EE 325: Probability and Random Processes

Github repository link: https://github.com/rookie-apoorv/Assignment_2.git

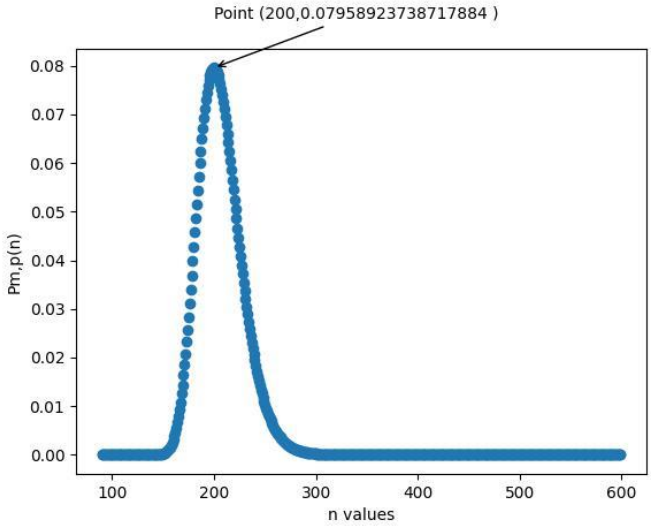
Question1 : Recall the capture-release-recapture problem: Catch m sh, mark them and release them back into the lake. Allow the sh to mix well and then you catch m sh. Of these p are those that were marked before. Assume that the actual sh population in the lakes is n and has not changed between the catches. Let Pmp(n) be the probability of the event (for a xed p recatches out of m) coming from n sh in the lake. Generate a plot for Pmp(n) as a function of n for the following values of m and p : m = 100 and p = 10205075 For each of these p use the plots to estimate (educated guess) the actual value of n i.e., what is the best guess for n if m = 100 and you catch p of the marked sh after mixing them up. Call these four estimates n1 n4 You de ne your notion of best guess. Do not search, THINK!

Approach : The above is nothing but a binomial random variable. We get $P_{m,p}(n)$ as follows:

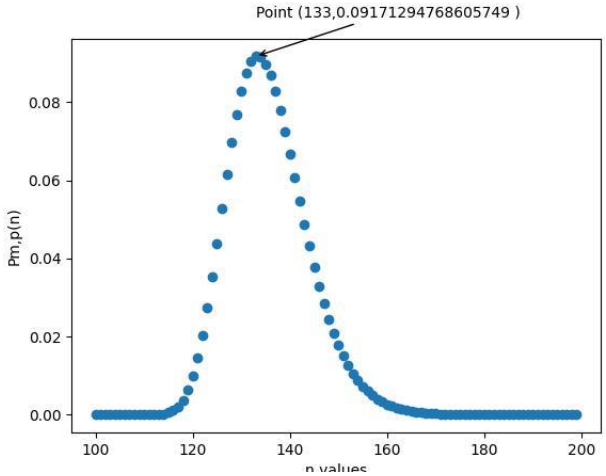
$$P_{m,p}(n) = \binom{m}{p}\left(\frac{m}{n}\right)^{p}\left(1-\frac{m}{n}\right)^{m-p}$$

Taking m = 100 and plotting this probability as a function of n for various p values we get:

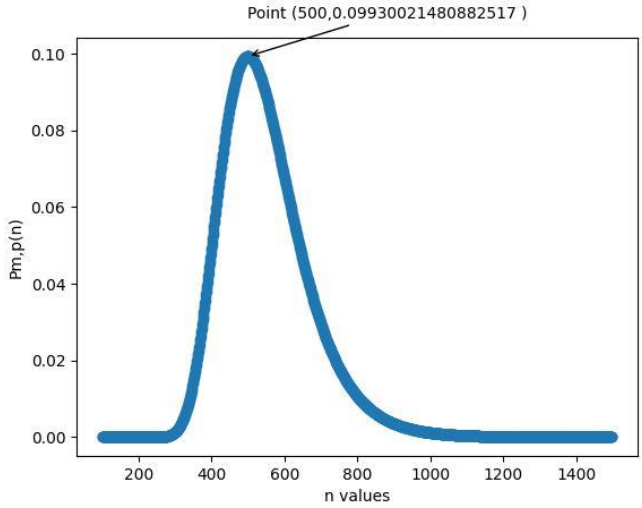
p = 50



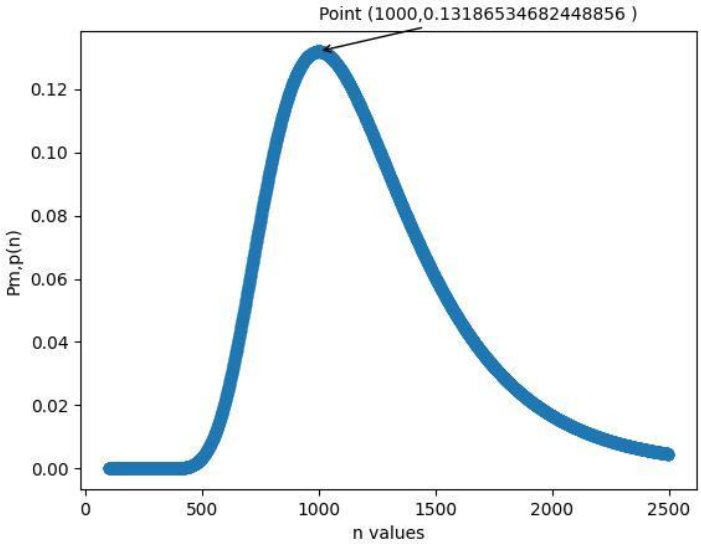
p = 75


Point (133,0.09171294768605749)
Pm,p(n)
0.00
0.02
0.04
0.06
0.08
100
120
140
160
180
200

