Topic 6: Individual and Agent-based Simulation

Learning Objectives

1. Describe what is an "object" in object oriented programming and why this programming paradigm is useful in ecology and evolution
2. Describe the pros and cons of individual (agent) based simulation.
3. The three main features of an IBM are: (1) agent behaviour, (2) agent- agent interactions, and (3) the environment. Explain how each of these could be modelled.
4. Analysis of agent-based simulations:
    - Hypothesis design and experimental controls (Null model)
    - Plotting and statistical tests
    - Testing and validating individual based simulations
5. Propose and justify an agent-based model for a biological phenomena
6. Use an agent-based model to simulate a biological process
7. Define a Cellular Automaton
8. Propose and justify a Cellular Automaton for a biological phenomena
9. Use a Cellular Automaton to simulate a biological process

Lecture 6.1 Object Oriented Programming and Individual Based Simulation.

Object Oriented Programming

Before we can define Object Oriented Programming (OOP) we have to disucss data types. Programs have **primitive data types**, these include integers, booleans, doubles/floats, and characters.

Example: 6.1 Primitive Data Types
What are examples of biological sample spaces or random variables of

1. **Integers**

- # of offspring

2. **Characters**

- Genomes

3. **Booleans**

- Survival

4. **Doubles**

- Waiting times

We can make these data types by considering **arrays** (e.g., vectors and matrices) of them as we have done extensively in this class.

Discussion What are some examples of arrays that we have used in this class?

However, arrays are limited in that they only let use combine multiple instances of the same primitive data types. This is okay as long as we aren't doing anything too complicated, but as we start to consider greater and greater biological complexity/realism it would be nice to be bring together multiple primitive data types. Data **structures** allow us to combine many different kinds of data

---

Example: 6.2 The prey structure

Suppose we are tyring to model the behaviour of prey and its impact on predator-prey population dynamics. We might want to create a 'prey' object that captures many features of the prey individual.

Sketch out the possible data types needed to describe a prey individual and its behaviour.

structure Prey:

- **int** age
- **double** energy
- **double***[2] location
- **bool** state (foraging/non-foraging)
- **bool** sex
- **double** perceptionRadius (how far the prey can see to avoid predators)

Sometimes, however, its convenient for a group together not only different types of data but functions that can use those pieces of data. This lets us grab a bunch of information and methods all at the same time. When we do this the thing we get is a **class**.

Example: 6.3 The prey structure cont.

Expand your structure above to include class functions.

structure Prey:

- *Variables*
  - **int** age
  - **double** energy
  - **double***[2] location
  - **bool** state (foraging/non-foraging)
  - **bool** sex
  - **double** perceptionRadius (how far the prey can see to avoid predators)
- *Functions*
  - **def** mature
  - **def** circadianRhytham
  - **def** forageFlight

So a **class** is "template", a recipe for describing a particular prey indivdiual. But if we are to simulate the dynamics of a whole population, we will need not just one **realization** of this class but many of them. These individual realizations are known as **objects**, the name sake of **object oriented programming** OOP.

There are many benefits to OOP, many of which have to do with memory control and working on programs in large collaborative groups, but the key attribute of OOP that I want to highlight in this class is "**Inheritance**". Despite being in a math-bio class, by inheritance I do not mean genetic inheritance. Rather this feature may be described as the hierarchy. Using objects we can easily structure programs into hierarchical levels.

Biological systems are naturally hierarchically structured.

Example: 6.4 Hierarchy in Biology

What are some examples of hierarchy in biology?

1. Linnaean Systematics: Kingdom->Phylum->Class->Order->Family->Genus->Species
2. Physiology: Cell->Tissue->Organs->Organ system-> Indivdidual
3. Levels of Selection: Gene->Genome->Individual->Family->Populations->Speices
4. Ecology: Population->Communities->Ecosystem->Biome

OOP let's us create classes and subclasses and subsubclasses etc. Each with a single or multiple realizations hence replicating natural biological hierarchical structure.

Individual-Based Simulations

- See Jørgensen and Fath 2011

An individual-based simulation (IBS) is a type of simulation model that represents and tracks the behaviour and interactions of individual entities aka agents, which are often coded with classes, within a system. Instead of considering the system as a whole, for example with a single differential equation or stochastic process, an IBS focuses on modeling the actions, characteristics, and interactions of each individual unit, allowing for more detailed and realistic representations of complex systems. Importantly for ecological applications individuals can adapt to changing conditions of the simulation, and can modify their environment and through their actions.

When and why might we want to build a IBM?

