Foundations of QBioS Homework 1

Elisa Rheaume

Homework Walk-through **2**
- ✍Problem 1 — September 9, 2022 2
- ✍Problem 2 — September 9, 2022 4
- ✍Problem 3 — September 9, 2022 6
- ✍Problem 4 — September 9, 2022 7

Codes **8**
- ✍Problem 1 Code — September 9, 2022 8
- ✍Problem 2 Code — September 9, 2022 10
- ✍Problem 3 Code — September 9, 2022 12
- ✍Problem 4 Code — September 9, 2022 13

Homework Walk-through

✍️Problem 1 — September 9, 2022

Description

Simulating the Luria-Delbruck Experiment, One Generation: Write a program to simulate just one generation of the LD experiment - stochastically. Simulate C=500 cultures each of which has N=1,000 cells and $\mu = 10^{-3}$, i.e., a very high mutation rate. What is the distribution of resistant mutants that you observe across all the cultures? Are they similar or dissimilar to each other? Specify your measurement of c(m), i.e., the number of cultures with m resistant mutants. Is this distribution well fit by a Poisson distribution? If so, what is the best fit shape parameter of the Poisson density function and how does that relate to the microscopic value of mutation you used to generate the output? Finally, to what extent are the fluctuations "large" or "small" ?

Simulating One Generation

Simulating one generation should be an easy task. By assuming an independent chance of mutation during each generation, and that generations are produced instantaneously at some set time (irrelevant time as we are only looking at one generation right now), we can simulate the expected number of mutants.

$$n_f = 2 * n_1 \tag{1}$$

Because there is a μ of 10^{-3}, then each generation there is a $1 - \mu$ chance that an unmutated cell will become 2 unmutated cells. There is a μ chance that it will instead produce 1 mutated cell and 1 unmutated cell. Because there is a guarenteed population in each culture (pursuant to Eq (1)) I need only record the mutated cells. To accomplis this, I will:

1. Create a 'mutant' array which will store the number of mutants created in each culture.
2. Iterate over 500 cultures with a For loop
3. For each culture, utilize **np.random.choice** to randomly assign binary values to my new population of 1000 (0 for no mutant, 1 for mutant). I then sum all elements of the output array, giving me the number of mutants for that culture in 1 generation. The new generation also has now 2000 cells.
4. I save this number of mutants in the 'mutant' array of length 500.

The results of this randomization can be viewed in the histogram Fig 1. I would characterize this distribution as heavily skewed around 1-3, with a few outlying data points 4 and 5. Most cultures are very similar with 0-3 mutants. Several lucky cultures have 4 and a few very lucky cultures have 5 mutants. In my simulation, I found that 180 of the cultures remained unchanged with 0 mutants. 199 cultures had 1 mutant. 82 cultures had 2 mutants. 32 cultures had 3 mutants. 6 cultures had 4 mutants. 1 culture had 5 mutants. These cultures now have 2000 cells total.

To compare whether or not this data fits a Poisson distribution, I will plot a random poisson distribution for 1 trial along side my data and compare them. The shape factor that would best suit my distribution should be

$$\lambda = np \tag{2}$$

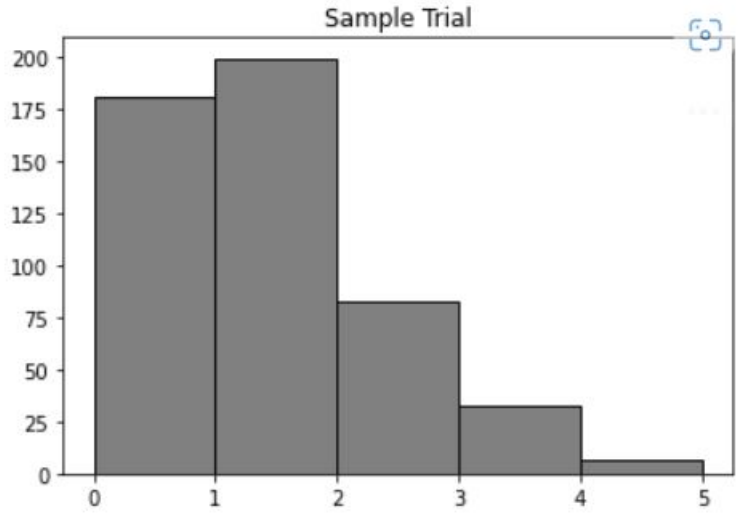

图 1. Histogram showing the number of mutant bacteria from one generation of 500 cultures with an initial population of 1000 bacteria and a mutation rate of $\mu = 10^{-3}$

