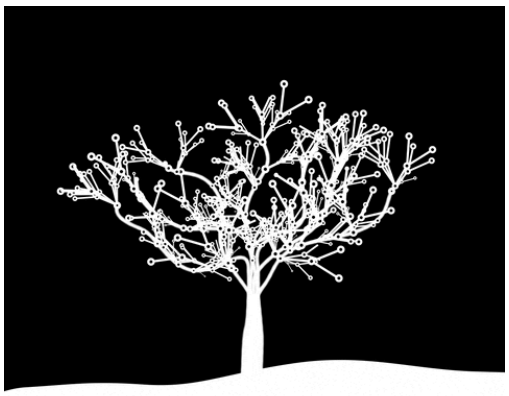
Branching Processes

What are Branching Processes?

A **branching process** is a stochastic model used in probability theory and statistics to describe the evolution of populations or systems over time. It is often used to model phenomena such as population growth, the spread of diseases, or the lifespan of particles in physics.



In a branching process, individuals can give rise to offspring according to certain probabilistic rules. Each individual in the current generation can produce a random number of offspring, and the numbers of offspring produced by different individuals are typically assumed to be independent and identically distributed (i.i.d.). The distribution governing the number of offspring produced by each individual is called the offspring distribution.

Branching processes are often represented using a branching tree diagram, where each node represents an individual and the branches represent their offspring. By analyzing the properties of the branching process and the offspring distribution, statisticians can make predictions about the long-term behavior of the population or system, such as the probability of extinction or the expected population size.

What Can They Teach Us?

To find out how a branching process can be useful in making preditions, lets start by defining the parameters of a branching process. Afterwards, we can ask what significant information we can

derive out of this model.

Definitions

In a branching process we are concerned with the size of the population at a given time.

Let Z_t denote the population of the process at time t, where time can be either discrete or continuous and where population is discete.

Also let $Z_0 = 1$.

Nodes (parents) will branch (have childeren) according to a distribution refered to as the **offspring distribution**.

$$\{p_0, p_1, p_2, \dots p_n\}$$

Where n is the amount of branches or offspring a node will have. Also remember the sum of these probabilities must equal 1. Every node will have the same offspring distribution and all nodes are independent from eachother.

Here we can see that the amount of offspring from a node will be a random variable which we will call $Y_{t,i}$ where t is time and i is the ith member of the population at the previous time. We can now rewrite X_t as the sum of all $Y_{t,i}$ for a given time.

$$Z_t = Y_{t,1} + Y_{t,2} + \dots + Y_{t,i} = \sum_{i=0}^{Z_{t-1}} Y_{t,i}$$

Note: We see the Markov property here. Future generations are conditionally independent of the past given that we know the present value. In other words the population of the next generation only depends on what the current population is.

What Questions Can we Ask?

There are several questions we can ask and seek out answers for pertaining to this stochastic model.

- What is the probability the population has a certain value at a certain time?
- What is the probability the population goes extinct?
- What is the expected size of the population at a certain time?
- What is the total population seen across generations?
- What is the variance of the process at a certain time?

The main focus for many statisticians is the 2nd bulletpoint, the probability of extinction.

Derivations and Results

Note: It is possible to derive the formulas for the values seen in the questions above, however this is a class about application of stocastic processes so I will just give an overview of the derivations and emphasize the final results. Full derivations can be found here.

Probability

To find probabilities for branching processes, a new statistical tool is needed. Proability Generating Functions (PGF) are exactly what they sound like. They are functions that create probabilties for certain inputs.

$$\Pi_X(s) = E(s^X) = \sum_{x=0}^{\infty} P(X = x)s^x$$

The function here is Π and pertains to a discrete random variable X with input s which is a notekeeping variable.

It can be shown that:

$$P(X = n) = \frac{\Pi_X(0)^{(n)}}{n!}$$

Which is a way we can calculate probabilities for different values of a random variable.

We can also use PGFs to derive p_e, the probability of extinction. This (along with the previous result) takes a good bit of math to prove and a few times reading over to get comfortable with. The final result is easier to grapple with.

$$p_e = \Pi_{X_0}(p_e)$$

Here we set the extiction probability equal to the PGF of the offspring distribution, Π_{X_0}, with the extiction probability as the input. Then we can solve for p_e to find our probability.

Note: This often leads to a polynomial equation with multiple possible solutions. It can be proven that the smallest solution is the correct one. Look here for more details.

Expected Value

We may also be interested in the population at a specific time or the long term total population.

