МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

ИНСТИТУТ КИБЕРНЕТИКИ

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 1

по курсу «**Теория вероятностей и математическая статистика, часть 2**»

ВАРИАНТ 67

Тема: **Первичная обработка выборки из дискретной генеральной совокупности**

Выполнил:
Студент 3-го курса
Мусатов Д. Ю.

Группа: КМБО-03-18

МОСКВА – 2021

Содержание

1 Задание

Задание 1. Получить выборку, сгенерировав 200 псевдослучайных чисел, распределенных по биномиальному закону с параметрами n и p.

$n = 7 + V mod 15$; $p = 0,2 + 0,005V$

Задание 2. Получить выборку, сгенерировав 200 псевдослучайных чисел, распределенных по геометрическому закону с параметром .

$p = 0,2 + 0,005V$

Задание 3. Получить выборку, сгенерировав 200 псевдослучайных чисел, распределенных по закону Пуассона с параметром .

$\lambda = 1 + 0,02V$

Следуя **Указаниям** для всех выборок построить:
1) статистический ряд;
2) полигон относительных частот;
3) график эмпирической функции распределения;

Найти:
1) выборочное среднее;
2) выборочную дисперсию;
3) выборочное среднее квадратическое отклонение;
4) выборочную моду;
5) выборочную медиану;
6) выборочный коэффициент асимметрии;
7) выборочный коэффициент эксцесса.

Провести сравнение рассчитанных характеристик с теоретическими значениями.

V - номер варианта. Вычисления проводить с точностью до 0,00001.

2 Краткие теоретические сведения

2.1 Биномиальное распределение

Ряд распределения

x_i	0	1	...	n
p_i	p_0	p_1	...	p_n

где $p_i = C_n^i \cdot p^i \cdot q^{n-i}, \quad i = 0, 1, ...,$
$0 < p \leqslant 1, \quad q = 1 - p$

Математическое ожидание: $M(X) = np$

Дисперсия: $D(X) = npq$

Среднее квадратическое отклонение: $\sigma = \sqrt{D(X)} = \sqrt{npq}$

Мода: $M_0 = \begin{cases} 1) & [(n+1)p], \text{ если } (n+1)p \text{ - дробное;} \\ 2) & (n+1)p - \dfrac{1}{2}, \text{ если } (n+1)p \text{ - целое;} \end{cases}$

Медиана: $M_e = Round(np)$

Коэффициент асимметрии: $a_s = \gamma_1 = \dfrac{q-p}{\sqrt{npq}}$

Коэффициент эксцесса: $\varepsilon_k = \gamma_2 = \dfrac{1-6pq}{npq}$

2.2 Геометрическое распределение

Ряд распределения

x_i	0	1	...
p_i	p_0	p_1	...

где $p_i = p \cdot q^i, \;\; i = 0, 1, ..., \;\; 0 < p < 1, \;\; q = 1 - p$

Математическое ожидание: $M(X) = \dfrac{q}{p}$

Дисперсия: $D(X) = \dfrac{q}{p^2}$

Среднее квадратическое отклонение: $\sigma = \sqrt{D(X)} = \dfrac{\sqrt{q}}{p}$

Мода: $M_0 = 0$

Медиана: $M_e = \begin{cases} 1) \quad \big[-\dfrac{\ln 2}{\ln q} \big], \text{ если } \dfrac{\ln 2}{\ln q} \text{ - дробное;} \\ 2) \quad -\dfrac{\ln 2}{\ln q} - \dfrac{1}{2}, \text{ если } \dfrac{\ln 2}{\ln q} \text{ - целое;} \end{cases}$

Коэффициент асимметрии: $a_s = \gamma_1 = \dfrac{2 - p}{\sqrt{q}}$

Коэффициент эксцесса: $\varepsilon_k = \gamma_2 = 6 + \dfrac{p^2}{q}$

2.3 Распределение Пуассона

Ряд распределения

x_i	0	1	...
p_i	p_0	p_1	...

где $p_i = \dfrac{\lambda^k}{k!} \cdot e^{-\lambda}, \quad \lambda > 0, \quad i = 0, 1, ...$

Математическое ожидание: $M(X) = \lambda$

Дисперсия: $D(X) = \lambda$

Среднее квадратическое отклонение: $\sigma = \sqrt{D(X)} = \sqrt{\lambda}$

Мода: $M_0 = [\lambda]$

Медиана: $M_e \approx \left[\lambda + \dfrac{1}{3} - \dfrac{0,02}{\lambda}\right]$

Коэффициент асимметрии: $a_s = \gamma_1 = \lambda^{-\frac{1}{2}}$

Коэффициент эксцесса: $\varepsilon_k = \gamma_2 = \lambda^{-1}$

3 Результаты расчётов

В программе расчёта был использован язык программирования Python. Работа осуществлялась в среде Jupyter Notebook.

3.1 Задание №1

$$p = 0.535 \quad N = 200 \quad q = 0.465 \quad n = 14$$

Полученная выборка

9	7	8	9	9	7	5	7	7	3
11	9	8	7	7	7	9	5	4	7
8	8	6	10	8	6	8	4	10	9
8	8	6	6	8	7	7	7	9	8
9	6	7	10	6	8	5	8	5	9
9	8	7	8	7	7	6	10	8	5
7	6	6	3	10	6	9	7	10	4
9	7	7	9	9	8	6	7	6	8
4	7	9	10	7	4	9	5	7	6
6	6	6	8	4	4	5	7	7	7
7	5	8	11	8	4	7	8	4	5
7	6	6	7	10	9	5	9	4	5
5	7	9	5	9	5	7	4	6	8
6	6	7	10	6	10	7	12	9	9
6	8	8	7	7	11	7	10	7	9
9	8	6	8	10	5	4	8	8	5
9	7	8	5	6	8	9	7	7	9
8	10	10	8	9	9	4	9	10	7
7	9	9	9	7	10	10	6	9	6
10	6	8	7	6	11	7	6	6	10

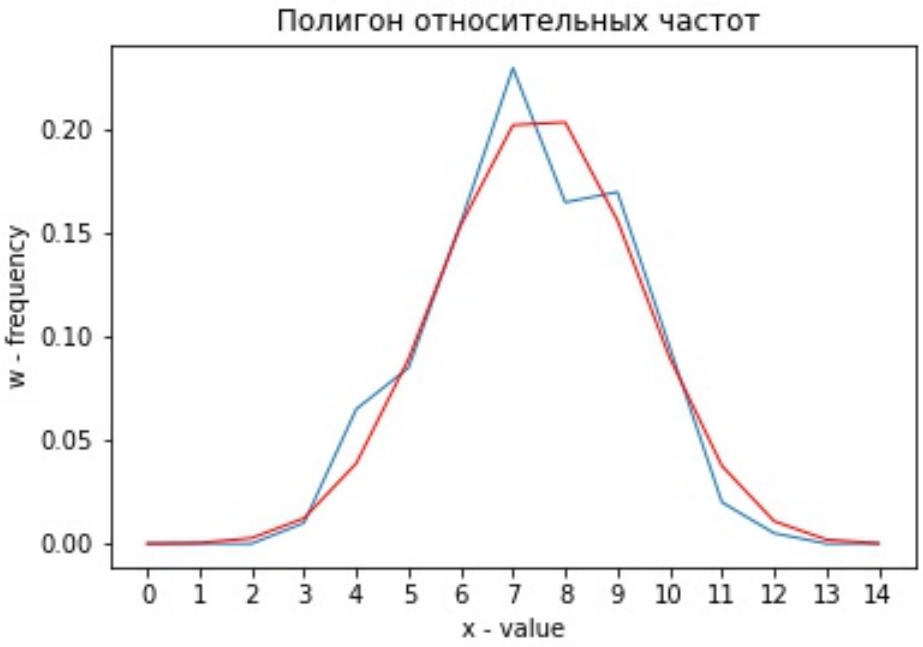
Упорядоченная выборка

3	3	4	4	4	4	4	4	4	4
4	4	4	4	4	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
6	6	6	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	10	10	10	10
10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	11	11	11	11	12

