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

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 1

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

ВАРИАНТ 6

Тема: _____**Первичная обработка выборки**_____

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

Группа: КМБО-01-20

МОСКВА – 2023

Задание

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

$$p_k = C_n^k \cdot p^k \cdot q^{n-k},\ k = 0,1,\ldots,n. \qquad n = 5 + V \bmod 19,\ p = 0,2 + 0,005V.$$

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

$$p_k = q^k \cdot p,\ k = 0,1,\ldots \qquad p = 0,2 + 0,005V.$$

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

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda},\ k = 0,1,\ldots \qquad \lambda = 0,5 + 0,01 \cdot V.$$

Задание 4. Получить выборку сгенерировав 200 псевдослучайных чисел распределенных равномерно на множестве {0 ... n-1}:

$$p_k = \frac{1}{n},\ k = 0,\ldots,(n-1).\ n = 5 + V \bmod 28.$$

Задание 5. Получить выборку сгенерировав 200 псевдослучайных чисел распределенных по гипергеометрическому закону с параметрами M K m:

$$p_k = \frac{C_K^k \cdot C_{M-K}^{m-k}}{C_M^m},\ k = 0,1,\ldots,m. \quad m = 5 + V \bmod 7,\ M = 2m + 3,\ K = m + 1 + V \bmod 2.$$

Для всех выборок в Заданиях 1-5 построить:

1) статистический ряд;
2) полигон относительных частот;
3) график эмпирической функции распределения;

Найти:

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

составить таблицы:

1) сравнения относительных частот и теоретических вероятностей;
2) сравнения рассчитанных характеристик с теоретическими значениями

Задание 6. Получить выборку сгенерировав 200 псевдослучайных чисел распределенных по нормальному закону с параметрами

$$a=(-1)^V\cdot 0{,}05\cdot V \quad \text{и} \quad \sigma^2 \text{, где } \sigma=0{,}005\cdot V+1.$$

Задание 7. Получить выборку сгенерировав 200 псевдослучайных чисел распределенных по показательному закону с параметром

$$\lambda=2+(-1)^V\cdot 0{,}01\cdot V\,.$$

Задание 8. Получить выборку сгенерировав 200 псевдослучайных чисел распределенных равномерно на отрезке [a b]

$$a=(-1)^V\cdot 0{,}02\cdot V\,,\quad b=a+6\,.$$

Для всех выборок в Заданиях 6-8

построить:

1) интервальный ряд и ассоциированный статистический ряд;
2) гистограмму относительных частот;
3) график эмпирической функции распределения;

найти:

1) выборочное среднее;
2) выборочную дисперсию с поправкой Шеппарда;
3) выборочное среднее квадратическое отклонение;
4) выборочную моду;
5) выборочную медиану;
6) выборочный коэффициент асимметрии;
7) выборочный коэффициент эксцесса;

составить таблицы:

1) сравнения относительных частот и теоретических вероятностей попадания в интервалы;
2) сравнения рассчитанных характеристик с теоретическими значениями.

Краткие теоретические сведения

Статистический ряд распределения представляет собой упорядоченное распределение единиц изучаемой совокупности на группы по определенному варьирующему признаку.

Полигоном относительных частот называют ломаную, отрезки которой соединяют точки (x_1; $W_{2)}$, (x_2; W_2), . . . (x_k; W_k). Для построения полигона относительных частот на оси абсцисс откладывают варианты x_i, а на оси ординат — соответствующие им относительные частоты W_I. Точки

(x_i; W_i) соединяют отрезками прямых и получают полигон относительных частот.

Эмпирической функцией распределения (функцией распределения выборки) называют функцию F *(x), определяющую для каждого значения x относительную частоту события X ≤ x.

Биномиальное распределение:

$$p_k = C_n^k \cdot p^k \cdot q^{n-k},\ k = 0,1,\ldots,n$$

Математическое ожидание: np

Дисперсия: npq q=1-p

Среднее квадратическое отклонение: $\sqrt{npq}$

Мода:$[(\mathrm{n}+\ 1)\mathrm{p}\ ]$ если $(\ \mathrm{n}+1)\mathrm{p}\ -$ дробное;

$(\mathrm{n}+\ 1)\mathrm{p}-\frac{1}{2}$ если $(\mathrm{n}+\ 1)\mathrm{p}\ -$ целое;

Медиана: Round(np)

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

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

Геометрическое распределение:

$$p_k = q^k \cdot p,\ k = 0,1,\ldots$$

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

Дисперсия: $\frac{q}{p^2}$ q=1-p

Среднее квадратическое отклонение: $\frac{\sqrt{q}}{p}$

Мода: 0

Медиана: $[-\frac{ln2}{lnq}]$ если $\frac{ln2}{lnq}$ – дробное

$-\frac{ln2}{lnq}-\ \frac{1}{2}$ если $\frac{ln2}{lnq}$ - целое

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

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

Распределение Пуассона:

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda},\ k = 0,1,\ldots$$

Математическое ожидание:λ

Дисперсия: λ

Среднее квадратическое отклонение:$\sqrt{\lambda}$

Мода: [λ]

Медиана: [$\lambda + \frac{1}{3} - \frac{0.02}{\lambda}$]

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

Коэффициент эксцесса: λ^{-1}

Равномерное распределение на множестве:

$$p_k = \frac{1}{n},\ k = 0,\ldots,(n-1)$$

Математическое ожидание: $\frac{n-1}{2}$

Дисперсия: $\frac{n^2-1}{12}$

Среднее квадратическое отклонение: $\frac{1}{2}\sqrt{\frac{n^2-1}{3}}$

Мода: $\frac{n-1}{2}$

Медиана: $\frac{n-1}{2}$

Коэффициент асимметрии: 0

Коэффициент эксцесса: $-\frac{6(n^2+1)}{5(n^2-1)}$

Гипергеометрическое распределение:

$$p_k = \frac{C_K^k \cdot C_{M-K}^{m-k}}{C_M^m},\ k = 0,1,\ldots,m$$

Математическое ожидание: $\frac{mK}{M}$

Дисперсия: $\frac{mK(M-K)(M-m)}{(M-1)M^2}$

Среднее квадратическое отклонение: $\frac{1}{M}\sqrt{\frac{mK(M-K)(M-m)}{(M-1)}}$

Мода: [$\frac{(K+1)(m+1)}{M+2}$]

Медиана: l при $\sum_{k=0}^{l-1} p_k$<0.5<$\sum_{k=0}^{l} p_k$

l+05 при $\sum_{k=0}^{l} p_k$=0.5

Коэффициент асимметрии: $\frac{(M-2K)(M-2m)}{M-2} * \sqrt{\frac{M-1}{mK(M-K)(M-m)}}$

Коэффициент эксцесса: $[\frac{(M-1)M^2}{m(M-2)(M-3)(M-m)}]*[\frac{M(M+1)-6M(M-m)}{K(M-K)} + \frac{3m(M+6)(M-m)}{M^2} - 6]$

Эмпирическая функция распределения:

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

где, x_i^* - значение встречающееся в выборке, w_i– относительная частота (частость) значения x_i^* ($w_i = \frac{n_i}{N}$, где N – размер выборки, n_i – частота x_i^* (число значений x_i^*, встречающихся в выборке).

Полигон относительных частот – ломаная линия, соединяющая последовательно точки с координатами (0, $\widetilde{w}_0$), (1, $\widetilde{w}_1$),…, (M, $\widetilde{w}_M$), где M = $x_m^* = \max\{x_i^* : 1 \le i \le m\}$; $\widetilde{w}_j = w_i$, если существует такое x_i^*, что j = x_i^*, и $\widetilde{w}_j = 0$ в противном случае.

Выборочное среднее:

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

Выборочный момент k-ого порядка (выборочный k-ый момент):

$\overline{\mu_k} = \sum_{i=1}^{m}(x_i^*)^k * w_i$, $\overline{\mu_1} = \bar{x}$.

Выборочная дисперсия:

$D_B = \sum_{i=1}^{m}(x_i^* - \bar{x})^2 * w_i$, = $\overline{\mu_2}$ - $(\overline{\mu_1})^2$.

Выборочный центральный момент k-ого порядка (выборочный центральный k-ый момент):

$\bar{\mu}_k^0 = \sum_{i=1}^{m}(x_i^* - \bar{x})^k * w_i$, $\bar{\mu}_1^0 = 0$, $\bar{\mu}_2^0 = D_B$,

$\bar{\mu}_3^0 = \overline{\mu_3} - 3\overline{\mu_2}\,\overline{\mu_1} + 2(\overline{\mu_1})^3$,

$\bar{\mu}_4^0 = \overline{\mu_4} - 4\overline{\mu_3}\,\overline{\mu_1} + 6\,\overline{\mu_2}\,(\overline{\mu_1})^2 - 3(\overline{\mu_1})^4$.

Выборочное среднее квадратическое отклонение:

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

Выборочная мода $\overline{M_0}$

$\overline{M_0}$ = { $x_i^* \mid n_i = \max n_k$ }, если $n_i = \max n_k > n_j$, i ≠ j;

если $n_i = n_{i+1} = \dots = n_{i+j} = \max n_k$, то $\overline{M_0} = \frac{1}{2}(x_i^* + x_{i+j}^*)$,

если $n_i = n_j = \max n_k > n_j$, i < l <j , то $\overline{M_0}$ – не существует.

Выборочная медиана:

$$\bar{M}_e = \begin{cases} x_i^*, & F_N^{Э}(x_{i-1}^*) < 0{,}5 < F_N^{Э}(x_i^*), \\ \frac{1}{2}(x_i^* + x_{i+1}^*), & F_N^{Э}(x_i^*) = 0{,}5 . \end{cases}$$

Выборочный коэффициент асимметрии:

$$\bar{\gamma}_1 = \frac{\bar{\mu}_3^0}{\bar{\sigma}^3}$$

Выборочный коэффициент эксцесса:

$$\bar{\gamma}_2 = \frac{\bar{\mu}_4^0}{\bar{\sigma}^4} - 3$$

Интервальный ряд - это вариационный ряд, варианты которого представлены в виде интервалов.

Гистограммой относительных частот называют ступенчатую фигуру, состоящую из прямоугольников, основаниями которых служат частичные интервалы длиною h, а высоты равны отношению W_i/h (плотность относительной частоты).

Нормальное распределение:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}, \; x \in (-\infty, +\infty)$$

Математическое ожидание: α
Дисперсия: σ^2
Среднее квадратическое отклонение: σ
Мода: α
Медиана: α
Коэффициент асимметрии: 0
Коэффициент эксцесса: 0

Показательное распределение:

$$f(x)=\lambda e^{-\lambda x},\ x\in[0,+\infty)$$

Математическое ожидание: λ^{-1}

Дисперсия: λ^{-2}

Среднее квадратическое отклонение: λ^{-1}

Мода: 0

Медиана: $\frac{\ln 2}{\lambda}$

Коэффициент асимметрии: 2

Коэффициент эксцесса: 6

Равномерное распределение на отрезке [a b]:

$$f(x)=\frac{1}{b-a},\ x\in[a,b]$$

Математическое ожидание: $\frac{a+b}{2}$

Дисперсия: $\frac{(b-a)^2}{12}$

Среднее квадратическое отклонение: $\frac{b-a}{2\sqrt{3}}$

Мода: $\frac{a+b}{2}$

Медиана: $\frac{a+b}{2}$

Коэффициент асимметрии: 0

Коэффициент эксцесса: $-\frac{6}{5}$

Эмпирическая функция распределения:

$$F_N^{\ni}(x;x_1,x_2,\ldots,x_N)=\sum_{x_k\le x}\frac{1}{N}=\begin{cases}0, & x<x_{(1)},\\ \frac{1}{N}, & x_{(1)}\le x<x_{(2)},\\ \frac{2}{N}, & x_{(2)}\le x<x_{(3)},\\ \frac{3}{N}, & x_{(3)}\le x<x_{(4)},\\ \ldots\ldots\ldots\ldots\ldots\ldots, & \\ 1, & x\ge x_{(N)}.\end{cases}$$

Выборочная дисперсия с поправкой Шеппарда:

$$s_B^2 = \sum_{i=1}^{m} (x_i^* - \bar{x})^2 \cdot w_i - \frac{h^2}{12}, \text{где } h = (a_m - a_0)/m$$

Если модальный интервал (на котором высота гистограммы максимально) один, то

$$\bar{M}_0 = a_k + h \frac{w_{k+1} - w_k}{2w_{k+1} - w_k - w_{k+2}}$$

где a_k – левая граница модального интервала (a_k, a_{k+1});

a_{k+1} – правая граница модального интервала (a_k, a_{k+1});

w_{k+1} – относительная частота на модальном интервале;

w_k, w_{k+2} – относительные частоты интервалов слева и справа от модального интервала.

Если модальных интервалов несколько и все они идут подряд

(т.е. интервалы $(a_k, a_{k+1}), \ldots (a_{k+l-1}, a_{k+l})$ – все модальные), то

$$\bar{M}_0 = a_k + l \cdot h \cdot \frac{w_{k+l} - w_k}{2w_{k+l} - w_k - w_{k+l+1}}.$$

Если между модальными интервалами находятся немодальные, то считаем, что выборочной моды не существует.

Выборочная медиана:

$$\bar{M}_e = a_{k-1} + \frac{h}{w_k}\left(\frac{1}{2} - \sum_{i=1}^{k-1} w_i\right), \text{ если } \quad \sum_{i=1}^{k-1} w_i < \frac{1}{2} < \sum_{i=1}^{k} w_i;$$

$$\bar{M}_e = a_k, \text{ если } \sum_{i=1}^{k} w_i = \frac{1}{2}.$$

В программе расчёта был использован язык python. Для получения выборок использовались следующие функции:

np.random.binomial(n, p, size) - генерация size псевдослучайных чисел, распределенных по биномиальному закону с параметрами n, p. n – количество испытаний, p – вероятность успеха.

binom.pmf(k, n, p) – расчёт вероятности успеха биномиального распределения с параметрами k, n и p. n – количество испытаний, p – вероятность успеха, k – количество успешных испытаний.

np.random.geometric(p, size) - генерация size псевдослучайных чисел, распределенных по геометрическому закону с параметрами p. p – вероятность успеха.

np.random.poisson(p, size) - генерация size псевдослучайных чисел, распределенных по по закону Пуассона с параметрами p. p – значение параметра λ.

np.random.random_integers(a, b, size) - генерация size псевдослучайных чисел, распределенных равномерно на множестве {a,..., b}.

np.random.hypergeometric(K, M-K, m, size) - генерация size псевдослучайных чисел, распределенных по гипергеометрическому закону с параметрами K, M-K, m. K – количество особых объектов, m - сколько всего объектов извлекается, M-K – количество обычных объектов.

hypergeom.pdf(number, K, M-K, m) - расчёт вероятности успеха гипергеометрического распределения с параметрами number, K, M-K, m. . K – количество особых объектов, m - сколько всего объектов извлекается, M-K – количество обычных объектов, number – сколько особых объектов извлекается.

np.random.normal(α, σ, size) - генерация size псевдослучайных чисел, распределенных по нормальному закону с параметрами α, σ^2. α – мат. ожидание, σ – стандартное отклонение.

norm.pdf(number, loc, scale) - расчёт вероятности успеха нормальному распределения с параметрами number – значение, до которого вычисляется распределение, loc – мат. ожидание, scale – стандартное отклонение.

expon.rvs(scale, size) - генерация size псевдослучайных чисел, распределенных по показательному закону с параметром scale– значение параметра 1/λ.

expon.cdf(x, scale) - расчёт вероятности успеха показательного распределения с параметрами x значение, до которого вычисляется распределение, scale – значение параметра 1/λ.

np.random.uniform(a1, b, size) - генерация size псевдослучайных чисел, распределенных равномерно на отрезке [a1, b].

uniform.cdf(x, loc, scale) - расчёт вероятности успеха показательного распределения с параметрами x - значение, до которого вычисляется распределение, loc-минимальное значение, scale – длина отрезка.

