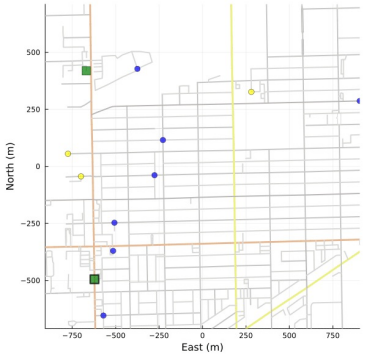
Agent-Based Modeling

FOR DUMMIES®



By: Anna Cobb

Table of Contents

A Refresher

- Anna Cobb definition: An agent-based model is defined by the pre-programmed behavior of its agents which dictates their interactions with each other and the model environment.
- Goal of using an ABM is to observe emergent macro-scale behaviors that are a result of each agent's individual decisions/actions
  - think disease spreading, animal migration patterns, etc.

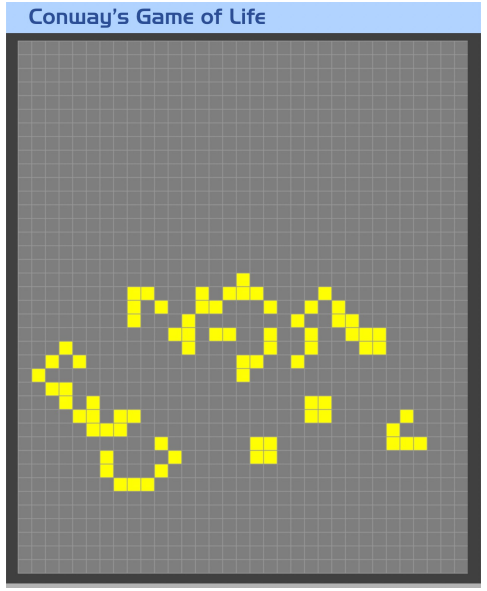
A Brief History of Space & Time & ABM

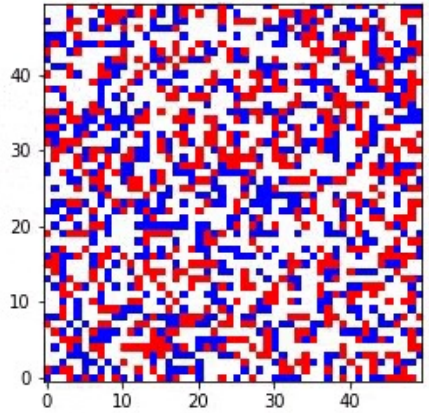
1940s:
they were thinking about it

1970s:
they got the computational power/language for it!

1990s:
"expansion" - Wikipedia

1970: John Conway - Game of Life
1971: Thomas Schelling – Segregation Model





Large scale "Sugarscape" model created by Joshua Epstein & Robert Axtell (EPP Alum!!!)

Santa Fe Institute became a thing: used ABMs for studying societal & social dynamics

Kathleen Carley at CMU was active in using ABMs for exploring the "co-evolution of social networks and culture"

A Brief History of Space & Time & ABM

2022: Anna Cobb started her PhD in EPP at CMU

2023: Anna decided she wanted to use an agent-based model to study Uber and Lyft…

