Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра прикладной математики

Практическая работа

по дисциплине «Современные проблемы прикладной математики и наукоёмкого прикладного обеспечения»

МЕТОДЫ УСТОЙЧИВОГО ОЦЕНИВАНИЯ ПАРАМЕТРА СДВИГА РАСПРЕДЕЛЕНИЙ НЕПРЕРЫВНЫХ СЛУЧАЙНЫХ ВЕЛИЧИН

Группа: ПММ-42 Будник Светлана

Вариант: 10.B Самсонов Семён

Судеревская Ирина

Преподаватель Лисицин Даниил Валерьевич

Новосибирск, 2024

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Изучить методы устойчивого оценивания параметра сдвига распределений непрерывных случайных величин.

Вариант задания: 10.B – обобщённое распределение Лапласа со значением параметра

$n = 4$.

$$f(x, n) = f(x, 4) = e^{-|x|} \cdot \frac{|x|^3 + 6|x|^2 + 15|x| + 15}{96}.$$

Дисперсия: $\sigma^2 = 8$, коэффициент эксцесса: $\gamma_2 = \frac{3}{4}$.

2. ХОД РАБОТЫ

2.1. ПРОВЕРКА ГЕНЕРАЦИИ ЧИСТОГО, ЗАСОРЯЮЩЕГО И ЗАСОРЁННОГО РАСПРЕДЕЛЕНИЙ

2.1.1. Чистое распределение

Моделирование случайной величины распределения Лапласа происходит по следующей формуле:

$$x = ln\left(\frac{\prod_{i:r_i \le \frac{1}{2}} 2r_i}{\prod_{i:r_i > \frac{1}{2}} 2(1-r_i)}\right),$$

где $r_i, i = \{1, \dots, n\} = \{1,2,3,4\}$ – реализация случайной величины, равномерно распределённой на интервале $(0; 1)$; если нет ни одного удовлетворяющего условия элемента в произведении, то произведение равно 1.

Для проверки корректности генератора чистого распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблица 1).

Таблица 1. Сравнение теоретических и выборочных характеристик для чистого распределения

Объём выборки	Мат ожидание		Медиана		Дисперсия		Асимметрия		Эксцесс	
	Т	П	Т	П	Т	П	Т	П	Т	П
10000	0	-0.00238	0	-0.00129	8	8.06043	0	-0.02783	0.75	0.69796
100000	0	0.00082	0	0.00538	8	8.01069	0	0.00379	0.75	0.73337
1000000	0	0.00096	0	0.00026	8	8.01254	0	0.00567	0.75	0.73396

2.1.2. Засоряющее распределение

Функция плотности случайной величины засоряющего распределения Лапласа принимает следующий вид:

$$f(x) = e^{-\left|\frac{x-\theta}{\lambda}\right|} \cdot \frac{\left|\frac{x-\theta}{\lambda}\right|^3 + 6\left|\frac{x-\theta}{\lambda}\right|^2 + 15\left|\frac{x-\theta}{\lambda}\right| + 15}{96\,\lambda}.$$

Моделирование случайной величины засоряющего распределения Лапласа происходит по следующей формуле:

$$x = \theta + \lambda\, ln\left(\frac{\prod_{i:r_i \le \frac{1}{2}} 2r_i}{\prod_{i:r_i > \frac{1}{2}} 2(1-r_i)}\right).$$

Теоретические значения характеристик для засоряющего распределения находятся следующим образом:

1. Матожидание: $\mu = \theta$;
2. Медиана: $M = \theta$;
3. Коэффициент асимметрии: $\gamma_1 = 0$;
4. Коэффициент эксцесса: $\gamma_2 = \frac{3}{4}$.

Для проверки корректности генератора засоряющего распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблицы 2 и 3).

Таблица 2. Сравнение теоретических и выборочных характеристик для засоряющего распределения ($\lambda = 2, \theta = 4$).

Объём выборки	Мат ожидание		Медиана		Дисперсия		Асимметрия		Эксцесс	
	Т	П	Т	П	Т	П	Т	П	Т	П
10000	4	4.07849	4	4.04918	32	32.09559	0	-0.00064	0.75	0.68993
100000	4	4.01401	4	4.02281	32	31.77302	0	0.01837	0.75	0.76731
1000000	4	4.00102	4	4.00004	32	32.00511	0	0.00548	0.75	0.74395

Таблица 3. Сравнение теоретических и выборочных характеристик для засоряющего распределения ($\lambda = 2, \theta = 0$).

Объём выборки	Мат ожидание		Медиана		Дисперсия		Асимметрия		Эксцесс	
	Т	П	Т	П	Т	П	Т	П	Т	П
10000	0	0.07849	0	0.04918	32	32.09559	0	-0.00064	0.75	0.68993
100000	0	0.01401	0	0.02281	32	31.77302	0	0.01837	0.75	0.76731
1000000	0	0.00102	0	0.00004	32	32.00511	0	0.00548	0.75	0.74395

2.1.3. Засорённое распределение

Функция плотности случайной величины засорённого распределения Лапласа принимает следующий вид:

$$g(x) = (1 - \varepsilon)f(x) + \varepsilon h(x)$$

Моделирование случайной величины засорённого распределения Лапласа происходит по следующему алгоритму:

1. Сгенерировать случайное число r, равномерно распределённое на интервале $(0; 1)$. Если $r \leq 1 - \varepsilon$, перейти на шаг 2, иначе перейти на шаг 3.
2. Сгенерировать случайное число с плотностью $f(x)$ (чистое распределение). Оно и будет искомым числом.
3. Сгенерировать случайное число с плотностью $h(x)$ (засоряющее распределение). Оно и будет искомым числом.

Теоретические характеристики засорённого распределения находятся следующим образом:

$$\mu = (1 - \varepsilon)\mu_1 + \varepsilon\mu_2,$$

$$D = (1 - \varepsilon)(\mu_1^2 + D_1) + \varepsilon(\mu_2^2 + D_2) - \mu^2,$$

$$\gamma_1 = \frac{1}{D^{1.5}}[(1-\varepsilon)\{(\mu_1-\mu)^3 + 3(\mu_1-\mu)D_1\} + \\ + \varepsilon\{(\mu_2-\mu)^3 + 3(\mu_2-\mu)D_2\}],$$

$$\gamma_2 = \frac{1}{D^2}\{(1-\varepsilon)[(\mu_1-\mu)^4 + 6(\mu_1-\mu)^2 D_1 + D_1^2(\gamma_{21}+3)] + \\ + \varepsilon[(\mu_2-\mu)^4 + 6(\mu_2-\mu)^2 D_2 + D_2^2(\gamma_{22}+3)]\} - 3.$$

Для проверки корректности генератора засорённого распределения, сгенерируем несколько выборок разного объёма и сравним выборочные характеристики полученных выборок с их теоретическими значениями (таблицы 4 и 5). Коэффициент зашумления $\varepsilon = 0.1$.

Таблица 4. Сравнение теоретических и выборочных характеристик для засорённого распределения ($\lambda = 2, \theta = 4$).

Объём выборки	Мат ожидание		Дисперсия		Асимметрия		Эксцесс	
	Т	П	Т	П	Т	П	Т	П
10000	0.4	0.43938	11.84	12.28219	0.74933	0.77483	3.22434	3.08007
100000	0.4	0.39444	11.84	11.75703	0.74933	0.76140	3.22434	3.34871
1000000	0.4	0.39887	11.84	11.81183	0.74933	0.75457	3.22434	3.21341

Таблица 5. Сравнение теоретических и выборочных характеристик для засорённого распределения ($\lambda = 2, \theta = 0$).

Объём выборки	Мат ожидание		Дисперсия		Асимметрия		Эксцесс	
	Т	П	Т	П	Т	П	Т	П
10000	0	0.01578	10.4	10.55490	0	0.01901	2.54734	2.40814
100000	0	-0.00456	10.4	10.36931	0	0.01925	2.54734	2.58238
1000000	0	-0.00027	10.4	10.38299	0	0.01286	2.54734	2.49837

Графики функций плотности получившихся распределений при объёме выборки 1000000 изображены на рисунках 1 и 2.

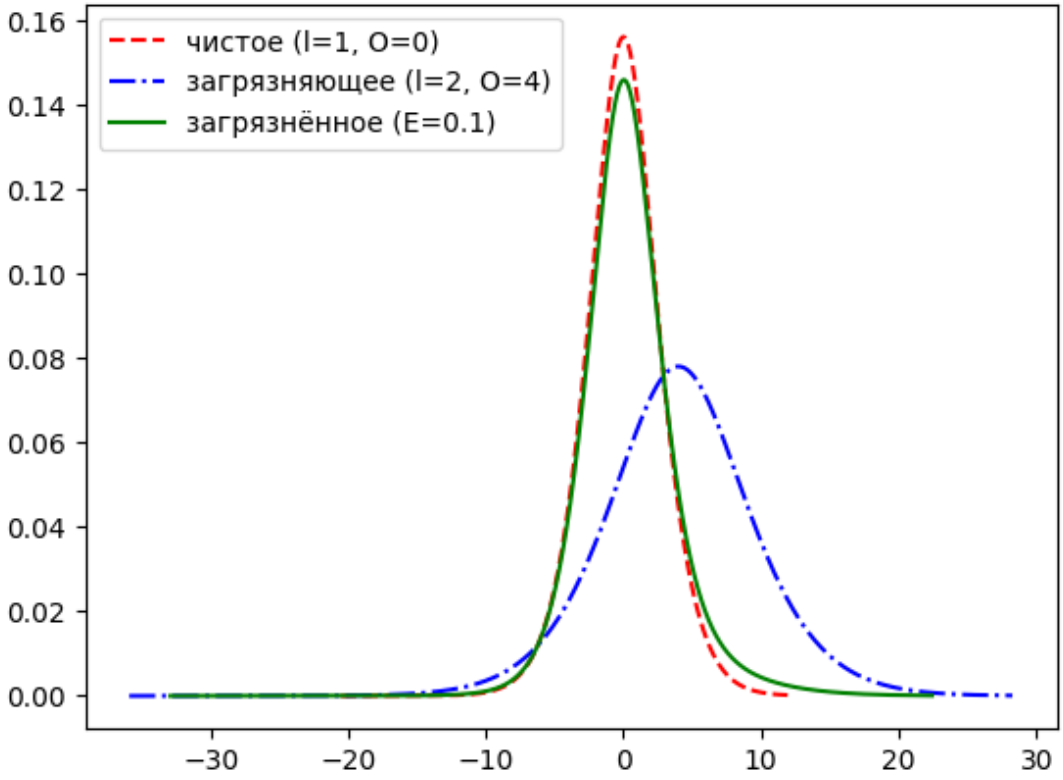

Рисунок 1 – графики функций плотности тестовых распределений для асимметричного зашумления

