Национальный исследовательский университет
Высшая школа экономики
Московский институт электроники и математики

Департамент прикладной математики
кафедра компьютерной безопасности

Долгосрочное домашнее задание

по матетматической статистике

Дискретное распределение: *Дискретное равномерное II*
Непрервное распределение: *Равномерное I*

Выполнил
Гаркин Д.А.

Проверил
Чухно А.Б.

Москва 2023

Оглавление

Домашнее задание 1.

Характеристики вероятностных распределений

1. Дискретное распределение

Функция вероятности распределения (Дискретное равномерное II):
$P(x) = \theta^{-1}, x \in \{\alpha, ..., \alpha + \theta - 1\}$
Параметры дискретного распределения: $\alpha = 234, \theta = 218$

1.1. Описание основных характеристик распредения

1.1.1. Функция распредения

$$F(x) = P(\xi \leq x) = \sum_{i=1}^{\lfloor x \rfloor} P_i(x) = \begin{cases} 0, & x < \alpha \\ (\lfloor x \rfloor - \alpha + 1) \cdot \theta^{-1}, & x \in \{\alpha, ..., \alpha + \theta - 1\} \\ 1, & x > \alpha + \theta - 1 \end{cases}$$

1.1.2. Математическое ожидание

$M_\xi = \sum\limits_{i \geq 1} i \cdot \theta^{-1} = \frac{\alpha + ... + \alpha + \theta - 1}{\theta} = \frac{(2\alpha + \theta - 1) \cdot \theta}{2\theta} = \frac{2\alpha + \theta - 1}{2}$

1.1.3. Дисперсия

$D_\xi = M_{\xi^2} - (M_\xi)^2$
$M_{\xi^2} = \sum\limits_{i \geq 1} i^2 \cdot P_\xi(i) = \big(\sum\limits_{i=1}^{\theta + \alpha - 1} i^2 - \sum\limits_{i=1}^{\alpha - 1} i^2\big)\theta^{-1} = \big(\frac{(\theta + \alpha - 1)(\theta + \alpha)(2\alpha + 2\theta - 1)}{6} - \frac{(\alpha - 1)(\alpha)(2\alpha - 1)}{6}\big)\theta^{-1} = \frac{6\alpha^2\theta + 6\alpha\theta^2 - 6\alpha\theta + 2\theta^3 - 3\theta^2 + \theta}{6} \cdot \theta^{-1} = \frac{6\alpha^2 + 6\alpha\theta - 6\alpha + 2\theta^2 - 3\theta + 1}{6}$
$(M_\xi)^2 = \big(\frac{2\alpha + \theta - 1}{2}\big)^2$
$D_\xi = \frac{6\alpha^2 + 6\alpha\theta - 6\alpha + 2\theta^2 - 3\theta + 1}{6} - \big(\frac{2\alpha + \theta - 1}{2}\big)^2 = \frac{\theta^2 - 1}{12}$

1.1.4. Квантиль уровня γ

$F_\xi(\lfloor x_\gamma \rfloor) = \gamma => (\lfloor x_\gamma \rfloor - \alpha + 1)\theta^{-1} = \gamma => \lfloor x_\gamma \rfloor = \gamma\theta + \alpha - 1$

1.2. Поиск примеров событий, которые могут быть описаны выбранными случайными величинами

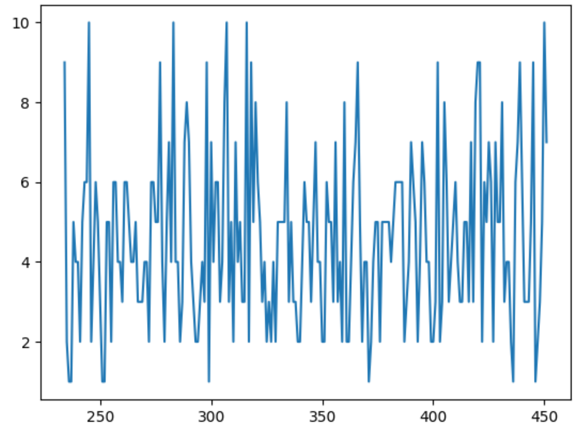
Дискретное равномерное распределение - это симметричное распределение вероятностей, в котором с равной вероятностью будет наблюдаться конечное число значений; каждое из n значений имеет равную вероятность 1 / n. Другой способ сказать «дискретное равномерное распределение» - это «известное конечное число результатов, которые с равной вероятностью произойдут».
Простой пример дискретного равномерного распределения - это бросок честной кости. Возможные значения: 1, 2, 3, 4, 5, 6, и каждый раз, когда бросается игральный кубик, вероятность получения данного результата равна 1/6. Если бросить две кости и сложить их значения, результирующее распределение больше не будет однородным, потому что не все суммы имеют равную вероятность. Хотя удобно описывать дискретные равномерные распределения по целым числам, таким как это, можно также рассматривать дискретные равномерные распределения по любому конечному множеству. Например, случайная перестановка - это перестановка, равномерно сгенерированная из перестановок заданной длины, а однородное остовное дерево - это остовное дерево генерируется равномерно из остовных деревьев данного графа.
Само по себе дискретное равномерное распределение по своей сути непараметрическое. Однако удобно представить его значения в целом всеми целыми числами в интервале [a, b], так что a и b становятся основными параметрами распределения (часто просто рассматривают интервал [1, n] с одним параметр n). Согласно этим соглашениям, кумулятивная функция распределения (CDF) дискретного равномерного распределения может быть выражена для любого k $\in$ [a, b] как
Бета-биномиальное распределение с параметром n и параметрами формы $\alpha = \beta = 1$ представляет собой дискретное равномерное распределение по целым числам от 0 до n.

1.3. Описание способа моделирования выбранных случайных величин

Для начала опишу процесс словами: разобьем отрезок [0,1] на θ частей. тогда они будут разделены значениями: $\{\frac{1}{\theta}, \frac{2}{\theta}, ..., \frac{\theta-1}{\theta}\}$
Тогда теперь нам достаточно определить лишь, в каком отрезке находится случайно сгененированное число из отрезка [0,1]

```python
import random
import matplotlib.pyplot as plt


# параметры распределения
a, t = 234, 218


def generate(n: int):
    res = []

    for i in range(n):
        x = random.random()
        for j in range(t-1, -1, -1):
            if x >= j/t:
                res.append(a + j)
                break
    return res


def create_plot(n: int):
    res = generate(n)
    y_values = []
    x_values = []

    for i in range(a, a+t, 1):
        y_values.append(res.count(i))
        x_values.append(i)

    plt.plot(x_values, y_values)
    plt.show()
```

График получившегося распределения:



2. Непрерывное распределение

Функция плотности распределения (Равномерное I):
$f(x) = \theta^{-1}, x \in [0, \theta], \theta \in \mathbb{R}^+$
Параметр непрерывного распределение: $\theta = 117$

2.1. Описание основных характеристик распредения

2.1.1. Функция распредения

$$F_\xi(x) = \int\limits_{-\infty}^{x} f(t)\mathbf{d}t = \begin{cases} 0, & x < 0 \\ \frac{x}{\theta}, & x \in [0, \theta] \\ 1, & x > \theta \end{cases}$$

2.1.2. Математическое ожидание

$$M_\xi = \int\limits_{-\infty}^{+\infty} t \cdot f(t)dt = \int\limits_{0}^{\theta} \frac{t}{\theta}dt = \frac{t^2}{2\theta}\Big|_0^\theta = \frac{\theta}{2}$$

2.1.3. Дисперсия

$D_\xi = M_{\xi^2} - (M_\xi)^2$

$$M_{\xi^2} = \int\limits_{-\infty}^{+\infty} t^2 \cdot f(t)dt = \int\limits_{0}^{\theta} \frac{t^2}{\theta}dt = \frac{t^3}{3\theta}\Big|_0^\theta = \frac{\theta^2}{3}$$

$D_\xi = \frac{\theta^2}{3} - \frac{\theta^2}{4} = \frac{\theta^2}{12}$

2.1.4. Квантиль уровня γ

$\frac{x_\gamma}{\theta} = \gamma => x_\gamma = \gamma\theta$

2.2. Поиск примеров событий, которые могут быть описаны выбранными случайными величинами

Равномерное распределение реализуется в экспериментах, в которых наудачу ставится точка на отрезке [a,b] (X– абсцисса поставленной точки), а также в экспериментах по измерению тех или иных физических величин с округлением (здесьX – ошибка округления).
Равномерное распределение имеют ошибки грубых измерений при помощи инструментов с крупными делениями, когда измеренное значение округляется до ближайшего целого (или до ближайшего меньшего, или до ближайшего большего). Например, ошибка (в см) измерения длины с помощью линейки с сантиметро- выми делениями имеет равномерное распределение на участке , если округление производится до ближайшего целого, и на участке [0; 1], если до ближайшего меньшего. Также равномерное распределение имеет ошибка (в мин.) указания времени часами со скачущей минутной стрелкой.
К случайным величинам, имеющим равномерное распределение, можно отнести также время ожидания пассажиром транспорта, курсирующего с определенным интервалом [0,T], угол поворота α хорошо уравно- вешенного колеса, если оно приводится во вращение и останавливается в результате трения.
Бета-распределение с параметрами формы $\alpha = \beta = 1$ представляет собой непрерывное равномерное распределение по действительным числам от 0 до 1.

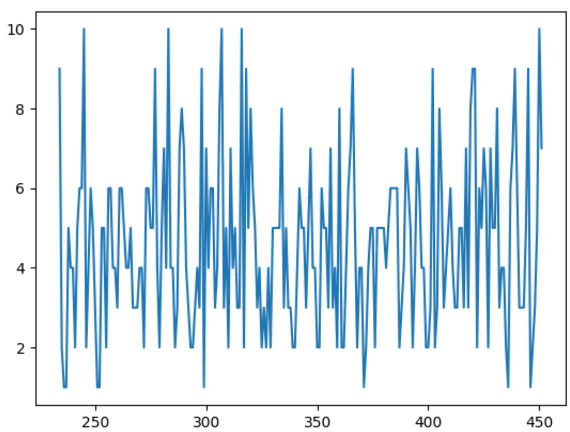
2.3. Описание способа моделирования выбранных случайных величин

Для начала опишу процесс словами: просто умножим полученное число из отрезка [0,1] на θ. так как случайное число (величина) равномерно распределена по определению, то и получившееся будет равномерным.
Заметим, что процесс умножения на θ, по сути, равносилен биективному отображению $[0, 1] \to [0, \theta]$. То есть, каждому числу из отрезка $[0, \theta]$ соотвествует Единсвтенное число из отрзека $[0, 1]$ и наоборот. А, потому, и вероятности у них у всех соответствующие. Таким образом, мы сгенерировали равномерное распределение

```
import random
import matplotlib.pyplot as plt
```

```python
# параметры распределения
t = 117


def generate(n: int):
    res = []

    for i in range(n):
        x = random.random()
        res.append(x*t)
    return res


def create_plot(n: int):
    res = generate(n)
    y_values = [0]*t

    for i in res:
        for j in range(1, t+1):
            if i <= j:
                y_values[j-1] += 1
                break

    plt.plot(range(t), y_values)
    plt.show()
```

График получившегося распределения:



Домашнее задание 2.

Основные понятия математической статистики

1. Дискретное распределение

2.1.1. Генерация выборок заданных случайных величин

252	318	411	308	338

Таблица 2.1: выборка размера 5

295	303	310	362	330	416	414	321	448	376

Таблица 2.2: выборка размера 10

369	391	259	309	277	363	348	384	380	270
267	344	304	260	342	308	249	407	410	262
364	323	250	291	268	325	394	272	442	413
345	354	347	246	358	309	349	246	363	440
420	390	367	429	395	259	431	346	253	254
412	374	321	401	307	392	238	338	384	372
303	243	428	243	316	448	404	319	408	247
378	445	255	282	343	398	308	285	326	341
305	302	355	292	449	302	258	262	437	301
272	239	293	278	448	295	264	234	430	411

Таблица 2.3: выборка размера 100

Программа для генерации выборок:

```
def generate_distribs(n_s):
    dis_s = []
    nepr_s = []

    for i in n_s:
        for j in range(5):
            dis_s.append(Diskretnoe.generate(i))
            nepr_s.append(Neprerivnoe.generate(i))
    return dis_s, nepr_s
```

Переделанные функции генирации по дискретному распределению для нормальной работы:

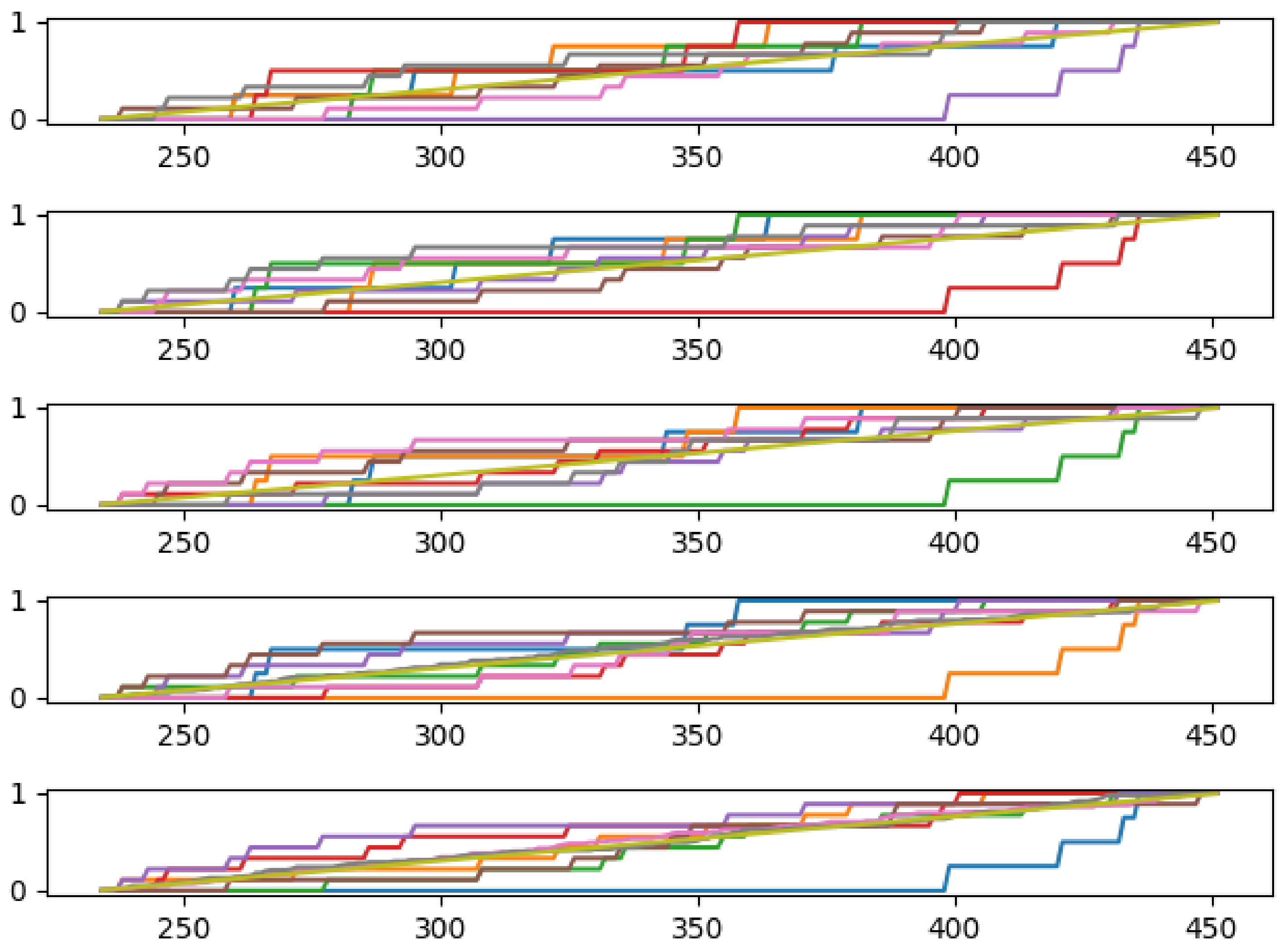
```
def generate(n: int):
    res = []

    for i in range(n):
        x = random.random()
        for j in range(t-1, -1, -1):
            if x >= j/t:
                res.append(a + j)
                break
    return res
```