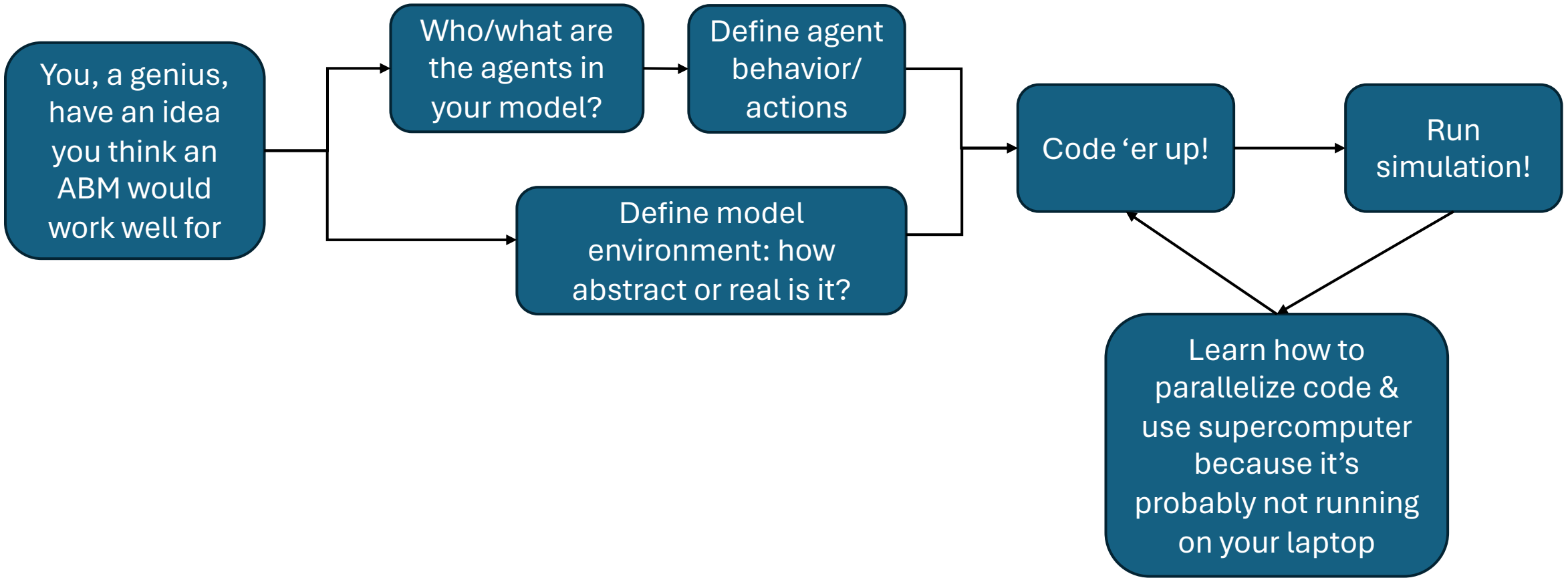
a decision she would come to deeply regret forever…

just kidding!

2024: Anna used said ABM to pass quals so it works

Shoutout to Aniruddh

Implementation: High Level



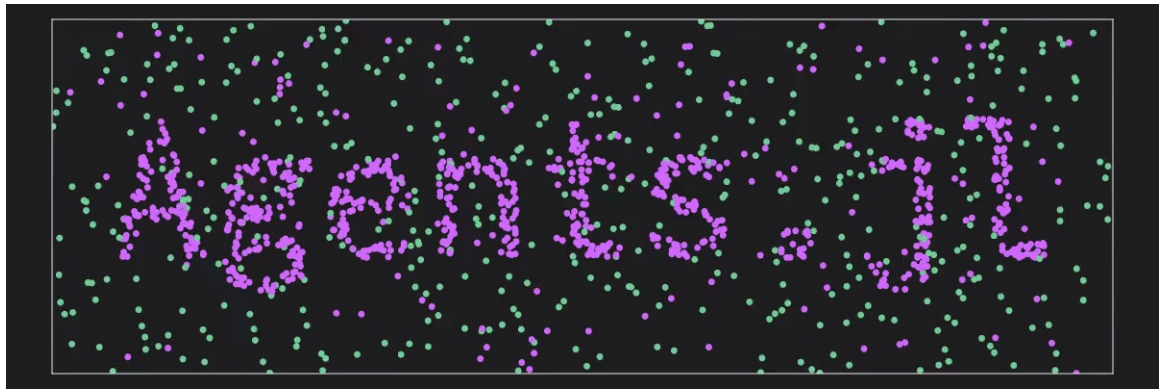
Implementation: but actually how?

- Well well well. I have had 2 experiences:

Use a pre-made model:

this nearly killed me.

Using a from-scratch model written in Julia



led me to greatness.

Implementation: Agents.jl

- Some pros of Julia:
  - syntax = python + MATLAB + you can make it act like R if you're doing data cleaning stuff
  - relatively easily parallelization (making use of multiple threads or cores)
  - known for efficiency and being a HPC language
- Some pros of the Agents package
  - Agents.jl is very well documented with a decent amount of online support
  - Agents.jl produces relatively fast ABMs compared to other options
  - Allows easy setup with OpenStreetMaps model environment (these are cool!)
  - A ton of built-in functionality

Implementation: Agents.jl

- Core Components:
  - Defining your agents:
    - can be multiple types, but each type shares a set of features/characteristics
    - these characteristics are not immutable & you must initialize them when adding agents to your model
    - think objects in object oriented programming or as the blueprints for each type of agent in your model
  - Example:

```
# EPP Student Agent
@agent Myagents OSMAgent begin
    status:: Int # 0 = sitting, 1 = standing, 2 = on ze run
    papers:: Vector{Int} # DOIs of papers glanced at
    snacks:: Int64 # snacks consumed
    sleeps:: Int64 # hours of sleep last night
    assignments_due:: Int # assignments due soon
    stress_level:: Float64 # how high is the stress
    name:: Symbol # Anna or Iana or Nana
end
```

