SCUDEM VIII 2023

Problem B: Punishing Infants

New York University

Edward Chang
Jordi Del Castillo
Paul Jaikaran

Problem B: Punishing Infants

- Study done on the tendency to punish antisocial behavior
- Researchers found that infants have the innate capacity to “punish”
- Infants will punish those who they believe are hurting others
- Based on this we will assume that is a human trait
- What does this natural tendency imply about society?
- What does this say about the long-term dynamics of different populations?

Goals of our Model

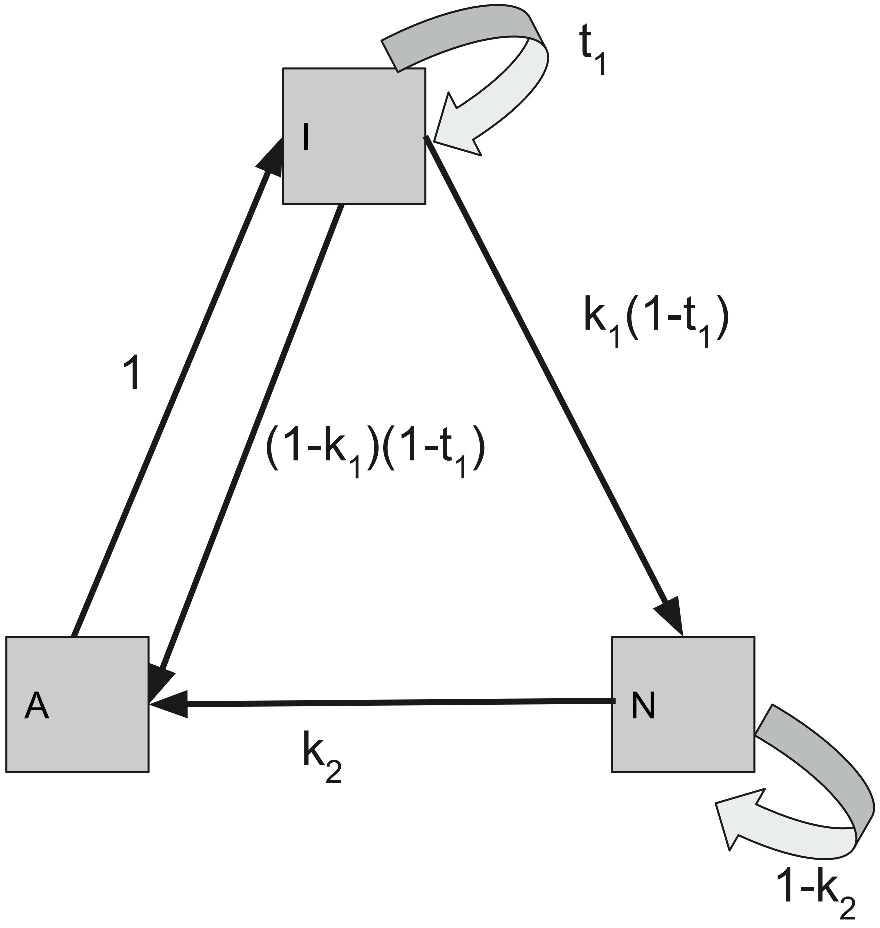
- Our model hopes to describe the effect punishment systems have towards reducing aggression within a population.
- We hope to see how changing the punishment system or the population would affect the equilibrium state of our population.

Rules of Model

- Closed population - no births or deaths over the time span
- Three distinct roles: Aggressors, Neutral, Incarcerated
- There is the ability to change
- Punishment for the aggressors was incarceration

Model

- *I* = incarcerated
- *A* = aggressive
- *N* = not-aggressive
- k_1 is the proportion of people leaving jail with corrected behavior
- k_2 is the proportion of neutral people who become aggressive
- t_1 is the proportion of people who stay incarcerated



Model (continued)

Solving for this Markov Chain's equilibrium point, where rates going in and out are equal, we get:

$$[k_1(1-t_1)]I = [k_2]N$$

$$[1]A = [1-t_1]I$$

This gives us:

$$I = \frac{1}{1+(1-t_1)+\frac{k_1}{k_2}(1-t_i)}$$

$$A = \frac{1-t_1}{1+(1-t_1)+\frac{k_1}{k_2}(1-t_i)}$$

$$N = \frac{\frac{k_1}{k_2}(1-t_1)}{1+(1-t_1)+\frac{k_1}{k_2}(1-t_i)}$$

Simulations

We implemented our simulation through code, and using 100 iterations with the initial sample variables of:

- I = 0
- A = 100
- N = 900

What would happen if people stayed in jail longer? Or if jail better corrected behavior? Or our population was more aggressive?

