# Assignment2

February 18, 2024

## 1 Assignment 2: Solutions

[19]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.stats import poisson
from scipy.stats import binom
from scipy.stats import nbinom
from scipy.optimize import fsolve
```

### 1.1 1. Modelling Population Dynamics in an Ecological System

**In population ecology, Markov chains are commonly used to model changes in the composition of populations over time. Consider a simplified ecological system with two competing species, Species A and Species B. Consider a markov chain modelling which species occupies the ecological survey site in a given year. The transition probabilities for the site are given by the transition probability matrix:**

$$P = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix}$$

**This matrix represents the transition probabilities for a one-year time step. For example, P[1,1] = 0.8 means that there is an 80% that the survey site will be dominated by individuals from Species A in two consecutive years. Given that the site currently is occupied by species A:**

**Part A: [3pt] Calculate the probability that the population will consist of Species A after one year. How about two years?**

The site is currently occupied by species A, this gives us the initial condition:

$$V(0) = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

[50]:
```
V0=np.array([1,0])
Pmtrx=np.array([[0.8,0.2],[0.4,0.6]])
```

(2pt) To calculate the probability that the site is dominated by species A the following year we have:

$$V(1) = V0.\mathbf{P}$$

[51]:
```
V0@ Pmtrx
```

[51]:
```
array([0.8, 0.2])
```

This tells us that **there is an 80% that the site remains dominated by Species A after 1 year.**

(1pt) To calculate the two-step transition probabilities we have to calculate: $\mathbf{P}^2$

[52]:
```
P2=Pmtrx @ Pmtrx
```

*Check:* The rows of P2 should still add to 1

[53]:
```
P2
```

[53]:
```
array([[0.72, 0.28],
       [0.56, 0.44]])
```

From this, we can conclude that **in 2 years there is a 28% chance that the site is dominated by species A** and a 28% chance that it is dominated by species B.

**Part B: [2pt] Using the Chapman-Kolmogorov equation, calculate the probability that the population will consist of Species A after three years.**

Recall the Chapman-Kolmogorov equation:

$$p_{ij}^{n+m} = \sum_k p_{ik}^n p_{kj}^m$$

Here we have $i = 0, j = 0$ (note that the index starts from 0) and using our previous calculations we have: $n = 2$ and $m = 1$.

[55]:
```
temp1=P2[0]
print(temp1)
```

```
[0.72 0.28]
```

[56]:
```
temp2=Pmtrx[:,0]
print(temp2)
```

```
[0.8 0.4]
```

[57]:
```
temp1@temp2
```

[57]:
```
0.6880000000000002
```

*Check:* We can check our answer by comparing the result to element 0 of:

$$V(3) = V(0).\mathbf{P}^3$$

[58]:
```
P3=Pmtrx @ Pmtrx @Pmtrx
V3=V0@P3
print(V3[0])
```

```
0.6880000000000002
```

**Part C:[4pt] What is the stationary distribution of this system?**

There are several ways of approaching this problem. Given that the matrix is small, let's try solving for the stationary distribution as a system of equations. Recall:

$$\vec{\pi} = \vec{\pi}.\mathbf{P}$$

There is also an additional equation that $\pi_1 + \pi_2 = 1$. Since our matrix is a 2x2 this means that we get 1 equation from the matrix and then the total probability equation. Specifically, from the first column of the matrix (recall the $\pi$ is a row vector) we have:

$$\pi_1 = 0.8\pi_1 + 0.4\pi_2$$

Rearranging we have:

$$\pi_1 = \frac{0.4}{0.2}\pi_2 = 2\pi_2$$

From the total probability equation we have:

$$2\pi_2 + \pi_2 = 1$$

Or $\pi_2 = \frac{1}{3}$ and $\pi_1 = \frac{2}{3}$.

[59]:
```
Pmtrx[:,0]
```

[59]:
```
array([0.8, 0.4])
```

*Check:* Let's double check with the eigenvector approach.

[60]:
```
eigenvalues, eigenvectors = np.linalg.eig(Pmtrx)
print(eigenvalues)
```

```
[1.  0.4]
```

So the first eigenvector is the leading eigenvector. So we want the first row of $A^{-1}$.

[61]:
```
rightEVs=np.linalg.inv(eigenvectors);
rightEVs[0]
```

Normalizing.

[63]:
```
rightEVs[0]/np.sum(rightEVs[0])
```

[63]:
```
array([0.66666667, 0.33333333])
```

**Part D: [1pt] Interpret your answers in terms of ecosystem stability.**

There are two ways of thinking about ecosystem stability.

First, we could think of an ecosystem as stable if it quickly reaches its "equilibrium" state. From parts A and B, we see that if we start in state 1 we are still in state 1 80% of the time in year 1, 72% of the time in year 2 and 68% of the time in year 3. The long-term probability of being in state 1 is 67% which means that we effectively reach this equilibrium state within 3 years. Under this metric, this ecosystem is quite stable.

Or we could think of ecosystem stability as a system that is likely to stay in the same state. In this case, we would want the diagonal elements of the matrix to be large and for one state to be nearly absorbing. Under this metric, this ecosystem is highly unstable.

## 1.2 2. Modelling Species Occupancy in an Ecological Network

**In the field of ecology, species occupancy in a network of interconnected habitats is a critical aspect to understand. Consider an ecological network consisting of three interconnected habitats: Habitat A, Habitat B, and Habitat C where $A \leftrightarrow B \leftrightarrow C \leftrightarrow A$. Ecologists want to model the occupancy dynamics of a particular species, "Species X," across these habitats over time using a discrete time discrete state stochastic process.**

**State Space:**

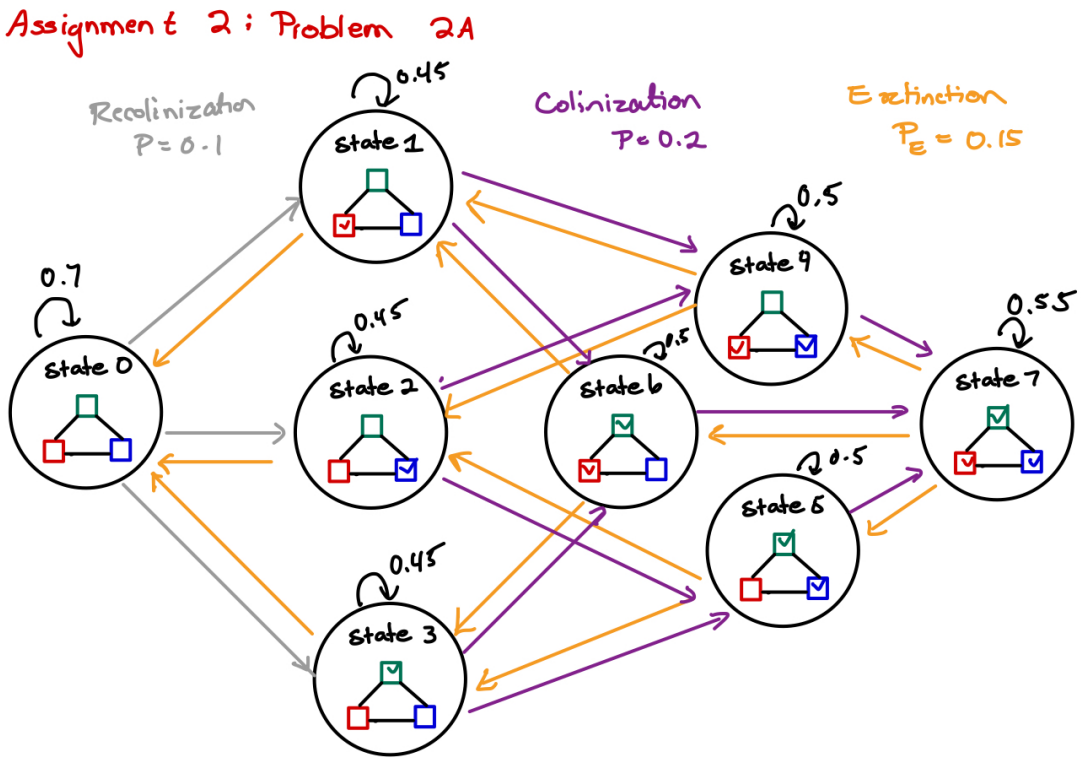
- **State 0: Species X is absent from all habitats.**
- **State 1: Species X is present in Habitat A.**
- **State 2: Species X is present in Habitat B.**
- **State 3: Species X is present in Habitat C.**
- **State 4: Species X is present in Habitats A&B.**
- **State 5: Species X is present in Habitats A&C.**
- **State 6: Species X is present in Habitats B&C.**
- **State 7: Species X is present in All Habitats.**

**Transition Probabilities:**

- **Rule 1: The probability of colonizing an adjacent habitat is 0.2.**
- **Rule 2: The probability of extinction from a habitat is 0.15.**
- **Rule 3: The probability of recolonization of a random habitat is 0.1**

**Part A. [4pt] Draw a schematic diagram of the Markov Process illustrating states and transitions. Define the transition rate matrix, $P$, for this discrete time discrete state stochastic process.**

Grading: 2pts for diagram, 2pts for matrix.




$$\mathbf{P} = \begin{bmatrix} 0.7 & 0.1 & 0.1 & 0.1 & 0 & 0 & 0 & 0 \\ 0.15 & 0.45 & 0 & 0 & 0.2 & 0 & 0.2 & 0 \\ 0.15 & 0 & 0.45 & 0 & 0.2 & 0.2 & 0 & 0 \\ 0.15 & 0 & 0 & 0.45 & 0 & 0.2 & 0.2 & 0 \\ 0 & 0.15 & 0.15 & 0 & 0.5 & 0 & 0 & 0.2 \\ 0 & 0 & 0.15 & 0.15 & 0 & 0.5 & 0 & 0.2 \\ 0 & 0.15 & 0 & 0.15 & 0 & 0 & 0.5 & 0.2 \\ 0 & 0 & 0 & 0 & 0.15 & 0.15 & 0.15 & 0.55 \end{bmatrix}$$

[2]:
```
PMtrx=np.array([[0.7,0.1,0.1,0.1,0,0,0,0],
   [0.15,0.45,0,0,0.2,0,0.2,0],
   [0.15,0,0.45,0,0.2,0.2,0,0],
  [0.15,0,0,0.45,0,0.2,0.2,0],
  [0,0.15,0.15,0,0.5,0,0,0.2],
  [0,0,0.15,0.15,0,0.5,0,0.2],
  [0,0.15,0,0.15,0,0,0.5,0.2],
  [0,0,0,0,0.15,0.15,0.15,0.55]])

np.sum(PMtrx, axis=1)
```

[2]:
```
array([1., 1., 1., 1., 1., 1., 1., 1.])
```

**Part B. [2pt] Calculate the stationary distribution,$\pi$, for this process. Interpret the stationary distribution in the context of Species X's long-term occupancy in the ecological network.**

[66]:
```
eigenvalues, eigenvectors = np.linalg.eig(PMtrx)
print(eigenvalues)
```

