-C and we are finished.

The world is not perfect. Almost nobody who wants an agent based model has the exhaustive mathematical characterization that is needed to write code for a simulation. My clients are generally empiricists who have concluded either that 1) the data is too hard to get, or 2) the statistical models can't manage the data. They often want a deeper, richer characterization of reality than the one that is implied by the usual statistical model. They are not necessarily against the idea that, at some point, a generalized linear model will be estimated with data, but they don't what the variables should be and they don't know what the formula ought to be.

There is the problem in a nutshell. The computer model requires absolute clarity (every variable must be explicitly set and updated) and the person who wants me to write the model is usually driven to the agent-based research strategy by the lack of clarity. The formulation of the computer model places demands on the researchers that they usually don't expect.

Thus I propose a regimen that we can impose on the process of translation between a substantive model and an agent-based computer model.

3.1 Step 1. Construct The Individual Agents

3.1.1 Make a list of variables that are internal to each individual agent

These are called the "instance variables" in computer science. Most obviously, each individual agent has two kinds of information.

1. Permanent (unchangeable) characteristics of the agent (which may differentiate that agent from other agents)

2. Individual variables that may change as the simulation progresses. This includes private, or internal, variables that may not be observable directly by other agents, as well as variables that are externally observable states (substantively, we might call these behaviors). As far as the model is concerned, all of these characteristics are just variables, information that is recorded in integers, real values, and so forth.

3. There may also be class variables, values that are simultaneously recognized and taken into account by sub-groups of agents. For example, consider an agent-based model of an army. If army X is at war with army Y, then each agent in X might have an instance variable that says "we are at war with army Y." However, it would be easier to write "we are at war with army Y" on a chalk board and have all of the agents in army X take notice of it. If a programming language does not include the possibility of class variables, we can work around this limitation. We could create instance variables, one for each agent. We would have to think very carefully about how to synchronization of instance variables of many agents. In a model with 1000s of agents, the storage ramifications are quite obvious. Instead of allocating space for each agent to keep its own copy of some commonly-held variable, we allocate space for one variable that all agents can talk notice of.

The model designer has to make a sequence of choices about information privacy. In object-oriented languages, variables can be characterized as "private" or "public." A public variable is one that other objects can easily "gather" without permission or cooperation from the object. Think of a public variable as the color of the agent's shirt, which the other agents can see without difficulty. On the other hand, some of the agent's variables should not be externally visible. A private variable can only be learned by other objects through a structured interface, a framework that the model must provide that allows information exchange.

To continue my checklist, while designing the agents, we need to consider the insertion of

4. Instance variables to record everything the agent "remembers" about

   (a) itself: the values of its instance variables as they existed in the past

   (b) other agents, either as individual agents or as summarized by recollections about groups or regions

   (c) the "world"

The agent-based model is always a dynamic model. Thus, we are studying agent instance variables that change over time. The emphasis on the emergence and maintenance of social patterns over time differentiates agent-based modeling from many traditional models in social science which presume the existence of equilibrium, rather than focusing on the achievement of it. I'd point to traditional microeconomics and game theory for evidence of that (Nash, 1951; Von Neumann and Morgenstern, 1953). There are tremendous mathematical benefits in that approach: one can bring to bear the power of mathematical fixed point theorems to argue for a the existence of a stable equilibrium by harmonizing the plans of actions of an arbitrarily large set of agents.

Because we are not usually concerned about the calculation of a particular outcome in agent-based modeling, we face a more diffuse, less-well organized modeling process. We have to make seemingly bizarre judgments about the self-awareness of the agents. The agent may need to have a model of itself, others and possibly a model of the world into which its experience may be accumulated. These records may be very comprehensive and code to maintain is difficult to maintain.

Finally, we probably need

5. An algorithm to initialize the agent's recollections.

The agent-based model begins at time 0. It "starts." What happened before that? How can the agents can behave reasonably? In the time before time 0–prehistory–we may have complicated maneuvers that create a "false history" upon which the model can continue into the future. In the Santa Fe artificial stock market (Palmer et al., 1994; Johnson, 2002), for example, the agents who begin trading at time 0 must have an accumulation of memories of past market events on order to begin trading. The model has a rather complicated series of steps to generate "false" memories for the agents, false memories that must be more-or-less internally consistent (both within the individual agent and across the various agents). I suppose that, in Bayesian MCMC models, they'd call these burn-in iterations.

3.1.2 Design methods to adjust the instance variables.

In object-oriented programming, an object (the thing that represents our agent) includes not only data, but also methods. For readers who have studied programming (such as Pascal, C or Fortran), but not object-oriented programming, the term method will be unfamiliar. A method is different from a function in the following sense. A function "floats" in the abstract, willing to receive inputs and generate outputs when it is instructed to do so. In C programming, as traditionally practiced, a function was not carried out "by" an object, it was carried out "on" a data structure. The term "method" is used (in large part) to avoid that way of thinking. A method is an action that a particular type of object can carry out. Objects send messages to one another, they exchange data and request actions. The difference between functions and methods translates into agent-based models fairly well (see the Apple Objective-C manual, Apple Computing, 2009). We think of objects, or artificial agents, as representations of people, and we ask those object to respond to requests.

All of the agent-based models, so far as I know, rely on the passage of time to allow agents the opportunity to change the values of their instance variables. It is fairly common to refer to the periodic actions that agents may perform as "step" methods. A step method may simply cause "book-keeping" to occur. An agent's age may increment by one unit with each time step. If the agent has a finite lifespan, then the step method will usually include the duty for the agent to check whether its time for removal from the simulation has come. A variety of other actions may be more-or-less automatic, such as a tree growing a certain amount when a particular amount of water is available.

The agent may be faced with decisions to update many of its instance variables, and these decisions may reflect a complicated mixture of input about the information states of other agents (or the "world" more generally). If we are working within the complexity tradition, of course, we are behooved to keep these internal machinations as simple as possible.

3.1.3 How is agent's information made "available"?

As far as the simulation design is concerned, there is no difference between an attitude and a behavior. These are just variables that exist within an agent.

The object-oriented paradigm of computer software design seeks to preserve information within separate objects. The objects reveal information and change values of instance variables only through a well-structured interface. In an object-oriented framework, we would be more likely to think of information updates as a process of "messaging" between objects. Agents send information with messages, they receive messages and take note of them. We don't allow one object to simply "change" the instance variables inside a separate object. Similarly, the variables that are inside an object might not be easily visible to the other objects, and information retrieval can be highly nuanced.

In the agent-based model, this translates into a series of design decisions.

1. Does the agent's existence in the model directly expose instance variables to access by other agents?

   Now the difference between public and private variables comes to the forefront. Can other agents simply "look" at at one agent and instantly "know" the values of some of its instance variables? Consider building a model of discussion in a cocktail party. Some participants have easily observable characteristics, possibly size, gender, and the style of clothing. Many of the other individual characteristics are likely to be private, and they are not directly observed. The party guests know their own religious beliefs, tastes in music, and favorite foods, but, at best, they can form educated guesses about the other guests.

2. Is the revelation of private information a focal point in the model? If so, think hard on the problem of private preferences and public declarations.

