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

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Типовой расчет 1

по курсу **«Специальные методы моделирования»**

Тема: **Моделирование дискретных распределений**

Выполнил:
Студент 1-го курса магистратуры
Малов И. М.

Группа: КММО-11-24

МОСКВА 2025

Задания

Задание 1. Моделирование биномиального распределения.
Получить две выборки из 200 псевдослучайных чисел, распределенных по биномиальному закону с параметрами $\boldsymbol{n}$ и $\boldsymbol{p}$:

1) используя общий (стандартный) метод моделирования дискретных распределений и псевдослучайные числа, равномерно распределенные на интервале $(0,1)$;

2) используя функцию Octave binornd ($\boldsymbol{n}$, $\boldsymbol{p}$) или функцию Python scipy.stats.binom.rvs($\boldsymbol{n}$, $\boldsymbol{p}$, **size**=200).

Полученные выборки упорядочить по возрастанию, построить статистические ряды вида:

x_i	n_i	w_i	p_i	s_i
0	n_0	w_0	p_0	s_0
1	n_1	w_1	p_1	s_1
…	…	…	…	…
m	n_m	w_m	p_m	s_m
	$\sum_{i=0}^{m} n_i$	$\sum_{i=0}^{m} w_i$	$\sum_{i=0}^{m} p_i$	–

где $m = \boldsymbol{n}$, n_i – частота значения i в выборке (проверить $\sum_{i=0}^{m} n_i = N = 200$);

w_i – относительная частота значения i, $w_i = \dfrac{n_i}{N}$, (проверить $\sum_{i=0}^{m} w_i = 1$),

$p_i = C_n^i \cdot p^i \cdot q^{n-i}$, $s_i = \sum_{j=0}^{i} p_j$.

Задание 2. Моделирование геометрического распределения.
Получить две выборки из 200 псевдослучайных чисел, распределенных по геометрическому закону с параметром $\boldsymbol{p}$:

1) используя общий (стандартный) метод моделирования дискретных распределений и псевдослучайные числа, равномерно распределенные на интервале $(0,1)$;

2) используя функцию Octave geornd ($\boldsymbol{p}$) или функцию Python scipy.stats.geom.rvs($\boldsymbol{p}$, **size**=200).

Полученные выборки упорядочить по возрастанию, построить статистические ряды вида:

x_i	n_i	w_i	p_i	s_i
0	n_0	w_0	p_0	s_0
1	n_1	w_1	p_1	s_1
…	…	…	…	…
m	n_m	w_m	p_m	s_m
	$\sum_{i=0}^{m} n_i$	$\sum_{i=0}^{m} w_i$	$\sum_{i=0}^{m} p_i$	–

где m – максимальное значение в двух полученных выборках, $p_i = p \cdot q^i$.

Задание 3. Моделирование распределения Пуассона.

Получить две выборки из 200 псевдослучайных чисел, распределенных по закону Пуассона с параметром **λ**:

1) используя общий (стандартный) метод моделирования дискретных распределений и псевдослучайные числа, равномерно распределенные на интервале $(0,1)$;

2) используя функцию Octave poissrnd (lam), lam = **λ,**
или функцию Python scipy.stats.poisson.rvs(**λ**, **size=**200).

Полученные выборки упорядочить по возрастанию, построить статистические ряды вида:

x_i	n_i	w_i	p_i	s_i
0	n_0	w_0	p_0	s_0
1	n_1	w_1	p_1	s_1
…	…	…	…	…
m	n_m	w_m	p_m	s_m
	$\sum\limits_{i=0}^{m} n_i$	$\sum\limits_{i=0}^{m} w_i$	$\sum\limits_{i=0}^{m} p_i$	–

где m – максимальное значение в двух полученных выборках, $p_i = \dfrac{\lambda^i}{i!} e^{-\lambda}$.

Для всех распределений построить полигоны относительных частот для двух выборок и полигон вероятностей $\{(i, p_i)\}$ (на одном рисунке, используя для линий синий, зелёный и красный цвета соответственно), найти для каждой выборки выборочное среднее и выборочную дисперсию и сравнить их с теоретическими значениями.

Для всех распределений проверить при уровне значимости $\alpha = 0{,}05$ следующие гипотезы:

1) о соответствии каждой выборки теоретическому распределению;

2) об однородности данных первой и второй выборок.

Результаты вычислений приводить в отчете с точностью до *0,00001.*

Краткие теоретические сведения

В данном разделе для каждого распределения представлены выражения для вероятностей ряда распределения, а также для математического ожидания (среднего значения) и дисперсии. В этом разделе описан общий (стандартный) метод моделирования дискретных распределений.

Сведения о распределениях:

– биномиальное:

Ряд распределения	$p_i = C_n^i \cdot p^i \cdot q^{n-i}$, $i = 0,\ldots,n$, $p \in (0,1)$, $q = 1 - p$;
Математическое ожидание	np
Дисперсия	npq , q = 1 - p

– геометрическое:

Ряд распределения	$p_i = p \cdot q^i$, $i = 0,\ldots;$ $p \in (0,1)$, $q = 1 - p$;
Математическое ожидание	$\frac{q}{p}$
Дисперсия	$\frac{q}{p^2}$

– Пуассона:

Ряд распределения	$p_i = \frac{\lambda^i}{i!} e^{-\lambda}$, $i = 0,\ldots$
Математическое ожидание	λ
Дисперсия	λ