```
[0.01139991 0.43860009 1.         0.75       0.3        0.65
 0.3        0.65       ]
```

So the leading eigenvalues is the third (index 2) eigenvalue.

[67]:
```
rightEVs=np.linalg.inv(eigenvectors);
print(rightEVs[2])
```

[69]:
```
#Checking that we have the right one:
rightEVs[2]@PMtrx
```

[69]:
```
array([0.41279747, 0.27519831, 0.27519831, 0.27519831, 0.36693109,
       0.36693109, 0.36693109, 0.48924145])
```

[70]:
```
#Normalize to obtain stationary distribution
rightEVs[2]/np.sum(rightEVs[2])
```

[70]:
```
array([0.14594595, 0.0972973 , 0.0972973 , 0.0972973 , 0.12972973,
       0.12972973, 0.12972973, 0.17297297])
```

Interpretation: 14% of the time the species is absent, 30% of the time it occupies a single habitat, 36% of the time it occupies 2 habitats simultaniously, and 17% of the time it occupies all three.

**Part C. [1pt] Characterize the states of the system as *transient*, *absorbing* or *recurrent*.**

All the states are recurrent.

**Part D. [3pt] Suppose that species $X$ presents an important ecosystem service. If as a conservation biologist, you can either support recolonization (upping the probability to 0.2) or reduce extinction probability to 0.1 what would be a better conservation strategy? Justify your answer.**

Grading: 2pts for math, 1pt for interpretation.

Let's look at the stationary distributions for these two alternatives

[71]:
```
PMtrxExt=np.array([[0.7,0.1,0.1,0.1,0,0,0,0],
   [0.1,0.5,0,0,0.2,0,0.2,0],
   [0.1,0,0.5,0,0.2,0.2,0,0],
  [0.1,0,0,0.5,0,0.2,0.2,0],
  [0,0.15,0.15,0,0.5,0,0,0.2],
  [0,0,0.15,0.15,0,0.5,0,0.2],
```

```
  [0,0.15,0,0.15,0,0,0.5,0.2],
  [0,0,0,0,0.15,0.15,0.15,0.55]])

PMtrxRe=np.array([[0.4,0.2,0.2,0.2,0,0,0,0],
   [0.15,0.45,0,0,0.2,0,0.2,0],
   [0.15,0,0.45,0,0.2,0.2,0,0],
  [0.15,0,0,0.45,0,0.2,0.2,0],
  [0,0.15,0.15,0,0.5,0,0,0.2],
  [0,0,0.15,0.15,0,0.5,0,0.2],
  [0,0.15,0,0.15,0,0,0.5,0.2],
  [0,0,0,0,0.15,0.15,0.15,0.55]])
```

[73]:
```
eigenvalues1, eigenvectors1 = np.linalg.eig(PMtrxExt)
print(eigenvalues1)

eigenvalues2, eigenvectors2 = np.linalg.eig(PMtrxRe)
print(eigenvalues2)
```

```
[0.0371488  0.47297908 1.         0.73987212 0.67320508 0.67320508
 0.32679492 0.32679492]
```

[74]:
```
rightEVs1=np.linalg.inv(eigenvectors1);
rightEVs1[2]/np.sum(rightEVs1[2])

rightEVs2=np.linalg.inv(eigenvectors2);
rightEVs2[2]/np.sum(rightEVs2[2])
```

[74]:
```
array([0.10227273, 0.10227273, 0.10227273, 0.10227273, 0.13636364,
       0.13636364, 0.13636364, 0.18181818])
```

As expected, both models reduce the probablilty of extinction from 14% to 10% (Reduced Extinction) and 7% (Increased Recolinization) respectively. The recolinization scenario leads to both a lower extinction probability and a higher probability that the species is present in all three habitats.

### 1.3 3. Genetic Drift

**In this lecture we modelled genetic drift in two ways, using the Wright-Fisher model and using the Moran model. Assume that we have a population of size $N = 50$ and two alleles 'A' and 'a' such that the initial frequency of the 'A' allele is initially $p_A = 0.5$. Assume that mutation from the $A \to a$ and $a \to A$ occurs with probability $u = 0.001$ per reproduction event.**

**Part A. [2pt] Write down the transition probabilities that describe each model of genetic drift.**

Grading: 1 for each model

In the Wright-Fisher Model with mutation the probability of going from $i$ to $j$ copies of the mutation in one time step is given by the binomial probaiblity:

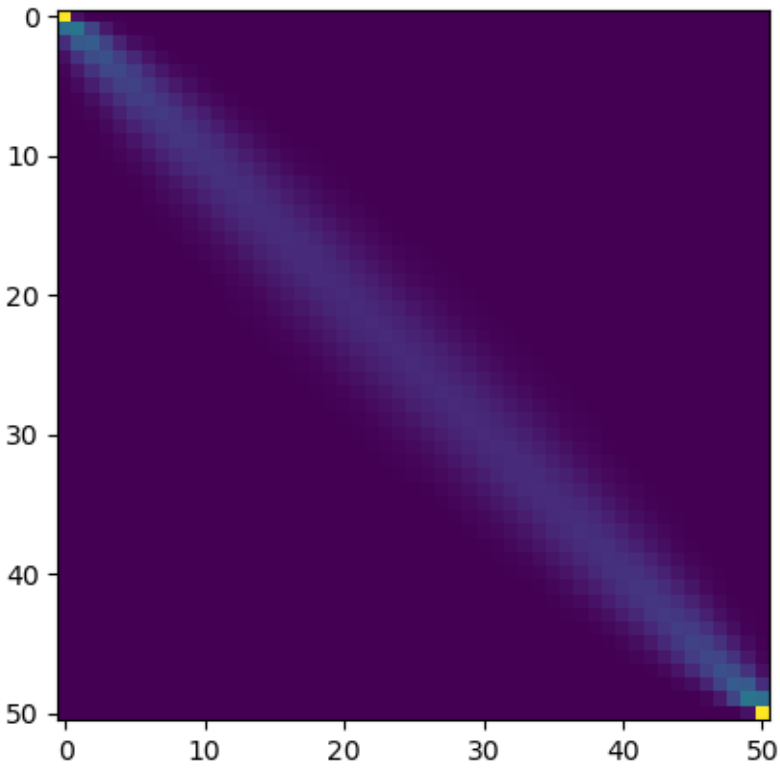
$$p(i,j) = \binom{n}{j} \left((1-u)\frac{i}{N} + u\frac{N-i}{N}\right)^{j} \left((1-u)\frac{N-i}{N} + u\frac{i}{N}\right)^{n-j}$$

[3]:
```
def PMtrxWFMu(N,u):
    P = np.zeros((N+1, N+1)) #Matrix of 0s

    for i in range(N+1):
        for j in range(N+1):
            x=i/N*(1-u)+(1 - i/N)*u
            y=i/N*(u)+(1 - i/N)*(1-u)
            P[i, j] = (np.math.comb(N, j) * (x ** j) * (y ** (N - j)))

    return P
```

[4]:
```
PWF50=PMtrxWFMu(50,0.001)
## Check
print(np.sum(PWF50, axis=1))
plt.imshow(PWF50, cmap='viridis', interpolation='nearest')
plt.show()
```

```
/tmp/ipykernel_774/1833759648.py:8: DeprecationWarning: `np.math` is a
deprecated alias for the standard library `math` module (Deprecated Numpy 1.25).
Replace usages of `np.math` with `math`
  P[i, j] = (np.math.comb(N, j) * (x ** j) * (y ** (N - j)))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1.]
```



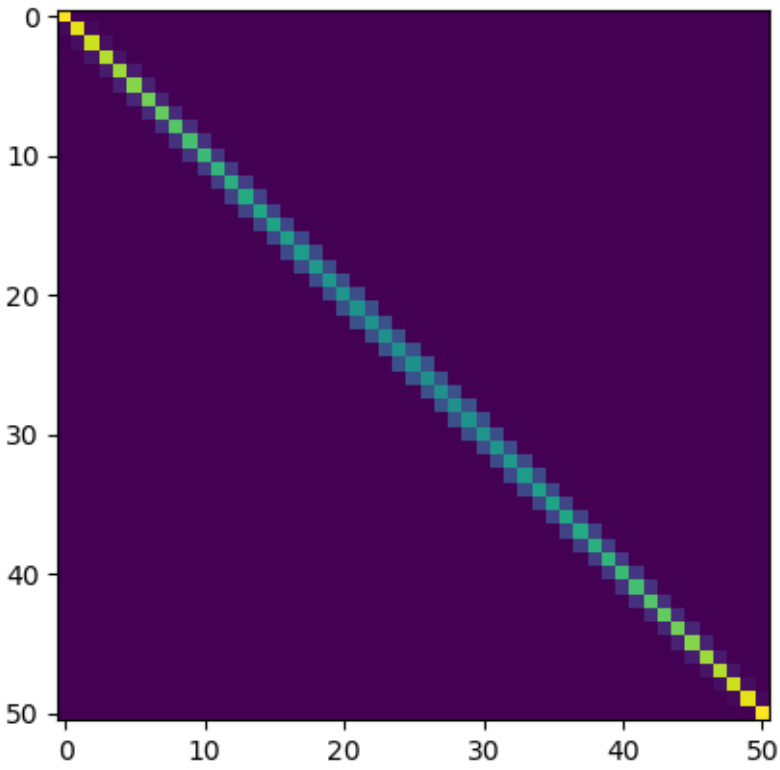

In the Moran Model with mutation we must consider the probability that an indivdiual dies and the probability it is replaced with either a mutant or non-mutant birth.

$$p(i,j) = \begin{cases} \frac{N-i}{N}\left((1-u)\frac{i}{N} + u\frac{N-i}{N}\right) & j = i+1 \\ \frac{i}{N}\left((1-u)\frac{N-i}{N} + u\frac{i}{N}\right) & j = i-1 \\ 1 - \frac{N-i}{N}\left((1-u)\frac{i}{N} + u\frac{N-i}{N}\right) - \frac{i}{N}\left((1-u)\frac{N-i}{N} + u\frac{i}{N}\right) & i = j \\ 0 & \text{otherwise} \end{cases}$$

[5]:
```
def PMtrxMoranMu(N,u):
    P = np.zeros((N+1, N+1))
    for i in range(0,N+1):
        P[i,i]=1;
        if i<N:
            P[i,i+1]= (1-i/N)*(i/N*(1-u)+(1-i/N)*u)
            P[i,i]-=P[i,i+1]
        if i>0:
            P[i,i-1]= i/N*((1-i/N)*(1-u)+(i/N)*u)
            P[i,i]-=P[i,i-1]
    return P
PMoran50=PMtrxMoranMu(50,0.001)
## Check
```

```
print(np.sum(PMoran50, axis=1))