   An agent-based model of political protest will typically hinge on each individual agent's willingness to take a risky personal action. The protest requires agents in sequence to announce "I am opposed to our government" (Granovetter and Soong, 1988; Brichoux, 2002).

3.1.4 Step 1b. Formulate Auxiliary Agents

Researchers are usually interested in their favorite topic, whether that is citizens, voters, consumers, students, teachers, or whatnot. When we begin with step 1, we usually start with those substantively important agents. Lets call those the primary agents, the ones that the researcher intends to study. After laying out the primary agents, we usually find that a secondary set of agents will be necessary to glue everything together. I'll call those auxiliary agents.

In almost all models, we need to introduce some substantively-unimportant auxiliary agents who keep records in the simulation. In Swarm, we call that the "observer swarm," it an draw graphs as the simulation progresses. It would be a breach of expectations to put substantively important actions into the observer agents. We don't want to think of these record-keepers as substantively important actors in the model.If we wanted the observer to systematically like or on the activities of the substantively important agents.

I can provide an example about the accidental profusion of auxiliary agents in an agent-based model. In Herron and Johnson (2005), we develop represent voters and candidates in a multi-district electoral model. The voters are relatively simple, they decide which party to join, attend a caucus with other people who join that party, and then they vote for a candidate in the general election. The candidates are fairly simple, they simply declare policy proposals to the voters within a district.

But the auxiliary agents are rather more involved than one might expect. Each district needs a political party object that will hold the caucus in which district-level candidates are chosen. It is necessary to have both national level political parties and district parties because they have separate functions. In particular, the national party decides whether to pay for a campaign in each of the separate districts. In addition, there must be district-level agents that keep a list of eligible voters and records about the candidates. That district-level agent, who we call the electioneer, has to hold the elections and then report the results to the national offices, for which another auxiliary agent must be created. The national offices combine the vote totals they receive from the district election managers and then use the rules of proportional representation to select the candidates who will become members of the national legislature. In that project, I'd guess that about 30% of the computer programming effort was dedicated to the substantively important actors, the voters and the candidates, and about 70% of the effort is in the auxiliary agents.

3.2 Step 2. Construct the Environment in which Agents are Situated

3.2.1 Where do agents "live"?

Here are some possibilities.

1. Arrange the agents in a pattern of discrete "cells".

   (a) Do we think of the cells as squares (a checker board), or triangles, or hexagons, or what? The implications for interactions with "neighbors" seem obvious.

   (b) Do we insist that only one agent can be "in" a cell at one time? It is scarcely possible to imagine Conway's game of life in a grid that allows more than one agent to exist in each cell .That has obvious implications for the feasibility of movement by the agents. In the Schelling neighborhood segregation model, the dynamical process is driven by the idea that many cells are empty, creating opportunities for agents to move. a grid, that allows multiple agents to inhabit the same cell.

2. Place the agents at points in a Euclidean plane.

3. Collect the agents in an unordered"list".

3.2.2 Do agents interact?

1. Do agents "find each other?" How? Perhaps the agents are "stuck in place" on the grid and they may "look up" or "look sideways" or "look diagonally".

2. When they meet, how do agents exchange information? Can some variables be automatically gleaned by simple recognition, while others must be guessed at? Note that the internal record keeping systems of the agents must support whatever approach is selected here.

3. Are agents oblivious to the larger artificial society? Are they aware, or do they care, about fluctuations in global indicators when they make their "local" decisions?

In some models, the agents don't keep track of any system-wide indicators. These agents are completely oblivious to everything that they do not experience "first hand." Axelrod's original culture model (Axelrod, 1997a) would be one example like that, because each individual agent would simply gather information about one neighbor at one time point. Schelling's model of neighborhood segregation (Schelling, 1971) is similar, in that each agent only surveys his immediate vicinity before deciding what to do. These models of very-limited-information decision-making are certainly consistent with the complexity literature's emphasis on the cultivation of emergent properties from the simplest possible agents.

In other models, the agents do not interact with each other at all. Instead, they interact with an aggregate. In the original Santa Fe artificial stock market (Palmer et al., 1994), for example, the agents buy and sell stock. They observe the same market-wide indicators, of course, and their individual behaviors will affect the development of the market. However, the agents in the ASM never say, “hey, agent 33, what are you buying next time?” Similarly, in the El Farol model (Casti, 1996), the agents care only that they can go to a not-too-crowded bar for Irish music. They are playing against the aggregate, not other individual agents. Their happiness does not depend on whether the bar is filled up with a particular client list. They only care if there is the right number of empty seats.

Between the extremes, when agents react to information about some of the other agents, we run into some interesting philosophical issues. The challenge in these models is to explain how the agents gain the information that they are putting to use. I believe it is important to avoid “social telepathy” (Erbring and Young, 1979), the assumption that the agents somehow magically know the individual opinions of many others within the society. Perhaps a model of social protest may plausibly assert that each agent can stand at its current position and “look around” to see how many others are marching in the streets. Or perhaps the agents can watch local television to observe activities in particular locations. In most cases, however, we find models are driven by some powerful assumptions about the information that the agents can gather. For example, in the Social Impact Model (Latane, 1996), each individual decides on its opinion, but in doing so it takes into account (simultaneously) the opinions of all the other agents, putting more weight on the opinions of the ones who are close by (Latane, 1996). The authors don’t explain the process through which the agents are all made aware, simultaneously, of the opinions of all the others. A slightly more believable approach is found in the adaptation of the the Axelrod culture model by Shibanai et al. (2001). They introduce a new type of agent which might be thought of as a polling agency with a mass media outlet. The other agents can learn about the state of the society from that single global source.

3.3 Simulation Infrastructure

3.3.1 What is time?

In an object-oriented program, we might create a large number of separate objects and collect them into a list. The objects just “sit there,” doing nothing, until some message is sent to them.

In a similar vein, lets think of our agents as a collection of objects that are created at time 0. They sit, waiting, until they are told to do something. We create their reality. We create time as well. Time is usually measured by the passage of discretely experienced opportunities for action.

3.3.2 How do we schedule actions in the model?

In the Swarm Simulation System, the schedule is an “objective” thing, an infinitely long conveyor belt. On each position in the belt, the schedule joins together 1) the objects being sent messages, and 2) the messages they are supposed to be sent. The framework is based on run time binding of objects to actions; there is flexibility in the selection of particular agents who may act. The hierarchy of levels, from individual agents, to aggregates, and then to the system as a whole, is harmonized by the principle that everything has to happen sometime. Einstein’s credited with the quip, ““The only reason for time is so that everything doesn’t happen at once.”

