MAT1630

Colin Gibbons-Fly

Bisection Method to Find Interest Rate

Rent-a-Center advertises renting an Amana refrigerator for \$112.61 per month for 18 months. You will own the refrigerator when you make all 18 payments. At the time of writing, the same refrigerator can be purchased from Lowes for \$649 with free delivery. Ignore the sales taxes in both cases. The goal of this activity is to determine the unknown interest rate that is associated with this loan. Use the Bisection Method or Newton's Method to determine the interest rate (to within an error of 0.0001) being charged in order to rent this refrigerator. Assume that the interest rate is compounded monthly. For simplicity, ignore any sales taxes. How much would you have to pay to Rent-A-Center in the form of interest relative to the cost of purchasing the refrigerator from Lowes?

```
# Given values
P0 = 649  # Lowes amount
c = 112.61  # Monthly payment
t = 18  # time periods
n = 12  # compounding schedule
# Define a Python function for the loan balance
def loan_balance(r,P0,n,t,c):
  Pnew = P0*(1+(r/n))**t - c*((1+r/n)**t  - 1)/(r/n)
  return Pnew
# Run the bisection method to find the interest rate (r)
def bisection(a,b,f,err):
  r = (b+a)/2
  while r-a >= err:
    if f(r,P0,n,t,c)==0:
      return r
    elif f(a,P0,n,t,c)*f(r,P0,n,t,c)>0:
      a=r
    else:
      b=r
    r = (b+a)/2
  return r
# Call bisection with the loan_balance function
result = bisection(0.001, 2, loan_balance, 0.0001)
print(f"Interest Rate = : {result:.2%}")
```

```
Interest Rate = : 194.22%
```

Coin Tosses

You flip a fair coin repeatedly. Using simulation, estimate the average number of coin tosses to get 4 consecutive heads.

```
import numpy as np
#Create function
def consecutive():
  #intiate variables for number of tosses and times it lands on heads
  tosses=0
  heads=0
  while heads < 4: #Flip coins until 4 heads in a row
    outcome = np.random.choice([1,0])
    tosses += 1 #Add 1 to your tosses tracker for every flip
    if outcome == 1:
      heads += 1 #If heads add 1 to tracker
    else:
      heads = 0
  return tosses
#Return MC simulation
sample = [consecutive() for i in range(10000)]
np.mean(sample)
```

```
29.4137
```

The Monty Hall Problem

In a popular game show in the 1970s , the host shows you a wall with three closed doors. Behind one of the doors is a fancy car, and behind each of the other two is a goat. The three doors all have equal chances of hiding the car. The host knows exactly what is behind each of the three doors. First, you choose a door without opening it, knowing that after you have done so, the host will open one of the two remaining doors to reveal a goat. When this has been done, you will be given the opportunity to switch doors and win whatever is behind the door you choose.

- What is the probability to win the car if you never switch doors? If you always switch doors?

```python
import numpy as np
def monty_hall_stay():
    # Setting stage
    doors = np.random.choice(['goat', 'goat', 'car'], size=3, replace=False)
    # First contestant guess
    guess = np.random.choice(3)
    # Host opens door
    host_open = np.random.choice([i for i in range(3) if i != guess and doors[i] != 'car']
    # Returns 1 or 0 depending on guess
    if doors[guess] == 'car':
        return 1
    elif doors[guess] == 'goat':
        return 0
# Run simulation
sample = [monty_hall_stay() for i in range(10000)]
stay_result = np.mean(sample)
print(f"Probability of winning by staying: {stay_result:.2%}")
```

```
Probability of winning by staying: 34.01%
```

```python
import numpy as np
def monty_hall_switch():
    # Setting stage
    doors = np.random.choice(['goat', 'goat', 'car'], size=3, replace=False)
    # First contestant guess
    guess = np.random.choice(3)
    # Host opens door
    host_open = np.random.choice([i for i in range(3) if i != guess and doors[i] != 'car']
    # Contestant switches guess
    guess = [i for i in range(3) if i != guess and i != host_open]
    # Returns 1 or 0 depending on guess
    if doors[guess] == 'car':
        return 1
    elif doors[guess] == 'goat':
        return 0
```

```
# Run simulation
sample = [monty_hall_switch() for i in range(10000)]
switch_result = np.mean(sample)
print(f"Probability of winning by switching: {switch_result:.2%}")
```

```
Probability of winning by switching: 67.25%
```

