Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Расчетное задание №2

Тема: Статистическая обработка случайных последовательностей.
Идентификация законов распределения.
Дисциплина: Теория вероятностей и математическая статистика

Выполнил студент гр. 3530901/10003 ____________________ Сергиенко К. А.
(подпись)

Руководитель ____________________ Никитин К. В.
(подпись)

“____” _____________ 2023 г.

Санкт-Петербург
2023

Оглавление

Исходные данные

В результате измерений получена выборка x1, x2, …, xN из генеральной совокупности с неизвестным законом распределения.

Задание

1.Статистическая обработка случайных последовательностей

1.1. Считать выборку X из файла. Создать на ее основе 10 подвыборок

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.

1.2.3. Построить оценки плотности с применением ядерного оценивания.

1.3. Определить точечные оценки (результаты представить в виде таблицы):

1.3.1. моментов:

•первого начального

•центральных моментов: второго-дисперсии, третьего, четвертого по выборочной функции распределения. Определить моду.

1.3.2. асимметрии и эксцесса.

1.3.3. границ интерквантильного промежутка для P = 0.95 только по полной выборке

1.3.4. характеристики по пп. 1.3.1-1.3.2 по подвыборкам, сформированным в п. 1.1.

1.4. Определить интервальные оценки с доверительной вероятностью Q=0.8:

•первого начального и второго центрального моментов (вычисления выполнить по полной выборке и по отдельным частям, как в п. 2.1.4-по N/10 значений в каждой частичной выборке). Нанести на эти характеристики соответствующие значения точечных оценок.

•интерквантильного промежутка J для P=0.95 (результаты представить только графически):

– по всей выборке с помощью непараметрических толерантных пределов, симметричных относительно среднего арифметического и относительно нуля.

– по частичным выборкам с помощью параметрических толерантных пределов.

Сделать выводы относительно ширины доверительных интервалов. Сравнить:

а) интерквантильные промежутки с толерантными пределами

б) параметрические и непараметрические толерантные пределы, симметричные относительно среднего арифметического и относительно нуля

2. Идентификация закона и параметров распределения

2.1. Начальный выбор распределения. Для начальной ориентировки в выборе закона использовать вид гистограммы, функции распределения, соотношения между моментами и полученные значения эксцесса и асимметрии. Определиться с основными распределениями, которые будут идентифицироваться.

2.2. Определение параметров теоретических распределений. Для выбранных теоретических распределений необходимо определить точные значения параметров, наиболее подходящие для описания выборки:

– с помощью метода моментов

– с помощью метода максимального правдоподобия

Сравнить оценки, полученные методом моментов и ММП. Для этого построить:

– эмпирическую и теоретические функцию распределения (на 1 графике)

– гистограмму и теоретические плотности распределения (на 1 графике)

2.3. Произвести проверку гипотез относительно выбранных теоретических законов распределения и их параметров (по методу ММП и моментов). Проверку провести по трем критериям - "хи-квадрат", Колмогорова-Смирнова, “омега-квадрат”.

2.4. Привести итоговую таблицу.

Практическое решение

Обрабатывать выборку, полученные в результате исследования, будем при помощи программы на языке python версии 3.10 с использованием следующих библиотек:

– matplotlib для построения графиков

– numpy для работы с данными

– prettytable для построения таблиц

– random для работы со случайными величинами

– scipy для работы с распределениями

Код программы с комментариями приведен в листинге.

Результат

Часть 1 Статистическая обработка случайной последовательности

1.2.1. Выборочная функция распределения F(x)

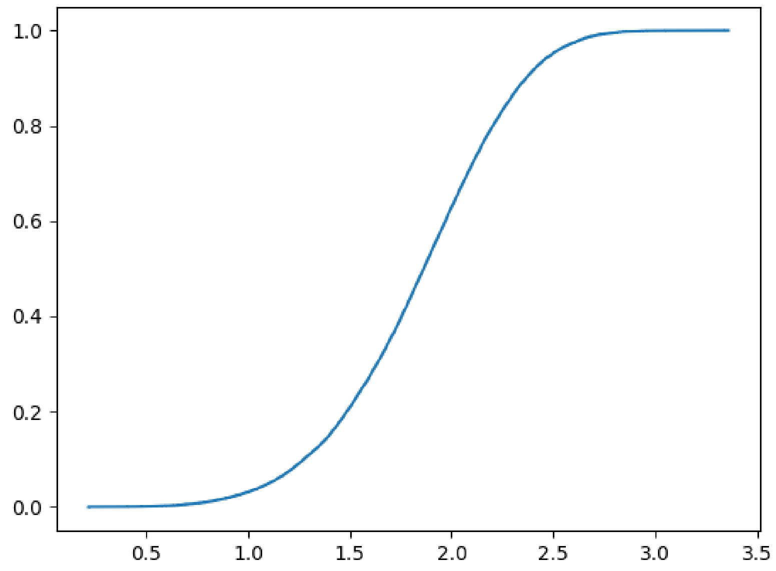

Рис. 1.1. График функции распределения F(x)

1.2.2. Абсолютная и относительная гистограммы

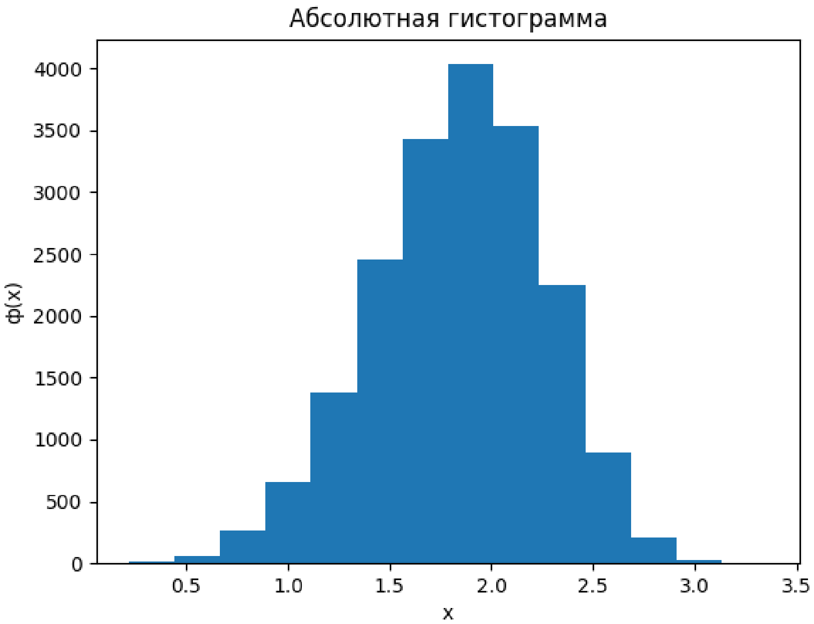

Рис. 1.2. Абсолютная гистограмма

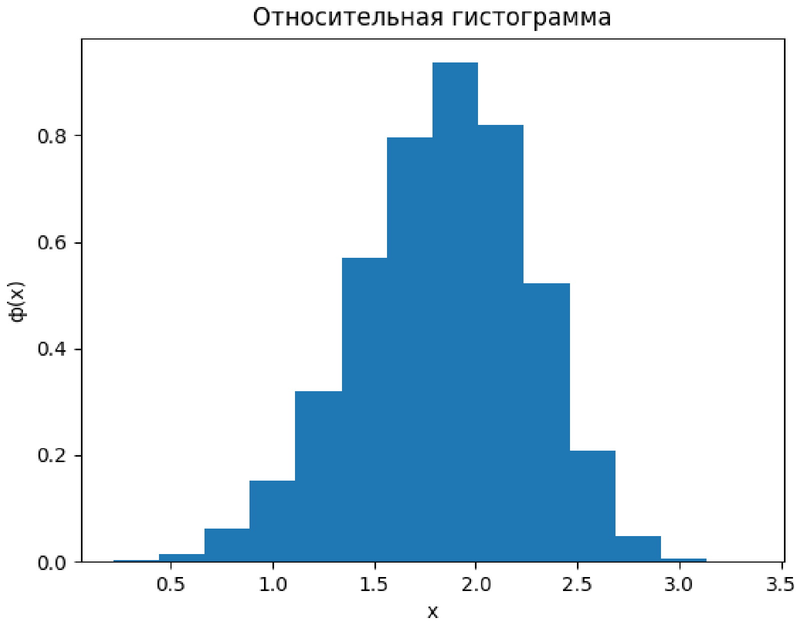

Рис. 1.3. Относительная гистограмма

1.2.3. Оценки плотности с применением ядерного оценивания

Проведём оценку плотности распределения с применением ядерного оценивания. Используем Гауссово, экспоненциальное ядро и ядро Коши. Будем варьировать ширину окна h.