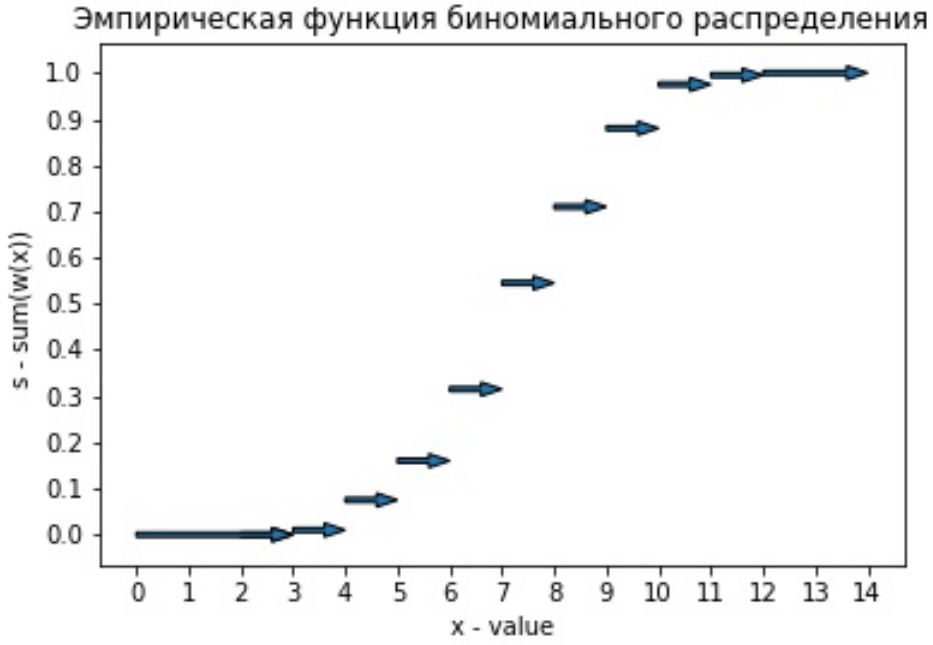
Статистический ряд

x_i	n_i	w_i	s_i
0	0	0	0
1	0	0	0
2	0	0	0
3	2	0.01	0.01
4	13	0.065	0.075
5	17	0.085	0.16
6	31	0.155	0.315
7	46	0.23	0.545
8	33	0.165	0.71
9	34	0.17	0.88
10	19	0.095	0.975
11	4	0.02	0.995
12	1	0.005	1
13	0	0	1
14	0	0	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчётов требуемых характеристик:

- Выборочное среднее: 7.35;
- Выборочная дисперсия: 3.27277;
- Среднее квадратичное: 1.80908;
- Выборочная мода: 7;
- Выборочная медиана: 7;
- Выборочный коэффициент асимметрии: -0.09488;
- Выборочный коэффициент эксцесса: -0.50712;

3.2 Задание №2

$$p = 0.535 \quad N = 200 \quad q = 0.465$$

Полученная выборка

1	0	1	0	0	2	0	2	0	4
3	0	2	2	0	0	0	0	0	1
1	0	3	1	0	0	0	1	0	1
3	1	5	3	0	1	0	0	2	0
2	0	0	0	2	0	3	1	1	3
0	1	2	0	2	1	0	0	0	2
1	0	2	1	1	2	1	1	0	1
0	3	0	0	0	0	1	3	3	1
0	2	1	0	0	0	0	0	0	0
0	4	0	0	3	1	5	1	0	1
0	0	0	0	0	0	1	1	0	1
1	5	5	0	1	0	1	0	2	0
3	1	0	0	0	0	0	2	3	0
2	0	1	0	3	1	5	0	1	1
1	0	1	0	0	3	2	0	0	4
0	1	0	0	3	0	1	1	0	1
1	1	0	3	4	1	2	0	0	0
2	0	7	2	0	1	0	1	1	1
4	0	0	1	1	0	0	0	3	0
7	0	1	0	0	0	0	2	0	0