Implementation: Agents.jl

- Core components:
  - Defining agent behavior: usually for one time step (let's say 1 min here)

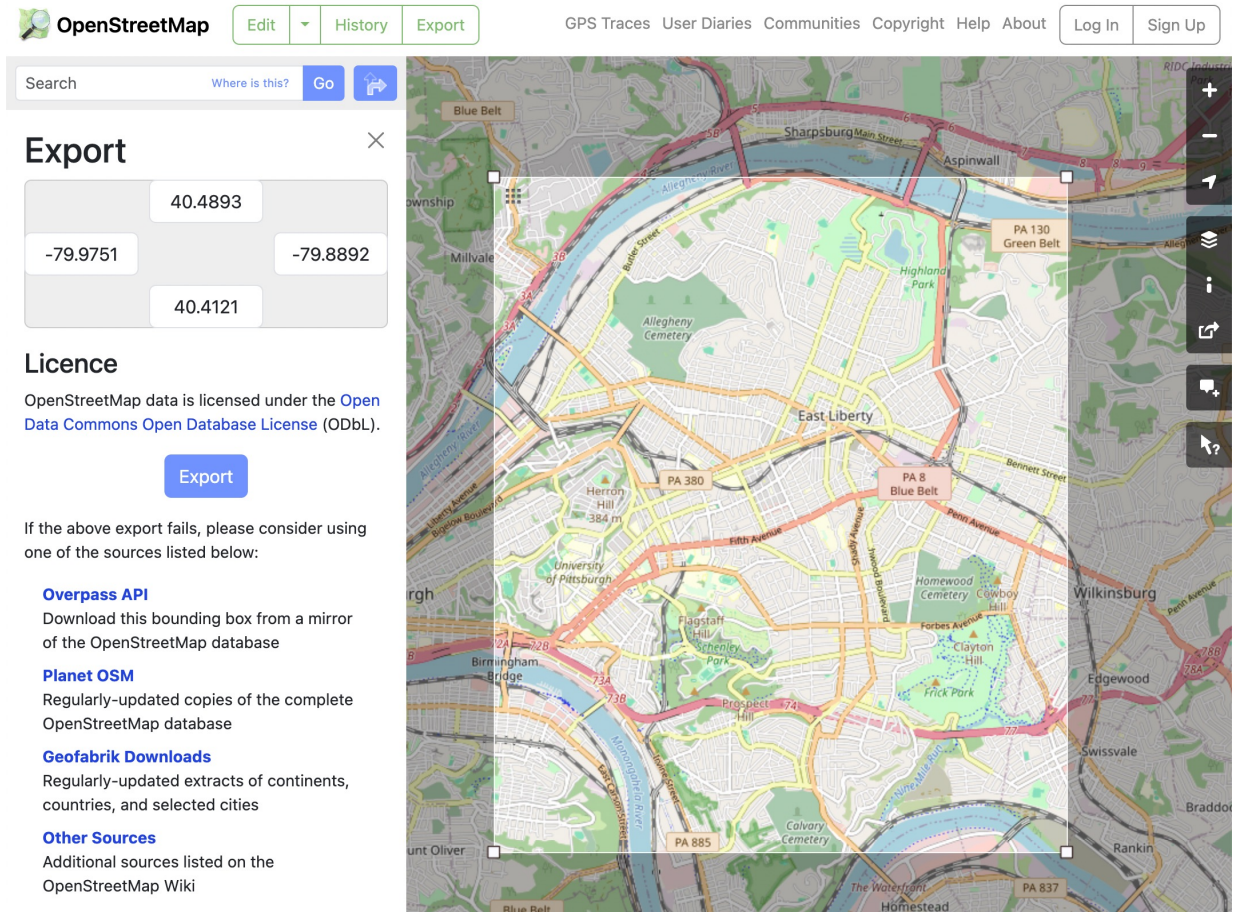
```
function Anna_step!(Anna,model)
    if model.time > 9 PM | model.time < 7 AM
        if Anna.assignments_due > 5
            work(Anna)
        else
            Anna.sleeps += 1
        end
...
```

```
function work(agent)
        agent.assignments_due -= 0.05
        agent.stress -= 0.1
end
```

Side note: Julia loves functions!