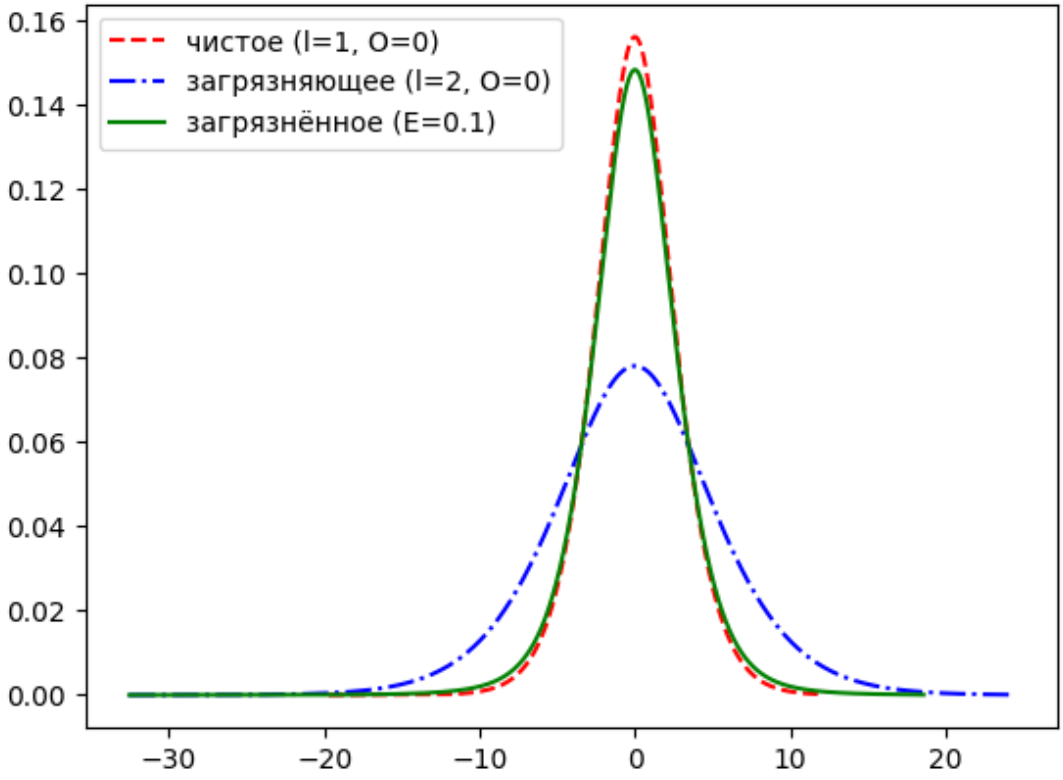

Рисунок 2 – графики функций плотности тестовых распределений для симметричного зашумления

2.2. ВЫЧИСЛЕНИЕ ОЦЕНОК ПАРАМЕТРА СДВИГА

2.2.1. О вычислении оценок параметра сдвига

При вычислении оценок параметра сдвига будут использоваться 3 распределения: чистое, засорённое с симметричным засорением и засорённое с асимметричным засорением. Для каждого из 3 распределений будут сделаны 3 различные выборки размером $N = 500$ элементов. Для генерации симметричного засорения будут использоваться параметры $\lambda = 2, \theta = 0$. Для генерации ассиметричного засорения: параметры $\lambda = 2, \theta = 7$.

Оценку параметра сдвига при помощи среднего арифметического можно провести по следующей формуле:

$$\theta = \frac{1}{N}\sum_{i=1}^{N} y_i.$$

Оценку параметра сдвига при помощи выборочной медианы можно провести путём подсчёта выборочной медианы. Аналогично и с оценкой по методу усечённого среднего.

Для оценки параметра сдвига по методу максимального правдоподобия, необходимо провести минимизацию следующей функции:

$$Q(\theta) = \frac{1}{N}\sum_{i=1}^{N} - ln\, f\left(\frac{y-\theta}{\lambda}\right),$$

где f – функция плотности распределения Лапласа без смещений.

Для оценки параметра сдвига по методу обобщённых радикальных оценок необходимо провести минимизацию следующей функции:

$$\rho(y,\theta) = -\frac{1}{f^{\delta}(0)} f^{\delta}\left(\frac{y-\theta}{\lambda}\right),$$

где $\delta > 0$ – параметр, регулирующий степень робастности оценки.

2.2.2. Оценка параметра сдвига для чистого распределения

Параметры выборки: $N = 500, \varepsilon = 0, \theta = 0, \lambda = 1$. Результаты оценок приведены в таблице 6. График функций плотности изображены на рисунке 3.

Таблица 6. Оценка параметра сдвига для чистого распределения

Оценка	1	2	3
Среднее арифметическое	0.031311	-0.207190	0.127137
Медиана	-0.087063	-0.160677	0.017835
Усечённое среднее (0.05)	0.005833	-0.235482	0.135109
Усечённое среднее (0.10)	-0.024193	-0.238936	0.124555
Усечённое среднее (0.15)	-0.040558	-0.238717	0.119639
Максимальное правдоподобие	-0.008228	-0.230535	0.119671
Обобщённые рад. оценки (0.1)	-0.024548	-0.238999	0.116672
Обобщённые рад. оценки (0.5)	-0.080853	-0.242184	0.096266
Обобщённые рад. оценки (1.0)	-0.123713	-0.237426	0.065782

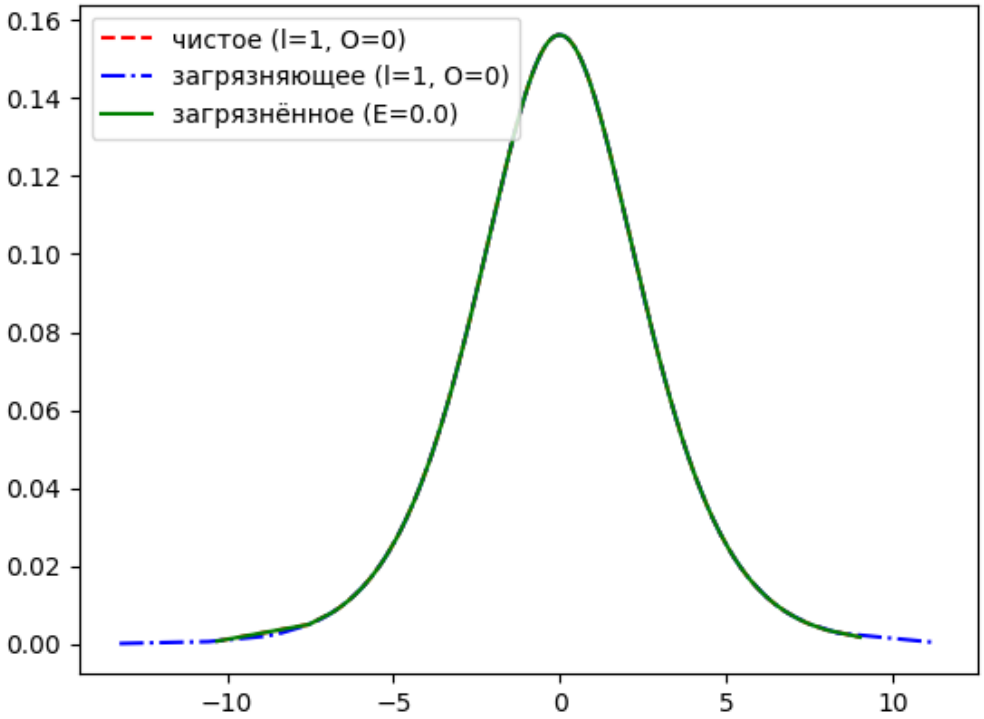

Рисунок 3 – график функций плотности чистого распределения

2.2.3. Оценка параметра сдвига симметричного засорения

Параметры выборки: $N = 500, \varepsilon = 0.4, \theta = 0, \lambda = 2$. Результаты оценок приведены в таблице 7. График функций плотности изображены на рисунке 4.

Таблица 7. Оценка параметра сдвига распределения с симметричным засорением

Оценка	1	2	3
Среднее арифметическое	-0.149536	0.279051	-0.155762
Медиана	-0.104019	0.131443	-0.059158
Усечённое среднее (0.05)	-0.132242	0.188167	-0.048394
Усечённое среднее (0.10)	-0.130925	0.163511	-0.047880
Усечённое среднее (0.15)	-0.137261	0.142980	-0.053193
Максимальное правдоподобие	-0.131394	0.170082	-0.068947
Обобщённые рад. оценки (0.1)	-0.121231	0.116904	-0.033999
Обобщённые рад. оценки (0.5)	-0.097883	0.035751	-0.035105
Обобщённые рад. оценки (1.0)	-0.085556	0.006322	-0.061179