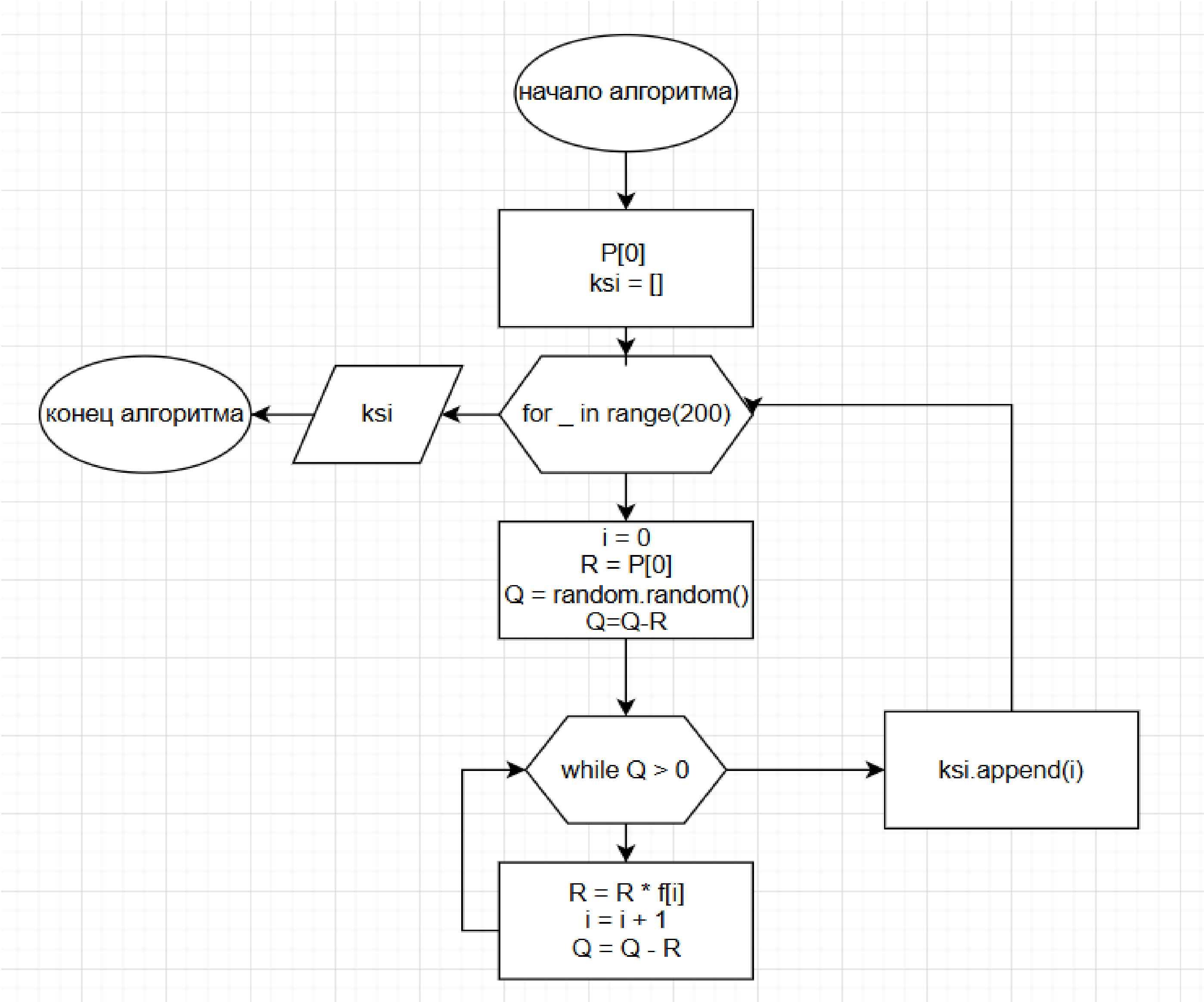

Рис.1 Стандартный алгоритм моделирования дискретных распределений

Распределение	$P(0)$	$f(i)$
Биномиальное	$(1-p)^n$	$\frac{(n-i)*p}{(i+1)*(1-p)}$
Геометрическое	p	q
Пуассона	$e^{-\lambda}$	$\frac{\lambda}{i+1}$

Средства языка программирования

В программе расчета был использован язык программирования Python. Работа осуществлялась в среде Jupyter Notebook с использованием библиотек numpy, scipy и matplotlib.

Были использованы стандартные функции и структуры данных, предоставляемые Python для вычисления функций распределения и плотностей распределений показательного и равномерного распределений, а также для вычислений промежуточных результатов.

Из numpy использовалась структура данных numpy.array и его методы для облегчения вычислений.

В библиотеке scipy использовались функции:

- binom.rvs(n,p,size) – функция генерирующая случайную выборку из биномиального распределения
- geom.rvs(p,size) – функция генерирующая случайную выборку из геометрического распределения
- poisson.rvs(λ,size) – функция генерирующая случайную выборку из распределения Пуассона
- chi2.ppf(alpha,l)- обратная функция кумулятивной функции распределения для распределения хи-квадрат, alpha – квантиль, l – степени свободы

Библиотеки matplotlib использовались для построения графика: были использованы функции настройки фигуры, а также функция matplotlib.pyplot.plot() для построения графиков.

Результаты расчетов

Задание 1

Вариант 8: p = 0.564, n = 15

Данные полученные с помощью САМДР:

12	8	9	9	9	7	10	5	10	10
9	9	8	7	4	11	7	10	8	10
10	9	7	6	9	11	8	7	6	11
8	8	6	10	13	5	9	6	7	9
6	6	6	8	8	8	9	9	11	10
7	7	10	11	11	8	10	9	9	7
10	6	10	9	6	8	6	7	11	10
8	11	9	8	6	11	7	5	7	9
8	12	6	4	9	6	9	9	6	8
11	6	10	12	9	11	7	6	7	10
4	9	9	7	11	8	11	9	10	9
9	10	8	8	9	9	9	8	10	9
9	8	12	7	7	8	9	6	5	7
6	10	7	10	9	8	8	10	9	9
9	6	10	9	5	8	10	8	10	11
9	11	7	9	8	9	6	12	11	9
8	9	9	10	6	6	9	7	7	6
5	6	9	7	7	7	8	10	13	13
3	10	9	6	10	6	12	11	12	6
10	11	9	9	8	8	11	7	10	11

Отсортированные данные:

3	4	4	4	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	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	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	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	10
10	10	10	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11
12	12	12	12	12	12	12	13	13	13

Данные полученные с помощью scipy.stats.binom.rvs

10	8	8	6	10	8	6	10	6	8
9	11	9	11	5	8	8	7	6	9
6	10	7	8	8	7	7	10	8	7
11	9	9	5	10	9	8	8	11	8
7	8	9	9	7	8	10	7	8	8
9	6	11	6	8	8	8	5	8	8
5	6	8	12	9	9	11	5	9	10
13	10	10	9	9	12	12	7	8	8
11	10	8	8	13	7	7	8	6	9
10	8	5	12	7	10	9	11	9	4
6	11	7	10	10	10	9	12	10	8
9	8	12	8	12	6	8	5	11	10
7	10	10	11	9	10	8	8	4	7
4	11	5	9	10	8	8	11	9	10
8	7	10	10	4	7	8	9	4	13
7	6	12	5	5	6	9	9	8	11
10	8	12	10	11	5	6	11	8	10
12	7	9	7	9	7	3	9	11	9
8	3	5	9	9	11	8	7	8	9
8	10	5	8	10	8	8	12	7	9

Отсортированные данные:

3	3	4	4	4	4	4	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	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
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	10	10

10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10	11	11
11	11	11	11	11	11	11	11	11	11
11	11	11	11	11	11	12	12	12	12
12	12	12	12	12	12	12	13	13	13

Статистические ряды САМДР

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	0	0.0	0.00000	0.00000
1	0	0.0	0.00008	0.00008
2	0	0.0	0.00069	0.00077
3	1	0.005	0.00385	0.00462
4	3	0.015	0.01495	0.01957
5	6	0.03	0.04254	0.06211
6	27	0.135	0.09172	0.15383
7	26	0.13	0.15254	0.30637
8	30	0.15	0.19733	0.50370
9	47	0.235	0.19853	0.70223
10	30	0.15	0.15409	0.85632
11	20	0.1	0.09060	0.94692
12	7	0.035	0.03907	0.98599
13	3	0.015	0.01166	0.99765
14	0	0.0	0.00216	0.99981
15	0	0.0	0.00019	1.0
∑	200	1.0	1.0	–

Статистические ряды scipy.stats.binom.rvs

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	0	0.0	0.00000	0.00000
1	0	0.0	0.00008	0.00008
2	0	0.0	0.00069	0.00077
3	2	0.01	0.00385	0.00462
4	5	0.025	0.01495	0.01957
5	13	0.065	0.04254	0.06211
6	14	0.07	0.09172	0.15383
7	23	0.115	0.15254	0.30637
8	48	0.24	0.19733	0.50370
9	33	0.165	0.19853	0.70223
10	30	0.15	0.15409	0.85632
11	18	0.09	0.09060	0.94692
12	11	0.055	0.03907	0.98599
13	3	0.015	0.01166	0.99765
14	0	0.0	0.00216	0.99981
15	0	0.0	0.00019	1.0
∑	200	1.0	1.0	–

