CSCU9YM

Practical 4: Links and Complex Networks

To prepare for this practical, you should first read through the section on Links in the Programming Guide in the NetLogo User Manual. This will help you to understand the rather complex code you will be looking at in this practical.

There are no checkpoints in this worksheet.

Part 1: Visualizing a social network

In this part of the lab we will have some fun using a simple NetLogo program for visualizing social network data extracted from Facebook. It used to be possible for students with Facebook accounts to do this exercise using their own Facebook data. Unfortunately (but perhaps understandably) Facebook no longer permit this data to be extracted. So you will have to use my Facebook data, extracted a few years ago.

- Download the model `Facebook-visualizer.nlogo` to your home folder and open it NetLogo. Read through the Info tab to find out how the model works. (Note that the instructions for downloading your Facebook data no longer work, unfortunately.)

- Download the file `sample-Facebook-data.gdf`. This file contains lightly anonymized data from my own social network.

- Try out the tasks listed in the "Things to try" section of the Info tab. Here is some relevant information to help you understand my social network. I attended a girls' high school, but haven't seen most of my school friends for many years. My husband knows many of my relatives and newer non-work-related friends, a few of my computer scientist friends, but none of my old school friends. Can you use this information to identify which nodes are a) my husband, b) my school friends, and c) my computer scientist friends? Some of the suggestions in the "Things to try" section may give you additional information that will help with this.

- Which nodes in my network are the most central or the best connected?

Part 2: Modelling Epidemic Spread

In the last lecture we learned about a number of different types of graph or network that can be used to model contact or linkage structures in real world systems. In this part of the practical we will use NetLogo to explore how the structure of the contact network can affect the dynamics of a process taking place on that network. The process we will look at is the spread of a disease epidemic.

The disease model we will use is called the Susceptible-Infected-Recovered (SIR) model. It is used to model infectious diseases where, after recovery, the ill person becomes immune and cannot catch the disease again. Many viral diseases are like this.

- Download the model `SIR-networks.nlogo` to your home folder and open it in NetLogo. Read through the Info tab to find out how the model works.

- Set the population size to 12 (a small value makes it easier to see the links) and then use the `network-type` chooser and the `setup-network` button to create small examples of different types of networks. Try reading the code for creating each of these types of network. Can you understand how it works? Look up any unfamiliar commands in the NetLogo Dictionary. (You may find some of the code rather difficult to understand – the procedure for creating the scale-free network is particularly tricky.)

- Now set the population size to 400. Create a few more networks of this larger size, just to see what they look like. When creating random networks with a large population size, keep the percentage of links (`linkage`) very low, otherwise the network will take a long time to set up. Similarly, for small world networks, keep the probability of rewiring a node (`rewiring-probability`) quite small.

The `setup-epidemic` button starts off the epidemic by making all the people in the population "susceptible" and then making a few of them "infected". The exact number to infect is controlled by a parameter, `initial-infected`.

The `go` button controls the spread of the epidemic. On each day, contacts take place between each susceptible and his/her infected link neighbours. The parameter `p-infect` represents the probability that each such contact causes the disease to be transmitted to the susceptible. After becoming infected, a person remains ill for `recovery-time` ticks, after which he/she recovers and becomes immune to re-infection.

- Using the parameter values below, set up and then run an epidemic on a square lattice (no wrapping).

`population-size`	`400`
`initial-infected`	`10`
`p-infect`	`0.25`
`recovery-time`	`10`

- Try running the model with very small and very large values for `p-infect`, say 0.01 and 0.99. Is there a difference in the result? Try varying `p-infect` systematically to see if you can find a tipping point (or narrow region) that separates the two outcomes. This is called the "epidemic threshold".

- Can you find thresholds on other network structures? Try it and see. Instead of manually varying `p-infect`, try using BehaviorSpace, which you learned about last week, to sweep through the values of `p-infect`. Use Excel or any statistical tool of your choice to create a graph of the final number of recovered individuals (the "impact" of the epidemic) plotted against `p-infect`. Use this graph to identify the value (or range of values) of `p-infect` that is the epidemic threshold. What can you say about how the structure of the network affects this threshold?

Caution: when using random or small world networks, stick to very small values for the number of random links to be introduced / rewired, otherwise your network will take a long time to set up.

- For the next task, make a copy of the model, calling it something like `vaccination.nlogo`. You are now going to explore how vaccinating some individuals in the population affects the spread of the epidemic.

- In your new model, add a button named `vaccinate` which invokes a command called `vaccinate`. Make it a "forever" button. You will use this button after the network and epidemic have been set up, but before the go button has been pressed. The effect of this button will be to allow you to "vaccinate" individuals by clicking on them with the mouse. An individual becomes "recovered" (i.e. immune) when vaccinated, regardless of their original status.

- Add the procedure below to the bottom of the code tab:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Vaccinate turtles by selecting them with the mouse
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to vaccinate
  if mouse-down?
  [
    ;; find the closest node
    let grabbed min-one-of turtles [distancexy mouse-xcor mouse-ycor]
    ask grabbed
    [
      set breed recovereds
      set color grey
    ]
  ]
end
```

If there is anything you do not understand about this code, please ask for help.
The `vaccinate` button allows you to vaccinate as many individuals as you choose, for as long as the button is down. When you have finished you can press the button again to stop vaccinating, and then press `go` to start the epidemic. Alternatively, you can leave the `vaccinate` button running to allow you to vaccinate people during the epidemic. (You might wish to use the speed slider to slow down execution if you are doing this.)

Explore the effect of vaccinating people before the epidemic starts. Does it matter *how many people* you vaccinate? Does it matter *which people* (in terms of their location on the network) are vaccinated? Are the answers to these questions the same on all types of network, or are they different for different network types?

Scale-free networks, in particular, have been suggested as a good model of contact structures for sexually transmitted diseases. Can you use the model to get any useful insights about how to protect people from STDs by vaccination? Does it matter who is vaccinated?

For further reading on this subject, see "Networks and Epidemic Models", Keeling and Eames (2005), *Journal of the Royal Society Interface* 2(4):295-307, `http://rsif.royalsocietypublishing.org/content/2/4/295.full`