Assignment - 3

Farhan Alam 12340740

January 30, 2025

Question 1

1. A company is testing a new marketing strategy to attract potential customers. They run several campaigns and analyze the response rates to understand their effectiveness.

(a) Suppose the marketing team reaches out to 15 potential customers, and the probability of a positive response (conversion) from each customer is 20%. Plot the probability mass function (pmf) for the number of successful responses. This will help the team understand the likelihood of achieving different levels of success.
[Hint: You can use the inbuilt packages]

(b) The team wants to explore how different response probabilities might affect campaign performance. Repeat part (a) by considering probabilities of conversion as 10%, 20%, ..., 90%. Analyze and comment on how the distribution changes as the probability of a successful response increases.

(c) To measure long-term campaign performance, the team introduces a success rate metric defined as the number of successful responses divided by the total number of people contacted. Let:

$$Y = \frac{X}{n} \tag{1}$$

where X represents the number of successful responses from a campaign targeting n individuals, with a conversion probability of 5%. Generate

plots for campaigns targeting 10, 20, 50, and 200 customers. Discuss how the success rate distribution changes as the number of targeted customers increases and what this might imply for large-scale marketing efforts.

Introduction

The question asks us to take the probability of success as 0.2 and the plot the pmf for the same process repeated 15 times. Then we need to perform the same for porbabilities between (0.1 to 0.9) and then the random variable must be divided by the no.of customers we are reaching out to.

Data

Generated during the code consists of a list with the number of success and its probability.

Question 1(a)

Methodology

```
import math
from math import comb
import seaborn as sns
import pandas as pd
l=[]
for i in range(0,16):
  l.append([i,comb(15,i)*pow(0.2,i)*pow(0.8,15-i)])
df=pd.DataFrame(l,columns=['no.of success','probablity'
   ])
sns.barplot(df,x='no.of success',y='probablity')
```

AS it is self explanatory by the code given above we iteratively store the probability of each no. of success (which is between 0 to 15) and the number itself which is later plotted on the graph the probability distribution follows

the binomial distribution so the probability of each success is given by.

$$\binom{15}{x}(0.2)^x(0.8)^{15-x} \tag{2}$$

Then the list data is plotted using sns.barplot function.

Result

The result observed was as follows:

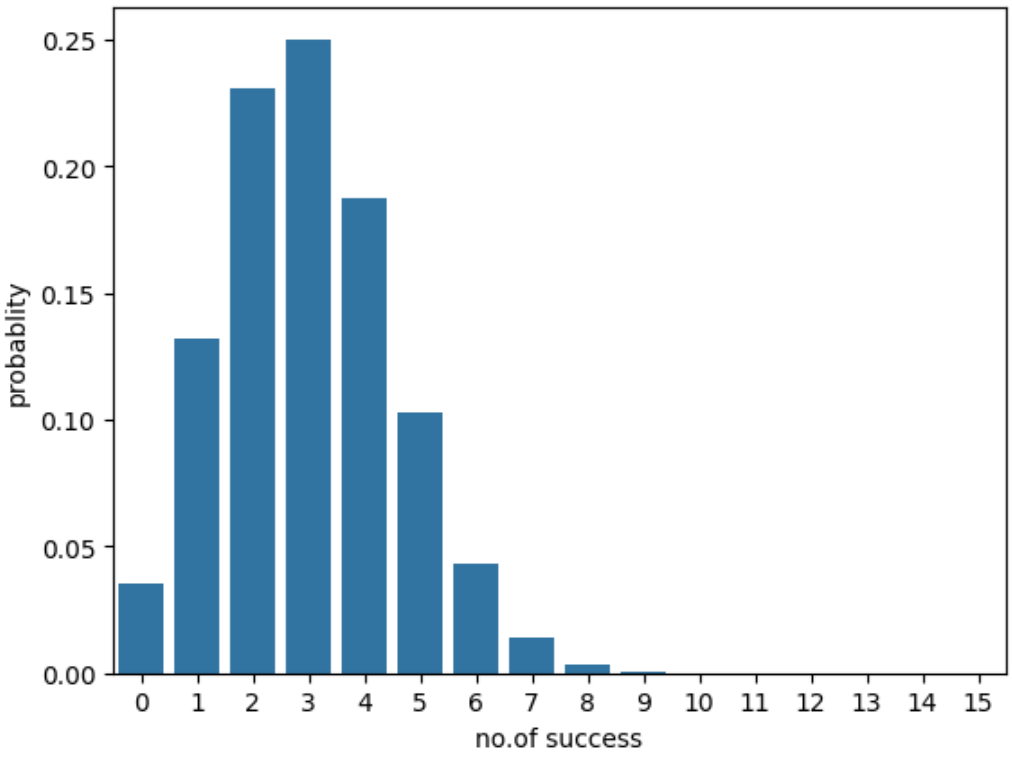

Figure 1:
From this we can infer that we have the maximum portability of success is at 3 and most of the max probabilities lies bear that which can also be proven by the mean of the binomial distribution which is given by n*p which results in 15*0.2=3 as the n value increase the probability nearly tends to zero the reason being the higher the power to a fraction the lower its value thus the result is minimal.

Question 1(b)

Data

Generated during the code just as produced for the above question but with varying values of the success probability from 0.1 to 0.9.

Methodology

The code used was as follows :-

```
1
2  j=0.1
3  import matplotlib.pyplot as plt
4  for k in range(0,8):
5    l=[]
6
7    for i in range(0,16):
8      l.append([i,comb(15,i)*pow(j,i)*pow(1-j,15-i)])
9    #s=comb(15,i)*pow(0.2,i)*pow(0.8,15-i)
10    #print(s)
11   df=pd.DataFrame(l,columns=['no.of success','
       probability'])
12   plt.figure()
13   sns.barplot(data=df, x='no.of success', y='probability
       ')
14   plt.title(f"Binomial Distribution (p = {j:.1f})")
15   plt.xlabel("No. of Successes")
16   plt.ylabel("Probability")
17   plt.figure()
18
19   j=j+0.2
20   if(j>0.9):
21     break
```

The Python script generates binomial distribution plots for different conversion probabilities. The key steps in the code are as follows:

1. Initialize the conversion probability j at 0.1.

2. Iterate through different probabilities from 0.1 to 0.8.

3. For each probability, compute the probability mass function (pmf) of the binomial distribution for 15 trials.

4. Store the computed values in a Pandas DataFrame for easier visualization.

5. Use Seaborn's barplot function to plot the distribution.

6. Display the plot for each probability and increment j by 0.1.

This process helps visualize how the probability distribution of successful responses shifts as the conversion probability changes.

