|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | |  |  |  | | 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 ***n*** и ***p***:

1. используя общий (стандартный) метод моделирования дискретных распределений и псевдослучайные числа, равномерно распределенные на интервале (0,1) ;
2. используя функцию Octave binornd (***n***, ***p***) или функцию Python scipy.stats.binom.rvs(***n***, ***p***, **size=**200).

Полученные выборки упорядочить по возрастанию, построить статистические ряды вида:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *ni* | *wi* | *pi* | *si* |
| 0 | *n*0 | *w*0 | *p*0 | *s*0 |
| 1 | *n*1 | *w*1 | *p*1 | *s*1 |
| … | … | … | … | … |
| *m* | *nm* | *wm* | *pm* | *sm* |
|  | *m*  ∑ *ni*  *i*=0 | *m*  ∑ *wi*  *i*=0 | *m*  ∑ *pi*  *i*=0 | – |

где *m=* ***n***,

*ni* – частота значения *i* в выборке (проверить

∑ *ni* = *N*

*i* = 0

*m*

= 20 0 );

*wi* – относительная частота значения *i* ,

*ni*

*wi* = *N* , (проверить

*m*

∑

*i* = 0

*wi* = 1 ),

*pi* = *Ci* ⋅ *pi* ⋅ *qn*−*i* , *si* =

*n*

*i*

∑ *p j* .

*j* = 0

**Задание 2.** Моделирование геометрического распределения.

Получить две выборки из 200 псевдослучайных чисел, распределенных по геометрическому закону с параметром ***p***:

1. используя общий (стандартный) метод моделирования дискретных распределений и псевдослучайные числа, равномерно распределенные на интервале (0,1) ;
2. используя функцию Octave geornd (***p***) или функцию Python scipy.stats.geom.rvs(***p***, **size=**200). Полученные выборки упорядочить по возрастанию, построить статистические ряды вида:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *ni* | *wi* | *pi* | *si* |
| 0 | *n*0 | *w*0 | *p*0 | *s*0 |
| 1 | *n*1 | *w*1 | *p*1 | *s*1 |
| … | … | … | … | … |
| *m* | *nm* | *wm* | *pm* | *sm* |
|  | *m*  ∑ *ni*  *i*=0 | *m*  ∑ *wi*  *i*=0 | *m*  ∑ *pi*  *i*=0 | – |

где *m* – максимальное значение в двух полученных выборках,

*pi* = *p* ⋅ *qi* .

**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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *ni* | *wi* | *pi* | *si* |
| 0 | *n*0 | *w*0 | *p*0 | *s*0 |
| 1 | *n*1 | *w*1 | *p*1 | *s*1 |
| … | … | … | … | … |
| *m* | *nm* | *wm* | *pm* | *sm* |
|  | *m*  ∑ *ni*  *i*=0 | *m*  ∑ *wi*  *i*=0 | *m*  ∑ *pi*  *i*=0 | – |

где *m* – максимальное значение в двух полученных выборках, *p* = *λi e*−*λ* .

*i i*!

Для всех распределений построить полигоны относительных частот для двух выборок и полигон

вероятностей {(*i* , *pi* )} (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 *α* = 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 | *pi* = ⋅ *pi* ⋅ *qn*−*i* , *i* = 0,...,*n* , *p*∈(0,1) , *q* =1− *p* ; |
| Математическое ожидание | np |
| Дисперсия | npq , q = 1 - p |

– геометрическое:

|  |  |
| --- | --- |
| Ряд распределения | *pi* = *p* ⋅ *qi* , *i* = 0,...; *p*∈(0,1) , *q* =1− *p* ; |
| Математическое ожидание |  |
| Дисперсия |  |

– Пуассона:

|  |  |
| --- | --- |
| Ряд распределения | *pi* = , *i* = 0,... |
| Математическое ожидание | *λ* |
| Дисперсия | *λ* |