2.1.2. Построение Эмпирической функции распределения

```
def emp_func_diskr(gen_raspr):
    y_s = []
    rise_points = []

    for raspr in gen_raspr:
        tmp = []
        tmp_rise_points = []
        t = sorted(raspr)
        count = Diskretnoe.a
        for elem_i in range(len(t)):
            while count <= t[elem_i]:
                count += 1
                tmp.append(elem_i/(len(t)-1))
```

```
            if not (count-Diskretnoe.a-1) in tmp_rise_points:
                tmp_rise_points.append(count-Diskretnoe.a-1)
        while len(tmp) < Diskretnoe.t:
            tmp.append(1)
        y_s.append(tmp)
        rise_points.append(tmp_rise_points)

    x = [i for i in range(Diskretnoe.a, Diskretnoe.a + Diskretnoe.t, 1)]
    return y_s, x, list(map(len, gen_raspr)), rise_points
```

Очень простой алгоритм: я просто отсортировал выборку и для каждого значения X (от α до θ) посчитал количество элементов. получился вот такой вот график:



код, с помощью которого это было построено:

```
def graph_emp(y_s, x):
    fig, axs = plt.subplots(5)
```

```
for i in range(0, 5):
    for j in range(8):
        axs[i].plot(x, y_s[i+j])
    axs[i].plot(x, y_s[-1])

fig.tight_layout()
plt.show()
```

2.1.3. $D_{m,n}$

Код, с помощью которого все считалось:

```
def D_mn(distr, dims, rise_points):
    Ds = []
    for i in range(len(distr)):
        for j in range(i+1, len(distr)):
            f_1, f_2 = distr[i], distr[j]
            r_p_1, r_p_2 = rise_points[i], rise_points[j]
            r_p = sorted(r_p_1 + r_p_2)

            while len(f_1) < len(f_2):
                f_1.append(1)
            while len(f_2) < len(f_1):
                f_2.append(1)
            n, m = dims[i], dims[j]

            const = sqrt((n*m)/(m+n))
            sup = -1

            for k in r_p:
                sup = max(sup, abs(f_1[k] - f_2[k]))
            Ds.append([const*sup, n, m])
    return Ds
```

Вот то, что получилось при выполнении такого кода (показан срез данных):

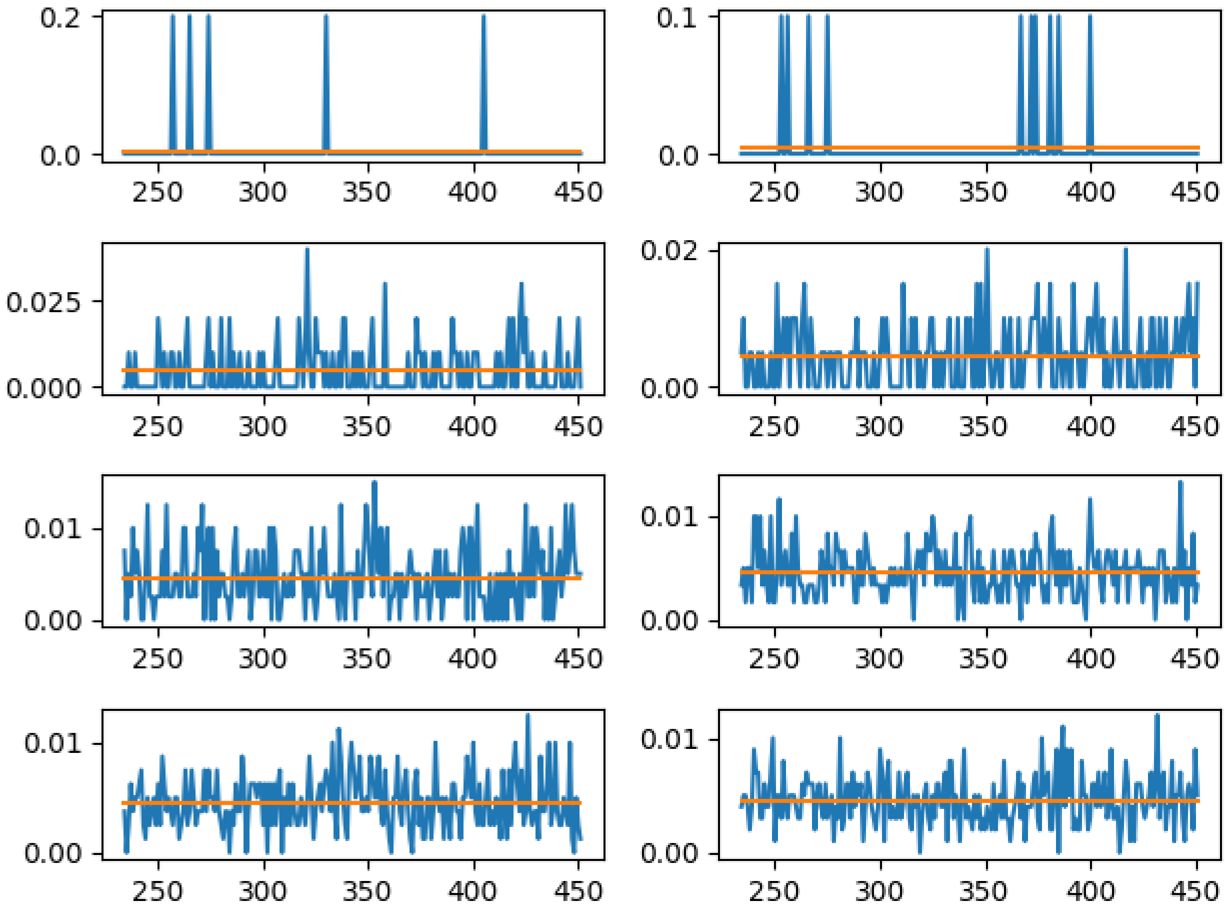
(0.7905694150420949, 5, 5)
(0.5071505162084872, 5, 10)
(0.6722874396159361, 5, 100)
(0.7491585939750895, 5, 200)
(0.6989696463380675, 5, 400)
(0.6747342363259172, 5, 600)
(0.6765455146409164, 5, 800)
(0.7088922243077421, 5, 1000)
(0.7905694150420949, 5, 5)
(0.9635859807961257, 5, 10)
(1.3225326680969234, 5, 100)
(1.1320618753401355, 5, 200)
(1.2642717905875802, 5, 400)
(1.2528123286051465, 5, 600)
(1.2805541905986833, 5, 800)
(1.2972169521977894, 5, 1000)
(0.8114408259335795, 5, 10)
(0.8816884453979489, 5, 100)
(0.6770173960367476, 5, 200)
(0.7630186577554998, 5, 400)
(0.736073712355546, 5, 600)
(0.7700064414057441, 5, 800)
(0.7368013669970233, 5, 1000)
(0.4057204129667897̶3, 5, 10)
(1.0414944761263272, 5, 100)
(0.9683568492492824, 5, 200)
(1.003898635477583, 5, 400)
(0.9712083704691231, 5, 600)

(0.9548358861272316, 5, 800)
(0.9963563940073383, 5, 1000)
(1.622881651867159, 5, 10)
(1.5429547794464107, 5, 100)
(1.4095280212568353, 5, 200)
(1.5873015873015874, 5, 400)
(1.6431544487936938, 5, 600)
(1.6962460738213494, 5, 800)
(1.600868424657169, 5, 1000)
(0.49690399499995325, 10, 10)
(0.7918479756587733, 10, 100)
(0.9701036965790928, 10, 200)
(0.8872026573454648, 10, 400)
(0.873798527524712, 10, 600)
(0.8906711290707112, 10, 800)
(0.9008238128172174, 10, 1000)
(0.49690399499995325, 10, 10)
(0.7613922842872818, 10, 100)
(0.7409317753623621, 10, 200)
(0.8950309160867482, 10, 400)
(0.8371478569294678, 10, 600)
(0.8946044166868232, 10, 800)
(0.9165724808734624, 10, 1000)
(1.2182276548596513, 10, 100)
(1.442232316228598, 10, 200)
(1.456056125878733, 10, 400)
(1.497441684319979, 10, 600)
(1.498582581738699, 10, 800)
(1.5087223997882766, 10, 1000)
(1.0659491980021945, 10, 100)
(0.9563189193630487, 10, 200)
(1.0176736363668564, 10, 400)
(1.0227282366101487, 10, 600)
(0.915144918682075, 10, 800)
(1.0047650219884348, 10, 1000)
(0.5482024446868432, 10, 100)
(0.46351313388947757, 10, 200)
(0.5010085594421447, 10, 400)
(0.5061283082200397, 10, 600)
(0.6398147855542301, 10, 800)
(0.503957377799842, 10, 1000)
(0.999948983496128, 100, 100)
(0.599700803310705, 100, 200)
(0.7479033472248107, 100, 400)
(0.6842834854898052, 100, 600)
(0.7444640742495756, 100, 800)
(0.7453129167003045, 100, 1000)
(0.8570991287109667, 100, 100)
(0.6527496649719143, 100, 200)
(0.8457217686602082, 100, 400)
(0.7198793409978307, 100, 600)
(0.9891622401852752, 100, 800)
(0.9223138887688274, 100, 1000)
(0.7530451690501385, 100, 200)
(0.6392162122965906, 100, 400)
(0.5120807283610678, 100, 600)
(0.5160950114474329, 100, 800)
(0.6464006087796581, 100, 1000)
(1.3469437531166488, 100, 200)
(1.0087524710525362, 100, 400)
(0.9317371301398945, 100, 600)
(0.8085885814455848, 100, 800)
(0.9821818646155346, 100, 1000)
(1.2155649316588097, 100, 200)
(1.0060352926793314, 100, 400)
(1.0221318684694851, 100, 600)
(0.7246784456351936, 100, 800)
(1.027299759456531, 100, 1000)
(1.5577889447236182, 200, 200)
(1.6585886374498258, 200, 400)
(1.7711388405339061, 200, 600)
(1.6137458842778496, 200, 800)
(1.8653736922820316, 200, 1000)
(1.0050251256281406, 200, 200)
(0.7165161084362378, 200, 400)
(0.7474785395570344, 200, 600)
(0.7354735371037076, 200, 800)
(0.4558720041712746, 200, 1000)
(0.5584375596499185, 200, 400)
(0.6251078260707912, 200, 600)
(0.48853899311561666, 200, 800)
(0.7643324058541174, 200, 1000)
(0.968394716070004, 200, 400)
(1.020783407628748, 200, 600)
(0.6684098062461169, 200, 800)
(1.0448040847737372, 200, 1000)

(0.6248974463061728, 200, 400)
(0.5558568933338222, 200, 600)
(0.61494975033117, 200, 800)
(0.6251031213892714, 200, 1000)
(0.6379910807698173, 400, 400)
(0.6622611762829651, 400, 600)
(0.6443848663003435, 400, 800)
(0.8115241543668796, 400, 1000)
(0.5316592339748485, 400, 400)
(0.6321200930910702, 400, 600)
(0.7742350758449661, 400, 800)
(0.605558077251348, 400, 1000)
(1.1407589743959956, 400, 600)
(1.0050969671339713, 400, 800)
(1.1842731388934455, 400, 1000)
(0.4967120870956595, 400, 600)
(0.9488029315167944, 400, 800)
(0.6200609387653501, 400, 1000)
(0.8556178454473059, 400, 600)
(1.4823413073025165, 400, 800)
(0.7481695488057097, 400, 1000)
(0.5204827134597627, 600, 600)
(0.5999054104388907, 600, 800)
(0.49803738377887263, 600, 1000)
(0.6361455386730448, 600, 600)
(0.9986301079742459, 600, 800)
(0.8104838386446944, 600, 1000)
(1.3658235976218345, 600, 800)
(1.126651820419845, 600, 1000)
(0.8829511980547118, 600, 800)
(0.8620027155112288, 600, 1000)
(1.1323765934564614, 600, 800)
(0.7123326115490922, 600, 1000)
(0.8760951188986232, 800, 800)
(0.9261793350223344, 800, 1000)
(1.1514392991239053, 800, 800)
(0.8768686777523472, 800, 1000)
(0.8681264026819844, 800, 1000)
(0.49471768683672357, 800, 1000)
(0.6179283007437925, 800, 1000)
(1.0520039533782806, 1000, 1000)
(0.8505563878377586, 1000, 1000)

Теперь проанализируем полученные результаты: Расстояние Колмагорова не велико, что говорит об общей схожести выборок (ну так, они же из одного распределения))). Те которые больше 1 - они просто сильно отличаются (отличие в размерах и в эмпирических функциях распределения). Думаю, их можно считать выбросами

2.1.4. Построение Гистограммы и полигона частот



Вот код, которые это делает:

```
def diskr_get_frequency_range(raspr):
    y_1 = []
    for i in range(Diskretnoe.t):
        y_1.append(1/Diskretnoe.t)
    fig, axs = plt.subplots(4, 2)
    for i in range(0, len(raspr), 5):

        x_values, y_values = Diskretnoe.create_plot(raspr[i])
        axs[i//10, (i%10)//5].plot(x_values, y_values)
        axs[i // 10, (i % 10) // 5].plot(x_values, y_1)
    fig.tight_layout()
    plt.show()
```

Применяемая функция:

```
def create_plot(raspr):
    y_values = []
    x_values = []

    for i in range(a, a+t, 1):
        y_values.append(raspr.count(i))
        x_values.append(i)

    return (x_values, y_values)
```

Видно, что чем больше выборка, тем больше полигон частот напоминает график функции вероятности. Так и должно быть.
Это выполняется за счет теоремы Гливенко-Кантели ($\sup\limits_{x\in\mathbb{R}} |F^*(x) - F(x)| \to 0, n \to \infty$)

2.1.5. Вычисление выборочных моментов

Вычисление выборочного среднего производится с помощью кода:

```
def sample_mean(raspr):
    res = []
    for i in raspr:
        res.append(sum(i)/len(i))
    return res
```

Вот то, что он выводит:

269.2	343.47	346.825	342.79875
352.2	343.28	343.1275	338.79125
365.0	342.09	340.0275	337.8016666666667
341.8	337.67	345.1275	342.43
364.6	346.42	340.14	349.305
358.9	347.145	341.79	343.309
334.6	343.61	341.325	344.129
322.1	341.655	343.3125	341.763
341.8	337.53	343.525	342.2316666666667
328.8	334.22	344.44375	343.02666666666664

Вычисление выборочной дисперсии производится с помощью кода:

```
def sample_variance(raspr, means):
    res = []
    for i in range(len(raspr)):
        S = 0
        for j in raspr[i]:
            S += (j - means[i])**2
        res.append(S/len(raspr[i]))
    return res
```

Вот то, что он выводит:

4758.96	3722.8570999999997	4125.168400000004
1271.3600000000001	3801.843600000001	4093.899900000003
1375.3600000000001	3994.360000000004	3983.6745484374974
4139.76	3843.7835999999998	3782.023873437499
4399.039999999999	3889.107774999998	3977.8273437499865
4456.6	3874.4799999999964	4086.2507749999972
4271.4400000000005	3862.8879749999983	3780.221060937498
2897.7599999999998	4108.424375000005	3812.172923999998
5164.09	4192.668693749997	3902.3485440000018
3757.2400000000007	3903.6936937499995	4122.038815999997
3520.8819000000008	4183.408943749998	3846.082095999999
4327.87	3964.5861972222215	3900.1178789999944
3926.7300000000014	3752.7206638888892	
3973.755599999998	4118.637600000003	

Математическое ожидание вычисляется по формуле: $\frac{\alpha+\alpha+\theta}{2}$, что равно $\alpha + \frac{\theta}{2} = 343$. Видим, что есть значения очень близкие к теоритическому и, чем больше выборка, тем лучше результаты (за исключением некоторых выбросов)

Дисперсия вычисляется по формуле: $\frac{\theta^2-1}{12}$, что равно 3960,25. Видим, что есть значения очень близкие к теоритическому и, чем больше выборка, тем лучше результаты (за исключением некоторых выбросов)

Все это так же иллюстрирует ЗБЧ: Закон больших чисел утверждает, что среднее значение выборки сходится к математическому ожиданию случайной величины по мере увеличения размера выборки.