PMoran50=PMtrxMoranMu(50,0.001)
plt.imshow(PMoran50, cmap='viridis', interpolation='nearest')
plt.show()
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1.]
```



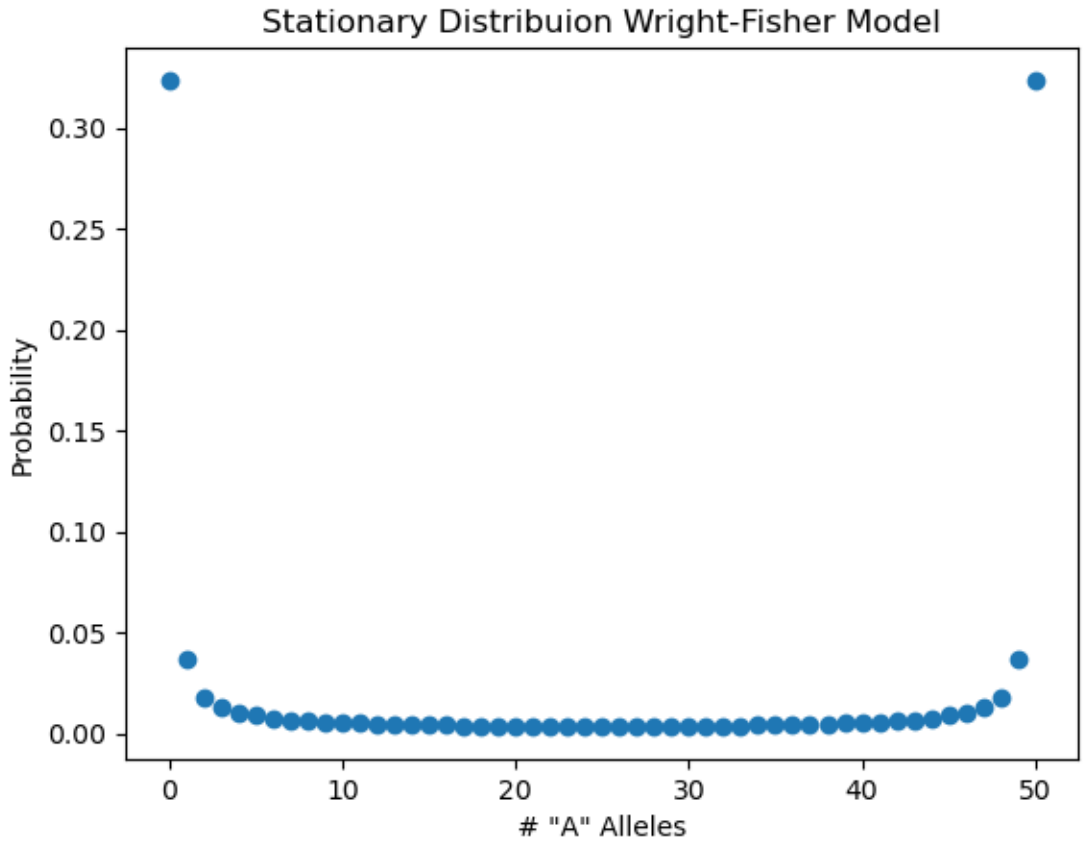

**Part B. [2pt] Calculate the stationary distribution in each model.**

Grading: 1 for each model

[8]:
```
eigenvaluesWF, eigenvectorsWF = np.linalg.eig(PWF50)
print(eigenvaluesWF)
index_of_largest = np.argmax(eigenvaluesWF)
print(index_of_largest)

rightEVsWF=np.linalg.inv(eigenvectorsWF);
SDWF=rightEVsWF[0]/np.sum(rightEVsWF[0])

x_vals=np.linspace(0, 50, 51);
```

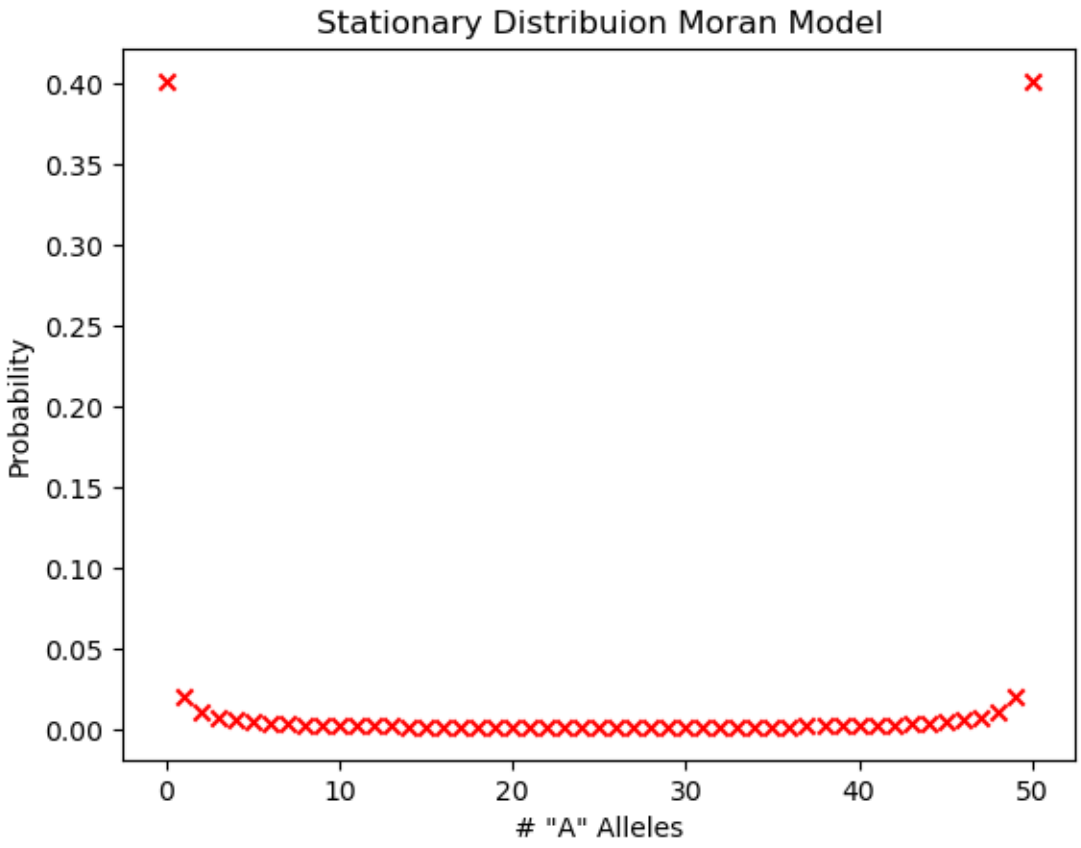
```
plt.scatter(x_vals,SDWF)
plt.xlabel('# "A" Alleles')
plt.ylabel('Probability')
plt.title('Stationary Distribuion Wright-Fisher Model');
plt.show()
```

```
[ 1.00000000e+00  9.98000000e-01  9.76083920e-01  9.35166482e-01
  8.77298380e-01  8.05500281e-01  7.23500352e-01  6.35406949e-01
  5.45357076e-01  4.57183744e-01  3.74140889e-01  2.98714086e-01
  2.32530993e-01  1.76370108e-01  1.30252852e-01  9.35944892e-02
  6.53851101e-02  4.43729512e-02  2.92275755e-02  1.86682370e-02
  1.15511583e-02  6.91683361e-03  4.00373996e-03  2.23761019e-03
  1.20589288e-03  6.25810171e-04  3.12279275e-04  1.49594264e-04
  6.86757348e-05  3.01568887e-05  1.26405614e-05  5.04611213e-06
  1.91368756e-06  6.87549668e-07  2.33299353e-07  7.45064815e-08
  2.23072406e-08  6.23353529e-09  1.61747774e-09  3.87418282e-10
  8.50615544e-11  1.69782724e-11  3.04997557e-12  4.87009870e-13
  6.80412875e-14  8.12708723e-15  8.01198528e-16  6.55118399e-17
  1.00028229e-17 -2.06879827e-17 -7.39212364e-18]
0
```



[7]:
```
eigenvaluesMoran, eigenvectorsMoran= np.linalg.eig(PMoran50)
print(eigenvaluesMoran)

index_of_largest = np.argmax(eigenvaluesMoran)
print(index_of_largest)

eigenvaluesMoran[index_of_largest]

rightEVsMoran=np.linalg.inv(eigenvectorsMoran);
SDMoran=rightEVsMoran[index_of_largest]/np.sum(rightEVsMoran[index_of_largest])

x_vals=np.linspace(0, 50, 51);
plt.scatter(x_vals,SDMoran, color='red', marker='x', label='Moran')
plt.xlabel('# "A" Alleles')
plt.ylabel('Probability')
plt.title('Stationary Distribuion Moran Model');
plt.show()
```

```
[0.01996   0.0591216 0.0974848 0.1350496 0.171816  0.207784  0.2429536
 0.2773248 0.3108976 0.343672  0.375648  0.4068256 0.4372048 0.4667856
 0.495568  0.523552  0.5507376 0.5771248 0.6027136 0.627504  0.651496
 0.6746896 0.6970848 0.7186816 0.73948   0.75948   0.7786816 0.7970848
 0.8146896 0.831496  0.847504  0.8627136 0.8771248 0.8907376 0.903552
 0.915568  0.9267856 0.9372048 0.9468256 0.955648  0.963672  0.9708976
 0.9773248 0.9829536 0.987784  0.991816  0.9950496 0.9974848 0.9991216
 0.99996   1.        ]
50
```

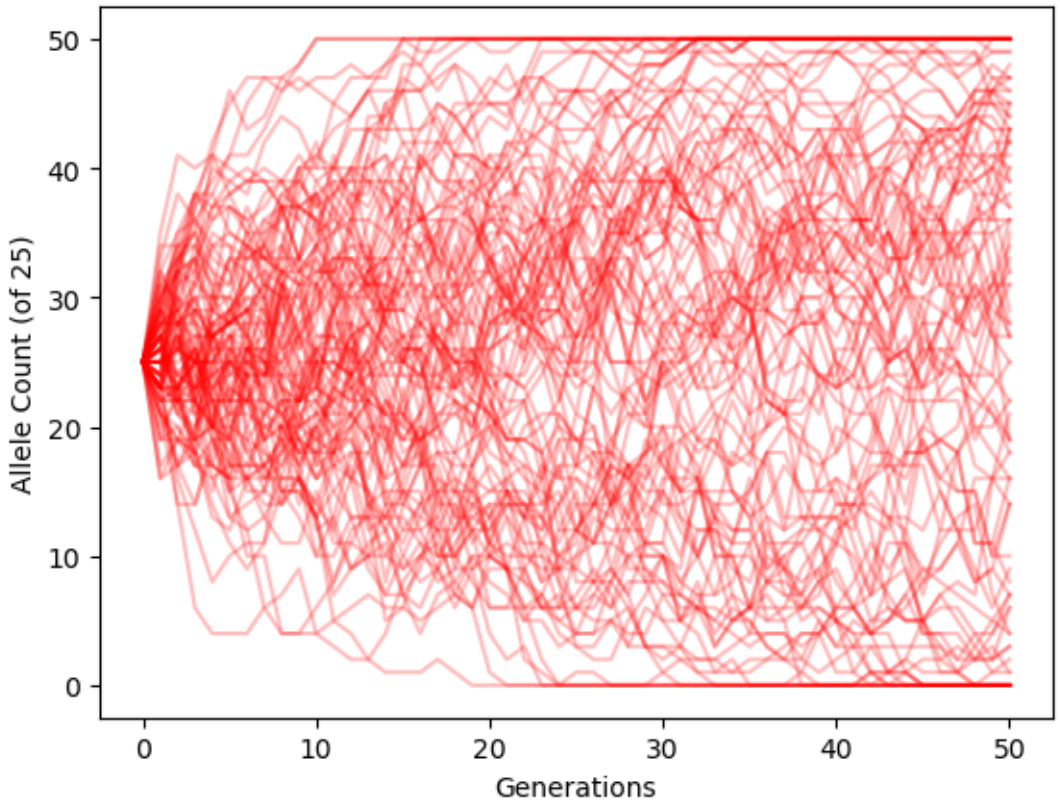



**Part C. [4pt] Simulate the dynamics of the 'A' allele in each model**

Grading: 2pts for each mdoel

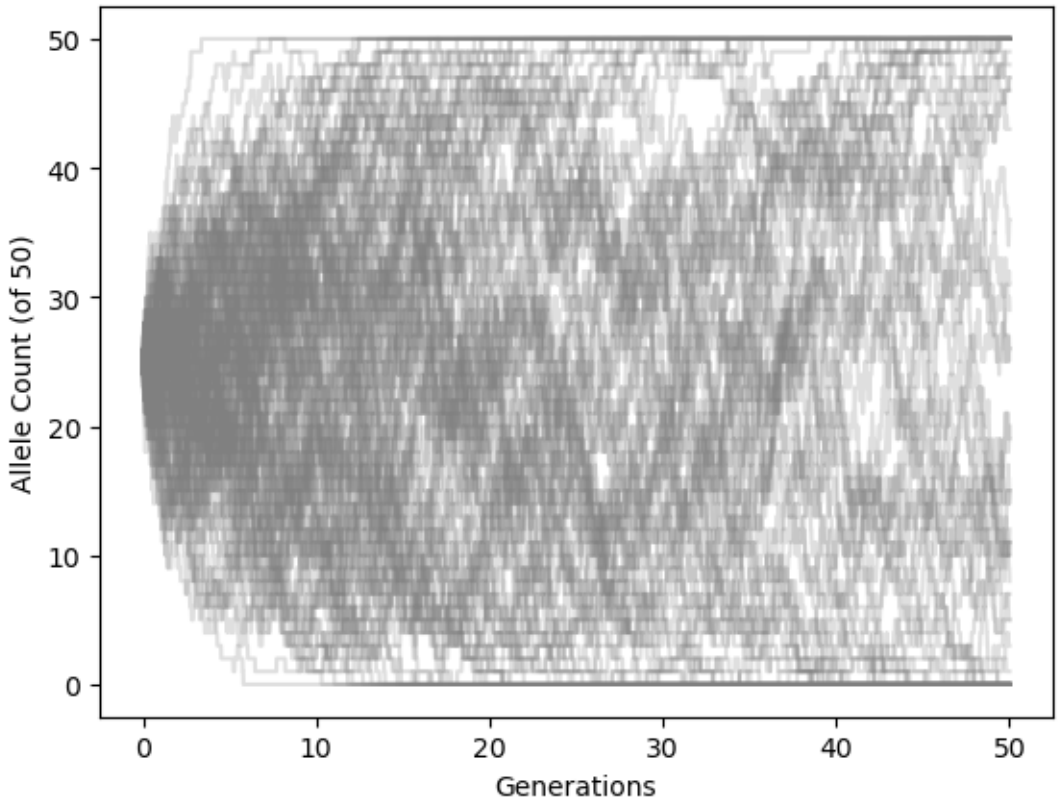
[9]:
```
def eventVec(N):
    return np.arange(N+1)
def randSimWF(tMax,index): #tMax in the number of generations, index is a dummy␣
 ↪variable we will use in a minute
    PMtrx=PWF50;
    #Initialize
    out=[25];
    temp=25;
    #Itterate
    for t in range(tMax):
        temp=np.random.choice(eventVec(50), p=PMtrx[temp])
        out.append(temp)
    return out
# Create a dictionary to store results
WF_dict = {}