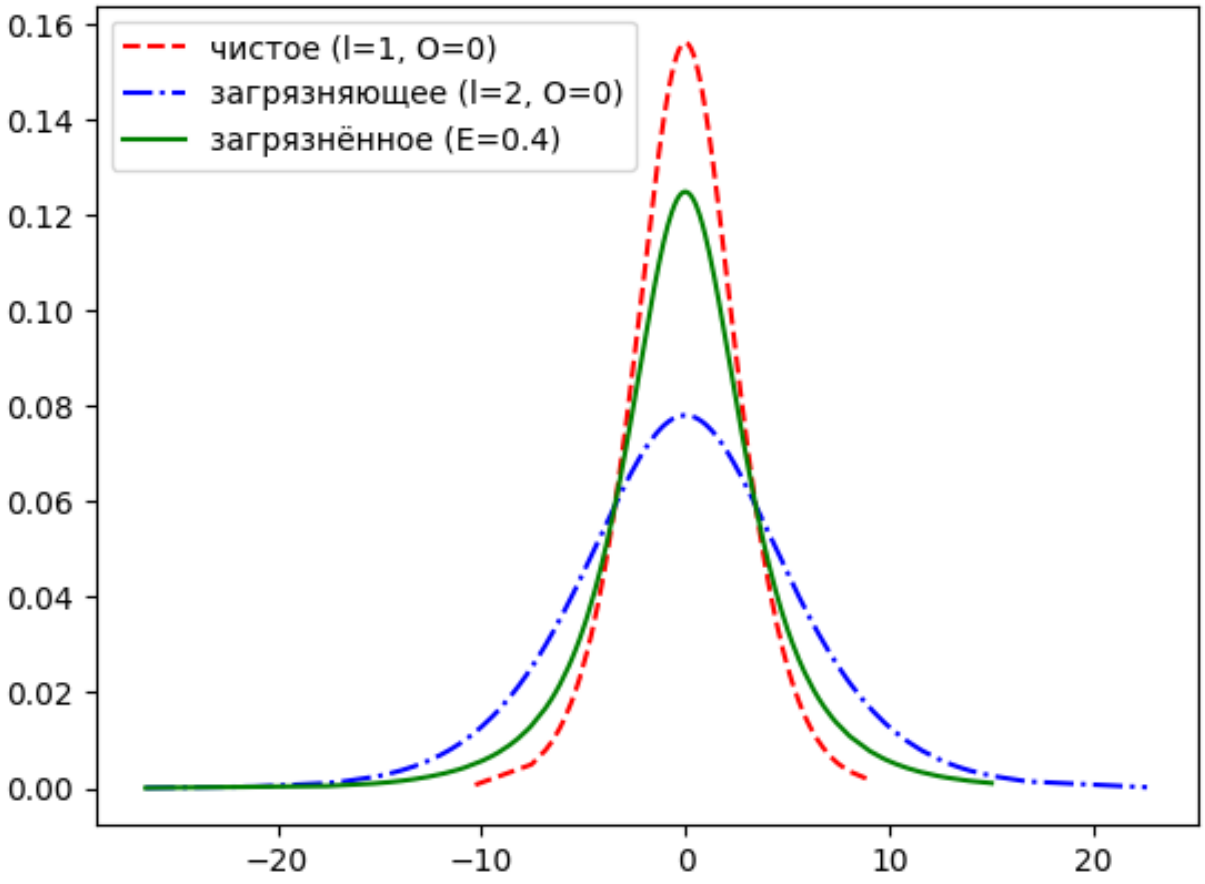

Рисунок 4 – график функций плотности распределения с симметричным засорением

2.2.4. Оценка параметра сдвига ассиметричного засорения

Параметры выборки: $N = 500, \varepsilon = 0.4, \theta = 7, \lambda = 2$. Результаты оценок приведены в таблице 8. График функций плотности изображены на рисунке 5.

Таблица 8. Оценка параметра сдвига распределения с асимметричным засорением

Оценка	1	2	3
Среднее арифметическое	2.650464	3.261051	2.588238
Медиана	1.223441	1.655123	1.439161
Усечённое среднее (0.05)	2.369304	2.949935	2.357698
Усечённое среднее (0.10)	2.194903	2.757692	2.171514
Усечённое среднее (0.15)	2.034749	2.574047	2.002744
Максимальное правдоподобие	2.015391	2.505499	2.005297
Обобщённые рад. оценки (0.1)	1.604532	1.950292	1.609359
Обобщённые рад. оценки (0.5)	0.666555	0.736931	0.747081
Обобщённые рад. оценки (1.0)	0.272267	0.279815	0.395158

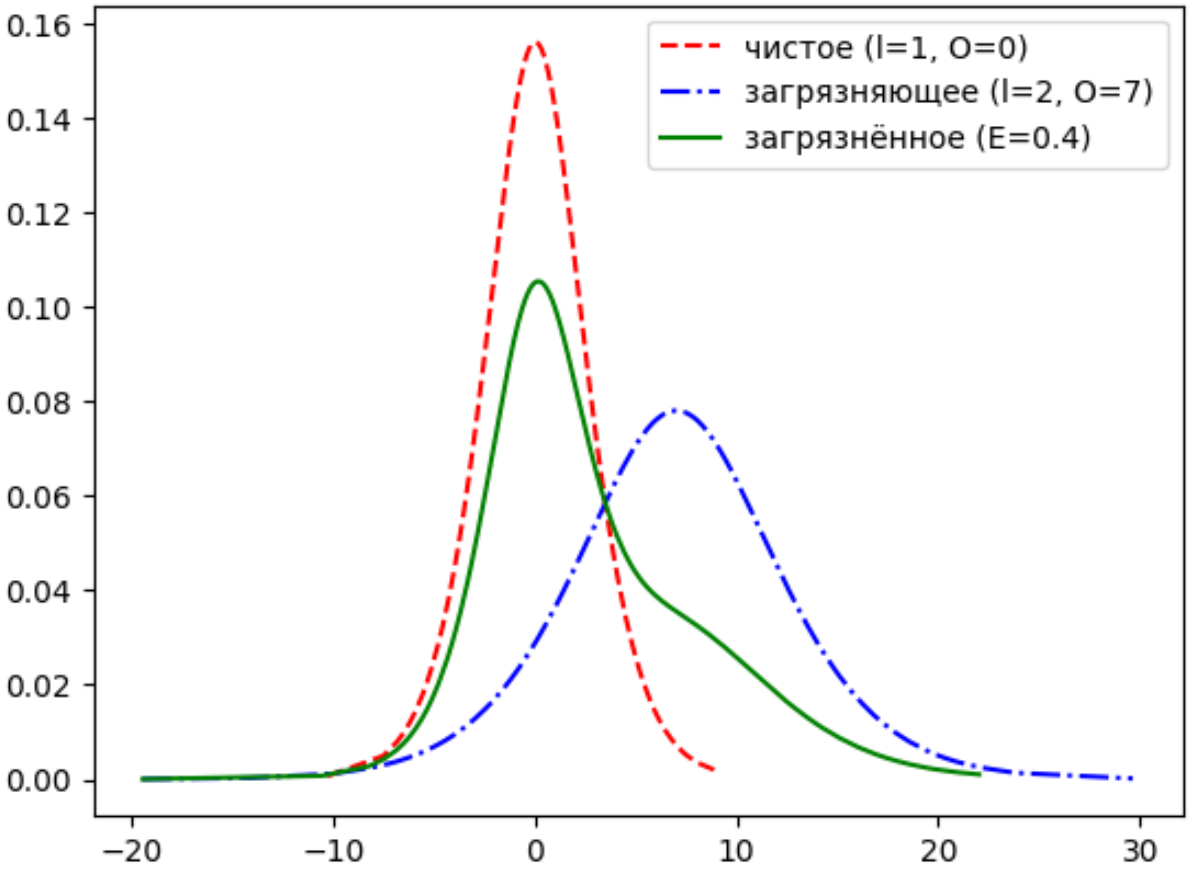

Рисунок 5 – график функций плотности распределения с асимметричным засорением

2.3. ГРАФИКИ ФУНКЦИЙ ВЛИЯНИЯ ОЦЕНОК

Функция влияния – функция, являющаяся одной из мер количественной робастности оценки. Функция влияния определяет воздействие на оценку, оказываемое добавлением к очень большой выборке одного наблюдения в точке y. В результате она отражает асимптотическое смещение оценки, вызываемое засорением наблюдений.

Функция влияния среднего арифметического выражается следующей формулой:

$$\mathrm{IF}(y) = y - \theta.$$

Для метода медианы, функция влияния следующая:

$$\mathrm{IF}(y) = \frac{\lambda \cdot \mathrm{sign}(y - \theta)}{2 \cdot f(0)}.$$

Учитывая, что $f(0) = \frac{5}{32}$, получаем

$$\mathrm{IF}(y) = \frac{\lambda \cdot \mathrm{sign}(y - \theta) \cdot 16}{5}.$$

График функций влияния среднего арифметического и медианы изображены на рисунке 6.

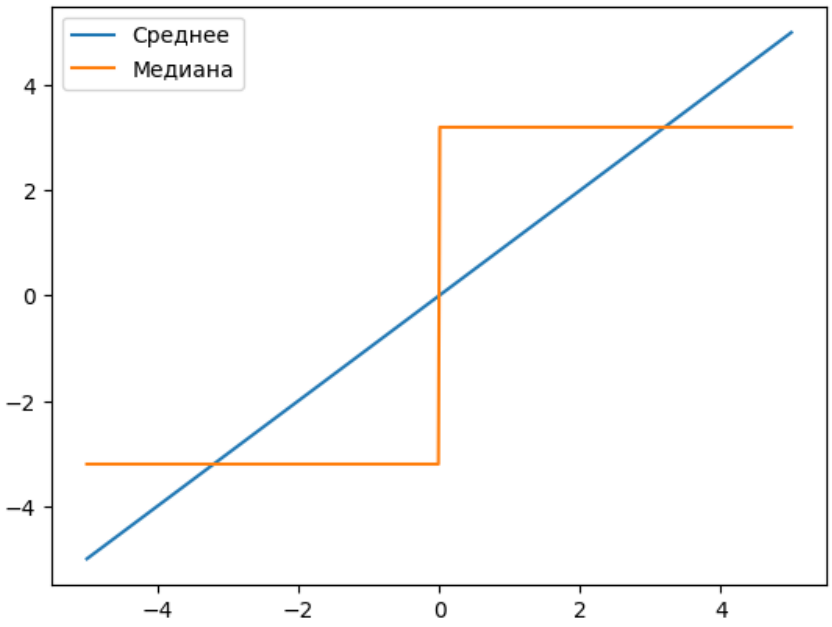

Рисунок 6 – график функций влияния среднего арифметического и медианы

Функция влияния усечённого среднего имеет следующий вид:

$$\mathrm{IF}(y) = \frac{1}{1-2\alpha}\begin{cases} -q, \dfrac{y-\theta}{\lambda} < -q, \\ \dfrac{y-\theta}{\lambda}, \left|\dfrac{y-\theta}{\lambda}\right| \le q, \\ q, \dfrac{y-\theta}{\lambda} > q, \end{cases}$$

где q – квантиль порядка $1 - \alpha$ стандартного распределения, то есть,

$$\int_{-\infty}^{q} f(y)dy = 1 - \alpha,$$

а α – уровень усечения ($\alpha = \{0.05, 0.10, 0.15\}$). Параметр q будем находить путём минимизации функционала

$$P(q) = \left|(1-\alpha) - \int_{-\infty}^{q} f(y)dy\right|.$$

График функции влияния усечённого среднего для разных параметров α представлен на рисунке 7.