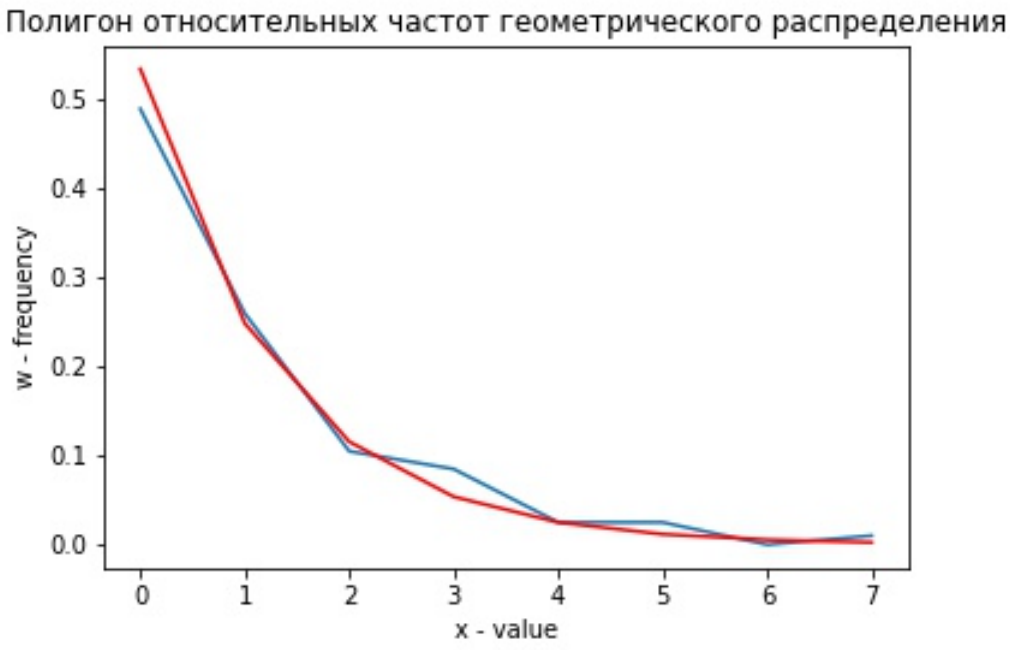
Упорядоченная выборка

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	4	4
4	4	4	5	5	5	5	5	7	7

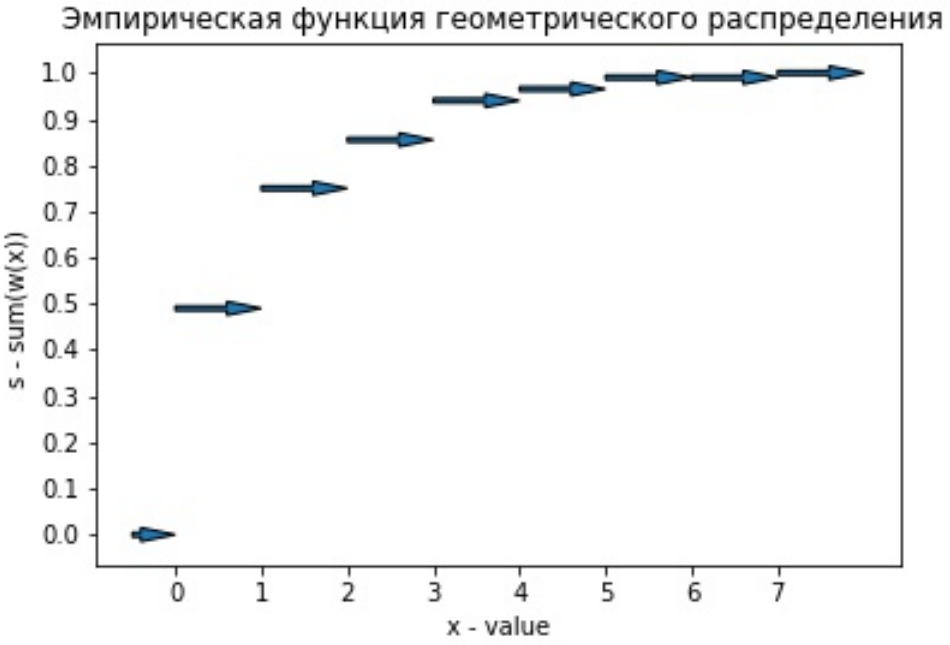
Статистический ряд

x_i	n_i	w_i	s_i
0	98	0.49	0.49
1	52	0.26	0.75
2	21	0.105	0.855
3	17	0.085	0.94
4	5	0.025	0.965
5	5	0.025	0.99
6	2	0	1
7	200	0.01	-
		1	

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчётов требуемых характеристик:

- Выборочное среднее: 1.02;
- Выборочная дисперсия: 1.9196;
- Среднее квадратичное: 1.3855;
- Выборочная мода: 0;
- Выборочная медиана: 1;
- Выборочный коэффициент асимметрии: 1.73516;
- Выборочный коэффициент эксцесса: 3.23258;

3.3 Задание №3

$$p = 0.535 \quad N = 200 \quad q = 0.465 \quad \lambda = 2.34$$

Полученная выборка

3	2	0	2	2	2	0	1	1	2
4	2	0	6	1	3	0	0	4	7
1	0	1	2	4	1	2	3	3	3
2	2	1	0	6	1	3	1	1	3
2	3	1	3	3	4	2	3	2	3
2	0	1	3	2	1	3	5	3	1
4	4	1	1	1	1	2	2	3	2
1	1	2	6	4	1	1	5	0	1
4	4	2	1	2	2	0	6	6	3
4	2	2	1	3	5	2	2	2	5
1	6	2	6	5	1	0	3	4	2
1	1	2	2	1	2	7	2	3	2
2	2	3	6	1	3	1	1	1	1
0	2	3	1	0	1	5	3	1	3
2	0	3	2	6	0	2	6	4	1
3	1	3	2	3	1	4	2	2	2
1	2	4	4	2	1	8	1	6	3
1	3	4	1	2	1	2	1	5	3
2	2	5	0	5	2	3	2	2	2
0	2	6	4	2	3	3	3	2	2

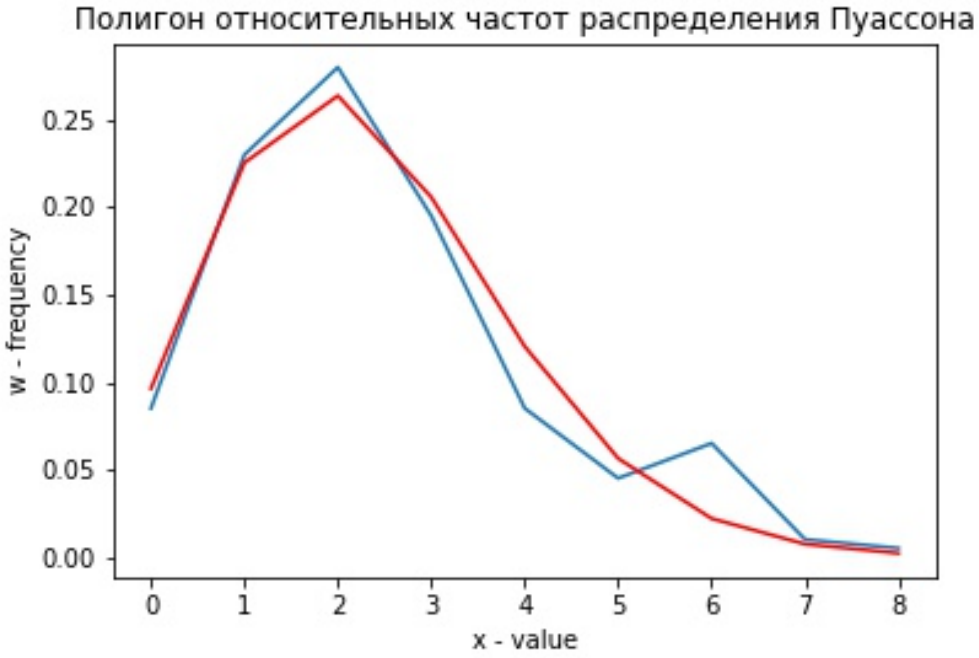
Упорядоченная выборка

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	4	4
4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	5	5	5	5	5
5	5	5	5	6	6	6	6	6	6
6	6	6	6	6	6	6	7	7	8

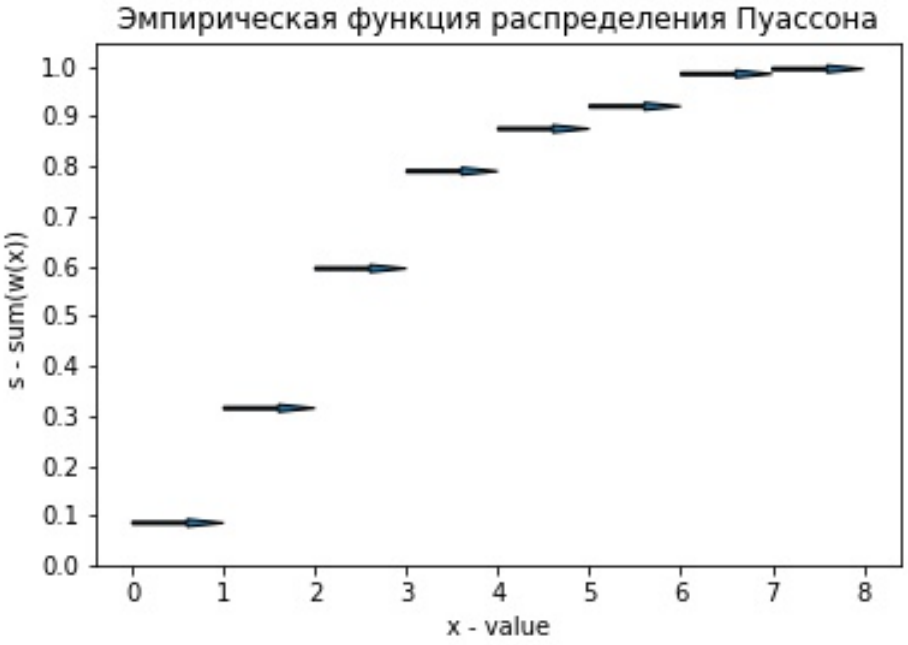
Статистический ряд

x_i	n_i	w_i	s_i
0	17	0.085	0.085
1	46	0.23	0.315
2	56	0.28	0.595
3	39	0.195	0.79
4	17	0.085	0.875
5	9	0.045	0.92
6	13	0.065	0.985
7	2	0.01	0.995
8	1	0.005	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчётов требуемых характеристик:

- Выборочное среднее: 2.44;
- Выборочная дисперсия: 2.7864;
- Среднее квадратичное: 1.66925;
- Выборочная мода: 2;
- Выборочная медиана: 2;
- Выборочный коэффициент асимметрии: 0.83995;
- Выборочный коэффициент эксцесса: 0.35006;

4 Анализ результатов и выводы

Таблица сравнения относительных частот и теоретических вероятностей

Биномиальное распределение

j	w`_j	p_j	|w`_j - p_j|
0	0	0.00002	0.00002
1	0	0.00036	0.00036
2	0	0.00266	0.00266
3	0.01	0.01225	0.00225
4	0.065	0.03876	0.02624
5	0.085	0.08919	0.00419
6	0.155	0.15392	0.00108
7	0.23	0.20239	0.02761
8	0.165	0.20375	0.03875
9	0.17	0.15628	0.01372
10	0.095	0.0899	0.0051
11	0.02	0.03761	0.01761
12	0.005	0.01082	0.00582
13	0	0.00192	0.00192
14	0	0.00016	0.00016
	1	1	0.03875

Геометрическое распределение

j	w`_j	p_j	|w`_j - p_j|
0	0.49	0.535	0.045
1	0.26	0.24878	0.01122
2	0.105	0.11568	0.01068
3	0.085	0.05379	0.03121
4	0.025	0.02501	0.00001
5	0.025	0.01163	0.01337
6	0	0.00541	0.00459
7	0.01	0.00251	0.045
	1	1	

Распределение Пуассона

j	w`_j	p_j	|w`_j - p_j|
0	0.085	0.09633	0.01133
1	0.23	0.22541	0.00459
2	0.28	0.26373	0.01627
3	0.195	0.20571	0.01071
4	0.085	0.12034	0.03534
5	0.045	0.05632	0.01132
6	0.065	0.02196	0.04304
7	0.01	0.00734	0.00266
8	0.005	0.00215	0.00285
	1	1	0.04304

Таблица сравнения рассчитанных характеристик с теоретическими значениями

Биномиальное распределение

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	7.335	7.49	0.155	0.02069
Выборочная дисперсия	3.27277	3.48285	0.21008	0.06032
Выборочное среднее квадратичное отклонение	1.80908	1.86624	0.05716	0.03063
Выборочная мода	7	8	1	0.125
Выборочная медиана	7	7	0	0
Выборочный коэффициент асимметрии	-0.09488	-0.03751	0.05737	1.52946
Выборочный коэффициент эксцесса	-0.50712	-0.14145	0.36567	2.58515

Геометрическое распределение

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	1.02	0.86916	0.15084	0.17355
Выборочная дисперсия	1.9196	1.6246	0.295	0.18158
Выборочное среднее квадратичное отклонение	1.3855	1.2746	0.1109	0.08701
Выборочная мода	0	0	0	-
Выборочная медиана	1	0	1	-
Выборочный коэффициент асимметрии	1.73516	2.14838	0.41323	0.19234
Выборочный коэффициент эксцесса	3.23258	6.61554	3.38296	0.51137

Распределение Пуассона

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	2.44	2.34	0.1	0.04274
Выборочная дисперсия	2.7864	2.34	0.4464	0.19077
Выборочное среднее квадратичное отклонение	1.66925	1.52971	0.13955	0.09123
Выборочная мода	2	2	0	0
Выборочная медиана	2	2	0	0
Выборочный коэффициент асимметрии	0.83995	0.65372	0.18623	0.28488
Выборочный коэффициент эксцесса	0.35006	0.42735	0.07729	0.18086

5 Список использованной литературы

1. Математическая статистика [Электронный ресурс]: метод. указания по выполнению лаб. работ / А.А. Лобузов — М.: МИРЭА, 2017.

2. Боровков А. А. Математическая статистика. — СПб.: Лань, 2010.- 704 с.

3. Гмурман В.Е. Теория вероятностей и математическая статистика. — М.: Юрайт, 2013. — 479 с.

4. Гмурман В.Е. Руководство к решению задач по теории вероятностей и математической статистике. — М.: Юрайт, 2013. — 404 с.

5. Емельянов Г.В.Скитович В.П. Задачник по теории вероятностей и математической статистике. — СПб.: Лань, 2007. — 336 с.

6. Ивченко Г. И., Медведев Ю. И. Введение в математическую статистику. — М.: Изд-во ЛКИ, 2010. — 599 с.

7. Кибзун А.И., Горяинова Е.Р., Наумов А.В. Теория вероятностей и математическая статистика. Базовый курс с примерами и задачам. Учебное пособие — М.:ФИЗМАТЛИТ, 2005. — 232 с.

8. Кобзарь А.И. Прикладная математическая статистика: Для инженеров и научных работников — М.: ФИЗМАТЛИТ, 2006. — 816 с.

9. Монсик В.Б., Скрынников А. А. Вероятность и статистика.— М. : БИНОМ, 2015 — 384 с.

10. Сборник задач по теории вероятностей, математической статистике и теории случайных функций: Учеб. пособие для вузов / Под ред. А. А. Свешникова. — СПб.: Лань, 2012. — 472 с.

11. Письменный Д.Т. Конспект лекций по теории вероятностей, математической статистике и случайным процессам: учеб. пособие для вузов. — М.: Айрис-пресс, 2013. — 288 с.

12. Ramachandran Kandethody M., Tsokos Chris P. Mathematical Statistics with Applications in R. — N-Y.: Academic Press, 2009. — 826 p.

