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

ИНСТИТУТ КИБЕРНЕТИКИ

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 2

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

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

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

Группа: КМБО-01-16

МОСКВА 2019

Лабораторная работа по Математической статистике № 2
«Первичная обработка выборки из непрерывной генеральной совокупности»

Задание 1. Получить выборку, сгенерировав 200 псевдослучайных чисел, распределенных по нормальному закону с параметрами

$a = (-1)^V \cdot 0{,}1 \cdot V$ и σ^2, где $\sigma = 0{,}01 \cdot V + 1$

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

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

Задание 3. Получить выборку, сгенерировав 200 псевдослучайных чисел, распределенных равномерно на отрезке $[a, b]$.

$a = (-1)^V \cdot 0{,}05 \cdot V$, $b = a + 0{,}05 \cdot V + 1$

V – номер варианта.

Для каждого Задания:

Построить:

1) группированную выборку (интервальный вариационный 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.

V=26 – номер варианта.

Вычисления проводить с точностью до 0,00001.

Краткие теоретические сведения

При построении группированной выборки (интервального вариационного ряда) число интервалов $[a_0, a_1], (a_1, a_2],\ldots, (a_{m-1}, a_m]$ определяется по формуле Стерджеса $m = 1 + [log_2 N]$, $a_0 = x_{(1)}$, $a_m = x_{(N)}$, $d = a_m - a_0$, $a_k - a_{k-1} = d/m$.

Интервальный ряд (группированная выборка) имеет вид:

$[a_{i-1},a_i]$	$[a_0,a_1]$	…	$(a_{m-1},a_m]$
$\boldsymbol{n_i}$	$\boldsymbol{n_1}$	…	$\boldsymbol{n_m}$
$\boldsymbol{w_i}$	$\boldsymbol{w_1}$	…	$\boldsymbol{w_m}$

Ассоциированный статистический ряд:

x_i^*	x_1^*	…	x_m^*
$\boldsymbol{n_i}$	$\boldsymbol{n_1}$	…	$\boldsymbol{n_m}$
$\boldsymbol{w_i}$	$\boldsymbol{w_1}$	…	$\boldsymbol{w_m}$

$x_i^* = \frac{a_{i-1}+a_i}{2}$ – середина интервала $(a_{i-1}, a_i]$Полигон

Эмпирическая функция распределения

$$F_N^{\text{Э}}(x; x_1, x_2, \ldots, x_N) = \sum_{x_i \le x} \frac{1}{N}$$

Выборочное среднее

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

Выборочная дисперсия с поправкой Шеппарда

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

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

$$\bar{\sigma} = \sqrt{S_B^2}$$

Выборочная медиана

$$\overline{M_e} = \begin{cases} a_{k-1} - \dfrac{h}{w_k}\left(\dfrac{1}{2} - \sum_{i=1}^{k-1} w_i\right), & \sum_{i=1}^{k-1} w_i < 0{,}5 < \sum_{i=1}^{k} w_i \\ a_k, & \sum_{i=1}^{k} w_i = 0{,}5 \end{cases}$$

Выборочная мода

$$\overline{M_0} = a_{k-1} - h\frac{w_k - w_{k-1}}{2w_k - w_{k-1} - w_{k+1}}$$

Выборочный коэффициент асимметрии

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

Выборочный коэффициент эксцесса

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

Нормальное распределение

Характеристика	Значение
Вероятность	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}$
Математическое ожидание	a
Дисперсия	σ^2
Среднее квадратичное отклонение	σ
Мода	a
Медиана	A
Коэффициент асимметрии	0
Коэффициент эксцесса	0

Показательное распределение

Характеристика	Значение
Вероятность	$\begin{cases} 0, x < 0 \\ \lambda e^{-\lambda x}, x \geq 0 \end{cases}$
Математическое ожидание	λ^{-1}
Дисперсия	λ^{-2}
Среднее квадратичное отклонение	λ^{-1}
Мода	0
Медиана	$\frac{\ln 2}{\lambda}$
Коэффициент асимметрии	2
Коэффициент эксцесса	6

Равномерное распределение на отрезке $[a,b]$

Характеристика	Значение
Вероятность	$\begin{cases} 0, x \notin (a,b) \\ \frac{1}{b-a}, x \in (a,b) \end{cases}$
Математическое ожидание	$\frac{a+b}{2}$
Дисперсия	$\frac{(b-a)^2}{12}$
Среднее квадратичное отклонение	$\frac{b-a}{2\sqrt{3}}$
Мода	$\frac{a+b}{2}$
Медиана	$\frac{a+b}{2}$
Коэффициент асимметрии	0
Коэффициент эксцесса	$-\frac{6}{5}$

Средства высокоуровневого интерпретируемого языка программирования Python, которые использованы в программе расчета

scipy.stats – модуль библиотеки scipy для работы со статистикой, в том числе и распределениями. Объектами класса являются типы распределений, методами – все методы, проходимые в курсе.

numpy – библиотека для эффективной работы с числами / массивами.

itertools.islice – метод библиотеки itertools для быстрого / эффективного взятия “слайсов”.

math.log2 – метод библиотеки math для взятия двоичного логарифма.

collections.Counter – класс словарь библиотеки collections, отличающийся от стандартного словаря возможностью автоматически считать частоту объектов.

matplotlib.pyplot – библиотека для построения графиков.

functools.reduce – метод reduce библиотеки functools.

copy.deepcopy – метод для копирования сложных структур.

random.choice – метод для выбора из предложенных вариантов.

collections.namedtuple – именованная структура.

Результаты расчетов с комментариями

Задание 1) Распределение по нормальному закону

a=-0.9

σ *=1.1881*

Полученная выборка

-2.3993	-1.7389	-1.0815	-0.367	-2.2783	-0.9521	-0.7503	-0.4684	1.2362	-4.3891
0.5003	-0.9611	1.1335	1.7393	-1.5632	-1.195	1.0155	-0.1805	-0.5313	-1.7189
-0.4256	1.7407	-1.4188	1.9883	-0.5631	-1.0645	1.6165	-0.7016	1.0234	-0.5911
-1.0171	0.1872	0.3002	0.2194	-0.1107	-1.1168	-0.9981	-1.2533	-1.771	-0.4383
-0.0837	-1.2035	-1.8183	-1.1974	-1.9848	-0.7361	0.7764	-1.8143	-2.4063	-3.0379
-1.2366	-1.7665	-2.2777	-0.1955	-1.6133	-0.8561	-1.6656	0.592	0.6732	1.2875
0.3746	-2.1051	-0.6499	-0.2786	-1.1681	-2.3038	-0.7265	-0.4171	-0.1571	-0.4209
-2.8051	-0.3417	-0.7333	-2.6228	-2.6545	-0.7916	-2.092	-1.1276	1.1469	-1.3506
-0.8446	-0.6282	-0.9288	-1.839	-2.0139	0.3232	-0.8271	1.2948	-1.1434	-1.3287
0.228	1.8374	-0.7405	-1.6601	0.8237	0.7983	1.3889	-1.5207	-2.0826	0.686
-0.5588	-0.8706	-2.3069	-2.0911	-1.923	1.2686	-2.2256	-0.3076	0.2485	-1.2484
-0.5235	-2.9539	-1.0536	-2.1953	-0.7066	-0.6087	-2.4247	0.2568	-2.2132	0.3765
-1.8532	0.0716	-0.5872	-0.1795	0.0486	-1.1564	-1.3259	-1.5277	-1.1109	0.1741
-0.3926	-0.9021	-2.9507	-1.3225	-1.4424	-0.5541	-1.4745	0.6616	0.8837	-2.6601
-2.8266	-1.5187	0.363	-0.9947	0.0714	0.2922	-0.7713	-1.5418	0.6166	-2.9889
0.4297	-2.0705	-1.3904	-2.1019	-2.3529	-1.0709	-1.7198	0.6176	0.1304	-1.4186
-1.102	-0.2659	-0.7778	1.2523	-1.7577	-1.2658	-0.4072	-1.6394	-1.577	-0.981
0.0503	0.213	-1.0547	1.106	-1.2962	-1.0399	-0.3781	0.9283	-1.9113	0.9272
-1.4359	-2.5064	-0.0109	0.955	-1.1303	-1.0483	-2.0837	-2.2515	-1.4114	-2.4135