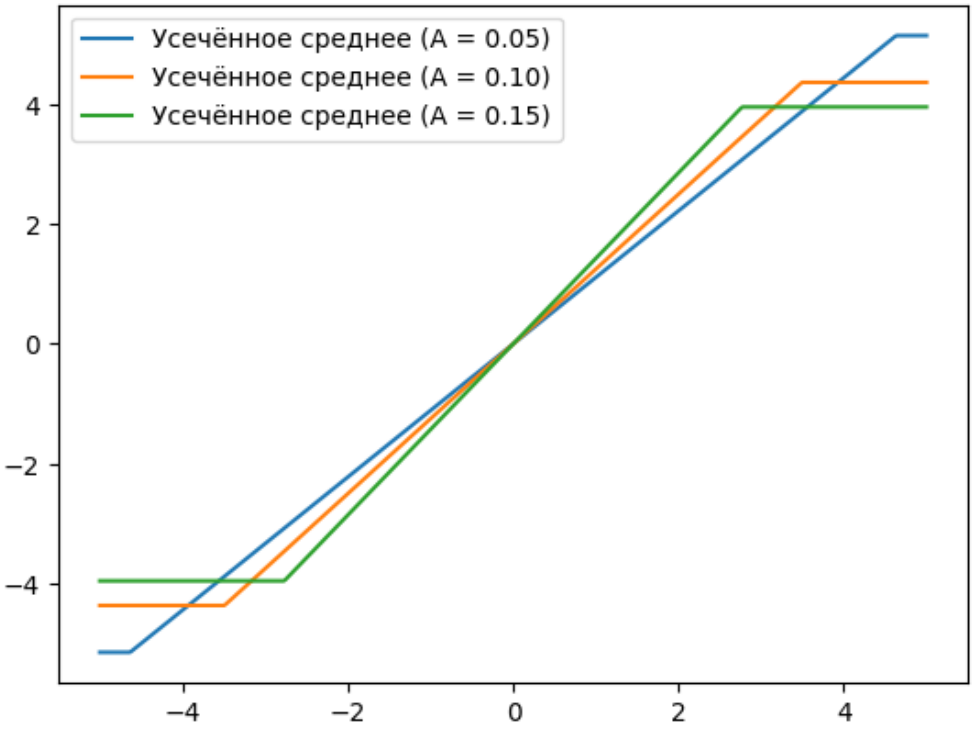

Рисунок 7 – график функции влияния усечённого среднего при разных параметрах α

Для метода максимального правдоподобия функция влияния задаётся следующим образом:

$$\mathrm{IF}(y) = \frac{-\lambda f'\left(\frac{y-\theta}{\lambda}\right)/f\left(\frac{y-\theta}{\lambda}\right)}{\int_{-\infty}^{+\infty}[f'(z)]^2/f(z)dz}.$$

Производная функции f равна

$$\frac{df}{dx} = -xe^{-|x|}\frac{(|x|^2+3|x|+3)}{96}.$$

График функции влияния для метода максимального правдоподобия изображён на рисунке 8.

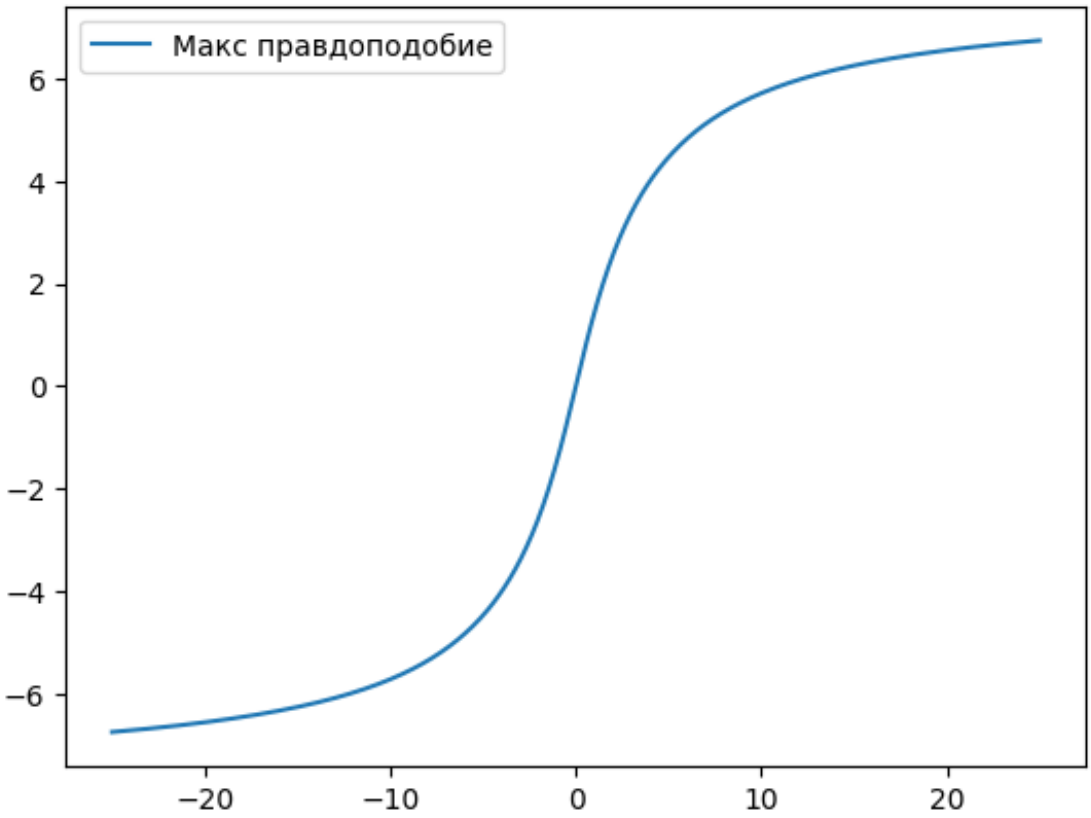

Рисунок 8 – график функции влияния для метода максимального правдоподобия

Для метода обобщённых радикальных оценок функция влияния задаётся следующим образом:

$$\mathrm{IF}(y) = \frac{-\lambda f'\left(\frac{y-\theta}{\lambda}\right)f^{\delta-1}\left(\frac{y-\theta}{\lambda}\right)}{\int_{-\infty}^{+\infty}[f'(z)]^2 f^{\delta-1}(z)\,dz}.$$

График функции влияния для метода обобщённых радикальных оценок с разными параметрами δ изображён на рисунке 9.

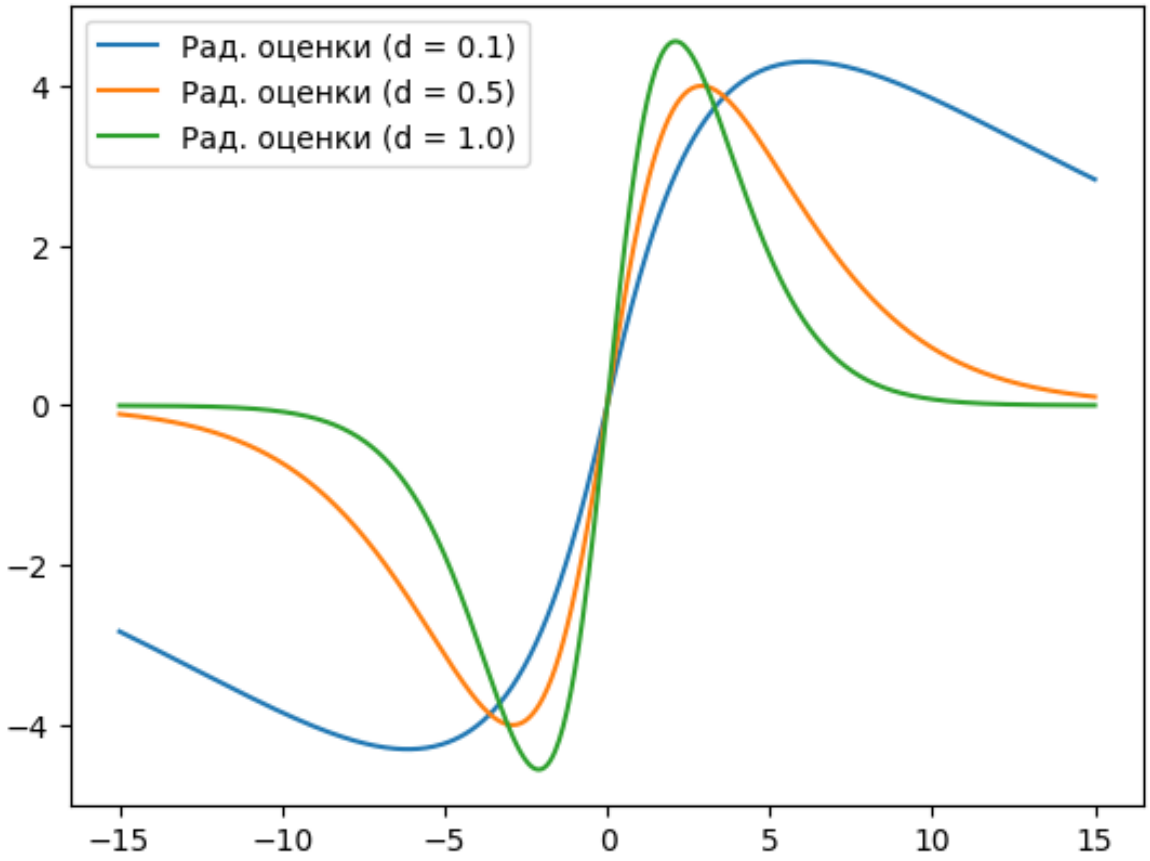

Рисунок 9 – график функции влияния для метода обобщённых радикальных оценок

График всех функций влияния изображён на рисунке 10.

