MEAN ABSORPTION TIME OF THE MORAN MODEL

JONATHON D'ARCY

Abstract. This report acts as a continuation of section 1.5 in [1]. In particular it aims to answer questions about the mean absorption time, in general, under conditioning of absorption type, and in respect to fitness. In conclusion we will answer questions about the shape of the distribution itself.

1. Introduction

The Moran model, which was first described by Patrick Moran in 1958, is a basic stochastic model which can be used to describe a finite population of mutants and non-mutants over a series of generations. Ultimately, for any Moran model it is known that the mutants will either become extinct or fixate given enough time; the time taken is called the absorption time, t_i. Being able to find this value is important for any stochastic process as it denotes the amount of time we expect the process to take. An example for the importance of this value is when modelling disease spread throughout a population, this value would denote the expect amount of time before either the entire population is infected or the disease dies out.

2. Deriving the Mean Absorption Time

When observing a neutral Moran model of i mutants in a population of size M, $Moran(i, M)$, we find the mean absorption time can be derived from the equation of mean time spent in state j when starting in state i given in [1] (1.34):

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

where $\delta_{i,j}$ denotes the kroenecker delta. Since this is a neutral model we have that $p_i = q_i$ allowing us to simplify (1) as $t_{i,j} = \frac{1}{2}(t_{i-1,j} + t_{i+1,j} + \frac{\delta_{i,j}}{q_i})$, which can then be rearranged to form the recurrence relation, $t_{i+1,j} = 2t_{i,j} - t_{i-1,j} - \frac{\delta_{i,j}}{q_i}$. By manipulating the terms in the recurrence we see it is equivalent to $t_{i+1,j} - t_{i,j} = t_{i,j} - t_{i-1,j} - \frac{\delta_{i,j}}{q_i}$ and by defining,

$$\tau_{i,j} = t_{i+1,j} - t_{i,j} \tag{2}$$

the relation becomes $\tau_{i+1,j} = \tau_{i,j} - \frac{\delta_{ij}}{q_i}$. We recognise this new form as a telescoping series, meaning we can define $\tau_{i+1,j}$ using any $\tau_{i-k,j}$ with integer $k < i$ as

$$\tau_{i+1,j} = t_{i-k,j} - \sum_{n=0}^{i-k} \frac{\delta_{i-n,j}}{q_{i-n}}. \tag{3}$$

Further we see that if we define $k = i - 1$ we can define $\tau_{i+1,j}$ as,

$$\tau_{i+1,j} = \begin{cases} t_{1,j} & j > i \\ t_{1,j} - \frac{1}{q_j} & j \leq i \end{cases} \tag{4}$$

This general form of $\tau_{i,j}$ can be used in a similar contraction of the telescoping series of (2) to get the general form of $t_{i,j}$,

$$t_{i,j} = t_{0,j} + \sum_{k=0}^{i-1} \tau_{k,j} = \sum_{k=0}^{i-1} \tau_{k,j} = \begin{cases} it_{1,j} & j > i \\ it_{1,j} - \frac{i-j}{q_j} & j \le i. \end{cases} \tag{5}$$

Using the fact that $t_{M,j} = 0$, where M is the fixed population size, and for any j we have $j \le M$. We can gather that $0 = Mt_{1,j} - \frac{M-j}{q_j}$ and using the definition of $q_j = \frac{j}{M}\frac{M-j}{M}$ we can calculate $t_{1,j} = \frac{M}{j}$. Therefore we can write (5) as

$$t_{i,j} = \begin{cases} \frac{iM}{j} & j > i \\ \frac{iM}{j} - \frac{(i-j)M^2}{j(M-j)} & j \le i \end{cases} = \begin{cases} \frac{iM}{j} & j > i \\ \frac{M^2 - iM}{M-j} & j \le i. \end{cases} \tag{6}$$