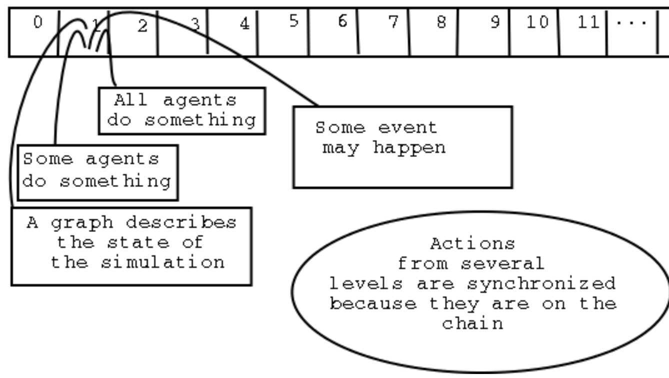


Figure 1: The Schedule

The important idea here is that agents carry out their actions at particular times, and time is strictly an ordinal concept. The actions of a whole society need to be interleaved into the time structure of the model

The design of a model's schedule requires us to ask a number of questions. Does each agent "step" every time period? Perhaps we should just pick one agent at random to step every time? Do we want to randomly shuffle the list of agents at each time step? Conceptually, do we believe that the world is a "frozen snapshot" in the mind of each agent when the agents are deciding what to do? If so, we are proceeding as though all of the agents act "at the same moment?" That kind of a model is easier to code, but it may not be as realistic. In contrast, perhaps that each agent has completely up-to-date information when its time to act arrives. In this view, each agent's action is instantaneously made available to each of the others.

Is the schedule flexible? A static schedule may require each agent to "step" in every time period, but perhaps we need something else. In Swarm, schedule can adapt to substantively important actions. Perhaps nothing happens most of the time, but then random disturbances are introduced that cause some agents to take steps, which cause others to take steps in the future. This is called dynamic scheduling, and we speak as though agents "put themselves on the schedule" at particular points in the future.

3.4 What data do we need during and after simulation?

Research projects will differ in their conceptualization of the record keeping problem. In my experience, there are at least two choices. First, the data to be collected might be wholly individualistic in nature, or it may be an aggregated characteristic of the agent society. Second, the way that we design data collection differs by whether we think of data as emerging from the individual agents, or whether we have an some functions that *steal* their instance variables and summarize them. This is summarized in Table 1.

On the left, I divide the data to be collected into 2 types. One is individualistic, meaning it is data reported by the agents in the model themselves. This data may be regarded as "subjective," in the sense that it only reflects the agent's actual experience of the world. The

Table 1: Record Keeping

		Record keeper	
		internal	external
Data to be collected	individualistic	door-to-door opinion survey	involuntary DNA collection
	aggregated	unemployment rate, social entropy	Relocate individual data into a communal record system

other data type, which I call "aggregated," is information that the program calculates about the agents. There are almost always some societal measurements that individual agents are not able to provide on their own. For example, one cannot learn the unemployment rate from interviewing particular people. It is necessary to interview a lot of people and then synthesize the information into an unemployment rate.

Next, how do we think of the record keeper. Is the record keeper given "super powers" to look inside all of the agents and tabulate their status? Is the record keeper forced to interact with the agents through the same, limited, interface that the agents follow to interact with each other? We can think of that record keeper as a surveyor who collects information on the phone, asking the agents for their information. Sometimes researchers want more information that, to have different or more accurate information than the agents would be willing to provide to each other. In simulations, the difference between these modeling strategies is readily observed in the way that the code renders individual agents in graphic displays. Does the simulation tell the individual agents to "draw themselves" on the landscape, or does it imbue the landscape with the ability to draw the agents (with or without their permission).

There may be a temptation to re-design a simulation in order to facilitate easier record keeping. In almost all projects there is a contradiction between agent privacy and the speed of computations. Usually, the substance of the problem would require us to have some sort of separate object–a surveyor, a government agency, or such–that surveys individuals and finds out about them. That would be most consistent with the idea of "information hiding" and the privacy of the agents. It is also the most realistic method, in the sense that the "real world" does it that way. On the other hand, a model will run more and more slowly as this record keeping is going on. The agents have to constantly answer survey questions. To obtain speed of execution, one is tempted to bend the principles a little bit so that the important information is kept in some public, easily tabulated, format. I know of at least one simulation that stores the individual agent information in a data base and the agents have to ask the data base for their own information when they need it!

In the model described in *Political Disagreement* (Huckfeldt et al., 2004), we have individual agents who have their own opinions and they have personal records about other agents they have encountered. There is also a more-or-less objective aggregated record of the opinions of all agents at a given moment. We never allow a meta-agent to simply glance through the private data of the agents, however.

In our model, the conflict between speed and agent privacy was navigated as follows. Each agent has an opinion vector, but most opinions are not changing most of the time. We found that surveying each of the agents at the end of each time interval was extremely time-consuming. There was, however, a way to accelerate the calculations. Almost all the time, the surveyor would ask the agent, "what are your opinions," and the agent would respond with the exact same opinion string that was reported in the previous time. To speed the calculations, a centralized tabulator object was created and it maintained records on the opinions of the agents. The graphs that are displayed when the model runs are drawn from the data in the tabulator. When agents change their opinions, they (of course) update their private variables, but we also created a method through which they notify the tabulator. As long as the number of agents who change their opinions is not huge (say, above one-half of the artificial society), this system of double-record keeping is efficient.

There was a similar tension between realism and computational speed in the political protest model (Brichoux, 2002). Each agent in that model lives at a point on a Euclidean plane, and the agent can view the activities of agents within a certain radius of vision. We found that the model ran very very slowly if each agent had to individually survey the state of the society. Each agent had to inquire of each of the other agents if it was protesting at a given instant. It is much faster, if less realistic, to retain a centralized record of agent actions and then ask each agent to report its changes to that central registrar. An even faster result was obtained by pre-aggregating agent activities into a social record keeping system that allowed the model to quickly answer this question for each agent. "If I am at location (x,y), how many other protesters can I see within a radius r." Since many agents were asking the same–or a very similar question–there were efficiencies of scale to be had by making the calculation about trouble at (x,y) only once, and re-distributing that information to the all of the agents who need to make the same calculation. One could try to provide a substantive justification for this communal information collection, perhaps by calling it a television station or such, but one has to strain quite a bit to accept such an argument. It is a pure-and-simple redesign for the speed of the simulation.

Some researchers want to have details saved on each agent's instance variables at each time point. That will make a model run more slowly and require more disk storage, of course. However, it may open up interesting possibilities that fall under the heading of "simulation serialization." We could use the saved data from any particular time point to re-start the model and subject it to new external influences. In the political disagreement model, that approach was used to generate quite a few interesting figures which demonstrated some conditions under which perturbations of the model might produce social behavior fluctuations.

4 The Inevitable Follow-Up Questions

In this section, some of the follow-up questions that arise most frequently are considered. I am tentatively proposing the following idea. There is a natural tension between the computer modeler and the substantive researcher. The computer modeler has a collection of routines for particular purposes and the substantive modeler has a few notions about what agents might be and how their interaction might unfold. Translating from the researcher's idea into a computer model forces the substantive researcher to become interested in (or pretend to be interested in, more likely) a series of specific modeling decisions. On the other hand, the program developer is supposed to try to represent the researcher's ideas in code and then clear up the additional decisions that are necessary to make the computer program run successfully.

Almost always, it happens that the practice of writing the computer model calls us to fill in a lot of empty spaces in our theory. Our theory usually describes the "interesting" parts of the social process that we think might be going on, but the act of tying the pieces together in a computer model usually creates a raft of design choices. I'm calling these additional decisions the "follow-up questions" and I'm concerned about the danger that they might derail the modeling exercise by drawing an excessive amount of attention to seemingly irrelevant detail. I'm especially worried that the seemingly irrelevant details may play a driving force in the model's behavior without the accommodating re-conceptualization of our theory.