P = 20


Point (500,0.09930021480882517)
Pm,p(n)
0.00
0.02
0.04
0.06
0.08
0.10
200
400
600
800
1000
1200
1400
n values

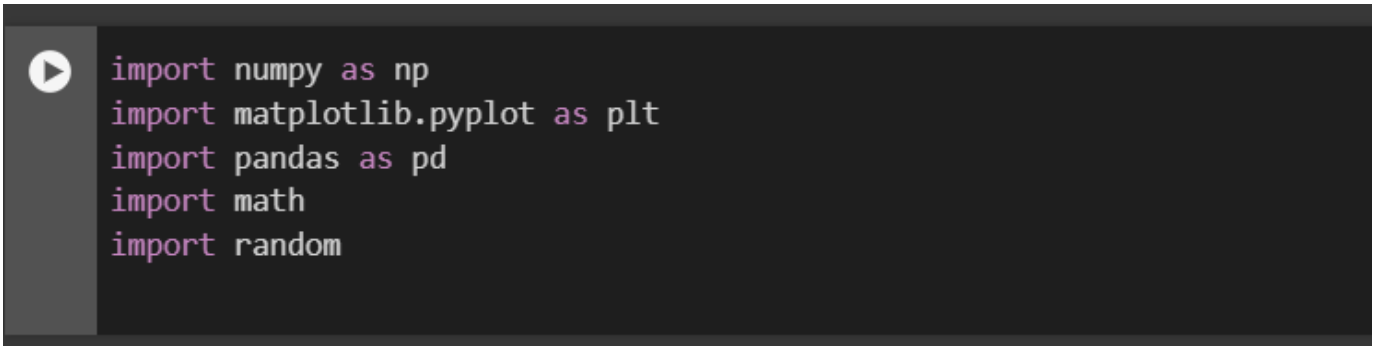
P = 10


Point (1000,0.13186534682448856)
Pm,p(n)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
500
1000
1500
2000
2500
n values

- According to us, the best guess for the value of n would be the value for which obtaining the corresponding p is most probable.
- We are using the notion that if an event has highest probability, it is most likely to happen.
- So, our guesses are :
    - N1 : 1000
    - N2 : 500
    - N3 : 200
    - N4 : 133

Question 2: Consider the following discrete time system. Packets arrive randomly at router to be transmitted on a link. The time between successive packets are independent geometric random variables with parameter λ. Packet transmission times are also random and have a geometric distribution with parameter μ. Only one packet can be transmitted at any time and packets that arrive during the transmission of a previously arrived packet wait in buffer memory. There is infinite memory and can accommodate any number of packets. This can be simulated as follows. At the beginning of each second, if there is a packet in the buffer, it leaves with probability μ. And a new packet is added to the queue with probability λ. Simulate this queue for 1,000,000-time steps. For n = 0, . . . 50, plot p(n), the fraction of time that there are n packets in the queue. Also find the time average of the number of packets in the memory. Use λ = 0.3 and μ = 0.4.

;;;; Note: There is one particular thing to care of when simulating this experiment, that is the fact that it is specified that at the start of each second you first check for already present packages and decide if they “leave” , the you decide whether or not you take in incoming packages, They both produce vastly different results.



