EE 325: Assignment 2

Ashish Prasad 21d180009
Shrey Agarwal 21d170039
Akhilesh Chauhan 21d070010

September 6, 2022

1 QUESTION 1

1.1 Approach

According to the question, we have been given n fishes in the lake, from these n fishes, we caught m fishes and marked them and then we mix them well with other fishes. After mixing them with other fishes, we caught only p fishes out of this m.

$$n \geqslant p \text{ and } p \geqslant m$$

m-p fishes that were caught the 2nd time were unmarked. This means that there were at least m-p unmarked fish. So

$$n - m \geqslant m - p \text{ or,}$$
$$n \geqslant 2m - p$$

The probability of the event for a fixed p recatches out of m coming from n fishes in the lake(P(X)) is 0 for

$$n < m \ , \ m < p \ , \ n < 2m - p$$

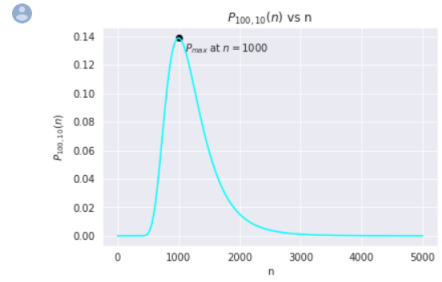
No. Of ways to have p recatches given m fish were caught = $\binom{m}{p}.n - \binom{m}{m} - p$
No of ways to catch m fish from the pond = $\binom{n}{m}$
P(X)=

$$P_{m,p}(n) = 0 \text{ if } n < \min\{m, 2m - p, 0\} \text{ or } p < m$$
$$P_{m,p}(n) = \frac{\binom{m}{p}.\binom{n-m}{m-p}}{\binom{n}{m}} \text{ otherwise}$$

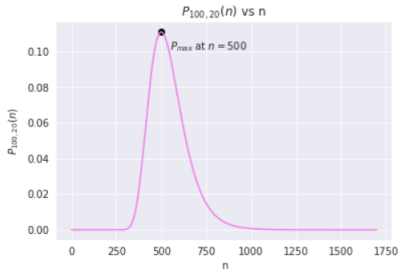
1.2 Graphs

Now, we will plot the graphs for m = 100, p = 10, 20, 50, 75:

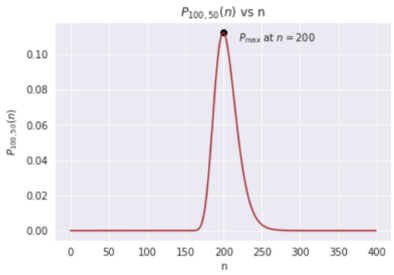
```
#sns.lineplot(N, p)
plt.xlabel('n')
plt.ylabel('$P_{100,10}(n)$')
plt.title('$P_{100,10}(n)$ vs n')
plt.plot(N1, p10,color='cyan')
plt.scatter([1000], [prob(100, 10, 1000)], marker='o',color ='black')
plt.annotate('$P_{max}$ at $n = 1000$', (1100, prob(100,10, 1000)-0.01))
plt.savefig('p10.pdf')
plt.show()
```



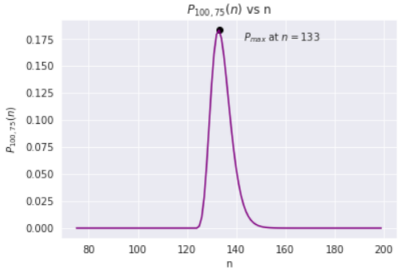
```
#sns.lineplot(N, p)
plt.xlabel('n')
plt.ylabel('$P_{100,20}(n)$')
plt.title('$P_{100,20}(n)$ vs n')
plt.plot(N2, p20, color='violet')
plt.scatter([500], [prob(100, 20, 500)], marker='o', color='black')
plt.annotate('$P_{max}$ at $n = 500$', (550, prob(100,20, 500)-0.01))
plt.savefig('p20.pdf')
plt.show()
```




```
#sns.lineplot(N, p)
plt.xlabel('n')
plt.ylabel('$P_{100,50}(n)$')
plt.title('$P_{100,50}(n)$ vs n')
plt.plot(N3, p50, color='brown')
plt.scatter([200], [prob(100, 50, 200)], marker='o', color='black')
plt.annotate('$P_{max}$ at $n = 200$', (220, prob(100,50, 200)-0.005))
plt.savefig('p50.pdf')
plt.show()
```




```
[ ] #sns.lineplot(N, p)
    plt.xlabel('n')
    plt.ylabel('$P_{100,75}(n)$')
    plt.title('$P_{100,75}(n)$ vs n')
    plt.plot(N4, p75, color='purple')
    plt.scatter([133], [prob(100, 75, 133)], marker='o', color='black')
    plt.annotate('$P_{max}$ at $n = 133$', (143, prob(100,75, 133)-0.01))
    plt.savefig('p75.pdf')
    plt.show()
```