# Calculate and save results for specified indices
```

```
for index in range(100):
    WF_dict[index] = randSimWF(50,index) #N=20, tMax=50

for index in range(100):
    plt.plot( WF_dict[index], label='Vectors',color='red',alpha=0.25)
plt.xlabel('Generations')
plt.ylabel('Allele Count (of 25)');
```




[10]:
```
def eventVec(N):
    return np.arange(N+1)
def randSimMoran(tMax,index): #tMax in the number of generations, index is a␣
 ↪dummy variable we will use in a minute
    PMtrx=PMoran50;
    #Initialize
    out=[25];
    temp=25;
    #Itterate
    for t in range(tMax*50):
        temp=np.random.choice(eventVec(50), p=PMtrx[temp])
        out.append(temp)
    return out
```

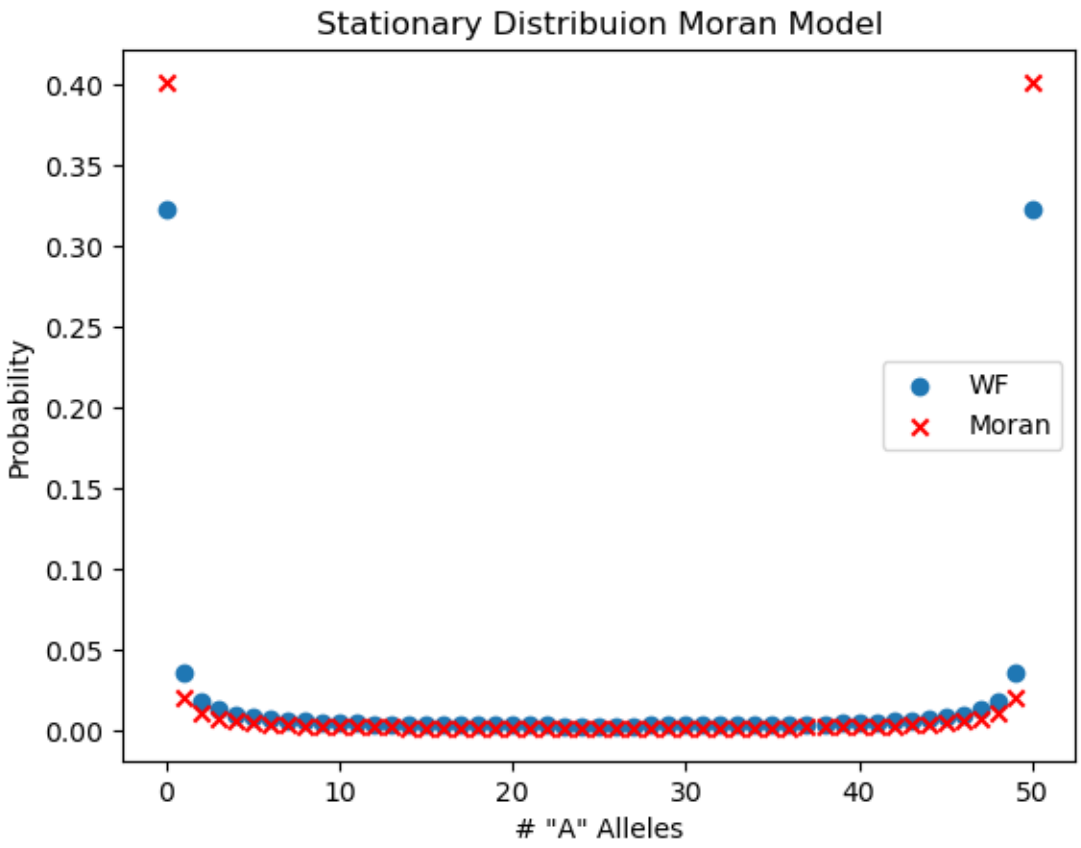
```
# Create a dictionary to store results
Moran_dict = {}

# Calculate and save results for specified indices
for index in range(100):
    Moran_dict[index] = randSimMoran(50,index)  #N=20, tMax=50=50*20 time steps