13. Ивановский Р.И. Теория вероятностей и математическая статистика: Основы, прикладные аспекты с примерами и задачами в среде Mathcad:Учеб. пособие для вузов — СПб.: БХВ-Петербург, 2008. — 528 с.

```
import matplotlib.pyplot as plt
import numpy as np
import math
import plotly.graph_objects as go
import copy
p=0.535
N=200
L=2.34
q=0.465
n=14
#Binomial=np.random.binomial(n, p, N)
#Geometric=np.random.geometric(p, N)
#Poisson=np.random.poisson(L, N)
#np.savetxt('binomial.txt', Binomial,'%d')
#np.savetxt('geometric.txt', Geometric,'%d')
#np.savetxt('poisson.txt', Poisson,'%d')
my_file = open("binomial.txt", "r")
Binomial=[]
for num in my_file:
   Binomial.append(int(num))
my_file.close()
Binomial_unsort = []
Binomial_unsort = copy.deepcopy(Binomial)
my_file1 = open("geometric.txt", "r")
Geometric=[]
for num in my_file1:
   Geometric.append(int(num))
my_file1.close()
my_file2 = open("poisson.txt", "r")
Geometric_unsort = []
Geometric_unsort = copy.deepcopy(Geometric)
Poisson=[]
for num in my_file2:
   Poisson.append(int(num))
my_file2.close()
Poisson_unsort = []
Poisson_unsort = copy.deepcopy(Poisson)
Binomial.sort()
Geometric.sort()
Poisson.sort()
dct_binom={}
for i in Binomial:
   if i in dct_binom:
      dct_binom[i] += 1
   else:
      dct_binom[i] = 1
dct_geometric={}
for i in Geometric:
   if i in dct_geometric:
      dct_geometric[i] += 1
   else:
      dct_geometric[i] = 1
dct_poisson={}
for i in Poisson:
```

```
    if i in dct_poisson:
        dct_poisson[i] += 1
    else:
        dct_poisson[i] = 1
matrix_b=[[0] * 4 for i in range(len(dct_binom))]
a=0
items=0
for item in dct_binom:
    print("'%d':%d" % (item, dct_binom[item]))
    matrix_b[items][0] = int(item)
    matrix_b[items][1] = int(dct_binom[item])
    matrix_b[items][2] = dct_binom[item]/200
    for i in range (items+1):
        a+=matrix_b[i][1]
    matrix_b[items][3] = a/200
    items+=1
    a=0
matrix_g=[[0] * 4 for i in range(len(dct_geometric))]
b=0
items=0
for item in dct_geometric:
    print("'%d':%d" % (item-1, dct_geometric[item]))
    matrix_g[items][0] = item-1
    matrix_g[items][1] = dct_geometric[item]
    matrix_g[items][2] = dct_geometric[item]/200
    for i in range (items+1):
        b+=matrix_g[i][1]
    matrix_g[items][3] = b/200
    items+=1
    b=0
matrix_p=[[0] * 4 for i in range(len(dct_poisson))]
c=0
items=0
for item in dct_poisson:
    print("'%d':%d" % (item, dct_poisson[item]))
    matrix_p[items][0] = item
    matrix_p[items][1] = dct_poisson[item]
    matrix_p[items][2] = dct_poisson[item]/200
    for i in range (items+1):
        c+=matrix_p[i][1]
    matrix_p[items][3] = c/200
    c=0
    items+=1
for i in range(len(matrix_b)):
    print(matrix_b[i][0], matrix_b[i][1], matrix_b[i][2], matrix_b[i][3], sep='\t')
for i in range(len(matrix_g)):
    print(matrix_g[i][0], matrix_g[i][1], matrix_g[i][2], matrix_g[i][3], sep='\t')
for i in range(len(matrix_p)):
    print(matrix_p[i][0], matrix_p[i][1], matrix_p[i][2], matrix_p[i][3], sep='\t')
w_b_1 = []
x_b_1 = []
for i in range(len(matrix_b)):
    w_b_1.append(matrix_b[i][2])
for i in range(len(matrix_b)):
```

```
    x_b_1.append(matrix_b[i][0])
fig = plt.figure()

plt.title(' Полигон относительных частот')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot( x_b_1, w_b_1,'r', linewidth=1)
### Вычисление вероятностей по стандартным формулам
P_b=[]
for i in range(n+1):
    P_b.append((p**i)*(q**(n-i))*(math.factorial(n)/(math.factorial(i)*(math.factorial(n-i)))))

P_p=[]
for i in range(len(matrix_p)):
    P_p.append((L**i*math.exp(-L))/math.factorial(i))

P_g=[]
for i in range(matrix_g[len(matrix_g)-1][0]+1):
    P_g.append(p*(q**i))
matrix_v_b_s=[[0] * 20 for i in range(10)]
matrix_v_b_uns=[[0] * 20 for i in range(10)]
matrix_v_g_s=[[0] * 20 for i in range(10)]
matrix_v_g_uns=[[0] * 20 for i in range(10)]
matrix_v_p_s=[[0] * 20 for i in range(10)]
matrix_v_p_uns=[[0] * 20 for i in range(10)]
for i in range(10):
    for j in range(20):
        matrix_v_b_s[i][j]=Binomial[i+j*10]
        matrix_v_b_uns[i][j]=Binomial_unsort[i+j*10]
        matrix_v_g_s[i][j]=(Geometric[i+j*10]-1)
        matrix_v_g_uns[i][j]=(Geometric_unsort[i+j*10]-1)
        matrix_v_p_s[i][j]=Poisson[i+j*10]
        matrix_v_p_uns[i][j]=Poisson_unsort[i+j*10]
### Биномиальное распределение и всё, что с ним связано
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_b_uns[0], matrix_v_b_uns[1],
matrix_v_b_uns[2], matrix_v_b_uns[3], matrix_v_b_uns[4], matrix_v_b_uns[5], matrix_v_b_uns[6],
matrix_v_b_uns[7], matrix_v_b_uns[8], matrix_v_b_uns[9]]))])
fig.show()
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_b_s[0], matrix_v_b_s[1], matrix_v_b_s[2],
matrix_v_b_s[3], matrix_v_b_s[4], matrix_v_b_s[5], matrix_v_b_s[6], matrix_v_b_s[7], matrix_v_b_s[8],
matrix_v_b_s[9]]))
])
fig.show()
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'w_i']),
cells=dict(values=[x_b_1, w_b_1]))
])
fig.show()
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'w_i']),
cells=dict(values=[x_b_1, w_b_1]))
])
fig.show()
w_b = []
n_b = []
k = 0
```

```
x_b = [i for i in range(0,n+1)]
for i in range(n+1):
   if(k<len(matrix_b) and matrix_b[k][0]==i):
      w_b.append(matrix_b[k][2])
      n_b.append(matrix_b[k][1])
      k+=1
   else:
      w_b.append(0)
      n_b.append(0)
plt.title(' Полигон относительных частот')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot( x_b, w_b, 'r', linewidth=1)
### График значений Биномиального расределения вычисленных с помощью стандартной
функции и по формулам
plt.title(' Полигон относительных частот')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.xticks(np.arange(0.0, n+1, step=1))
