Exam1

March 5, 2024

1 Midterm 1: Take Home

Consider a population in which the number of offspring per parent is geometrically distributed with success probability $p = 0.3$ such that the probability that a parent has k offspring is:

$$\Pr(k) = (1-p)^k p$$

Note: There are two alternative parameterizations of the geometric distribution, use the one corresponding to the PMF above. This is NOT the notation used in Python.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.stats import nbinom
from scipy.integrate import solve_ivp
import random as rand
from scipy.special import comb
from scipy.linalg import expm
from scipy.stats import geom
from scipy.optimize import fsolve
```

Part A [1pt]: What is the expected number of offspring per parent?

The mean of the geometric distribution is $E[X] = \frac{1}{p} - 1 = 2.22$

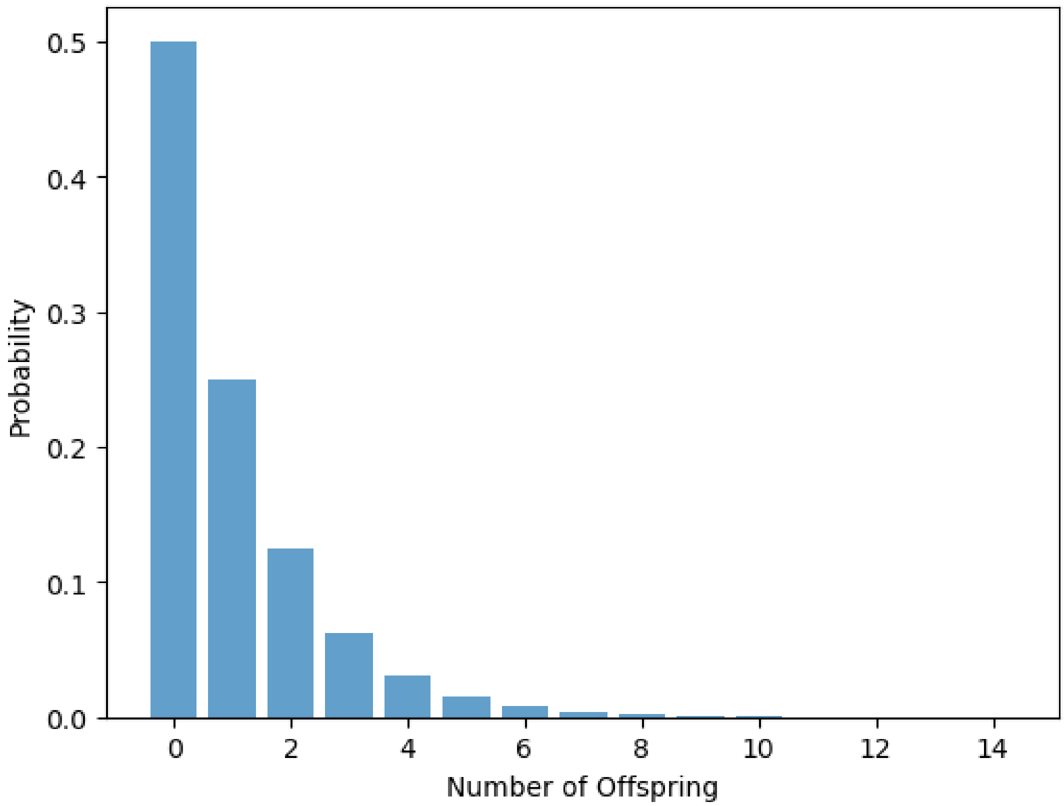
[10]:
```
p=0.3
1/p-1
```

[10]: 2.3333333333333335

Part B [1pt]: Plot the distribution of offspring per parent.

[32]:
```
# Generate x values (number of trials) from 1 to 10
k_values = np.arange(0, 15, 1)