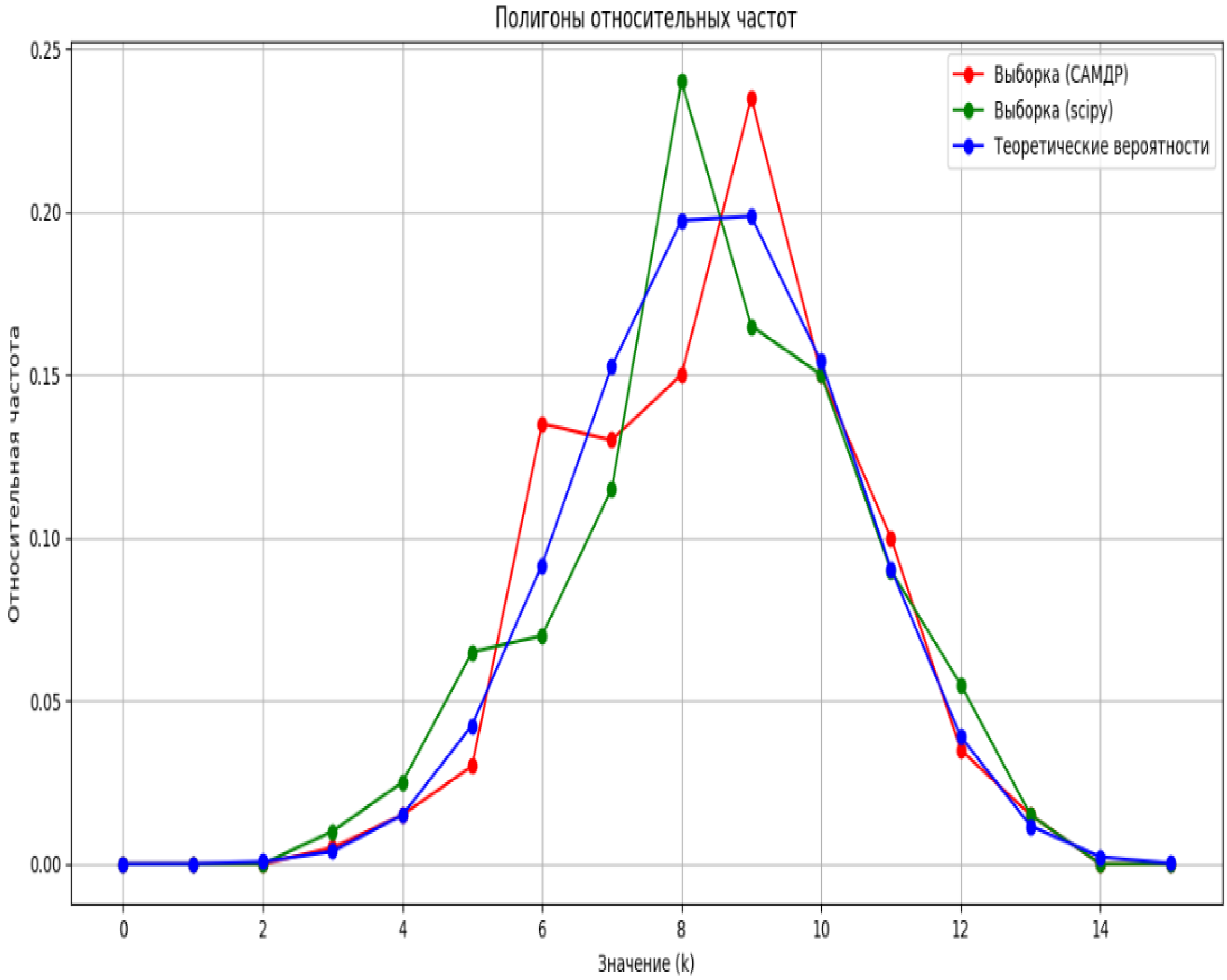

Рис.2 Биномиальное распределение

Расчет критерия хи-квадрат САМДР

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.0	0.00000	0.0	0.00078
1	0.0	0.00008	0.00008	0.01518
2	0.0	0.00069	0.00069	0.13744
3	0.005	0.00385	0.00115	0.06843
4	0.015	0.01495	0.00005	0.00004
5	0.03	0.04254	0.01254	0.73949
6	0.135	0.09172	0.04318	4.08486
7	0.13	0.15254	0.02254	0.66632

8	0.15	0.19733	0.04733	2.2702
9	0.235	0.19853	0.03647	1.33959
10	0.15	0.15409	0.00409	0.02173
11	0.1	0.09060	0.00940	0.19487
12	0.035	0.03907	0.00407	0.08471
13	0.015	0.01166	0.00334	0.19102
14	0.0	0.00216	0.00216	0.43104
15	0.0	0.00019	0.00019	0.03717
	1.0	1.0	0.04733	10.28286

Расчет критерия хи-квадрат scipy

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.0	0.00000	0.0	0.00078
1	0.0	0.00008	0.00008	0.01518
2	0.0	0.00069	0.00069	0.13744
3	0.01	0.00385	0.00615	1.9625
4	0.025	0.01495	0.01005	1.35171
5	0.065	0.04254	0.02246	2.37118
6	0.07	0.09172	0.02172	1.02857
7	0.115	0.15254	0.03754	1.84803
8	0.24	0.19733	0.04267	1.84564
9	0.165	0.19853	0.03353	1.13283
10	0.15	0.15409	0.00409	0.02173
11	0.09	0.09060	0.0006	0.00081
12	0.055	0.03907	0.01593	1.29944
13	0.015	0.01166	0.00334	0.19102
14	0.0	0.00216	0.00216	0.43104
15	0.0	0.00019	0.00019	0.03717
	1.0	1.0	0.04267	13.67507

Набор данных САМДР:

Хи-квадрат: 10.28286

Критическое значение: 18.30704

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с биномиальным распределением.

Набор данных scipy:

Хи-квадрат: 13.67507

Критическое значение: 18.30704

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с биномиальным распределением.

Расчет критерия хи-квадрат для проверки однородности

i	w_{i1}	w_{i2}	$\frac{(w_{i1})^2+(w_{i2})^2}{w_{i1}+w_{i2}}$
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.005	0.01	0.00833
4	0.015	0.025	0.02125
5	0.03	0.065	0.05395
6	0.135	0.07	0.1128
7	0.13	0.115	0.12296
8	0.15	0.24	0.20538
9	0.235	0.165	0.20612
10	0.15	0.15	0.15
11	0.1	0.09	0.09526
12	0.035	0.055	0.04722
13	0.015	0.015	0.015
14	0.0	0.0	0
15	0.0	0.0	0
	1.0	1.0	15.31590

Обе выборки

Хи-квадрат: 15.31590

Критическое значение: 18.30703

Заключение: Не можем отвергнуть нулевую гипотезу. Выборки однородны.

