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. Считать выборку X из файла. Создать на ее основе 10 подвыборок – для этого перемешать выборку
   2. Представить визуально оценки функции плотности распределения:
      1. Построить выборочную функцию распределения F(x)
      2. Построить абсолютную и относительную гистограммы на разных графиках
      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
   3. Определить точечные оценки:
      1. первого начального момента
      2. центральных моментов: второго - дисперсии, третьего, четвертого по выборочной функции распределения
      3. асимметрии и эксцесса
      4. границ интерквантильного промежутка Jp для P = 0.95 только по полной выборке
      5. представить результаты графически
   4. Определить интервальные оценки с доверительной вероятностью Q = 0.8:
      1. первого начального и второго центрального моментов
      2. интерквантильного промежутка J для P=0.95:
         1. по всей выборке с помощью непараметрических толерантных пределов, симметричных относительно среднего арифметического и относительно нуля:
         2. по частичным выборкам с помощью параметрических толерантных пределов, считая закон распределения генеральной совокупности нормальным
         3. Результаты представить графически
   5. Сделать выводы относительно ширины доверительных интервалов. Сравнить:
      1. интерквантильные промежутки с толерантными пределами
      2. параметрические и непараметрические толерантные пределы, симметричные относительно среднего арифметического и относительно нуля
2. **Идентификация закона и параметров распределения**
   1. Начальный выбор распределения
   2. Определение параметров теоретических распределений. Для выбранных теоретических распределений необходимо определить точные значения параметров, наиболее подходящие для описания выборки. Это необходимо сделать двумя способами:
      1. С помощью метода моментов
      2. С помощью метода максимального правдоподобия
      3. Сравнить оценки, полученные методом моментов и ММП
   3. Произвести проверку гипотез относительно выбранных теоретических законов распределения и их параметров. Проверку провести по трем критериям - "хи-квадрат", Колмогорова - Смирнова, "омега-квадрат".
   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.

Изображение выглядит как текст, диаграмма, снимок экрана, График

Автоматически созданное описание

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

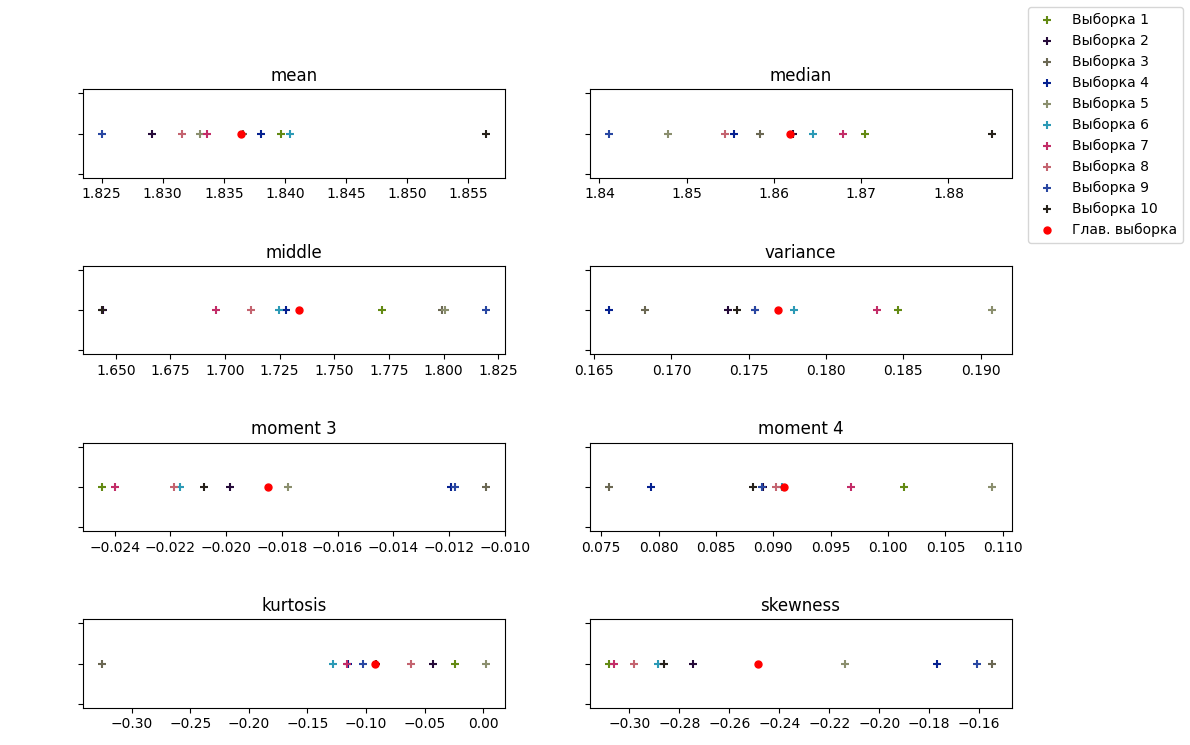
Далее построим абсолютные и относительные гистограммы оценки плотности, также применим kde алгоритмы следующих типов: Епанечниково, треугольное и Гауссово. Очевидно, что все оценки ядер, кроме последнего, крайне похожи.