Эти оценки обладают свойствами:

1. **Состоятельность:**

$$\forall\epsilon > 0 \to \lim_{n\to\infty} P(|\overline{X} - M(x_1)| > \epsilon) = 0$$

$$\forall\epsilon > 0 \to \lim_{n\to\infty} P(|\frac{1}{n}\sum_{i=1}^{n} x_i - \alpha - \frac{\theta}{2}| > \epsilon) =$$

$$\lim_{n\to\infty} P(|\frac{1}{n}\frac{\alpha+\alpha+\theta}{2}\cdot n - \alpha - \frac{\theta}{2}| > \epsilon) = 0$$

$$\forall\epsilon > 0 \to \lim_{n\to\infty} P(|\overline{S}^2 - D(x_1)| > \epsilon) = 0$$

Неравенство Чебышева:

$$P(|\overline{S}^2 - D_\theta| > \epsilon) < \frac{D_\theta}{n} \to 0, n \to \infty$$

как выборочное среднее, так и выборочная дисперсия стремятся к истинным значениям (естественно, с увеличением выборки)

2. **Несмещенность:**
$M_\theta\overline{X} = M_\theta\frac{\sum x_i}{n} = \frac{1}{n}\sum M_\theta x_i = \frac{1}{n}\sum\limits_{i=1}^{n}\frac{2\alpha+\theta-1}{2} = \frac{2\alpha+\theta-1}{2} = M_\theta x_i =>$ оценка в виде выборочного среднего является несмещенной.

$$M_\theta\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})^2 = \frac{1}{n}\sum_{i=1}^{n}(M_\theta(x_i)^2 - 2M_\theta(x_i\overline{X}) + M_\theta(\overline{X})^2) =$$

$$\frac{1}{n}\sum_{i=1}^{n}(M_\theta(x_i^2) - 2(M_\theta(x_i))^2 + M_\theta(M(x_i))^2) = \frac{1}{n}D_\theta(X)$$

Получается, что оценка смещена, При том, на $\frac{n-1}{n}D_\theta(X)$

2. Непрерывное распределение

2.2.1. Генерация выборок заданных случайных величин

116.93467306155557	114.18182388823688	62.79263767630565
20.16118519167	65.6754749839694	

Таблица 2.4: размерность 5

81.01841917212971	93.98033216585513	49.114817437895
63.114908160073604	84.70124347672925	72.4955103191863
33.483385402736396	4.761125041973717	23.70576653407245
61.97679613537279		

Таблица 2.5: размерность 10

Программа для генерации выборок:

```
def generate_distribs(n_s):
    dis_s = []
    nepr_s = []

    for i in n_s:
        for j in range(5):
            dis_s.append(Diskretnoe.generate(i))
            nepr_s.append(Neprerivnoe.generate(i))
    return dis_s, nepr_s
```

Переделанные функции генерации по непрерывному распределению для нормальной работы:

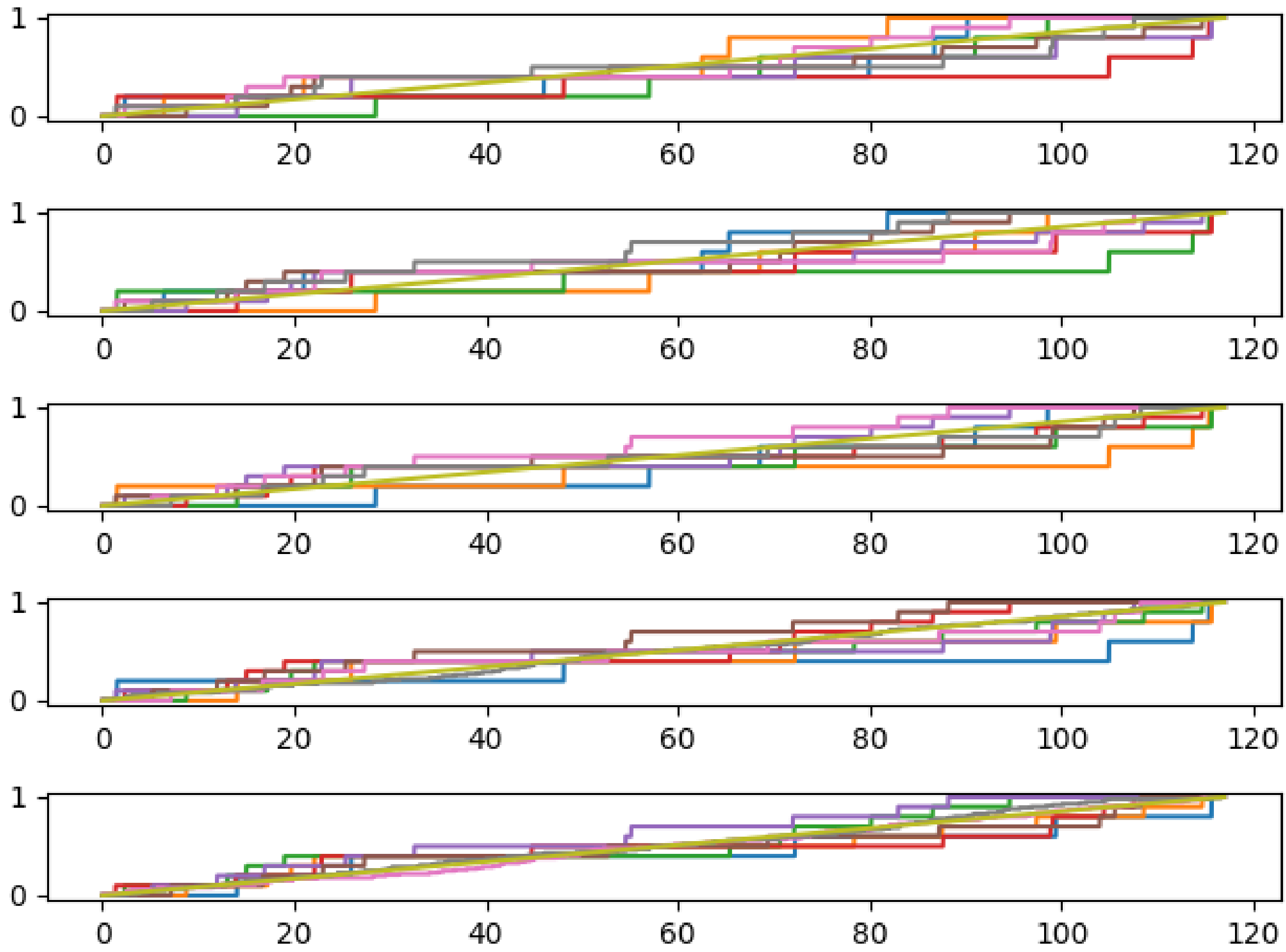
```
def generate(n: int):
    res = []

    for i in range(n):
        x = random.random()
        res.append(x*t)
    return res
```

2.2.2. Построение Эмпирической функции распределения

```
def emp_func_nepr(gen_raspr):
    y_s = []
```

```
    x = [0]
    rise_points = []

    while x[-1] <= Neprerivnoe.t:
        x.append(x[-1] + 0.001)

    for i in gen_raspr:
        tmp = []
        tmp_rise_points = []
        t = sorted(i)
        count = 0
        for j in range(len(t)):
            while t[j] >= x[count]:
                count += 1
                tmp.append(j/len(t))
            tmp_rise_points.append(count-1)
        while len(tmp) < len(x):
            tmp.append(1)
        y_s.append(tmp)
        rise_points.append(tmp_rise_points)

    return y_s, x, list(map(len, gen_raspr)), rise_points
```

Очень простой алгоритм: я просто бегу итератором с шагом 0.001 и для каждого значения считаю количесво элементов в отсортированной выборке меньше данного. получился вот такой вот график:



код, с помощью которого это было построено:

```
def graph_emp(y_s, x):
    fig, axs = plt.subplots(5)
    for i in range(0, 5):
        for j in range(8):
            axs[i].plot(x, y_s[i+j])
        axs[i].plot(x, y_s[-1])

    fig.tight_layout()
    plt.show()
```

Выводы:
Ассимптотика стремления графика к теоретической функции распределения по теореме Глевенко-Кантеле равна $\frac{1}{\sqrt{n}}$

2.2.3. $D_{m,n}$

Код, с помощью которого все считалось:

```
def D_mn(distr, dims, rise_points):
    Ds = []
    for i in range(len(distr)):
        for j in range(i+1, len(distr)):
            f_1, f_2 = distr[i], distr[j]
            r_p_1, r_p_2 = rise_points[i], rise_points[j]
            r_p = sorted(r_p_1 + r_p_2)

            while len(f_1) < len(f_2):
                f_1.append(1)
            while len(f_2) < len(f_1):
                f_2.append(1)
            n, m = dims[i], dims[j]

            const = sqrt((n*m)/(m+n))
            sup = -1

            for k in r_p:
                sup = max(sup, abs(f_1[k] - f_2[k]))
            Ds.append([const*sup, n, m])
    return Ds
```

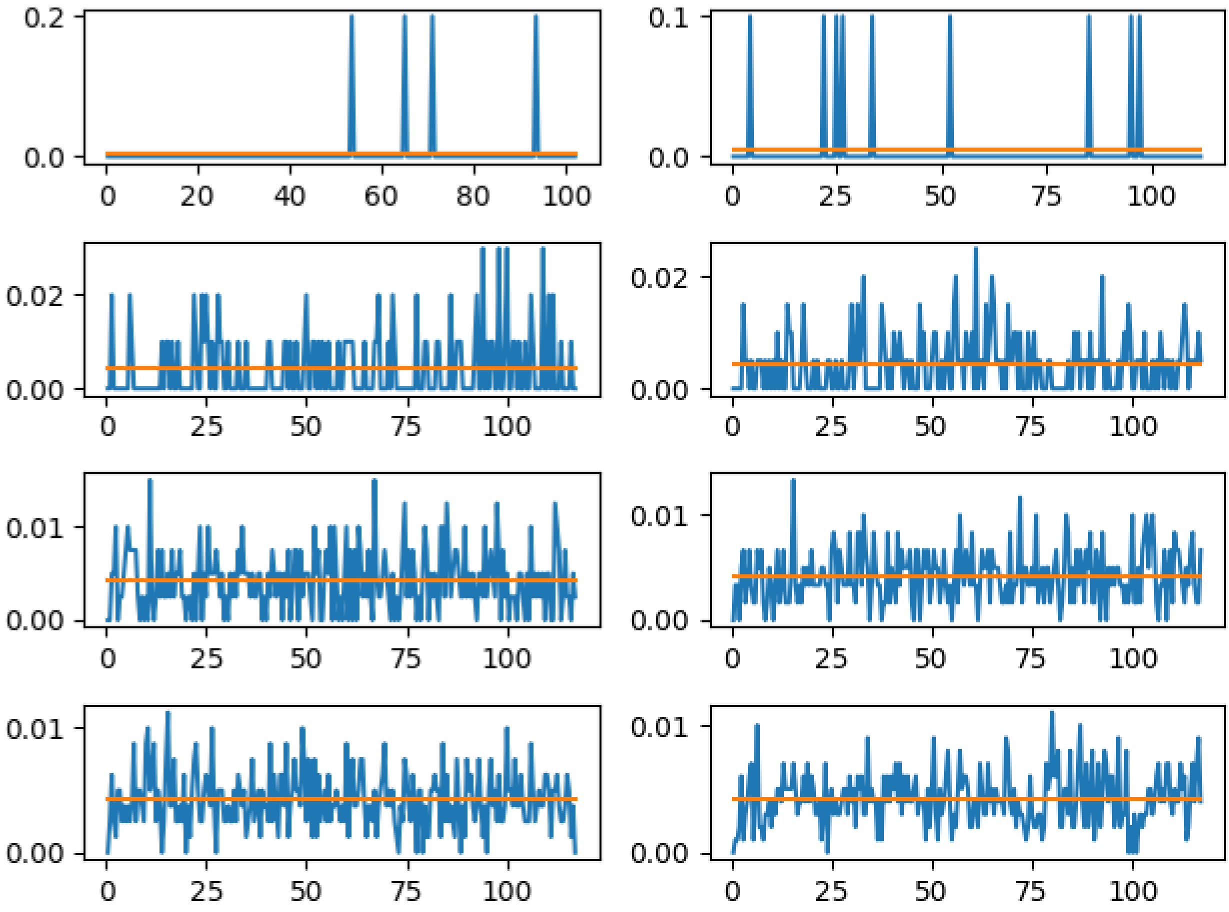
Вот то, что получилось при выполнении такого кода (показан срез данных):

(0.948683298050514, 5, 5)
(1.0954451150103324, 5, 10)
(1.0692676621563626, 5, 100)
(0.9828405820661343, 5, 200)
(0.9444444444444445, 5, 400)
(1.068868251396296, 5, 600)
(0.9613049166924836, 5, 800)
(1.005954906361036, 5, 1000)
(0.9486832980505138, 5, 5)
(1.0954451150103324, 5, 10)
(1.025624084109164, 5, 100)
(1.1484787711784041, 5, 200)
(1.0944444444444443, 5, 400)
(1.0428888147304136, 5, 600)
(1.1061972519620753, 5, 800)
(1.09517485371013, 5, 1000)
(0.7302967433402214, 5, 10)
(0.7637626158259734, 5, 100)
(0.8171932929538642, 5, 200)
(0.8055555555555556, 5, 400)
(0.7868057961667178, 5, 600)
(0.8498492741774131, 5, 800)
(0.8052100248255744, 5, 1000)
(0.9128709291752769, 5, 10)
(0.5673665146135801, 5, 100)
(0.7840638351314104, 5, 200)
(0.6055555555555555, 5, 400)
(0.6420632204568028, 5, 600)