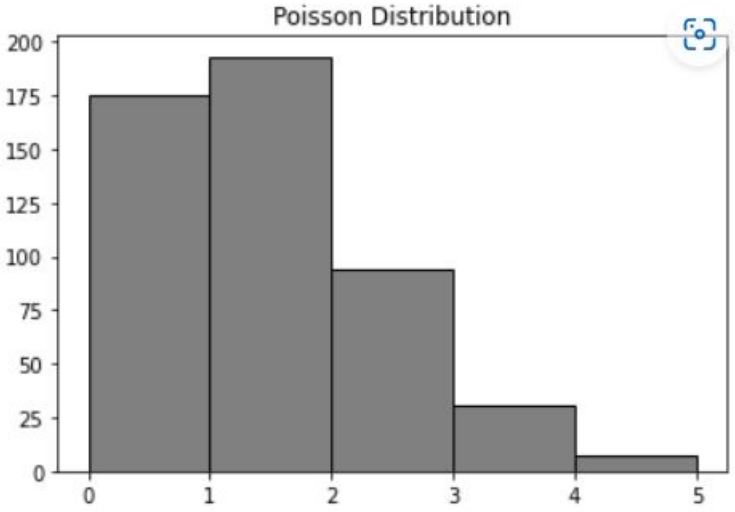

图 2. The Poisson Approximation of my data, using $\lambda = np$ as the optimal shape factor.

Where λ is the shape factor, n is the number of trials, and p is the probability of mutation. Thus, λ is the expected average number of mutations in each culture. $n = 1000$, as there are 1000 cells in each culture. I will set the sample size to 500, as there are 500 cultures.

To examine the fluctuations of the experiment, I calculate the variance of the sample that I generated as well as the poisson random variance. My sample had a variance of 0.9194 where the poisson sample had a variance of 0.9260. These fluctuations could be considered large as the average mutants generated for my sample and the poisson sample were 0.976 and 1.006 respectively.

✍️Problem 2 September 9, 2022

Description

Simulating the Luria-Delbruck Experiment Forward, One Generation at a Time: Extend the program in part 1 by setting C = 1000, N=400 and μ =10-7, while having the population grow over g = 15 generations. What choice did you make with respect to modeling the population? If you decided to model each individual cell in each individual culture, explain your rationale? Could you, instead, model the emergence of new resistance in each generation in each culture en masse (that is, all at once)? Hint: think about how prudent use of the Poisson random generation function could help. Bonus hint: compare the speed when you use Poisson vs. Binomial random generation functions. In doing so, report back your findings, specifically what is c(m) – the number of cultures with m resistant mutants. Describe and characterize the shape of this distribution and contrast it to that expected under the "acquired" hypothesis? What is the mean and variance, and are fluctuations "large" or "small" . Finally, to what extent are their "jackpot" cultures? Can you define a principled way to identify those "jackpots" and answer: to what extent do fluctuations become small when jackpots are excluded?

Simulating 15 Generations

To extend my current model to include 15 generations, there are several changes that need to be made. Additionally, I need to choose whether or not to have all of my populations reproduce instantaneously together or to have them randomly reproduce over the course of the simulation. For simplicity's sake, and because I want exactly 15 generations, I will have all of my cultures divide and multiply exactly 15 times, at the same time. Additionally, I will not alter the method of my modeling decision to record only the mutant populations. This will save lots and lots of data and we already know the total number of cells in each culture ($400 * 2^{15}$).

1. Set initial conditions (N=500, C=1000, $\mu = 10^{-7}$
2. Create mutant array of zeros
3. Iterate over 15 generations
4. Iterate over all samples, using np.random.choice to determine whether or not each non-mutated bacteria will mutate or not.
5. remove already mutated bacteria from the pool of possible future mutants, Those bacteria need to have a guaranteed mutant.