Выборка САМДР (биномиальный)

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	8.47500	8.46	0.01500	0.00177
Выборочная дисперсия	3.78559	3.68856	0.09703	0.02631

Выборка scipy.stats.binom.rvs

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	8.43000	8.46	0.03000	0.00952
Выборочная дисперсия	4.29658	3.68856	0.60802	0.16484

Задание 2

p = 0.564

Данные полученные с помощью САМДР:

0	0	3	2	0	0	1	0	1	2
0	0	2	0	0	0	0	0	0	0
0	6	0	0	0	0	0	0	0	0
0	0	0	2	1	1	0	0	1	2
3	1	0	2	0	0	0	0	1	0
0	0	1	0	0	2	0	4	0	1
1	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	2	1
5	1	2	2	0	2	0	1	0	1
0	0	1	0	2	0	0	1	1	1
0	0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0	2
0	1	0	1	1	1	2	0	1	0
0	0	2	0	0	1	1	2	1	0
0	0	3	1	0	0	0	0	5	0
3	0	0	1	0	0	0	1	1	0
2	0	1	0	0	2	1	0	0	1
3	1	2	0	2	2	2	1	0	0
2	3	0	1	2	0	2	0	1	0
0	3	1	0	0	1	0	0	0	2

Отсортированные данные:

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	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	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	4	5	5	6

Данные полученные с помощью scipy.stats.geom.rvs:

1	0	0	0	1	2	1	2	0	0
2	1	0	0	0	3	0	2	0	3
0	1	0	3	0	1	0	0	1	0
0	1	2	0	0	0	0	1	2	0
1	2	0	1	1	2	0	0	0	2
1	1	0	0	0	0	0	0	2	0
1	1	0	0	0	0	2	3	1	0
1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	4	0	0	0	0
1	1	1	0	2	3	0	0	3	0
1	1	2	4	3	0	0	0	0	0
0	5	0	4	0	0	0	2	0	1
0	0	0	0	5	0	0	0	0	0
1	1	2	0	1	0	1	1	1	0
0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	1	0	1	0
0	0	0	3	0	3	3	1	1	2
2	2	1	1	1	0	1	0	1	1
4	1	2	6	0	1	1	0	0	3

Отсортированные данные:

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	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	3	3	3	3	3	3	3	3
3	3	3	4	4	4	4	5	5	6

Статистические ряды САМДР

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	117	0.585	0.564	0.564
1	45	0.225	0.2459	0.8099
2	27	0.135	0.10721	0.91711
3	7	0.035	0.04675	0.96386
4	1	0.005	0.02038	0.98424
5	2	0.01	0.00889	0.99313
6	1	0.005	0.00387	0.997
∑	200	1.0	0.997	–

Статистические ряды scipy.stats.geom.rvs

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	111	0.555	0.564	0.564
1	52	0.26	0.2459	0.8099
2	19	0.095	0.10721	0.91711
3	11	0.055	0.04675	0.96386
4	4	0.02	0.02038	0.98424
5	2	0.01	0.00889	0.99313
6	1	0.005	0.00387	0.997
∑	200	1.0	0.997	–

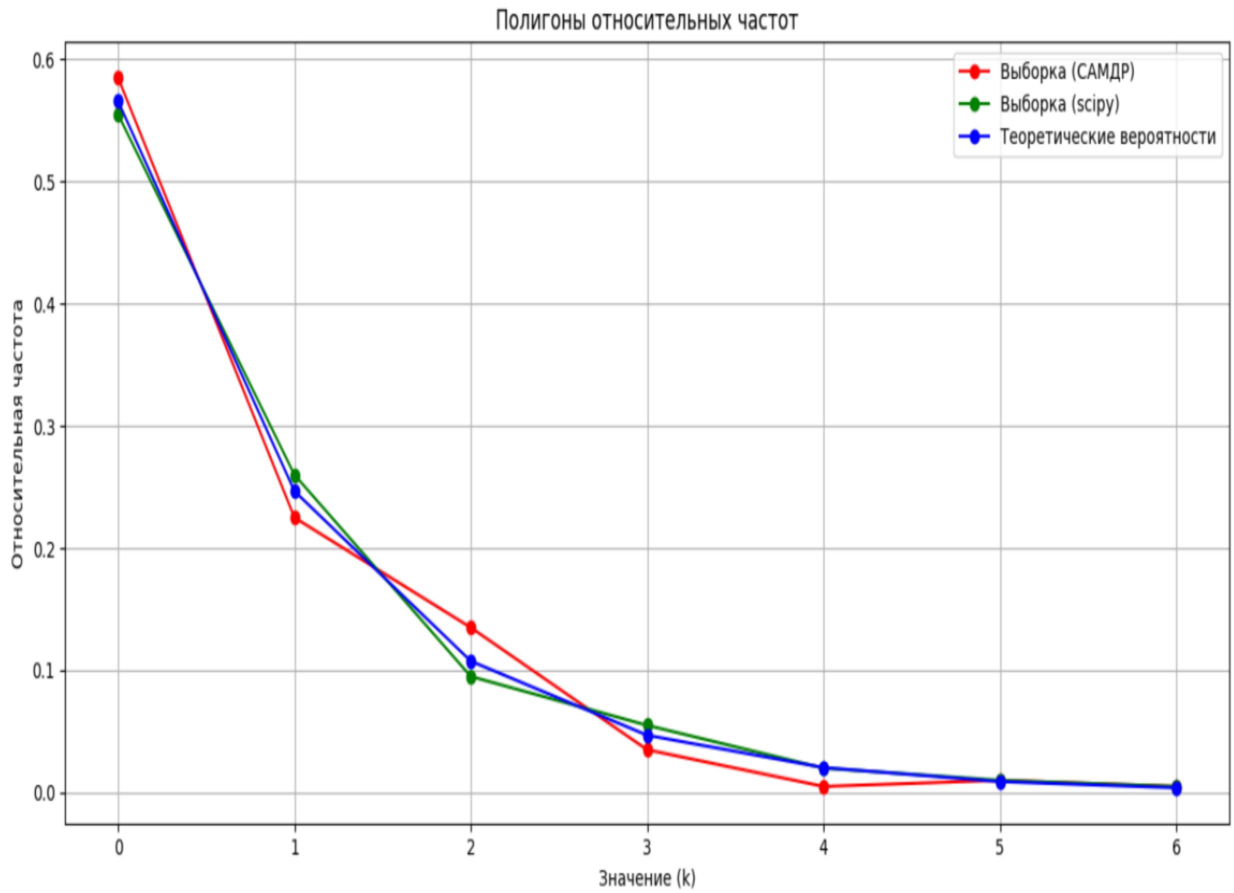

Рис.2 Геометрическое распределение

Расчет критерия хи-квадрат САМДР:

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.585	0.564	0.021	0.15638
1	0.225	0.2459	0.0209	0.3554
2	0.135	0.10721	0.02779	1.44021
3	0.035	0.04675	0.01175	0.59023
4	0.005	0.02038	0.01538	2.32152
5	0.01	0.00889	0.00111	0.02793
6	0.005	0.00387	0.00113	0.06541
	1.0	0.997	0.02779	4.95709

Расчет критерия хи-квадрат scipy:

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.555	0.564	0.009	0.02872
1	0.26	0.2459	0.0141	0.16161
2	0.095	0.10721	0.01221	0.27829
3	0.055	0.04675	0.00825	0.29153
4	0.02	0.02038	0.00038	0.00142
5	0.01	0.00889	0.00111	0.02793
6	0.005	0.00387	0.00113	0.06541
	1.0	0.997	0.0141	0.85492

Набор данных САМДР:

Хи-квадрат: 4.95709

Критическое значение: 12.59159

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с геометрическим распределением.

Набор данных scipy:

Хи-квадрат: 0.85492

Критическое значение: 12.59159

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с геометрическим распределением.