From here we calculate t_i as the sum of $t_{i,j}$ over j, ie as,

$$t_i = \sum_{j=1}^{M-1} t_{i,j} = \sum_{j=1}^{i} t_{i,j} + \sum_{j=i+1}^{M-1} t_{i,j} = \sum_{j=1}^{i} M\frac{M-i}{M-j} + \sum_{j=i+1}^{M-1} M\frac{i}{j}. \tag{7}$$

Which we can rewrite using the harmonic numbers as,

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

To verify this result python was used to simulate a set of independent Moran models. These models were then used to calculate the set's true mean absorption time which was then compared to the expected mean absorption time. In figure 1, we see a simulation of a 10000 Moran(5,10) models which gave a true mean absorption of time of 64.81 generations. For comparison the expected mean absorption time for a Moran(5,10) is 64.56 generations.

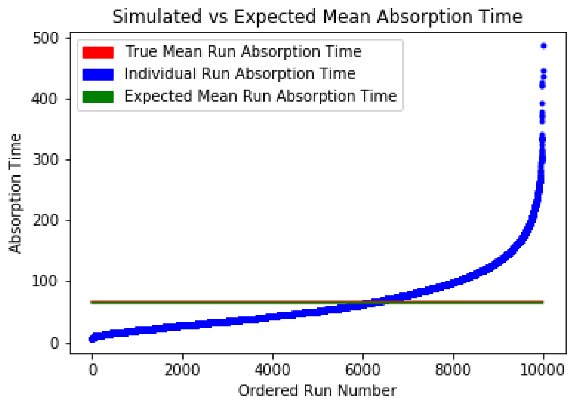


FIGURE 1

3. Conditional Absorption

As we have studied the mean time of either fixation or extinction, it would be fitting to study them individually. We have from [1] section 1.5.3 a derivation for the mean fixation time of a general Moran(i,M) model giving,

$$t_i^* = M^2\frac{M-i}{i}(H_{M-1} - H_{M-i-1}) - M. \tag{9}$$

Where t_i^* denotes the mean absorption time conditioned on fixation. A similar method can be imposed to conduct a derivation for mean extinction time, however there is a more efficient way. As currently we are only observing neutral models we have symmetry of the expected fixation time and expected extinction time around $M/2$ for the initial number of mutants. This allows the expected extinction time to be calculated from the expected fixation time by taking the initial number of mutants to be $M-i$ instead of i, implementing this we gather that t_i^{**}, the mean time of absorption conditioned on extinction is of the form,

$$t_i^{**} = t_{M-i}^* = M^2\frac{i}{M-i}(H_{M-1} - H_{i-1}) - M, \tag{10}$$

for neutral models. We can convince ourselves that this is true by deriving the mean extinction time from the mean absorption and fixation time. To do this we use the knowledge that the Moran must either fixate with probability ϵ_i or become extinct with probability $1-\epsilon_i$. This allows us to say that,

$$\epsilon_i t_i^* + (1-\epsilon_i)t_i^{**} = t_i \tag{11}$$

We have from [1] that in neutral Moran models $\epsilon_i = i/M$, and substituting in our result in (8) and the mean fixation time we can derive the following

$$M(M-i)(H_{M-1} - H_{M-i-1}) - i + \frac{M-i}{M} t_i^{**} = t_i \tag{12}$$

$$\frac{M-i}{M} t_i^{**} = Mi(H_{M-1} - H_i) + i \tag{13}$$

$$t_i^{**} = \frac{M}{M-i}(Mi(H_{M-1} - H_{i-1} - \frac{1}{i}) + i) \tag{14}$$

$$t_i^{**} = M^2 \frac{i}{M-i}(H_{M-1} - H_{i-1}) - M. \tag{15}$$

4. Fitness

The past sections have only considered neutral Moran models, however we can implement a fitness, f, to the mutants, which will make mutants more or less likely to be chosen for reproduction by a factor. We will denote Moran models of this type as $Moran(i, M, f)$. When we do this the probability of fixation over extinction, denoted as ϵ_i, changes meaning we have to discuss it in the form,

$$\epsilon_i = \frac{1 - (\frac{1}{f})^i}{1 - (\frac{1}{f})^M}. \tag{16}$$

The deviation for this ϵ_i can be found once again in [1]. Though we can also verify this fact once again in simulation. Below (Figure 2) is the output for a program which runs 10000 Moran(7,10,0.7) and calculates the true proportions which end in fixation. As we can see the approximated propor-

