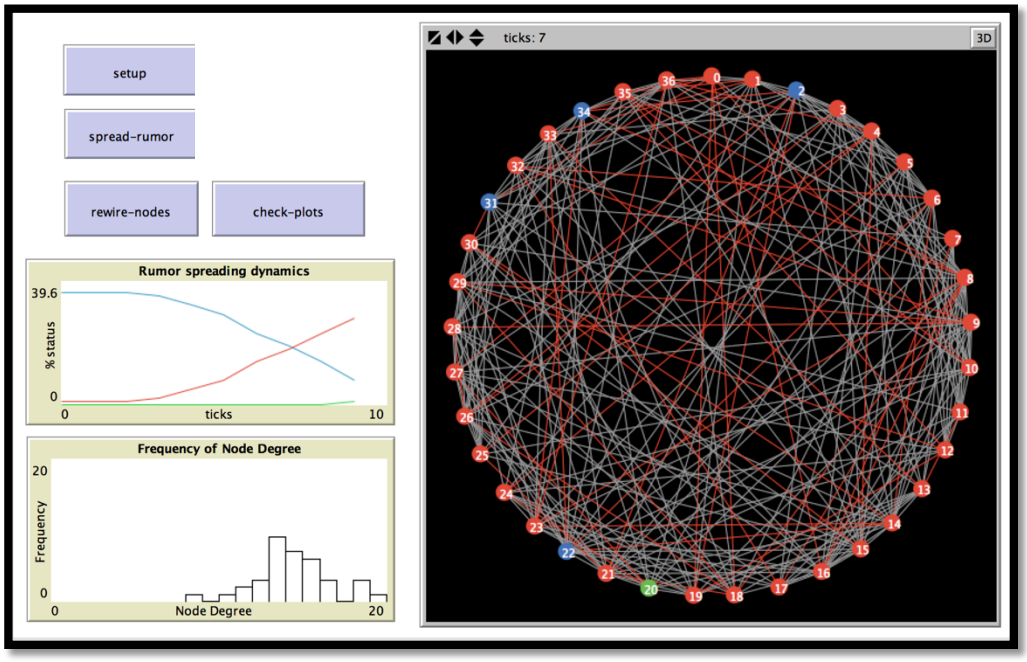
Information Spreading on Networks



Have you ever had the power go out in your home? Chances are, a lot of your neighbors had their power go out too. Why does this happen?

A power grid is a small-world network!

Recall from the previous lesson that *in a small world network, the path from any given node to any other node will be relatively short.* That means that if your neighbor's power line is down, yours is likely to go down too. We can look at a power outage as a type of information spreading on a network: in this case, the information takes the form of failure.

We can also model these phenomena (and many more!) as information spreading:

- a computer virus
- the brain's response to stimuli
- a disease epidemic

Let's get started!

Activities Overview

This module will be spread out over two weeks. You will:

1. Create a network using File I/O and modify it. **(Week 1)**
2. Simulate how a rumor spreads on a social network. **(Week 1)**
3. Create plots to investigate the behavior of the network. Run behavior space experiments to examine how changing values in the model effects the spread of disease. **(Week 2)**

You'll use these concepts from **previous** activities (including the first semester of the course):

1. Breeds
2. Links
3. Labels
4. Passing a variable to a procedure
5. Reporter procedures
6. The network extension/network formatting

You'll learn the following **new** concepts:

1. Using File I/O (Input/Output) in NetLogo
2. Behavior Space

Refresh your memory on breeds, links, labels, passing variables, reporter procedures, and the networks extension by reviewing previous material. When you're ready, move on to the new material below.

Activity 1: Information Spreading on Networks

Create a new NetLogo file. Name it *yourlastname_yourfirstname*_networksio.nlogo.