x_vals=np.linspace(0, 50, 50*50+1);
for index in range(100):
    plt.plot(x_vals,Moran_dict[index], label='Vectors',color='gray',alpha=0.25)
plt.xlabel('Generations')
plt.ylabel('Allele Count (of 50)');
```



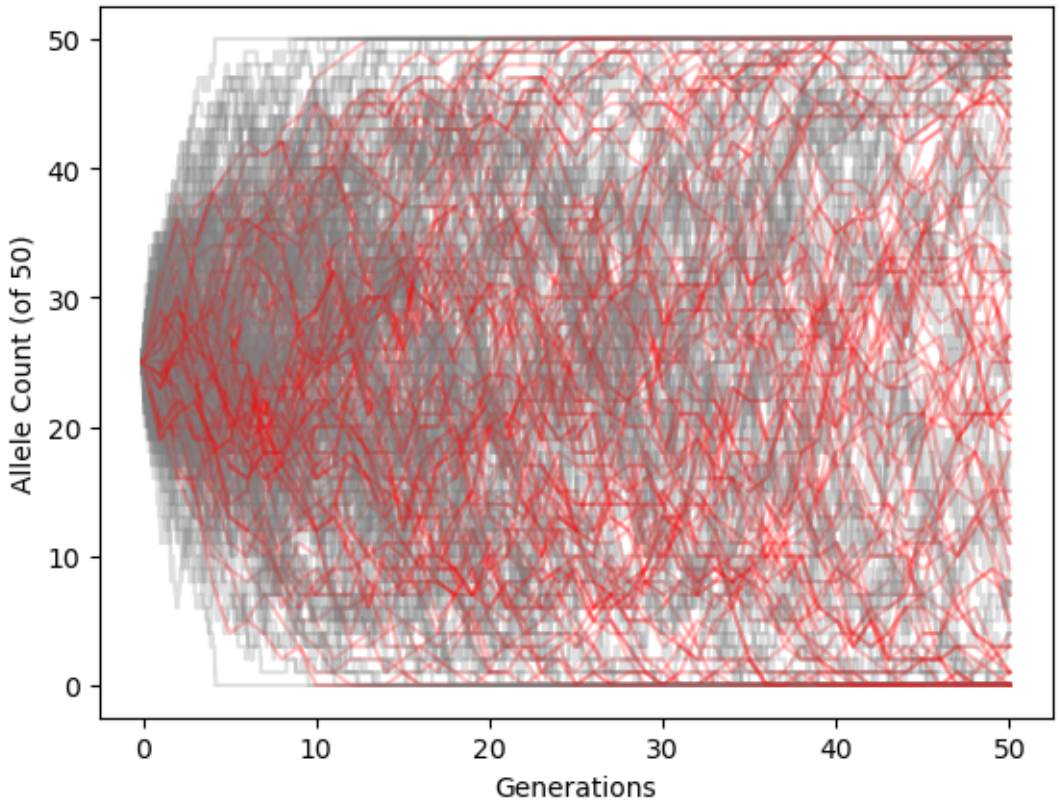

**Part D. [2pts] Use your understanding of both models and your answers to A-C to compare and contrast the dynamics of these two models.**

There are two key differences between the Moran and WF models.
1) Drift occurs twice as fast in the Moran model than in the WF model. We see this in the elevated probability of fixation in the Moran model as given by their stationary distributions. 2) The Moran model is effectively continuous whereas the WF model captures discrete non-overlapping generations. This is reflected in the smoothness of the jumps in the simulated dynamics.

[95]:
```
x_vals=np.linspace(0, 50, 51);
plt.scatter(x_vals,SDWF, label='WF')
plt.scatter(x_vals,SDMoran, color='red', marker='x', label='Moran')
plt.xlabel('# "A" Alleles')
plt.ylabel('Probability')
plt.title('Stationary Distribuion Moran Model');
plt.legend()
plt.show()
```




[96]:
```
x_vals=np.linspace(0, 50, 50*50+1);
for index in range(100):
    plt.plot(x_vals,Moran_dict[index], color='gray',alpha=0.25)
    plt.plot(WF_dict[index], color='red',alpha=0.25)
plt.xlabel('Generations')
plt.ylabel('Allele Count (of 50)');
plt.show()
```




### 1.4 4. Extinction Probabilities in a Branching Process

**In biology, branching processes are often used to model the population growth of species. Extinction probabilities are crucial in understanding the long-term survival of a population. Consider a simple branching process where each individual has a Poisson number of offspring with on average, 2.5 offspring.**

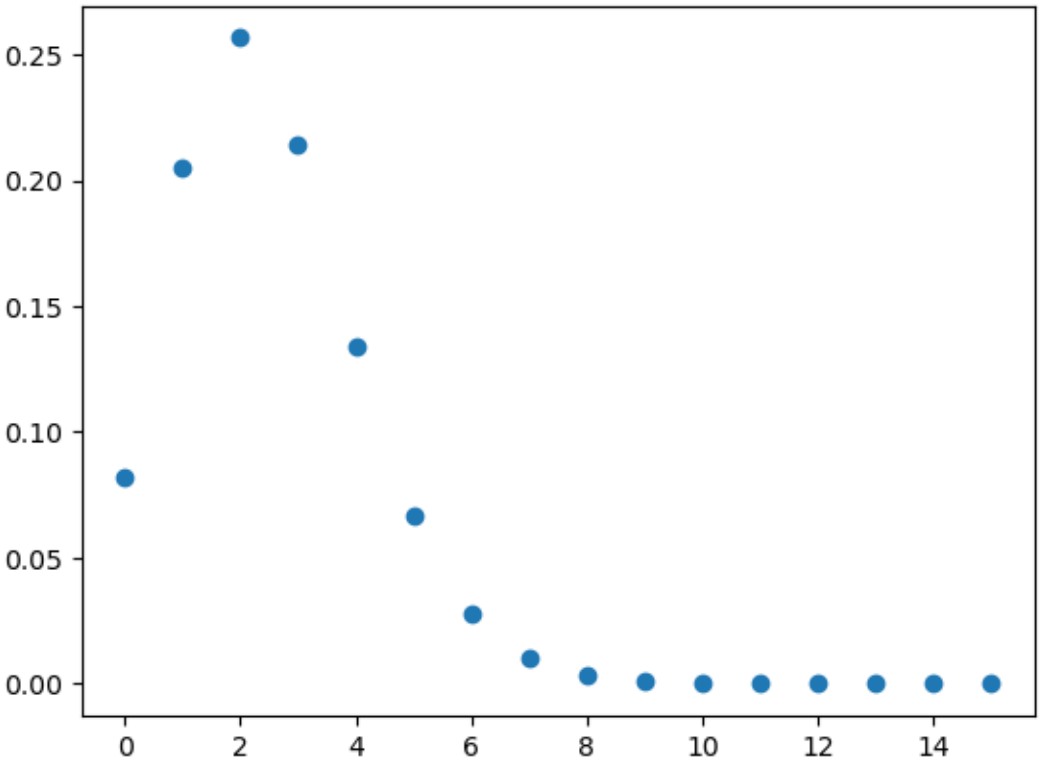
**Part A. [3pts] Suppose that we start with a single individual and calculate the probability that the population goes extinct after one generation. What is the extinction prob. in a single generation if there are n>1 individual?**

Note that the offspring distribution (the probability of having $k$ offspring) is:

$$p_k = \frac{2.5^k e^{-2.5}}{k!}$$

[97]:
```
k_values = np.arange(0, 16, 1)
plt.plot(k_values, poisson.pmf(k_values, 2.5),'o')  # Replace with your function
```

[97]:
```
[<matplotlib.lines.Line2D at 0x7fd04122d880>]
```




(2pts) The probability of a single initial individual is the probability that that individual having 0 offspring is:

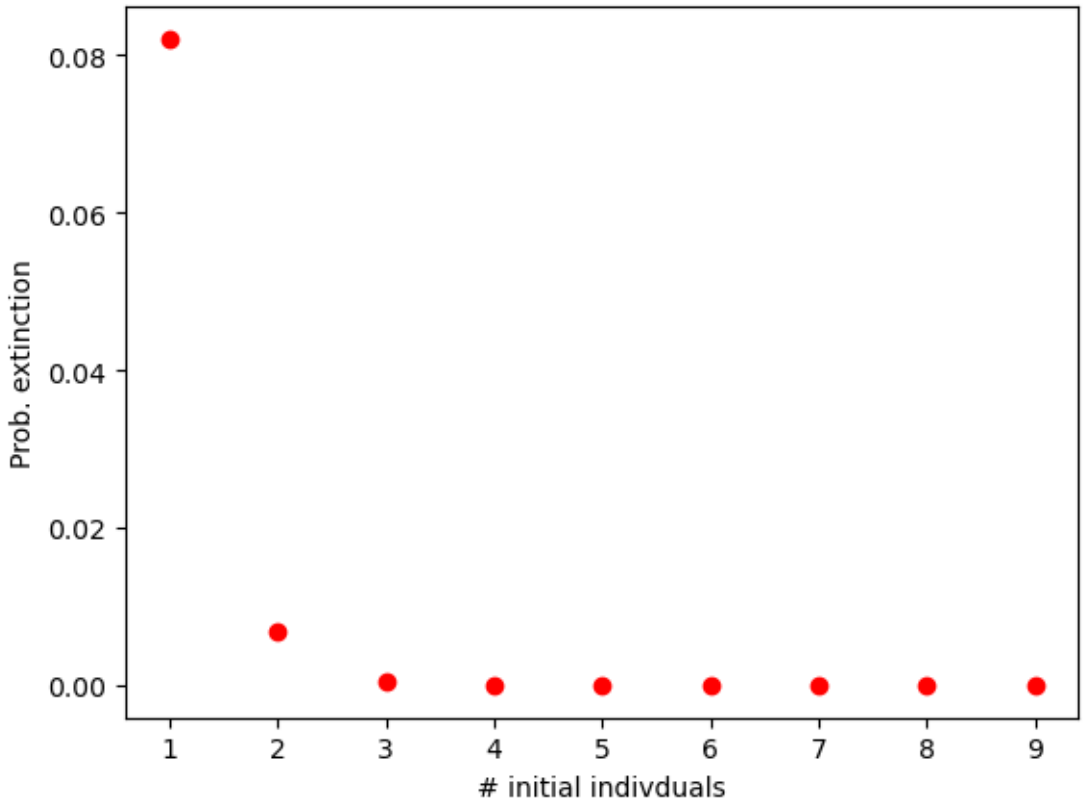
[98]:
```
poisson.pmf(0, 2.5)
```

[98]:
```
0.0820849986238988
```

In words, 8% of the time the population will go extinct immediately.

(1pts) To find the extinction probability with $n$ initial individuals we raise this inital probability to the $n$.

[99]:
```
n_vals=np.arange(1, 10, 1)
prob_ext=poisson.pmf(0, 2.5)**n_vals
plt.plot(n_vals, prob_ext,'o',color='red')
plt.xlabel('# initial indivduals')
plt.ylabel('Prob. extinction')
plt.show()
```



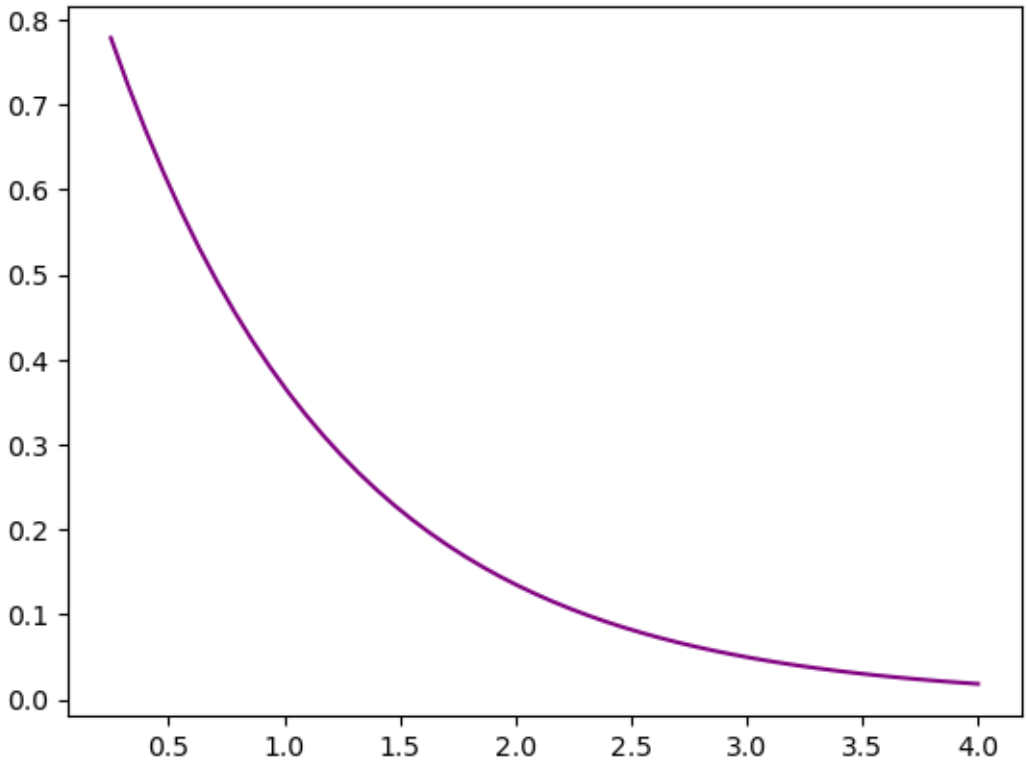

**Part B. (3pts) Let's explore how sensitive your answer is to the number of individuals per parent. What is the probability of extinction in 1 generation from 1 initial individual if the mean number of offspring is $\lambda = [0.5, 4]$? Calculate the extinction probability in *exactly* two generations given that there is a single initial individual across various values of $\lambda$. What is the probability that the population goes extinct in the long term across various values of $\lambda$?**

**Note you can work this problem out either numerically or analytically if you choose.**

(1pts) Let $\lambda$ be the expected number of offspring.

[14]:
```
def g1(lam):
    return np.exp(-lam)