In Fig 3 I have created a histogram detailing the number of mutants in each culture. The distribution appears to be a Poisson distribution with a few exceptional outliers, also known as 'Jackpots'. The mean is calculated to be 41064 and the variance is calculated to be $1.32 * 10^9$. Under Lamark's 'Acquired' model, mean (Eq (3a)) and Variance ((3b)) are detailed. By those equations, the expected mean should be 1.31 and the variance should be 1.31 as well. However, these values do not match the expected results (by a lot). As a result, I must discard the acquired hypothesis.

$$< m >= \mu * N + / - N_0 \tag{3a}$$

$$Var = \mu * N \tag{3b}$$

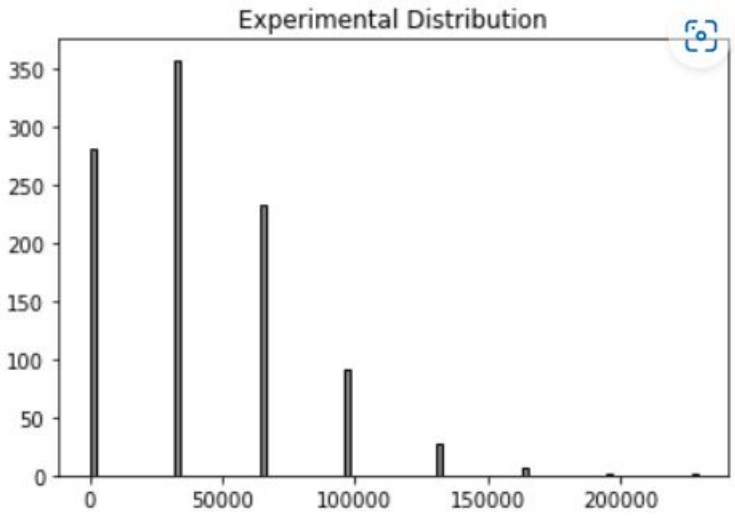

图 3. A histogram detailing the number of cultures with a specified number of mutants that arose in my experiment after 15 generations.

Fluctuations can be considered large because of the existence of jackpots. These are cultures in which a mutant is created early in the generation process, thereby creating many more mutants than the average. A simple way of defining jackpots would be cultures who's mutation rate is beyond a certain number of standard deviations from the mean. Those cultures would skew data and give unreliable results. Whether or not fluctuations become small depends on how many standard deviations greater than the average one considers a jackpot. If one were to select 4 or 5 standard deviations, it is likely that no sizable change would occur to the total fluctuations. However, if, for example, jackpots were half of a standard deviation or greater, there would be a much larger impact as we begin to discard data that extends too far from the average. In my example, if I exclude data that is 2 or more standard deviations from the mean, my variance is reduced to $6.30 * 10^8$, much lower than my previous variance.

✍🏻Problem 3 September 9, 2022

Description

Why Are There Jackpots? In this problem you should characterize the "age" of each mutant? That is, if you have not already done so, keep track of the first appearance of each mutant and characterize the relative importance of different "age" mutants to the total number at the end. Do early/late mutants contribute disproportionately to the total number? Do early/late mutants more strongly influence variation?

Counting Jackpots

To tackle this next section, I simply need to amend my code from section 2 to output an array that tracks the number of mutants and when they arise. The simplest way to do this is during my For loop, I can add any mutants that are discovered to a zero'd array, in a position determined by the current generation. Then, afterwards, I should be able to see when mutants were developed.

1. Use code for Problem 2 as a base.
2. Create an array of 15 zeros, one for each generation.
3. After np.random.choice outputs the number of NEW mutants, save this number to the zero array. The position will indicate when the mutant appeared.

Using the same code as in section 2, but adding in a counter for when mutants appeared, I can see that the first mutant appeared in the 4th generation, another fresh mutant appeared in the 5th and 6 generations. Then, due to the rapidly increasing population size, many more mutants began to emerge. Early mutants have a much more significant impact on the total number of mutants than late mutants. This is because every generation the total number of mutants doubles as well as additional mutants emerge. A single mutant in the first generation could produce 2^{15} mutants whereas a mutant in the last generation is just a singular mutant. Because of these early 'jackpot' mutants, the variation is HEAVILY skewed, resulting in a variation that is orders of magnitude larger than my mean.

✍️Problem 4 September 9, 2022

Description

Problem 4. Moving Backwards, like L&D, from Observations to Estimates Take the results from the first 100 of your experiments in Problem 2, that is the number of resistant mutants, m1, m2, ⋯, m100. Treat these 100 numbers as your "data" . Now, write a program that leverages results of new simulations to infer the most likely value of μ, the mutation rate, and (if possible) a 95% confidence interval for it – in doing so, assume you do not μ is in advance. Consider using one of two pieces of evidence: (i) the number of cultures with no resistant mutants; (ii) the mean number of resistant mutants. Finally, ask: is your data also consistent with the acquired resistance or spontaneous resistance hypothesis? Why or why not?