4.1 Design and the belt of auxiliary assumptions

Follow-up questions are inevitable. A project begins with a "pencil and paper sketch" of the parts that are substantively interesting to the researcher. Programmers would like this sketch to be as detailed as possible, including a list of the variables in the agents, the structure in which they interact, and the record keeping system to be used, and so forth. That kind of detail is usually lacking at the outset, but even if it is available, it is never sufficient for development of computer code. The computer code has to be built up bit by bit.

The best programmers I know still follow the advice that is emphasized in introductory programming: add small pieces, recompile, make sure it runs, and try to understand the effect of what has been changed. Superficially, of course, this is about avoiding programming errors (bugs). But there's a more important reason. This is an agent-based modeling exercise. We started with the idea that we have a complex system, an on-going series of information exchanges that can (and will, we hope) produce unintended effects (emergent properties). If a model is built up in stages, it is much more likely that we will be able to understand emergent properties (and differentiate them from flaws in the code).

This is an exciting, but dangerous part of the agent-based modeling exercise. On the one hand, we want to keep our focus on the part of the model that is truly interesting to our substantive research. On the other hand, we need to be faithful to our audience. We want to be able to say to our readers that a system with certain characteristics behaves in certain ways, and we should not present a model whose overall behavior is driven by an unmentioned, obscure detail in computer code. Thus the research process has difficulty balancing the emphasis we place on the research topic and the auxiliary details.

The problem that we are "distracted by auxiliary details" is not peculiar to agent-based modeling. It seems to be inherent in the practice of scientific research itself. The theories that we are interested in usually have a small core of ideas and formulas, and also a possibly wide belt of *other stuff* that connects them together and to our observations (Hempel, 1966). We don't care about the *other stuff* so much, we wish we didn't have to put up with it. But we need it, without it our idea*s are obviously wrong. With the other stuff*, our predictions are less obviously incorrect. Kuhn argued that as theories accumulate more and more of that *other stuff*, they seem less and less interesting and useful, so that eventually they reach a point of crisis in which their value is outweighed by their inconvenient, cumbersome apparatus (1962). These increasingly ill-suited theories can be displaced by new, simpler characterizations of the central ideas that, at least at first, are guarded by smaller belts of tedious, inconvenient, arguments. As such, new theories don't necessarily fit the data better, they are just more satisfying, usually on an abstract level, partly because they are not so full of *other stuff* (Toulmin, 1961).

As a result of the fact that a modeling strategy that ads more and more of the *other stuff* is preparing itself for an internal crisis and eventual rejection, we need to be cautious. It seems to me agent-based modeling makes it particularly easy for us to throw in auxiliary appliances that seem necessary, but are irrelevant to the substance at hand. We are able to see the problem more clearly in agent-based modeling because the computer programming framework does not structure our choices. Programming allows a nearly infinite-dimensional space of auxiliary assumptions that can be brought to bear.

4.2 Do you really want us to write n^m computer programs?

Every simulation project will, *with certainty*, run up against the research boundaries described by Bankes (1993). Suppose we come to a design point at which there are 3 choices, none of which is substantively more appealing than the others. The only responsible choice is to work out all 3 options, to literally write 3 computer models, to find out if any of these apparently indifferent choices imply a difference. Thus we have allowed the auxiliary design issue, about which we had no opinion at the outset, to become part of our research problem.

The work on the auxiliary details compounds, like interest on a bank account. While we are working on the first 3 models, we will come to a second decision point that presents 3 more choices. Being responsible scholars, we decide to write those three models. And since this choice may affect our conclusion on the previous decision, we find ourselves wading through $3^2 = 9$ models. After just a few of these decision points, we quickly find ourselves in an untenable situation, trying to implement n^m different models.

I'll call this the n^m problem. Either we devote a huge amount of effort to experiments designed to help us choose among auxiliary assumptions, or we ignore the issue and proceed as if we did not notice the problem in the first place. The first option is commonly self-destructive, while the latter is dishonest. We would like to steer a course between the extremes, on in which we rigorously evaluate alternatives that are likely to make a difference, while while we honestly admit that we recognize, but not pursue, all of them. At the current time, however, we have only weak guidance in separating the real problems from the epiphenomena.

I can point quite a few examples of this problem. Consider the public opinion models. The substantive purpose is to study public opinion via social influence. When I started writing agent-based models, the leading models used a checker board analogy to lay out the agents on a square grid of square cells (Latane, 1996; Axelrod, 1997a). The substantive issue, the part that truly interests us, is the way that agents decide their own opinions after taking notice of the opinions of others around them. All of the rest is *other stuff*.

We've already put a big pile of *other stuff* in the middle of our project. The square grid, by itself, is a huge auxiliary assumption. Do we really want a model with agents that are evenly spaced across a rectangular array? Its not realistic, but it may be useful. After that, then we have to make a lot of other auxiliary modeling decisions that arise simply because of our original decision to place the agents on a grid.

1. Should we use von Neumman or Moore neighborhoods: Should the agents find neighbors by looking up, down, left, and right or can they also look diagonally?

2. Should we limit the agents to interact only with the others that are immediately adjacent to them? What if some have a larger radius of vision than that?

3. If an agent can interact with agents are further away in the grid, should the opinions of far-away others be given less weight? If so, what might be a good formula for the effect of distance?

4. What about the agents who are positioned on the edge of the grid? Are they on the edge, with fewer opportunities for interaction than the other agents? Perhaps we need to redesign our world, think of the grid as a torus, with edges that wrap around so that there are no agents on the edges.

It seems irresponsible at the current time to develop a grid based model that does not allow for different types and sizes of neighborhoods. I don't know when, or if, the neighborhood type is likely to affect the simulation, but I expect it might. Thus we should have to design it both ways This is frustrating because code has to be designed in an ever more complicated maze. One that implements agents with a variety of neighborhoods, and worlds that wrap or don't wrap around.

After we have finished with those ad hoc modeling decisions, then we are confronted by *more other stuff*. Should the agents update opinions synchronously (meaning that they all update their variables at the same instant)? Synchronous updating is the classic approach in cellular automata, such as Conway's Game of life (Gardner, 1970). Quite a few of the models that generate interesting fractal patterns are based on synchronous updates (Wolfram, 2002). In the simulation model, synchronous updates are implemented as a "double buffered" grid: the agents have access to the "old" state of the world, and they register their new opinions on the "new" state of the world. After all of the agents have made their changes, then the "new" grid becomes the "old" grid and the situation is repeated.

Is the choice between synchronous or asynchronous updating an important choice? Many of us suspect that the answer is yes, but we don't know for sure. There is a famous paper which seemed to indicate that synchronous updating can substantially distort our understanding of simulated spatial iterated prisoner's dilemma (Huberman and Glance, 1993), and then an equally famous rebuttal which seems to indicate that the difference is trivial, usually (Nowak and May, 1992; Nowak et al., 1996). The whole problem seems like an afterthought, and it is. It is an artifact of our conceptualization of the way that agents are connected together.

