Problem Set 4: Simulating the Spread of Disease and Bacteria Population

Out: 11:59pm, Tuesday, November 15th, 2016.
Due: 11:59pm, Wednesday, November 23rd, 2016.

Introduction

In this problem set, you will design and implement a stochastic simulation of bacteria population dynamics, and reach conclusions about how various treatment regimens affect the spread of the bacteria based on the simulation results.

You should submit:

1. Code for your solutions in **`ps4.py`**
2. Write-up for the discussions in **`ps4_writeup.pdf`**

Background: Bacteria, Antibiotics, and Computational Models

Bacteria are single-celled organisms that reproduce asexually. Some bacteria cause diseases, some are harmless, and some are beneficial. Bad bacteria can cause infections such as strep throat and tuberculosis. Bacterial infections are treated with antibiotics targeted to kill only the bad bacterial cells.

Bacteria that cause infections can resist and develop a resilience to antibiotics in two ways: (1) naturally and (2) via inappropriate use of antibiotics. Thus, populations of bacteria can undergo substantial evolutionary changes within a single patient over the course of treatment.

In this problem set, we will make use of simulations to explore the effect of introducing antibiotics to the bacteria population and determine how best to address these treatment challenges within a simplified model. We will implement a highly simplified stochastic model of bacteria population dynamics within a person. Nevertheless, our model exhibits biologically relevant characteristics and will give you a chance to analyze and interpret interesting simulation data.

We've provided you with skeleton code in **`ps4.py`**. Please do not change any of the provided skeleton code except to delete # TODO after writing your implementation.

Problem 1: Implementing a Simple Simulation (No Antibiotic Treatment)

We start with a trivial model of the bacteria population - the patient does not take any antibiotics and the bacteria do not acquire resistance to antibiotics. We simply model the growth of the bacterial population inside a patient as if it were left untreated:

Implement the `SimpleBacteria` and `Patient` classes, according to the behavior described in the docstrings in the provided code.

`SimpleBacteria` class

The `SimpleBacteria` class maintains the state of a single bacteria. You will implement the following methods according to the specifications:

- `__init__`
- `is_killed`
- `reproduce`

Use `random.random()` for generating random numbers between 0 and 1 to ensure that your results are consistent with ours. **Hint**: During debugging, use `random.seed(0)` so that your results are reproducible. When you put this at the top of your file, all "random" calls will be the same each time you run your file, and so your code will behave the same way each time you run it, which can be useful for debugging purposes. Make sure to remove this before you hand in your pset!

`Patient` class

The `Patient` class maintains the state of a bacterial population associated with a patient. You will implement the following methods according to the docstring spec:

- `__init__`
- `get_total_pop`
- `update`

Problem 2: Running and Analyzing a Simple Simulation (No Antibiotic Treatment)

In this part you will understand the behavior of a group of bacteria cells as time passes before introducing any antibiotic through a simulation.

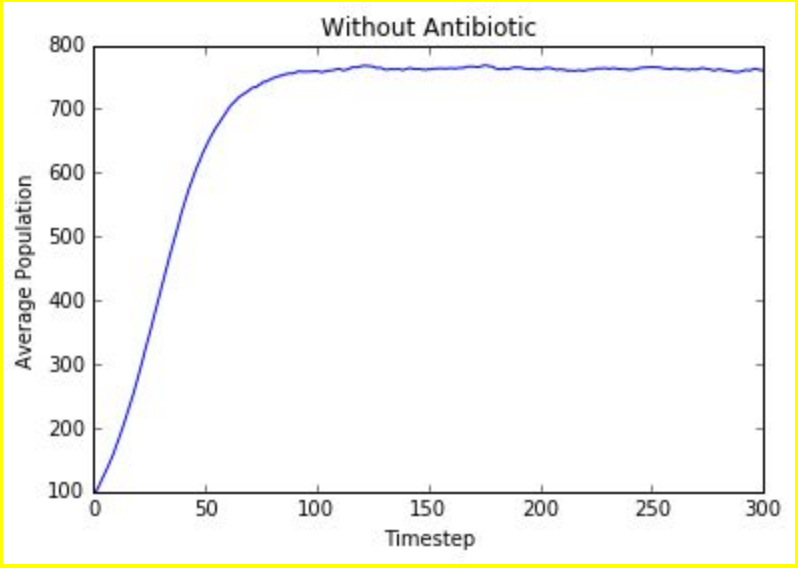
Implement the functions `calc_pop_avg` and `simulation_without_antibiotic` according to the behavior described in the docstrings in the provided code. While you must implement `calc_pop_avg`, you are not required to use it while implementing `simulation_without_antibiotic` (though you may find it useful). You can test `calc_pop_avg` using the test suite that we've provided.