Результаты расчетов

Для всех заданий вариант равен 6.
Задание 1:
n=11 ; p=0.23

Полученная выборка

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

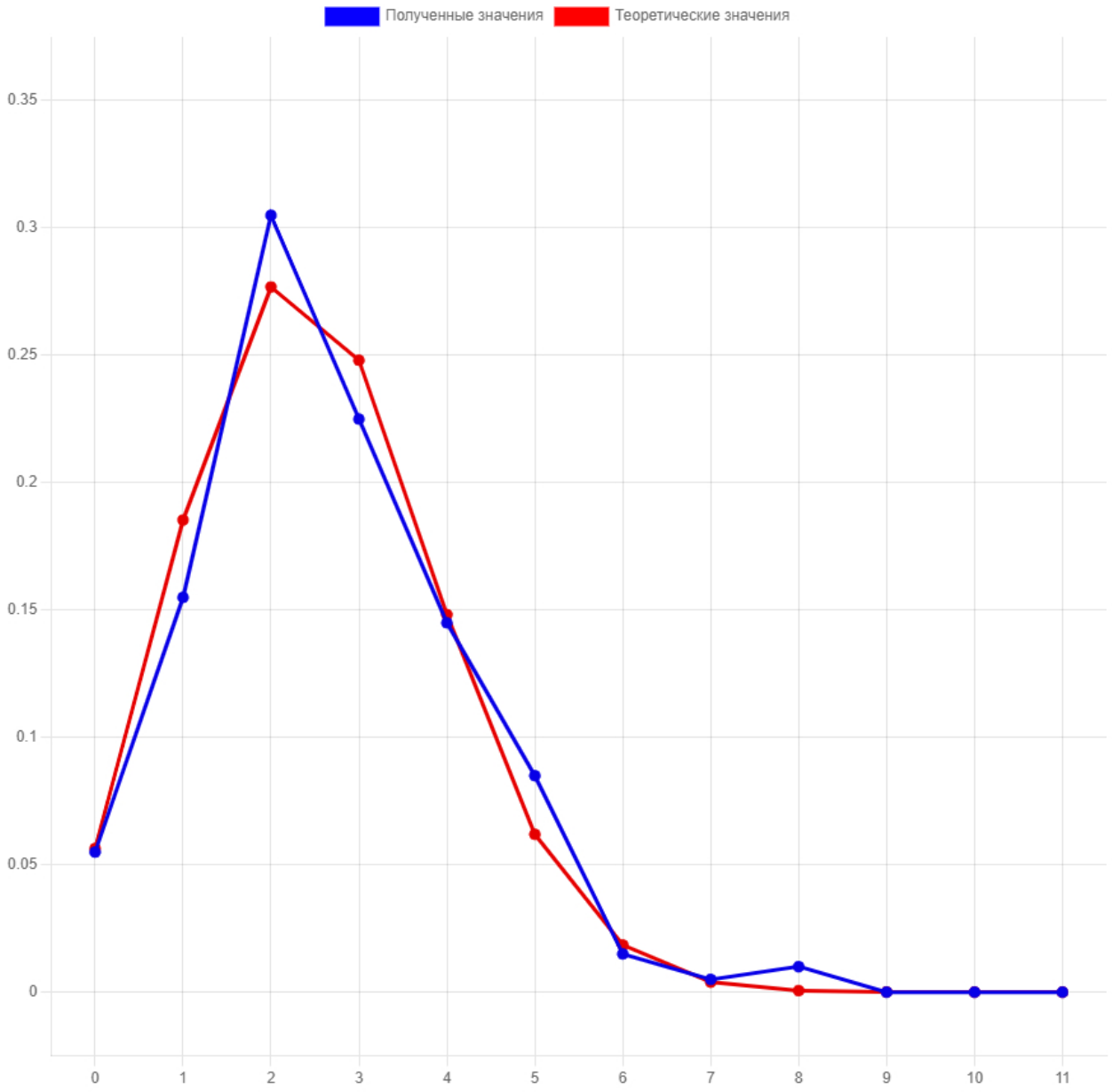
Отсортированная выборка

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	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	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	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	4	4	4	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	6	6	6	7	8	8

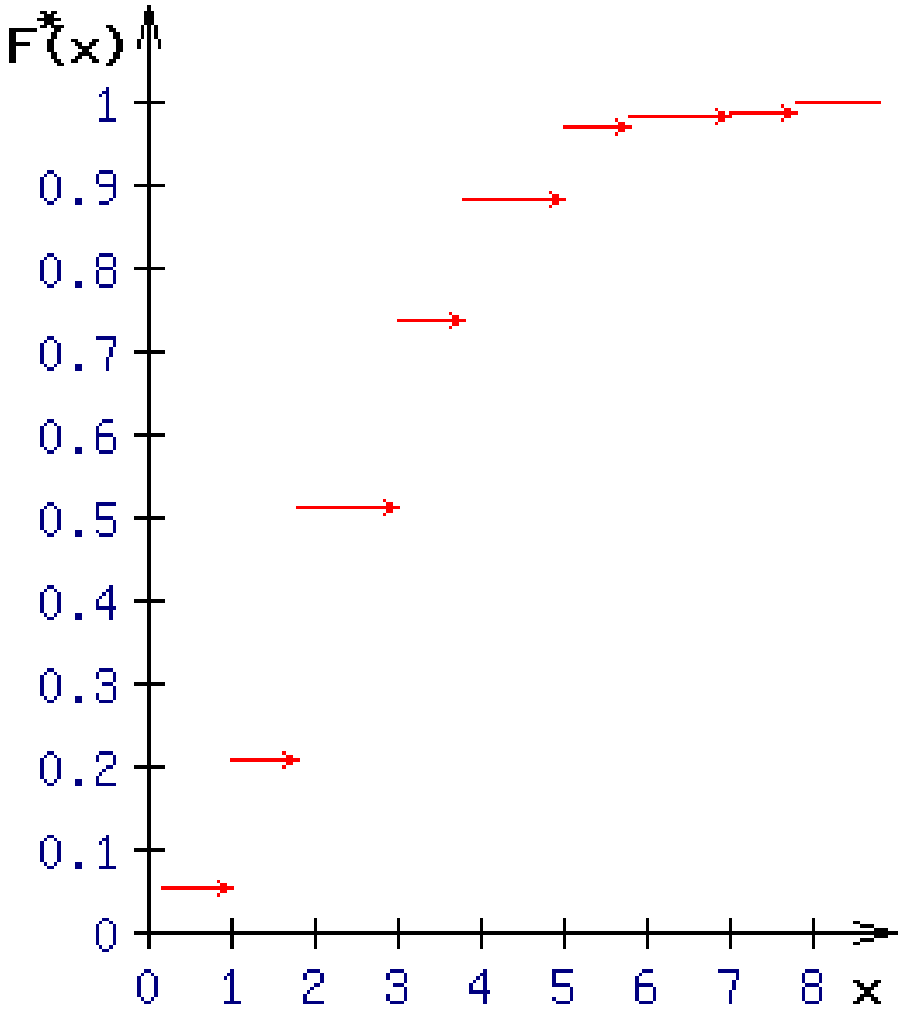
Статистический ряд

x_i	n_i	w_i	s_i
0	11	0.055	0.055
1	31	0.155	0.21
2	61	0.305	0.515
3	45	0.225	0.74
4	29	0.145	0.885
5	17	0.085	0.97
6	3	0.015	0.985
7	1	0.005	0.99
8	2	0.01	1
9	0	0	1
10	0	0	1
11	0	0	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 2.65
Выборочную дисперсию: 2.24749
Выборочное среднее квадратическое отклонение: 1.499167
Выборочная мода: 2
Выборочная медиана: 2
Выборочный коэффициент асимметрии: 0.644852
Выборочный коэффициент эксцесса: 0.73470

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

j	w_j	p_j	$\lvert w_j - p_j \rvert$
0	0.055	0.05642	0.00142
1	0.155	0.18537	0.03037
2	0.305	0.27684	0.02816
3	0.225	0.24808	0.02308
4	0.145	0.14820	0.0032
5	0.085	0.06198	0.02302
6	0.015	0.01851	0.00351
7	0.005	0.00395	0.00105
8	0.01	0.00059	0.00941
9	0	0.00006	0.00006
10	0	0	0
11	0	0	0
	1	1	0.03037

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	2.65	2.53	0.12	0.047431
Выборочная дисперсия	2.2475	1.9481	0.2994	0.15369
Выборочное среднее квадратичное отклонение	1.49917	1.39574	0.10342	0.0741
Выборочная мода	2	2	0	0
Выборочная медиана	2	3	1	0.33333
Выборочный коэффициент асимметрии	0.64485	0.60521	0.03963	0.065479
Выборочный коэффициент эксцесса	0.73470	-0.321338	1.05604	-3.28638

Задание 2:
p=0.23

Полученная выборка

0	1	0	0	2	0	9	6	0	18
0	0	0	2	5	11	7	5	2	4
5	7	2	1	0	0	0	0	0	2
5	5	1	1	5	2	19	1	1	1
14	0	1	1	3	0	2	1	0	10
7	0	15	0	1	7	3	2	1	0
0	3	4	4	3	6	2	1	2	2
0	3	5	1	9	1	2	1	2	1
1	0	6	6	2	0	4	1	0	6
5	3	4	1	20	0	2	8	3	2
3	1	0	0	10	0	3	0	5	2
7	1	3	0	6	6	7	0	2	0
7	4	2	6	0	2	8	3	0	0
7	2	10	1	0	11	4	0	1	2
6	0	3	2	3	6	0	3	0	0
1	0	0	3	9	4	8	1	0	4
4	13	1	6	15	1	10	0	1	0
0	0	2	4	1	11	1	10	0	13
1	4	6	1	1	0	6	0	3	1
3	6	8	1	4	0	1	1	5	1

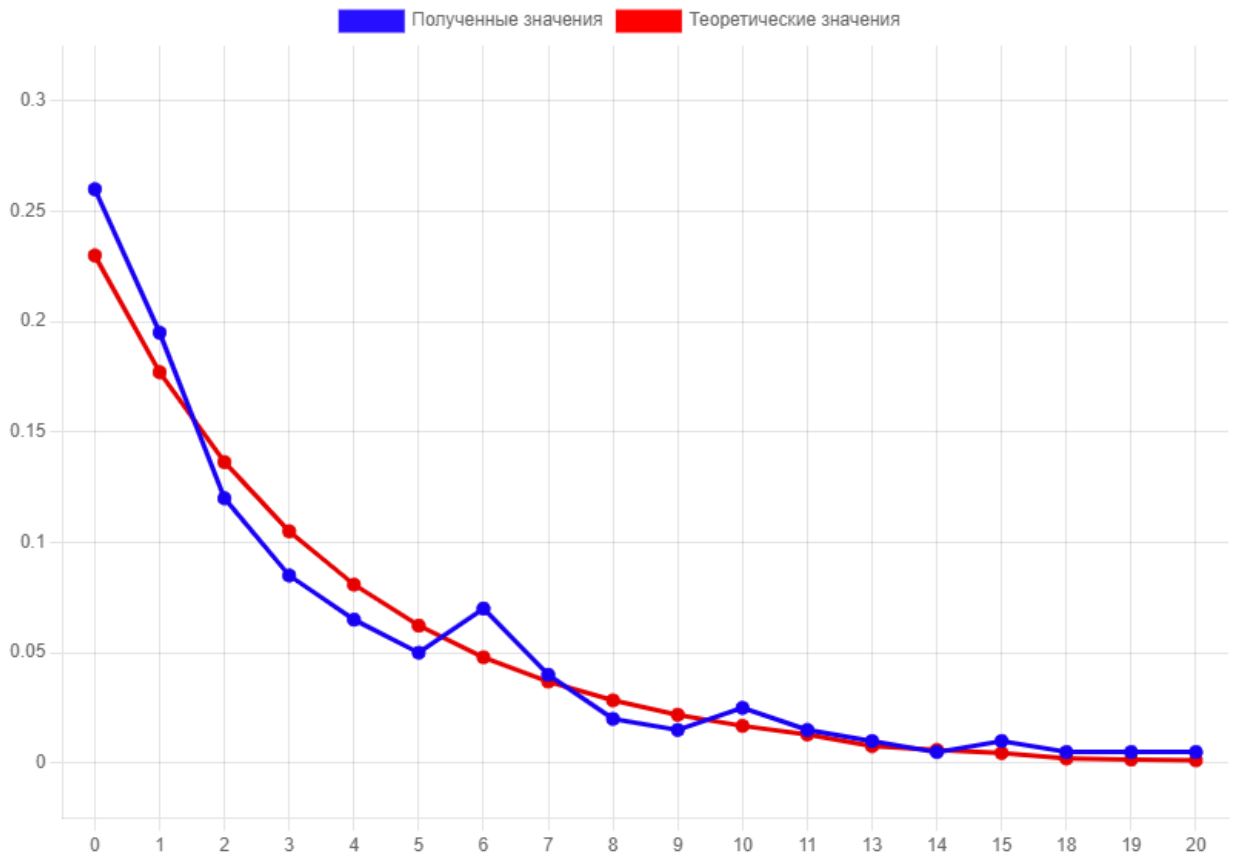
Отсортированная выборка

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	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	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	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	8	8	8
8	9	9	9	10	10	10	10	10	11
11	11	13	13	14	15	15	18	19	20

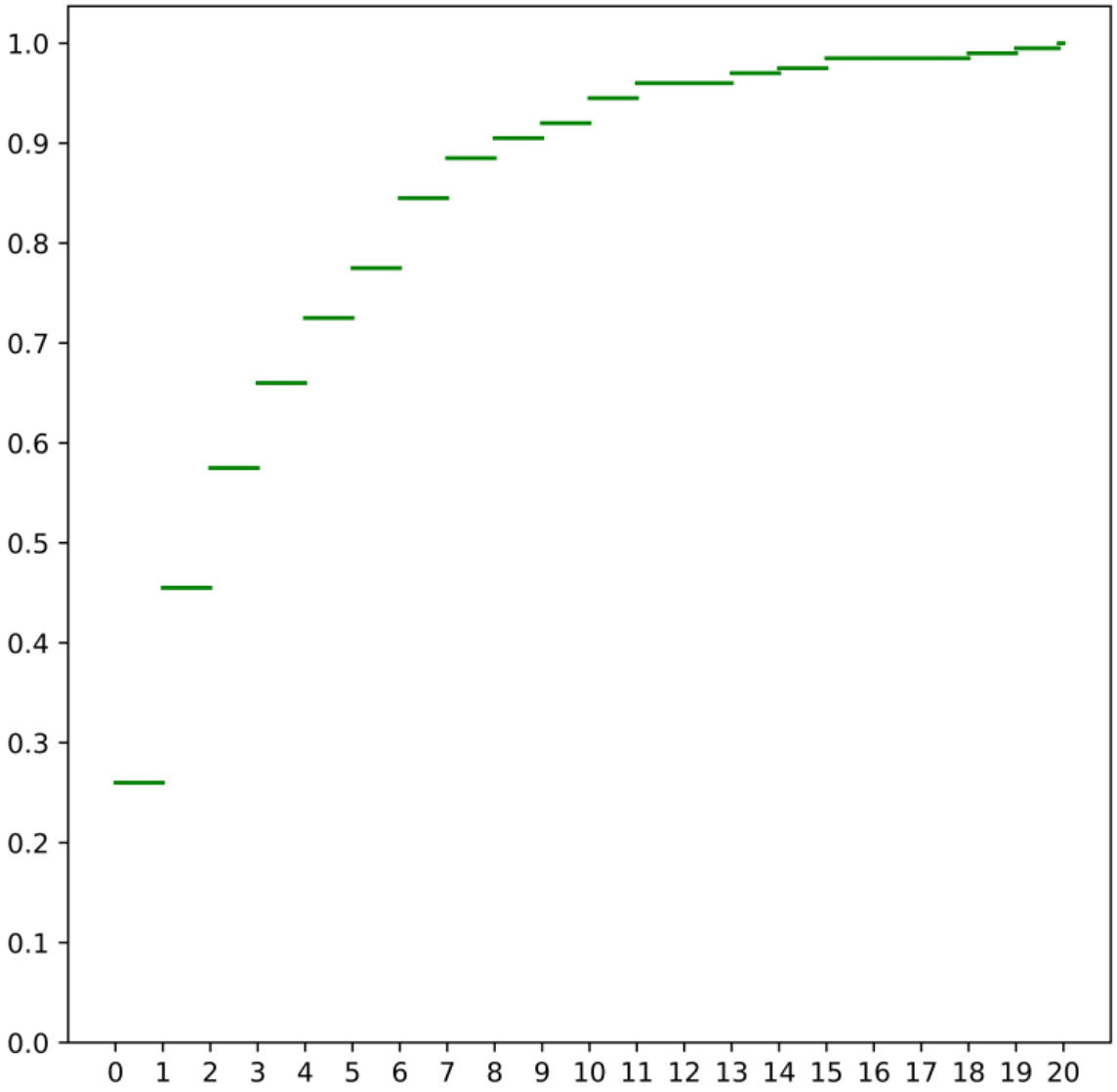
Статистический ряд

x_i	n_i	w_i	s_i
0	52	0.26	0.26
1	39	0.195	0.455
2	24	0.12	0.575
3	17	0.085	0.66
4	13	0.065	0.725
5	10	0.05	0.775
6	14	0.07	0.845
7	8	0.04	0.885
8	4	0.02	0.905
9	3	0.015	0.92
10	5	0.025	0.945
11	3	0.015	0.96
13	2	0.01	0.97
14	1	0.005	0.975
15	2	0.01	0.985
18	1	0.005	0.99
19	1	0.005	0.995
20	1	0.005	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 3.245
Выборочную дисперсию: 14.83498
Выборочное среднее квадратическое отклонение: 3.85162
Выборочная мода: 0
Выборочная медиана: 2
Выборочный коэффициент асимметрии: 1.80520
Выборочный коэффициент эксцесса: 3.66598

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

j	w_j	p_j	$\lvert w_j - p_j \rvert$
0	0.26	0.23	0.03
1	0.195	0.1771	0.0179
2	0.12	0.136367	0.016367
3	0.085	0.105003	0.02003
4	0.065	0.080852	0.015852
5	0.05	0.062256	0.012256
6	0.07	0.047937	0.022068
7	0.04	0.036912	0.003088
8	0.02	0.028422	0.08422
9	0.015	0.021885	0.006885
10	0.025	0.016851	0.008149
11	0.015	0.012976	0.002024
12	0.01	0.007693	0.092307
13	0.005	0.005924	0.000924
14	0.01	0.004561	0.005439
15	0.005	0.002082	0.002918
16	0.005	0.001603	0.003397
17	0.005	0.001235	0.003765
	1	1	0.092307

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	3.245	3.34782	0.10282	0.03071
Выборочная дисперсия	14.83498	14.55576	0.27921	0.01918
Выборочное среднее Квадратичное отклонение	3.85161	3.81520	0.03641	0.00954
Выборочная мода	0	0	0	0
Выборочная медиана	2	2	0	0
Выборочный коэффициент асимметрии	1.80519	2.15202	0.21190	0.10505
Выборочный коэффициент эксцесса	3.66598	6.06870	2.40272	0.39592

Задание 3:
λ =0.56

Полученная выборка

1	0	0	0	0	0	2	0	0	2
0	0	0	1	0	1	0	1	1	0
0	0	3	0	2	0	0	1	1	0
0	1	0	0	0	0	1	0	0	1
0	1	2	0	0	0	2	0	0	4
0	0	0	0	0	1	0	2	0	1
1	0	0	0	0	0	0	1	0	0
0	0	0	1	1	3	2	2	0	0
1	1	1	0	2	0	1	1	1	0
0	1	0	0	0	0	0	2	0	0
1	1	0	0	0	1	0	1	1	1
0	0	1	0	0	2	0	2	2	1
1	1	0	1	0	2	0	0	1	1
0	0	1	0	0	0	1	1	0	0
0	0	0	1	1	3	0	0	2	0
0	1	3	2	0	1	0	0	0	1
0	2	0	1	0	0	1	1	0	2
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	1	0