2 QUESTION 2

2.1 Approach

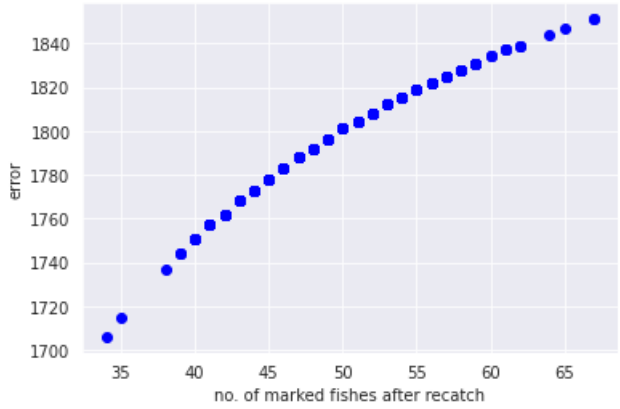
First of all, we have to assume n=2000 according to the question, after that we will create an array of 2000 integers. Then we will randomly catch 100 fishes and mark them and then again recatch fishes to find the marked recatch fishes (that is p) similarly as we did in the first question. Then we will find the best estimate of no. Of fishes as in the first question. Then we found the error in the estimate as defined in the question for 500 iterations. At last, I found the sample mean of the error and sample variance of the error.

```
import numpy as np
import statistics
import matplotlib.pyplot as plt
import array as arr
import random
import operator as op
from functools import reduce
import math
def ncr(n, r):
    r = min(r, n-r)
    numer = reduce(op.mul, range(n, n-r, -1), 1)
    denom = reduce(op.mul, range(1, r+1), 1)
    return (numer/denom)

p_values = []
n_values = []
error = []
for i in range(0, 500):
    arr = np.random.randint(2,size=(2000))
    arr_after_catch = np.array(np.random.choice(arr,size=100))
    p_values.append(sum(arr_after_catch))

for j in p_values :
    n1=np.array(range(200-j,2000 ))
    arr = np.array([(ncr(100,j)*ncr(i-100,100-j))/ncr(i,100) for i in n1 ] )
    max_index=np.argmax(arr)
    n_values.append(n1[max_index])
    error.append(abs(n1[max_index]-2000))

plt.xlabel('no. of marked fishes after recatch')
plt.ylabel('error')
plt.scatter(p_values, error, c ="blue")
print(statistics.mean(error))
print(statistics.variance(error))
```



3 QUESTION 3

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

value = np.array((1,-1,0)) #array containing values if the signal coming then 1, going out then -1, if nothing happens 0

weights = np.array((0.3,0.4,0.12)) #probability of each value


time = np.zeros(1000000) # array of time intervals

for t in range(0,999999):
    choice = rand.choices(value,weights,k=1) #generate value acc to weight
    time[t+1] += time[t]+choice[0] # if signal comes or goes at time = t then the value will change of the next time t+1.
    if time[t+1]<0:
        time[t+1] = 0 #making sure that the value if positive

fraction = np.zeros(50)

for k in range(0,50):
    for t in range(0,1000000):
        if k == time[t]: # if n is equal to the value in the time interval then add the instances.
            fraction[k] +=1

plt.plot(range(0,50),fraction)
plt.show()
avg = []

for i in range(0,50):
  avg.append (float(fraction[k])/1000000)
```