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import random
```

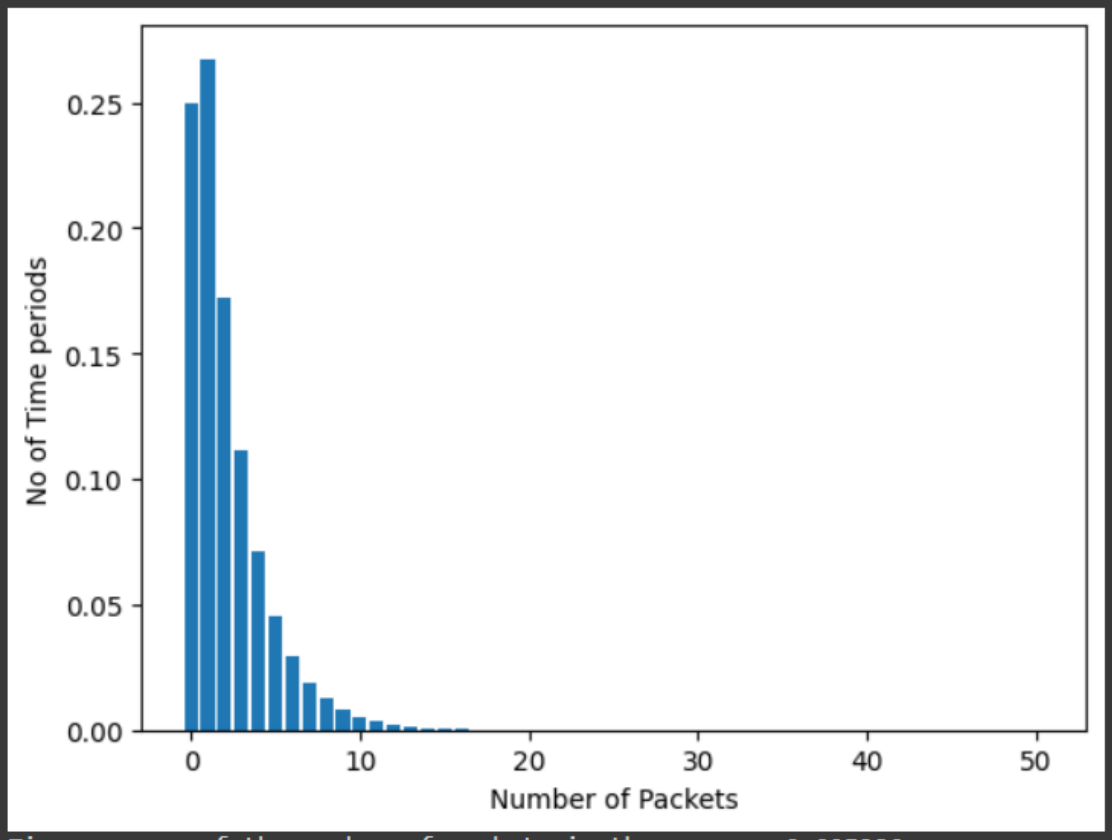
```
[2]
    lambda1 = 0.3
    mu = 0.4
    times_repeat = 1000000
    max_n = 50
    queue = 0
    queue_lengths = np.zeros(times_repeat)
```

```
[3] for t in range(times_repeat):
        if queue > 0 and np.random.rand() < mu :
            queue -= 1

        if np.random.rand() < lambda1:
            queue += 1

        queue_lengths[t] = queue
```

```
[4]
    p_n = np.zeros(max_n + 1)
    for n in range(max_n + 1):
        p_n[n] = np.sum(queue_lengths == n)/times_repeat


    average_queue_length = np.mean(queue_lengths)
```

The time average is being defined as the average number of packets in storage over any time period (1 second.) The time average is 2.095

```
[6]  plt.bar(range(max_n + 1), p_n)
     plt.xlabel("Number of Packets")
     plt.ylabel(" No of Time periods ")
     plt.show()

     print("Time average of the number of packets in the memory:" , average_queue_length )
```




```
Time average of the number of packets in the memory: 2.095286
```

The time average is 2.095

Question3 : Now extend the program from the previous part to simulate 10,000 queues in simultaneously parallel. When you stop the simulation after 100,000 time steps you have 10,000 values for the number of packets in the system. Use this data to plot p(n) the fraction of queues that have n packets in the system and calculate the sample average from this 10,000 samples.

The logic is the same. And since looking at the number of times it is run it can be also concluded the results should not differ much.

CODE:

```
[2]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     import math
     import random
```

```
[3]  lambda1 = 0.3
     mu = 0.4
     times_repeat = 100000
     num_queues = 10000
     max_n = 50
```

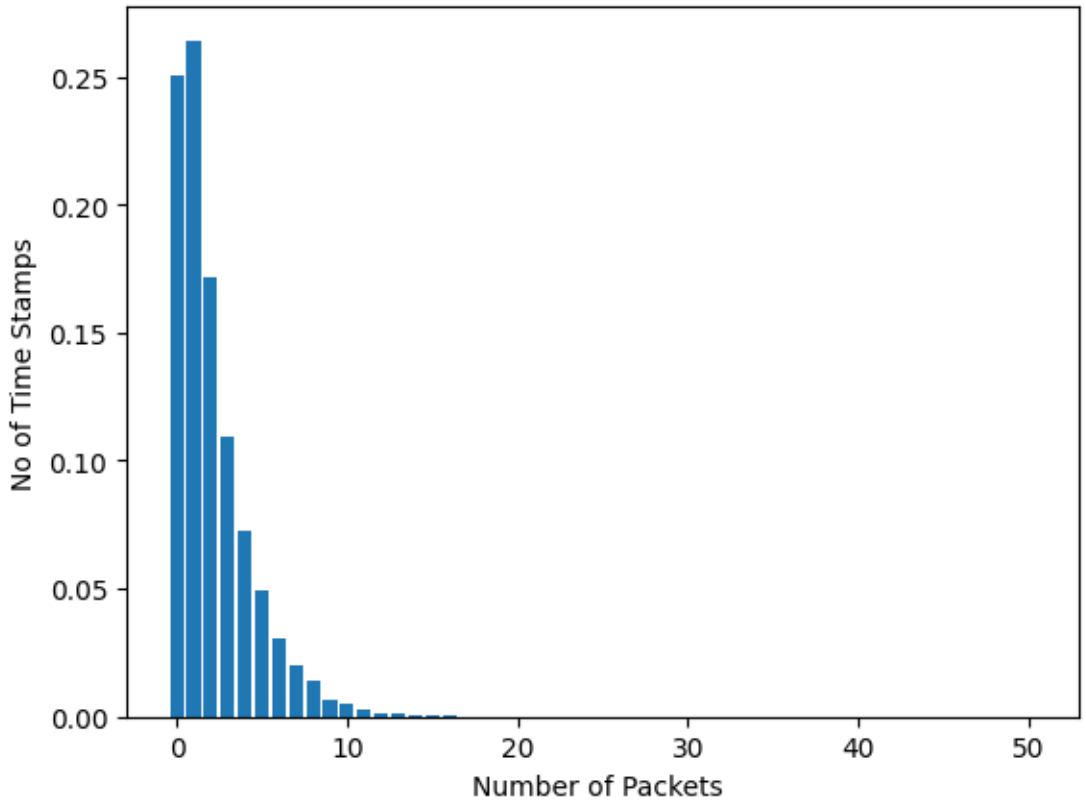
```
[4]
     queues = np.zeros(num_queues, dtype=int)
     final_len = np.zeros(num_queues)
```

