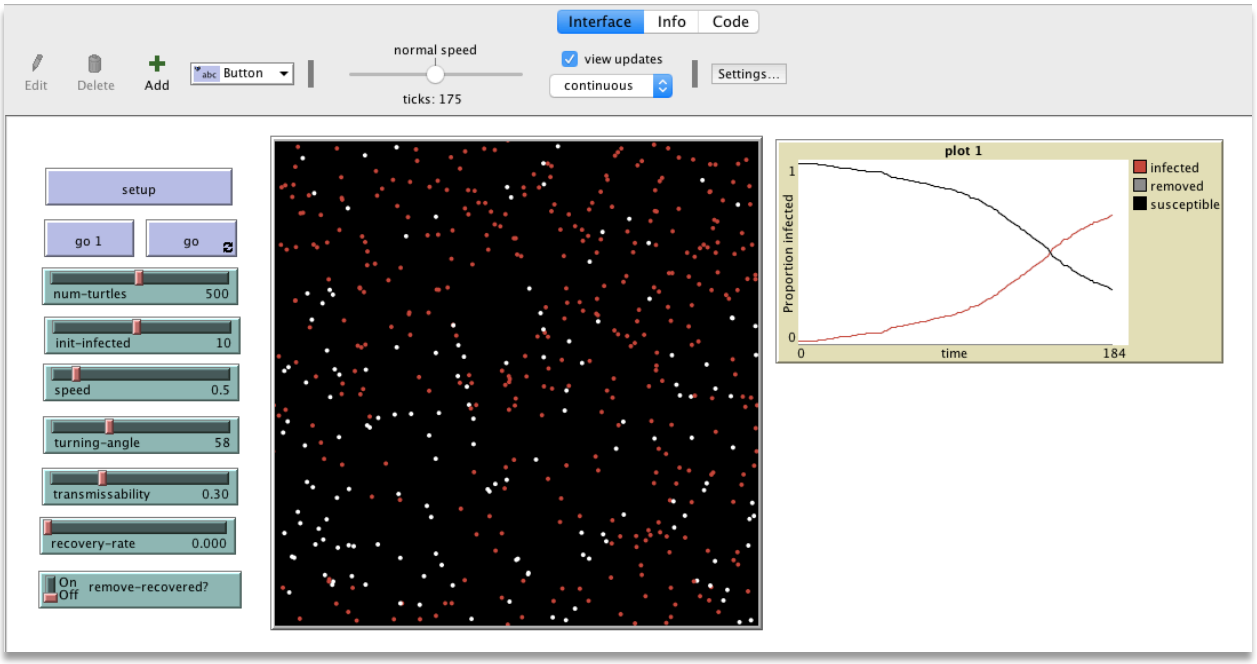
Computational Modeling of Social Behavior

Coding Challenges – Day 2

You will build an epidemiological model in NetLogo in which agents move around in space and infect one another. With the simulation you will be able to vary (1) how the infection starts, (2) how agents move through space and encounter one another, (3) how effectively the infection spreads, (4) how quickly infected agents recover, and (5) whether recovery conveys immunity. You will be able to plot the proportion of agents who are susceptible, infected, and immune. In the end, your Interface tab should look something like this:



Designing our model

Before you start coding, look through the following model description. This will help you break up the coding of the model into small, discrete tasks.

<u>Agent and their world</u>

The world is a toroidal square grid with dimensions 101 x 101 patches.

Agents are turtles that move through this space. They move constantly, and their movement is not influenced by other agents. Their color indicates their infection state: white = susceptible, red = infected, gray = removed. As their move through space, susceptible agents can become infected if they get too close to infected agents. Infected agents can recover. Depending on the disease, recovered agents may become susceptible again, or they may be "removed," meaning that they cannot become re-infected.

Initialization

First, all variables and agentsets are cleared.
Next, a user-defined number of turtles are created and placed in random locations on the grid. Their statuses as "infected" and "removed" should be set to false, and their color set to white.
Next, the disease is seeded among some number of randomly selected turtles. These turtles should have their color set to red and their status as "infected" set to true.
Finally, we reset the tick counter.