- Varying values for t_1, k_1, and k_2
- Everything else constant

Simulations (continued)

```
# simulator
def sim(iter, I_p, A_p, N_p, t_1, k_1, k_2): # initial I, A, N values
  history = [(I_p,A_p,N_p)]
  I_res=0
  A_res=0
  N_res=0

  # running simulation for given iterations
  for itr in range(iter):
    I_res,A_res,N_res=0,0,0
    # running random individually for each I
    for i in range(I_p):
      r = np.random.rand() # [0,1)
      if r<1.0*t_1:      # 1/2
        I_res+=1
      elif r<t_1+k_1*(1-t_1):  # 1/4
        N_res+=1
      elif r<1.0:   # 1/4
        A_res+=1
    # running random individually for each N
    for n in range(N_p):
      r = np.random.rand() # [0,1)
      if r<1-k_2:   # 9/10
        N_res+=1
      elif r<1.0: # 1/10
        A_res+=1
    # All A goes to I
    I_res += A_p

    I_p=I_res
    A_p=A_res
    N_p=N_res
    history.append((I_p,A_p,N_p))
  return history
```

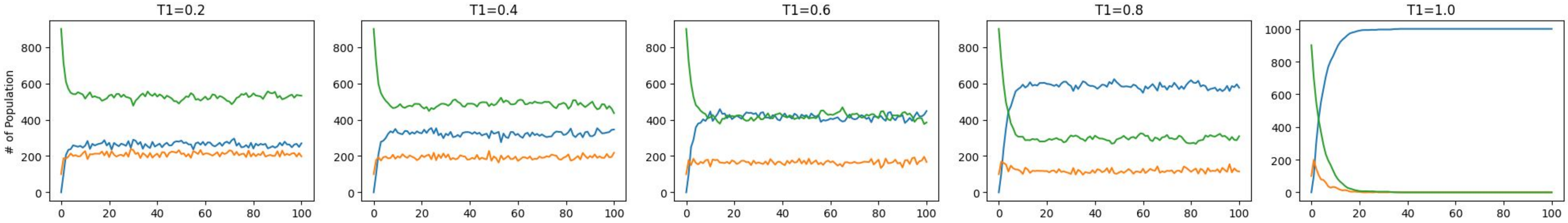
Simulations (continued)

```python
I = 0     # incarcerated
A = 100   # aggressive
N = 900   # neutral

t = 100   # time (simulation iteration)

default_k_1=0.5 # rate of incarcerated becoming neutral
default_k_2=0.2 # rate of neutral becoming aggressor
default_t_1=0.5 # rate of staying incarcerated

plt.figure(figsize=(20, 9))  # Adjusted figure size for 3x5 subplots

# We'll start by plotting the T1 variations in the first row
t_values = [0.2, 0.4, 0.6, 0.8, 1.0]
k1_values = [0.2, 0.4, 0.6, 0.8, 1.0]
k2_values = [0.2, 0.4, 0.6, 0.8, 1.0]

# Plot for T1 variations
for j, t_1 in enumerate(t_values, start=1):
    history = sim(t, I, A, N, t_1, default_k_1, default_k_2)
    I_value = [item[0] for item in history]
    A_value = [item[1] for item in history]
    N_value = [item[2] for item in history]
    indices = list(range(len(history)))

    plt.subplot(3, 5, j)
    plt.plot(indices, I_value, label='Population I')
    plt.plot(indices, A_value, label='Population A')
    plt.plot(indices, N_value, label='Population N')
    plt.title(f'T1={t_1:.1f}')
    if j == 1:  # Add y-label to the first column
        plt.ylabel('# of Population')
```

```python
# Plot for K1 variations
for j, k_1 in enumerate(k1_values, start=1):
    history = sim(t, I, A, N, default_t_1, k_1, default_k_2)
    I_value = [item[0] for item in history]
    A_value = [item[1] for item in history]
    N_value = [item[2] for item in history]
    indices = list(range(len(history)))

    plt.subplot(3, 5, j + 5)  # Offset by 5 to start on the second row
    plt.plot(indices, I_value, label='Population I')
    plt.plot(indices, A_value, label='Population A')
    plt.plot(indices, N_value, label='Population N')
    plt.title(f'K1={k_1:.1f}')
    if j == 1:  # Add y-label to the first column
        plt.ylabel('# of Population')
```