```
[5]  for t in range(times_repeat):
         departures = (queues > 0) & (np.random.rand(num_queues) < mu)
         queues[departures] -= 1


         arrivals = (np.random.rand(num_queues) < lambda1)
         queues[arrivals] += 1
```

```
[6]
     final_len = queues.copy()

     p_n = np.zeros(max_n + 1)
     for n in range(max_n + 1):
         p_n[n] = np.sum(final_len == n)/ num_queues

     sample_average = np.mean(final_len)
```

```
[11] plt.bar(range(max_n + 1), p_n)
     plt.xlabel("Number of Packets ")
     plt.ylabel("No of Time Stamps")
     plt.show()

     print(sample_average)
```

The time average is being defined as the average number of packets in storage over any time period (1 second.) The time average is 2.102



Once again, the time average is 2.102.

Question4 : A jury of N members is to be constituted to decide on a complaint. In the population from which the jury has to be selected, each member makes the correct (fair) decision with probability (05 + c) 005 c 025 The majority rule is applied, i.e., all members vote yes/no and the majority vote is the decision of the panel. If the probability of the correct decision has to be at least 0.75, what combinations of c and N are feasible. Discretize c in steps of 0.01. You can assume N to be odd numbered integers. Repeat if the requirement is to be correct 90% of the time. Provide a short discussion of your ndings. Submit the plots for the combinations of c and N for the two correctness requirements.

Note that there is typically a cost involved with the choices of c and NA juror with a higher c is both rare and expensive. Similarly higher N makes the logistics of managing them complex. Suggest suitable cost functions that depend on c and N and comment on the right combination of c and N for each of the two preceding correctness requirements.