-0.6401	-0.1019	0.546	-2.8047	-2.1055	-0.9257	-0.9403	-1.3253	0.8394	0.4329

Упорядоченная выборка

-4.3891	-3.0379	-2.9889	-2.9539	-2.9507	-2.8266	-2.8051	-2.8047	-2.6601	-2.6545
-2.6228	-2.5064	-2.4247	-2.4135	-2.4063	-2.3993	-2.3529	-2.3069	-2.3038	-2.2783
-2.2777	-2.2515	-2.2256	-2.2132	-2.1953	-2.1055	-2.1051	-2.1019	-2.092	-2.0911
-2.0837	-2.0826	-2.0705	-2.0139	-1.9848	-1.923	-1.9113	-1.8532	-1.839	-1.8183
-1.8143	-1.771	-1.7665	-1.7577	-1.7389	-1.7198	-1.7189	-1.6656	-1.6601	-1.6394
-1.6133	-1.577	-1.5632	-1.5418	-1.5277	-1.5207	-1.5187	-1.4745	-1.4424	-1.4359
-1.4188	-1.4186	-1.4114	-1.3904	-1.3506	-1.3287	-1.3259	-1.3253	-1.3225	-1.2962
-1.2658	-1.2533	-1.2484	-1.2366	-1.2035	-1.1974	-1.195	-1.1681	-1.1564	-1.1434
-1.1303	-1.1276	-1.1168	-1.1109	-1.102	-1.0815	-1.0709	-1.0645	-1.0547	-1.0536
-1.0483	-1.0399	-1.0171	-0.9981	-0.9947	-0.981	-0.9611	-0.9521	-0.9403	-0.9288
-0.9257	-0.9021	-0.8706	-0.8561	-0.8446	-0.8271	-0.7916	-0.7778	-0.7713	-0.7503
-0.7405	-0.7361	-0.7333	-0.7265	-0.7066	-0.7016	-0.6499	-0.6401	-0.6282	-0.6087
-0.5911	-0.5872	-0.5631	-0.5588	-0.5541	-0.5313	-0.5235	-0.4684	-0.4383	-0.4256
-0.4209	-0.4171	-0.4072	-0.3926	-0.3781	-0.367	-0.3417	-0.3076	-0.2786	-0.2659
-0.1955	-0.1805	-0.1795	-0.1571	-0.1107	-0.1019	-0.0837	-0.0109	0.0486	0.0503
0.0714	0.0716	0.1304	0.1741	0.1872	0.213	0.2194	0.228	0.2485	0.2568
0.2922	0.3002	0.3232	0.363	0.3746	0.3765	0.4297	0.4329	0.5003	0.546
0.592	0.6166	0.6176	0.6616	0.6732	0.686	0.7764	0.7983	0.8237	0.8394
0.8837	0.9272	0.9283	0.955	1.0155	1.0234	1.106	1.1335	1.1469	1.2362
1.2523	1.2686	1.2875	1.2948	1.3889	1.6165	1.7393	1.7407	1.8374	1.9883

Группированная выборка (интервальный вариационный ряд)

(-0.4032, 0.394)	(-1.2004, -0.4032)	(-1.9976, -1.2004)	(-2.7947, -1.9976)	(-3.5919, -2.7947)	(-4.3891, -3.5919)	(0.394, 1.1911)	(1.1911, 1.9883)
33	58	41	26	7	1	23	11

Ассоциированный статистический ряд

-3.9905	-3.1933	-2.3961	-1.599	-0.8018	-0.0046	0.7925	1.5897
1.0	7.0	26.0	41.0	58.0	33.0	23.0	11.0
0.005	0.035	0.13	0.205	0.29	0.165	0.115	0.055

Проверка выполнения условия была выполнена в программе. $\sum_{i=1}^{n} w_i = 1$, где n – количество x_i

Гистограмма относительных частот

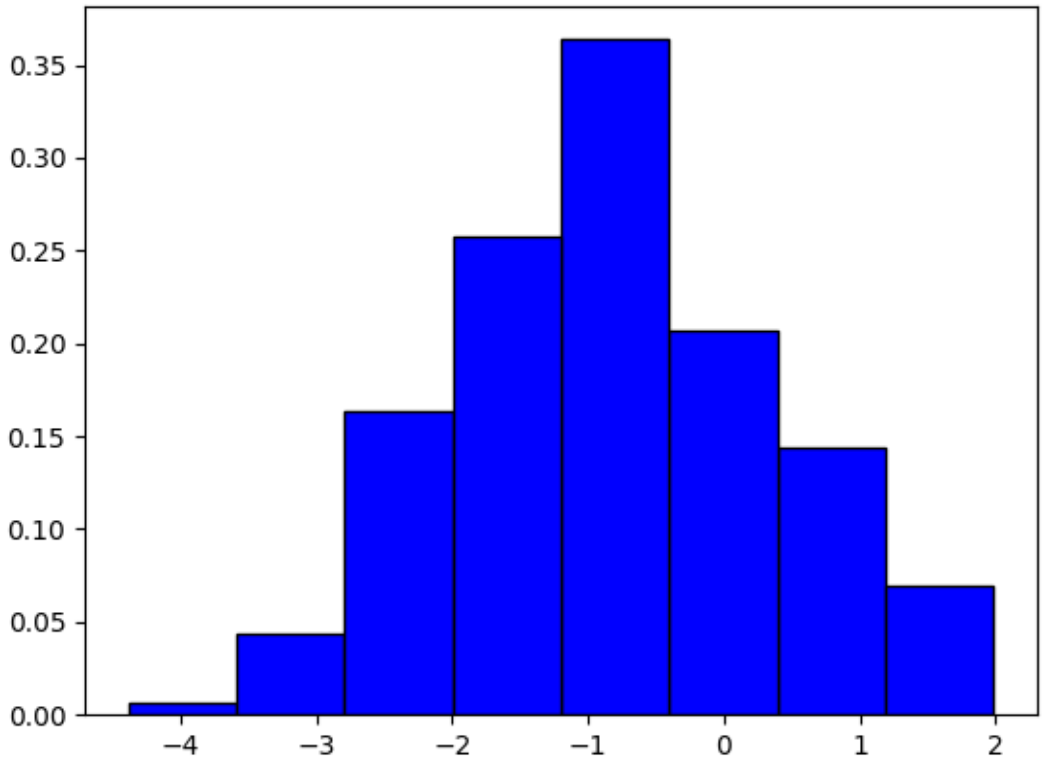
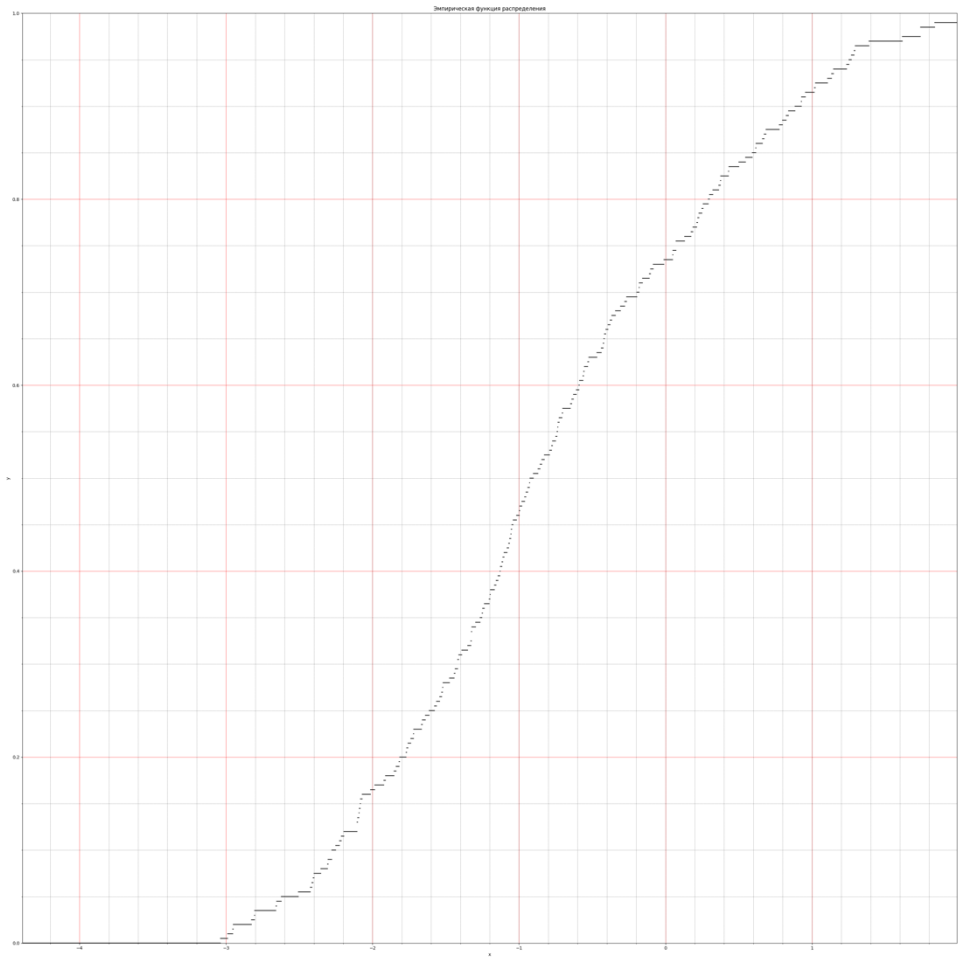