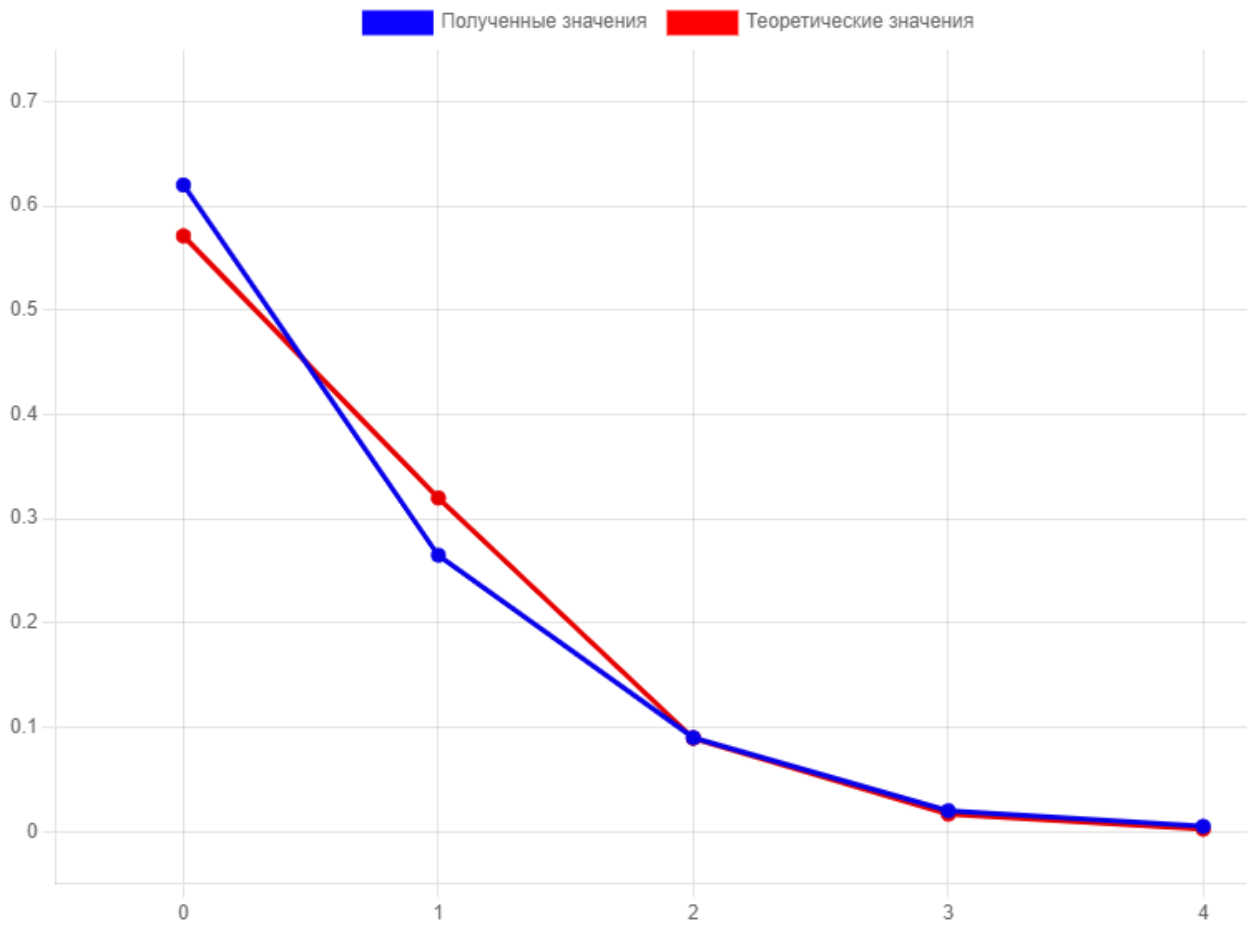
Отсортированная выборка

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
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	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	3	3	3	3	4

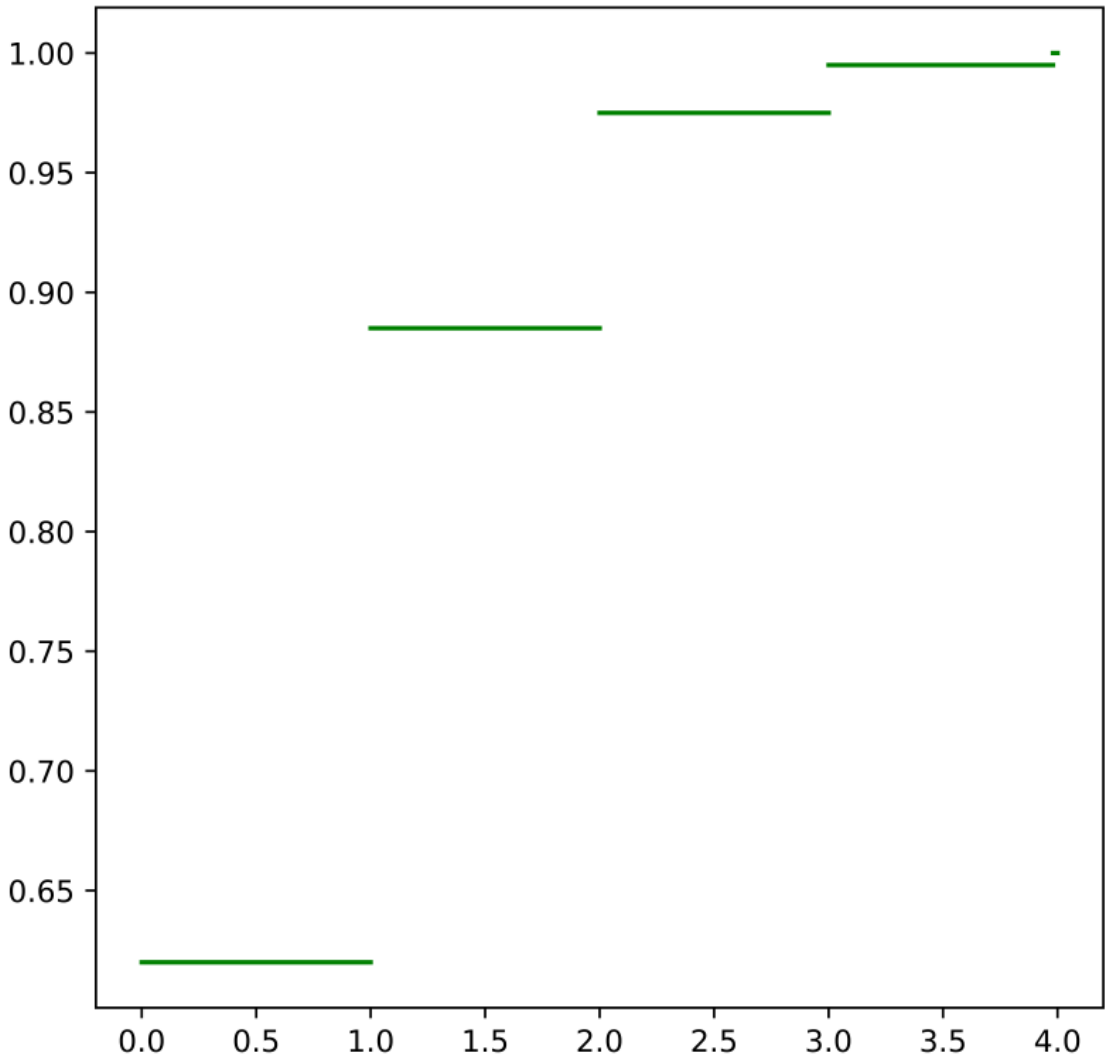
Статистический ряд

x_i	n_i	w_i	s_i
0	124	0.62	0.62
1	53	0.265	0.885
2	18	0.09	0.975
3	4	0.02	0.995
4	1	0.005	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 0.525
Выборочную дисперсию: 0.60937
Выборочное среднее квадратическое отклонение: 0.78062
Выборочная мода: 0
Выборочная медиана: 0
Выборочный коэффициент асимметрии: 1.55674
Выборочный коэффициент эксцесса: 2.2948

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

j	w_j	p_j	$\lvert w_j - p_j \rvert$
0	0.62	0.57121	0.04879
1	0.265	0.31988	0.05488
2	0.09	0.08957	0.00043
3	0.02	0.01672	0.00328
4	0.005	0.00234	0.00266
	1	1	0.05488

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	0.525	0.56	0.035	0.0625
Выборочная дисперсия	0.60937	0.56	0.04937	0.08817
Выборочное среднее Квадратичное отклонение	0.78062	0.74833	0.03229	0.04315
Выборочная мода	0	0	0	0
Выборочная медиана	0.0	0	0.0	0
Выборочный коэффициент асимметрии	1.55674	1.3363	0.22044	0.16496
Выборочный коэффициент эксцесса	2.2948	1.78571	0.50909	0.28509

Задание 4:

n = 11; $p_k = \frac{1}{11}$

Полученная выборка

5	5	8	0	0	6	8	0	1	3
2	6	10	5	7	0	2	5	0	7
0	0	0	2	6	3	10	8	5	6
7	2	3	4	4	5	5	10	2	6
3	2	8	5	8	10	0	9	5	6
1	0	4	10	5	3	8	7	10	0
5	3	4	2	7	7	3	5	9	9
2	6	1	1	1	6	6	4	1	9
1	9	9	10	6	10	3	7	6	9
8	3	9	3	9	2	7	7	4	9
4	7	8	6	4	5	2	0	3	6
4	1	2	6	9	8	1	3	0	5
10	4	9	9	1	0	0	0	1	9
0	6	8	1	8	2	8	8	0	6
6	8	7	8	5	10	0	9	9	7
1	1	1	10	0	6	0	9	0	7
7	9	5	5	0	0	6	5	2	0
4	6	5	6	9	2	2	4	8	10
2	9	0	4	7	5	0	6	9	5
2	9	4	4	7	1	8	3	5	7

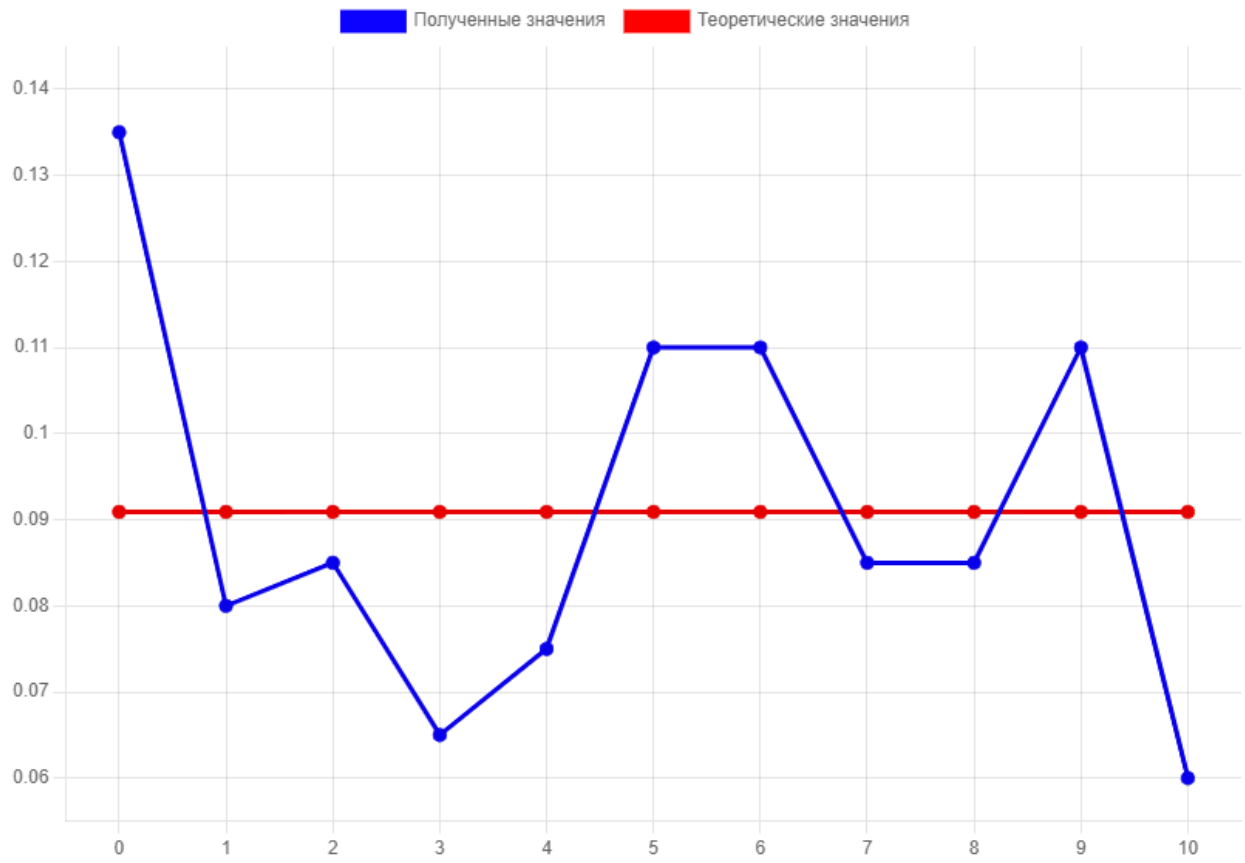
Отсортированная выборка

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	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
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	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
6	6	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	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

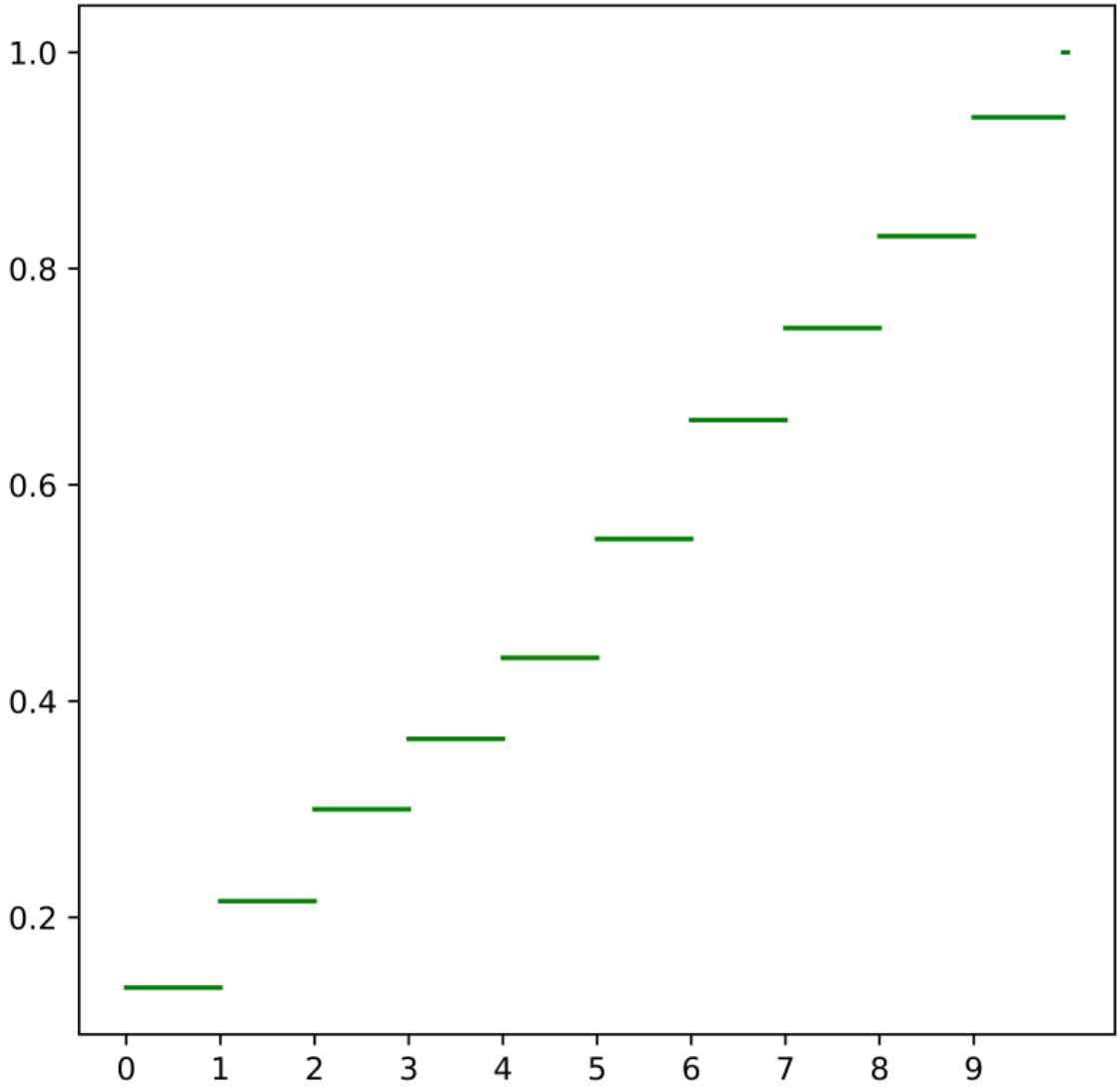
Статистический ряд

x_i	n_i	w_i	s_i
0	27	0.135	0.135
1	16	0.08	0.215
2	17	0.085	0.30
3	13	0.065	0.365
4	15	0.075	0.44
5	22	0.11	0.55
6	22	0.11	0.66
7	17	0.085	0.745
8	17	0.085	0.83
9	22	0.11	0.94
10	12	0.06	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 4.82
Выборочную дисперсию: 10.1976
Выборочное среднее квадратическое отклонение: 3.19337
Выборочная мода: 0
Выборочная медиана: 5
Выборочный коэффициент асимметрии: -0.05366
Выборочный коэффициент эксцесса: -1.23419

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

j	w_j	p_j	$\lvert w_j - p_j \rvert$
0	0.135	0.09091	0.044091
1	0.08	0.09091	0.01091
2	0.085	0.09091	0.00591
3	0.065	0.09091	0.02591
4	0.075	0.09091	0.01591
5	0.11	0.09091	0.01909
6	0.11	0.09091	0.01909
7	0.085	0.09091	0.00591
8	0.085	0.09091	0.00591
9	0.11	0.09091	0.01909
10	0.06	0.09091	0.03091
	1	1	0.044091

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	4.82	5.0	0.18	0.036
Выборочная дисперсия	10.1976	10.0	0.1976	0.01976
Выборочное среднее Квадратичное отклонение	3.19337	3.16228	0.03109	0.00983
Выборочная мода	0	5.0	5.0	1.0
Выборочная медиана	5	5.0	0.0	0.0
Выборочный коэффициент асимметрии	-0.05366	0	0.05366	∞
Выборочный коэффициент эксцесса	-1.23419	-0.00992	1.22427	-123.44756