Approach – Let X be the random variable which denotes the number of people who gave a fair decision.

$$P(\text{Decision is correct }) = P(X > [\frac{N}{2}])$$

Given N is an odd integer, say N = 2M +1, where M is a non negative integer. Then

$$P(\text{Decision is correct }) = P(X > M)$$

Which can be given as

$$P(X > M) = \sum_{M+1}^{2M+1} P(X = i)$$

Where

$$P(X = i) = \binom{N}{i} p^i (1-p)^{N-i}$$

Where p = probability that a given member makes a fair decision, here

p = 0.5 + c , 0.05 <= c <= 0.25

We get

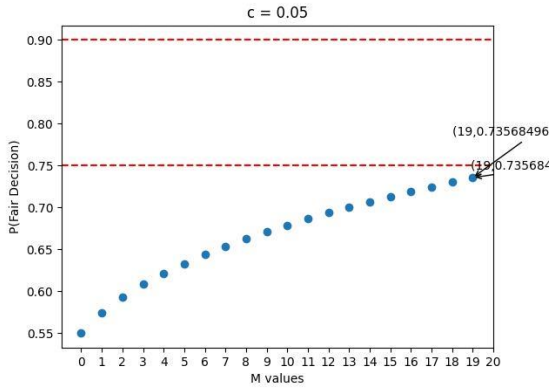
P(Fair Decision) =

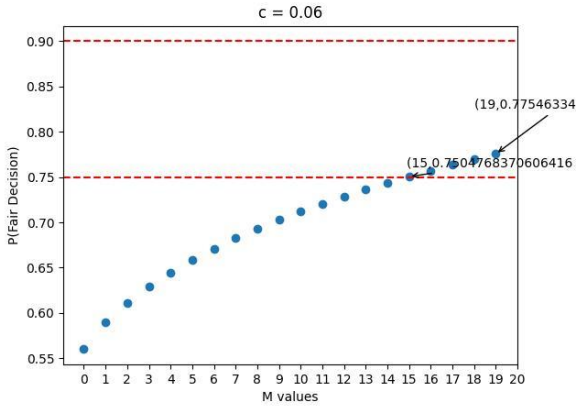
$$\sum_{[\frac{N}{2}]+1}^{N} \binom{N}{i} p^i (1-p)^{N-i}$$

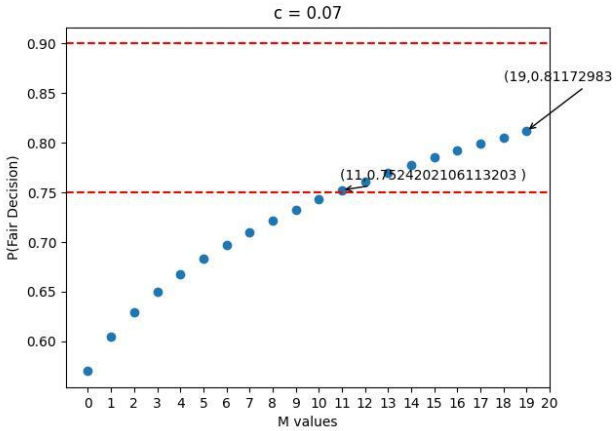
Now let's look at some real situations:

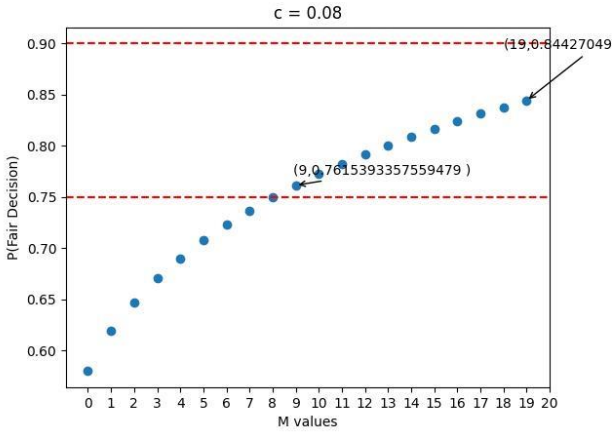
- Where do you think such kind of juries are required?
    - One immediate answer that comes to our mind is the DAC committee!

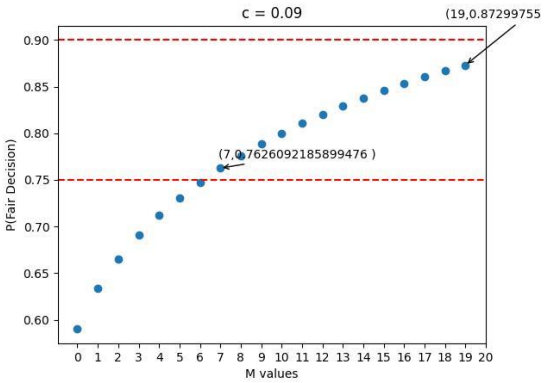
- Jokes apart, any authority which needs to decide on certain complaint need a jury, worst case the jury has only one member.
- A more important example that we see is jury in the Supreme Court.
- You might have often seen Supreme Court benches with 3, 5 or more judges.
- Why only these numbers?
- In many foreign countries they sometimes make juries of sizes more than 10/20.
- When is this done?
- Let us first look at some plots we have generated for the final fair decision made probability, for various values of c, then we will try to answer these questions.
- **NOTE:**
