CSE 280A HW 3

Sharad Venkateswaran

February 2021

Problem 1

We have from Fu that $Exp(m_i) = \frac{\theta}{i}$. We can use these to demonstrate all the equalities below.

1a

$Exp(\theta_{FL}) = Exp(m_1) = Exp(\frac{\theta}{1}) = Exp(\theta)$

1b

Proof. Note that $\sum_{i=1}^{n-1} m_i = \sum_{i=1}^{n-1} \frac{1}{i} i m_i = \sum_{i=1}^{n-1} \frac{\theta}{i}$
Then $Exp(\theta_W)$

$$= Exp(\frac{\sum_{i=1}^{n-1} \frac{\theta}{i}}{\sum_{i=1}^{n-1} \frac{1}{i}})$$

$$= Exp(\theta) \qquad \text{From Fu's result with } \alpha_i = \frac{1}{i}$$

□

1c

Proof. $Exp(\theta_L)$

$$= Exp(\frac{1}{n-1} \sum_{i=1}^{n-1} i m_i)$$

$$= Exp(\frac{1}{n-1} \sum_{i=1}^{n-1} \theta) \qquad \text{from Fu's result}$$

$$= Exp(\frac{1}{n-1}(n-1)\theta) = Exp(\theta)$$

□

1d

Proof. $Exp(\theta_\pi)$

$$
\begin{aligned}
&= Exp(\frac{2}{n(n-1)}\sum_{i=1}^{n-1}(n-i)im_i) \\
&= Exp(\frac{2}{n(n-1)}\sum_{i=1}^{n-1}(n-i)\theta) && \text{from Fu's result} \\
&= Exp(\frac{2}{n(n-1)}\sum_{i=1}^{n-1}i\theta) && \text{Summing in the reverse order} \\
&= Exp(\frac{2}{n(n-1)}\frac{n(n-1)}{2}\theta) && \text{closed form of the sum} \\
&= Exp(\theta)
\end{aligned}
$$

□

1e

Proof. $Exp(\theta_H)$

$$
\begin{aligned}
&= Exp(\frac{2}{n(n-1)}\sum_{i=1}^{n-1}i^2m_i) \\
&= Exp(\frac{2}{n(n-1)}\sum_{i=1}^{n-1}i\theta) && \text{from Fu's result} \\
&= Exp(\frac{2}{n(n-1)}\frac{n(n-1)}{2}\theta) && \text{closed form of the sum} \\
&= Exp(\theta)
\end{aligned}
$$

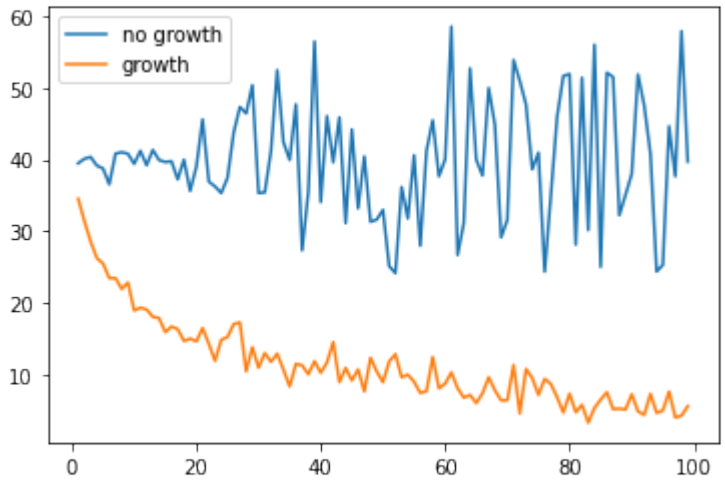
□

1 Problem 2

We can compute the mutation frequencies f_j for all j mutations as $\frac{a_j}{n_j}$. From there we can choose some number of samples n, like $n = max_j(n_j)$, and for each mutation we can determine the number of samples with that mutation as $f_j * n$. After rounding this gives us our mutation counts from which we can compute m_i by simply counting how many mutations affected i individuals for $1 \leq i \leq n-1$. With this we have all the information needed to compute any of the estimates of θ, including Tajima's θ.