Approach

Given that we already have data stored for the trials, I can approximate by using the first 100 rows of data, from which I can calculate the average total number of mutants. Additionally, because we can analytically calculate the total number of bacteria using a simple formula, Eq (4). Then, using Equation 1.25 in the textbook (Eq (5)) we can calculate directly the expected mutation rate μ.

$$N_f = N_0 * 2^{\tau} \tag{4}$$

$$m = \mu \tau_f N_f \tag{5}$$

1. Select first 100 rows of data from previous trial (2)
2. Average the total number of mutants from those 100 rows
3. Plus into Eq (5) and solve.

For the purpose of being painfully explicit, let me put this here...

$$\mu = \frac{m}{\tau_f N_f} \tag{6}$$

Finally, after plugging in $\tau = 15, N_f = 400 * 2^{15}, and m = 42605$ I find that the expected μ should be approximately .00022 which is significantly larger than the actual value. Realistically, this value is not consistent with either the acquired resistance or spontaneous resistance model hypothesis because neither model returns an accurate value for μ. I would guess that this error is occurring largely because of jackpots which are corrupting the average and variance of the data.

Codes

✍🏻Problem 1 Code

September 9, 2022

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

### Problem 1. Simulating the Luria-Delbruck Experiment, One Generation
# Write a program to simulate just one generation of the LD experiment - stochastically. Simulate C=500
# cultures each of which has N=1,000 cells and   = 10-3, i.e., a very high mutation rate. What is the
# distribution of resistant mutants that you observe across all the cultures? Are they similar or
                                                    dissimilar
# to each other? Specify your measurement of c(m), i.e., the number of cultures with m resistant
# mutants. Is this distribution well fit by a Poisson distribution? If so, what is the best fit shape
                                                    parameter
# of the Poisson density function and how does that relate to the microscopic value of mutation you
                                                    used
# to generate the output? Finally, to what extent are the fluctuations “large” or “small” ?

mu = 10**-3

# Create mutant array
mutant_array = np.zeros(500)

for i in range(0, np.size(mutant_array)):
    x=np.random.choice([0,1],size=(1000,1),p=[1-mu,mu])
    mutant_array[i] = np.sum(x)

plt.hist(mutant_array,bins=5,color='gray',edgecolor='k');

mutant_array = np.array(mutant_array)
zero_mut = (np.where(mutant_array==0)[0]).size
one_mut = (np.where(mutant_array==1)[0]).size
two_mut = (np.where(mutant_array==2)[0]).size
three_mut = (np.where(mutant_array==3)[0]).size
four_mut = (np.where(mutant_array==4)[0]).size
five_mut = (np.where(mutant_array==5)[0]).size
print(zero_mut,one_mut,two_mut,three_mut,four_mut,five_mut)

# Now I will prepare a set of Poisson data using the optimal shape factor lambda = np

lam = mu*1000
numsamps = 500
poss_dist = np.random.poisson(lam,numsamps)
plt.hist(poss_dist,bins=5,color='gray',edgecolor='k');
#plt.xticks([0,1,2]);
plt.title('Poisson Distribution');
```

```
# Finally, to find the "fluctuations" in my data, I calculate the varience.