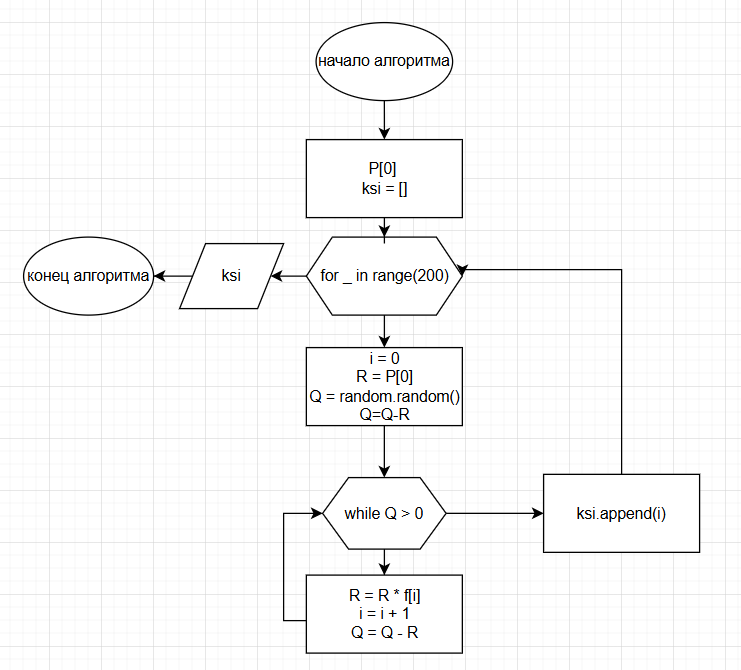


Рис.1 Стандартный алгоритм моделирования дискретных распределений

|  |  |  |
| --- | --- | --- |
| *Распределение* |  |  |
| Биномиальное |  |  |
| Геометрическое |  |  |
| Пуассона |  |  |

**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 | − |

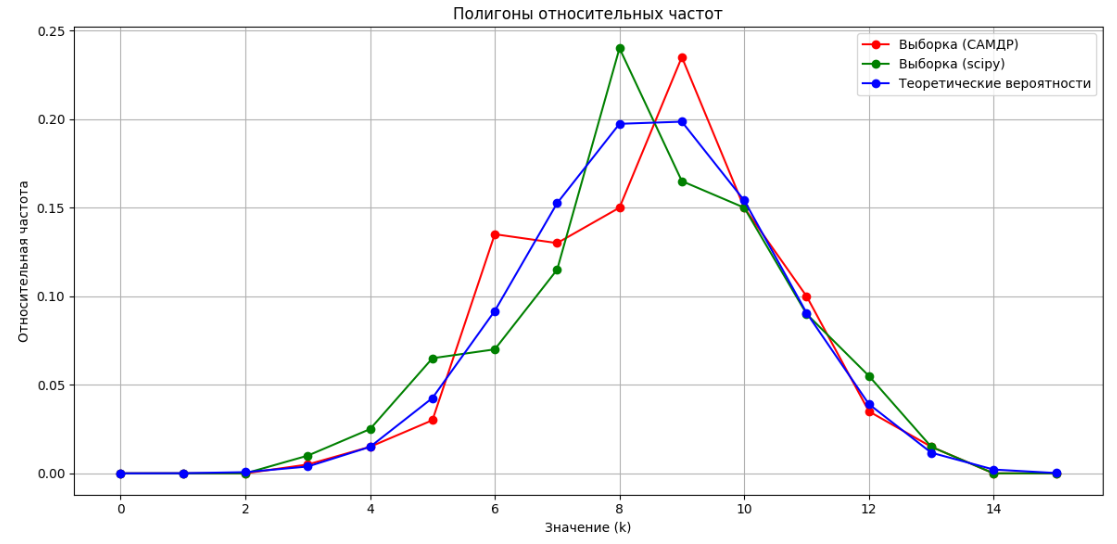


Рис.2 Биномиальное распределение

Расчет критерия хи-квадрат САМДР

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i* | *wi* | *pi* | | *wi −* *pi* | |  |
| 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* | *wi* | *pi* | | *wi −* *pi* | |  |
| 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* |  |  |  |
| 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 САМДР (биномиальный)