Расчет критерия хи-квадрат для проверки однородности

i	w_{i1}	w_{i2}	$\frac{(w_{i1})^2 + (w_{i2})^2}{w_{i1} + w_{i2}}$
0	0.585	0.555	0.57039
1	0.225	0.26	0.24376
2	0.135	0.095	0.11848
3	0.035	0.055	0.04722
4	0.005	0.02	0.017
5	0.01	0.01	0.01

6	0.005	0.005	0.005
	1.0	1.0	4.74324

Обе выборки:

Хи-квадрат: 4.74324

Критическое значение: 12.59159

Заключение: Не можем отвергнуть нулевую гипотезу. Выборки однородны.

Выборка САМДР (геометрический)

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	0.7	0.77305	0.07305	0.0945
Выборочная дисперсия	1.10553	1.37066	0.26513	0.19343

Выборка scipy.stats.geom.rvs

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	0.775	0.77305	0.00195	0.00252
Выборочная дисперсия	1.29083	1.37066	0.07963	0.05810

Задание 3

lambda = 2.1

Данные полученные с помощью САМДР:

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

Отсортированные данные:

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	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	4	4	4	4	4	4	4

4	4	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	6	6	7

Данные полученные с помощью scipy.stats.poisson.rvs:

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

Отсортированные данные:

0	0	0	0	0	0	0	0	0	0
0	1	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	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	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	4	4	5	5
5	5	5	5	5	6	6	6	6	7

Статистические ряд САМДР

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	26	0.13	0.12246	0.12246
1	51	0.255	0.25716	0.37962
2	56	0.28	0.27002	0.64964
3	30	0.15	0.18901	0.83865
4	26	0.13	0.09923	0.93788
5	8	0.04	0.04168	0.97956
6	2	0.01	0.01459	0.99415
7	1	0.005	0.00438	0.99853
∑	200	1.0	0.99853	−

Статистические ряды scipy.stats.poisson.rvs

x_i	**n_i**	**w_i**	**P_i**	**s_i**
0	19	0.095	0.12246	0.12246
1	46	0.23	0.25716	0.37962
2	57	0.285	0.27002	0.64964
3	47	0. 235	0.18901	0.83865
4	19	0.095	0.09923	0.93788
5	7	0.035	0.04168	0.97956
6	4	0.02	0.01459	0.99415
7	1	0. 005	0.00438	0.99853
∑	200	1.0	0.99853	−

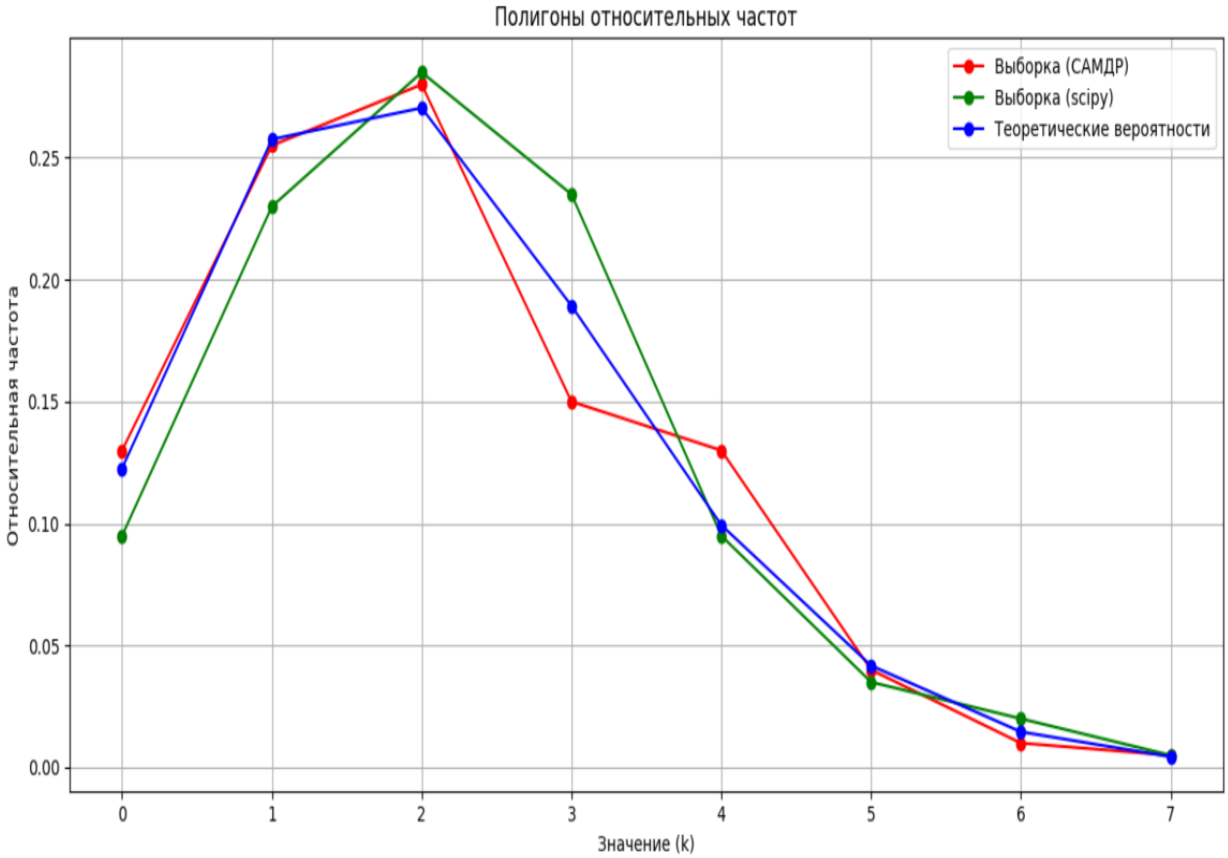

Рис.3 Распределения Пуассона

Расчет критерия хи-квадрат САМДР

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.13	0.12246	0.00754	0.09294
1	0.255	0.25716	0.00216	0.00362
2	0.28	0.27002	0.00998	0.07383
3	0.15	0.18901	0.03901	1.61037
4	0.13	0.09923	0.03077	1.90813
5	0.04	0.04168	0.00168	0.0135
6	0.01	0.01459	0.00459	0.28848
7	0.005	0.00438	0.00062	0.01779
	1.0	0.99853	0.03901	4.00866

Расчет критерия хи-квадрат scipy

i	w_i	p_i	$\lvert w_i - p_i \rvert$	$\frac{N(w_i - p_i)^2}{p_i}$
0	0.095	0.12246	0.02746	1.23122
1	0.23	0.25716	0.02716	0.57364
2	0.285	0.27002	0.01498	0.16629
3	0. 235	0.18901	0.04599	2.2379
4	0.095	0.09923	0.00423	0.03608
5	0.035	0.04168	0.00668	0.21394
6	0.02	0.01459	0.00541	0.40174
7	0. 005	0.00438	0.00062	0.01779
	1.0	0.99853	0.04599	4.87861

Набор данных САМДР:

Хи-квадрат: 4.00866

Критическое значение: 14.06714

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с распределением Пуассона.

Набор данных scipy:

Хи-квадрат: 4.87861

Критическое значение: 14.06714

Заключение: Не можем отвергнуть нулевую гипотезу. Данные согласуются с распределением Пуассона.

