CS1210 Computer Science I: Foundations
Homework 2: Discrete Event Simulation, Revised
Due Friday, November 17 at 11:59PM

Introduction

In this assignment, we're going to be making a slate of changes to the solution to our discrete even simulator from HW2. Like for HW2, you'll be given my solution to the previous homework (in this case, HW2) as a a template.

Beyond Random Mixing

Up to now, our simulator does not distinguish between types of agents; each agent is interchangeable with other every other one. In other words, this kind of simulation makes what's called the *random mixing assumption*: every agent is equally likely to interact with any other. Unfortunately, while this may be a reasonable model for interactions amongs, say, sheep in a herd, it is not a reasonable approach to modeling human behavior. Here, we experiment with the ability to create subpopulations with different mixing behaviors within the overall population.

Subpopulations can be used to model different types of agent characteristics. We might divide a population into subgroups based on, geographic residence or age group. But unlike *e.g.*, vaccination status or masking habits, these groupings reflect differences in who the agents interact with. Kids, for example, tend to interact with large numbers of other children at school. Senior citizens may be more isolated, and tend to interact only within family groups. These are examples of the kinds of grouping we wish to model with this first rudimentary step beyond random mixing (real simulators get to be wildly more complex in this regard).

We will modify out system as follows. First, we will allow some of the configuration file parameters that currently have one value to have multiple values. So, for example:

```
N: 50, 30, 10, 10    # number of agents
```

means there are 4 types of agents (call them 0, 1, 2, and 3). Here, we are specifying a total of 100 agents, where each agent will carry an additional 'type' designation, and the number of agents of each type will be 50, 20, 10 and 10, respectively.

If you are dealing with a population divided into subgroups, you may also wish to specify the types or group memberships of the initial infecteds. So you will also need to accomodate:

```
I: 0, 2, 0, 1          # initial infecteds
```

but your code should also work if you don't care which subgroup your initial infecteds belong to, in which case you could still say:

```
I: 3
```

Aside from specifying initial infecteds, groups are meant to allow you to regulate mixing by agent type to reflect group association preferences. If your population is divided into 4 groups of type 0 through 3, your configuration file should also contain four collection of mixing parameters, indicating the types of agents a particular agent type will mix with. For example, one set of mixing parameters for this example:

```
0: 50,40,10,0
1: 40,50,10,0
2: 80,10,0,10
3: 0,0,0,100
```

tells us that agents of type 3 will initiate interactions only with other agents of type 3. Note that this does

not mean that agents of type 3 never interact with agents of other types: only that agents of type 3 do not initiate those interactions. Indeed, agents of type 2, for example, will initiate interactions with agents of type 3 10% of the time.

Where to Start

The modifications to the template file are not many, but they can be tricky and do require some thought.

First, you should extend your readConfig() function to allow for lists of values rather than just individual values. In other words, you will need to accomodate lines like those shown above. In these cases, the values returned in the configuration dictionary should be tuples of ints, such as in this configuration returned by readConfig():

```
{'N': (50, 30, 10, 10), 'I': (0, 2, 0, 1), 'm': 4, 'de': 3, 'di': 5,
 'tpe': 0.05, 'tpi': 0.2, 'rp': 1.0, 'vp': 0.2, 'mp': 0.2, 'ap': 0.8,
 'ip': 0.7, 'max': 100, 'verbose': True, 'seed': None,
 0: (50, 40, 10, 0), 1: (40, 50, 10, 0), 2: (80, 10, 0, 10), 3: (0, 0, 0, 100)}
```

Note that the keys for mixing values in the returned configurations are ints and not strings; also, you should return a warning if there is, for example, an extra mixing line:

```
4: 0,0,0,100
```

your readConfig() might complain:

```
Unexpected line '4' in configuration file.
```

Finally, note that you need only handle lists of values for N, I, and the mixing parameters in range(len(N)).

Second, you should implement a function binnedSample(k, N) which produces a set (of size specified by k) of random integers selected without replacement from th range specified by N. The function constructs its result differently, depending on the types of k and N, as described in the HW3 template. You will use binnedSample() in two places: first, in newPop() when selecting initial infecteds, and second, in sim(), when determining which agents and infected agent interacts with in the main infection loop.

Third, you need to modify newPop() to create the appropriate numbers of agents of each type, where each agent contains an additional key:value pair, *e.g.*,

```
{'state': -1, 'vaccine': 0, 'mask': 0.7741903898826571, 'type': 1}
```

for agent 72 in the configuration shown above. You will also need to make sure your initial infecteds are selected across the different types in accordance with the specification of the N and I parameters in the configuration file (see the binnedSample() specification). So if I have the following configuration:

```
N: 6, 4
I: 0, 1
```

newPop() will return a population of 10 agents, with the sole infected agent selected from {6, 7, 8, 9}.

Finally, you will need to change the way agents are selected to interact with an infected agent in the main infected loop. To see how this should work, consider an example where the configuration file contains:

```
N: 6, 4          # number of agents
I: 0, 1          # initial infecteds
0: 80, 20        # mixing params group 0
1: 50, 50        # mixing params group 1
m: 4             # mixing parameter
```

and agent 9 (from group 1) is the initial infected. In HW1 and HW2 we would simply pick randint(0, m) agents from range(N) for agent 9's interactions. Here, we still want to pick randint(0, m) agents, with half coming from each group. If at some point we are to pick interactions for agent 3 (from group 0), we would want to pick randint(0, m) agents, but with 80% of these coming from group 0 and only 20% of these coming from group 1.

Conclusion

Note that if done carefully, you should not need to make any other changes: everything else should still work as before. Indeed, loading a configuration file form HW2 should not cause problems, because none of the changes would be triggered at all. Good luck!