x_vals=np.linspace(0.25, 4, 50);
g1_vals=g1(x_vals)
plt.plot(x_vals,g1_vals,color='purple')
```

[14]: `[<matplotlib.lines.Line2D at 0x7ff3e5e93010>]`




(2pts) To calculate the probability that the population goes extinct in exactly two generations we consider the probability that the initial individual has $k$ offspring and the probability that all $k$ of those offspring have 0 offspring.

$$g_2 = \sum_{k=1}^{\infty} p_k p_0^k$$

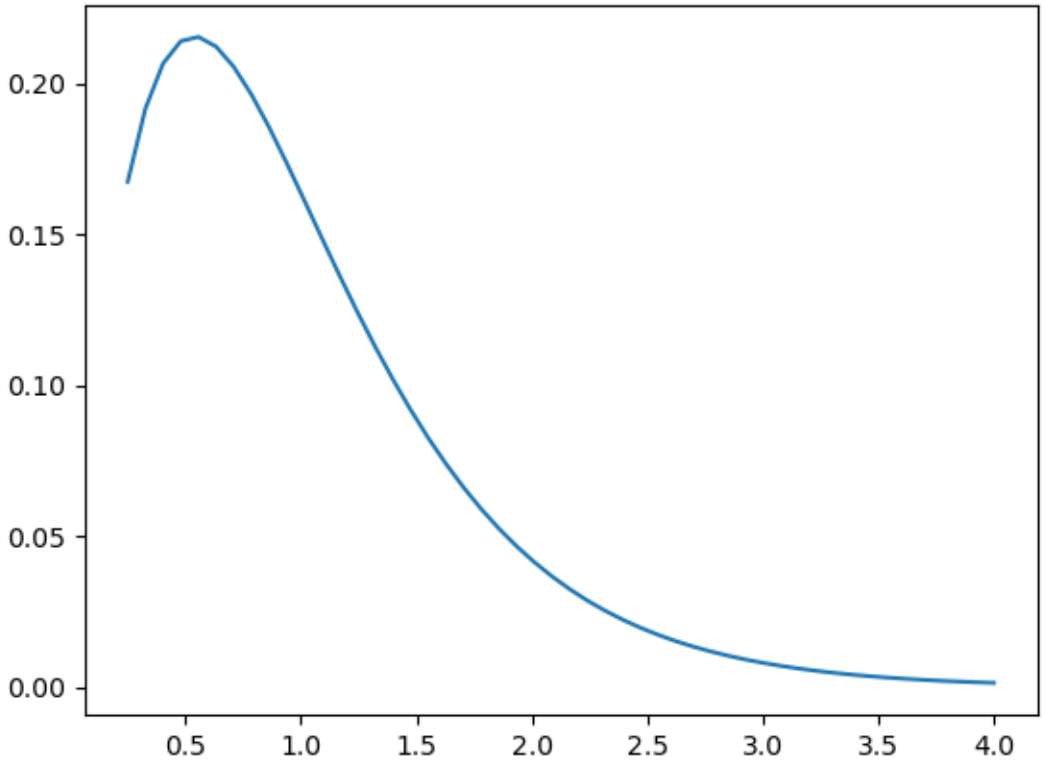
Let $\lambda$ be the expected number of offspring.

$$g_2 = \sum_{k=1} \left( \frac{\lambda^k e^{-\lambda}}{k!} \right) \left( \frac{\lambda^0 e^{-\lambda}}{0!} \right)^k$$

$$g_2 = e^{-\lambda} \left( e^{e^{-\lambda}\lambda} - 1 \right)$$

[12]:
```
def g2(lam):
    return np.exp(-lam)*(np.exp(np.exp(-lam)*lam)-1)

x_vals=np.linspace(0.25, 4, 50);
g2_vals=g2(x_vals)
plt.plot(x_vals,g2_vals)
```

[12]: [<matplotlib.lines.Line2D at 0x7ff3e650c410>]




We see that by in large the probability of extinction decreases with increasing mean number of offspring. Note that for very few offspring per parent the probability actually decreases because the popualtion usually goes extinct in the first generaiton.

The probability of ultimate extinction must satisfy:

$$g = \sum_{k=0}^{\infty} p_k g^k$$

For the Poisson distribution this simplfies to:

$$g = e^{(g-1)\lambda}$$

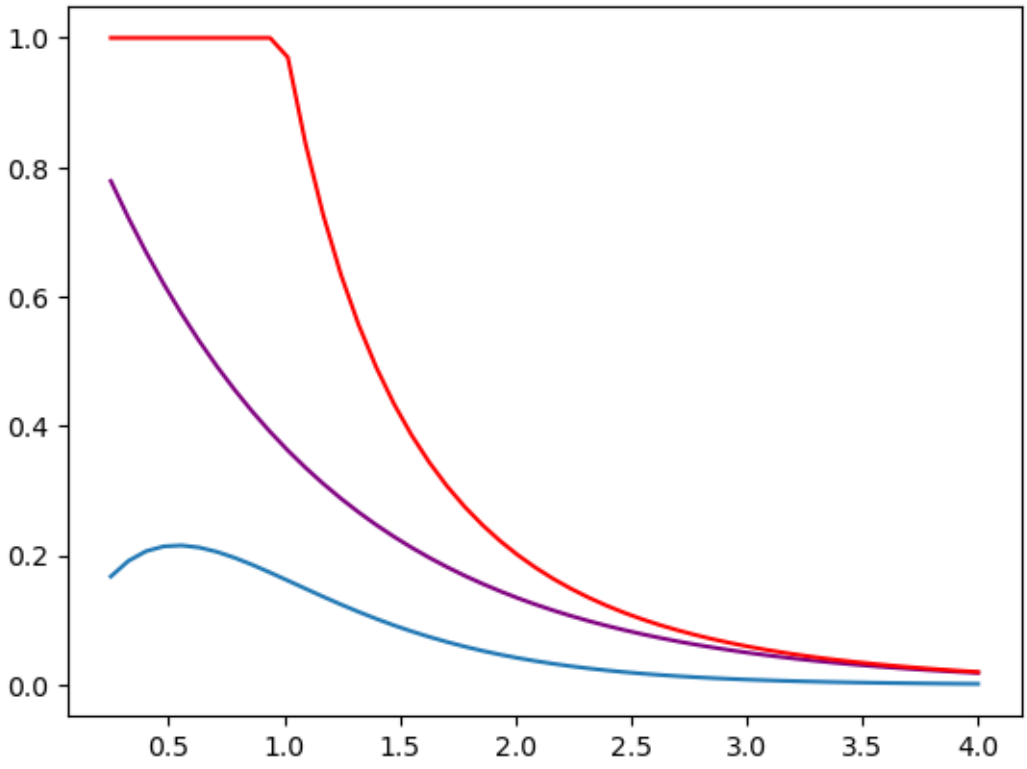
Solving we have:

$$g = -\frac{W\left(-e^{-\lambda}\lambda\right)}{\lambda}$$

Where $W$ is the product log or Lambert W function.

[ ]:
```
from scipy.special import lambertw
def g(lam):
    return lambertw(-np.exp(-lam)*lam)/-lam

x_vals=np.linspace(0.25, 4, 50);
g_vals=g(x_vals)
g2_vals=g2(x_vals)
g1_vals=g1(x_vals)
plt.plot(x_vals,g1_vals,color='purple')
plt.plot(x_vals,g2_vals)
plt.plot(x_vals,y_vals,'red')
plt.xlabel('Mean # of offspring')
plt.ylabel('Probability of Extinction')
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f4913224ed0>]
```




**Part C. (3pts) The distribution of offspring per parent in human populations is not Poisson but rather negative binomial. The negative binomial distribution has two parameters $0 < r$ and $0 < p < 1$ with a mean of $\frac{r(1-p)}{p}$ and variance $\frac{r(1-p)}{p^2}$ (hence the variance is greater than the mean whereas in the Poisson distribution the variance equals the mean). Setting $p = 0.5$ and $r = 2.5$ (such that the mean is the same as in part A) and starting with a single individual, what is the probability of extinction in**

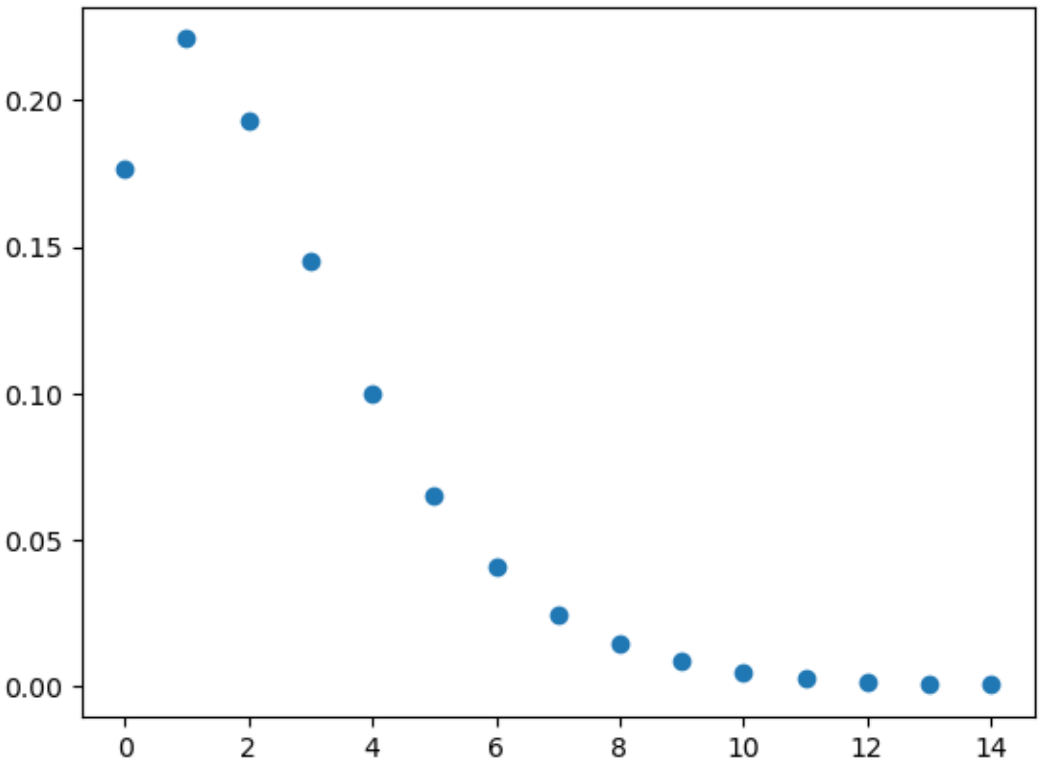
**a single generation? How does this compare to your result from A?**

Grading: 2pts for prob. of extinction, 1pt for comparison.

Let's start by plotting the negative binomial distribution

[ ]:
```
from scipy.stats import nbinom
# Define a range of values for the random variable (k)
k_values = np.arange(0, 15)
# Calculate the probability mass function (PDF) for each value of k
nBin_values = nbinom.pmf(k_values, 2.5, 0.5)
plt.scatter(k_values,nBin_values)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7f4475c65010>
```




To go extinct in a single generation the focal indivdiual must have 0 offspring.

[20]:
```
## Negative binomial offspring distribution
print(nbinom.pmf(0, 2.5, 0.5))
## poisson offspring distribution
print(poisson.pmf(0, 2.5))
```