График эмпирической функции распределения



Результаты расчетов требуемых характеристик

выборочное среднее	-0.8257
выборочная дисперсия с поправкой Шеппарда	1.3699
выборочное среднее квадратическое отклонение	1.1704
выборочная мода	-0.8777
выборочная медиана	-1.5115
выборочный коэффициент асимметрии	0.0826
выборочный коэффициент эксцесса	-0.2287

Задание 2) Распределение по показательному закону

λ=1.91

Полученная выборка

0.7741	1.9751	4.9167	1.3741	0.9706	2.1148	0.5921	2.5107	2.8157	2.9328
2.3945	0.9645	3.0172	2.918	1.8904	0.5675	0.9178	1.9575	1.0501	1.1733
2.044	1.6977	1.278	1.0772	1.3496	1.4861	0.6034	1.582	1.839	1.2331
2.1209	0.8161	1.0494	0.8361	1.8763	2.2705	2.229	1.6048	1.6996	2.2702
0.7399	2.0142	1.0274	0.82	1.3193	1.8951	1.5966	0.5742	0.5531	1.9227
2.7368	0.6176	3.2029	2.2002	1.4283	0.859	8.1421	1.2411	0.561	1.0169
1.5876	1.4291	4.3861	0.7568	0.9749	0.5481	1.4123	2.4004	1.4312	0.674
2.1665	1.1486	2.4324	0.6684	1.0114	1.7163	0.9437	0.7869	2.0262	0.5782
1.5042	1.3446	2.3326	2.5733	0.9808	2.9792	1.6906	5.1585	1.1853	0.644
1.27	0.9064	2.1483	1.6265	0.6131	1.1702	1.2199	2.9828	3.5719	2.7807
1.4856	1.0648	0.5426	0.7964	1.0541	5.9394	0.7429	0.8103	1.8961	0.5261
1.6248	2.1596	2.875	2.9596	2.4873	1.0419	1.0823	0.6854	2.9896	2.3622
0.7942	1.2256	1.9481	2.3581	1.0373	1.3842	0.744	0.7335	2.0375	0.8538
0.9429	0.6857	3.1229	3.2772	1.8596	1.8382	1.1656	1.1736	0.5912	0.7183
0.5315	1.9388	0.6603	1.3231	1.5934	4.5081	0.8637	0.7801	0.5464	1.7318

1.6305	0.8879	0.9199	1.1802	1.8083	2.211	0.6183	0.6735	0.7343	1.2228
1.587	1.3489	1.1378	2.6968	0.6354	2.2001	1.3105	0.8747	1.1326	0.5946
0.6265	0.6823	1.1674	0.5753	2.6938	1.047	3.1257	2.0542	0.9743	1.317
0.6406	1.6532	1.1655	0.7416	0.6164	0.8356	0.8584	0.9209	0.9965	2.2912
0.7335	0.8182	1.2238	1.83	0.8629	6.2586	4.351	1.2147	1.2358	0.9151

Упорядоченная выборка

0.5261	0.5315	0.5426	0.5464	0.5481	0.5531	0.561	0.5675	0.5742	0.5753
0.5782	0.5912	0.5921	0.5946	0.6034	0.6131	0.6164	0.6176	0.6183	0.6265
0.6354	0.6406	0.644	0.6603	0.6684	0.6735	0.674	0.6823	0.6854	0.6857
0.7183	0.7335	0.7335	0.7343	0.7399	0.7416	0.7429	0.744	0.7568	0.7741
0.7801	0.7869	0.7942	0.7964	0.8103	0.8161	0.8182	0.82	0.8356	0.8361
0.8538	0.8584	0.859	0.8629	0.8637	0.8747	0.8879	0.9064	0.9151	0.9178
0.9199	0.9209	0.9429	0.9437	0.9645	0.9706	0.9743	0.9749	0.9808	0.9965
1.0114	1.0169	1.0274	1.0373	1.0419	1.047	1.0494	1.0501	1.0541	1.0648
1.0772	1.0823	1.1326	1.1378	1.1486	1.1655	1.1656	1.1674	1.1702	1.1733
1.1736	1.1802	1.1853	1.2147	1.2199	1.2228	1.2238	1.2256	1.2331	1.2358
1.2411	1.27	1.278	1.3105	1.317	1.3193	1.3231	1.3446	1.3489	1.3496
1.3741	1.3842	1.4123	1.4283	1.4291	1.4312	1.4856	1.4861	1.5042	1.582
1.587	1.5876	1.5934	1.5966	1.6048	1.6248	1.6265	1.6305	1.6532	1.6906
1.6977	1.6996	1.7163	1.7318	1.8083	1.83	1.8382	1.839	1.8596	1.8763
1.8904	1.8951	1.8961	1.9227	1.9388	1.9481	1.9575	1.9751	2.0142	2.0262
2.0375	2.044	2.0542	2.1148	2.1209	2.1483	2.1596	2.1665	2.2001	2.2002
2.211	2.229	2.2702	2.2705	2.2912	2.3326	2.3581	2.3622	2.3945	2.4004
2.4324	2.4873	2.5107	2.5733	2.6938	2.6968	2.7368	2.7807	2.8157	2.875
2.918	2.9328	2.9596	2.9792	2.9828	2.9896	3.0172	3.1229	3.1257	3.2029
3.2772	3.5719	4.351	4.3861	4.5081	4.9167	5.1585	5.9394	6.2586	8.1421

Группированная выборка (интервальный вариационный ряд)

(0.5261, 1.4781)	(1.4781, 2.4301)	(2.4301, 3.3821)	(3.3821, 4.3341)	(4.3341, 5.2861)	(5.2861, 6.2381)	(6.2381, 7.1901)	(7.1901, 8.1421)
116	54	21	1	5	1	1	1

Ассоциированный статистический ряд

1.0021	1.9541	2.9061	3.8581	4.8101	5.7621	6.7141	7.6661
116.0	54.0	21.0	1.0	5.0	1.0	1.0	1.0
0.58	0.27	0.105	0.005	0.025	0.005	0.005	0.005

Проверка выполнения условия была выполнена в программе.