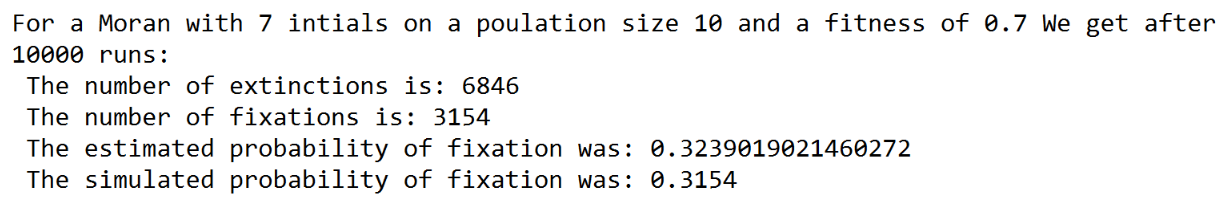

```
For a Moran with 7 intials on a poulation size 10 and a fitness of 0.7 We get after
10000 runs:
 The number of extinctions is: 6846
 The number of fixations is: 3154
 The estimated probability of fixation was: 0.3239019021460272
 The simulated probability of fixation was: 0.3154
```

Figure 2

tion of 0.3154 for a model of this time is close to this simulations true proportion of approximately 0.3239. We also can see that with fitness implemented we can no longer have symmetry over the central point of $M/2$ and thus $t_1^* \neq t_{M-1}^{**}$. However, we can recreate this symmetry by manipulating the fitness in the extinction model. Consider the mean fixation time on $Moran(i, M, f_1)$ here we find we are not symmetric with the mean extinction time of $Moran(M-i, M, f_1)$ as we have implimented a skew, however there does exist symmetry in the mean extinction time of

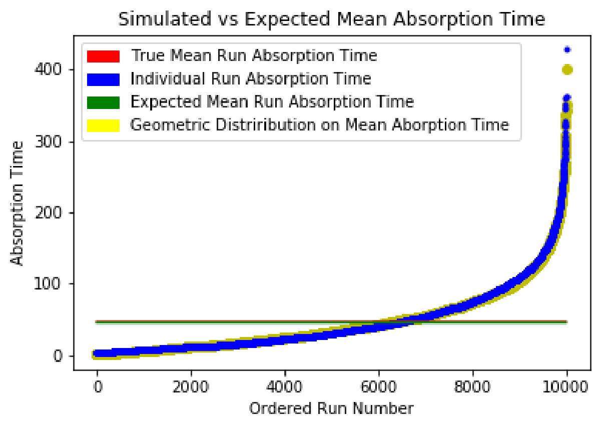
$Moran(M-i, M, f_2)$ where f_2 is defined as the positive real solution to:

$$1 - \frac{1-(\frac{1}{f_1})^i}{1-(\frac{1}{f_1})^M} = \frac{1-(\frac{1}{f_2})^{M-i}}{1-(\frac{1}{f_2})^M} \tag{17}$$

