Домашние задания

Фомин Андрей, СКБ 172

Содержание

1 Домашнее задание 1. Геометрическое распределение 3
- 1.1 Мат. ожидание, дисперсия, х.ф и п.ф . . . 3
- 1.2 Типичная интерпретация: . . . 5
- 1.3 Нетипичная интерпретация: . . . 5
- 1.4 Известные соотношения между распределениями. . . . 5
- 1.5 Моделирование случайной величины. Описание. . . . 6

2 Домашнее задание 1. Нормальное распределение 11
- 2.1 Мат. ожидание, дисперсия и др. . . . 11
- 2.2 Типичная интерпретация: . . . 13
- 2.3 Нетипичная интерпретация: . . . 13
- 2.4 Известные соотношения между распределениями. . . . 14
- 2.5 Моделирование случайной величины . . . 14

3 Домашнее задание 2. Геометрическое распределение 19
- 3.1 Построение выборок . . . 19
- 3.2 Графики эмпирической функции распределения и функции распределения. . . . 19
- 3.3 Построение вариационного ряда выборки . . . 25
- 3.4 Построение гистограмм и полигонов частот . . . 27

4 Домашнее задание 2. Нормальное распределение 39
- 4.1 Построение выборок . . . 39
- 4.2 Графики эмпирической функции распределения и функции распределения . . . 41
- 4.3 Построение вариационного ряда выборки . . . 49
- 4.4 Построение гистограмм и полигонов частот . . . 51

5 Домашнее задание 3. Геометрическое распределение. 57
5.1 Нахождение выборочного среднего и выборочной дисперсии 57
5.2 Нахождение параметров распределений событий. 62
5.3 Работа с данными 66

6 Домашнее задание 3. Нормальное распределение 73
6.1 Нахождение выборочного среднего и выборочной дисперсии. 73
6.2 Нахождение параметров распределений событий. 77
6.2.1 Оценим параметр θ_1 в распределении $N(\theta_1, \sigma^2)$ . . . 77
6.2.2 Оценим параметр θ_2 в распределении $N(\mu, \theta^2)$ . . . 81
6.3 Работа с данными 85

7 Домашнее задание 4. Геометрическое распределение. 89
7.1 Критерий согласия хи-квадрат для простой гипотезы. . . . 89
7.1.1 Описание применяего критерия. 89
7.1.2 Описание свойств. 93
7.1.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$ 93
7.2 Критерий согласия хи-квадрат для сложной гипотезы. . . . 97
7.2.1 Описание применяемого критерия. 97
7.2.2 Описание свойств. 99
7.2.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$ 99

8 Домашнее задание 4. Нормальное распределение 102
8.1 Критерий Колмогорова для простой гипотезы 102
8.1.1 Описание критерия 102
8.1.2 Описание свойств. 104
8.1.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$. 104
8.2 Критерий Колмогорова для сложной гипотезы. 105
8.2.1 Описание критерия. 105
8.2.2 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$ 106
8.3 Критерий согласия хи-квадрат для простой гипотезы . . . 108
8.4 Критерий хи-квадрат для сложной гипотезы 111

9 Домашнее задание 4.Часть 2. 113

10 Домашнее задание 5. 116

Замечания:
представленный ниже код находится по адресу /код/Python/МатСтат.ipynb;
все картинки находятся в папке images;

1 Домашнее задание 1. Геометрическое распределение

1.1 Мат. ожидание, дисперсия, х.ф и п.ф

ВАЖНО. В 1-ом и 2-ом домашних заданиях вероятность успеха в геометрическом распределении принимаем за $p = 0.3$

$$p(\eta = x) = pq^x,\ x \in N \cup \{0\}, 0 < p < 1, q = 1 - p$$

Производящая функция:

$$\phi_\eta(z) = \sum_{k=0}^{\infty} z^k p q^k = p \sum_{k=0}^{\infty} z^k q^k = p \sum_{k=0}^{\infty} (zq)^k = \frac{p}{1 - qz}$$

Характеристическая функция:

$$g(t) = \phi(e^{it}) = \frac{p}{1 - qe^{it}}$$

Математическое ожидание:

$$\phi_\eta^{'}(z) = (-1)(-q)\frac{p}{(1 - qz)^2} = \frac{qp}{(1 - qz)^2}$$

$$E\eta = \phi_\eta^{(1)}(1) = \frac{qp}{(1 - q)^2} = \frac{qp}{p^2} = \frac{q}{p}$$

Дисперсия:

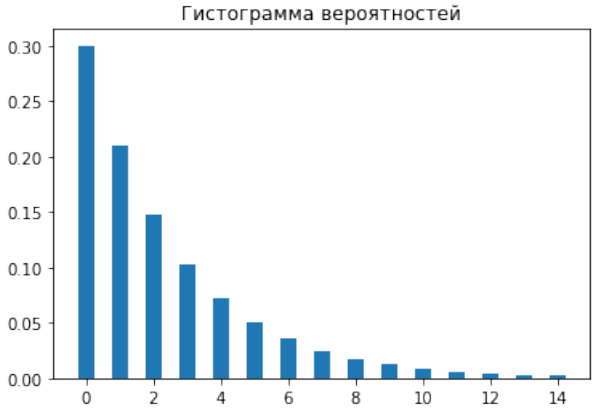
$$D\eta = \phi_\eta^{(2)}(1) + \phi_\eta^{(1)}(1) - (\phi_\eta^{(1)}(1))^2$$

$\phi_\eta^{(1)}(1)$ нам уже известно, посчитаем $\phi_\eta^{(2)}(1)$:

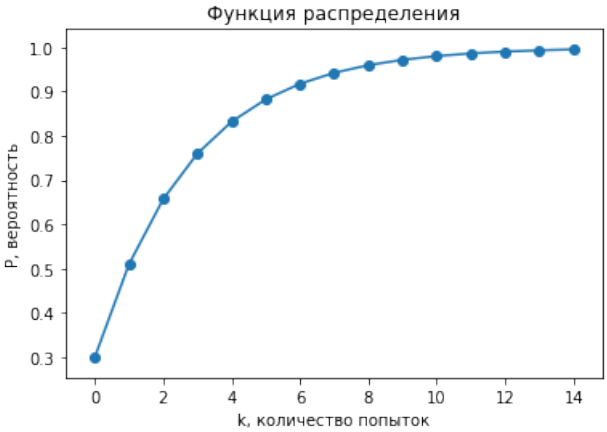
$$\phi_\eta^{(2)}(z) = (\phi_\eta^{(1)}(z))^{'} = \left(\frac{qp}{(1 - qz)^2}\right)^{'} = (-2)(-q)\frac{qp}{(1 - qz)^3} = \frac{2q^2p}{(1 - qz)^3}$$

$$\phi_\eta^{(2)}(1) = \frac{2q^2p}{(1 - q)^3} = \frac{2q^2p}{p^3} = \frac{2q^2}{p^2}$$

$$D\eta = \frac{2q^2}{p^2} + \frac{q}{p} - \frac{q^2}{p^2} = \frac{q^2}{p^2} + \frac{q}{p} = \frac{q^2 + qp}{p^2} = \frac{q(q + p)}{p^2} = \frac{q}{p^2}$$




```python
import numpy as np
import matplotlib.pyplot as plt

def geomm(p, i):
    return (1 - p) ** i * p

s = 0
mas = []
for k in range(15):
    for i in range(k+1):
        s += geomm(0.3, i)
    mas.append(s)
    s = 0

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(list(range(15)), mas)
ax.scatter(list(range(15)), mas)
ax.set_ylabel('P, вероятность')
ax.set_xlabel('k, количество попыток')
ax.set_title('Функция распределения')
plt.show()
```



1.2 Типичная интерпретация:

X - число испытаний *Бернулли*, предшествовавших первому успеху, p - вероятность успеха, $q = 1 - p$ - вероятность неудачи в одиночном испытании

1.3 Нетипичная интерпретация:

В газете NZ Herald (это ежедневная газета, которая издается в Окленде, Новая Зеландия) приводится статься акушера доктора Фредди Грэхэма (Dr Freddie Graham), который утверждает, что шансы на успешную беременность в результате имплантации замороженного эмбриона составляют примерно 1 успешная беременность на 10 попыток. Предположим, пара, которая отчаянно хочет иметь детей, продолжит проводить эту "операцию"до тех пор, пока женщина не забеременеет. Тогда X - количество неуспешных попыток, будет иметь геометрическое распределение, с вероятностью успеха $p = 0.1$ и с вероятностью провала $q = 0.9$. (Значения вероятности успехи и неудачи даны доктором Фредди Грэхэмом). Следовательно, вероятность забеременеть при k-ой попытке равна $p(k) = (1 - p)^k \cdot p$. Тогда функция распределения имеет вид $F(k) = \sum_{i=0}^{k} (1 - p)^k \cdot p$

1.4 Известные соотношения между распределениями.

1. Геометрическое распределение является частным случаем отрицательного биномиального распределения $p(x = z) = C_z^{z+m-1} p^m q^z$ с параметром $m = 1$
2. *Распределение Эйлера* $p(x) = \frac{p(0)\alpha^x}{(1-q)(1-q^2)...(1-q^x)}$, $x = 1,2,3\ldots$ стремится к геометрическому при $q \to 0$
3. Связь с *распределением Юла.*
Распределение Юла было введено для описания числа биологических выводов в семействе. Это распределение является смесью смещенных геометрических распределений, определенных вероятностями:

$$e^{-\beta u}(1 - e^{-\beta u})^{y-1},\ y = 1,2,3,\ldots$$

где u имеет эксопненциальное распределение с плотностью:

$$f(u) = \theta^{-1} exp(\frac{-u}{\theta}), \theta > 0, u \geq 0$$

Функция вероятности распределения Юла имеет вид:

$$p(y) = \int_0^\infty e^{-\beta u}(1 - e^{-\beta u})^{y-1}\theta^{-1}exp(\frac{-u}{\theta})$$

4. Дискретное *распределение Вейбулла* $p(x) = q^{x^\beta} - q^{(x+1)^\beta}$, определяющееся через частичные суммы, при $\beta = 1$ является геометрическим распределением:

$$p(x) = q^x - q^(x + 1) = q^x(1 - q) = pq^x$$

1.5 Моделирование случайной величины. Описание.

Алгоритм GeomB (Geometric Bernoully)
Моделирование геометрического распределения Geom(p) через испытания Бернулли

Входные данные: p.
Результат: ξ.

1. (*Инициализация*) $i \leftarrow k \leftarrow 0$;

2. (*Моделирование числа неудач*) While $i = 0$ do $\Big($Get(α); If $\alpha < p$ then $i \leftarrow 1$ else $k \leftarrow k + 1\Big)$;

3. (*Завершение*) $\xi \leftarrow k$; STOP.

На вход подается значение p, i - число успехов, то есть $i = 1$, k - число неуспехов, наша случайная величина. $Get(\alpha)$ - моделирование случайной величины,равномерно распределенной на отрезке (0,1]

Обоснование моделирования

В данном моделировании мы считаем, сколько попыток потребуется равномерно распределенной случайной величине α на отрезке (0,1), чтобы α была меньше p. Это и есть смысл геометрического распределения, где успехом считается событие: $\alpha < p$ и неудачей событие $\alpha >\geq$. В каждом испытании вероятность успеха и неудачи одна и та же, так как α - равномерно распределенная случайная величина. Следовательно, смоделированные случайные величины имеют геометрическое распределение.

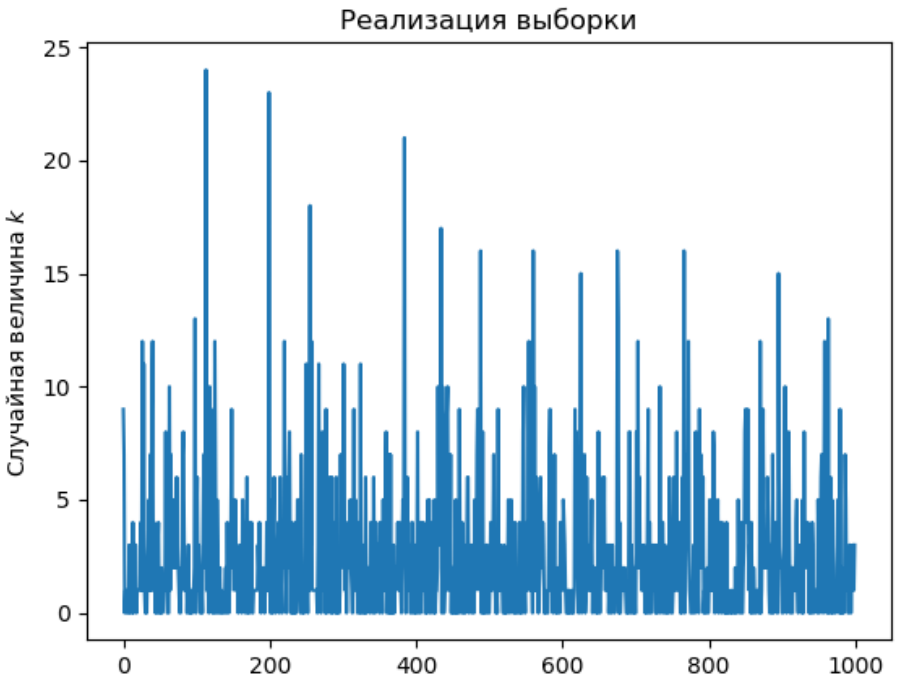
Моделирование на языке Python:

```
# моделирование случайной величины, имеющей геометрическое распределение
def my_geom_model(p):
    i = 0
    k = 0
    while i == 0:
        alpha = np.random.rand()
        if alpha < p:
            i = 1
        else:
            k = k + 1
    return k
for i in range(1000000):
    print(my_geom_model(0.3), end=' ')
```

функция np.random.rand() осуществляет моделирование равномерно распределенной случайной величины на отрезке $[0, 1)$

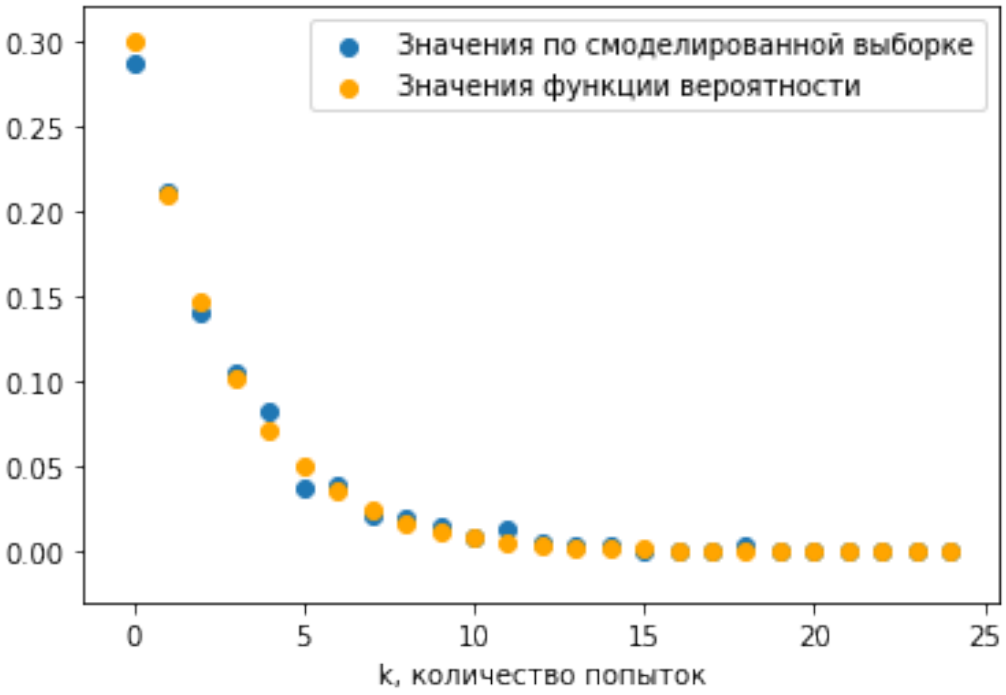
```
def rand(*dn): # known case of numpy.random.mtrand.rand
    """
    rand(d0, d1, ..., dn)

        Random values in a given shape.

        Create an array of the given shape and populate it with
        random samples from a uniform distribution
        over ``[0, 1)``.
```

График смоделированной выборки:


Реализация выборки
Случайная величина k
25
20
15
10
5
0
0
200
400
600
800
1000

Сравнение нашей выборки с функцией вероятностей геометрического распределения

```
def my_geom_model(p):
    i = 0
    k = 0
    while i == 0:
        alpha = np.random.rand()
        if alpha < p:
            i = 1
        else:
            k = k + 1
    return k


def geomm(p, i):
    return (1 - p) ** i * p


geom_arr = []
for i in range(1000):
    geom_arr.append(my_geom_model(0.3))
z = np.array(geom_arr)

mas_prob = []
for i in range(25):
    mas_prob.append((z == i).sum() / 1000)

mas = []
for i in range(25):
    mas.append(geomm(0.3, i))

fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(list(range(25)), mas_prob, label='Значения по смоделированной выборке')
ax.scatter(list(range(25)), mas, color='orange', label='Значения функции вероятности')
ax.set_xlabel('k, количество попыток')
ax.legend()
plt.show()
```


Значения по смоделированной выборке
Значения функции вероятности
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
k, количество попыток

2 Домашнее задание 1. Нормальное распределение

2.1 Мат. ожидание, дисперсия и др.

$f(x) = \frac{1}{\sigma\sqrt{2\Pi}} \cdot exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, где μ - математическое ожидание, а параметр σ - среднеквадратичное отклонение (σ^2 - дисперсия) распределения.
Найдем **характеристическую функцию** нормального распределения:

$$g(t) = \frac{1}{\sqrt{2\Pi}\sigma} \int\limits_{-\infty}^{\infty} e^{itx} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \frac{1}{\sigma\sqrt{2\Pi}} \int\limits_{-\infty}^{\infty} e^{-\frac{(x-(\mu+it\sigma^2))^2}{2\sigma^2}} \cdot e^{\frac{2\mu it\sigma^2 + (it\sigma^2)^2}{2\sigma^2}} dx = \frac{1}{\sqrt{2\Pi}\sigma} e^{\mu it - \frac{t^2\sigma^2}{2}} \cdot$$

$$\cdot \int\limits_{-\infty}^{\infty} e^{-\frac{(x-(\mu+it\sigma^2))^2}{2\sigma^2}}$$

Сделаем замену: $u = \frac{(x-(\mu+it\sigma^2))^2}{2\sigma^2}$. Тогда:

$$du = \frac{1}{\sqrt{2}\sigma} dx$$

$$dx = \sqrt{2}\sigma du$$

При подстановке получается:

$$\frac{1}{\sqrt{2}} e^{\mu it - \frac{t^2\sigma^2}{2}} \int\limits_{-\infty}^{\infty} e^{-u^2} du$$

Таким образом, мы свели наш интеграл к интегралу Пуассона: $\int\limits_{-\infty}^{\infty} e^{-u^2} du = \sqrt{\Pi}$
Следовательно $g(t) = e^{\mu it - \frac{t^2\sigma^2}{2}}$
Найдем **мат. ожидание**: Так как $g^{(r)}(t)\big|_{t=0} = i^r E\xi^r$ Тогда $g^{(1)}(t) = (\mu i - \frac{2t\sigma^2}{2}) \cdot e^{\mu it - \frac{t^2\sigma^2}{2}}$
$g^{(1)}(0) = \mu i = iE\xi$
Тогда $E\xi = \mu$
Найдем дисперсию:

$$g^{(2)}(t) = \left(g^{(1)}(t)\right)' = -\sigma^2 \cdot e^{\mu it - \frac{t^2\sigma^2}{2}} + (\mu i - \frac{2t\sigma^2}{2}) \cdot e^{\mu it - \frac{t^2\sigma^2}{2}} \cdot (\mu i - \frac{2t\sigma^2}{2})$$

$$g^{(2)}(0) = -\sigma^2 + \mu^2 i^2 = -\mu^2 - \sigma^2 = i^2 E\xi^2$$

$$E\xi^2 = \mu^2 + \sigma^2$$

Тогда $D\xi = E\xi^2 - (E\xi)^2 = \mu^2 + \sigma^2 - \mu^2 = \sigma^2$

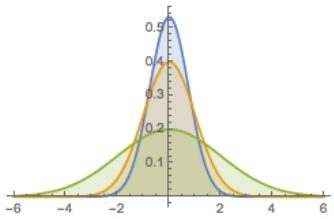
Функция распределения $F(x) = \frac{1}{\sigma\sqrt{2\Pi}} \int\limits_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$

График **плотности вероятности**:

1) при $\mu = 0$ и $\sigma = 0.75, 1, 2$

```
In[1]:= Plot[Table[PDF[NormalDistribution[0, σ], x], {σ, {.75, 1, 2}}] // Evaluate, {x, -6, 6}, Filling → Axis]
```

Out[1]=



2) при $\mu = -1, 1, 2$ и $\sigma = 1$

```
In[1]:= Plot[Table[PDF[NormalDistribution[μ, 1], x], {μ, {-1, 1, 2}}] // Evaluate, {x, -6, 6}, Filling → Axis]
```

Out[1]=

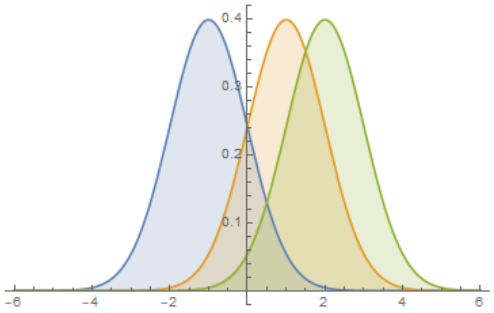
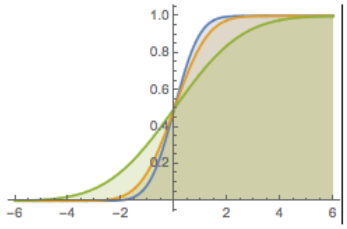


График **функции распределения**:

1) при $\mu = 0$ и $\sigma = 0.75, 1, 2$

```
In[1]:= Plot[Table[CDF[NormalDistribution[0, σ], x], {σ, {.75, 1, 2}}] // Evaluate, {x, -6, 6}, Filling → Axis]
```

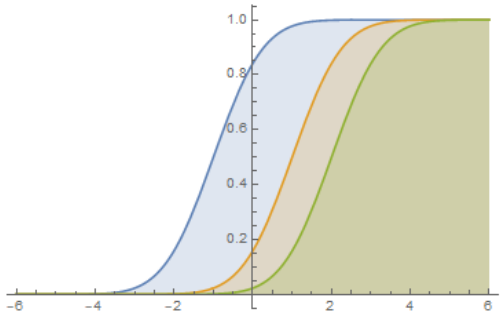
Out[1]=



2) при $\sigma = 1$ и $\mu = -1, 1, 2$

```
In[2]:= Plot[Table[CDF[NormalDistribution[μ, 1], x], {μ, {-1, 1, 2}}] // Evaluate, {x, -6, 6}, Filling → Axis]
```

Out[2]=



2.2 Типичная интерпретация:

Нормальное распределение заключается в том, что чаще всего встречаются средние значения соответствующих показателей, и отклонения в большую и меньшие стороны встречаются одинаково редко по сравнению со средними значениями.

2.3 Нетипичная интерпретация:

Рассмотрим рост игроков чемпионата мира 2018. (Данные находятся на сайте www.championat.com). Большинство игроков будут иметь рост μ. Очень высоких или низких игрков будет очень мало - $\approx 1 - 2\%$. Также рост 68 - 69 % игроков будет находиться между $\mu - \sigma$ и $\mu + \sigma$. Рост 95 - 96 % игроков будет находиться между $\mu - 2 \cdot \sigma$ и $\mu + 2 \cdot \sigma$. И рост практически всех игроков будет находиться между $\mu - 3 \cdot \sigma$ и $\mu + 3\sigma$. Это нормальное распределения с плотностью: $f(x) = \frac{1}{\sigma\sqrt{2\Pi}} \cdot exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

2.4 Известные соотношения между распределениями.