Изображение выглядит как снимок экрана, диаграмма, График, текст

Автоматически созданное описание

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.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 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 |
| 1. N/10 | 1.8298 | 1.8530 | 1.7591 | 0.1758 | -0.0180 | 0.0897 | -0.0921 | -0.2446 |
| 1. N/10 | 1.8317 | 1.8551 | 1.7588 | 0.1706 | -0.0227 | 0.0836 | -0.1243 | -0.3228 |
| 1. N/10 | 1.8563 | 1.8980 | 1.7032 | 0.1711 | -0.0226 | 0.08626 | -0.0482 | -0.3200 |
| 1. N/10 | 1.8466 | 1.8647 | 1.7338 | 0.1779 | -0.0154 | 0.0907 | -0.1270 | -0.2057 |
| 1. N/10 | 1.8411 | 1.8620 | 1.6573 | 0.1676 | -0.0149 | 0.0799 | -0.1507 | -0.2170 |
| 1. N/10 | 1.8462 | 1.8726 | 1.6630 | 0.1810 | -0.0221 | 0.0975 | -0.0171 | -0.2870 |
| 1. N/10 | 1.8269 | 1.8399 | 1.7079 | 0.1720 | -0.0118 | 0.0842 | -0.1490 | -0.1662 |
| 1. N/10 | 1.8289 | 1.8558 | 1.7717 | 0.1812 | -0.0236 | 0.0962 | -0.0632 | -0.3067 |
| 1. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| N | 1.8316 | 1.8411 | 0.1741 | 0.1798 |
| 1. N/10 | 1.8244 | 1.8549 | 0.1757 | 0.1944 |
| 1. N/10 | 1.8143 | 1.8440 | 0.1653 | 0.1828 |
| 1. N/10 | 1.8212 | 1.8511 | 0.1602 | 0.1772 |
| 1. N/10 | 1.8236 | 1.8526 | 0.1579 | 0.1746 |
| 1. N/10 | 1.8175 | 1.8486 | 0.1816 | 0.2008 |
| 1. N/10 | 1.8254 | 1.8554 | 0.1693 | 0.1872 |
| 1. N/10 | 1.8184 | 1.8488 | 0.1745 | 0.1929 |
| 1. N/10 | 1.8167 | 1.8464 | 0.1670 | 0.1846 |
| 1. N/10 | 1.8101 | 1.8399 | 0.1669 | 0.1846 |
| 1. N/10 | 1.8416 | 1.8712 | 0.1658 | 0.1834 |