    - The x-axis in this graph is the values of M , where N = 2M+1. M can take all non negative integer values
    - The y-axis has the values of P(Decision is Fair).
    - The two red lines correspond to the probability being 0.75 and 0.9.
    - We have also labelled the points which are the closest to these two boundary lines in each plot, so we can get the corresponding values of N .

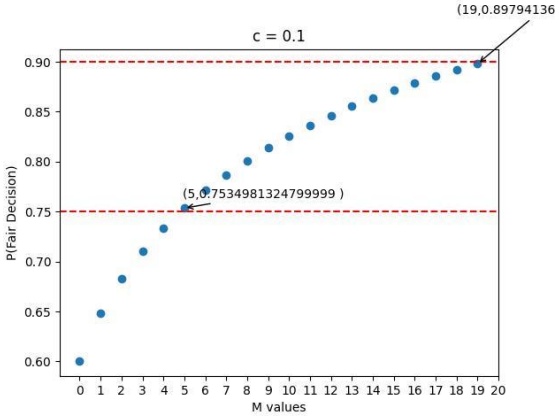


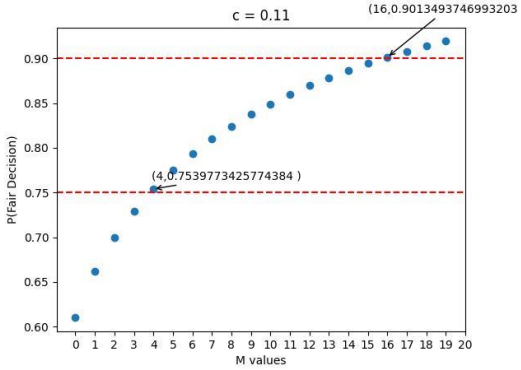


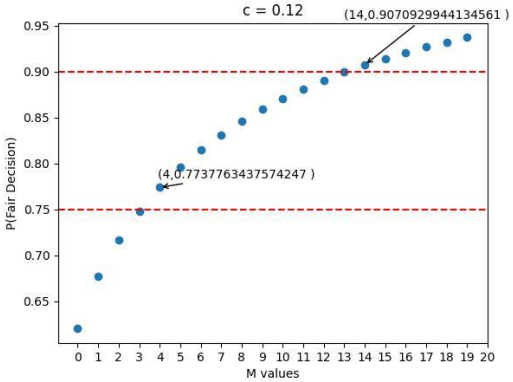


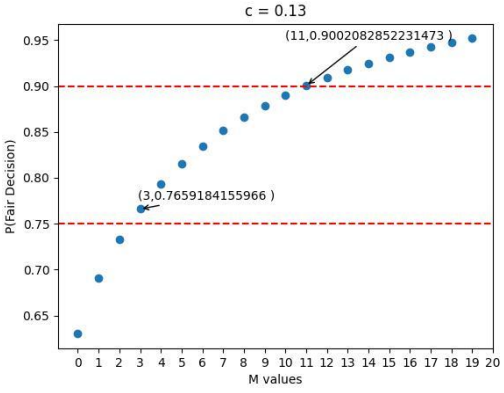


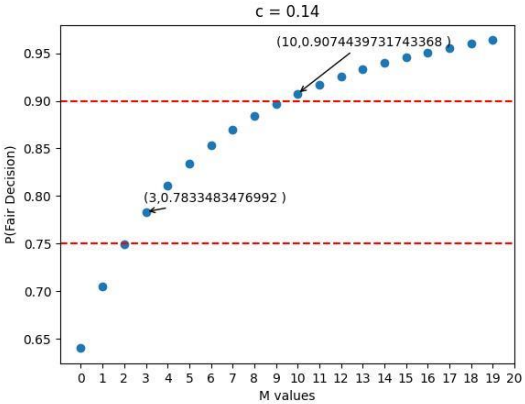

c = 0.09
(19,0.87299755
(7,0.7626092185899476)
P(Fair Decision)
M values

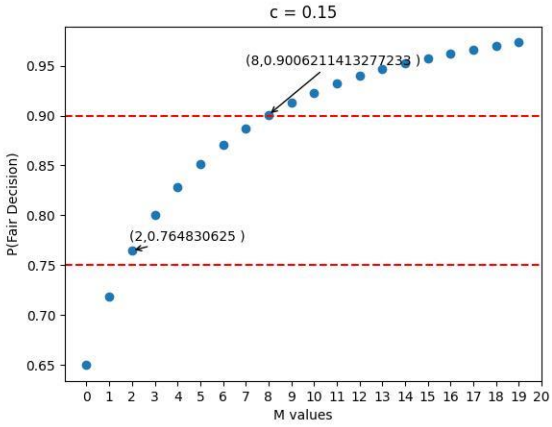

c = 0.1
(19,0.89794136
(5,0.7534981324799999)
P(Fair Decision)
M values

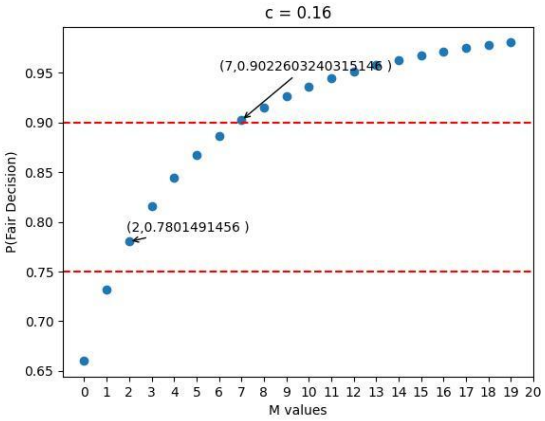

c = 0.11
(16,0.9013493746993203
(4,0.7539773425774384)
P(Fair Decision)
M values

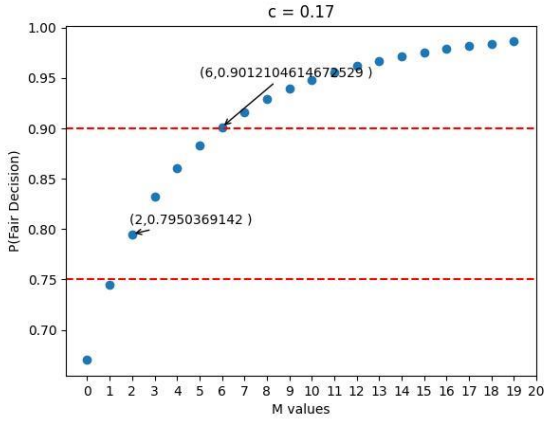

c = 0.12
(14,0.9070929944134561)
(4,0.7737763437574247)
P(Fair Decision)
M values

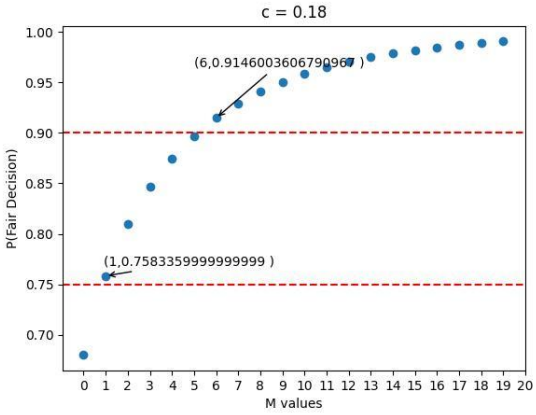

c = 0.13
(11,0.9002082852231473)
(3,0.7659184155966)
P(Fair Decision)
M values

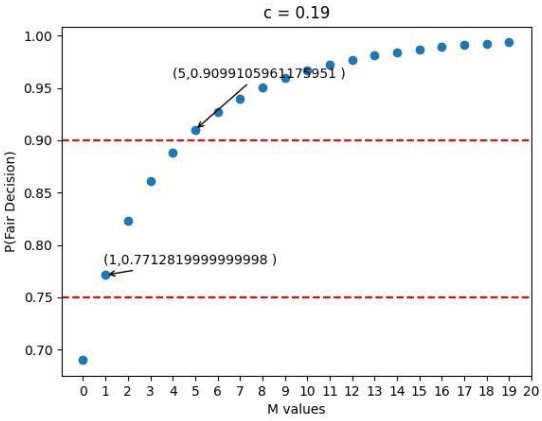

c = 0.14
(10,0.9074439731743368)
(3,0.7833483476992)
P(Fair Decision)
M values

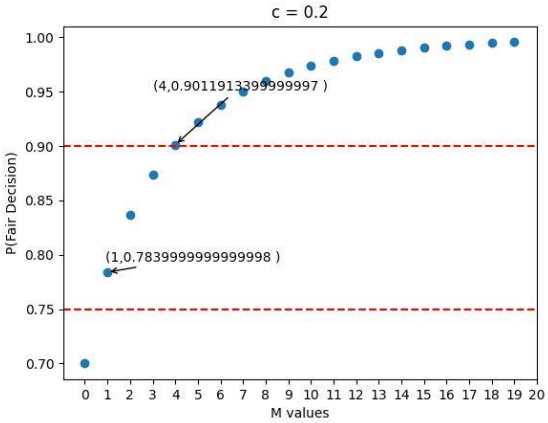

c = 0.15
(8,0.9006211413277233)
(2,0.764830625)
P(Fair Decision)
M values

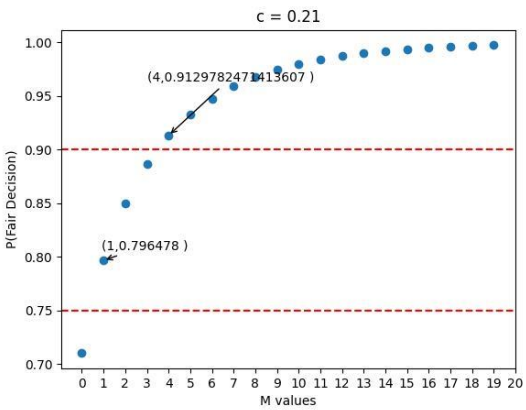

c = 0.16
(7,0.9022603240315146)
(2,0.7801491456)
P(Fair Decision)
M values

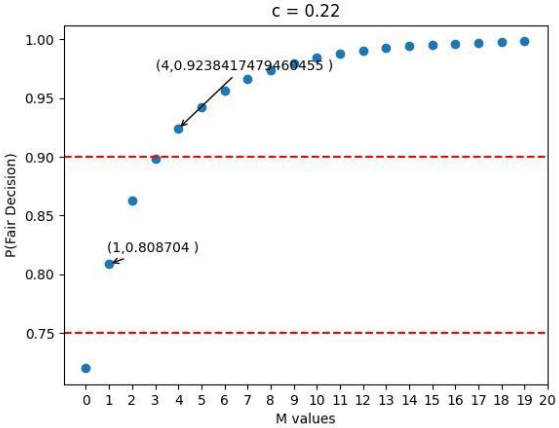

c = 0.17
(6,0.9012104614678529)
(2,0.7950369142)
P(Fair Decision)
M values

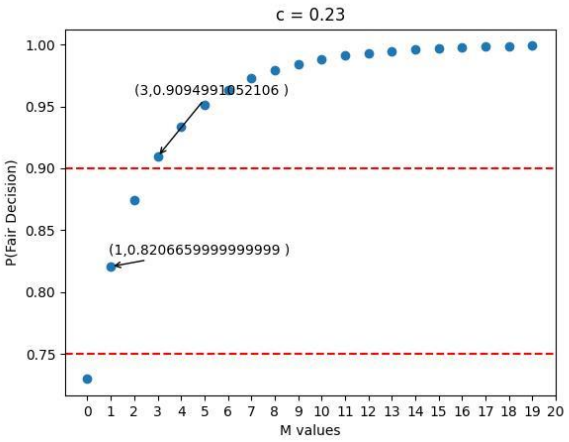

c = 0.18
(6,0.9146003606790967)