At each tick

- If every turtle is infected (global spread) or if no turtles are infected (the disease is eradicated), stop the simulation.
- For each susceptible agent:
    - If not removed, if there are any infected agents close by, become infected with a probability equal to the transmission rate.
    - [NOTE: avoid having agents able to infect other agents on the same tick on which they become newly infected]
- For each infected agent
    - With a probability equal to the recovery rate, recover.
    - If *remove-recovered?* is true, set recovered to "removed," otherwise set to "susceptible." Update turtles-own boolean variables and agents' colors accordingly.
- For each agent
    - Move:
        - First, turn left and right a random number of degrees as determined by the *turning-angle* slider.
        - Then move forward a distance equal to *speed*.

Plotting

Create a plot with 3 lines as a function of time: the proportion of agents who are (1) susceptible, (2) infected, and (3) removed.

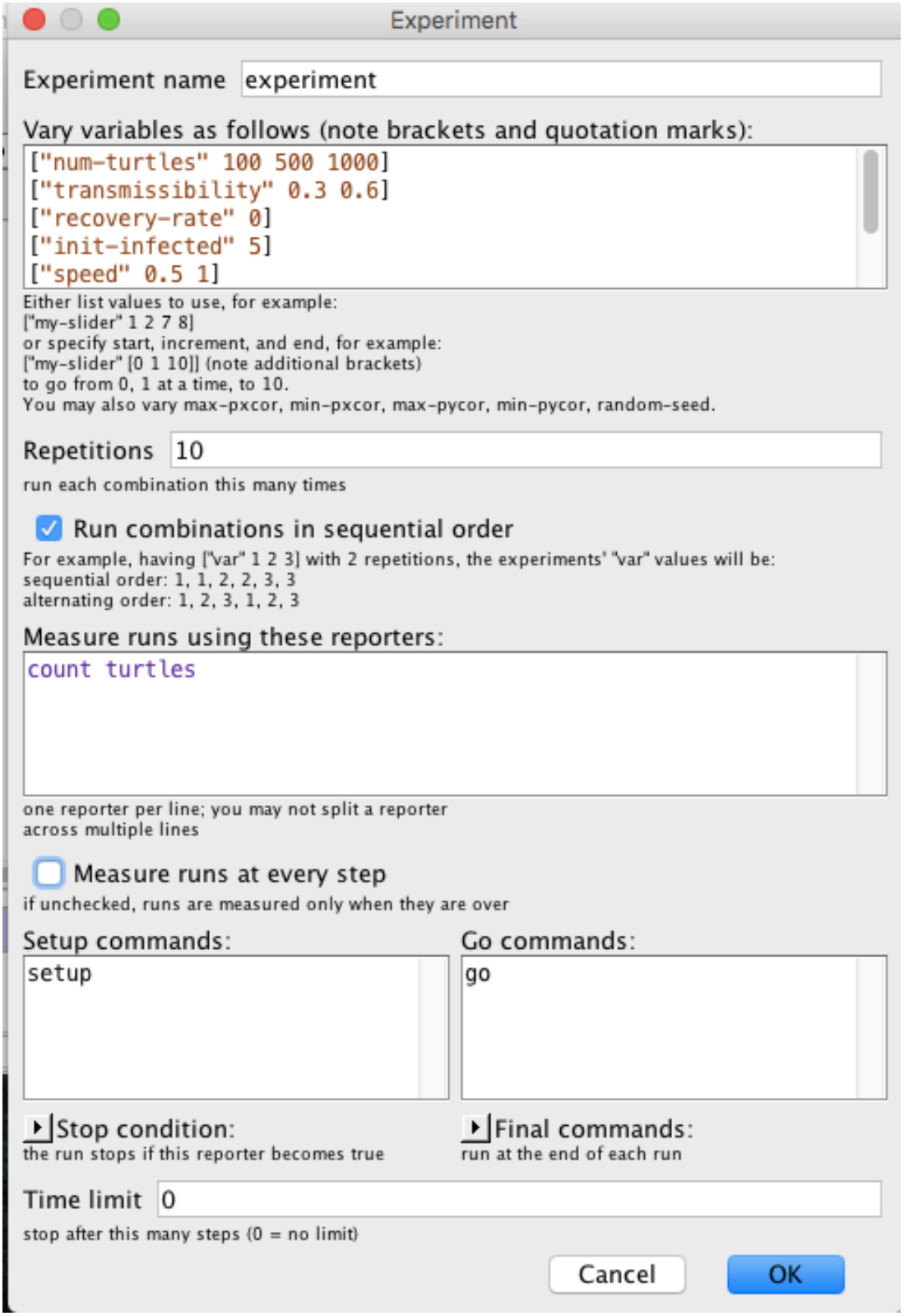
1. ANALYZING BATCH RUNS

We want to systematically study this model. This involves running many simulations and sweeping across parameter values. NetLogo has a feature to do this called BehaviorSpace. Setup a BehaviorSpace experiment to explore how diffusion rates respond to population density and mobility. For all simulations, set *recovery-rate* to zero (so this is a pure SI diffusion model). Fix *init-infected* at 5. Run 10 simulations for each parameter combination, dealing with the following values:

- *num-turtles* = {100, 500, 1000}
- *speed* = {0.5, 1}
- *turning-angle* = {0, 45, 180}
- *transmissibility* = {0.3, 0.6}

Note how the number of simulations gets very large very quickly, every for this sort of initial exploration! 10 x 3 x 2 x 3 x 2 = 360 simulations. Make sure you only collect data at the end of each run, and that each run stops automatically when all agents are infected. Note that NetLogo will automatically record the number of ticks, which is the variable interest. You will want "Table output."

Analyze the resulting data using analytical tools of your choice. What do you find? In other words, how do the four factors influence the speed of diffusion? Do they interact?



2. DETERMINING CONTACT RATE

In the derivation of the mathematical SIS model, we were able to calculate the basic reproduction number, R0, as the contact rate times the ratio of the transmissibility and recovery rates. While the latter two numbers may be considered intrinsic properties of the disease, the contact rate results from demographics and behavior. In the agent-based model we have built, three parameters influence contact rate: *num-turtles* (the population density), *speed,* and *turning-angle* (the rate at which agents move through the population and contact other agents. Experiment with the simulation to see if you change the equilibrium infection rate without altering the transmissibility or recovery rates. What do you find? Think about how would might test this systematically using batch runs.