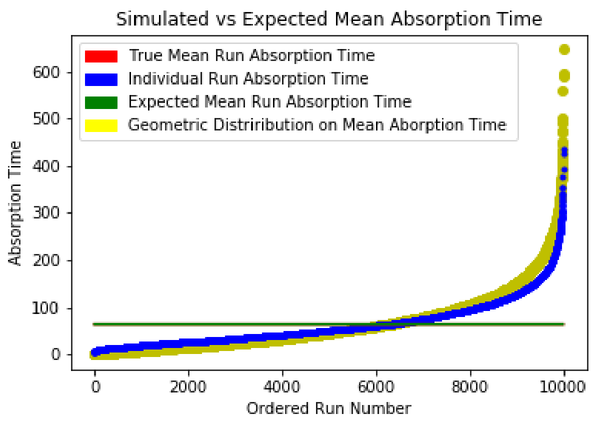
These solutions are difficult to solve algebraically, however using any root finding algorithm the value of f_2 can be derived.

5. The Distribution of t_i

In section 3 a neutral Moran model was simulated 10000 and sorted by absorption time. It may have been noticed that the Moran appeared to follow the geometric distribution. Using our knowledge of the theoretical absorption mean and the mean of the geometric distribution we can gather the parameter, p, of the geometric distribution this may fit. We thus find that if the Moran model does follow the geometric distribution then, Moran$(i, M) \sim$ Geo$(1/t_i)$. This approximation is plotted in figure 3(A) and 3(B) for 10000 runs of Moran(2,10) and Moran(5,10) respectively. Interestingly, we see that the approximation for an initial position of 2 is much better than that of



(A) 10000 runs of Moran(2,10)



(B) 10000 runs of Moran(5,10)

Figure 3

5. The reason behind this is due to the variance of the Moran model and the geometric distribution not being the same form. For the Moran(5,10) simulation the variance of the absorption times was 2468.92 while the variance of the geometric is 1403.88. For the Moran(2,10) the observed variance was 2186.1936 which laid much closer to the value of the exponential distributions 2021.89. Further, it can be found that in general only Neutral Moran models with initial values of approximately $[M/5, M/4]$ or $[3M/4, 4M/5]$ have similar variances to this geometric distribution.

6. Discussion and conclusions

The Moran model while simple in its assumptions of the population it describes, provides within it many interesting questions. This simplicity also allows the creation of many extension questions of the model to provide new and unexpected results on the Model. One such unexpected result came from the final anecdote that only Neutral Moran models of certain initial values follow the Geometric model in simulation, unfortunately we could not gather a closed form solution nor educated guess for why this is happening.

References

[1] Lecture notes.
[2] P. Moran: Random processes in genetics, Mathematical Proceedings of the Cambridge Philosophical Society, 1958.
[3] S. Ross: A First Course in Probability, Pearson, 8th edition, 2010.

Python Code

Moranmodels

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 25 14:26:36 2020

@author: s1607860
"""


import math, random as rnd, numpy as np,
    matplotlib.pyplot as plt, matplotlib.patches as mpatches


def moran(i,total, fitness=1):
    path = [i]
    while 0<i and i<total:
        p=(fitness*i*(total-i))/(total*(i*fitness+total-i))
        q=p/fitness
        change=np.random.choice([i+1,i+0,i-1],p=[p,1-p-q,q])
        i=change
        path.append(change)
    return path


def multimoran(i,total,runs,fitness=1):
    All=[]
    for j in range(0,runs):
        run=moran(i,total,fitness)
        All.append(run)
    return All

def Harmonic(N):
    l=sum(1/i for i in range(1,N+1))
    return l

def approxmeanabsorptiontime(i,total):
    if 1000 < total:
        t = (total**2)*((1-(i/total))*math.log(1/(1-(i/total)))+
        (i/total)*math.log(total/i))
    else:
        t = total*((total-i)*(Harmonic(total-1)-Harmonic(total-i-1))+
        i*(Harmonic(total-1)-Harmonic(i)))
    return t
```

MoranmodelsContinued