Implementation: Agents.jl

- Core components:
  - Defining the environment: there are a few built-in types
    - Graphs – each node of the graph can hold certain number of agents and agents can move from node to node. Agents.jl has an example of this being used to model spread of COVID-19
    - Grids – Schelling's segregation model is an example of a grid-type space
    - Continuous spaces – "agent position, orientation, and speed are true floats"
    - OpenStreetMap files – can be obtained via https://www.openstreetmap.org



Implementation: Other Options

- NetLogo: NetLogo (as far as I can tell it has its own language)
- Repast: Java, C++, Python
- MASON: Java
- Mesa: Python

Table 1. A comparison of four ABM frameworks covering objective categories focusing on ease of use, available functionality and performance. Colours represent implementation quality. Red: poor/none, Yellow: basic, Green: good, Blue: clear class leader. Further details corresponding to the superscript numbers are given in the main text.

	Agents.jl 4.2	Mesa 0.8	NetLogo 6.2	Mason 20.0
	Objective property comparisons.			
Core	Core design decisions and aspects that cannot be changed or implemented by users			
Continuous Space	Yes	Yes	Yes	Yes
Graph Space	Yes, and mutable	Only undirectional	Link Agents (not a Space)	Networks (not a Space)
Grid Space	Yes	Yes (+Hexagonal)	Yes	Yes (+Hexagonal, Triangular)
OpenStreetMap Space	Yes	No	No	No
Dimensionality	Any[1]	2D	2D & 3D (separate applications)	2D & 3D (complicated install for 3D)
License permissiveness	MIT	Apache v2.0	GPL v2	Academic Free License
Mixed-agent models	Yes	Yes	Yes	Yes
Simulation termination	After 'n' steps or user-provided boolean condition of model state	Explicitly written user loop	Manually by pressing a button on the interface, stop command in code	When Schedule is empty, or user provided custom finish function
Parameter types	Anything	Anything	Float64, Lists Hashtables and Assoc. Arrays in the Table extension	Anything
Modeling and Analysis in the same language	Yes, Julia v1.5+	Yes, Python v3+	No	Yes, Java but designed to work within the console or GUI of the applet
Maximum memory capacity	Hardware limits	Hardware limits	1 GB Manually expanded by increasing JVM heap	1 GB Manually expanded by increasing JVM heap
Distributed computing[2]	Yes	No. BatchRunnerMP is only multithreaded	No. BehaviorSpace is only multithreaded	Yes
Interop with external libraries	Yes, also couples to anything in Python / R / C / C++ seamlessly.	Yes, modular design.	Partial, via the Extensions API. JVM languages (Scala, Clojure)	Partial. Extensions in the 'contrib' directory. No simple user API
Language ecosystem integration	By Design. Examples: black box optimization, differential equations	Any of Python's analytical tools can be used	Complex. Must create plugins or use Control API	Warned against (e.g. Random), provides custom types in place of Java primitives
Browser-based online ABM execution	No	No	Yes (NetLogo Web)	No

Potential Drawbacks

- The outcomes of these models are very dependent on how agent behavior is defined
  - If people don't like the way your agents make decisions, they may not trust any of your results!
- Quite computationally intensive—I think the norm is needing to use a supercomputer, not the exception
- You don't exactly know why the outcomes of your model are what they are
- "Parametric" studies may not be as straight forward as usual…

Potential Drawbacks: An Illustration

- In my model of TNCs, we want to understand the effect of reducing the frequency with which drivers are matched to customers (trying to represent taxi operations)
- However, adjusting this parameter impacts the way other model heuristics work: for example, drivers come online in our model in response to unmet customer demand & now unmet demand has skyrocketed due to (in some ways) less efficient matching.
- So now we want to restrict the number of drivers and suppress our usual driver coming online mechanism, but then we end up with fewer customers getting served in the same amount of time…

Resources

- Agent-based modeling generally: Our alum's book Growing Artificial Societies was touted in 701. I have not read but likely worth checking out!
- Agents.jl: their homepage is great and full of examples and built-in function descriptions
- @ CMU: Kathleen Carley did respond to a couple of my emails. There aren't many classes heavily focused on this. Feel free to reach out (to me or others *Aaron* with experience)!
- I think reading papers that use these can help you get a feel for when they're useful!
- I have a presentation with lots of resources for using the PSC if that ends up being a part of your ~modeling journey~