3. COMPLEX CONTAGION

In 2007, sociologists Damon Centola and Michael Macy introduced a model of "complex contagion." This model is based on the premise that the adoption of some beliefs, behaviors, or products may require influence from multiple sources. As such, the dynamics of diffusion may be different from class SI models in which exposure to a single "infected" individual is sufficient for spread.

(a) Create a new diffusion model on a 121 x 121 grid, using patches only. Call it *ComplexContagion.nlogo*. Initialize the model with a 3 x 3 square of patches infected. Assume each patch is connected to their nearest eight neighbors, so we can use the reporter *neighbors.* Establish a slider for a parameter *threshold* that varies from 1 to 20. At each time step, each uninfected patch that has at least *threshold* infected neighbors becomes infected. When *threshold* > 1, the contagion is considered "complex." Is there a maximum threshold for the spread of infection? If so, why?

(b) Plot the proportion of patches that are infected over time. How long does it take for the infection to completely diffuse through the population as a function of *threshold?*

(c) **CHALLENGE:** Create a switch for a Boolean variable called *asynchronous?* If this is true, patches can become infected in random order. If false, all patches first determine if they will become infected on a given tick, then all of those who will change status (from uninfected to infected) do so at the same time. How does this change the spatial dynamics?

(d) **CHALLENGE:** Centola and Macy showed that the maximum threshold for the spread of infection depends in part on the network connectivity—that is, how many other agents each is connected to. Let's make a more strongly connected network. Create a switch for a Boolean variable called *bigger-neighborhood.* If this is true, do the following:

(i) Initialize the simulation with a 5 x 5 square of patches infected.

(ii) Assume each patch is connected to their nearest 24 neighbors. To do this, create a new reporter called *big-neighbors*, which can be called by a patch (this can be done just like setting up a procedure, except the first line will be "to-report big-neighbors" and you must use the command "report" followed by an agentset of the appropriate 24 patches. One way to do this is to use the NetLogo primitive *patch-set*.

How does changing the network size change the maximum threshold for whether or not diffusion will occur?