Testing and Validating Models

EES 4760/5760

Agent-Based and Individual-Based Computational Modeling

Jonathan Gilligan

Class #7: Thursday, September 16 2021

Organization

Organization

- Download culture dissemination model from the Download page on the course web site, https://ees4760.jgilligan.org/files/models/chapter_06/CultureDissemination_Untested.nlogo, or Brightspace.
  - The paper describing the culture dissemniation model is also on the Download page, https://ees4760.jgilligan.org/files/models/chapter_06/axelrod_culture_dissemination_1997.pdf.
- Sit together in pairs, with a partner.
- Team Projects:
  - By the end of the day Friday (tomorrow) fill out the survey on Brightspace
    - Choose which project you prefer
      - Adaptive behavior (Business investor model, Ch. 10)
      - Agent interactions (Telemarketer model, Ch. 13)
    - Optionally name team members

Finding and Fixing Errors

Classes of Errors

- Typographical (typing `pxcor` when you mean `pycor`)
- Misunderstanding NetLogo language:

```
ask turtle 5 [
  let neighbor-patches patches in-radius 2
  ask neighbor-patches [set pcolor green]
]
```

versus

```
ask [patch-here] of turtle 5 [
  let neighbor-patches patches in-radius 2
  ask neighbor-patches [set pcolor green]
]
```

- Wrong display settings (wrapping)
- Run-time errors (e.g., division by zero, forgetting to initialize globals, etc.)
- Logic errors **(hard to find)**
- Formulation errors **(hard to find)**

Independent Re-Implementation of Submodels

- If your model needs a tricky calculation:
  - Try it in another format:
    - spreadsheet,
    - scripting language (Python, R, Matlab, etc.)
  - Compare to NetLogo results

Culture-Diffusion Model

Culture-Diffusion Model

- **Entities** and **State Variables**:
  - Each patch is a village (10 × 10)
    - Culture is characterized by 5 numbers (integers 0–9)
      - example: `58354`
    - Similarity of two villages = (# matching numbers) / 5
      - Goes from 0–1
      - `04976` and `44873` have a similarity of 0.4 (2 matches)

Culture-Diffusion Model

- **Process Overview**
  1. *Cultural interaction*
     - Each tick *one* random village is active.
     - Active village picks a random partner from neighbors sharing an “edge.”
     - Maybe the active village interacts with partner
       - Probability of interacting = similarity.
       - The more similarity, the more likely to interact
       - Two villages that have nothing in common won’t interact.
     - If they interact, active village copies one of the partner’s culture numbers.
  2. *Output*: Update patch colors and graphs
  3. If no patches have interacted for 1000 ticks, it stops.

Expected behavior

- We expect clusters of villages with:
  - Clusters represent different cultures
  - Different villages in the same cluster become increasingly simmilar.
- Color patches according to mean similarity with neighbors:
  - Compare to four neighbors: up, down, left, right
  - black if mean *similarity* = 0 (nothing in common with any neighbor),
  - white if *similarity* = 1 (identical to all neighbors),
  - shades of red in between.

New NetLogo Primitives

- `myself`:
  - `self` refers to the current turtle or patch
  - `myself` refers to the turtle or patch that asked the current turtle or patch to do something.

```
ask turtles-here [ set color [color] of myself ]
```

- `neighbors4` of a patch gives the four neighboring patches that share an edge (up, down, right, left, but not diagonal).
  - A patch on an edge or corner will have fewer than four neighbors.

NetLogo Lists

- Each patch has 5 variables (one for each cultural feature)

```
patches-own
[
  ; Patches have five "cultural feature" variables
  Var1 Var2 Var3 Var4 Var5
  mean-similarity  ; The mean similarity over four neighbor patches
]
```

- To compare patches, make a **list** of the differences between the variables

```
let var-list (list)
if (Var1 != [Var1] of the-neighbor) [set var-list fput 1 var-list]
if (Var2 != [Var2] of the-neighbor) [set var-list fput 2 var-list]
if (Var3 != [Var3] of the-neighbor) [set var-list fput 3 var-list]
if (Var4 != [Var4] of the-neighbor) [set var-list fput 4 var-list]
if (Var5 != [Var5] of the-neighbor) [set var-list fput 5 var-list]
```

  - `(list)` creates an empty list
  - `fput 1 var-list` puts a `1` at the front (first place) of the list
    - `lput 1 var-list` would put a `1` at the end (last place) of the list
  - If this patch has `04976` and `the-neighbor` has `44873`, then `var-list` will end up with `[5 3 1]`

Updating Cultural Variables

- On each tick, the model picks **one** patch and updates its culture
  - That patch will randomly select one of its 4 neighbors (N, S, E, and W),
    - Make a list of the features of the neighbor that differ.
    - Cultural similarity is the fraction of the 5 features that match
      - If this patch has `04976` and the neighbor has `44873`, then 2 of the 5 features match, so the similarity is 0.4
    - Probability of interaction is the cultural similarity
    - If the patch interacts with the neighbor,
      - randomly pick one of the cultural features that don’t match and change this patch’s feature to the neighbor’s value.

Test the model

Work with your partner to examine the culture diffusion model.

- Run it and see what happens.
  - Look for weird behavior that might indicate an error.
  - Save data to files with “test-output-on?” switch.
- Examine the model code
- Inspect agents. Use the `show-similarities` procedure in the agent monitor.
- Announce to the class if you find an error