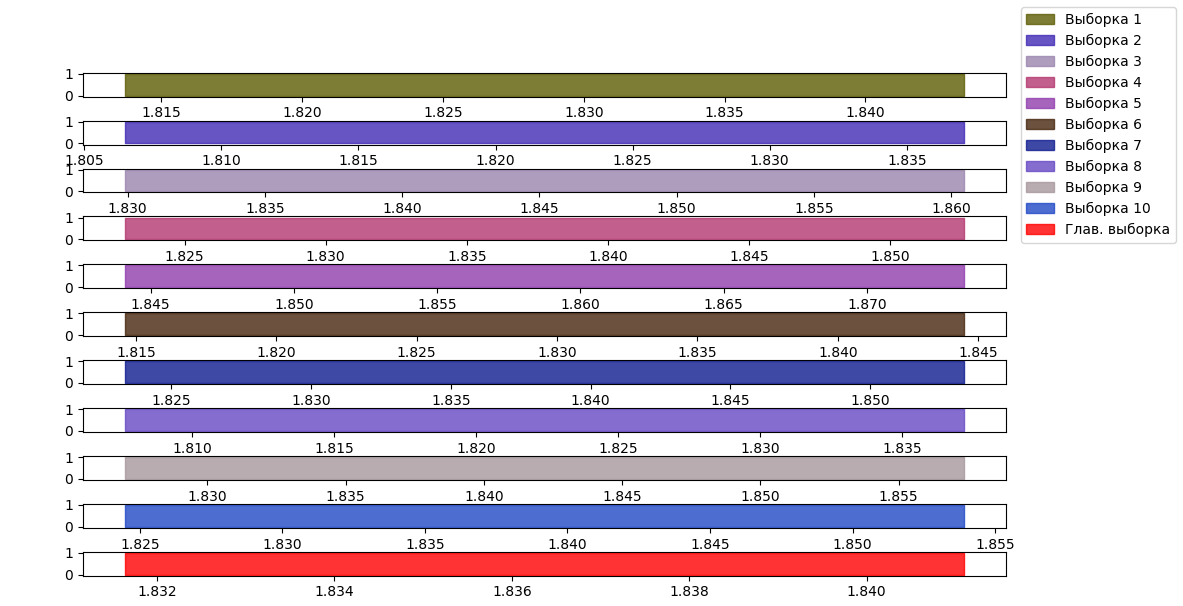
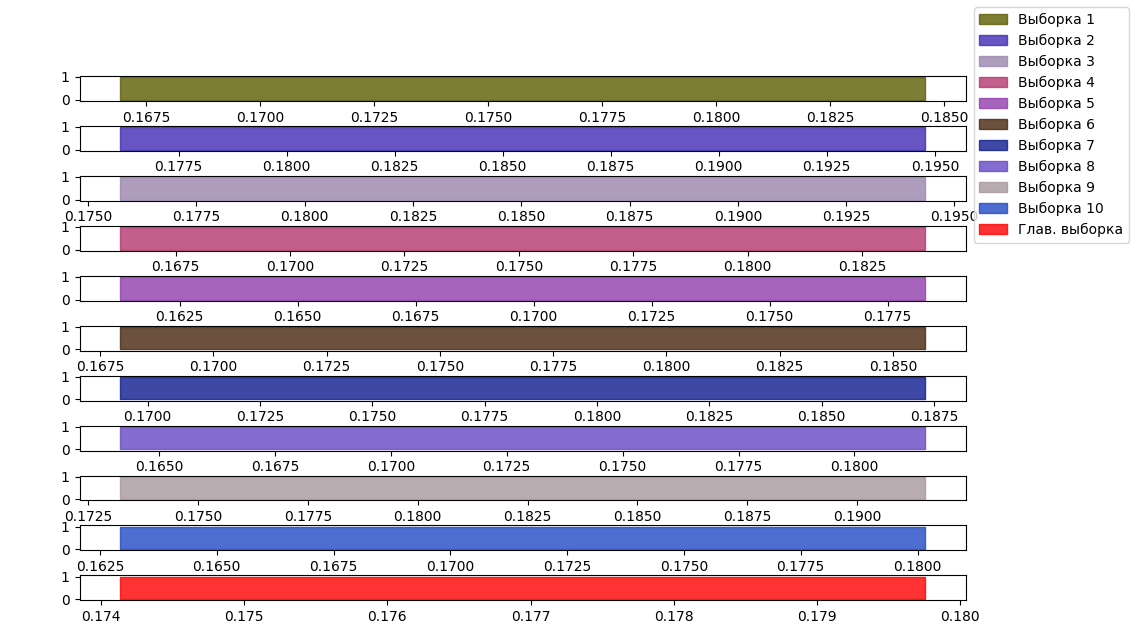
Рис.6 Интервальные оценки мат. ожидания

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:

Результирующий предел будет равен при чётном k или 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

Полученное k = 127. Очевидно, k – нечётное число, следовательно используем формулу .