If we let μ be the mean of the offspring distribution then we can find the expected size of the population at time t as the following:

$$\mu_t = E(X_t) = \mu^t$$

To find the total population across all generations lets define a new variable M that is the sum of the differnt populations.

$$M_t = X_0 + X_1 + \ldots + X_t$$

Then we can find the expected value of M which results in:

$$E(M_t) = 1 + \mu + \mu^2 + \ldots + \mu^t$$

Variance

The variance of the population at time t is as follows:

$$\sigma_t^2 = \mu^{t-1}\sigma^2(1 + \mu + \mu^2 + \ldots + \mu^{t-1})$$

where σ is the variance of the offspring distribution.

This results in a few cases based on the value of μ.

μ	σ^2
> 1	$\text{as } t \to \infty;\ \sigma_t^2 \to \infty$
$= 1$	$t \cdot \sigma^2$
< 1	$\frac{\mu^{t-1}\sigma^2}{1-\mu}$

Application of Branching Processes

Now that we have established what a branching process is, some of its properties and some of its values of interest, lets take some time to apply what we have just talked about.

Finding Probabilities

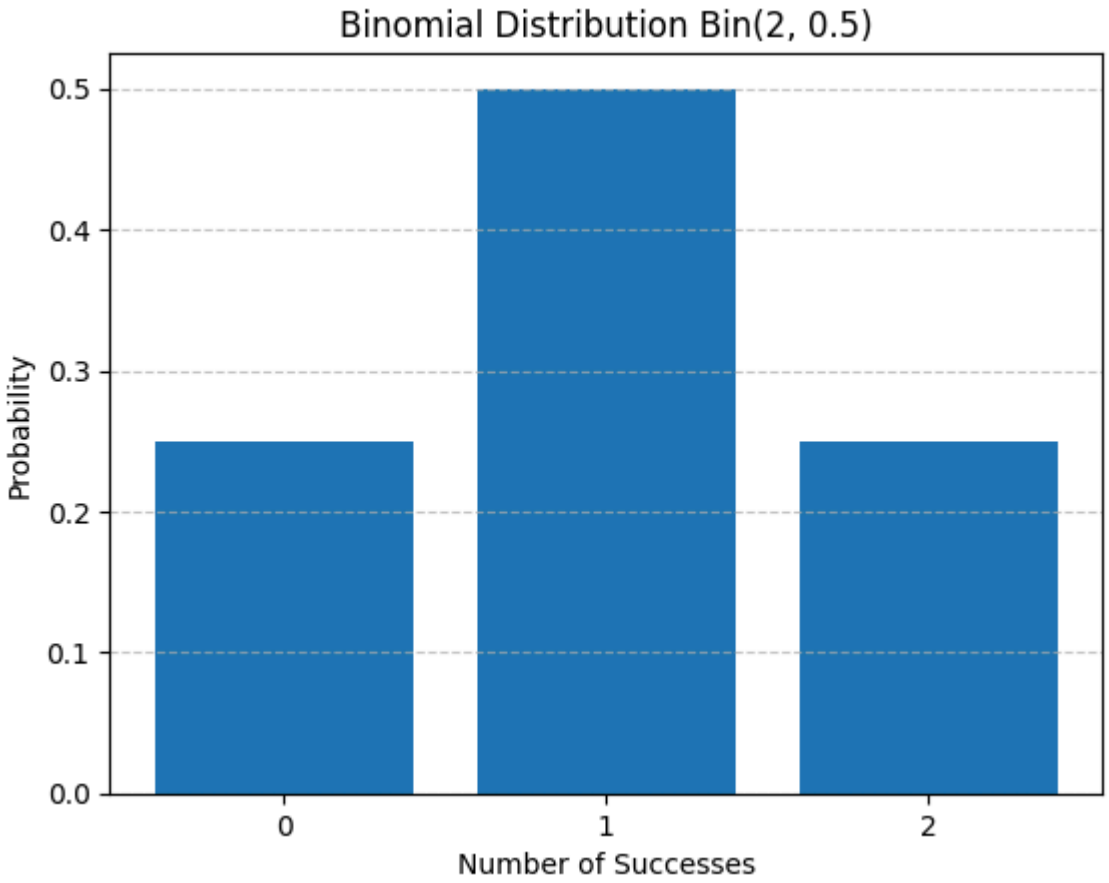
Blurb about binomial distribution and finding its PGF and offspring distribution and link to bookdown...

Lets take some time to practice finding probabilities.

We want to be able to find the extinction probability and random variable probabilities using PGFs. First however lets get a little refresher on the binomial distribution and how it behaves. This will help us later in our practice.