1) Нормальное распределение является предельным для биномиального распределения. Вспомним интегральную теорему **Муавра-Лапласа:** Если вероятность p наступления события A в каждом испытании постоянна и отлична от нуля и единицы, то вероятность $P_n(k_1 \leq m \leq k_2)$ (число успехов из указанного интервала) может быть найдена по приближенной формуле:

$$P_n(k_1 \leq m \leq k_2) \approx \frac{1}{\sqrt{2\Pi}} \int_{x_1}^{x_2} e^{-\frac{x^2}{2}} dx$$

где $x_1 = \frac{k_1 - np}{\sqrt{npq}}$ и $x_2 = \frac{k_2 - np}{\sqrt{npq}}$
Получается, что случайная величина X распределена по нормальному закону $X \sim N(np, npq)$
2) Предельная форма распределения **нормированной пуассоновской** случайной величины $(X - \theta)\theta^{-\frac{1}{2}}$, где X имеет распределение

$$P(X = x) = \frac{e^{-\theta}\theta^x}{x!}$$

- стандартное нормальное распределение. Это означает, что

$$\lim_{\theta \to \infty} P(\alpha < (X - \theta)\theta^{-\frac{1}{2}} < \beta) = \frac{1}{\sqrt{2\Pi}} \int_{\alpha}^{\beta} e^{-\frac{u^2}{2}} du$$

2.5 Моделирование случайной величины

Моделирование нормального распределения $\mathrm{N}(0,1)$ **модифицированным полярным методом**

Результат: (ξ_1, ξ_2).

1. (*Отбор в круг*)

- Do

  $\mathrm{Get}(\alpha_1, \alpha_2)$; $\beta_1 \leftarrow 2 * \alpha_1 - 1$; $\beta_2 \leftarrow 2 * \alpha_2 - 1$; $d \leftarrow \beta_1 * \beta_1 + \beta_2 * \beta_2$;
- While $d > 1$;

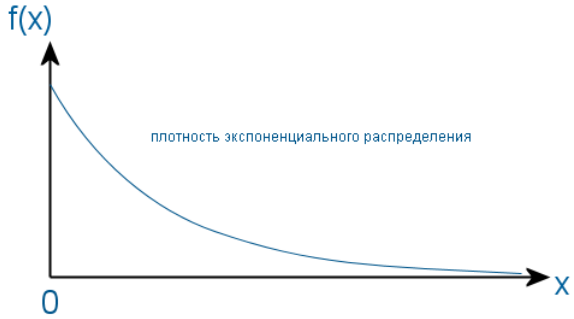
3. (*Моделирование*) $t \leftarrow \mathrm{sqrt}(-\ln(2 * d)/d)$; $\xi_1 \leftarrow \beta_1 * t$; $\xi_2 \leftarrow \beta_2 * t$; STOP.

ВАЖНО. На третьем пункте у нас происходит генерация $Get(\alpha)$. Затем под логарифмом мы эту величину умножаем на 2: $ln(2 * \alpha)$. Так как

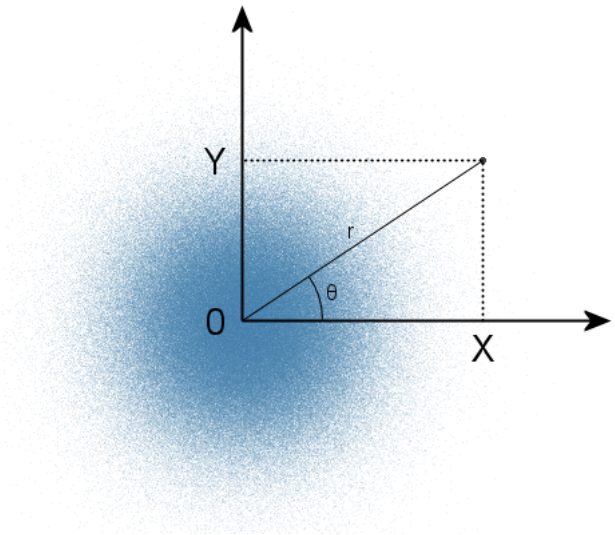
α может быть больше 0.5, то значение $ln(2 * \alpha)$ может быть положительно. Следовательно под корнем $\sqrt{-ln(2 * \alpha)/d}$ может быть отрицательное число. Следовательно, корень отрицательного числа - комплексное число. В интернете я также нашел данный алгоритм, но там двойка была за ln: $\sqrt{-2 * ln(\alpha)}$. Видимо, это опечатка.

Обоснование моделирования

Распределение χ^2 при $k = 2$ является экспоненциальным.



То есть у точки в прямоугольной системе координат будут случайные величины X и Y, распределенные нормально. Тогда после перевода этих координат в полярную систему координат (r, θ) квадрат радиуса будет распределен по экспоненциальному закону, так как квадрат радиуса - это сумма квадратов координат. Плотность распределения таких точек:



Угол θ будет иметь равномерное распределение в диапазоне от $[0, 2\pi]$. Также работает и в "обратную"сторону: если задать точку в полярной системе координат с помощью двух независимых случайных величин(угла, распределенного равномерно и раиуса, распределенного экспоненциально), то прямоугольные координаты данной точки будут яв-

ляться независимыми нормальными случайными величинами. Экспоненциальное распределение из равномерного можно получить с помощью метода того же обратного преобразования. Это называется полярный метод Бокса-Миллера.

$$X = r \cdot \cos\theta$$

$$Y = r \cdot \sin\theta$$

это нормально распределенные независимые случайные величины. Для получения r и θ необходимо сгенерировать две равномерное распределенные случайные величины (назовем их u и v) на отрезке (0,1), распределение одной из которых (например, v) надо преобразовать в экспоненциальное для получения радиуса. Функция экспоненциального распределения:

$$f(x) = 1 - e^{-\lambda x}$$

Тогда обратная функция:

$$f_1(x) = -\frac{\ln(1-x)}{x}$$

Из-за симметрии равномерного распределения аналогичное преобразование будет работать и с функцией:

$$f_2(x) = -\frac{\ln(x)}{\lambda}$$

Экспоненциальное распределение совпадает с распределеним χ^2 при $\lambda = \frac{1}{2}$. Подставим в f_1 λ, v и получим квадрат радиуса:

$$r^2 = -2\ln(v)$$

$$r = \sqrt{-2\ln(v)}$$

Получим угол θ:

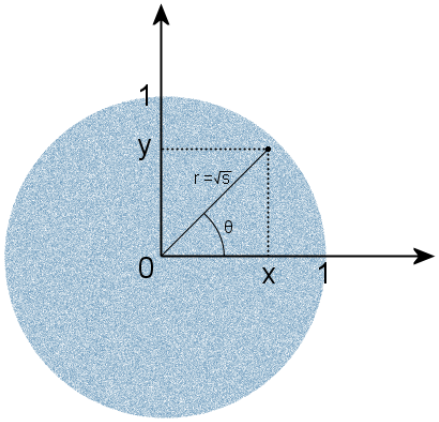
$$\theta = 2\pi u$$

Надо подставить r и θ в формулы для X и Y:

$$X = \sqrt{-2\ln(v)} \cdot \cos 2\pi u$$

$$Y = \sqrt{-2\ln(v)} \cdot \sin 2\pi v$$

Но в моделировании используется немного измененый алгоритм: выберем случайную точку из равномерно-распределенных в круге единичного радиуса и обозначим квадрат длины радиус-вектора s (в нашем алгоритме это величина d).



Выбор осуществляется заданием случайных прямоугольных координат (x,y), равномерно распределенных на интервале (-1,1), а у s следующее условие:

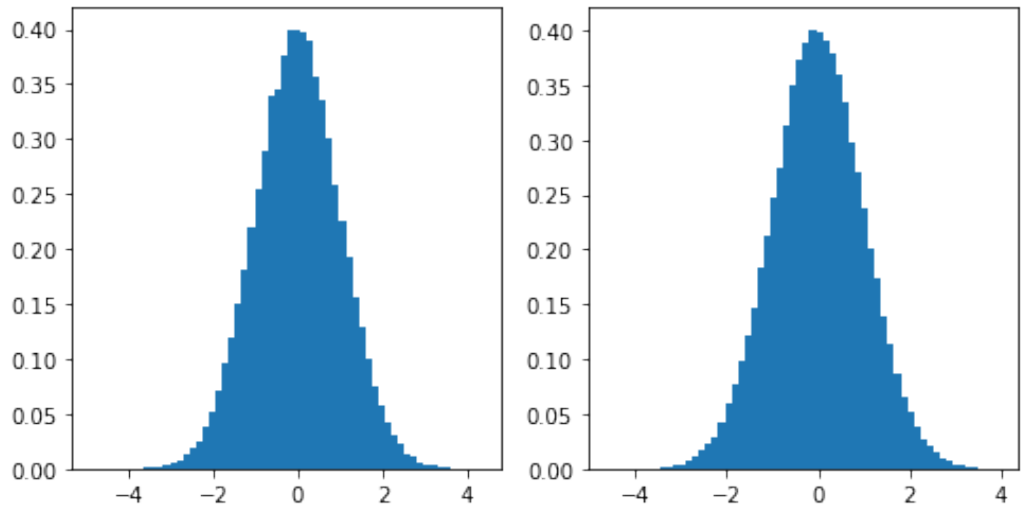
$$0 < s < 1$$

Тогда угол θ будем равномерно распределен. Также s имеет равномерное распределение. Полученные s и θ будут независимы друг от друга. Значит мы можем воспользоваться s для получения экспоненциального распределения. Теперь надо подставить s в формулы X, Y вместо v, и поделить координаты на длину радиус-вектора ($\sqrt{s}$)

Реализация на Python. Выходные случайные величины имеют стандартное нормальное распределение:

```
# моделирование нормального распределения
import numpy as np
import math
def normal():
    while True:
        alpha1 = np.random.rand()
        alpha2 = np.random.rand()
        beta1 = 2 * alpha1 - 1
        beta2 = 2 * alpha2 - 1
        d = beta1 ** 2 + beta2 ** 2
        if d <= 1:
            break
    t = ((-2) * math.log(d) / d) ** (0.5)
    return beta1 * t
```

График выборки. На левом графике находится выборка, сгенерированная моей функцией, на правом графике выборка, сгенерированная функцией numpy.random.import()



3 Домашнее задание 2. Геометрическое распределение

3.1 Построение выборок

```
# создает 25 файлов с выборками
mas = [5, 10, 100, 1000, 100000]
for i in range(len(mas)):
    for j in range(1, 6):
        mas1 = []
        for k in range(mas[i]):
            mas1.append(my_geom_model(0.3))
#         with open('fileeeeeee{}.txt'.format('{}_{}'.format(mas[i], j)), 'w') as file:
#             print(*mas1, file=file)
```

То есть при выполнении этой программы создается 25 файлов с выборками. Ниже примеры для $n = 5, 10$
10,0,1,10,0
0,0,0,1,0
0,0,1,0,2
6,2,0,3,1
0,3,0,3,2
4,1,1,8,2,4,2,1,0,3
7,1,2,0,0,0,3,0,1,2
6,5,8,9,4,0,5,1,0,1
1,1,2,1,0,4,1,0,4,5
0,1,4,3,2,0,5,0,0,0

3.2 Графики эмпирической функции распределения и функции распределения.

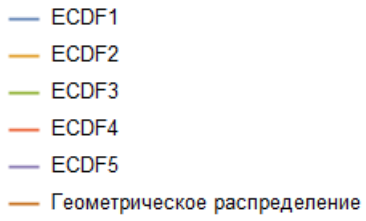
Построение будет осуществляться в Wolfram Mathematica, но с выборками, построенными в Python.

пример кода:

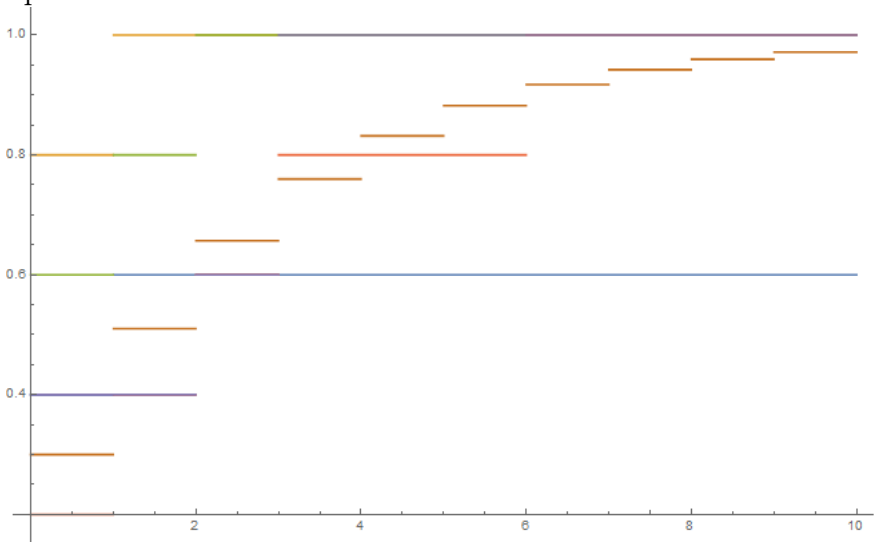
```
a1 = ReadList["D:\\Python\\file1000_1.txt", Number];
     считать в список                       число
a2 = ReadList["D:\\Python\\file1000_2.txt", Number];
     считать в список                       число
a3 = ReadList["D:\\Python\\file1000_3.txt", Number];
     считать в список                       число
a4 = ReadList["D:\\Python\\file1000_4.txt", Number];
     считать в список                       число
a5 = ReadList["D:\\Python\\file1000_5.txt", Number];
     считать в список                       число
c1 = EmpiricalDistribution[a1];
     эмпирическое распределение
c2 = EmpiricalDistribution[a2]; c3 = EmpiricalDistribution[a3];
     эмпирическое распределение      эмпирическое распределение
c4 = EmpiricalDistribution[a4]; c5 = EmpiricalDistribution[a5];
     эмпирическое распределение      эмпирическое распределение
```

```
In[55]:= Plot[{CDF[c1, x], CDF[c2, x], CDF[c3, x], CDF[c4, x], CDF[c5, x], CDF[GeometricDistribution[0.3], x]},
          граф… функция ра… функция ра… функция ра… функция ра… функция ра… ф… геометрическое распределение
  {x, 0, 10}, PlotLegends → {"ECDF1", "ECDF2", "ECDF3", "ECDF4", "ECDF5", "Геометрическое распределение"}]
                легенды графика
```

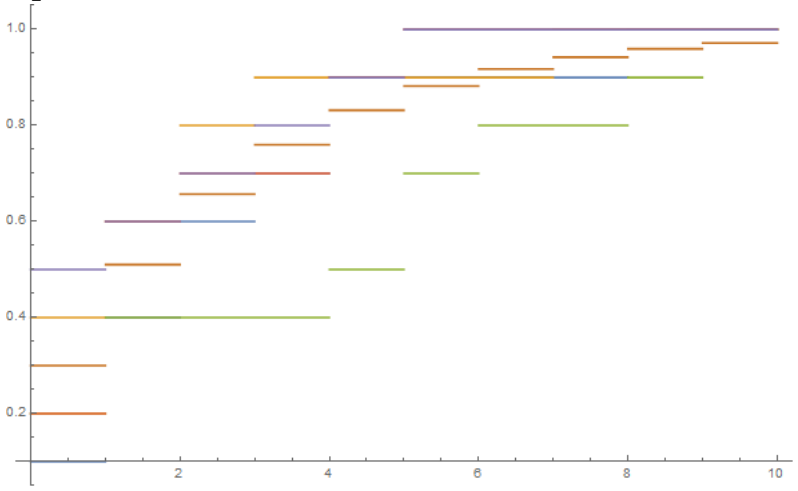
Далее меняться будет только имя файла(например, file5_1.txt) Легенда графиков:



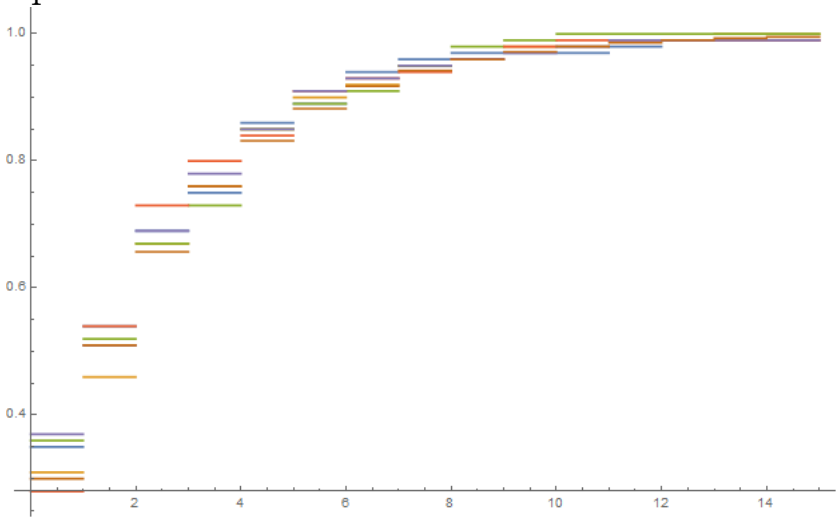
при $n = 5$



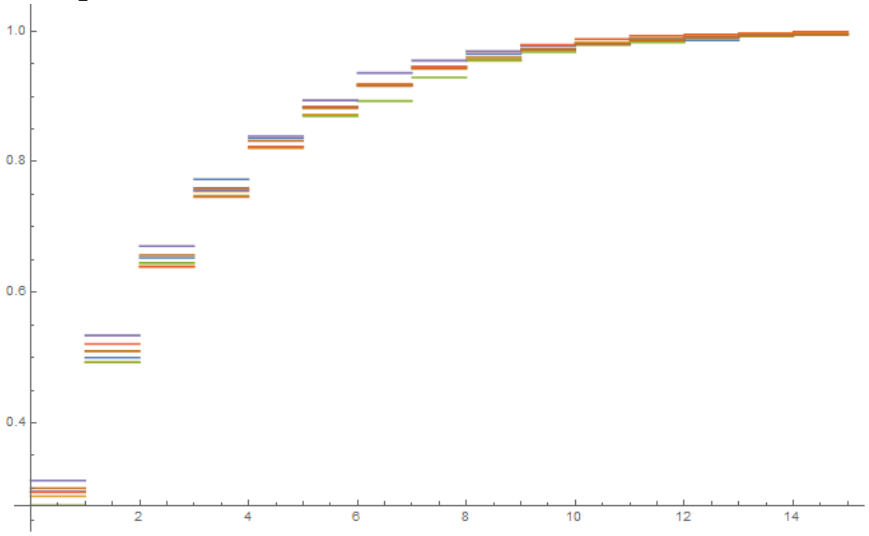
при $n = 10$



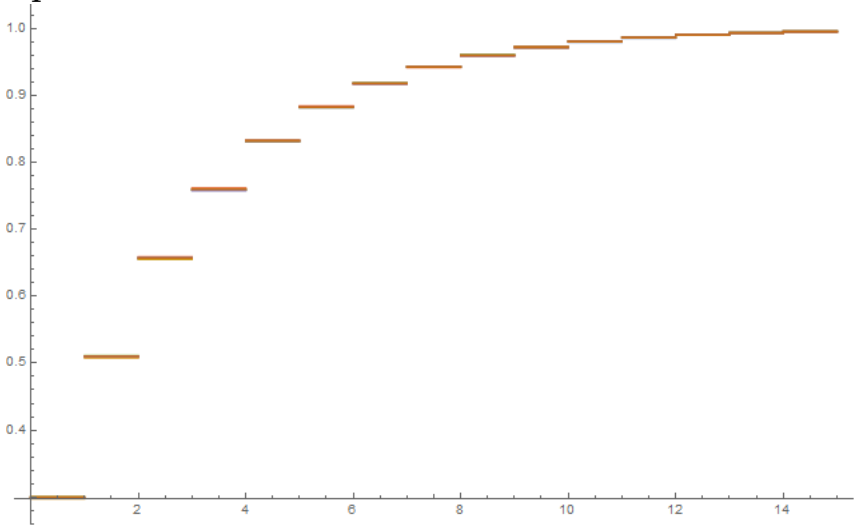
при $n = 100$



при $n = 1000$



при $n = 100000$



Как видно, чем больше n, тем более эмпирическая фунцкция распределения совпадает с функцией распределения. Это подтверждает теорему:
Для $\forall x \in R$ и для $\forall \varepsilon > 0$ при $n \to \infty$

$$P(|\hat{F}_n(x) - F(x)| < \varepsilon) \to 1$$

Тогда

$$\hat{F}_n(x) \xrightarrow{P} F(x)$$

```
# нахождение значений эмпирической функции для выборок и нахождение верхних границ э.ф.
from collections import Counter
import numpy as np
import math
# функция, которая убирает повторяющиеся элементы массива
def f(l):
    n = []
    for i in l:
        if i not in n:
            n.append(i)
    return n

# эфр на выборке ( массив a) от значения b
def funct(a, b):
    s = 0
    for i in range(len(a)):
        if a[i] < b:
            s +=1
        else:
            break
    return(s / len(a))



mas = [5, 10, 100, 1000, 100000]
for k1 in range(5):
    data = []
    maxs = []
    for i in range(len(mas)):
        data.append (np.loadtxt("file{}_{}.txt".format(mas[k1], i + 1), delimiter=' ',
                                dtype=np.float))
    for i in range(len(data)):
        data[i].sort()
    for i in range(4):
        for j in range(i + 1, 5):
            max = -100000
            mas1 = sorted(f(np.concatenate([data[i], data[j]])))
            for k in range(len(mas1)):
                if math.fabs(funct(data[i],mas1[k]) - funct(data[j],mas1[k])) > max:
                    max = math.fabs(funct(data[i],mas1[k]) - funct(data[j],mas1[k]))
            maxs.append(max)
    print('n={}'.format(mas[k1]),maxs)
```

Максимальные границы разности пар эмпирических функций

n = 5	n = 10	n = 100
0.4	0.30000000000000004	0.08000000000000002
0.4	0.4	0.03000000000000027
0.4	0.19999999999999996	0.06999999999999995
0.4	0.4	0.03000000000000027
0.20000000000000007	0.5	0.06
0.6000000000000001	0.20000000000000007	0.08000000000000002
0.6	0.10000000000000009	0.06
0.4	0.4	0.07999999999999996
0.4	0.4	0.05000000000000044
0.2	0.3	0.08999999999999997

n = 1000	n = 100000
0.018000000000000016	0.0010000000000000009
0.025000000000000022	0.0017599999999999838
0.027000000000000024	0.002509999999999901
0.03400000000000003	0.0015499999999999403
0.02400000000000002	0.0026200000000000667
0.028000000000000025	0.0031199999999999006
0.041000000000000036	0.0021599999999999397
0.028000000000000025	0.0018700000000000383
0.04300000000000004	0.0013199999999999878
0.03200000000000003	0.0029000000000000137

3.3 Построение вариационного ряда выборки

```
mas = [5, 10, 100, 1000, 100000]
data = []
for i in range(len(mas)):
    with open("file{}.txt".format('{}_{}'.format(mas[0], i + 1))) as f:
        for line in f:
            data.append(list([float(x) for x in line.split()]))
for i in range(5):
    data[i] = sorted(data[i])
```

примеры для $n = 5$ и $n = 10$:

```
[0.0, 0.0, 1.0, 10.0, 10.0]
[0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 0.0, 0.0, 1.0, 2.0]
[0.0, 1.0, 2.0, 3.0, 6.0]
[0.0, 0.0, 2.0, 3.0, 3.0]
```

```
[0.0, 1.0, 1.0, 1.0, 2.0, 2.0, 3.0, 4.0, 4.0, 8.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 2.0, 2.0, 3.0, 7.0]
[0.0, 0.0, 1.0, 1.0, 4.0, 5.0, 5.0, 6.0, 8.0, 9.0]
[0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 2.0, 4.0, 4.0, 5.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
```

Функция, высчитывающая квантили:

```
def quantiles(mas, value1 = 0.1, value2 = 0.5, value3 = 0.7):
    our_quantiles1 = []
    our_quantiles2 = []
    our_quantiles3 = []
    number1 = int(value1 * len(mas[1]) + 1)
    number2 = int(value2 * len(mas[1]) + 1)
    number3 = int(value3 * len(mas[1]) + 1)
    for i in range(5):
        our_quantiles1.append(mas[i][number1 - 1])
    for i in range(5):
        our_quantiles2.append(mas[i][number2 - 1])
    for i in range(5):
        our_quantiles3.append(mas[i][number3 - 1])
    print('при n = {} квантили уровня 0.1:'.format(len(mas[1]))),
    for i in range(len(our_quantiles1)):
        print(our_quantiles1[i])
    print('при n = {} квантили уровня 0.5:'.format(len(mas[1]))),
    for i in range(len(our_quantiles2)):
        print(our_quantiles2[i])
    print('при n = {} квантили уровня 0.7:'.format(len(mas[1]))),
    for i in range(len(our_quantiles3)):
        print(our_quantiles3[i])

quantiles(data)
```

Результаты:

```
при n = 5 квантили уровня 0.1: [0.0, 0.0, 0.0, 0.0, 0.0]
при n = 5 квантили уровня 0.5: [1.0, 0.0, 0.0, 2.0, 2.0]
при n = 5 квантиль уровня 0.7: [10.0, 0.0, 1.0, 3.0, 3.0]

при n = 10 квантили уровня 0.1: [1.0, 0.0, 0.0, 0.0, 0.0]
при n = 10 квантили уровня 0.5: [2.0, 1.0, 5.0, 1.0, 1.0]
при n = 10 квантили уровня 0.7: [4.0, 2.0, 6.0, 4.0, 3.0]

при n = 100 квантили уровня 0.1: [0.0, 0.0, 0.0, 0.0, 0.0]
при n = 100 квантили уровня 0.5: [1.0, 2.0, 1.0, 1.0, 1.0]
при n = 100 квантили уровня 0.7: [3.0, 3.0, 3.0, 2.0, 3.0]

при n = 1000 квантили уровня 0.1: [0.0, 0.0, 0.0, 0.0, 0.0]
при n = 1000 квантили уровня 0.5: [2.0, 2.0, 2.0, 1.0, 1.0]
при n = 1000 квантили уровня 0.7: [3.0, 3.0, 3.0, 3.0, 3.0]

при n = 100000 квантили уровня 0.1: [0.0, 0.0, 0.0, 0.0, 0.0]
при n = 100000 квантили уровня 0.5: [1.0, 1.0, 1.0, 1.0, 1.0]
при n = 100000 квантили уровня 0.7: [3.0, 3.0, 3.0, 3.0, 3.0]
```

Формула квантилей для случайной величины, имеющей геометрическое распределение:

The **quantile function** for a **geometric** random variable is

$$Q(r;p) = \left\lceil \frac{\ln(1-r)}{\ln(1-p)} \right\rceil$$

for `0 < r < 1`, where `p` is the success probability.