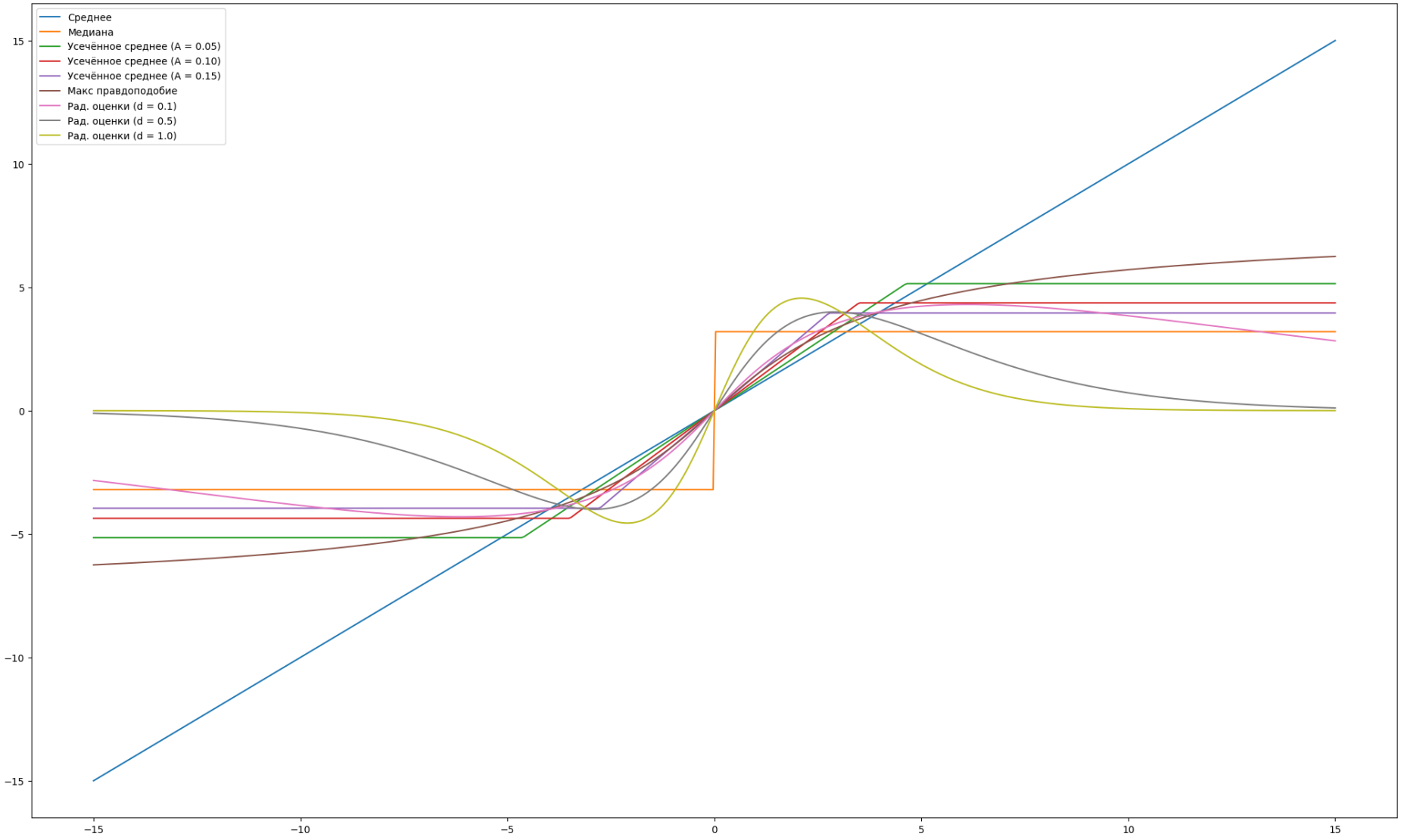

Рисунок 10 – график всех функций влияния

3. ВЫВОДЫ

1. На чистом распределении все оценки показали примерно одинаковые, близкие к истине, результаты. Отклонения от истинного значения возникали вследствие неравномерной генерации выборки случайных значений при сравнительно небольшом объёме выборки.

2. На распределении с симметричным засорением оценки также показали примерно одинаковые результаты.

3. На распределении с ассиметричным засорением лучше всего себя показали оценки на основе медианы и обобщённых радикальных оценок, причём с наибольшим параметром δ. Причину этому наглядно видно по графикам функций влияния каждого из методов оценки. Функция влияния медианы при значениях $y = 7$ и её окрестности (область основного множества значений зашумляющего распределения) постоянна и имеет достаточно низкое значение. Значения функций влияния обобщённых радикальных оценок так же достаточно низкие и при этом стремятся к нулю при удалении от центра (причём чем больше δ, тем более резко график устремляется к нулю). При этом график функции обобщённых радикальных оценок при $\delta = 0.1$ находится выше, чем график медианы, поэтому и оценка при этом значении параметра показала результат хуже, чем медиана. Остальные же оценки имеют значения функции влияния в этой области гораздо выше, поэтому они показывают менее верный результат. Хуже всего себя проявляет оценка по среднему арифметическому, поскольку значение её функции влияния не ограничено сверху и линейно растёт по мере удаления от нулевой точки.