Below we have a code cell where you can explore the distribution of $Bin(n, p)$ by playing around with n and p. Feel free to play around with it, progress in the application section and come back to it to verify that values match.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats

#-------------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Define parameters
n = 2  # Number of trials
p = 0.5  # Probability of success

#-------------------------------------------------------------------------------#

# Generate the distribution
x = np.arange(0, n+1)  # Possible number of successes
pmf = scipy.stats.binom.pmf(x, n, p)  # Probability mass function
distribution_df = pd.DataFrame({'Number of Successes': x, 'Probability': pmf})



# Plot the distribution
plt.bar(x, pmf)
plt.xlabel('Number of Successes')
plt.ylabel('Probability')
plt.title('Binomial Distribution Bin({}, {})'.format(n, p))
plt.xticks(x)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

print(distribution_df)

# Find Expectec Value of Binom
ev = 0
for i in range(0,len(pmf)):
  ev = ev + i*pmf[i]

# n*p and ev are eqivalent
print("\nFor Binom({:d},{:0.4f}), Expected Value = {:0.4f}".format(n,p,ev))
```



	Number of Successes	Probability
0	0	0.25
1	1	0.50
2	2	0.25

For Binom(2,0.5000), Expected Value = 1.0000

Now we will explore how to find the probability of a certain value of a random value using a PGF.

Through this derivation, PGF of a $Bin(n, p)$ can be found to be:

$$\Pi_X(s) = (1 - p + p \cdot s)^n$$

We can use this in combination with the following to determine the probability of some value from a random variable.

$$P(X = x) = \frac{\Pi_X(0)^{(x)}}{x!}$$

So we are going to take the xth derivative of the binomial PGF with respect to s. Then we can plug in 0 for s and divide by the factorial. This should result in the same probability as values generated above for the binomial distribution.

```
import sympy as sp
import math

#-------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Number of trials
n = 2

# Probability of success
p_val = 0.5
q_val = 1-p_val

# Random Variable value of interest
x = 1

#-------------------------------------------------------------------------#

# Define symbols
p = sp.symbols('p')
q = sp.symbols('q')
s = sp.symbols('s')

# Define the PGF
pgf_Binom = (q+p*s)**n

# Take the derivative of the expression with respect to a variable
derivative = sp.diff(pgf_Binom, s, x)
# print("Derivative:", derivative)

# Find the factorial
factorial = math.factorial(x)

# Suppose you want to substitute specific values
values = {p: p_val, q: q_val, s:0}

# Find probability
P_x = derivative.subs(values)/factorial
print("For Binom({:d},{:0.4f}), P(X = {:d}) = {:0.4f}".format(n,p_val,x,P_x))
```

```
For Binom(2,0.5000), P(X = 1) = 0.5000
```

Lastly for probability, we can explore how to find the probability of extinction p_e.

As a reminder we can find p_e by setting it equal to the PGF of the offspring distribution with p_e as the input and solving p_e.

$$p_e = \Pi_{X_0}(p_e)$$

In this case the PGF of our offspring distribution is the same as the PGF of our binomial random variable

$$\Pi_{X_0}(p_e) = \Pi_X(p_e) = (1 - p + p \cdot s)^n = p_e$$

```
import sympy as sp

#-------------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Number of trials
n = 2

# Probability of success
p_val = 0.5
q_val = 1-p_val

#---Note---:
# The math and debugging gets nasty after n = 2, so instead of getting a
# symbolic solution and trying to solve for p_e after the fact, we will just
# skip finding the symbolic solution and jump straight to finding the
# probability of extinction.

#-------------------------------------------------------------------------------#

if n <=2 :
  # Define symbols
  p = sp.symbols('p')
  q = sp.symbols('q')
  s = sp.symbols('s')

  # Define the PGF when set equal to p_e but solved so all terms are on one side
  pgf_Binom_ext = (q+p*s)**n - s

  # Solve the equation
  solutions = sp.solve(pgf_Binom_ext, s)
  print("Solutions:", solutions, "\n")

  # Suppose you want to substitute specific values for 'p' and 'q'
  values = {p: p_val, q: q_val}

  # Substitute the values into the solution(s) and find probability of extinction
  p_e = 1.1 # set higher than maximum to allow for it to find minimum

  for i in range(0,len(solutions)):
    solution_with_values = solutions[i].subs(values).evalf()
    print("Solution {:d} with values substituted: {:0.4f}".format(i+1,solution_with_values))

