ASSIGNMENT 1, MATHS IN ACTION A, 2018

TIANWEI LIU

Abstract. The goal of this project is to study population genetics and major mathematical models in this field. Among those models, the Wright-Fisher Model and the Moran Model are the most popular ones. In this writeup, we will be first investigating different aspects of the Moran Model, including its fixation probability, mean Absorption time, mean Fixation time, mean Extinction time, and at the end, we will also survey some aspects of the Wright-Fisher model.

1. Introduction

The Wright-Fisher Model and the Moran Model provide us a way to quantify the average time for mutants to disappear or dominate in the whole population and the probability that it will dominate. Those metrics are important for us to comprehend the stochastic process of genetic mutations. In the sections below, I will be describing the calculations and simulations that I performed. Section 2 will be about Neutral Moran Model without mutant fitness. I will derive the mean absorption time t_i, compare it with my simulation, and simulate the mean extinction time and the mean Fixation time. Those correspond to my answer to question 1, 2, and 3. Section 3 will be about Moran Model with mutant fitness. I will repeat the same simulations for question 4. Section 5 will be about the Wright-Fisher Model. I will be talking about what I have done for question 5.

2. Neutral Moran Model without Mutant Fitness

I chose $M = 25$ total genes, and iterate through the number of mutants $i = 1, 2, 3, ..., 24$ with 10000 simulations in each iteration. The transition probabilities are

$$\begin{aligned} p_i &\equiv p_{i,i+1} = \frac{i(M-i)}{M^2} \\ q_i &\equiv p_{i,i-1} = \frac{i(M-i)}{M^2} \\ p_{i,i} &= 1 - p_i - q_i \end{aligned} \tag{1}$$

from equation (1.20) on page 6 of the Lecture Notes [1]. Corresponding python code can be found in Appendix B. Neutral Moran Model without Mutant Fitness.

2.1. **Question 1.** The result of Question 1 are shown below.

Because $\tau_i = t_{i+1,j} - t_{i,j}$,

$$\begin{aligned} p_i\tau_i &= q_i\tau_{i-1} \\ \tau_i &= \prod_{k=1}^{i} \frac{q_k}{p_k} t_{1,j} \end{aligned} \tag{2}$$

From the boundary condition $t_{M-1,j} = 0$ and $t_{1,j} = 0$,

$$\tau_{M-1,j} = t_{M-1,j} = \prod_{k=1}^{M-1} \frac{q_k}{p_k} t_{1,j} \tag{3}$$
$$t_{1,j} = (1 - p_1 - q_1)t_{1,j} + p_1 t_{2,j}$$

The mean local times are

$$t_{i,j} = \begin{cases} M\frac{i}{j} & i < j \\ M\frac{M-i}{M-j} & i \geq j \end{cases} \tag{4}$$

The derivation from $t_{i,j}$ to t_i

$$\begin{aligned} t_i &= \sum_{j=1}^{M-1} t_{i,j} = t_{i,j} + t_{i,2} + ... + t_{i,i} + t_{i,M-1} \\ &= M\Big[(M-i)\sum_{j=1}^{i} \frac{1}{M-j} + i\sum_{j=i+1}^{M-1} \frac{1}{j}\Big] \end{aligned} \tag{5}$$

Then

$$t_i = M((M-i)(H_{M-1} - H_{M-i-1}) + i(H_{M-1} - H_i)) \tag{6}$$

.

For detailed derivation, please see Appendix A. Complete Derivation of Question 1

2.2. **Question 2.** I counted the total steps for each initial number of mutants to fixate or to extinct and the number of fixations for each initial number of mutants during simulation. I calculated the simulated mean absorption time as

$$t_{mean\ absorption\ time} = \frac{(t_{fixation} + t_{extinction})}{number\ of\ simulations} \tag{7}$$

The theoretical absorption time is calculated via the equation (1.37) on page 9 of the Lecture Notes, and it is shown below [1].

$$t_i = M[(M-i)(H_{M-1} - H_{M-i-1}) + i(H_{M-1} - H_i)] \tag{8}$$

In Figure 1 shown below, I compare the mean absorption time t_i obtained from the derived equation with that from my simulation with respect to the number of mutants.

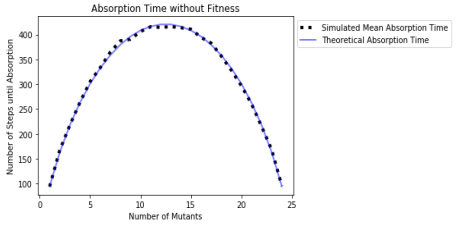

Figure 1. Absorption Time

It is clear that they align with each other closely. The theoretical value is confirmed by the simulation result.