Задание 5:
m = 11; M = 25 ; K = 12

Полученная выборка

7	6	5	4	4	5	3	6	8	5
7	6	5	7	8	5	5	6	3	4
6	5	6	5	5	6	5	6	5	6
5	6	7	3	6	6	5	6	4	6
4	6	7	4	4	6	4	4	6	5
3	5	5	5	6	5	6	6	4	5
4	7	5	4	4	8	5	5	5	6
7	5	5	5	0	3	5	6	5	2
3	4	7	5	4	6	4	4	8	6
5	6	7	4	3	5	6	5	6	5
5	4	5	6	6	6	6	5	5	5
7	5	7	4	5	4	6	4	7	5
5	6	5	7	6	7	6	8	7	5
6	6	6	6	6	6	4	4	7	8
5	4	5	5	6	7	3	7	5	5
7	7	8	6	5	4	4	5	5	6
5	5	5	6	4	5	5	6	4	7
6	5	6	3	6	5	7	5	7	9
6	5	6	7	8	5	5	6	4	4
5	5	4	5	6	7	7	6	6	5

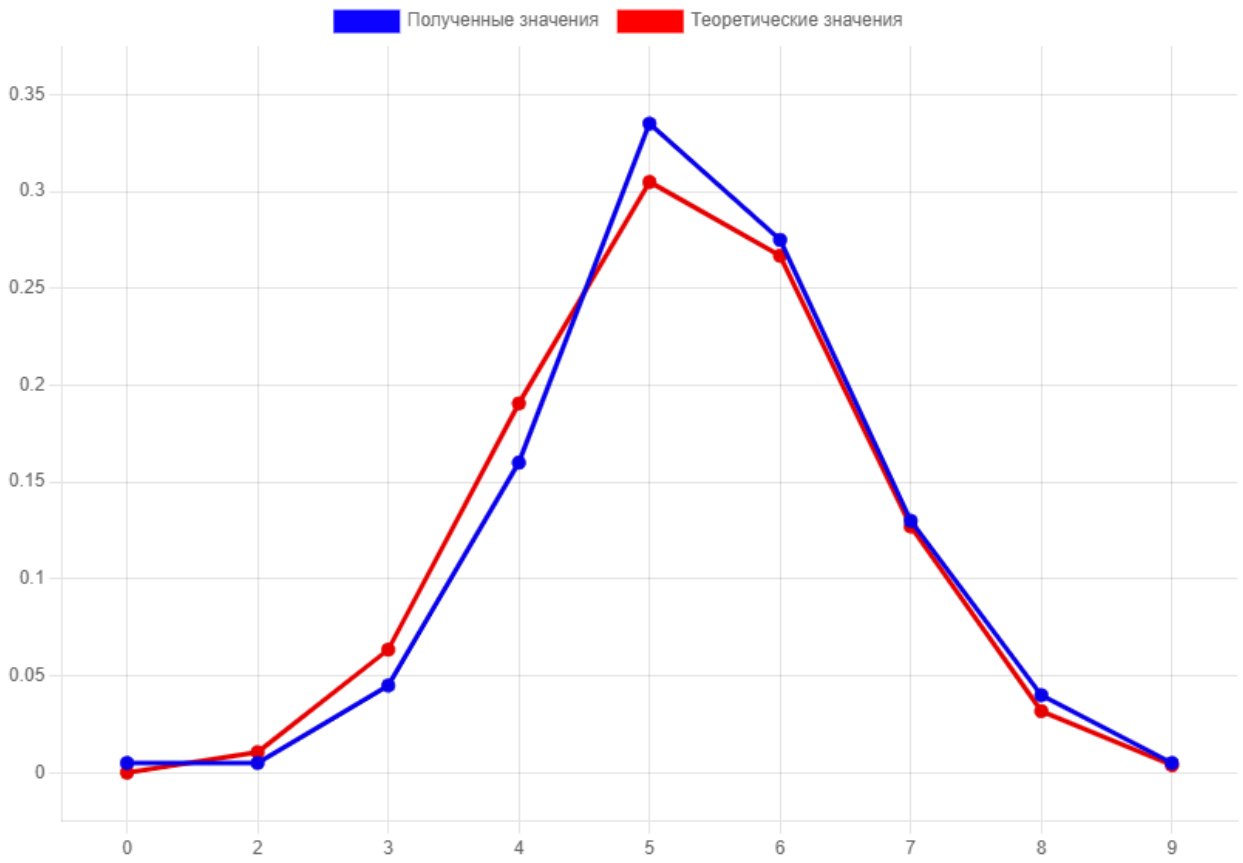
Отсортированная выборка

0	2	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	4	4	4
4	4	4	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
6	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	9

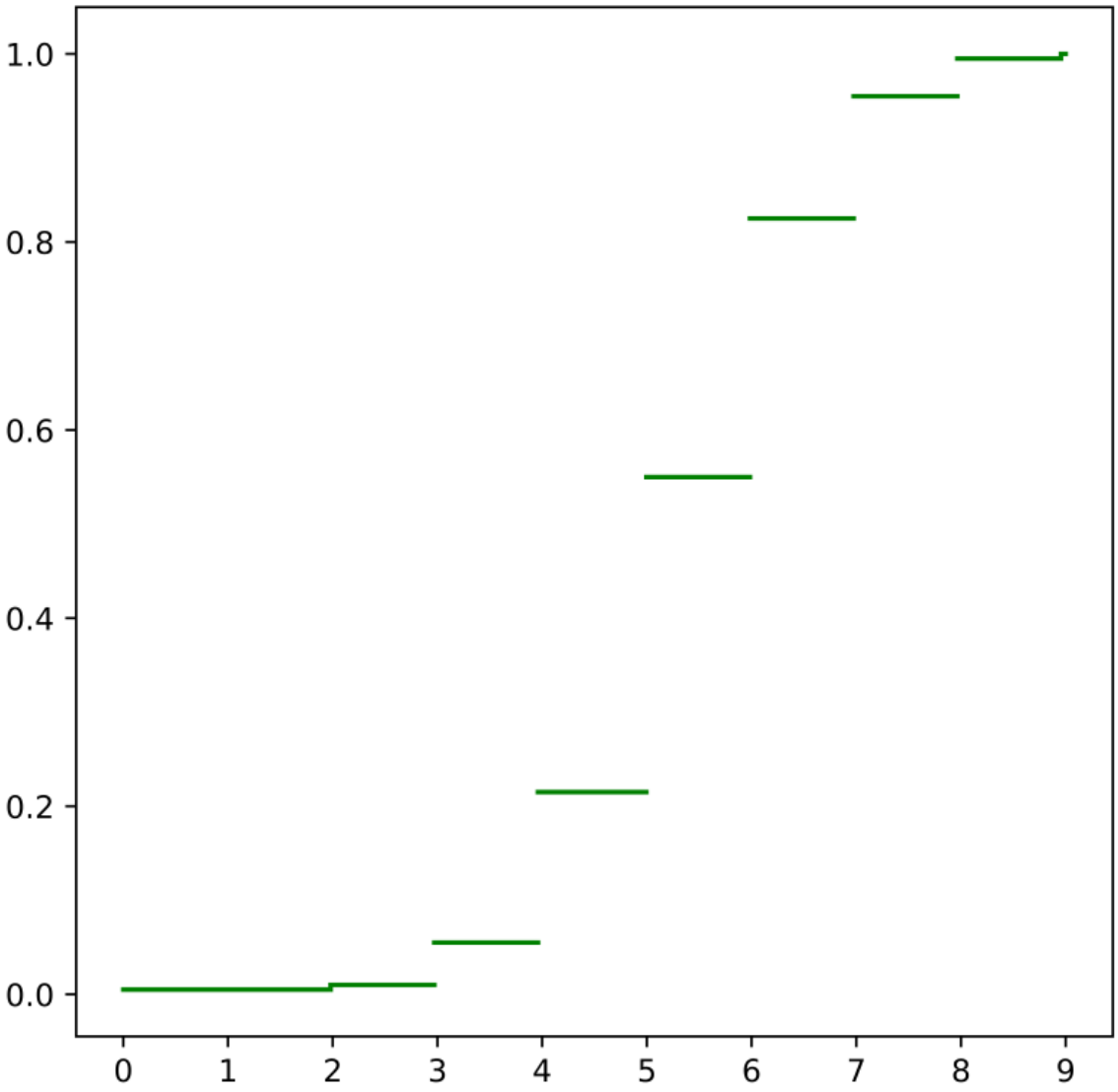
Статистический ряд

x_i	n_i	w_i	s_i
0	1	0.005	0.005
2	1	0.005	0.01
3	9	0.045	0.055
4	32	0.16	0.215
5	67	0.335	0.55
6	55	0.275	0.825
7	26	0.13	0.955
8	8	0.04	0.995
9	1	0.005	1
	200	1	-

График полигона относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 5.385
Выборочную дисперсию: 1.59677
Выборочное среднее квадратическое отклонение: 1.26364
Выборочная мода: 5
Выборочная медиана: 5
Выборочный коэффициент асимметрии: 0.64485
Выборочный коэффициент эксцесса: 0.7347

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

j	w_j	p_j	$\lvert w_j - p_j \rvert$
0	0.005	0.000017	0.004983
1	0.005	0.010587	0.005587
2	0.045	0.063521	0.018251
3	0.16	0.190564	0.030564
4	0.335	0.304902	0.030098
5	0.275	0.26679	0.00821
6	0.13	0.127043	0.002957
7	0.04	0.031761	0.008239
8	0.005	0.00385	0.00115
	1	1	0.030564

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	5.385	5.28	0.105	0.01987
Выборочная дисперсия	1.59677	1.6016	0.00483	0.00301
Выборочное среднее Квадратичное отклонение	1.26364	1.26554	0.00191	0.00151
Выборочная мода	5	4	1	0.25
Выборочная медиана	5.0	5	0	0
Выборочный коэффициент асимметрии	-0.23112	0.00412	0.23524	57.06119
Выборочный коэффициент эксцесса	1.14211	0	1.14211	∞

Задание 6:
$\alpha = 0.3$; $\sigma = 1.03$

Полученная выборка

1.1658	0.03992	1.01534	0.07553	-0.92043	0.70086	0.35958	0.39276	-1.48112	-1.2337
-1.30807	-2.14042	0.45475	-1.31484	-0.4506	-0.69883	1.72924	-0.71028	-0.39715	0.81071
-0.6385	0.40144	0.92914	0.6065	1.07279	0.74965	1.60671	2.28143	-0.07828	-0.1886
0.64163	-0.45412	0.89457	4.23547	1.1833	-0.13018	-0.51238	-0.61987	0.62851	0.23584
3.21512	-1.08879	1.81319	-0.34664	-0.05514	-1.39536	0.85688	0.71801	-0.31807	2.36985
-0.43662	0.18998	-0.17463	0.46352	0.60267	1.0832	-0.55948	0.15433	-1.33625	-1.15913
1.23691	-1.02543	-0.47616	0.99325	1.10945	-1.32368	0.31241	0.45591	0.65407	-0.53396
1.04851	-2.02812	-2.1909	1.68378	0.33701	0.15507	-0.64901	0.23099	-1.00966	0.80459
-1.05305	-0.13572	0.29378	0.01389	0.27026	0.82507	1.00571	2.0791	0.16196	1.2796
0.86837	0.11846	-0.36415	-0.20819	0.14569	0.67373	-1.22727	-0.8153	-0.03751	0.92959
0.01598	0.13447	-0.82175	0.95921	1.02299	0.89841	-1.11187	0.47599	-0.117	-0.73716
-0.80665	-0.83037	-0.90396	-1.93622	0.88252	0.87349	0.84677	0.19989	0.17481	-0.5227
0.59634	0.56526	0.75848	-0.31772	-1.12639	0.40775	0.6983	0.95638	1.16627	1.30625
1.00296	0.90736	0.72462	1.44972	0.86584	-1.35144	-1.11026	0.39271	1.04747	-1.65366
0.37406	-0.65462	-0.0471	0.62038	1.40446	0.97301	-0.62792	-1.89807	0.04357	0.34359
-0.03187	0.91351	0.8047	1.29796	-0.65203	0.44832	0.87969	-1.10788	1.38339	-0.86818
-0.05037	-1.73752	-0.0647	-0.43745	0.3148	-0.52213	1.62558	0.2308	2.37026	0.40379
0.64067	-0.49419	0.3358	0.85429	-0.61423	1.34232	0.94634	0.33478	-0.75317	0.82504
-0.91629	-1.78466	-0.6901	0.98902	1.65466	2.0217	-2.10385	-1.0686	-0.51435	-0.24632
1.07106	-0.2019	1.37149	1.14914	-0.04509	0.02509	0.18447	1.84519	0.30137	0.24753

Отсортированная выборка

-2.1909	-2.14042	-2.10385	-2.02812	-1.93622	-1.89807	-1.78466	-1.73752	-1.65366	-1.48112
-1.39536	-1.35144	-1.33625	-1.32368	-1.31484	-1.30807	-1.2337	-1.22727	-1.15913	-1.12639
-1.11187	-1.11026	-1.10788	-1.08879	-1.0686	-1.05305	-1.02543	-1.00966	-0.92043	-0.91629
-0.90396	-0.86818	-0.83037	-0.82175	-0.8153	-0.80665	-0.75317	-0.73716	-0.71028	-0.69883
-0.6901	-0.65462	-0.65203	-0.64901	-0.6385	-0.62792	-0.61987	-0.61423	-0.55948	-0.53396
-0.5227	-0.52213	-0.51435	-0.51238	-0.49419	-0.47616	-0.45412	-0.4506	-0.43745	-0.43662
-0.39715	-0.36415	-0.34664	-0.31807	-0.31772	-0.24632	-0.20819	-0.2019	-0.1886	-0.17463
-0.13572	-0.13018	-0.117	-0.07828	-0.0647	-0.05514	-0.05037	-0.0471	-0.04509	-0.03751
-0.03187	0.01389	0.01598	0.02509	0.03992	0.04357	0.07553	0.11846	0.13447	0.14569
0.15433	0.15507	0.16196	0.17481	0.18447	0.18998	0.19989	0.2308	0.23099	0.23584
0.24753	0.27026	0.29378	0.30137	0.31241	0.3148	0.33478	0.3358	0.33701	0.34359
0.35958	0.37406	0.39271	0.39276	0.40144	0.40379	0.40775	0.44832	0.45475	0.45591
0.46352	0.47599	0.56526	0.59634	0.60267	0.6065	0.62038	0.62851	0.64067	0.64163
0.65407	0.67373	0.6983	0.70086	0.71801	0.72462	0.74965	0.75848	0.80459	0.8047
0.81071	0.82504	0.82507	0.84677	0.85429	0.85688	0.86584	0.86837	0.87349	0.87969
0.88252	0.89457	0.89841	0.90736	0.91351	0.92914	0.92959	0.94634	0.95638	0.95921
0.97301	0.98902	0.99325	1.00296	1.00571	1.01534	1.02299	1.04747	1.04851	1.07106
1.07279	1.0832	1.10945	1.14914	1.1658	1.16627	1.1833	1.23691	1.2796	1.29796
1.30625	1.34232	1.37149	1.38339	1.40446	1.44972	1.60671	1.62558	1.65466	1.68378
1.72924	1.81319	1.84519	2.0217	2.0791	2.28143	2.36985	2.37026	3.21512	4.23547

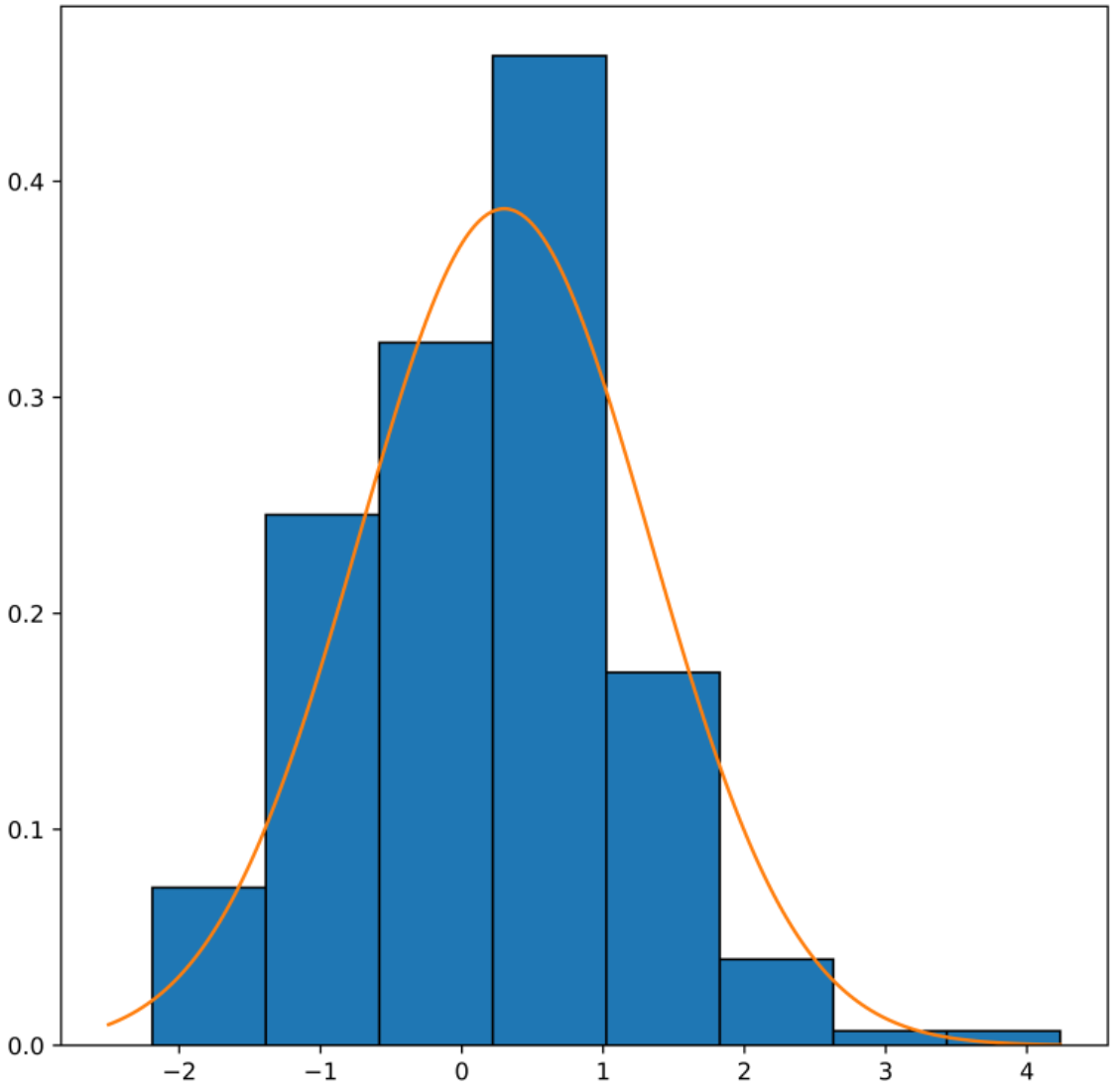
Интервальный ряд

Интервалы	n_i	w_i
[-2.1909, -1.3876]	11	0.055
(-1.3876, -0.58431]	37	0.185
(-0.58431, 0.21899]	49	0.245
(0.21899, 1.02228]	69	0.345
(1.02228, 1.82558]	26	0.13
(1.82558, 2.62887]	6	0.03
(2.62887, 3.43217]	1	0.005
(3.43217, 4.23547]	1	0.005
	200	1