plt.plot( x_b, w_b, x_b, P_b,'r', linewidth=1)
plt.savefig('binomial_polygon.jpg')
fig = plt.figure()
s_b = [0]
k = 0
for i in range(15):
   if(k<len(matrix_b) and matrix_b[k][0]==i):
      s_b.append(matrix_b[k][3])
      k+=1
   else:
      s_b.append(s_b[i])
s_b.pop(0)
for i in range(x_b_1[0],x_b_1[len(x_b_1)-1]):
   plt.arrow(x_b[i],s_b[i],x_b[i+1]-x_b[i], s_b[i]-s_b[i] , width = 0.01, length_includes_head = True,
head_length = 0.4)
plt.arrow(0,0,x_b_1[0], 0, width = 0.01, length_includes_head = True, head_length = 0.4)
plt.arrow(x_b_1[len(x_b_1)-1],1,n-x_b_1[len(x_b_1)-1], 0, width = 0.01, length_includes_head = True,
head_length = 0.4)
plt.title('Эмпирическая функция биномиального распределения')
plt.ylabel('s - sum(w(x)) ')
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.xticks(np.arange(0.0, n+1, step=1))
plt.xlabel('x - value')
plt.savefig('binomial_emp.jpg')
plt.show()
x_b_average = 0
for i in range(len(matrix_b)):
   x_b_average += (matrix_b[i][0]*matrix_b[i][2])
print(x_b_average)
D_b_average = 0
for i in range(len(matrix_b)):
   D_b_average += ((matrix_b[i][0]-x_b_average)**2)*matrix_b[i][2]
print(round(D_b_average,5))
sqrt_D_b_average = math.sqrt(D_b_average)
print(sqrt_D_b_average)
```

```
c_m_b_3=0
for i in range(len(matrix_b)):
   c_m_b_3 += ((matrix_b[i][0]-x_b_average)**3)*matrix_b[i][2]
print(c_m_b_3)
c_m_b_4=0
for i in range(len(matrix_b)):
   c_m_b_4 += ((matrix_b[i][0]-x_b_average)**4)*matrix_b[i][2]
print(c_m_b_4)
c_b_as = c_m_b_3/(sqrt_D_b_average**3) #coefficiet asymmetry
print(c_b_as)
c_b_ex = c_m_b_4/(sqrt_D_b_average**4) #coefficiet excess
c_b_ex -= 3
print(c_b_ex)
n_b_1=[]
for i in range(len(matrix_b)):
   n_b_1.append(matrix_b[i][1])
Moda_b = -1
max_n_b = max(n_b_1)
counter = 0
first_max = -1;
k = -1
for i in range(len(n_b_1)):
   if(max_n_b == n_b_1[i]):
      counter+=1
      k = i
if(counter==1):
   Moda_b = matrix_b[k][0]
else:
   for i in range(k+1):
      if(max_n_b == n_b_1[i]):
         first_max = i
         break
   for i in range(first_max,k+1):
      counter-=1
   if(counter>=0):
      Moda_b = 0.5*(matrix_b[first_max][0]+matrix_b[k][0])
print(Moda_b)
number = 0
Med_b = 0
for i in range(len(matrix_b)):
   if(matrix_b[i][3]==0.5):
      Med_b = 0.5*(matrix_b[i][0]+matrix_b[i+1][0])
      break
   if(matrix_b[i][3]>0.5):
      Med_b = matrix_b[i][0]
      break
print(Med_b)
D_b_t=n*p*q
print(D_b_t)
sqrt_D_b_average_t=math.sqrt(n*p*q)
print(sqrt_D_b_average_t)
Moda_b_t=0
if(((n+1)*p).is_integer()):
   Moda_b_t=(n+1)*p-0.5
```

```
else:
   Moda_b_t=int((n+1)*p)
print(Moda_b_t)
Med_b_t=round(n*p)
print(Med_b_t)
x_b_average_t=n*p
print(x_b_average_t)
c_b_as_t = (q-p)/(math.sqrt(n*p*q))
print(c_b_as_t)
c_b_ex_t = (1-6*p*q)/(n*p*q)
print(c_b_ex_t)
abs_b_w_p = []
for i in range(len(P_b)):
   abs_b_w_p.append(abs(w_b[i]-P_b[i]))
max_b_delt = max(abs_b_w_p)
x_b_h =[]
w_b_h = []
P_b_h = []
abs_b_w_p_h = []
for i in range(len(x_b)):
   x_b_h.append(round(x_b[i], 5))
x_b_h.append(' ')
for i in range(len(w_b)):
   w_b_h.append(round(w_b[i], 5))
w_b_h.append(sum(w_b))
for i in range(len(P_b)):
   P_b_h.append(round(P_b[i], 5))
P_b_h.append(round(sum(P_b)))
for i in range(len(abs_b_w_p)):
   abs_b_w_p_h.append(round(abs_b_w_p[i], 5))
abs_b_w_p_h.append(round(max_b_delt,5))
fig = go.Figure(data=[go.Table(header=dict(values=['j', 'w`_ j' , 'p_ j', '|w`_ j - p_ j|']),
cells=dict(values=[x_b_h, w_b_h, P_b_h, abs_b_w_p_h]))
])
fig.show()
n_b_h = []
s_b_h = []
for i in range(len(n_b)):
   n_b_h.append(round(n_b[i], 5))
n_b_h.append(sum(n_b))
for i in range(len(s_b)):
   s_b_h.append(round(s_b[i], 5))
s_b_h.append(' - ')
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'n_i' , 'w_i', 's_i']),
cells=dict(values=[x_b_h, n_b_h, w_b_h,s_b_h]))
])
fig.show()
n_b_h = []
s_b_h = []
for i in range(len(n_b)):
   n_b_h.append(round(n_b[i], 5))
n_b_h.append(sum(n_b))
for i in range(len(s_b)):
   s_b_h.append(round(s_b[i], 5))
```

```
s_b_h.append(' - ')
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'n_i' , 'w_i', 's_i']),
cells=dict(values=[x_b_h, n_b_h, w_b_h,s_b_h]))
])
fig.show()
names = ['Выборочное среднее', 'Выборочная дисперсия', 'Выборочное среднее квадратичное
отклонение', 'Выборочная мода', 'Выборочная медиана', 'Выборочный коэффициент асимметрии',
'Выборочный коэффициент эксцесса']
experimental_b = [round(x_b_average, 5), round(D_b_average, 5), round(sqrt_D_b_average, 5),
round(Moda_b, 5), round(Med_b, 5), round(c_b_as, 5), round(c_b_ex, 5)]
theoretical_b = [round(x_b_average_t, 5), round(D_b_t, 5), round(sqrt_D_b_average_t, 5),
round(Moda_b_t, 5), round(Med_b_t, 5), round(c_b_as_t, 5), round(c_b_ex_t, 5)]
absolute_deviation_b = [round(abs(x_b_average-x_b_average_t), 5), round(abs(D_b_average - D_b_t),
5), round(abs(sqrt_D_b_average-sqrt_D_b_average_t), 5), round(abs(Moda_b-Moda_b_t), 5),
round(abs(Med_b-Med_b_t), 5), round(abs(c_b_as-c_b_as_t), 5), round(abs(c_b_ex-c_b_ex_t), 5)]
relative_deviation_b = []
for i in range(len(absolute_deviation_b)):
    if(theoretical_b[i] == 0):
        relative_deviation_b.append(' - ')
    else:
        relative_deviation_b.append(round(abs(absolute_deviation_b[i]/theoretical_b[i]) , 5))
fig = go.Figure(data=[go.Table(header=dict(values=['Название показателя', 'Экспериментальное
значение' , 'Теоретическое значение', 'Абсолютное отклонение', 'Относительное отклонение']),
cells=dict(values=[names, experimental_b, theoretical_b, absolute_deviation_b, relative_deviation_b]))
])
fig.show()
### Геометрическое распределение и всё, что с ним связано
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_g_uns[0], matrix_v_g_uns[1],
matrix_v_g_uns[2], matrix_v_g_uns[3], matrix_v_g_uns[4], matrix_v_g_uns[5], matrix_v_g_uns[6],