2.3. **Question 3.** As for question 3, I plotted the Mean Fixation Time and Mean Extinction Time in the same graph with respect to the number of initial mutants. Additionally, to give a better global view, I plotted the Mean Absorption time on the same chart. They are shown in Figure 2 below.

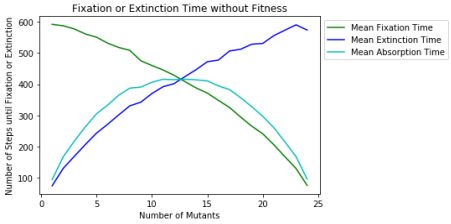

Figure 2. Absorption/Fixation/Extinction Time

It is clear that the Mean Fixation Time and the Mean Extinction Time are symmetrical to each other, and the Mean Absorption Time itself is symmetrical as well.

3. Biased Moran Model with Mutant Fitness

Again, I chose $M = 25$ total genes, and iterate through the number of mutants $i = 1, 2, 3, ..., 24$ with 10000 simulations in each iteration. The fitness was set to 0.8. The transition probabilities are

$$
\begin{aligned}
p_i \equiv p_{i,i+1} &= \frac{if}{if + M - i}\frac{M - i}{M} \\
q_i \equiv p_{i,i-1} &= \frac{M - i}{if + M - i}\frac{i}{M} \\
p_{i,i} &= 1 - p_i - q_i
\end{aligned}
\tag{9}
$$

from equation (1.29) on page 8 of the Lecture Notes [1]. Corresponding python code can be found in Appendix C. Biased Moran Model without Mutant Fitness.

3.1. **Question 4.** I first plotted the theoretical fixation probability and the simulated Fixation probability in Figure 3. The theoretical Fixation probability is calculated by

$$\epsilon_i = \frac{1 - 1/f^i}{1 - 1/f^M} \tag{10}$$

from equation (1.30) on page 8 of the Lecture Notes [1]. Then I plotted its Absorption, Fixation, Extinction Time on Figure 4. The theoretical Fixation probability is again confirmed by the

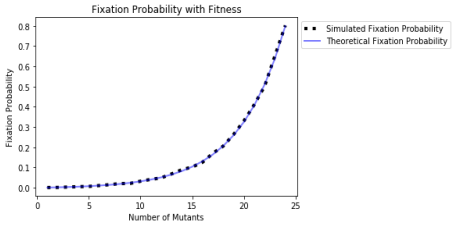

Figure 3. Fixation Probability

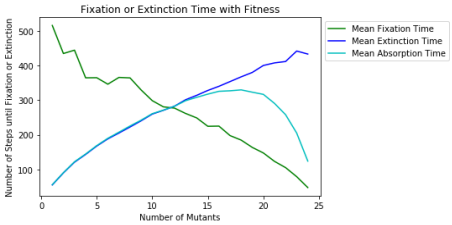

Figure 4. Absorption, Fixation, Extinction Time

simulated value. It is much harder for mutants to fixate because it is not as fit as its wild type counterpart. The Mean Absorption Time is left skewed. It takes much longer time for mutants to fixate when the number of initial mutants are small, and it does not take longer time for mutants to extinct when the number of initial mutants are increased.

4. Wright-Fisher Model

After an in-depth investigation into the Moran Model, I question myself that what is the relationship between the Absorption, Fixation, Extinction time of the Wright-Fisher model and the number of initial mutants? To survey on this question, I simulated the Wright-Fisher model with transition probability

$$p_{i,j} = binomial(M, \frac{i}{M}) \tag{11}$$

derived from equation (1.7) on page 4 of the Lecture Notes [1]. Corresponding python code can be found in Appendix D. Wright-Fisher Model.

4.1. **Question 5.** I initialized $M = 1000$ total genes, and picked 25 points from $[1, 999]$ interval using *np.linspace* to be the number of initial mutants. For each number of initial mutants, I did 10000 simulations. I plotted the Absorption, Fixation, Extinction time in Figure 5 below. We

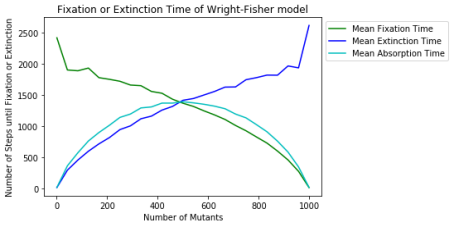

Figure 5. Absorption/Fixation/Extinction Time

can see that just like the neutral Moran Model, the Fixation time and the Extinction time are symmetrical to each other and the Absorption time is symmetrical itself. However, with extremely small or large number of initial mutants, it is much difficult for mutants to fixate or extinct as the time skyrockets.