(0.5823557321412437, 5, 800)
(0.6334616261785682, 5, 1000)
(1.0954451150103324, 5, 10)
(0.6764754597315763, 5, 100)
(0.4638124095143555, 5, 200)
(0.5555555555555556, 5, 400)
(0.44165042331999727, 5, 600)
(0.5628509947011063, 5, 800)
(0.5197061933084731, 5, 1000)
(1.565247584249853, 10, 10)
(1.567858991804371, 10, 100)
(1.4813121596360825, 10, 200)
(1.4524159855640364, 10, 400)
(1.651758460226241, 10, 600)
(1.4534972724390145, 10, 800)
(1.557559019430693, 10, 1000)
(0.894427190999916, 10, 10)
(0.6331738236133035, 10, 100)
(0.8332380897952965, 10, 200)
(0.7418253689708787, 10, 400)
(0.7056563042105776, 10, 600)
(0.7778174593052022, 10, 800)
(0.7237142918566856, 10, 1000)
(0.7236272269866327, 10, 100)
(0.6480740698407863, 10, 200)
(0.640312423743285, 10, 400)
(0.6690667180663254, 10, 600)
(0.6049691350151574, 10, 800)
(0.6387565271604662, 10, 1000)
(0.6030226891555273, 10, 100)
(0.9258200997725516, 10, 200)
(0.7418253689708788, 10, 400)
(0.8101979789084409, 10, 600)
(0.7346053782326911, 10, 800)
(0.7803528016541653, 10, 1000)
(0.9045340337332911, 10, 100)
(0.7560864148142504, 10, 200)
(0.8589556903873333, 10, 400)
(0.8990584024016249, 10, 600)
(0.852456508430449, 10, 800)
(0.8590173985951094, 10, 1000)
(1.0606601717798214, 100, 100)
(0.9797958971132713, 100, 200)
(1.0285912696499033, 100, 400)
(1.465881824639873, 100, 600)
(0.9074537025227366, 100, 800)
(1.0488088481701514, 100, 1000)
(0.5656854249492378, 100, 100)
(0.8164965809277259, 100, 200)
(0.6260990336999412, 100, 400)
(0.7715167498104594, 100, 600)
(0.8013876853447541, 100, 800)
(0.7246315678266503, 100, 1000)
(0.5715476066494078, 100, 200)
(0.8273451516749225, 100, 400)
(0.5554920598635303, 100, 600)
(0.8485281374238572, 100, 800)
(0.7341661937191062, 100, 1000)
(0.9797958971132713, 100, 200)
(0.6484597134749386, 100, 400)
(0.6635044048369951, 100, 600)
(0.6481812160876689, 100, 800)
(0.5148697981926192, 100, 1000)
(0.4898979485566361, 100, 200)
(1.0285912696499033, 100, 400)
(0.9721111047611789, 100, 600)
(1.2020815280171304, 100, 800)
(1.0297395963852396, 100, 1000)
(0.8499999999999996, 200, 200)
(0.8948929172439202, 200, 400)
(0.8573214099741131, 200, 600)
(1.2649110640673515, 200, 800)
(1.0715253924507195, 200, 1000)
(0.7499999999999996, 200, 200)
(0.6350852961085883, 200, 400)
(0.5511351921262142, 200, 600)
(0.5217758139277827, 200, 800)
(0.5551276129563955, 200, 1000)
(0.8082903768654762, 200, 400)
(1.2247448713915894, 200, 600)
(0.8696263565463048, 200, 800)
(1.0198856145012867, 200, 1000)
(0.6350852961085877, 200, 400)
(1.041033140682851, 200, 600)
(0.5059644256269412, 200, 800)
(0.6842270578299768, 200, 1000)

(0.6639528095680697, 200, 400)
(0.612372435695795, 200, 600)
(0.8221921916437787, 200, 800)
(0.52930772398168, 200, 1000)
(0.6010407640085668, 400, 400)
(1.2522646152737311, 400, 600)
(0.46948553403344245, 400, 800)
(0.7437357441610953, 400, 1000)
(0.5656854249492386, 400, 400)
(0.8778762251403492, 400, 600)
(0.6327848502189888, 400, 800)
(0.6423172335936732, 400, 1000)
(1.4717336715588185, 400, 600)
(0.8777338244973059, 400, 800)
(1.0987005311470717, 400, 1000)
(1.1360751148875086, 400, 600)
(0.6531972647421805, 400, 800)
(0.7606388292556647, 400, 1000)
(0.671317113342619, 400, 600)
(0.9185586535436921, 400, 800)
(0.7352842016138091, 400, 1000)
(1.0392304845413267, 600, 600)
(1.4813121596360828, 600, 800)
(0.8197814749472365, 600, 1000)
(1.2413030787576957, 600, 600)
(1.535318332122814, 600, 800)
(1.400728976878349, 600, 1000)
(0.6094982323502628, 600, 800)
(0.942425947577138, 600, 1000)
(0.5786375623578447, 600, 800)
(0.6454972243679027, 600, 1000)
(1.033832444746015, 600, 800)
(0.826236447190917, 600, 1000)
(0.8250000000000002, 800, 800)
(0.9803060746521961, 800, 1000)
(0.6500000000000006, 800, 800)
(0.8011103405759902, 800, 1000)
(0.5428576649955711, 800, 1000)
(0.537587202228625, 800, 1000)
(0.6746192341692543, 800, 1000)
(0.5142956348249514, 1000, 1000)
(0.5366563145999494, 1000, 1000)

2.2.4. Построение Гистограммы и полигона частот



Вот код, который это делает:

```
def nepr_get_frequency_range(raspr):
    fig, axs = plt.subplots(4, 2)
    for i in range(0, len(raspr), 5):
        x_values, y_values = Neprerivnoe.create_plot(raspr[i])
        axs[i // 10, (i % 10) // 5].plot(x_values, y_values)
    fig.tight_layout()
    plt.show()
```

Применяемая функция:

```
def create_plot(raspr):
    rp = sorted(raspr)
    y_values = []
    x_values = []
    number = 0
    count = 0
```

```
    for i in range(len(rp)):
        while number < rp[i]:
            print(rp[i], number)
            number += 0.5
            y_values.append(count)
            x_values.append(number)
            count = 0
        count += 1

    return (x_values, y_values)
```

Видно, что чем больше выборка, тем больше полигон частот напоминает график функции вероятности. Так и должно быть.
Это выполняется за счет теоремы Гленевко-Кантели ($\sup\limits_{x\in\mathbb{R}} |F^*(x) - F(x)| \to 0, n \to \infty$)

2.2.5. Вычисление выборочных моментов

Вычисление выборочного среднего производится с помощью кода:

```
def sample_mean(raspr):
    res = []
    for i in raspr:
        res.append(sum(i)/len(i))
    return res
```

Вот то, что он выводит:

64.75146434472164	60.67255866465042	58.58120992551063
72.15356260333613	57.805769437677455	57.87070821813645
64.73118975046268	59.81972501146823	58.12948967408372
64.62222414946905	58.964951685176345	58.010958134468204
88.3074163185711	62.53550248459022	57.65297196641613
47.996982651137486	58.53577274591262	57.27146778594104
69.21954061878057	61.454272619825254	59.08997382295189
55.823242181340376	58.10458443402046	59.05503762080276
77.7534893972359	57.43702534645371	60.83208643264995
50.74769587813061	60.28518462184048	57.86618902157024
55.17189164483355	59.4499624400634	59.24502157005728
57.717916102961496	58.81176179673298	57.23271855015885
66.81956480156896	58.75974834633596	59.724573115803594

59.529329091168904

Вычисление выборочной дисперсии производится с помощью кода:

```
def sample_variance(raspr, means):
    res = []
    for i in range(len(raspr)):
        S = 0
        for j in raspr[i]:
            S += (j - means[i])**2
        res.append(S/len(raspr[i]))
    return res
```

Вот то, что он выводит:

1563.1011623624659
318.00603941194885
1030.2144873186041
1313.992370034874
1462.0039348696507
1312.7070821788677
957.9212719749343
1567.671361902339
664.7758238166263
1045.2681020530192
1154.9930758533073
1168.2424357200323
1223.8398739829943
1307.9702422164794
1215.3993514199149
1179.121093911478
1044.2951544735886
1124.9667035220543
1197.935568160755
1204.5591687667568
1132.3435312358927
1194.8012519927665
1238.5445080052202
1115.253125334508
1105.4101918854744
1046.1070443207216
1136.9141245072065
1171.5462832275034
1048.5597805353007
1239.7106733058645
1136.6916065258438
1141.663912862754
1175.5268527088353
1119.5030796903236
1122.1907196728505
1141.5213117707144
1183.114288324195
1198.025805263531
1165.8496029623793
1157.7255757217022

Математическое ожидание вычисляется по формуле: $\frac{\theta}{2}$, что равно 58,5. Видим, что есть значения очень близкие к теоритическому и, чем больше выборка, тем лучше результаты (за исключением некоторых выбросов)

Дисперсия вычисляется по формуле: $\frac{\theta^2-1}{12}$, что равно 1140,66. Видим, что есть значения очень близкие к теоритическому и, чем больше выборка, тем лучше результаты (за исключением некоторых выбросов)

Домашнее задание 3.

Построение точечных оценок параметра распределения

Задание: Необходимо получить оценки неизветного параметра методом моментов и методом максимального правдоподобия.
Для каждой выборки, сгенерированной в пункте 2.1, необходимо привести значения полученных оценок.

1. Дискретное распределение

3.1.1. Получение оценок методом моментов и методом максимального правдоподобия.

Если все моменты существуют (конечны) и тотображение, заданное системой, является биекцией, то метод моментов обладает следующими свойствами:
- состоятельность
- ассимптотическая нормальность

Оценка методом моментов:

$$M(X) = \sum_{x=\alpha}^{\alpha+\theta-1} x \cdot P(x) = \alpha + \frac{\theta - 1}{2}$$

Выборочный момент первого порядка:

$$\overline{\theta} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Тогда для нахождения этого параметра нужно решить уравнение:

$$\alpha + \frac{\theta - 1}{2} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

```
def moment_method_diskr(raspr):
    res = []
    means = sample_mean(raspr)
    for i in means:
```

```
        res.append((i - Diskretnoe.a)*2+1)
    return res
```

Вывод програмы:
Оценка методом моментов длины 5: 118.60000000000002
Оценка методом моментов длины 5: 177.0
Оценка методом моментов длины 5: 240.60000000000002
Оценка методом моментов длины 5: 279.0
Оценка методом моментов длины 5: 232.60000000000002
Оценка методом моментов длины 10: 217.79999999999995
Оценка методом моментов длины 10: 165.60000000000002
Оценка методом моментов длины 10: 222.0
Оценка методом моментов длины 10: 203.39999999999998
Оценка методом моментов длины 10: 254.39999999999998
Оценка методом моментов длины 100: 218.53999999999996
Оценка методом моментов длины 100: 199.55999999999995
Оценка методом моментов длины 100: 216.24
Оценка методом моментов длины 100: 204.41999999999996
Оценка методом моментов длины 100: 216.96000000000004
Оценка методом моментов длины 200: 221.77999999999997
Оценка методом моментов длины 200: 212.87
Оценка методом моментов длины 200: 220.27999999999997
Оценка методом моментов длины 200: 214.51999999999998
Оценка методом моментов длины 200: 204.63
Оценка методом моментов длины 400: 221.27499999999998
Оценка методом моментов длины 400: 223.11
Оценка методом моментов длины 400: 211.24
Оценка методом моментов длины 400: 219.45500000000004
Оценка методом моментов длины 400: 213.515
Оценка методом моментов длины 600: 220.9766666666667
Оценка методом моментов длины 600: 217.04999999999995
Оценка методом моментов длины 600: 218.15666666666664
Оценка методом моментов длины 600: 211.75333333333333
Оценка методом моментов длины 600: 216.69333333333338
Оценка методом моментов длины 800: 215.81500000000005
Оценка методом моментов длины 800: 213.72000000000003
Оценка методом моментов длины 800: 221.8025
Оценка методом моментов длины 800: 213.8075
Оценка методом моментов длины 800: 223.73249999999996
Оценка методом моментов длины 1000: 222.88
Оценка методом моментов длины 1000: 219.34000000000003
Оценка методом моментов длины 1000: 214.48400000000004
Оценка методом моментов длины 1000: 216.062

Оценка методом моментов длины 1000: 220.664
Видно, что с увеличением размера выборки получаемое значение приближается к теоретическому, с которым генерировалось: 218

Оценка методом максимального правдоподобия:

$$L(\theta; x_1, x_2, x_3, ..., x_n) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \theta^{-1} \cdot I(x_i \in [\alpha, \alpha + \theta - 1]) =$$

$$\theta^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha + \theta + 1)$$

$$lnL(\theta, X) = \sum_{i=1}^{n} ln(\theta^{-1} \cdot I(x_i \in [\alpha, \alpha+\theta-1])) = \sum_{i=1}^{n} ln(\theta^{-1}) + \sum_{i=1}^{n} ln(I(x_i \in [\alpha, \alpha+\theta-1]))$$

$$= nln(\theta^{-1}) + \sum_{i=1}^{n} ln(I(x_i \in [\alpha, \alpha + \theta - 1]))$$

$$\frac{d}{d\theta} lnL(\theta, X) = n\theta \cdot (-1)\theta^{-2} = \frac{-n}{\theta}$$

этот результат можно оценить максимумом из всех $x_i = x_{(n)} = \alpha + \theta - 1$

```
def max_pod_diskr(raspr):
    res = []
    for i in raspr:
        res.append(max(i))
    return res
```

Вывод программы:
Оценка методом максимального правдоподобия длины 5: 441
Оценка методом максимального правдоподобия длины 5: 385
Оценка методом максимального правдоподобия длины 5: 448
Оценка методом максимального правдоподобия длины 5: 371
Оценка методом максимального правдоподобия длины 5: 448
Оценка методом максимального правдоподобия длины 10: 387
Оценка методом максимального правдоподобия длины 10: 412
Оценка методом максимального правдоподобия длины 10: 447
Оценка методом максимального правдоподобия длины 10: 381
Оценка методом максимального правдоподобия длины 10: 432
Оценка методом максимального правдоподобия длины 100: 449
Оценка методом максимального правдоподобия длины 100: 451
Оценка методом максимального правдоподобия длины 100: 451
Оценка методом максимального правдоподобия длины 100: 451
Оценка методом максимального правдоподобия длины 100: 446
Оценка методом максимального правдоподобия длины 200: 449
Оценка методом максимального правдоподобия длины 200: 451

Оценка методом максимального правдоподобия длины 200: 450
Оценка методом максимального правдоподобия длины 200: 450
Оценка методом максимального правдоподобия длины 200: 451
Оценка методом максимального правдоподобия длины 400: 450
Оценка методом максимального правдоподобия длины 400: 451
Оценка методом максимального правдоподобия длины 400: 451
Оценка методом максимального правдоподобия длины 400: 451
Оценка методом максимального правдоподобия длины 400: 451
Оценка методом максимального правдоподобия длины 600: 451
Оценка методом максимального правдоподобия длины 600: 451
Оценка методом максимального правдоподобия длины 600: 451
Оценка методом максимального правдоподобия длины 600: 451
Оценка методом максимального правдоподобия длины 600: 451
Оценка методом максимального правдоподобия длины 800: 451
Оценка методом максимального правдоподобия длины 800: 451
Оценка методом максимального правдоподобия длины 800: 451
Оценка методом максимального правдоподобия длины 800: 451
Оценка методом максимального правдоподобия длины 800: 451
Оценка методом максимального правдоподобия длины 1000: 451
Оценка методом максимального правдоподобия длины 1000: 451
Оценка методом максимального правдоподобия длины 1000: 451
Оценка методом максимального правдоподобия длины 1000: 451
Оценка методом максимального правдоподобия длины 1000: 451
Видно, что реальные значения с величением выборки стремятся, и, даже в точности равны, теоретическим значениям (451).