1. When life cycles are complex
2. When individuals within the same life cycle stage exhibit significant vairability (e.g., in the amount of resources they have)
3. When we want to count the number of indvdiuals as integers. This feature is also true of stochastic processes but differs from that of ODE approaches.
4. When we care about modelling stochasticity in these complex systems.

There are three necessarily elements to developing an IBM

1. Specifying agent properties and individual behaviour (these are our class variables and functions)
2. Specifying agent-agent interactions (these are global functions)
3. Specifying the environment (these are the global vairables)

How can we analyze the outcome of IBM?

1. We have to consider multiple runs. Like individual trajectories, each trajetory is useful but we must consider the consistancy between and amoung trajectories/runs.
2. Similarly, we have analyze IBM using explicit statistical methods
3. An IBM is in between a full analytical model and an experimental approach. Hence we can use properties of both but that means we also have use scientific priciples for both of these approaches.
    1. We want the IBM to be as simple as possible
    2. We can make simplifying assumptions (e.g., one directional mutation)
    3. We should establish hypotheses, controls, and experimental lines. Controls are important for understanding if we have **run-time errors**
    4. We can test the results by going to the extremes (e.g., limits of no natural selection or no mutation), conditions which we could maybe never fully match in the lab.

Example: 6.5 Early IBMs in Ecology, Density-dependence in smallmouth bass

See DeAngelis et al. 1991: "An individual-based approach for predicting density-dependent dynamics in smallmouth bass populations"

- Density-dependence in fish growth and surivival is particularly important in fisheries for proper management.
- It is also potentially determined by many complicated feedback loops.

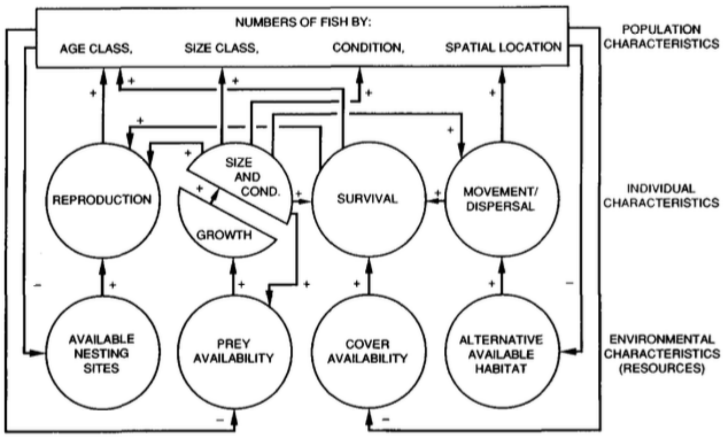


Fig. 1. Some of the feedback connections linking population level characteristics, individual characteristics, and environmental resources in a fish population. Most of the feedback loops shown here are negative or compensatory feedbacks.

- de Angelis et al. 1991 provides one of the early examples of how IBS were used to capture and understand the result of this level of biological complexity.
- They lay out the following four points to their rationale for using an IBS approach:
  1. "Each individual in a biological population is unique, differing in its set of characteristics (age, sex, size, condition, social status, genotype,...) from all other individuals. This is a result not only of genetic differences, but also of the different experiences (movements through space, encounters with prey and predators, mating, and accidents) of each organism, which are subject to stochasticity."
  2. "In populations (such as fish) in which growth is highly plastic and the relative sizes of individuals is important, a small fraction of a population may play a dominant role (e.g. in recruitment, reproduction, etc.). In such cases the 'average individual' in the population may be rather unimportant compared with the small fraction of extraordinary individuals."
  3. "Decision making by the individual organism, which can depend in complex ways on day-to-day circumstances, should be factored into population behavior."
  4. "Modeling at the individual level is compatible with taking into account short-time-scale variability (daily changes in weather, water level, ...) and small-scale spatial heterogeneity, both of which are important in critical life stages of organisms."
- They are particularly interested in how density impacts the surival and growth of young fish. They assume that adults give birth at the beginning of the simulation and then follow the indivdiual level dynamics of growth and survavial of the young (Age-0) fish through the first year.
- They model the effect of density and the enviornment through temperature.