matrix_v_g_uns[7], matrix_v_g_uns[8], matrix_v_g_uns[9]]))
])
fig.show()
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_g_s[0], matrix_v_g_s[1], matrix_v_g_s[2],
matrix_v_g_s[3], matrix_v_g_s[4], matrix_v_g_s[5], matrix_v_g_s[6], matrix_v_g_s[7], matrix_v_g_s[8],
matrix_v_g_s[9]]))
])
fig.show()
w_g = []
x_g = []
s_g = []
n_g = []
k=0
for i in range(len(matrix_g)+1):
    if(k<len(matrix_g) and matrix_g[k][0]==i):
        x_g.append(matrix_g[k][0])
        w_g.append(matrix_g[k][2])
        n_g.append(matrix_g[k][1])
        s_g.append(matrix_g[k][3])
        k+=1
    else:
        x_g.append(i)
        w_g.append(0)
        n_g.append(0)
        s_g.append(s_g[i-1])
```

```
fig = plt.figure()
plt.title('Полигон относительных частот геометрического распределения')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot(x_g,w_g,'r',linewidth=1)
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'w_i']),
cells=dict(values=[x_g, w_g]))])
fig.show()
### График значений Геометрического распределения вычисленных с помощью стандартной
функции и по формулам
fig = plt.figure()
plt.title('Полигон относительных частот геометрического распределения')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot(x_g,w_g,x_g,P_g,'r')
plt.savefig('geometrical_polygon.jpg')
fig = plt.figure()
for i in range(len(s_g)-1):
   plt.arrow(x_g[i],s_g[i],x_g[i+1]-x_g[i], s_g[i]-s_g[i] , width = 0.01, length_includes_head = True,
head_length = 0.4)
plt.arrow(-0.5,0,0.5, 0 , width = 0.01, length_includes_head = True, head_length = 0.4)
plt.arrow(x_g[i+1],s_g[i+1],1, 0 , width = 0.01, length_includes_head = True, head_length = 0.4)
plt.title(' Эмпирическая функция геометрического распределения')
plt.ylabel('s - sum(w(x)) ')
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.xticks(np.arange(0.0, x_g[len(x_g)-1]+1, step=1))
plt.xlabel('x - value')
plt.savefig('geometrical_emp.jpg')
plt.show()
sqrt_D_g_average = math.sqrt(D_g_average)
print(sqrt_D_g_average)
c_m_g_3=0
for i in range(len(matrix_g)):
   c_m_g_3 += ((matrix_g[i][0]-x_g_average)**3)*matrix_g[i][2]
print(c_m_g_3)
c_m_g_4=0
for i in range(len(matrix_g)):
   c_m_g_4 += ((matrix_g[i][0]-x_g_average)**4)*matrix_g[i][2]
print(c_m_g_4)
c_g_as = c_m_g_3/(sqrt_D_g_average**3) #coefficiet asymmetry
print(c_g_as)
c_g_ex = c_m_g_4/(sqrt_D_g_average**4)#coefficiet excess
c_g_ex -=3
print(c_g_ex)
Moda_g = -1
max_n_g = max(n_g)
counter = 0
first_max = -1;
k = -1
for i in range(len(n_g)):
   if(max_n_g == n_g[i]):
      counter+=1
      k = i
if(counter==1):
```

```
    Moda_g = matrix_g[k][0]
else:
    for i in range(k+1):
        if(max_n_g == n_g[i]):
            first_max = i
            break
    for i in range(first_max,k+1):
        counter-=1
    if(counter>=0):
        Moda_g = 0.5*(matrix_g[first_max][0]+matrix_g[k][0])
print(Moda_g)
number = 0
Med_g = 0
for i in range(len(matrix_g)):
    if(matrix_g[i][3]==0.5):
        Med_g = 0.5*(matrix_g[i][0]+matrix_g[i+1][0])
        break
    if(matrix_g[i][3]>0.5):
        Med_g = matrix_g[i][0]
        break
print(Med_g)
D_g_t = q/(p**2)
print(D_g_t)
x_g_average_t = q/p
print(x_g_average_t)
sqrt_D_g_average_t = math.sqrt(q)/p
print(sqrt_D_g_average_t)
Moda_g_t = 0
Med_g_t=0
if((math.log1p(1)/math.log1p(q-1)).is_integer()):
    Med_g_t=-0.5-(math.log1p(1)/math.log1p(q-1))
else:
    Med_g_t=int(-(math.log1p(1)/math.log1p(q-1)))
print(Med_g_t)
c_g_as_t = (2-p)/math.sqrt(q)
print(c_g_as_t)
c_g_ex_t = 6+(p**2/q)
print(c_g_ex_t)
abs_g_w_p = []
for i in range(len(P_g)):
    abs_g_w_p.append(abs(w_g[i]-P_g[i]))
max_g_delt = max(abs_g_w_p)
x_g_h =[]
w_g_h = []
P_g_h = []
abs_g_w_p_h = []
for i in range(len(x_g)):
    x_g_h.append(round(x_g[i], 5))
x_g_h.append(' ')
for i in range(len(w_g)):
    w_g_h.append(round(w_g[i], 5))
w_g_h.append(sum(w_g))
for i in range(len(P_g)):
    P_g_h.append(round(P_g[i], 5))
```

```
P_g_h.append(round(sum(P_g)))
for i in range(len(abs_g_w_p)):
   abs_g_w_p_h.append(round(abs_g_w_p[i], 5))
abs_g_w_p_h.append(round(max_g_delt,5))
fig = go.Figure(data=[go.Table(header=dict(values=['j', 'w`_ j' , 'p_ j', '|w`_ j - p_ j|']),
cells=dict(values=[x_g_h, w_g_h, P_g_h, abs_g_w_p_h]))])
fig.show()
n_g_h = []
s_g_h = []
for i in range(len(n_g)):
   n_g_h.append(round(n_g[i], 5))
n_g_h.append(sum(n_g))
for i in range(len(s_g)):
   s_g_h.append(round(s_g[i], 5))
s_g_h.append(' - ')
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'n_i' , 'w_i', 's_i']),
cells=dict(values=[x_g_h, n_g_h, w_g_h,s_g_h]))
])
fig.show()
experimental_g = [round(x_g_average, 5), round(D_g_average, 5), round(sqrt_D_g_average, 5),
round(Moda_g, 5), round(Med_g, 5), round(c_g_as, 5), round(c_g_ex, 5)]
theoretical_g = [round(x_g_average_t, 5), round(D_g_t, 5), round(sqrt_D_g_average_t, 5),
round(Moda_g_t, 5), round(Med_g_t, 5), round(c_g_as_t, 5), round(c_g_ex_t, 5)]
absolute_deviation_g = [round(abs(x_g_average-x_g_average_t), 5), round(abs(D_g_average - D_g_t),
5), round(abs(sqrt_D_g_average-sqrt_D_g_average_t), 5), round(abs(Moda_g-Moda_g_t), 5),
round(abs(Med_g-Med_g_t), 5), round(abs(c_g_as-c_g_as_t), 5), round(abs(c_g_ex-c_g_ex_t), 5)]
relative_deviation_g = []
for i in range(len(absolute_deviation_g)):
   if(theoretical_g[i] == 0):
      relative_deviation_g.append(' - ')
   else:
      relative_deviation_g.append(round(abs(absolute_deviation_g[i]/theoretical_g[i]) , 5))
fig = go.Figure(data=[go.Table(header=dict(values=['Название показателя', 'Экспериментальное
значение' , 'Теоретическое значение', 'Абсолютное отклонение', 'Относительное отклонение']),
cells=dict(values=[names, experimental_g, theoretical_g, absolute_deviation_g, relative_deviation_g]))
])
fig.show()