$$\sum_{i=1}^{n} w_i = 1, \text{где } n - \text{количество } x_i$$

Гистограмма относительных частот

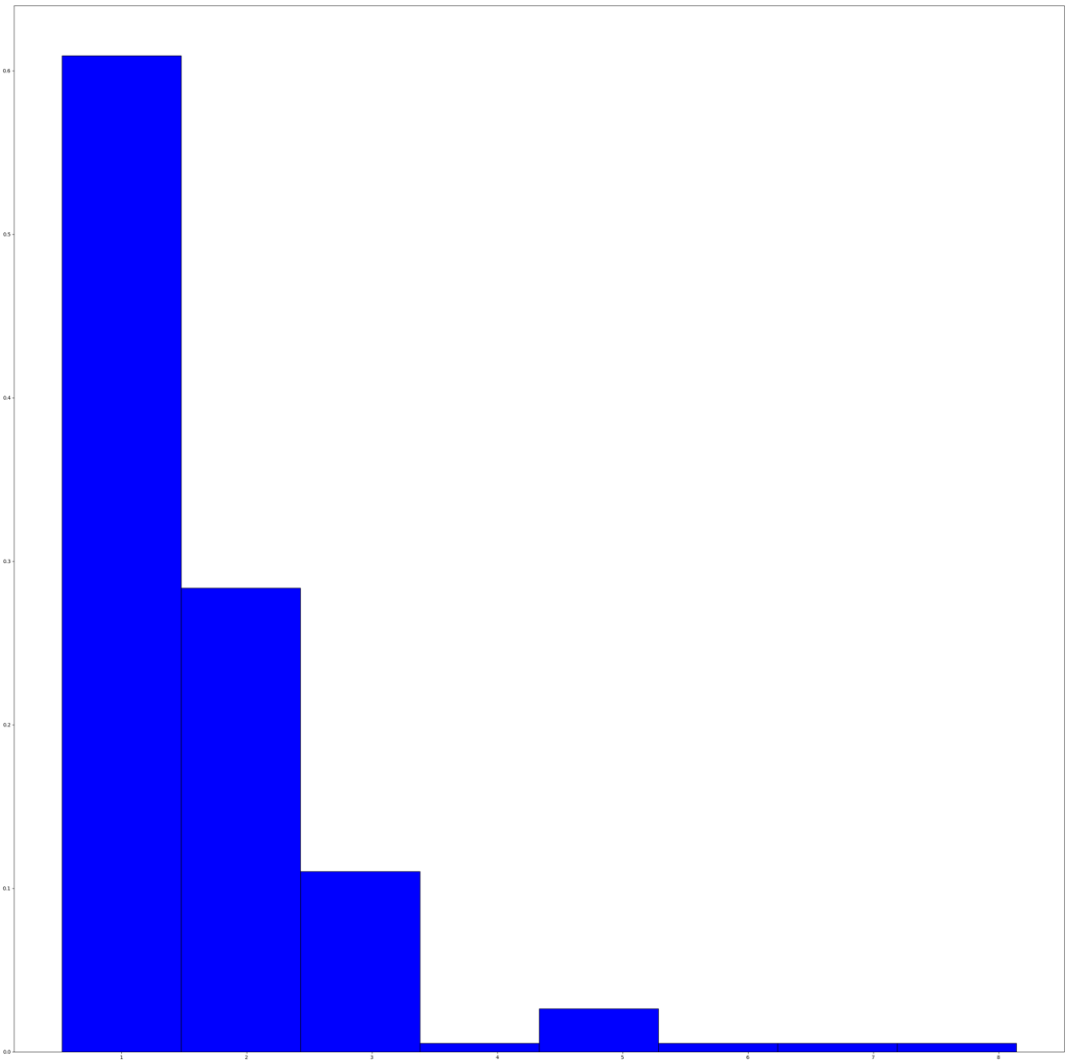
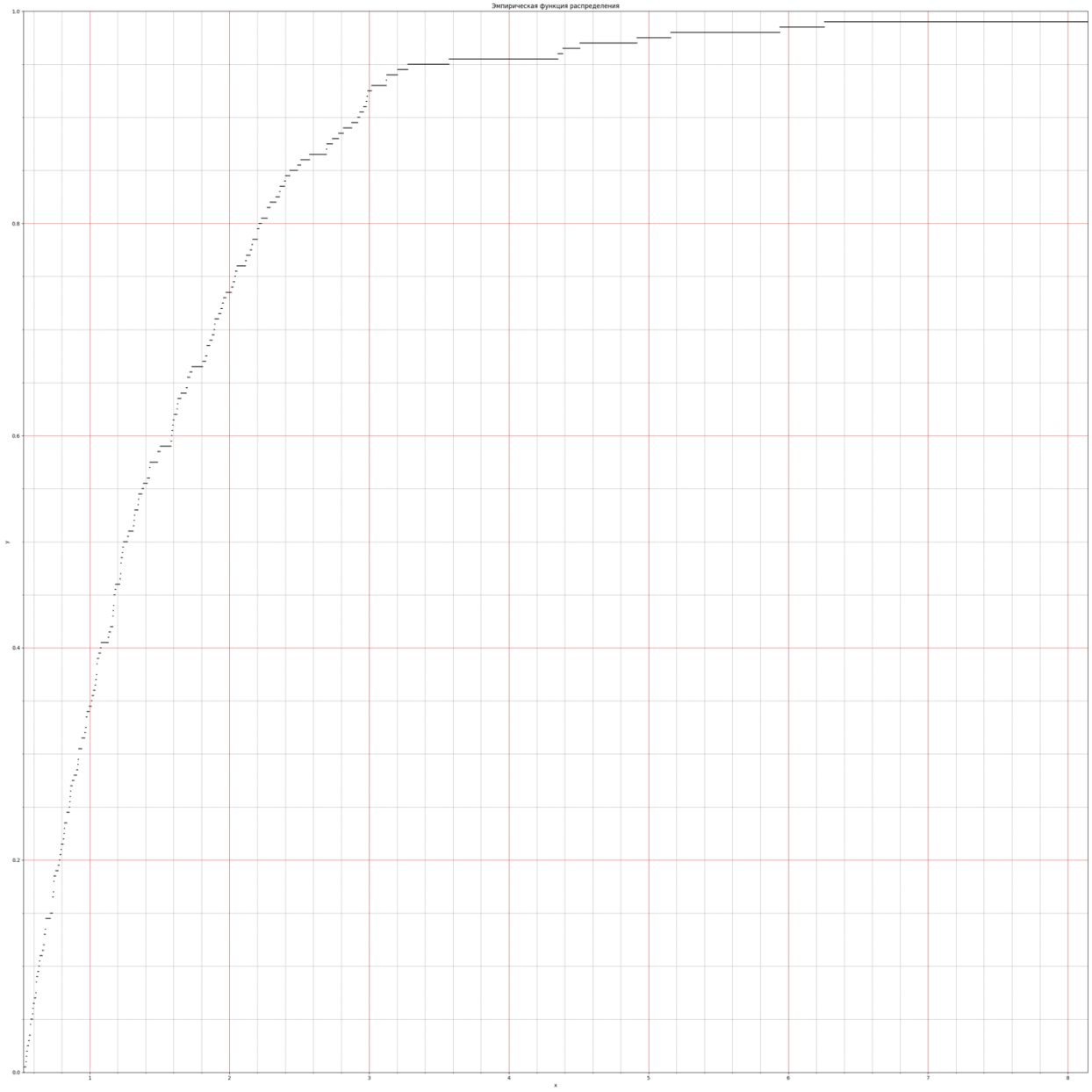


График эмпирической функции распределения


Эмпирическая функция распределения
1.0
0.8
0.6
0.4
0.2
0.0
F
1
2
3
4
5
6
7
8
x

Результаты расчетов требуемых характеристик

Характеристика	Значение
выборочное среднее	1.6542
выборочная дисперсия с поправкой Шеппарда	1.0264
выборочное среднее квадратическое отклонение	1.0131
выборочная мода	1.1446
выборочная медиана	3.9475
выборочный коэффициент асимметрии	2.8588
выборочный коэффициент эксцесса	10.4762