Значения, которые должны получиться:

```
In[5]:= IntegerPart[Log[1 - 0.1] / Log[1 - 0.3] // N]
        целая часть  натуральный ло··· натуральный лога··· чис

Out[5]= 0

In[6]:= IntegerPart[Log[1 - 0.5] / Log[1 - 0.3] // N]
        целая часть  натуральный ло··· натуральный лога··· чис

Out[6]= 1

In[7]:= IntegerPart[Log[1 - 0.7] / Log[1 - 0.3] // N]
        целая часть  натуральный ло··· натуральный лога··· чис

Out[7]= 3
```

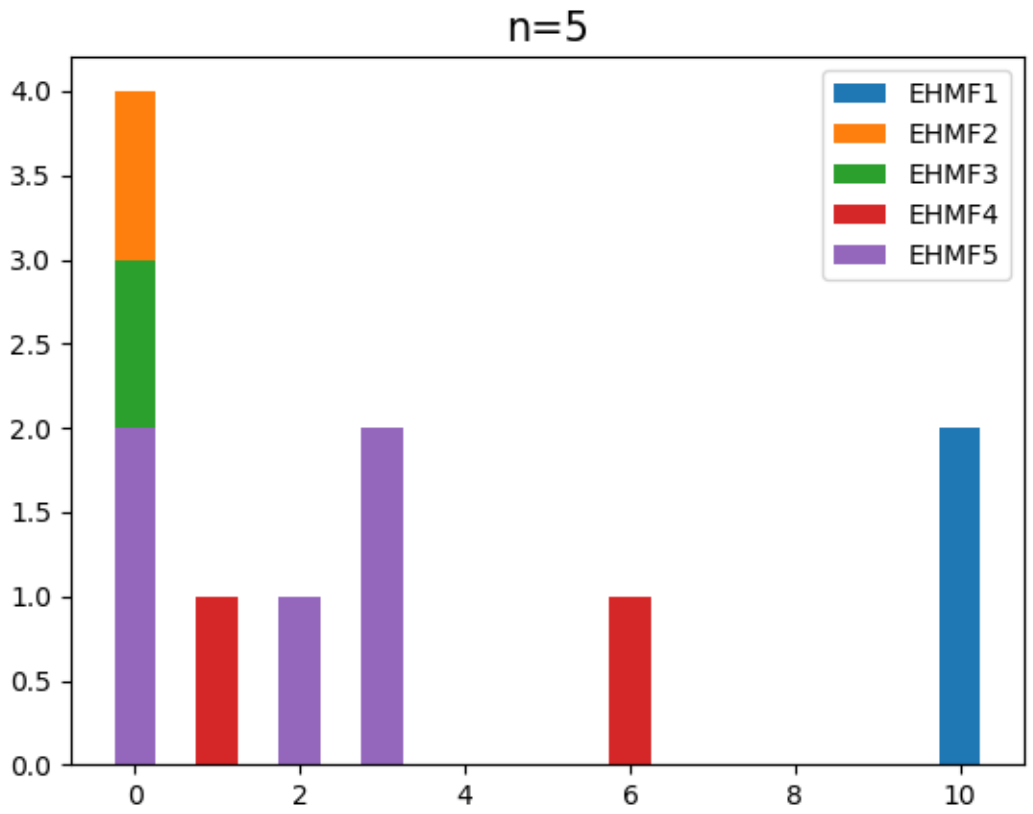
Как видно, чем больше n, тем точнее квантиль.

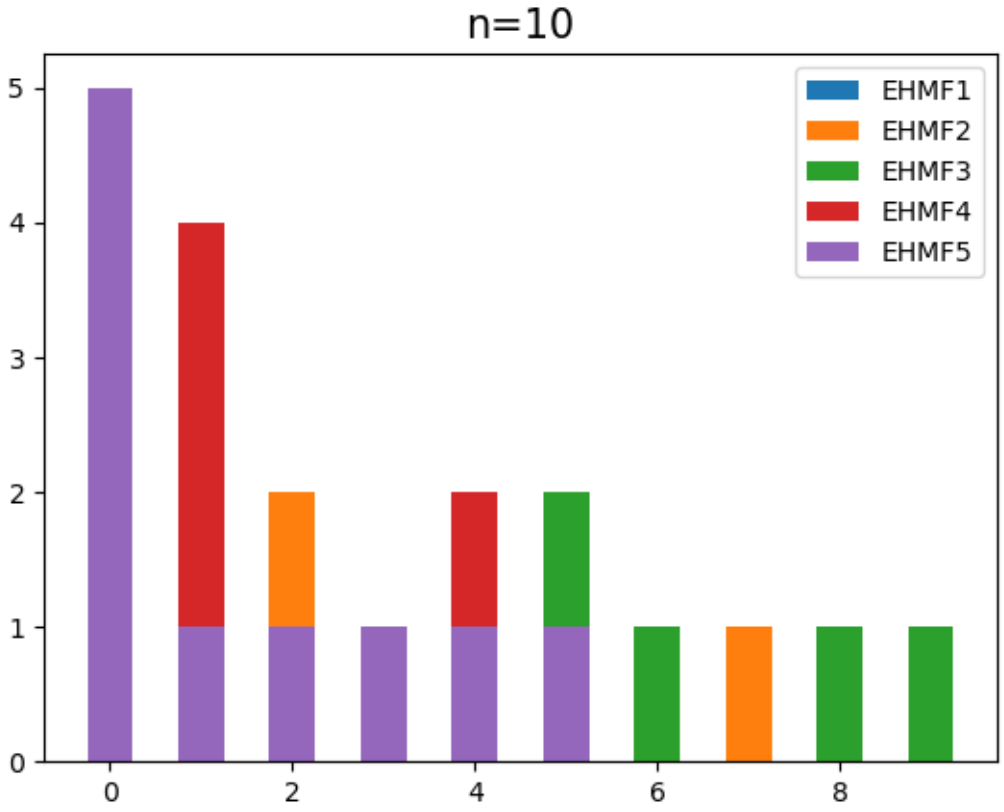
3.4 Построение гистограмм и полигонов частот

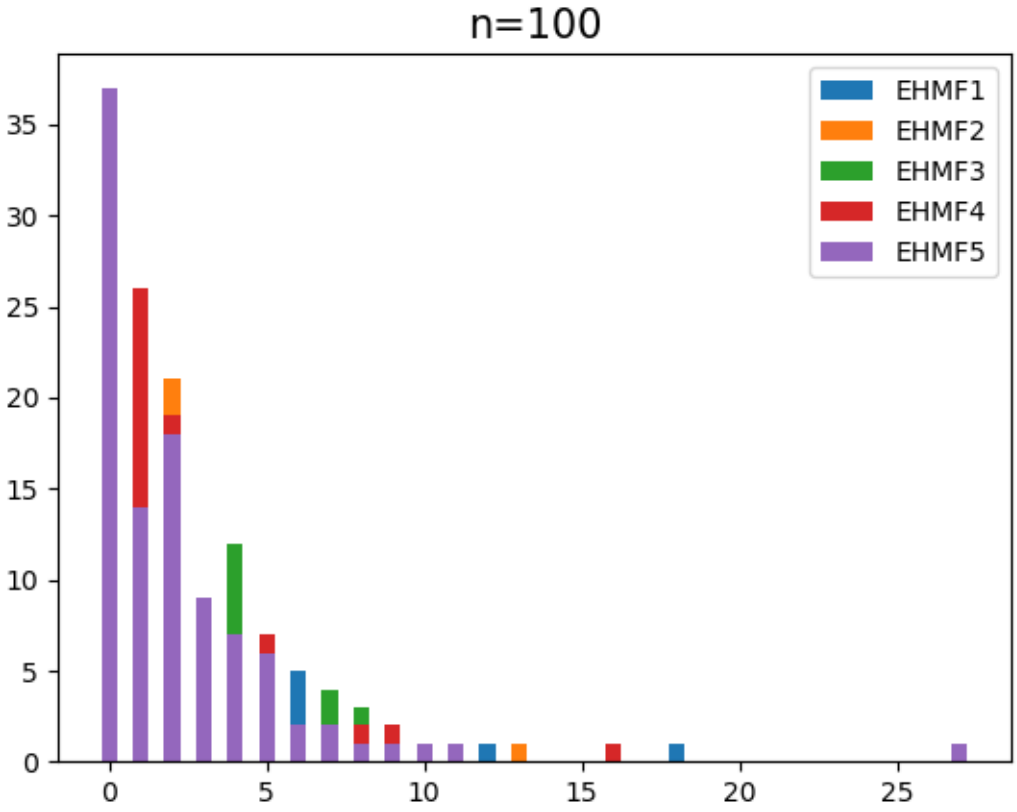
Способ построения гистограмм:

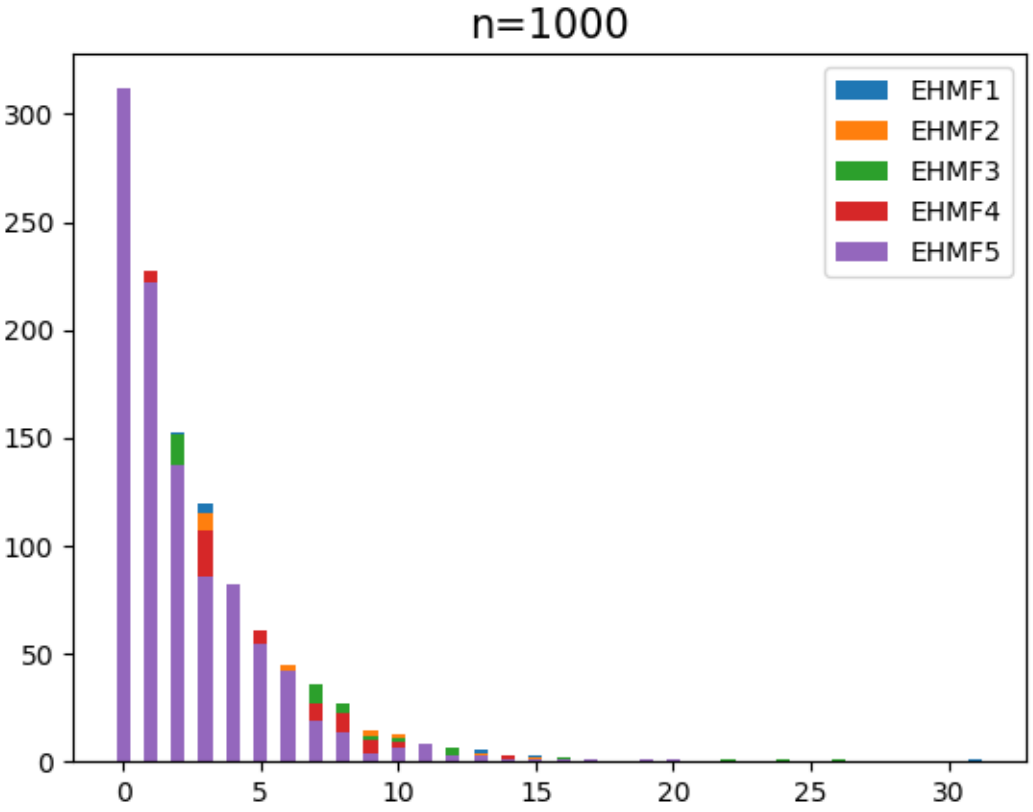
```
from collections import Counter
import matplotlib.pyplot as plt
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("file{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for i in range(5):
        data[i] = Counter(data[i])
    width = 0.5
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for i in range(5):
        ax.bar(list(data[i].keys()), list(data[i].values()), width, label='EHMF{}'.format(i+1))
    ax.set_title('n={}'.format(sum(list(data[0].values()))))
    ax.legend()
    plt.show()
```

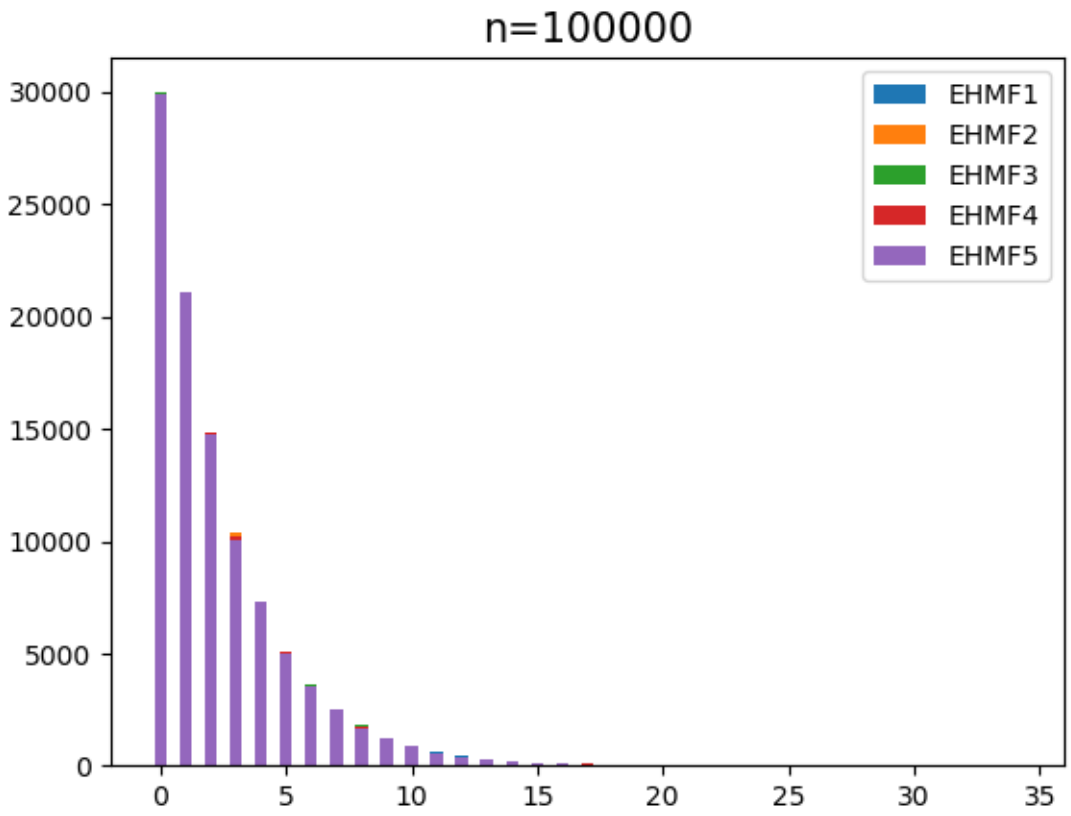
Гистограммы:




n=10
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0
1
2
3
4
5
0
2
4
6
8


n=100
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0
5
10
15
20
25
30
35
0
5
10
15
20
25

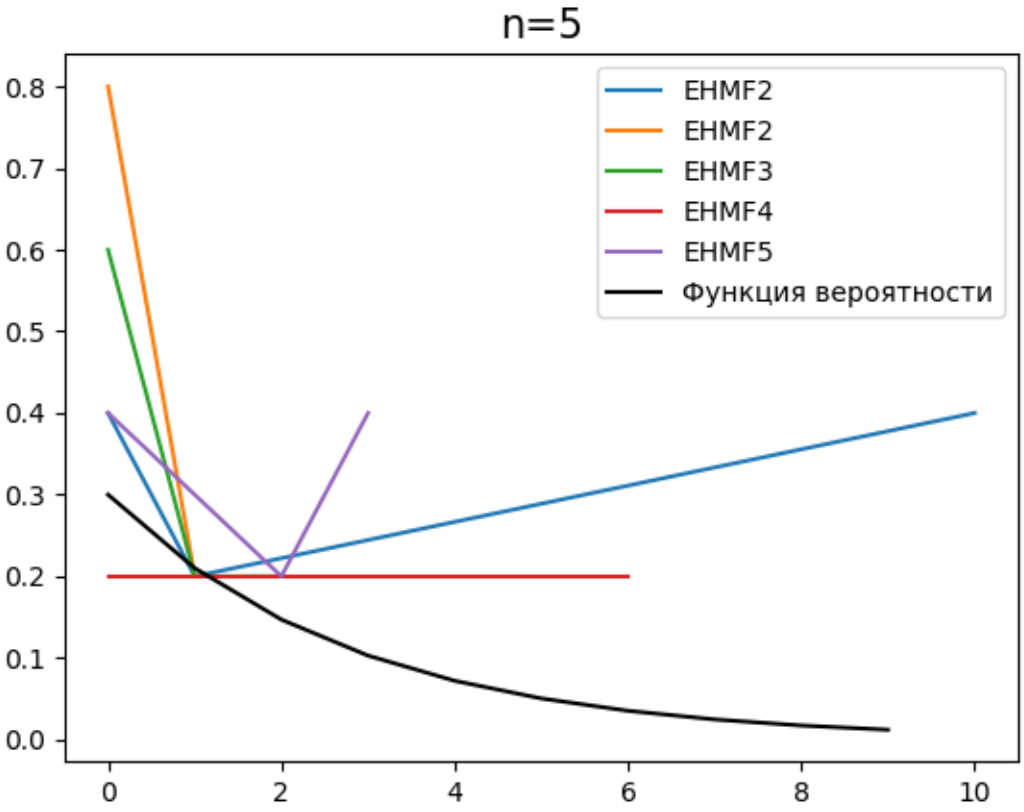

n=1000
EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
300
250
200
150
100
50
0
0
5
10
15
20
25
30

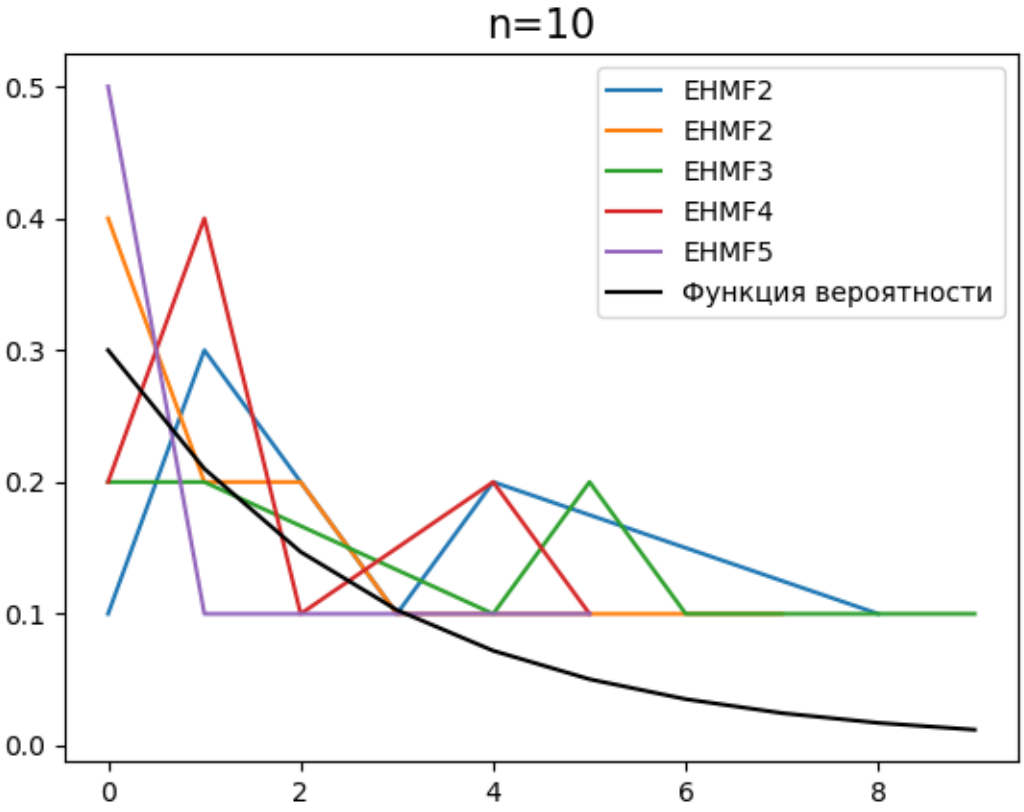


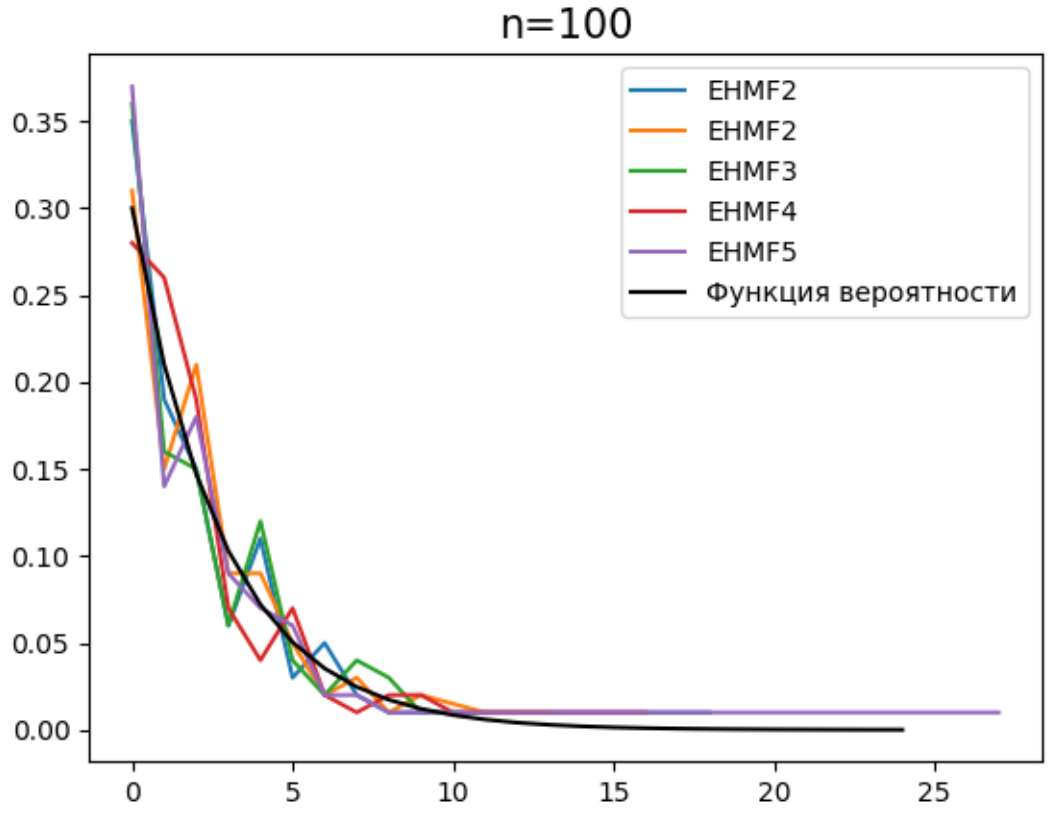
Способ построения полигонов частот:

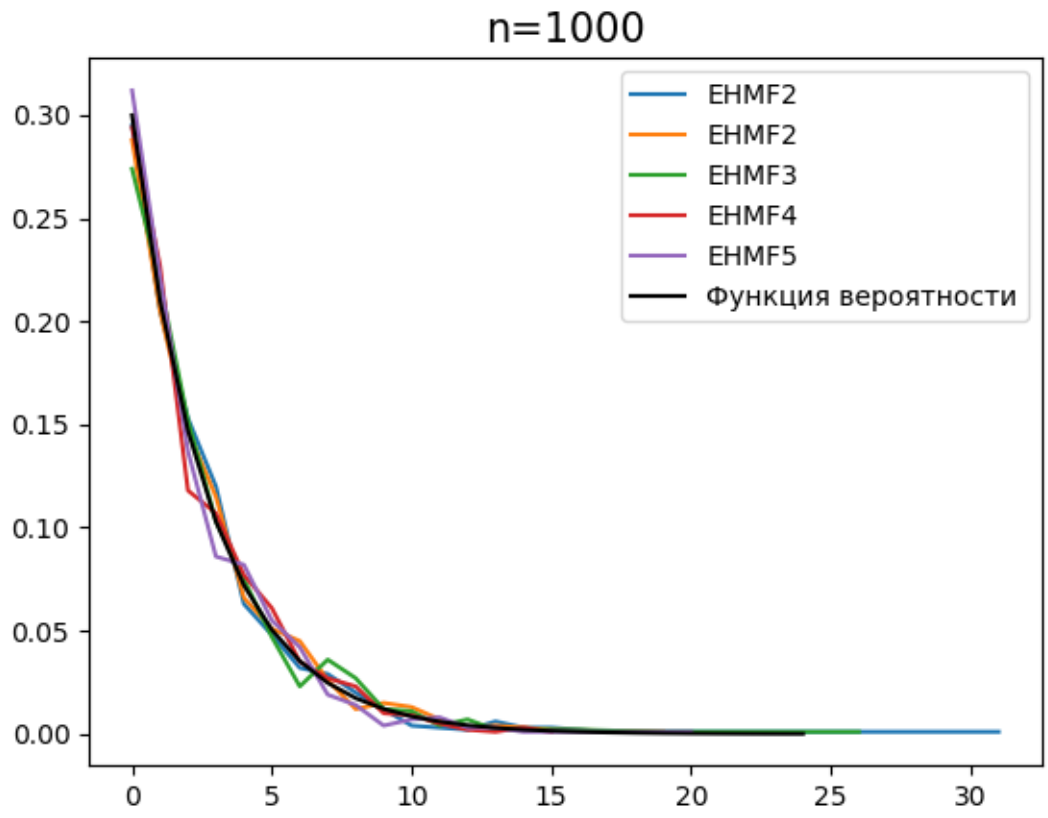
```python
# построение полигонов частот
from collections import Counter
import matplotlib.pyplot as plt

def geomm(p, i):
    return (1 - p) ** i * p

mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("file{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for j in range(5):
        data[j] = sorted(data[j])
    for j in range(5):
        data[j] = Counter(data[j])
    summ = sum(data[1].values())
    our_mas = []
    for i in range(5):
        our_dict = {}
        for key in data[i].keys():
            our_dict[key] = data[i][key] / summ
        our_mas.append(our_dict)

    d = []
    for i in range(20):
        d.append(geomm(0.3, i))
    width = 0.5
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for i in range(5):
        ax.plot(list(our_mas[i].keys()), list(our_mas[i].values()), label='EPMF{}'.format(i + 1))
    ax.set_title('n={}'.format(mas[k]))
    ax.legend()
    plt.show()
```

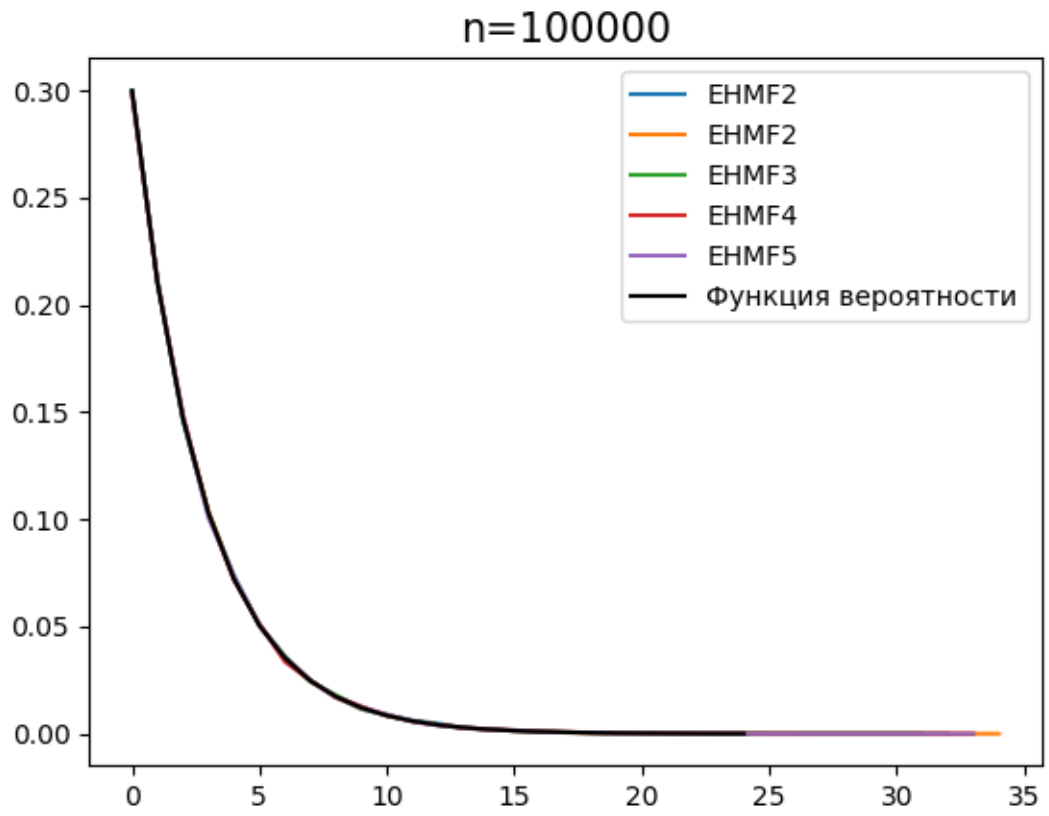
Полигоны:


n=5
EHMF2
EHMF2
EHMF3
EHMF4
EHMF5
Функция вероятности
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
10


n=10
EHMF2
EHMF2
EHMF3
EHMF4
EHMF5
Функция вероятности
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8


n=100
EHMF2
EHMF2
EHMF3
EHMF4
EHMF5
Функция вероятности
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25


n=1000
EHMF2
EHMF2
EHMF3
EHMF4
EHMF5
Функция вероятности
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30


n=100000
EHMF2
EHMF2
EHMF3
EHMF4
EHMF5
Функция вероятности
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30
35

4 Домашнее задание 2. Нормальное распределение

4.1 Построение выборок

```
# моделирование случайной величины, имеющей нормальное стандартное распределение
import numpy as np
import math
def normal():
    while True:
        alpha1 = np.random.rand()
        alpha2 = np.random.rand()
        beta1 = 2 * alpha1 - 1
        beta2 = 2 * alpha2 - 1
        d = beta1 ** 2 + beta2 ** 2
        if d <= 1:
            break
    t = ((-2) * math.log(d) / d) ** (0.5)
    return beta1 * t

# создает 25 файлов с выборками
mas = [5, 10, 100, 1000, 100000]
for i in range(len(mas)):
    for j in range(1, 6):
        mas1 = []
        for k in range(mas[i]):
            mas1.append(normal())
        with open('filenormal{}.txt'.format('{}_{}'.format(mas[i], j)), 'w') as file:
            print(*mas1, file=file)
```

примеры.n = 5
{{-1.7390618353554481, -1.5335815661663137, -0.3324430261430819, 0.6451154424542936, 1.6057681960841894}, {-1.4663782014740543, -0.9937204626803992, -0.9085481168601001, -0.531839634752429, 0.7115414286636189}, {-1.866110253802929, -1.0357089008187481, -0.059111174991449425, 0.15454806044145045, 1.4346289165411896}, {0.29072532012175395, 0.5219380952264526, 0.6270236428518352,

0.7541925090301437, 1.5820058340085246}, {-1.1293334585305543,
-0.34308308565560247, 0.08372356892872686, 0.7125824978566664,
0.8897263957166648}}
n = 10
{{-1.764961063251503, -1.0415077097071515, -0.8843918327849775,
-0.13697787620632346, -0.004426115033130548, 0.0249558277375861,
0.24174922190184212, 0.25103779932907544, 0.5287107933785409,
1.5479879004455182}, {-2.0514665381617343, -1.144500703901591,
-1.1156285847624876, -0.902588847659232, -0.8114010742641512,
0.3681879851526466, 0.3834079319245003, 0.3952706915719456,
0.8750467809940337, 1.3713916167503966},{-2.0710202031760514,
-1.0594206730366618, -0.9740680054292715, -0.7336122261108688,
-0.6888245249700418, 0.33103038377748617, 0.6892299584411499,
0.8107479033464768, 1.1978042509516247, 1.5790372901836052},
{-0.8792787792929041, -0.37060959609048055, -0.07066395702827302,
0.2428843342495532, 0.26660109967850604, 0.38737700258721214,
0.7280729247075317, 0.9320330643179978, 1.3643021553210717,
1.9164934437109193}, {-1.1846985827170662, -1.0865409434045394,
-0.5318755296541551, -0.3977483353510811, -0.065696920192765 86,
0.3243871600268893 3, 0.4379150644166077, 0.7853323485849844,
0.9161010376064156, 1.243104440869849}}

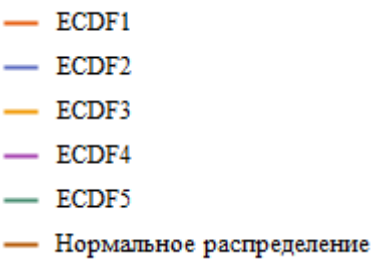
4.2 Графики эмпирической функции распределения и функции распределения

.

Построение будет осуществляться, как и геометрическое распределение, в **Wolfram Mathematica**

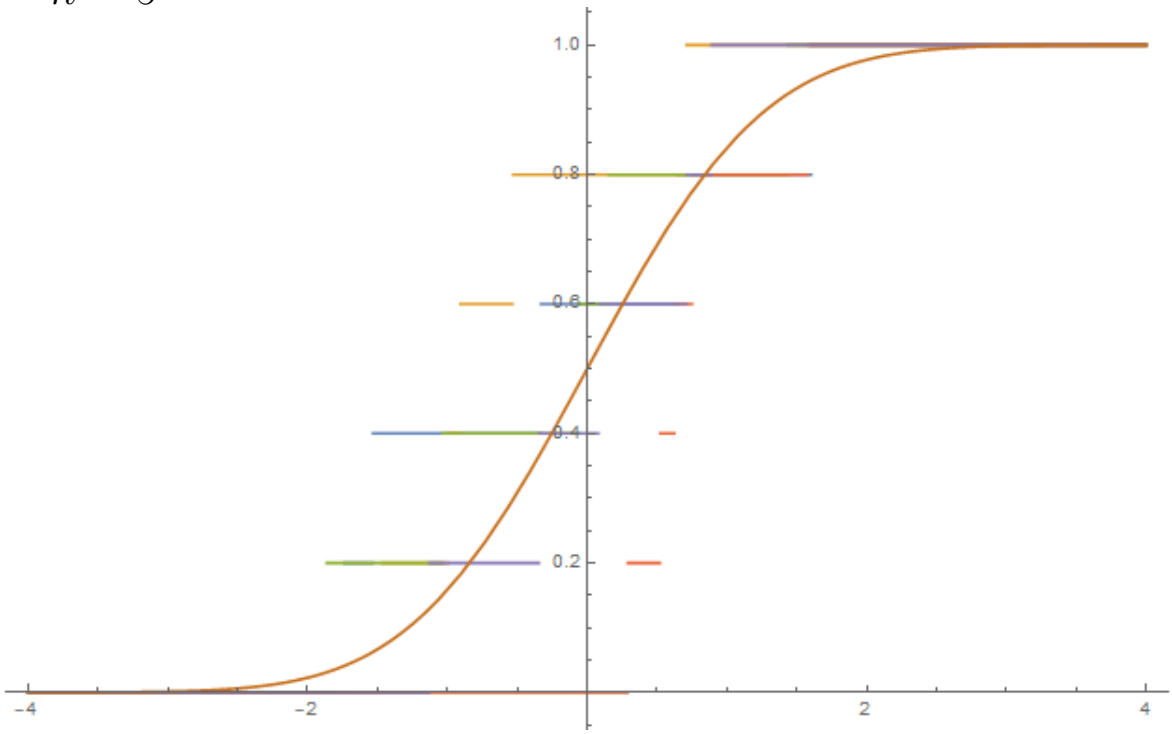
Код:

```
a1 = ReadList["D:\\Python\\filenormal5_1.txt", Number];
     считать в список                          число
a2 = ReadList["D:\\Python\\filenormal5_2.txt", Number];
     считать в список                          число
a3 = ReadList["D:\\Python\\filenormal5_3.txt", Number];
     считать в список                          число
a4 = ReadList["D:\\Python\\filenormal5_4.txt", Number];
     считать в список                          число
a5 = ReadList["D:\\Python\\filenormal5_5.txt", Number];
     считать в список                          число
c1 = EmpiricalDistribution[a1]; c2 = EmpiricalDistribution[a2];
     эмпирическое распределение   эмпирическое распределение
c3 = EmpiricalDistribution[a3]; c4 = EmpiricalDistribution[a4] ; c5 = EmpiricalDistribution[a
     эмпирическое распределение   эмпирическое распределение   эмпирическое распределение


Plot[{CDF[c1, x], CDF[c2, x], CDF[c3, x], CDF[c4, x], CDF[c5, x], CDF[NormalDistribution[],
 графи··  функция рас···  функция рас···  функция рас···  функция рас···  функция рас···  ф···  нормальное распределение
 {x, -5, 5}, PlotTheme → "Scientific", PlotStyle → Thick,
             тематический стиль графика   стиль графика  жирный
 PlotLegends → {"ECDF1", "ECDF2", "ECDF3", "ECDF4", "ECDF5", "Нормальное распределение"} ]
 легенды графика
```

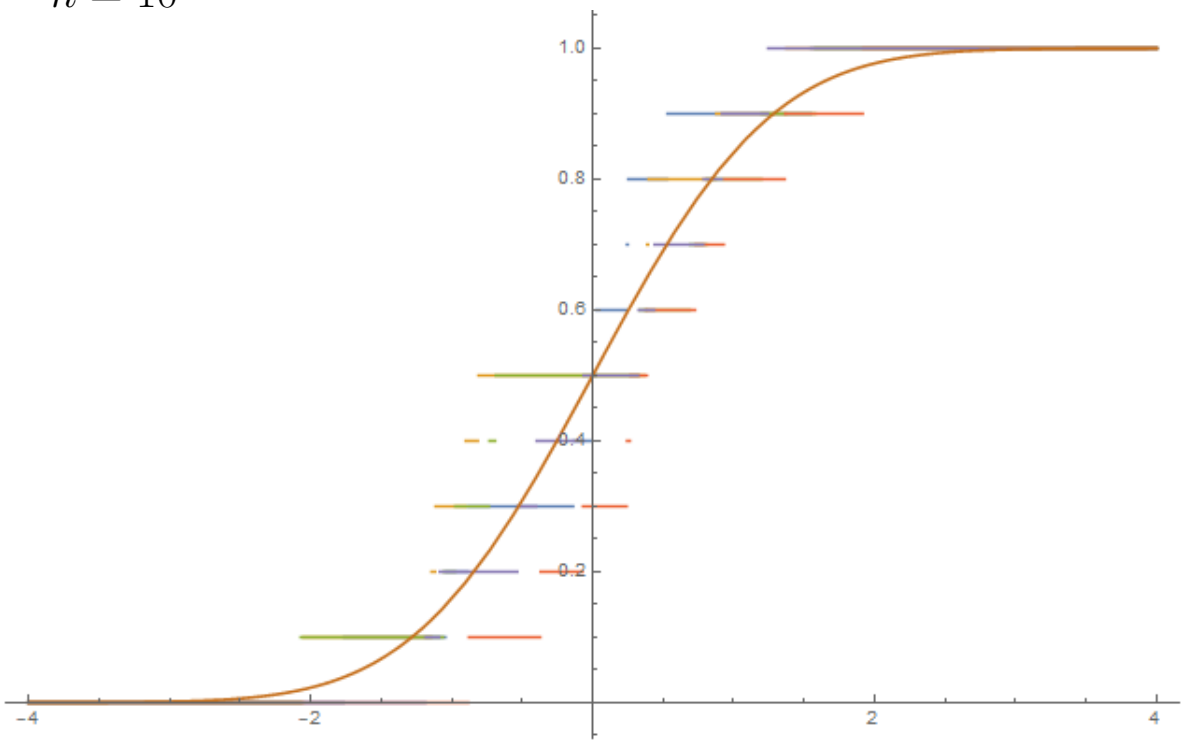


Легенды графиков:

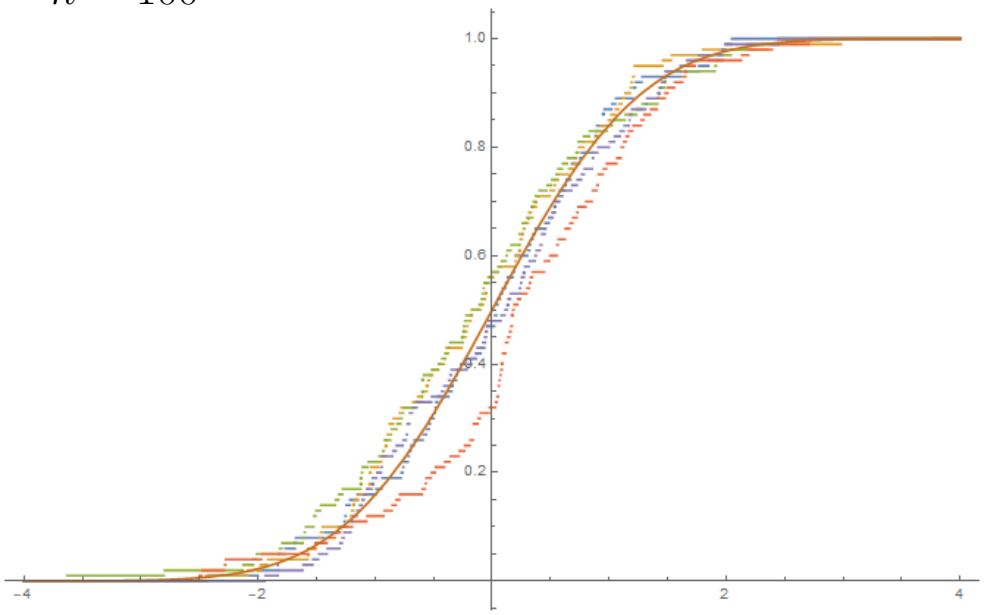
$n = 5$



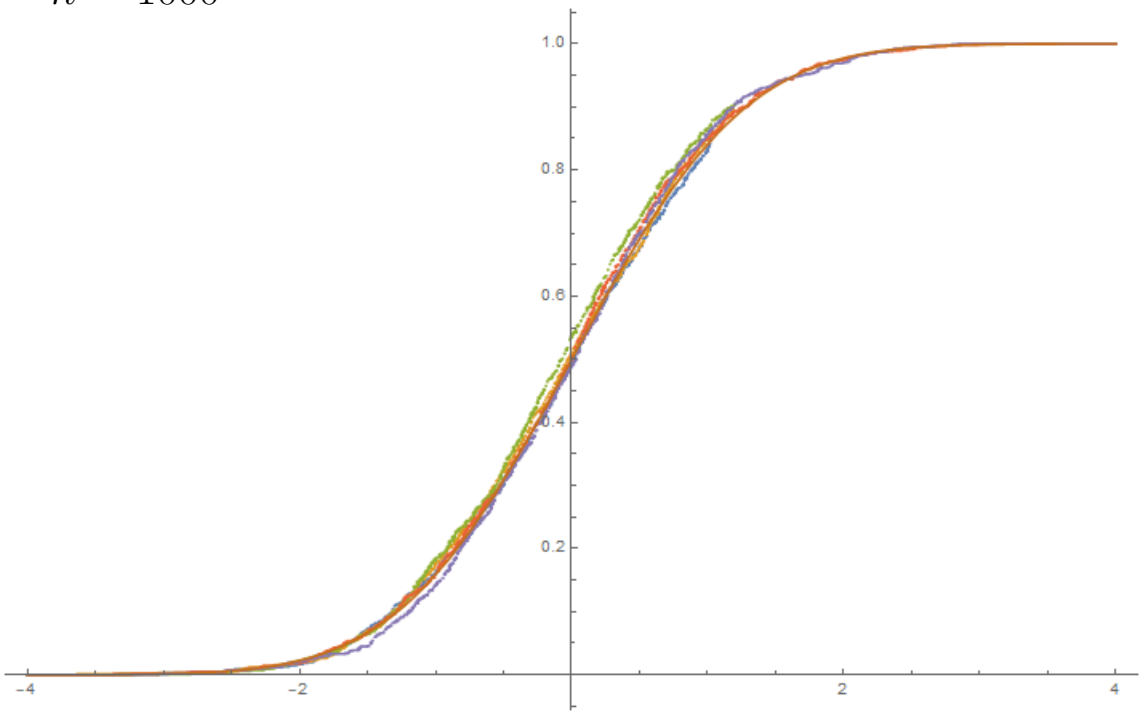
$n = 10$



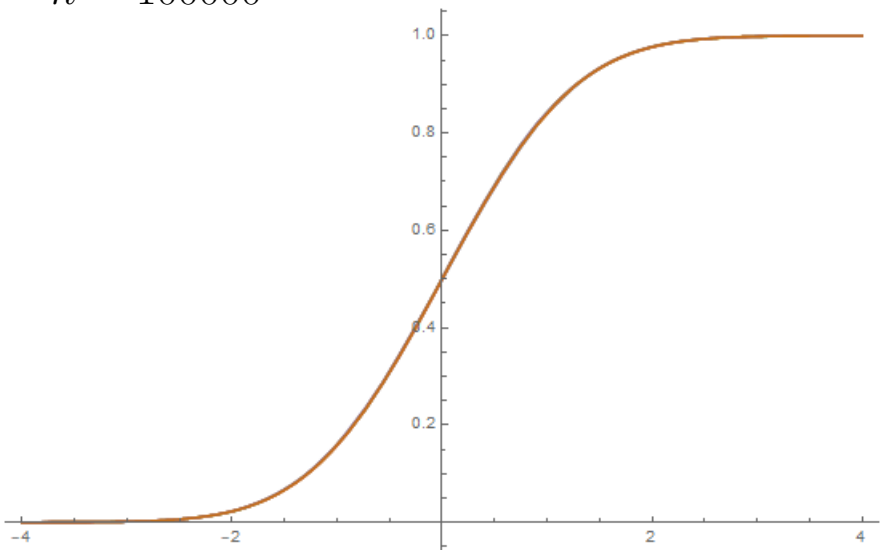
$n = 100$



$n = 1000$



$n = 100000$



Нахождение максимальных границ эмпирических функций

пример кода:

```
# нахождение значений эмпирической функции для выборок и нахождение верхних границ э.ф.
from collections import Counter
import numpy as np
import math
# функция, которая убирает повторяющиеся элементы массива
def f(l):
    n = []
    for i in l:
        if i not in n:
            n.append(i)
    return n

# эфр на выборке ( массив a) от значения b
def funct(a, b):
    s = 0
    for i in range(len(a)):
        if a[i] < b:
            s +=1
        else:
            break
    return(s / len(a))



mas = [5, 10, 100, 1000, 100000]
for k1 in range(4):
    data = []
    maxs = []
    for i in range(len(mas)):
        data.append (np.loadtxt("filenormal{}_{}.txt".format(mas[k1], i + 1), delimiter=' ',
                                dtype=np.float))
    for i in range(len(data)):
        data[i].sort()
    for i in range(4):
        for j in range(i + 1, 5):
            max = -100000
            mas1 = sorted(f(np.concatenate([data[i], data[j]])))
            for k in range(len(mas1)):
                if math.fabs(funct(data[i],mas1[k]) - funct(data[j],mas1[k])) > max:
                    max = math.fabs(funct(data[i],mas1[k]) - funct(data[j],mas1[k]))
            maxs.append(max)
    print('n={}'.format(mas[k1]),maxs)
```

Максимальные границы разности пар эмпирических функций.