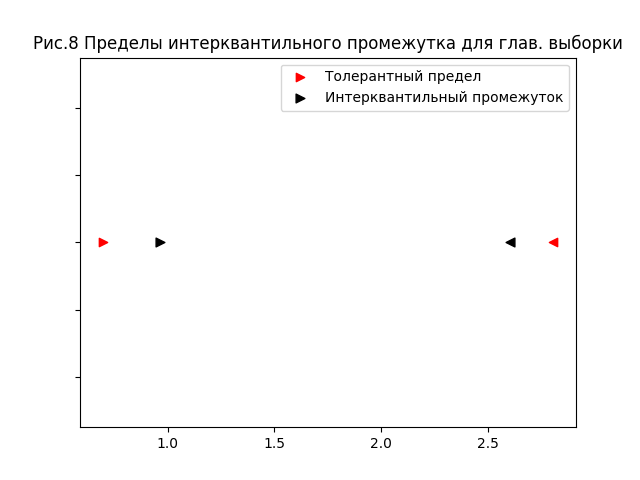
;

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)

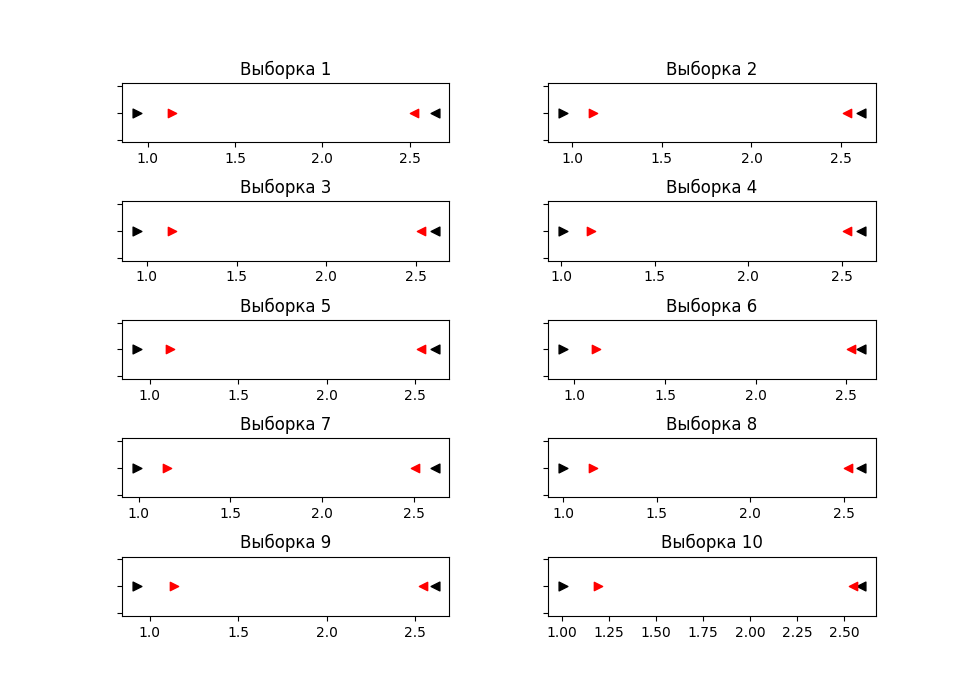
Представим полученные результаты графически:

Определим толерантные пределы симметричные относительно нуля:



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 -

Коши -

Стьюдента -

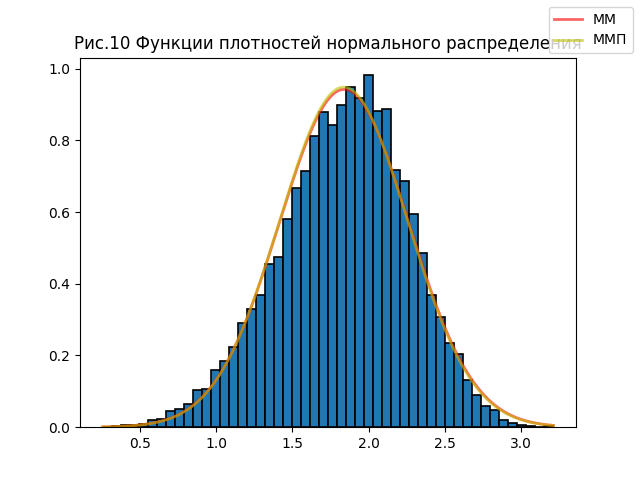
Метод максимального правдоподобия:

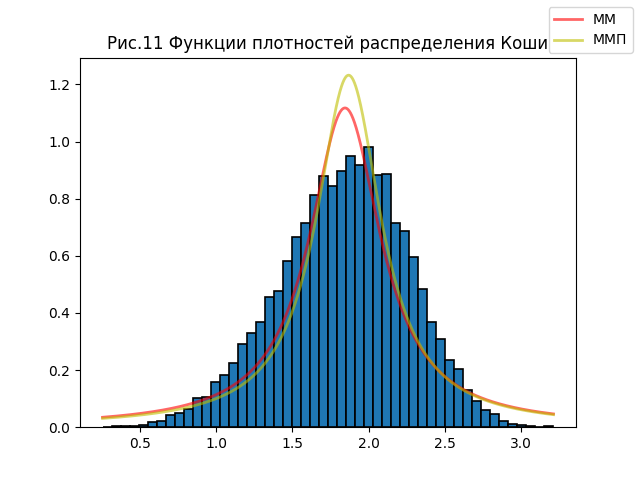
Нормальное -

Коши -

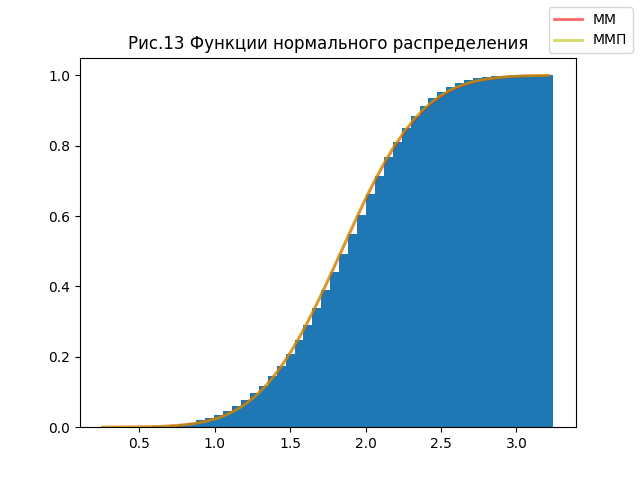
Стьюдента -

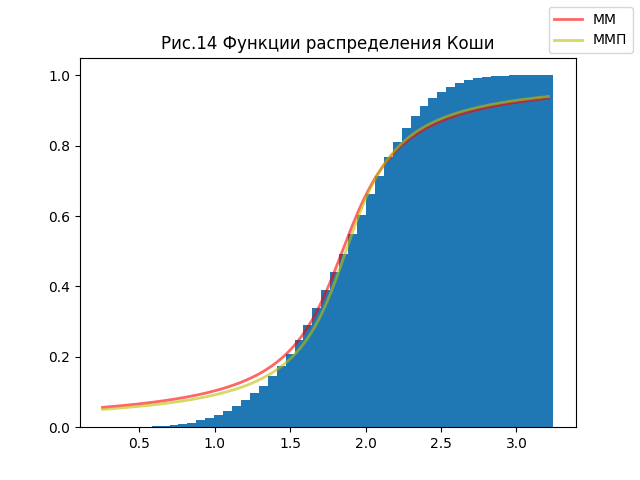
Представим результаты графически:

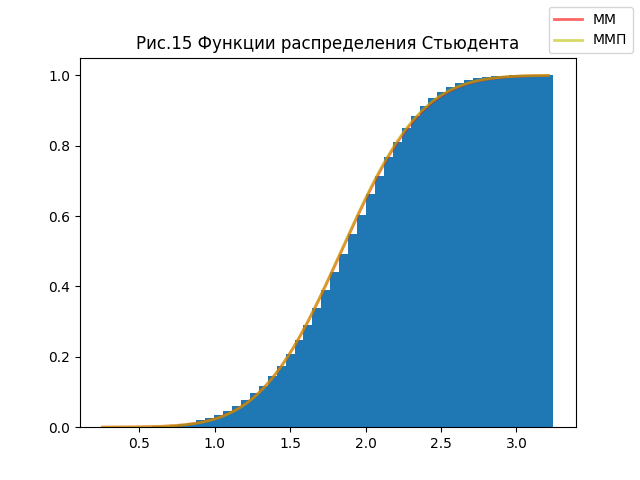












На основе полученных графиков можно сделать вывод, что данная выборка описывается либо законом нормального распределения, либо законом распределения Стьюдента. Также следует отметить, что метод моментов показал большую точность, чем ММП.