5. Discussion and conclusions

The neural Moran Model and neural Wright-Fisher model are symmetric, while the biased Moran Model is not. The biased Moran Model is probably the most realistic as it takes into consideration that different gene have different fitness to the environment. If one allele has a fitness advantage over the other allele, it will be more likely to be chosen for reproduction. Yet, all of those three models are not realistic because they assume finite population which is extremely rare in nature. The population of most species fluctuates from time to time and can be modelled as a dynamical system. Unfortunately, I am not a quantitative biologist who is capable of inventing a new model that solves this conundrum.

References

[1] Lecture notes.
[2] W. Ewens: Mathematical Population Genetics, Springer 2004, New York.

Appendix A. Complete Derivation of Question 1

Below is my derivation for question 1:

$$t_{i,j} = q_i t_{i-1,j} + (1 - p_i - q_i)t_{i,j} + p_i t_{i+1,j} + \delta_{i,j} \tag{12}$$

because $\delta_i = 1(i = j), \delta_i = 0(i \neq j)$, When $i \neq j$,

$$\begin{aligned} t_{i,j} &= q_i t i - 1, j + (1 - p_i - q_i)t_{i,j} + p_i t_{i+1,j} \\ q_i t_{i-1,j} - p_i t_{i,j} &= q_i t_{i,j} + p_i t_{i+1,j} \\ p_i(t_{i+1,j} - t_{i,j}) &= q_i(t_{i,j} - t_{i-1,j}) \end{aligned} \tag{13}$$

Because $\tau_i = t_{i+1,j} - t_{i,j}$,

$$\begin{aligned} p_i \tau_i &= q_i \tau_{i-1} \\ \tau_i &= \prod_{k=1}^{i} \frac{q_k}{p_k} t_{1,j} \end{aligned} \tag{14}$$

From the boundary condition $t_{M-1,j} = 0$ and $t_{1,j} = 0$,

$$\begin{aligned} \tau_{M-1,j} = t_{M-1,j} &= \prod_{k=1}^{M-1} \frac{q_k}{p_k} t_{1,j} \\ t_{1,j} &= (1 - p_1 - q_1)t_{1,j} + p_1 t_{2,j} \end{aligned} \tag{15}$$

Using

$$t_{M-1,j} = (1 - p_{M-1} - q_{M-1})t_{1,j} + p_{M-1} t_{2,j} \tag{16}$$

The mean local times are

$$t_{i,j} = \begin{cases} M\frac{i}{j} & i < j \\ M\frac{M-i}{M-j} & i \geq j \end{cases} \tag{17}$$

The derivation from $t_{i,j}$ to t_i

$$\begin{aligned} t_i &= \sum_{j=1}^{M-1} t_{i,j} = t_{i,j} + t_{i,2} + ... + t_{i,i} + t_{i,M-1} \\ &= \sum_{j=1}^{i} M\frac{M-i}{M-j} + \sum_{j=i+1}^{M-1} M\frac{i}{j} = M\Big[\sum_{j=1}^{i} \frac{M-i}{M-j} + \sum_{j=i+1}^{M-1} \frac{i}{j}\Big] \\ &= M\Big[(M-i)\sum_{j=1}^{i} \frac{1}{M-j} + i\sum_{j=i+1}^{M-1} \frac{1}{j}\Big] \end{aligned} \tag{18}$$

Then

$$\begin{aligned} \sum_{j=1}^{i} \frac{1}{M-j} &= \Big(\frac{1}{M-1} + ... + 1\Big) - \Big(\frac{1}{M-i-1} + ... + 1\Big) \\ &= H_{M-1} - H_{M-i-1} \end{aligned} \tag{19}$$

$$\begin{aligned}\sum_{j=1+1}^{M-1}\frac{1}{j} &= \Big(1+...+\frac{1}{M-1}\Big)-\Big(1+...+\frac{1}{i}\Big)\\ &= H_{M-1}-H_i\end{aligned} \tag{20}$$

Then, we can get t_i,

$$t_i = M((M-i)(H_{M-1}-H_{M-i-1})+i(H_{M-1}-H_i)) \tag{21}$$

.