Задание 3) Равномерное распределение на отрезке [*a*,*b*]

a=-0.45 b=1.0

Полученная выборка

-0.0443	-0.0756	0.1481	0.3628	0.0138	0.2695	-0.0279	0.3181	0.2113	-0.2423
0.3969	-0.1734	0.1998	0.0137	0.1716	-0.4329	0.1224	0.1037	-0.4328	-0.2231
0.0487	-0.2308	-0.4489	0.0059	0.4408	-0.4171	0.1027	-0.1931	0.2473	0.1736
0.0053	-0.4153	0.094	0.3824	0.3223	-0.3722	-0.3708	-0.389	0.5352	-0.4055
0.1673	0.0434	0.3616	-0.1563	-0.2369	-0.2955	0.038	-0.4127	0.2587	0.2636
0.3621	-0.2458	-0.2664	-0.2621	0.5067	0.439	0.0264	-0.3482	0.1828	-0.4047
-0.4375	0.4455	0.1203	0.0187	0.533	-0.2601	-0.4043	-0.3666	0.4813	-0.0272
0.4913	0.0492	-0.3125	0.3688	0.0935	0.2288	0.2992	0.1822	0.5499	-0.1853
-0.055	-0.0973	0.103	-0.1189	0.3321	0.0929	0.3915	0.3578	0.1034	0.1558
0.5232	0.3628	-0.2449	0.1081	-0.0237	0.4218	-0.0849	0.1113	0.4159	0.361
-0.0979	-0.0989	0.0789	-0.4456	-0.449	-0.3226	0.0128	-0.4479	0.2159	-0.4117
0.5236	0.4401	0.3225	-0.0889	0.2193	-0.1456	0.3406	0.172	-0.18	0.0318
-0.1704	-0.2514	-0.1114	-0.2742	-0.3016	-0.1225	-0.4381	-0.337	0.4603	-0.4377
-0.3427	0.3125	0.1909	0.13	-0.2378	-0.3369	0.2214	0.4218	0.4986	-0.3308
-0.3913	-0.3906	0.4757	0.264	0.0337	-0.1548	0.5349	0.3609	0.2085	0.3793
0.1251	-0.0085	-0.2321	0.0807	0.2043	-0.3621	-0.2086	0.1217	0.5169	-0.3454
0.4438	-0.1462	0.4563	0.1027	-0.0861	-0.2864	-0.2445	0.0612	0.4984	0.0587
0.3021	0.4495	0.2683	-0.2639	-0.064	-0.105	0.1101	-0.2209	-0.2623	0.0858
0.113	-0.2686	-0.3795	0.5028	-0.2206	0.0479	-0.2506	-0.2154	-0.2407	0.0326
0.5066	-0.0335	-0.0783	0.1424	0.1442	0.0571	0.3647	0.2215	0.4816	-0.0493

Упорядоченная выборка

-0.449	-0.4489	-0.4479	-0.4456	-0.4381	-0.4377	-0.4375	-0.4329	-0.4328	-0.4171
-0.4153	-0.4127	-0.4117	-0.4055	-0.4047	-0.4043	-0.3913	-0.3906	-0.389	-0.3795
-0.3722	-0.3708	-0.3666	-0.3621	-0.3482	-0.3454	-0.3427	-0.337	-0.3369	-0.3308
-0.3226	-0.3125	-0.3016	-0.2955	-0.2864	-0.2742	-0.2686	-0.2664	-0.2639	-0.2623
-0.2621	-0.2601	-0.2514	-0.2506	-0.2458	-0.2449	-0.2445	-0.2423	-0.2407	-0.2378
-0.2369	-0.2321	-0.2308	-0.2231	-0.2209	-0.2206	-0.2154	-0.2086	-0.1931	-0.1853
-0.18	-0.1734	-0.1704	-0.1563	-0.1548	-0.1462	-0.1456	-0.1225	-0.1189	-0.1114
-0.105	-0.0989	-0.0979	-0.0973	-0.0889	-0.0861	-0.0849	-0.0783	-0.0756	-0.064
-0.055	-0.0493	-0.0443	-0.0335	-0.0279	-0.0272	-0.0237	-0.0085	0.0053	0.0059
0.0128	0.0137	0.0138	0.0187	0.0264	0.0318	0.0326	0.0337	0.038	0.0434
0.0479	0.0487	0.0492	0.0571	0.0587	0.0612	0.0789	0.0807	0.0858	0.0929
0.0935	0.094	0.1027	0.1027	0.103	0.1034	0.1037	0.1081	0.1101	0.1113
0.113	0.1203	0.1217	0.1224	0.1251	0.13	0.1424	0.1442	0.1481	0.1558
0.1673	0.1716	0.172	0.1736	0.1822	0.1828	0.1909	0.1998	0.2043	0.2085
0.2113	0.2159	0.2193	0.2214	0.2215	0.2288	0.2473	0.2587	0.2636	0.264
0.2683	0.2695	0.2992	0.3021	0.3125	0.3181	0.3223	0.3225	0.3321	0.3406
0.3578	0.3609	0.361	0.3616	0.3621	0.3628	0.3628	0.3647	0.3688	0.3793
0.3824	0.3915	0.3969	0.4159	0.4218	0.4218	0.439	0.4401	0.4408	0.4438
0.4455	0.4495	0.4563	0.4603	0.4757	0.4813	0.4816	0.4913	0.4984	0.4986
0.5028	0.5066	0.5067	0.5169	0.5232	0.5236	0.533	0.5349	0.5352	0.5499

Группированная выборка (интервальный вариационный ряд)

(-0.0744, 0.0504)	(-0.1993, -0.0744)	(-0.3242, -0.1993)	(-0.449, -0.3242)	(0.0504, 0.1753)	(0.1753, 0.3002)	(0.3002, 0.425)	(0.425, 0.5499)

24	21	28	30	31	19	23	24

Ассоциированный статистический ряд

-0.3866	-0.2617	-0.1369	-0.012	0.1129	0.2377	0.3626	0.4875
30.0	28.0	21.0	24.0	31.0	19.0	23.0	24.0
0.15	0.14	0.105	0.12	0.155	0.095	0.115	0.12

Проверка выполнения условия была выполнена в программе. $\sum_{i=1}^{n} w_i = 1$, где n – количество x_i