Suppose we throw away synchronous updating altogether. What should we do? Axelrod placed his agents evenly among the cells of a grid, but did not update them all at the same instant. He proposed to randomly choose one agent and allow it interact and adjust, supposing all of the others are fixed at the same moment (1997a). Epstein and Axtel replicated that model, using the seemingly equivalent assumption that the list of agents would be shuffled at each time point. The list was traversed in order, allowing each agent to act according to the exact same logic as was used in Axelrod's model. To their surprise, there were differences in the long-run behavior of the model because the "shuffled list" approach never allowed one agent to have more chances to step than the others (Axtell et al., 1996).

The fact that we have made a seemingly endless series of important decisions about the computer model for reasons that are completely external to the substantive research questions is the major problem. If the research problem had somehow been posed to us as a controversy about whether people speak to their neighbors in a strictly north and south pattern, for example, then these things would be relevant. On the other hand, if we want to study political influence and persuasion, then these decisions are truly auxiliary. They are other stuff we can easily throw away without a second thought.

In the best case scenario, we should stop and re-think our theory. If the computer model's development is really dominated by this string of seemingly irrelevant decisions, perhaps the problem is not in the computer model, but rather in the theory itself. The theory is lacking in detail in some important ways, and that void is filled by computer implementation ideas, not substantive research ideas. That is to say, we hope we will are sufficiently self-aware so as to recognize the proliferation of ad hoc decisions and then we should think more clearly about the theory that necessitates them.

In the political disagreement model, we were plagued by a series of these problems because we started with "agents smeared evenly on a square grid". After a long list of computer modeling questions, Bob Huckfeldt wondered out lout, "why are we having to bother with all of this?" To guide the decisions about the agent neighborhoods, we tried to make substantively motivated design. Substantively speaking, our topics–humans–to move from one context to another (from home, work, church, softball). People are not spread evenly across the terrain, some places are more crowded than others. People may sometimes occupy a common meeting area. That model adopted various implementations of a decentralized scheduling framework, dividing the agent's day into a fixed series of time steps and allowing agents to initiate interactions with others that they encountered (this is explained more fully in the Political Disagreement book, but the most complete explanation is, as always, in the source code itself: `http://pj.freefaculty.org/Swarm/MySwarmCode/OpinionFormation`).

I don't think we solved the problem of the auxiliary assumptions, but rather we tried to answer some of them with principles that we were willing to put within the core of the theory that we were developing. I'm certain we did not avoid the n^m problem entirely. A scan through the code still reveals a lot of re-designs for exploration of special cases. My notes indicate that, during a 6 month stretch in 2002, I was preoccupied with three different code implementations of the dynamic scheduling framework. We wanted the agents to move about autonomously, but there were several competing ideas about how that should be implemented in a complex system. There is a power point presentation about this (`http://pj.freefaculty.org/ResearchPapers/Presentations/Swarmfest02/DynamicScheduling`) with example code, of course.

4.3 When will the simulation end?

Suppose the program is finished and the model runs. How long should we wait for our results? How many timesteps should we monitor?

The answer depends on a host of factors. There are two cases in which we have an easy answer. First, the topic being considered calls for a model that is finished after a certain number of steps. Most modeling projects don't dictate, a priori, a finite number of steps, but there are a few that might. A model of a series of weekly political primaries, ending with a general election, describes a finite sequence of events. An agent-based model of a football game would have a finite length Casti (1997). Perhaps an even simpler example would be a simulation of a television contest, such as the *American Idol*. One of my students wrote an MA thesis that about an agent-based model of political nominations following the *American Idol* format that sequentially eliminated the contenders (Struemph, 2006).

Second, if the model exhibits a tendency to stabilize to a fixed pattern, an equilibrium configuration, then we also do not have much trouble. The model grinds to a halt on its own term: our agents stop changing, the environmental variable become constants. The political opinion models generally do stabilize into a locked pattern of agreement and disagreement, some more quickly than others (very quickly: Latane (1996); less quickly: Axelrod (1997a); and slowly: Huckfeldt et al. (2004)). The Schelling segregation model does generally reach a stable pattern (Schelling, 1971; Zhang, 2011).

If our modeling exercise falls into either of those two categories, we can feel fortunate. We are relieved of the need to make a number of other decisions about how much data to gather and how to simplify it for a presentation. These models make me uncomfortable; it is more difficult to persuade the audience that the implications of the model are relevant if we have to equivocate over the question of whether we have observed the model for enough timesteps. If a model reaches a stable point, and shows no sign of budging (e.g., Johnson, 1996), the presentation is much more manageable. Many of the problems of reporting and presentation are solved automatically. If we run the model a thousand times, then a histogram or other summaries of the outcomes may convey the important message.

What do we do if the simulation model does not grind to a halt? Without a computational stable point, the research problem is open-ended and difficult to present to readers. We have no good way of knowing if we have enough iterations with a run of the model. If repeated runs indicate that the model does not track to a particular kind of outcome, then we worry that we have not run enough different re-starts as well. I wouldn't want to be on the receiving end of a critique like Binmore's review of *The Complexity of Cooperation* because my simulation was terminated too suddenly (Binmore, 1998).

This is not a purely idle concern, since a very large segment of the agent-based models do not exhibit fixed behavior patterns over the long run. Models of bees, ants, bit-forecasting stock market investors, do not stabilize to a pattern of fixed individual behavior. However, they do seem to stabilize to an understandable pattern that can be characterized as a statistically stable distribution. I've not seen the diagnostic tools for convergence of Markov Chains (from Gibbs sampling in Bayesian statistics) applied to these models, but it might be fruitful to do so. We would, at least, have a rigorous, well documented set of arguments that would justify the collection of a sample of, say, the last 2000 timesteps from a simulation upon which to make our analysis.

Some models seem to have neither a tendency to form stable individual patterns of behavior nor stable aggregates (Brichoux, 2002). Many of the agent-based simulations in artificial life, particularly in the "edge of chaos" literature (Langton, 1990; Kauffman, 1993), are designed with the idea that the transformation matrix is not a fixed probability model, but rather it is an adaptive process that may never stabilize. In these cases, even the Bayesian tools will not afford us any insight. The strong theoretical results about the convergence in

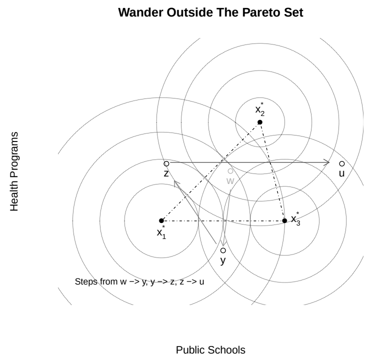


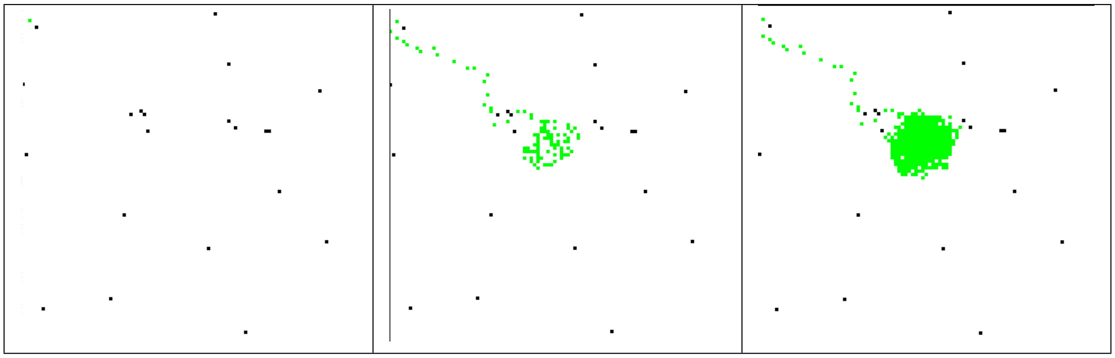
Figure 2: Instability of Majority Rule

distribution of a Markov chain Monte Carlo (MCMC) simulation presuppose that the system is in a stable stable.