Для n = 100 000 вычисление занимает слишком много времени, поэтому этих значений нет.

n = 5	n = 10	n = 100
0.4	0.30000000000000004	0.027000000000000024
0.20000000000000007	0.30000000000000004	0.05399999999999994
0.6	0.4	0.041000000000000036
0.4	0.30000000000000004	0.040999999999999925
0.4	0.20000000000000007	0.049000000000000044
0.8	0.4	0.03200000000000003
0.6000000000000001	0.3	0.03900000000000001
0.8	0.4	0.038000000000000034
0.20000000000000007	0.3	0.050000000000000044
0.6	0.30000000000000004	0.031000000000000028

n = 1000	n = 100000
0.027000000000000024	?
0.05399999999999994	?
0.041000000000000036	?
0.040999999999999925	?
0.049000000000000044	?
0.03200000000000003	?
0.03900000000000001	?
0.038000000000000034	?
0.050000000000000044	?
0.031000000000000028	?

4.3 Построение вариационного ряда выборки

Построение осуществляется также, как и в геометрическом распределении.
Примеры для n = 5:

-1.7390618353554481	-1.4663782014740543	-1.866110253802929	0.29072532012175395	-1.1293334585305543
-1.5335815661663137	-0.9937204626803992	-1.0357089008187481	0.5219380952264526	-0.34308308565560247
-0.3324430261430819	-0.9085481168601001	-0.059111174991449425	0.6270236428518352	0.08372356892872686
0.6451154424542936	-0.531839634752429	0.15454806044145045	0.7541925090301437	0.7125824978566664
1.6057681960841894	0.7115414286636189	1.4346289165411896	1.5820058340085246	0.8897263957166648

n=10:

-1.764961063251503	-2.0514665381617343	-2.0710202031760514	-0.8792787792929041	-1.1846985827170662
-1.0415077097071515	-1.144500703901591	-1.0594206730366618	-0.37060959609048055	-1.0865409434045394
-0.8843918327849775	-1.1156285847624876	-0.9740680054292715	-0.07066395702827302	-0.5318755296541551
-0.13697787620632346	-0.902588847659232	-0.7336122261108688	0.2428843342495532	-0.3977483353510811
-0.004426115033130548	-0.8114010742641512	-0.6888245249700418	0.26660109967850604	-0.06569692019276586
0.0249558277375861	0.3681879851526466	0.33103038377748617	0.38737700258721214	0.32438716002688933
0.24174922190184212	0.3834079319245003	0.6892299584411499	0.7280729247075317	0.4379150644166077
0.25103779932907544	0.3952706915719456	0.8107479033464768	0.9320330643179978	0.7853323485849844
0.5287107933785409	0.8750467809940337	1.1978042509516247	1.3643021553210717	0.9161010376064156
1.5479879004455182	1.3713916167503966	1.5790372901836052	1.9164934437109193	1.243104440869849

Нахождение кванителй такое же, как и в геометрическом распределении:

```
при n = 5 квантили уровня 0.1:
-1.7390618353554481
-1.4663782014740543
-1.866110253802929
0.29072532012175395
-1.1293334585305543
при n = 5 квантили уровня 0.5:
-0.3324430261430819
-0.9085481168601001
-0.059111174991449425
0.6270236428518352
0.08372356892872686
при n = 5 квантили уровня 0.7:
0.6451154424542936
-0.531839634752429
0.15454806044145045
0.7541925090301437
0.7125824978566664
```

```
при n = 10 квантили уровня 0.1:
-1.0415077097071515
-1.144500703901591
-1.0594206730366618
-0.37060959609048055
-1.0865409434045394
при n = 10 квантили уровня 0.5:
0.0249558277375861
0.3681879851526466
0.33103038377748617
0.38737700258721214
0.32438716002688933
при n = 10 квантили уровня 0.7:
0.25103779932907544
0.3952706915719456
0.8107479033464768
0.9320330643179978
0.7853323485849844
```

```
при n = 100 квантили уровня 0.1:
-1.2636063422998884
-1.2364511312094493
-1.5230624763947744
-1.1920100481844014
-1.204483013923443
при n = 100 квантили уровня 0.5:
0.04203562089952837
-0.0953127830278428
-0.1113750289731512
0.1874190457720955
0.13158604472674834
при n = 100 квантили уровня 0.7:
0.5700570034213173
0.3813643867854186
0.3709275514636254
0.8543799987439257
0.5761927192794468
```

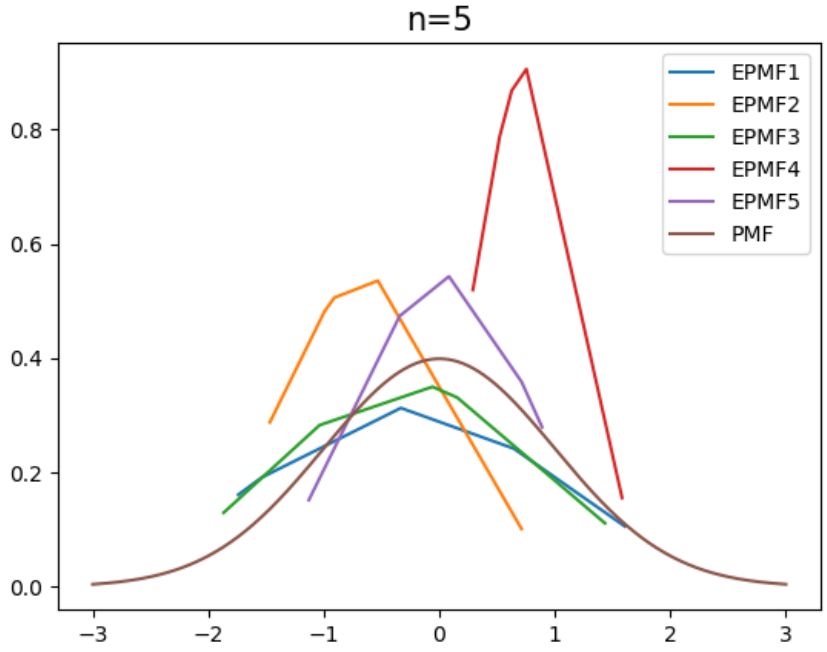
```
при n = 1000 квантили уровня 0.1:
-1.3181070995013082
-1.3087698992786059
-1.2998166315546684
-1.286559577006271
-1.187859969623271
при n = 1000 квантили уровня 0.5:
0.017476186189650633
-0.021919393124056205
-0.06866827829294705
-0.005083283094032185
0.03490806237374252
при n = 1000 квантили уровня 0.7:
0.5667390204217078
0.551305058298311
0.4221235435819775
0.4887929206816705
0.5025878240821313
```

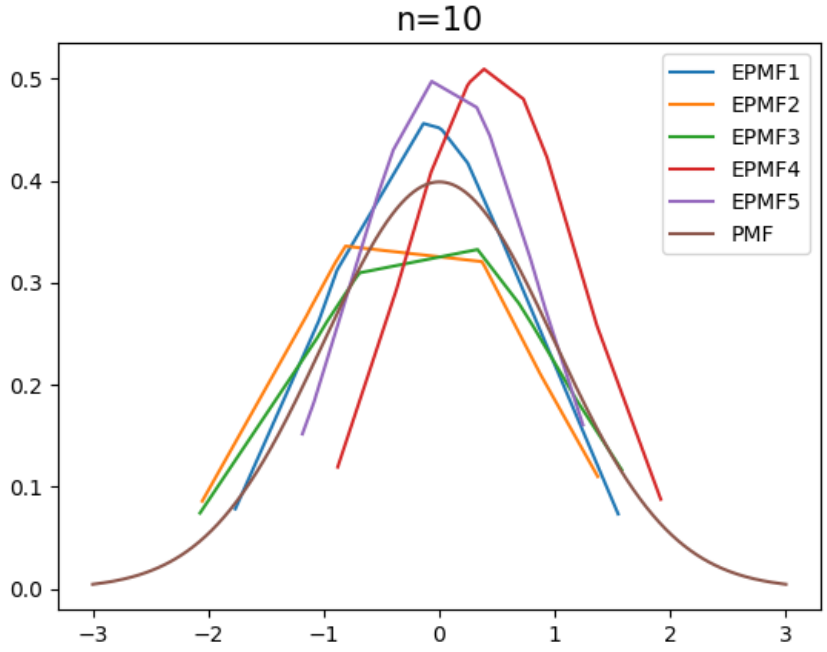
```
при n = 100000 квантили уровня 0.1:
-1.2813289745128595
-1.282082970554599
-1.2851306150076756
-1.2781314347756172
-1.288717064106276
при n = 100000 квантили уровня 0.5:
0.00418523668145405
-7.710575837067435e-05
-0.0025021563603506098
0.0015523476199955969
-0.005528668081748688
при n = 100000 квантили уровня 0.7:
0.5343613355088271
0.5244799120400663
0.5194197461854507
0.5266728861566355
0.5183228050506635
```

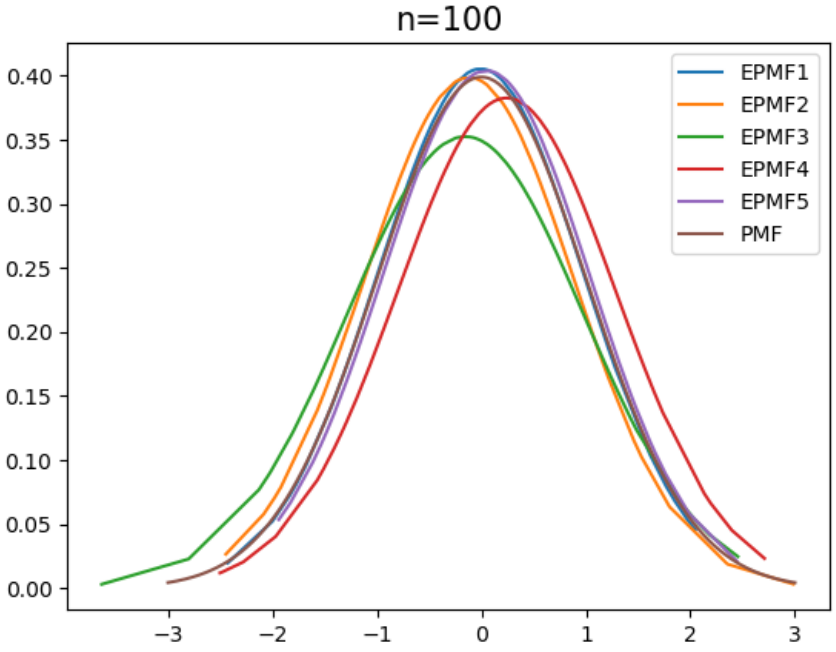
4.4 Построение гистограмм и полигонов частот

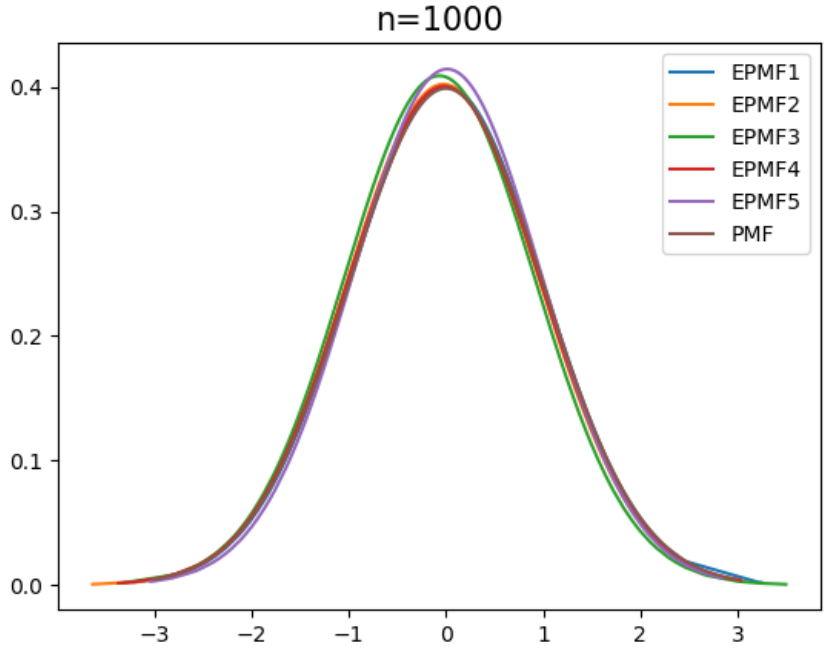
Способ построения полигонов частот:

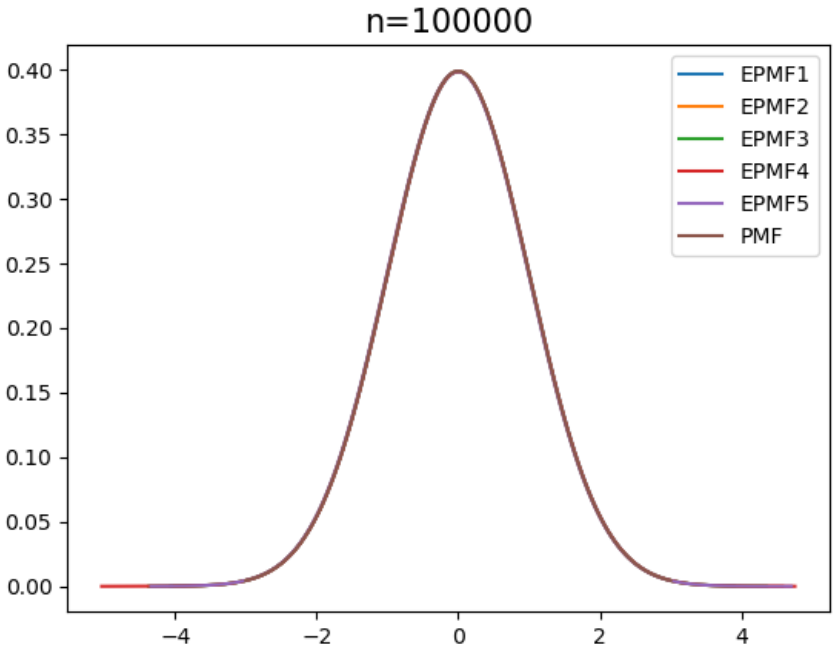
```
# построение полигонов частот
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(5):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for i in range(5):
        data[i] = sorted(data[i])
    x = np.linspace(-3, 3, 100000)
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for i in range(5):
        ax.plot(data[i], stats.norm.pdf(data[i], np.mean(data[i]), np.std(data[i])),
                label='EPMF{}'.format(i+1))
    ax.plot(x, stats.norm.pdf(x, 0, 1), label='EPMF')
    ax.set_title('{}'.format(mas[k]))
    ax.legend()
    plt.show()
```


n=5
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
0.0
0.2
0.4
0.6
0.8
−3
−2
−1
0
1
2
3


n=10
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
0.0
0.1
0.2
0.3
0.4
0.5
−3
−2
−1
0
1
2
3

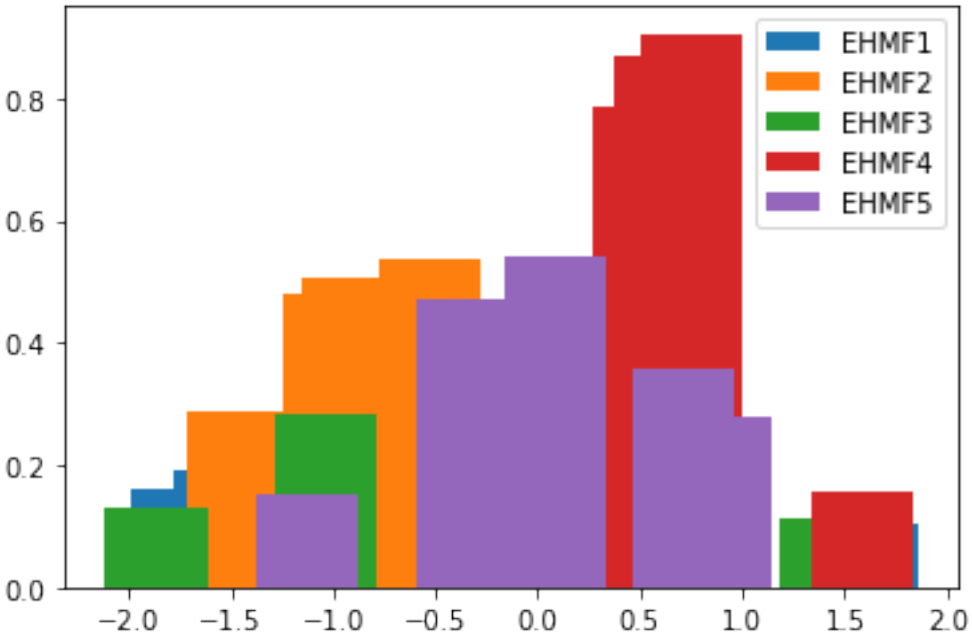

n=100
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−3
−2
−1
0
1
2
3

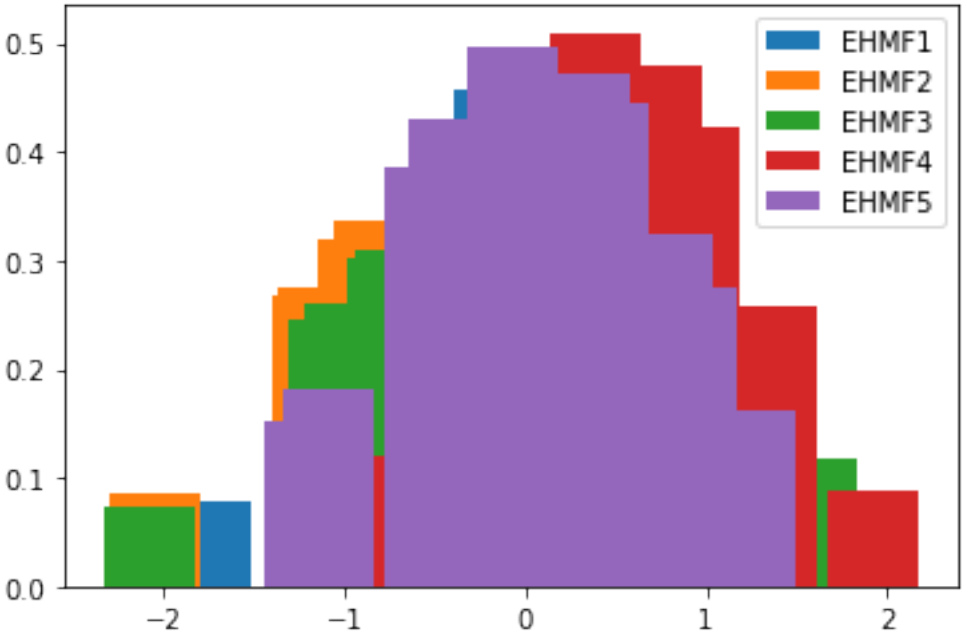

n=1000
EPMF1
EPMF2
EPMF3
EPMF4
EPMF5
PMF
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3

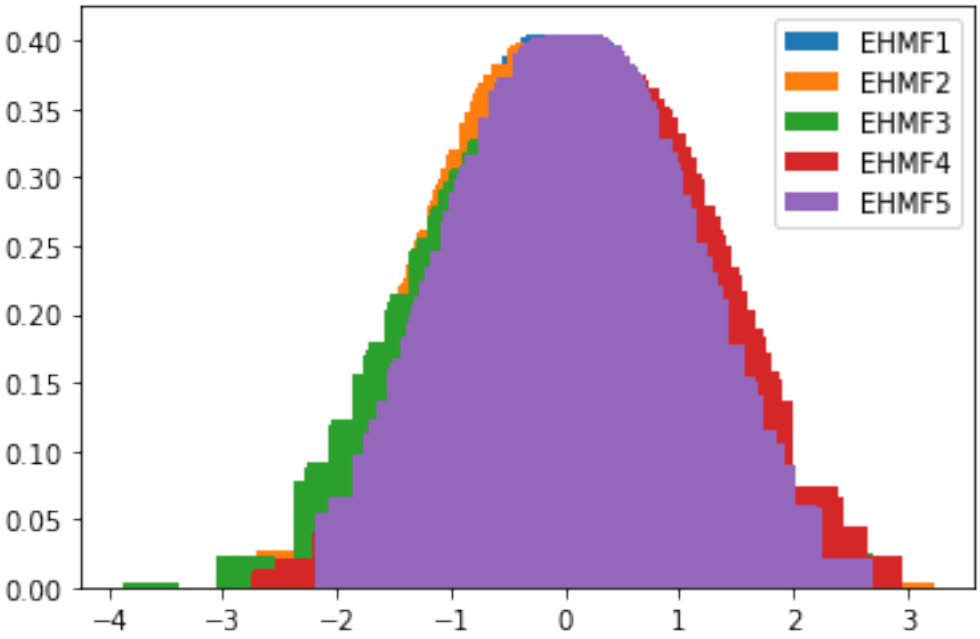


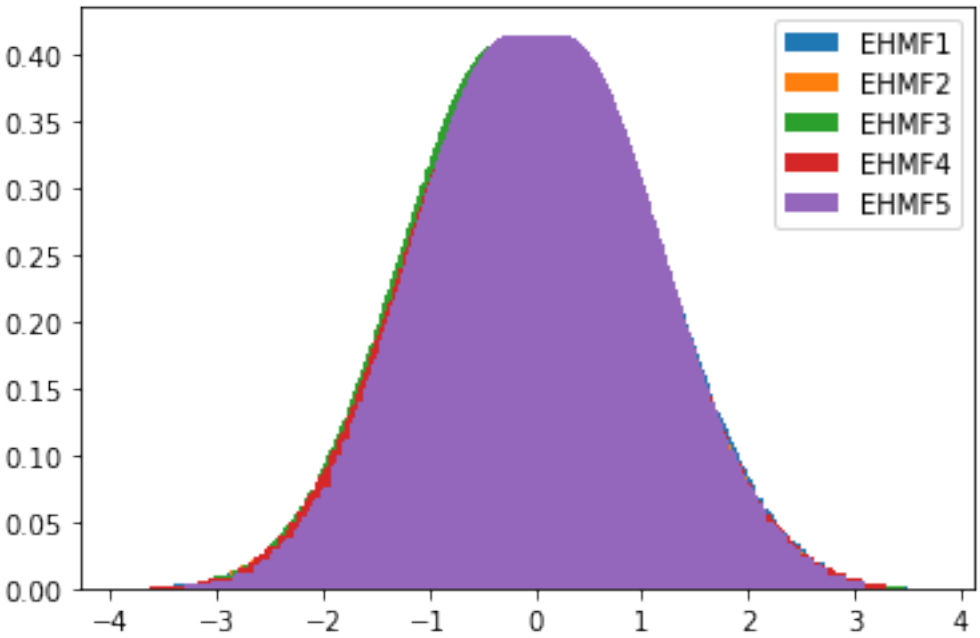
Способ построения гистограмм частот:

```
# построение гистограмм частот
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    #for i in range(5):
    #    data[i] = Counter(data[i])
    width = 0.5
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for i in range(5):
        ax.bar(data[i], stats.norm.pdf(data[i], np.mean(data[i]), np.std(data[i])), width,
               label='EHMF{}'.format(i+1))
    #ax.set_title('n={}'.format(sum(list(data[0].values()))))
    ax.legend()
    plt.show()
```


EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0.0
0.2
0.4
0.6
0.8
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0.0
0.1
0.2
0.3
0.4
0.5
−2
−1
0
1
2


EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3


EHMF1
EHMF2
EHMF3
EHMF4
EHMF5
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4

5 Домашнее задание 3. Геометрическое распределение.