Гистограмма относительных частот

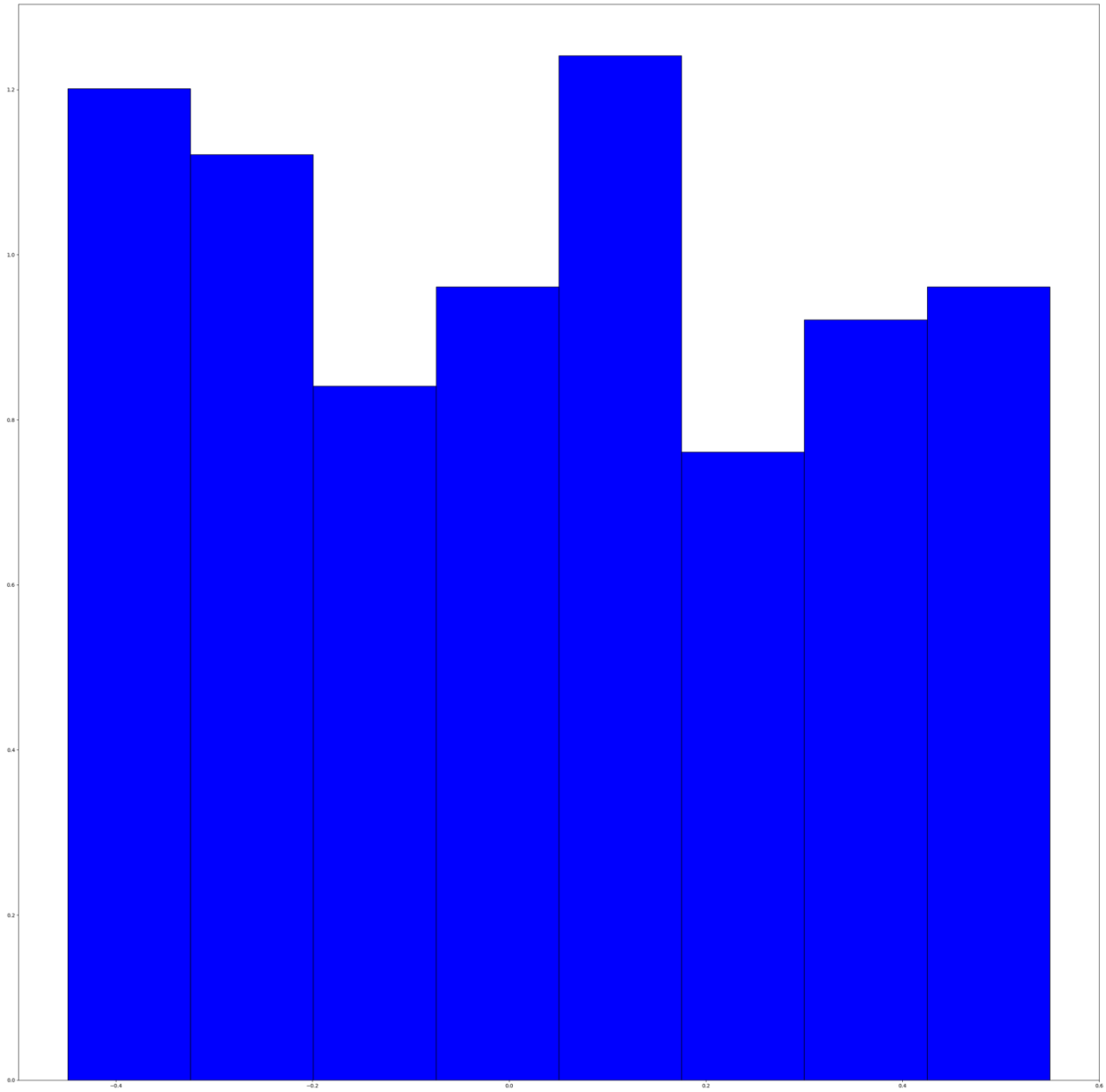
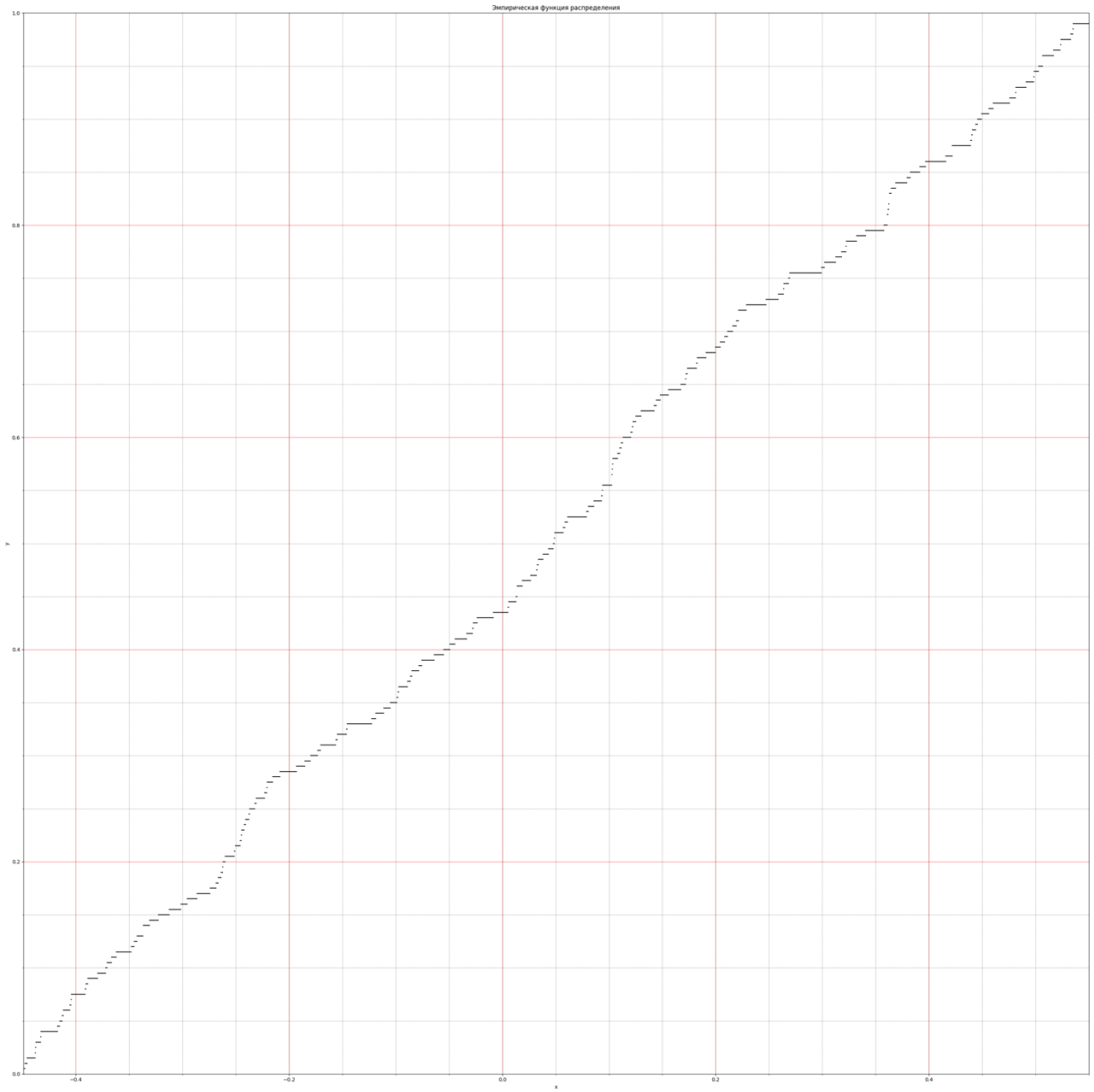


График эмпирической функции распределения



Результаты расчетов требуемых характеристик

Характеристика	Значение
выборочное среднее	0.0298
выборочная дисперсия с поправкой Шеппарда	0.0828
выборочное среднее квадратическое отклонение	0.2877
выборочная мода	0.0964
выборочная медиана	-0.0744
выборочный коэффициент асимметрии	0.0791
выборочный коэффициент эксцесса	-1.1815

Анализ результатов и выводы

1) Распределение по нормальному закону

Название показателя	Эксперимен-тальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	-1.1815	-1.2	0.0185	-0.0
Выборочная дисперсия с поправкой Шеппарда	0.0828	0.0833	0.0006	0.0
Выборочное среднее квадратичное отклонение	0.0964	0.275	0.1786	1.0
Выборочная мода	0.0791	0.0	0.0791	inf
Выборочная медиана	0.2877	0.2887	0.001	0.0

Интервал	w_i	p_i	$\lvert w_i - p_i \rvert$

Выборочный коэффициент асимметрии	0.0298	0.05	0.0202	0.0
Выборочный коэффициент эксцесса	-0.0744	0.05	0.1244	2.0

[-0.4032, 0.394]	0.005	0.1999	0.1949
(-1.2004, -0.4032]	0.035	0.2619	0.2269
(-1.9976, -1.2004]	0.13	0.2224	0.0924
(-2.7947, -1.9976]	0.205	0.1224	0.0826
(-3.5919, -2.7947]	0.29	0.0437	0.2463
(-4.3891, -3.5919]	0.165	0.0101	0.1549
(0.394, 1.1911]	0.115	0.0988	0.0162
(1.1911, 1.9883]	0.055	0.0317	0.0233
	$\sum_{i=1}^{n} w_i =$ 1	$\sum_{i=1}^{n} p_i =$ 0.9908	Δ_{max}=0.2463