```
def meanabsorptiontime(i, total, runs, fitness=1, plot=0, save =0):
    data=multimoran(i,total,runs,fitness)
    holder=[]
    for k in range(0,runs):
        A=len(data[k])
        holder.append(A)
    M=sum(holder)/runs
    est=approxmeanabsorptiontime(i,total)
    s=np.random.geometric(1/approxmeanabsorptiontime(i,total),runs)
    var=np.var(holder)
    if plot==1:
        holder.sort()
        s.sort()
        Mlist=[M]*runs
        estlist=[est]*runs
        fig,ax=plt.subplots()
        ax.plot(range(runs),s,'yo', range(0,runs), holder, ...
            'b.',range(0,runs), Mlist, 'r', range(0,runs), estlist, 'g')
        ax.set(xlabel='Ordered Run Number', ylabel='Absorption Time', ...
            title='Simulated vs Expected Mean Absorption Time')
        red_patch = mpatches.Patch(color='red', label='True Mean Run ...
            Absorption Time')
        green_patch = mpatches.Patch(color='green', label='Expected ...
            Mean Run Absorption Time')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Absorption Time')
        yellow_patch = mpatches.Patch(color='yellow', label='Geometric ...
            Distriribution on Mean Aborption Time ')
        plt.legend(handles=[red_patch,blue_patch,green_patch,yellow_patch])
        fig1,ax1=plt.subplots()
        ax1.hist(holder,50,alpha=0.5)
        ax1.hist(s,50,alpha=0.5)
        ax1.set(xlabel='Absorption Time', ylabel='Frequency', ...
            title='Histogram of Mean Absorption Times')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Absorption Time')
        plt.legend(handles=[blue_patch])
        if save==1:
            fig.savefig('Plot_Q2'+str(i)+'_'+str(runs)+'.png')
            fig1.savefig('Hist_Q2_'+str(i)+'_'+str(runs)+'.png')
        else:
            pass
    else:
        ax=1
        ax1=1
    return M,est,ax,ax1,var

a,b,c,d,e=meanabsorptiontime(2,10,100,plot=1,save=0)
print("\nThe true absorption mean of the runs is:", a ,"\nThe expected ...
    absorption mean of the runs was:", b)
```

MoranmodelsContinued

```
def approxmeanextinctiontime(i,total):
    if 1000 < total:
        t = (total**2)*(1-(i/total))/(i/total)*math.log(1/(1-(i/total)))
    else:
        t = ((total**2)*((total-i)/i)*(Harmonic(total-1)-
        Harmonic(total-i-1)))-total
    return t

def meanextinctiontime(i,total,runs,fitness=1,plot=0,save=0):
    data=multimoran(i,total,runs,fitness)
    holder=[]
    count=0
    for k in range(0,runs):
        if (0 in data[k]):
            A=len(data[k])
            holder.append(A)
            count=count+1
        else:
            pass
    M=sum(holder)/count
    est=approxmeanextinctiontime(total-i,total)
    if plot==1:
        holder.sort()
        Mlist=[M]*count
        estlist=[est]*count
        fig,ax=plt.subplots()
        ax.plot(range(0,count), holder, 'b.',range(0,count), Mlist, ...
            'r', range(0,count), estlist, 'g')
        ax.set(xlabel='Ordered Run Number', ylabel='Extinction Time', ...
            title='Simulated vs Expected Mean Extinction Time')
        red_patch = mpatches.Patch(color='red', label='True Mean Run ...
            Extinction Time')
        green_patch = mpatches.Patch(color='green', label='Expected ...
            Mean Run Extinction Time')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Extinction Time')
        plt.legend(handles=[red_patch,blue_patch,green_patch])
        fig1,ax1=plt.subplots()
        ax1.hist(holder,50)
        ax1.set(xlabel='Extinction Time', ylabel='Frequency', ...
            title='Histogram of Mean Extinction Times')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Extinction Time')
        plt.legend(handles=[blue_patch])
        if save==1:
            fig.savefig('Q3_Extin_Plot'+str(i)+'_'+str(runs)+'.png')
            fig1.savefig('Q3_Extin_Hist'+str(i)+'_'+str(runs)+'.png')
        else:
            pass
    else:
        ax=1
        ax1=1
    return M,est,ax,ax1
```

MoranmodelsContinued