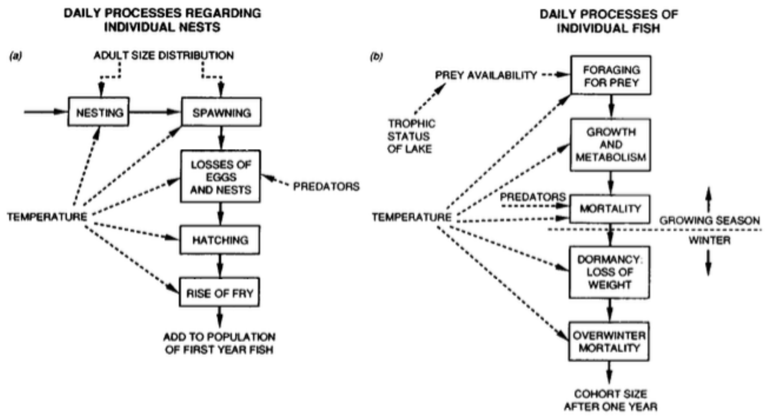


Fig. 2. Schematic of smallmouth bass model: (a) model of reproduction and pre-swim-up stage of fish, at which point each nest is considered as individual, and (b) model of population after swim-up from the nest, at which point the fish are considered as individuals.

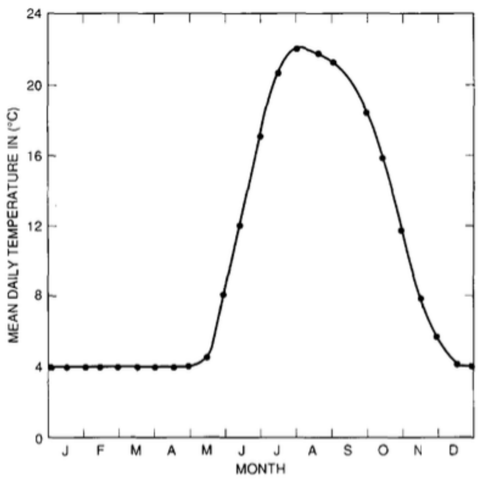


Fig. 3. Annual temperature scenario used in all of the simulations. The temperature is 4 °C during the period of ice cover.

- The details of the model take several pages to define, but if you are interested you can find this in the paper. Processes include:
  - Reproduction
  - Foraging
  - Growth
  - Prey population dynamics
  - Morality
  - Environmental dependence
- They then look at how the size and number of the fish change throughout the year.

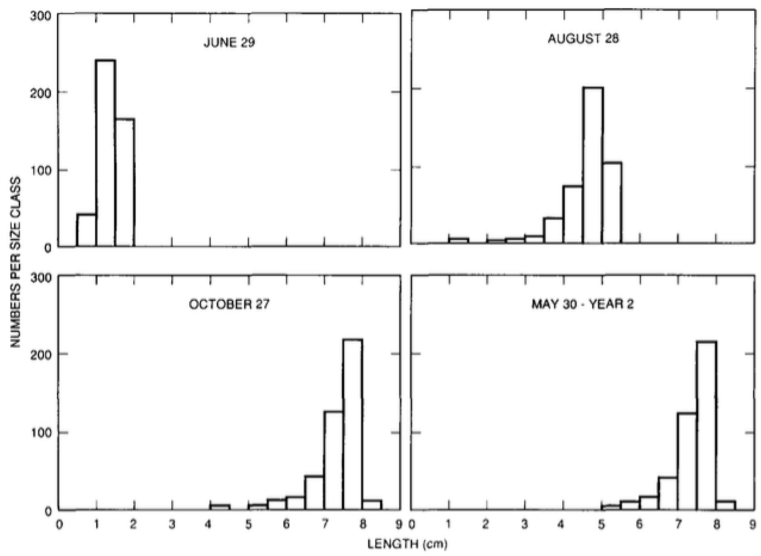


Fig. 4. Size–frequency distribution of smallmouth bass on four dates following spawning. The number of nests was 5, resulting in a cumulative 49928 eggs during the spawning reason (i.e., before June 29 of Year 1) and 7660 swim-up fry. A sample of 500 individuals were simulated. The 407 survivors of the sample population (corresponding to 6235 total surviving individuals) at May 30 of Year 2 are assumed to have survived the winter and to be recruits to the yearling class. Predation mortality bass beyond the swim-up stage was assumed to be very low.

- They then change indivdiual parameters to analyze the effect. For example in the figure below they compare the control set of parameters (Panel A) to experimental parameter sets (Panel B and C). For example the Panel A-C contrast illustrates the effect of changing the behaviour of the fish prey (the water flea *Daphnia*).