# Calculate the PMF for each x value
pmf_values =(1 - p) ** k_values * p
```

```
# Plot the PMF
plt.bar(k_values, pmf_values, align='center', alpha=0.7)
plt.xlabel('Number of Offspring')
plt.ylabel('Probability')
plt.show()
```



Part C [1.5pt]: Given that the population starts with 2 individuals, what is the probability that the population eventually goes extinct?

We start by considering the probability of extinction for a single individual.

$$P_{\text{Ext}} = \sum_{k=0}^{\infty} (1-p)^k p P_{\text{Ext}}^k$$

[20]:
```
# Define the probability of success
p = 0.3

# Define the function representing the equation
def equation(P_Ext):
```

```
    return P_Ext - np.sum((1 - p) ** np.arange(0, 100) * p * P_Ext ** np.
↪arange(0, 100))

# Initial guess for P_Ext
initial_guess = 0.5

# Solve the equation numerically
solution = fsolve(equation, initial_guess)[0]

print(f'Probability of extinction of a single indivdiual, P_Ext: {solution:.
↪4f}')
```

```
Probability of extinction of a single indivdiual, P_Ext: 0.4286
```

To get the probability of extinction of two initial individuals we have to calculate P_{Ext}^2.

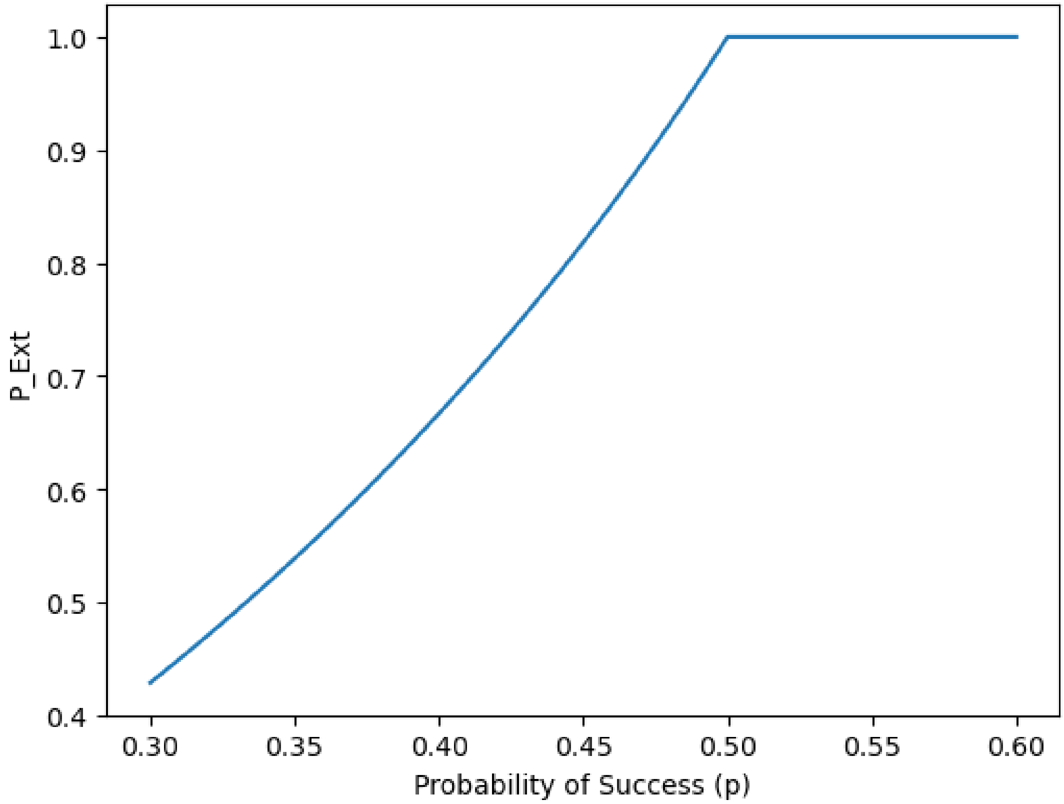
[21]:
```
0.4286**2
```

[21]:
```
0.18369796
```

Part D [1.5pt]: For what value of p is the population guarenteed to go extinct?

[29]:
```
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
import numpy as np