4. ИСХОДНЫЙ КОД ПРОГРАММЫ

Проект Google Colab

```
# %% [markdown]
# # Вариант 10.B

# %% [markdown]
# ## О чём работа в блокноте
#
# Обобщенное распределение Лапласа со следующим значением параметра $n=4$.
#
# $$
# f(x,n) = f(x,4) = \frac{e^{-|x|}}{(n-1)!\cdot 16}
\sum\limits_{j=0}^{3}\frac{(3+j)!}{(3-j)!\cdot j!} \frac{|x|^{3-j}}{2^{j}} =
\
# $$
#
# $$
# \ = e^{-|x|}\cdot\frac{|x|^{3}+6|x|^{2} + 15|x| + 15}{96}.
# $$
#
# Дисперсия: $\sigma^{2}= 8$,
#
# Коэффициент эксцесса: $\large\gamma_{2} = \frac{3}{4}$.
#

# %% [markdown]
# ## Как пользоваться блокнотом?
#
# В блокноте будут идти вычисления ТОЛЬКО ПО ОДНОЙ ВЫБОРКЕ (все графики, всё
прочее строятся только для одной выборки). Поэтому для того, чтобы получить
значения на разных выборках, необходимо менять глобальные параметры,
расположенные в блоке кода ниже.
#
# Для того, чтобы быстро перезапустить вычисление во всех ячейках, можно
нажать `ctrl+F9`, либо через верхнее меню: `Среда выполнения -> выполнить
все`.

# %%
# базовые импорты

import math
import statistics
import matplotlib.pyplot as plt

import numpy as np

from scipy import stats as st
from scipy import integrate
```

```
from scipy.integrate import quad
from scipy.optimize import minimize

import pandas as pd

# Глобальные параметры
count = 10000 # количество которое хочешь сгенерить
eps = 0.1  # Эпсилон (коэффициент зашумления)

rng = np.random.default_rng(
    12345 # 12345, 23456, 34567
)  # Генератор рандомных чисел с фиксированным сидом (для повторяемости
результатов)

# Параметры для чистого распределения
N_net = 4  # Параметр распределения
l_net = 1  # Масштаб (лямбда) НЕ МЕНЯТЬ
O_net = 0  # Смещение (Тетта) НЕ МЕНЯТЬ

# Параметры для шума
N_scattener = N_net  # Параметр распределения
l_scattener = 2  # Масштаб (лямбда). 1 по умолчанию
O_scattener = 4  # Смещение (Тетта). 0 по умолчанию

# %% [markdown]
# ## 1 Разработка программы

# %% [markdown]
# ### 1.1. Генерация данных

# %% [markdown]
# > Полное задание пункта
# > Разработать программу, которая реализует генерацию наборов данных с
заданным в варианте *чистым распределением и засоренным распределением*,
использовать засоряющие распределения, совпадающие с чистым с точностью до
значений параметров сдвига и масштаба.
#

# %% [markdown]
# #### Генерация чистого и загрязняющего распределения
#
# Обобщённое распределение Лапласа с параметром $n=4$ задаётся следующим
образом:
#
# $$
# f(x) = e^{-|x|}\cdot\frac{|x|^{3}+6|x|^{2} + 15|x| + 15}{96}.
# $$
#
# Дисперсия при этом равна $\sigma^{2}=8$, коэффициент эксцесса:
$\large\gamma_{2}=\frac{3}{4}$.
#
# Моделирование случайных величин осуществляется по следующей формуле:
```

```
#
# $$
# x = \ln\left(\frac{\prod\limits_{i:r_{i} \leqslant \frac{1}{2}}
2r_{i}}{\prod\limits_{i: r_{i} > \frac{1}{2}}2(1-r_{i})}\right),
# $$
#
# где $r_{i}, i=\{1, \dots, n\}= \{1,2,3,4\}$ - *реализация случайной
величины*, равномерно распределённой на интервале $(0; 1)$ (просто 4
значения с генератора равномерных случайных чисел); если нет ни одного
элемента, удовлетворяющего условию в произведении, то оно равно 1.
#
# #### Генерация засоряющего распределения
#
# Засоряющее распределение должно (судя по всему) повторять исходное
распределение, но также должно иметь возможность изменять свой масштаб и
сдвиг.  Сдвиг-масштабное преобразование осуществляется по следующей формуле:
#
# $$
# x = \theta + \lambda x_{1},
# $$
#
# где $\theta$ - параметр сдвига, $\lambda$ - параметр масштаба, $x_{1}$ -
значение случайной величины с нулевым сдвигом и единичным масштабом.
#
# Применимо к формуле нашего распределения, получаем:
#
# $$
# x = \theta + \lambda \ln\left(\frac{\prod\limits_{i:r_{i} \leqslant
\frac{1}{2}} 2r_{i}}{\prod\limits_{i: r_{i} > \frac{1}{2}}2(1-
r_{i})}\right).
# $$
#
#


# %%
def gen_rand_laplas(n_lap: int, theta: float = 0.0, lamb: float = 1.0) ->
float:
    """
    Генератор случайной величины, соответствующей обобщенному распределению
Лапласа, с возможностью задания сдвига и масштабирования

    Args:
        n_lap (int): Натуральное число, соответствующее количеству случайных
величин. Является параметром обобщённого распределения Лапласа `n`
        theta (float, optional): параметр сдвига распределения тэтта. По
умолчанию 0.0
        lamb (float, optional): параметр масштабироваиня сдвига лямбда. По
умолчанию 1.0

    Returns:
        float: Случайная величина, соответствующая обобщенному распределению
Лапласа.
```

```
    """
    global rng

    list_rand = rng.random((n_lap,))

    res = 1.0
    for item in list_rand:
        if item <= (1 / 2):
            res *= 2 * item
        else:
            res /= 2 * (1 - item)

    return np.log(res) * lamb + theta

def gen_full_laplas(
    count: int, theta: float = 0.0, lamb: float = 1.0
) -> np.ndarray[np.float64]:
    """Генератор множества случайных величин по распределению Лапласа

    Args:
        count (int): размер выборки случайных чисел по данному распределению
        theta (float, optional): параметр сдвига распределения тэтта. По
умолчанию 0.0
        lamb (float, optional): параметр масштабироваиня сдвига лямбда. По
умолчанию 1.0

    Returns:
        ndarray[float64]: множество значений полученной выборки
    """
    global N_net

    res = [gen_rand_laplas(N_net, theta=theta, lamb=lamb) for _ in
range(count)]
    return np.asarray(res)

print(f"Размер выборки: {count} элементов...")

print("Строим чистое распределение...")
clean = gen_full_laplas(count)

print("Строим загрязняющее распределение...")
scattener = gen_full_laplas(count, theta=0_scattener, lamb=l_scattener)

# %% [markdown]
# #### Генерация засорённого распределения
#
# Засорённое распределение представляет собой смесь двух распределений с
плотностью, задаваемой следующей формулой:
#
# $$
# g(x) = (1 - \varepsilon) f(x) + \varepsilon h(x),
# $$
```

```
#
# где $0 < \varepsilon < 0.5$ - уровень засорения, $f(x), h(x)$ - плотности
чистого и засоряющего распределений соответственно.
#
# Моделирование случайной величины с засорённым распределением можно
проводить по следующему алгоритму:
#
# 1. Сгенерировать случайное число $r$, равномерно распределённое на
интервале $(0; 1)$. Если $r \leqslant 1 - \varepsilon$, перейти на шаг 2,
иначе перейти на шаг 3.
# 2. Сгенерировать случайное число с плотностью $f(x)$ (чистое
распределение). Оно и будет искомым числом.
# 3. Сгенерировать случайное число с плотностью $h(x)$ (загрязняющее
распределение). Оно и будет искомым числом.

# %%
def get_scattened_laplas(
    clean: np.ndarray[np.float64], scattener: np.ndarray[np.float64], eps:
float
) -> np.ndarray[np.float64]:
    """Соединяет чистое и загрязняющее распределения в единое загрязнённое

    Args:
        clean (np.ndarray[np.float64]): множество случайных величин из
чистого распределения
        scattener (np.ndarray[np.float64]): множество случайных величин из
загрязняющего распределения (должно совпадать по размеру с чистым)
        eps (float): коэффициент загрязнения (0 <= `eps` <= 0.5)

    Returns:
        np.ndarray[np.float64]: полученное загрязнённое распределение
    """
    global rng
    res = np.zeros_like(clean)
    rand = rng.random((clean.size,))

    for ind in range(res.size):
        if rand[ind] <= (1 - eps):
            res[ind] = clean[ind]
        else:
            res[ind] = scattener[ind]

    return res

scattened = get_scattened_laplas(clean, scattener, eps)

# %% [markdown]
# #### Вывод графиков
#
#

# %%
```

```python
def f_laplas(data: np.ndarray, theta: float = 0.0, lamb: float = 1.0) ->
np.ndarray:
    """
    Функция плотности распределения Лапласа от значений X

    Args:
        data (np.ndarray): значения X текущего распределения (желательно
должна быть отсортированной)
        theta (float): значение параметра сдвига функции (по умолчанию 0)
        lamb (float): значение параметра масштабирования функции (по
умолчанию 1)

    Returns:
        np.ndarray: значения функции плотности распределения Лапласа
    """
    ordered = np.sort(data) # Сортируем последовательность
    ordered = (ordered - theta) / lamb # Убираем сдвиги
    ordered = np.abs(ordered) # Убираем знаки значений

    result = np.exp(-1 * ordered)
    result *= (ordered**3 + 6 * ordered**2 + 15*ordered + 15) / (96 * lamb)
    return result

def f_laplas_mixed(mixed: np.ndarray) -> np.ndarray:
    """
    Функция плотности смешанного распределения от значений X чистого и
загрязняющего распределений

    Args:
        mixed (np.ndarray): значения X смешанного распределения

    Returns:
        np.ndarray: значения функции плотности смешанного распределения
Лапласа
    """
    mixed_sorted = np.sort(mixed) # Сортируем последовательность
    f_clean = f_laplas(mixed_sorted, O_net, l_net)
    f_scattener = f_laplas(mixed_sorted, O_scattener, l_scattener)

    return (1 - eps) * f_clean + eps * f_scattener

def plot_graphics(
    clean: np.ndarray[np.float64],
    scattener: np.ndarray[np.float64],
    scattened: np.ndarray[np.float64],
):
    """Выводит графики сгенерированных распределений на экран (приводя
размеры к единичному по графику `clean`)

    Args:
        clean (np.ndarray[np.float64]): массив значений из чистого
распределения
```

```
        scattener (np.ndarray[np.float64]): массив значений из загрязняющего
распределения
        scattened (np.ndarray[np.float64]): массив значений из грязного
распределения
    """
    step = 1
    if clean.size > 10000:
        step = int(clean.size / 5000)
    clean_sorted = np.sort(clean)[::step]
    scattener_sorted = np.sort(scattener)[::step]
    scattened_sorted = np.sort(scattened)[::step]

    f_clean = f_laplas(clean_sorted, O_net, l_net)
    f_scattener = f_laplas(scattener_sorted, O_scattener, l_scattener)
    f_scattened = f_laplas_mixed(scattened_sorted)

    # Выводим полученные графики
    fig, axes = plt.subplots()
    axes.plot(clean_sorted, f_clean, label=f"чистое (l={l_net}, O={O_net})",
linestyle="--", color="red")
    axes.plot(scattener_sorted, f_scattener, label=f"загрязняющее
(l={l_scattener}, O={O_scattener})", linestyle="-.", color="blue")
