Agent-Based Modeling: Week 1

Winter 2023

Jean Clipperton

Agenda

- Overview of the course
- Basics / things to make our lives more pleasant
- ABMs: what are they and how do they work?
- Conway's Game of Life

Overview

- Assuming some background in Python (equivalent to MACS 30111 / 30121)
- Course balance between **ABM principles** and **substantive content** (e.g. applications)
    - In-class programming sessions on 'canonical models' (e.g. Conway's game of life)
- Opportunity to apply design principles in assignments and (ultimately) develop your own model

Course Expectations

- Do the readings and come with notes and questions
- Think about broader applications / relevancy
- Be an engaged group member

Things to install / have

This will make your life easier

- ZOTERO: both the desktop and extension (chrome and/or google docs)
- Notetaking system: **FOLLOW THE SCIENTIFIC METHOD** whenever possible
  - *Puzzle*
  - *Theory*
  - *Test*
  - *Method*
  - *Conclusions*

Assignments

- Smaller assignments due in weeks 3, 6, 7
  - Each assignment applies a concept from previous course discussions
  - You have the option to select the concept from two choices
- Final assignment due creating an original model and applying course concepts

Grading

Grades are a function of:

- Assignments
- In-class **engagement**
- Group contributions (posting summaries and questions on ED)

NOW, on to the fun stuff!

ABM Overview

Agent-Based Models (ABMs) are ways we can model systems or situations starting from a set of simple assumptions.

What is an Agent-Based Model (ABM)?

ABMs allow you to focus on

1. the evolution of behavior / system and
2. explicitly engage with heterogeneous actors.

In contrast to other types of models, ABMs permit heterogeneous agents who can have different experiences and allow you to explore system-level effects, the path to equilibrium, and emergent phenomena.

ABMs vs. other models:

- **Linear Regression**: focus upon large-scale relationships instead of individual-level factors.
- **Systems Dynamics**: focus on the system from a high level; typically need an understanding of the equations governing the system.
- **Game Theory**: can look at final equilibrium of the system, without information about what occurs on the way there nor how long it will take.
- **ABM**: 'Bottom-up' approach using simple assumptions, analyzes how the system behaves and emergent phenomena that result from the model.

ABM setup and components:

Who does what, how?

- **Actors** these can be people, animals, businesses, etc. The *units* you care about.
- **Parameters** (AKA *variables*) these are the things you're going to vary.
- **Framework** structure for interaction.
  -Structure ecosystem / environment for actor.
    - Define movement / behavior (where are they placed, how they interact, move, etc.).

ABM setup and components:

Who does what, how?

- **Actors** (what are the units / agents /actors)
  - Initialize with identical or varied attributes

- **Parameters** (AKA *variables*)
  - Consider range and other variation / combination potentials

- **Framework** structure for interaction
  -Structure ecosystem / environment for actor
  - Define movement / behavior (where are they placed, how they interact, move, etc.)

Parts of the ABM: Initialization and Steps

For both the agent portion and the model portion, you set up (**initialize**) your universe (give the agents qualities, put them in the world), and then specify what happens with each tick of time. This ticking of time is known as the **step function**.

You may also choose to display different pieces of information about what is occuring (which we'll talk about in a few weeks in more detail).

Applications

ABMs allow you to evaluate situations from relatively simple initial conditions and have broad applications.

- **Political Science** emergence of markets (Padgett), effects of institutional change (Bednar/Page)
- **Sociology** neighborhood segregation, evolution of cultural groups (Axelrod).
- **Economics** tragedy of the commons (Hardin), market bubbles.
- **Ecology** flocking, predator/prey models, ecosystems.
- **Epidemiology** pandemic modeling (Zelner), health disparities, public health policy analysis.

Why Model (Epstein reading)

Why are we here?

Trying our best to make some sense of things -- we have a question, issue, topic, etc that we want to learn more about.