5.1 Нахождение выборочного среднего и выборочной дисперсии

Выборочное среднее это:

$$\widehat{\alpha}_1 = \widehat{m}_1^* = \widehat{m}_1^*(\bar{x}) = \overline{X} = \frac{1}{n}\sum_{\infty}^{j=1} X_j = \int_{j=1}^{\infty} t d\widehat{F}_n(t)$$

Выборочная дисперсия имеет вид:

$$\mu_2 = \widehat{\mu}_2^* = \widehat{\mu}_2^*(\widehat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\alpha}_1)^2 = \int_{-\infty}^{\infty}(t - \widehat{\alpha}_1)^2 d\widehat{F}_n(t)$$

Способ нахождения выборочного среднего:

```
# нахождение выборочного среднего
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("file{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean = sum(data[m]) / len(data[m])
        print("Выборочное среднее для {} выборки при n = {}:".format(m+1, len(data[m])), sample_mean)
```

Получившиеся значения:

```
Выборочное среднее для 1 выборки при n = 5: 4.2
Выборочное среднее для 2 выборки при n = 5: 0.2
Выборочное среднее для 3 выборки при n = 5: 0.6
Выборочное среднее для 4 выборки при n = 5: 2.4
Выборочное среднее для 5 выборки при n = 5: 1.6
Выборочное среднее для 1 выборки при n = 10: 2.6
Выборочное среднее для 2 выборки при n = 10: 1.6
Выборочное среднее для 3 выборки при n = 10: 3.9
Выборочное среднее для 4 выборки при n = 10: 1.9
Выборочное среднее для 5 выборки при n = 10: 1.5
Выборочное среднее для 1 выборки при n = 100: 2.19
Выборочное среднее для 2 выборки при n = 100: 2.28
Выборочное среднее для 3 выборки при n = 100: 2.15
Выборочное среднее для 4 выборки при n = 100: 2.15
Выборочное среднее для 5 выборки при n = 100: 2.26
Выборочное среднее для 1 выборки при n = 1000: 2.336
Выборочное среднее для 2 выборки при n = 1000: 2.39
Выборочное среднее для 3 выборки при n = 1000: 2.496
Выборочное среднее для 4 выборки при n = 1000: 2.309
Выборочное среднее для 5 выборки при n = 1000: 2.224
Выборочное среднее для 1 выборки при n = 100000: 2.33808
Выборочное среднее для 2 выборки при n = 100000: 2.33922
Выборочное среднее для 3 выборки при n = 100000: 2.3294
Выборочное среднее для 4 выборки при n = 100000: 2.33341
Выборочное среднее для 5 выборки при n = 100000: 2.33618
```

Свойства выборочного среднего.

Найдем математическое ожидание выборочного среднего:

$$M_\theta \alpha_1 = M_\theta(\overline{X}) = M_\theta \frac{1}{n} \sum_{i=1}^{n} X_i = \frac{1}{n} \sum_{i=1}^{n} M_\theta(X_i) = \\ = \frac{n M_\theta(X_1)}{n} = M_\theta X_1$$

Получается, что выборочное среднее является **несмещенной** оценкой для математического ожидания нашей случайной величины.

Проверим данную оценку на состоятельность. Найдем дисперсию выборочного среднего и посмотрим,

стремится ли она к 0 (по неравенству Чебышева).

$$D\overline{X} = D\frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{n^2}\sum_{i=1}^{n} DX_i = \frac{n \cdot DX_1}{n^2} = \frac{DX_1}{n}$$

Видно, что при $n \to \infty$ дисперсия

$$D\overline{X} = \frac{DX_1}{n}$$

стремится к 0.Следовательно $\overline{X}$ является **состоятельной**.

Также следует отметить еще одно свойство: выборочное среднее при $n \to \infty$ стремится к 1-ому моменту нашей величины, то есть к MX_1. Действительно,

$$MX_1 = \frac{q}{p} = 2.3333$$

Уже при $n = 100$ выборочное среднее находится около 2, а при $n = 10^5$ совпадает почти везде до двух знаков после запятой.

Также выборочное среднее является **асимптотически нормальной** оценкой.

Способ построения выборочной дисперсии:

```
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
#  нахождение выборочных средних и запись их в массив sample_means
for k in range(5):
    sample_variance = []
    sample_mean = []
    data = []
    for i in range(len(mas)):
        with open("file{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean.append( sum(data[m]) / len(data[m]))
    for n in range(5):
        my_summ = 0
        for n1 in range(len(data[n])):
            my_summ = my_summ + (data[n][n1] - sample_mean[n])**2
        my_summ = my_summ / len(data[n])
        sample_variance.append(my_summ)
    for k in range(len(sample_variance)):
        print("Для {} выборки при n = {}:".format(k+1, len(data[1])),
              sample_variance[k])
```

Получившиеся значения:

```
Для 1 выборки при n = 5: 22.56
Для 2 выборки при n = 5: 0.16000000000000003
Для 3 выборки при n = 5: 0.6399999999999999
Для 4 выборки при n = 5: 4.24
Для 5 выборки при n = 5: 1.8400000000000003
Для 1 выборки при n = 10: 4.84
Для 2 выборки при n = 10: 4.24
Для 3 выборки при n = 10: 9.689999999999998
Для 4 выборки при n = 10: 2.8899999999999997
Для 5 выборки при n = 10: 3.25
Для 1 выборки при n = 100: 8.353900000000019
Для 2 выборки при n = 100: 6.641599999999995
Для 3 выборки при n = 100: 6.007499999999997
Для 4 выборки при n = 100: 7.087499999999989
Для 5 выборки при n = 100: 11.832400000000009
Для 1 выборки при n = 1000: 8.051103999999942
Для 2 выборки при n = 1000: 7.3898999999999635
Для 3 выборки при n = 1000: 9.121984000000017
Для 4 выборки при n = 1000: 6.727519000000014
Для 5 выборки при n = 1000: 7.281824000000039
Для 1 выборки при n = 100000: 7.791401913602224
Для 2 выборки при n = 100000: 7.75286979159975
Для 3 выборки при n = 100000: 7.717455640006899
Для 4 выборки при n = 100000: 7.824847771901656
Для 5 выборки при n = 100000: 7.760743007603177
```

Свойства выборочной дисперсии.

Мат. ожидание выборочной дисперсии имеет вид:

$$D\overline{X} = \frac{n-1}{n}\mu_2$$

Заметим, что при $n \to \infty$ значение выборочной дисперсии сходится к дисперсии нашей случайной величины.

Дисперсия нашей случайной величины, которая имеет геометрическое распределение равна:

$$D\xi = \frac{q}{p^2} = \frac{0.7}{0.3^2} = 7.777777$$

при $n = 10^5$ это выполняется.
Также выборочная дисперсия является **смещенной** оценкой теоретической дисперсии.

5.2 Нахождение параметров распределений событий.

Параметризуем:

$$p(\xi = k) = \theta(1-\theta)^{x_i}$$

Найдем оценку **методом моментов**:

$$M\xi = \overline{X}$$

Тогда

$$\frac{1-\theta}{\theta} = \overline{X}$$

$$\theta = \frac{1}{1+\overline{X}}$$

Данная оценка является **состоятельной**, так как $\theta = \frac{1}{1+\overline{X}}$ - непрерывная функция. Найдем оценку **методом максимального правдоподобия**:

$$L(\bar{x}, \theta) = \prod_{i=1}^{n} \theta(1-\theta)^{x_i} = \theta^n \cdot (1-\theta)^{\sum\limits_{i=1}^{n} x_i}$$

Тогда

$$\ln L(\bar{x}, \theta) = n \cdot \ln\theta + \sum_{i=1}^{n} x_i \ln(1-\theta)$$

Соответственно:

$$\frac{\partial \ln L(\bar{x},\theta)}{\partial\theta} = \frac{n}{\theta} - \frac{\sum\limits_{i=1}^{n} x_i}{1-\theta} = \frac{n(1-\theta) - \theta\sum\limits_{i=1}^{n} x_i}{\theta(1-\theta)} =$$

$$= \frac{\frac{1}{n}\sum\limits_{i=1}^{n} x_i - \frac{1-\theta}{\theta}}{\theta(\theta-1)\frac{1}{n}} = 0$$

Тогда

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

Упростим:

$$\theta = \frac{1}{1+\overline{\overline{X}}}$$

Получается, что оценки, найденные методом моментов и методом максимального правдоподобия, совпадают. Посмотрим, смещенная это оценка или несмещенная. Если оценка несмещенная, то математическое ожидание оценки при $\forall n$ будет равняться параметру. Рассмотрим случай $n = 1$ и найдем мат.

ожидание:

$$M\left(\frac{1}{1+X_1}\right) = \sum_{X_1=0}^{\infty} \frac{1}{1+X_1} \cdot \theta(1-\theta)^{X_1} =$$

$$= \frac{1}{1+0} \cdot \theta(1-\theta)^0 + \sum_{X_1=1}^{\infty} \frac{1}{1+X_1} \cdot \theta(1-\theta)^{X_1} =$$

$$= \theta + \sum_{X_1=1}^{\infty} \frac{1}{1+X_1} \cdot \theta(1-\theta)^{X_1} > \theta$$

Получается, что при $n = 1$ мат. ожидание оценки не совпало с параметром, следовательно, оценка является **смещенной**.

Так как оценка смещенная, то она **не является оптимальной и эффективной.** Докажем, что оценка $\theta = \frac{1}{1+\overline{X}}$ является достаточной.

По **критерию факторизации**:

$$L(\bar{x}, \theta) = \theta^n \cdot (1-\theta)^{\sum\limits_{i=1}^{n} x_i} = \theta^n \cdot (1-\theta)^{n \cdot \frac{1}{n} \sum\limits_{i=1}^{n} x_i} =$$

$$= \theta^n \cdot (1-\theta)^{n \cdot (\frac{1}{n} \sum\limits_{i=1}^{n} x_i + 1 - 1)} = \theta^n (1-\theta)^{n \cdot (\overline{X} + 1 - 1)} =$$

$$= \theta^n (1-\theta)^{n \cdot \left(\frac{1}{\frac{1}{\overline{X}+1}} - 1\right)} = \theta^n (1-\theta)^{n \cdot \left(\frac{1}{T(X)} - 1\right)} = g(T(X), \theta) \cdot h(x)$$

Получается, что $g(T(X), \theta) = \theta^n (1-\theta)^{n \cdot \left(\frac{1}{T(X)} - 1\right)}$, а $h(x) = 1$. Тогда данная статистика является **достаточной**. Так как оценка $T(X) = \frac{1}{1+\overline{X}}$ является со-

стоятельной, то данная оценка при $n \to \infty$ по P стремится к оцениваемому ею параметру, то есть к θ. Проверим, так ли это. Код:

```
#сравнение истинного значения и оценки
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("file{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean = sum(data[m]) / len(data[m])
        print("параметр theta  = 0.3,", 'разница: ',abs(0.3- 1 / (1 + sample_mean)),
              'при n = {}'.format(mas[k]))
```

Получившиеся значения:

```
параметр theta  = 0.3, разница:  0.1076923076923077 при n = 5
параметр theta  = 0.3, разница:  0.5333333333333334 при n = 5
параметр theta  = 0.3, разница:  0.325 при n = 5
параметр theta  = 0.3, разница:  0.00588235294117645 при n = 5
параметр theta  = 0.3, разница:  0.08461538461538459 при n = 5
параметр theta  = 0.3, разница:  0.2222222222222222 при n = 10
параметр theta  = 0.3, разница:  0.08461538461538459 при n = 10
параметр theta  = 0.3, разница:  0.09591836734693879 при n = 10
параметр theta  = 0.3, разница:  0.044827586206896586 при n = 10
параметр theta  = 0.3, разница:  0.10000000000000003 при n = 10
параметр theta  = 0.3, разница:  0.013479623824451448 при n = 100
параметр theta  = 0.3, разница:  0.004878048780487809 при n = 100
параметр theta  = 0.3, разница:  0.017460317460317454 при n = 100
параметр theta  = 0.3, разница:  0.017460317460317454 при n = 100
параметр theta  = 0.3, разница:  0.006748466257668728 при n = 100
параметр theta  = 0.3, разница:  0.00023980815347718343 при n = 1000
параметр theta  = 0.3, разница:  0.005014749262536866 при n = 1000
параметр theta  = 0.3, разница:  0.013958810068649852 при n = 1000
параметр theta  = 0.3, разница:  0.002206104563312161 при n = 1000
параметр theta  = 0.3, разница:  0.01017369727047146 при n = 1000
параметр theta  = 0.3, разница:  0.00042659253223409355 при n = 100000
параметр theta  = 0.3, разница:  0.000528866022604102 при n = 100000
параметр theta  = 0.3, разница:  0.0003544182134919138 при n = 100000
параметр theta  = 0.3, разница:  6.899841303642518e-06 при n = 100000
параметр theta  = 0.3, разница:  0.00025598139189131 34 при n = 100000
```

5.3 Работа с данными

. Данные для моего геометрического распределения были взяты со следующих сайтов (на одном сайте может быть несколько форумов, тем, обсуждений, поэтому некоторые ссылки могут указывать именно на сайт.)
Ссылки:

- http://www.probirka.org/forum/viewtopic.php?f=17&t=20136
- https://eva.ru/kids/messages-2665663.htm

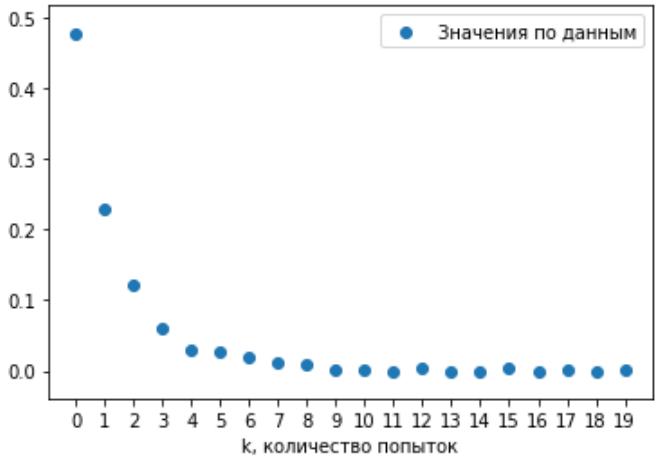
- https://formama.online/planirovanie-beremennosti/eko-zachatie-rebenka/
- https://altravita-ivf.ru/eko/rezultativnost-i-statisti
  html
- https://myzachatie.ru/vrt/statistika-eko.html
- https://www.7ya.ru/article/Beremennost-pri-pomowi-JeK0
- https://detkam.su/forum/33-884-1
- https://www.u-mama.ru/forum/waiting-baby/want-baby/41293/2.html
- https://forum.sibmama.ru/viewtopic.php?t=43209
- https://sovet.kidstaff.com.ua/question-1144504
- https://deti.mail.ru/forum/v_ozhidanii_chuda/planirovanie_beremennosti/jeko_s_kakogo_raza_poluchilos/
- https://forum.say7.info/topic7278.html
- https://beremennost.net/forum/showthread.php?3413-kto-zaberemenel-posle-eko/page2
- https://touch.otvet.mail.ru/question/193407436
- https://forum.2mm.ru/viewtopic.php?t=16208
- http://www.detkityumen.ru/forum/thread/18897/

- https://www.nn.ru/community/user/be_mother/sobirayu_informatsiyu_kto_delal_eko_rodil_i_poshel_eshche_raz_na_eko_za_vtorym_malyshem.html
- http://www.my-bt.ru/talk/post9343.html
- http://38mama.ru/forum/index.php?topic=301514.50
- https://conf.7ya.ru/fulltext-thread.aspx?cnf=Planning&trd=43883
- http://eka-mama.ru/forum/part277/topic18075/
- http://www.woman.ru/health/Pregnancy/thread/3829077/
- https://www.babyblog.ru/community/post/sterility/3044829
- https://www.babyplan.ru/questions/141304-posle-kakoj-p

Файл, в котором находятся данные: код/python/pregnant.txt.
Количество данных: n = 1336.
Построим график из этих данных. Код:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('pregnant.txt')
mas_prob = []
data = data - 1
for i in range(20):
    mas_prob.append((data == i).sum() / len(data))
print(mas_prob)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(list(range(20)), mas_prob, label='Значения по данным')
#ax.scatter(list(range(25)), mas, color='orange', label='Значения функции вероятности')
ax.set_xlabel('k, количество попыток')
ax.set_xticks(list(range(20)))
ax.legend()
plt.show()
```

Результат:



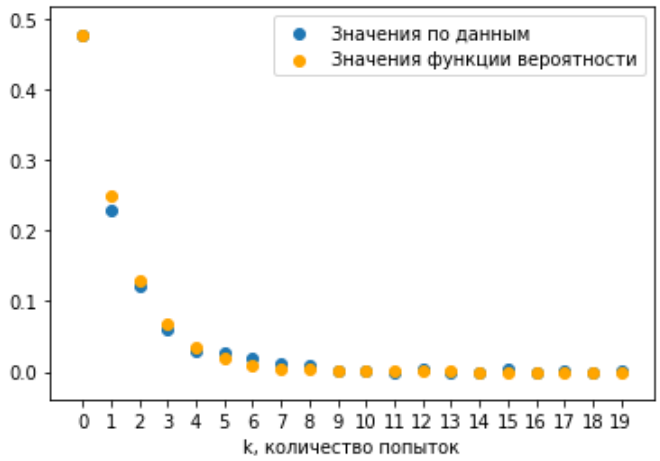
Также ниже представлена частота появления величин:
{0.4782934131736527, 0.22904191616766467, 0.12125748502994012, 0.05913173652694611, 0.030688622754491017, 0.02694610778443114, 0.020209580838323353, 0.010479041916167664, 0.009730538922155689, 0.002245508982035928, 0.002245508982035928, 0.0, 0.004491017964071, 0.0, 0.0, 0.0029940119760479044, 0.0, 0.0014970059880239522, 0.0, 0.0007485029940119761}
Как видно, вероятность того, что получится забере-

менить с первого раза (то есть не будет неудач) равна **0.4782934131736527**. Интересный факт. Многие источники(сайты, врачи) утверждают, что вероятность забеременеть лежит между 0.4 и 0.6. То есть наше получившееся значение **укладывается** в общепринятые ожидания от данной процедуры. Теперь возьмем эту вероятность как за вероятность успеха, и построим выборку, имеющую геометрическое распределение с $p = 0.4782934131736527$ и $q = 1 - 0.4782934131736527$. Сравним с нашим графиком по данным.

Код:

```
#сравнение
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('pregnant.txt')
mas_prob = []
data = data - 1
for i in range(20):
    mas_prob.append((data == i).sum() / len(data))

def my_geom(n):
    p = 0.4782934131736527
    q = 1 - 0.4782934131736527
    mas = []
    for i in range(n):
        mas.append(p * q **i)
    return mas

mas = my_geom(20)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(list(range(20)), mas_prob, label='Значения по данным')
ax.scatter(list(range(20)), mas, color='orange', label='Значения функции вероятности')
ax.set_xlabel('k, количество попыток')
ax.set_xticks(list(range(20)))
ax.legend()
plt.show()
```

Получившийся график:



Как видно, разница совсем небольшая.
Найдем **выборочное среднее.** Код:

```
# нахождение выборочного среднего:
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('pregnant.txt')
data = data - 1
print('выборочное среднее:', sum(data)/ len(data), )
print('мат. ожидание: ',(1 - 0.4782934131736527) /0.4782934131736527 )
```

Выборочное среднее и мат ожидание, к которому выборочное среднее должно стремиться:

```
выборочное среднее: 1.3600299401197604
мат. ожидание:  1.0907668231611893
```

Значения отличаются, но это объяснимо довольно малым $n = 1336$. Найдем выборочную дисперсию:

```python
# нахождение выборочной дисперсии
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('pregnant.txt')
data = data - 1
sample_mean = sum(data) / len(data)
data = data - sample_mean
data = data**2
print('выборочная дисперсия:', sum(data) / len(data))
```

```
выборочная дисперсия: 4.854659879343125
```

Найдем дисперсию:
Посчитаем выборочную дисперсию, которая должна получится при геометрическом распределении с вероятностью успеха $p = 0.4782934131736527$ и вероятностью неудачи $q = 1 - 0.4782934131736527$

$$\frac{q}{p^2} = 2.2805390856703425$$

Разница между выборочной дисперсией и дисперсий большая, что, возможно, объясняется малым n.
Найдем значение оценки:

```python
# нахождение значения оценки
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt('pregnant.txt')
data = data - 1
sample_mean = sum(data) / len(data)
print('значение оценки: ',1 / (1 + sample_mean))
```

```
значение оценки:  0.4237234379955598
```

Значение оценки отличается от оцениваемого параметра ($p = 0.4782934131736527$) на 0.06, что можно назвать хорошим результатом.

6 Домашнее задание 3. Нормальное распределение

6.1 Нахождение выборочного среднего и выборочной дисперсии.

Способ нахождения выборочного среднего : (такой же, как и в геометрическом распределении.)