# Define a range of probability of success (p) values
p_values = np.linspace(0.3, 0.6, 100)

# Function to solve for P_Ext given p
def solve_equation(p):
    def equation(P_Ext):
        return P_Ext - np.sum((1 - p) ** np.arange(0, 100) * p * P_Ext ** np.
↪arange(0, 100))

    # Initial guess for P_Ext
    initial_guess = 0.5

    # Solve the equation numerically
    solution = fsolve(equation, initial_guess)[0]
    return solution

# Solve for P_Ext for each p
solutions = [solve_equation(p) for p in p_values]

# Plot the solutions
plt.plot(p_values, solutions, label='Numerical Solution')
plt.xlabel('Probability of Success (p)')
```

```
plt.ylabel('P_Ext')
plt.show()
```



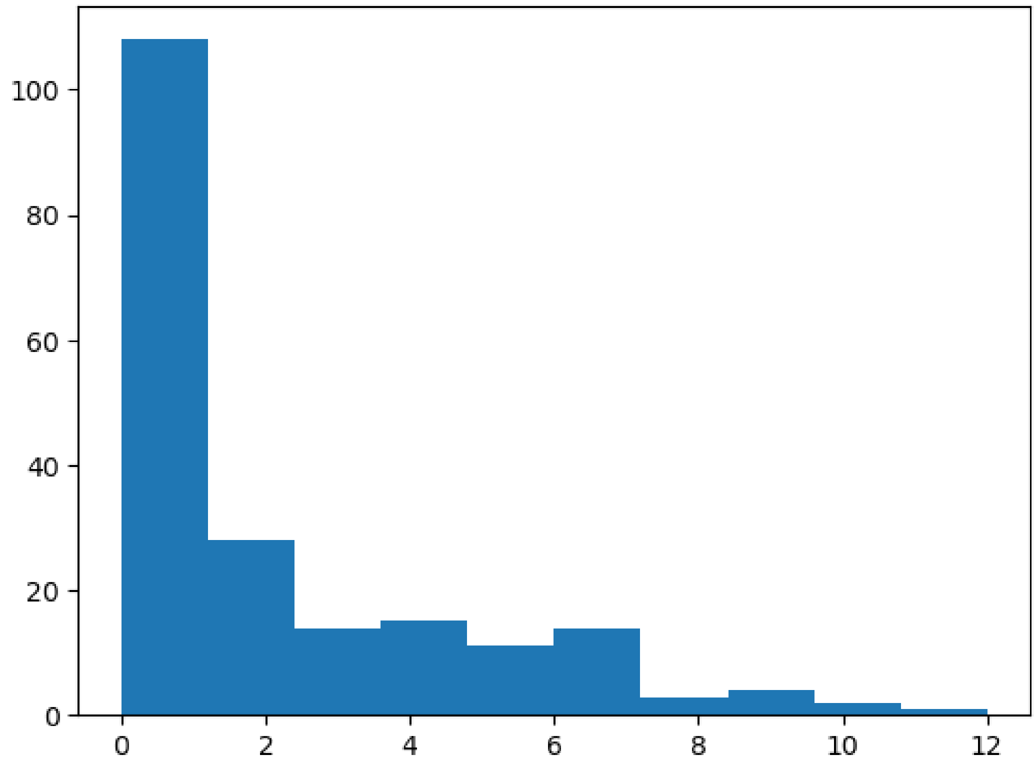
Part E [2pt]: Write a simulation of this branching process and plot one example trajectory

[2]:
```
p=0.3
# Generate x values (number of trials) from 1 to 10
k_values = np.arange(0, 15, 1)
# Calculate the PMF for each x value
pmf_values =(1 - p) ** k_values * p
# Calculate the CDF for each x value
cdf_values=np.cumsum(pmf_values)
def randGeom():
    r=rand.random()
    i=0;
    while r>cdf_values[i] and i<14:
        i=i+1
    return i
```

[3]:
```
temp=np.array([randGeom() for i in range(200)])
```

[4]:

```
plt.hist(temp);
```



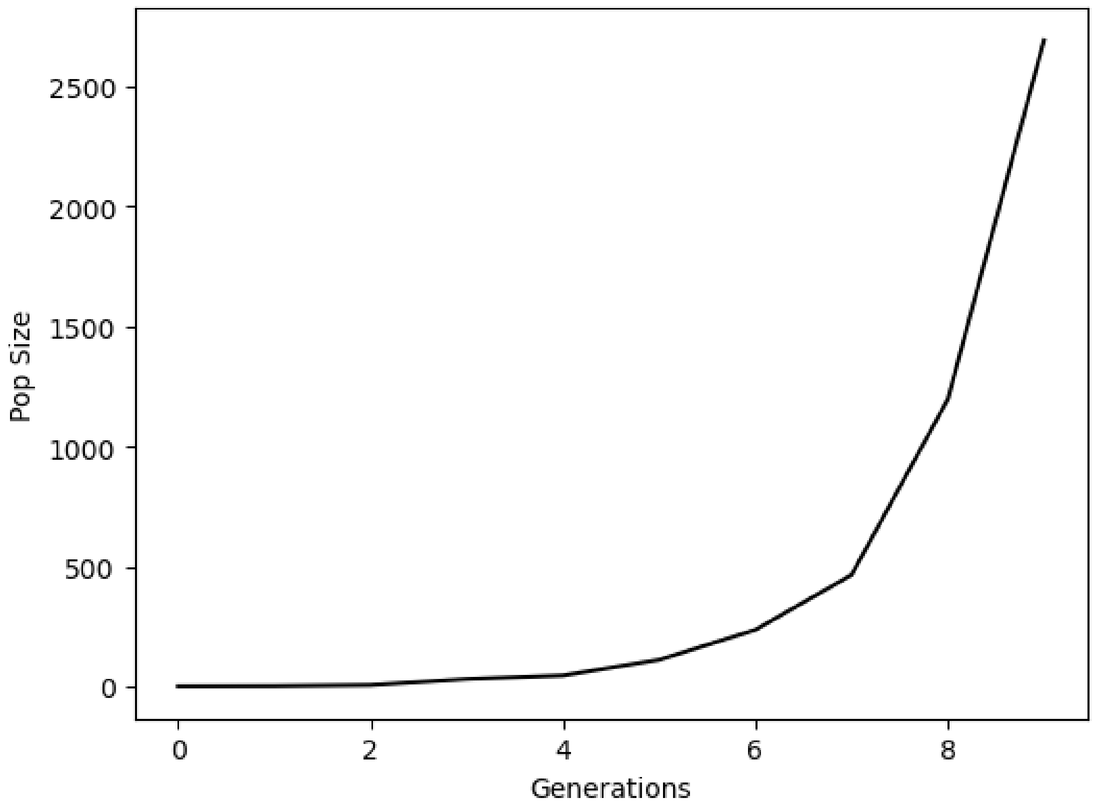
[7]:

```
import numpy as np

def simulate_branching_process(initial_population_size, p, num_generations):
    population_sizes = [initial_population_size]

    for generation in range(1, num_generations):
        # offspring_counts = np.random.geometric(p,␣
 ↪size=population_sizes[generation - 1])
        if population_sizes[generation - 1]>0:
            offspring_counts=np.array([randGeom() for i in␣
 ↪range(population_sizes[generation - 1])])
            new_population_size = np.sum(offspring_counts)
        else:
            new_population_size=0
        population_sizes.append(new_population_size)

    return population_sizes

# Parameters
initial_population_size = 1  # Initial population size
```

```
p = 0.3  # Probability of success for geometric distribution
num_generations =10  # Number of generations to simulate

