CS1210 Computer Science I: Foundations
Homework 1: Discrete Event Simulation
Due Friday, October 20 at 11:59PM

Introduction

A discrete event simulation (DES) is a useful tool used in many areas of science and engineering. In domains where experiments are *expensive* (*e.g.*, completely reconfiguring Amazon's supply chain or Uber's pricing structure), *unethical* (*e.g.*, human genetics, where offspring cannot be arbitrarily generated from specific parents), or *impractical* (*e.g.*, astronomy, where distant stars cannot be made to implode to test a hypothesis), simulation provides a practical means for for understanding how processes work and interact. By playing with the parameters of the model, the experimenter is then able to observe the downstream effects of these changes on the outcome. DES is used extensively to study industrial workflows, machine placements, scheduling, digital design, infection control, etc. Here, we will be building a simulator to experiment with measures to control the spread of infectious diseases like influenza, Covid19, tuberculosis, and so on. Such diseases are typically caused by a virus (*e.g.*, influenza, Covid19, RSV, cold) or sometimes by bacteria (*e.g.*, tuberculosis) and are spread by direct physical contact (*e.g.*, ebola), droplets (*e.g.*, cold), or even finer airborne particles (*e.g.*, measels, Covid19).[1]

Infectious Disease Simulation

Infectious diseases like influenza, tuberculosis or Covid19 are transmitted when an infected agent comes into contact with a susceptible (*e.g.*, uninfected and unprotected) agent. Here, we'll be working on what's called an *agent based model*, that is, a model that consists of a collection of agents (*i.e.*, independent actors), each having their own internal state and proclivities for interacting with the other agents.

Traditionally, infection simulations use models with names like SIS, SIR, SEIS or SEIR, where the acronym represents the possible states of an individual agent (S=*susceptible*, E=*exposed*, I=*infected*, R=*recovered*, although some models permit other states as well). Different diseases map different progressions through these states on different time schedules; for example, for influenza, an agent, once infected, enters an *exposed* state for roughly 2 days (sometimes referred to as an *incubation period*), then progresses to an *infected* state for roughly 7 days. When exposed, an agent begins to actively shed virus which can then lead to infections in other agents, but does not yet feel sick or exhibit any symptoms. Once the agent reaches the infected state, they continue to shed virus, but they are now symptomatic, and therefore hopefully less likely to choose to interact with other people.[2] Finally, once the infection runs its course, the agent recovers and then may, for some diseases, achieve lifelong immunity (the *recovered* state, which is no longer susceptible to reinfection; this is typical with, for example, the mumps) or may re-enter the susceptible pool, as seems to be the case for Covid19 or the common cold. Other variations are both possible and common.

With this in mind, it's fairly easy to see how a simulation might unfold. Starting on day 1, we model interactions between infected and susceptible agents, and decide whether an infection is transmitted or not. If a susceptible

---

[1] There are other forms of transmission that do not require the infecting agent to be in close proximity to the susceptible agent. We won't consider these here, but think of, for example, the transmission of malaria by an animal vector, like a mosquito, or by contact with animal droppings. The subject is both complex and fascinating.

[2] There are many variations: in an SE(IA)R disease model, some number of agents may fail to become symptomatic, entering a parallel asymptomatic state where they continue to infect others but don't feel sick or exhibit symptoms. Such agents (consider Covid19, or look up "Typhoid Mary" for another real world example) can have a profound effect on how infections spread.

agent is so infected, they are then able to infect other agents. At the end of each day, we update the state of all infected agents to model the progression of their individual disease states; in ds + de days, the agent either progresses to the recovered state or returns to the susceptible state, where they are once more prone to infection. The key aspects of this process that remain to be worked out are (1) who interacts with whom on which day, and (2) what, exactly, is the mathematical model that determines if an interaction leads to an infection.

We're going to address these last two questions in the most rudimentary way possible. For the first question (who interacts with whom), we adopt the *random mixing assumption*, which states that any agent is equally likely to interact with any other agent on any given day. While perhaps a reasonable assumption when modeling cows wandering in a field, this is clearly an over simplification when it comes to approximating human behavior (a little introspection should suffice to convince you). We make a similarly sweeping simplification to address the second question: an infection results during interaction between infected and susceptible agents with a fixed *transmission probability*, tpi, while an infection during an interaction between exposed and susceptible agents has probability tpe. We further assume that a vaccinated susceptible agent cannot be infected. Real models would consider, *e.g.*, the size and ventilation of the room, the duration of the encounter, whether the agents were wearing masks, the effectiveness of the vaccine, and so on.

Assignment

We're going to start building a fairly straightforward infectious disease simulation engine. We'll make some implementation choices that may at first blush seem odd, but will help lay the groundwork for some interesting extensions in Homework2 (should things move in that direction).

At the beginning of the simulation, you will generate a population of agents of size N. Each agent will be characterized by two values: their disease state (an integer) and their vaccination state (a Boolean). The vaccination state of the agent will be determined at creation time by flipping a weighted coin that reflects the *vaccination probability*, vp, that is, the prevalence of vaccination in the general population expressed as a float between 0 and 1. A certain number of agents will be initially infected, and then the simulation will be allowed to unfold so we can see what happens. The simulation should for as long as there are still infected agents in the population, modeling interactions and updating agent disease states each day, and finally returning the number of infecteds in the population over the course of the simulation.[3]

Our simulator will adopt an SEIR model with no asymptomatic infections. The disease we are modeling is characterized by three parameters: the *days of exposure*, de, the *days of infection*, di, and the *transmission probabilities*, a tuple of two floats denoted (tpe, tpi) between 0 and 1 that describe the probability that a susceptible agent is infected following an interaction with an exposed agent or an infected agent, respectively (the latter is generally greater). Using a tuple of values allows us to model an agent in the exposed state being slightly less infectious than an agent in the infected state thanks to the difference in viral shedding rates.

Closing Thoughts

Download the template file and follow the instructions to complete each function. The complexity of the assignment is much greater than that of a Lab, for example, but there is sufficient guidance to make it easy if you work through the problems methodically. Note that you are responsible for documenting what your code does and why you wrote it the way you did: justify any assumptions or design choices accordingly.

---

[3] Depending on the simulation parameters, we have some expectation of what these results should look like: a *pandemic curve* displays the number of infecteds as a function of the duration of the simulation. You will have become familiar with these curves during the Covid19 pandemic; remember "flatten the curve"?