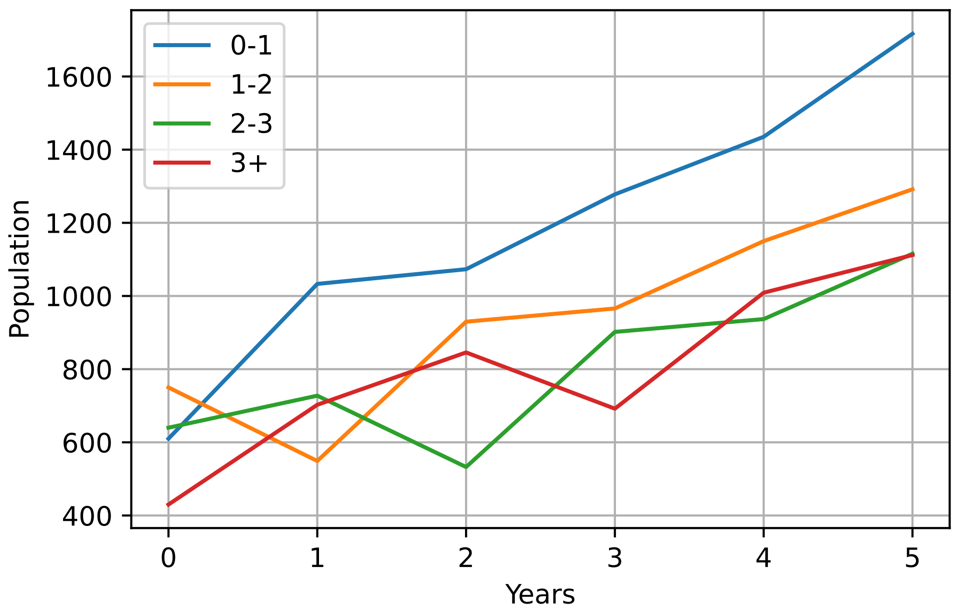
Discrete Modeling

The goal of this project is to model the population of rats at a local park in 2025 using the table of birth and survival rates given below. The average rat can live up to the age of 4. Below is a table of birth and survival rates for rats observed at the local park. Estimate the total population of rats, across all age groups, at the local park in 2025 using the Table above, using a discrete-time population model. The total rats population in 2025 should be about 6056 rats of all age groups.

Age (years)	Birth Rate	Survival Rate	Population in 2019
0-1	0.00	0.90	610
1-2	0.65	0.97	750
2-3	0.55	0.95	640
3+	0.45	0.22	430

```
import numpy as np
import matplotlib.pyplot as plt
plt.clf()
# Given data
t = 2025-2019
r1 = np.empty(t)
r2 = np.empty(t)
r3 = np.empty(t)
r4 = np.empty(t)
#Rat populations
r1[0] = 610  # rats 0-1 pop
r2[0] = 750  # rats 1-2 pop
r3[0] = 640  # rats 2-3 pop
r4[0] = 430  # rats 3+ pop
#Birth Rates
r1br = 0.00  # r1 birth rate
```

```
r2br = 0.65  # r2 birth rate
r3br = 0.55  # r3 birth rate
r4br = 0.45  # r4 birth rate
#Survival Rates
r1sr = 0.90  # r1 survival rate
r2sr = 0.97  # r2 survival rate
r3sr = 0.95  # r3 survival rate
r4sr = 0.22  # r4 survival rate
# Evolution of the system using a for loop
for i in range(t-1):
    r1[i+1] = r2br*r2[i] + r3br*r3[i] + r4br*r4[i]
    r2[i+1] = r1sr*r1[i]
    r3[i+1] = r2sr*r2[i]
    r4[i+1] = r3sr*r3[i] + r4sr*r4[i]
#Plot graph
plt.plot(range(len(r1)),r1, label='0-1')
plt.plot(range(len(r2)),r2, label='1-2')
plt.plot(range(len(r3)),r3, label='2-3')
plt.plot(range(len(r4)),r4, label='3+')
plt.legend()
plt.grid()
plt.ylabel('Population')
plt.xlabel('Years')
plt.show()
```


0-1
1-2
2-3
3+
Population
Years
1600
1400
1200
1000
800
600
400
0
1
2
3
4
5