# Simulate branching process
population_sizes = simulate_branching_process(initial_population_size, p,␣
↪num_generations)

population_sizes
```

[7]: [1, 2, 6, 31, 46, 111, 236, 465, 1196, 2693]

[20]:
```
plt.plot(population_sizes, label='Vectors',color='black')
plt.xlabel('Generations')
plt.ylabel('Pop Size');
```

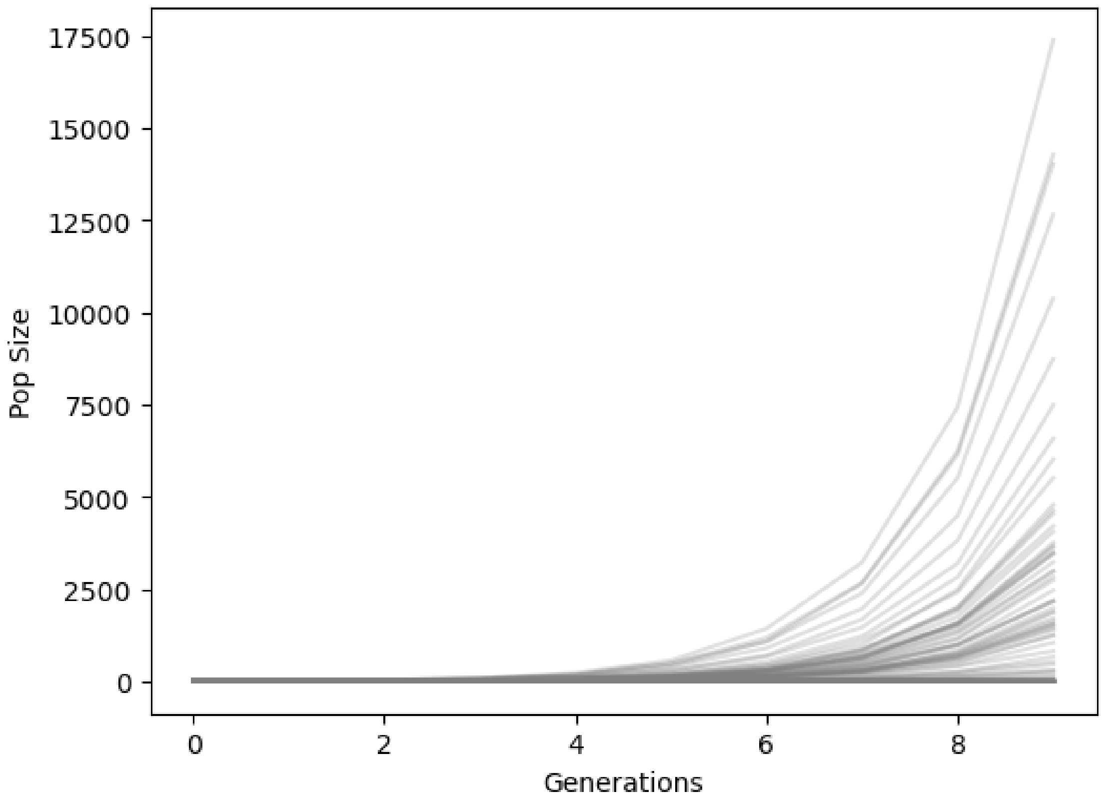


Part F [2pt]: Simulate 100 sample trajectories for $t = 10$ generations for $p = 0.3$ and a single initial indivdiual. What is the observed probability of extinction in these simulations?

[9]:
```
# Create a dictionary to store results
sim_dict = {}
```