(1,0.7583359999999999)
P(Fair Decision)
M values

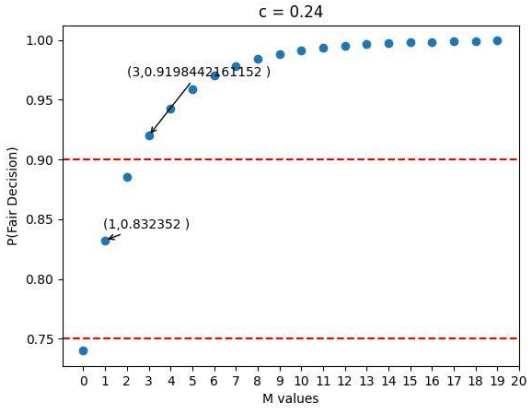

c = 0.19
(5,0.9099105961179351)
(1,0.7712819999999998)
P(Fair Decision)
M values

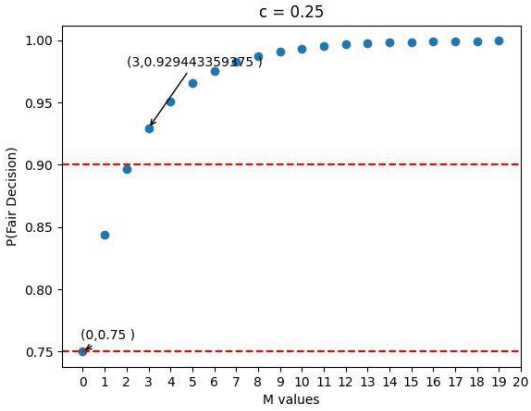

c = 0.2
(4,0.9011913999999997)
(1,0.7839999999999998)
P(Fair Decision)
M values


c = 0.21
(4,0.9129782471413607)
(1,0.796478)
P(Fair Decision)
M values


c = 0.22
(4,0.9238417479460455)
(1,0.808704)
P(Fair Decision)
M values







The following points are noteworthy:

- Why have we plotted M till 20 only, that is N till 41?
    - In a real situation we would not want N to be even 20 as it would greatly affect our operating cost. Hence, we are taking only these into considerations.

The following is a table to easily see the feasible values of c and N

NOTE: Here '–' means the values is greater than 41 (our max interest value).

C values	**Least N for P(fair decision)>0.75**	**Least N for P(fair decision)>0.90**
0.05	-	-
0.06	31	-
0.07	23	-
0.08	19	-
0.09	15	-
0.1	11	-
0.11	9	33
0.12	9	29
0.13	7	23
0.14	7	21
0.15	5	17
0.16	5	15
0.17	5	13
0.18	3	13
0.19	3	11
0.2	3	9
0.21	3	9
0.22	3	9
0.23	3	7
0.24	3	7
0.25	1	7

Some answers:

- Observe that for p = 0.5 + c = 0.75 , we need only 7 judges to get more than 90% probability of making a fair decision.
- We can safely assume (considering their experience and rigorous selection procedure) that a supreme court judge has a much higher value of c and consequently a higher probability of giving a fair decision. Hence the number of judges required in a supreme court bench can even be lesser than 7. And as we see there are benches with even only 3 judges.