Расчет критерия хи-квадрат для проверки однородности

i	w_{i1}	w_{i2}	$\frac{(w_{i1})^2 + (w_{i2})^2}{w_{i1} + w_{i2}}$
0	0.13	0.095	0.11522
1	0.255	0.23	0.24314
2	0.28	0.285	0.28252
3	0.15	0. 235	0.20188
4	0.13	0.095	0.11522
5	0.04	0.035	0.03767

6	0.01	0.02	0.01667
7	0.005	0. 005	0.005
	1.0	1.0	6.93094

Обе выборки

Хи-квадрат: 6.93094

Критическое значение: 14.06714

Заключение: Не можем отвергнуть нулевую гипотезу. Выборки однородны.

Выборка САМДР (Пуассона)

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	2.08	2.1	0.02	0.00952
Выборочная дисперсия	2.09407	2.1	0.00593	0.00282

Выборка scipy.stats.poisson.rvs

Название показателя	Эксперементальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	2.215	2.1	0.115	0.05476
Выборочная дисперсия	1.94852	2.1	0.15148	0.07213

Список литературы

1. Лобузов А.А. Статистическое моделирование [Электронный ресурс]: методические указания. – М.: МИРЭА – Российский технологический университет, 2023.
2. Ермаков С.М., Михайлов Г.А. Статистическое моделирование. – М.: Наука, 1982 г. – 296 с.
3. Соболь И.М. Численные методы Монте-Карло. – М.: Наука, 1973 г. – 312 с.
4. Бусленко Н.П., Голенко Д. И., Соболь И. М., Срагович В. Г., Шрейдер Ю.А. Метод статистических испытаний (метод Монте-Карло). – М.: Гос. изд-во физико-математической литературы, 1962 г. – 332 с.

