Reading Assignment Sheet #1

EES 4760/5760 Agent- and Individual-Based Computational Modeling

Reading for Aug. 22–Dec. 5, 2019

General instructions for reading assignments

- Do the assigned reading *before* you come to class on the date for which it is assigned. If you have questions or find the ideas presented in the readings confusing, I encourage you to ask questions in class.
- Questions in the "Reading Notes" sections of the assignments are for you to think about to make sure you understand the material, but you do not have to write up your answers or turn them in. You are responsible for all the assigned readings, but topics I have highlighted in the reading notes are particularly important.
- In addition to the questions I ask in the reading notes, look over the "Conclusions" section at the end of each chapter in *Agent-Based and Individual-Based Modeling* to check whether you understand the key facts and concepts from the chapter.

Thu., Aug. 22: Introduction

Reading

No new reading for today.

Tue., Aug. 27: The computer modeling cycle

Homework

Homework #1: Install NetLogo and Set up Box account is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 1
- "Artificial Societies" (Handout)

Reading Notes

This reading sets the stage for answering the questions:

1. What is computational modeling and why is it useful in social and natural science research?
2. What are agent based models? How are they different from other kinds of models? What makes them useful for scientific research?

The reading introduces the idea of a **modeling cycle**. You should understand the different steps in the modeling cycle. You should also think about why Railsback and Grimm describe modeling as a cycle, as opposed to a linear process with a start and stop.

As to what makes agent-based modeling special, Steven Railsback and Volker Grimm are ecologists and *Agent-Based Modeling* emphasizes aspects of agent-based modeling that are well suited for studying ecological systems. Others, such as social scientists, emphasize the aspects of agent-based

modeling that are well suited for problems in social science. And still others, such as computer scientists, emphasize aspects of automated and autonomous things (ranging from packets of data on a network to swarms of robots or flying drones that need to coordinate their activities and avoid collisions). What all of these approaches have in common are their use of individuals or **agents** (what is an agent?), which inhabit some kind of space or **environment** (this could be physical space or an abstract space, such as a computer network). Agents **interact** with each other and with the environment, and they make **decisions** according to rules.

The article "Artificial Life" gives you a feel for how an early agent-based model called "Sugarscape" was used as part of a very influential research project in the 1990s. Joshua Epstein and Robert Axtell who wrote Sugarscape are highly respected pioneers in agent-based modeling and the Sugarscape model set off a revolution in agent-based modeling by showing that a very simple model could reproduce complex phenomena that are observed in real societies. As you read through this article, think about what the different applications of agent-based models have in common. Do these suggest questions that you might be interested in exploring with agent-based models. Do you have questions, as you read this, about whether computer modeling can really tell you about real societies?

Agent-based models are often used to examine **emergent** phenomena. Neither reading describes clearly what *emergence* means. There is no simple definition, but during the semester we will pay a lot of attention to learning about emergence and trying to understand it. Do not worry if you don't understand emergence at this point. Emergence is difficult to put into words, and it's much easier to understand from experience. Over the course of the semester, we will work together to understand what emergence is and how to study it.

Thu., Aug. 29: Introduction to NetLogo

Homework

Homework #2: Mushroom hunt is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 2

Reading Notes

Familiarize yourself with NetLogo. I recommend that you read through the chapter with NetLogo open on your computer. Feel free to play around with NetLogo and try things out. The homework consists of following the step-by-step creation of a model in section 2.3.

Tue., Sep. 3: Specifying models: The ODD protocol

Homework

Homework #3: Ex. 2.1–2.2 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 3

Reading Notes

Read carefully through the different design principles. Some of them have meanings that are a bit different from what you might infer from colloquial use.

For instance:

- **Adaptation** does not mean a persistent change in a turtle's behavior similar to the biological/-Darwinian sense of adaptation in species. Rather, it means the way an agent changes its behavior in response to its *immediate* conditions. Thus, adaptation in the ODD sense might include behaviors such as eating when you are hungry (*eating* is an **adaptation** to *hunger*), putting on warmer clothing when it's cold out (bundling up is an adaptation to cold), and running away from a predator.
- The kind of persistent changes that arise over time from experience fall under the ODD design concept of **learning**: If there is more food near a river than on hills, turtles may **learn** to go to rivers when they are hungry.