```

#a,b,c,d=meanextinctiontime(1,10,10000,fitness=1.42857,plot=1,save=0)
#print("\nThe true extinction mean of the runs is:", a ,"\nThe expected ...
    extinction mean of the runs was:", b)

def approxmeanfixationtime(i,total):
    if 1000 < total:
        t = (total**2)*(1-(i/total))/(i/total)*math.log(1/(1-(i/total)))
    else:
        t = ((total**2)*((total-i)/i)*(Harmonic(total-1)-
        Harmonic(total-i-1)))-total
    return t

def meanfixationtime(i,total,runs,fitness=1,plot=0,save=0):
    data=multimoran(i,total,runs,fitness)
    holder=[]
    count=0
    for k in range(0,runs):
        if (total in data[k]):
            A=len(data[k])
            holder.append(A)
            count=count+1
        else:
            pass
    M=sum(holder)/count
    est=approxmeanfixationtime(i,total)
    if plot==1:
        holder.sort()
        Mlist=[M]*count
        estlist=[est]*count
        fig,ax=plt.subplots()
        ax.plot(range(0,count), holder, 'b.',range(0,count), Mlist, ...
            'r', range(0,count), estlist, 'g')
        ax.set(xlabel='Ordered Run Number', ylabel='Fixation Time', ...
            title='Simulated vs Expected Mean Fixation Time')
        red_patch = mpatches.Patch(color='red', label='True Mean Run ...
            Fixation Time')
        green_patch = mpatches.Patch(color='green', label='Expected ...
            Mean Run Fixation Time')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Fixation Time')
        plt.legend(handles=[red_patch,blue_patch,green_patch])
        fig1,ax1=plt.subplots()
        ax1.hist(holder,50)
        ax1.set(xlabel='Fixation Time', ylabel='Frequency', ...
            title='Histogram of Mean Fixation Times')
        blue_patch = mpatches.Patch(color='blue', label='Individual Run ...
            Fixation Time')
        plt.legend(handles=[blue_patch])
```

MoranmodelsContinued

```
        if save==1:
            fig.savefig('Q3_Fixa_Plot'+str(i)+'_'+str(runs)+'.png')
            fig1.savefig('Q3_Fixa_Hist'+str(i)+'_'+str(runs)+'.png')
        else:
            pass
    else:
        ax=1
        ax1=1
    return M,est,ax,ax1

#a,b,c,d=meanfixationtime(9,10,10000,fitness=0.7,plot=1,save=0)
#print("\nThe true fixation mean of the runs is:", a ,"\nThe expected ...
    fixation mean of the runs was:", b)

def meanbothtime(i,total,runs,fitness=1):
    data=multimoran(i,total,runs,fitness)
    holder=[]
    holder1=[]
    count=0
    count1=0
    for k in range(0,runs):
        if (0 in data[k]):
            A=len(data[k])
            holder.append(A)
            count=count+1
        elif(total in data[k]):
            B=len(data[k])
            holder1.append(B)
            count1=count1+1
        else:
            pass
    M=sum(holder)/count
    N=sum(holder1)/count1
    Guess=(1-((1/(fitness**i))))/(1-((1/(fitness**total))))
    actual=len(holder1)/runs
    return i,total,fitness,runs,len(holder),len(holder1),Guess, actual

#a,b,c,g,k,d,e,f=meanbothtime(7,10,10000,fitness=0.7)
#print("\nFor a Moran with", a,"intials on a poulation size",b ,"and a ...
    fitness of", c,"We get after",g,"runs: \n The number of extinctions ...
    is:", k ,"\n The number of fixations is:", d, "\n The estimated ...
    probability of fixation was:", e, "\n The simulated probability of ...
    fixation was:",f)
```