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

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 1

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

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

Выполнил:
Студент 3-го курса
Жолковский Д. А.

Группа: КМБО-01-16

МОСКВА 2019

Лабораторная работа по Математической статистике № 1 «Первичная обработка выборки из дискретной генеральной совокупности»

Задание 1. Получить выборку, сгенерировав 150 псевдослучайных чисел, распределенных по биномиальному закону с параметрами n и p. $n=5 + V \bmod 17$ $p=0,1 + 0,01V$

Задание 2. Получить выборку, сгенерировав 150 псевдослучайных чисел, распределенных по геометрическому закону с параметром p.

$p =0,1 + 0,01V$

Задание 3. Получить выборку, сгенерировав 150 псевдослучайных чисел, распределенных по закону Пуассона с параметром λ . $\lambda = 0,7 + 0,07V$

Для всех выборок построить:

1) статистический ряд;

2) полигон относительных частот;

3) эмпирическую функцию 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.

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

Теоретические сведения

Полученную выборку $\{x1 , x2 , x3 , ..., xN\}$ упорядочить по возрастанию, определить частоты n_i и относительные частоты (частости) w_i, построить статистический ряд:

x_i	x_1^*	…	x_m^*
n_i	n_1	…	n_m
w_i	w_1	…	w_m

Полигон относительных частот – ломаная линия, соединяющая последовательно точки с координатами $(x_1^*, w_1), (x_2^*, w_2), \ldots, (x_m^*, w_m)$.

Эмпирическая функция распределения

$$F_N^{\text{Э}}(x) = \sum_{x_i^* \le x} w_i = \begin{cases} 0, & x < x_1^* \\ w_1, & x_1^* \le x < x_2^* \\ w_1 + w_2, & x_2^* \le x < x_3^* \\ w_1 + w_2 + w_3, & x_3^* \le x < x_4^* \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 1, & x \ge x_m^* \end{cases}$$

Выборочное среднее

$$\bar{x} = \sum_{i=1}^{m} x_i^* w_i$$

Выборочная дисперсия

$$D_B = \sum_{i=1}^{m} (x_i^* - \bar{x})^2 w_i$$

Выборочный момент k-ого порядка

$$\overline{\mu_k} = \sum_{i=1}^{m} (x_i^*)^k w_i$$

Выборочный центральный момент k-ого порядка

$$\overline{\mu_k^o} = \sum_{i=1}^{m} (x_i^* - \bar{x})^k w_i$$

Выборочное среднее квадратическое отклонение

$$\bar{\sigma} = \sqrt{\overline{D_B}}$$

Выборочная медиана

$$\overline{M_e} = \begin{cases} x_i^*, & F_N^э(x_{i-1}^*) < 0{,}5 < F_N^э(x_i^*) \\ \dfrac{1}{2}(x_i^* + x_{i+1}^*), & F_N^э(x_i^*) = 0{,}5 \end{cases}$$

Выборочная мода – это значение x_i, которому соответствует максимальная частота.

$$\begin{cases} \text{если } n_i = \max n_k > n_j, \text{i} \neq \text{j}, & \overline{M_0} = \{x_i^* \mid n_i = \max n_k\} \\ \text{если } n_i = n_i + 1 = \ldots = n_{i+j} = \max n_k, & \text{то } \overline{M_0} = \dfrac{1}{2}(x_i^* + x_{i+1}^*) \\ \text{если } n_i = n_j = \max n_k > n_l, i < \text{l} < j, & \text{то } \overline{M_0} - \text{не существует} \end{cases}$$

Выборочный коэффициент асимметрии

$$\overline{\alpha_s} = \frac{\overline{\mu_3^o}}{\bar{\sigma}^3}$$

Выборочный коэффициент эксцесса

$$\overline{\varepsilon_k} = \frac{\overline{\mu_4^o}}{\bar{\sigma}^4} - 3$$

Ряд распределения - структурная группировка с целью выделения характерных свойств и закономерностей изучаемой совокупности.

Математическое ожидание – понятие среднего значения случайной величины в теории вероятностей.

Дисперсия – отклонение величины от ее математического ожидания.

Среднеквадратическое отклонение – показатель рассеивания значений случайной величины относительно ее математического ожидания.

Мода – значение во множестве наблюдений, которое встречается наиболее часто.

Медиана – возможное значение признака, которое делит вариационный ряд выборки на две равные части.

Коэффициент асимметрии используется для проверки распределения на симметричность, а также для грубой предварительной проверки на нормальность.

Если плотность распределения симметрична, то выборочный коэффициент асимметрии равен нулю, если левый хост распределения тяжелее – больше нуля, легче – меньше.

Коэффициент эксцесса используется для проверки на нормальность.

Нормальное распределение имеет нулевой эксцесс. Если хвосты распределения «легче», а пик острее, чем у нормального распределения, то коэффициент эксцесса положительный; если хвосты распределения «тяжелее», пик «приплюснутый», чем у нормального распределения, то отрицательный.

Биномиальное распределение

Биномиальное распределение – распределение количества «успехов» в последовательности из n независимых случайных экспериментов, таких что вероятность «успеха» в каждом из них равна p.

Математическое ожидание: np

Дисперсия: npq, $q=1-p$

Среднеквадратическое отклонение: $\sqrt{npq}$

Мода: [(n+1)p], если (n+1)p – дробное; (n+1)p- $\frac{1}{2}$, если np – целое;

Медиана: $Round(np)$

Коэффициент асимметрии: $\frac{q-p}{\sqrt{npq}}$

Коэффициент эксцесса: $\frac{1-6pq}{npq}$

Геометрическое распределение

Геометрическое распределение – распределение величины, равной количеству испытаний случайного эксперимента до наблюдения первого «успеха».

Математическое ожидание: $\frac{q}{p}$, $q=1-p$

Дисперсия: $\frac{q}{p^2}$, $q=1-p$

Среднее квадратичное отклонение: $\sqrt{\frac{q}{p^2}}$

Мода: 0

Медиана: $\left[-\frac{ln2}{lnq}\right]$, если $\frac{ln2}{lnq}$ – дробное; $-\frac{ln2}{lnq}-\frac{1}{2}$, если $\frac{ln2}{lnq}$ - целое

Коэффициент асимметрии: $\frac{2-p}{\sqrt{1-p}}$

Коэффициент эксцесса: $6+\frac{p^2}{1-p}$

Распределение Пуассона

Распределение Пуассона – вероятностное распределение дискретного типа, моделирует случайную величину, представляющую собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средне интенсивностью и независимо друг от друга.