1. Create the following at the top of the file:
   - Don't forget to include the Networks extension!
   - Create a breed of **people** or **nodes** (your choice).
   - The members of your breed need their own variable; call it `status`.
   - Create two `global` variables: `spread-chance` and `stifle-chance`.

```
extensions[nw]
breed [nodes node]
nodes-own [status]
globals [spread-chance stifle-chance]
```

2. Create a `setup` procedure and add a button for it on the **interface** tab.

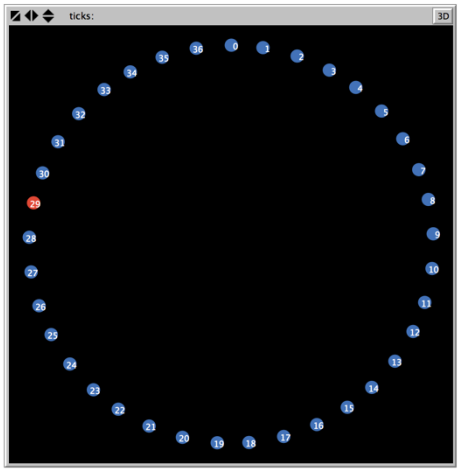
Your `setup` procedure should `clear-all` and call the procedure import-graph (you will make this next) , passing it the name of the txt file included with this module. Lastly, set both `spread-chance` and `stifle-chance` to a value between 0.1 and 1.0. You will experiment with values for these parameters in Activity 2.

```
to setup
  ca
  let file-name "networks3.txt"
  import-graph file-name
  set spread-chance 0.8
  set stifle-chance 0.1
end
```

3. Create the `import-graph` procedure by following the directions in the table below.

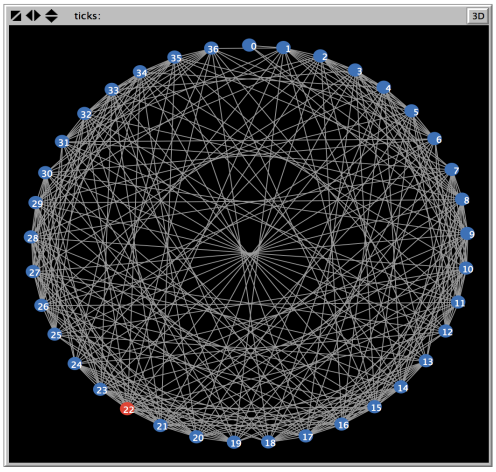
check your input file before starting	The input file must be in the same directory as your NetLogo file!
read in the file and create the nodes from the information; set node properties	We will use the txt file here. We'll open the file, read it in, and then create nodes from the information in the file. Then, we'll choose a node as the starter node for the simulation. Do this by asking one of your nodes or people to set their status to "spreader", and their color to red. This will be the one that starts the spread of disease. Other nodes will be blue—they haven't encountered the disease.

	yet. Their status is “ignorant.” REMEMBER! status is a agents-own variable; each node has its own copy so it knows its own status.
	Lastly, we’ll call the procedure import-links to create the edges between the nodes. We’ll write that next.
	
	`to import-graph[file-name]`
	`  file-open file-name`
	`  create-nodes read-from-string file-read-line`
	`  [`
	`  set shape "circle"`
	`  set label who`
	`  layout-circle (sort nodes) max-pxcor - 1`
	`  set color blue`
	`  set status "ignorant"`
	`  ]`
	`  ask one-of nodes [`
	`  set status "spreader"`
	`  set color red`
	`  ]`
	`  import-links file-name`
	`  file-close`
	`end`

test your code	Click your setup button. The result should be similar to the picture. If it is not, keep working on this part. 

4. Now we need to create the `import-links` procedure that we called in import-graph. Do this by following the instructions in the table below.

read in the file again, but create the links instead of the nodes	Use the code below to read in the txt file again. But now, for each pair of nodes that's in the file, we'll create a link between them.
	
	`to import-links[file-name]`
	`  file-open file-name`
	`  while [not file-at-end?]`
	`  [`
	`  let items read-from-string (word "[" file-read-line "]")`
	`  ask node get-node (item 0 items)`
	`  [`
	`  create-link-with node get-node (item 1 items)`
	`  ]`
	`  ]`
	`  file-close`
	`end`
	
	Notice that we're using the command **get-node**. Let's write a reporter block that will implement that command.
	
	`to-report get-node [node-number]`
	`  report node-number`
	`end`

test your code	Click your setup button. The result should be similar to the picture. If it is not, keep working on this part. 

We're ready to simulate information or infection spreading on this network!

- The first spreader node is the red node we already created. This models the information or disease originating from this node.
- Recall that you created 2 global variables, `spread-chance` and `stifle-chance`. We will use the value of those variables to determine how the info or disease will spread.

