Using NetLogo for Modeling of Virtual Worlds

Glenn Ledder

Department of Mathematics
University of Nebraska-Lincoln
gledder@unl.edu

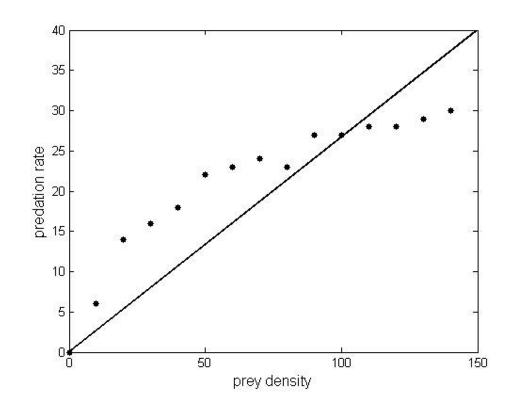
July 18, 2023

Talk Structure

1. Introduction

- ‘Beanbag Biology’ (Jungck, Gaff, Weisstein, 2010) is the teaching of mathematical models in biology by
    1. use of physical manipulatives
    2. interactive exploration of computer simulations
    3. derivation of mathematical relationships from core principles
    4. analysis of real data sets
- Independently, I coined the phrase ‘virtual laboratory’ for item 2, above (PRIMUS, 2007).
- One of my original examples was based on C.S. Holling’s 1959 paper on predation models (ie: *functional response*).
- Here I present an update for 2023, using NetLogo instead of the original Python implementation.

1.1 Predation and Prey Density



- As prey density increases, predator capture rate increases, but saturating to an upper bound.

1.2 Holling's Manipulative

1. The 'prey' are sandpaper disks attached to a plywood board with thumbtacks.
2. The 'predator' is a blindfolded student.
3. The student taps the board in random locations at a steady rate.
4. If (s)he find a disk, (s)he removes it and places it in a cup before returning to the search.
5. Students record prey density and captures over a fixed time.

1.2 BUGBOX-predator-original

- My 2007 insight: We can use a computer simulation instead!
- **BUGBOX-predator-original.nlogo** implements a virtual predation experiment in NetLogo.
  - A lady beetle wanders through an arena scattered with aphids.
  - It stops to feed when it finds an aphid.
  - In the standard (replacement) mode, new aphids appear each time one is 'eaten'. This keeps the prey population constant.
  - The simulation ends at a fixed time.
  - The aphid population and total number of captures are recorded.

1.3 NetLogo Overview

NetLogo is a platform for coding agent-based models.

- Primary control is in the **Interface** tab, which contains a **display** window, **data entry boxes**, **buttons**, and **output**.
- An **Info** tab contains program documentation.
- A **Code** tab contains the program code.
  - Variable Declarations
  - Procedures
    - Blocks of code triggered by **buttons** or other procedures.
  - Reporters
    - Functions that do calculations and are called by procedures.

1.3 NetLogo Features

- The "world" is composed of **patches** arranged in a 2-dimensional rectangular structure (usually wrap-around).
- **Turtles** (agents) have built-in attributes as well as user-defined attributes.
- Different breeds of turtles (eg. rabbits and coyotes) can have different user-defined attributes.
- Agent-based rules can be applied to all turtles of a given breed without writing loops.
- Program statements allow for easy filtering of turtles by attribute so as to apply rules to subgroups.
  - Example: In a foraging model, we can apply some rules to turtles that are moving and others to turtles that are feeding.

1.3 Some NetLogo Code Snippets

```
to go
  ...
  foreach range maxticks
   ifelse move? [do-move] [do-wait]
  ...
end

to do-move
  let x xcor ; xcor is a built-in turtle variable
  let y ycor
  ask cells with [xcor = x and ycor = y] ;; here always one
    if shape = "aphid" ;; shape is either aphid or hex
    ··· ;; update shape, captures, move?, endtick (for not moving)
end
```

2. Mechanistic Modeling with NetLogo

- A NetLogo program for mechanistic modeling should contain
  1. An agent-based model in the **Code**.
  2. A **visual representation** of the agent-based model.
  3. **Data entry boxes** for choosing experiment parameters.
  4. A **setup** button to prepare for experiments.
  5. A **data entry box** for choosing an independent variable value.
  6. A **go** button to do individual experiment runs.
  7. Numerical and visual **output**.
- Students use insights from the visual representation to derive a mechanistic model.
- Students fit their model to the data outside of NetLogo.

2.1 BUGBOX-predator-original Interface Components

1. A **Chooser** and a **Switch** give the user 4 experiment choices.
2. The **setup** button prepares for the experiment.
3. A **Display** window shows the system in real time.
4. A **Slider** allows the user to set the aphid population value.
5. The **go** button runs the simulation for a fixed duration.
6. Two **Monitor** windows show the experiment status.
7. An **Output** window shows the parameters and outcomes.
8. A **Plot** window shows the accumulated data.
9. The **save output** button writes the data to a .csv file.

