Previous   Next

Part B: Problem 2

Bookmark this page

Calculator   Hide Notes

Part B: Problem 2-1: Running and Analyzing a Simple Simulation (No Drug Treatment)

10.0/10.0 points (graded)
You should start by understanding the population dynamics before introducing any drug.

Fill in the function `simulationWithoutDrug(numViruses, maxPop, maxBirthProb, clearProb, numTrials)` that instantiates a `Patient`, simulates changes to the virus population for 300 time steps (i.e., 300 calls to `update`), and plots the average size of the virus population as a function of time; that is, the x-axis should correspond to the number of elapsed time steps, and the y-axis should correspond to the average size of the virus population in the patient. The population at time=0 is the population after the first call to `update`.

Run the simulation for `numTrials` trials, where `numTrials` in this case can be up to 100 trials. Use pylab to produce a plot (with a single curve) that displays the average result of running the simulation for many trials. Make sure you run enough trials so that the resulting plot does not change much in terms of shape and time steps taken for the average size of the virus population to become stable. **Don't forget to include axes labels, a legend for the curve, and a title on your plot.**

You should call `simulationWithoutDrug` with the following parameters:

- `numViruses` = 100

- `maxPop` (maximum sustainable virus population) = 1000

- `maxBirthProb` (maximum reproduction probability for a virus particle) = 0.1

- `clearProb` (maximum clearance probability for a virus particle) = 0.05

Thus, your simulation should be instantiating one `Patient` with a list of 100 `SimpleVirus` instances. Each `SimpleVirus` instance in the viruses list should be initialized with the proper values for `maxBirthProb` and `clearProb`.

Hint: graphing

Your graph should contain one line that represents the average of many different trials. One way of computing the average involves holding all of your data in one list, with one element for each of 300 time steps, and adding to each data point during each trial. Then, at the end, each element of the list is divided by the total number of trials, thus taking the average of each element of the list. However, if this idea confuses you, feel free to ignore the hint and implement it however makes the most sense to you.

Consult <u>reference documentation on pylab</u> as reference. Scroll down on the page to find a list of all the <u>plotting commands</u> in pylab.

Hint: testing your simulation

Compared to the previous problem sets, testing your simulation code is more challenging, because the behavior of the code is stochastic, and the expected output is not exactly known. How do you know whether your plots are correct or not? One way to test this is to run the simulation with extreme input values (i.e. initialization parameters), and check that the output matches your intuition. For example, if `maxBirthProb` is set to 0.99 instead of 0.1, then you would expect that the virus population rapidly increases over a short period of time. Similarly, if you run your simulation with `clearProb` = 0.99 and `maxBirthProb` = 0.1, then you should see the virus population quickly decreasing within a small number of steps. You can also try to vary the input values, and check whether the output plots change as you expect. For example, if you run multiple simuation runs, each time increasing `maxBirthProb`, the curves in the successive plots should show an "upward" trend, since the virus will reproduce faster with a higher `maxBirthProb`.

For further testing, we have provided the .pyc (compiled Python) files for the completed `Patient` and `SimpleVirus` classes (and for Problem 5, the `ResistantVirus` and `TreatedPatient` classes) that you can use to confirm that your code is generating the correct results during simulation.

If you comment out your versions of these classes in `ps3b.py`, and add the following ⊞ Calculator ; to the

top of your file ~~u~~ to make sure
you are obtain~~i~~ s. s correctly.
Make sure to comment out the import statement and uncomment your implementations before moving to Problem 4.

Notes:

to be continued

- **Note: If you want to use numpy arrays, you should add the following lines at the beginning of your code for the grader:**
  `import os`
  `os.environ["OPENBLAS_NUM_THREADS"] = "1"`
  Then, do `import numpy as np` and use `np.METHOD_NAME` in your code.

- Use this code in the grader, to plot:

```
pylab.plot(YOUR_Y_AXIS_VALUES, label = "SimpleVirus")
pylab.title("SimpleVirus simulation")
pylab.xlabel("Time Steps")
pylab.ylabel("Average Virus Population")
pylab.legend(loc = "best")
pylab.show()
```

For Python 3.5: `from ps3b_precompiled_35 import *`

For Python 3.6: `from ps3b_precompiled_36 import *`

For Python 3.7: `from ps3b_precompiled_37 import *`

```
1 # Part B: Problem 2
2 #  Bookmark this page
3 # Part B: Problem 2-1: Running and Analyzing a Simple Simulation (No Drug Treatment)
4 # 10.0/10.0 points (graded)
5 # You should start by understanding the population dynamics before introducing any drug.
6
7 # Fill in the function simulationWithoutDrug(numViruses, maxPop, maxBirthProb, clearProb, numTrials) that
8 # simulates changes to the virus population for 300 time steps (i.e., 300 calls to update), and plots the
9 # population as a function of time; that is, the x-axis should correspond to the number of elapsed time s
10 # correspond to the average size of the virus population in the patient. The population at time=0 is the
11 # to update.
12
13 # Run the simulation for numTrials trials, where numTrials in this case can be up to 100 trials. Use pyla
14 # single curve) that displays the average result of running the simulation for many trials. Make sure you
15
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

See full output

CORRECT

See full output

Submit   You have used 1 of 30 attempts

Part B: Problem 2-2

edX®

edX

About   Calculator

Affiliates

edX for Business

Open edX

Careers

News

Legal

Terms of Service & Honor Code

Privacy Policy

Accessibility Policy

Trademark Policy

Sitemap

Connect

Blog

Contact Us

Help Center

Security

Media Kit

Calculator