2) Распределение по показательному закону

Название показателя	Эксперимен-тальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	-0.2287	0.0	0.2287	inf
Выборочная дисперсия с поправкой Шеппарда	1.3699	1.4116	0.0416	0.0
Выборочное среднее квадратичное отклонение	-0.8777	-0.9	0.0223	-0.0
Выборочная мода	0.0826	0.0	0.0826	inf
Выборочная медиана	1.1704	1.1881	0.0177	0.0
Выборочный коэффициент асимметрии	-0.8257	-0.9	0.0743	-0.0
Выборочный коэффициент эксцесса	-1.5115	-0.9	0.6115	-1.0

Интервал	w_i	p_i	$\lvert w_i - p_i \rvert$
[0.5261, 1.4781]	0.58	0.6125	0.0325
(1.4781, 2.4301]	0.27	0.2364	0.0336
(2.4301, 3.3821]	0.105	0.0912	0.0138
(3.3821, 4.3341]	0.005	0.0352	0.0302
(4.3341, 5.2861]	0.025	0.0136	0.0114
(5.2861, 6.2381]	0.005	0.0052	0.0002
(6.2381, 7.1901]	0.005	0.002	0.003
(7.1901, 8.1421]	0.005	0.0008	0.0042
	$\sum_{i=1}^{n} w_i =$ 1	$\sum_{i=1}^{n} p_i =$ 0.997	Δ_{max}=0.0336

3) Равномерное распределение на отрезке $[a,b]$

Название показателя	Эксперимен-тальное значение	Теоретическое значение	Абсолютное отклонение	Относительное отклонение
Выборочное среднее	-1.1815	-1.2	0.0185	-0.0
Выборочная дисперсия с поправкой Шеппарда	0.0828	0.0833	0.0006	0.0
Выборочное среднее квадратичное отклонение	0.0964	0.275	0.1786	1.0
Выборочная мода	0.0791	0.0	0.0791	inf
Выборочная медиана	0.2877	0.2887	0.001	0.0
Выборочный коэффициент асимметрии	0.0298	0.05	0.0202	0.0
Выборочный коэффициент эксцесса	-0.0744	0.05	0.1244	2.0

Интервал	w_i	p_i	$\lvert w_i - p_i \rvert$
[-0.0744, 0.0504]	0.15	0.1248	0.0252
(-0.1993, -0.0744]	0.14	0.1249	0.0151
(-0.3242, -0.1993]	0.105	0.1249	0.0199
(-0.449, -0.3242]	0.12	0.1248	0.0048
(0.0504, 0.1753]	0.155	0.1249	0.0301
(0.1753, 0.3002]	0.095	0.1249	0.0299
(0.3002, 0.425]	0.115	0.1248	0.0098
(0.425, 0.5499]	0.12	0.1249	0.0049
	$\sum_{i=1}^{n} w_i =$ 1	$\sum_{i=1}^{n} p_i =$ 0.9989	Δ_{max}=0.0301

Вывод: теоретические и экспериментальные в основном не сильно отличаются друг от друга, но были случаи, в которых достаточно большое относительное отклонение, но это из-за того, что взяли только 200 чисел.

Список использованной литературы

1. Лобузов А.А. Математическая статистика [Электронный ресурс]: Методические указания по выполнению лабораторных работ / под ред. Ю. И. Худака. Москва: Московский технологический университет (МИРЭА), 2017. 36 с.

2. Чернова Н. И. Математическая статистика: Учеб. пособие / Новосиб. гос. ун-т. Новосибирск, 2007. 148 с

Приложение (Листинг программы)

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

import sys
import argparse

from scipy import stats as st
import numpy as np
import pandas as pd

from copy import deepcopy
from collections import namedtuple
from collections import Counter
from functools import reduce
from copy import deepcopy
from random import choice
from itertools import islice
from math import log2
import string

import matplotlib.pyplot as plt
```

```
def findInterv(val: float, arr: list) -> int:
    test = lambda x, item: True if item[0] <= x <= item[1] else False
    for i, item in enumerate(arr):
        if test(val, item):
            return i
    return None

def expStats(sec_counter, first_counter, h):
    tmp = list(sec_counter.items())
    tmp.insert(0, 0.0)
    mean = reduce(lambda a, b: a + b[0] * b[1][1], tmp)
    s2 = reduce(lambda a, b: a + (b[0] - mean) ** 2 * b[1][1], tmp) - (h **
2) / 12
    ds = s2 ** 0.5

    ak, most_common = first_counter.most_common(1)[0]
    wn = sorted(first_counter.items(), key=lambda x: x[0])
    k = {val: i for i, (key, val) in enumerate(sorted(first_counter.items(),
key=lambda x: x[0]))}[most_common]
    wk = lambda k: sorted(sec_counter.items(), key=lambda x: x[0])[k][1][1]
    mode = ak[0] + h * ((wk(k) - wk(k-1)) / (2 * wk(k) - wk(k-1) - wk(k+1)))

    w = [val[1] for key, val in sorted(sec_counter.items(), key=lambda x: x)]
    sums = [reduce(lambda a, b: a + b, w[:i+1]) for i in range(len(w))]
    dsums = {el: i for i, el in enumerate(sums)}
    try:
        prk = [(dsums[l], dsums[r]) for l, r in list(zip(sums, islice(sums,
1, None))) if l <= 0.5 < r][0]
        a = [key for key, val in sorted(first_counter.items(), key=lambda x:
(x[0]))]
        med = a[prk[0]][1] if sums[prk[0]] == 0.5 else a[prk[0]][0] + (h /
w[prk[0]]) * (0.5 - sums[prk[0]])
    except:
        prk = (0, 1)
        a = [key for key, val in sorted(first_counter.items(), key=lambda x:
(x[0]))]
        med = a[prk[0]][1] if sums[prk[0]] == 0.5 else a[prk[0]][0] + (h /
w[prk[0]]) * (0.5 - sums[prk[0]])
        med = abs(med*10)

    mk = lambda k: sum(map(lambda item: item[0] ** k * item[1][1],
sec_counter.items()))
    mck = lambda k: sum(map(lambda item: (item[0] - mean) ** k * item[1][1],
sec_counter.items()))
    skew, kurtosis = mck(3) / (ds ** 3), (mck(4) / (ds ** 4)) - 3

    return mean, s2, ds, mode, med, skew, kurtosis

class LabFitter3000(object):
    '''IN: distribution name, params, scipy class
    OUT: path to figs, experimental stats, theoretical stats'''
```

```
    def __init__(self, obj, N, name, *params):
        self.distribution = obj
        self.N = N
        self.distribution_name = name
        self.params = params
        self.rvs = None
        self.h = None
        self.first_counter = None
        self.sec_counter = None
        self.pathes = []
        self.experimental_stats = None
        self.theoretical_stats = None
        self.Stats = namedtuple('Stats', 'mean variance std mode med skew
kurtosis')

    def create_rvs(self):
        self.rvs = list(self.distribution.rvs(size=self.N))

    def create_first_counter(self):
        m = int(1 + log2(self.N))
        sr = sorted(self.rvs)
        f, l = sr[0], sr[-1]
        d = abs(l - f)
        self.h = d / m
        interv = [f + self.h * i for i in range(m)]
        interv.append(l)
        interv = list(zip(interv, islice(interv, 1, None)))

        self.first_counter = Counter()
        for val in sr:
            self.first_counter[interv[findInterv(val, interv)]] += 1

    def create_second_counter(self):
        self.sec_counter = {(key[1] + key[0]) / 2: (val, val / self.N)
                            for key, val in self.first_counter.items()}

    def hist(self, show: bool = False):#, sec_counter: dict, h: float, show:
bool = False, path: str = 'Data/hist.png'):
        tmp = {key: val[1] / self.h for key, val in self.sec_counter.items()}
        plt.bar(list(tmp.keys()),
                list(tmp.values()),
                color='b',
                edgecolor='black',
                width=self.h)
        if show:
            plt.show()
        else:
            path = 'Data/' + self.distribution_name + '_hist.png'
            plt.savefig(path)
            plt.clf()
```