Математическое ожидание: λ

Дисперсия: λ

Среднеквадратическое отклонение: $\sqrt{\lambda}$

Мода: $\lfloor\lambda\rfloor$

Медиана: $\left\lfloor \lambda + \frac{1}{3} - \frac{0.002}{\lambda} \right\rfloor$

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

Коэффициент эксцесса: λ^{-1}

Средства языка Octave

В программе расчёта используются следующие средства языка:

Функции:

- binornd(n, p, s, z) - возвращает матрицу случайных значений из биномиального распределения с параметрами n и p , где n есть число испытаний и p есть вероятность успеха, s – количество строк в возвращаемой матрице, z – количество столбцов.
- geornd(p, s, z) - возвращает матрицу случайных значений из геометрического распределения с параметром p, s – количество строк в возвращаемой матрице, z – количество столбцов.
- poissrnd(λ, s, z) - возвращает матрицу случайных значений из распределения Пуассона с параметром λ, s – количество строк в возвращаемой матрице, z – количество столбцов.
- sort(x) - возвращает копию x с элементами, расположенными в порядке возрастания.
- sqrt(x) – возвращает квадратный корень из числа x.
- max(X) в случае одномерного массива возвращает наибольший элемент; в случае двумерного массива - это вектор-строка, содержащая максимальные элементы каждого столбца.

Операторы управления:

- for – endfor
- if – else – endif
- break

А также арифметические и логические операторы.

Результаты расчетов с комментариями

Задание 1:

n = 14, p = 0,19

Выборка:

Неупорядоченная:

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

Упорядоченная:

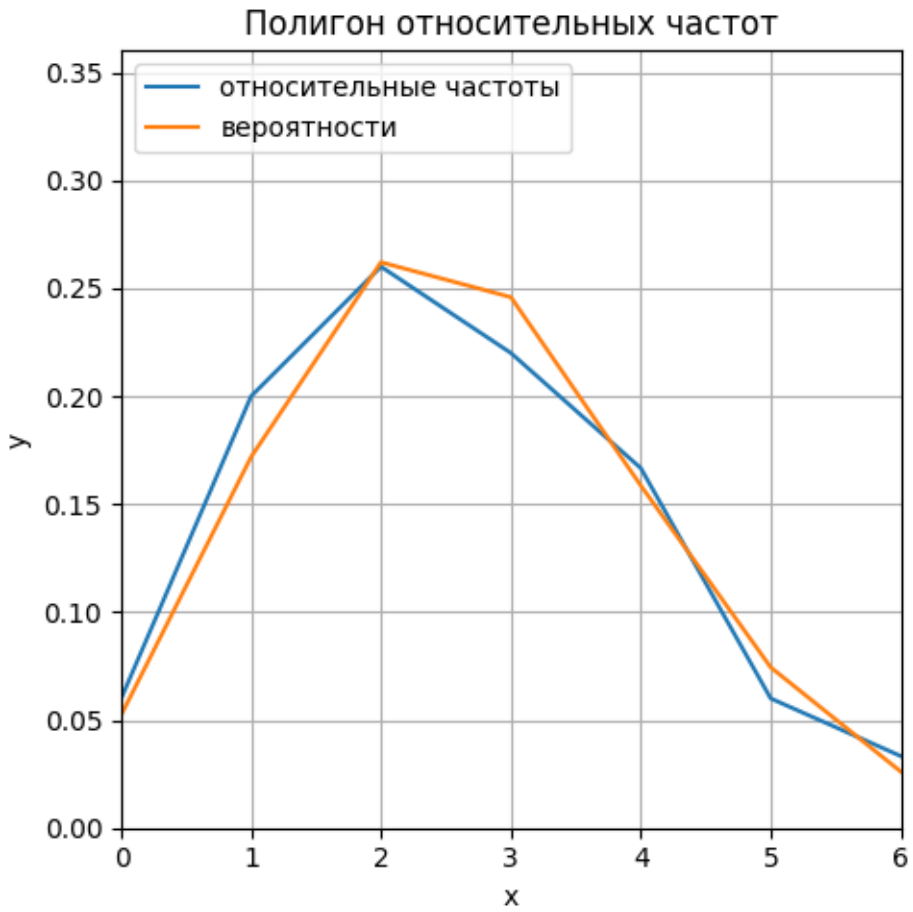
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	5	5	5	5	5	5	5	5	5	6	6	6	6	6

Статистический ряд:

x_i	0	1	2	3	4	5	6
n_i	9	30	39	33	25	9	5

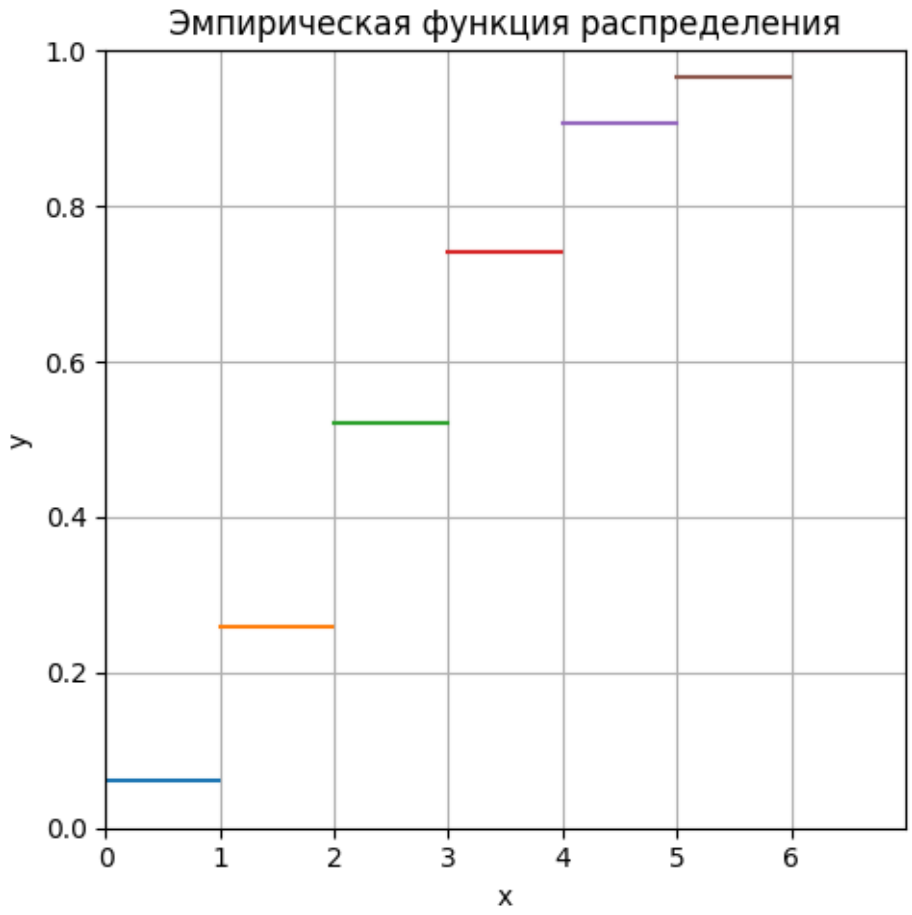
w_i	0.06	0.2	0.26	0.22	0.1666	0.06	0.0333