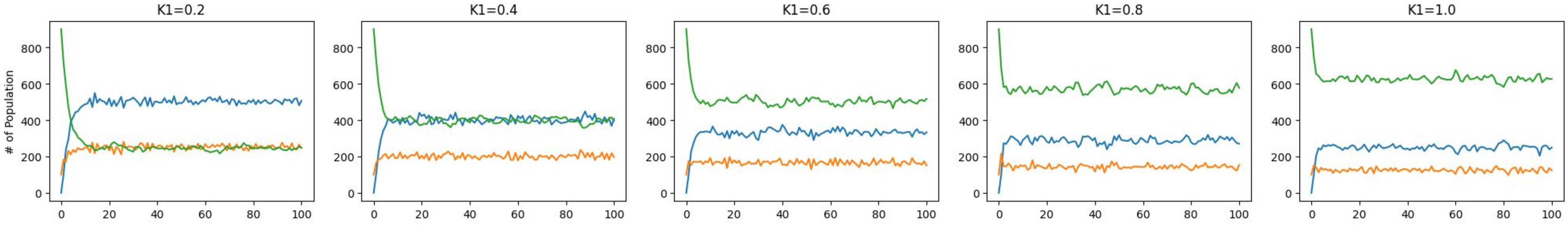
```python
# Plot for K2 variations
for j, k_2 in enumerate(k2_values, start=1):
    history = sim(t, I, A, N, default_t_1, default_k_1, k_2)
    I_value = [item[0] for item in history]
    A_value = [item[1] for item in history]
    N_value = [item[2] for item in history]
    indices = list(range(len(history)))

    plt.subplot(3, 5, j + 10)  # Offset by 10 to start on the third row
    plt.plot(indices, I_value, label='Population I')
    plt.plot(indices, A_value, label='Population A')
    plt.plot(indices, N_value, label='Population N')
    plt.title(f'K2={k_2:.1f}')
    if j == 1:  # Add y-label to the first column
        plt.ylabel('# of Population')
    if j <= 5:  # Add x-labels to the bottom row
        plt.xlabel('Time')

# Adjust layout
plt.tight_layout()

# Now show the plot
plt.show()
```

Observations



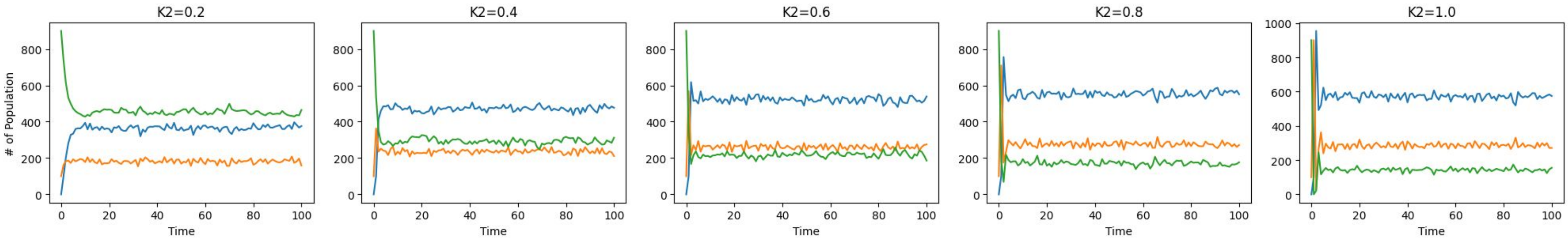
Key:
Neutral - Green
Incarcerated - Blue
Aggressors - Orange

Observations (continued)



Key:
Neutral - Green
Incarcerated - Blue
Aggressors - Orange

Observations (continued)



Key:
Neutral - Green
Incarcerated - Blue
Aggressors - Orange

Conclusion

- Within our model, effective punishments are ideal
- This means that punishment results in people changing and becoming less aggressive
- Results in the highest neutral population
- Furthermore, we want to avoid excessive punishment as this ultimately hurts the population
- Results in the lowest neutral population