Problem 3

I used the msms simulator to generate both populations. The command I used for the population without a growth factor is $msms\ 100\ 1000\ -N\ 1000000\ -t\ 40$, while the command for the exponentially growing population is $msms\ 100\ 1000\ -N\ 1000000\ -t\ 40\ -G\ 7$. The scaled allele frequency spectrums for both populations is shown below:



The population that is not growing has a noisy curve that settles around 40, which is consistent with the expectation as the scaled allele frequency should be around θ for a population that is not under selection. There are a few interesting things about the curve for the population that is growing exponentially. One is that all the datapoints are below the expected value of 40 for a population that is not under selection - we can reason about this from the perspective of the coalescent. Both populations have the same current size, but the exponentially growing population would have taken less time to reach that size. Therefore the coalescent tree for this population will be shorter than the neutrally evolving population, which will then leave less room for mutations to be dropped along the branches. We should then expect less mutations overall in the exponentially growing population, which explains why all datapoints are below the expectation of the neutral population. We can also see that mutations that are present in more individuals are even less common relative to the neutral population than mutations that are present in few individuals. We can again reason about this in terms of coalescent theory. A population that is growing exponentially as time progresses forwards is shrinking exponentially as time moves backwards. Therefore it becomes much easier for two individuals to find a parent at the smaller epochs (higher in the tree), so the final epochs will be short. Then few mutations will appear high in the tree compared to lower in the tree, but mutations high in the tree are exactly the mutations that appear in many individuals.

Problem 4

To generate a sample according to this population diagram I ran msms with the following options, using 25 years per generation:

- -N 14474 (using $N_e = 14474$)
- -t 14.474 (I computed $\theta = \mu * l * 4N_e$ and used $l = 10000bp$ to get a reasonable value of θ)
- -ms 300 100 (Total samples = 300 with 100 replicates)
- -I 3 100 100 100 (Sample 100 individuals from 3 populations)
- -g 1 278 (Population 1 is EAS, growth rate is $0.0048 * 4N_e \approx 278$)
- -g 2 220 (Population 2 is EUR, growth rate is $0.0038 * 4N_e \approx 220$)
- -n 1 3.13 (Current population size of EAS is $45370 \approx 3.13 * N_e$ I computed current population size based on exponential growth over 920 generations)
- -n 2 2.33 (Current population size of EUR is $33814 \approx 2.33 * N_e$ computed current population size based on exponential growth over 920 generations)
- -ej 0.0159 1 2 (EAS joins EUR 23kya = 920 generations ago using generations of 25 years. $\frac{920}{4N_e} \approx 0.0159$)

  -
- ej 0.0352 2 3 (EAS/EUR joins AFR 51kya = 2040 generations ago. $\frac{2040}{4N_e} \approx 0.0352$)
- -en 0.0159 1 0.038 (EAS population size 23kya is 554. $\frac{554}{14474} \approx 0.038$)
- -en 0.0159 2 0.071 (EUR population size 23kya is 1032. $\frac{1032}{14474} \approx 0.071$)
- -en 0.0352 2 0.128 (EAS/EUR population size 51kya is 1861. $\frac{1861}{14474} \approx 0.128$)
- -en 0.102 3 0.505 (AFR population size 148kya is 7310. $\frac{7310}{14474} \approx 0.505$)

Problem 5

In rapidly growing populations we expect more mutations to occur towards the bottom of the coalescent tree, because the branches at the bottom of the tree will be longer due to the population shrinking exponentially as time moves backwards - this translates to more mutations that appear in very few individuals. In Problem 3 we saw that this was true, but since the population sizes were equal the number of "rare" mutations in the exponentially growing population was

still less than that of the neutrally evolving population due to the shorter coalescent tree. However here the population sizes are not equal and demonstrate the exponential growth parameters of the population, so we may be able to leverage the frequency of singleton mutations as a decision metric to determine which population is which.

Given 3 population data samples, we would expect that the population with the most singleton mutations is the fastest-growing population, i.e. EAS, and the population with the least singleton mutations is AFR, which EUR in the middle. The results of this prediction metric is shown below.

