HIV Mini-Project

Learning Objectives

The main goals of this worksheet are:

- Gain experience reading and understanding scientific literature that employs modeling and simulation methods.
- Practice implementing stock-and-flow models in Python, starting from a new Jupyter notebook.
- Replicate a published model and validate the results.

Submission Checklist

Please be sure to do the following:

☐ Write your name here: Lilo

☐ Write the name(s) of your studio partner(s) here: Jasper

☐ By midnight on Friday, October 18: Scan this worksheet and submit it on Canvas. Also submit a link to your CoCalc notebook as a comment.

Then stay tuned for instructions from the NINJAs on how to get checked off. (You will need to meet with your studio NINJA and explain your work.)

The HIV Paper

Please answer the following questions based on your reading of Phillips (1996) and the discussion in class.

1. Here are the four equations of the model:

$$\frac{dR}{dt} = \Gamma\tau - \mu R - \beta RV \quad (1)$$

$$\frac{dL}{dt} = p\beta RV - \mu L - \alpha L \quad (2)$$

$$\frac{dE}{dt} = (1-p)\beta RV + \alpha L - \delta E \quad (3)$$

$$\frac{dV}{dt} = \pi E - \sigma V \quad (4)$$

(a) What are the 4 state variables? What are the 9 parameters?[1]

[1] Note that there seems to be a typo in Table 1; the parameter labeled ρ is the same as the one labeled p in the text.

state var

- R - activated uninfected
- L - latently infected
- E - actively infected
- V - free virons

parameters

- Γ - rate of production of R
- γ - proportion of cells activated R
- μ - rate of death of R
- β - rate of infection of R per V
- p - proportion of cells becoming L
- α - rate of activation of L (→ E)
- δ - rate of removal of R
- π - rate of production of V
- σ - rate of removal of cell-free virus V

(b) What are the units of each equation? For each equation, verify that the units are consistent (i.e., the same for each term).

1) $\frac{dR}{dt} = \Gamma\left[\frac{R}{t}\right] \cdot p\,[1] - \mu\left[\frac{1}{t}\right]\cdot[R] - \beta\left[\frac{1}{Vt}\right]\cdot[R]\cdot[V] = \frac{\text{cells}}{\text{day}}$

2) $\frac{dL}{dt} = \left(p[1]\cdot\beta\left[\frac{1}{Vt}\right]\cdot[R]\cdot[V]\right) - \left(\mu\left[\frac{1}{t}\right]\cdot[L]\right) - \left(\alpha\left[\frac{1}{t}\right]\cdot[L]\right)$

$= \text{cells / days}$

3) $\frac{dE}{dt} = \left((1-p[1])\cdot\beta\left[\frac{1}{Vt}\right]\cdot[R]\cdot[V]\right) + \left(\alpha\left[\frac{1}{t}\right]\cdot[L]\right) - \left(\delta\left[\frac{1}{t}\right]\cdot[E]\right)$

$= \text{cells / day}$

4) $\frac{dV}{dt} = \left(\pi\left[\frac{1}{t}\right]\cdot[E]\right) - \left(\sigma\left[\frac{1}{t}\right]\cdot[V]\right) = \frac{\text{cells}}{\text{day}}$

(c) The parameter values are given in Table 1 of the paper. What are the initial values of the four state variables (stocks)?[2]

(Note that Table 2 summarizes the results of the model for different values of σ, δ, π, Γ, and β. You are welcome to explore these on your own, but for this worksheet we'll stick with the values in Table 1.)

[2] See Figure 1 and the text on p. 497. Two of them are a little tricky!