```
0.17677669529663695
0.0820849986238988
```

The probability of extinction in the negative binomial case is more than twice as high as in the Poisson case.

The probabilitly of long term extinction under the negative binomial model is $g = 0.24$ whereas it is $g = 0.10$ in the Poisson model. This is characteristic of branching processes that the more variable the distribution the more likely there will be extinction.

**Part D. [1pt] Use your answers to the above parts to describe how extinction probability depends on the average number of offspring per parent, the mean and variance of the offspring distribution, and how these features influence the population's short-term and long-term survival.**

We have learned alot about branching processes in this example. First we have that the probability that the population goes extinct dramatically decreases with the initial number of indivdiuals. It also decreases with as the average number of offspring per parent increases. Finally, the less variance there is in the distribution the less likely the popualtion will go extinct.

When the number of offspring per parent<1 then the probability of ultimate extinction is guarenteed although it may take a few generations.

## 1.5 5. Wright-Fisher Model with Selection

**In class we discussed the Wright-Fisher model of genetic drift both with and without mutation.**

**Part A: [3pts]What is the state space in this model? What are the transition probabilities in the Wright-Fisher Model with and without mutation for a population with $N$ gene copies? In each of these models, what is the probability that a particular offspring individual carries the 'A' allelle? Assume that the probability of mutation from an 'A' to 'a' and the probability of a mutation from a 'a' to a 'A'are both $u$.**

(1pts) The state space of the model is 0 to N alleles.

(2pts) Without mutation, the transition probabilities are:

$$P_{i,j} = \binom{n}{j} \left(\frac{i}{N}\right)^j \left(\frac{N-i}{N}\right)^{N-j}$$

Here the probability of having a 'A' parent is: $\frac{i}{N}$ and the probability of having a little 'a' parent is $1 - \frac{i}{N} = \frac{N-i}{N}$

In the model with mutation, we have:

$$P_{i,j} = \binom{n}{j} \left(\frac{i}{N}(1-u) + \frac{N-i}{N} * u\right)^j \left(\frac{i}{N}u + \frac{N-i}{N}(1-u)\right)^{N-j}$$

Here the probability that the individual is a 'A' is: $\frac{i}{N}(1-u) + \frac{N-i}{N} * u$

**Part B: [2pts] Suppose that there are currently $i$ focal 'A' individuals in the population what is the frequency $p$ of that allele? What is the frequency of the alternate 'a' allele? Rewrite the transition probabilities in part A in terms of this allele frequency $p$.**

The frequency of the 'A' allele is $p = \frac{i}{N}$.

[1pts] We can rewrite the transition probability without mutation as:

$$P_{i,j} = \binom{N}{j} p^j (1-p)^{N-j}$$

[1pts] We can rewrite the transition probabilities with mutation as:

$$P_{i,j} = \binom{N}{j} (p(1-u) + (1-p)u)^j (pu + (1-p)(1-u))^{N-j}$$

**Part C: [2pts] Now, let's introduce selection favouring the focal 'A' allele. Suppose that haploid individuals with allele 'A' have fitness $W_A = 1$ and that individuals with the other 'a' allele have reduced fitness $W_a = 1$ such that the probability that a particular offspring carries the 'A' allele is given by the 'relative fitness' of that allele:**

$$w_A = \frac{pW_A}{pW_A + (1-p)W_a}$$

**What is the probability that a particular offspring carries the 'a' allele?**

There are several ways of deriving this probability. One way is to use the definition of relative fitness:

$$w_a = \frac{(1-p)W_a}{pW_A + (1-p)W_a}$$

Or we can use the fact that the probability that an individual is a 'a' has to be 1 minus the probability it is a 'A'.

$$w_a = 1 - W_A = 1 - \frac{pW_A}{pW_A + (1-p)W_a} = \frac{(1-p)W_a}{pW_A + (1-p)W_a}$$

**Part D: [2pts] Modify the Wright Fisher transition probabilities to incorporate selection.**

$$\begin{aligned} P_{i,j} =& \binom{N}{j} (w_A)^j (w_a)^{N-j} \\ =& \binom{N}{j} \left(\frac{pW_A}{pW_A + (1-p)W_a}\right)^j \left(\frac{(1-p)W_a}{pW_A + (1-p)W_a}\right)^{N-j} \end{aligned}$$

**Part E: [2pts] Use this transition probability to express the probability of having allele frequency $p(t+1) = \frac{j}{N} = p'$ in the next generation in terms of the current allele frequencies $p(t) = p$.**

We already have the probabilities in terms of $p(t)$ but now we just need to replace the $j$ with $Np(t+1) = Np'$.

$$\Pr(p(t+1)=p'|p(t)=p)=\binom{N}{Np'}\left(\frac{pW_A}{pW_A+(1-p)W_a}\right)^{Np'}\left(\frac{(1-p)W_a}{pW_A+(1-p)W_a}\right)^{N(1-p')}$$

### 1.6 6. Challenge Question: Life History Evolution

**Life history theory is the study of how organisms are born, age, reproduce, and die. Stochastic processes are important in formulating predictions and understanding of life history evolution. Consider a population with synchronous (aka discrete-time) reproduction and survival (e.g., plants). Suppose that the focal plant can exhibit one of two reproductive strategies:**

**Strategy 1: Each adult plant produces 10 seeds every year each of which have a 15% chance of surviving to become a reproductive adult in the following year.**

**Strategy 2: Each adult plant produces 50 seeds every year each of which has a 3% chance of surviving to become a reproductive adult in the following year.**

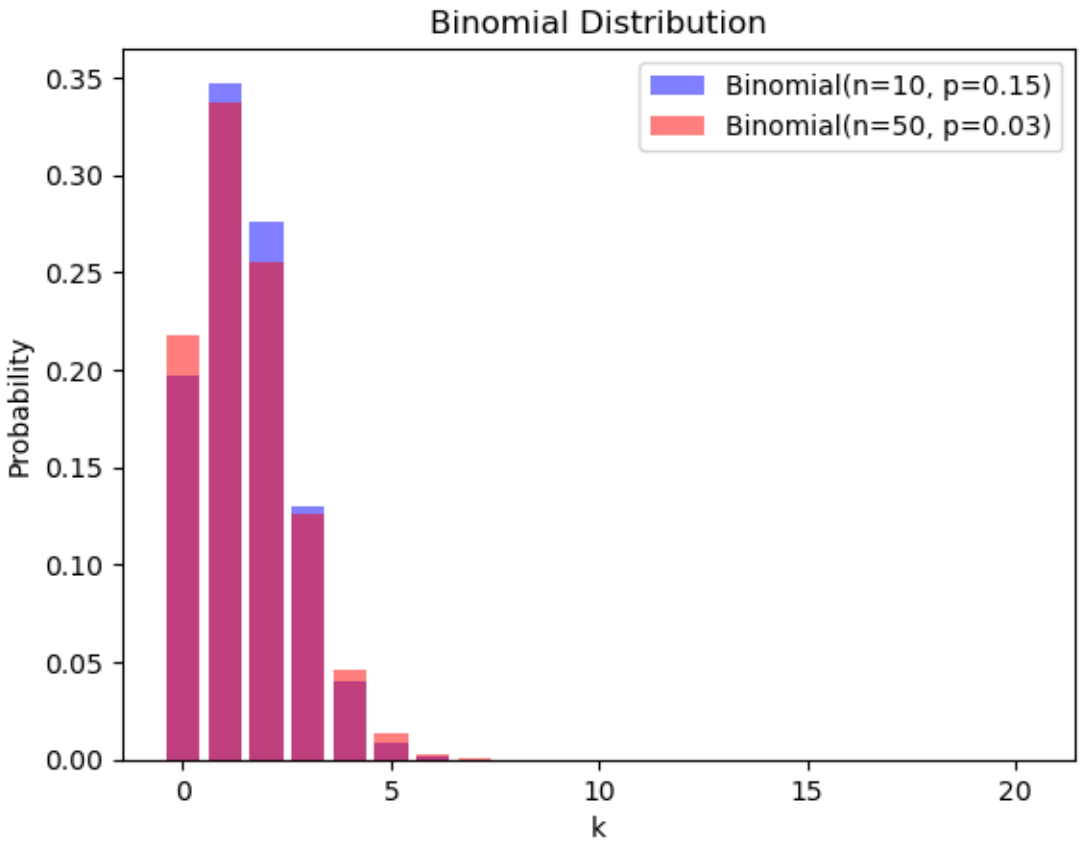
**Part A: [3pts] Propose a branching process describing the number of adult plants each year. What are the relevant model parameters for Strategy 1 and 2. What is the mean # of offspring per parent in each strategy? Plot the two distributions.**

The "offspring" distribution for both of these models are binomial distributions. For strategy 1 the distribution has parameters $N_1 = 10$ and $p = 0.1$. For strategy 2 the distribution has parameters $N_2 = 50$ and $p_2 = 0.03$. The mean number of offspring per parent is then given by the mean of the binomial distributions. Both strategies have the same mean of 1.5 offspring per parent.

[ ]:
```
print(10*0.15)
print(50*0.03)
```

[ ]:
```
1.5
```

[ ]:
```
# Generate k values from 0 to 20
k_values = np.arange(0, 21)

# Calculate the probability mass function (PMF) for each k
pmf_values1 = binom.pmf(k_values, 10, 0.15)
pmf_values2 = binom.pmf(k_values, 50, 0.03)

# Plot the binomial distribution
plt.bar(k_values, pmf_values1, label=f'Binomial(n={10}, p={0.15})',␣
 ↪color='blue', alpha=0.5)
plt.bar(k_values, pmf_values2, label=f'Binomial(n={50}, p={0.03})',␣
 ↪color='red', alpha=0.5)
plt.title('Binomial Distribution')
plt.xlabel('k')
plt.ylabel('Probability')
plt.legend()
plt.show()
```




**Part B: [4pts] What is the probability of extinction for each strategy?**

The probability of extinction $g$ is given by:

$$g = \sum_{k=0}^{\infty} \binom{n}{k} p^k (1-p)^{n-k} g^k$$

[9]:
```
# Define the equation
def equation(g, p, n):
    return g - np.sum(np.array([np.math.comb(n, k) * p**k * (1-p)**(n-k) * g**k␣
 ↪for k in range(n+1)]))

# Initial guess for g
initial_guess = 0.5

# Solve the equation numerically
result = fsolve(equation, initial_guess, args=(0.15, 10))

print("Numerical solution for g:", result[0])
```

```
# Solve the equation numerically
result = fsolve(equation, initial_guess, args=(0.03, 50))