Most of the influential social science theories are oriented to producing stable equilibrium distributions. The mainstream social scientists are not so interested chaos theory, for example. There is one exception, however, in mainstream political science. This is the problem of the generic instability of majority rule in multidimensional policy spaces (McKelvey, 1976; Schofield, 1978). A small example with just three voters on a two dimensional space is presented in Figure 2. There is generally no point is safe from defeat by that majority rule; majority rule will wander. If we choose proposals carefully, we can lead a majority wherever we want to go (in this case, we can hop from w to y, and then y to z, and z to u, in a sequence of majority approved proposals. Captivated by the apparent illogic of majority rule, many scholars have tried to understand conditions under which majority rule is likely to be the most incoherent (for a review, see Johnson (1998)).

In the shadow of that burgeoning literature on the instability of majority rule, I wrote an agent-based simulation model that randomly assigns voters to positions in the two dimensional space. We allow proposals to be put to the audience completely at random, and the voters accept of reject each proposal strictly on the grounds of whether or not the proposal moves policy closer to the agent's most preferred position (`http://pj.freefaculty.org/Swarm/MySwarmCode/MajorityRule`). A snapshot of one of these models is displayed in Figure 3. The left panel displays the initial state, with the policy proposal in the top left. Proposals emanate from a random process that we can adjust. We can control the length of the jump in policy that is allowed, and we can regulate the range of motion allowed to the proposer. This particular run allows the proposals to be drawn from a 180 degree angle that is centered on the most recent winning proposal (the agenda proposal process cannot turn backwards, that it to say. At most it can turn 90 degrees to the left or right, but the choice within that range is uniformly random.

This model never reaches a stable point, but it does not wander so widely as we might have expected. In this particular run, the random proposal process finds majority support for change about 22% of the time. The striking thing about the model, however, is that it can very seldom find two randomly chosen points in a row that lead away from the center. This model of the agenda setter encourages the proposer to keep wandering away from the



Time 0. 21 voter ideal points (black) 1 policy position (green)	Time 500. Green points are majority rule winners	Time 5000.

Figure 3: Majority Rule Simulation

middle by allowing proposals only in the half-space that is "in front" of the agenda setter (facing in the direction of the most recently approved change). Finding a string of randomly chosen points that wander widely, as illustrated in Figure 2, would be about as likely as the proverbial monkeys typing the Bible. That is to say, not very likely. This was proven formally for a specific random model of the proposal mechanism in Ferejohn, McKelvey, and Packel (1984).

Although the simulation may convey some visual suggestion, it is far from a rigorous mathematical proof. An effort to establish the stability of majority rule purely on the basis of the agent-based model would be an uphill battle because readers would always be suspicious that the model ought to be allowed to run longer, or that we lack a rigorous way to characterize observed instability. The simulation allows us to tweak the proposal and decision process in ways that are impossible in the mathematical analysis, however. I have often wished I could travel back in time to write simulations for that research team, so as to make to power of their argument more visually compelling.

4.4 Should we re-design a stable simulation by inserting noise?

Suppose we want to model an on-going (not time bound) social process that, empirically speaking, does not reach a stable point. The simulation model, however, "grinds" to a halt. That model is tractable, and possibly useful, but is it realistic? If we want to keep using the simulation model that grinds to a stop, we have some rhetorical devices at our disposal with which we try to finesse that problem. We argue, for example, that a model is an isolated example of a pristine system that operates under idealized conditions. We recognize that, in reality, many social processes are not allowed to run until they are completed. Instead, they are subject to disturbances of many sorts, and thus reality never equilibrates the way our model does. The study of the "ideal type" model may still be useful.

We might conclude, instead, that the simulation model is "just wrong." In order to prevent it from locking at a certain stage, we have to make some changes that make the model less stable. We might throw in exogenous randomness.

That has been a discouraging realization for me. We are going to intentionally obscure the underlying tendencies of the model we have prepared. It has been such a discouraging thing that I have avoided it, for the most part, by ignoring it.

Misery loves company, and recently I have realized that this same problem arises in empirical research as well as in game theoretic modeling. Consider each in turn.

4.4.1 Mixed effects regression analysis

Suppose a model makes a clear prediction:

$$y_i = \beta_0 + \beta_1 x_i.$$

That model will almost always be instantly rejected. We almost never gather columns of x_i and y_i data that are perfectly aligned. This model is obviously wrong.

It is necessary to infuse an error term, some noise, so that the model is not so easily rejected. This error, e_i is, to many of us, just unmeasured *other stuff* (that is very pleasantly behaved, with an expected value of 0 and fixed variance):

$$y_i = \beta_0 + \beta_1 x_i + e_i.$$

This model is not so obviously wrong as the first one. So far, the auxiliary assumption involves the simple assertion that the effects besides x_i are well behaved.

Now suppose we apply that model to data, but there appears to be trouble because some of the outcomes have more dispersion in them than is implied by the model. Historically, we called heteroskedasticity and developed a set of tools (weighted least squares, etc) to try to manage it. Our auxiliary hypothesis about e_i has to be made more ad hoc. We suppose different random processes are at work and then redesign the model accordingly. Witness the development of regression models for count data and the debate over whether the outcome is better represented by a Poisson or Negative Binomial distribution (Long, 1997; Cameron and Trivedi, 1998). One model, the simple one, is obviously wrong; the more complicated model is less provably wrong. All in all, this seems like a distracting exertion of effort on *other stuff*.

The way we view heteroskedastic regression is changing, however, because new perspectives are helping applied researchers transform the "corrections for heteroskedasticity" into substantively meaningful components in social theory. The use of clustered random errors and frailty in survival analysis would be primary examples. Random effect regression models lead us to incorporate cluster-specific random effects into the core of a new type of social theory. Membership within a school or community may have a "contextual effect" that is not directly measurable. Clusters cause heteroskedasticity, but that's not the main point anymore (Goldstein, 1995; Pinheiro and Bates, 2000; Raudenbush and Bryk, 2002). Hence the rise of the contextual model. Another difference is found in the way additional randomness is incorporated in survival analysis (for a nice discussion, see Congdon (2006)). Frailty is individual-level randomness, inserted on the substantive grounds that two people in the same situation do not always behave in the same way. Instead of talking about heteroskedasticity and correcting for it, we talk about how to design a theoretical model that generates "over dispersion" in order to more closely match observed data.