    axes.plot(scattened_sorted, f_scattened, label=f"загрязнённое
(E={eps})", linestyle="-", color="green")
    # axes.plot([10.0, 10.0], [0.0, 1.1], label="Центр", color="purple")
    axes.legend()

    plt.show()

print("Графики функций плотности полученных распределений")
print(f"Чистое распределение: Тетта = {O_net}, Лямбда = {l_net}")
print(f"Загрязняющее распределение: Тетта = {O_scattener}, Лямбда =
{l_scattener}")
plot_graphics(clean, scattener, scattened)

# %% [markdown]
# ### 1.4. Вычисление математического ожидания, дисперсии, коэффициентов
асимметрии и эксцесса случайной величины с засоренным распределением по
заданным характеристикам чистого и засоряющего распределений

# %% [markdown]
# Для того, чтобы получить значения данных характеристик для загрязнённого
распределения, надо найти их для чистого и загрязняющего. Ищуются они по
следующим формулам:
#
# $$
# M = 0 + \theta,
# $$
#
# $$
# D = \sigma^{2} = 2\cdot n \cdot \lambda^{2},
# $$
```

```
#
# $$
# \gamma_{1} = 0, ???
# $$
#
# $$
# \gamma_{2} = \frac{3}{n}.
# $$
#

# %%
dataframe_labels = [
    "Мат ожидание M",
    "Дисперсия D",
    "Ассиметрия gamma_1",
    "Эксцесс gamma_2",
]

# Теоретические величины
M1_ = 0 + O_net            # Матожидание
D1_ = 2 * N_net * l_net**2 # Дисперсия
G11_ = 0                   # Ассиметрия
G12_ = 3 / N_net           # Эксцесс

print(f"Для чистого распределения ({count} точек):")
pd.DataFrame(np.array([dataframe_labels, [M1_, D1_, G11_, G12_]]).T,
columns=["Параметр чистого", "Значение"])

# %%
# Теоретические величины
M2_ = 0 + O_scattener                  # Матожидание
D2_ = 2 * N_scattener * l_scattener**2 # Дисперсия
G21_ = 0 # Ассиметрия
G22_ = 3 / N_scattener # Эксцесс

print(f"Для загрязняющего распределения ({count} точек):")
pd.DataFrame(np.array([dataframe_labels, [M2_, D2_, G21_, G22_]]).T,
columns=["Параметр загрязняющего", "Значение"])

# %% [markdown]
# Для вычисления всяких там параметров полученного засорённого распределения
можно воспользоваться следующими формулами. Обозначим $M_{i}, D_{i},
\gamma_{2i}, i=1,2$ мат ожидание, дисперсию и коэффициент эксцесса,
соответствующие при $i=1$ и $i=2$ плотностям $f(x)$ и $h(x)$. Тогда
матожидание, дисперсия, коэффициенты асимметрии и эксцесса случайной
величины с засорённым распределением выражается формулами:
#
# $$
# M = (1-\varepsilon)M_{1} + \varepsilon M_{2},
# $$
#
# $$
```

```
# D = (1 - \varepsilon)(M_{1}^{2} + D_{1}) + \varepsilon(M_{2}^{2} + D_{2})
- M^{2},
# $$
#
# $$
# \gamma_{1} = \frac{1}{D^{1.5}} \left[(1 - \varepsilon) \left\{(M_{1} -
M)^{3} + 3(M_{1} - M)D_{1}\right\} + \varepsilon \left\{(M_{2}-M)^{3} +
3(M_{2}-M)D_{2} \right\} \right],
# $$
#
# $$
# \gamma_{2} = \frac{1}{D^{2}} \left\{ (1-\varepsilon) \left[ (M_{1} -
M)^{4} + 6(M_{1} - M)^{2}D_{1} + D_{1}^{2}(\gamma_{21} + 3) \right] + \
\right.
# $$
#
# $$
# \ + \left. \varepsilon [(M_{2}-M)^{4} +6(M_{2}-M)^{2}D_{2} +
D_{2}^{2}(\gamma_{22} + 3) ] \right\} - 3.
# $$

# %%
# Теоретические величины
M_ = (1 - eps) * M1_ + eps * M2_
D_ = (1 - eps) * (M1_**2 + D1_) + eps * (M2_**2 + D2_) - M_**2
G1_ = (1 / D_**1.5) * (
    (1 - eps) * ((M1_ - M_) ** 3 + 3 * (M1_ - M_) * D1_)
    + eps * ((M2_ - M_) ** 3 + 3 * (M2_ - M_) * D2_)
)
G2_ = (1 / D_**2) * (
    (1 - eps) * ((M1_ - M_)**4 + 6 * (M1_ - M_)**2 * D1_ + D1_**2 * (G12_ +
3))
        + eps * ((M2_ - M_)**4 + 6 * (M2_ - M_)**2 * D2_ + D2_**2 * (G22_ +
3))
) - 3

print(f"Для загрязнённого распределения ({count} точек):")
pd.DataFrame(
    np.array([dataframe_labels, [M_, D_, G1_, G2_]]).T,
    columns=["Параметр загрязнённого", "Значение"],
)

# %% [markdown]
# ### 1.2. Вычисление выборочных характеристик полученных распределений
#
# Здесь фактически совпадёт со 2 пунктом, поэтому пропускаю

# %% [markdown]
# ## 2 Провести проверку генератора чистого и засорённого распределений

# %% [markdown]
# Выборочные характеристики считаются следующим образом:
```

```
#
# |Название|Формула|Функция|
# |---|:---:|---|
# |Среднее арифметическое $M$ | $\frac{1}{N}\sum\limits_{i=1}^{N}y_{i}$ |
`numpy.mean(array) -> float` |
# |Медиана $m$ | тю-тю (просто средний элемент) | `numpy.median(array) ->
float` |
# |Дисперсия $D$ | $\frac{1}{N}\sum\limits_{i=1}^{N}(y_{i}-M)^{2}$ |
`numpy.var(array) -> float` |
# |Коэф. асимметрии $\gamma_1$| $\frac{\sum\limits_{i=1}^{N} (y_{i} -
\bar{y})^{3}}{N \cdot D^{1.5}}$ |`scipy.stats.skew(array) -> float`|
# |Коэф. эксцесса $\gamma_2$|$\frac{\sum\limits_{i=1}^{N}(y_{i} -
\hat{y})^{4}}{N\cdot D^{2}} - 3$|`scipy.stats.kurtosis(array) -> float`|
#
# Для чистого и загрязняющего распределений, медиана равна $0$ и $\theta$
соответственно в силу их симметричности. Для загрязнённого распределения, в
силу его возможной несимметричности, теоретическую медиану найти неизвестно
как.
#

# %%
dataframe_labels = [
    "Мат ожидание M",
    "Медиана m",
    "Дисперсия D",
    "Ассиметрия gamma_1",
    "Эксцесс gamma_2",
]

# Практические величины
PM1_ = np.mean(clean)      # Матожидание
Pm1_ = np.median(clean)    # Медиана
PD1_ = np.var(clean)       # Дисперсия
PG11_ = st.skew(clean)     # Ассиметрия
PG12_ = st.kurtosis(clean) # Эксцесс

print(f"Теоретические и практические величины для чистого распределения
({count} точек)")
pd.DataFrame(
    np.array([
        dataframe_labels,
        [M1_, M1_, D1_, G11_, G12_],
        [PM1_, Pm1_, PD1_, PG11_, PG12_],
    ]).T,
    columns=["Параметр чистого", "Теоретические", "Практические"],
)

# %%
# Практические величины
PM2_ = np.mean(scattener)      # Матожидание
Pm2_ = np.median(scattener)    # Медиана
PD2_ = np.var(scattener)       # Дисперсия
```

```
PG21_ = st.skew(scattener)     # Ассиметрия
PG22_ = st.kurtosis(scattener) # Эксцесс

print(f"Теоретические и практические величины для загрязняющего
распределения ({count} точек)")
pd.DataFrame(
    np.array([
        dataframe_labels,
        [M2_, M2_, D2_, G21_, G22_],
        [PM2_, Pm2_, PD2_, PG21_, PG22_],
    ]).T,
    columns=["Параметр загрязняющего", "Теоретические", "Практические"],
)

# %%
dataframe_labels = [
    "Мат ожидание M",
    "Дисперсия D",
    "Ассиметрия gamma_1",
    "Эксцесс gamma_2",
]

# Практические величины
PM_ = np.mean(scattened)      # Матожидание
PD_ = np.var(scattened)       # Дисперсия
PG1_ = st.skew(scattened)     # Ассиметрия
PG2_ = st.kurtosis(scattened) # Эксцесс

print(f"Теоретические и практические величины для загрязнённого
распределения ({count} точек)")
pd.DataFrame(
    np.array([
        dataframe_labels,
        [M_, D_, G1_, G2_],
        [PM_, PD_, PG1_, PG2_],
    ]).T,
    columns=["Параметр загрязнённого", "Теоретические", "Практические"],
)

# %% [markdown]
# ## 3 Для выборок с разными видами распределений вычислить следующие оценки
параметра сдвига:

# %% [markdown]
# ### 3.1. Среднее арифметическое
#
# Само среднее арифметическое выглядит следующим образом:
#
# $$
# M = \frac{1}{N}\sum\limits_{i=1}^{N}y_{i}.
# $$
#
```

```
# Среднее арифметическое является M-оценкой параметра сдвига $\theta$ с функцией потерь
#
# $$
# \rho(y, \theta) = (y - \theta)^{2}.
# $$
#
# Для нахождения параметра сдвига необходимо решить следующую задачу минимизации:
#
# $$
# Q(\theta) = \frac{1}{N}\sum\limits_{i=1}^{N} (y_{i} - \theta)^{2} \rightarrow \min
# $$
#
# В нашем случае (видимо из-за симметричности распределения и того, что по умолчанию матожидание равно 0), для нахождения параметра сдвига по среднему арифметическому достаточно просто посчитать среднее арифметическое.

# %%
def theta_approximation_by_mean(data: np.ndarray) -> float:
    """
    Функция оценки параметра сдвига при помощи среднего арифметического

    Args:
        data (np.ndarray): последовательность, для которой необходимо выполнить оценку

    Returns:
        float: значение оценки параметра
    """
    return np.mean(data)

theta_by_mean = theta_approximation_by_mean(scattened)
print(f"Параметр сдвига по методу среднего арифметического: {theta_by_mean}")

# %% [markdown]
# ### 3.2. Выборочная медиана
#
# Выборочная медиана также является M-оценкой, но проще найти параметр сдвига через само значение медианы

# %%
def theta_approximation_by_median(data: np.ndarray) -> float:
    """
    Функция оценки параметра сдвига при помощи медианы

    Args:
        data (np.ndarray): последовательность, для которой необходимо выполнить оценку
```

```
    Returns:
        float: значение оценки параметра
    """
    return np.median(data)

theta_by_median = theta_approximation_by_median(scattened)
print(f"Параметр сдвига по методу медианы: {theta_by_median}")

# %% [markdown]
# ### 3.3. Усечённое среднее
#
# Для оценки параметра сдвига при помощи усечённого среднего можно
воспользоваться `scipy.stats.trim_mean(array, alpha: float) -> float`.
Необходимо будет также сделать 3 таких оценки с коэффициентами $0.05$,
$0.10$, $0.15$.

# %%
def theta_approximation_by_trim_mean(data: np.ndarray, alpha: float) ->
float:
    """
    Функция оценки параметра сдвига при помощи усечённого среднего

    Args:
        data (np.ndarray): последовательность, для которой необходимо
выполнить оценку
        alpha (float): процент усечения диапазона (от 0.0 до 0.5)

    Returns:
        float: значение оценки параметра
    """
    return st.trim_mean(data, alpha)

theta_by_trim_mean_05 = theta_approximation_by_trim_mean(scattened, 0.05)
theta_by_trim_mean_10 = theta_approximation_by_trim_mean(scattened, 0.10)
theta_by_trim_mean_15 = theta_approximation_by_trim_mean(scattened, 0.15)
print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.05):
{theta_by_trim_mean_05}")
print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.10):
{theta_by_trim_mean_10}")
print(f"Параметр сдвига по методу усечённого среднего (alpha = 0.15):
{theta_by_trim_mean_15}")

# %% [markdown]
# ### 3.4. Метод максимального правдоподобия
#
# Для оценки максимального правдоподобия необходимо провести минимизацию
функционала вида
#
# $$
# Q(\theta) = \frac{1}{N} \sum\limits_{i=1}^{N} -\ln f\left(\frac{y-
\theta}{\lambda}\right),
# $$
```

```