```
    def cdf(self, show: bool = False, sizex: int = 30, sizey: int = 30):#,
sr: list, N: int, sizex: int = 30, sizey: int = 30, show: bool = False, path:
str = 'Data/cdf.png'):

        sr = sorted(self.rvs)
        items = list(zip(sr, islice(sr, 1, None)))
        delta = 0.00001
        Xlist = [[x * delta for x in range(int(item[0] / delta), int(item[1]
/ delta))]
                 for item in items]
        Ylist = [[i / self.N] * len(Xlist[i]) for i in range(self.N-1)]

        fig, ax = plt.subplots(figsize=(sizex, sizey))

        for X, Y in zip(Xlist, Ylist):
            ax.plot(X, Y, label='', color='black')
        ax.set_title('Эмпирическая функция распределения')
        ax.legend(loc='upper left')
        ax.set_ylabel('y')
        ax.set_xlabel('x')
        ax.set_xlim(xmin=sr[0], xmax=sr[-1])
        ax.set_ylim(ymin=0, ymax=1)
        fig.tight_layout()
        if sizex != 5:
            ax.set_axisbelow(True)
            ax.minorticks_on()
            ax.grid(which='major', linestyle='-', linewidth='0.5',
color='red')
            ax.grid(which='minor', linestyle=':', linewidth='0.5',
color='black')
        else:
            ax.grid()
        if show:
            plt.show()
        else:
            path = 'Data/' + self.distribution_name + '_cdf.png'
            plt.savefig(path)
            plt.clf()

    def create_theoretical_stats(self, mode_generator):
        mean, variance, skew, kurtosis =
self.distribution.stats(moments='mvsk')
        med = self.distribution.median()
        std = self.distribution.std()
        mode = mode_generator(*self.params)

        self.theoretical_stats = self.Stats(mean=float(mean),
                                            variance=float(variance),
                                            std=std,
                                            mode=mode,
                                            med=med,
```

```
                                                skew=float(skew),
                                                kurtosis=float(kurtosis))

    def create_experimental_stats(self, exp_stats_generator):
        mean, variance, std, mode, med, skew, kurtosis =
exp_stats_generator(self.sec_counter,

self.first_counter,

self.h)
        self.experimental_stats = self.Stats(mean=mean,
                                                variance=variance,
                                                std=std,
                                                mode=mode,
                                                med=med,
                                                skew=skew,
                                                kurtosis=kurtosis)

    def save_rvs(self, shape: int = 20, dec = 4):
        rvs = np.array(self.rvs)
        rvs = rvs.reshape(shape, -1)
        rvs = np.round(rvs, dec)
        df_rvs = pd.DataFrame(rvs,

columns=list(string.ascii_lowercase)[:rvs.shape[1]])
        df_rvs.to_csv('Data/' + self.distribution_name + '_rvs.csv',
                      sep=';',
                      encoding='utf-8',
                      index=False)

        sorted_rvs = np.array(sorted(self.rvs))
        sorted_rvs = sorted_rvs.reshape(shape, -1)
        sorted_rvs = np.round(sorted_rvs, dec)
        df_sorted_rvs = pd.DataFrame(sorted_rvs,

columns=list(string.ascii_lowercase)[:sorted_rvs.shape[1]])
        df_sorted_rvs.to_csv('Data/' + self.distribution_name +
'_sorted_rvs.csv',
                             sep=';',
                             encoding='utf-8',
                             index=False)

    def save_dict(self):
        res = {str((round(key[0], 4), round(key[1], 4))): [val]
               for key, val in self.first_counter.items()}
        df = pd.DataFrame(res)
        df.to_csv('Data/' + self.distribution_name + '_first_counter.csv',
                  sep=';',
                  encoding='utf-8')
        res = {round(key, 4): val
               for key, val in self.sec_counter.items()}
```

```
        df = pd.DataFrame(res)
        df.to_csv('Data/' + self.distribution_name + '_second_counter.csv',
                  sep=';',
                  encoding='utf-8')

    def save_stats(self, dec = 4):
        experimental = self.experimental_stats._asdict()
        df = pd.DataFrame(list(np.round(list(experimental.values()), dec)),
                          index=experimental.keys(),
                          columns=[['values']])
        df.to_csv('Data/' + self.distribution_name +
'_experimental_stats.csv',
                  sep=';',
                  encoding='utf-8')

        theoretical = self.theoretical_stats._asdict()
        df = pd.DataFrame(list(np.round(list(theoretical.values()), dec)),
                          index=theoretical.keys(),
                          columns=[['values']])
        df.to_csv('Data/' + self.distribution_name +
'_theoretical_stats.csv',
                  sep=';',
                  encoding='utf-8')

        all_stats = list(zip(self.experimental_stats._asdict().keys(),
list(zip(self.experimental_stats, self.theoretical_stats))))
        result = {key: (round(e, dec), round(t, dec), round(abs(e-t), dec),
round(abs(e-t)/t, dec))
                  for key, (e, t) in all_stats}
        df = pd.DataFrame(list(result.values()),
                          index=list(result.keys()))
        df.to_csv('Data/' + self.distribution_name + '_all_stats.csv',
                  sep=';',
                  encoding='utf-8')

    def create_and_save_p(self, dec = 4):
        cur_r = self.distribution
        cur_x = sorted(list(self.first_counter.keys()))
        cur_w = [self.sec_counter[key][1]
                 for key in sorted(list(self.sec_counter.keys()))]


        p = [cur_r.cdf(b) - cur_r.cdf(a) for a, b in cur_x]
        s = sum(p)
        wp = [abs(a - b) for a, b in zip(cur_w, p)]
        m = max(wp)
        new_w = deepcopy(cur_w)
        new_w.append(1.0)
        p.append(s)
        wp.append(m)
        new_x = deepcopy(cur_x)
```

```
        new_x.append('-')

        df = pd.DataFrame(np.round(np.array([new_w, p, wp]).T, 4),
                          index=new_x)
        df.to_csv(self.distribution_name + '_wp' + '.csv',
                  sep=';',
                  encoding='utf-8')


def fitter(obj, mode_generator):
    obj.create_rvs()
    obj.create_first_counter()
    obj.create_second_counter()
    obj.hist()
    obj.cdf()
    obj.create_theoretical_stats(mode_generator)
    obj.create_experimental_stats(expStats)
    obj.save_rvs()
    obj.save_dict()
    obj.save_stats()
    obj.create_and_save_p()
    return obj


if __name__ == '__main__':
    variant = 9
    N = 200

    mode = {'norm': lambda a, sgm: a,
            'expon': lambda lmbd: 0,
            'uniform': lambda a, b: sum([a, b])/2}

    Obj = namedtuple('Obj', 'name params distribution')

    mu = (-1) ** variant * 0.1 * variant
    sgm = (0.01 * variant + 1) ** 2

    lmbd = 2 + (-1) ** variant * 0.01 * variant

    a = (-1) ** variant * 0.05 * variant
    b = a + 0.05 * variant + 1

    print('mu = {0}\nsgm = {1}\nlmbd = {2}\na = {3}\nb = {4}'.format(mu,
                                                                     sgm,
                                                                     lmbd,
                                                                     a,
                                                                     b))

    Data = [Obj(name='norm',
                params=[mu, sgm],
                distribution=st.norm(mu, sgm)),
```

```
            Obj(name='expon',
                params=[1/lmbd],
                distribution=st.expon(1/lmbd)),
            Obj(name='uniform',
                params=[a, b],
                distribution=st.uniform(a, b))]

    result = []
    for obj in Data:
        result.append(LabFitter3000(obj.distribution, N, obj.name,
*obj.params))

    result = [fitter(lf, mode[Data[i].name]) for i, lf in enumerate(result)]
```