Predicted \ Correct	EAS	EUR	AFR
EAS	63	36	1
EUR	37	58	5
AFR	0	6	94

Note that each population is predicted once per sample. With 100 samples we can treat these entries as percentages. This prediction metric does exceedingly well at distinguishing the African population from the other two - this makes sense as the African population is the "odd one out" among the 3, as it is the only one without any growth factor. The other two are both exponentially growing, but the EAS population is growing at a faster rate. Therefore we would expect more mixups between EUR and AFR than EAS and AFR. This is indeed the case - when the correct answer was EAS the model never predicted AFR, but when the correct answer was EUR it predicted AFR 6 times. Likewise when the correct answer was AFR the model predicted EUR 5 times but predicted EAS only once. Since the EAS and EUR populations are growing at roughly similar rates we would expect the model to have a difficult time distinguishing EAS and EUR, and this is also demonstrated in the results. Most incorrect guesses of EAS are when the correct answer is EUR, and vice versa.

In [35]:

```
import string
import math
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
```

In [48]:

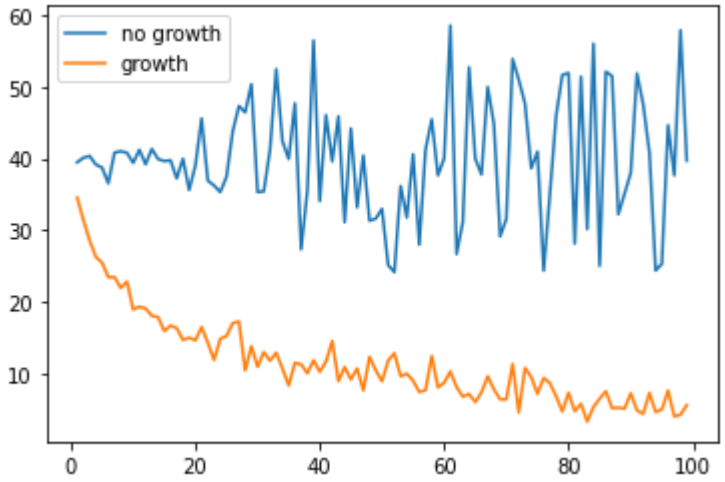
```
# Problem 3 code

def lineSplit(line):
    retList = []
    for c in line:
        retList.append(int(c))

    return retList

data_all_ns = open("./100_selection.txt")
data_all_s = open("./100_no_selection.txt")
num_samples = 100
num_reps = 1000
data_all_ns = [line for line in data_all_ns.read().splitlines()]
data_all_s = [line for line in data_all_s.read().splitlines()]
sample_data_s = []
sample_data_ns = []
allele_freqs_s = defaultdict(int)
allele_freqs_ns = defaultdict(int)


# Data starts on line 6
line_num = 6

for i in range(num_reps):
    sample_data_s = data_all_s[line_num:line_num + num_samples]
    sample_data_ns = data_all_ns[line_num:line_num + num_samples]
    data_split_s = [lineSplit(line) for line in sample_data_s]
    data_split_ns = [lineSplit(line) for line in sample_data_ns]
    data_np_s = np.array(data_split_s)
    data_np_ns = np.array(data_split_ns)

    for m in range(data_np_s.shape[1]):
        mut_count_s = sum(data_np_s[:,m])
        allele_freqs_s[mut_count_s] += 1
    for m in range(data_np_ns.shape[1]):
        mut_count_ns = sum(data_np_ns[:,m])
        allele_freqs_ns[mut_count_ns] += 1



    line_num += num_samples + 4 # Skip 4 lines for metadata
```

```
43
44
45 for key in allele_freqs_s.keys():
46     allele_freqs_s[key] = float(allele_freqs_s[key] * key) / num_reps
47 for key in allele_freqs_ns.keys():
48     allele_freqs_ns[key] = float(allele_freqs_ns[key] * key) / num_reps
```

In [49]:

```
1 x_vals = [i for i in range(1,num_samples)]
2 y_vals_s = [allele_freqs_s[i] for i in range(1,num_samples)]
3 y_vals_ns = [allele_freqs_ns[i] for i in range(1,num_samples)]
4
5 plt.plot(x_vals, y_vals_s, label="no growth")
6 plt.plot(x_vals, y_vals_ns, label="growth")
7 plt.legend()
```