4.4.2 Insertion of randomness in game theory

In game theory, a model may make a definite prediction about human behavior that is not consistent with observation. There have been a variety of efforts to deal with the problem, but the approaches that I'm referring to in this section are trembling hand and quantal response equilibrium models.

Noticing that some equilibrium points in games were not empirically plausible, Nobel Prize winner Reinhard Selten suggested that we might impose some mandatory randomness in agent behavior Selten, 1975. A little bit of randomness may eliminate some of the most unrealistic, least plausible predictions of these models. Tn the quantal response equilibrium approach to game theory, McKelvey and Palfrey, 1988 propose the same idea write large. These changes seem, at least on the surface, to be a dry and uninteresting as corrections for heteroskedasticity. However, in political science, there is a growing literature associated with the idea that the models imposing some intrinsic randomness are more consistent with the facts and that we can attempt to integrate this randomness into theories of international politics (Signorino, 1999).

4.4.3 Plausible randomness in social theories and agent-based models.

A complex system is often defined as a collection of autonomous agents who are only loosely linked together. The characterization "loosely linked" means that the agents are not directly in "control" of one another. Instead, the agent behaviors reflect a large number of seemingly unpredictable small influences that they encounter. An agent-based model may seem random, even though it is not ("deterministic randomness").

The computer code for a complex system model is, almost by definition, complicated. We sometimes have difficulty believing that a result is correct–we worry that some flaw in the code my cause a pattern to emerge. We have difficulty feeling sure that the theoretically relevant settings in the model are responsible for a given change in its behavior. To avoid that confusion, and to diminish the n^m problem, we are urged to "keep it simple, stupid" (Axelrod, 1997b). In particular, we probably should not design extremely complicated agents. We should not gratuitously insert random variables because the model should–if things work right–already have a lot of uncertainty in it. Add to that the "something for nothing" research strategy that leads to interesting models of bee hives and ant trails.

In most agent-based modeling projects, we will arrive at a point where the substantive researcher says, "I did not mean to say people always behave in a certain way. I meant to say they usually do." Thus the code has to be redesigned so that an agent usually does one thing, but not always. As long as these changes are motivated by the substance of the social theory being investigated, these changes should not be discouraged. They were simply an oversight in the initial design of the agents.

However, it often happens that we design models that grind to a halt, and they do so because our theory was not well considered from the start. The social science theories we usually hold describe one or two choices in a short time span. Most of them are not intended to describe hundreds or thousands of decisions that humans make on a day-to-day basis. Let's consider the agent-based model of political protest . Usually, we start thinking about this from Granovetter's threshold perspective (Granovetter, 1978; Granovetter and Soong, 1988). People may be unhappy "on the inside," but they do not show it on the outside. If some people express their unhappiness, that may encourage others to do the same, and the aggregate level of discontent gradually "tips" the society toward a state of rebellion. A simulation that plots the behaviors of the agents over time will generally accumulate agents who are willing to show their discontent on the outside.

The fact that the model grinds to a fixed state of widespread protest should probably be considered a flaw. We need to introduce some component at the individual level that will make the outcomes more realistic. One change that is required is individual frailty, a "maybe I will or maybe I won't" sort of randomness that is simply thrown into the model. This is note a completely ad hoc decision, however, because it has a real-life referent: protesters may become "exhausted." If we change the model so that each individual agent has a personal characteristic that determines the number of consecutive days that the agent is willing to protest, then we can arrive at a more realistic model in which there are occasional flare-ups of political discontent (Brichoux, 2002).

5 Conclusion

This essay has presented an overview of the agent-based model building process. It began with a characterization of the agents and the agent based model. A checklist for the creation of agent-based models was offered and a number of specification issues in agent-based modeling were considered.

One of the creative sources of friction in the agent-based modeling process is the mismatch between patterns that exist in the observable world and patterns that arise in the computer simulation. One contribution of this paper is a taxonomy of the mismatches that may arise and some hits about the conditions under which they are likely to be important. Sometimes we are not concerned when the model's aggregate behaviors do not match our observations because we view the model as an ideal type system operating in an enclosed environment.

On the other hand, there are cases of concern when models generate results that are grossly out of line with our empirical reality. Rather than adjusting the agent-based model in a wholly ad hoc way so as to force it into line with our empirical expectations, I would urge modelers to strive for the cultivation of substantively meaningful changes in the behavior of individual agents that might lead to changes in the aggregate behavior of the model.

References

Apple Computing. 2009. "The Objective-C Programming Language." http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/ URL `http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html`

Axelrod, Robert. 1997a. "The Dissemination of Culture: A Model with Local Convergence and Global Polarization." *The Journal of Conflict Resolution* 41(2):203–226.

Axelrod, Robert M. 1997b. *The complexity of cooperation: agent-based models of competition and collaboration*. Princeton studies in complexity, Princeton, N.J: Princeton University Press.

Axtell, Robert, Robert Axelrod, Joshua M. Epstein, and Michael D. Cohen. 1996. "Aligning simulation models: A case study and results." *Computational and Mathematical Organization Theory* 1(2):123–141.
URL `http://www.springerlink.com/content/h2371t226h133785/`

Bak, Per. 1999. *How Nature Works: The Science of Self-Organized Criticality*. Springer, 1 ed.

Bankes, Steve. 1993. "Exploratory modeling for policy analysis." *Operations Research* 41(3):435–449.
URL `http://portal.acm.org/citation.cfm?id=158055`

Barnsley, Michael F. 1993. *Fractals Everywhere*. Morgan Kaufmann Pub, 2 sub ed.

Bonabeau, Eric, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm Intelligence from Natural to Artificial Isystems*. New York: Oxford University Press.

Bonabeau, Eric and Guy Theraulaz. 2000. "Swarm Smarts." *Scientific American* 282(3):73–79.

Brichoux, David; Johnson. 2002. "The Power of Commitment in Cooperative Social Action." *Journal of Artificial Societies and Social Simulation* 5(3).
URL `http://jasss.soc.surrey.ac.uk/5/3/1.html`

Cameron, Adrian Colin and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. No. no. 30 in Econometric Society monographs, Cambridge ; New York: Cambridge University Press.

Casti, J. L. 1997. *Would-be worlds: how simulation is changing the frontiers of science*. Perseus Books Group.

Casti, John. 1996. "Seeing the Light at El Farol." *Complexity* 1(5):7–10.

Congdon, P. 2006. *Bayesian statistical modelling*. John Wiley & Sons.

Epstein, Joshua M and Robert Axtell. 1996. *Growing Artificial Societies Social Science from the Bottom Up*. Washington, D.C: Brookings Institution Press.

Erbring, L. and A. A. Young. 1979. "Individuals and Social Structure: Contextual Effects as Endogenous Feedback." *Sociological Methods & Research* 7(4):396–430.

Ferejohn, J. A., R. D. McKelvey, and E. W. Packel. 1984. "Limiting distributions for continuous state Markov voting models." *Social Choice and Welfare* 1(1):45–67.

Gardner, Martin. 1970. "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game "Life"." *Scientific American* 223:120–123.