You can download several useful documents related to the ODD protocol from the class web site:

- The journal article, V. Grimm *et al.* (2010). "The ODD protocol: A review and first update" *Ecological Modeling* **221**, 2760–68.. `https://ees4760.jgilligan.org/files/odd/Grimm_2010_ODD_update.pdf`
- A Word document that provides a template for writing ODDs: `https://ees4760.jgilligan.org/files/odd/Grimm_2010_odd_template.docx`
- Lists of scientific publications using agent-based and individual-based models that either do or don't use the ODD protocol (this appeared as Appendix 1 of the Grimm *et al.* paper):
  - `https://ees4760.jgilligan.org/files/odd/Grimm_2010_appendix_1.pdf`,
  - `https://ees4760.jgilligan.org/files/odd/ch3_ex1_pubs_with_no_ODD.pdf`,
  - `https://ees4760.jgilligan.org/files/odd/ch3_ex2_pubs_with_ODD.pdf`.

Thu., Sep. 5: Your first model

Homework

Homework #4: Ex. 2.3–2.4 and Ex. 3.3 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 4

Reading Notes

For the reading, read Chapter 4 in *Agent-Based Modeling* first and focus mostly on this chapter.

It is worth noting that the Sugarscape model you read about for Aug. 27 is very similar to the hilltopping model. Sugarscape was part of a very influential research project in the 1990s, in which Joshua Epstein and Robert Axtell showed that a very simple model could reproduce complex biological and economic phenomena that are observed in real societies and ecosystems. You may find it interesting at this point to look back at the "Artificial Societies" article and to play with the Sugarscape model in NetLogo.

You can download an ODD for the butterfly model, which is suitable to paste into the "Info" tab in NetLogo, from the class web site: `https://ees4760.jgilligan.org/files/models/chapter_04/butterfly_model_ODD.txt`

Tue., Sep. 10: Using models for science

Reading

- Railsback & Grimm, Ch. 5

Reading Notes

This reading sets the stage for answering the big question, "How can we use agent-based models to do science?" There are several aspects to this question, which this chapter will introduce:

1. How can we produce quantitative output from our models?
2. How can your models read and write data to and from files? (This is important for connecting your model to other parts of your project)
3. How should we test our models to make sure they do what we think they do? (More on this in Chapter 6)
4. Making your research reproducible by using version control and documentation.

A number of you may like to use Excel or statistical analysis tools, such as R, SPSS, or Stata. The material in this chapter about importing and exporting data using text or `.csv` files will be very useful for this. By default, NetLogo only allows you to read in data in simple text files. However, if comes with some extensions that you can use to read in data from other common file formats, including `.csv` and ArcGIS shapefiles and raster (grid) files.

If you want to read in data from csv files, you may want to look at the documentation for the \texttt{csv} extension to NetLogo. To use it, you just put the line `includes [csv]` as the first line of your model, and then use functions from the extension, such as `let data csv:from-file "myfile.csv"`.

To read in date from ArcGIS files, look at the documentation for the GIS extension. You would put the line `extensions [ gis ]` as the first line of your model, and then use functions, such as `gis:load-dataset`, which can load vector shape files (`.shp`) and raster grid files (`.grd` or `.asc`). The GIS extension offers a lot of functions for working with vector and raster GIS data. If you're interested in using GIS data in your models, take a look at the GIS examples in the NetLogo model library.

You can download the data file with the elevations for the realistic butterfly model from `https://ees4760.jgilligan.org/files/models/chapter_05/ElevationData.txt`

Thu., Sep. 12: Testing and validating models

Homework

Homework #5: Ex. 4.2, 4.4, Ex. 5.1, 5.2, 5.4, and 5.7, and Ex. 5.5 and 5.8 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 6

Reading Notes

No one writes perfect programs. Errors in programs controlling medical equipment have killed people. Errors in computer models and data analysis code have not had such dire results, but have wasted lots of time for researchers and have caused public policy to proceed on incorrect assumptions. In

many cases, these errors were uncovered only after a great deal of frustration because the original researchers would not share their computer codes with others who were suspicious of their results.