Out[49]: <matplotlib.legend.Legend at 0x22594cc33d0>



In [ ]:

```
1
```

In [1]:

```
import string
import math
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
```

In [31]:

```
# Problem 3 code

def lineSplit(line):
    retList = []
    for c in line:
        retList.append(int(c))

    return retList

data_all = open("./Q5.txt")
num_samples = 100
num_reps = 100
data_all = [line for line in data_all.read().splitlines()]
sample_data_EAS = []
sample_data_EUR = []
sample_data_AFR = []

allele_freqs_EAS = defaultdict(int)
allele_freqs_EUR = defaultdict(int)
allele_freqs_AFR = defaultdict(int)



# Data starts on line 6
line_num = 6

correct_EAS = 0
correct_EUR = 0
correct_AFR = 0
predictions_list = [] # list of (predicted, correct) values for confusion matrix
for i in range(num_reps):
    sample_data_EAS = data_all[line_num:line_num + num_samples]
    sample_data_EUR = data_all[line_num+num_samples:line_num + 2*num_samples]
    sample_data_AFR = data_all[line_num+2*num_samples:line_num + 3*num_samples]
    data_split_EAS = [lineSplit(line) for line in sample_data_EAS]
    data_split_EUR = [lineSplit(line) for line in sample_data_EUR]
    data_split_AFR = [lineSplit(line) for line in sample_data_AFR]

    data_np_EAS = np.array(data_split_EAS)
    data_np_EUR = np.array(data_split_EUR)
    data_np_AFR = np.array(data_split_AFR)
    singletons_EAS = 0
```

```
    singletons_EUR = 0
    singletons_AFR = 0

    for m in range(data_np_EAS.shape[1]):
        if sum(data_np_EAS[:,m]) == 1:
            singletons_EAS += 1
        if sum(data_np_EUR[:,m]) == 1:
            singletons_EUR += 1
        if sum(data_np_AFR[:,m]) == 1:
            singletons_AFR += 1
    singletons_list = [(singletons_EAS, "EAS"), (singletons_EUR, "EUR"), (singletons_AFR, "AFR")]
    sorted_singletons = sorted(singletons_list, key = lambda x: x[0], reverse=True)

    most_singletons = sorted_singletons[0][1]
    middle_singletons = sorted_singletons[1][1]
    least_singletons = sorted_singletons[2][1]

    predictions_list.append(("EAS"), most_singletons))
    predictions_list.append(("EUR"), middle_singletons))
    predictions_list.append(("AFR"), least_singletons))

    line_num += 3*num_samples + 4 # Skip 4 lines for metadata


print("Predicted EAS when correct answer was EAS " + str(predictions_list.count(("EAS", "EAS"))) + " times"
print("Predicted EAS when correct answer was EUR " + str(predictions_list.count(("EAS", "EUR"))) + " times"
print("Predicted EAS when correct answer was AFR " + str(predictions_list.count(("EAS", "AFR"))) + " times"
print("Predicted EUR when correct answer was EUR " + str(predictions_list.count(("EUR", "EUR"))) + " times"
print("Predicted EUR when correct answer was EAS " + str(predictions_list.count(("EUR", "EAS"))) + " times"
print("Predicted EUR when correct answer was AFR " + str(predictions_list.count(("EUR", "AFR"))) + " times"
print("Predicted AFR when correct answer was AFR " + str(predictions_list.count(("AFR", "AFR"))) + " times"
print("Predicted AFR when correct answer was EAS " + str(predictions_list.count(("AFR", "EAS"))) + " times"
print("Predicted AFR when correct answer was EUR " + str(predictions_list.count(("AFR", "EUR"))) + " times"
```

```
Predicted EAS when correct answer was EAS 63 times
Predicted EAS when correct answer was EUR 36 times
Predicted EAS when correct answer was AFR 1 times
Predicted EUR when correct answer was EUR 58 times
Predicted EUR when correct answer was EAS 37 times
Predicted EUR when correct answer was AFR 5 times
Predicted AFR when correct answer was AFR 94 times
Predicted AFR when correct answer was EAS 0 times
Predicted AFR when correct answer was EUR 6 times
```