```
# Calculate and save results for specified indices
for index in range(100):
    sim_dict[index] = simulate_branching_process(initial_population_size, p,␣
 ↪num_generations)
```

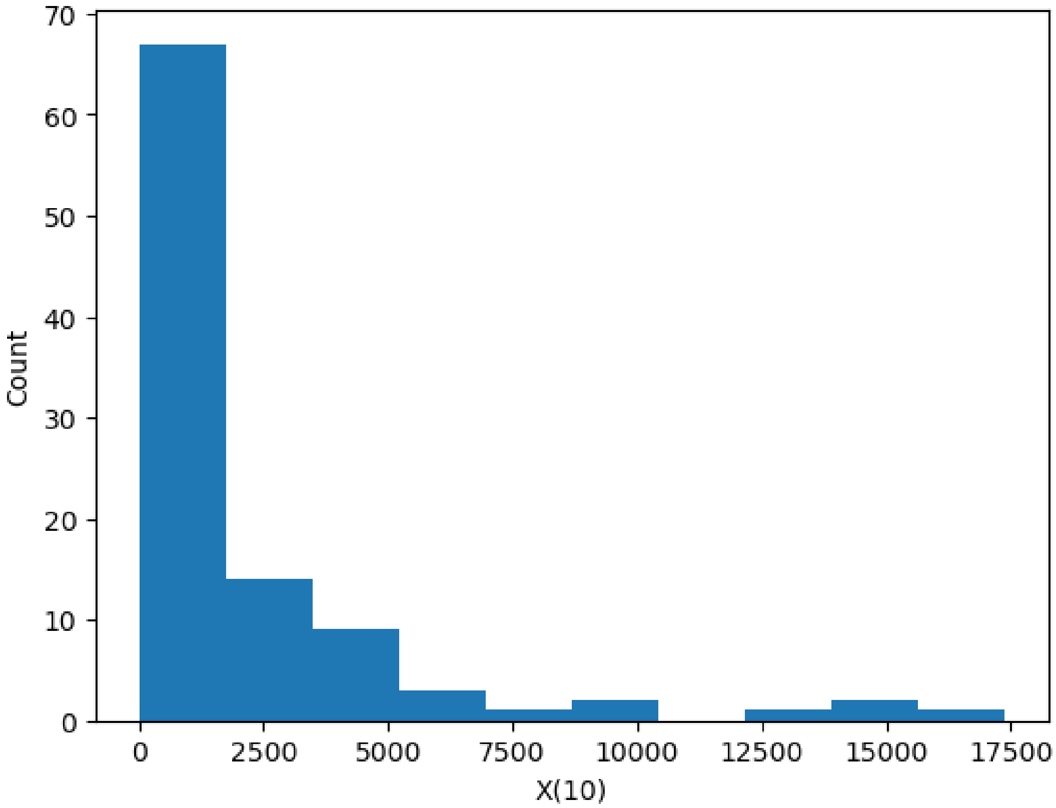
[21]:
```
for index in range(100):
    plt.plot( sim_dict[index], label='Vectors',color='gray',alpha=0.25)
plt.xlabel('Generations')
plt.ylabel('Pop Size');
```



[16]:
```
finalsize=np.array([sim_dict[index][-1] for index in range(100)])
print(finalsize)
plt.hist(finalsize);
plt.xlabel('X(10)')
plt.ylabel('Count');
```

```
[14280  4209     0     0     0 17387     0     0     0     0     0   284
  3463  2185  8738  1244     0     0  2978     0  1985  6015  3229  1046
  1386     0   111     0   232     0  2179   531     0  1249     0  1890
  5514  2837     0  3674  2158  4786   454     0     0  2465     0  4641
```

```
    0     0     0     5     0  3506  2738     0  7486     0  1549     0
    0  1709     0  2993  3629     0   816  1862 12658     0     0  4543
  661     0     0   301     0  3755  1504  1443     0  1632     0  6578
    0 10383     0  3454     0  1559   141     0 14026     0     0     0
    0     0     0  4058]
```



[18]:
```python
count_zeros = np.sum(finalsize == 0)
print("Number of 0s:", count_zeros)
print("Estimated Prob of Extinction:", count_zeros/100)
```

```
Number of 0s: 47
Estimated Prob of Extinction: 0.47
```

Part G [1pt]: Do you think your answer to part F would change substantially if you were to simulate the trajectories for $t = 100$ generations, why or why not?

No. The populations that are not extinct at $t = 10$ are very large.