Appendix B. Neutral Moran Model without Mutant Fitness

```
# The setup for the programming component of this project
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from itertools import product

# Set the seed
np.random.seed(0)
# Set graph size
plt.rcParams['figure.figsize'] = (20,10)
plt.rcParams['figure.dpi'] = 80

def H(num_mut):
    res = 0
    for i in range(num_mut):
        res += 1 / (i + 1)
    return res

# Simulate Moran Process without fitness
total_gene = 25
mutant = np.arange(1, total_gene)
num_simulation = 10000
fixed = np.zeros(mutant.shape[0])   # the fixation frequency
t_fixation = np.zeros(mutant.shape[0])   # the fixation time
t_extinction = np.zeros(mutant.shape[0])   # the extinction time
theoretical_absorption_time = np.zeros(mutant.shape[0]) # the theoretical absorption time h
theoretical_fixation_probability = mutant / total_gene

for i in range(theoretical_absorption_time.shape[0]):
    theoretical_absorption_time[i] = total_gene * \
        ((total_gene-mutant[i]) * \
         (H(total_gene-1) - H(total_gene-mutant[i]-1)) + \
         mutant[i] * (H(total_gene-1) - H(mutant[i])))
```

```
ti = np.zeros((total_gene-1, num_simulation))
for i, j in product(range(mutant.shape[0]), range(num_simulation)):
    mut = mutant[i]
    step = 0
    while mut > 0 and mut < total_gene:
        rng = np.random.uniform()
        # the probability that there is an increment in the number of mutants
        p_inc_mut = mut * (total_gene-mut) / total_gene**2
        # the probability that there is an increment in the number of wild type
        p_dec_mut = mut * (total_gene-mut) / total_gene**2
        if rng < p_inc_mut:
            mut += 1
        elif rng > 1 - p_dec_mut:
            mut -= 1
        step += 1
    if mut == total_gene: # The mutant fixates
        fixed[i] += 1
        t_fixation[i] += step
    else:
        t_extinction[i] += step

mean_t_fixation = t_fixation / fixed
mean_t_extinction = t_extinction / (num_simulation-fixed)
mean_t_absorption = (t_fixation + t_extinction) / num_simulation

# Plot the Mean Absorption Time without Mutant Fitness
plt.plot(mutant, mean_t_absorption, "k:", linewidth = 4, label = "Mean Absorption Time")
plt.plot(mutant, theoretical_absorption_time, "b-", linewidth = 2, alpha = 0.5, label = "Th
plt.xlabel("Number of Mutants")
plt.ylabel("Number of Steps until Absorption")
plt.title("Absorption Time without Fitness")
plt.legend(bbox_to_anchor=(1, 1))
plt.savefig("Absorption_Time_without_Fitness", bbox_inches="tight")

# Plot the Mean Fixation, Extinction, Absorption Time
plt.plot(mutant, mean_t_fixation, "g", label = "Mean Fixation Time")
plt.plot(mutant, mean_t_extinction, "b", label = "Mean Extinction Time")
plt.plot(mutant, mean_t_absorption, "c", label = "Mean Absorption Time")
plt.xlabel("Number of Mutants")
plt.ylabel("Number of Steps until Fixation or Extinction")
plt.title("Fixation or Extinction Time without Fitness")
plt.legend(bbox_to_anchor=(1, 1))
plt.savefig("Fixation_Extinction_Time_without_Fitness", bbox_inches="tight")
```

Appendix C. Biased Moran Model with Mutant Fitness

```
# The setup for the programming component of this project
```

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from itertools import product

# Set the seed
np.random.seed(0)
# Set graph size
plt.rcParams['figure.figsize'] = (20,10)
plt.rcParams['figure.dpi'] = 80

def local_time(total_gene, mut, fitness):
    res = 0
    for j in range(1, total_gene):
        if mut < j:
            res += total_gene * mut * fitness / j
        else:
            res += total_gene * (total_gene - mut * fitness) / (total_gene - j)
    return res