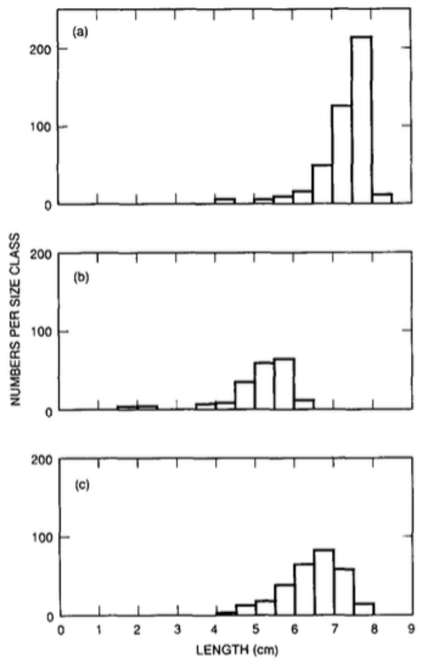


Fig. 5. Sensitivity of the young-of-the-year size–frequency distribution at the end of the growing season to an increase in the loss rate through respiration and a decrease in prey encounters per day. (a) Unperturbed case, same as Fig. 4. (b) Increase in rate of respiratory loss by factor of 1.6. (c) Increase in prey avoidance by factor of 1.6.

Lecture 6.2 Cellular Automata

Cellular automata (CA) is a computational model ideal for capturing interactions in a highly spatially-structured environment. CA models consist of a grid of **cells**. Individual cells often represent an individual (or the empty place for an individual). Each cells can be in one of a finite number of **states** (examples of states could be empty/full or susceptible/infected/recovered/dead). The state of each cell is updated based on a set of **rules** that define how it responds to the states of its neighbouring cells. Cellular automata have been used to model and simulate a variety of complex systems in with numerous applications particularly in ecology and epidmeiology.

Key characteristics of cellular automata include:

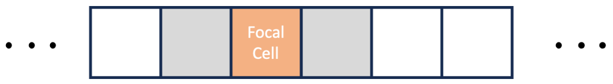
1. **Grid Structure:**
   - The cellular automaton is represented as a regular grid or lattice of cells. The grid can be one-dimensional, two-dimensional, or even three-dimensional.
2. **States:**
   - Each cell can be in one of a finite number of discrete states. The set of possible states is typically predefined.
3. **Neighbourhood:**
   - The state update of a cell depends on the states of its neighbouring cells. The neighbourhood of a cell defines which neighbouring cells influence its state. Common neighbourhood configurations include **von Neumann** (four neighbours) or **Moore** (eight neighbours) neighbourhoods.
4. **Rules:**
   - The rules of the cellular automaton define how the state of each cell evolves over time. The rules are applied simultaneously to all cells in the grid during each time step.
5. **Time Steps:**
   - The simulation progresses in discrete time steps. At each time step, the state of each cell is updated based on the defined rules and the current states of its neighbours.
6. **Boundary Conditions:**
   - Boundary conditions determine how the simulation handles cells at the edges of the grid. Common choices include periodic boundary conditions (the grid wraps around) or fixed boundary conditions.

Example: 6.6 The 1D Contact Process

The simplest lattice is a 1D lattice. Let's model an SIS on a 1D lattice, where the transmission rate is proportional to the number of infected neighbours and the recovery rate is constant.

1. Draw the lattice and denote the 'neighbourhood' of a focal cell

The *neighbourhood* in this model is simply the cells to the left and right of the focal cell.



2. What are the possible states in the model?

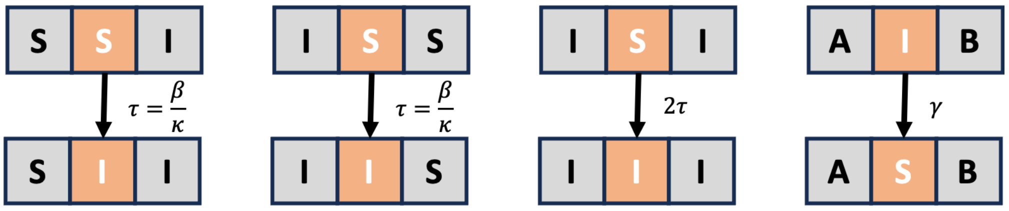
Each cell can be susceptible (S) or infected (I). Let 'A' and 'B' denote the state of neighbouring cells when the rules do not depend on this state.

3. What would be a reasonable initial condition? What would be a reasonable boundary condition?

If we want to understand early spread of the disease, lets start by considering a single initially infected cell at a random location.

One easy boundary condition is to wrap the lattice in a cricle or 'torus' in 2D. We can implement this computationally by using the Modulus command.

4. What are the rules of the contact process?



5. Build the cellular automata model in python.

**6. How does the number of infected individuals change over time? How fast does the disease spread in space?

7. How does this compare to a non-spatial stochastic model?