```
# нахождение выборочного среднего
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean = sum(data[m]) / len(data[m])
        print("Выборочное среднее для {} выборки при n = {}:".format(m+1, len(data[m])),
              sample_mean)
```

Получившиеся значения:

```
Выборочное среднее для 1 выборки при n = 5: -0.2708405578252721
Выборочное среднее для 2 выборки при n = 5: -0.6377889974206727
Выборочное среднее для 3 выборки при n = 5: -0.2743506705260973
Выборочное среднее для 4 выборки при n = 5: 0.755177080247742
Выборочное среднее для 5 выборки при n = 5: 0.042723183663180266
Выборочное среднее для 1 выборки при n = 10: -0.1237823054190523
Выборочное среднее для 2 выборки при n = 10: -0.2632280742355674
Выборочное среднее для 3 выборки при n = 10: -0.09190958460225529
Выборочное среднее для 4 выборки при n = 10: 0.45172116921611344
Выборочное среднее для 5 выборки при n = 10: 0.04402797401851383
Выборочное среднее для 1 выборки при n = 100: -0.008602079524551037
Выборочное среднее для 2 выборки при n = 100: -0.11789027740965569
Выборочное среднее для 3 выборки при n = 100: -0.15479204315871456
Выборочное среднее для 4 выборки при n = 100: 0.24174822544426394
Выборочное среднее для 5 выборки при n = 100: 0.045750024689120955
Выборочное среднее для 1 выборки при n = 1000: 0.0032225205413249484
Выборочное среднее для 2 выборки при n = 1000: -0.026637175287282818
Выборочное среднее для 3 выборки при n = 1000: -0.06858668205861544
Выборочное среднее для 4 выборки при n = 1000: -0.02020407377527684
Выборочное среднее для 5 выборки при n = 1000: 0.011448469792738086
Выборочное среднее для 1 выборки при n = 100000: 0.003708552459099596
Выборочное среднее для 2 выборки при n = 100000: -0.0011015237968186836
Выборочное среднее для 3 выборки при n = 100000: -0.0021705407740259274
Выборочное среднее для 4 выборки при n = 100000: 0.001824486961231062
Выборочное среднее для 5 выборки при n = 100000: -0.0071960611131343735
```

Свойства, которыми обладает выорочное среднее, были расписаны в разделе про среднее выборочное геометрического распределения. Кратко:

- Выборочное среднее - это несмещенная оценка для математического ожидания.
- Выборочное среднее является состоятельной оценкой.
- Выборочное среднее при $n \to \infty$ стремится к 1-ому моменту нашей случайной величины, то есть к MX_1. Действительно, мат. ожидание случайной величины, имеющей стандартное нормальное распределение есть $MX_1 = \mu = 0$. Исходя из

полученных значений это выполняется.

- Выборочное среднее является асимптотически нормальной оценкой, то есть

Способ нахождения выборочной дисперсии:

```
# нахождение выборочной дисперсии
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
#  нахождение выборочных средних и запись их в массив sample_means
for k in range(5):
    sample_variance = []
    sample_mean = []
    data = []
    for i in range(len(mas)):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean.append( sum(data[m]) / len(data[m]))
    for n in range(5):
        my_summ = 0
        for n1 in range(len(data[n])):
            my_summ = my_summ + (data[n][n1] - sample_mean[n])**2
        my_summ = my_summ / len(data[n])
        sample_variance.append(my_summ)
    for k in range(len(sample_variance)):
        print("Для {} выборки при n = {}:".format(k+1, len(data[1])),
              sample_variance[k])
```

Полученные значения:

```
Для 1 выборки при n = 5: 1.6229238495478202
Для 2 выборки при n = 5: 0.5436951288003978
Для 3 выборки при n = 5: 1.2528516633328497
Для 4 выборки при n = 5: 0.1940371845120735
Для 5 выборки при n = 5: 0.538074044717748
Для 1 выборки при n = 10: 0.7645423815066699
Для 2 выборки при n = 10: 1.0628396887856302
Для 3 выборки при n = 10: 1.2458522731469277
Для 4 выборки при n = 10: 0.6088129087854264
Для 5 выборки при n = 10: 0.6308402024247972
Для 1 выборки при n = 100: 0.9685855080959697
Для 2 выборки при n = 100: 1.0017977333772685
Для 3 выборки при n = 100: 1.2808710400896026
Для 4 выборки при n = 100: 1.0872740702926316
Для 5 выборки при n = 100: 0.9752242061118053
Для 1 выборки при n = 1000: 0.990550326390373
Для 2 выборки при n = 1000: 0.9829726165711664
Для 3 выборки при n = 1000: 0.9507956791408639
Для 4 выборки при n = 1000: 0.991346861969709
Для 5 выборки при n = 1000: 0.9263292215864997
Для 1 выборки при n = 100000: 1.0025223449606422
Для 2 выборки при n = 100000: 1.0004955612913855
Для 3 выборки при n = 100000: 1.0009415119807175
Для 4 выборки при n = 100000: 1.0020259091587813
Для 5 выборки при n = 100000: 1.001320056786746
```

Свойства выборочной дисперсии:

- при $n \to \infty$ значение выборочной дисперсии сходится к дисперсии случайной величины. Дисперсия случайной величины, имеющей стандартное нормальное распределение равна $DX_1 = \sigma^2 = 1$. Полученные значения подтверждают это.
- Выборочная дисперсия является смещенной оценкой теоретической дисперсии.

6.2 Нахождение параметров распределений событий.

6.2.1 Оценим параметр θ_1 в распределении $N(\theta_1, \sigma^2)$

Параметризуем плотность нормального распределения:

$$f(\theta_1,\sigma^2) = \frac{1}{\sigma\sqrt{2\Pi}} \cdot exp\left(-\frac{(x-\theta_1)^2}{2\sigma^2}\right)$$

Найдем оценку сначала **методом моментов**.Так как у нас два параметра, то нам нужные первые два момента (далее я обозначаю случайную величину как X_1, так как все X_i независимые одинаково распределенные случайные величиныи и без разницы, какую X_i взять):

$$M_\theta X_1 = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Получается, что:

$$\theta_1 = \overline{X}$$

Данная оценка является **состоятельной**, так как $\overline{X}$ - непрерывная функция. Проверим, что эта оценка является состоятельной(дисперсия этой оценки должна стремиться к 0 при $n \to \infty$):

$$D(\overline{X}) = D(\frac{1}{n}\sum_{i=1}^{n} X_i) = \frac{1}{n^2}\sum_{i=1}^{n} DX_i = \frac{DX_i}{n}$$

Видно, что при $n \to \infty$ дисперсия оценки стремится к 0.

Теперь найдем оценку **методом максимального правдоподобия.**

$$L(\bar{x},\theta_1)=\prod_{i=1}^{n} f_i(\theta_1,\sigma^2)=\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\Pi}}\cdot\exp\left(-\frac{(x_i-\theta_1)^2}{2\sigma^2}\right)=$$
$$=\frac{1}{\sigma^n\cdot(2\Pi)^{\frac{n}{2}}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\theta_1)^2\right)$$

Прологарифмируем:

$$\ln L(\bar{x},\theta_1)\ =\ \ln\sigma^{-n}+\ln(2\Pi)^{\frac{n}{2}}-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\theta_1)^2$$

Продифференцируем:

$$\frac{\partial\ln L(\bar{x},\theta_1)}{\partial\theta_1}=2\cdot(-1)\cdot\left(-\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\theta_1)=\frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\theta_1)=0$$

$$0=\sum_{i=1}^{n}(x_i-\theta_1)=\sum_{i=1}^{n}x_i-n\cdot\theta_1$$

Тогда получается, что θ_1 имеет следующую оценку:

$$\theta_1=\frac{1}{n}\sum_{i=1}^{n}x_i=\overline{X}$$

Так как данная оценка - ОМП, то она обладает следующими свойствами:

- состоятельна

- асимптотически нормальная
- асимптотически эффективна

Найдем эффективную оценку с помощью **критерия эффективности**. Воспользуемся выкладками выше:

$$\frac{\partial \ln L(\bar{x}, \theta_1)}{\partial \theta_1} = \frac{\sum\limits_{i=1}^{n}(x_i - \theta_1)}{\sigma^2} = \frac{\sum\limits_{i=1}^{n} x_i - n \cdot \theta_1}{\sigma^2} = \frac{\frac{1}{n}\sum\limits_{i=1}^{n} x_i - \theta_1}{\sigma^2} = \\ = \frac{T(\bar{x}) - \tau(\theta_1)}{a(\theta_1)}$$

Тогда получается, что $T(\bar{x}) = \frac{1}{n}\sum\limits_{i=1}^{n} x_i$ является эффективной оценкой для параметра θ_1. Заметим, что эффективная оценка совпала с ОМП и с оценкой, найденной методом моментов и является выборочным средним. Эффективная оценка обладает следующими свойствами: Тогда данная оценка обладает следующими свойствами:

- несмещенная (как выборочное среднее)
- состоятельная (как ОМП)
- асимптотически нормальная (как ОМП)
- асимптотически эффективная (как ОМП)
- достаточная (как эффективная)
- оптимальная (как эффективная)

- эффективная

Так как оценка $T(X) = \overline{X}$ состоятельная, то при $n \to \infty$ она должна стремиться к оцениваемому параметру по вероятности, то есть к μ. Сравним истинное значение и его оценку.

Код:

```
#сравнение истинного значения и оценки для мю
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
for k in range(5):
    data = []
    for i in range(len(mas)):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean = sum(data[m]) / len(data[m])
        print("Для {} выборки при n = {} разница: ".format(m+1, len(data[m])),
              abs(sample_mean - 0))
```

Выкладки:

```
Для 1 выборки при n = 5 разница:  0.2708405578252721
Для 2 выборки при n = 5 разница:  0.6377889974206727
Для 3 выборки при n = 5 разница:  0.2743506705260973
Для 4 выборки при n = 5 разница:  0.755177080247742
Для 5 выборки при n = 5 разница:  0.042723183663180266
Для 1 выборки при n = 10 разница:  0.1237823054190523
Для 2 выборки при n = 10 разница:  0.2632280742355674
Для 3 выборки при n = 10 разница:  0.09190958460225529
Для 4 выборки при n = 10 разница:  0.45172116921611344
Для 5 выборки при n = 10 разница:  0.04402797401851383
Для 1 выборки при n = 100 разница:  0.008602079524551037
Для 2 выборки при n = 100 разница:  0.11789027740965569
Для 3 выборки при n = 100 разница:  0.15479204315871456
Для 4 выборки при n = 100 разница:  0.24174822544426394
Для 5 выборки при n = 100 разница:  0.045750024689120955
Для 1 выборки при n = 1000 разница:  0.0032225205413249484
Для 2 выборки при n = 1000 разница:  0.026637175287282818
Для 3 выборки при n = 1000 разница:  0.06858668205861544
Для 4 выборки при n = 1000 разница:  0.02020407377527684
Для 5 выборки при n = 1000 разница:  0.011448469792738086
Для 1 выборки при n = 100000 разница:  0.003708552459099596
Для 2 выборки при n = 100000 разница:  0.0011015237968186836
Для 3 выборки при n = 100000 разница:  0.0021705407740259274
Для 4 выборки при n = 100000 разница:  0.001824486961231062
Для 5 выборки при n = 100000 разница:  0.0071960611131343735
```

6.2.2 Оценим параметр θ_2 в распределении $N(\mu, \theta^2)$

Параметризуем плотность нормального распределения:

$$f(\mu,\theta_2^2) = \frac{1}{\theta_2\sqrt{2\Pi}} \cdot exp\left(-\frac{(x-\mu)^2}{2\theta_2^2}\right)$$

Найдем оценку θ_2 **методом моментов**.

$$MX_1^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2$$

$$\mu^2 + \theta_2^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2$$

$$\theta_2^2 = \overline{X^2} - \mu^2$$

Так как $\overline{X^2} - \mu^2$ непрерывная, то оценка для θ_2 является состоятельной.

Проверим, что оценка является состояльной:

$$D(\overline{X^2} - \mu^2) = D(\overline{X}) = \frac{1}{n^2} D\left(\sum_{i=1}^{n} X_i^2\right) = \frac{DX_1^2}{n}$$

стремится к 0 при $n \to \infty$.

Найдем оценку **методом максимального правдоподобия.**

$$L(\bar{x}, \theta_2) = \prod_{i=1}^{n} f_i(\mu, \theta_2^2) = \prod_{i=1}^{n} \frac{1}{\theta_2\sqrt{2\Pi}} \cdot exp\left(-\frac{(x_i - \mu)^2}{2\theta_2^2}\right) =$$

$$= \frac{1}{\theta_2^n \cdot (2\Pi)^{\frac{n}{2}}} \cdot \exp\left(-\frac{1}{2\theta_2^2} \sum_{i=1}^{n} (x_i - \mu)^2\right)$$

Прологарифмируем:

$$\ln L(\bar{x}, \theta_2^2) = -n \ln \theta_2 - \frac{n}{2} \ln 2\Pi - \frac{1}{2\theta_2^2} \sum_{i=1}^{n} (x_i - \mu)^2$$

Продифференцируем:

$$\frac{\partial \ln L(\bar{x}, \theta_2^2)}{\partial \theta_2} = \frac{-n}{\theta_2} + \frac{\sum (x_i - \mu)^2}{\theta_2^3} = \frac{-n \cdot \theta_2^2 + \sum\limits_{i=1}^{n} (x_i - \mu)^2}{\theta_2^3} = 0$$

Тогда

$$\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 = \theta_2^2$$

Найдем эффективную оценку по критерию эффективности. Воспользуемся предыдущими выкладками. Как мы знаем,

$$\frac{\partial \ln L(\bar{x}, \theta_2^2)}{\partial \theta_2} = \frac{- \cdot \theta_2^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}{\theta_2^3 \cdot \frac{1}{n}} = \frac{T(\bar{x}) - \tau(\theta_2)}{a(\theta_2)}$$

Получается, что оценка $\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$ является эффективной для параметра θ_2^2. Покажем, что оценка несмещенная:

$$M\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 = \frac{1}{n}M\sum_{i=1}^{n}\left(x_i^2 + \mu^2 - 2x_i\mu\right) =$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n}Mx_i^2 + \sum_{i=1}^{n}M\mu^2 - \sum_{i=1}^{n}M2x_i\mu\right) =$$

$$= \frac{1}{n}\left(n \cdot Mx_1^2 + n \cdot \mu^2 - 2\mu \cdot nMx_1\right) = Mx_1^2 + \mu^2 - 2\mu Mx_1 =$$

$$= \mu^2 + \theta_2^2 + \mu^2 - 2\mu^2 = \theta_2^2$$

Значит, оценка является несмещенной. Заметим, что во всех 3 случаях оценки для θ_2^2 совпали. Тогда подытожим. Оценка $\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$:

- несмещенная (доказано выше)
- состоятельная (как ОМП)
- асимптотически нормальная (как ОМП)

- асимптотически эффективная (как ОМП)
- достаточная (как эффективная)
- оптимальная (как эффективная)
- эффективная

Данная оценка является состоятельной, то есть она при $n \to \infty$ стремится к оцениваемому параметру по вероятности. Проверим это. Код:

```
#сравнение истинного значения и оценки
from collections import Counter
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
mas = [5, 10, 100, 1000, 100000]
#  нахождение выборочных средних и запись их в массив sample_means
for k in range(5):
    sample_variance = []
    data = []
    for i in range(len(mas)):
        with open("filenormal{}.txt".format('{}_{}'.format(mas[k], i + 1))) as f:
            for line in f:
                data.append(list([float(x) for x in line.split()]))
    for m in range(len(data)):
        sample_mean.append( sum(data[m]) / len(data[m]))
    for n in range(5):
        my_summ = 0
        for n1 in range(len(data[n])):
            my_summ = my_summ + (data[n][n1] - 0**2)**2
        my_summ = my_summ / len(data[n])
        sample_variance.append(my_summ)
    for k in range(len(sample_variance)):
        print("Для {} выборки при n = {} разница: ".format(k+1, len(data[1])),
              abs(sample_variance[k]-1))
```

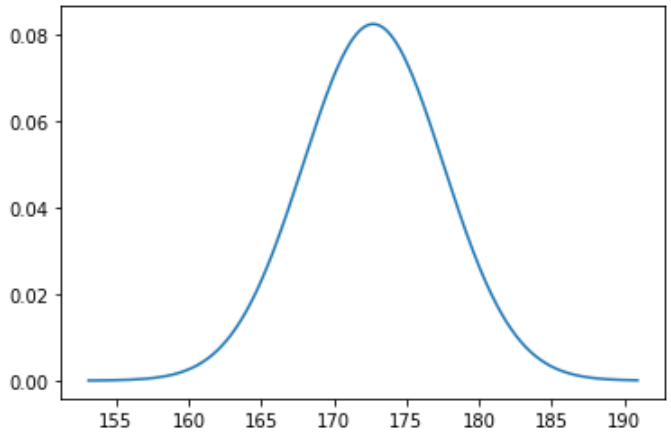
Выкладки:

```
Для 1 выборки при n = 5 разница:  0.6962784573109251
Для 2 выборки при n = 5 разница:  0.049530065968735215
Для 3 выборки при n = 5 разница:  0.3281199537509685
Для 4 выборки при n = 5 разница:  0.23567039295642178
Для 5 выборки при n = 5 разница:  0.46010068485993405
Для 1 выборки при n = 10 разница:  0.22013555935847462
Для 2 выборки при n = 10 разница:  0.13212870785139552
Для 3 выборки при n = 10 разница:  0.2542996448886865
Для 4 выборки при n = 10 разница:  0.18713507649660122
Для 5 выборки при n = 10 разница:  0.36722133507902777
Для 1 выборки при n = 100 разница:  0.031340496131883944
Для 2 выборки при n = 100 разница:  0.01569585088499381
Для 3 выборки при n = 100 разница:  0.30483161671485215
Для 4 выборки при n = 100 разница:  0.14571627479808247
Для 5 выборки при n = 100 разница:  0.02268272912913971
Для 1 выборки при n = 1000 разница:  0.009439288970986115
Для 2 выборки при n = 1000 разница:  0.016317844321548036
Для 3 выборки при n = 1000 разница:  0.04450018790332677
Для 4 выборки при n = 1000 разница:  0.008244933433172896
Для 5 выборки при n = 1000 разница:  0.07353971095290324
Для 1 выборки при n = 100000 разница:  0.002536098321990643
Для 2 выборки при n = 100000 разница:  0.0004967746460591282
Для 3 выборки при n = 100000 разница:  0.000946223227989984
Для 4 выборки при n = 100000 разница:  0.0020292379114605907
Для 5 выборки при n = 100000 разница:  0.0013718400822855248
```

6.3 Работа с данными

Нетипичная интерпретация почти не изменилась - мы также берем рост, но не игроков, а людей в возрасте 18 лет. Всего у нас есть 25 000 записей о росте реальных людей. Данные находятся в /код/Python/height.txt. Данные взяты с сайта http://socr.ucla.edu/docs/resources/SOCR_Data/SOCR_Data_Dinov_020108_HeightsWeights.html Построим график:

```
# построим график на основе данных
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
c.sort()
c = np.array(c)
# перевод в метрическую систему
c = c * 2.54
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(c, stats.norm.pdf(c, np.mean(c), np.std(c)))
plt.show()
```



Найдем мат. ожидание и дисперсию:

```
# найдем мат ожидание и дисперсию
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
c.sort()
c = np.array(c)
c = c * 2.54
print('мат. ожидание: ', np.mean(c))
print('дисперсия: ', np.std(c)**2)
print('сигма: ', np.std(c))
```

```
мат. ожидание:  172.70250853587203
дисперсия:  23.33051781350413
сигма:  4.830167472614602
```

Найдем выборочное среднее и выборочную дисперсию:

```
# найдем выборочное среднее и выборочную дисперсию:
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
c.sort()
c = np.array(c)
c = c * 2.54
sample_mean = sum(c) / len(c)
print('выборочное среднее: ', sample_mean)
print('выборочная дисперсия: ', sum((c - sample_mean)** 2) / len(c) )
```

```
выборочное среднее:  172.70250853587117
выборочная дисперсия:  23.330517813504198
```

Проверим, что рост 68-69 % людей находится между $\mu - \sigma$ и $\mu + \sigma$

```
# найдем выборочное среднее и выборочную дисперсию:
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
length = len(c)
c.sort()
c = np.array(c)
c = c * 2.54
c = c[c < (172.70250853587117 + 4.830167472614602)]
c = c[c > (172.70250853587117 - 4.830167472614602) ]
print(len(c) / length * 100)
```

```
68.356
```

Проверим, что рост 95 - 96 % людей находится между $\mu - 2\sigma$ и $\mu + 2\sigma$

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
length = len(c)
c.sort()
c = np.array(c)
c = c * 2.54
c = c[c < (172.70250853587117 + 2 *4.830167472614602)]
c = c[c > (172.70250853587117 - 2 *4.830167472614602) ]
print(len(c) / length * 100)
```

```
95.46
```

И рост почти всех людей должен находиться между $\mu - 3\sigma$ и $\mu + 3\sigma$

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
a = pd.read_csv('height.txt',sep = '\t')
c = a['Height'].tolist()
length = len(c)
c.sort()
c = np.array(c)
c = c * 2.54
c = c[c < (172.70250853587117 + 3 *4.830167472614602)]
c = c[c > (172.70250853587117 - 3 *4.830167472614602) ]
print(len(c) / length * 100)
```

```
99.79599999999999
```

7 Домашнее задание 4. Геометрическое распределение.

7.1 Критерий согласия хи-квадрат для простой гипотезы.

7.1.1 Описание применяего критерия.

Это один из самых универсальных методов проверки статистических гипотез. Этот критерий может применяться к данным **любой** природы.
Пусть $\xi_1, \xi_2 ... \xi_n$ - выборка значений наблюдаемой случайной величины, то есть произведено n независимых испытаний. Сгруппируем наблюдения. Область определения случайной величины разбивается на k непересекающихся интервалов:

$$x_0 < x_1 < x - 2 < \ldots < x_k$$

Для нахождения оптимального k есть следующие способы (на самом деле, их больше, но рассмотрим только несколько).

- $k = \log_2 n + 1 = 3.3 \lg n + 1$ по эвристической формуле Старджесса. Этот метод упоминается во многих источниках.
- $k = 5 \lg n$ по формуле Брукса и Каррузера
- $k = \sqrt{n}$
- $k =\approx 4\sqrt[5]{2}(n/t)^{0.4}$ для равновероятных интервалов, где t - квантиль стандартного нормального распределения для заданного уровня значимости
- $k = 4 \lg n$
- $k = 5 \lg n - 5$

Также существуют следующие рекомендации:

При больших объемах выборок N разброс значений k, задаваемых различными формулами, достаточно велик. Поэтому на практике при выборе числа интервалов больше руководствуются тем, чтобы в интервалы попадало число наблюдений не менее 5—10. Так, например, в рекомендациях ВНИИМ им. Д. И. Менделеева [34] в зависимости от N предлагают следующие значения k:

N	k
40—100	7—9
100—500	8—12
500—1000	10—16
1000—10000	12—22.

Известно, что оптимальное значение k зависит не только от объема выборки, но и от вида и закона распределения и от способа группирования.

Группировка осуществляется следующим образом: область определения функции разбивают на k непересекающихся интервалов граничными точкамии:

$$x_{(0)},x_{(1)},\dots x_{(k)}$$

где x_0 - нижняя грань области определения, $x_{(k)}$ - верхняя грань области определения.
Далее подсчитаем число v_i выборочных значений, попавших в i интервал и вероятность попадания в интервал:

$$p_i = F(x_{(i)}) - F(x_{(i-1)})$$

где $F(x_{(i)})$ - функция распределения. Получается, что

$$\sum_{i=1}^{k} v_i = n$$

и

$$\sum_{i=1}^{k} p_i = 1$$

Величина

$$X_k^2 = X_k^2(\overline{v}) = \sum_{i=1}^{k} \frac{(v_i - np_i)^2}{n \cdot p_i} = \sum_{i=1}^{k} \frac{v_i^2}{np_i} - n$$

является статистикой, на которой основан **критерий хи-квадрат Пирсона**
Если гипотеза H_0 справедлива (при этом разности

$\frac{v_i}{i} - p_i$ при больших n должны быть малы), то тогда и значение статистики X_n^2 не должно быть слишком большим. Тогда зададим критическую границу t_α,которая при заданном уровне значимости α должна быть выбра из условия:

$$P\left(X_k^2 > t_\alpha | H_0\right) = \alpha$$

Тогда критерий формулируется следующим образом:

$$\mathscr{X}_{1\alpha} = \left\{X_n^2 \geq t_\alpha\right\}$$

Известно, что $\mathscr{L}(X_k^2|H_0) \to \chi^2(k-1)$ при $n \to \infty$
Тогда при $n \geq 50$ и $v_i \geq 5$ можно записать, что:

$$P\left(X_k^2 > t_\alpha | H_0\right) \approx 1 - F_{k-1}(t_\alpha)$$

Следовательно

$$1 - F_{k-1}(t_\alpha) = \alpha$$

т.е

$$t(\alpha) = \chi^2_{1-\alpha,k-1}$$

Таким образом, классический критерий согласия имеет следующий вид: пусть объем выборки n и наблюдавшиеся значения вектора частот $\overline{n}$ удовлетворяют условиям $n \geq 50$, $v_i > 5$ $\forall i$. Тогда при заданном уровне значимости α:

$$\boxed{H_0 \text{ отвергается} \iff X_n^2 > \chi^2_{1-\alpha,k-1}}$$

7.1.2 Описание свойств.

- **Недостатком** метода является некоторая потеря информации, которая происходит при группировке.

- **Достоинством** является то, что при применении этого критерия нет необходимости учитывать точные значения наблюдений (бывают случаи, когда исходные данные имеют нечисловой характер). Также критерий Пирсона прост, нагляден и универсален.

- Использование асимпотически оптимального группирования максимизирует мощность критериев χ^2 при сложных и простых гипотезах.

7.1.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$

Обозначим гипотезу H_0 : наши данные имеют геометрическое распределения с известными нам параметрами. Возможно следующее разбиение:

i	значение с.в.	v_i
1	0	639
2	1	306
3	2	162
4	3	79
5	4	41
6	5	36
7	6	27
8	7	14
9	8	13
10	9	3
11	10	3
12	12	6
13	15	4
14	17	2
15	19	1

Тогда