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 | − |

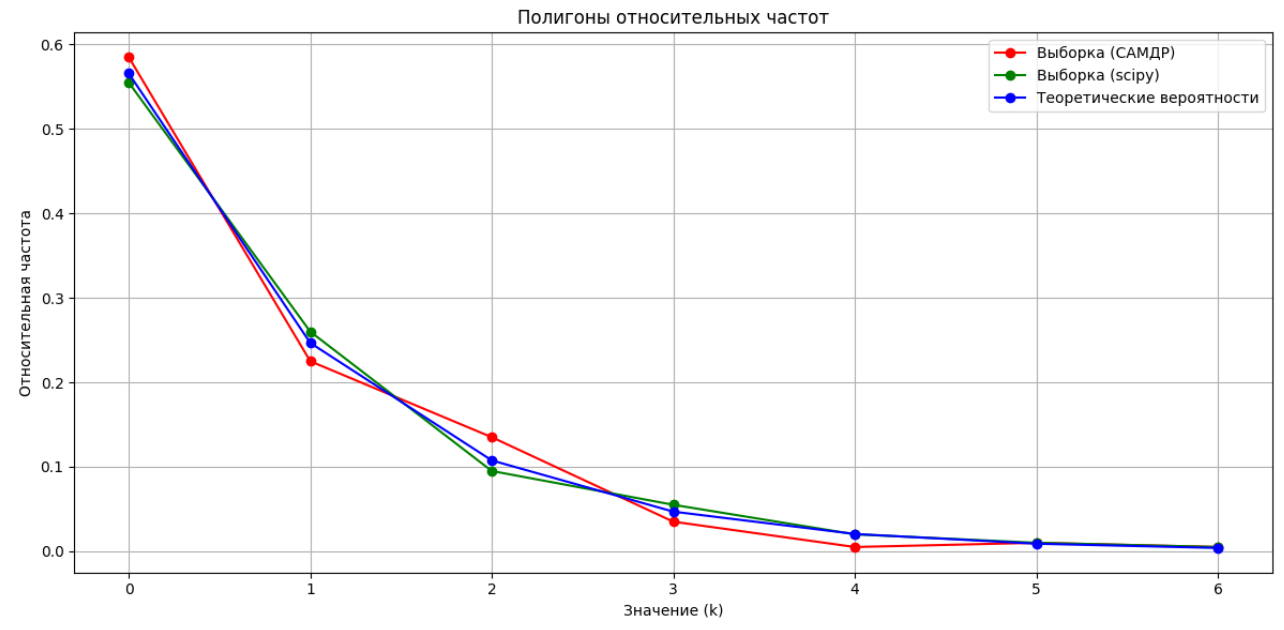


Рис.2 Геометрическое распределение

Расчет критерия хи-квадрат САМДР:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i* | *wi* | *pi* | | *wi −* *pi* | |  |
| 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* | *wi* | *pi* | | *wi −* *pi* | |  |
| 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* |  |  |  |
| 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 | − |

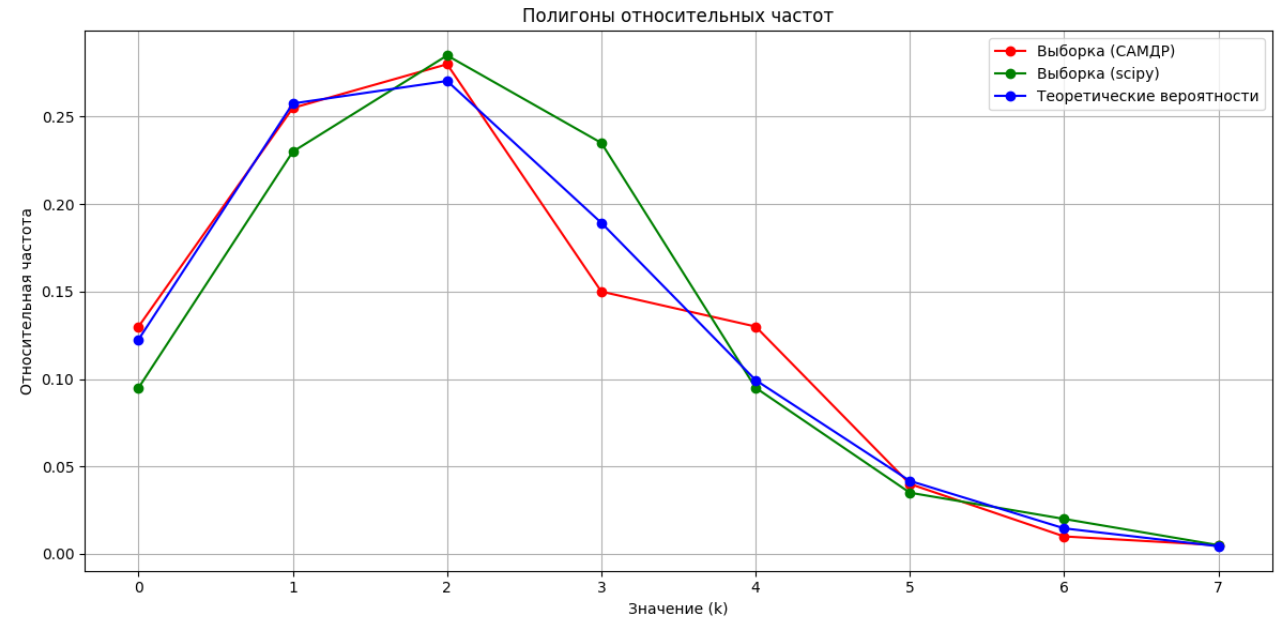


Рис.3 Распределения Пуассона

Расчет критерия хи-квадрат САМДР

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i* | *wi* | *pi* | | *wi –* *pi* | |  |
| 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* | *wi* | *pi* | | *wi –* *pi* | |  |
| 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* |  |  |  |
| 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

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

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

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\_пауссоновского = {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))