Приложение

```
import math
import random
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
from collections import Counter
import numpy as np
from scipy.stats import chi2_contingency
from scipy.stats import chi2


n =15
p =0.564
q = 1 - p
lamb = 2.1

def frequancy(data):
    counts = Counter(data)
    n_arr = [counts.get(i, 0) for i in range(min(data), max(data) + 1)]
    return n_arr

def relative_frequancy(n_arr,size = 200):
    return [count/size for count in n_arr]

def element_is_zero1(frequancy1):
    for k in range(len(frequancy1)):
        if frequancy1[k] == 0:
            return True
    return False

def element_is_zero2(arr1, arr2):  # Переименовал для ясности
  """Проверяет, есть ли нулевые элементы хотя бы в одном из массивов."""
  if not arr1 or not arr2:
    return True # Считаем, что если один из массивов пустой, условие
выполняется

  min_len = min(len(arr1), len(arr2))
  for i in range(min_len):
    if arr1[i] == 0 or arr2[i] == 0:
      return True
  return False

def pad_array(arr, min_val, max_val, n):


    for _ in range(min_val):
        arr.insert(0, 0)

    for _ in range(n - max_val -1): # -1 потому что множество не может
содержать дубликаты
        arr.append(0)   # append быстрее, чем insert(len(arr), ...)

    return arr


 def generate_ksi_binom(n, p, size=200):
    ksi = []
    P = np.zeros(n + 1)
    # Вычисление вероятностей для биномиального распределения
    P[0] = (1 - p) ** n   # Вероятность 0 успехов
    for i in range(1, n + 1):
```

```
        P[i] = P[i - 1] * (n - (i - 1)) * p / (i * (1 - p))

    for _ in range(size):
        k = 0
        R = P[0]
        alpha = random.random()
        Q = alpha - R
        while Q>0:
            R = R*(n-k)*p/((k+1)*(1-p))
            Q=Q-R
            k=k+1
        ksi.append(k)

    return ksi, P

s = 0
S_arr_binom = []
n_arr1_binom = []
n_arr2_binom = []


while element_is_zero2(n_arr1_binom, n_arr2_binom) or len(n_arr1_binom) == 0
or len(n_arr1_binom) != len(n_arr2_binom):
    n_arr1_binom = []  # Обнуляем перед каждым заполнением
    n_arr2_binom = []  # Обнуляем перед каждым заполнением

    ksi_binom, P_binom = generate_ksi_binom(n, p)
    ksi_binom.sort()
    n_arr1_binom = frequancy(ksi_binom)

    data_binom = sps.binom.rvs(n, p, size=200)
    data_binom.sort()
    n_arr2_binom = frequancy(data_binom)


print("n_arr1_binom:", n_arr1_binom)
print("n_arr2_binom:", n_arr2_binom)


x_i_binom = np.array(set(ksi_binom))
# Преобразование частот в относительные частоты
relative_frequencies1_binom = relative_frequancy(n_arr1_binom)
relative_frequencies2_binom = relative_frequancy(n_arr2_binom)

print("Сгенерированные значения:", ksi_binom)
print("Сгенерированные значения (scipy):", data_binom)

print(f"P_binom ={P_binom} ")

for i in range(len(P_binom)):
    s+=P_binom[i]
    S_arr_binom.append(s)
print(f"sum_P = {s}")
print(f"S_arr_binom = {S_arr_binom}")
print(f"частоты САМ биномиального распределения{n_arr1_binom}")
print(f"частоты пакетного биномиального распределения{n_arr2_binom}")

var_sum = 0
srednee = (sum(ksi_binom)/200)
print(f" среднее_биномиальное = {srednee:.6f}")
for i in ksi_binom:
    var_sum +=(i-srednee)**2
var_sum/=199
```

```python
print(f" дисперсия = {var_sum}")

print(f"относительные частоты САМ биномиального распределения{relative_frequencies1_binom}")
print(f"относительные частоты пакетного биномиального распределения{relative_frequencies2_binom}")

print(x_i_binom)

print(len(relative_frequencies1_binom))
print(len(relative_frequencies2_binom))

# Подготовка данных для построения полигонов
x1 = [k for k in range(len(relative_frequencies1_binom))]
y1 = [s for s in relative_frequencies1_binom]
x2 = [k for k in range(len(relative_frequencies2_binom))]
y2 = [s for s in relative_frequencies2_binom]

N_abs_sq_div_p = [(200*(relative_frequencies1_binom[i]-P_binom[i])**2)/P_binom[i] for i in range(len(relative_frequencies1_binom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_binom[i]-P_binom[i]) for i in range(len(relative_frequencies1_binom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)
print(f"abs_wi_p = {abs_wi_p}")
print(f"N_abs_sq_div_p = {N_abs_sq_div_p}")

x_prob = np.arange(0, n + 1)
y_prob = P_binom
teor_binom_n = np.array([n_w * 200 for n_w in y_prob], dtype=float)

# Построение полигонов относительных частот
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)  # 1 row, 2 columns, first plot
plt.plot(x1, y1, marker='o', linestyle='-', label='Выборка (САМДР)')
plt.plot(x2, y2, marker='x', linestyle='--', label='Выборка (scipy)')
plt.xlabel('Значение (k)')
plt.ylabel('Относительная частота')
plt.title('Полигоны относительных частот')
plt.legend()
plt.grid(True)


plt.subplot(1, 2, 2)  # 1 row, 2 columns, second plot
plt.plot(x_prob, y_prob, marker='o', linestyle='-', color='blue',
label='Теоретические вероятности')
plt.xlabel('Значение (k)')
plt.ylabel('Вероятность P(k)')
plt.title('Полигон вероятностей')
plt.legend()
plt.grid(True)


plt.tight_layout()  # Предотвращает перекрывание графиков
plt.show()



min_x1 = min(ksi_binom)
max_x1 = max(ksi_binom)
n_arr1_binom = pad_array(n_arr1_binom, min_x1, max_x1, n+1)
```

```
min_x2 = min(data_binom)
max_x2 = max(data_binom)
n_arr2_binom = pad_array(n_arr2_binom, min_x2, max_x2, n+1)

print(n_arr1_binom)
print(n_arr2_binom)

relative_frequencies1_binom = relative_frequancy(n_arr1_binom)
relative_frequencies2_binom = relative_frequancy(n_arr2_binom)

N_abs_sq_div_p = [(200*(relative_frequencies1_binom[i]-
P_binom[i])**2)/P_binom[i] for i in range(len(relative_frequencies1_binom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_binom[i]-P_binom[i]) for i in
range(len(relative_frequencies1_binom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)

var_sum = 0
srednee = (sum(ksi_binom)/200)
print(f" среднее_биномиальное = {srednee:.6f}")
for i in ksi_binom:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

for i in abs_wi_p:
    k = round(i,5)
    print(k)

for i in N_abs_sq_div_p:
    k = round(i,5)
    print(k)
print(sum(N_abs_sq_div_p))


N_abs_sq_div_p = [(200*(relative_frequencies2_binom[i]-
P_binom[i])**2)/P_binom[i] for i in range(len(relative_frequencies2_binom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies2_binom[i]-P_binom[i]) for i in
range(len(relative_frequencies2_binom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)

var_sum = 0
srednee = (sum(data_binom)/200)
print(f" среднее_биномиальное = {srednee:.6f}")
for i in data_binom:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

for i in abs_wi_p:
    k = round(i,5)
    print(k)

for i in N_abs_sq_div_p:
    k = round(i,5)
    print(k)
print(sum(N_abs_sq_div_p))


k = 0
wi12 = []
```

```
for a,b in zip(relative_frequencies1_binom,relative_frequencies2_binom):
    if a==0 and b==0:
        k=0
    else:
        k=(a**2 + b**2)/(a+b)
    print(k)
    wi12.append(k)
print(400*(sum(wi12)-1))



def generate_ksi_geom(p, size=200):
    ksi = []

    # Генерируем значения ksi
    for _ in range(size):
        k = 0
        # Генерируем случайные числа до первой удачи
        while random.random() >= p:  # Пока случайное число > p (неудача)
            k += 1
        ksi.append(k + 1)  # Добавляем 1 для учета первой удачи

    max_ksi = max(ksi)

    # Расчет вероятности P для всех возможных значений k
    P = np.array([p * (1 - p) ** i for i in range(max_ksi)])

    return ksi, P

s = 0
S_arr_geom = []
n_arr1_geom=[]
n_arr2_geom=[]
while len(n_arr1_geom) != len(n_arr2_geom) or len(n_arr1_geom) == 0 or
element_is_zero1(n_arr1_geom) or element_is_zero1(n_arr2_geom):
    ksi_geom,P_geom = generate_ksi_geom(p)
    ksi_geom.sort()

    data_geom = sps.geom.rvs(p,size = 200)
    data_geom.sort()

    n_arr1_geom = frequancy(ksi_geom)
    n_arr2_geom = frequancy(data_geom)

    relative_frequencies1_geom = relative_frequancy(n_arr1_geom)
    relative_frequencies2_geom = relative_frequancy(n_arr2_geom)


print(f"частоты САМ геометрического распределения{n_arr1_geom}")
print(f"частоты пакетного геометрического распределения{n_arr2_geom}")

# Преобразование частот в относительные частоты
print(f"относительные частоты САМ биномгеометрического
распределения{relative_frequencies1_geom}")
print(f"относительные частоты пакетного геометрического
распределения{relative_frequencies2_geom}")


for i in range(len(ksi_geom)):
    ksi_geom[i] -= 1
    data_geom[i] -= 1
print(ksi_geom)
print(data_geom)
```

```
S_arr_geom = []
s = 0
var_sum = 0
print("\nВероятности P(k) ( округленные до 5 знаков):")
for i, prob in enumerate(P_geom):
    print(f"P_geom({i}) = {round(prob, 6)}")
    s=round(s+prob,5)
    S_arr_geom.append(s)
print(f"сумма вероятностей = {round(s,6)}")

x_i_geom =list(set(ksi_geom))
print(f"x_i_geom = {len(x_i_geom)}")
print(len(relative_frequencies1_geom))
print(len(relative_frequencies2_geom))
srednee = (np.sum(data_geom)/200)
print(f" среднее_геометрического = {srednee:.6f}")
for i in data_geom:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")