We will run the simulation using the following rules:

a) Whenever a spreader contacts an ignorant, the ignorant becomes a spreader with chance `spread-chance`. (Ignorant has a chance to turn red and become a spreader.)
b) When a spreader contacts another spreader or a stifler, the initiating spreader becomes a stifler with chance `stifle-chance`. (Spreader can't spread the info or disease anymore; it becomes a stifler and turns green)

We will run the simulation until no spreaders remain.

Follow the instructions in the table below to create the `spread` procedure that contains a loop that implements the rules.

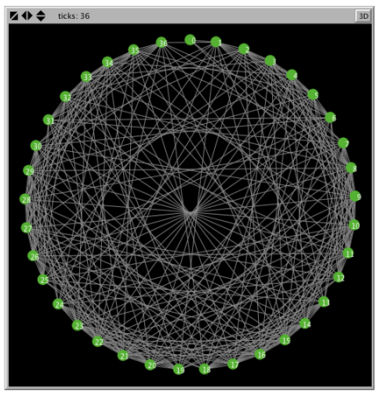
check code and create button	If something isn't working in what you've written so far, go back and fix it now. Create a button on your interface called `spread`.

loop	Enter the code below to create the simulation loop. Notice how we use the global variables as probablilites. Also notice how the agents-own variable `status` is important in this simulation. There's a command here that you may not have seen before: `ask myself.` We use this because we want to change *our* status—not the status of the agent we are asking--if we encountered another spreader or a stifler and the probability check passed.
test your code	Click your setup button, and then your spread button. The simulation should run, and at the end, you should see something similar to this. If some blue nodes remain, that's ok too.

```
to spread
  reset-ticks
  while [count nodes with [status = "spreader"] > 0][
    ask nodes
    [
      if status = "spreader"
      [
        let target one-of link-neighbors
        ask target
        [
          if status = "ignorant" and random-float 1 <= spread-chance
          [
            set color red
            set status "spreader"
          ]
          if (status = "spreader" or status = "stifler") and random-float 1 <= stifle-chance
          [
            ask myself
            [
```

END WEEK ONE!

You may work ahead and start Activity 2 if you wish.



Pre-submission checklist

☐ Is your name, date, and project title at the top of your file in comments?

- ☐ Is your code organized and formatted to course standards?
- ☐ Does it work without errors?
- ☐ Is your file named correctly?

If you answered YES to all of these questions, congrats! You are ready to turn in your project. Your instructor or TA will tell you how to turn it in.

Rubric

	Grading Rubric: **Information Spreading on Networks I–first week (10 pts)** **Information Spreading on Networks II—second week (10 pts)**
Points	Task to be completed
1	**Clarity and professionalism:** The file is named and commented appropriately, and the code is formatted and readable. **[1 point]**
9	**Activity 1:** File I/O and Network Setup Breeds and variables are used correctly. **[4 points]** The network sets up correctly when the setup button is clicked. **[5 points]**

START WEEK TWO!