- Why do then some juries have 10/20 people. Often in foreign countries a jury is selected randomly from the audience present inside the courtroom. We can not expect all of these randomly selected people to have a high c value, and hence we need more number of such people to increase the probability of making a fair decision.
- A trivia : The largest ever Supreme Court bench had 13 judges in it! Imagine the gravity of the situation, that they needed 13 judges, whom we can assume the c value to be more than 0.8, if we calculate it would mean a probability of around 0.9929964388352007! (p.s. that's not a factorial).

Cost function

- What should the cost function tell us?
    - Increasing c after a certain point should become increasingly tougher, and hence should increasingly affect our costs.
    - Increasing N should make a constant increase in our costs.
    - Also its tougher to increase c than N.
- Hence we make the following choice for the cost function:

$$\textbf{Cost} = a \cdot c^2 + b \cdot N$$

- Here a and b are constants.
- C^2 would be around 10^{-4} so we take a to be 10000, and say b = 10
    - Cost = $10000\ c^2 + 10\ N$

We get the following costs :

	N for P(fair decision)>0.75	N for P(fair decision)>0.90	Cost for P > 0.75	Cost for P > 0.90
0.05	-	-	-	-
0.06	31	-	34.6	-
0.07	23	-	27.9	-
0.08	19	-	25.4	-
0.09	15	-	23.1	-
0.1	11	-	21	-
0.11	9	33	21.1	45.1
0.12	9	29	23.4	43.4
0.13	7	23	23.9	39.9
0.14	7	21	26.6	40.6
0.15	5	17	27.5	39.5
0.16	5	15	30.6	40.6
0.17	5	13	33.9	41.9
0.18	3	13	35.4	45.4
0.19	3	11	39.1	47.1
0.2	3	9	43	49
0.21	3	9	47.1	53.1
0.22	3	9	51.4	57.4
0.23	3	7	55.9	59.9
0.24	3	7	60.6	64.6
0.25	1	7	63.5	69.5

As you can see for a cost of 60 you can take c =0.23 and N=7 or take lesser c say 0.16 and you can get a much larger N.