All models are wrong

(but some models are useful)

- *George Box*

- Explain (very distinct from predict)
- Guide data collection
- Illuminate core dynamics
- Suggest dynamical analogies
- Discover new questions
- Promote a scientific habit of mind
- Bound (bracket) outcomes to plausible ranges
- Illuminate core uncertainties.
- Offer crisis options in near-real time
- Demonstrate tradeoffs / suggest efficiencies
- Challenge the robustness of prevailing theory through perturbations
- Expose prevailing wisdom as incompatible with available data
- Train practitioners
- Discipline the policy dialogue

How we'll do this

You'll have some kind of goal, problem, or objective to satisfy. There is a lot that goes into the model and **only some of it is the programming portion!**

Keep in mind / Designing:

We'll reference this over time but keep in mind that we will often have a goal in mind for the final model.

- Think about where you want to go and the steps / pieces needed to get you there
- Sketch out how it should look (**THIS WILL SAVE YOU TIME OVERALL!!**)

Examples

'Classic' or 'Canonical' Models

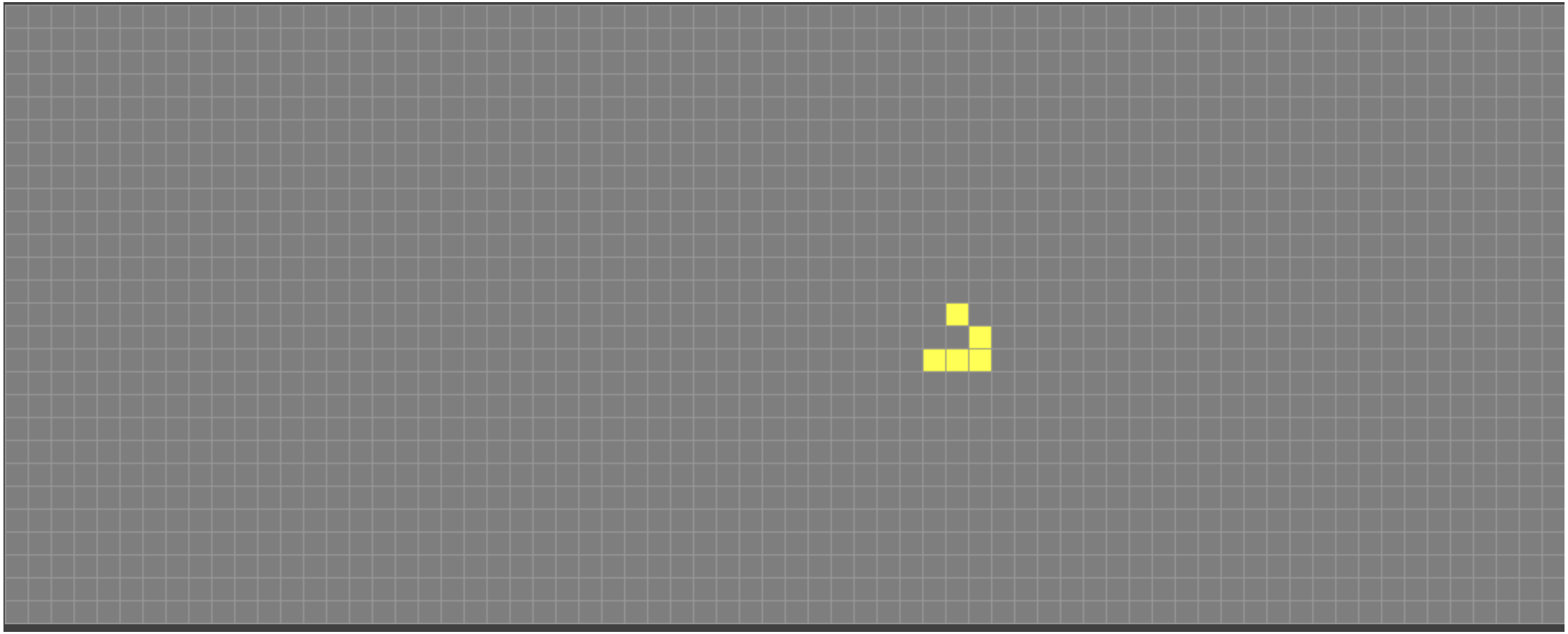
Throughout our course, we'll be discussing popular models, such as Conway's Game of Life, Schelling's Segregation, Boids / flocking, etc.

