Assignment #5 (Due April 2)

This assignment will give you a hands-on introduction to agent-based simulation via the NetLogo package. For historical reasons, agents in NetLogo are called "turtles" and they live on a grid of "patches", or rectangular regions. The can also be "links" between agents, which model relationships (neighborhoods, family relationships, and so on). The assignment consists of the steps given below. **You may work in teams of two.**

1. Set up a working NetLogo installation. You can download the latest version of NetLogo from https://ccl.northwestern.edu/netlogo/download.shtml (you do not need to fill in the information on the initial web page —just click on the download button).

2. Go to the NetLogo user manual (either on the web at https://ccl.northwestern.edu/netlogo/docs/ or via the help menu in the NetLogo app). Read the Introduction ("What is NetLogo?" and "Sample Model: Party"), and work through Tutorials #1, #2, and #3.

3. After building the basic turtle model in Tutorial #3, modify the model into a simple infectious-disease model, which works as follows. Initially, all turtles are healthy (yellow-colored) except for one randomly selected turtle (red colored), and every baby turtle is born healthy. If, at any tick, the closest turtle to a given turtle j is infected and this turtle is within a radius of 3 units of turtle j, then turtle j becomes infected as well (and its icon turns red). If a turtle becomes infected at the nth tick, it will die at the $(n+k)$ th tick, where k is the value of a parameter called *illness-length*. Implementation of this model requires the following tasks.

   a. Besides the existing turtle attribute *energy*, add two new attributes: *sick* is a Boolean variable that indicates whether the turtle is infected or not, and *time-sick* is an integer variable indicating the time in ticks since the turtle initially became infected.

   b. When creating the initial set of turtles, set the color equal to yellow and *sick* equal to false. Then choose a turtle at random, set its *sick* variable to true and its color to red. Whenever you hatch a new turtle, sets its *sick* value to false and its color to yellow.

   c. In the *go* procedure, add a call to a new procedure called *check-disease* right after the call to the *eat-grass* procedure. The check-disease procedure for a turtle first tests if the turtle is sick. If so, it first increments the turtle's *time-sick* variable by 1. Then, if *time-sick* equals or exceeds *illness-length*, the turtle dies. If the turtle is not sick we identify the closest turtle within a radius of 3 units, if one exists. The NetLogo statement

```
let x min-one-of other turtles in-radius 3 [distance myself]
```

   will return this turtle as x (figure out why this works). The statement

```
if (x != nobody) and ([sick] of x)
```

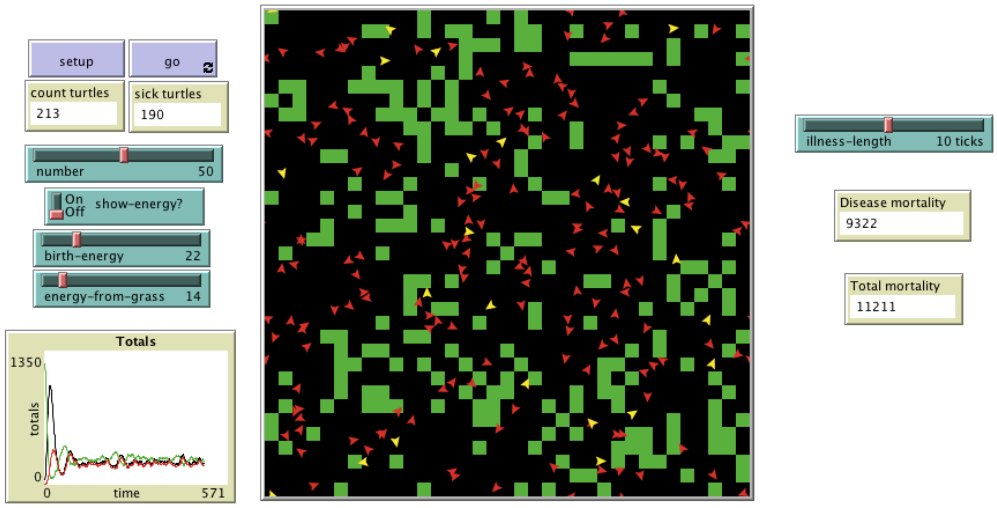
will test if there exists a turtle within radius 3 and, if so, whether the turtle is sick. If so, turn the turtle red, mark it as sick, and initialize the *time-sick* variable to 0.