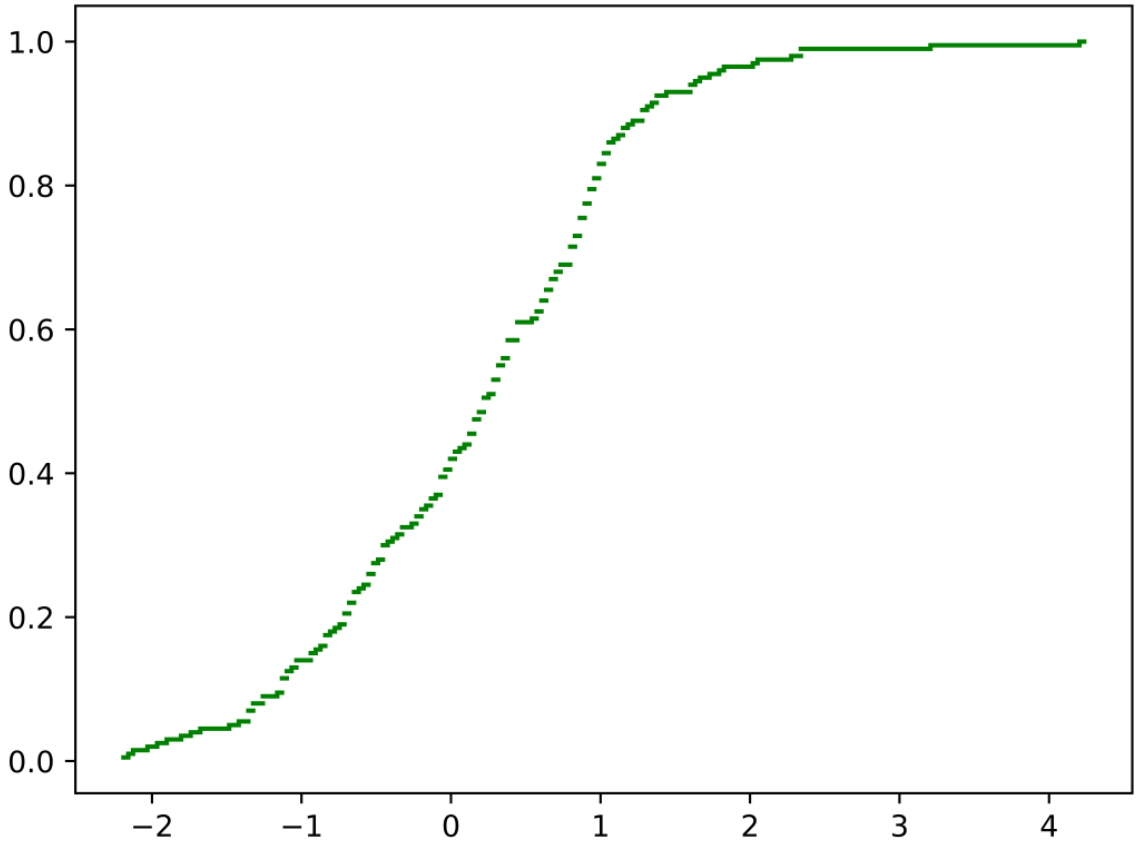
Ассоциированный статистический ряд

x_i	n_i	w_i
-1.78925	11	0.055
-0.98596	37	0.185
-0.18266	49	0.245
0.62063	69	0.345
1.42393	26	0.13
2.22723	6	0.03
3.03052	1	0.005
3,83382	1	0.005
	200	1

Гистограмма относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 0.15564
Выборочная дисперсия с поправкой Шеппарда: 0.68175
Выборочное среднее квадратическое отклонение: 0.82568
Выборочная мода: 0.44213
Выборочная медиана: 0.24169
Выборочный коэффициент асимметрии: -0.00701
Выборочный коэффициент эксцесса: 2.05813

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

Интервалы	w_i	p_i	$\lvert w_i - p_i \rvert$
[-2.1909, -1.3876]	0.055	0.04287	0.01213
(-1.3876, -0.58431]	0.185	0.14463	0.04037
(-0.58431, 0.21899]	0.245	0.27336	0.02836
(0.21899, 1.02228]	0.345	0.28977	0.05523
(1.02228, 1.82558]	0.13	0.17229	0.04229
(1.82558, 2.62887]	0.03	0.05741	0.02741
(2.62887, 3.43217]	0.005	0.0107	0.0057
(3.43217, 4.23547]	0.005	0.00111	0.00389
	1	0.99214	0.05523

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	0.15564	0.3	0.14436	0.48121
Выборочная дисперсия	0.68175	1.0609	0.37915	0.35739
Выборочное среднее Квадратичное отклонение	0.82568	1.03	0.20432	0.19837
Выборочная мода	0.44213	0.3	0.14213	0.47376
Выборочная медиана	0.24169	0.3	0.05831	0.19438
Выборочный коэффициент асимметрии	-0.00701	0	0.00701	-
Выборочный коэффициент эксцесса	2.05813	0	2.05813	-

Задание 7:
$\lambda = 2.06$

Полученная выборка

0.37721	0.03169	0.37151	0.0769	0.34037	1.13088	0.2575	0.56454	1.82127	0.05064
0.10365	0.70715	0.21093	0.37443	0.19333	0.09039	0.04419	0.12236	0.77516	0.40823
1.4607	1.83618	1.37025	0.25452	0.68557	1.03029	1.14552	0.2294	0.28135	0.73949
0.07434	0.10757	1.42173	0.07836	0.04259	0.01033	0.21435	0.38462	0.9403	1.08397
0.95688	0.10289	0.0546	1.92442	1.40552	0.09963	0.59998	0.12916	0.11015	0.39198
0.52947	0.03863	0.1736	0.10894	1.18496	0.12376	1.5779	0.68347	0.66878	1.07169
1.08322	0.4471	1.59374	0.04513	0.16902	0.67763	0.25521	0.359	0.34856	0.21629
2.68677	1.53046	0.24074	0.15679	0.18262	0.12134	1.59947	1.40134	0.83365	1.5909
0.00551	0.16201	0.19019	0.46259	1.81955	0.03282	0.98149	0.56862	0.49926	0.65135
0.0878	0.06003	1.27631	0.02527	0.12644	0.53766	0.44759	0.01381	0.75243	0.096
0.43248	0.079	0.06444	0.08693	0.05088	0.62982	0.66711	0.00541	0.28463	0.1487
0.63705	0.19284	0.03226	0.38898	1.77783	0.3157	0.05891	0.54325	0.71747	1.06411
0.32912	0.34705	0.55192	0.01571	0.44481	0.43602	0.59003	0.24869	0.1213	0.16289
0.46451	0.59355	0.35801	1.29891	0.18403	0.15223	2.21691	1.21771	0.02043	0.00464
0.17036	1.13631	0.33452	0.09827	2.42315	0.94445	0.25844	0.36272	0.56341	0.19087
0.1771	0.26055	0.09611	0.48435	0.09605	0.57185	0.46519	1.0077	0.18623	0.2779
0.08348	0.01262	0.05042	0.47721	0.36026	0.74537	0.05519	0.49033	1.72484	0.15356
0.30864	0.19948	1.60141	0.00222	0.0428	1.12284	0.75679	0.30892	0.00987	0.91402
0.01588	0.1943	0.14746	0.35594	1.59502	1.02987	0.4608	0.60957	0.19554	0.74813
0.01426	0.11249	1.66962	0.44219	0.71825	1.16398	0.04346	0.45743	0.60865	0.14083

Отсортированная выборка

0.00222	0.00464	0.00541	0.00551	0.00987	0.01033	0.01262	0.01381	0.01426	0.01571
0.01588	0.02043	0.02527	0.03169	0.03226	0.03282	0.03863	0.04259	0.0428	0.04346
0.04419	0.04513	0.05042	0.05064	0.05088	0.0546	0.05519	0.05891	0.06003	0.06444
0.07434	0.0769	0.07836	0.079	0.08348	0.08693	0.0878	0.09039	0.096	0.09605
0.09611	0.09827	0.09963	0.10289	0.10365	0.10757	0.10894	0.11015	0.11249	0.1213
0.12134	0.12236	0.12376	0.12644	0.12916	0.14083	0.14746	0.1487	0.15223	0.15356
0.15679	0.16201	0.16289	0.16902	0.17036	0.1736	0.1771	0.18262	0.18403	0.18623
0.19019	0.19087	0.19284	0.19333	0.1943	0.19554	0.19948	0.21093	0.21435	0.21629
0.2294	0.24074	0.24869	0.25452	0.25521	0.2575	0.25844	0.26055	0.2779	0.28135
0.28463	0.30864	0.30892	0.3157	0.32912	0.33452	0.34037	0.34705	0.34856	0.35594
0.35801	0.359	0.36026	0.36272	0.37151	0.37443	0.37721	0.38462	0.38898	0.39198
0.40823	0.43248	0.43602	0.44219	0.44481	0.4471	0.44759	0.45743	0.4608	0.46259
0.46451	0.46519	0.47721	0.48435	0.49033	0.49926	0.52947	0.53766	0.54325	0.55192
0.56341	0.56454	0.56862	0.57185	0.59003	0.59355	0.59998	0.60865	0.60957	0.62982
0.63705	0.65135	0.66711	0.66878	0.67763	0.68347	0.68557	0.70715	0.71747	0.71825
0.73949	0.74537	0.74813	0.75243	0.75679	0.77516	0.83365	0.91402	0.9403	0.94445
0.95688	0.98149	1.0077	1.02987	1.03029	1.06411	1.07169	1.08322	1.08397	1.12284
1.13088	1.13631	1.14552	1.16398	1.18496	1.21771	1.27631	1.29891	1.37025	1.40134
1.40552	1.42173	1.4607	1.53046	1.5779	1.5909	1.59374	1.59502	1.59947	1.60141
1.66962	1.72484	1.77783	1.81955	1.82127	1.83618	1.92442	2.21691	2.42315	2.68677

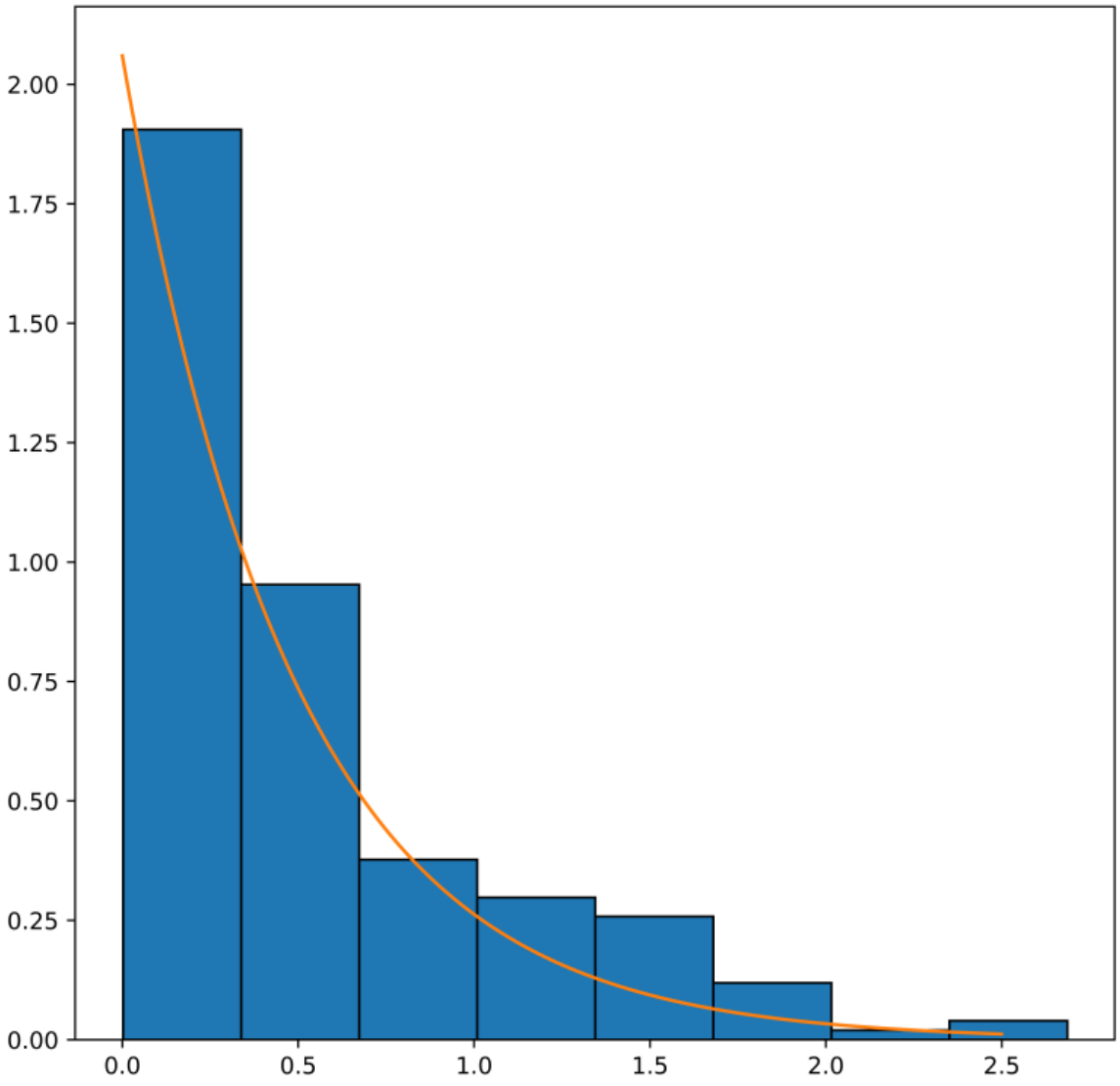
Интервальный ряд

Интервалы	n_i	w_i
[0, 0.33585]	96	0.48
(0.33585, 0.67169]	48	0.24
(0.67169, 1.00754]	18	0.09
(1.00754, 1.34339]	16	0.08
(1.34339, 1.67923]	13	0.065
(1.67923, 2.01508]	6	0.03
(2.01508, 2.35092]	1	0.005
(2.35092, 2.68677]	2	0.01
	200	1

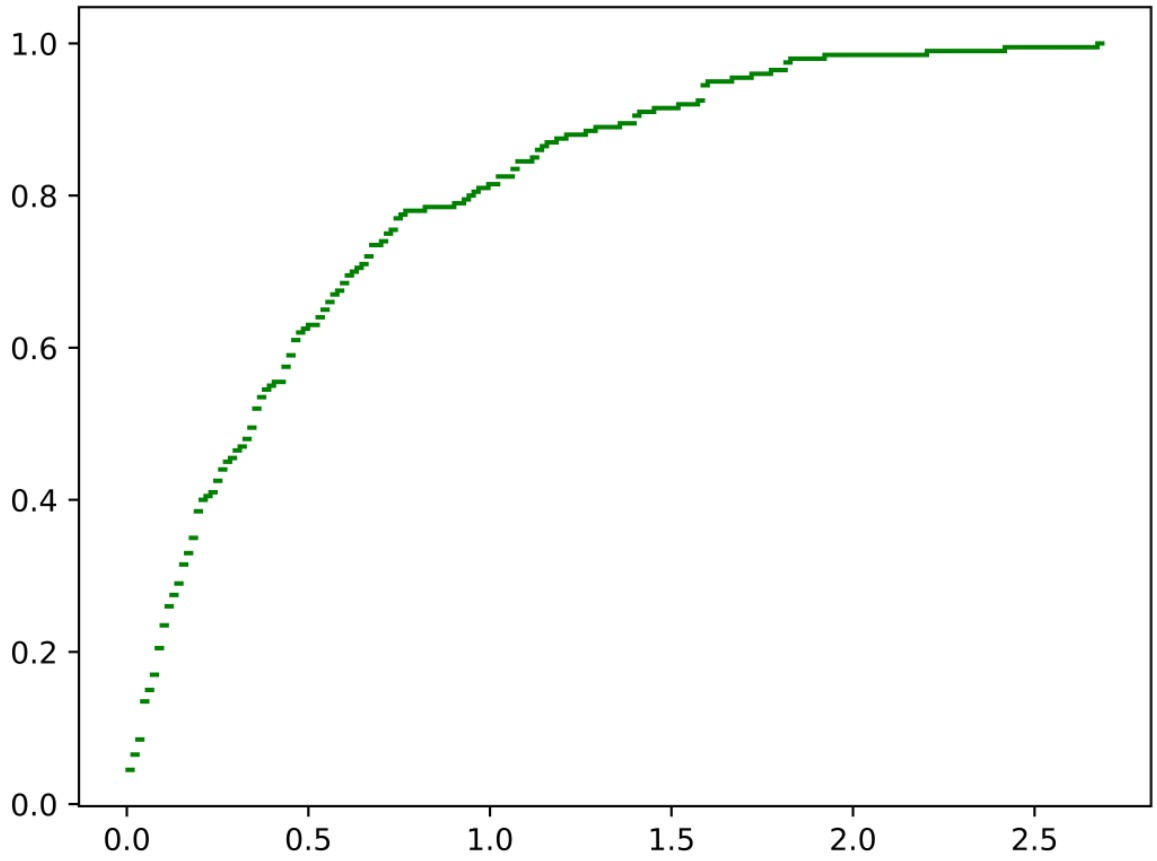
Ассоциированный статистический ряд

x_i	n_i	w_i
0.16792	96	0.48
0.50377	48	0.24
0.83962	18	0.09
1.17546	16	0.08
1.51131	13	0.065
1.84715	6	0.03
2.183	1	0.005
2,51885	2	0.01
	200	1