    if p_e > solution_with_values:
      p_e = solution_with_values

  # Present results on extinction probability
  print("\n-------------------------------------------------------------------------")
  print("For Binom({:d},{:0.4f}), extinction probability = {:0.4f}".format(n,p_val,p_e))

else :
```

```
# Define the PGF
pgf_Binom_ext = (q_val+p_val*s)**n - s

# Solve the equation
solutions = sp.solve(pgf_Binom_ext, s)
print("Solutions:", solutions)

# Find probability of extinction - note p_e should be between 0-1
p_e = 1.1 # set higher than maximum to allow for it to find minimum
for i in range(len(solutions)):
  if solutions[i].is_real:
      if solutions[i] > 0:
          if p_e > solutions[i]:
             p_e = solutions[i]

# Present results on extinction probability
print("\n----------------------------------------------------------------------")
print("For Binom({:d},{:0.4f}), extinction probability = {:0.4f}".format(n,p_val,p_e))
```

```
Solutions: [(-2*p*q - sqrt(-4*p*q + 1) + 1)/(2*p**2), (-2*p*q + sqrt(-4*p*q + 1) + 1)/(2

Solution 1 with values substituted: 1.0000
Solution 2 with values substituted: 1.0000

----------------------------------------------------------------------
For Binom(2,0.5000), extinction probability = 1.0000
```

Finding Expected Value

Lets continue the exploration of branching processes and find the expected population at a given time when our offspring distribution is binomially distributed.

If we have $Bin(n, p)$ describe our distribution of decendents for each node, then we can take the average of our offspring distribution μ to be the expected value of $Bin(n, p)$. In otherwords we can let $\mu = n \cdot p$.

Then to find the expected value of the population we can refer to our equation above.

$$\mu_t = \mu^t$$

```
#-------------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Define parameters
n = 2  # Number of trials
p = 0.5  # Probability of success
t = 3  # number of generations

#-------------------------------------------------------------------------------#

#offspring distribution
mu = n*p

#Expected Value at current generation
mu_t = mu**t

print("\nFor Binom({:d},{:0.4f}), Expected Value at generation {:d} = {:0.4f}".format(n,p,t,
```

```
     For Binom(2,0.5000), Expected Value at generation 3 = 1.0000
```

We could extend this to find the total population across generations.

```
#-------------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Define parameters
n = 2  # Number of trials
p = 0.5  # Probability of success
t = 3  # number of generations

#-------------------------------------------------------------------------------#

#offspring distribution
mu = n*p

#Expected Value to current generation
E_M = 0
for i in range(0,t+1):
  E_M += mu**t

print("\nFor Binom({:d},{:0.4f}), Expected Value up to generation {:d} = {:0.4f}".format(n,p
```