1 Result

The result obtained were as follows:-

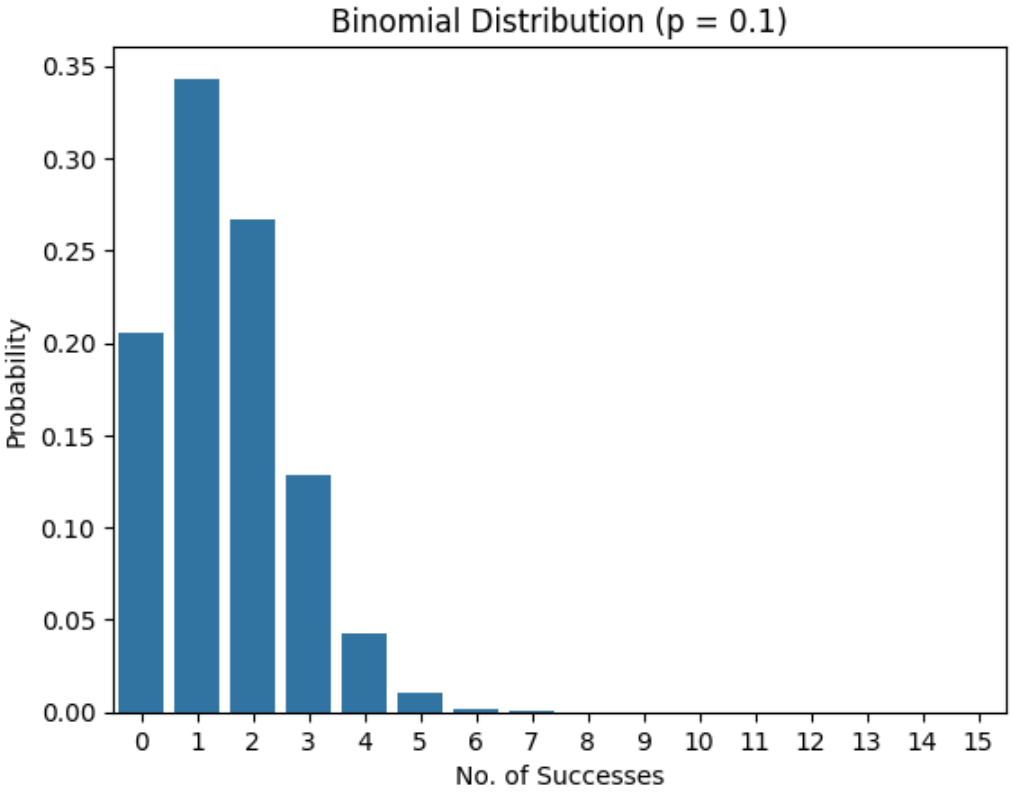
Binomial Distribution (p = 0.1)
Probability
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
No. of Successes

Figure 2:

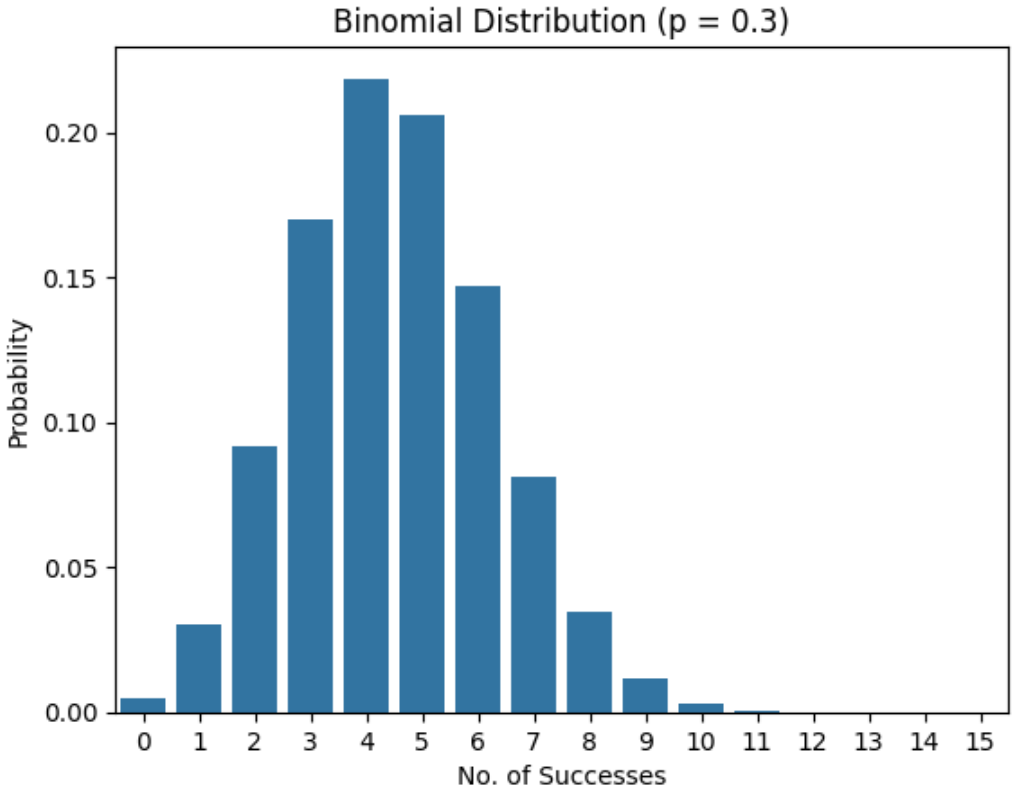
Binomial Distribution (p = 0.3)
Probability
0.00
0.05
0.10
0.15
0.20
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
No. of Successes

Figure 3:

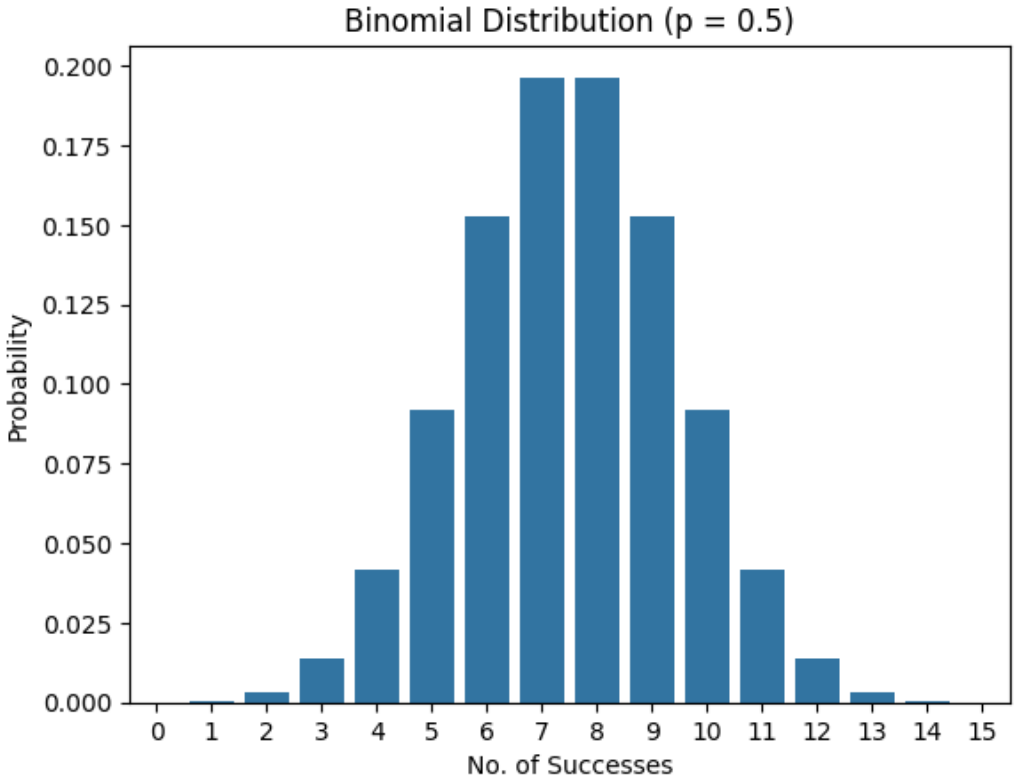
Binomial Distribution (p = 0.5)
Probability
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
No. of Successes

Figure 4:

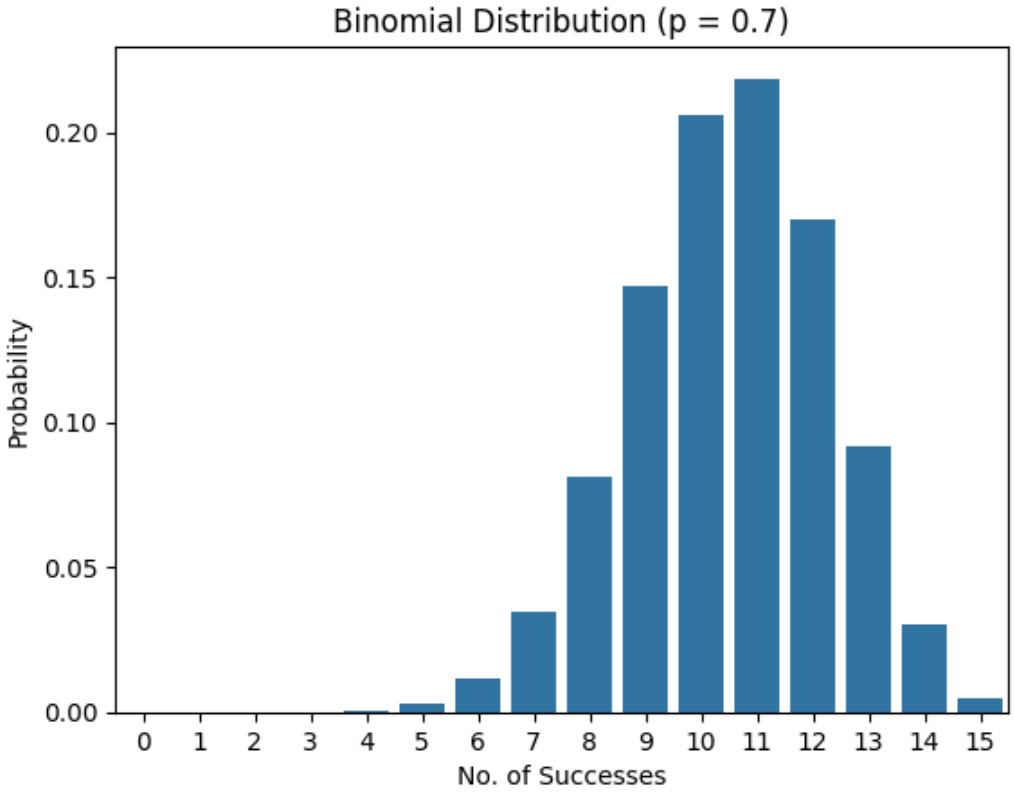
Binomial Distribution (p = 0.7)
Probability
0.00
0.05
0.10
0.15
0.20
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
No. of Successes

Figure 5:

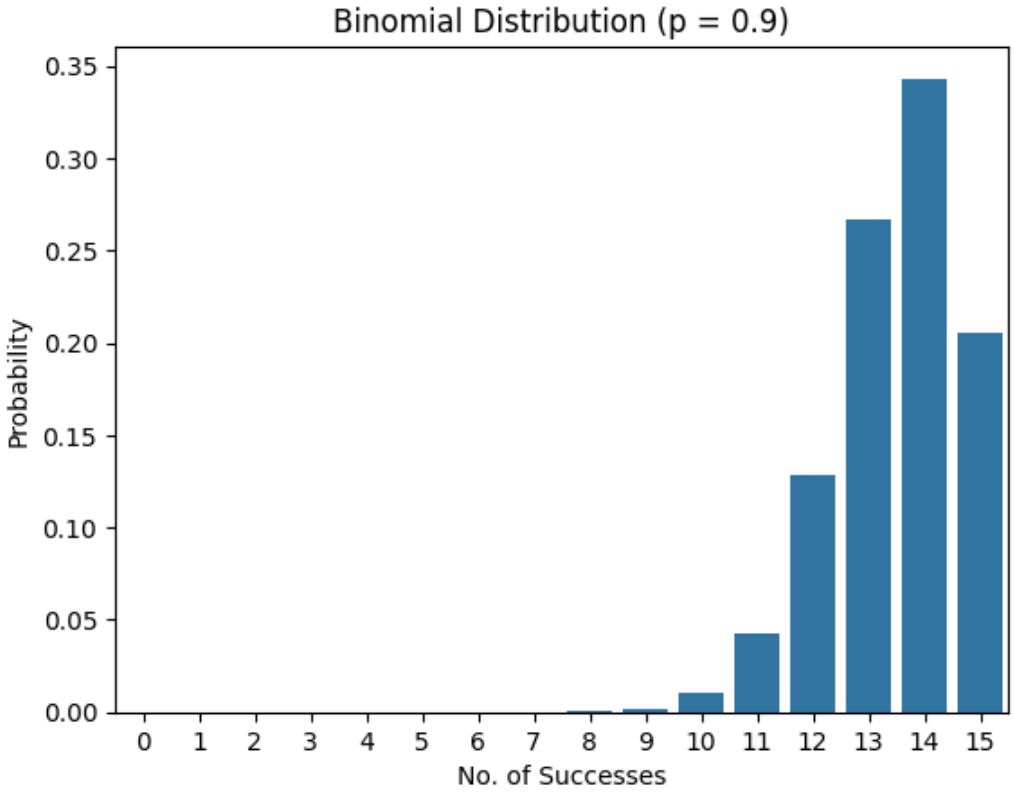

Figure 6:

As we can see as the value of p increases the max probabilities start to shift around the higher values of n where as when the porbability is low the max values are around the lower section of the numbers.This can also be explained by the mean of the distribution as the p increases the mean increases which results in the given shift.

2 Question 1(c)

3 Introduction

For this question we have value of p fixed at 0.05 the no.of people reached out to varies from [10,20,50,200] then the final probablity for each succes is normalized by the no. of people reached out to.

4 Data

Generated during the code just the porbablities are divided by n

5 Methodology

The code used is as follows:-

```
j=0.05
import matplotlib.pyplot as plt
ra=[10,20,50,200]
for k in ra:


  l=[]

  for i in range(0,k):
    l.append([i,(comb(k,i)*pow(j,i)*pow(1-j,k-i))/k])

  df=pd.DataFrame(l,columns=['no.of success','probablity
     '])
  sns.barplot(df,x='no.of success',y='probablity')
  plt.figure()
```

The Python script generates binomial distribution plots for different campaign sizes. The key steps in the code are as follows:

1. Initialize the conversion probability j at 0.05.
2. Define an array ra containing different campaign sizes: 10, 20, 50, and 200.
3. Iterate through each campaign size in ra.
4. For each campaign size k:
   - Compute the probability mass function (pmf) of the binomial distribution for k trials.
   - Store the computed values in a Pandas DataFrame for easier visualization.

- Use Seaborn's barplot function to plot the distribution of successful responses.

5. Display the plot for each campaign size.

This process helps visualize how the success rate distribution shifts as the number of targeted customers increases, providing insights into large-scale marketing efforts.

6 Result

The results were as follows:-

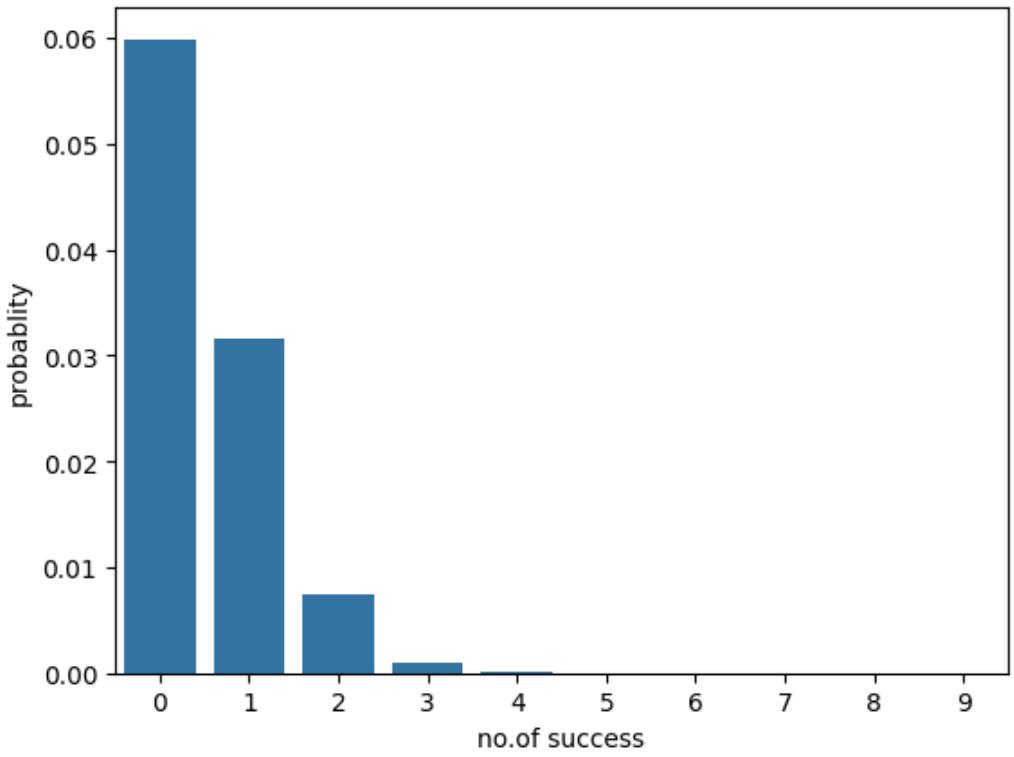

Figure 7:

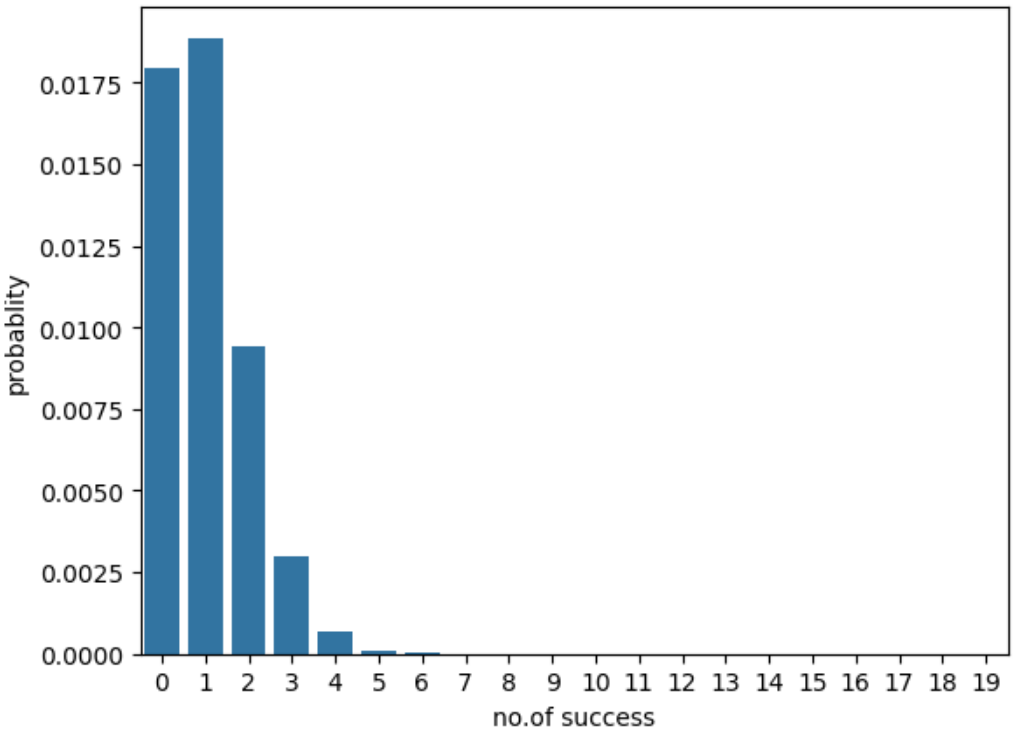
probablity
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
no.of success

Figure 8:

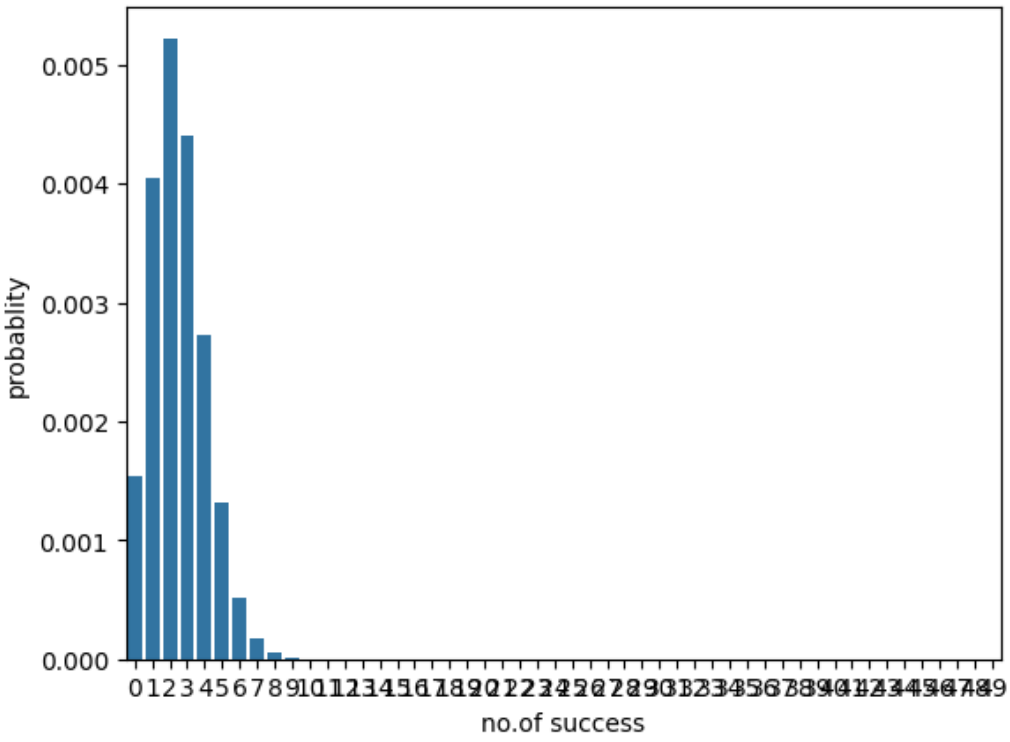

Figure 9:

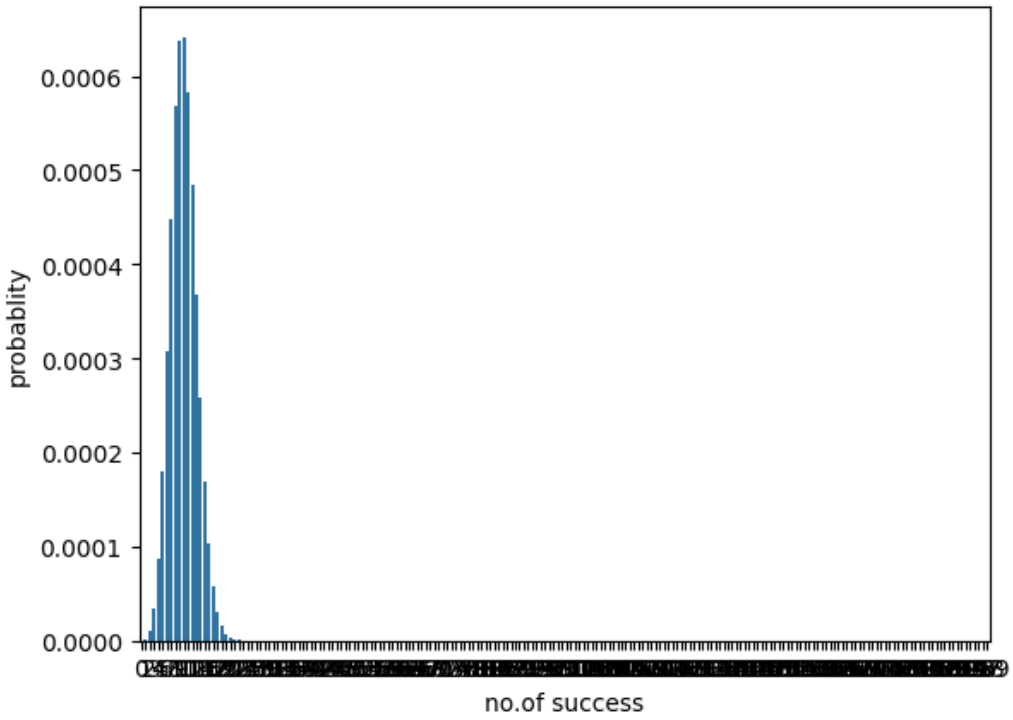

Figure 10:

The insights are similar to the on in question no.1 as we have only divided the total porbablity y the no. of customers thus the mean becomes equal to p which is 0.05 0.05 hence the majority data is centred around the numbers 0,1,2,3.