2.1 Using BUGBOX-predator-original

- Start with *P. speedius* and replacement-on.
- Do some runs with different aphid populations.
- Watch the simulation and look at the plot of captures vs density.
- Key insight to be gained (from the manipulative or the virtual laboratory):
  - Scarce prey → most time is spent in searching
  - Abundant prey → most time is spent in handling

2.1 Derivation of the Holling Type 2 Model

$\mathbf{x}$: prey density

$\mathbf{y}(\mathbf{x})$ captures in unit time

$\mathbf{s}$: search speed (area/time)

$\mathbf{h}$: handling time per capture

$$\frac{\text{food}}{\text{total } t} = \frac{\text{search } t}{\text{total } t} \cdot \frac{\text{area}}{\text{search } t} \cdot \frac{\text{food}}{\text{area}}$$

$$\mathbf{y}(\mathbf{x}) = f(\mathbf{y}) \cdot \mathbf{s} \cdot \mathbf{x}$$

$$\frac{\text{search } t}{\text{total } t} = 1 - \frac{\text{handling } t}{\text{total } t}$$

$$f(\mathbf{y}) = 1 - \mathbf{h}\,\mathbf{y}(\mathbf{x})$$

$$\mathbf{y} = \frac{\mathbf{sx}}{1+\mathbf{shx}} = \frac{\mathbf{h}^{-1}\mathbf{x}}{\mathbf{s}^{-1}\mathbf{h}^{-1}+\mathbf{x}} = \frac{\mathbf{qx}}{\mathbf{a}+\mathbf{x}}$$

2.2 Some Coding Details

- Each patch contains a single 'turtle' of breed 'cell'.
  - NetLogo makes it easy to identify neighboring cells.
  - Aphids differ from empty cells only in the choice of image to display. Kill aphid: change cell shape from "aphid" to "hex".
- **reset-ticks** cannot be used between runs because that would erase any plot.
  - Simulations are timed using a **foreach** loop, with ticks used only to trigger display updates.
- To obtain a plot with any independent variable other than time, the plot commands have to be in the **Code** rather than the **Interface**.

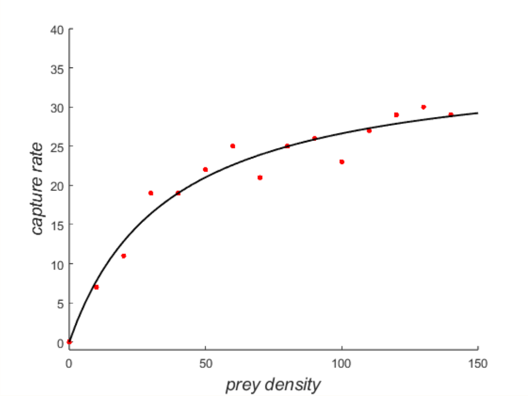
3. Empirical Modeling with NetLogo

- A NetLogo program for empirical modeling should contain
  1. All elements from the mechanistic modeling version.
  2. A **do a set** button that automates the selection of independent variable values.
  3. Additional code that fits appropriate models to the data and calculates AIC.
  4. A **plot models** button that runs the fitting code and plots the different models.
- Students use the results to check their curve fitting algorithm and AIC calculation.

3.1 Using BUGBOX-predator-analysis

- Run a complete experiment.
  1. Choose the species and replacement policies and hit **setup**.
  2. Hit **do a set** and **plot models**.
  3. Hit **save output**.
  4. Use a CAS or Matlab/R to parameterize the models and calculate AIC. Compare results with those calculated by the program.
- Make sure you do both species. *P. steadius* is usually better fit by the less accurate linear model.
- Without replacement, how do you know what value to use for prey density?
  - Changing from start value to some weighted average is a simple modification to the program. How much does this change the results?

3.1 Results



3.2 Parameter Studies

- A NetLogo program for a parameter study should contain
  1. All elements from the empirical modeling version.
  2. At least one **data entry** box that allows a parameter to be specified by the user.
  3. A **plot** that shows the effect of the parameter in some way.
- Students use experiments to collect data on the effect of a parameter.
- It is important for students to design their own scheme for visualizing that effect.
  - Example: For effect of handling time, we could plot best fit q vs h and check against theory.

4 More Examples

- BUGBOX-population.nlogo shows the population dynamics of a stage-structured population in discrete time.
  - Students develop models for a sequence of four species, working from overly simple to realistic.
- forager.nlogo is a platform for studying optimal giving-up time. foraging-auto.nlogo is a version intended for parameter studies.
  - Unlike most NetLogo library examples, this uses a realistic differential equation model for patch regrowth.

4 Take-Home Messages

- Virtual laboratories give students a way to collect real data, albeit not for a real-world setting. They can be used to teach mechanistic modeling and empirical modeling.
- NetLogo is a convenient platform for creating virtual laboratories using agent-based models.
- Experiments and data analysis can be automated in NetLogo if you know how to do it.
- The agent-based models used in NetLogo can include differential equations (eg, forager.nlogo)

5 Shameless Self Promotion

- My new book, **Mathematical Modeling for Epidemiology and Ecology**, is now available from Springer (call your school librarian). Features include
  - empirical modeling, mechanistic modeling, and dynamical systems analysis
  - a LOT of epidemiology models and many ecology models
  - case studies, such as onchocerciasis models and analysis of linearization methods for fitting the Michaelis-Menten model
  - linked problem sets that are like projects, but distributed across multiple sections
  - original models (eg, vaccination population dynamics) and methods (eg, asymptotic simplification for dynamical systems analysis)
- Contact me at gledder@unl.edu with questions or comments. If anyone wants to have me visit, I'm interested.