Полигон относительных частот:



Эмпирическая функция распределения:

$$F_{150}^{э}(x) = \sum_{x_i \le x} w_i = \begin{cases} 0, & x < 0 \\ 0.06,, & 0 \le x < 1 \\ 0.26, & 1 \le x < 2 \\ 0.52, & 2 \le x < 3 \\ 0.74, & 3 \le x < 4 \\ 0.9066, & 4 \le x < 5 \\ 0.9666, & 5 \le x < 6 \\ 1, x \ge 6 \end{cases}$$



Выборочное среднее: 2.5466

Выборочная дисперсия: 2.1011

Выборочное среднее квадратическое отклонение: 1.4495

Выборочная мода: 2

Выборочная медиана: 2

Выборочный коэффициент асимметрии: 0.3340

Выборочный коэффициент эксцесса: -0.4272

Задание 2

p = 0.19

Выборка:

Неупорядоченная:

1	3	24	0	2	8	0	7	0	11	10	0	2	9	1
6	1	12	6	2	1	3	0	0	6	4	16	1	2	1
14	6	0	14	3	0	1	10	13	7	11	2	3	2	1
0	6	1	5	4	8	3	5	2	2	0	10	4	1	5
0	10	4	5	3	0	17	1	0	0	2	2	11	1	32
2	2	4	5	0	1	4	7	5	2	11	11	0	2	3
4	7	1	11	0	3	3	9	1	5	2	9	0	0	0
4	0	5	0	3	5	3	1	0	0	1	10	5	0	0
15	3	21	1	1	3	1	10	7	4	8	3	10	15	8
3	0	10	34	2	20	4	5	9	1	8	3	11	10	5

Упорядоченная:

0	0	0	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	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	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	6	6	6
6	6	7	7	7	7	7	8	8	8	8	8	9	9	9
9	10	10	10	10	10	10	10	10	10	11	11	11	11	11
11	11	12	13	14	14	15	15	16	17	20	21	24	32	34

Статистический ряд:

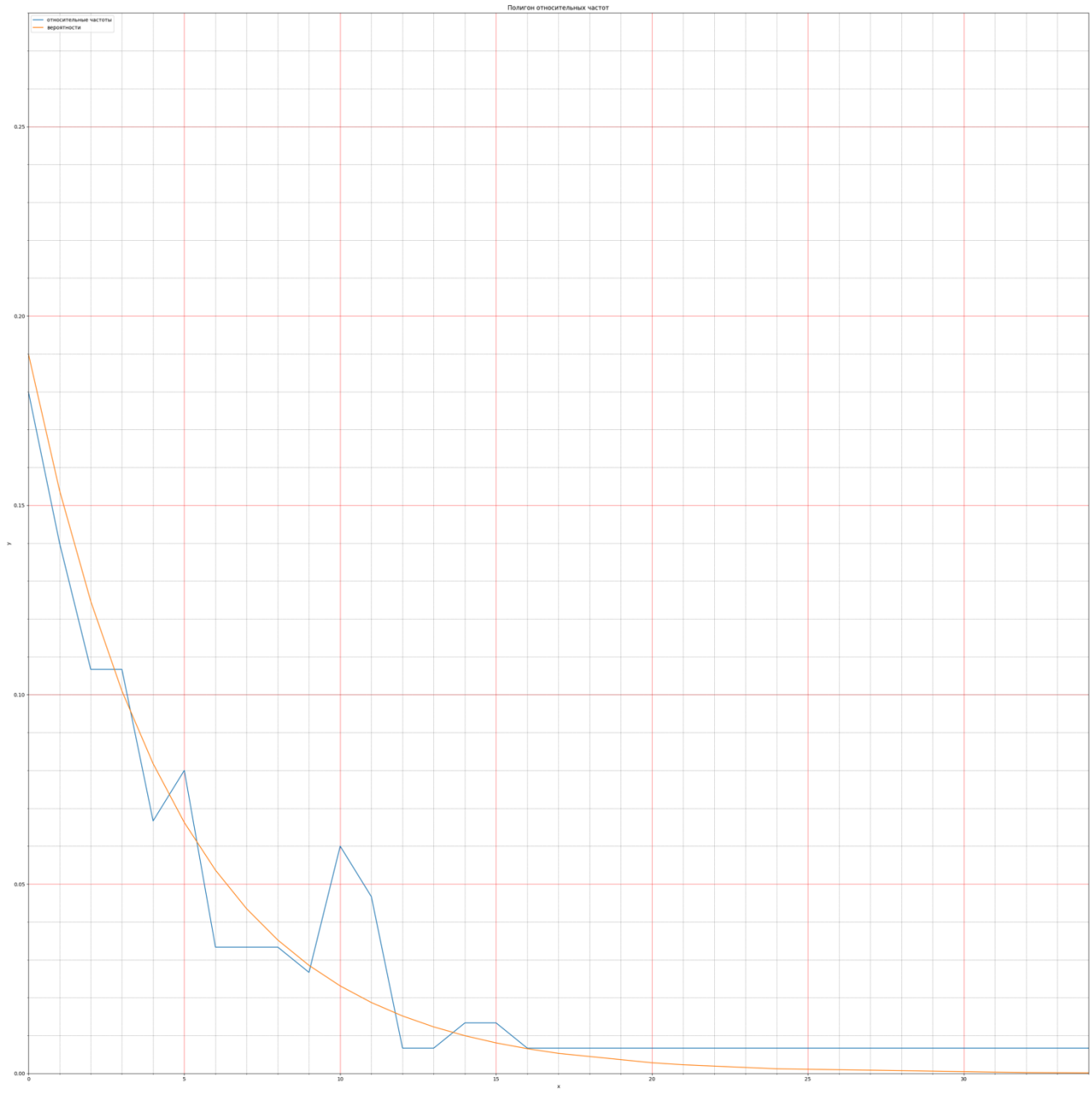
x_i	0	1	2	3	4	5
n_i	27	21	16	16	10	12
w_i	0.18	0.14	0.10667	0.10667	0.0667	0.08