Activity 2: Information Spreading Experiments

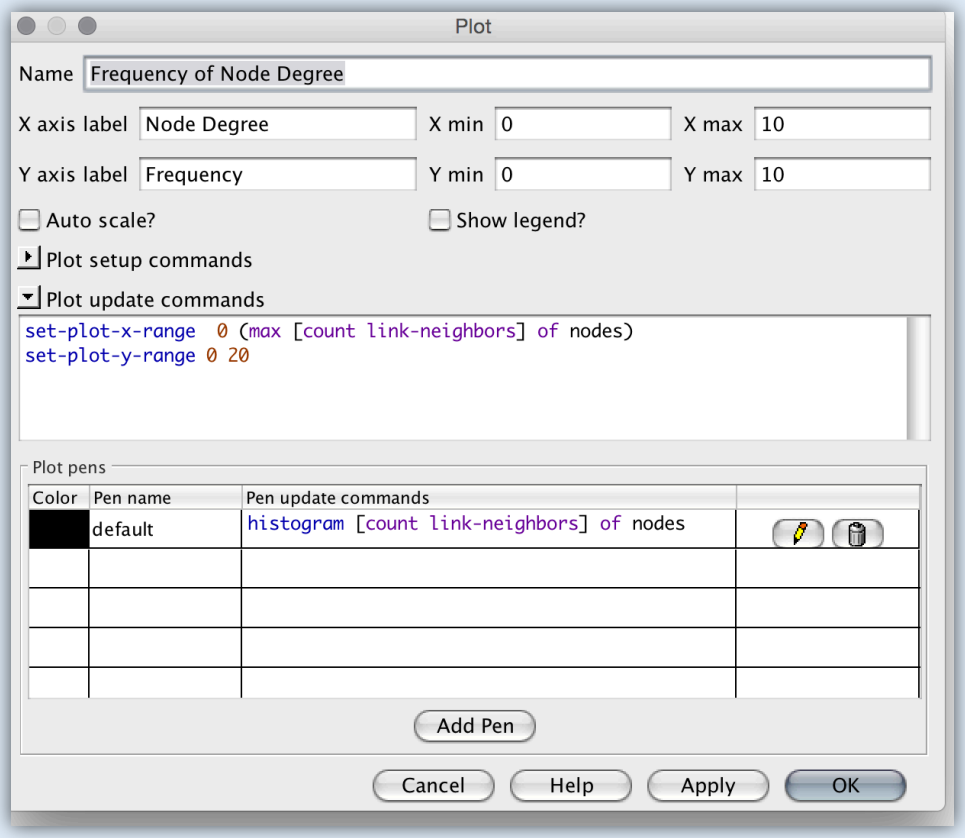
Follow the instructions in the table below to set up your program to perform experiments.

1. Write a procedure called `rewire-nodes` and create a button for it on the interface. When the button is pressed, nodes should break their current links (`ask` the link to `die`) and form a link with another node with a 40% probability.

```
to rewire-nodes
  ask nodes[
    if count link-neighbors > 0[
      if random-float 1 > 0.4[
        ask one-of links [die]
        create-link-with one-of other nodes[set color red]
      ]
    ]
  ]
end
```

Try it out! Your rewired links will show up as red.

2. Create a plot on your interface that tracks the degree of each node (see next page).
   - x-axis: node degree
   - make a histogram
   - y-axis: frequency of node degree

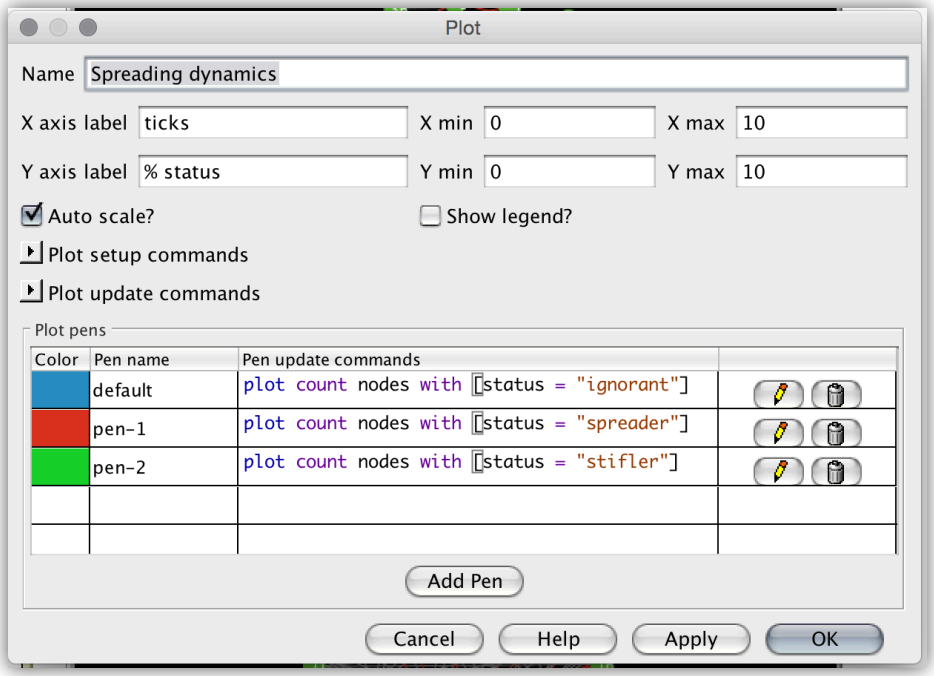


3. Create a procedure called check-plots that refreshes the plot. Add a button for it on the Interface.

```
to check-plots
      update-plots
end
```