You can never be certain that your model is correct, but the more aggressively you check for errors the more confident you can be that it does not have major problems.

Two very important things you can do to ensure that your research does not suffer a similar fate are:

1. Test your code. Assume your program has errors in it and make the search for those errors a priority in your programming process. Some things you can do in this regard are:

   - Write your code with tests that will help you find errors.
   - Work with a partner: after one of you writes code, the other should read it and check for errors.
   - Break your program up into small chunks. It is easier to test and find bugs if you are looking at a short block of code than if you are looking at hundreds of lines of code.
   - Independently reimplement submodels and check whether they agree with the submodel you are using.

2. Publish your code. If you trust your results and believe they are important enough to publish in a book or journal, then you should make your code available (there are many free sites, such as `github.com` and `openabm.org` where people can publish their models and other computer code).

   The more that other researchers can read your code, the greater the probability that they will find any errors, and if you make it easy for others to use your code, it will help science because other people can build on your work, and it will help your reputation because when other people use your model or other code they are likely to cite the publication in which you first announced it, so your work will get attention.

   Many scholarly journals demand that you make your code available as a condition for publishing your paper, and federal funding agencies are increasingly requiring that any research funded by their grants must make its code and data available to other researchers and the public.

The Marriage Model

The ODD for the marriage model and a NetLogo model that implements the ODD, but with many errors, can be downloaded from the class web site:

- `https://ees4760.jgilligan.org/files/models/chapter_06/Marriage_age_model_ODD.pdf`,
- and `https://ees4760.jgilligan.org/files/models/chapter_06/Marriage_model_with_errors.nlogo`.

Tue., Sep. 17: Choosing Research Projects

Reading

- Railsback & Grimm, Ch. 7

Reading Notes

There is not very much reading for today. Read Chapter 7 of *Agent-Based Modeling* carefully (it's very short) The point of this chapter is to help you get ideas for your term research project.

Before class, I want you to read through the research project assignment and think about what you might want to do for a research project. We will spend class talking about possible research projects.

Thu., Sep. 19: Emergence

Homework

Homework #6: Ex. 5.11 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 8

Reading Notes

This is a major chapter. Emergence is one of the most important concepts in agent-based modeling, so pay close attention to the discussion in this chapter and think about how you can measure and assess emergence.

This chapter also introduces a very important tool for doing experiments in NetLogo: **BehaviorSpace**. BehaviorSpace lets you repeatedly run a NetLogo model while varying the settings of any of the controls on your user interface. Where there is randomness (stochasticity) in the model, you can perform many runs at each set of control settings. This will let us perform **sensitivity analysis** to determine whether a certain emergent phenomenon we are investigating happens only for values of the parameters within a narrow range, or whether it happens over a wide range of the parameters. It will let us determine which parameters are most important for the phenomenon.

For homework and your modeling projects you will use BehaviorSpace extensively. BehaviorSpace outputs large amounts of data to `.csv` files, which you can read into Excel, R, SPSS, or another tool where you can do statistical analysis and generate plots such as the ones in figures 8.3, 8.5, and 8.6.

The format in which BehaviorSpace saves its data is very annoying to deal with in many tools. Indeed, it's almost impossible to do anything useful with it in Excel. Because of this, I have written a tool called `analyzeBehaviorspace` that can read the output of a BehaviorSpace run and allow you to interactively graph it and re-organize the data to make it more useful.

You can either use this tool online in a web browser at `https://analyze-behaviorspace.jgilligan.org` or install it on your own computer. For details, see the description of `analyzeBehaviorspace` on the "Reading Resources and Computing Tools for Research" handout.

As you read the chapter, be sure to try out the experiments with the birth-and-death model and the flocking model. Try reading the output of the behaviorspace runs into analyzeBehaviorspace (the web version or a local version installed on your computer), or your favorite statistical analysis software and try to generate plots similar to figures 8.3, 8.5, and 8.6.