3.1.2. Поиск оптимальных оценок

Теорема:
Для того, чтобы несмещенная оценка $t = t(x)$ для $\tau(\theta)$ была эффективной, необходимо и досточно, чтобы выполнялось следующее представление:

$$\frac{dlnL}{d\theta} = A(\theta)(t(x) - \tau(\theta))$$

Где $A(\theta)$ - некоторая функция, зависящая от θ
Определение:
Достаточная статистика называетсая полной, если для любой функции ϕ из того, что $M_\theta(\phi(T)) = 0$ следует, что ϕ - тождественно равна нулю.
Теорема:
Если существует полная достаточная статистика, то произвольная функция от нее будет являться оптимальной оценкой своего математического ожидания.

$$L(\overline{x},\theta) = g(T(\overline{x}),\theta)\cdot h(\overline{x})$$

$$f(x,\theta) = \theta^{-1}\cdot I(x\in[\alpha,\alpha+\theta-1]) =>$$

$$L(x,\theta) = \prod \theta^{-1} I(x\in[\alpha,\alpha+\theta-1]) = \theta^{-n}\cdot I(x_{(1)}\geq\alpha)\cdot I(x_{(n)}\leq\alpha+\theta-1)$$

Отсюда можно сделать вывод, что нашей оцентой должен быть парарметр:

$$T(x) = x_{(n)}$$

Тогда мы имеем, что:

$$g(T(x),\theta) = \theta^{-n}\cdot I(x_{(n)}\leq\alpha+\theta-1)$$

$$h(x) = I(x_{(1)}\geq\alpha)$$

Далее пользуемся критерием факторизации: Для того, чтобы статистика $T = T(x)$ была достаточной необходимо и достаточно, чтобы функция правдоподобия представлялась в следующем виде: $L(\overline{x},\theta) = g(T(x),\theta)\cdot h(x)$, где g, h - не отрицательные функции и h - не зависит от θ

Тепень надо понять, какой же вид имеет наша оценка. Заметим, что:

$$M(\phi(x_{(n)})) = 0$$

$$F_{x_{(n)}}(t) = (F(t))^n = (\frac{t}{\theta})^n => f_{x_{(n)}}(t) = F_{x_{(n)}}(t) - F_{x_{(n)}}(t-1) = \frac{1}{\theta^n}(t^n - (t-1)^n)$$

$$\forall t\in\overline{\alpha,\alpha+\theta-1}$$

$$\sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\frac{1}{\theta^n}(t^n-(t-1)^n) = 0 => \sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\cdot t^n(1-(1-\frac{1}{t})^n) = 0 =>$$

$$\sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\cdot t^n - \sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\cdot(t-1)^n = 0 =>$$

$$\phi(\alpha)\cdot\alpha^n + \phi(\alpha+1)\cdot(\alpha+1)^n + \phi(\alpha+2)\cdot(\alpha+2)^n + ... + \phi(\alpha+\theta-1)\cdot(\alpha+\theta-1)^n -$$

$$\phi(\alpha)\cdot(\alpha-1)^n - \phi(\alpha+1)\cdot\alpha^n - \phi(\alpha+2)\cdot(\alpha+1)^n - ... - \phi(\alpha+\theta-1)\cdot(\alpha+\theta-2)^n =>$$

$$\alpha^n(\phi(\alpha)-\phi(\alpha+1)) + ... + (\alpha+\theta-2)^n(\phi(\alpha+\theta-2)-\phi(\alpha+\theta-1)) +$$

$$+\phi(\alpha+\theta-1)(\alpha+\theta-1)^n - \phi(\alpha)(a-1)^n =$$

$$\sum_{t=\alpha}^{\alpha+\theta-2} t^n(\phi(t)-\phi(t+1)) + \phi(\alpha+\theta-1)(\alpha+\theta-1)^n - \phi(\alpha)(a-1)^n$$

короче тут можно проще!

$$\sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\frac{1}{\theta^n}(t^n-(t-1)^n)=0 => \sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\frac{1}{\theta^n}t^n=\sum_{t=\alpha}^{\alpha+\theta-1}\phi(t)\frac{1}{\theta^n}(t-1)^n$$

это верно тогда и только тогда, когда $\phi(t)=0\forall t\in\overline{\alpha,\alpha+\theta-1}$
Тогда можно смело утверждать, опираясь на теорему Рао-Бернулли-Колмагорова: Оптимальная оценка, если существует, является функцией от достаточной статистики.
Запишем:

$$MH(x_{(n)})=\theta => \sum_{\alpha}^{\alpha+\theta-1}H(t)\frac{1}{\theta^n}(t^n-(t-1)^n)=\theta =>$$

$$\sum_{t=\alpha}^{\alpha+\theta-1}H(t)t^n=\theta^{n+1}+\sum_{t=\alpha}^{\alpha+\theta-1}H(t)(t-1)^n$$

$$\sum_{t=\alpha}^{\alpha+\theta-1}H(t)t^n=\sum_{t=\alpha}^{\alpha+\theta-1}(H(t)(t-1)^n+\theta^n)$$

$$H(t)t^n=H(t)(t-1)^n+\theta^n => H(t)=\frac{\theta^n}{t^n-(t-1)^n}$$

$$H(x_{(n)})=\frac{\theta^n}{x_{(n)}^n-(x_{(n)}-1)^n}$$

Хорошая попытка, но оно зависит от тетта. Теперь новый вариант: на вот этом моменте

$$\sum_{\alpha}^{\alpha+\theta-1}H(t)\frac{1}{\theta^n}(t^n-(t-1)^n)=\theta$$

Заметим, что

$$\frac{1}{\theta^n}\sum_{i=\alpha}^{\alpha+\theta-1}(i^n-(i-1)^n)=1 => \sum_{i=\alpha}^{\alpha+\theta-1}(i^n-(i-1)^n)=\theta^n$$

А значит, что чтобы получить

$$\sum_{\alpha}^{\alpha+\theta-1}H(t)\frac{1}{\theta^n}(t^n-(t-1)^n)=\theta$$

Нужно чтобы внутри суммы (если вынести $\frac{1}{\theta^n}$) θ^{n+1} =>

$$H(i)=\frac{i^{n+1}-(i-1)^{n+1}}{i^n-(i-1)^n} =>$$

$$H(x_{(n)})=\frac{x_{(n)}^{n+1}-(x_{(n)}-1)^{n+1}}{x_{(n)}^n-(x_{(n)}-1)^n}$$

2. Непрерывное распределение

Так уж вышло, что у меня дискретное и непрерывное распределения - одни и те же. Отличаются только границы и параметр. Так что, все применяемые теоремы и методы будут одни и те же, поменяются лишь вычисления.

Параметры тут: $[0, \theta]$, $\theta = 117$

3.2.1. Получение оценок методом моментов и методом максимального правдоподобия.

Оценка методом моментов:

$$M(X) = \int_0^\theta t \cdot \theta^{-1} dt = \frac{\theta}{2}$$

Выборочный момент первого порядка:

$$\overline{\theta} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Тогда для нахождения этого параметра нужно решить уравнение:

$$\frac{\theta}{2} = \frac{1}{n} \sum_{i=1}^{n}$$

```
def moment_method_nepr(raspr):
    res = []
    means = sample_mean(raspr)
    for i in means:
        res.append(i*2)
    return res
```

Вывод программы:
Оценка методом моментов длины 5: 49.2870273561051
Оценка методом моментов длины 5: 37.07863963955923
Оценка методом моментов длины 5: 93.91597231650425
Оценка методом моментов длины 5: 160.42715879865787
Оценка методом моментов длины 5: 89.38965153269979
Оценка методом моментов длины 10: 103.56789216300687
Оценка методом моментов длины 10: 116.9554763931449
Оценка методом моментов длины 10: 135.69882655958247
Оценка методом моментов длины 10: 128.35876378711467
Оценка методом моментов длины 10: 86.58337038352578

Оценка методом моментов длины 100: 115.10251562001618
Оценка методом моментов длины 100: 124.25310599114977
Оценка методом моментов длины 100: 117.43881823876569
Оценка методом моментов длины 100: 114.71767329587193
Оценка методом моментов длины 100: 119.21098248067445
Оценка методом моментов длины 200: 126.21444776220831
Оценка методом моментов длины 200: 115.9176054990178
Оценка методом моментов длины 200: 118.72057403093692
Оценка методом моментов длины 200: 114.00583413698139
Оценка методом моментов длины 200: 123.17693361047883
Оценка методом моментов длины 400: 117.10215829360888
Оценка методом моментов длины 400: 114.13487379518426
Оценка методом моментов длины 400: 120.15346654789958
Оценка методом моментов длины 400: 114.6080462610332
Оценка методом моментов длины 400: 122.35304912048669
Оценка методом моментов длины 600: 121.25716208068644
Оценка методом моментов длины 600: 120.10644667013219
Оценка методом моментов длины 600: 117.65650251324446
Оценка методом моментов длины 600: 117.3589637403698
Оценка методом моментов длины 600: 121.91416334056613
Оценка методом моментов длины 800: 119.19958218496737
Оценка методом моментов длины 800: 114.69300914201534
Оценка методом моментов длины 800: 119.3800933065129
Оценка методом моментов длины 800: 116.95301653077773
Оценка методом моментов длины 800: 117.7429494501997
Оценка методом моментов длины 1000: 117.47658178770483
Оценка методом моментов длины 1000: 121.1022539527951
Оценка методом моментов длины 1000: 118.87979917811604
Оценка методом моментов длины 1000: 112.63137226159525
Оценка методом моментов длины 1000: 118.64992885678151
Видно, что с увеличением размера выборки получаемое значение приближается к теоретическому, с которым генерировалось: 117

Оценка методом максимального правдоподобия:

$$L(\theta; x_1, x_2, x_3, ..., x_n) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \theta^{-1} \cdot I(x_i \in [0, \theta]) =$$

$$\theta^{-n} \cdot I(x_{(1)} \geq 0) \cdot I(x_{(n)} \leq \theta)$$

Вот тут давайте представим, как выглядит график этого дела: это будет просто прямая горизонтальная линия на уровне θ^{-n} на отрезке $[0, \theta]$. Наша задача - методом максимального правдоподобия найти максимум, зависимый от θ. так как это горизонтальная линия - тут все есть максимум, но, по сути, только 1 место зависит от θ: $x_{(n)} \leq \theta$. так что мы получаем, что мы можем оценить θ как $x_{(n)}$

$$lnL(\theta, X) = \sum_{i=1}^{n} ln(\theta^{-1} \cdot I(x_i \in [0, \theta])) = \sum_{i=1}^{n} ln(\theta^{-1}) + \sum_{i=1}^{n} ln(I(x_i \in [0, \theta]))$$

$$= nln(\theta^{-1}) + \sum_{i=1}^{n} ln(I(x_i \in [0, \theta]))$$

Мы генерировали наше распределение так, что все элементы попадают в отрезок $[0, \theta]$ => $\sum_{i=1}^{n} ln(I(x_i \in [0, \theta]) = \sum_{i=1}^{n} ln(1) = 0$. Теперь мы избавились от индикаторов и можем смело считать производную для поиска максимума.

$$\frac{d}{d\theta} lnL(\theta, X) = \frac{d}{d\theta}(-n)ln(\theta) = \frac{-n}{\theta}$$

этот результат можно оценить максимумом из всех $x_i = x_{(n)} = \theta$

```
def max_pod_nepr(raspr):
    res = []
    for i in raspr:
        res.append(max(i))
    return res
```

Вывод программы:
Оценка методом максимального правдоподобия длины 5: 115.63061331259054
Оценка методом максимального правдоподобия длины 5: 113.46248146215741
Оценка методом максимального правдоподобия длины 5: 71.62945494067884
Оценка методом максимального правдоподобия длины 5: 100.72629172341861
Оценка методом максимального правдоподобия длины 5: 103.9651044401549
Оценка методом максимального правдоподобия длины 10: 100.06977345252568
Оценка методом максимального правдоподобия длины 10: 110.30645049164137
Оценка методом максимального правдоподобия длины 10: 103.3098304447322
Оценка методом максимального правдоподобия длины 10: 87.28547326674587
Оценка методом максимального правдоподобия длины 10: 107.97188113921983
Оценка методом максимального правдоподобия длины 100: 116.94578552976174
Оценка методом максимального правдоподобия длины 100: 115.96227015987787
Оценка методом максимального правдоподобия длины 100: 113.72669580366878
Оценка методом максимального правдоподобия длины 100: 114.79923804788868
Оценка методом максимального правдоподобия длины 100: 114.19321262488124
Оценка методом максимального правдоподобия длины 200: 116.36302139230948
Оценка методом максимального правдоподобия длины 200: 116.84169042816427
Оценка методом максимального правдоподобия длины 200: 116.51210353635857
Оценка методом максимального правдоподобия длины 200: 116.67279185376869

Оценка методом максимального правдоподобия длины 200: 115.91891587490423
Оценка методом максимального правдоподобия длины 400: 116.9786947713713
Оценка методом максимального правдоподобия длины 400: 116.88661794485292
Оценка методом максимального правдоподобия длины 400: 116.91721738294227
Оценка методом максимального правдоподобия длины 400: 116.49812496373961
Оценка методом максимального правдоподобия длины 400: 116.85730504686238
Оценка методом максимального правдоподобия длины 600: 116.51089495599614
Оценка методом максимального правдоподобия длины 600: 116.9764082337241
Оценка методом максимального правдоподобия длины 600: 116.91691821206136
Оценка методом максимального правдоподобия длины 600: 116.84731300560449
Оценка методом максимального правдоподобия длины 600: 116.39179941513885
Оценка методом максимального правдоподобия длины 800: 116.99595420297285
Оценка методом максимального правдоподобия длины 800: 116.86773244452112
Оценка методом максимального правдоподобия длины 800: 116.89168303465934
Оценка методом максимального правдоподобия длины 800: 116.86354928775776
Оценка методом максимального правдоподобия длины 800: 116.81651190070698
Оценка методом максимального правдоподобия длины 1000: 116.78082955225953
Оценка методом максимального правдоподобия длины 1000: 116.98864396858512
Оценка методом максимального правдоподобия длины 1000: 116.94994746737888
Оценка методом максимального правдоподобия длины 1000: 116.87601800713078
Оценка методом максимального правдоподобия длины 1000: 116.89825536938316
Видно, что реальные значения с величением выборки стремятся к теоретическим значениям (117).