6.1 Question 2

The lifetime (in hours) of a certain battery used under extremely cold conditions follows a Gamma distribution with parameters:

- Shape parameter ($\alpha = 5$)
- Scale parameter ($\theta = 4$)

Using any programming language, answer the following questions:

(a) Calculate the average expected lifetime of the battery and its variability. Display the values.

(b) Find the median lifetime of the battery, representing the point at which half of the batteries are expected to last longer and half shorter. The median can be found using the inverse cumulative distribution function:

$$P(X \leq \text{median}) = 0.5$$

(c) Generate a plot of the probability density function (PDF) for battery lifetimes ranging from 0.1 to 50 hours to better understand the distribution of expected lifetimes.

(d) Repeat the same for the following cases of parameters:

(i) Shape parameter ($\alpha = 10$), Scale parameter ($\theta = 0.9$)

(ii) Shape parameter ($\alpha = 7$), Scale parameter ($\theta = 2$)

(iii) Shape parameter ($\alpha = 1.5$), Scale parameter ($\theta = 0.2$)

7 Introduction

The qusetion asks us to find the mean median for given paramteres of gamma function and draw its cdf too.We know that The expected lifetime (mean) of a Gamma distribution is given by:

$$E[X] = \alpha\theta \tag{3}$$

and its variance is given by:

$$Var(X) = \alpha\theta^2 \tag{4}$$

and then we will plot the pdf of each distribution.

8 Methodlogy

The code is as follows :-

```
import numpy as np
from scipy.stats import gamma

print(f"the mean is alpha*beta that is {5*4} ")
```

```
y=np.linspace(1,100,1000)
x=gamma.pdf(y,5,4)
z=gamma.ppf(0.5,5,scale=4)
print(f"the meadian is {z}")
y=np.linspace(0.1,50,1000)
x=gamma.pdf(y,5,scale=4)
plt.plot(y,x)
```

The code snippet provided uses the `numpy` and `scipy.stats` libraries to compute values related to the Gamma distribution.

1. **Mean Calculation:** The expected value (mean) of a Gamma distribution with shape parameter α and scale parameter θ is given by:

   $$\mu = \alpha \cdot \theta$$

   In the code, the mean is calculated as $5 \times 4 = 20$, and the result is printed using the `print` function:

   print(f"the mean is alpha*beta that is 5*4")

2. **Median Calculation:** The median of the Gamma distribution is calculated using the inverse cumulative distribution function (CDF), denoted as `ppf` in `scipy.stats.gamma`. The median represents the point at which half of the values of the random variable are expected to fall below and half above.

   $$P(X \leq \text{median}) = 0.5$$

   The median is calculated as:

   $$z = \text{gamma.ppf}(0.5, 5, \text{scale} = 4)$$

   The result is printed:

   print(f"the meadian is z")

3. **Plotting the PDF:** A range of values for the lifetime of the battery is generated using `np.linspace(0.1, 50, 1000)`, creating an array `y`

with 1000 points between 0.1 and 50. The probability density function (PDF) of the Gamma distribution is then calculated using:

$$x = \text{gamma.pdf}(y, 5, \text{scale} = 4)$$

The PDF is then plotted using `plt.plot(y, x)`. This visualizes the distribution of the battery lifetime for the given parameters ($\alpha = 5$, $\theta = 4$).

9 Result

The result obtained were as follows- the mean is alpha*beta that is 20 the meadian is 18.68363553118394 The pdf obtained was as follows

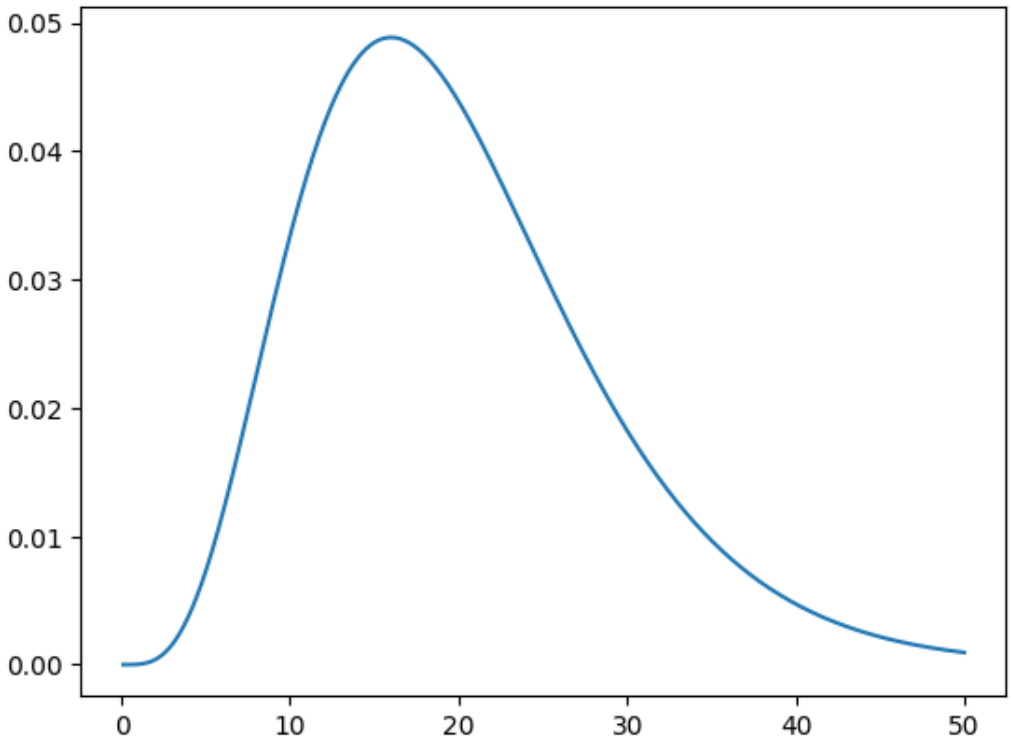

Figure 11:

Similar porcess was done for question 2(c) and the results obtained were as follows :

the mean is 9.0
the meadian is 8.701843153242715
the mean is 14
the meadian is 13.33927414909954
the mean is 0.30000000000000004]] the meadian is 0.23659738843753378

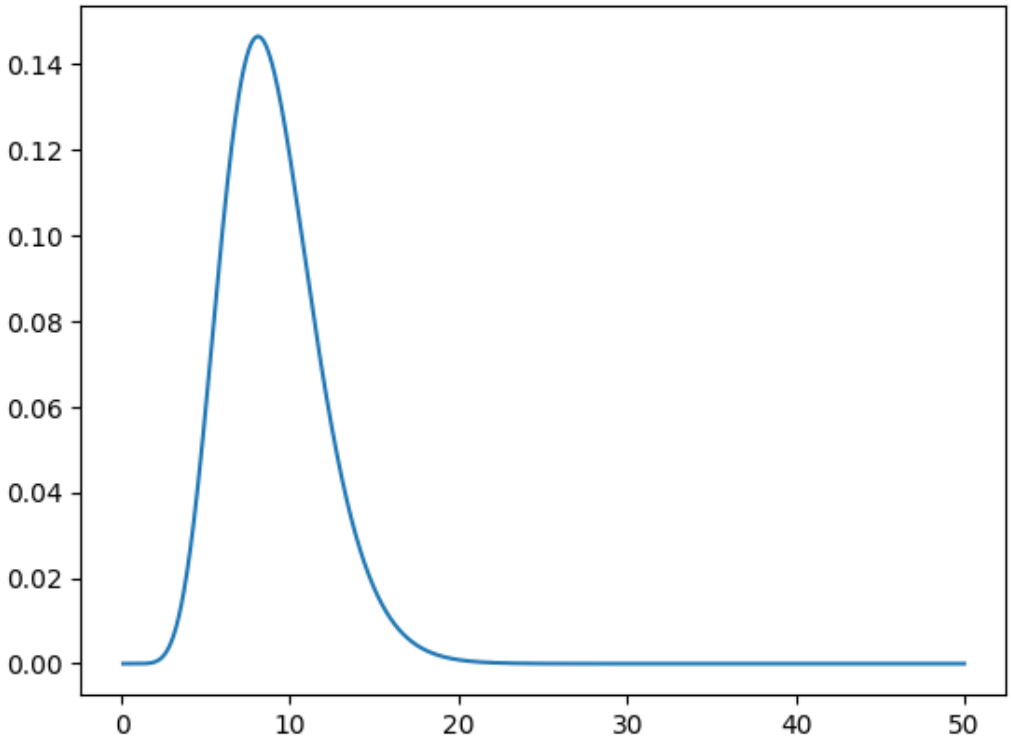

Figure 12:

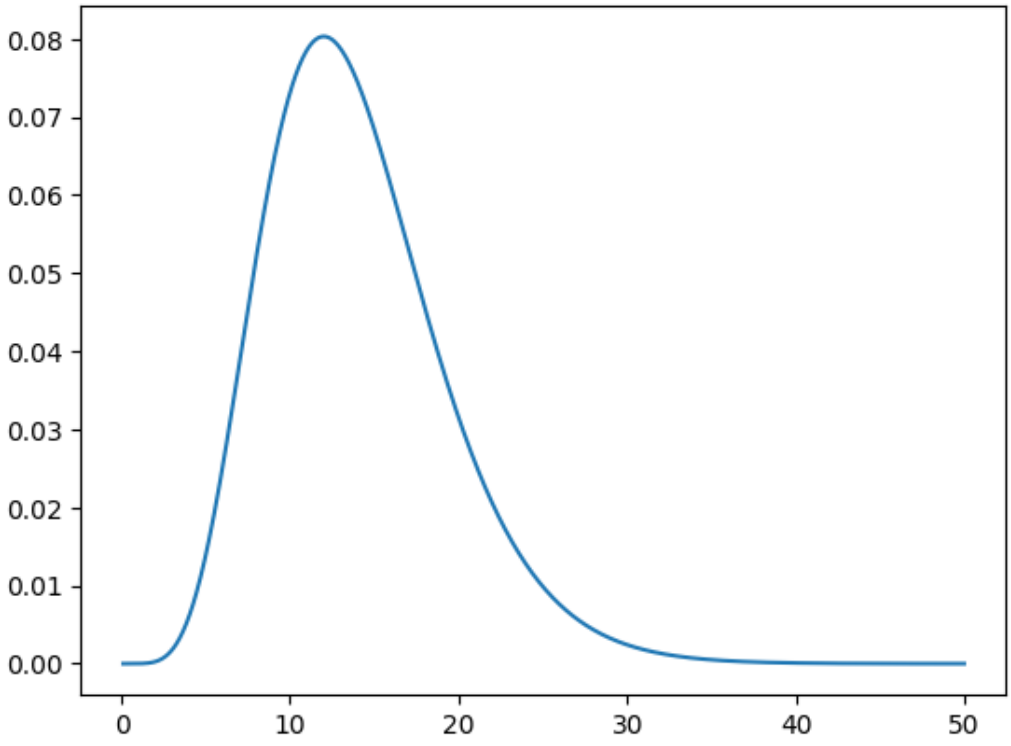

Figure 13:

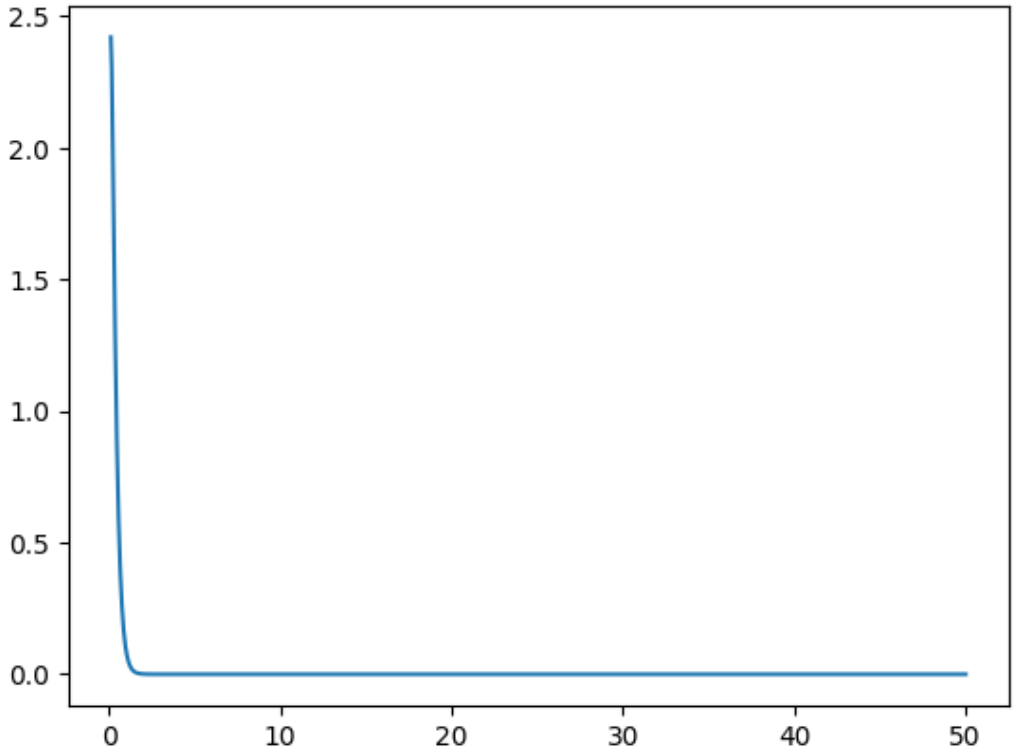

Figure 14:

9.1 Question 3

3. A company has developed a new type of battery whose lifetime (in hours) follows a chi-squared distribution with 10 degrees of freedom. Using any programming language, answer the following questions:

(a) Compute the mean and standard deviation of the battery lifetime. Also, compute the median, first quartile (Q1), and third quartile (Q3) of the lifetime.

(b) Generate a plot of the battery lifetime distribution (PDF) from 0 to 24 hours, clearly indicating the mean, median, and quartiles on the graph. Also, mark the mean, median, and quartiles on the plot.

(c) Using the Moment Generating Function (MGF), generate the first, second, and third moments of the chi-squared distribution.

9.2 Intorudction

The question asks us to caculate mean of a chi squared distribution with degree of freedom 10 and the the 1st 2nd and 3rd quartile and the first second and thir moment using mgf.

9.3 Methodology

The code used is as follows:

```
from scipy.stats import chi2
l=chi2(10)
print(f"mean is {10}")
print(f"standard deviation is {20}")
print(f"the first quartile is {chi2.ppf(0.25,df=10)}")
print(f"the median is {chi2.ppf(0.5,df=10)}")
print(f"the third quartile  is {chi2.ppf(0.75,df=10)}")
import sympy as sp
k_value = 10
t = sp.symbols('t')
MGF = (1 - 2*t)**(-k_value/2)
first_derivative = sp.diff(MGF, t)
first_moment = first_derivative.subs(t, 0)
second_derivative = sp.diff(first_derivative, t)
second_moment = second_derivative.subs(t, 0)
third_derivative = sp.diff(second_derivative, t)
third_moment = third_derivative.subs(t, 0)
print(f"First moment (mean) is {first_moment}")
print(f"Second moment is=  {second_moment}")
print(f"Third moment is  {third_moment}")
```

article amsmath

Code Explanation

The code provided calculates several properties of a Chi-squared distribution and computes the moments using its Moment Generating Function (MGF).

1. Chi-Squared Distribution (Degrees of Freedom = 10)

The code uses the Chi-squared distribution with 10 degrees of freedom. The mean and standard deviation of a Chi-squared distribution are calculated as:

$$\text{Mean} = \mu = 10$$

$$\text{Standard Deviation} = \sigma = 20$$

These values are printed directly in the code.

Next, the code calculates the quantiles, such as the first quartile (Q_1), median (Q_2), and third quartile (Q_3), using the Percent-Point Function (PPF), which is the inverse of the CDF. The code computes:

$$Q_1 = \text{PPF}(0.25, \text{df} = 10)$$

$$Q_2 = \text{PPF}(0.5, \text{df} = 10)$$

$$Q_3 = \text{PPF}(0.75, \text{df} = 10)$$

These give the values of the chi-squared distribution at the specified percentiles.

2. Moment Generating Function (MGF)

The code uses the Moment Generating Function (MGF) to calculate the moments. For a chi-squared distribution with k degrees of freedom, the MGF is given by:

$$M(t) = (1 - 2t)^{-k/2}$$

For $k = 10$, the MGF is:

$$M(t) = (1 - 2t)^{-10/2}$$

3. Moments Calculation

The first, second, and third moments are calculated by taking the derivatives of the MGF and evaluating them at $t = 0$.

First Moment (Mean)

The first derivative of the MGF is:

$$M'(t) = \frac{d}{dt}\left((1-2t)^{-5}\right)$$

Evaluating at $t = 0$ gives the first moment (mean):

$$\text{First Moment} = M'(0) = 10$$

Second Moment

The second derivative of the MGF is:

$$M''(t) = \frac{d^2}{dt^2}\left((1-2t)^{-5}\right)$$

Evaluating at $t = 0$ gives the second moment:

$$\text{Second Moment} = M''(0) = 120$$

Third Moment

The third derivative of the MGF is:

$$M'''(t) = \frac{d^3}{dt^3}\left((1-2t)^{-5}\right)$$

Evaluating at $t = 0$ gives the third moment:

$$\text{Third Moment} = M'''(0) = 1680$$

9.4 Result

The result obtained were as follows: mean is (mathematically given as n) 10
standard deviation is (mathematically given as n) 20
the first quartile is 6.737200771954642
the median is 9.34181776559197
the third quartile is 12.548861396889377
First moment (mean) is 10.0000000000000
Second moment is= 120.000000000000

Third moment is 1680.00000000000