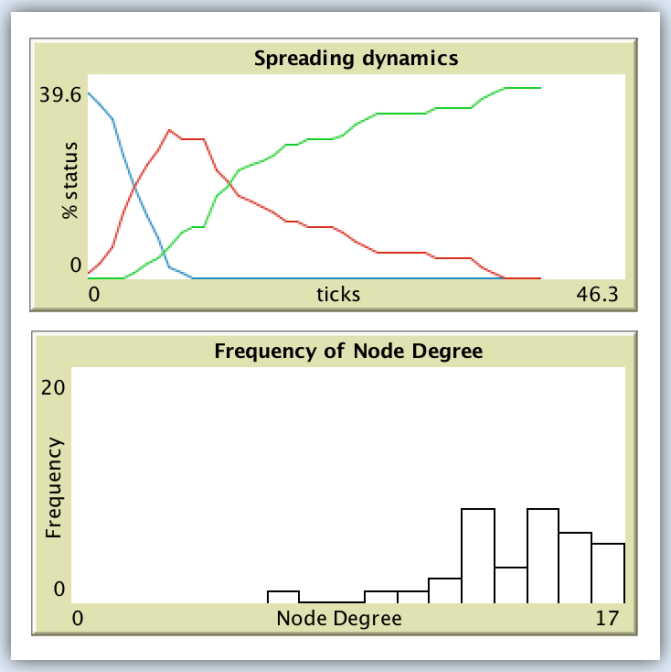
4. **Now, click rewire-nodes several times and refresh the plot. How does the plot change? What does that tell us about how the structure of the network has changed?**

5. Create another plot on your interface. This one will show trends of what status nodes are in while the simulation is running. (See next page)



6. Click `setup`, then `spread`, then `check-plots`.

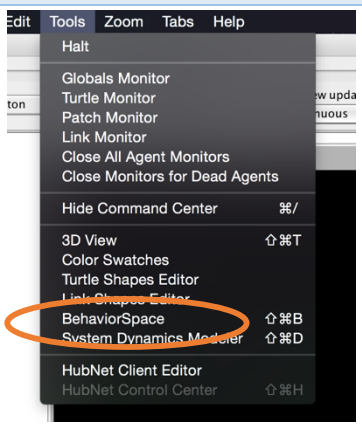
If you created both plots correctly, you will see something like the picture below.



HELP! MY HISTOGRAM ISN'T SHOWING BARS!

Right-click the Frequency of Node Degree plot to edit. In the plot pens section, click the pencil. Choose Bar from the dropdown menu.