3.2.2. Поиск оптимальных оценок

$$L(\overline{x}, \theta) = g(T(\overline{x}), \theta) \cdot h(\overline{x})$$

Заметим то, что $\sqrt{f(x, \theta)} = \sqrt{\theta^{-1}}$ - ограниченная, что является достаточным условием для того, чтобы можно было утвердлать, что оптимальная оценка существует и представима в виде формулы, что написана выше.

$$f(x, \theta) = \theta^{-1} \cdot I(x \in [0, \theta]) =>$$
$$L(x, \theta) = \prod \theta^{-1} I(x \in [0, \theta]) = \theta^{-n} \cdot I(x_{(1)} \geq 0) \cdot I(x_{(n)} \leq \theta)$$

Отсюда можно сделать вывод, что нашей оцентой должен быть парарметр:

$$T(x) = x_{(n)}$$

Тогда мы имеем, что:

$$g(T(x), \theta) = \theta^{-n} \cdot I(x_{(n)} \leq \theta)$$

$$h(x) = I(x_{(1)} \geq 0)$$

Тепень надо понять, какой же вид имеет наша оценка. Заметим, что:

$$M(\phi(x_{(n)})) = 0$$

$$F_{x_{(n)}}(t) = (F(t))^n = (\frac{t}{\theta})^n => f_{x_{(n)}} = \frac{nt^{n-1}}{\theta^n}, t \in [0, \theta]$$

$$\int\limits_0^\theta \phi(t)\frac{nt^{n-1}}{\theta^n}dt = 0$$

продифференцируем это дело по θ
Лирическое отсутпление в матан:

$$F(\theta) = \int\limits_{a(\theta)}^{b(\theta)} f(t, \theta)dt$$

$$F'_\theta = f(b(\theta), \theta)b'_\theta(\theta) - f(a(\theta), \theta)a'_\theta(\theta) + \int\limits_{a(\theta)}^{b(\theta)} \frac{d}{d\theta}f(t, \theta)dt$$

теперь к подсчетам:

$$\int\limits_0^\theta \phi(t)\frac{nt^{n-1}}{\theta^n}dt = 0 => \int\limits_0^\theta \phi(t)t^{n-1}dt = 0$$

теперь заметим, что тут 2е и 3е слогаемые занулятся, так что можно просто посчитать.

$$\phi(\theta)(\theta)^{n-1} = 0$$

Тогда:

$$\phi(\theta) = 0$$

Тогда мы понимаем, что $x_{(n)}$ - полная достаточная статистика.
То есть, для любого ϕ это будет оптимальная оценка.

$$M(H(x_{(n)})) = \theta$$

$$\int\limits_0^\theta H(t)\frac{nt^{n-1}}{\theta^n}dt = \theta =>$$

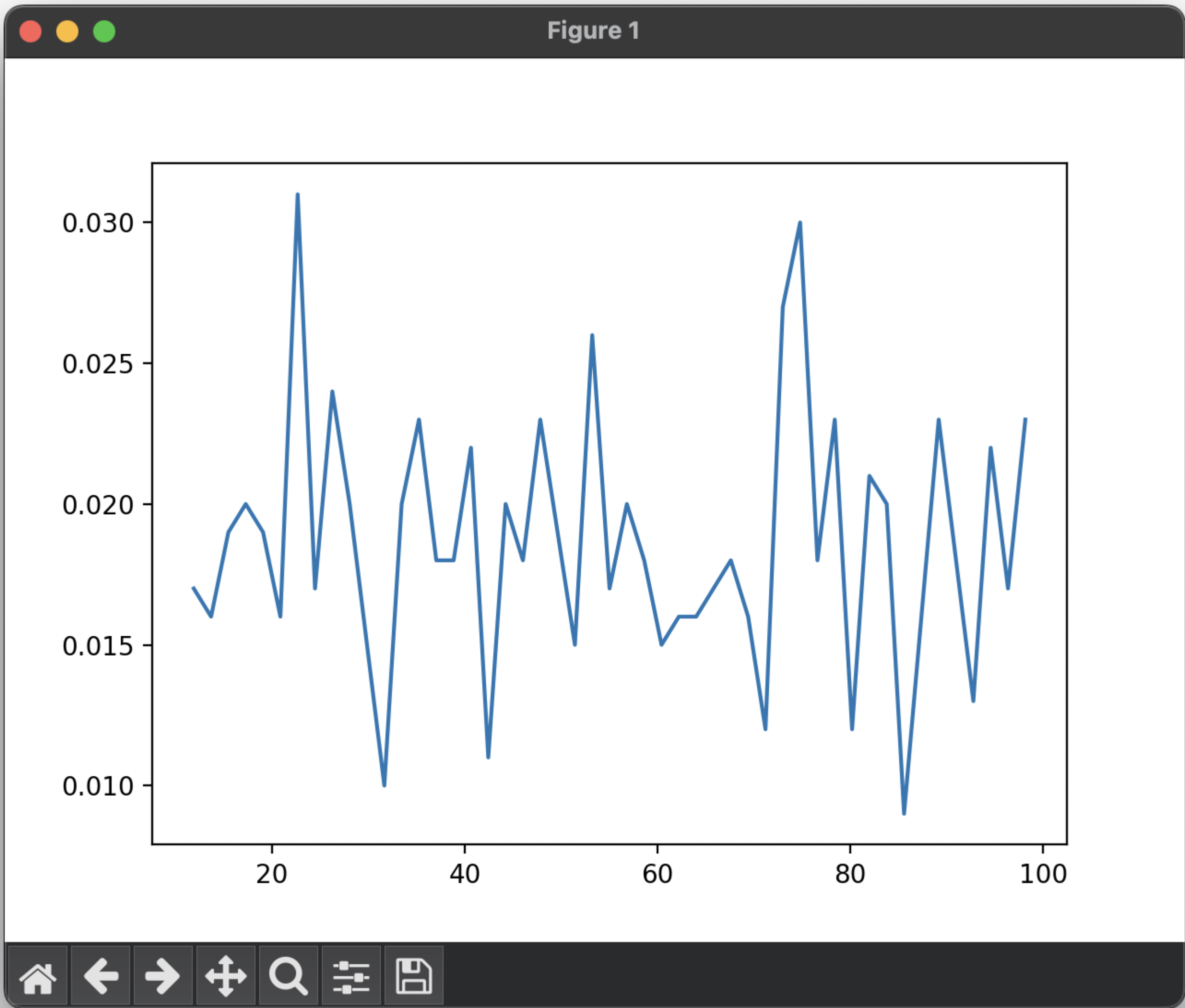
$$\int\limits_0^\theta H(t)nt^{n-1}dt = \theta^{n+1} =>$$

$$H(\theta)n\theta^{n-1} = (n+1)\theta^n => H(\theta) = \theta\frac{n+1}{n} =>$$

$$H(x_{(n)}) = x_{(n)} \cdot \frac{n+1}{n}$$

3. Работа с данными

Я очень долго искал и наконец нашел хоть что-то похожее на равновмерное распределение. Это распределение цен в магазине: Данные. Отсюда я пользовался колонкой "Unit price". Далее привожу ее полигон частот:



Он строился с помощью данного кода (важно, что данные были мной харанее отсортированы):

```
def freq_range(raspr):
    _raspr = sorted(raspr)
    # print(_raspr)
    delta = (_raspr[-1] - _raspr[0])/50
    # print(delta)
    tmp = _raspr[0] + delta
    x = []
    y = []
    count = 0

    for i in _raspr:
        # print('1')
        if i <= tmp:
            count += 1
        else:
            y.append(count / len(_raspr))
            count = 0
            x.append(tmp)
            tmp += delta
    # print(y)
    plt.plot(x, y)
    plt.show()
```

Оно и правда выглядит +- как равномерное, да и по логике, думаю, пожходит: распределение цен за каждый продукт.

Плотность распределения будем определять по формуле:

$$f(x) = \frac{1}{x_{(n)} - x_{(1)}}$$

теперь посчитаем это значение:

```
def get_max_min(reaspr):
    return(reaspr[-1], reaspr[0])
```

Здачения максимума, минимума: $99.96; 10.08$ Функция распределения:

$$f(x) = \frac{1}{89.88}$$

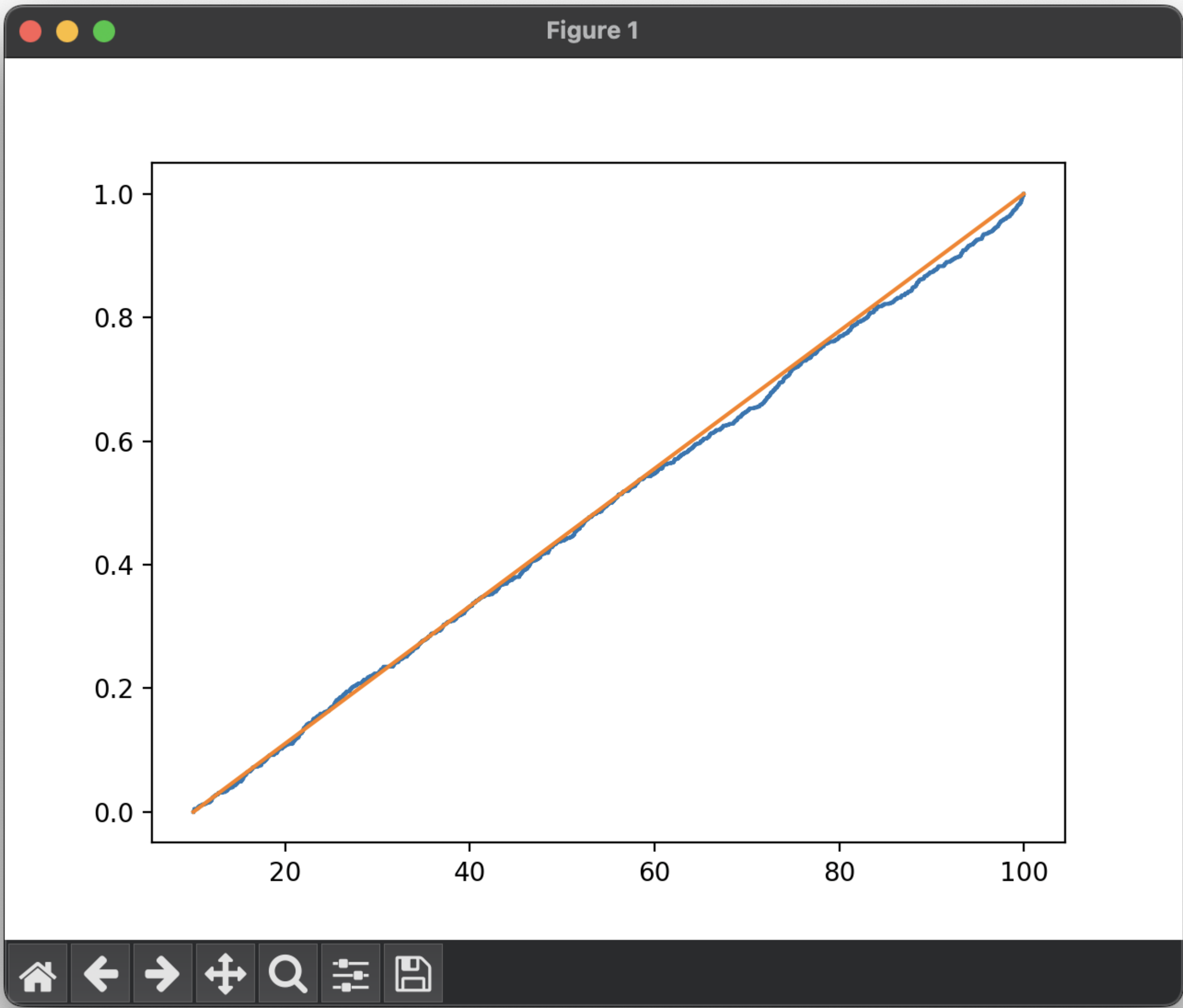
Теперь вычислим Мат. Ожидание:

$$M_\xi = \int\limits_{-\infty}^{+\infty} t \cdot f(t) dt = \int\limits_{0}^{x_{(n)}-x_{(1)}} \frac{t}{x_{(n)} - x_{(1)}} dt = \frac{t^2}{2(x_{(n)} - x_{(1)})} \Big|_0^{x_{(n)}-x_{(1)}} = \frac{x_{(n)} - x_{(1)}}{2} = 44.94$$

Далее для простоты обозначим $x_{(n)} - x_{(1)}$ за θ Теперь построим Эмпирическую функцию распределениея:

```
# Построение эмпирической функции распределения непрерывного распределения
def emp_func_nepr(gen_raspr):
    y_s = []
    x = [min(gen_raspr[0])]
    rise_points = []

    while x[-1] <= max(gen_raspr[-1]):
        x.append(x[-1] + 0.001)

    for i in gen_raspr:
        tmp = []
        tmp_rise_points = []
        t = sorted(i)
        count = 0
        for j in range(len(t)):
            while t[j] >= x[count]:
                count += 1
                tmp.append(j/len(t))
            tmp_rise_points.append(count-1)
        while len(tmp) < len(x):
            tmp.append(1)
        y_s.append(tmp)
        rise_points.append(tmp_rise_points)

    return y_s, x

def gen_raspr_func():
    res = []
    count = 10.08

    while count <= 99.96:
        res.append((count - 10.08)/89.88)
        count += 0.001
    res.append(1)
    return res

# отрисовка графика эмпирической функции
def graph_emp(y_s, x):
    plt.plot(x,y_s[0])
    plt.plot(x,y_s[1])
```

```
plt.show()
```



Можно заметить, что графики очень похожи, это радует.

Домашнее задание 4.