#
# где $f$ - функция плотности распределения Лапласа без смещений и прочего.

# %%
def theta_approximation_by_MLE(data: np.ndarray) -> float:
    """
    Функция оценки параметра сдвига при помощи метода максимального
правдоподобия

    Args:
        data (np.ndarray): последовательность, для которой необходимо
выполнить оценку

    Returns:
        float: значение оценки параметра
    """
    f = lambda O, x: np.sum(-np.log(f_laplas(x - O)))
    return minimize(f, x0=0, args=(data), tol=1e-6).x[0]

theta_by_MLE = theta_approximation_by_MLE(scattened)
print(f"Параметр сдвига по методу максимального правдоподобия:
{theta_by_MLE}")

# %% [markdown]
# ### 3.5. Метод обобщённых радикальных оценок
#
# Обобщённые радикальные оценки задаются следующей функцией потерь:
#
# $$
# \rho(y, \theta) = - \frac{1}{f^{\delta}(0)}f^{\delta}\left( \frac{y-
\theta}{\lambda} \right),
# $$
#
# где $\delta > 0$ - параметр, регулирующий степень робастности оценки.

# %%
def theta_approximation_by_MGD(data: np.ndarray, delta: float) -> float:
    """
    Функция оценки параметра сдвига при помощи метода обобщённых радикальных
оценок

    Args:
        data (np.ndarray): последовательность, для которой необходимо
выполнить оценку
        delta (float): параметр робастности оценки

    Returns:
        float: значение оценки параметра
    """
    f0 = 15/96
    f = lambda O, x: np.sum(-1 / (f0**delta) * f_laplas(x - O)**delta)
    return minimize(f, x0=0, args=(data), tol=1e-6).x[0]
```

```
theta_by_MGD_01 = theta_approximation_by_MGD(scattened, 0.1)
theta_by_MGD_05 = theta_approximation_by_MGD(scattened, 0.5)
theta_by_MGD_10 = theta_approximation_by_MGD(scattened, 1.0)
print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр =
0.1): {theta_by_MGD_01}")
print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр =
0.5): {theta_by_MGD_05}")
print(f"Параметр сдвига по методу обобщённых радикальных оценок (параметр =
1.0): {theta_by_MGD_10}")

# %% [markdown]
# ### 3.6. Все оценки в общем виде

# %%
results = [
    [
        "Среднее арифметическое", theta_approximation_by_mean(scattened),
    ],
    [
        "Медиана", theta_approximation_by_median(scattened),
    ],
    [
        "Усечённое среднее (0.05)",
theta_approximation_by_trim_mean(scattened, 0.05),
    ],
    [
        "Усечённое среднее (0.10)",
theta_approximation_by_trim_mean(scattened, 0.10),
    ],
    [
        "Усечённое среднее (0.15)",
theta_approximation_by_trim_mean(scattened, 0.15),
    ],
    [
        "Максимальное правдоподобие", theta_approximation_by_MLE(scattened),
    ],
    [
        "Обобщённые рад. оценки (0.1)",
theta_approximation_by_MGD(scattened, 0.1),
    ],
    [
        "Обобщённые рад. оценки (0.5)",
theta_approximation_by_MGD(scattened, 0.5),
    ],
    [
        "Обобщённые рад. оценки (1.0)",
theta_approximation_by_MGD(scattened, 1.0),
    ],
]

pd.DataFrame(results, columns=["Оценка", "Значение"])
```

```
# %% [markdown]
# ## 4 Графики функций влияния

# %% [markdown]
# ### 4.1. График среднего арифметического, медианы
#
# Функция влияния среднего арифметического выражается следующей формулой:
#
# $$
# \text{IF}(y) = y-\theta
# $$
#
# Для метода медианы, функция влияния следующая:
#
# $$
# \text{IF}(y) = \frac{\lambda\cdot \text{sign}(y - \theta)}{2\cdot f(0)}
# $$
#
# Учитывая, что $f(0) = \frac{5}{32}$, получаем:
#
# $$
# \text{IF}(y) = \frac{\lambda\cdot \text{sign}(y - \theta) \cdot 16}{5}
# $$

# %%
def IF_mean(data: np.ndarray, theta: float) -> np.ndarray:
    """
    Функция влиялия для среднего арифметического

    Args:
        data (np.ndarray): значения X распределения
        theta (float): параметр тетта смещения

    Returns:
        np.ndarray: значения функции влияния
    """
    return data - theta

def IF_median(data: np.ndarray, theta: float, lamb: float) -> np.ndarray:
    """
    Функция влиялия для медианы

    Args:
        data (np.ndarray): значения X распределения
        theta (float): параметр тетта смещения
        lamd (float): параметр лямбда распределения

    Returns:
        np.ndarray: значения функции влияния
    """
    return 16 * lamb * np.sign(data - theta) / 5
```

```python
x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()

axes.plot(x, IF_mean(x, 0), label="Среднее")
axes.plot(x, IF_median(x, 0, 1), label="Медиана")
axes.legend()

plt.show()

# %% [markdown]
# ### 4.2. Графики для усечённого среднего
#
# Функция влияния усечённого среднего имеет следующий вид:
#
# $$
# \large
# \text{IF}(y) = \frac{1}{1 - 2\alpha} \begin{cases}
# -q,~\frac{y - \theta}{\lambda} < -q \\
# \frac{y - \theta}{\lambda}, \left| \frac{y - \theta}{\lambda} \right|
\leqslant q, \\
# q, \frac{y - \theta}{\lambda} > q,
# \end{cases}
# $$
#
# где $q$ - квантиль порядка $1 - \alpha$ стандартного распределения, то
есть,
#
# $$
# \int_{-\infty}^{q} f(y) dy = 1 - \alpha,
# $$
#
# а $\alpha$ - уровень усечения ($\alpha = \{0.05, 0.10, 0.15\}$).

# %%
def IF_trim_mean(data: np.ndarray, alpha: float, theta: float, lamb: float)
-> np.ndarray:
    """
    Функция влиялия для усечённого среднего

    Args:
        data (np.ndarray): значения X распределения
        alpha (float): параметр усечения (в диапазоне 0 <= alpha < 0.5)
        theta (float): параметр тетта смещения
        lamd (float): параметр лямбда распределения

    Returns:
        np.ndarray: значения функции влияния
    """
    # Определяем параметр q
    right = 1 - alpha
```

```
    fn = lambda x: np.exp(-1 * np.abs(x)) * (np.abs(x)**3 + 6 * np.abs(x)**2
+ 15*np.abs(x) + 15) / 96
    fn_m = lambda q: np.abs(right - quad(fn, -np.inf, q)[0])
    q = minimize(fn_m, x0=2).x[0]

    # Считаем
    left = 1 / (1 - 2 * alpha) # Базовый множитель
    x = (data - theta) / lamb # преобразованная переменная y (массив)
    res = np.ndarray(x.shape) # Результат
    for i in range(x.size):
        if x[i] < -q:
            res[i] = -q
        elif x[i] > q:
            res[i] = q
        else:
            res[i] = x[i]
    return left * res

x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()
axes.plot(x, IF_trim_mean(x, 0.05, 0, 1), label="Усечённое среднее (A =
0.05)")
axes.plot(x, IF_trim_mean(x, 0.1, 0, 1), label="Усечённое среднее (A =
0.10)")
axes.plot(x, IF_trim_mean(x, 0.15, 0, 1), label="Усечённое среднее (A =
0.15)")
axes.legend()

plt.show()

# %% [markdown]
# ### 4.3. Графики для метода максимального правдоподобия
#
# Для максимального правдоподобия функция влияния задаётся следующим
образом:
#
# $$
# \text{IF}(y) = \frac{-\lambda f'(\frac{y-\theta}{\lambda})/f(\frac{y -
\theta}{\lambda})}{\int_{-\infty}^{+\infty}[f'(z)]^{2}/f(z)~dz}
# $$
#
# Производная функции $f$ равна
#
# $$
# \frac{df}{dx} = -xe^{-|x|}\frac{\left( |x|^2 + 3|x| + 3 \right) }{96}.
# $$
#

# %%
def IF_MLE(data: np.ndarray, theta: float, lamb: float) -> np.ndarray:
    """
```

```
    Функция влиялия для метода максимального правдоподобия

    Args:
        data (np.ndarray): значения X распределения
        theta (float): параметр тетта смещения
        lamd (float): параметр лямбда распределения

    Returns:
        np.ndarray: значения функции влияния
    """
    fn = lambda x: np.exp(-1 * np.abs(x)) * (np.abs(x)**3 + 6 * np.abs(x)**2
+ 15*np.abs(x) + 15) / 96
    dfn = lambda x: -1 * x * np.exp(-1 * np.abs(x)) * (np.abs(x)**2 +
3*np.abs(x) + 3) / 96
    fndfn = lambda x: dfn(x)**2 / fn(x)

    denumenator = quad(fndfn, -700, 700)[0] # По идее надо -np.inf; np.inf,
но там всё ломается
    x = (data - theta) / lamb

    return -lamb / denumenator * dfn(x) / fn(x)

x = np.linspace(-5, 5, 500)

fig, axes = plt.subplots()
axes.plot(x, IF_MLE(x, 0, 1), label="Макс правдоподобие")
axes.legend()

plt.show()

# %% [markdown]
# ### 4.4. Графики для метода обобщённых радикальных оценок
#
# Для обобщённых радикальных оценок функция влияния задаётся следующим
образом:
#
# $$
# \text{IF}(y) = \frac{-\lambda f'(\frac{y-\theta}{\lambda})f^{\delta -
1}(\frac{y - \theta}{\lambda})}{\int_{-\infty}^{+\infty}[f'(z)]^{2}
f^{\delta - 1}(z)~dz}
# $$
#
# Производная функции $f$ равна
#
# $$
# \frac{df}{dx} = -xe^{-|x|}\frac{\left( |x|^2 + 3|x| + 3 \right) }{96}.
# $$

# %%
def IF_MGD(data: np.ndarray, delta: float, theta: float, lamb: float) ->
np.ndarray:
    """
```

```
    Функция влиялия для метода обобщённых радикальных оценок

    Args:
        data (np.ndarray): значения X распределения
        delta (float): параметр регулировки степени робастности оценки
        theta (float): параметр тетта смещения
        lamd (float): параметр лямбда распределения

    Returns:
        np.ndarray: значения функции влияния
    """
    fn = lambda x: np.exp(-1 * np.abs(x)) * (np.abs(x)**3 + 6 * np.abs(x)**2
+ 15*np.abs(x) + 15) / 96
    dfn = lambda x: -1 * x * np.exp(-1 * np.abs(x)) * (np.abs(x)**2 +
3*np.abs(x) + 3) / 96
    fndfn = lambda x: dfn(x)**2 * fn(x)**(delta - 1)

    denumenator = quad(fndfn, -700, 700)[0] # По идее надо -np.inf; np.inf,
но там всё ломается
    x = (data - theta) / lamb

    return -lamb / denumenator * dfn(x) * fn(x)**(delta - 1)

x = np.linspace(-15, 15, 500)

fig, axes = plt.subplots()
axes.plot(x, IF_MGD(x, 0.1, 0, 1), label="Рад. оценки (d = 0.1)")
axes.plot(x, IF_MGD(x, 0.5, 0, 1), label="Рад. оценки (d = 0.5)")
axes.plot(x, IF_MGD(x, 1.0, 0, 1), label="Рад. оценки (d = 1.0)")
axes.legend()

plt.show()

# %% [markdown]
# ### 4.5. Графики функций влияния всех оценок

# %%
x = np.linspace(-15, 15, 500)

fig, axes = plt.subplots(figsize=(25,15))
axes.plot(x, IF_mean(x, 0), label="Среднее")
axes.plot(x, IF_median(x, 0, 1), label="Медиана")
axes.plot(x, IF_trim_mean(x, 0.05, 0, 1), label="Усечённое среднее (A =
0.05)")
axes.plot(x, IF_trim_mean(x, 0.1, 0, 1), label="Усечённое среднее (A =
0.10)")
axes.plot(x, IF_trim_mean(x, 0.15, 0, 1), label="Усечённое среднее (A =
0.15)")
axes.plot(x, IF_MLE(x, 0, 1), label="Макс правдоподобие")
axes.plot(x, IF_MGD(x, 0.1, 0, 1), label="Рад. оценки (d = 0.1)")
axes.plot(x, IF_MGD(x, 0.5, 0, 1), label="Рад. оценки (d = 0.5)")
axes.plot(x, IF_MGD(x, 1.0, 0, 1), label="Рад. оценки (d = 1.0)")
```

```
axes.legend()

plt.show()
```