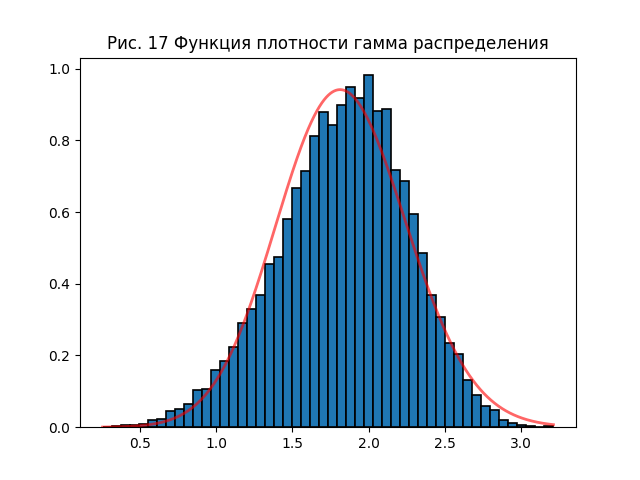
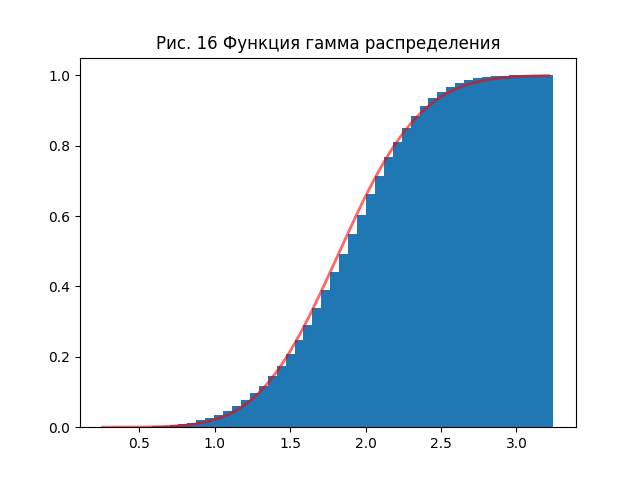
Произведём проверку гипотез относительно выбранных теоретических законов распределения и их параметров. Проверку проведём по трем критериям - "хи-квадрат", Колмогорова - Смирнова, "омега-квадрат". Альфа возьмём как 0.05.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Нормальное | | Коши | | Стьюдента | |
| Формула плотности |  | |  | |  | |
|  | 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.



**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 Jp 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. К сожалению, ни одно из распределений не обладает удовлетворительной точностью, хотя нормальное распределение с параметрами, определёнными методом моментов обладают наибольшей схожестью с основной выборкой по всем трём критериям.

**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)  
  
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()  
  
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 c 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 c 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.gamma.fit(values)  
print("g ", a, b, c)  
  
  
fig\_11, chart = plt.subplots(1, 1)  
chart.set\_title("Рис. 16 Функция гамма распределения")  
chart.bar(columns\_values, step\_function / N, values\_step)  
chart.plot(x, stats.gamma.cdf(x, 359.2836728552438, -6.204869331992612, 0.0223779807198273), 'r-', lw=2, alpha=0.6)  
  
fig\_12, chart = plt.subplots(1, 1)  
chart.set\_title("Рис. 17 Функция плотности гамма распределения")  
hist\_prob(values, chart)  
x = np.linspace(values.min(), values.max(), 13000)  
chart.plot(x, stats.gamma.pdf(x, 359.2836728552438, -6.204869331992612, 0.0223779807198273), 'r-', lw=2, alpha=0.6)  
  
  
ks\_stat\_t, ks\_p\_value\_t = stats.kstest(values, 'gamma', args=(359.2836728552438, -6.204869331992612, 0.0223779807198273))  
print('Колмогоров-Смирнов г', ks\_stat\_t, ks\_p\_value\_t)  
cvm\_stat\_gamma = stats.cramervonmises(values, 'gamma', args=(359.2836728552438, -6.204869331992612, 0.0223779807198273))  
print('Мизес г :', cvm\_stat\_gamma)  
ks\_stat\_t, ks\_p\_value\_t = chi\_square\_test(values, stats.gamma, (359.2836728552438, -6.204869331992612, 0.0223779807198273))  
print('Хи-квадрат г', ks\_stat\_t, ks\_p\_value\_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()