print("Numerical solution for g:", result[0])
```

```
Numerical solution for g: 0.37151052951556324
Numerical solution for g: 0.4085508884312274
```

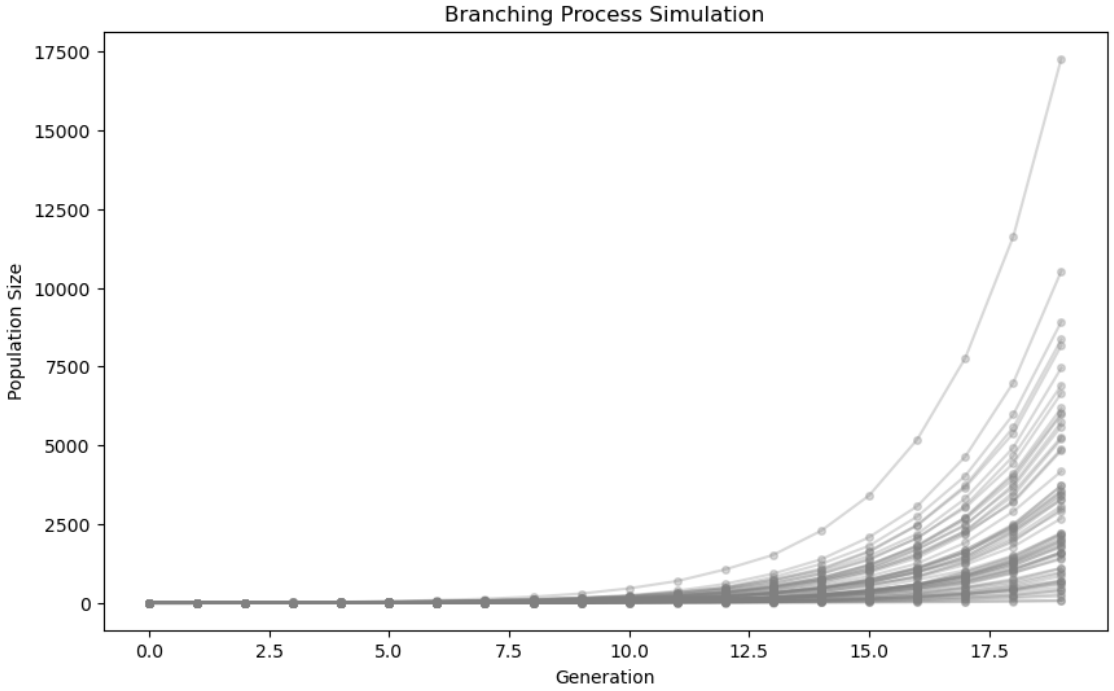
```
/tmp/ipykernel_758/2611087869.py:3: DeprecationWarning: `np.math` is a
deprecated alias for the standard library `math` module (Deprecated Numpy 1.25).
Replace usages of `np.math` with `math`
  return g - np.sum(np.array([np.math.comb(n, k) * p**k * (1-p)**(n-k) * g**k
for k in range(n+1)]))
```

Here we have that the probability of extinction in the first strategy is 37% compared with 40% for strategy 2.

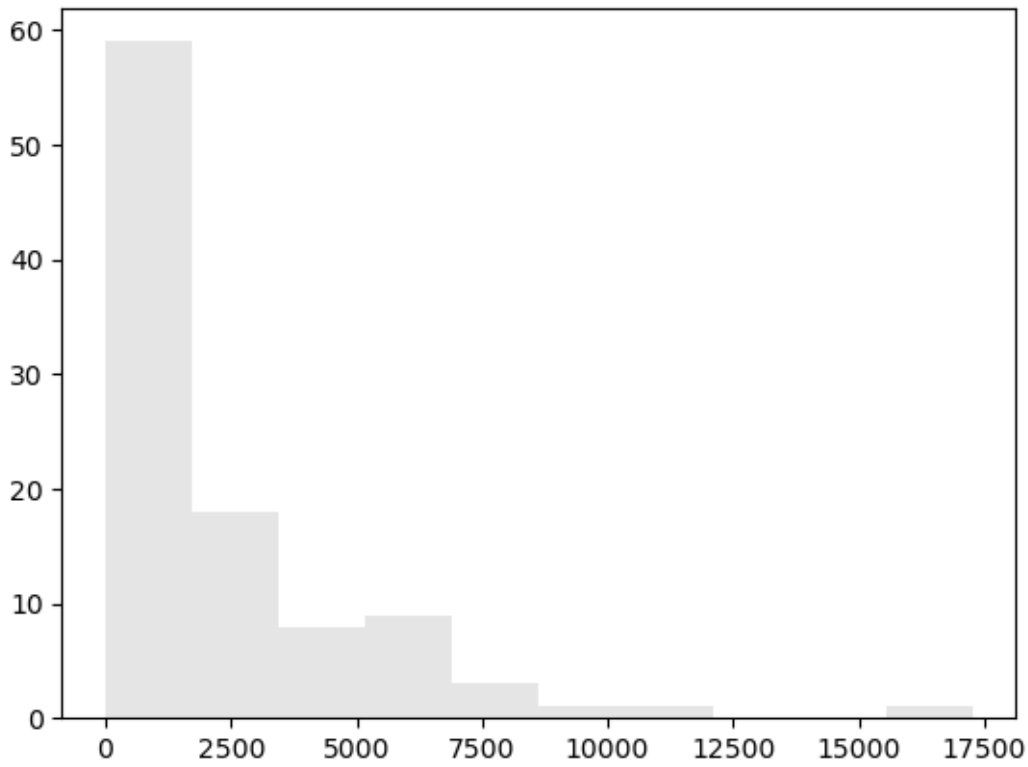
**Part C: [2pts] Simulate 100 trajectories for each strategy. How does the probability of extinction compare to your prediction in Part B? How does the distribution in population sizes compare between the two models?**

[101]:
```
# Parameters
n = 10  # Number of trials in the binomial distribution
p = 0.15  # Probability of success in each trial
num_trajectories = 100  # Number of trajectories
max_generations = 20  # Maximum number of generations

# Function to simulate a single trajectory
def simulate_trajectory(n, p, max_generations):
    population_sizes = [1]  # Initial population size
    for generation in range(1, max_generations):
        offspring = np.random.binomial(n, p, population_sizes[-1])
        new_population_size = np.sum(offspring)
        if new_population_size == 0:
            break  # Extinction event
        population_sizes.append(new_population_size)
    return population_sizes

# Simulate multiple trajectories
trajectories = []
for _ in range(num_trajectories):
    trajectory = simulate_trajectory(n, p, max_generations)
    trajectories.append(trajectory)

# Plot the trajectories
plt.figure(figsize=(10, 6))
for trajectory in trajectories:
    plt.plot(trajectory, marker='o', linestyle='-',␣
 ↪markersize=4,color='gray',alpha=0.3)
```

```
plt.title('Branching Process Simulation')
plt.xlabel('Generation')
plt.ylabel('Population Size')
plt.show()
```

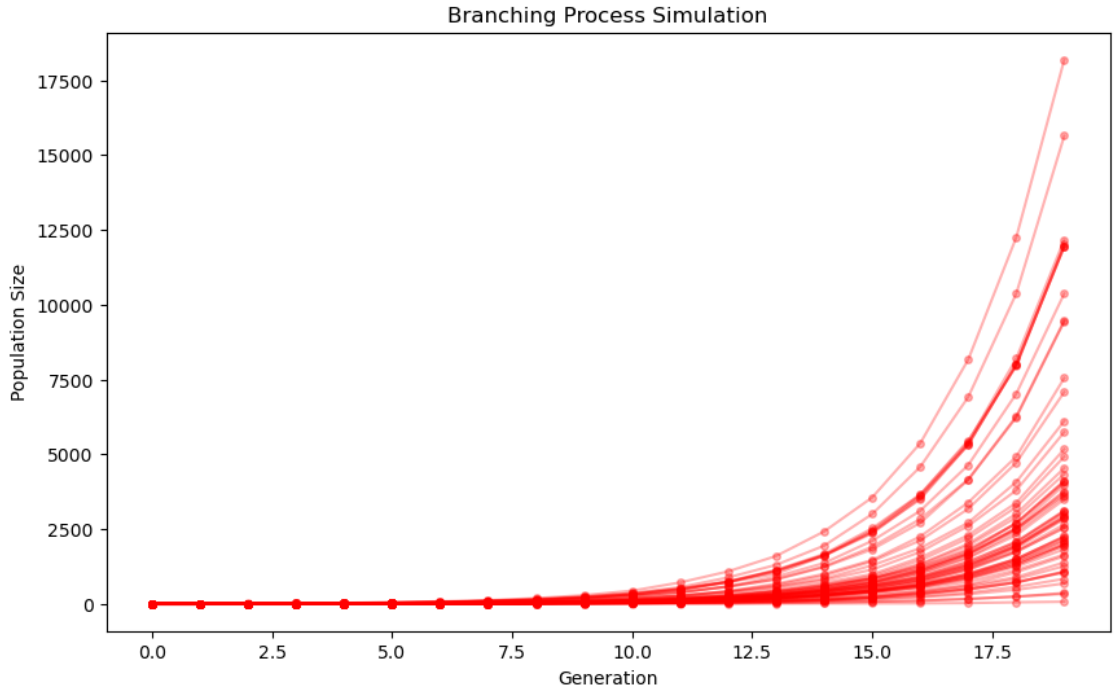



[104]:
```
finalSize = []
for i in range(num_trajectories):
    finalSize.append(trajectories[i][-1])
plt.hist(finalSize,color='gray',alpha=0.2);
```

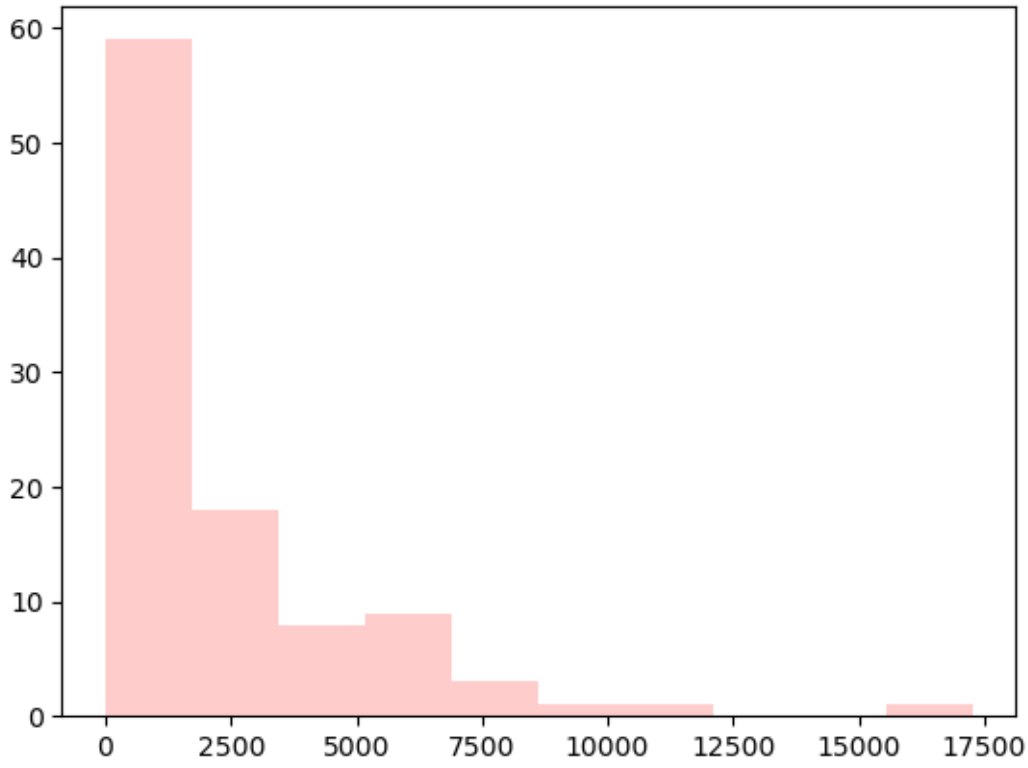



[43]:
```python
# Parameters
n = 50  # Number of trials in the binomial distribution
p = 0.03  # Probability of success in each trial
num_trajectories = 100  # Number of trajectories
max_generations = 20  # Maximum number of generations

# Function to simulate a single trajectory
def simulate_trajectory(n, p, max_generations):
    population_sizes = [1]  # Initial population size
    for generation in range(1, max_generations):
        offspring = np.random.binomial(n, p, population_sizes[-1])
        new_population_size = np.sum(offspring)
        if new_population_size == 0:
            break  # Extinction event
        population_sizes.append(new_population_size)
    return population_sizes

# Simulate multiple trajectories
trajectories = []
for _ in range(num_trajectories):
    trajectory = simulate_trajectory(n, p, max_generations)
    trajectories.append(trajectory)
```

```
# Plot the trajectories
plt.figure(figsize=(10, 6))
for trajectory in trajectories:
    plt.plot(trajectory, marker='o', linestyle='-',␣
 ↪markersize=4,color='red',alpha=0.3)

plt.title('Branching Process Simulation')
plt.xlabel('Generation')
plt.ylabel('Population Size')
plt.show()
```




[103]:
```
finalSize = []
for i in range(num_trajectories):
    finalSize.append(trajectories[i][-1])
plt.hist(finalSize,color='red',alpha=0.2);
```




**Part D:[1pts] Use your answers to part B and part C to formulate a hypothesis about which strategy is favoured by natural selection.**

Strategy 1 will be favoured.

[ ]: