Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по расчётному заданию № 2

Дисциплина: Теория вероятностей и математическая статистика

Тема: Статистическая обработка случайных последовательностей

Выполнил студент гр. 5130901/20003 _____________ А.А Вагнер

(подпись)

Преподаватель __________ К.В. Никитин

(подпись)

“___” _____________ 2024 г.

Санкт-Петербург

2024

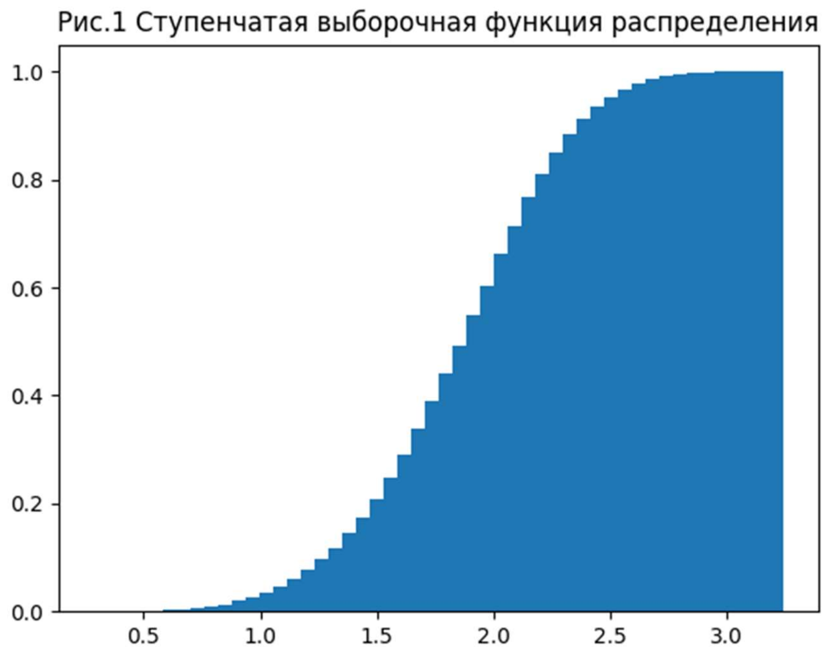
Задание

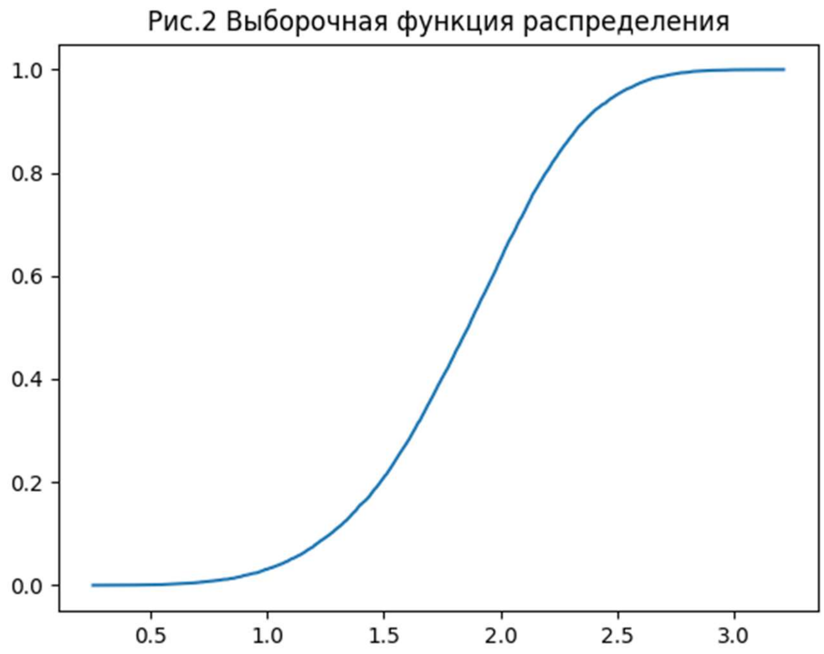
1. **Статистическая обработка случайных последовательностей**
   1.1. Считать выборку X из файла. Создать на ее основе 10 подвыборок – для этого перемешать выборку
   1.2. Представить визуально оценки функции плотности распределения:
      1.2.1. Построить выборочную функцию распределения F(x)
      1.2.2. Построить абсолютную и относительную гистограммы на разных графиках
      1.2.3. Построить оценки плотности с применение ядерного оценивания (kernel density estimation).Рассмотреть 3-4 разных вариантов ядра и для каждого из них выбрать оптимальное значение ширины окна *h*. В качестве начальной оценки использовать одну из параметрических оценок *h*
   1.3. Определить точечные оценки:
      1.3.1. первого начального момента
      1.3.2. центральных моментов: второго - дисперсии, третьего, четвертого по выборочной функции распределения
      1.3.3. асимметрии и эксцесса
      1.3.4. границ интерквантильного промежутка J_p для P = 0.95 только по полной выборке
      1.3.5. представить результаты графически
   1.4. Определить интервальные оценки с доверительной вероятностью Q = 0.8:
      1.4.1. первого начального и второго центрального моментов
      1.4.2. интерквантильного промежутка J для P=0.95:
         1.4.2.1. по всей выборке с помощью непараметрических толерантных пределов, симметричных относительно среднего арифметического и относительно нуля:
         1.4.2.2. по частичным выборкам с помощью параметрических толерантных пределов, считая закон распределения генеральной совокупности нормальным
         1.4.2.3. Результаты представить графически
   1.5. Сделать выводы относительно ширины доверительных интервалов. Сравнить:
      1.5.1. интерквантильные промежутки с толерантными пределами
      1.5.2. параметрические и непараметрические толерантные пределы, симметричные относительно среднего арифметического и относительно нуля
2. **Идентификация закона и параметров распределения**
   2.1. Начальный выбор распределения
   2.2. Определение параметров теоретических распределений. Для выбранных теоретических распределений необходимо определить точные значения параметров, наиболее подходящие для описания выборки. Это необходимо сделать двумя способами:
      2.2.1. С помощью метода моментов
      2.2.2. С помощью метода максимального правдоподобия
      2.2.3. Сравнить оценки, полученные методом моментов и ММП
   2.3. Произвести проверку гипотез относительно выбранных теоретических законов распределения и их параметров. Проверку провести по трем критериям - "хи-квадрат", Колмогорова - Смирнова, "омега-квадрат".
   2.4. Привести итоговую таблицу с оценками.

Ход работы

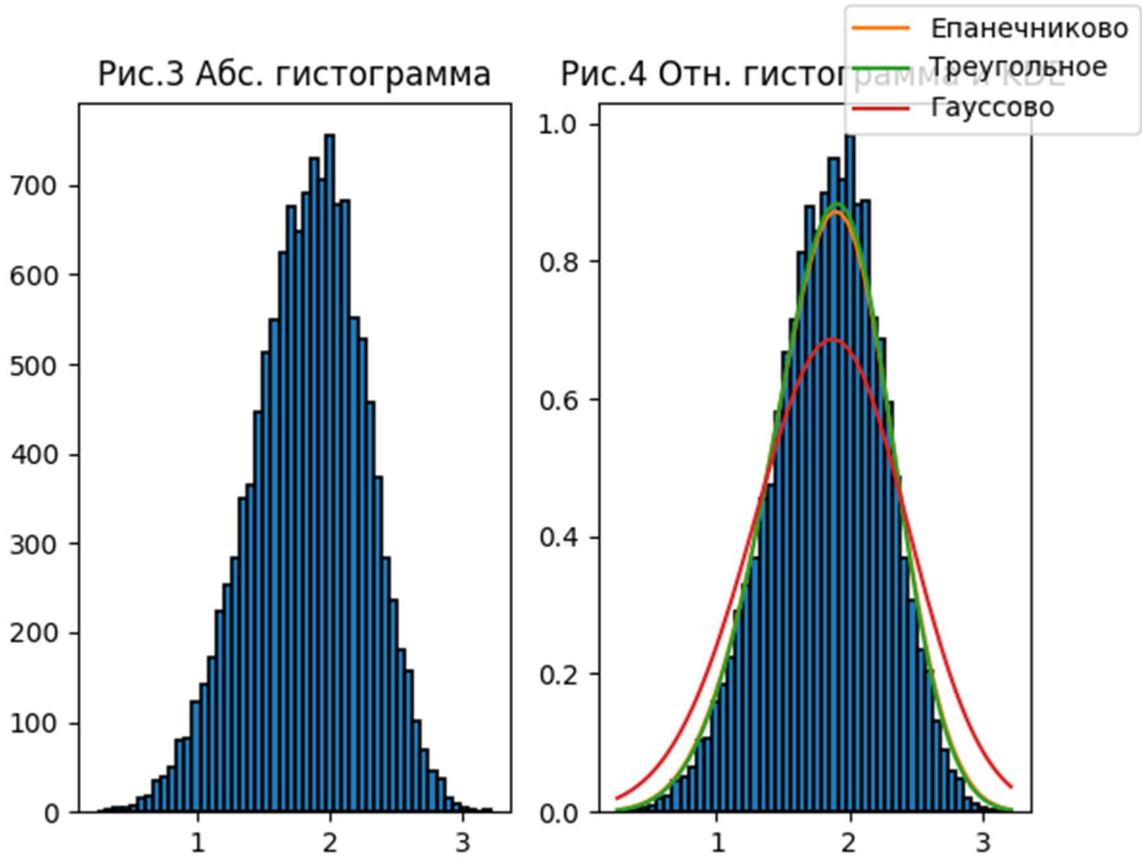
Так как в моём, полученном методами альтернативными легальным, пакете Matlab отсутствует Statistic Toolbox, все последующие вычисления будут выполнены на языке программирования Python.

Для начала считаем данные из файла и построим 10 выборок. Отсортируем все выборки, включая изначальную и построим выборочную функцию распределения.





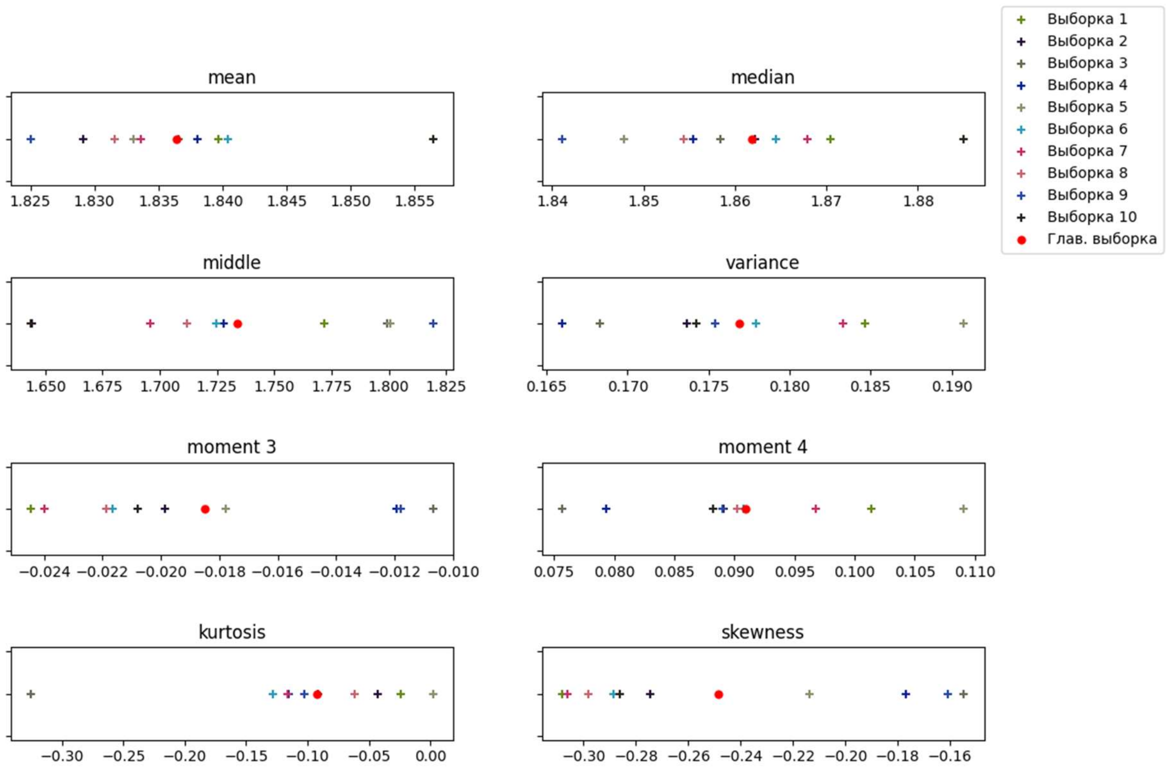
Далее построим абсолютные и относительные гистограммы оценки плотности, также применим kde алгоритмы следующих типов: Епанечниково, треугольное и Гауссово. Очевидно, что все оценки ядер, кроме последнего, крайне похожи.



Определим следующие точечные оценки по сформированным подвыборкам и изначальной выборке: mean, median, middle, variance, moment 3, 4, skewness, kurtosis; представим их в виде таблицы, а также графически – точками на осях.

	$\bar{x}$	x_{med}	$x_{\text{ср}}$	$varince$	m_3	m_4	A_s	E_x
N	1.8289	1.8708	1.7786	0.1898	-0.0179	0.1066	-0.0369	-0.2162
1. N/10	1.8271	1.8370	1.6949	0.1823	-0.0153	0.0942	-0.1610	-0.1965
2. N/10	1.8298	1.8530	1.7591	0.1758	-0.0180	0.0897	-0.0921	-0.2446
3. N/10	1.8317	1.8551	1.7588	0.1706	-0.0227	0.0836	-0.1243	-0.3228
4. N/10	1.8563	1.8980	1.7032	0.1711	-0.0226	0.08626	-0.0482	-0.3200
5. N/10	1.8466	1.8647	1.7338	0.1779	-0.0154	0.0907	-0.1270	-0.2057
6. N/10	1.8411	1.8620	1.6573	0.1676	-0.0149	0.0799	-0.1507	-0.2170
7. N/10	1.8462	1.8726	1.6630	0.1810	-0.0221	0.0975	-0.0171	-0.2870
8. N/10	1.8269	1.8399	1.7079	0.1720	-0.0118	0.0842	-0.1490	-0.1662
9. N/10	1.8289	1.8558	1.7717	0.1812	-0.0236	0.0962	-0.0632	-0.3067
10. N/10	1.8363	1.8618	1.7337	0.1769	-0.0184	0.0909	-0.0927	-0.2486

Рис.5 Точечные оценки.



Посчитаем интервальные оценки мат. ожидания и дисперсии с доверительной вероятностью Q=0.8 и представим в виде таблицы и графика.

	$\bar{x}, l$	$\bar{x}, r$	var, l	var, r
N	1.8316	1.8411	0.1741	0.1798
1. N/10	1.8244	1.8549	0.1757	0.1944
2. N/10	1.8143	1.8440	0.1653	0.1828
3. N/10	1.8212	1.8511	0.1602	0.1772
4. N/10	1.8236	1.8526	0.1579	0.1746
5. N/10	1.8175	1.8486	0.1816	0.2008
6. N/10	1.8254	1.8554	0.1693	0.1872
7. N/10	1.8184	1.8488	0.1745	0.1929
8. N/10	1.8167	1.8464	0.1670	0.1846
9. N/10	1.8101	1.8399	0.1669	0.1846
10. N/10	1.8416	1.8712	0.1658	0.1834

Рис.6 Интервальные оценки мат. ожидания

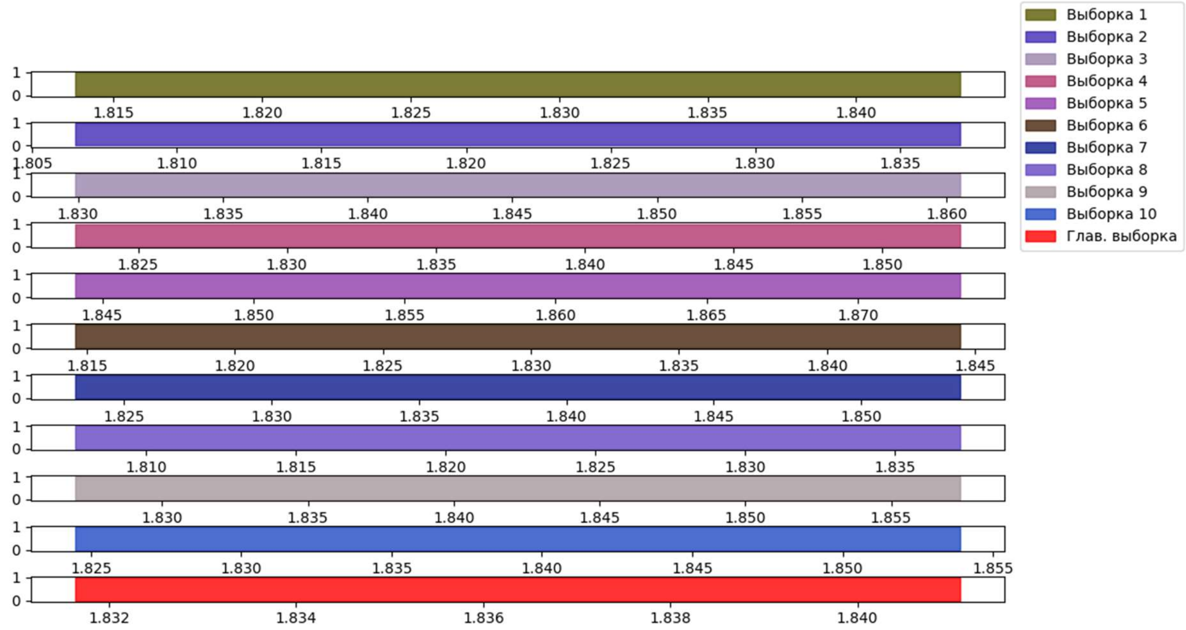
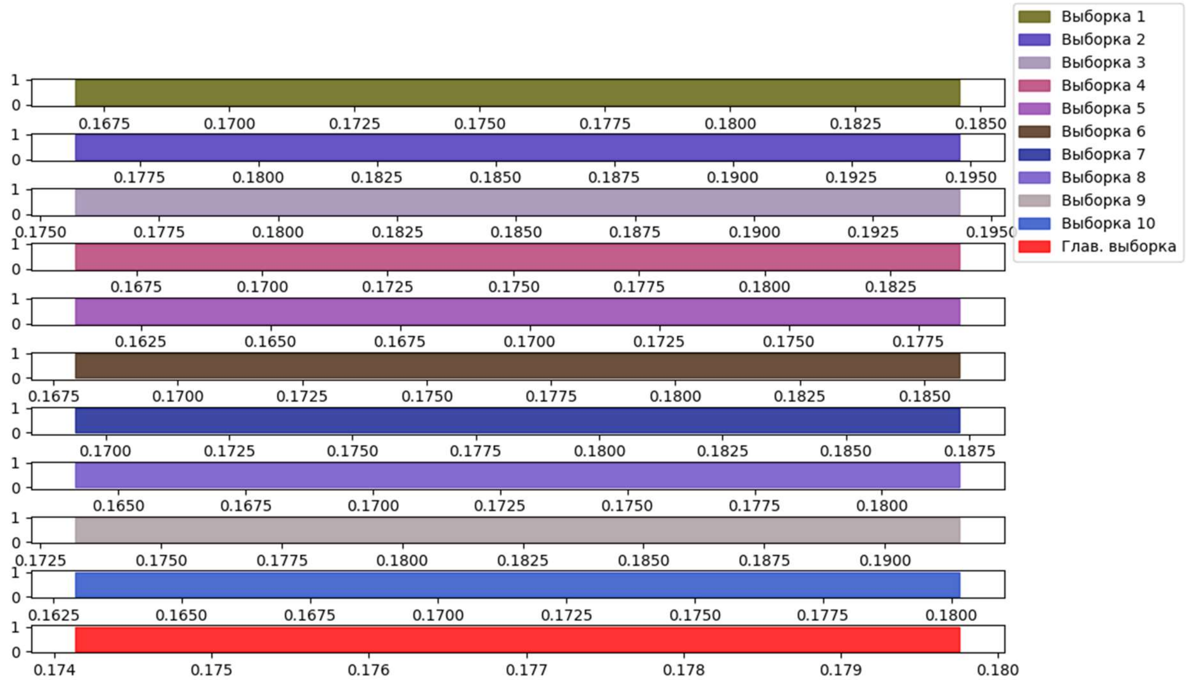


Рис.7 Интервальные оценки дисперсии



Посчитаем интерквантильный промежуток J для P=0.95, Q=0.8:
При помощи непараметрических толерантных пределов, симметричных относительно среднего арифметического. Количество статистически эквивалентных блоков k отбрасываемых от выборки при нахождении непараметрических толерантных пределов, симметричных относительно среднего арифметического определяется из неравенства: $\sum_{m=n-k}^{n} C_n^m P^m (1-P)^{n-m} \le 1-Q$
Результирующий предел будет равен $[X_{\frac{k}{2}}, X_{N-\frac{k}{2}}]$ при чётном k или $[X_{\frac{k-1}{2}}, X_{N-\frac{k-1}{2}}]$ при нечётном k. В случае если пределы симметричны

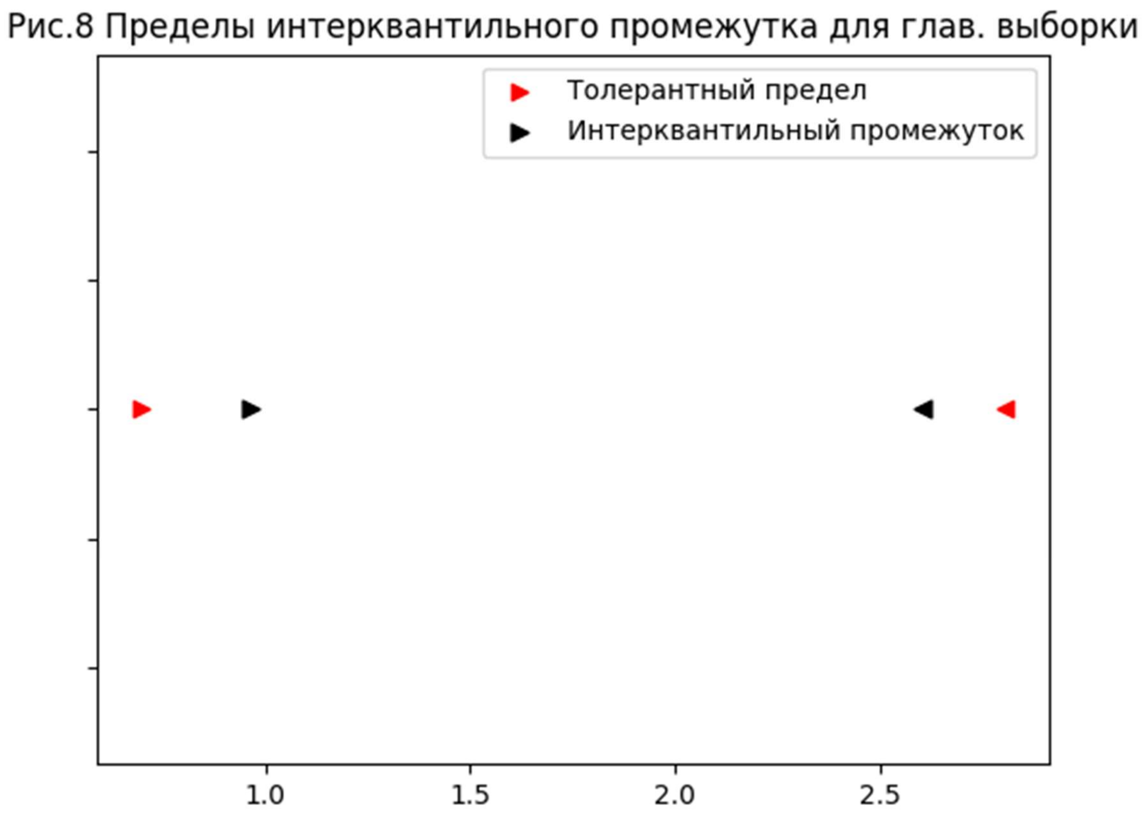
относительно нуля, то необходимо преобразовать выборку, заменив отрицательные значения на их модуль и отбросить справа эквивалентных блоков. Результирующий предел будет равен $[-X_{N-k+1}, X_{N-k+1}]$
Полученное k = 127. Очевидно, k – нечётное число, следовательно используем формулу $[X_{\frac{k-1}{2}}, X_{N-\frac{k-1}{2}}]$.

$[X_{63}, X_{12937}] = [0.694123, 2.80923]$;

Квантиль посчитаем при помощи функции quantile:

```
X = dlmread('Task_2a.txt');
    QL = quantile(X, (1 - 0.95 + 1) /2)
    QR = quantile(X, (0.95 + 1) /2)
```

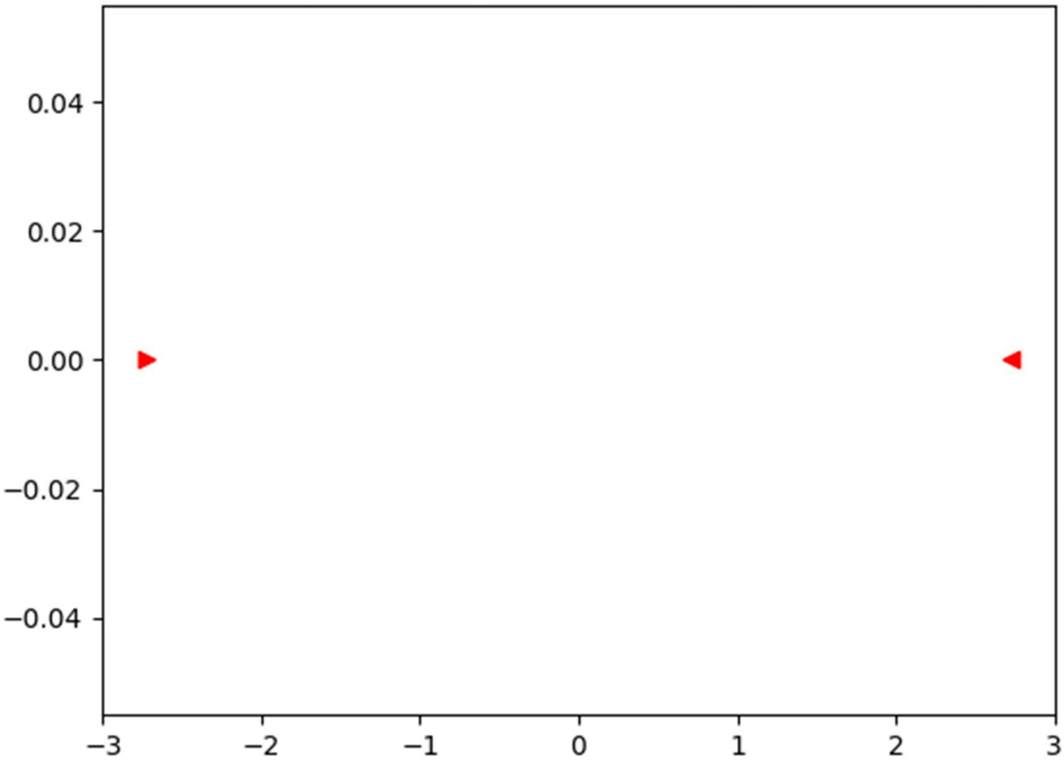
Представим полученные результаты графически:



Определим толерантные пределы симметричные относительно нуля:

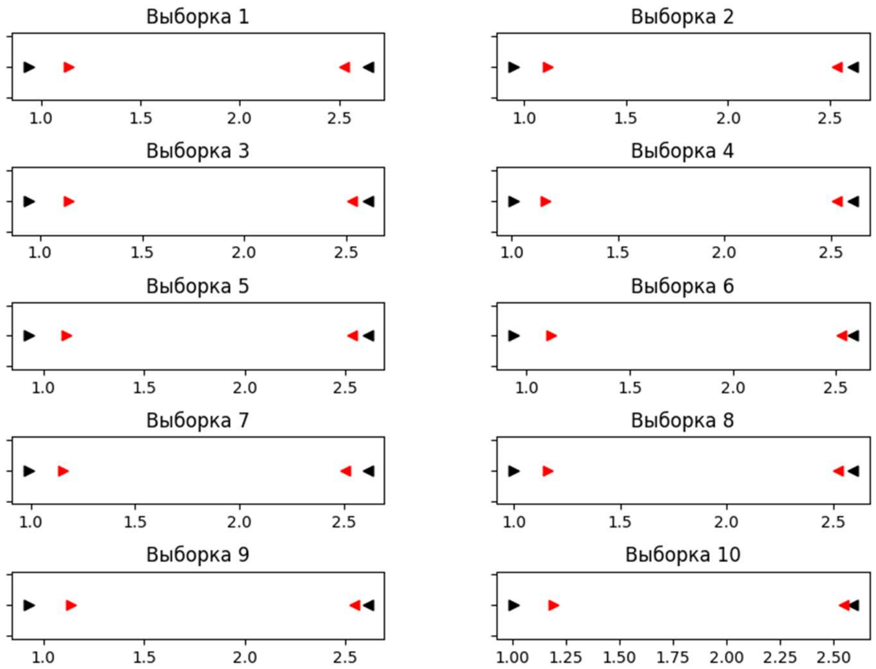
$$[-X_{N-k+1}, X_{N-k+1}] = [-X_{12872}, X_{12872}] = [-2.73271, 2.73271]$$

Рис.9 Толерантные пределы симметричные отн. нуля



Посчитаем интерквантильные промежутки по частичным выборкам с помощью параметрических толерантных пределов, считая закон распределения генеральной совокупности нормальным. Легенда графиков та же, что и у рис.8.

Рис.10 Пределы интерквантильных промежутков подвыборок



Определим закон и параметры распределения. Учитывая форму гистограммы мною были выбораны функции распределения: нормальное, Коши, Стьюдента.
Определим параметры для данных функций.
Метод моментов:
Нормальное - $a = \bar{x} = 1.8390, \sigma = \sqrt{var} = 0.4234$
Коши - $a = x_{med} = 1.8452, \gamma = \frac{IQR}{2} = 0.2848$
Стьюдента - $\mu = \bar{x} = 1.8390, \sigma = \sqrt{var} = 0.4234$
Метод максимального правдоподобия:
Нормальное - $a = 1.8364, \sigma = 0.4206$
Коши - $a = 1.8689, \gamma = 0.2583$
Стьюдента - $\mu = 1.8364, \sigma = 0.4206$

Представим результаты графически:

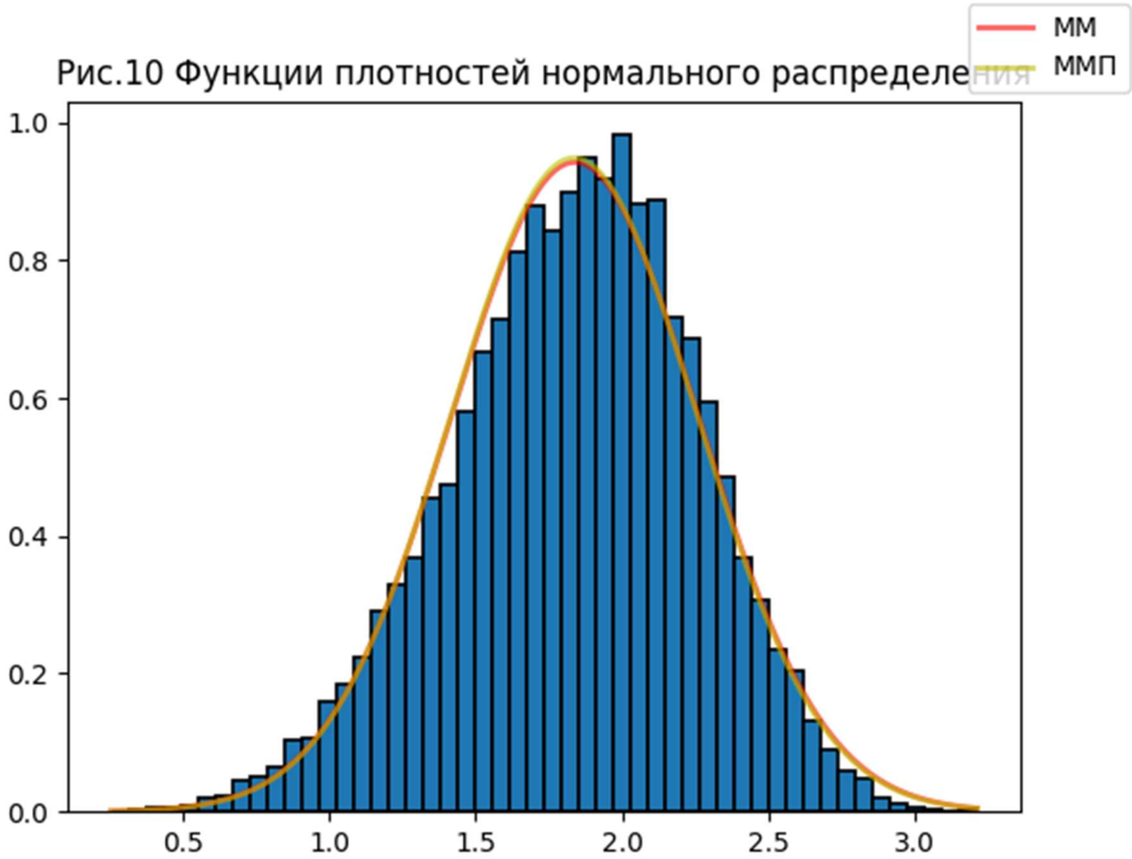


Рис.10 Функции плотностей нормального распределения

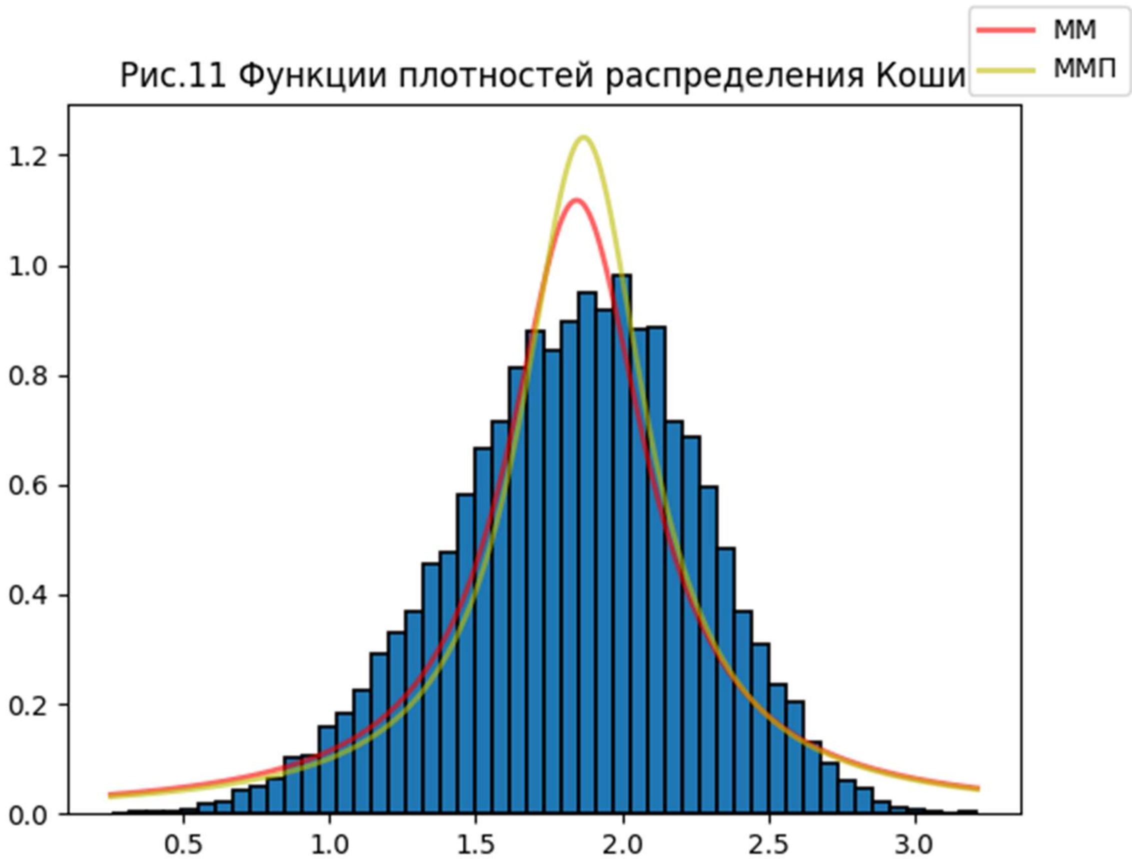
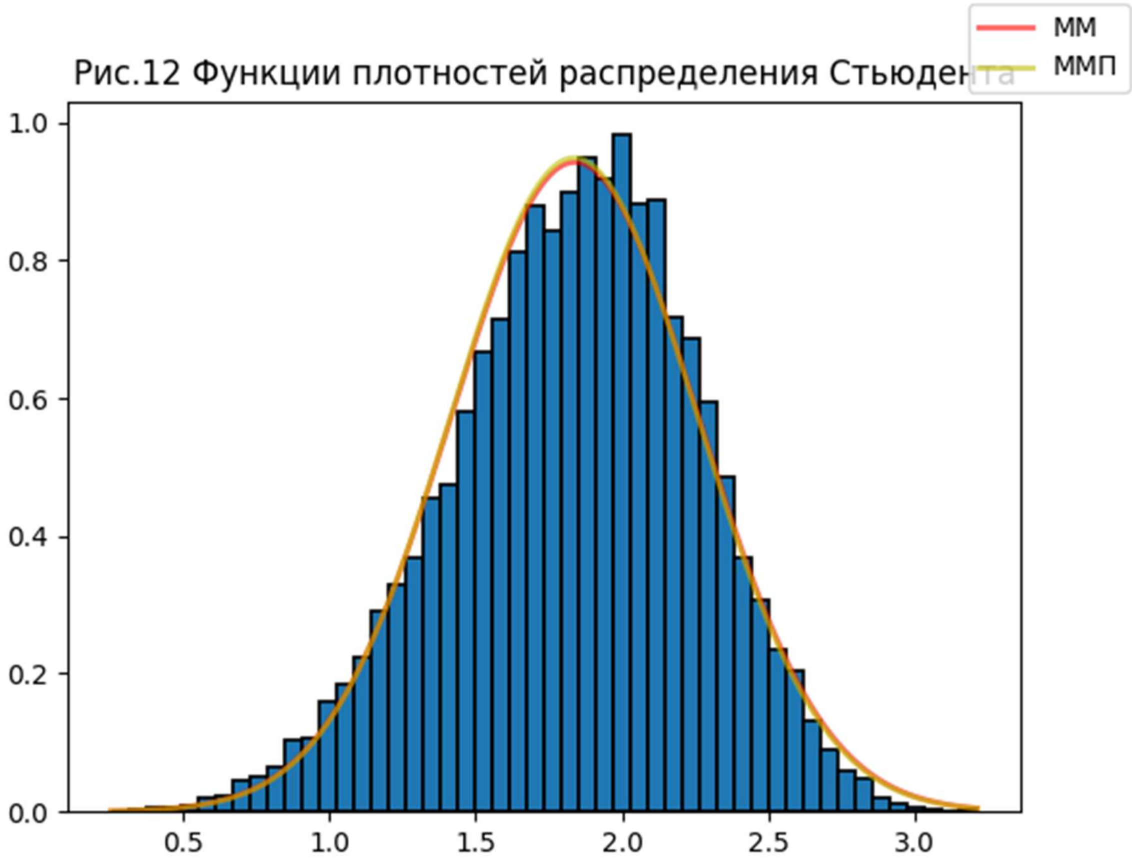
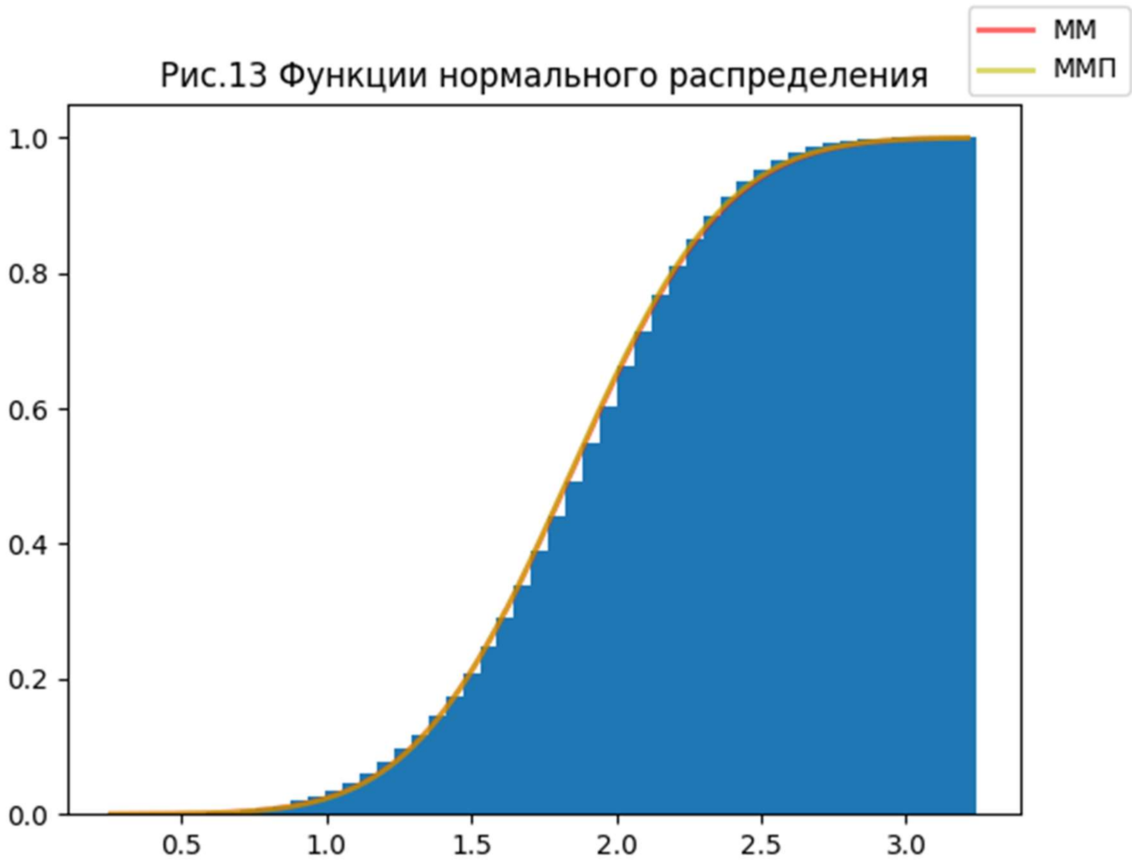
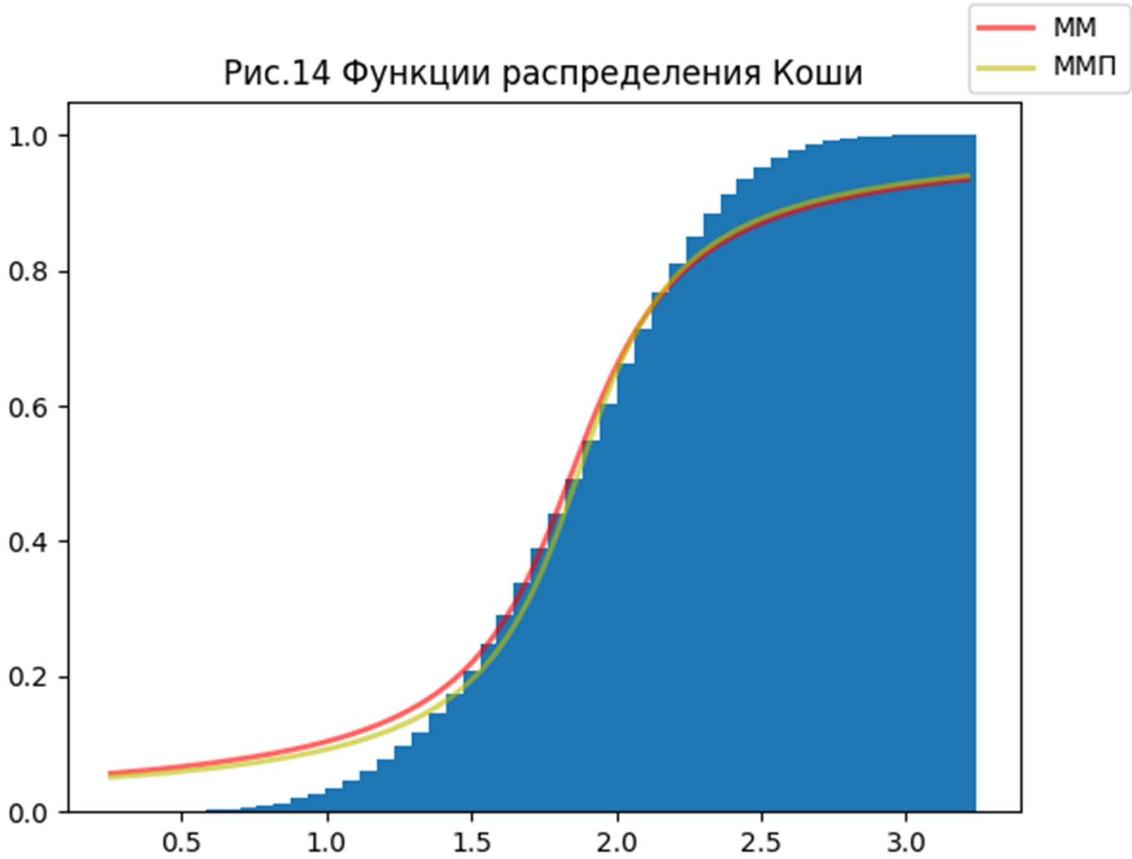
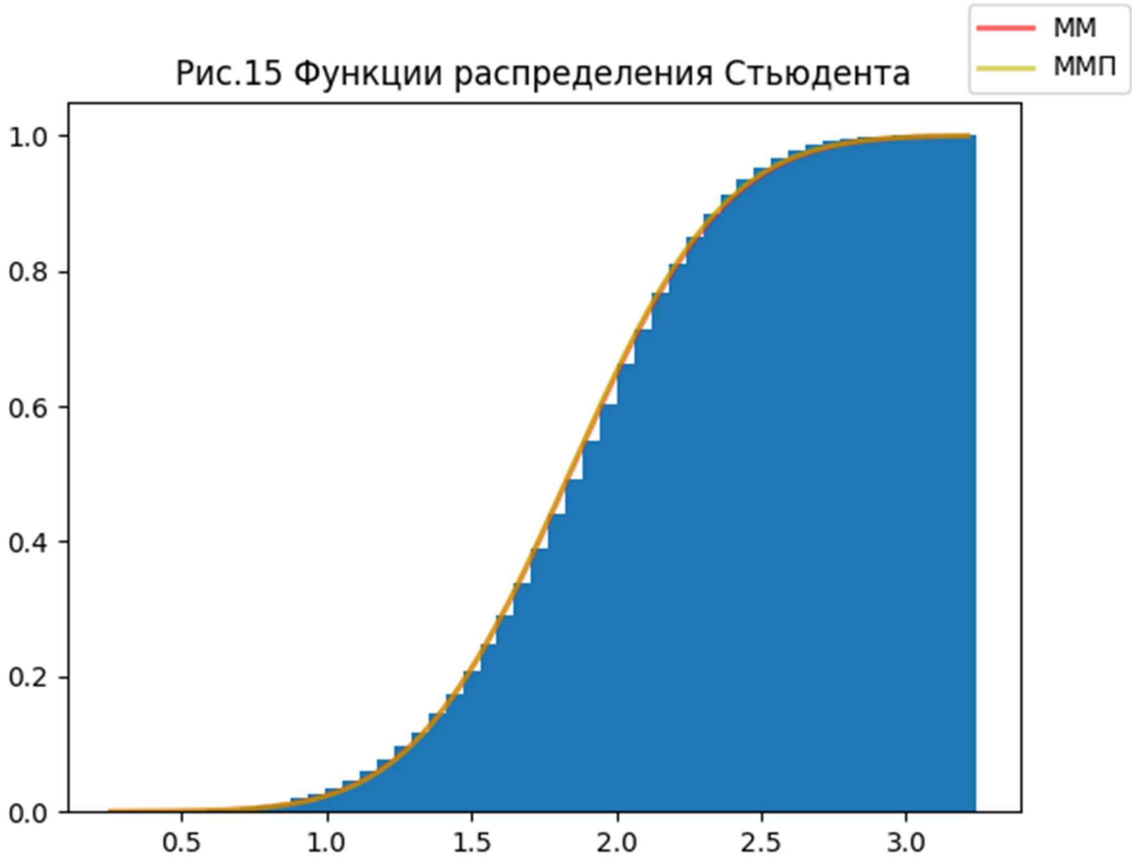


Рис.11 Функции плотностей распределения Коши


MM
ММП
Рис.12 Функции плотностей распределения Стьюдента
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0


MM
ММП
Рис.13 Функции нормального распределения
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0





На основе полученных графиков можно сделать вывод, что данная выборка описывается либо законом нормального распределения, либо законом распределения Стьюдента. Также следует отметить, что метод моментов показал большую точность, чем ММП.

Произведём проверку гипотез относительно выбранных теоретических законов распределения и их параметров. Проверку проведём по трем критериям - "хи-квадрат", Колмогорова - Смирнова, "омега-квадрат". Альфа возьмём как 0.05.

Название	Нормальное		Коши		Стьюдента	
Формула плотности	$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-a)^2}{2\sigma^2}\right)$		$\frac{\Delta}{\pi(\Delta^2+(x-c)^2}$		$\frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{\pi n}\Gamma\left(\frac{n}{2}\right)}\left(1+\frac{y^2}{n}\right)^{\frac{n+1}{2}}$	
	ММ	ММП	ММ	ММП	ММ	ММП
Хи-квадрат статистика	252.682205	257.393945	1579.06402	1852.86967	252.682203	257.393943
Хи-квадрат критич. знач.	178.870612					
Хи-Квадрат вывод	Ни одно из распределений не удовлетворяет					
Колм. - Смирнова статистика	0.02211660	0.02473514	0.09116960	0.08457930	0.02211660	0.02473514
Колм. - Смирнова критич. знач.	0.0201746					
Колм.- Смирнова вывод	Ни одно из распределений не удовлетворяет					
Мизеса статистика	1.88160970	2.35601497	19.03415323	17.15292637	1.88160870	2.35601496
Мизеса критич. знач.	0.4614					
Мизеса вывод	Ни одно из распределений не удовлетворяет					

Следует проверить другие распределения. Для этого напишем скрипт, который применяет метод максимального правдоподобия для всех распределений из следующего списка: арксинус, треугольное, Коши, Симпсона, Лапласа, Хи-квадрат, экспоненциальное, нормальное, равномерное, Симпсона, Стьюдента, логнормальное, гамма, Рэлея, Парето; и определяет, для какого из них статистика критерия Мизеса была наименьшей. Код этого скрипта также будет включён в листинг.
По результатам работы программы ближайшими були распределения Стьюдента и нормальное, в зависимости от того, как была перемешана выборка. В большинстве случаев и для отсортированной выборки нормальное распределение оказывалось точнее.

Также следует отметить гамма распределение, которое занимало третье по точности место. Критерий Мизеса для неё составил 4.3726, Колмагорова-Смирнова - 0.032625, Хи-квадрат - 387.8556 при аргументах 359.28, -6.2049, 0.02237 для k, θ и a соответственно.

Рис. 16 Функция гамма распределения

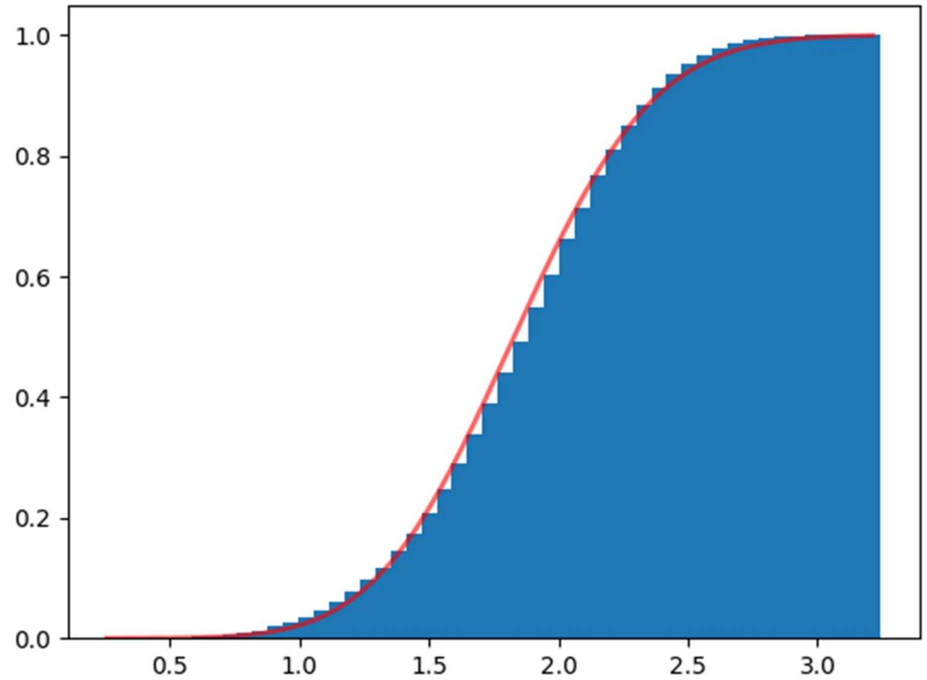
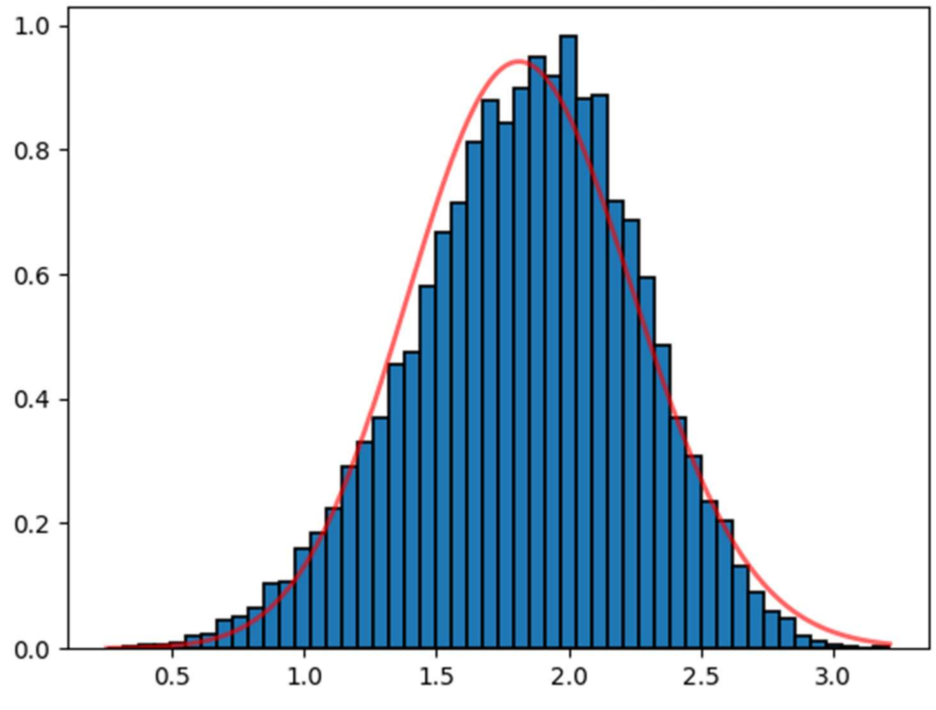
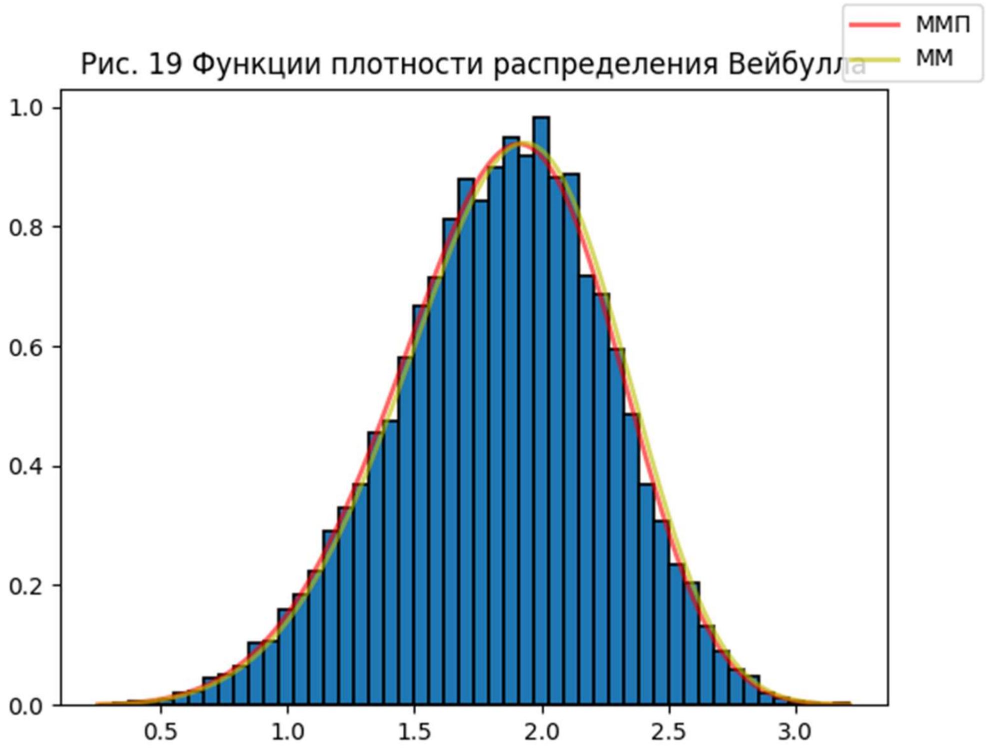


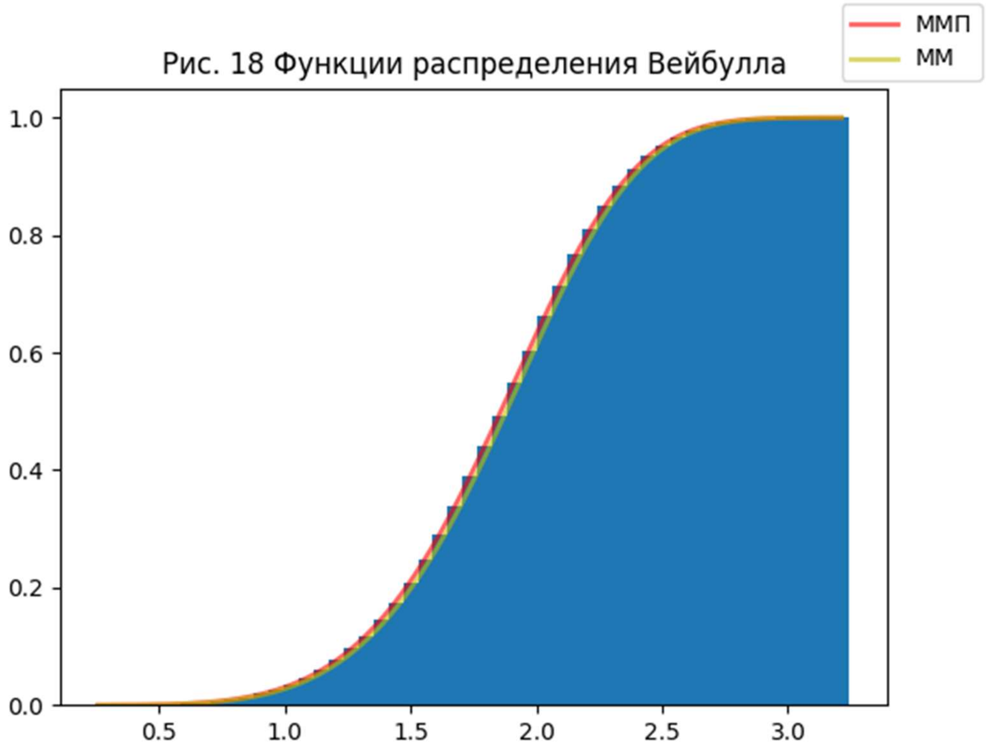
Рис. 17 Функция плотности гамма распределения



Также проверим распределение Вейбулла. Определим параметры.
Метод моментов:

k = 5.00; λ = 2.00
Метод максимального правдоподобия:
k = 4.9338; λ = 1.9784





Оценим для него ранее названные критерии.

Название	Вейбулла	
Формула плотности	$f(x;k;\lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right), x \geq 0 \\ 0, \qquad x < 0 \end{cases}$	
	ММ	ММП
Хи-квадрат статистика	141.83310	108.9981984
Хи-квадрат критич. знач.	178.870612	
Хи-Квадрат вывод	Удовлетворяют оба	
Колм. - Смирнова статистика	0.022705379	0.00533975
Колм. - Смирнова критич. знач.	0.0201746	
Колм.-Смирнова вывод	Удоавл. только ММП	
Мизеса статистика	3.02910	0.0623
Мизеса критич. знач.	0.4614	
Мизеса вывод	Удовл. только ММП	

Вывод

В данной лабораторной работе для данной выборки были построены графики распределения и плотности распределения. Так же были определены точечные оценки: первого начального момента, центральных моментов: второго - дисперсии, третьего, четвертого по выборочной функции распределения, асимметрии и эксцесса, границ интерквантильного промежутка J_p для $P = 0.95$.

Так же были получены интервальные оценки. С помощью гистограмм и точечных оценок были проверены гипотезы о различных распределениях. Затем была произведена проверка гипотез относительно выбранных теоретических законов распределения и их параметров. Проверка была проведена по трем критериям - "хи-квадрат", Колмогорова - Смирнова, Мизеса. Из пяти выбранных гипотез распределение Вейбулла оказалось единственным удовлетворяющим всем критическим значениям, но лишь с параметрами, определёнными методом максимального правдоподобия.

Листинг

```
import random
from math import sqrt
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from scipy.special import comb
from sklearn.neighbors import KernelDensity
from scipy.stats import (arcsine, triang, cauchy, invgauss, laplace, chi2,
expon, norm,
                         uniform, t, lognorm, gamma, rayleigh, pareto)



def random_color():
    r = random.randint
    # exclude white
    return '#%02X%02X%02X' % (r(0, 200), r(0, 200), r(0, 200))


# 1.1
with open('Task_2a.txt') as file:
    N = int(file.readline().split('=')[1])
    values = np.fromstring(file.readline(), dtype=float, sep=' ')

parts = 10
np.random.shuffle(values)
selections = np.array_split(np.copy(values), parts)
m = 50

step_function = np.zeros(m)
values_step = (max(values) + 0.001 * max(values) - min(values)) / m
columns_values = []
for t in range(1, m + 1):
    columns_values.append(min(values) + t * values_step)
for selection in selections + [values]:
    selection.sort()

linespace = np.linspace(values[0], values[-1], 100, dtype=float)
colors = [random_color() for _ in range(parts)]


def hist_prob(sample, plot, density=True, cumulative=False, **kwargs):
    kwargs['edgecolor'] = kwargs.get('edgecolor', 'k')
    kwargs['linewidth'] = kwargs.get('linewidth', 1.2)

    plot.hist(sample, 50, density=density, cumulative=cumulative, **kwargs)


def kde(sample, x, kernel='gaussian', bandwidth=0.15) -> np.ndarray:
    kde = KernelDensity(kernel=kernel, bandwidth=bandwidth)
    kde.fit(sample[:, np.newaxis])
    log_pdf = kde.score_samples(x[:, np.newaxis])
    return np.exp(log_pdf)


def kde_prob(sample, plot, kernel='gaussian', bandwidth=0.15, **kwargs):
    kwargs['label'] = kwargs.get('label', 'kde')
    density = kde(sample, linespace, kernel, bandwidth)
    plot.plot(linespace, density, **kwargs)
```

```
# 1.2.1

fig_1_1, chart = plt.subplots(1, 1)
chart.set_title("Рис.1 Ступенчатая выборочная функция распределения")
index = 0
for value in values:
    if value > columns_values[index]:
        p = int(abs(columns_values[index] - value) / values_step) + 1
        for i in range(1, p):
            step_function[index + i] = step_function[index]
        index += p
        step_function[index] = step_function[index - 1]
    step_function[index] += 1
chart.bar(columns_values, step_function / N, values_step)

fig_1_2, chart = plt.subplots(1, 1)
cdf = np.arange(0, 1, 1 / N)
chart.set_title("Рис.2 Выборочная функция распределения")
chart.plot(values, cdf)

# 1.2.2
fig_2, chart = plt.subplots(1, 2)
chart = chart.reshape(2)

hist_prob(values, chart[0], density=False)
chart[0].set_title("Рис.3 Абс. гистограмма")

hist_prob(values, chart[1])
chart[1].set_title("Рис.4 Отн. гистограмма и KDE")
kde_prob(values, chart[1], kernel="epanechnikov", bandwidth=0.4,
label="Епанечниково")
kde_prob(values, chart[1], kernel="linear", bandwidth=0.4,
label="Треугольное")
kde_prob(values, chart[1], kernel="gaussian",
         bandwidth=0.4, label="Гауссово")
fig_2.legend()


def point_estimations(sample):
    data = {}
    data['mean'] = np.mean(sample)
    data['median'] = np.median(sample)
    data['middle'] = (max(sample) + min(sample)) / 2
    data['variance'] = np.var(sample, ddof=1)
    data['moment 3'] = stats.moment(sample, 3)
    data['moment 4'] = stats.moment(sample, 4)
    data['kurtosis'] = stats.kurtosis(sample)
    data['skewness'] = stats.skew(sample)
    return data


fig_3_1, chart = plt.subplots(4, 2)
fig_3_1.subplots_adjust(hspace=1, left=.07, bottom=.05, right=.85)
chart = chart.reshape(8)


def var_interval(sample, q):
    var = np.var(sample, ddof=1)
    n = len(sample)

    left = var * (n - 1) / stats.chi2.ppf((1 + q) / 2, n - 1)
    right = var * (n - 1) / stats.chi2.ppf((1 - q) / 2, n - 1)
    return np.array([left, right])
```

```
def mean_interval(sample, q):
    mean = np.mean(sample)
    std = np.std(sample, ddof=1)
    n = len(sample)

    x = std / sqrt(n) * stats.t.ppf(0.5 + q / 2, n - 1)
    return np.array([mean - x, mean + x])


for selection_index, selection in enumerate(selections):
    selection_info = point_estimations(selection)
    print(selection_info)
    for i, data in enumerate(selection_info.keys()):
        chart[i].set_title(data)
        chart[i].set_yticklabels([])

        # adds labels
        if i == 0:
            chart[i].scatter(selection_info[data], 0,
color=colors[selection_index],
                             marker='+', s=35, label='Выборка
{0}'.format(selection_index + 1))
        else:
            chart[i].scatter(selection_info[data], 0,
                             color=colors[selection_index], marker='+', s=35)

main_selection_info = point_estimations(values)
print(main_selection_info)
for i, data in enumerate(main_selection_info):
    # adds labels
    if i == 0:
        #data['mean'], data['variance'], data['skewness'], data['kurtosis'] =
stats.laplace.stats(moments='mvsk')
        chart[i].scatter(main_selection_info[data], 0,
                         color='r', marker='.', s=100, label='Глав. выборка')
    else:
        chart[i].scatter(main_selection_info[data], 0,
                         color='r', marker='.', s=100)
fig_3_1.legend()

fig_3_2, chart = plt.subplots(11)
fig_3_3, chart1 = plt.subplots(11)
fig_3_2.subplots_adjust(hspace=1, left=.07, bottom=.05, right=.85)
fig_3_3.subplots_adjust(hspace=1, left=.07, bottom=.05, right=.85)
chart1 = chart1.reshape(11)
chart = chart.reshape(11)

q = 0.8
p = 0.95
lp = stats.laplace

for selection_index, selection in enumerate(selections):
    mean = mean_interval(selection, q)
    print("Mean L: " + str(mean[0]) + " R: " + str(mean[-1]))
    var = var_interval(selection, q)
    print("Var L: " + str(var[0]) + " R: " + str(var[-1]))
    chart[selection_index].fill_between(
        mean, 0, 1, color=colors[selection_index], alpha=0.8, label='Выборка
{0}'.format(selection_index + 1))
    chart1[selection_index].fill_between(
        var, 0, 1, color=colors[selection_index], alpha=0.8, label='Выборка
{0}'.format(selection_index + 1))
```

```
mean = mean_interval(values, q)
print("Mean L: " + str(mean[0]) + " R: " + str(mean[-1]))
var = var_interval(values, q)
print("Var L: " + str(var[0]) + " R: " + str(var[-1]))
chart[10].fill_between(mean, 0, 1, color='r', alpha=0.8, label='Глав.
выборка')
chart1[10].fill_between(var, 0, 1, color='r', alpha=0.8, label='Глав.
выборка')
fig_3_2.legend()
fig_3_3.legend()


# 1.4
def equivalent_k(n):
    def get_sum(k):
        return sum(comb(n, m) * p ** m * (1 - p) ** (n - m)
                   for m in range(n - k, n + 1))

    k = 0
    while get_sum(k) < 1 - q:
        k += 1

    return k


def interval_plot(plot, l_q, l_p, r_p, r_q):
    plot.scatter(l_q, 0, color='r', marker='>', label="Толерантный предел")
    plot.scatter(l_p, 0, color='k', marker='>', s=40, label="Интерквантильный
промежуток")
    plot.scatter(r_p, 0, color='k', marker='<', s=40)
    plot.scatter(r_q, 0, color='r', marker='<')
    #plot.legend()


fig_4, chart = plt.subplots(1, 1)
chart.set_yticklabels([])
chart.set_title('Рис.8 Пределы интерквантильного промежутка для глав.
выборки')
k = equivalent_k(N)
print(k)

if k % 2:
    k -= 1

left_q = values[k // 2]
right_q = values[N - k // 2]
left_q_zero = -values[N - k + 1]
right_q_zero = values[N - k + 1]
left_p = np.quantile(values, 0.5 - p / 2)
right_p = np.quantile(values, 0.5 + p / 2)
print(left_q, right_q)
print(left_p, right_p)
print(left_q_zero, right_q_zero)
interval_plot(chart, left_q, left_p, right_p, right_q)
fig_4_2, chart1 = plt.subplots(1)
chart1.scatter(left_q_zero, 0, color='r', marker='>')
chart1.scatter(right_q_zero, 0, color='r', marker='<')
chart1.set_title("Рис.9 Толерантные пределы симметричные отн. нуля")

fig_5, chart = plt.subplots(5, 2)
fig_5.subplots_adjust(hspace=1, wspace=0.3)
chart = chart.reshape(10)
```

```python
for i, selection in enumerate(selections):
    chart[i].set_title("Выборка {0}".format(i + 1))
    chart[i].set_yticklabels([])
    mean = np.mean(selection)
    std = np.std(selection, ddof=1)
    k = 1.65
    left_p = np.quantile(selection, 0.5 - p / 2)
    right_p = np.quantile(selection, 0.5 + p / 2)
    interval_plot(chart[i], mean - k * std, left_p,
                  right_p, mean + k * std)


series = pd.Series(values)
IQR = series.quantile(0.75) - series.quantile(0.25)
print(IQR / 2)

a, b = stats.norm.fit(values)
print("n ", a, b)
a, b = stats.cauchy.fit(values)
print("c ", a, b)
a, b, c = stats.t.fit(values)
print("t ", a, b, c)

values.sort()

fig_6, chart = plt.subplots(1, 1)
chart.set_title("Рис.10 Функции плотностей нормального распределения")
hist_prob(values, chart)
x = np.linspace(values.min(), values.max(), 13000)
chart.plot(x, stats.norm.pdf(x, 1.8390, 0.4234), 'r-', lw=2, alpha=0.6,
label='ММ')
chart.plot(x, stats.norm.pdf(x, 1.8364, 0.4206), 'y-', lw=2, alpha=0.6,
label='ММП')
fig_6.legend()

fig_7, chart = plt.subplots(1, 1)
chart.set_title("Рис.11 Функции плотностей распределения Коши")
hist_prob(values, chart)
x = np.linspace(values.min(), values.max(), 13000)
chart.plot(x, stats.cauchy.pdf(x, 1.8452, 0.2848), 'r-', lw=2, alpha=0.6,
label='ММ')
chart.plot(x, stats.cauchy.pdf(x, 1.8689, 0.2583), 'y-', lw=2, alpha=0.6,
label='ММП')
fig_7.legend()

fig_8, chart = plt.subplots(1, 1)
chart.set_title("Рис.12 Функции плотностей распределения Стьюдента")
hist_prob(values, chart)
x = np.linspace(values.min(), values.max(), 13000)
chart.plot(x, stats.t.pdf(x, 601624759, 1.8390, 0.4234), 'r-', lw=2,
alpha=0.6, label='ММ')
chart.plot(x, stats.t.pdf(x, 601624759, 1.8364, 0.4206), 'y-', lw=2,
alpha=0.6, label='ММП')
fig_8.legend()

fig_9, chart = plt.subplots(1, 1)
chart.set_title("Рис.13 Функции нормального распределения")
chart.bar(columns_values, step_function / N, values_step)
stats.ecdf(values).cdf.plot()
chart.plot(x, stats.norm.cdf(x, 1.8390, 0.4234), 'r-', lw=2, alpha=0.6,
label='ММ')
chart.plot(x, stats.norm.cdf(x, 1.8364, 0.4206), 'y-', lw=2, alpha=0.6,
label='ММП')
fig_9.legend()
```

```python
fig_10, chart = plt.subplots(1, 1)
chart.set_title("Рис.14 Функции распределения Коши")
chart.bar(columns_values, step_function / N, values_step)
chart.plot(x, stats.cauchy.cdf(x, 1.8452, 0.2848), 'r-', lw=2, alpha=0.6,
label='ММ')
chart.plot(x, stats.cauchy.cdf(x, 1.8689, 0.2583), 'y-', lw=2, alpha=0.6,
label='ММП')
fig_10.legend()

fig_11, chart = plt.subplots(1, 1)
chart.set_title("Рис.15 Функции распределения Стьюдента")
chart.bar(columns_values, step_function / N, values_step)
chart.plot(x, stats.t.cdf(x, 601624759, 1.8390, 0.4234), 'r-', lw=2,
alpha=0.6, label='ММ')
chart.plot(x, stats.t.cdf(x, 601624759, 1.8364, 0.4206), 'y-', lw=2,
alpha=0.6, label='ММП')
fig_11.legend()

# Нормальное распределение
ks_stat, ks_p_value = stats.kstest(values, 'norm', args=(1.8390, 0.4234))
print('Колмогоров-Смирнов n ММ', ks_stat, ks_p_value)
ks_stat, ks_p_value = stats.kstest(values, 'norm', args=(1.8364, 0.4206))
print('Колмогоров-Смирнов n ММП', ks_stat, ks_p_value)

# Распределение Коши
ks_stat_cauchy, ks_p_value_cauchy = stats.kstest(values, 'cauchy',
args=(1.8452, 0.2848))
print('Колмогоров-Смирнов c ММ', ks_stat_cauchy, ks_p_value_cauchy)
ks_stat_cauchy, ks_p_value_cauchy = stats.kstest(values, 'cauchy',
args=(1.8689, 0.2583))
print('Колмогоров-Смирнов c ММП', ks_stat_cauchy, ks_p_value_cauchy)

# t-распределение
ks_stat_t, ks_p_value_t = stats.kstest(values, 't', args=(601624759, 1.8390,
0.4234))
print('Колмогоров-Смирнов t ММ', ks_stat_t, ks_p_value_t)
ks_stat_t, ks_p_value_t = stats.kstest(values, 't', args=(601624759, 1.8364,
0.4206))
print('Колмогоров-Смирнов t ММП', ks_stat_t, ks_p_value_t)


def chi_square_test(data, distribution, params, bins=114):
    observed_freq, bin_edges = np.histogram(data, bins=bins, density=False)
    cdf = distribution.cdf(bin_edges, *params)
    expected_freq = len(data) * np.diff(cdf)

    # Ensure the sums are the same
    scale_factor = observed_freq.sum() / expected_freq.sum()
    expected_freq *= scale_factor

    chi_stat, chi_p_value = stats.chisquare(observed_freq, expected_freq)
    return chi_stat, chi_p_value


# Нормальное распределение
ks_stat, ks_p_value = chi_square_test(values, stats.norm, (1.8390, 0.4234))
print('Хи-квадрат n ММ', ks_stat, ks_p_value)
ks_stat, ks_p_value = chi_square_test(values, stats.norm, (1.8364, 0.4206))
print('Хи-квадрат n ММП', ks_stat, ks_p_value)

# Распределение Коши
ks_stat_cauchy, ks_p_value_cauchy = chi_square_test(values, stats.cauchy,
(1.8452, 0.2848))
```

```
print('Хи-квадрат с ММ', ks_stat_cauchy, ks_p_value_cauchy)
ks_stat_cauchy, ks_p_value_cauchy = chi_square_test(values, stats.cauchy,
(1.8689, 0.2583))
print('Хи-квадрат с ММП', ks_stat_cauchy, ks_p_value_cauchy)

# t-распределение
ks_stat_t, ks_p_value_t = chi_square_test(values, stats.t, (601624759,
1.8390, 0.4234))
print('Хи-квадрат t ММ', ks_stat_t, ks_p_value_t)
ks_stat_t, ks_p_value_t = chi_square_test(values, stats.t, (601624759,
1.8364, 0.4206))
print('Хи-квадрат t ММП', ks_stat_t, ks_p_value_t)


# Нормальное распределение
cvm_stat = stats.cramervonmises(values, 'norm', args=(1.8390, 0.4234))
print('Мизес n ММ:', cvm_stat)
cvm_stat = stats.cramervonmises(values, 'norm', args=(1.8363647545384614,
0.42058517731132417))
print('Мизес n ММП:', cvm_stat)

# Распределение Коши
cvm_stat_cauchy = stats.cramervonmises(values, 'cauchy', args=(1.8452,
0.2848))
print('Мизес c ММ:', cvm_stat_cauchy)
cvm_stat_cauchy = stats.cramervonmises(values, 'cauchy', args=(1.8689,
0.2583))
print('Мизес c ММП:', cvm_stat_cauchy)

# t-распределение
cvm_stat_t = stats.cramervonmises(values, 't', args=(601624759, 1.8390,
0.4234))
print('Мизес t ММ:', cvm_stat_t)
cvm_stat_t = stats.cramervonmises(values, 't', args=(6.01624760e+08,
1.83636478e+00, 4.20585177e-01))
print('Мизес t ММП:', cvm_stat_t)

# Уровень значимости
alpha = 0.0001
df = 114

# Критическое значение
ks_critical_value = stats.kstwo.ppf(1 - alpha / 2, 13000)
print('Критическое значение Колмогорова-Смирнова:', ks_critical_value)


# Критическое значение
chi_critical_value = stats.chi2.ppf(1 - alpha, df)
print('Критическое значение хи-квадрат:', chi_critical_value)

print('Критическое значение Cramér-von Mises:', 0.4614)


a, b, c = stats.weibull_min.fit(values)
print("w ", a, b, c)


fig_11, chart = plt.subplots(1, 1)
chart.set_title("Рис. 18 Функции распределения Вейбулла")
chart.bar(columns_values, step_function / N, values_step)
chart.plot(x, stats.weibull_min.cdf(x, a, b, c), 'r-', lw=2, alpha=0.6,
label='ММП')
chart.plot(x, stats.weibull_min.cdf(x, 5.00, b, 2.00), 'y-', lw=2, alpha=0.6,
label='ММ')
```

```
fig_11.legend()

fig_12, chart = plt.subplots(1, 1)
chart.set_title("Рис. 19 Функции плотности распределения Вейбулла")
hist_prob(values, chart)
x = np.linspace(values.min(), values.max(), 13000)
chart.plot(x, stats.weibull_min.pdf(x, a, b, c), 'r-', lw=2, alpha=0.6,
label='ММП')
chart.plot(x, stats.weibull_min.pdf(x, 5.00, b, 2.00), 'y-', lw=2, alpha=0.6,
label='ММ')
fig_12.legend()

ks_stat_t, ks_p_value_t = stats.kstest(values, 'weibull_min', args=(a, b, c))
print('Колмогоров-Смирнов в', ks_stat_t)
cvm_stat_gamma = stats.cramervonmises(values, 'weibull_min', args=(a, b, c))
print('Мизес в:', cvm_stat_gamma)
ks_stat_t, ks_p_value_t = chi_square_test(values, stats.weibull_min, (a, b,
c))
print('Хи-квадрат в', ks_stat_t)




# Генерируем случайные данные для примера (можно заменить на свои данные)
data = values

# Функция для вычисления критерия Омега-квадрат
def omega_squared(data, dist, params):
    # Получаем эмпирическую функцию распределения данных
    ecdf = np.arange(1, len(data) + 1) / len(data)
    # Сортируем данные
    sorted_data = np.sort(data)
    # Получаем теоретическую функцию распределения
    cdf = dist.cdf(sorted_data, *params)
    # Вычисляем Омега-квадрат
    omega2 = np.sum((cdf - ecdf)**2) + 1 / (12 * len(data))
    return omega2

# Определяем распределения, которые хотим проверить
distributions = {
    'arcsine': arcsine,
    'triang': triang,
    'cauchy': cauchy,
    'invgauss': invgauss,
    'laplace': laplace,
    'chi2': chi2,
    'expon': expon,
    'norm': norm,
    'uniform': uniform,
    't': t,
    'lognorm': lognorm,
    'gamma': gamma,
    'rayleigh': rayleigh,
    'pareto': pareto
}

# Выбираем наилучшее распределение и его параметры
best_distribution = None
best_params = None
best_omega2 = np.inf

for name, dist in distributions.items():
    try:
        # Оцениваем параметры распределения
```

```
        params = dist.fit(data)
        # Вычисляем критерий Омега-квадрат
        omega2 = omega_squared(data, dist, params)
        if omega2 < best_omega2:
            best_distribution = dist
            best_params = params
            best_omega2 = omega2
    except Exception as e:
        print(f"Ошибка при обработке распределения {name}: {e}")

# Выводим результаты
if best_distribution:
    print("Ближайшее распределение:", best_distribution.name)
    print("Параметры:", best_params)
else:
    print("Не удалось определить наилучшее распределение")



plt.show()
```