Как видно из рисунков 1.5, при уменьшении значения ширины окна h функции принимают более резкий вид. В то время как при увеличении – более пологий.

гауссово	$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$
Экспоненциальное (Лапласа)	$K(u) = \frac{1}{2} \exp(-\lvert u \rvert)$
Коши	$K(u) = \frac{1}{\pi} \frac{1}{1+u^2}$

Рис. 1.4. Одномерные уравнения для выбранных типов ядер.

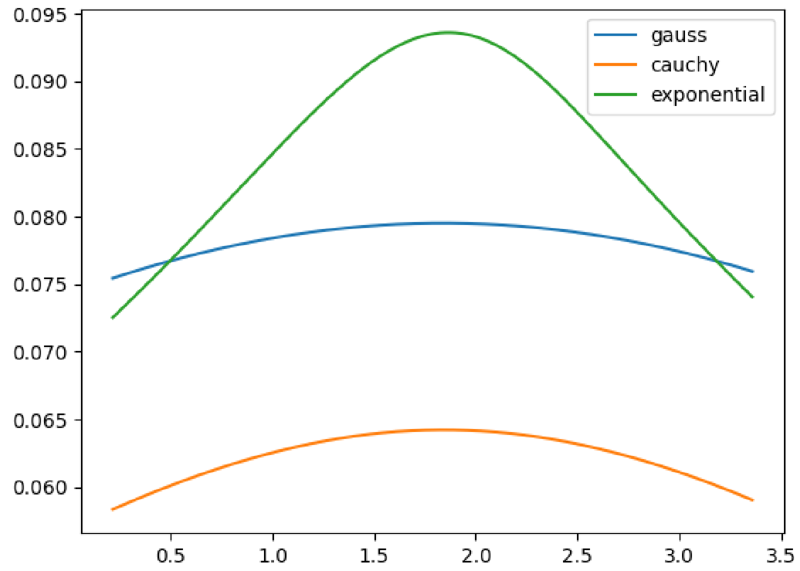

Рис. 1.5.1. Графики аппроксимирующих функций плотностей для h = 5

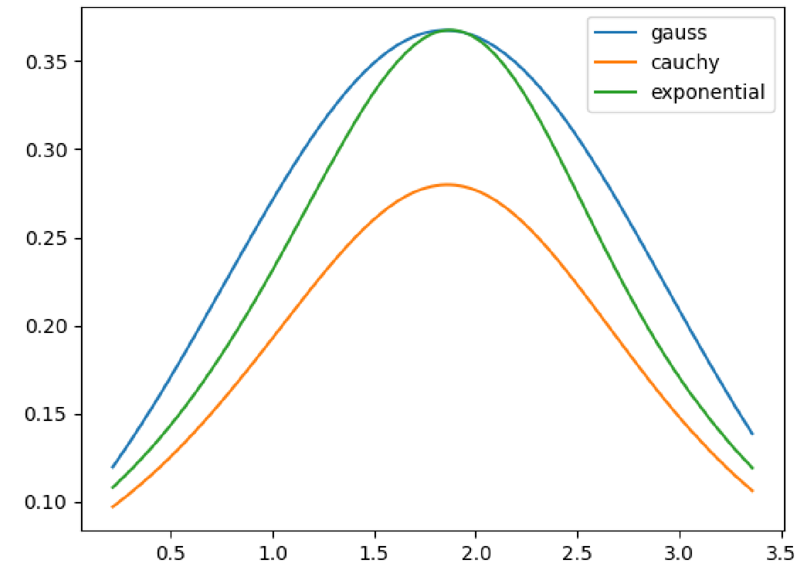

Рис. 1.5.2. Графики аппроксимирующих функций плотностей для h = 1

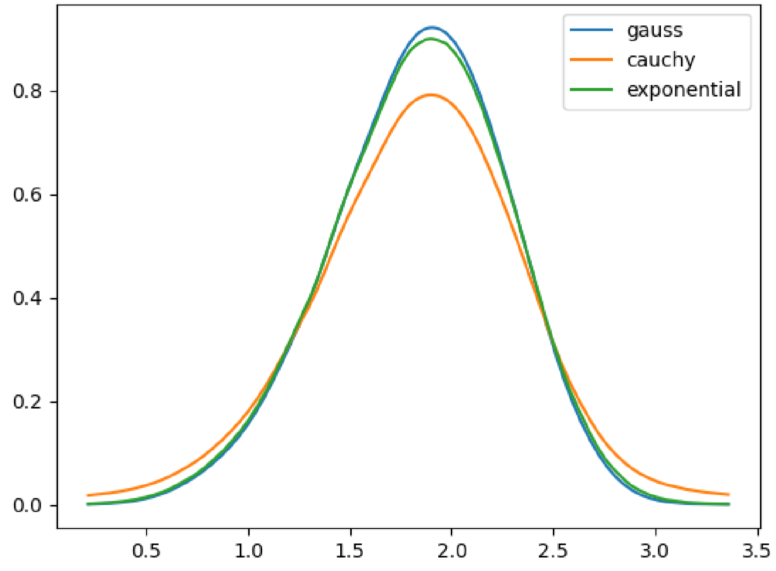

Рис. 1.5.3. Графики аппроксимирующих функций плотностей для h = 0,1

1.3. Точеные оценки

Для нахождения характеристик были использованы следующие формулы:

Центральный момент порядка k – $\mu_k = M[(X - MX)^k]$

Асимметрия – $\gamma_1 = \frac{\mu_3}{\sigma^3}$

Эксцесса – $\gamma_2 = \frac{\mu_4}{\sigma^4} - 3$

Интерквантильный промежуток – $\mu_1 - \sigma * k(n, P, Q);\ \mu_1 + \sigma * k(n, P, Q)$, где k = 1.9602111525053565 (коэффициент из таблицы).

Результаты для подвыборок и выборки в табличном и графическом виде:

Объём подвыборки	xmean	xmed	xave	var	m3	m4	As	Ex
1920.0	1.849	1.874	1.634	0.181	-0.024	0.095	-8.094789	-0.1
1920.0	1.848	1.871	1.849	0.177	-0.021	0.093	-7.574073	-0.032
1920.0	1.845	1.879	1.685	0.162	-0.02	0.076	-9.408382	-0.104
1920.0	1.81	1.838	1.855	0.178	-0.017	0.093	-6.028634	-0.065
1920.0	1.844	1.862	1.827	0.182	-0.023	0.098	-7.630336	-0.041
1920.0	1.836	1.857	1.666	0.179	-0.017	0.089	-6.028634	-0.191
1920.0	1.844	1.873	1.689	0.183	-0.02	0.094	-6.526896	-0.193
1920.0	1.851	1.874	1.84	0.181	-0.022	0.097	-7.420223	-0.039
1920.0	1.848	1.868	1.82	0.175	-0.013	0.088	-4.851312	-0.127
1920.0	1.828	1.845	1.673	0.174	-0.021	0.091	-7.972629	0.006
19200	1.84	1.864	1.788	0.177	-0.02	0.091	-7.213403	-0.095

Мода: 2.11

Интерквартильный промежуток: [1.015 , 2.665]

Рис. 1.6. Точечные оценки характеристик

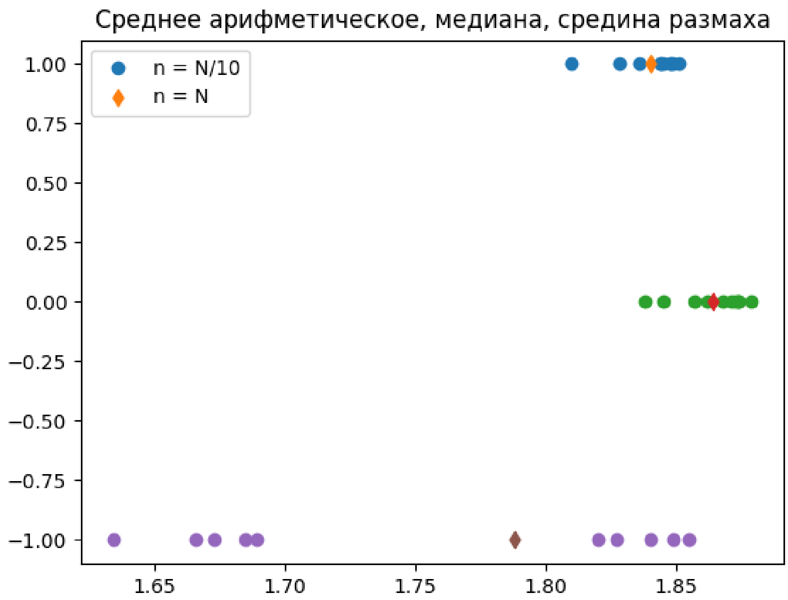

Рис. 1.7.1. Графическое представление оценок м. о.

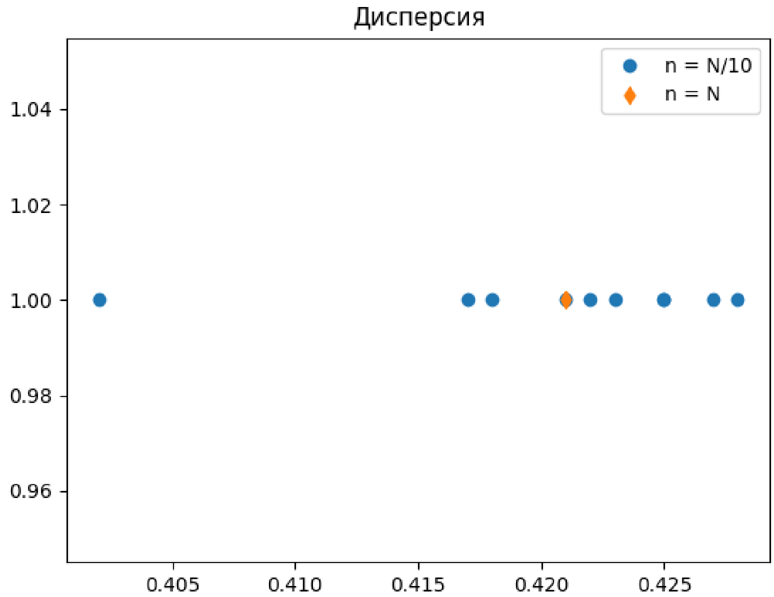

Рис. 1.7.2. Графическое представление оценок дисперсии

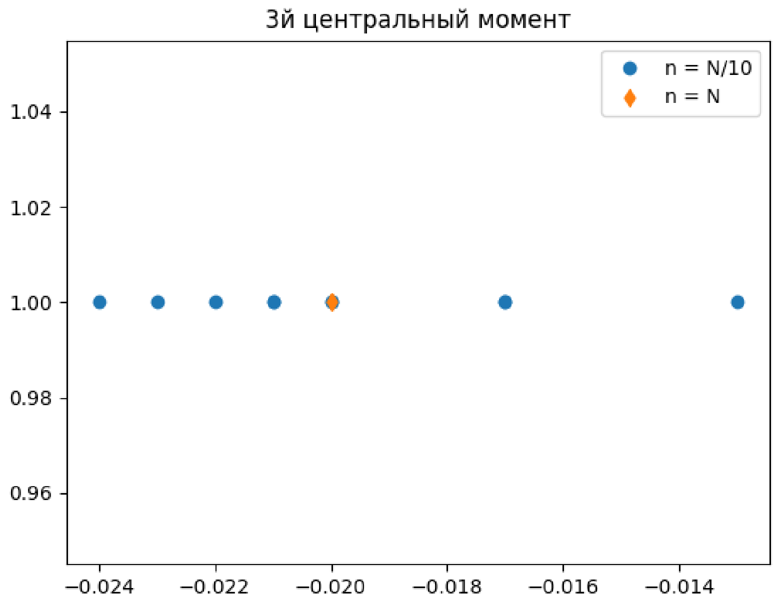

Рис. 1.7.3. Графическое представление оценок 3 ц. м.

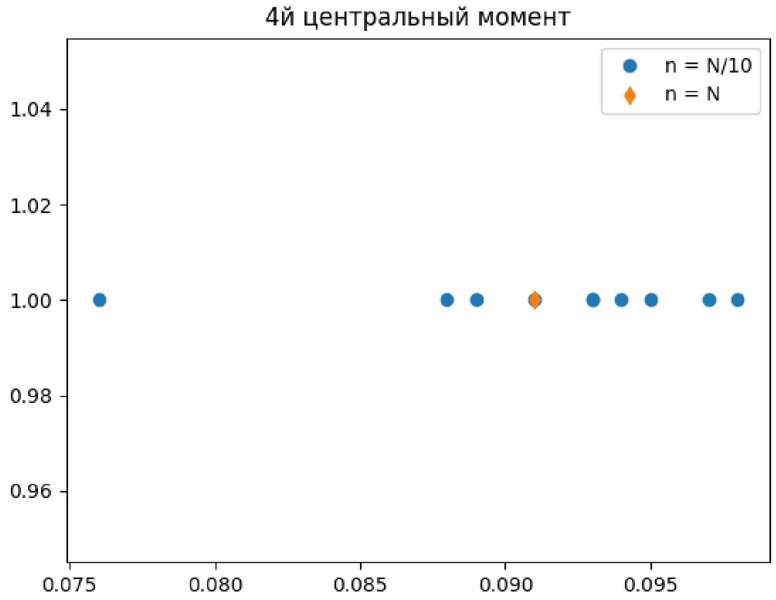

Рис. 1.7.4. Графическое представление оценок 4 ц. м.

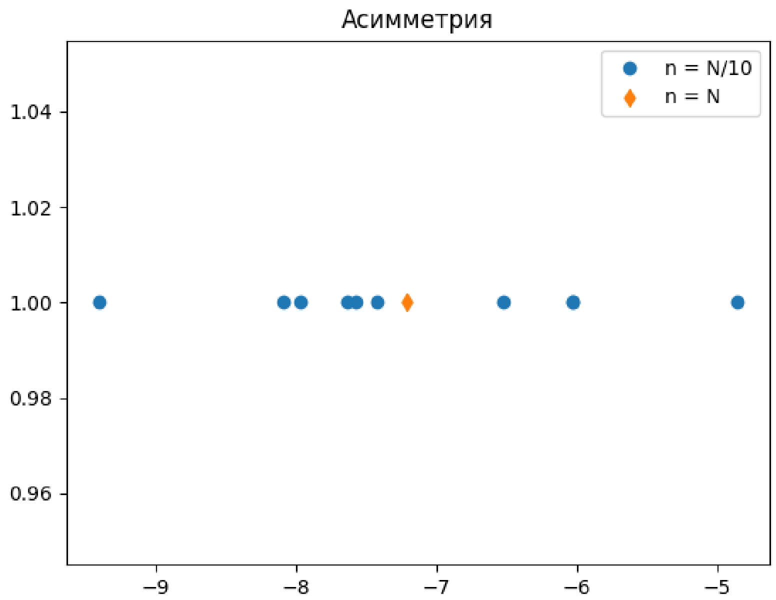

Рис. 1.7.5. Графическое представление оценок асимметрии

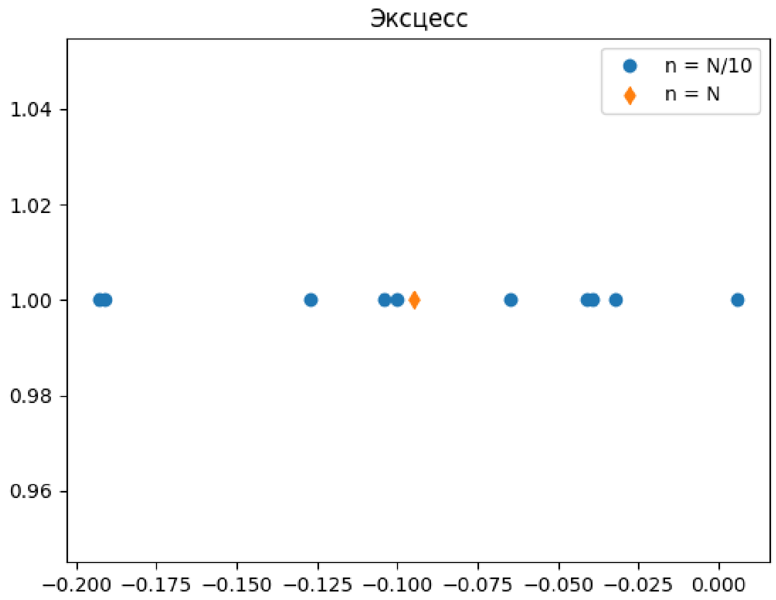

Рис. 1.7.6. Графическое представление оценок эксцесса

1.4. Интервальные оценки

Доверительную вероятность примем за Q = 0,8.

Математическое ожидание будем оценивать по следующей формуле: $[\bar{x} - \frac{k*\sigma}{\sqrt{n}}; \bar{x} + \frac{k*\sigma}{\sqrt{n}}]$, где k = 1.281639766840975 для полной выборки и k = 1.2824349652798663 для частичной - $\frac{1+Q}{2} * 100\%$-ная квантиль Стьюдента.

Дисперсию будем оценивать по следующей формуле: $[\frac{\sigma^2*(n-1)}{k_1}, \frac{\sigma^2*(n-1)}{k_2}]$, где k1 и k2 - $\frac{1+Q}{2} * 100\%$-ная и $\frac{1-Q}{2} * 100\%$-ная квантили распределения хи-квадрат соответственно. Для их расчёта был использован Microsoft Excel и формула =ХИ2.ОБР(вероятность; степеней свободы):

k1 = 19450.54961

k2 = 19450.54961

Аналогично рассчитали параметры для подвыборки.

Интерквантильный промежуток будем оценивать для P = 0,95, как по всей выборке при помощи непараметрических толерантных пределов, так и по подвыборке при помощи толерантных пределов.

Результаты:

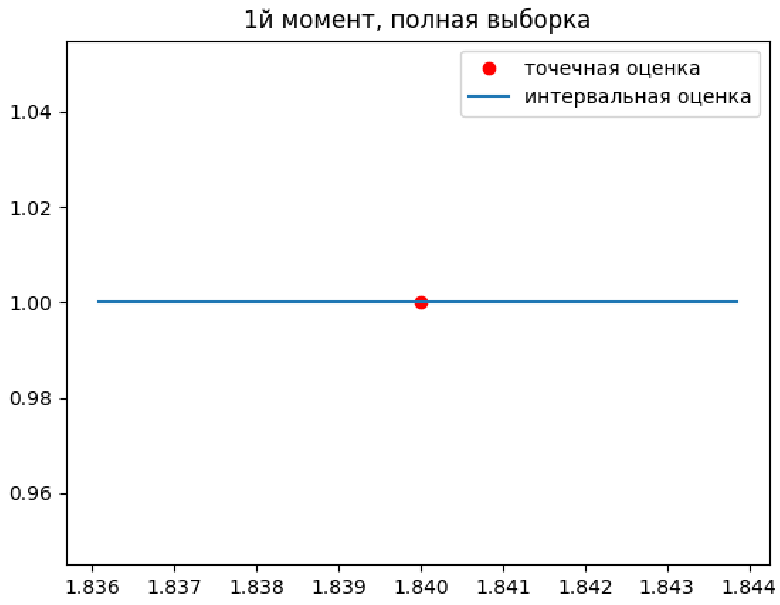

Рис. 1.8.1. Сравнение точечной и интервальной оценок м. о. для полной выборки

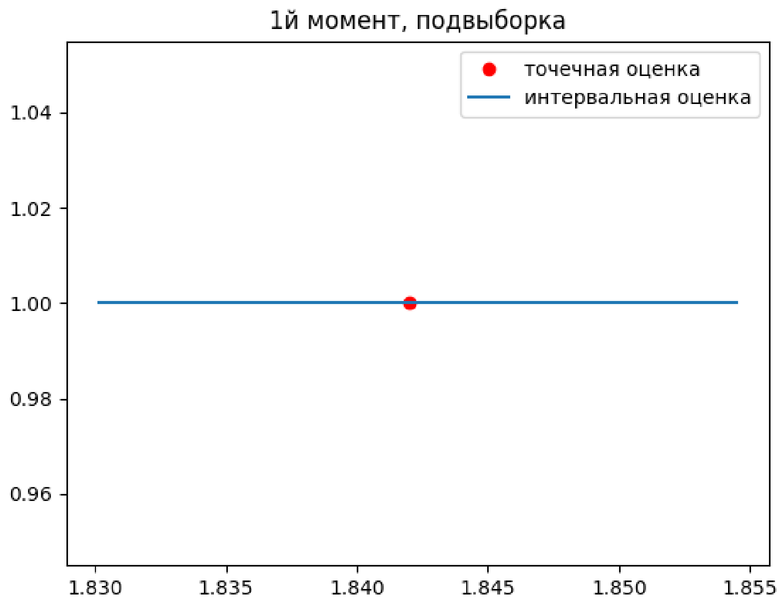

Рис. 1.8.2. Сравнение точечной и интервальной оценок м. о. для подвыборк

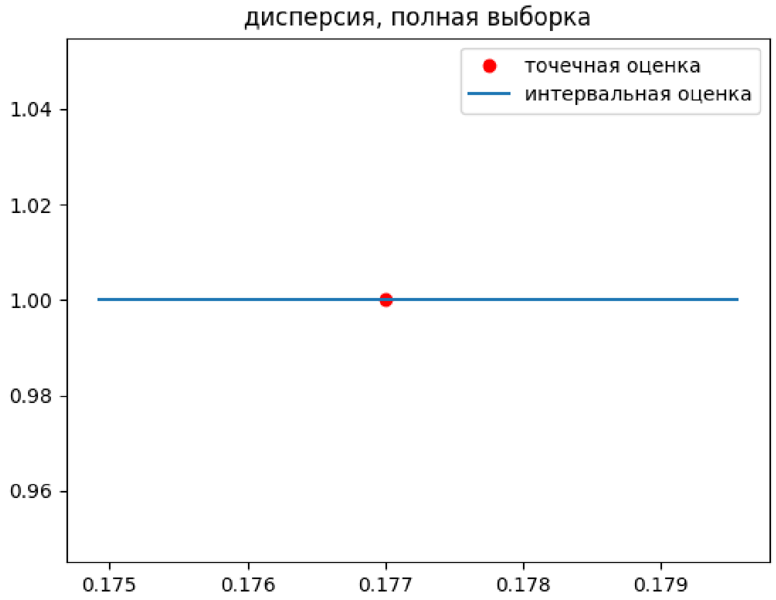

Рис. 1.9.1. Сравнение точечной и интервальной оценок дисперсии для полной выборки

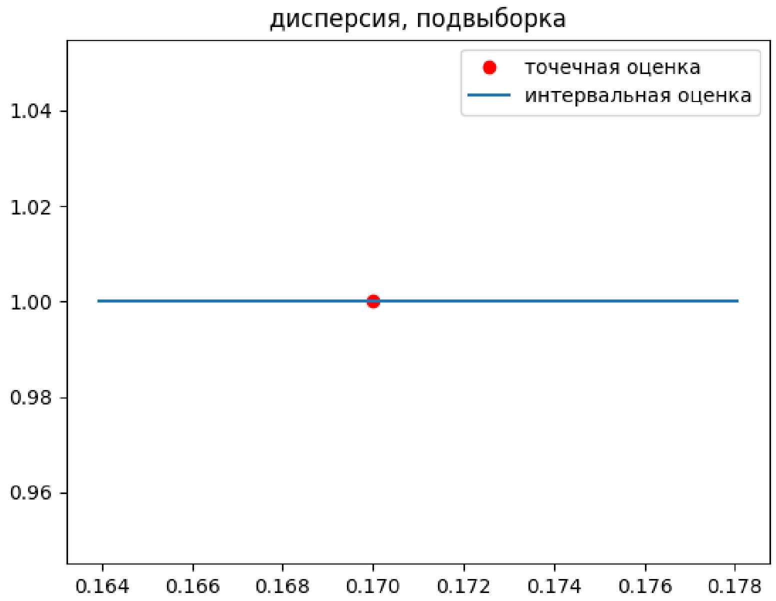

Рис. 1.9.2. Сравнение точечной и интервальной оценок дисперсии для подвыборки

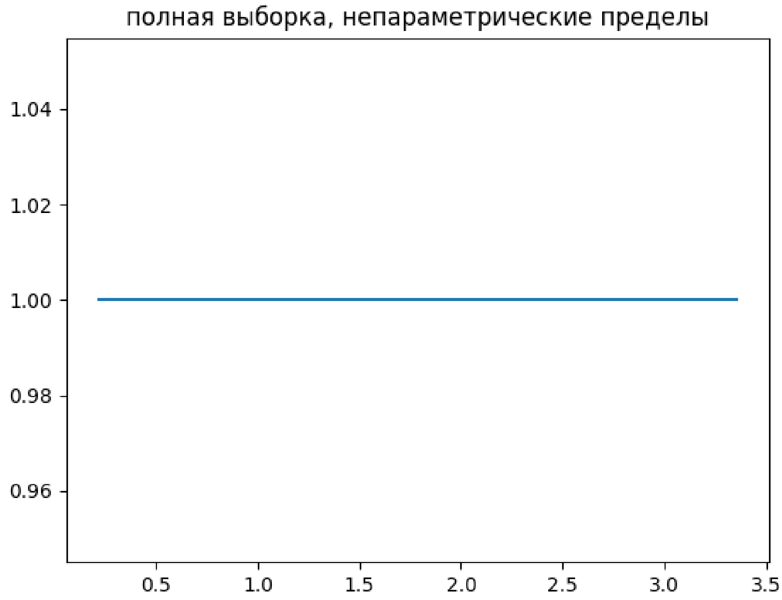

Рис. 1.10.1. Оценка интерквантильного промежутка по полной выборке

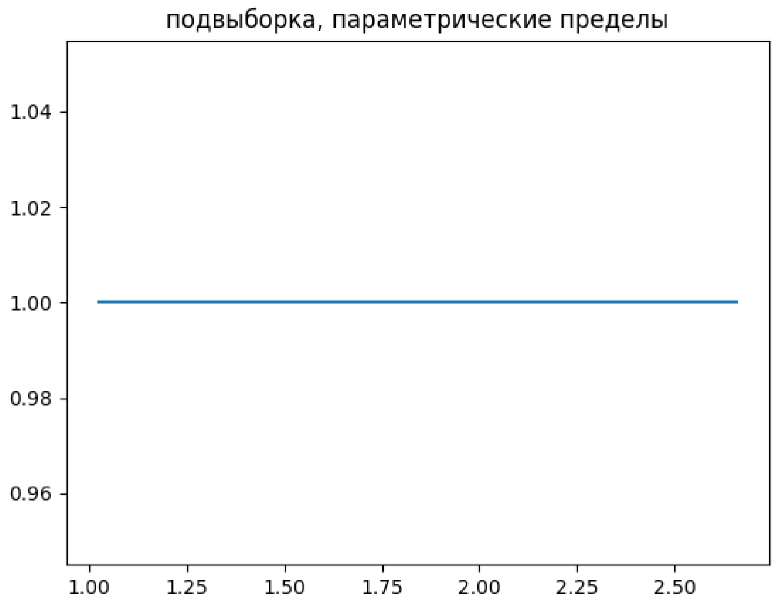

Рис. 1.10.2. Оценка интерквантильного промежутка по подвыборке

Часть 2 – Идентификация закона и параметров распределения

2.1. Начальный выбор распределения

Сравним получившееся распределение для идентификации со следующими распределениями: гамма, логнормальное и Вейбулла. Из полученных выше данных: асимметрия ≈ -7,2. и эксцесс ≈ -0,1. Благодаря отрицательному значению асимметрии, можно отмести большое количество распределений, например, нормальное.

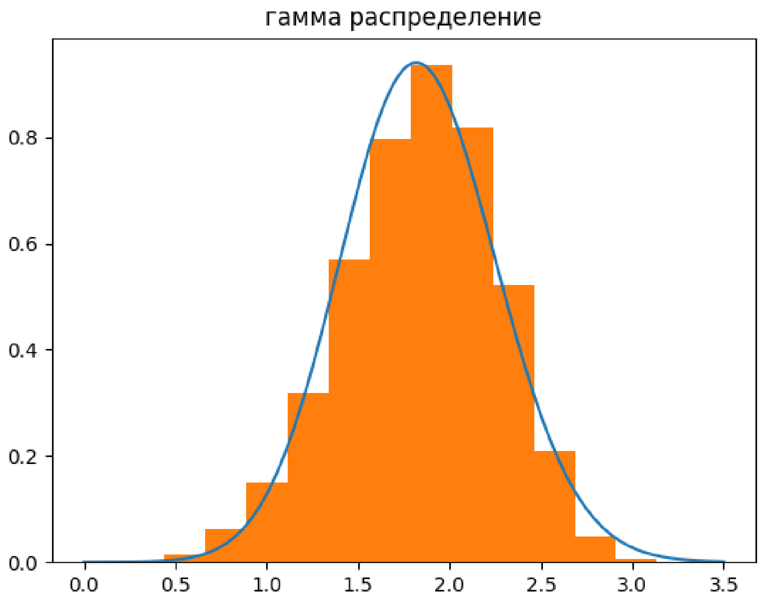

Рис. 2.1.1. Сравнение гистограммы и плотности гамма распределения

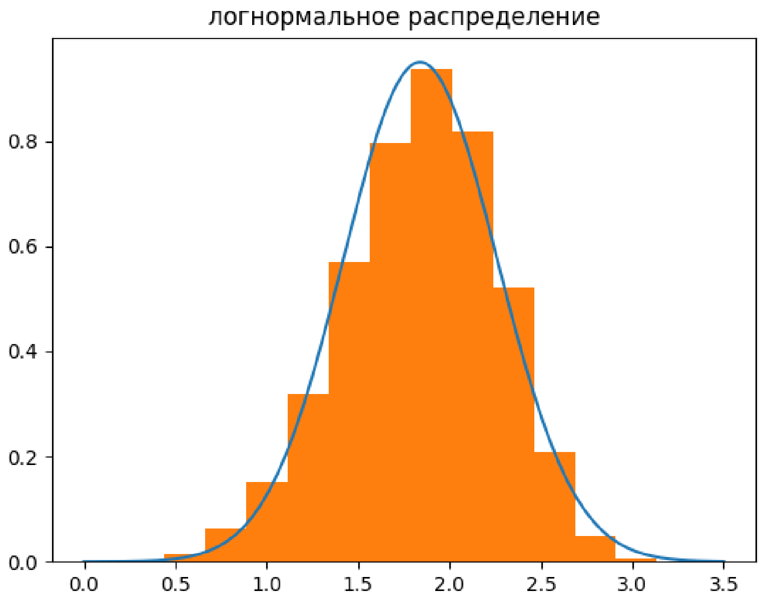

Рис. 2.1.1. Сравнение гистограммы и плотности логнормального распределения

Рис. 2.1.1. Сравнение гистограммы и плотности распределения Вейбулла

2.2. Определение параметров теоретических распределений`

Определим параметры выбранных выше теоретических распределений двумя способами:

1) с помощью метода моментов. Для этого будем приравнять эмпирические моменты (оценки теоретических) к самим теоретическим и по ним оценивать неизвестные параметры распределения путём решения систем уравнений.

Для гамма:

Математическое ожидание: $k\theta$

Дисперсия: $k\theta^2$

Для логнормального:

Математическое ожидание: $e^{\mu+\sigma^2/2}$

Дисперсия: $(e^{\sigma^2}-1)e^{2\mu+\sigma^2}$

Для Вейбулла:

Математическое ожидание:

$$\lambda\Gamma\left(1+\frac{1}{k}\right)$$

Дисперсия:

$$\lambda^2\Gamma\left(1+\frac{2}{k}\right)-\mu^2$$

2) с помощью метода максимального правдоподобия. Для этого воспользуемся функциями .fit() библиотеки Python scipy.stats.

Как видно из результатов методов (рисунок 2.2.), графиков сравнения плотностей (рисунки 2.1.) и графиков сравнения функций (рисунки 2.3), для распределения Вейбулла методы дали приблизительно одинаковые параметры, в то время как для остальных расхождения достаточно сильные.

```
гамма:
443.21329639889206 0.01999585174591994
метод моментов: a = 443.21329639889206, scale = 0.01999585174591994
ммп: a = 370.53147183229044, scale = 0.022069396458606645
логнормальное:
метод моментов: a = 0.35197068533337295, scale = 0.6226576539289345
ммп: a = 0.005193360943465061, scale = 80.89951518642067
Вейбулла:
метод моментов: a = 5.0, scale = 2.005634035688463
ммп: a = 5.0076417973578184, scale = 2.0036777182293193
```

Рис. 2.2. Сравнение метода оценок параметров

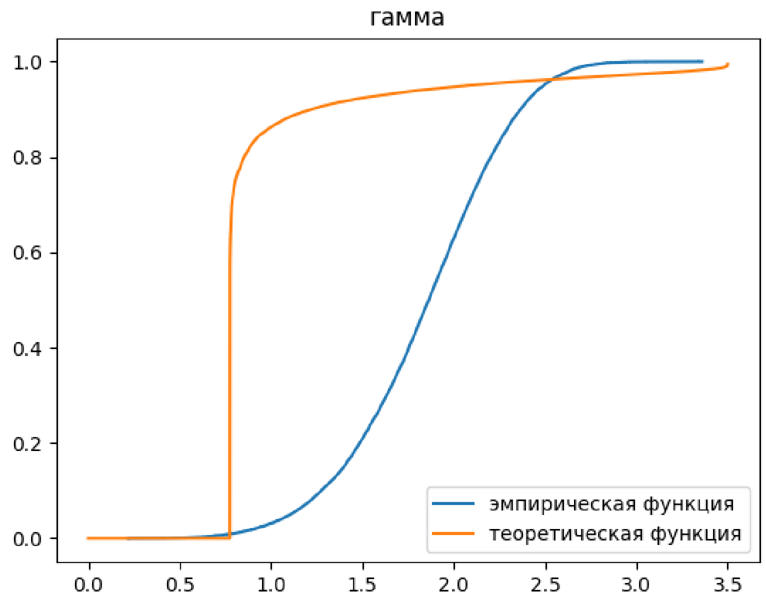

Рис. 2.3.1. Сравнение графиков теоретической и эмпирической функций для гамма распределения

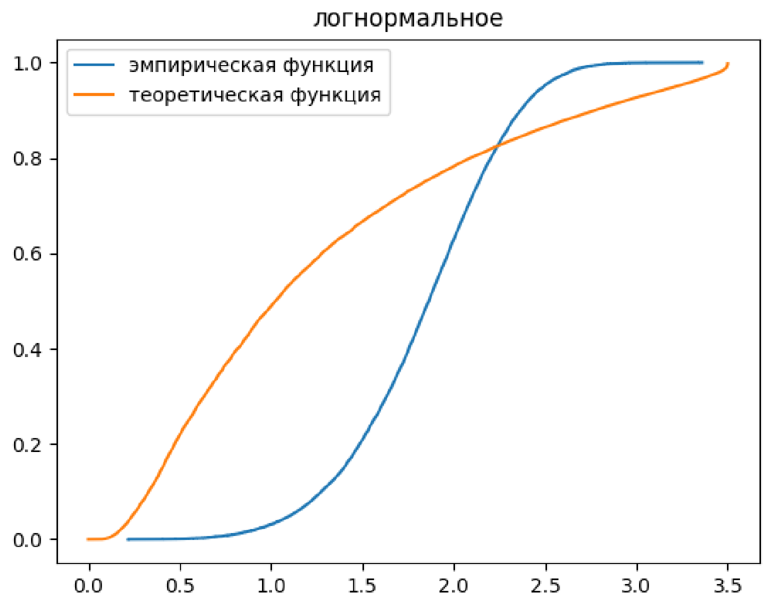

Рис. 2.3.2. Сравнение графиков теоретической и эмпирической функций для логнормального распределения.

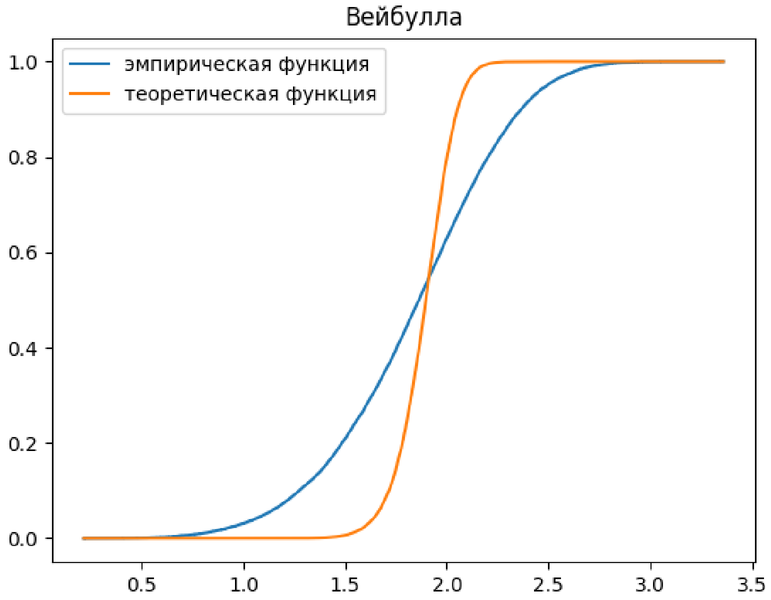

Рис. 2.3.3. Сравнение графиков теоретической и эмпирической функций для распределения Вейбулла.

Таким образом, большую точность даёт метод максимального правдоподобия.

2.3. Проверка гипотез

Проведём проверку гипотез о природе распределения. Воспользуемся тремя критериями: «хи-квадрат», Колмогорова-Смирнова, Мизеса. Критерии реализуем самостоятельно.

Распределение	Гамма		Логнормальное		Вейбулла	
	Мет. мом.	ММП	Мет. мом.	ММП	Мет. мом.	ММП
Пар. 1	443.213	370.531	0.35	0.005	5.0	5.008
Пар. 2	0.020	0.022	0.62	80.900	2.006	2.004
Хи-квадрат - статистика	4282.882	20998.429	2801.579	3558.824	6.364	6.188
Хи-квадрат – крит. знач.	19033,89					
Хи-квадрат - вывод	+	–	+	+	+	+

Колм.-Смирнова - статистика	0.477	0.970	0.0254	0.1758	0.0065	0.0078
Колм.-Смирнова – крит. знач.	0.0098					
Колм.-Смирнова - вывод	–	–	–	–	+	+
Мизеса - статистика	0.612	0.643	0.381	0.290	0.030	0.025
Мизеса – крит. знач.	0.2415					
Мизеса – вывод	–	–	–	–	+	+

Вывод

В результате исследования выборки был определен наиболее вероятный вид распределения – Вейбулла.

Приложение

Листинг

my_main.py

```
import random
from point1_2 import kernal_res
from point1_3 import point_res
from point1_4 import interval_res
from point2_1 import compare_with_theory_plot
from point2_2 import compare_with_theory_params
from point2_3 import check_hypothsis

with open("Task_2a.txt") as f:
    n = int(f.readline().split(" ")[2])
    x = f.readline().split(" ")
    x.pop()
    for i in range(len(x)):
        x[i] = float(x[i])


# Получение массива подвыборок из выборки x
def get_subselection(x):
    x_sub = []
    x_rand = x
    random.shuffle(x_rand)
    final = int(n / 10 + 9)
    for i in range(0, 10):
        t = []
        for j in range(0, final):
            t.append(x_rand[j + (int(n / 10) - 1) * i])
        x_sub.append(t)
    return x_sub


if __name__ == "__main__":
    # 1

    # 1.1
    x_sub = get_subselection(x)
    # 1.2
    kernal_res((x))
    # 1.3
    asym, ex = point_res(x, x_sub)
    # 1.4
    interval_res(x, x_sub)

    # 2

    # 2.1
    compare_with_theory_plot(x)
    # 2.2
    P, MMP = compare_with_theory_params(x, ex)
    # 2.3
    check_hypothsis(x, P, MMP)
```

point1_2.py

```
# Получение ядерного оценивания
def kernel(x, h):
    # Вложенные функции для получения
    def Gauss(p):
        return pow(e, (-pow(p, 2) / 2)) / sqrt(2 * pi)

    def Cauchy(p):
        return 1 / (pi * (1 + pow(p, 2)))

    def Exponential(p):
        return pow(e, -abs(p)) / 2

    x_sorted = sorted(x)
    n = len(x)
    nh = n * h
    pkde_gauss, pkde_cauchy, pkde_exponential = np.zeros(n), np.zeros(n),
np.zeros(n)
    for i in range(0, n):
        print(i)
        t_gauss, t_cauchy, t_exponential = 0, 0, 0
        for j in range(0, n):
            diff = x_sorted[i] - x_sorted[j]
            t_gauss += Gauss(diff / h)
            t_cauchy += Cauchy(diff / h)
            t_exponential += Exponential(diff / h)
        pkde_gauss[i] = t_gauss / nh
        pkde_cauchy[i] = t_cauchy / nh + 0.001
        pkde_exponential[i] = t_exponential / nh
    return x_sorted, [pkde_gauss, pkde_cauchy, pkde_exponential]

def kernal_res(x):
    plot_hist(x)
    x_sorted, PKDE = kernel(x, 0.1)
    plot_sdf(x_sorted)
    plot_kernal(x_sorted, PKDE)
```

point1_3.py

```
import matplotlib.pyplot as plt
import random
import numpy as np
from prettytable import PrettyTable
import statistics as stat

def get_subselection(x):
    x_sub = []
    x_rand = x
    n = len(x)
    random.shuffle(x_rand)
    final = int(n / 10 + 9)
    for i in range(0, 10):
        t = []
        for j in range(0, final):
            t.append(x_rand[j + (int(n / 10) - 1) * i])
        x_sub.append(t)
    return x_sub

# Получение момента
def calculate_moment(x, degree, avr):
```

```python
    result = 0.0
    for i in range(0, len(x)):
        result += (x[i] - avr) ** degree
    return round(result / len(x), 3)

# Построение таблицы
def point_table(x):
    table = PrettyTable()
    table.add_column("Объём подвыборки", ns)
    table.add_column("xmean", x_mean)
    table.add_column("xmed", x_med)
    table.add_column("xave", x_ave)
    table.add_column("var", var2)
    table.add_column("m3", m3)
    table.add_column("m4", m4)
    table.add_column("As", asym)
    table.add_column("Ex", ex)
    print(table)
    print("Мода: ", round(stat.mode(x), 3))
    print("Интерквартильный промежуток: [", lower_bound, ", ", upper_bound,
"]")

# Построение графиков
def plot_point():
    y = [np.zeros(10), np.ones(10), np.zeros(10) - 1]
    plt.title("Среднее арифметическое, медиана, средина размаха")
    plt.scatter(x_mean[:10], y[1], label='n = N/10')
    plt.scatter(x_mean[10], y[1][0], label='n = N', marker='d')
    plt.scatter(x_med[:10], y[0])
    plt.scatter(x_med[10], y[0][0], marker='d')
    plt.scatter(x_ave[:10], y[2])
    plt.scatter(x_ave[10], y[2][0], marker='d')
    plt.legend()
    plt.show()

    plt.title("Дисперсия")
    plt.scatter(var[:10], y[1], label='n = N/10')
    plt.scatter(var[10], y[1][0], label='n = N', marker='d')
    plt.legend()
    plt.show()

    plt.title("3й центральный момент")
    plt.scatter(m3[:10], y[1], label='n = N/10')
    plt.scatter(m3[10], y[1][0], label='n = N', marker='d')
    plt.legend()
    plt.show()

    plt.title("4й центральный момент")
    plt.scatter(m4[:10], y[1], label='n = N/10')
    plt.scatter(m4[10], y[1][0], label='n = N', marker='d')
    plt.legend()
    plt.show()

    plt.title("Асимметрия")
    plt.scatter(asym[:10], y[1], label='n = N/10')
    plt.scatter(asym[10], y[1][0], label='n = N', marker='d')
    plt.legend()
    plt.show()

    plt.title("Эксцесс")
    plt.scatter(ex[:10], y[1], label='n = N/10')
    plt.scatter(ex[10], y[1][0], label='n = N', marker='d')
    plt.legend()
```

```
    plt.show()

# Получение точечной оценки
def point_estimates(x, x_sub):
    global ns, x_mean, x_med, x_ave, var, var2, asym, ex, m3, m4,
lower_bound, upper_bound, y
    n = len(x_sub)
    ns = []
    x_mean, x_med, x_ave = [], [], []  # Среднее арифметическое, медиана и
середина размаха
    var, var2 = [], []  # Дисперсия и её квадрат
    asym = []  # Асимметрия
    ex = []  # Эксцесс
    m3, m4 = [], []  # Третий и четвертый центральные моменты
    for i in range(0, n + 1):
        if i != 10:
            u = x_sub[i]
            ns.append(len(x) / 10)
        else:
            u = x
            ns.append(len(x))
            lower_bound = round(stat.mean(u) - stat.stdev(u) *
1.9602111525053565, 3)
            upper_bound = round(stat.mean(u) + stat.stdev(u) *
1.9602111525053565, 3)
        x_mean.append(round(stat.mean(u), 3))
        x_med.append(round(stat.median(u), 3))
        x_ave.append(round((min(u) + max(u)) / 2, 3))
        var2.append(round(stat.variance(u), 3))
        var.append(round(var2[i] ** (1 / 2), 3))
        m3.append(calculate_moment(u, 3, x_mean[i]))
        asym.append(round(m3[i] / (calculate_moment(u, 2, x_mean[i]) ** 3 /
2), 6))
        m4.append(calculate_moment(u, 4, x_mean[i]))
        ex.append(round(m4[i] / calculate_moment(u, 2, x_mean[i]) ** 2 - 3,
3))
    return [x_mean[0], x_mean[10], var2[0], var2[10]]

def point_res(x, x_sub):
    point_estimates(x, x_sub)
    #plot_point()
    #point_table(x)
    return asym[10], ex[10]
```

point1_4.py

```
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import statistics as stat
from point1_3 import point_estimates


# Получение сигма
def calculate_sigma(x, ave):
    s = 0.0
    for i in x:
        s += (i - ave) ** 2
    s /= round((len(x) - 1), 3)
    s = s ** (1 / 2)
    return s
```

```
# Построение таблицы
def interval_table():
    table = PrettyTable()
    table.add_column(" ", ns)
    table.add_column("Параметрические толерантные пределы", bounds_param)
    table.add_column("Непараметрические пределы", bounds_nonparam)
    table.add_column("1й момент", m[0])
    print(table)

# Построение графиков
def plot_interval():
    plt.title("1й момент, полная выборка")
    plt.scatter(points[1], [1], color="red", label="точечная оценка")
    plt.plot(m[0][10], [1, 1], label="интервальная оценка")
    plt.legend()
    plt.show()

    plt.title("1й момент, подвыборка")
    plt.scatter(points[0], [1], color="red", label="точечная оценка")
    plt.plot(m[0][0], [1, 1], label="интервальная оценка")
    plt.legend()
    plt.show()

    plt.title("дисперсия, полная выборка")
    plt.scatter(points[3], [1], color="red", label="точечная оценка")
    plt.plot(m[1][10], [1, 1], label="интервальная оценка")
    plt.legend()
    plt.show()

    plt.title("дисперсия, подвыборка")
    plt.scatter(points[2], [1], color="red", label="точечная оценка")
    plt.plot(m[1][0], [1, 1], label="интервальная оценка")
    plt.legend()
    plt.show()

    plt.title("полная выборка, непараметрические пределы")
    plt.plot(bounds_nonparam[10], [1, 1])
    plt.show()

    plt.title("подвыборка, параметрические пределы")
    plt.plot(bounds_param[0], [1, 1])
    plt.show()

# Получение интервальной оценки
def interval_estimates(x, x_sub):
    global ns, bounds_param, bounds_nonparam, m, points

    # Получение моментов
    def get_1st_moment(x):
        ave = stat.mean(x)
        s = calculate_sigma(x, ave)
        n = len(x)
        if n == 19200:
            k = 1.281639766840975
        else:
            k = 1.2824349652798663
        return [ave - k * pow(n, -0.5) * s, ave + k * pow(n, -0.5) * s]

    def get_2nd_moment(x):
        ave = stat.mean(x)
        s = calculate_sigma(x, ave)
        n = len(x)
        if n == 19200:
```

```
            k1 = 19450.54961
            k2 = 18948.30689

        else:
            k1 = 1998.810088
            k2 = 1840.046392
        return [pow(s, 2) * (n - 1) / k1, pow(s, 2) * (n - 1) / k2]

    # Получение параметрических толерантных пределов
    def get_param(x):
        ave = stat.mean(x)
        s = calculate_sigma(x, ave)
        lower_bound = float(ave - s * 1.9602111)
        upper_bound = ave + s * 1.9602111525053565
        return [round(lower_bound, 3), round(upper_bound, 3)]

    # Получение непараметрических толерантных пределов
    def get_nonparam(x):
        sorted_x = sorted(x)
        return [sorted_x[0], sorted_x[len(x) - 1]]

    inner_x_sub = x_sub
    points = point_estimates(x, x_sub)
    inner_x_sub.append(x)
    n = len(inner_x_sub)
    ns = []
    m = [[], []]
    bounds_param = []
    bounds_nonparam = []
    for i in range(n):
        if i == n - 1:
            ns.append(n)
            bounds_nonparam.append(get_nonparam(inner_x_sub[i]))
        else:
            ns.append(n / 10)
            bounds_nonparam.append("")
        bounds_param.append(get_param(inner_x_sub[i]))
        m[0].append(get_1st_moment(inner_x_sub[i]))
        m[1].append(get_2nd_moment(inner_x_sub[i]))
    print(m[0])
    print(m[1])


def interval_res(x, x_sub):
    interval_estimates(x, x_sub)
    interval_table()
    plot_interval()
```