If you have time, try to play around with BehaviorSpace using those models (varying different parameters) or other models from the NetLogo library to explore the ways that changing parameters affects the models' behavior.

Tue., Sep. 24: Observation

Homework

Homework #7: Ex. 6.2, 6.3, 6.5 and Ex. 6.4, 6.7 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 9

Reading Notes

In this chapter, we examine how to detect and record the properties of a model that we want to study.

The article, D. Kornhauser, U. Wilensky, and W. Rand. (2009). "Design guidelines for agent-based model visualization," *Journal of Artificial Societies and Social Simulation* **12**, 1 is available online at `http://jasss.soc.surrey.ac.uk/12/2/1.html`.

I have posted a refresher guide for NetLogo programming on the class web site at `https://ees4760.jgilligan.org/files/models/chapter_09/ch9_ex8_Netlogo_exercises.pdf`

Thu., Sep. 26: Sensing

Reading

- Railsback & Grimm, Ch. 10

Reading Notes

Important programming concepts in this chapter include:

Links: ~ Agents interact with their physical environment (patches around them) and with other agents nearby, but they can also interact in social networks, which can be represented by links.

Variable scope: ~ Understand the differences between global variables, local variables, patch variables, agent variables, and link variables. Understand how an agent can get the value of a global variable or variables belonging to a certain patch or link or another agent.

Entity detection: ~ Understand different ways to detect which agents or patches meet certain conditions (e.g., within a certain distance, have a certain color, have the largest or smallest values of some variable, etc.).

The agents' interactions, both with their environment and with each other through sensing. Part of the design concepts section of a model's ODD consists of specifying what the agents can sense: They might be able to sense other agents within a certain distance. They might only be able to detect other agents if they are within a certain angle (e.g., the agent might be able to look forward, but might not be able to see behind itself unless it turns around). Agents might be able to detect certain qualities of one another (e.g., I can see how tall you are, but I can't see how much money you have).

Agents can interact both spatially and through networks of links. You can create many kinds of links so that agents can belong to many networks (e.g., family, co-workers, members of a church congregation, etc.).

Sensing involves two steps:

1. Detect which entities your agent (or patch) will sense.
2. Get the values of the sensed variables from those entities.

Be sure to code the Business Investor model as you read section 10.4. You will use it in Chapter 11 and it will form the basis for one of the team projects you will present. The other team project will use the Telemarketer model from section 13.3. This would be a good time for you and your teammates to get started on your team projects.

Tue., Oct. 1: Adaptive Behavior and Objectives

Homework

Homework #8: Ex. 8.1, 8.2, Ex. 8.3, 8.4, Ex. 9.1, 9.3, 9.4, and Ex. 9.6 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 11

Reading Notes

Agents' behavior often consists of trying to achieve some **objective**.

I have discussed the way that Adam Smith's "invisible hand of the market" is a kind of agent-based view of a nation's economy: Each person (agent) has an objective of trying to maximize his or her own wealth (that's the agent's **micromotive**), and in doing so, the population of agents manages unintentionally to maximize the total wealth of the nation (an emergent **macrobehavior** that results from the collective interactions of the agents and their micromotives).

For Darwin, agents whose objectives are to survive and reproduce under changing environmental conditions achieve emergent phenomena of evolution and speciation.

If we are going to program an agent-based model to simulate such an economy (we saw this in the Sugarscape models), you need to program your agents to try to achieve their objective (maximize their wealth). There are two approaches to this:

1. You could program a sophisticated strategy into your agents.
2. You could program a simple strategy into your agents, but give them the ability to learn from their experience and adapt their behavior according to what they learn. (see section 11.3 for details and an example)

This chapter discusses different kinds of objectives you might have your agents employ. An important concept from decision theory and behavioral economics that might be new to you is **satisficing**. This term, introduced by Herbert Simon[^1] in 1956, refers to making decisions by choosing a "good-enough" option when it would take too much time and effort to determine which option is the absolute best. See section 11.4 for details and an example.

[^1]: Herbert Simon (1916–2001) was a fascinating intellectual. Kind of a renaissance scholar, he made major contributions to political science, economics, cognitive psychology, and artificial intelligence. He won the Nobel Prize for economics in 1978. His publications have been cited more than 250,000 times and even fifteen years after his death, they are still cited more than 10,000 times per year.

