Justin Edwards
PHYS 2300
Scientific Computing

Modeling an Epidemic in GlowScript

Introduction

An epidemic is defined as a widespread occurrence of an infectious disease in a community over a particular time frame. There are several ways we could show this growth, but for this lab we will be using something called the SIR model which computes the theoretical number of people infected. SIR, which stands for Susceptible – Infected – Recovered, uses a variable called R0, pronounced R naught, to determine how easily spread the infection is. Using this variable, we can compute how many in the population are in each category and display this information. On top of this, we can use the projected mortality rate to determine an estimate of how many deaths would be caused if the virus ran its course through a population of n size.

Displaying Data

There are also several ways we could display the data to make it easier to understand. Because it doesn't make much sense to show an individual dot for each person in the population, we will be using spheres for each group in the model (SIR + D) and setting their radius based on the amount in each group. In order to keep the display reasonable, we will be taking the log of the amount of people in each group and setting the radius of the sphere to that value. On this scale, even with 1 billion people in a category the radius would be set at 9 which is reasonable to display. We will also be adding graphs that show each category over time, an easy addition once the model has been set up.

Justin Edwards
PHYS 2300
Scientific Computing

Setting Up the Scene

When setting up a program, some people choose to set up the model first then worry about displaying the data, but I prefer to build the interface before building the model so I can build the model with the UI in mind. The program is going to allow the user to set values in two different categories – Population Details and Infection Details. In the population details there will be sliders for the size of the population, initial immunity (percentage), initial infections, and contact rate. In the Infection Details there will be sliders for infection rate (percentage), disease period (days), and mortality rate (percentage). From these values, R0 will be calculated and displayed and the user can start the simulation.

Start by creating variables for each of the slider values at the top of the program and set them to a reasonable default like so:

```
popSize, immunity, infections, contact, infectRate,
period, mortality = 1000000, .05, 5, 4, .06, 6, .02
```

Create the sliders with the values in the variables created and labels showing their value. Create a function for each of the sliders that updates the variable first, then the label from the variable. For the labels, python provides a handy method of formatting large numbers to have commas. When setting the value of the label, use string formatting to add commas like so:

```
text = "{:,}".format(1000)
```

Set the ranges as follows for each of the sliders:

Size – 1,000 to 100,000,000 in increments of 1,000

Immunity – 0 to 99

Infections – 1 to 1,000

Contact Rate – 0 to 20

Infection Rate – 1 to 100

Disease Period – 1 to 100

Mortality Rate – 1 to 100

Be sure to store the percentages as their decimal amount, but be extra sure to multiply the value by 100 in the label to ensure it is displaying correctly as a percentage. Add a start and stop button, the start button enabled and stop button disabled by default, and create a function for each one, binding it to each button respectively. In the start button function, disable all sliders and enable the stop button. In the stop button function, enable all sliders and disable the stop button. Ensure this is working correctly before moving on.

Calculating R0

Now that the UI is in place to allow the user to simulate a virus to their heart's content, we need to calculate R0. R0 is a surprisingly simple formula in the context of a simulation – Contact Rate * Infection Rate * Disease Period. In the real world, those numbers are tough to come up with but something that is important to note is that the only variable that we, as humans, can have any effect on is the Contact Rate. Infection Rate and Disease Period are inherent to the disease itself, even if we don't know those exact values. This is why social distancing is so important, it's the only thing we can do to decrease R0 and slow the spread of a virus. (Ramirez)

Add a label above the buttons that shows the calculation of R0 and a function that calculates R0 and sets it to the label. In each of the functions bound by the sliders for Contact Rate, Infection Rate, and Disease Period, add a call to the function that calculates and displays R0.

Building the model

With the UI in place we can now shift our focus to the model. After creating the model programmatically, it will be fairly easy to display a graph and visual representation of the SIR+D model. Since we added an initial immunity, we will be using a variable called immune to store the total of recovered and initially immune people. This better models a population as the immune people are still an important part of the statistic for the overall population. Create 4 variables on the line under the creation of the variables from earlier like so:

```
susceptible, infected, immune, dead = 0, 0, 0, 0
```

In the start button function, set the following variables:

Infected – Amount of people set to be initially infected (stored in infections)

Immune - Amount of people in the population who are immune (population size * immunity)

Susceptible - Size of the population less the amount of people in the population who are immune and those infected.

Dead – 0

We also need a variable for units of time for running the simulation. Create a variable t and set it to 0. Something important to note is that this program is structured around variables at the

global level. Ensure that every function accessing or changing a global variable has a line at the top specifying the variables are to be accessed in the global scope. For example:

```
global infected
infected = infections
```

This will become increasingly important as we continue on to the next part.