Проверка статистических гипотез

1. Теория

4.1.1. Критерий согласия Колмогорова (Смирнова)

Критерий Колмогорова основан на тереме Колмогорова. Статистика критерия определяется формулой:

$$D_n = D_n\left(X\right) = \sup_{x\in\mathbb{R}} \left| F_n(x) - F(x) \right|$$

где D_n — это отклонение эмпирической функции распределения от теоретической функции.
Мы знаем, что F_n является оптимальной, несмещенной и состоятельной оценкой для $F(x)$. Отсюда следует, что D_n не должно «сильно» отклоняться от 0.
По теореме Колмогорова для непрерывных функций распределения F и при $20 \le n$:

$$P(\sqrt{n}D_n \ge \lambda_\alpha | H_0) = 1 - K(\lambda_\alpha) = \alpha$$

«сильно» отклоняться от 0.
При этом по значению α возможно однозначно определить величину λ_α Критерий формулируется следующим образом. Проверяем, выполняется ли неравенство: $\sqrt{n}D_n \ge \lambda_\alpha$, если да, то отвергаем гипотезу H_0. Данному критерию соответсвует критическая область

$$\mathbf{X}_1 = \overline{x} : D_n(\overline{x})\sqrt{n} \ge \lambda_\alpha$$

Вместо статистики $D_n\sqrt{n} \ge \lambda_\alpha$при малых значениях n рекомендуется использовать статистику

$$S_n = \frac{6nD_n + 1}{6\sqrt{n}}$$

которая также сходится к распределению Колмогорова. Опишем способ вычисления значения $D_n = \sup_{x\in\mathbb{R}\left|\cup F_n(x) - F(x)\right|}$ Вычисление супремума функции, вообще говоря, не является тривиальной задачей. Однако в виду того, что $\cup F_n(x)$ принимает конечное число значений: $0, \frac{1}{n}, \frac{2}{n}, ..., \frac{n}{n}$, задача нахождения супремума функции сильно упрощается. Пусть имеется вариационный ряд реализации выборки: $x_{(1)},x_{(2)},...,x_{(n)}$. Определим следующие две функции:

$$D_n^+ = \max_{1\leq k\leq n} \left| \frac{k}{n} - F(x_{(k)}) \right|,$$

$$D_n^- = \max_{1\leq k\leq n} \left| F(x_{(k)}) - \frac{k-1}{n} \right|.$$

Тогда вычислить значение D_n можно следующим образом:

$$D_n = \max D_n^+, D_n^-$$

4.1.2. Критерий согласия хи-квадрат

Данный критерий позволяет проверять как простую гипотезу о согласии, когда мы проверяем гипотезу о согласии с некоторым полностью заданным законом распределения с точностью до параметров:

$$H_0 : F_n(x) = F(x, \theta)$$

А также и сложную гипотезу, когда мы проверяем согласие с некоторым семейством распределений и нам необходимо оценить параметры этого семейства по имеющейся у нас выборке:

$$H_0 : F_n(x) \in F(x, \theta \in \Theta$$

Критерий подразумевает, что мы работаем с группированными данными. То есть первым этапом в применении данного критерия является разбиение области определения нашей случайной величины на k непересекающихся интервалов:

$$x_{(0)}, x_{(1)}, x_{(2)}, .., x_{(k)}$$

После того как мы разбили область определения случайной величины на интервалы мы должны посчитать количество наблюдений, которые попадают в каждые из интервалов: После этого мы должны рассчитать наблюдаемую частоту попадания в каждый из интервалов, то есть отношение Затем считаем теоретическую вероятность попадания в каждый интерва- лов соответствии с тем законом распределения, с которым мы проверяем со- гласие, то есть через функцию распределения

$$P_i(\theta) = F(x_i, \theta) - F(x_{i-1}, \theta)$$

Статистика критерия имеет вид:

$$\xi_n^2 = n \sum_{i=1}^{k} \frac{\left(\frac{n_i}{n} - P_i(\theta)\right)^2}{P_i(\theta)}$$

В пределе она имеет Хи-квадрат распределение с количеством степеней свободы, которое зависит от вида проверяемой гипотезы. То есть, если мы про- веряем простую гипотезу о согласии, тогда данная статистика имеет Хи-квад- рат распределение с k – 1 степенями свободы, где k – количество интервалов, на которое мы разбили область определения случайной величины. Если же мы проверяем сложную гипотезу о согласии и оценивали параметры по выборке, то данная статистика будем иметь Хи-квадрат распределение с k – m – 1 степе- нями свободы, где m – количество параметров, которые мы оцениваем по вы- борке. Например, если мы проверяли гипотезу о согласии с нормальным зако- ном распределения и оценивали по выборке параметры сдвига и параметры масштаба, то m в данном случае будет равно двум. В зависимости от значения критерия 2 гипотеза H0 может приниматься либо отвергаться:
$\xi^{21} < \xi^2 < \xi^{22}$— гипотеза H0 выполняется.
$x^2 \leq x^{21}$ — попадает в левый «хвост» распределения. Следовательно, теоретические и практические значения очень близки. Если, к примеру, происходит проверка генератора случайных чисел, который сгенерировал n чисел из отрезка [0,1] и выборка Xn распределена равномерно на [0,1], то генератор нельзя называть случайным (гипотеза случайности не выполняется), т.к. выборка распределена слишком равномерно, но гипотеза H0 выполняется.
$\xi^2 \geq \xi^2 2$ — попадает в правый «хвост» распределения, гипотеза H0 от- вергается.

Сама идея критерия основана на том, что мы измеряем разницу между наблюдаемой частотой попадания в какой-либо интервал и теоретической вероятностью попадания в этот же интервал.

2. Практика

Стоит сразу уточнить, что я по незнаю сделал работу так же и для дскретного случая, хоть оно там и не работает, но теперь жалко удалять, так что давайте просто не обращать внимания)
Если $\sqrt{n}D_n \geq \lambda_\alpha$, то отвергаем H_0, иначе – принимаем. Значение α возьмем равным 0.05, тогда $\lambda = 1.36$.

n	d_n	$\sqrt{n}D_n$	
5	0.4246575342465753	0.9495631137327873	принимаем
10	0.5986672104143272	1.8931519453685497	отвергаем
100	0.25573449266437115	2.5573449266437116	отвергаем
200	0.06964043964112798	0.9848645423010811	принимаем
400	0.04016891543928314	0.8033783087856627	принимаем
600	0.032131177533331834	0.7870498979144163	принимаем
800	0.023333961973014183	0.6599841097226945	принимаем
1000	0.03293704663567676	1.041560867679247	принимаем

код, который это считает:

```
def D_n(emp, raspr):
    res = []
    for i in range(0, len(emp), 5):
        tmp = -1
        for j in range(len(emp[i])):
            tmp = max(tmp, abs(emp[i][j] - raspr[j]))
        res.append(tmp)
    for i in range(1,3,1):
        res[i] = (6*5*i*res[i] + 1)/ (6*((5*i)**0.5))
    return res

def main():
    # Diskretoe

    n_s = [5, 10, 100, 200, 400, 600, 800, 1000]
    diskr, nepr = generate_distribs(n_s)
    emp_diskr_y, emp_diskr_x, emp_diskr_n, rise_points_diskr = emp_func_di
    diskr_raspr_func = Diskretnoe.gen_raspr_func()
    # print(emp_diskr_y)
    diskr_D_n = D_n(emp_diskr_y, diskr_raspr_func)
    diskr_nDn = []
    for i in range(len(diskr_D_n)):
        diskr_nDn.append(diskr_D_n[i] * (n_s[i]**0.5))
    diskr_condition = []
    for i in diskr_nDn:
        if i < 1.36:
            diskr_condition.append("принимаем")
        else:
            diskr_condition.append("отвергаем")
    # print(len(diskr_D_n))
    for i in range(len(diskr_D_n)):
        print(f'{n_s[i]} & {diskr_D_n[i]} & {diskr_nDn[i]} &
        {diskr_condition[i]} \\\\\\hline')
```

теперь для второго распределения:

n	d_n	$\sqrt{n}D_n$	
5	0.49470085470079106	1.1061847396382152	принимаем
10	0.5415680194737343	1.712588449443424	отвергаем
100	0.2690909092707572	2.690909092707572	отвергаем
200	0.10176068376042169	1.4391133909034792	отвергаем
400	0.06009829059862126	1.2019658119724252	принимаем
600	0.039213675213771115	0.9605349521296329	принимаем
800	0.059044871794686316	1.6700411696125204	отвергаем
1000	0.019034188033774546	0.6019138759864954	принимаем

код тот же самый, только для непрерывного

Теперь перейдем к сложной функции:

n	θ	d_n	$\sqrt{n}D_n$	
100	243.60000000000002	0.09849967384213953	0.9849967384213953	принимаем
200	200.27999999999997	0.23511665367401272	3.325051603655668	отвергаем
400	238.24	0.3067404251098597	6.1348085021971945	отвергаем
600	255.15999999999997	0.049693804309647094	1.2172446393635505	принимаем
800	186.64	0.01720053558553919	0.4865046141030114	принимаем
1000	229.62	0.05150941151709815	1.6288706132893929	отвергаем

код, который это считает:

```
def diskr_sloznaya_D_n(raspr):
n_s = [100, 200, 400, 600, 800, 1000]

data_1 = []
data_2 = []
for i in raspr:
    data_1.append(i[:len(i)//2])
    data_2.append(i[len(i)//2:])
otsenki = moment_method_diskr(data_1)
# print(otsenki)

emp_diskr_y, emp_diskr_x, emp_diskr_n, rise_points_diskr =
emp_func_diskr(data_2)
diskr_raspr_func = Diskretnoe.gen_raspr_func()

diskr_D_n = D_n(emp_diskr_y, diskr_raspr_func)
diskr_nDn = []

for i in range(len(diskr_D_n)):
```

```
        diskr_nDn.append(diskr_D_n[i] * (n_s[i]**0.5))

    diskr_condition = []
    for i in diskr_nDn:
        if i < 1.36:
            diskr_condition.append("принимаем")
        else:
            diskr_condition.append("отвергаем")

    for i in range(len(diskr_D_n)):
        print(f'{n_s[i]} & {otsenki[i]} & {diskr_D_n[i]} & {diskr_nDn[i]}
        {diskr_condition[i]} \\\\\\hline')
```

теперь непрерывнй случай

n	θ	d_n	$\sqrt{n}D_n$	
100	109.57687679275749	0.08619658119657092	0.8619658119657092	принимаем
200	116.26272996803887	0.29109018260929415	4.116636841197247	отвергаем
400	114.25854491391628	0.23933306565091717	4.786661313018343	отвергаем
600	116.16899267150573	0.042307692308029554	1.0363225834934515	принимаем
800	114.3520237447326	0.04314102564115843	1.220212471128234	принимаем
1000	115.52860208412338	0.029452991453373349	0.931385368981332	принимаем

код, который это считает:

```
def nepr_sloznaya_D_n(raspr):
    n_s = [100, 200, 400, 600, 800, 1000]

    data_1 = []
    data_2 = []
    for i in raspr:
        data_1.append(i[:len(i)//2])
        data_2.append(i[len(i)//2:])
    otsenki = max_pod_nepr(data_1)
    # print(otsenki)

    emp_nepr_y, emp_nepr_x, emp_nepr_n, rise_points_nepr = emp_func_nepr(d
    nepr_raspr_func = Neprerivnoe.gen_raspr_func()
    nepr_D_n = D_n(emp_nepr_y, nepr_raspr_func)
    nepr_nDn = []
    for i in range(len(nepr_D_n)):
        nepr_nDn.append(nepr_D_n[i] * (n_s[i]**0.5))
    nepr_condition = []
```

```
    for i in nepr_nDn:
        if i < 1.36:
            nepr_condition.append("принимаем")
        else:
            nepr_condition.append("отвергаем")
    for i in range(len(nepr_D_n)):
        print(f'{n_s[i]} & {otsenki[i]} & {nepr_D_n[i]} & {nepr_nDn[i]} &
```

Теперь перейдем к критерию Хи-квадрат. Как уже было указано в теории, будем пользоваться следующими функцими для подсчета этого значения:

$$X_N^2 = \sum_{i=1}^{N} \frac{(\eta_i^{(n)} - np_i)^2}{np_i} = \sum_{i=1}^{N} \frac{(\eta_i^{(n)})^2}{np_i} - n$$

где

$$\eta_k^{(n)} = \sum_{i=1}^{n} I(\xi_i = k)$$

Вормула из лекций в печатном виде, она немного проще и понятнее для перегона программу:

$$\xi_n^2 = n \sum_{i=1}^{k} \frac{\left(\frac{n_i}{n} - P_i(\theta)\right)^2}{P_i(\theta)}$$

В общем, грубо говоря, что тут происходит: мы бьем наш отрезок, в котором все происходит на меньшие отрезки каким-то образом, считаем вероятность попадания в них в теории и потом, сколько всего в них попало элементов выборки и смотрим сумму квадратов разностей этих величин.
В лекции было указано, что работает корректно (а потому и будет рассмотрено) для выборок длины не меньше чем 50 и при разбиении минимум на 3 отрезка с вхождением минимум 5 элементов в каждый отрезок

Дискретный случай:
Для начала, важно сказать, что $P_i(\theta) = F(x_i, \theta) - F(x_{i-1}, \theta)$
Для начала, будем бить на равные отрезки: на 3 и 5 штук. Тогда $P_i = \frac{1}{3}$ и $P_i = \frac{1}{5}$ соответственно для любого отрезка.

```
# кси-квадрат
def diskr_ksi(raspr, k): # к - количество отрезков. Сначала поделим на рав
    delta = Diskretnoe.t//k
    res = []
    n_s = [100, 200, 400, 600, 800, 1000]
    for i in range(0, len(raspr), 5):
        r = sorted(raspr[i])
```

```
        tmp = Diskretnoe.a + delta
        res_i = 0
        count = 0
        for j in r:
            if j < tmp:
                count += 1
            else:
                print(count/n_s[i//5])
                res_i += (((count/n_s[i//5]) - (1/k))**2)/(1/k)
                count = 1
                tmp += delta
        res_i *= n_s[i//5]
        res.append(res_i)
    return res
```

квантили для сравнения я брал отсюда:мега ссылка, альфа берем за 0.05.
Вывод для N = 3:

n	X_3^2	$X^2_{1-\alpha,N-1}$	решение
100	1.0666666666666653	5.9915	Принимаем
200	0.4499999999999992	5.9915	Принимаем
400	1.9824999999999995	5.9915	Принимаем
600	0.885	5.9915	Принимаем
800	0.4474999999999987	5.9915	Принимаем
1000	0.6619999999999985	5.9915	Принимаем

Вывод для N = 5:

n	X_5^2	$X^2_{1-\alpha,N-1}$	решение
100	8.649999999999999	9.4877	Принимаем
200	4.725000000000001	9.4877	Принимаем
400	5.549999999999999	9.4877	Принимаем
600	2.4583333333333353	9.4877	Принимаем
800	4.9375000000000036	9.4877	Принимаем
1000	1.730000000000003	9.4877	Принимаем

Вывод для N = 20:

n	X^2_{20}	$X^2_{1-\alpha,N-1}$	решение
100	21.800000000000004	30.1435	Принимаем
200	19.1	30.1435	Принимаем
400	16.750000000000004	30.1435	Принимаем
600	25.200000000000003	30.1435	Принимаем
800	28.35000000000001	30.1435	Принимаем
1000	33.36000000000001	30.1435	Отвергаем

Текперь сложная гипотеза:
Для нее надо сначала посчитать новое $\overline{\theta}$ методом максимального правдоподобия. дальше уже работаем с новым параметром, а алгоритмы все те же самые:

$$\hat{X}^2_n = X^2_n(\hat{\theta}) = n\sum_{i=1}^{N}\frac{\frac{n_i}{n} - P_i(\hat{\theta})^2}{P_i(\hat{\theta})}$$

так что тепреь просто сначала методом максимального правдоподбия в коде посчитаем параметр и сделаем то же самое). Также, сравнивать будем с квантилем уровня $N-1-r$, где r - это 2), то есть уровень $N-2$

```
# кси-квадрат сложная
def diskr_ksi_sl(raspr, k): # к - количество отрезков. Сначала поделим на
    res = []
    n_s = [100, 200, 400, 600, 800, 1000]
    for i in range(0, len(raspr), 5):
        r = sorted(raspr[i])
        delta = (r[-1] - Diskretnoe.a + 1)//k
        tmp = Diskretnoe.a + delta
        res_i = 0
        count = 0
        for j in r:
            if j < tmp:
                count += 1
            else:
                # print(count/n_s[i//5])
                res_i += (((count/n_s[i//5]) - (1/k))**2)/(1/k)
                count = 1
                tmp += delta
        res_i *= n_s[i//5]
        res.append(res_i)
    return res
```

