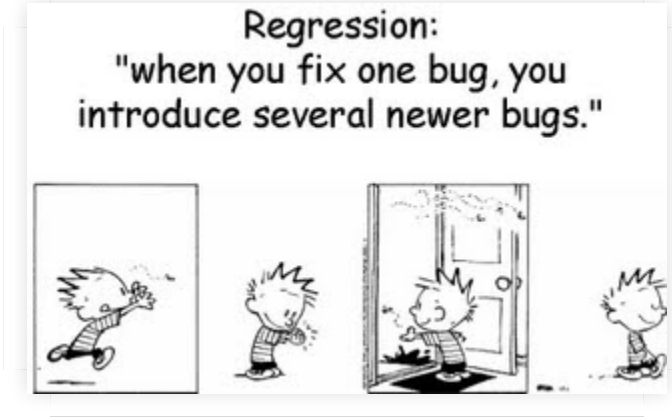
Implementing and Testing an Agent-Based Model

M. Drew LaMar
October 14, 2016



Introduction to Quantitative Biology, Fall 2016

Class announcements

- Reading assignment for Monday - Chapter 6 (THERE IS A QUIZ)
- There is no lab assignment due next week - only Homework #4
- We will use lab next week as a review session for the second exam

The ODD Protocol - Learning Objectives

- Be able to name the benefits of using a “Materials and Methods” protocol (e.g. ODD) for reporting and organizing the steps of agent-based model design and analysis.
  - Describing
  - Understanding
  - Replicating
  - Formulating

The ODD Protocol - Learning Objectives

- Develop a firm understanding of the "Overview" and "Details" elements of ODD.

	Elements of the ODD protocol
Overview	1. Purpose
	2. Entities, state variables, and scales
	3. Process overview and scheduling
Design concepts	4. Design concepts • Basic principles • Emergence • Adaptation • Objectives • Learning • Prediction • Sensing • Interaction • Stochasticity • Collectives • Observation
Details	5. Initialization
	6. Input data
	7. Submodels

The ODD Protocol - Learning Objectives

- Develop an introductory understanding of the “Design concepts” element of ODD.

	Elements of the ODD protocol
Overview	1. Purpose
	2. Entities, state variables, and scales
	3. Process overview and scheduling
Design concepts	4. Design concepts • Basic principles • Emergence • Adaptation • Objectives • Learning • Prediction • Sensing • Interaction • Stochasticity • Collectives • Observation
Details	5. Initialization
	6. Input data
	7. Submodels

The ODD Protocol - Learning Objectives

- Understand, from its ODD description, the model we will program and use in chapters 4 and 5.

Butterfly Model ODD - Purpose

The model was designed to explore questions about virtual corridors. Under what conditions do the interactions of butterfly hilltopping behavior and landscape topography lead to the emergence of virtual corridors, that is, relatively narrow paths along which many butterflies move? How does variability in the butterflies' tendency to move uphill affect the emergence of virtual corridors?

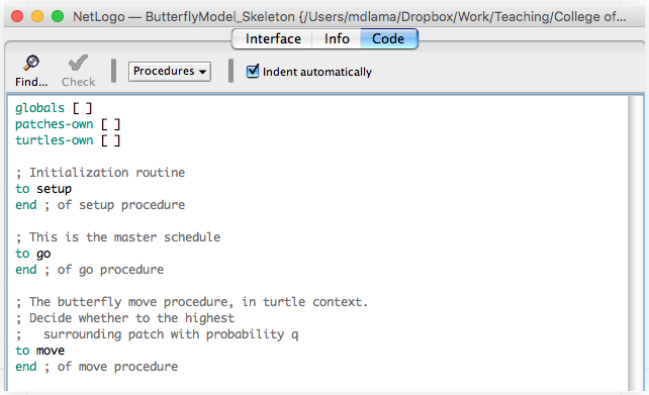
Question: What are the explanatory variables and processes and what is the system output we are interested in?

Chapter 4: Implementing an ABM - Learning Objectives

- Be able to translate a model from its written description in ODD format into NetLogo code.
- Be able to define global, turtle, and patch variables.
- Become familiar with NetLogo's most important primitives, such as `ask`, `set`, `let`, `create-turtles`, `ifelse`, and `one-of`.
- **Start learning good programming practices**, such as making very small changes and constantly checking them, and writing comments in your code.
- Produce your own software for the Butterfly model described in chapter 3.