point2_1.py

```
import matplotlib.pyplot as plt
import numpy as np
import math
from scipy.stats import gamma, lognorm, weibull_min


def compare_with_theory_plot(x):
    x_theory = np.linspace(0, 3.5, 1000)

    a, b, c = gamma.fit(x)
    d, e, f = lognorm.fit(x)
```

```
    k, h, l = weibull_min.fit(x)

    gam = gamma.pdf(x_theory, a, b, c)
    logn = lognorm.pdf(x_theory, d, e, f)
    w_min = weibull_min.pdf(x_theory, k, h, l)


    print(gamma.fit(x), " ; ", lognorm.fit(x), " ; ", weibull_min.fit(x))

    print("Медиана: ", l * pow(math.log(2, math.e), 1 / k))
    print("Мода: ", l * pow(k - 1, 1 / k) / pow(k, 1 / k))

    plt.title("гамма распределение")
    plt.plot(x_theory, gam)
    plt.hist(x, 14, density=True)
    plt.show()

    plt.title("логнормальное распределение")
    plt.plot(x_theory, logn)
    plt.hist(x, 14, density=True)
    plt.show()

    plt.title("распределение Вейбулла")
    plt.plot(x_theory, w_min)
    plt.hist(x, 14, density=True)
    plt.show()
```

point2_2.py

```
import matplotlib.pyplot as plt
from math import log, sqrt
import numpy as np
import statistics as stat

from scipy import special
from scipy.stats import gamma, lognorm, weibull_min, weibull_max


def moment_method_gamma(x, asym):
    global p_gamma
    k = (2 / asym) ** 2
    t = sqrt(var / k)
    r = "a = " + str(k) + ", scale = " + str(t)
    plt.title("гамма")
    x_sorted = sorted(x)
    y = np.linspace(0, 1, 19200)
    plt.step(x_sorted, y, label="эмпирическая функция")

    x_theor = np.linspace(0, 3.5, 19200)
    p_gamma = sorted(gamma.cdf(x, k, t))
    plt.plot(x_theor, p_gamma, label="теоретическая функция")
    plt.legend()
    plt.show()
    return r


def moment_method_lognorm(x):
    global p_lognorm
    c = log(med)
    s2 = log(med / mod)
    s = sqrt(abs(s2))
```

```
    r = "a = " + str(s) + ", scale = " + str(c)
    plt.title("логнормальное")
    x_sorted = sorted(x)
    y = np.linspace(0, 1, 19200)
    plt.step(x_sorted, y, label="эмпирическая функция")

    x_theor = np.linspace(0, 3.5, 19200)
    p_lognorm = sorted(lognorm.cdf(x, s, c))
    plt.plot(x_theor, p_lognorm, label="теоретическая функция")
    plt.legend()
    plt.show()
    return r


def moment_method_weibull(x):
    global p_weibull
    ks = np.arange(0.1, 10, 0.1)
    for k in ks:
        l1 = med / log(2) ** (1 / k)
        l2 = mean / special.gamma(1 + (1 / k))
        if abs(l2 - l1) < 0.002 and k == round((log(med / l1, log(2))) ** -1,
1):
            l = l1
            break
    r = "a = " + str(k) + ", scale = " + str(l)

    plt.title("Вейбулла")
    x_sorted = sorted(x)
    y = np.linspace(0, 1, 19200)
    plt.step(x_sorted, y, label="эмпирическая функция")

    x_theor = np.linspace(0, 3.5, 19200)
    p_weibull = weibull_min.cdf(x_theor, k, 0, l)
    plt.plot(x_sorted, p_weibull, label="теоретическая функция")
    plt.legend()
    plt.show()
    return r


def mmp_gamma(x):
    global p_mmp_gamma
    a, b, c = gamma.fit(x)
    r = "a = " + str(a) + ", scale = " + str(c)
    x_theor = np.linspace(0, 3.5, 19200)
    p_mmp_gamma = gamma.pdf(x_theor, a, b, c)
    plt.title("гамма распределение")
    plt.plot(x_theor, p_mmp_gamma)
    plt.hist(x, 14, density=True)
    plt.show()
    return r


def mmp_lognorm(x):
    global p_mmp_lognorm
    d, e, f = lognorm.fit(x)
    r = "a = " + str(d) + ", scale = " + str(f)
    x_theor = np.linspace(0, 3.5, 19200)
    p_mmp_lognorm = lognorm.pdf(x_theor, d, e, f)
    plt.title("логнормальное распределение")
    plt.plot(x_theor, p_mmp_lognorm)
    plt.hist(x, 14, density=True)
    plt.show()
    return r
```

```
def mmp_weibull(x):
    global p_mmp_weibull
    k, h, l = weibull_min.fit(x, floc=0)
    r = "a = " + str(k) + ", scale = " + str(l)
    x_theor = np.linspace(0, 3.5, 19200)
    p_mmp_weibull = weibull_min.pdf(x_theor, k, h, l)
    plt.title("распределение Вейбулла")
    plt.plot(x_theor, p_mmp_weibull)
    plt.hist(x, 14, density=True)
    plt.show()
    return r


def compare_with_theory_params(x, ex):
    global mean, var, mod, med
    mean = stat.mean(x)
    var = stat.variance(x)
    mod = stat.mode(x)
    med = stat.median(x)

    print("гамма:")
    print("метод моментов:", moment_method_gamma(x, ex))
    print("ммп:", mmp_gamma(x))

    print("логнормальное:")
    print("метод моментов:", moment_method_lognorm(x))
    print("ммп:", mmp_lognorm(x))

    print("Вейбулла:")
    print("метод моментов:", moment_method_weibull(x))
    print("ммп:", mmp_weibull(x), "\n")
    return [p_gamma, p_lognorm, p_weibull], [p_mmp_gamma, p_mmp_lognorm,
p_mmp_weibull]
```

point2_3.py

```
from scipy.stats import chi2_contingency, ks_2samp
from scipy.stats import gamma, lognorm, weibull_min

def chi_square(x, theor):
    observed = []
    for i in range(len(x)):
        if (x[i] * theor[i] == 0):
            observed.append([x[i], 0.000000000000001])
        else:
            observed.append([x[i], theor[i]])
    chi2, p, dof, expected = chi2_contingency(observed)
    print("chi_square: ", chi2, p)


def kolmogorov_smirnov(x, theor):
    ks_statistic, p_value = ks_2samp(x, theor)
    print("kolmogorov_smirnov: ",ks_statistic, p_value)


def von_mises(x, f):
    n = len(x)
    mises = 1 / (12 * len(x))
    for i in range(0, len(x)):
```

```python
        mises += (f[i] - (2 * i - 1) / (2 * n)) ** 2
    print("von_mises: ", mises / n)

def check_hypothsis(x, P, MMP):
    print("P: \n")
    for p in P:
        chi_square(x, p)
        kolmogorov_smirnov(x, p)
        von_mises(x, p)
        print("\n")
    print("MMP: \n")
    for mmp in MMP:
        chi_square(x, mmp)
        kolmogorov_smirnov(x, mmp)
        von_mises(x, mmp)
        print("\n")
```