# Подготовка данных для построения полигонов
x1 = [k for k in range(len(relative_frequencies1_geom))]
y1 = [s for s in relative_frequencies1_geom]
x2 = [k for k in range(len(relative_frequencies2_geom))]
y2 = [s for s in relative_frequencies2_geom]

N_abs_sq_div_p = [(200*(relative_frequencies1_geom[i]-
P_geom[i])**2)/P_geom[i] for i in range(len(relative_frequencies1_geom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_geom[i]-P_geom[i]) for i in
range(len(relative_frequencies1_geom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)
print(f"abs_wi_p = {abs_wi_p}")
print(f"N_abs_sq_div_p = {N_abs_sq_div_p}")


teor_geom_n = []
x_prob = np.arange(0, len(x1))
y_prob = P_geom / np.sum(P_geom)

# Количество необходимых значений
n_w = 200

# Вычисляем teor_geom_n
teor_geom_n = n_w * y_prob

# Проверка значений в teor_geom_n:
print("Проверка значений teor_geom_n:")
for i, val in enumerate(teor_geom_n):
    print(f"Index: {i}, Value: {val}")

# Проверка суммы
print("Сумма teor_geom_n:", np.sum(teor_geom_n))

# Построение полигонов относительных частот
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)  # 1 row, 2 columns, first plot
plt.plot(x1, y1, marker='o', linestyle='-', label='Выборка (САМДР)')
plt.plot(x2, y2, marker='x', linestyle='--', label='Выборка (scipy)')
plt.xlabel('Значение (k)')
plt.ylabel('Относительная частота')
plt.title('Полигоны относительных частот')
```

```
plt.legend()
plt.grid(True)


plt.subplot(1, 2, 2)  # 1 row, 2 columns, second plot
plt.plot(x_prob, y_prob, marker='o', linestyle='-', color='green',
label='Теоретические вероятности')
plt.xlabel('Значение (k)')
plt.ylabel('Вероятность P(k)')
plt.title('Полигон вероятностей')
plt.legend()
plt.grid(True)


plt.tight_layout()  # Предотвращает перекрывание графиков
plt.show()

print(len(ksi_geom))
print(len(data_geom))


var_sum = 0
srednee = (np.sum(ksi_geom)/200)
print(f" среднее_пауссоновского = {srednee:.6f}")
for i in ksi_geom:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

N_abs_sq_div_p = [(200*(relative_frequencies1_geom[i]-
P_geom[i])**2)/P_geom[i] for i in range(len(relative_frequencies1_geom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_geom[i]-P_geom[i]) for i in
range(len(relative_frequencies1_geom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)

for i in abs_wi_p:
    print(round(i,5))

for i in N_abs_sq_div_p:
    print(round(i,5))

print(np.sum(N_abs_sq_div_p))


var_sum = 0
srednee = (np.sum(data_geom)/200)
print(f" среднее_пауссоновского = {srednee:.6f}")
for i in data_geom:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

k = 0
summa = 0
N_abs_sq_div_p = [(200*(relative_frequencies2_geom[i]-
P_geom[i])**2)/P_geom[i] for i in range(len(relative_frequencies2_geom))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies2_geom[i]-P_geom[i]) for i in
range(len(relative_frequencies2_geom))]
abs_wi_p = np.array(abs_wi_p,dtype=float)

for i in abs_wi_p:
```

```
    print(round(i,5))

for i in N_abs_sq_div_p:
    print(round(i,5))
t=0
t = np.sum(N_abs_sq_div_p)
print(round(t,5))


for i in range(len(relative_frequencies2_geom)):
    k = (relative_frequencies1_geom[i]**2 +
relative_frequencies2_geom[i]**2)/(relative_frequencies1_geom[i] +
relative_frequencies2_geom[i])
    summa +=k
    print(round(k,5))
print(400*(summa-1))



def generate_ksi_poisson(lambd, size=200):
    ksi = []
    # Генерируем значения ksi
    for _ in range(size):
        k = 0
        p = math.exp(-lambd)  # Начальная вероятность P(X=0)
        alpha = random.random()

        while alpha > p: # Пока случайное число > p (неудача)
            k += 1
            alpha *= random.random()  # Вычисляем P(X=k) и сравниваем с alpha

        ksi.append(k)

    max_ksi = max(ksi)

    P = np.zeros(max_ksi + 1)
    P[0] = math.exp(-lambd)

    # Расчет вероятности P для всех возможных значений k
    for i in range(max_ksi):
        P[i+1] = P[i] * lambd / (i+1)

    return ksi, P

s = 0
S_arr_poisson = []
n_arr1_poisson=[]
n_arr2_poisson=[]
while len(n_arr1_poisson) != len(n_arr2_poisson) or len(n_arr1_poisson) == 0
:
    ksi_poisson,P_poisson = generate_ksi_poisson(lamb)
    ksi_poisson.sort()

    data_poisson = sps.poisson.rvs(lamb,size = 200)
    data_poisson.sort()

    n_arr1_poisson = frequancy(ksi_poisson)
    n_arr2_poisson = frequancy(data_poisson)

    relative_frequencies1_poisson = relative_frequancy(n_arr1_poisson)
    relative_frequencies2_poisson = relative_frequancy(n_arr2_poisson)

s=0
```

```python
print(ksi_poisson)
print(data_poisson)

print("\nВероятности P(k) ( округленные до 5 знаков):")
for i, prob in enumerate(P_poisson):
    print(f"P_poisson({i}) = {round(prob, 6)}")
    s=round(s+prob,5)
    S_arr_poisson.append(s)
print(f"сумма вероятностей = {round(s,6)}")

print(f"частоты САМ пауссоновского распределения{n_arr1_poisson}")
print(f"частоты пакетного пауссоновского распределения{n_arr2_poisson}")

# Преобразование частот в относительные частоты
print(f"относительные частоты САМ пауссоновского
распределения{relative_frequencies1_poisson}")
print(f"относительные частоты пакетного пауссоновского
распределения{relative_frequencies2_poisson}")
x_i_poisson =list(set(ksi_poisson))
print(f"x_i_geom = {len(x_i_poisson)}")
print(len(relative_frequencies1_poisson))
print(len(relative_frequencies2_poisson))

var_sum = 0
srednee = (sum(ksi_poisson)/200)
print(f" среднее_пауссоновского = {srednee:.6f}")
for i in ksi_poisson:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия")
# Подготовка данных для построения полигонов
x1 = [k for k in range(len(relative_frequencies1_poisson))]
y1 = [s for s in relative_frequencies1_poisson]
x2 = [k for k in range(len(relative_frequencies2_poisson))]
y2 = [s for s in relative_frequencies2_poisson]

N_abs_sq_div_p = [(200*(relative_frequencies1_poisson[i]-
P_poisson[i])**2)/P_poisson[i] for i in
range(len(relative_frequencies1_poisson))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_poisson[i]-P_poisson[i]) for i in
range(len(relative_frequencies1_poisson))]
abs_wi_p = np.array(abs_wi_p,dtype=float)
print(f"abs_wi_p = {abs_wi_p}")
print(f"N_abs_sq_div_p = {N_abs_sq_div_p}")


teor_poisson_n = []
x_prob = np.arange(0, len(x1))
y_prob = P_poisson / np.sum(P_poisson)

# Количество необходимых значений
n_w = 200

# Вычисляем teor_geom_n
teor_poisson_n = n_w * y_prob

# Проверка значений в teor_geom_n:
print("Проверка значений teor_poisson_n:")
for i, val in enumerate(teor_poisson_n):
    print(f"Index: {i}, Value: {val}")

# Проверка суммы
```

```python
print("Сумма teor_poisson_n:", np.sum(teor_poisson_n))

# Построение полигонов относительных частот
plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)  # 1 row, 2 columns, first plot
plt.plot(x1, y1, marker='o', linestyle='-', label='Выборка (САМДР)')
plt.plot(x2, y2, marker='x', linestyle='--', label='Выборка (scipy)')
plt.xlabel('Значение (k)')
plt.ylabel('Относительная частота')
plt.title('Полигоны относительных частот')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)  # 1 row, 2 columns, second plot
plt.plot(x_prob, y_prob, marker='o', linestyle='-', color='red',
label='Теоретические вероятности')
plt.xlabel('Значение (k)')
plt.ylabel('Вероятность P(k)')
plt.title('Полигон вероятностей')
plt.legend()
plt.grid(True)


plt.tight_layout()  # Предотвращает перекрывание графиков
plt.show()

print(len(ksi_poisson))
print(len(data_poisson))



var_sum = 0
srednee = (np.sum(ksi_poisson)/200)
print(f" среднее_пауссоновского = {srednee:.6f}")
for i in ksi_poisson:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

k = 0
summa = 0
N_abs_sq_div_p = [(200*(relative_frequencies1_poisson[i]-
P_poisson[i])**2)/P_poisson[i] for i in
range(len(relative_frequencies1_poisson))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies1_poisson[i]-P_poisson[i]) for i in
range(len(relative_frequencies1_poisson))]
abs_wi_p = np.array(abs_wi_p,dtype=float)


for i in abs_wi_p:
    print(round(i,5))

for i in N_abs_sq_div_p:
    print(i)

print(sum(N_abs_sq_div_p))



var_sum = 0
```

```
srednee = (np.sum(data_poisson)/200)
print(f" среднее_паусcоновского = {srednee:.6f}")
for i in data_poisson:
    var_sum +=(i-srednee)**2
var_sum/=199
print(f" дисперсия = {var_sum}")

N_abs_sq_div_p = [(200*(relative_frequencies2_poisson[i]-
P_poisson[i])**2)/P_poisson[i] for i in
range(len(relative_frequencies2_poisson))]
N_abs_sq_div_p = np.array(N_abs_sq_div_p,dtype=float)
abs_wi_p = [abs(relative_frequencies2_poisson[i]-P_poisson[i]) for i in
range(len(relative_frequencies2_poisson))]
abs_wi_p = np.array(abs_wi_p,dtype=float)

for i in abs_wi_p:
    print(round(i,5))

for i in N_abs_sq_div_p:
    print(i)

print(sum(N_abs_sq_div_p))


for i in range(len(relative_frequencies1_poisson)):
    k = (relative_frequencies1_poisson[i]**2 +
relative_frequencies2_poisson[i]**2)/(relative_frequencies1_poisson[i] +
relative_frequencies2_poisson[i])
    summa +=k
    print(k)
print(400*(summa-1))
```