Гистограмма относительных частот


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5

Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 0.53567
Выборочная дисперсия с поправкой Шеппарда: 0.16115
Выборочное среднее квадратическое отклонение: 0.40143
Выборочная мода: 0.19453
Выборочная медиана: 0.35697
Выборочный коэффициент асимметрии: 2.29475
Выборочный коэффициент эксцесса: 5.0189

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

Интервалы	w_i	p_i	$\lvert w_i - p_i \rvert$
[0, 0.33585]	0.48	0.49935	0.01935
(0.33585, 0.67169]	0.24	0.25	0.01
(0.67169, 1.00754]	0.09	0.12516	0.03516
(1.00754, 1.34339]	0.08	0.06266	0.01734
(1.34339, 1.67923]	0.065	0.03137	0.03363
(1.67923, 2.01508]	0.03	0.01571	0.01429
(2.01508, 2.35092]	0.005	0.00786	0.00286
(2.35092, 2.68677]	0.01	0.00394	0.00606
	1	0.99605	0.03516

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	0.53567	0.48544	0.05024	0.10349
Выборочная дисперсия	0.16115	0.23565	0.0745	0.31616
Выборочное среднее Квадратичное отклонение	0.40143	0.48544	0.08401	0.17305
Выборочная мода	0.19453	0	0.19453	-
Выборочная медиана	0.35697	0.33648	0.0205	0.06091
Выборочный коэффициент асимметрии	2.29475	2	0.29475	0.14738
Выборочный коэффициент эксцесса	5.0189	6	0.9811	0.16352

Задание 8:
a = 0.12 ; b = 6.12

Полученная выборка

4.51619	0.64791	5.74012	2.40527	4.79316	1.02785	1.68067	0.79872	0.63617	2.65827
3.46123	5.4018	2.56591	2.29086	4.60367	3.10783	3.49316	3.98583	1.99792	5.6337
5.97344	2.93144	3.61848	5.65853	4.59439	2.69284	3.7595	3.92595	5.10626	0.46189
4.27802	3.77599	1.93278	1.11825	3.73949	1.37507	2.81062	3.74309	1.76431	4.71168
4.02514	4.29168	2.25315	1.42468	4.9224	3.52073	3.81906	4.09599	5.80177	6.10689
1.71819	1.89865	3.05749	4.4658	5.74957	4.31756	3.74666	6.01089	1.5953	2.5564
4.09231	1.39821	0.74256	4.98158	1.98988	1.78326	2.00176	2.86743	3.53877	0.62853
3.69936	1.70682	2.40115	5.65735	3.36533	3.60259	3.86498	3.89272	3.6226	2.65251
3.69958	3.19559	0.1808	3.98894	4.91258	3.63122	3.27958	1.52191	0.17533	0.76452
4.56522	3.47247	1.70136	5.93382	3.02981	4.39671	6.10375	2.26584	4.79656	2.38918
5.84915	5.93756	0.42735	1.35956	0.13694	1.15312	3.17122	2.67776	5.32927	0.87669
2.2269	3.18344	1.83923	0.97814	5.29894	1.30082	5.98346	1.52759	1.75303	0.95461
2.66872	3.76358	1.24846	0.34444	2.15164	1.34363	0.70764	2.59511	0.50725	0.39954
1.31598	0.81105	1.64008	2.04608	0.12563	2.63582	5.1322	4.68766	4.42812	4.4732
1.28256	3.06777	1.36024	4.04054	0.12723	3.34874	0.85071	5.87172	5.83924	0.24436
1.20543	3.41301	3.42896	0.93058	0.79317	4.15512	4.66328	2.80795	0.99277	3.54192
1.63507	5.51285	3.84307	4.43255	0.69026	0.48988	0.99017	5.54017	5.43263	1.05205
3.56077	2.67718	3.75287	5.94408	4.00894	3.55984	3.42971	5.18887	4.14362	0.19922
2.77651	1.39758	3.92923	4.49154	0.99509	1.40039	2.41583	0.19218	1.75065	4.74384
6.04676	4.65954	5.81203	0.86442	5.57319	0.4329	3.67384	4.09807	2.42238	3.92409

Отсортированная выборка

0.12563	0.12723	0.13694	0.17533	0.1808	0.19218	0.19922	0.24436	0.34444	0.39954
0.42735	0.4329	0.46189	0.48988	0.50725	0.62853	0.63617	0.64791	0.69026	0.70764

0.74256	0.76452	0.79317	0.79872	0.81105	0.85071	0.86442	0.87669	0.93058	0.95461
0.97814	0.99017	0.99277	0.99509	1.02785	1.05205	1.11825	1.15312	1.20543	1.24846
1.28256	1.30082	1.31598	1.34363	1.35956	1.36024	1.37507	1.39758	1.39821	1.40039
1.42468	1.52191	1.52759	1.5953	1.63507	1.64008	1.68067	1.70136	1.70682	1.71819
1.75065	1.75303	1.76431	1.78326	1.83923	1.89865	1.93278	1.98988	1.99792	2.00176
2.04608	2.15164	2.2269	2.25315	2.26584	2.29086	2.38918	2.40115	2.40527	2.41583
2.42238	2.5564	2.56591	2.59511	2.63582	2.65251	2.65827	2.66872	2.67718	2.67776
2.69284	2.77651	2.80795	2.81062	2.86743	2.93144	3.02981	3.05749	3.06777	3.10783
3.17122	3.18344	3.19559	3.27958	3.34874	3.36533	3.41301	3.42896	3.42971	3.46123
3.47247	3.49316	3.52073	3.53877	3.54192	3.55984	3.56077	3.60259	3.61848	3.6226
3.63122	3.67384	3.69936	3.69958	3.73949	3.74309	3.74666	3.75287	3.7595	3.76358
3.77599	3.81906	3.84307	3.86498	3.89272	3.92409	3.92595	3.92923	3.98583	3.98894
4.00894	4.02514	4.04054	4.09231	4.09599	4.09807	4.14362	4.15512	4.27802	4.29168
4.31756	4.39671	4.42812	4.43255	4.4658	4.4732	4.49154	4.51619	4.56522	4.59439
4.60367	4.65954	4.66328	4.68766	4.71168	4.74384	4.79316	4.79656	4.91258	4.9224
4.98158	5.10626	5.1322	5.18887	5.29894	5.32927	5.4018	5.43263	5.51285	5.54017
5.57319	5.6337	5.65735	5.65853	5.74012	5.74957	5.80177	5.81203	5.83924	5.84915
5.87172	5.93382	5.93756	5.94408	5.97344	5.98346	6.01089	6.04676	6.10375	6.10689

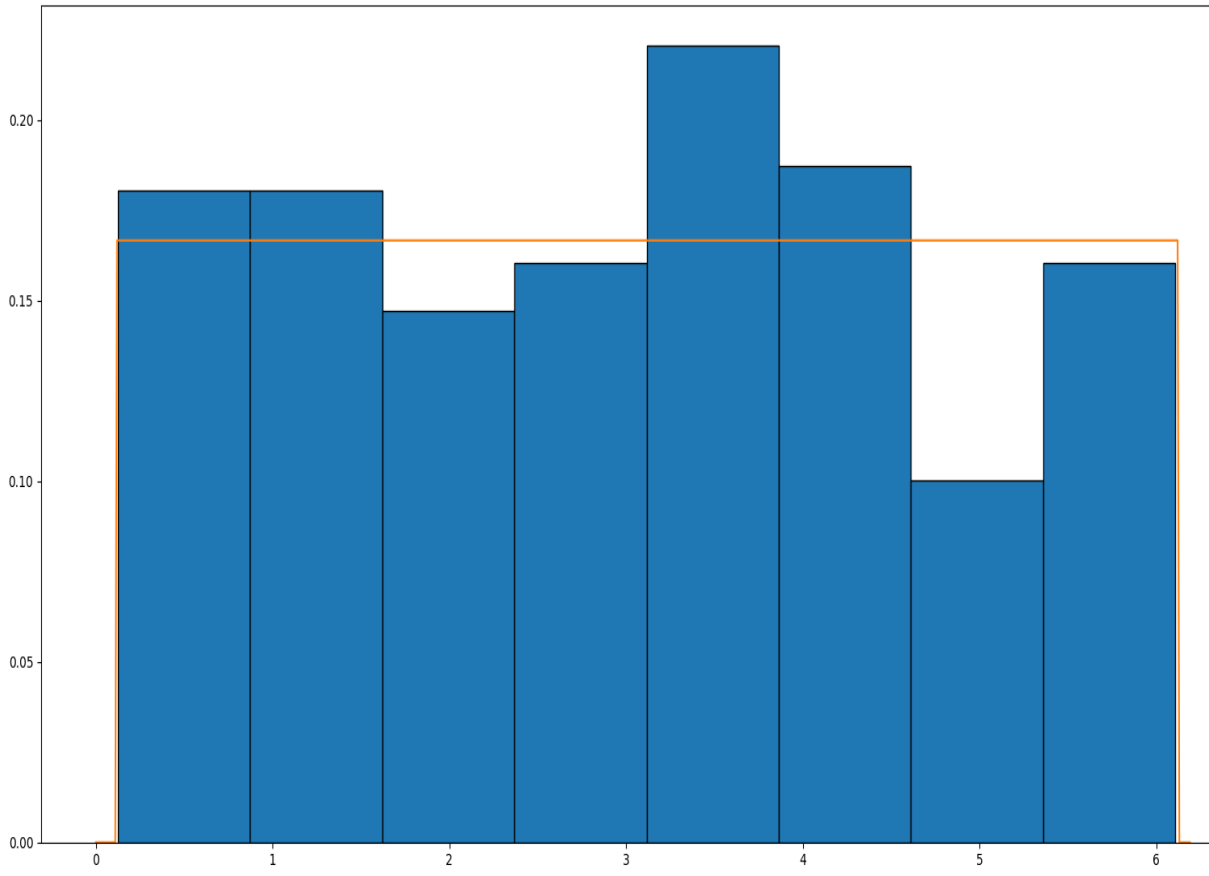
Интервальный ряд

Интервалы	n_i	w_i
[0.12, 0.87]	27	0.135
(0.87, 1.62]	27	0.135
(1.62, 2.37]	22	0.11
(2.37, 3.12]	24	0.12
(3.12, 3.87]	34	0.17
(3.87, 4.62]	27	0.135
(4.62, 5.37]	15	0.075
(5.37, 6.12]	24	0.12
	200	1

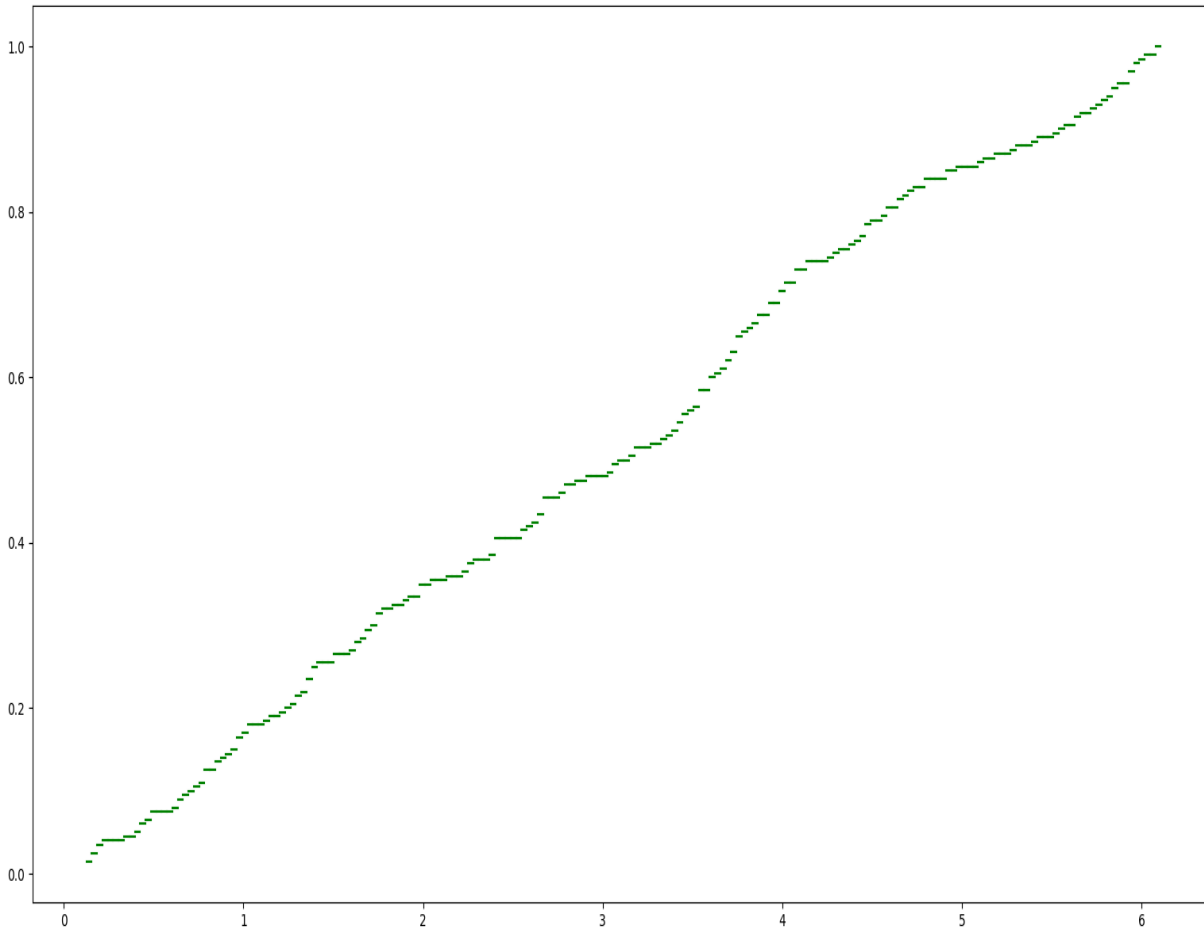
Ассоциированный статистический ряд

x_i	n_i	w_i
0.495	27	0.135
1.245	27	0.135
1.995	22	0.11
2.745	24	0.12
3.495	34	0.17
4.245	27	0.135
4.995	15	0.075
5.745	24	0.12
	200	1

Гистограмма относительных частот



Эмпирическая функция распределения



Результаты расчетов требуемых характеристик

Выборочное среднее: 3.015
Выборочная дисперсия с поправкой Шеппарда: 2.78835
Выборочное среднее квадратическое отклонение: 1.66984
Выборочная мода: 3.56118
Выборочная медиана: 3.13953
Выборочный коэффициент асимметрии: 0.05706
Выборочный коэффициент эксцесса: -1.06747

Сравнения относительных частот и теоретических вероятностей попадания в интервалы

Интервалы	w_i	p_i	$\lvert w_i - p_i \rvert$
[0.12, 0.87]	0.135	0.125	0.01
(0.87, 1.62]	0.135	0.125	0.01
(1.62, 2.37]	0.11	0.125	0.015
(2.37, 3.12]	0.12	0.125	0.005
(3.12, 3.87]	0.17	0.125	0.045
(3.87, 4.62]	0.135	0.125	0.01
(4.62, 5.37]	0.075	0.125	0.05
(5.37, 6.12]	0.12	0.125	0.005
	1	1	0.05

Сравнения рассчитанных характеристик с теоретическими значениями

Название показателя	Экспериментальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	3.015	3.12	0.105	0.03365
Выборочная дисперсия	2.78835	3	0,21165	0,07055
Выборочное среднее Квадратичное отклонение	1.66984	1.73205	0.06222	0.03592
Выборочная мода	3.56118	3.12	0.44118	0.1414
Выборочная медиана	3.13953	3.12	0.01953	0.00626
Выборочный коэффициент асимметрии	0.05706	0	0.05706	-
Выборочный коэффициент эксцесса	-1.06747	-1.2	0.13253	-0.11044

Список литературы

1. Математическая статистика [Электронный ресурс]: метод. указания по выполнению лаб. работ / А.А. Лобузов — М.: МИРЭА, 2017.
2. Гмурман В. Е. Теория вероятностей и математическая статистика. Учеб. пособие для вузов. Изд. 7-е, стер.— М.: Высш. шк., 1999.— 479 с.: ил.
3. Письменный Д.Т. Конспект лекций по теории вероятностей, математической статистике и случайным процессам: учеб. пособие для вузов. – М.: Айрис-пресс, 2020.