```
     For Binom(2,0.5000), Expected Value up to generation 3 = 4.0000
```

Finding Variance

Finally we can conclude with finding variance.

If we have $Bin(n, p)$ describe our distribution of offspring, then we can take the offspring variance σ to be the variance of $Bin(n, p)$. Alternatively, $\sigma = n \cdot p \cdot q$.

Then to find the variance of the population we can refer to our equation above.

$$\sigma_t^2 = \mu^{t-1}\sigma^2(1 + \mu + \mu^2 + \ldots + \mu^{t-1})$$

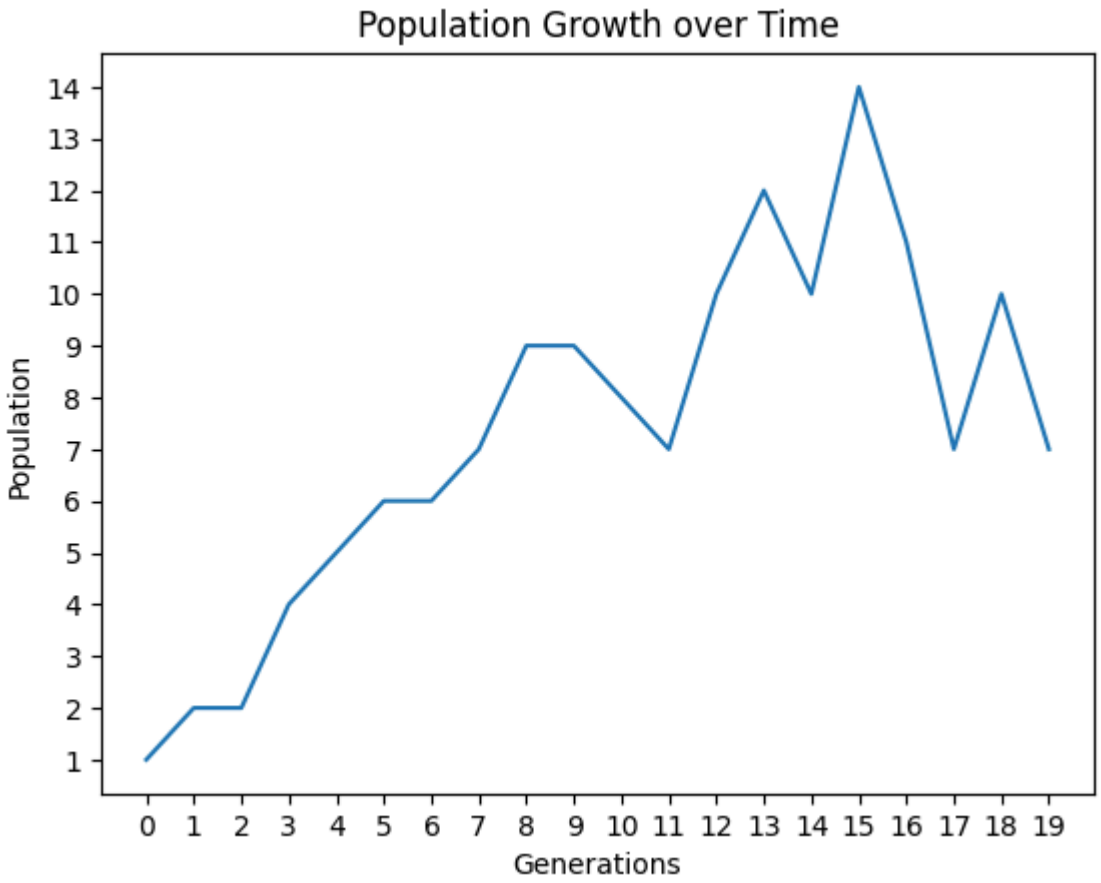
```
#------------------------------------------------------------------------------#
#---USER EXPLORATION HERE!!!---#

# Define parameters
n = 2  # Number of trials
p = 0.5  # Probability of success
q = 1-p
t = 3  # number of generations

#------------------------------------------------------------------------------#

#offspring distribution
mu = n*p

#offspring variance
var = n*p*q

#Expected Value to t-1
sum = 0
for i in range(0,t):
  sum += mu**t

var_t = (mu**(t-1))*var*sum
print("\nFor Binom({:d},{:0.4f}), Variance at generation {:d} = {:0.4f}".format(n,p,t,var_t)
```

```
For Binom(2,0.5000), Variance at generation 3 = 1.5000
```

Simulation of Branching Processes

While we can find analytical solutions for branching processes, that give us a sense of their general behavior, we can also simulate these processes to and verify they behave as we predict.

```
import numpy as np
import matplotlib.pyplot as plt

# Select generations to simulate
time = np.arange(20)

# Initialize storage for population RV
X = np.zeros(len(time), dtype=int)

# Initial population
X[0] = 1

# offspring distribution parameters
n = 2  # Number of trials
p = 0.5  # Probability of success

# Simulate evolution of population
for t in time:
    for i in range(0, X[t-1]):
        offspring = np.random.binomial(n, p)
        X[t] += offspring

# Plot
plt.plot(time, X)

# Set axis ticks to whole numbers
plt.xticks(np.arange(min(time), max(time)+1, 1))
plt.yticks(np.arange(min(X), max(X)+1, 1))

plt.xlabel('Generations')
plt.ylabel('Population')
plt.title('Population Growth over Time')
plt.show()

# Total population
M = X.sum()
print("Total:",M)
```




```
Total: 147
```

Conclusion

In this study, we delved into the realm of branching processes, a powerful mathematical tool for modeling population growth and various dynamic systems. Through our exploration, we uncovered their fundamental principles and applications like their role in modeling reproduction dynamics. Branching processes offer valuable insights into the probability of ultimate extinction and the dynamics of evolving systems.

This is not the end, but rather the beginning. There are many variations out there of the branching process all which cater to different modeling needs. Some of these variations have changing rates or certain criteria that must be met for reproduction. Some are dependent on the size or age of the population. There are even variations like Multitype branching processes where not all nodes behave the same. This is all to say there are plenty of branches to explore.

References

1. https://en.wikipedia.org/wiki/Branching_process

2. https://bookdown.org/probability/bookdown-demo/branching-processes.html#long-term-population-metrics
3. https://www.youtube.com/playlist?list=PLRogqfr-vZMfvcLjjmXnb0wk6MPTzhph-