# Simulate Moran Process  with fitness
total_gene = 25
mutant = np.arange(1, total_gene)
num_simulation = 10000
fixed = np.zeros(mutant.shape[0])   # the fixation frequencyqa
t_fixation = np.zeros(mutant.shape[0])   # the fixation time
t_extinction = np.zeros(mutant.shape[0])   # the extinction time
fitness = 0.8 # the fitness of the mutant
theoretical_fixation_probability = (1 - 1 / fitness**mutant)/((1 - 1 / fitness**total_gene)

for i, j in product(range(mutant.shape[0]), range(num_simulation)):
    mut = mutant[i]
    step = 0
    while mut > 0 and mut < total_gene:
        rng = np.random.uniform()
        # the probability that there is an increment in the number of mutants
        p_inc_mut = mut * fitness * (total_gene - mut) / \
            (total_gene * (mut*fitness + total_gene - mut))
        # the probability that there is an increment in the number of wild type
        p_dec_mut = mut * (total_gene-mut) / \
            (total_gene * (mut*fitness + total_gene - mut))
        if rng < p_inc_mut:
            mut += 1
        elif rng > 1 - p_dec_mut:
            mut -= 1
        step += 1
```

```
    if mut == total_gene: # The mutant fixates
        fixed[i] += 1
        t_fixation[i] += step
    else:
        t_extinction[i] += step

prob_fixation = fixed / num_simulation
mean_t_fixation = t_fixation / fixed
mean_t_extinction = t_extinction / (num_simulation-fixed)
mean_t_absorption = (t_fixation + t_extinction) / num_simulation

# Plot Fixation Probability without Mutant Fitness
plt.plot(mutant, prob_fixation, "k:", linewidth = 4, label = "Simulated Fixation Probabilit
plt.plot(mutant, theoretical_fixation_probability, "b-", linewidth = 2, alpha = 0.5, label
plt.xlabel("Number of Mutants")
plt.ylabel("Fixation Probability")
plt.title("Fixation Probability with Fitness")
plt.legend(bbox_to_anchor=(1, 1))
plt.savefig("Fixation_Probability_with_Fitness", bbox_inches="tight")

# Plot the Mean Fixation, Extinction, Absorption Time
plt.plot(mutant, mean_t_fixation, "g", label = "Mean Fixation Time")
plt.plot(mutant, mean_t_extinction, "b", label = "Mean Extinction Time")
plt.plot(mutant, mean_t_absorption, "c", label = "Mean Absorption Time")
plt.xlabel("Number of Mutants")
plt.ylabel("Number of Steps until Fixation or Extinction")
plt.title("Fixation or Extinction Time with Fitness")
plt.legend(bbox_to_anchor=(1, 1))
plt.savefig("Fixation_Extinction_Time_with_Fitness", bbox_inches="tight")
```

Appendix D. Wright-Fisher Model

```
# The setup for the programming component of this project
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from itertools import product

# Set the seed
np.random.seed(0)
np.seterr(divide='ignore', invalid='ignore')
# Set figure size
plt.rcParams['figure.figsize'] = (20,10)
plt.rcParams['figure.dpi'] = 80

# Simulate Wright-Fisher model
total_gene = 1000
```

```
mutant = np.linspace(1, total_gene - 1, num = 25)
num_simulation = 10000
fixed = np.zeros(mutant.shape[0])   # the fixation frequency
t_fixation = np.zeros(mutant.shape[0])   # the fixation time
t_extinction = np.zeros(mutant.shape[0])   # the extinction time
fitness = 0.75 # the fitness of the mutant
theoretical_absorption_time = np.zeros(total_gene-1) # the theoretical absorption time by i

for i, j in product(range(mutant.shape[0]), range(num_simulation)):
    mut = mutant[i]
    step = 0
    while 0 < mut and mut < total_gene:
        mut = np.random.binomial(total_gene, mut / total_gene)
        step += 1
    if mut == total_gene:
        fixed[i] += 1
        t_fixation[i] += step
    else:
        t_extinction[i] += step
    if j == 0:
        print(i)

mean_t_fixation = t_fixation / fixed
mean_t_extinction = t_extinction / (num_simulation-fixed)
mean_t_absorption = (t_fixation + t_extinction) / num_simulation

# Plot the Mean Fixation, Extinction, Absorption Time
plt.plot(mutant, mean_t_fixation, "g", label = "Mean Fixation Time")
plt.plot(mutant, mean_t_extinction, "b", label = "Mean Extinction Time")
plt.plot(mutant, mean_t_absorption, "c", label = "Mean Absorption Time")
plt.xlabel("Number of Mutants")
plt.ylabel("Number of Steps until Fixation or Extinction")
plt.title("Fixation or Extinction Time of Wright-Fisher model")
plt.legend(bbox_to_anchor=(1, 1))
plt.savefig("Fixation_Extinction_Time_with_WF", bbox_inches="tight")
```