Приложение

```
import numpy as np
import math
from scipy.stats import hypergeom
from scipy.stats import binom
from scipy.stats import norm
from scipy.stats import expon
from scipy.stats import uniform
import statistics
import pandas as pd

f = open('answer.txt', 'r+')
selection = np.random.binomial(n=11, p=0.23, size=200)
selection = [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, 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, 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, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 6, 6, 6, 7, 8, 8]
n = 11
p = 0.23
q = 0.77
selection_list = list(selection)
selection_numbers = list(set(selection))
frequency = []
relative_frequency = []
ter_frequency = []
rz = []
O_rz = []

for number in selection_numbers:
    frequency.append(selection_list.count(number))
    relative_frequency.append(selection_list.count(number) / 200)
    ter_frequency.append(binom.pmf(k=number, n=11, p=0.23))

for i in range(len(relative_frequency)):
    rz.append(abs(relative_frequency[i] - ter_frequency[i]))

sum_frequency = []
for _ in range(len(relative_frequency)):
    sum_frequency.append(None)

for i in range(len(relative_frequency)):
```

```
            sum_frequency[i] = 0
            for j in range(i):
                sum_frequency[i] += relative_frequency[j]

        mean = 0.0
        for i in range(len(selection_numbers)):
            mean += frequency[i] * selection_numbers[i]
        mean = mean / 200

        s_m1 = s_m2 = s_m3 = s_m4 = 0.0
        for i in range(len(selection_numbers)):
            s_m1 += relative_frequency[i] * selection_numbers[i]
            s_m2 += relative_frequency[i] * (selection_numbers[i] ** 2)
            s_m3 += relative_frequency[i] * (selection_numbers[i] ** 3)
            s_m4 += relative_frequency[i] * (selection_numbers[i] ** 4)
        Sample_variance = s_m2 - (s_m1 ** 2)
        Smqd = math.sqrt(Sample_variance)

        k = relative_frequency.index(max(relative_frequency))
        mode = selection_numbers[k]
        i=0
        while(sum_frequency[i]<0.5):
            i+=1
            if(sum_frequency[i]==0.5):
                break
        i = i-1
        if(sum_frequency[i]==0.5):
            median = (selection_numbers[i] +selection_numbers[i+1])/2
        else:
            median = selection_numbers[i]

        print(mode)
        S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
        S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

        last_table = [mean, Sample_variance, Smqd, mode, mode, median, S_a_c,
S_k_c]
        last_table_T = [n * p, n * p * q, math.sqrt(n * p * q), math.floor((n + 1) *
p), (n + 1) * p - 0.5, math.ceil(n * p),
                        (q - p) / math.sqrt(n * p * q), (1 - 6 * p * q) / (n * p * q)]
        L_rz = []
        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
```

```
    if last_table_T[i] != 0:
        O_rz.append(L_rz[i] / last_table_T[i])
    if last_table_T[i] == 0 and L_rz[i] == 0:
        O_rz.append(0)

f.write(str(selection))
f.write('\n')
selection_list.sort()
f.write(str(selection_list))
f.write('\n')
f.write(str(selection_numbers))
f.write('\n')
f.write(str(frequency))
f.write('\n')
f.write(str(relative_frequency))
f.write('\n')
f.write(str(sum_frequency))
f.write('\n')
f.write(str(ter_frequency))
f.write('\n')
f.write(str(rz))
f.write('\n')
f.write(str(last_table))
f.write('\n')
f.write(str(last_table_T))
f.write('\n')
f.write(str(L_rz))
f.write('\n')
f.write(str(O_rz))
f.write('\n')
f.write('\n')

selection = np.random.geometric(p=0.23, size=200)
selection = [0, 1, 0, 0, 2, 0, 9, 6, 0, 18,
             0, 0, 0, 2, 5, 11, 7, 5, 2, 4,
             5, 7, 2, 1, 0, 0, 0, 0, 0, 2,
             5, 5, 1, 1, 5, 2, 19, 1, 1, 1,
             14, 0, 1, 1, 3, 0, 2, 1, 0, 10,
             7, 0, 15, 0, 1, 7, 3, 2, 1, 0,
             0, 3, 4, 4, 3, 6, 2, 1, 2, 2,
             0, 3, 5, 1, 9, 1, 2, 1, 2, 1,
             1, 0, 6, 6, 2, 0, 4, 1, 0, 6,
             5, 3, 4, 1, 20, 0, 2, 8, 3, 2,
             3, 1, 0, 0, 10, 0, 3, 0, 5, 2,
```

```
        7, 1, 3, 0, 6, 6, 7, 0, 2, 0,
        7, 4, 2, 6, 0, 2, 8, 3, 0, 0,
        7, 2, 10, 1, 0, 11, 4, 0, 1, 2,
        6, 0, 3, 2, 3, 6, 0, 3, 0, 0,
        1, 0, 0, 3, 9, 4, 8, 1, 0, 4,
        4, 13, 1, 6, 15, 1, 10, 0, 1, 0,
        0, 0, 2, 4, 1, 11, 1, 10, 0, 13,
        1, 4, 6, 1, 1, 0, 6, 0, 3, 1,
        3, 6, 8, 1, 4, 0, 1, 1, 5, 1, ]
q = 0.77
p = 0.23
selection_list = list(selection)
selection_numbers = list(set(selection))
frequency = []
relative_frequency = []
ter_frequency = []
rz = []
O_rz = []

for number in selection_numbers:
    frequency.append(selection_list.count(number))
    relative_frequency.append(selection_list.count(number) / 200)
    ter_frequency.append((q ** (number - 1)) * p)

for i in range(len(relative_frequency)):
    rz.append(abs(relative_frequency[i] - ter_frequency[i]))

sum_frequency = []
for _ in range(len(relative_frequency)):
    sum_frequency.append(None)

for i in range(len(relative_frequency)):
    sum_frequency[i] = 0
    for j in range(i):
        sum_frequency[i] += relative_frequency[j]

mean = 0.0
for i in range(len(selection_numbers)):
    mean += frequency[i] * selection_numbers[i]
mean = mean / 200

s_m1 = s_m2 = s_m3 = s_m4 = 0.0
for i in range(len(selection_numbers)):
    s_m1 += relative_frequency[i] * selection_numbers[i]
```

```
            s_m2 += relative_frequency[i] * (selection_numbers[i] ** 2)
            s_m3 += relative_frequency[i] * (selection_numbers[i] ** 3)
            s_m4 += relative_frequency[i] * (selection_numbers[i] ** 4)
        Sample_variance = s_m2 - (s_m1 ** 2)
        Smqd = math.sqrt(Sample_variance)

        k = relative_frequency.index(max(relative_frequency))
        mode = selection_numbers[k]
        i=0
        while(sum_frequency[i]<0.5):
            i+=1
            if(sum_frequency[i]==0.5):
                break
        i = i-1
        if(sum_frequency[i]==0.5):
            median = (selection_numbers[i] +selection_numbers[i+1])/2
        else:
            median = selection_numbers[i]
        S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
        S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

        last_table = [mean, Sample_variance, Smqd, mode, median, median, S_a_c,
S_k_c]
        last_table_T = [q / p, q / (p * p), math.sqrt(q) / p, 0, math.floor(-math.log(2)
/ math.log(q)),
                        -math.log(2) / math.log(q) - 0.5, (2 - p) / math.sqrt(q), 6 + (p * p)
/ q]
        L_rz = []

        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
            if last_table_T[i] != 0:
                O_rz.append(L_rz[i] / last_table_T[i])
            if last_table_T[i] == 0 and L_rz[i] == 0:
                O_rz.append(0)

        f.write(str(selection))
        f.write('\n')
        selection_list.sort()
        f.write(str(selection_list))
        f.write('\n')
        f.write(str(selection_numbers))
        f.write('\n')
```

```
        f.write(str(frequency))
        f.write('\n')
        f.write(str(relative_frequency))
        f.write('\n')
        f.write(str(sum_frequency))
        f.write('\n')
        f.write(str(ter_frequency))
        f.write('\n')
        f.write(str(rz))
        f.write('\n')
        f.write(str(last_table))
        f.write('\n')
        f.write(str(last_table_T))
        f.write('\n')
        f.write(str(L_rz))
        f.write('\n')
        f.write(str(O_rz))
        f.write('\n')
        f.write('\n')

        selection = np.random.poisson(0.56, 200)
        p = 0.56
        selection = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 3, 3, 3, 3, 4]
        selection_list = list(selection)
        selection_numbers = list(set(selection))
        frequency = []
        relative_frequency = []
        ter_frequency = []
        rz = []
        O_rz = []

        for number in selection_numbers:
            frequency.append(selection_list.count(number))
            relative_frequency.append(selection_list.count(number) / 200)
            ter_frequency.append(((p ** number) / (math.factorial(number))) *
math.e ** (-p))

        for i in range(len(relative_frequency)):
```

```
        rz.append(abs(relative_frequency[i] - ter_frequency[i]))

    sum_frequency = []
    for _ in range(len(relative_frequency)):
        sum_frequency.append(None)

    for i in range(len(relative_frequency)):
        sum_frequency[i] = 0
        for j in range(i):
            sum_frequency[i] += relative_frequency[j]

    mean = 0.0
    for i in range(len(selection_numbers)):
        mean += frequency[i] * selection_numbers[i]
    mean = mean / 200

    s_m1 = s_m2 = s_m3 = s_m4 = 0.0
    for i in range(len(selection_numbers)):
        s_m1 += relative_frequency[i] * selection_numbers[i]
        s_m2 += relative_frequency[i] * (selection_numbers[i] ** 2)
        s_m3 += relative_frequency[i] * (selection_numbers[i] ** 3)
        s_m4 += relative_frequency[i] * (selection_numbers[i] ** 4)
    Sample_variance = s_m2 - (s_m1 ** 2)
    Smqd = math.sqrt(Sample_variance)

    k = relative_frequency.index(max(relative_frequency))
    mode = selection_numbers[k]
    i=0
    while(sum_frequency[i]<0.5):
        i+=1
        if(sum_frequency[i]==0.5):
            break
    i = i-1
    if(sum_frequency[i]==0.5):
        median = (selection_numbers[i] +selection_numbers[i+1])/2
    else:
        median = selection_numbers[i]

    S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
    S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

    last_table = [mean, Sample_variance, Smqd, mode, median, S_a_c, S_k_c]
```

```
        last_table_T = [p, p, math.sqrt(p), math.floor(p), math.floor(p + 1 / 3 - 0.02
/ p), 1 / math.sqrt(p),
                        1 / p]
        L_rz = []

        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
            if last_table_T[i] != 0:
                O_rz.append(L_rz[i] / last_table_T[i])
            if last_table_T[i] == 0 and L_rz[i] == 0:
                O_rz.append(0)

        f.write(str(selection))
        f.write('\n')
        selection_list.sort()
        f.write(str(selection_list))
        f.write('\n')
        f.write(str(selection_numbers))
        f.write('\n')
        f.write(str(frequency))
        f.write('\n')
        f.write(str(relative_frequency))
        f.write('\n')
        f.write(str(sum_frequency))
        f.write('\n')
        f.write(str(ter_frequency))
        f.write('\n')
        f.write(str(rz))
        f.write('\n')
        f.write(str(last_table))
        f.write('\n')
        f.write(str(last_table_T))
        f.write('\n')
        f.write(str(L_rz))
        f.write('\n')
        f.write(str(O_rz))
        f.write('\n')
        f.write('\n')
        selection = np.random.random_integers(0, 10, 200)
        selection = [0, 0, 0, 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, 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, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 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, 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, 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]
        n = 11
        selection_list = list(selection)
        selection_numbers = list(set(selection))
        frequency = []
        relative_frequency = []
        ter_frequency = []
        rz = []
        O_rz = []

        for number in selection_numbers:
            frequency.append(selection_list.count(number))
            relative_frequency.append(selection_list.count(number) / 200)
            ter_frequency.append(1 / 11)

        for i in range(len(relative_frequency)):
            rz.append(abs(relative_frequency[i] - ter_frequency[i]))

        sum_frequency = []
        for _ in range(len(relative_frequency)):
            sum_frequency.append(None)

        for i in range(len(relative_frequency)):
            sum_frequency[i] = 0
            for j in range(i):
                sum_frequency[i] += relative_frequency[j]

        mean = 0.0
        for i in range(len(selection_numbers)):
            mean += frequency[i] * selection_numbers[i]
        mean = mean / 200

        s_m1 = s_m2 = s_m3 = s_m4 = 0.0
        for i in range(len(selection_numbers)):
            s_m1 += relative_frequency[i] * selection_numbers[i]
            s_m2 += relative_frequency[i] * (selection_numbers[i] ** 2)
            s_m3 += relative_frequency[i] * (selection_numbers[i] ** 3)
            s_m4 += relative_frequency[i] * (selection_numbers[i] ** 4)
        Sample_variance = s_m2 - (s_m1 ** 2)
        Smqd = math.sqrt(Sample_variance)

        k = relative_frequency.index(max(relative_frequency))
        mode = selection_numbers[k]
```

```
        i=0
        while(sum_frequency[i]<0.5):
            i+=1
            if(sum_frequency[i]==0.5):
                break
        i = i-1
        if(sum_frequency[i]==0.5):
            median = (selection_numbers[i] +selection_numbers[i+1])/2
        else:
            median = selection_numbers[i]

        S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
        S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

        last_table = [mean, Sample_variance, Smqd, mode, median, S_a_c, S_k_c]
        last_table_T = [(n - 1) / 2, ((n * n) - 1) / 12, 0.5 * math.sqrt(((n * n) - 1) / 3),
(n - 1) / 2, (n - 1) / 2,
                        0, -(6 / 5) * (((n * n) + 1) / ((n * n)) - 1)]
        L_rz = []

        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
            if last_table_T[i] != 0:
                O_rz.append(L_rz[i] / last_table_T[i])
            if last_table_T[i] == 0 and L_rz[i] == 0:
                O_rz.append(0)

        f.write(str(selection))
        f.write('\n')
        selection_list.sort()
        f.write(str(selection_list))
        f.write('\n')
        f.write(str(selection_numbers))
        f.write('\n')
        f.write(str(frequency))
        f.write('\n')
        f.write(str(relative_frequency))
        f.write('\n')
        f.write(str(sum_frequency))
        f.write('\n')
        f.write(str(ter_frequency))
        f.write('\n')
        f.write(str(rz))
```

```
        f.write('\n')
        f.write(str(last_table))
        f.write('\n')
        f.write(str(last_table_T))
        f.write('\n')
        f.write(str(L_rz))
        f.write('\n')
        f.write(str(O_rz))
        f.write('\n')
        f.write('\n')
        selection = np.random.hypergeometric(12, 13, 11, 200)
        selection = [0, 2, 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, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 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, 9]
        m = 11
        M = 25
        K = 12
        selection_list = list(selection)
        selection_numbers = list(set(selection))
        frequency = []
        relative_frequency = []
        ter_frequency = []
        rz = []
        O_rz = []

        for number in selection_numbers:
            frequency.append(selection_list.count(number))
            relative_frequency.append(selection_list.count(number) / 200)
            ter_frequency.append(hypergeom.cdf(number, 25, 11, 12))

        for i in range(len(relative_frequency)):
            rz.append(abs(relative_frequency[i] - ter_frequency[i]))

        sum_frequency = []
        for _ in range(len(relative_frequency)):
            sum_frequency.append(None)

        for i in range(len(relative_frequency)):
            sum_frequency[i] = 0
            for j in range(i):
```

```
            sum_frequency[i] += relative_frequency[j]

        mean = 0.0
        for i in range(len(selection_numbers)):
            mean += frequency[i] * selection_numbers[i]
        mean = mean / 200

        s_m1 = s_m2 = s_m3 = s_m4 = 0.0
        for i in range(len(selection_numbers)):
            s_m1 += relative_frequency[i] * selection_numbers[i]
            s_m2 += relative_frequency[i] * (selection_numbers[i] ** 2)
            s_m3 += relative_frequency[i] * (selection_numbers[i] ** 3)
            s_m4 += relative_frequency[i] * (selection_numbers[i] ** 4)
        Sample_variance = s_m2 - (s_m1 ** 2)
        Smqd = math.sqrt(Sample_variance)

        k = relative_frequency.index(max(relative_frequency))
        mode = selection_numbers[k]
        i=0
        while(sum_frequency[i]<0.5):
            i+=1
            if(sum_frequency[i]==0.5):
                break
        i = i-1
        if(sum_frequency[i]==0.5):
            median = (selection_numbers[i] +selection_numbers[i+1])/2
        else:
            median = selection_numbers[i]

        S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
        S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

        last_table = [mean, Sample_variance, Smqd, mode, median, median, S_a_c,
S_k_c]
        last_table_T = [m * K / M, (m * K * (M - K) * (M - m)) / ((M - 1) * (M *
M)),
                        (1 / M) * math.sqrt((m * K * (M - K) * (M - m)) / (M - 1)),
                        math.floor(((K + 1) * (m - 1)) / (M + 2)), m, m + 0.5,
                        ((M - 2 * K) * (M - 2 * m) / (M - 2)) * math.sqrt((M - 1) / (m * K
* (M - K) * (M - m))),
                        math.floor(((M - 1) * M * M) / (m * (M - 2) * (M - 3) * (M - m)))
* math.floor(
```

```
                    (M * (M + 1) - 6 * M * (M - m)) / (K * (M - K)) + (3 * m * (M
+ 6) * (M - m)) / (M * M) - 6)]
        L_rz = []

        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
            if last_table_T[i] != 0:
                O_rz.append(L_rz[i] / last_table_T[i])
            if last_table_T[i] == 0 and L_rz[i] == 0:
                O_rz.append(0)

        f.write(str(selection))
        f.write('\n')
        selection_list.sort()
        f.write(str(selection_list))
        f.write('\n')
        f.write(str(selection_numbers))
        f.write('\n')
        f.write(str(frequency))
        f.write('\n')
        f.write(str(relative_frequency))
        f.write('\n')
        f.write(str(sum_frequency))
        f.write('\n')
        f.write(str(ter_frequency))
        f.write('\n')
        f.write(str(rz))
        f.write('\n')
        f.write(str(last_table))
        f.write('\n')
        f.write(str(last_table_T))
        f.write('\n')
        f.write(str(L_rz))
        f.write('\n')
        f.write(str(O_rz))
        f.write('\n')
        f.write('\n')
        f.close()

        f = open('answer1.txt', 'r+')
        selection = np.random.normal(0.3, 1.03 , 200)
        selection = [1.1658, 0.03992, 1.01534, 0.07553, -0.92043, 0.70086,
0.35958, 0.39276, -1.48112, -1.2337, -1.30807, -2.14042, 0.45475, -1.31484, -
0.4506, -0.69883, 1.72924, -0.71028, -0.39715, 0.81071, -0.6385, 0.40144,
```