Вывод для N = 4:

n	$\hat{X}_4^2$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	1.72	5.9915	Принимаем
200	1.1200000000000008	5.9915	Принимаем
400	1.8100000000000012	5.9915	Принимаем
600	4.6866666666666665	5.9915	Принимаем
800	0.8200000000000015	5.9915	Принимаем
1000	1.9919999999999993	5.9915	Принимаем

Вывод для N = 10:

n	$\hat{X}_{10}^2$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	11.2	15.5073	Принимаем
200	6.7	15.5073	Принимаем
400	7.775	15.5073	Принимаем
600	4.216666666666669	15.5073	Принимаем
800	6.400000000000003	15.5073	Принимаем
1000	13.190000000000003	15.5073	Принимаем

Вывод для N = 20:

n	$\hat{X}_{20}^2$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	13.800000000000004	28.8693	Принимаем
200	16.300000000000004	28.8693	Принимаем
400	34.699999999999996	28.8693	Отвергаем
600	32.30000000000001	28.8693	Отвергаем
800	11.125000000000004	28.8693	Принимаем
1000	34.68	28.8693	Отвергаем

непрерывный случай:
важно сказать, что в данной ситуации это будет длина отрезка/общую длину. $P_i(\theta) = \frac{l_i}{\theta}$. При сложной гипотезе это всегда будет $\frac{1}{k}$, где k = колиество отрезков.

```
# кси-квадрат
def diskr_nepr(raspr, k): # к - количество отрезков. Сначала поделим на ра
    delta = Neprerivnoe.t//k
    res = []
    n_s = [100, 200, 400, 600, 800, 1000]
    for i in range(0, len(raspr), 5):
        r = sorted(raspr[i])
        print(r)
        tmp = delta
```

```
        res_i = 0
        count = 0
        for j in r:
            if j < tmp:
                count += 1
            else:
                # print(count/n_s[i//5])
                res_i += (((count/n_s[i//5]) - (1/k))**2)/(1/k)
                count = 1
                tmp += delta
        res_i *= n_s[i//5]
        res.append(res_i)
    return res
```

квантили для сравнения я брал отсюда:мега ссылка, альфа берем за 0.05.
Вывод для N = 3:

n	X_3^2	$X^2_{1-\alpha,N-1}$	решение
100	0.06666666666666664	5.9915	Принимаем
200	0.8833333333333334	5.9915	Принимаем
400	0.7541666666666668	5.9915	Принимаем
600	0.33999999999999847	5.9915	Принимаем
800	0.020833333333333319	5.9915	Принимаем
1000	0.2816666666666695	5.9915	Принимаем

Вывод для N = 5:

n	X_5^2	$X^2_{1-\alpha,N-1}$	решение
100	1.3000000000000005	9.4877	Принимаем
200	2.8500000000000005	9.4877	Принимаем
400	5.050000000000001	9.4877	Принимаем
600	4.225000000000002	9.4877	Принимаем
800	11.418750000000003	9.4877	Отвергаем
1000	3.0300000000000025	9.4877	Принимаем

Вывод для N = 20:

n	X^2_{20}	$X^2_{1-\alpha,N-1}$	решение
100	11.200000000000005	30.1435	Принимаем
200	15.500000000000005	30.1435	Принимаем
400	35.25000000000001	30.1435	Отвергаем
600	34.733333333333335	30.1435	Отвергаем
800	37.45000000000001	30.1435	Отвергаем
1000	38.28000000000002	30.1435	Отвергаем

Теперь перейдем к сложной гипотезе. делаем все то же самое, только для непрерывного распределения:

```
# кси-квадрат сложная
def nepr_ksi_sl(raspr, k): # к - количество отрезков. Сначала поделим на р
    res = []
    n_s = [100, 200, 400, 600, 800, 1000]
    for i in range(0, len(raspr), 5):
        r = sorted(raspr[i])
        delta = (r[-1])//k
        tmp = delta
        res_i = 0
        count = 0
        for j in r:
            if j < tmp:
                count += 1
            else:
                # print(count/n_s[i//5])
                res_i += (((count/n_s[i//5]) - (1/k))**2)/(1/k)
                count = 1
                tmp += delta
        res_i *= n_s[i//5]
        res.append(res_i)
    return res
```

Вывод для $N = 4$:

n	$\hat{X}^2_4$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	11.600000000000001	5.9915	Отвергаем
200	4.759999999999999	5.9915	Принимаем
400	1.6200000000000006	5.9915	Принимаем
600	2.379999999999998	5.9915	Принимаем
800	3.984999999999999	5.9915	Принимаем
1000	1.1480000000000021	5.9915	Принимаем

Вывод для N = 10:

n	$\hat{X}^2_{10}$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	19.000000000000004	15.5073	Отвергаем
200	10.550000000000002	15.5073	Принимаем
400	7.400000000000001	15.5073	Принимаем
600	13.566666666666668	15.5073	Принимаем
800	6.987500000000009	15.5073	Принимаем
1000	10.040000000000012	15.5073	Принимаем

Вывод для N = 20:

n	$\hat{X}^2_{20}$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
100	32.60000000000001	28.8693	Отвергаем
200	18.70000000000001	28.8693	Принимаем
400	27.150000000000006	28.8693	Принимаем
600	25.833333333333346	28.8693	Принимаем
800	38.750000000000014	28.8693	Отвергаем
1000	39.460000000000015	28.8693	Отвергаем

Работа с данными: критерий согласия Колмагорова для сложной гипотезы:
Формулы все те же)

n	θ	d_n	$\sqrt{n}D_n$	
1000	99.96	0.040196261682490575	1.2711174034082218	принимаем

код, который это считает:

```
def nepr_sloznaya_D_n(raspr):
    n_s = 1000

    data_1 = raspr[:len(raspr)//2]
    data_2 = raspr[len(raspr)//2:]
    # методом моментов была посчитана такая оценка, так что сразу просто п
    otsenki = [max(data_2)]
    n_s = [len(raspr)]

    emp_nepr_y, x = emp_func_nepr([data_1])
    nepr_raspr_func = gen_raspr_func()
    nepr_D_n = D_n(emp_nepr_y, nepr_raspr_func)
    nepr_nDn = []
```

```
    for i in range(len(nepr_D_n)):
        nepr_nDn.append(nepr_D_n[i] * (n_s[i]**0.5))
    nepr_condition = []
    for i in nepr_nDn:
        if i < 1.36:
            nepr_condition.append("принимаем")
        else:
            nepr_condition.append("отвергаем")
    for i in range(len(nepr_D_n)):
        print(f'{n_s[i]} & {otsenki[i]} & {nepr_D_n[i]} & {nepr_nDn[i]} &
```

теперь оценка хи-квадрат для сложной гипотезы:

```
# кси-квадрат сложная
def nepr_ksi_sl(raspr, k): # к - количество отрезков. Сначала поделим на р
    res = []
    n_s = [100, 200, 400, 600, 800, 1000]
    for i in range(0, len(raspr), 5):
        r = sorted(raspr[i])
        delta = (r[-1])//k
        tmp = delta
        res_i = 0
        count = 0
        for j in r:
            if j < tmp:
                count += 1
            else:
                # print(count/n_s[i//5])
                res_i += (((count/n_s[i//5]) - (1/k))**2)/(1/k)
                count = 1
                tmp += delta
        res_i *= n_s[i//5]
        res.append(res_i)
    return res
```

Вывод для N = 4:

n	$\hat{X}_4^2$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
1000	5.06400000000001	5.9915	Принимаем

Вывод для N = 10:

n	$\hat{X}^2_{10}$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
1000	7.85000000000002	15.5073	Принимаем

Вывод для N = 20:

n	$\hat{X}^2_{20}$	$X^2_{1-\alpha,N-1}(\hat{\theta})$	решение
1000	54.00000000000009	28.8693	Отвергаем

Домашнее задание 5.

Различение статистических гипотез

1. Описание критерия отношения правдоподобия
2. Вычисление функции отношения правдоподобия
3. Вычисление критической области.
4. Вычисление минимального необходимого количества материала при фиксации минимального возможного значения ошибок первого и второго рода.
5. Необходимо ответить на вопросы:
   (a) Что является гипотезой H0, что H1?
   (b) Что такое ошибка первого и второго рода, функция мощности?

1. Теория:

5.1.1. Описание критерия отношения правдоподобия

Пусть имеется выборка $X = (X_1, ...X_n)$ из распределения. Рассмотрим две гипотезы:

$$H_0 : \theta = \theta_0$$
$$H_1 : \theta = \theta_1$$

Выбор из двух простых гипотез представляется в виде параметрической функции. Пусть $\theta = 0, 1$ и $F_\theta(x) = (1 - \theta)F_0(x) + \theta F_1(x)$
В случае параметрических гипотез функцию мощности критерия можно переписать в виде

5.1.2. функция мощности

$$W(\theta) = W(\theta; \mathfrak{X}_{1,\alpha}) = P_\theta(X \in \mathfrak{X}_{1,\alpha})$$

5.1.3. ошибка первого и второго рода

Ошибкой первого рода является:

$$P(X \in \mathfrak{X}_1 | H_0) = \alpha$$

Ошибкой второго рода является:

$$P(X \in \mathfrak{X}_1 | H_1) = \beta$$

Вероятность ошибок первого и второго рода через функцию мощности:

$$\alpha = W(\theta_0, \mathfrak{X}_{1,\alpha}))$$

$$\beta = 1 - W(\mathfrak{X}_{1,\alpha}))$$

Параметрический критерий, минимизирующий ошибку второго рода при заданной ошибке первого рода, называется наиболее мощным критерием уровня значимости α

5.1.4. критическая область

Критическую область можно построить следующим образом: множество $\mathfrak{X}_{1,\alpha}$ сострoит из таких $\overline{x}$ для которых правдоподобие $L(\overline{x}, \theta_1)$,будет больше правдоподобия $L(\overline{x}, \theta_0)$. Функция, имеющая вид

$$l(\overline{x}) = \frac{L(\overline{x}, \theta_1)}{L(\overline{x}, \theta_0)} = \frac{\prod\limits_{i=1}^{n} f_1(x_i)}{\prod\limits_{i=1}^{n} f_0(x_i)}$$

называется функцией отношения правдоподобия. Выберем некоторую границу c. Если $l(\overline{x}) > c$, принимаем H_1, иначе принимаем H_0. Критическим множеством Неймана-Пирсона называется множество $\mathfrak{X}^*_{1,\alpha}$, имеющее вид

$$\mathfrak{X}^*_{1,\alpha} = \overline{x} \in \mathfrak{X} : l(\overline{x}) > c_\alpha$$

где c_α такое, что ошибка первого рода равна α. Для данного множества верно

$$W(\theta_0; \mathfrak{X}^*_{1,\alpha}) = P_0(\overline{x} \in \mathfrak{X}^*_{1,\alpha}) = \alpha$$

Лемма Неймана-Пирсона гласит, что критическая область $\mathfrak{X}^*_{1,\alpha}$ задает наиболее мощный критерий для гипотезы H_0 относительно H_1 среди всех критериев с уровнем значимости α. Также данный критерий является несмещенным.

5.1.5. гипотеза H_0 и H_1

- ▷ гипотеза H_0 — основная гипотеза (нулевая гипотеза), заключается в том, что $F_X \in F_0$
- ▷ гипотеза H_1— альтернативная гипотеза, заключается в том, что $F_X \in F_1$

Определение 8.1. Если множество F_0 состоит из одного распределения, то говорят, что H_0 — простая гипотеза, иначе H_0 - сложная гипотеза.
Определение 8.2. Если множество F_1 состоит из одного распределения, то говорят, что H_1 - простая гипотеза, иначе H_1 - сложная гипотеза.
Определение 8.5. Статистический критерий - это правило, по которому каждой реализации выборки ставится в соответствие решение: принимаем гипотезу H_0 или отвергаем ее (то есть принимаем гипотезу H_1)

2. Дискретное

5.2.1. Вычисление функции отношения правдоподобия

Запишем функцию правдоподобия:

$$L(\theta,\overline{x}) = \prod \theta^{-1} I(x \in [\alpha, \alpha+\theta-1]) = \theta^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha+\theta-1)$$

Пусть есть выборка независимых случайных величин $X = (X_1, ..., X_n)$ из равномерного распределения.
Выдвинем две гипотезы:

$$H_0 : \theta = \theta_0$$

$$H_1 : \theta = \theta_1$$

Уровень значимости возьмём равным α
Предполагая, что верна нулевая гипотеза, получаем:

$$L_n(\theta_0|X) = \theta_0^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha+\theta_0-1)$$

Аналогично, при верности альтернативной гипотезы:

$$L_n(\theta_1|X) = \theta_1^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha+\theta_1-1)$$

Тогда функция отношения правдоподобия примет вид:

$$l(X) = \frac{L(\theta_1|X)}{L(\theta_0|X)} = \frac{\theta_1^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha+\theta_1-1)}{\theta_0^{-n} \cdot I(x_{(1)} \geq \alpha) \cdot I(x_{(n)} \leq \alpha+\theta_0-1)} =$$
$$= \left(\frac{\theta_1}{\theta_0}\right)^n \cdot \frac{I(x_{(n)} \leq \alpha+\theta_1-1)}{I(x_{(n)} \leq \alpha+\theta_0-1)}$$

А логарифмическая функция отношения правдоподобия будет равна:

$$ln(l(X)) = ln\left(\left(\frac{\theta_1}{\theta_0}\right)^n \cdot \frac{I(x_{(n)} \leq \alpha+\theta_1-1)}{I(x_{(n)} \leq \alpha+\theta_0-1)}\right) = n{\cdot}ln\left(\frac{\theta_1}{\theta_0}\right) + ln\left(\frac{I(x_{(n)} \leq \alpha+\theta_1-1)}{I(x_{(n)} \leq \alpha+\theta_0-1)}\right)$$

5.2.2. Вычисление критической области

Важно заметить, что выполняется следующее свойство:
Допустим, что БОО $\theta_1 > \theta_0$ => если $x_n \in [\theta_0; \theta_1]$ => нулевая гипотеза отвергается и принимается первая гипотеза. Если же $x \in [0; \theta_0]$ => ошибка первого рода отсутствует, а ошибка второго рода будет иметь вид: $\left(\frac{\theta_0}{\theta_1}\right)^n$

$$P(l(X) \leq k|H_0) = \alpha$$

$$\alpha = P(l(X) \leq k|H_0) = P(\ln(l(X)) \leq \ln(k)|H_0) =$$

$$P\left(n \cdot ln\left(\frac{\theta_1}{\theta_0}\right) + ln\left(\frac{I(x_{(n)} \leq \alpha + \theta_1 - 1)}{I(x_{(n)} \leq \alpha + \theta_0 - 1)}\right) \leq ln(k)\right) =$$

$$P\left(lnI(x_{(n)} \leq \alpha + \theta_1 - 1) - ln(I(x_{(n)} \leq \alpha + \theta_0 - 1) \leq ln(k) - n \cdot ln\left(\frac{\theta_1}{\theta_0}\right)\right) =$$

5.2.3. Вычисление минимального необходимого количества материала при фиксации минимального возможного значения ошибок первого и второго рода

Теперь зафиксируем ошибку второго рода $= \beta$, вспомним, что это равно

$$=> \beta = \left(\frac{\theta_0}{\theta_1}\right)^n => n = ln(\beta) : ln(\frac{\theta_0}{\theta_1})$$