Thu., Oct. 3: Prediction

Homework

Homework #9: Ex. 10.1, 10.2, Ex. 11.1, 11.2, and Ex. 11.3 is due today. See the homework assignment sheet for details.

Reading

- Railsback & Grimm, Ch. 12

Reading Notes

In class, we will discuss the telemarketer model.

For the teams working on the telemarketer model, the steps are:

1. Build the model as described in the ODD.
2. Do the analyses in section 13.3.2
3. Next work on two extensions:

(a) Mergers (section 13.5)

(b) Customers remember (section 13.6)

Before class on Thursday, everyone should read Chapter 13 and the ODD for the telemarketer model and be ready to discuss the model in class. This will be a chance for the teams working on the model to ask questions.

Tue., Oct. 8: Interaction

Reading

- Railsback & Grimm, Ch. 13

Thu., Oct. 10: Team Presentations

Homework

Team Project Assignment: Team project presentations in class is due today. See the assignment sheet for details.

Reading

No new reading for today.

Tue., Oct. 15: Research Project ODDs

Homework

Semester Project Assignment: Analysis of the model you chose for your semester research project is due today. See the assignment sheet for details.

Reading

No new reading for today.

Thu., Oct. 17: Scheduling

Reading

- Railsback & Grimm, Ch. 14

Reading Notes

A good example of asynchronous updating is the model of breeding synchrony, described in Jovani and Grimm (2008) and in Chapter 23, which I have posted on the class web site: `https://ees4760.jgilligan.org/files/models/chapter_23/Ch_23_4_breeding_synchrony.nlogo`

Tue., Oct. 22: Stochasticity

Reading

- Railsback & Grimm, Ch. 15

Thu., Oct. 24: Fall Break

Fall Break. No class. Enjoy yourselves.

Tue., Oct. 29: Collectives

Reading

- Railsback & Grimm, Ch. 16

Thu., Oct. 31: Patterns

Reading

- Railsback & Grimm, Ch. 17–18

Reading Notes

You can download the ODD for the BEFORE beech forest model, which is described in section 18.3, from the class web site: `https://ees4760.jgilligan.org/files/models/chapter_18/ch18_before_ODD.pdf`.

I have also posted a list of published models in which observed patterns are important: `https://ees4760.jgilligan.org/files/models/chapter_18/ch18_ex1_models_list.pdf`

Tue., Nov. 5: Theory Development

Reading

- Railsback & Grimm, Ch. 19

Reading Notes

There is an implementation of the wood hoopoe model, suitable for Exercise 2, on the class web site: `https://ees4760.jgilligan.org/files/models/chapter_19/ch19_ex2_wood_hoopoes.nlogo`

Thu., Nov. 7: Parameterization and Calibration

Reading

- Railsback & Grimm, Ch. 20

Tue., Nov. 12: Parameterization and Calibration 2

Reading

- Railsback & Grimm, Ch. 20

Thu., Nov. 14: Analyzing ABMs

Reading

- Railsback & Grimm, Ch. 22

Tue., Nov. 19: Sensitivity and Robustness

Reading

- Railsback & Grimm, Ch. 23

Reading Notes

You can download the model of breeding synchrony, described in Jovani and Grimm (2008) and in section 23.4, from the class web site: `https://ees4760.jgilligan.org/files/models/chapter_23/Ch_23_4_breeding_synchrony.nlogo`

Thu., Nov. 21: Looking Ahead: ABMs Beyond this Course

Reading

- Railsback & Grimm, Ch. 24

Tue., Nov. 26–Thu., Nov. 28: Thanksgiving Break

Thanksgiving Break. No class. Enjoy yourselves.

Tue., Dec. 3: Presentations

Homework

Semester Project Assignment: Presentation of research project is due today. See the assignment sheet for details.

Reading

No new reading for today.

Thu., Dec. 5: Presentations

Homework

Semester Project Assignment: Presentation of research project is due today. See the assignment sheet for details.

Reading

No new reading for today.