Logic of a Spreading Infection

In order to simulate a math model that changes at every time step, we will always need some sort of loop. In this program, there will be a global variable to keep track of if the program is running or not. The start button will set this value to True and start the simulation whereas the stop button will set this value to False, terminating the loop. Inside the loop in the start button function, it will call a function called updatePopulations() that will perform the logic for one time step, then increment time.

```
running = False
def startSimulation():
. . .
global running, t
t = 0
running = True
While running:
    global t
    rate(50)
```

```
updatePopulations()

t += 1
```

After implementing the loop in the start button function, create the updatePopulations() function above the UI creation. Add a line or two specifying global scope for the variables for the populations as well as the variables set by the sliders. The logic for updating each individual population is thankfully fairly simple. The steps are as follows:

(Note: You will need a new variable for each of steps 1-7)

1. Calculate proportion of population susceptible `(susceptible/popSize)`
2. Calculate removal rate of individuals infected `(1 / period)`
3. Calculate contact between susceptible people and infected people `(infected * proportionSusceptible * contact)`
4. Calculate number of people infected from contact `(infectedContact * infectRate)`
5. Calculate number of people either recovered or dead `(infected * removalRate)`
6. Calculate number of people recovered from previous number `(removedCases * (1 – mortality))`
7. Calculate number of people dead from number calculated in 5 `(removedCases * mortality)`
8. Update populations based on the above numbers. I'll let you figure this one out, it shouldn't be too much of a challenge.

   (SIR algorithm derived from formulas in Source (Smith))

Once this loop has been implemented the bulk of the program is pretty much done. I created labels temporarily to display the numbers for each loop to ensure it was working correctly, but that's entirely optional. They will be omitted from the program anyway so if you are confident in your implementation then it's time to move on to the displaying of data. If you're not confident, I suggest adding a label that totals up all of the populations together and displays it. This is a great way to make sure your proportions are working correctly since the total of the populations should stay constant. Because the populations aren't representative of real life due to containing decimals (.23781 of a person?) we need to make a modification to the loop that drives the simulation. Find the loop in the start function and modify it like so:

```
while running and infected > .05:
. . .
```

This will ensure that our program has an end point when the number of infected is reasonably close to 0.

Displaying Results

Now that the program logic is in place we can get around to displaying our results. Start by adding a few lines at the top of your program to make sure the scene displays correctly. I'll leave all the coloring up to you, the below code just ensures the scene is purely for display and not interaction and setting the FOV low ensures that the spheres don't look stretched if they're not center.

```
scene.fov = .01
scene.userzoom = False
```

```
scene.autoscale = False
scene.userspin = False
```

Next we need 4 spheres with 4 labels – susceptible, infected, immune, and dead. Design these however you like, just make sure the spheres don't overlap or go out of the screen width too far with large inputs. Check the source below for an idea of how to set up the spheres (Collinridge).

In the updatePopulations() functions, set the radius of each sphere like so:

```
if (susceptible < 1):
    susceptibleSphere.radius = .1
    susceptibleSphereLabel.text = "Susceptible: 0"
else:
    susceptibleSphere.radius = log(susceptible) / n
    susceptibleSphereLabel.text = "Susceptible:
    {:,}".format(susceptible)
```

A few notes about the code above:

- Checking whether the population total is below is important because log(n) where n < 1 returns a negative number which will throw off the radius. The calculations will continue normally but displaying it in this way makes more sense.
- On line 5 where I divided the log(susceptible) by n, that's purely done for display purposes. Depending on how you set up your scene, this number may be different so long as it's the same for each population.
- Recall "{:,}.format(…) from earlier as a way to add commas to large numbers.

Add code for each population to be updated and ensure everything is updating correctly. Add a label for what day it is in the simulation at the top corner and add code in the updatePopulations() function to update it. At this point, the first data visualization is complete. Now on to the graphs.

Fortunately, GlowScript makes it quite trivial to make graphs. Add a graph below where you added spheres to the scene and set it to have a reasonable width, height, and labels indicating what the data means. Add a set of gdots for each population, coloring them to match your spheres, and set their interval to 1 and size to 5. In the updatePopulations() function, add a couple lines to plot each point on the graph in the form (t, population). Lastly, add a couple lines to the top of the start function to delete each set of dots from the graph so each simulation produces a new graph.

Wrapping Up

At this point, the program is complete and all that's left to do is test it. Don't worry if the spheres overlap or go out of the screen bounds with large data so long as it's still easy to interpret what's happening. Go ahead and fiddle with the sliders to produce an R0 near some of the following values for common viruses and test the results. Use a population size of 100,000,000, slightly less than 1/3 of the US population. For some of these you will need to manually set the mortality rate in your start function as the slider isn't precise enough (cold, flu, covid).