x_i	6	7	8	9	10	11
n_i	5	5	5	4	9	7
w_i	0.0333	0.0333	0.0333	0.0266	0.06	0.04

x_i	12	13	14	15	16	17
n_i	1	1	2	2	1	1
w_i	0.0066	0.0066	0.0133	0.0133	0.0066	0.0066

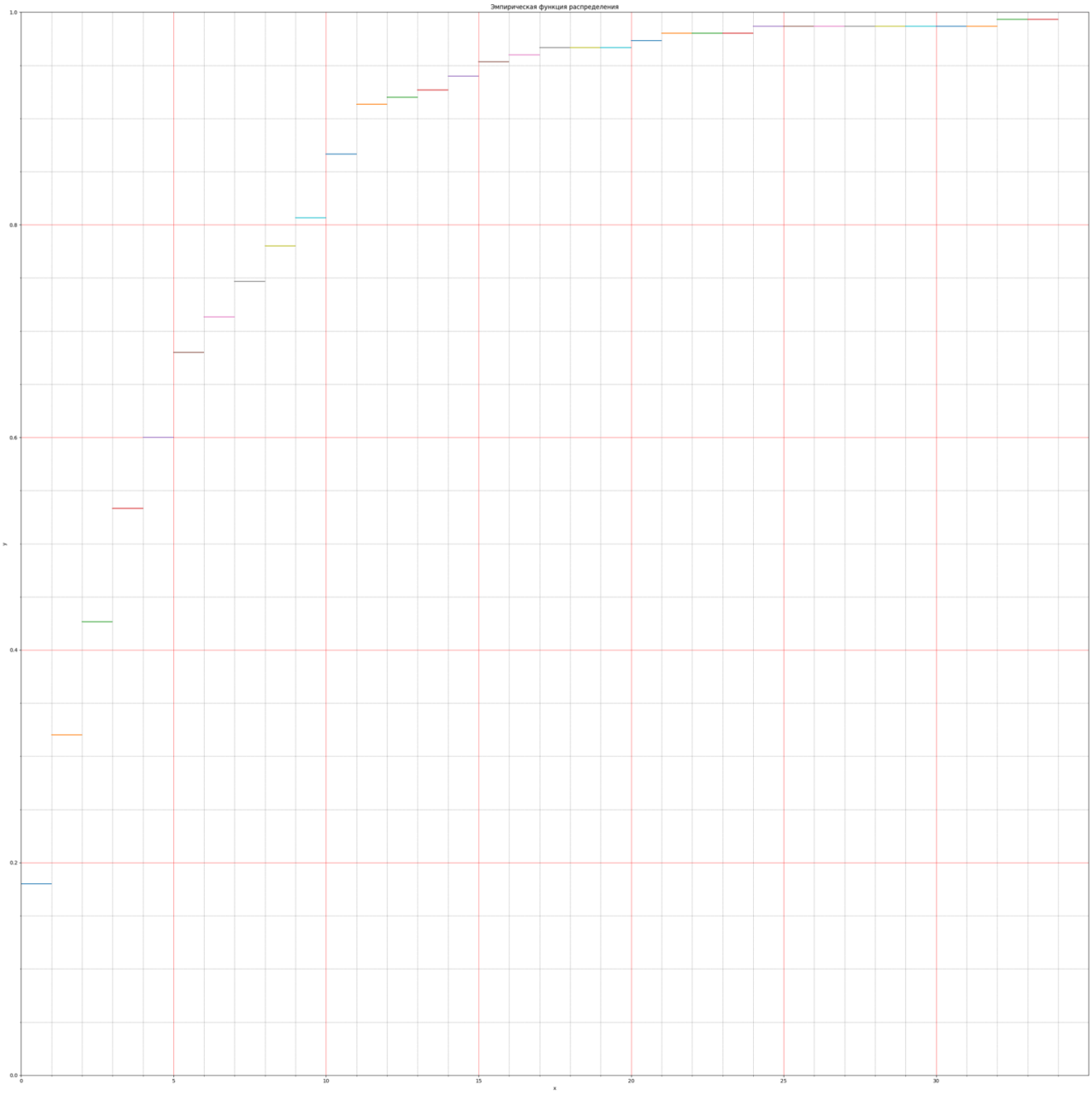
x_i	20	21	24	32	34
n_i	1	1	1	1	1
w_i	0.0066	0.0066	0.0066	0.0066	0.0066

Полигон относительных частот:



Эмпирическая функция распределения:

$$F_{150}^{Э}(x) = \sum_{x_i \le x} w_i = \begin{cases} 0, & x < 0 \\ 0.18, & 0 \le x < 1 \\ 0.32, & 1 \le x < 2 \\ 0.4266, & 2 \le x < 3 \\ 0.5333, & 3 \le x < 4 \\ 0.6, & 4 \le x < 5 \\ 0.6799, & 5 \le x < 6 \\ 0.7133, & 6 \le x < 7 \\ 0.7466, & 7 \le x < 8 \\ 0.7799, & 8 \le x < 9 \\ 0.8066, & 9 \le x < 10 \\ 0.8666, & 11 \le x < 12 \\ 0.9133, & 12 \le x < 13 \\ 0.9199, & 13 \le x < 14 \\ 0.9266, & 14 \le x < 15 \\ 0.9399, & 15 \le x < 16 \\ 0.9533, & 17 \le x < 18 \\ 0.9599, & 18 \le x < 19 \\ 0.9666, & 19 \le x < 20 \\ 0.9666, & 20 \le x < 21 \\ 0.9666, & 21 \le x < 22 \\ 0.9733, & 22 \le x < 23 \\ 0.98, & 23 \le x < 24 \\ 0.98, & 24 \le x < 25 \\ 0.98, & 25 \le x < 26 \\ 0.9866, & 26 \le x < 27 \\ 0.9866, & 27 \le x < 28 \\ 0.9866, & 28 \le x < 29 \\ 0.9866, & 29 \le x < 30 \\ 0.9866, & 30 \le x < 31 \\ 0.9866, & 31 \le x < 32 \\ 0.9933, & 32 \le x < 33 \\ 0.9933, & 33 \le x < 34 \\ 1, & x \ge 34 \end{cases}$$



Выборочное среднее: 5.04

Выборочная дисперсия: 33.2784

Выборочное среднее квадратическое отклонение: 5.7687

Выборочная мода: 0

Выборочная медиана: 3.0

Выборочный коэффициент асимметрии: 2.1863

Выборочный коэффициент эксцесса: 6.5721

Задание 3

$\lambda = 1.33$

Выборка:

Неупорядоченная:

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

Упорядоченная:

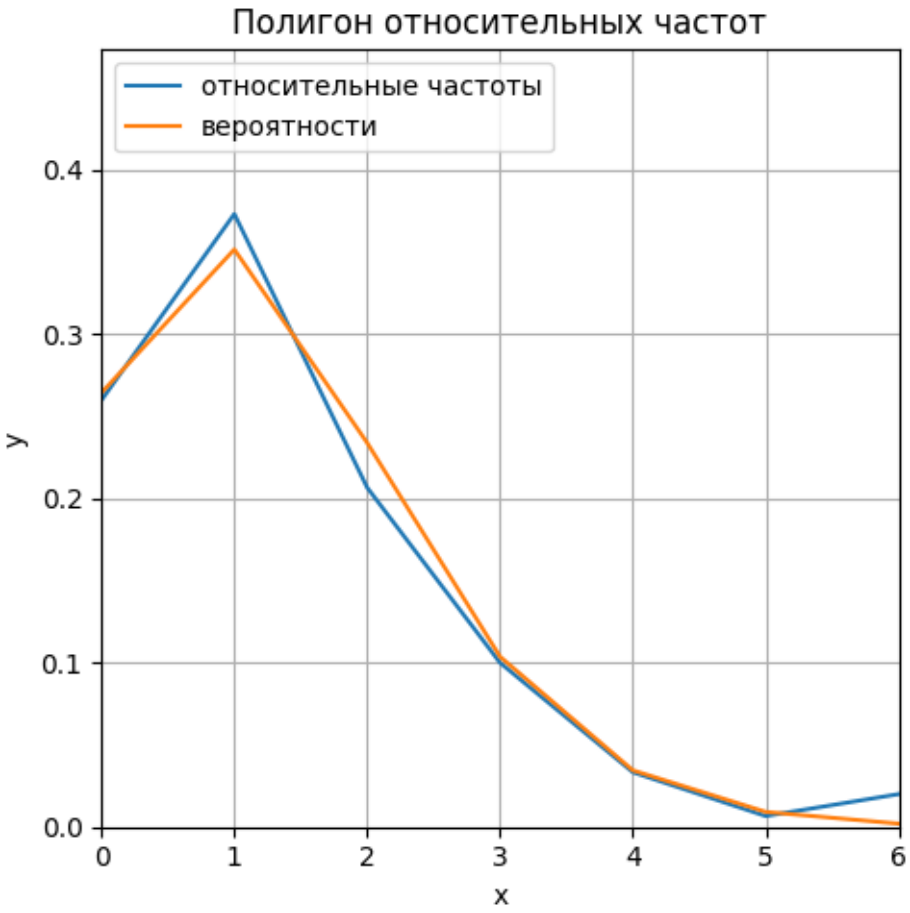
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	4	4	4	4	4	5	6	6	6

Статистический ряд:

x_i	0	1	2	3	4
n_i	39	56	31	15	5
w_i	0.426	0.3733	0.2066	0.01	0.0333

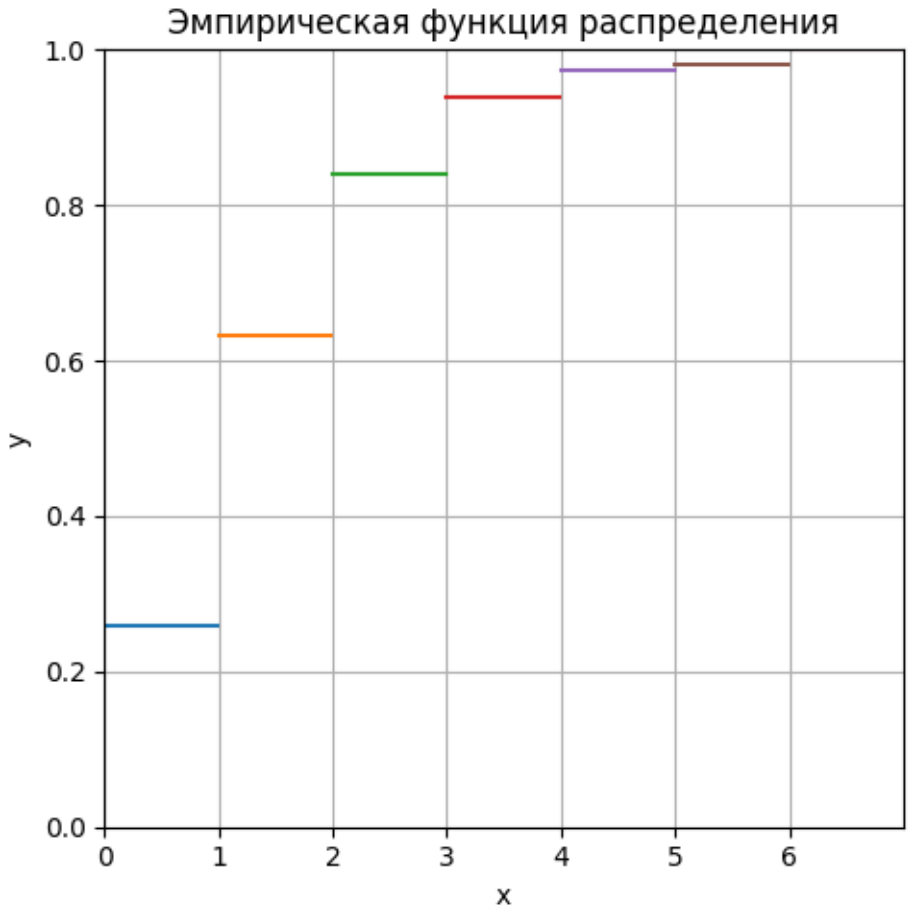
x_i	5	1
n_i	1	3
w_i	0.0067	0.02

Полигон относительных частот:



Эмпирическая функция распределения:

$$F_{150}^{э}(x) = \sum_{x_i \le x} w_i = \begin{cases} 0, & x < 0 \\ 0.26, & 0 \le x < 1 \\ 0.6333, & 1 \le x < 2 \\ 0.84, & 2 \le x < 3 \\ 0.94, & 3 \le x < 4 \\ 0.9733, & 4 \le x < 5 \\ 0.98, & 5 \le x < 6 \\ 1, & x \ge 6 \end{cases}$$



Выборочное среднее: 1.3733

Выборочная дисперсия: 1.6339

Выборочное среднее квадратическое отклонение: 1.2782

Выборочная мода: 1

Выборочная медиана: 1

Выборочный коэффициент асимметрии: 1.2885

Выборочный коэффициент эксцесса: 2.0822

Анализ результатов

Задание 1

n = 14, q = 0,81, p = 0,19

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	2.5466	2.66	0.1134	4.4529%
Выборочная дисперсия	2.1011	2.1546	0.0535	2.5462%
Выборочное среднее квадратичное отклонение	1.4495	1.4678	0.0183	1.2625%
Выборочная мода	2	2	0	0%
Выборочная медиана	2	3	1	50%
Выборочный коэффициент асимметрии	0.334	0.4223	0.3889	26.4371%
Выборочный коэффициент эксцесса	-0.4272	0.0355	0.4627	203.3802%

$\{|w_i-p_i|\}$=[0.0076, 0.0281, 0.0020407421518351954, 0.0258, 0.008, 0.0144, 0.0071]

$\max\{|w_i-p_i|, i=1,\ldots,m\}$=0.0281

Задание 2

q = 0,81, p = 0,19

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	5.04	4.2631	0.7769	15.4146%
Выборочная дисперсия	33.2784	22.4376	10.8408	32.576%
Выборочное среднее квадратичное отклонение	5.7687	4.7368	1.0319	17,8879%
Выборочная мода	0.0	0.0	0	0%
Выборочная медиана	3.0	3.0	0	0%
Выборочный коэффициент асимметрии	2.1863	2.0111	0.1752	8.0135%
Выборочный коэффициент эксцесса	6.5721	6.0445	0.5276	8.0278%

$\{|w_i-p_i|\}$=[0.01, 0.0138, 0.0179, 0.0056, 0.0151, 0.0137, 0.0203, 0.0101, 0.0018, 0.0018, 0.0369, 0.0279, 0.0084, 0.0056, 0.0033, 0.0052, 0.0001, 0.0013, 0.0038, 0.0043, 0.0054, 0.0064, 0.0065]

$\max\{|w_i-p_i|, i=1,\ldots,m\} = 0.0369$

Задание 3

$\lambda = 1.33$

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	1.3733	1.33	0.0433	3.2556%
Выборочная дисперсия	1.6339	1.33	0.3039	22.8496%
Выборочное среднее квадратичное отклонение	1.2782	1.1532	0.125	10.8394%
Выборочная мода	1	1	0	0%
Выборочная медиана	1	1	0	0%
Выборочный коэффициент асимметрии	1.2885	0.8671	0.4214	48.5987%
Выборочный коэффициент эксцесса	2.0822	0.7518	1.3304	76.9619%

$\{|w_i-p_i|\}=[0.0044, 0.0215, 0.0272, 0.0037, 0.0011, 0.0025, 0.0179]$
$\max\{|w_i-p_i|, i=1,\ldots,m\}=0.0179$

Литература по математической статистике

1. Гмурман В.Е. Теория вероятностей и математическая статистика: Учеб. пособие
для вузов — М.: Высш. образов., 2006. — 480 с.
2. Математическая статистика [Электронный ресурс]: метод. указания по выполнению лаб. работ / А.А.Лобузов — М.: МИРЭА, 2017. — Электрон. опт. диск (ISO)
3. Справочное пособие по теории вероятностей и математической статистике
(законы распределения): Учеб.пособие для вузов / Г.А.Соколов, Н.А. Чистякова. —
М.: Высш. шк., 2007. — 248 с.

Приложение

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import sys
import argparse
from collections import Counter
from itertools import islice

from functools import reduce
from scipy import stats as st
import numpy as np
from glob import glob
import pickle
import re

import matplotlib.pyplot as plt

variant = 9

def createParser():
    parser = argparse.ArgumentParser()
    parser.add_argument('-w', '--write', default=0, type=int)
    parser.add_argument('-r', '--read', default=1, type=int)
    return parser

def save(r, type_ryad):
    reg = 'data/' + type_ryad + '*'
    files = glob(reg) # type: ryadN.pickle
    if files:
        n = int(re.sub(r'[^\d]', '', files[0]))
        n += 1
    else:
        n = 0
    filename = reg[:-1] + str(n) + '.pickle'
    with open(filename, 'wb') as f:
        pickle.dump(r, f)

def load(type_ryad, n = -1):
    reg = 'data/' + type_ryad + '*'
    files = glob(reg)
    if not files:
        return
    if abs(n) > len(files):
        return
    with open(files[n], 'rb') as f:
        return(pickle.load(f))

def draw_poligon(count, pr, name, sizex=5, sizey=5):
    X = list(count.keys())
```