d. Keep track of the number of deaths by disease and the total number of deaths. You can do this by defining some global variables at the top of the file:

```
Globals [ill-deaths total-deaths]
```

and incrementing these variables appropriately when a turtle dies. (Make sure to increment the variables **before** you execute the *die* command, or the update will get lost.)

e. Modify the interface so that (i) the monitors under the *setup* and *go* buttons now display the total number of turtles and the number of sick turtles, (ii) the plot now displays a red curve giving the number of sick turtles over time, (iii) there is now a slider to the right of the turtle field that lets you set the *illness-length* parameter to a value between 1 and 20, and (iv) there are now monitors showing disease mortality (# of deaths due to the disease) and total mortality (# of deaths due to both disease and running out of energy). The interface should look as follows:



f. Play with the model to explore its behavior! Watch how the results change when you vary the lethality of the disease. (The smaller the *illness-length* parameter, the more lethal the disease. Ebola is an example of a highly lethal disease, whereas this year's version of the flu represents a highly contagious, but not as lethal disease. Our disease is a bit strange in that a parent will not pass on the disease to its child.) Look at how other parameter changes affect the model. Think about other factors that you might want to model, e.g., a vaccination policy.

g. Now systematically explore the impact of the lethality parameter on model behavior by running a series of experiments. Specifically, read about the BehaviorSpace experiment tool at https://ccl.northwestern.edu/netlogo/docs/behaviorspace.html (or via the help menu on the app). Set up and run a series of experiments in which the *illness-length* parameter varies from 1 to 20

and the remaining parameters are fixed at *number* = 50, *energy-from-grass* = 14, and *birth-energy* = 22. Run 10 Monte Carlo replications per parameter setting, so that there will be a total of 200 experiments; the experiments don't take that long to run. For each experiment, measure the final turtle population size at the end of the simulation, i.e., after 500 ticks; call this variable turtlePop. Have BehaviorSpace write the results to a .csv file and use this file create a plot of E[turtlePop] as a function of lethality (i.e., of *illness-length*). (Use the "table" and not the "spreadsheet" option for the output.) For each lethality value from 1 to 20, you will of course estimate E[turtlePop] by the average of 10 Monte Carlo replications. On your plot, show error bars for each point that correspond to 95% simultaneous Bonferroni confidence intervals. (Recall that such intervals are based on the Bonferroni inequality and have the property that, with probability 95%, all 20 confidence intervals shown in the plot will be valid at the same time.) See if you can explain the overall shape of the curve based of your observation of the model's behavior. You can use any software that you would like to manipulate the data and create the plot. Some R code that does the job is given below, but you can also use Excel, Matlab, hand-coded Python, and so on.

So for this assignment you will need to turn in (i) the .nlogo file, (ii) a printout of your code, (iii) a screenshot as above, and (iv) the plot described in part g, along with a brief attempted explanation of the shape of the curve. For the screenshot, you can use "File/Export/Export Interface" from the Netlogo menu to create a .png image file, or you can print to pdf.

Below is some R code for creating the experiment plot.

```
setwd("./Desktop") # directory where csv file is located
z = qnorm(.99875) # for 95% Bonferroni CI with 20 points
reps = 10 # number of Monte Carlo reps per illness.length value
# Read output from NetLogo BehaviorSpace, ignore first 6 lines of metadata
T = read.csv("HW5 Lethality experiment-table.csv",skip=6)
# compute avg of final pop per illness.length
M = aggregate(T$count.turtles, list(T$illness.length), mean)
# compute std of final pop per illness.length
S = aggregate(T$count.turtles, list(T$illness.length), sd)
M$sd = unlist(S[2]) # append sd column to dataframe M
colnames(M) = c("lethality","turtle pop.","sd")
M$lower = M$"turtle pop." - z * M$sd / sqrt(reps) # lower CI bound
M$upper = M$"turtle pop." + z * M$sd / sqrt(reps) # upper CI bound
plot(M$lethality,M$`turtle pop.`,
     ylim=1.05*range(c(M$lower,M$upper)), # leave enough room for error bars
     xlab="lethality",
     ylab="final population size",
     pch=16, type="o") # black dots, with dots connected by lines
# The next command adds error bars to the plot
arrows(M$lethality, M$lower, M$lethality, M$upper, length=0.05, angle=90,
code=3)
```

If you would like to read more about agent-based simulation, you can look at Section 13.2 in Law, the tutorial at https://link.springer.com/content/pdf/10.1057%2Fjos.2010.3.pdf, or the several books about agent-based simulation on the course reference list.