Disease	R0	Mortality Rate	Dead	Time to Peak Infections
Measles	12-18	~2% (Unvaccinated)	2m	46 days

HIV/AIDS	2-5	~85% (untreated)	83.1m	211 days
SARS	2-5	~11%	10.75m	387 days
Common Cold	2-3	~0%	0	80 days
Influenza	1.4-2.8	~.01%	8.8k	80 days
Ebola	1.5-2.5	~85% (untreated)	66.5m	114 days
COVID-19	1.4-3.9	.6%-1.4%	57.5k-134k	56 days

All R0 and Mortality Rates courtesy of Wikipedia (Sources 1 and 4)

Question: Is the above information accurate to real world experience?

Because we are working with imperfect approximations in a simulation that doesn't consider the complex variation that occurs in the real world, the information isn't entirely accurate. That being said, it is a good representation of how a virus spreads, what sort of damage it can do if it's not contained or treated, and what can be done to slow the spread of a virus. Since the viral infection of COVID-19 is currently ongoing, the range for R0 is fairly wide (1.4-3.9) so it's a great example. If we run a simulation (disregarding mortality rate) with an R0 of 3.6, near the higher end, the peak of infections happens on Day 48 with nearly 40 million infected at the same time. If we decrease that R0 to 1.7, on the lower end, the peak of infections happens on day 150 with only 1 million infected at the same time.

Like mentioned earlier, the only variable in R0 that we can have any effect on is the Contact Rate. By decreasing the Contact Rate, we can decrease R0 to a point where we won't overload health services and everyone who needs medical care can receive it. The mortality rate isn't the most alarming part of COVID-19. Rather, it's the rate of people who need to visit the ICU

for their symptoms. If people are unable to receive care for their symptoms, that mortality rate skyrockets and a worrisome pandemic becomes a global catastrophe. This is why social distancing is so important right now and why people need to be individually responsible when it comes to washing their hands and limiting to essential services. (McMurry)

Question: Why wasn't Ebola a global emergency in the same way that COVID-19 is?

There were a few factors that played into Ebola dying out relatively quickly compared to what it could have been. One of these reasons, and an unfortunate one at that, is that it has such a high mortality rate once symptoms kick in. Simply put, at an 85% mortality rate, people who contract Ebola are more likely to die than they are to go out and spread it. Another reason is that because it's so serious, in the recent Ebola outbreak the efforts to quarantine and nip the spread in the bud were incredibly focused and we were able to prevent a pandemic. How that varies from COVID-19 is that people with symptoms can continue to go about their day and spread it to others. That combined with how easy it is to spread and its long incubation period is a recipe for disaster. (Del Rio)

Question: Do some research on R0 and explain your understanding of what the term means.

R0 is the number that tells us about the potential spread or decline of a disease. There are three particular categories for R0:

- If R0 is less than 1 then each infection will cause less than one new infection. In this scenario, the disease will decline and die out.
- If R0 is 1 then each infection will cause one new infection. The virus will continue to spread, but it won't cause an outbreak or epidemic.

- If R0 is greater than 1, each infection will cause more than 1 new infection. The disease spreads quickly and can cause an outbreak or epidemic.

Something to note about R0 is that it only applies directly if everyone is susceptible, meaning no one has been vaccinated, no one has had the disease before, and there's no way to control the spread of the disease. For example, though Ebola has an R0 of 1.5-2.5, it doesn't directly apply since considerable efforts went into controlling the spread. For COVID-19 it's also highly variable because of the efforts that have gone into quarantining and social distancing. R0 is not the be all end all of understanding a disease, but it's absolutely a helpful metric for getting an idea of how deadly it can be. (Ramirez)

Justin Edwards
PHYS 2300
Scientific Computing

Works Cited

"Basic Reproduction Number." *Wikipedia*, Wikimedia Foundation, 30 Mar. 2020, en.wikipedia.org/wiki/Basic_reproduction_number.

Collinridge, Peter. "Modelling an Epidemic | Ebola Outbreak." *Khan Academy*, Khan Academy, 2015, www.khanacademy.org/science/health-and-medicine/current-issues-in-health-and-medicine/ebola-outbreak/pi/modelling-an-epidemic.

Del Rio, Carlos, and Jeannette Guarner. "Ebola: Implications and Perspectives." *Transactions of the American Clinical and Climatological Association*, American Clinical and Climatological Association, 2015, www.ncbi.nlm.nih.gov/pmc/articles/PMC4530678/.

"List of Human Disease Case Fatality Rates." *Wikipedia*, Wikimedia Foundation, 3 Apr. 2020, en.wikipedia.org/wiki/List_of_human_disease_case_fatality_rates.

McMurry, Julie. "Flatten the Curve." Flatten the Curve, 2020, www.flattenthecurve.com/.

Ramirez, Vanessa. "What Is R0?: Gauging Contagious Infections." *Healthline*, 24 June 2016, www.healthline.com/health/r-nought-reproduction-number.

Smith, David, and Lang Moore. "The SIR Model for Spread of Disease - The Differential Equation Model." *The SIR Model for Spread of Disease - The Differential Equation Model | Mathematical Association of America*, Dec. 2004, www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model.