```
    Y = [count[x][1] for x in X]
    fig, ax = plt.subplots(figsize=(sizex, sizey))
    ax.plot(X, Y, label='относительные частоты')
    ax.plot(X, pr, label='вероятности')
    ax.set_title('Полигон относительных частот')
    ax.legend(loc='upper left')
    ax.set_ylabel('y')
    ax.set_xlabel('x')
    ax.set_xlim(xmin=0, xmax=max(X))
    ax.set_ylim(ymin=0, ymax=max(Y) + 0.1)
    if sizex != 5:
        # Don't allow the axis to be on top of your data
        ax.set_axisbelow(True)

        # Turn on the minor TICKS, which are required for the minor GRID
        ax.minorticks_on()

        # Customize the major grid
        ax.grid(which='major', linestyle='-', linewidth='0.5', color='red')
        # Customize the minor grid
        ax.grid(which='minor', linestyle=':', linewidth='0.5', col-
or='black')
    else:
        ax.grid()
    fig.tight_layout()
    fig.savefig('data/poligon_' + name + '.png')

def draw_cdf(Xlist, Ylist, name, sizex=5, sizey=5):
    fig, ax = plt.subplots(figsize=(sizex, sizey))

    for X, Y in zip(Xlist, Ylist):
        ax.plot(X, Y, label='')
    ax.set_title('Эмпирическая функция распределения')
    ax.legend(loc='upper left')
    ax.set_ylabel('y')
    ax.set_xlabel('x')
    ax.set_xlim(xmin=0, xmax=max(X))
    ax.set_ylim(ymin=0, ymax=1)
    fig.tight_layout()
    if sizex != 5:
        # Don't allow the axis to be on top of your data
        ax.set_axisbelow(True)

        # Turn on the minor TICKS, which are required for the minor GRID
        ax.minorticks_on()

        # Customize the major grid
        ax.grid(which='major', linestyle='-', linewidth='0.5', color='red')
        # Customize the minor grid
        ax.grid(which='minor', linestyle=':', linewidth='0.5', col-
or='black')
```

```
101     else:
102         ax.grid()
103     fig.savefig('data/cdf_' + name + '.png')
104 
105 def expect(nv, key):
106     if key in nv.keys():
107         return nv[key][1]
108     else:
109         return 0.0
110     # elif key not in nv.keys() and key != 0:
111     #     return expect(nv, key-1)
112     # else:
113     #     return 0.0
114 
115 def generate_ef(nv, r):
116     delta = 0.01
117     items = list(zip(list(range(max(r) + 2)), islice(list(range(max(r) +
118 2)), 1, None))) # [(0, 1), (1, 2), ...]
119     keys = list(range(list(nv.keys())[-1] + 1)) # list(nv.keys())
120     Y = [sum([expect(nv, key) for key in keys[:i + 1]]) for i, el in enume-
121 rate(keys)]
122     Xlist = [[x * delta for x in range(item[0] * int(1 / delta), item[1] *
123 int(1 / delta))]
124                 for item in items]
125     Ylist = [[y] * len(Xlist[i]) for i, y in enumerate(Y)]
126     # Ylist = [[st.binom.cdf(x, n, p) for x in line] for line in Xlist]
127     return Xlist, Ylist, Y
128 
129 def exp_stats(nv, r, st_r):
130     mean = sum([key * val[1] for key, val in nv.items()]) # mean
131     var = sum([(key - mean) ** 2 * val[1] for key, val in nv.items()])
132     standart = var ** 0.5
133     mu = lambda k: sum([(key - mean) ** k * val[1] for key, val in
134 nv.items()])
135     skew = mu(3) / (standart ** 3)
136     kurtosis = mu(4) / (standart ** 4) - 3
137     mode = st.mode(r)
138     if len(st_r) % 2 != 0:
139         med = st_r[int(len(st_r) / 2)]
140     else:
141         med = 0.5 * (st_r[int(len(st_r) / 2)] + st_r[int(len(st_r) / 2) -
142 1])
143     return mean, var, standart, skew, kurtosis, mode, med
144 
145 def binomial(np, size, write, read):
146     n, p = np
147     r = load('binomial') if read else st.binom.rvs(n, p, size=size)
148     if write:
149         save(r, 'binomial')
150 
151     st_r = sorted(r)
```

```
152     p_ch = Counter(r)
153     nv = {key: (val, val / sum(p_ch.values())) for key, val in p_ch.items()}
154 
155     ex_mean, ex_var, ex_standart, ex_skew, ex_kurtosis, ex_mode, ex_med =
156 exp_stats(nv, r, st_r)
157 
158     pr = [st.binom.pmf(x, n, p) for x in list(nv.keys())]
159     draw_poligon(nv, pr, 'binomial')
160 
161     # delta = 0.01
162     # items = list(zip(list(range(max(r) + 2)), islice(list(range(max(r) +
163 2)), 1, None))) # [(0, 1), (1, 2), ...]
164     # Xlist = [[x * delta for x in range(item[0] * int(1 / delta), item[1] *
165 int(1 / delta))]
166     #           for item in items]
167     # Ylist = [[st.binom.cdf(x, n, p) for x in line] for line in Xlist]
168 
169     Xlist, Ylist, Y = generate_ef(nv, r)
170 
171     draw_cdf(Xlist, Ylist, 'binomial')
172 
173     cdf_func = [el[0] for el in Ylist]
174     # cdf_func.append(1.0)
175     # cdf_func.insert(0, 0.0)
176 
177     mean, variance, skew, kurtosis = st.binom.stats(n, p, moments='mvsk') #
178 среднее, дисперсия, ассиметрия, эксцесс
179     standart, med = st.binom.std(n, p), st.binom.median(n, p) # стантартное
180 отклонение, медиана
181     mode = st.mode(r) # мода
182 
183     pmf = [abs(val[1] - st.binom.pmf(key, n, p)) for key, val in nv.items()]
184 
185     return (r, st_r, cdf_func, nv, float(mean),
186                                    float(variance),
187                                    float(skew),
188                                    float(kurtosis),
189                                    standart,
190                                    med,
191                                    float(mode.mode), ex_mean,
192                                                      ex_var,
193                                                      ex_standart,
194                                                      ex_skew,
195                                                      ex_kurtosis,
196                                                      float(ex_mode.mode),
197                                                      ex_med, pmf, Y)
198 
199 def geometr(p, size, write, read):
200     r = load('geometr') if read else st.geom.rvs(p, loc=-1, size=size)
201     if write:
202         save(r, 'geometr')
```