At the end of your `simulation_without_antibiotic` , use the provided helper function `make_one_curve_plot` to generate a plot of your results. The plot should display the average bacteria population over time. You plot the average population across `num_trials`, so that the resulting curve is smooth and shows a general trend rather than the particular details of any one trial. **It may take up to a minute to generate the plot.**

Hint: `simulation_without_antibiotic` asks you to return a 2D list of populations at each time step for each trial such that `populations[i][j]` would be the population for the trial i at time step j. This means that if we are running 2 trials for 5 time steps, `populations` may look something like:

[[10, 34, 22, 40, 21],
[15, 27, 32, 25, 41]]

Paste in a copy of the graph you created by running your simulation with our parameters (in the call to `simulation_without_antibiotic` that we provided in `ps4.py`) in your write-up (`ps4_writeup.pdf`).
You should aim to have your graphs look like the one below:



Problem 3: Calculating a Confidence Interval

Using data collected in the simulation from Problem 2 as our sample, we want to construct a 95% confidence interval of an estimate of the average bacteria population at a certain time step (zero-indexed).

Implement the functions `calc_pop_std` and `calc_95_ci` according to the behavior described in the docstrings in the provided code. You can test your code using the tests we provided in ps4_tests.py

Given data $x_i \; for \; i = 1, 2, \ldots, n$, some formulas you might find useful are:

mean $$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

standard deviation $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

standard error of the mean $SEM = \frac{\sigma}{\sqrt{n}}$

95% confidence interval $\bar{x} \pm 1.96(SEM)$

Using the sample data collected from running the call to `simulation_without_antibiotic` provided in `ps4.py`, compute and record the 95% confidence interval for the population estimate at time step 299 in your write-up (`ps4_writeup.pdf`).

Problem 4: Implementing a Simulation with an Antibiotic

In this problem, we consider the effects of both administering an antibiotic to the patient and the ability of bacteria cell to inherit or mutate genetic traits that provide antibiotic resistance. As the bacteria population reproduces, mutations will occur in the bacteria offspring. Some bacteria cells gain favorable mutations like antibiotic resistance. Note that bacteria in lower population densities mutate at a faster rate.

`ResistantBacteria` class

`ResistantBacteria` is a subclass of `SimpleBacteria`, which has the potential to gain antibiotic resistance through a mutation. The scientists on the 6.0002 staff have found that bacteria that develop antibiotic resistance also seem to gain other mutations that cause a different death rate. If a `ResistantBacteria` gains antibiotic resistance, it will die with the given death probability. `ResistantBacteria` that do not have antibiotic resistance will die with probability `death_prob / 4`. Whether or not the bacteria have this mutation is determined at birth.

Implement the `ResistantBacteria` class according to the docstrings and descriptions provided above. The methods you need to implement are:

- `__init__`
- `get_resistant`
- `is_killed`
- `reproduce`

`TreatedPatient` class

The `TreatedPatient` class is a subclass of `Patient` and represents a patient with antibiotic treatment and manages a collection of `ResistantBacteria` instances.

- `TreatedPatient` must make use of the new methods in `ResistantBacteria` and maintain whether the antibiotic is administered to the patient.

- The antibiotic is given to the patient using the `TreatedPatient` class's `set_on_antibiotic` method.
- If the patient is on the antibiotic then only bacteria cells that have a resistance will survive. Bacteria cells with resistance to the antibiotic continue to reproduce normally.

Implement the methods of `TreatedPatient` according to the docstring. The methods you need to implement are:

- `__init__`
- `set_on_antibiotic`
- `get_resist_pop`
- `update`

Problem 5: Running and Analyzing a Simulation with an Antibiotic

In this problem, we will use the implementation you filled in for Problem 4 to run a simulation. The simulation explores the effects of resistant bacteria and antibiotic treatments.

Implement the function `simulation_with_antibiotic` according to the behavior described in the docstrings in the provided code.