Implementing an ABM - Tips & Tricks

- Program the overall structure of a model first, before starting any of the details.



- Before adding each new element, conduct some basic tests of the existing code and save the file.
- For the love of all things made of chocolate, indentation and organization is your friend!

NetLogo Brainteaser

Discuss: When you click the new setup button, why does NetLogo seem to color the patches in spots all over the View, instead of simply starting at the top and working down?

Answer: NetLogo always, by default, goes through turtles and patches in random order. This is an attempt to remove a possible confounding process - ***agent ordering***. This is an example of ***asynchronous updating.***

Code Commenting

This is the programmers “Do as I say and not as I do!” Think of commenting as taking notes in your learning.

> “Comments are needed to make code easier for others to understand, but they are also very useful to ourselves: after a few days or weeks go by, you might not remember why you wrote some part of your program as you did instead of in some other way.”

Code Commenting - What do I comment?

- Briefly describe each procedure (**section**) or nontrivial piece of code (**paragraphs**).
- Explain meaning of variables (*Pull from ODD if possible*)
- Document *context* of each procedure (*specific to ABMs - turtle, patch?*)
- Label ending of procedures and sometimes `if` or `ifelse` statements (i.e. after closing using `]` character)
- Use `;----------` to separate procedures in long programs.
- Comment pieces of code that are used for *unit testing* or *temporary output*.

```
show (word elev1 " " elev2 " " elevation)
```

Code Commenting - Caveats

The point is to **make your code readable and understandable**, so some caveats:

- There is such a thing as too much commenting - try and make your code unobfuscated (e.g. if there are multiple ways to accomplish something, choose the readable over the elegant and short)
- Use tabs and blank lines to show code organization.

Chapter 5: From Animations to Science - Learning Objectives

- Learn the importance of version control.
- Understand the concept that using ABMs for science means:
  1. measuring system response and behavior through producing quantitative output, and
  2. conduct simulation experiments (**in silico**).
- Perform your first simulation experiment.
- Define and initialize a global variable by creating a slider or switch.
- Develop an understanding of what reporters are and how to write them.

Chapter 5: From Animations to Science - Learning Objectives

- Start learning how to find and fix mistakes in your code.
- Learn to create output by:
  1. writing text to an output window,
  2. creating a time-series plot, and
  3. exporting plot results to a file.
- Create a version of the Butterfly model that uses real topographic data by importing data into NetLogo.

Programming Note: Moving global variables to the Interface

- GUI (Graphical User Interface) elements on the Interface both *define* and *initialize* a global variable. What this means if you move a variable to the Interface:
  - You should **comment out** the variable in the `globals [ ]` chunk of code. I would also put an additional comment at the end stating it is on the Interface. For example:

```
globals
[
  ; q  ; (Interface Slider) Probability that ...
]
```

Programming Note: Moving global variables to the Interface

- GUI (Graphical User Interface) elements on the Interface both *define* and *initialize* a global variable. What this means if you move a variable to the Interface:
  - You should **remove** or **comment out** the variable initialization in the `setup` procedure. If you choose to comment and not remove, I would also put the same comment from the `globals` chunk in here as well. For example:

```
to setup
[
  ; set q 0.2  ; (Interface Slider)
]
```

Programming Note: Initializing variables

- New variables have a default value of **zero** until assigned another value.
- Good practice is to explicitly set it to zero anyways in the code.

Programming Note: Troubleshooting tips

Tips to reduce probability of getting stuck and increase probability of getting unstuck.

- Proceed *slowly*! Add small chunks of code at a time, check for *syntax* (using `Check` button) and *runtime* (using `Go` button) errors, and verify and validate code (i.e. is it doing what it is supposed to do?) For verification and validation:
  1. Create a `Step` button that steps through the program one tick at a time.
  2. Use `show` statements throughout your code.
  3. Use Agent Monitors.

Programming Note: Troubleshooting tips

Tips to reduce probability of getting stuck and increase probability of getting unstuck.

- Think logically!
  1. “It should work this way because…”, i.e. critically assess output and results and see if they make sense
  2. “Let me try this and see what happens…”