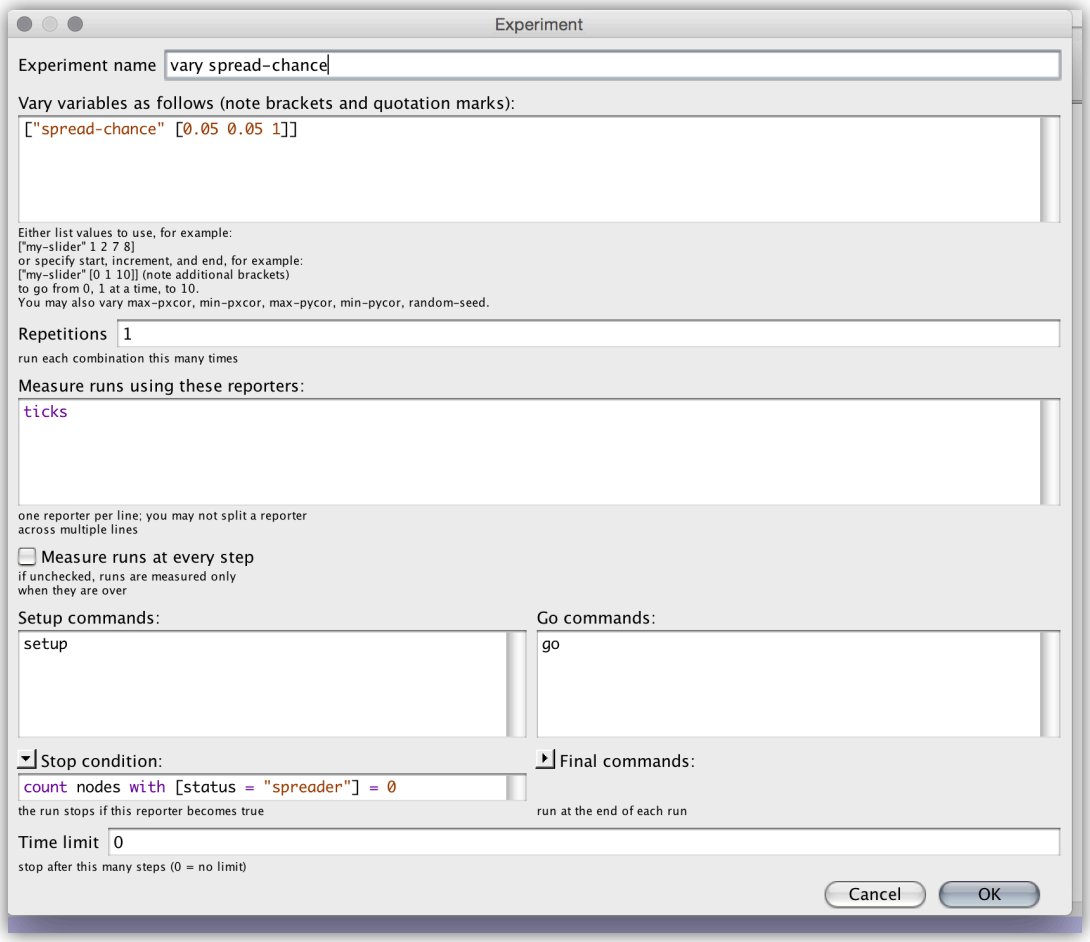
7. For the final piece, we'll run an experiment using NetLogo's Behavior Space. We want to investigate if changing the probability that a node will spread the disease effects how long it takes for the simulation to finish.
GO to TOOLS -> BehaviorSpace and create a new experiment.



EXPERIMENT 1

Fill out the experiment as in the picture below and click OK.

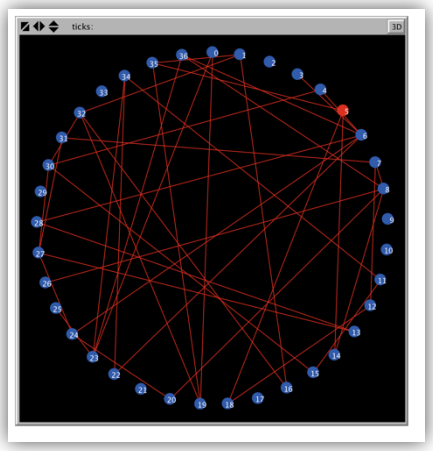
BE CAREFUL THAT ALL FIELDS MATCH—INCLUDING CHECKBOXES—OR YOUR EXPERIMENT WILL NOT WORK.



Now go to Tools -> BehaviorSpace again. Select your experiment and run it.
Choose a table output and only 1 simultaneous run. Save the outputted sheet. You will submit it.

EXPERIMENT 2

Let's change the structure of the graph and see if that effects the info spreading. Click your `rewire-nodes` procedure many times until the graph becomes sparse, like in the picture here.



Now go to Tools -> BehaviorSpace and run the experiment in the same way. Save the outputted sheet. You will submit it.

Look at the results. How are they different from the first experiment you ran? Why do you think that is?

Pre-submission checklist

- ☐ Is your name, date, and project title at the top of your file in comments?
- ☐ Is your code organized and formatted to course standards?
- ☐ Does it work without errors?
- ☐ Is your file named correctly?

If you answered YES to all of these questions, congrats! You are ready to turn in your project.
Your instructor or TA will tell you how to turn it in.

You will be evaluated on the highlighted section of the rubric this week.

Rubric

	Grading Rubric: **Information Spreading on Networks I–first week (10 pts)** **Information Spreading on Networks II—second week (10 pts)**
Points	Task to be completed
10	**Activity 2:** Information Spreading Experimental Data Plots are correctly implemented. **[5 points]** The experiments are completed and there are two tables. **[5 points]**