We'll use these as a means to learn about the week's topic (e.g. agent types) while familiarizing ourselves with the literature.

We'll dedicate time in our Thursday sessions to digging in deeper to the code.

Conway's Game of Life

This canonical model is a great *jumping*-off point: it starts with a grid of cells that can be alive or dead. The model (game) allows us to find patterns that emerge from a set of simple starting rules.



Game of Life: Rules

- Any live cell with fewer than two live neighbors dies (referred to as underpopulation).
- Any live cell with more than three live neighbors dies (referred to as overpopulation).
- Any live cell with two or three live neighbors lives, unchanged, to the next generation.
- Any dead cell with exactly three live neighbors comes to life.

Breaking down Life: ABMs + ODD

Next week, we'll learn about different frameworks for describing our models. For right now, we'll focus on a precursor to that, WHO, does WHAT, HOW.

Overview, Description, Details

We're living in the land of object-oriented programming (OOP) and so we're dealing with object and methods. We'll talk about classes of objects and how they come together.

Conway's **Game of Life** is designed to help us see how patterns can emerge from rules (there's a lot more to it than this, but we're doing a simplistic overivew right now). We have *cellular automata*, which are our agents (**who**) that operate under a certain set of rules (**does what**) when various conditions are met (**how**).

Game of Life: Give it a go

Game of Life: Anatomy of code

We're going to follow the code conventions set out in the Mesa Reading (Kazil et al 2013). At first this will feel a bit rigid, but it can help us have a strong start in the where and how to approach things.

As they lay out, we are going to begin with an informatively-named file (conway), that contains at least four python files (run.py, agents.py, model.py, and server.py.) AND a README.

We'll do a bit of an overview today, and then dive in a bit more in our next meeting.

Agents (WHO)

We have a series of agents (yellow squares) that are 'alive'. In the model, there's a user-settable parameter that allows you to adjust the number of alive agents.

Parameters (DOES WHAT) (agent step function)

Typically, we look at our variables and other elements of the model to explore how they fit together. In this instance, we only have the rules for the game of life and the number of agents.

Framework (HOW) (model file including step function)

The framework, our structure for how things come together, can be easily overlooked.

Goals: NOTICING AND OBSERVING

- What do you see happening?
- What do you think are the crucial pieces of code that are functioning?

THE SEED

Note: I have made it so that we all have the same seed (10) because it gave a somewhat interesting run of the model and then we could all replicate the same scenario.

Having the seed can be useful for replicability (*Try it out and notice that you can reset to your heart's desire. Same results!*).

Playing around: let's go back to setting our seed to 10

AGENTS: (Who)

Now, let's look at how we're initializing our agents. How are they placed in the world?

What are other choices I could have made?

Possible Choices:

- How the agents are placed (at random vs via some other method)
- Why not just have living agents?

Does what (step function)

In each step, what do we want to do?

- What choices did I make here?
- What are some of the costs of my choice?
- What else could I have done?
- *What if I don't know how to do this yet?? How do I get this information??*

The beginning of a love - hate relationship: MESA DOCUMENTATION!

- We're using Mesa (as opposed to Netlogo or Java) because we can carry over our understanding of Python.
- Con: Mesa is newer and there're fewer resources out there
- Pro: More documentation / information than other add-ons for Python

How: Deep dive into model framework

Generally, models have the initialization and then the step method (you can also include additional elements for activation where you have a sort of holding period before you actually complete the setp).

There are different ways you can specify the step function (for all agents of all types, etc).

Notes on how we're doing this

I'm going to teach this using VS Code and script files. You can choose to use other IDEs (e.g. PyCharm). I discourage you from using Jupyter Notebooks (even though they're great!!) because they can sometimes be a bit challenging to get to work nicely with the graphics and because it's harder to have the nice tidy code setup that we're going to be following here.

There are circumstances where you might want to later use Jupyter notebooks, but for all assignments in this course, you'll need separate script files.

Looking ahead:

I will take MULTIPLE volunteers to talk us through the two readings. Please be ready to walk through the logic of the readings and think critically about what they contribute to our understanding.