

Intro to ABMs

2024-04-09

What is an agent-based model (ABM)?

- No rigorous definition (like most applied science)
- But: can characterize as a model consisting of
 1. **entities** (the agents)...
 2. ...which **act**¹ upon each other...
 3. ...to create some **emergent** outcome
- “Emergent” = collective is more than the sum of its parts
- Implemented computationally, sometimes also analysed mathematically

What is the benefit of ABMs?

“Like equation-based modeling, but **unlike prose**, agent-based models must be **complete, consistent, and unambiguous if they are to be capable of being executed on a computer**” (Gilbert 2020, xii, my emphasis)

- **complete**: the modeller cannot leave anything out of the model description
- **consistent**: no part of the model can logically contradict another part of the same model
- **unambiguous**: the meaning of every part of the model must be objectively clear

Examples

These concepts are best explained through the use of examples...

...so let’s look at a few!²

¹Eng. *agent* < Lat. *agens*, pres. part. of *ago* ‘act’

²The examples are taken from the [Example Zoo](#) of the Agents.jl package (released under the MIT license).

Example 1: Conway's Game of Life

- Early example of a **cellular automaton** (subclass of ABMs) (Gardner 1970)
- Lattice; each cell either “live” (L) or “dead” (D)
- Rules:
 1. Underpopulation: if a L cell is surrounded by fewer than two L neighbours, it dies
 2. Sustenance: if a L cell is surrounded by two or three L neighbours, it continues to live
 3. Overpopulation: If a L cell is surrounded by more than three L neighbours, it dies
 4. Reproduction: If a D cell is surrounded by three L neighbours, it becomes L

Example 1: Conway's Game of Life

[../videos/game_of_life.mp4](#)

Question

Earlier we said ABMs are complete, consistent and unambiguous.

What have I left out of the definition of Conway's Game of Life? (I.e. why is my description so far incomplete?)

Answer

Two very important things:

1. Is the lattice **finite** or **infinite**? If finite, then what happens at the boundaries? *It is infinite.*
2. Are the agents (the cells) updated **synchronously** (all at once) or **asynchronously**? *Synchronously.*

Species

The game supports many life forms (“species”), categorized into:³

- still lifes
- oscillators
- spaceships (moving oscillators)

³Images of Game of Life species from Wikimedia Commons (public domain).

Emergence

- The game has simple rules, complex behaviour
- It is **undecidable**: given a starting state S and a proposed other state O, we can prove that it is impossible to prove whether O will ever be reached from S!
- New facts about the game are still being discovered: 2018 discovery of “knightships” (spaceships that move like the knight in chess)

Example 2: Flocking

- A simple model of the emergence of collective behaviour, namely flocking in birds
- Birds follow three rules:
 - Collision avoidance: maintain a minimum distance to other birds
 - Tracking: fly towards the average position of neighbouring birds
 - Navigation: fly in the average direction of your neighbours

Example 2: Flocking

[../videos/flocking.mp4](#)

Example 3: Social Distancing

Bounded rationality / Locality

- Common to all these examples is the following observation: the agents have **bounded rationality**
 - In Flocking, individual birds follow only three simple rules defined over the bird’s neighbours
 - A bird **does not know** what flocking means, nor does it have a rule to accomplish flocking
 - Rather, flocking emerges as the collective behaviour of a group of birds
- In other words, **global** patterns arise from numerous **local** interactions
- Similar remarks apply to Game of Life and Social Distancing, indeed to **any** ABM

Challenges in ABM

- How do we know what to model?
- How do we test our models against empirical data?
- How do we implement our models computationally?

Challenges of computational implementation

- Speed: we want simulations to be fast
- Randomness: when our code calls for random numbers, we want them to be really random!
- Cleanliness: we want our code to be understandable to other users
- Reproducibility: when others run our code, they should get the same results we do

Why is speed an issue?

- Central processing units (CPUs) in modern computers carry out billions of instructions each second
- However, with ABMs, computational requirements may be significant, and may not **scale** nicely

Speed issues: an example

- You have a model such that one simulation run, with a given set of parameter values, takes 1 minute to complete.
 - Suppose your model has 2 parameters, each of which can assume 100 different values. This means a total of $100 \times 100 = 10,000$ parameter combinations.
 - Suppose you want to replicate the simulation for each parameter combination 100 times for statistical reasons.
 - Then you're looking at a total of 1 million runs.
- With 1 minute / run, we're looking at **2 years** to get the results!

How to deal with issues of speed

1. Choose a suitable programming language
2. Write **performant** code
3. Whenever possible, **parallelize** your code
 - This means running it simultaneously across many CPUs/computers; we will see later how it's done

Why is randomness needed?

- Quite simple: real-world processes are complex, and to model such complex processes we resort to **stochastic processes**
- A stochastic process is a sequence of random variables
- For example, consider a “navigating” agent that turns into a random direction whenever it doesn’t know how to proceed otherwise. In this case, the random direction needs to be generated using a **random number**.
- Or consider a linguistic example: suppose Mary is friends with Bob, Fiona and Charles. Unless we want to claim that Mary’s interactions with the other people are **deterministic** (which does not seem particularly sensible), we need some way of selecting interlocutors at random.

Why is randomness an issue?

- Conventional computers are deterministic devices
- So, if we need, say, a random number between 0 and 1, how is that accomplished?
- The answer is a **pseudorandom number generator** (PRNG)
 - This is an algorithm that generates a (long, but not infinite!) sequence of numbers which has the *appearance* of being random
 - The sequence is generated from a **seed** number. If you give the PRNG the same seed, you will get the same “random” sequence of numbers (this takes care of the reproducibility requirement).
 - **However**, there are significant issues...

Issues with PRNGs

- Suppose your PRNG generates a sequence of 1M numbers...
- ...but in your simulation you need to generate 10M random numbers⁴
- Then your “random” numbers will repeat 10 times
- This means that different parts of your simulation are **not** independent of each other – a major problem!
- Further issues can arise when we look at parallel processing... but more on that later!

Summary

- ABM is a powerful framework for modelling real-world processes
- Models are complete, consistent and unambiguous

⁴We’ll see later that this is by no means a crazy requirement!

- Individual agents exhibit bounded rationality
- Challenges involve, among other things, simulation speed and proper implementation of randomness

The scientific community

- ABMs are created and explored by people in all manners of disciplines from physics and chemistry to linguistics and economics
- Examples of professional organisation in social sciences and linguistics:
 - [European Social Simulation Association](#)
 - [The Journal of Artificial Societies and Social Simulation](#)
 - [The International Society for Computational Social Science](#) (+IC2S2 conference)
 - [Language Dynamics and Change](#) (journal)

Homework

Next week, we will start programming. To prepare your computer for this, complete the homework [Installing Julia](#).

References

- Gardner, Martin. 1970. “The Fantastic Combinations of John Conway’s New Solitaire Game ‘Life.’” *Scientific American* 223 (4): 120–23. <https://doi.org/10.1038/scientificamerican1070-120>.
- Gilbert, Nigel. 2020. *Agent-Based Models*. Second edition. London: SAGE.