### Распределение Пуассона и всё, что с ним связано
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_p_s[0], matrix_v_p_s[1], matrix_v_p_s[2],
matrix_v_p_s[3], matrix_v_p_s[4], matrix_v_p_s[5], matrix_v_p_s[6], matrix_v_p_s[7], matrix_v_p_s[8],
matrix_v_p_s[9]]))])
fig.show()
fig = go.Figure(data=[go.Table(cells=dict(values=[matrix_v_p_uns[0], matrix_v_p_uns[1],
matrix_v_p_s[2], matrix_v_p_uns[3], matrix_v_p_uns[4], matrix_v_p_uns[5], matrix_v_p_uns[6],
matrix_v_p_uns[7], matrix_v_p_uns[8], matrix_v_p_uns[9]]))])
fig.show()
w_p = []
x_p = []
s_p = []
for i in range(len(matrix_p)):
   w_p.append(matrix_p[i][2])
for i in range(len(matrix_p)):
   x_p.append(matrix_p[i][0])
for i in range(len(matrix_p)):
```

```
    s_p.append(matrix_p[i][3])
fig = plt.figure()
plt.title('Полигон относительных частот распределения Пуассона')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot( x_p, w_p, 'r')
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'w_i']),
cells=dict(values=[x_p, w_p]))
])
fig.show()
### График значений распределения Пуассона вычисленных с помощью стандартной функции и
по формулам
fig = plt.figure()
plt.title('Полигон относительных частот распределения Пуассона')
plt.ylabel('w - frequency')
plt.xlabel('x - value')
plt.plot( x_p, w_p, x_p, P_p,'r')
plt.savefig('poisson_polygon.jpg')
fig = plt.figure()
for i in range(len(matrix_p)-1):
    plt.arrow(matrix_p[i][0],matrix_p[i][3],matrix_p[i+1][0]-matrix_p[i][0], matrix_p[i][3]-matrix_p[i][3] ,
width = 0.005, length_includes_head = True, head_length = 0.4)
plt.title('Эмпирическая функция распределения Пуассона')
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel('s - sum(w(x)) ')
plt.xlabel('x - value')
plt.savefig('poisson_emp.jpg')
plt.show()
x_p_average = 0
for i in range(len(matrix_p)):
    x_p_average += (matrix_p[i][0]*matrix_p[i][2])
print(x_p_average)
D_p_average = 0
for i in range(len(matrix_p)):
    D_p_average += ((matrix_p[i][0]-x_p_average)**2)*matrix_p[i][2]
print(D_p_average)
sqrt_D_p_average = math.sqrt(D_p_average)
print(sqrt_D_p_average)
c_m_p_3=0
for i in range(len(matrix_p)):
    c_m_p_3 += ((matrix_p[i][0]-x_p_average)**3)*matrix_p[i][2]
print(c_m_p_3)
c_m_p_4=0
for i in range(len(matrix_p)):
    c_m_p_4 += ((matrix_p[i][0]-x_p_average)**4)*matrix_p[i][2]
print(c_m_p_4)
c_p_as = c_m_p_3/(sqrt_D_p_average**3) #coefficiet asymmetry
print(c_p_as)
c_p_ex = c_m_p_4/(sqrt_D_p_average**4) #coefficiet excess
c_p_ex -= 3
print(c_p_ex)
n_p=[]
for i in range(len(matrix_p)):
    n_p.append(matrix_p[i][1])
```

```
Moda_p = -1
max_n_p = max(n_p)
counter = 0
first_max = -1;
k = -1
for i in range(len(n_p)):
   if(max_n_p == n_p[i]):
      counter+=1
      k = i
if(counter==1):
   Moda_p = matrix_p[k][0]
else:
   for i in range(k+1):
      if(max_n_p == n_p[i]):
         first_max = i
         break
   for i in range(first_max,k+1):
      counter-=1
   if(counter>=0):
      Moda_p = 0.5*(matrix_p[first_max][0]+matrix_p[k][0])
print(Moda_p)
number = 0
Med_p = 0
for i in range(len(matrix_p)):
   if(matrix_p[i][3]==0.5):
      Med_p = 0.5*(matrix_p[i][0]+matrix_p[i+1][0])
      break
   if(matrix_p[i][3]>0.5):
      Med_p = matrix_p[i][0]
      break
print(Med_p)
D_p_t=L
print(D_p_t)
x_p_average_t = L
print(x_p_average_t)
sqrt_D_p_average_t = math.sqrt(L)
print(sqrt_D_p_average_t)
Moda_p_t=int(L)
print(Moda_p_t)
Med_p_t=int(L+1/3-0.02/L)
print(Med_p_t)
c_p_as_t = L**(-0.5)
print(c_p_as_t)
c_p_ex_t = L**(-1)
print(c_p_ex_t)
abs_p_w_p = []
for i in range(len(P_p)):
   abs_p_w_p.append(abs(w_p[i]-P_p[i]))
max_p_delt = max(abs_p_w_p)
x_p_h =[]
w_p_h = []
P_p_h = []
abs_p_w_p_h = []
for i in range(len(x_p)):
```

```
    x_p_h.append(round(x_p[i], 5))
x_p_h.append(' ')
for i in range(len(w_p)):
    w_p_h.append(round(w_p[i], 5))
w_p_h.append(sum(w_p))
for i in range(len(P_p)):
    P_p_h.append(round(P_p[i], 5))
P_p_h.append(round(sum(P_p)))
for i in range(len(abs_p_w_p)):
    abs_p_w_p_h.append(round(abs_p_w_p[i], 5))
abs_p_w_p_h.append(round(max_p_delt,5))
fig = go.Figure(data=[go.Table(header=dict(values=['j', 'w`_ j' , 'p_ j', '|w`_ j - p_ j|']),
cells=dict(values=[x_p_h, w_p_h, P_p_h, abs_p_w_p_h]))])
fig.show()
n_p_h = []
s_p_h = []
for i in range(len(n_p)):
    n_p_h.append(round(n_p[i], 5))
n_p_h.append(sum(n_p))
for i in range(len(s_p)):
    s_p_h.append(round(s_p[i], 5))
s_p_h.append(' - ')
fig = go.Figure(data=[go.Table(header=dict(values=['x_i', 'n_i' , 'w_i', 's_i']),
cells=dict(values=[x_p_h, n_p_h, w_p_h, s_p_h]))
])
fig.show()
experimental_p = [round(x_p_average, 5), round(D_p_average, 5), round(sqrt_D_p_average, 5),
round(Moda_p, 5), round(Med_p, 5), round(c_p_as, 5), round(c_p_ex, 5)]
theoretical_p = [round(x_p_average_t, 5), round(D_p_t, 5), round(sqrt_D_p_average_t, 5),
round(Moda_p_t, 5), round(Med_p_t, 5), round(c_p_as_t, 5), round(c_p_ex_t, 5)]
absolute_deviation_p = [round(abs(x_p_average-x_p_average_t), 5), round(abs(D_p_average - D_p_t),
5), round(abs(sqrt_D_p_average-sqrt_D_p_average_t), 5), round(abs(Moda_p-Moda_p_t), 5),
round(abs(Med_p-Med_p_t), 5), round(abs(c_p_as-c_p_as_t), 5), round(abs(c_p_ex-c_p_ex_t), 5)]
relative_deviation_p = []
for i in range(len(absolute_deviation_p)):
    if(theoretical_p[i] == 0):
        relative_deviation_p.append(' - ')
    else:
        relative_deviation_p.append(round(abs(absolute_deviation_p[i]/theoretical_p[i]) , 5))
fig = go.Figure(data=[go.Table(header=dict(values=['Название показателя', 'Экспериментальное
значение' , 'Теоретическое значение', 'Абсолютное отклонение', 'Относительное отклонение']),
cells=dict(values=[names, experimental_p, theoretical_p, absolute_deviation_p, relative_deviation_p]))
])
fig.show()
```