R – (1000 lymphocytes total)
200 activated

L – 0 cells

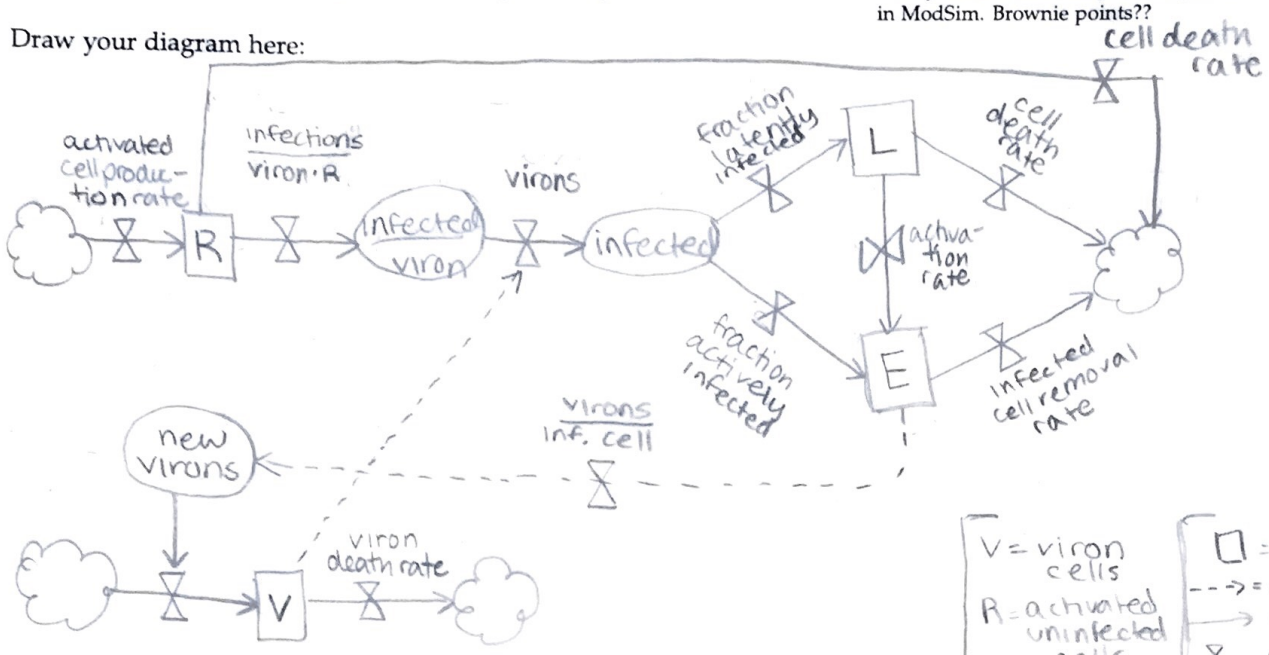
E – 0 cells

V – ~~100~~ $4 \cdot 10^{-7}$ virons

2. Use the equations above to construct a stock-and-flow diagram for the model in the paper. Be sure to label it appropriately; bonus points for indicating the flows in a sufficiently precise way that someone could reconstruct the four equations from your diagram.[3]

[3] Okay, there aren't really bonus points in ModSim. Brownie points??

Draw your diagram here:



3. There are just a few more pieces of information you need in order to implement the HIV model in Python — specifically, the start and end time, and the step length. What are the values of each of these parameters?[4]

[4] See Figure 1 again. Sometimes we call these "simulation parameters" to distinguish them from "model parameters" that appear in the flow equations.

simulation parameters

start time – 0 days infected

end time – 120 days infected

step length – 1 hour 20 min

Implementing the HIV Model

You are now ready to implement the HIV model.

You and your partner(s) should decide how you are going to work together on this task. If pair programming has worked well for you, and/or you would like to try it again, we suggest choosing driver and navigator roles for Tuesday's studio session and reversing them on Wednesday. You could also try a "modified pair programming" approach where each of you works in your own notebook but communicates continually with your partner(s). Regardless of how you organize yourselves, we encourage all members of the group to be actively engaged and learning from each other.

You should start from a new notebook that you create in CoCalc. The goal is to replicate the results that are presented in Figure 1 of the paper. You are welcome (but not obliged) to follow the QMRI format we are using for projects in the course.[5] Keep in mind that you and your partner(s) will need to explain your work to your studio NINJA during your check-off meeting.

[5] One advantage of doing so is that your HIV notebook could make an excellent starting point for Project 2 next week.