Goldstein, Harvey. 1995. "Hierarchical Data Modeling in the Social Sciences." *Journal of Educational and Behavioral Statistics* 20(2):201–204.
URL `http://www.jstor.org/stable/1165357`

Granovetter, Mark. 1978. "Threshold Models of Collective Behavior." *American Journal of Sociology* 83(6):1420–1443, ArticleType: research-article / Full publication date: May, 1978 / Copyright Â© 1978 The University of Chicago Press.

Granovetter, Mark and Roland Soong. 1988. "Threshold Models of Diversity: Chinese Restaurants, Residential Segregation, and the Spiral of Silence." *Sociological Methodology* 18:69–104.

Heinrich, Bernd. 1981. "The Mechanisms and Energetics of Honeybee Swarm Temperature Regulation." *Journal of Experimental Biology* 91(1):25–55.

Hempel, Carl G. 1966. *Philosophy of Natural Science*. Foundations of philosophy series, Englewood Cliffs, N.J: Prentice-Hall.

Hiebeler, David. 1994. "The Swarm Simulation System and Individual-Based Modeling." Tech. Rep. 94-11-065, Santa Fe Institute, Santa Fe, NM.

Holland, John H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Complex adaptive systems, Cambridge, Mass: MIT Press, 1st MIT press ed ed.

Holland, John H. 1998. *Emergence: From Chaos to Order*. Reading, Mass: Addison-Wesley.

Holland, John Henry. 1996. *Hidden order: how adaptation builds complexity*. Basic Books.

Huberman, B A and N S Glance. 1993. "Evolutionary games and computer simulations." *Proceedings of the National Academy of Sciences* 90(16):7716 –7718.
URL `http://www.pnas.org/content/90/16/7716.abstract`

Huckfeldt, R. Robert, P. E Johnson, and John D Sprague. 2004. *Political Disagreement: The Survival of Diverse Opinions Within Communication Networks*. Cambridge, UK: Cambridge University Press.

Jacob, Christian J., Gerald Hushlak, Jeffrey E. Boyd, Paul Nuytten, Maxwell Sayles, and Marcin Pilat. 2007. ""SwarmArt:" Interactive Art from Swarm Intelligence." *Leonardo* 40(3):248–255.

Johnson, Paul E. 1996. "Unraveling in a Variety of Institutional Settings." *Journal of Theoretical Politics* 8(3):299 –330.

Johnson, Paul E. 1998. *Social Choice: Theory and Research*. Thousand Oaks, CA: Sage.

Johnson, Paul E. 2002. "Agent-Based Modeling: What I Learned From the Artificial Stock Market." *Social Science Computer Review* 20(2):174 –186.

Kauffman, Stuart A. 1993. *The Origins of Order: Self Organization and Selection in Evolution*. New York: Oxford University Press.

Kauffman, Stuart A. 1995. *At Home in the Universe: The Search for Laws of Self-Organization and Complexity*. Oxford University Press US.

Kuhn, Thomas S. 1962. *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press.

Langton, Christopher G. 1986. "Studying Artificial Life with Cellular Automata." *Physica D: Nonlinear Phenomena* 22D(1-3):120–149.

Langton, Christopher G. 1990. "Computation at the Edge of Chaos." *Physica D* 42:134—144.

Langton, Christopher G. 1995. *Artificial Life: an Overview*. Complex adaptive systems, Cambridge, Mass: MIT Press.

Latane, Bibb. 1996. "Dynamic Social Impact: The Creation of Culture by Communication." *Journal of Communication* 46(4):13–23.

Long, J. Scott. 1997. *Regression Models for Categorical and Limited Dependent Variables*. No. 7 in Advanced quantitative techniques in the social sciences, Thousand Oaks: Sage Publications.

Mandelbrot, Benoit B. 1983. *The fractal geometry of nature*. San Francisco: W.H. Freeman.

McKelvey, Richard D. 1976. "Intransitivities in multidimensional voting models and some implications for agenda control." *Journal of Economic Theory* 12:472–82.

McKelvey, Richard D. and Thomas R. Palfrey. 1988. "Quantal Response Equilibria for Normal Form Games." *Games and Economic Behavior* 10(1):9–41.

Miller, John H. and Scott E. Page. 2007. *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton University Press, illustrated edition ed.

Mitchell, Melanie. 2009. *Complexity: A Guided Tour*. Oxford University Press, USA.

Nash, John. 1951. "Non-Cooperative Games." *The Annals of Mathematics* 54(2):286–295.

Nowak, Martin A., Sebastian Bonhoeffer, and Robert M. May. 1996. "Robustness of cooperation." *Nature* 379(6561):126.
URL `http://dx.doi.org/10.1038/379126a0`

Nowak, Martin A. and Robert M. May. 1992. "Evolutionary games and spatial chaos." *Nature* 359(6398):826–829.
URL `http://dx.doi.org/10.1038/359826a0`

Palmer, R.G., Brian W. Arthur, John Holland, Blake Lebaron, and P Tayler. 1994. "Artificial economic life: a simple model of a stockmarket." *Physica D* 75(1-3):264–274.
URL `http://deepblue.lib.umich.edu/handle/2027.42/31402`

Pinheiro, Jose C. and Douglas M. Bates. 2000. *Mixed Effects Models in S and S-Plus*. Springer.

Raudenbush, Stephen W. and Anthony S. Bryk. 2002. *Hierarchical linear models: applications and data analysis methods*. Thousand Oaks, CA: Sage.

Schelling, Thomas C. 1971. "Dynamic Models of Segregation." *Journal of Mathematical Sociology* 1:143–186.

Schofield, Norman. 1978. "Instability of Simple Dynamic Games." *The Review of Economic Studies* 45(3):575–594.
URL `http://www.jstor.org/stable/2297259`

Selten, Reinhard. 1975. "A Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games." *International Journal of Game Theory* 4:25–55.

Shibanai, Yasufumi, Satoko Yasuno, and Itaru Ishiguro. 2001. "Effects of Global Information Feedback on Diversity: Extensions to Axelrod's Adaptive Culture Model." *The Journal of Conflict Resolution* 45(1):80–96.
URL `http://www.jstor.org/stable/3176284`

Signorino, Curtis. 1999. "Strategic Interaction and the Statistical Analysis of International Conflict." *American Political Science Review* 93(2):279–97.

Strogatz, Steven H. 2003. *SYNC: The Emerging Science of Spontaneous Order*. Hyperion, 1 ed.

Theraulaz, G. and J.-L. Deneubourg. 1992. "Swarm Intelligence in Social Insects and the Emergence of Cultural Swarm Patterns." Tech. Rep. 92-09-046, Santa Fe Institute, Santa Fe, NM.

Toulmin, Stephen. 1961. *Foresight and understanding; an enquiry into the aims of science*. Bloomington: Indiana University Press.

Von Neumann, John and Oskar Morgenstern. 1953. *Theory of games and economic behavior*. Princeton: Princeton University Press, 3d ed. ed.

Wolfram, Stephen. 2002. *A New Kind of Science*. Wolfram Media.

Zhang, Junfu. 2011. "Tipping and Residential Segregation: A Unified Schelling Model*." *Journal of Regional Science* 51(1):167–193.
URL `http://dx.doi.org/10.1111/j.1467-9787.2010.00671.x`