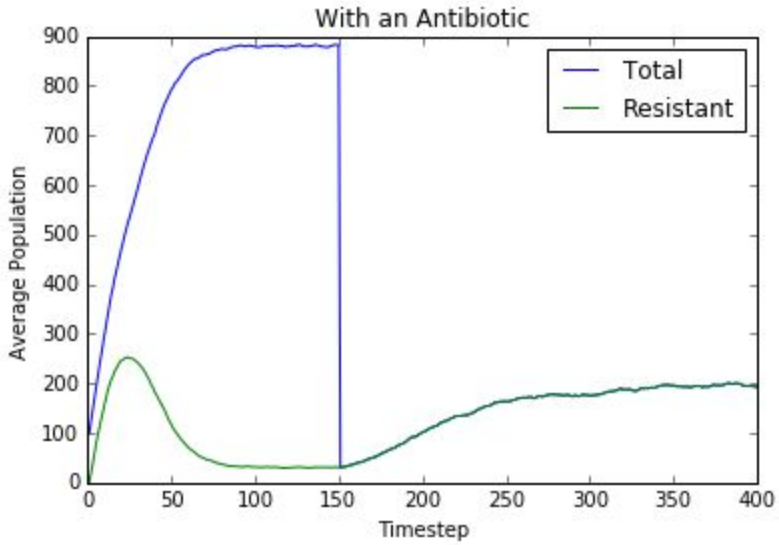
Use the provided helper function `make_two_curve_plot` to generate a plot of your results at the end of your `simulation_with_antibiotic` function. The plot should have the following two curves:

- The average total bacteria population size over time
- The average population size of antibiotic-resistant bacteria population over time.

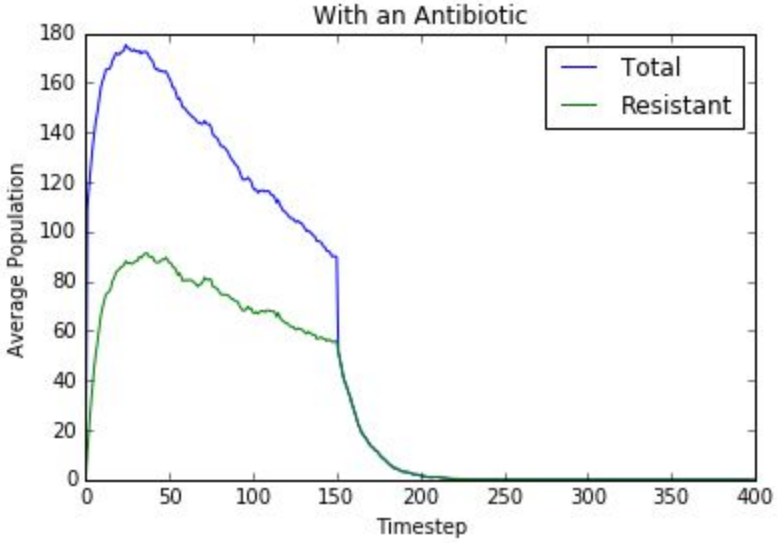
Run the two calls (simulation A and simulation B) to `simulation_with_antibiotic` that we provided in `ps4.py`. Paste copies of the two graphs you created by running your simulation with our parameters into your write-up (`ps4_writeup.pdf`). The first simulation (Simulation A) has a higher birth prob than the second (Simulation B).

Your graphs should look something like the ones below, where the two curves eventually converge (the actual dropoff might look less drastic in your implementation):

Simulation A:



Simulation B:



Use the sample data collection from running each call to `simulation_with_antibiotic` to compute the 95% confidence interval for the total population estimate and resistant bacteria estimate at time step 299 in your write-up (`ps4_writeup.pdf`). Note that there should be a total of 4 confidence intervals for this part.

Problem 6 Write-up

In your write-up (`ps4_writeup.pdf`), include the plot for the simulation from Problem 2 and the two plots for each simulation from Problem 5. For each plot from Problem 5, answer the following questions and explain why there is a difference (if it exists) between simulation A and B:

Trends of Simulation A and Simulation B

1. What happens to the total population before introducing the antibiotic?
2. What happens to the resistant bacteria population before introducing the antibiotic?
3. What happens to the total population after introducing the antibiotic?
4. What happens to the resistant bacteria population after introducing the antibiotic?

Hand-In Procedure

1. Save

Save your solutions as **`ps4.py`** and **`ps4_writeup.pdf`** .
For **`ps4_writeup.pdf`** , please do not submit a .doc, .odt, .docx, etc.

2. Time and Collaboration Info

At the start of each file, in a comment, write down the number of hours (roughly) you spent on the problems in that part, and the names of the people you collaborated with. For example:

```
# Problem Set 4
# Name:
# Collaborators (Discussion):
# Time:
#
 ... your code goes here ...
```

3. Sanity checks

- After you are done with the problem set, do sanity checks. Run the code and make sure it can be run without errors. You should never submit code that immediately generates an error when run!
- Make sure that your write-up contains everything we've asked for. You should include all 3 plots and answers to all the questions that we ask you.

6.0002 Introduction to Computational Thinking and Data Science

Fall 2016