np.var(poss_dist)
np.var(mutant_array)
```

✍️Problem 2 Code

September 9, 2022

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

###Problem 2. Simulating the Luria-Delbruck Experiment Forward, One Generation at a Time
# Extend the program in part 1 by setting C = 1000, N=400 and   =10-7, while having the population grow
# over g = 15 generations. What choice did you make with respect to modeling the population? If you
# decided to model each individual cell in each individual culture, explain your rationale? Could you,
# instead, model the emergence of new resistance in each generation in each culture en masse (that is,
                                                    all
# at once)? Hint: think about how prudent use of the Poisson random generation function could help.
# Bonus hint: compare the speed when you use Poisson vs. Binomial random generation functions.
# In doing so, report back your findings, specifically what is c(m) – the number of cultures with m
                                                    resistant
# mutants. Describe and characterize the shape of this distribution and contrast it to that expected
                                                    under
# the “acquired” hypothesis? What is the mean and variance, and are fluctuations “large” or
                                                     “small” .
# Finally, to what extent are their “jackpot” cultures? Can you define a principled way to identify
                                                    those
#  “jackpots” and answer: to what extent do fluctuations become small when jackpots are excluded#

cult = 1000
N = 400
mu = 10**(-7)

# Create mutant array
mutant_array = np.zeros(cult)
mutant_array = mutant_array.astype(int)

for gens in range (0,16):
    for i in range(0, np.size(mutant_array)):
        x=np.random.choice([0,1],size=(N*2**(gens)-mutant_array[i],1),p=[1-mu,mu])
        mutant_array[i] = np.sum(x) + 2*mutant_array[i]

plt.hist(mutant_array,bins=20,color='gray',edgecolor='k');
#plt.xticks([0,1,2]);
plt.title('Experimental Distribution');

np.max(mutant_array)

mutant_array = np.array(mutant_array)
zero_mut = (np.where(mutant_array==0)[0]).size
one_mut = (np.where(mutant_array==1)[0]).size
two_mut = (np.where(mutant_array==2)[0]).size
three_mut = (np.where(mutant_array==3)[0]).size
four_mut = (np.where(mutant_array==4)[0]).size
```

```
five_mut = (np.where(mutant_array==5)[0]).size
six_mut = (np.where(mutant_array==6)[0]).size
seven_mut = (np.where(mutant_array==7)[0]).size
eight_mut = (np.where(mutant_array==8)[0]).size
nine_mut = (np.where(mutant_array==9)[0]).size
ten_mut = (np.where(mutant_array==10)[0]).size
print(zero_mut,one_mut,two_mut,three_mut,four_mut,five_mut,six_mut,seven_mut,eight_mut,nine_mut,ten_mut
                                                    )

np.mean(mutant_array)
np.var(mutant_array)

# Calculating the expected average mutants in an acquired regime
mu * (400*2**15-400)
# Calculating the expected variance in an acquired regime
mu * (400*2**15)
np.std(mutant_array)
trimmed=mutant_array[mutant_array<np.std(mutant_array)*3]
np.var(trimmed)
```

✍🏻Problem 3 Code September 9, 2022

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

###Problem 3. Why Are There Jackpots?
# In this problem you should characterize the "age" of each mutant? That is, if you have not already
                                                    done
# so, keep track of the first appearance of each mutant and characterize the relative importance of
# different "age" mutants to the total number at the end. Do early/late mutants contribute
# disproportionately to the total number? Do early/late mutants more strongly influence variation?

cult = 1000
N = 400
mu = 10**(-7)

# Create mutant array
mutant_array = np.zeros(cult)
mutant_array = mutant_array.astype(int)
# Create time index array
mut_time = np.zeros(15)

for gens in range (0,15):
    for i in range(0, np.size(mutant_array)):
        x=np.random.choice([0,1],size=(N*2**(gens)-mutant_array[i],1),p=[1-mu,mu])
        sumx = np.sum(x)
        mutant_array[i] = sumx + 2*mutant_array[i]
        mut_time[gens] = mut_time[gens] + sumx

mut_time

plt.hist(mutant_array,bins=50,color='gray',edgecolor='k');
#plt.xticks([0,1,2]);
plt.title('Experimental Distribution');
```

✍🏻Problem 4 Code

September 9, 2022

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
import matplotlib.pyplot as plt

###Problem 4. Moving Backwards, like L&D, from Observations to Estimates
# Take the results from the first 100 of your experiments in Problem 2, that is the number of resistant
# mutants, m1, m2, ⋯, m100. Treat these 100 numbers as your “data”. Now, write a program that
                                                    leverages
# results of new simulations to infer the most likely value of  , the mutation rate, and (if possible)
                                                    a 95%
# confidence interval for it – in doing so, assume you do not  is in advance. Consider using one of
                                                    two
# pieces of evidence: (i) the number of cultures with no resistant mutants; (ii) the mean number of
# resistant mutants. Finally, ask: is your data also consistent with the acquired resistance or
                                                    spontaneous
# resistance hypothesis? Why or why not?

myfirst100 = mutant_array[:100]
average_mutants = np.average(myfirst100)
average_mutants

est_mu = average_mutants / (15*N*2**15)
est_mu
```