```
203
204     st_r = sorted(r)
205     p_ch = Counter(r)
206     nv = {key: (val, val / sum(p_ch.values())) for key, val in p_ch.items()}
207
208     ex_mean, ex_var, ex_standart, ex_skew, ex_kurtosis, ex_mode, ex_med =
209 exp_stats(nv, r, st_r)
210
211     pr = [st.geom.pmf(x, p, loc=-1) for x in list(nv.keys())]
212     draw_poligon(nv, pr, 'geometr', 30, 30)
213
214     # delta = 0.01
215     # items = list(zip(list(range(max(r) + 2)), islice(list(range(max(r) +
216 2)), 1, None))) # [(0, 1), (1, 2), ...]
217     # Xlist = [[x * delta for x in range(item[0] * int(1 / delta), item[1] *
218 int(1 / delta))]
219     #           for item in items]
220     # Ylist = [[st.geom.cdf(x, p) for x in line] for line in Xlist]
221
222     Xlist, Ylist, Y = generate_ef(nv, r)
223
224     draw_cdf(Xlist, Ylist, 'geometr', 30, 30)
225
226     cdf_func = [el[0] for el in Ylist]
227     cdf_func.append(1.0)
228     cdf_func.insert(0, 0.0)
229
230     mean, variance, skew, kurtosis = st.geom.stats(p, loc=-1, mo-
231 ments='mvsk') # среднее, дисперсия, ассиметрия, эксцесс
232     standart, med = st.geom.std(p, loc=-1), st.geom.median(p, loc=-1) #
233 стантартное отклонение, медиана
234     mode = st.mode(r) # мода
235
236     pmf = [abs(val[1] - st.geom.pmf(key, p, loc=-1)) for key, val in
237 nv.items()]
238
239     return (r, st_r, cdf_func, nv, float(mean),
240                                     float(variance),
241                                     float(skew),
242                                     float(kurtosis),
243                                     standart,
244                                     med,
245                                     float(mode.mode), ex_mean,
246                                                       ex_var,
247                                                       ex_standart,
248                                                       ex_skew,
249                                                       ex_kurtosis,
250                                                       float(ex_mode.mode),
251                                                       ex_med, pmf, Y)
252
253 def puasson(mu, size, write, read):
```

```
254     r = load('puasson') if read else st.poisson.rvs(mu, size=size)
255     if write:
256         save(r, 'puasson')
257 
258     st_r = sorted(r)
259     p_ch = Counter(r)
260     nv = {key: (val, val / sum(p_ch.values())) for key, val in p_ch.items()}
261 
262     ex_mean, ex_var, ex_standart, ex_skew, ex_kurtosis, ex_mode, ex_med =
263 exp_stats(nv, r, st_r)
264 
265     pr = [st.poisson.pmf(x, mu) for x in list(nv.keys())]
266     draw_poligon(nv, pr, 'puasson')
267 
268     # delta = 0.01
269     # items = list(zip(list(range(max(r) + 2)), islice(list(range(max(r) +
270 2)), 1, None))) # [(0, 1), (1, 2), ...]
271     # Xlist = [[x * delta for x in range(item[0] * int(1 / delta), item[1] *
272 int(1 / delta))]
273     #           for item in items]
274     # Ylist = [[st.poisson.cdf(x, mu) for x in line] for line in Xlist]
275 
276     Xlist, Ylist, Y = generate_ef(nv, r)
277 
278     draw_cdf(Xlist, Ylist, 'puasson')
279 
280     cdf_func = [el[0] for el in Ylist]
281     cdf_func.append(1.0)
282     cdf_func.insert(0, 0.0)
283 
284     mean, variance, skew, kurtosis = st.poisson.stats(mu, moments='mvsk') #
285 среднее, дисперсия, ассиметрия, эксцесс
286     standart, med = st.poisson.std(mu), st.poisson.median(mu) # стантартное
287 отклонение, медиана
288     mode = st.mode(r) # мода
289 
290     pmf = [abs(val[1] - st.poisson.pmf(key, mu)) for key, val in nv.items()]
291 
292     return (r, st_r, cdf_func, nv, float(mean),
293                                    float(variance),
294                                    float(skew),
295                                    float(kurtosis),
296                                    standart,
297                                    med,
298                                    float(mode.mode), ex_mean,
299                                                      ex_var,
300                                                      ex_standart,
301                                                      ex_skew,
302                                                      ex_kurtosis,
303                                                      float(ex_mode.mode),
304                                                      ex_med, pmf, Y)
```

```
305
306 def save_result(result):
307     reg = 'data/result*'
308     files = glob(reg)
309     if files:
310         n = int(re.sub(r'[^\d]', '', files[0]))
311         n += 1
312     else:
313         n = 0
314     filename = reg[:-1] + str(n) + '.txt'
315     with open(filename, 'w') as f:
316         f.write(result)
317
318
319 if __name__ == '__main__':
320     parser = createParser()
321     namespace = parser.parse_args(sys.argv[1:])
322
323     result = ['выборка: {0}',
324               'упорядоченная: {1}',
325               'эмпирическая функция распределения: {2}',
326               'статистический ряд: {3}',
327               'теор. среднее: {4} <=> среднее: {11}',
328               'теор. дисперсия: {5} <=> дисперсия: {12}',
329               'теор. ассиметрия: {6} <=> ассиметрия: {14}',
                  'теор. эксцесс: {7} <=> эксцесс: {15}',
                  'теор. среднее квадратичное отклонение: {8} <=> среднее квад-
    ратичное отклонение: {13}',
                  'теор. медиана: {9} <=> медиана: {17}',
                  'теор. мода: {10} <=> мода: {16}',
                  'pmf: {18}',
                  'ex_pmf {19}']

         line = '\n\n'.join(result)

         task = {'binomial': (binomial, [5 + variant % 17, 0.1 + 0.01 * va-
    riant]),
                 'geometr': (geometr, 0.1 + 0.01 * variant),
                 'puasson': (puasson, 0.7 + 0.07 * variant)}

         out = ''
         for r_type, fdata in task.items():
             func, params = fdata
             res = func(params, size=150,
                                write = namespace.write,
                                read = namespace.read)

             out += r_type + '\n\n' + line.format(*res) + '\n\n<' + '='*50 +
    '>\n\n'

         save_result(out)
```