Introduction to Complexity

Unit 2 Homework

(optional)

You can choose whether you want to do the Beginner, Intermediate, or Advanced level. free to discuss this part of the homework with anyone, or to ask questions about it on the course forum. Not that homework in this course is not turned in or graded.

Video solutions to these three options will be posted on the Lectures page.

Beginner Level:

1. In the logistic map, R is always between 0 and 4. Why is this the case? What happens if R is set to be greater than 4?

2. Consider the logistic map:

$$x_{t+1} = R\,x_t\left(1 - x_t\right)$$

If $R = 1$, what fixed point does x_t always go to? (You can use a calculator or LogisticMap.nlogo to help solve this.)

3. Download SimplePopulationGrowth.nlogo from the Course Materials page. Do the following modifications of the code, and see if they work!

- Change the shape of the reproducing individuals from bunnies to another shape. (To see the list of possible shapes, go to the "observer >" box below the Command Center window, and type "show shapes", followed by a carriage return, or see the list of "default shapes" at http://ccl.northwestern.edu/netlogo/docs/index2.html (and click on "Shapes Editor" under "Features" on the left sidebar).

- Change the color of the patches, using "set pcolor green" or whatever color you prefer. The list of color names you can use includes: black, gray, white, red, orange, brown, yellow, green, lime, turquoise, cyan, sky, blue, violet, magenta, pink. You can see all the color codes by going to the Tools menu and clicking on "Color Swatches".

4. Download SensitiveDependence.nlogo from the Course Materials page. Modify the code to color the background green, and have each of the two dots show a label with its coordinates (x_t, x_{t+1}). (Hint: use the "set label" command in an "ask turtles" section.)

Intermediate Level:

1. Suppose that a population of organisms grows according to the following rule:

$$n_{t+1} = 1.1 n_t$$

where n_t is the population size at generation t and n_{t+1} is the population size at the next generation. If the population starts at $n_0 = 100$ individuals, how many generations will it take until the population has more than doubled (that is, is greater than 200)? (Assume that if n_t is not an integer, the actual population size is rounded off.)

2. Prove algebraically that $x_{t+1} = 2\ (x_t - x_t^2)$ has fixed point 0.5 .

3. Download and modify SensitiveDependence.nlogo (available on the Course Materials page) as follows: Add a third initial condition, x_0'', that can be set by a slider and that is updated and plotted by the program in the same way that x_0' is updated and plotted, but with a color different from red or blue.

Advanced Level: (for people who finished the Intermediate level and want to go deeper into NetLogo programming):

Our LogisticModel.nlogo updates the population using the Logistic Model equation. This assignment is to build an agent-based (rather than equation-based) model of logistic population growth.

Build a model that has the following features:

- A slider to set the initial population.
- Turtles that reproduce (no mating necessary) with a reproduction rate set by a slider on the interface.
- Some aspect of the environment (food, space, etc.) that either limits reproduction or increases the death-rate.
- A plot that shows the population as it changes over time.
- Your code should contain as least three procedures/reporters.

Does your model show population growth dynamics that are qualitatively similar to the Logistic Model, or are the dynamics different? If different, why are they different?

Hints:

If you are new to programming and/or new to NetLogo creating this model may be quite challenging. Use the NetLogo dictionary and look at samples for the model library to get ideas. Persevere. Building a model that works is very rewarding.

It is often useful to start with pseudo-code to "outline" the situation. Most models have a high level "Go" button that runs other procedures. Perhaps your Go procedure will look like this

```
to go
ask turtles [
        look-around
        reproduce
        expire
]
end
```

Now the challenge is to just create the three simpler procedures.

You will likely want to "customize" the turtles with *turtles-own* variables. You do this be starting off the code with a block that looks something like this:

```
turtles-own [ energy-level]
```

After you do this, any turtle that gets created will have this variable. You can now say things like, "ask turtles [if energy-level = 0 [die]]" that is, to ask turtles to die if their energy-level gets to zero.

Finally, know that when you use the *create-turtles* command OR the *hatch* command, you have an opportunity to set variables in the newly created turtles. Hatched turtles will have all the attributes of the parents, including turtles-own variables. Here's an example that assumes that a *turtles-own* "age" variable has been created.

```
ask turtles with [color = green and age = 24 ] [ hatch 1 [set color red]]
```

I have asked a subset of the turtles (an "*agentset*" in NetLogo parlance) to each have one offspring. Green, 24 year old turtles will have red offspring. Unless you add additional code, all of these newborns will also have an *age* variable set at 24. Probably not what we intended.

Good luck with this assignment. Try to use online resources and the NetLogo dictionary and models library (File -> Models) when you get stuck. Getting stuck is inevitable but it is what makes getting unstuck fun. In a week or so we will post a video that explains an example program that would satisfy this assignment.