0.92914, 0.6065, 1.07279, 0.74965, 1.60671, 2.28143, -0.07828, -0.1886, 0.64163, -0.45412, 0.89457, 4.23547, 1.1833, -0.13018, -0.51238, -0.61987, 0.62851, 0.23584, 3.21512, -1.08879, 1.81319, -0.34664, -0.05514, -1.39536, 0.85688, 0.71801, -0.31807, 2.36985, -0.43662, 0.18998, -0.17463, 0.46352, 0.60267, 1.0832, -0.55948, 0.15433, -1.33625, -1.15913, 1.23691, -1.02543, -0.47616, 0.99325, 1.10945, -1.32368, 0.31241, 0.45591, 0.65407, -0.53396, 1.04851, -2.02812, -2.1909, 1.68378, 0.33701, 0.15507, -0.64901, 0.23099, -1.00966, 0.80459, -1.05305, -0.13572, 0.29378, 0.01389, 0.27026, 0.82507, 1.00571, 2.0791, 0.16196, 1.2796, 0.86837, 0.11846, -0.36415, -0.20819, 0.14569, 0.67373, -1.22727, -0.8153, -0.03751, 0.92959, 0.01598, 0.13447, -0.82175, 0.95921, 1.02299, 0.89841, -1.11187, 0.47599, -0.117, -0.73716, -0.80665, -0.83037, -0.90396, -1.93622, 0.88252, 0.87349, 0.84677, 0.19989, 0.17481, -0.5227, 0.59634, 0.56526, 0.75848, -0.31772, -1.12639, 0.40775, 0.6983, 0.95638, 1.16627, 1.30625, 1.00296, 0.90736, 0.72462, 1.44972, 0.86584, -1.35144, -1.11026, 0.39271, 1.04747, -1.65366, 0.37406, -0.65462, -0.0471, 0.62038, 1.40446, 0.97301, -0.62792, -1.89807, 0.04357, 0.34359, -0.03187, 0.91351, 0.8047, 1.29796, -0.65203, 0.44832, 0.87969, -1.10788, 1.38339, -0.86818, -0.05037, -1.73752, -0.0647, -0.43745, 0.3148, -0.52213, 1.62558, 0.2308, 2.37026, 0.40379, 0.64067, -0.49419, 0.3358, 0.85429, -0.61423, 1.34232, 0.94634, 0.33478, -0.75317, 0.82504, -0.91629, -1.78466, -0.6901, 0.98902, 1.65466, 2.0217, -2.10385, -1.0686, -0.51435, -0.24632, 1.07106, -0.2019, 1.37149, 1.14914, -0.04509, 0.02509, 0.18447, 1.84519, 0.30137, 0.24753]

```
mu = 0.3
sigma = 1.03 ** 2
selection_list = list(selection)
m = 1 + math.floor(math.log2(200))
a = [None] * (m+1)
x = []
a[0] = min(selection_list)
a[m] = max(selection_list)
for i in range(1, m):
    a[i] = a[i - 1] + (a[m] - a[0]) / m
a.sort()
for i in range(1, m):
    x.append((a[i - 1] + a[i]) / 2)


pan = pd.Series(selection_list)
print(pan)
frequency = pan.groupby(pd.cut(pan, bins=a, right=True)).count()
frequency = frequency.tolist()
```

```
        frequency[0]+=1
        relative_frequency = []
        ter_frequency = []
        rz = []
        O_rz = []

        for i in range(len(frequency)):
            relative_frequency.append(frequency[i] / 200)
            ter_frequency.append(abs(norm.cdf(a[i], loc=mu, scale=1.03) -
norm.cdf(a[i + 1], loc=mu, scale=1.03)))

        for i in range(len(relative_frequency)):
            rz.append(abs(relative_frequency[i] - ter_frequency[i]))

        mean = 0.0
        for i in range(len(x)):
            mean += frequency[i] * x[i]
        mean = mean / 200
        s_m1 = s_m2 = s_m3 = s_m4 = 0.0
        for i in range(len(x)):
            s_m1 += relative_frequency[i] * x[i]
            s_m2 += relative_frequency[i] * (x[i] ** 2)
            s_m3 += relative_frequency[i] * (x[i] ** 3)
            s_m4 += relative_frequency[i] * (x[i] ** 4)
        Sample_variance = 0.0
        h = (a[m-1]-a[0])/m
        for i in range(1, m-1):
            Sample_variance += ((x[i] - mean)**2)*relative_frequency[i]
        Sample_variance = Sample_variance - (h*h)/12
        Smqd = math.sqrt(Sample_variance)

        k = relative_frequency.index(max(relative_frequency))
        ak = a[k]
        mode = [ak + h * (relative_frequency[k] - relative_frequency[k - 1]) / (
        2 * relative_frequency[k] - relative_frequency[k - 1] - relative_frequency[
        (k + 1) % len(relative_frequency)])]
        median = statistics.median(selection_list)

        S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
        S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

        last_table = [mean, Sample_variance, Smqd, mode[0], median, S_a_c,
S_k_c]
```

```
last_table_T = [0.3, 1.03 ** 2, 1.03, 0.3, 0.3, 0, 0]
L_rz = []
for i in range(len(last_table_T)):
    L_rz.append(abs(last_table[i] - last_table_T[i]))
    if last_table_T[i] != 0:
        O_rz.append(L_rz[i] / last_table_T[i])
    if last_table_T[i] == 0 and L_rz[i] == 0:
        O_rz.append(0)
f.write(str([ round(elem, 5) for elem in selection ]))
f.write('\n')
selection_list.sort()
f.write(str([ round(elem, 5) for elem in selection_list ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in a ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in x ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in frequency ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in relative_frequency ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in ter_frequency ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in rz ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in last_table ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in last_table_T ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in L_rz ]))
f.write('\n')
f.write(str([ round(elem, 5) for elem in O_rz ]))
f.write('\n')
f.write('\n')


selection = expon.rvs(scale=1/2.06, size=200)
selection = [0.37721, 0.03169, 0.37151, 0.0769, 0.34037, 1.13088, 0.2575,
0.56454, 1.82127, 0.05064, 0.10365, 0.70715, 0.21093, 0.37443, 0.19333,
0.09039, 0.04419, 0.12236, 0.77516, 0.40823, 1.4607, 1.83618, 1.37025,
0.25452, 0.68557, 1.03029, 1.14552, 0.2294, 0.28135, 0.73949, 0.07434,
0.10757, 1.42173, 0.07836, 0.04259, 0.01033, 0.21435, 0.38462, 0.9403,
1.08397, 0.95688, 0.10289, 0.0546, 1.92442, 1.40552, 0.09963, 0.59998,
0.12916, 0.11015, 0.39198, 0.52947, 0.03863, 0.1736, 0.10894, 1.18496,
```

0.12376, 1.5779, 0.68347, 0.66878, 1.07169, 1.08322, 0.4471, 1.59374, 0.04513, 0.16902, 0.67763, 0.25521, 0.359, 0.34856, 0.21629, 2.68677, 1.53046, 0.24074, 0.15679, 0.18262, 0.12134, 1.59947, 1.40134, 0.83365, 1.5909, 0.00551, 0.16201, 0.19019, 0.46259, 1.81955, 0.03282, 0.98149, 0.56862, 0.49926, 0.65135, 0.0878, 0.06003, 1.27631, 0.02527, 0.12644, 0.53766, 0.44759, 0.01381, 0.75243, 0.096, 0.43248, 0.079, 0.06444, 0.08693, 0.05088, 0.62982, 0.66711, 0.00541, 0.28463, 0.1487, 0.63705, 0.19284, 0.03226, 0.38898, 1.77783, 0.3157, 0.05891, 0.54325, 0.71747, 1.06411, 0.32912, 0.34705, 0.55192, 0.01571, 0.44481, 0.43602, 0.59003, 0.24869, 0.1213, 0.16289, 0.46451, 0.59355, 0.35801, 1.29891, 0.18403, 0.15223, 2.21691, 1.21771, 0.02043, 0.00464, 0.17036, 1.13631, 0.33452, 0.09827, 2.42315, 0.94445, 0.25844, 0.36272, 0.56341, 0.19087, 0.1771, 0.26055, 0.09611, 0.48435, 0.09605, 0.57185, 0.46519, 1.0077, 0.18623, 0.2779, 0.08348, 0.01262, 0.05042, 0.47721, 0.36026, 0.74537, 0.05519, 0.49033, 1.72484, 0.15356, 0.30864, 0.19948, 1.60141, 0.00222, 0.0428, 1.12284, 0.75679, 0.30892, 0.00987, 0.91402, 0.01588, 0.1943, 0.14746, 0.35594, 1.59502, 1.02987, 0.4608, 0.60957, 0.19554, 0.74813, 0.01426, 0.11249, 1.66962, 0.44219, 0.71825, 1.16398, 0.04346, 0.45743, 0.60865, 0.14083]

```
lambd = 2.06
selection_list = list(selection)
m = 1 + math.floor(math.log2(200))
a = [None] * (m+1)
x = []
a[0] = min(selection_list)
a[m] = max(selection_list)

a[0] = 0
a[m] = max(selection_list)
for i in range(1, m):
    a[i] = a[i - 1] + (a[m] - a[0]) / m
for i in range(1, m):
    x.append((a[i - 1] + a[i]) / 2)

pan = pd.Series(selection_list)
frequency = pan.groupby(pd.cut(pan, bins=a, right=True)).count()
frequency = frequency.tolist()
relative_frequency = []
ter_frequency = []
rz = []
O_rz = []
print(sum(frequency))
for i in range(len(frequency)):
    relative_frequency.append(frequency[i] / 200)
```

```
        ter_frequency.append(abs(expon.cdf(x = a[i], scale = 1/lambd) -
expon.cdf(x = a[i+1], scale = 1/lambd)))

    for i in range(len(relative_frequency)):
        rz.append(abs(relative_frequency[i] - ter_frequency[i]))

    mean = 0.0
    for i in range(len(x)):
        mean += frequency[i] * x[i]
    mean = mean / 200
    s_m1 = s_m2 = s_m3 = s_m4 = 0.0
    for i in range(len(x)):
        s_m1 += relative_frequency[i] * x[i]
        s_m2 += relative_frequency[i] * (x[i] ** 2)
        s_m3 += relative_frequency[i] * (x[i] ** 3)
        s_m4 += relative_frequency[i] * (x[i] ** 4)
    Sample_variance = 0.0
    h = (a[m-1]-a[0])/m
    for i in range(1, m-1):
        Sample_variance += ((x[i] - mean)**2)*relative_frequency[i]
    Sample_variance = Sample_variance - (h*h)/12
    Smqd = math.sqrt(Sample_variance)

    k = relative_frequency.index(max(relative_frequency))
    ak = a[k]
    mode = [ak+h*(relative_frequency[k]-
relative_frequency[len(relative_frequency)-1])/(2*relative_frequency[k]-
relative_frequency[len(relative_frequency)-1]-
relative_frequency[(k+1)%len(relative_frequency)])]
    median = statistics.median(selection_list)

    S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
    S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

    last_table = [mean, Sample_variance, Smqd, mode[0], median, S_a_c,
S_k_c]
    last_table_T = [lambd**(-1), lambd**(-2), lambd**(-1), 0,
math.log(2)/lambd, 2, 6]
    L_rz = []

    for i in range(len(last_table_T)):
        L_rz.append(abs(last_table[i] - last_table_T[i]))
        if last_table_T[i] != 0:
```

```
            O_rz.append(L_rz[i] / last_table_T[i])
        if last_table_T[i] == 0 and L_rz[i] == 0:
            O_rz.append(0)
    f.write(str([ round(elem, 5) for elem in selection ]))
    f.write('\n')
    selection_list.sort()
    f.write(str([ round(elem, 5) for elem in selection_list ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in a ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in x ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in frequency ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in relative_frequency ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in ter_frequency ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in rz ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in last_table ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in last_table_T ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in L_rz ]))
    f.write('\n')
    f.write(str([ round(elem, 5) for elem in O_rz ]))
    f.write('\n')
    f.write('\n')

    selection = np.random.uniform(0.12, 6.12, 200)
    selection = [0.12562551492260876, 0.1272326588196918,
0.13693962957418 97, 0.17533268565879212, 0.18079993056914 69,
                 0.1921757167092718, 0.19922495406104213,
0.24435789958973364, 0.34443804186878835, 0.3995417773692066,
                 0.42734823985142023, 0.4329020890151357,
0.46189051602031606, 0.48988127406484605, 0.5072492218424186,
                 0.628531535239479, 0.6361746981421076,
0.6479075503856279, 0.6902630579657861, 0.7076354415522476,
                 0.7425551559881848, 0.7645196884125206,
0.7931707333494434, 0.7987184351648705, 0.811048132234817,
                 0.8507106637667311, 0.86442346827153, 0.876692275639506,
0.9305788546920793, 0.9546073496588595,
```

0.9781384063000343, 0.9901717653170478, 0.992767961783935, 0.9950895358772692, 1.0278481166156426,
1.052054688287091, 1.1182467556776534, 1.153118323529029, 1.2054330726271774, 1.2484579728911331,
1.282558646818837, 1.3008212154995515, 1.3159847155442832, 1.3436283386544376, 1.3595550986589968,
1.3602380389897575, 1.3750705715803093, 1.3975777101672864, 1.398207992944255, 1.4003853099675103,
1.4246849631058036, 1.5219092875121407, 1.5275875048183982, 1.5952994081713032, 1.6350676382701672,
1.6400779088455586, 1.68066908371132, 1.701360573842325, 1.7068232222260717, 1.7181919302738398,
1.7506514707796978, 1.7530338313747982, 1.7643062656788473, 1.7832642822541285, 1.839229338785609,
1.8986498908953924, 1.9327810326298858, 1.9898800135065988, 1.9979161756900243, 2.0017580685950485,
2.046079510022451, 2.1516443376470162, 2.2269025853612874, 2.2531484934685, 2.2658406730212564,
2.2908572477682, 2.3891815415675968, 2.401152044373532, 2.405267589004157, 2.4158320858197246,
2.422384286776534, 2.556404467799983, 2.565914416211781, 2.595107396043554, 2.6358160002948114,
2.652512980596657, 2.658273841307581, 2.668717784784176, 2.677177824325412, 2.6777581700601427,
2.6928353019824725, 2.776512530004556, 2.807953558606706, 2.8106224456249422, 2.8674268022000646,
2.931440242786512, 3.029814884530647, 3.0574862508707166, 3.06776575268203, 3.107825604136611,
3.1712246494681198, 3.183444247937006, 3.19558988827834, 3.279580535680761, 3.34874262456642,
3.3653266087751876, 3.4130117547429895, 3.4289558467876464, 3.429707884597394, 3.4612265217230136,
3.472472681463942, 3.4931555108173145, 3.520725244369217, 3.538771657082056, 3.5419189052203004,
3.5598404007997533, 3.56077259680631, 3.6025890658399033, 3.6184842317147483, 3.622598963028352,
3.6312163624616987, 3.673844689419713, 3.699360311574848, 3.699584238184599, 3.7394901362126474,
3.743093645304036, 3.7466573687129108, 3.7528664752726737, 3.759502542199381, 3.7635750497138556,
3.7759918444216707, 3.8190581090850637, 3.8430687878151777, 3.8649772373242564, 3.8927232051271354,
3.924090156694729, 3.9259508707491255, 3.929232585775269, 3.9858348504115755, 3.9889380803560135,

4.008942857938251, 4.025138255026329, 4.040541262518161, 4.0923095856343705, 4.095987037332115,
4.098072389259218, 4.143618301255864, 4.155121974493067, 4.278022360228854, 4.29167788368445,
4.317557031823923, 4.396709002172995, 4.428118100839357, 4.432551318956099, 4.465796351565596,
4.473200322801036, 4.491543410966609, 4.516186991997199, 4.565216838242331, 4.59439355855376,
4.603669341946267, 4.6595402461291275, 4.663275676373524, 4.687657717534638, 4.711675845002831,
4.743835306041133, 4.793160932614812, 4.796558074792212, 4.9125812290668565, 4.922395516845044,
4.981580640571454, 5.106257268972576, 5.132195235230196, 5.1888707752868, 5.29894406056668,
5.329265876480405, 5.401803167875951, 5.4326323551708935, 5.512847023208698, 5.540167462105237,
5.573194892165703, 5.633699796719307, 5.657349195565742, 5.658525137804989, 5.740116246629289,
5.74956982419598, 5.80176546829411, 5.8120331764405435, 5.839235088038437, 5.849153649751552,
5.871716631227106, 5.933824017837832, 5.9375642131599005, 5.944084152326927, 5.973435517157857,
5.983455810514716, 6.010887918991185, 6.046756999572009, 6.103750055130712, 6.106887898494262]

```
a1 = 0.12
b = 6.12
selection_list = list(selection)
m = 1 + math.floor(math.log2(200))
a = [None] * (m + 1)
x = []
a[0] = a1
a[m] = b

for i in range(1, m):
    a[i] = a[i - 1] + (a[m] - a[0]) / m
a.sort()
for i in range(1, m + 1):
    x.append((a[i - 1] + a[i]) / 2)

pan = pd.Series(selection_list)
frequency = pan.groupby(pd.cut(pan, bins=a, right=True)).count()
frequency = frequency.tolist()
relative_frequency = []
```

```
    ter_frequency = []
    rz = []
    O_rz = []

    for i in range(len(frequency)):
        relative_frequency.append(frequency[i] / 200)
        ter_frequency.append(abs(uniform.cdf(x=a[i], loc=a1, scale=6) -
uniform.cdf(x=a[i + 1], loc=a1, scale=6)))

    for i in range(len(relative_frequency)):
        rz.append(abs(relative_frequency[i] - ter_frequency[i]))

    mean = 0.0
    for i in range(len(x)):
        mean += frequency[i] * x[i]
    mean = mean / 200
    s_m1 = s_m2 = s_m3 = s_m4 = 0.0
    for i in range(len(x)):
        s_m1 += relative_frequency[i] * x[i]
        s_m2 += relative_frequency[i] * (x[i] ** 2)
        s_m3 += relative_frequency[i] * (x[i] ** 3)
        s_m4 += relative_frequency[i] * (x[i] ** 4)
    Sample_variance = 0.0
    h = (a[m] - a[0]) / m
    for i in range(len(x)):
        Sample_variance += ((x[i] - mean) ** 2) * relative_frequency[i]
    Sample_variance = Sample_variance - (h * h) / 12
    Smqd = math.sqrt(Sample_variance)

    k = relative_frequency.index(max(relative_frequency))
    ak = a[k]
    mode = [ak + h * (relative_frequency[k] - relative_frequency[k - 1]) / (
            2 * relative_frequency[k] - relative_frequency[k - 1] -
relative_frequency[
        (k + 1) % len(relative_frequency)])]
    median = statistics.median(selection_list)

    S_a_c = (s_m3 - 3 * s_m2 * s_m1 + 2 * (s_m1 ** 3)) / (Smqd ** 3)
    S_k_c = (s_m4 - 4 * s_m3 * s_m1 + 6 * s_m2 * (s_m1 ** 2) - 3 * (s_m1 **
4)) / (Smqd ** 4) - 3

    last_table = [mean, Sample_variance, Smqd, mode[0], median, S_a_c,
S_k_c]
```

```
        last_table_T = [(a1 + b) / 2, ((a1 + b) * (a1 + b)) / 12, (b - a1) / (2 *
math.sqrt(3)), (a1 + b) / 2, (a1 + b) / 2, 0,
                        -6 / 5]
        L_rz = []
        for i in range(len(last_table_T)):
            L_rz.append(abs(last_table[i] - last_table_T[i]))
            if last_table_T[i] != 0:
                O_rz.append(L_rz[i] / last_table_T[i])
            if last_table_T[i] == 0 and L_rz[i] == 0:
                O_rz.append(0)
        f.write(str([round(elem, 5) for elem in selection]))
        f.write('\n')
        selection_list.sort()
        f.write(str([round(elem, 5) for elem in selection_list]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in a]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in x]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in frequency]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in relative_frequency]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in ter_frequency]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in rz]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in last_table]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in last_table_T]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in L_rz]))
        f.write('\n')
        f.write(str([round(elem, 5) for elem in O_rz]))
        f.write('\n')
        f.write('\n')
        f.close()
```