You are welcome to implement the model in any way you like, and very welcome to "cut and paste" code you've already worked with in the ModSimPy chapter notebooks. Here are some starting points that might be helpful:

1. You'll probably want to create a `System` object. You could use a `make_system` function like the one in Chapter 11 (p. 89).
2. You'll definitely want an update function of some kind. The one in Chapter 11 (p. 90) is a fine starting point, except that it implicitly assumes a time step of 1. Peek ahead to Chapter 15 (p. 128) for an example that allows a variable time step, `dt`, to be passed as part of a `System` object. If you peek even further ahead to Chapter 18 (p. 150), you'll see an example of an update function for two different stocks; this might be an even better place to start.
3. If you're following the pattern in the book, you'll also need a `run_simulation` function. Again, the ones you've already seen in Chapter 11 are not a bad starting point (especially the one on p. 94 that uses a `TimeFrame`), but here too you'll need to handle a variable time step. For this, see the `run_simulation` functions in Chapter 15 (p. 129) and Chapter 18 (p. 151); they are identical.[6]

make_system # chp 11

update # chp 11 p90
or chp 15 p128 for dt

read chp 18 p150
for stock updt func

run_simulation # chp 11 p94
TimeFrame

As always, be sure to test your code as you go (especially your update function), and seek help if you need it.

[6] We are actually going to say goodbye to `run_simulation` altogether at the end of Chapter 18, when we learn how to use the ODE solver. If you're interested, you're welcome to peek ahead at pp. 152–153 and implement a slope function instead of an update function. (No worries if you don't — we'll come back to this later.)

Plotting and Comparing the Results

Congratulations — if you've gotten this far, you've successfully implemented the HIV model and are ready to compare your results with those of the paper!

We're going to leave this part mostly to you. The thing we really want you to do is to verify that your *numerical* results are consistent with the results presented in Figure 1. But if you want to get fancy and try to replicate the graphs themselves, you are most welcome to do that.

$\frac{24}{1.2} \cdot 120 = 20 \cdot 120 = 2400$ steps of 1.2 hours

loglog(x, y)

semilog(x, y, x2, y2 . . .)

(And if you do that, you'll probably need to refer to the Matplotlib documentation, especially the parts about creating axes with logarithmic scales[7] and combining two different scales into the same graph.[8])

[7] https://matplotlib.org/3.1.1/gallery/pyplots/pyplot_scales.html

[8] https://matplotlib.org/gallery/api/two_scales.html

Most of this work will be in your notebook on CoCalc, a link to which you will turn in with this assignment, so we don't need you to reproduce it here. That said, feel free to use this space to capture notes or thoughts you want to share about your simulation results.

I am happy with our version of the model. It essentially exactly replicates the results of the paper, including the graph formatting.

Finally, try playing around with the time step (`dt`). How short does it need to be in order to produce the smooth-looking curves you see in the paper? What happens if you set it to 1 day? Why?

dt: smooth:

1 day: when dt=1 day, all virons die immediately caused by the virons only having a 1/2 day lifespan and the fact that it takes time to produce infected cells, since it starts with none, and infected cells cause the production of virons.

$V(t=1) = \gamma E - \sigma V < 0$ when $E=0, V>0, \sigma \geq 1$

Reflection Questions

1. Did you succeed in replicating the model and results of the HIV paper? What differences were there (if any), and what do you think accounts for them?

Yes, we were sucessful.

Actually, I compared the graphs and they were essentially identical.

To graph total CD4 lymphocytes, however, I had to add the 800 initial unactivated lymphocytes to R, E, and L to get the same overall result.

2. What do you think of the model itself? Thinking back to our modeling mantras, in what ways is it (perhaps intentionally) wrong, and in what ways is it useful? To what extent do you think it achieves the goal of being as simple as possible but no simpler?

① In assuming no specific immune response, it simplifies the problem and allows us to see the underlying trend, which is that the infection decreases over time after the initial spike due to depletion of uninfected cells. This is good.

② Two of the variables, Γ & γ could have been combined, since γ is the rate of activation of lymphocytes, and unactivated lymphocytes are essentially irrelevant because they don't factor into this model.

3. What was the most difficult part of this assignment? Where did you get stuck, and how did you get unstuck?

I initially wrote dt as 1, causing very spiky results, and had to ① return to the graph in the paper to find a better value of dt and ② refer to Chapter 15 to find a good implementation of dt ③ modified the model by adding it as a state var, used it in the update_func and changed the simulation to run days · dt num. times.