```
# проверка простой гипотезы в критерии согласия для РЕАЛЬНЫХ ДАННЫХ
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt
p = 0.4782934131736527 # значение успеха
p1 = 0.4237234379955598 # значение оценки
data = np.loadtxt('pregnant.txt')
n = len(data)
data = np.sort(data)
data = data - 1
for k in [p,p1]:
    def geom(x):
        return (1-k)**x * k
    mas_x = []
    mas_ni = []
    a = Counter(data)
    for i in a.keys():
        mas_x.append(i)
    for i in a.values():
        mas_ni.append(i)
    sum = 0
    for i in range(len(mas_x)):
        sum = sum +  mas_ni[i]**2 / (n * geom(mas_x[i]))
    sum = sum - n
    print(sum, 'при p = {}'.format(k))
print('значения, которые принимает наша случайная величина: ',mas_x)
```

```
1376.4568138724821 при p = 0.4782934131736527
317.5574954134611 при p = 0.4237234379955598
```

$$\chi^2_{0.95,14} = 23.685$$

и

$$\chi^2_{0.99,14} = 29.141$$

В обоих случаях гипотеза H_0 отвергается.

Теперь применим данный критерий для наших сгенерированных выборок объемом $n = 10^5$.
Код:

```
# проверка простой гипотезы в критерии согласия для СМОДЕЛИРОВАННЫХ ДАННЫХ
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt
n = 10**5
def geom(x):
        return (1-0.3)**x * 0.3
for k in range(1,6):
    data = []
    data = np.loadtxt('file100000_{}.txt'.format(k))
    data = np.sort(data)
    mas_x = []
    mas_ni = []
    sum = 0
    a = Counter(data)
    for i in a.keys():
        mas_x.append(i)
    for i in a.values():
        mas_ni.append(i)
    sum = 0
    for i in range(len(mas_x)):
        sum = sum +  mas_ni[i]**2 / (n * geom(mas_x[i]))
    sum = sum - n
    print(sum, ' у {} выборки'.format(k))
    print('ст.свободы: k - 1 = {}'.format(len(mas_x) - 1))
```

Результаты:

```
38.76156576279027  у 1 выборки
ст.свободы: k - 1 = 29
47.640462554612895  у 2 выборки
ст.свободы: k - 1 = 29
37.67748948727967  у 3 выборки
ст.свободы: k - 1 = 30
34.63693856995087  у 4 выборки
ст.свободы: k - 1 = 32
34.990158120621345  у 5 выборки
ст.свободы: k - 1 = 30
```

	r	значение X_n^2	$\alpha = 0.05$	$\alpha = 0.01$
1 выборка	29	38.761565	$\chi^2_{0.95,29} = 42.557$	$\chi^2_{0.99,29} = 49.558$
2 выборка	29	47.640462	$\chi^2_{0.95,29} = 42.557$	$\chi^2_{0.99,29} = 49.558$
3 выборка	30	37.677489	$\chi^2_{0.95,30} = 43.773$	$\chi^2_{0.99,30} = 50.892$
4 выборка	32	34.636938	$\chi^2_{0.95,32} > 43.773$	$\chi^2_{0.99,32} > 50.892$
5 выборка	30	34.9901581	$\chi^2_{0.95,30} = 43.773$	$\chi^2_{0.99,30} = 50.892$

Заметим, что гипотеза отвергается только в одном случае: у 2 выборки при $\alpha = 0.05$. Во всех остальных случаях она не отвергается.

7.2 Критерий согласия хи-квадрат для сложной гипотезы.

7.2.1 Описание применяемого критерия.

Сложные гипотезы в общем случае имеют вид:

$$H_0\text{: } \overline{p} = p(\theta),\ \theta = (\theta_1, \ldots, \theta_r) \text{ при } r < k - 1$$

Таким образом, при гипотезе H_0 вероятности исходов являются некоторыми функциями от неизвестного параметра θ с множеством возможных значений Θ. Для построение критерия проверки такой гипотезы по наблюдению вектора частот $\overline{v} = (v_1, \ldots, v_k)$ для начала построим статистику:

$$X_n^2(\theta) = \sum_{i=1}^{k} \frac{(v_i - np_i(\theta))^2}{np_i(\theta)}$$

Так как данная статистика зависит от неизвестного параметра θ, то ее нельзя использовать. Заменим θ

некоторой оценкой $\tilde{\theta}_n = \tilde{\theta}_n(\overline{v})$ и получим статистику $\tilde{X}_n^2 = X_n^2(\tilde{\theta})$. Эта функция от наблюдений $\overline{v}$, следовательно ее значение можно вычислить для каждой реализации вектора $\overline{v}$. Но для рассчета данного критерия надо знать распределение $\tilde{X}_n^2$ при гипотезе H_0(хотя бы приближенно). Также теперь величины $p_i(\tilde{\theta})$не постоянные, а представлюят собой функции от наблюдений $\overline{v}$. Еще следует ожидать, что распределение $\tilde{X}_n^2$ будет зависеть от способоя построения оценки $\tilde{\theta}_n$. Распределение $\tilde{X}_n^2$ при $n \to \infty$ нашел американский статистик **Р. Фишер** в 1924 г. Он показал, что существуют методы оценивания параметра θ при которых наше предельное распределение имеет простой вид χ^2_{k-1-r}. В частности, это будет иметь место при оценке максимального правдоподобия:

$$\widehat{\theta} = \arg\max \prod_{i=1}^{k} (p_i(\theta))^{v_i}$$

В этом случае для построения критерия испольуется статистика:

$$\widehat{X}_n^2 = X_n^2(\widehat{\theta}_n) = \sum_{i=1}^{k} \frac{(v_i - np_i(\widehat{\theta}_n))^2}{np_i(\widehat{\theta}_n)}$$

На основании следующей теоремы рассчитывается критерий:

Теорема 4. Пусть функции $p_j(\theta)$, $j = 1, \quad , N$, удовлетворяют условиям:

1) $\sum_{j=1}^{N} p_j(\theta) = 1, \forall\, \theta \in \Theta$;

2) $p_j(\theta) \geqslant c > 0, \forall\, j$, и существуют непрерывные производные

$$\frac{\partial p_j(\theta)}{\partial \theta_k}, \quad \frac{\partial^2 p_j(\theta)}{\partial \theta_k \partial \theta_l}, \quad k, l = 1, \quad , r;$$

3) $(N \times r)$-матрица $\left\| \partial p_j(\theta)/\partial \theta_k \right\|$ имеет ранг r для всех $\theta \in \Theta$. Тогда

$$\mathcal{L}(\widehat{X}_n^2 | H_0) \to \chi^2(N - 1 - r),$$

где статистика $\widehat{X}_n^2$ определена в (11).

При выполнении условий **теоремы 4** (на картинке выше) критерий для сложной гипотезы H_0 имеет следующий вид:

$$\boxed{H_0 \text{ отвергается} \iff X_n^2 > \chi^2_{1-\alpha,k-1-r}}$$

7.2.2 Описание свойств.

Свойства такие же, как и для простой гипотезы.

7.2.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$

Обозначим гипотезу H_0 : наши **реальные** данные имеют геометрическое распределение с неизвестным нам параметром $\theta \in \Theta$. Найдем (еще раз) функцию максимального правдоподобия:

$$L(\theta) = \prod_{i=1}^{k} (p_i(\theta))^{v_i} = p_1^{v_1} \cdot p_2^{v_2} \cdot \ldots \cdot p_{14}^{v_{14}}$$

Прологарифмируем:

$$\ln L(\theta) = v_1 \cdot \ln\theta + v_2 \ln\theta(1-\theta) + v_3 \ln\theta(1-\theta)^2 +$$
$$+v_4 \ln\theta(1-\theta)^3 + v_5 \ln\theta(1-\theta)^4 + v_6 \ln\theta(1-\theta)^5 + v_7 \ln\theta(1-\theta)^6 +$$
$$+ v_8 \ln\theta(1-\theta)^7 + v_9 \ln\theta(1-\theta)^8 + v_{10} \ln\theta(1-\theta)^9 +$$
$$+v_{11} \ln\theta(1-\theta)^{10} + v_{12} \ln\theta(1-\theta)^{12} + v_{13} \ln\theta(1-\theta)^{15} + v_{14} \ln\theta(1-\theta)^{17} +$$
$$+ v_{15} \ln\theta(1-\theta)^{19} =$$
$$= n\cdot\ln\theta + v_2 \ln(1-\theta) + 2v_3 \ln(1-\theta) + 3v_4 \ln(1-\theta) + 4v_5 \ln(1-\theta) +$$
$$+5v_6 \ln(1-\theta) + 6v_7 \ln(1-\theta) + 7v_8 \ln(1-\theta) + 8v_9 \ln(1-\theta) + 9v_{10} \ln(1-\theta) +$$
$$+10v_{11} \ln(1-\theta) + 12v_{12} \ln(1-\theta) + 15v_{13} \ln(1-\theta) + 17v_{14} \ln(1-\theta) +$$
$$+ 19v_{15} \ln(1-\theta) = 1336 \ln\theta + 1817 \ln(1-\theta)$$

Продифференцируем:

$$\frac{\partial \ln L(\theta)}{\partial\theta} = \frac{1336}{\theta} - \frac{1817}{1-\theta} = 0$$

Следовательно:

$$1336(1-\theta) - 1817\theta = 0$$

$$\widehat{\theta} = \frac{1336}{1336 + 1817} = 0.423723$$

Мы получили ожидаемый ответ. Для этого значения параметра мы уже посчитали статистику X_n^2.(посчитали для критерия согласия для простой гипотезы) $X_n^2 = 317.5574954134611$. А $\chi^2_{0.99,13} = 27.688$ и $\chi^2_{0.95,13} = 22.362$. Следовательно сложная гипотеза H_0 отклоняется.

Проверим критерий для **смоделированных** данных для $n = 10^5$. Оценку, найденную методом максимального правдоподобия, мы уже находили.

```
# проверка сложной гипотезы хи-квадрат для СМОДЕЛИРОВАННЫХ выборок
from collections import Counter
import numpy as np
def geom(x, p):
    return (1 - p)**x * p
for i in range(1,6):
    data = np.loadtxt('file100000_{}.txt'.format(i))
    data.sort()
    n = len(data)
    mas_x = []
    mas_ni = []
    sum = 0
    a = Counter(data)
    for k in a.keys():
        mas_x.append(k)
    for k in a.values():
        mas_ni.append(k)
    est = 1 / (1 + np.mean(data))
    for j in range(len(mas_x)):
        sum = sum + mas_ni[j]**2 / (n * geom(mas_x[j], est))
    sum = sum - n
    print(sum, 'ст. свободы = {}'.format(len(mas_x) - 1 - 1))
```

```
38.14385515511094 ст. свободы = 28
46.4263772018312 ст. свободы = 28
37.79800222792255 ст. свободы = 29
34.63245646916039 ст. свободы = 31
34.708318259596126 ст. свободы = 29
```

	r	значение X_n^2	$\alpha = 0.05$	$\alpha = 0.01$
1 выборка	28	38.14385	$\chi^2_{0.95,28} = 41.337$	$\chi^2_{0.99,28} = 48.278$
2 выборка	28	46.42637	$\chi^2_{0.95,28} = 41.337$	$\chi^2_{0.99,28} = 48.278$
3 выборка	29	37.798	$\chi^2_{0.95,29} = 42.557$	$\chi^2_{0.99,29} = 49.558$
4 выборка	31	34.63245	$\chi^2_{0.95,31} > 43.773$	$\chi^2_{0.99,31} > 50.892$
5 выборка	29	34.70831	$\chi^2_{0.95,29} = 42.557$	$\chi^2_{0.99,29} = 49.558$

Заметим, что отвергается гипотеза все также на 2 выборке при $\alpha = 0.05$.

8 Домашнее задание 4. Нормальное распределение

8.1 Критерий Колмогорова для простой гипотезы

8.1.1 Описание критерия

Данный критерий применяется, если $F(x)$ - фукнция распределения - непрерывная.Статистика критерия определяется следующей формулой:

$$D_n = D_n(\overline{X}) = \sup_{-\infty<x<\infty} |\widehat{F}_n(x) - F(x)|$$

то есть представляет собой максимальное отклонение эмпирической функции распределения $\widehat{F}_n(x)$ от функции распределения $F(x)$. Разумность данной оценки для проверки гипотезы H_0 следует из известных нам свойств э.ф.р. $\widehat{F}_n(x)$. Мы знаем, что при $n \to \infty$ $\widehat{F}_n(x)$ стремится к $F(x)$ (если гипотеза H_0 справедлива). Следовательно при истинности гипотезы H_0 значение D_n не должно существенно отклоняться от 0. Получается, что критическую область критерия, основанного на статистике $T = D_n$, следует задавать в виде:

$$\mathscr{X}_{1\alpha} = \{t \geq t_\alpha\}$$

Значит, большие D_n нужно интерпретировать как свидетельство против проверямоей гипотезы H_0. Критическая граница t_α при заданном уровне значимости α рассчитывается при этом на основании теоремы Колмогорова(https://ru.wikipedia.org/wiki/

%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%9A%D0%BE%D0%BB%D0%BC%D0%BE%D0%B3%D0%BE%D1%80%D0%BE%D0%B2%D0%B0)

Если n достаточно большое($n \geq 20$), то, обозначив $t_\alpha \approx \frac{\lambda_\alpha}{\sqrt{n}}$, где $K(\lambda_\alpha) = 1 - \alpha$,получим:

$$P\left\{D_n \in \mathscr{X}_{1\alpha} | H_0\right\} = P\left\{\sqrt{n}D_n \geq | H_0\right\} \approx 1 - K(\lambda_\alpha) = \alpha$$

Сформулируем критерий Колмогорова: если $n \geq 20$ и при выбранном уровне значимости α число λ_α определено соотношением $K(\lambda_\alpha) = 1 - \alpha$, то

$$\boxed{H_0 \text{ отвергается} \iff \sqrt{n}D_n \geq \lambda_\alpha}$$

Также есть эквивалентная формула для практических рассчетов $D_n = \max(D_n^+, D_n^-)$, где

$$D_n^+ = \max_{1 \leq i \leq n} \left(\frac{i}{n} - F(X_{(i)}) \right)$$

и

$$D_n^- = \max_{1 \leq i \leq n} \left(F(X_{(i)}) - \frac{i-1}{n} \right)$$

Также мы можем использовать не $\sqrt{n}D_n$, а статистику с поправкой Большева, которая сходится к закону распределения Колмогорова существенно быстрее:

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

8.1.2 Описание свойств.

8.1.3 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$.

Гипотеза H_0 состоит в том, что наши данные имеют нормальное распределение с параметрами $\mu = 172.70250853587203$ и $\sigma = 4.830167472614602$.

Проверим ее для **реальных** данных. Найдем nD_n через формулу максимальной разности э.ф.р. и ф.р.

```
# Критерий Колмогорова. Найдем D_n через максимальную разность э.ф.р. и ф.р. для РЕАЛЬНЫХ ДАННЫХ
import numpy as np
import pandas as pd
import scipy.integrate as integrate
data = np.loadtxt('height.txt',usecols=1,skiprows=1)
n = len(data)
data.sort()
data = data * 2.54
mu = np.mean(data)
sigma = np.std(data)
smallest_height = data[0]
edf = [] # значения эмпирической функции
cdf = [] # значения функции распределения
for i in range(len(data)):
    edf.append((data < data[i]).sum() / n)
for i in range(len(data)):
    result = integrate.quad(lambda x: 1 / ((2*np.pi)**(0.5) * sigma) *
                            np.e ** (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                            smallest_height, data[i])
    cdf.append(result[0])
max = -1000000
for i in range(len(edf)):
    value = abs(edf[i] - cdf[i])
    if value > max:
        max = value
print( n **(0.5) * max)
```

```
0.46928220729459763
```

По таблице узнаем: $\lambda_{0.05} = 1.3581$ и $\lambda_{0.01} = 1.2238$. Следовательно, гипотеза H_0 для реальных данных не отвергается.

Проделаем все это для **смоделированных** данных($n = 10^5$).

```
for k in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(k))
    n = len(data)
    data.sort()
    data = data * 2.54
    mu = np.mean(data)
    sigma = np.std(data)
    smallest_height = data[0]
    edf = [] # значения эмпирической функции
    cdf = [] # значения функции распределения
    for i in range(len(data)):
        edf.append((data < data[i]).sum() / n)
    for i in range(len(data)):
        result = integrate.quad(lambda x: 1 / ((2*np.pi)**(0.5) * sigma) *
                                np.e ** (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                                smallest_height, data[i])
        cdf.append(result[0])
    max = -1000000
    for i in range(len(edf)):
        value = abs(edf[i] - cdf[i])
        if value > max:
            max = value
    print( n **(0.5) * max)
```

```
0.658271890777777
0.5373154839020897
0.7793021994739959
0.412382220042106
0.5771031031313629
```

Получается, что для всех 5 выборок гипотеза H_0 не отвергается.

8.2 Критерий Колмогорова для сложной гипотезы.

8.2.1 Описание критерия.

Гипотеза H_0 состоит в том, что мы знаем закон распределения, но не знаем точных параметров. Методика проверки критерия Колмогорова для сложной гипотезы практически совпадает с методикой проверки для простой гипотезы. Значение параметра $\widehat{\theta}$ находим методом максимального правдоподобия. И

статистика D_n имеет вид:

$$\widehat{D}_n = \sup_{-\infty \leq x \leq \infty} |\widehat{F}_n(x) - F(x, \widehat{\theta})|$$

8.2.2 Уровень значимости $\alpha = 0.05$ и $\alpha = 0.01$

Проверим гипотезу H_0 для **реальных** данных. В критерии Колмогорова для простой гипотезы мы использовали μ и σ (значения этих параметров - все что нам известно о параметрах данной выборки), найдены с помощью функций np.mean() и np.std(), которые имеют формулу выборочного среднего и выборочной дисперсии соотвественно. То есть оценки, найденные методом максимального правдоподобия будут полностью совпадать с этими параметрами, для которых уже проверялась простая гипотеза в критерии Колмогорова. Тогда бессмысленно повторяться, ведь результат будет таким же.
Проверим критерий для **смоделированных** данных ($n = 10^5$).Гипотеза H_0 состоит в том, что нам неизвестен параметр μ, а $\sigma = 1$.
Код:

```python
#Критерий Колмогорова для сложной гипотезы для СМОДЕЛИРОВАННЫХ ДАННЫХ.
#пусть нам неизвестен первый параметр ти = theta_1
import numpy as np
import scipy.integrate as integrate
sigma = 1
for i in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(i))
    data.sort()
    n = len(data)
    theta_1 = data.sum() / n
    smallest_height = data[0]
    edf = [] # значения эмпирической функции
    cdf = [] # значения функции распределения
    for i in range(len(data)):
        edf.append((data < data[i]).sum() / n)
    for i in range(len(data)):
        result = integrate.quad(lambda x: 1 / ((2*np.pi)**(0.5) * sigma) *
                                np.e ** (- 1 / 2 * (x - theta_1)**2 / (sigma)**2),
                                smallest_height, data[i])
        cdf.append(result[0])
    max = -1000000
    for i in range(len(edf)):
        value = abs(edf[i] - cdf[i])
        if value > max:
            max = value
    print( n **(0.5) * max)
```

Результаты:

```
0.7288624279086076
0.5333310109537133
0.7476982030097858
0.4731003600602837
0.5703889255472361
```

Таким образом, для всех выборок при обоих значениях α гипотеза h_0 не отвергается. Теперь проверим гипотезу H_0, которая заключается в том, что неизвестен параметр $\sigma = \theta_2$, а $mu = 0$. Код:

```python
#Критерий Колмогорова для сложной гипотезы для СМОДЕЛИРОВАННЫХ ДАННЫХ.
#пусть нам неизвестен первый параметр sigma = theta_2
import numpy as np
import scipy.integrate as integrate
mu = 0
for i in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(i))
    data.sort()
    n = len(data)
    theta_2 = ((data - mu)**2).sum() / n
    smallest_height = data[0]
    edf = [] # значения эмпирической функции
    cdf = [] # значения функции распределения
    for i in range(len(data)):
        edf.append((data < data[i]).sum() / n)
    for i in range(len(data)):
        result = integrate.quad(lambda x: 1 / ((2*np.pi)**(0.5) * theta_2) *
                                np.e ** (- 1 / 2 * (x - mu)**2 / (theta_2)**2),
                                smallest_height, data[i])
        cdf.append(result[0])
    max = -1000000
    for i in range(len(edf)):
        value = abs(edf[i] - cdf[i])
        if value > max:
            max = value
    print( n **(0.5) * max)
```

Результат:

```
0.9979061032329927
0.6790589912897645
1.0310447698676213
0.588198296118071
1.166013014182024
```

8.3 Критерий согласия хи-квадрат для простой гипотезы

Проверим критерий для **реальных данных**. Гипотеза H_0 заключается в том, что нам известно распределение и параметры. Число разбиений равно

$$k = \log_2 n + 1 = \log_2 25000 + 1 = 15.6$$

Результат:

```
# критерий хи-квадрат для простой гипотезы для РЕАЛЬНЫХ ДАННЫХ
import numpy as np
import math
import scipy.integrate as integrate
data = np.loadtxt('height.txt',usecols=1,skiprows=1)
data = data *2.54
data.sort()
n = len(data)
k = int(math.log2(n) + 1) # число интервалов
a = np.array_split(data, k)
mu = np.mean(data)
sigma = np.std(data)
sum = 0
for i in range(k):
    res = integrate.quad(lambda x: 1 /((2*np.pi)**(0.5) * sigma) * np.e **
                         (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                                                  a[i][0], a[i][-1])[0]
    v_i = len(a[i])
    sum = sum + (v_i ** 2) / (n * res)
sum = sum - n
print('Статистика равна:',sum)
print('ст. свободы k - 1 = ', k-1)
```

```
Статистика равна: 26.866030804660113
ст. свободы k - 1 =  14
```

$\chi^2_{0.95,14} = 23.685$ и $\chi^2_{0.99,14} = 29.141$. Следовательно, гипотеза H_0 при $\alpha = 0.05$ отвергается, а при $\alpha = 0.01$ не отвергается.

Проверим критерий для **смоделированных данных**.

```python
# критерий хи-квадрат для простой гипотезы для СМОДЕЛИРОВАННЫХ ДАННЫХ
import numpy as np
import math
import scipy.integrate as integrate
for i in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(i))
    n = len(data)
    data.sort()
    mu = 0
    k = int(math.log2(n) + 1)
    a = np.array_split(data, k)
    sigma = 1
    sum = 0
    for j in range(k):
        res = integrate.quad(lambda x: 1 /((2*np.pi)**(0.5) * sigma) * np.e **
                        (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                                                a[j][0], a[j][-1])[0]
        v_i = len(a[j])
        sum = sum + (v_i ** 2) / (n * res)
    sum = sum - n
    print('Статистика для {} выборки:'.format(i),sum)
    print('ст. свободы k - 1 = ', k-1)
```

Результат:

```
Статистика для 1 выборки: 45.40081630050554
ст. свободы k - 1 =  16
Статистика для 2 выборки: 25.48987041703367
ст. свободы k - 1 =  16
Статистика для 3 выборки: 32.435541303188074
ст. свободы k - 1 =  16
Статистика для 4 выборки: 26.873726456338773
ст. свободы k - 1 =  16
Статистика для 5 выборки: 34.92294737337215
ст. свободы k - 1 =  16
```

$\chi^2_{0.95,16} = 26.296$ и $\chi^2_{0.99,16} = 32.000$

Таким образом, при $\alpha = 0.05$ гипотеза H_0 отвергается на всех выборках, а при $\alpha = 0.01$ гипотеза H_0 не отвергается у 2 и 4 выборок.

8.4 Критерий хи-квадрат для сложной гипотезы

Оценки параметров, найденные методом максимального правдоподобия, являются результатом функций np.mean() и np.std(). Таким образом, вычисления для **реальных данных** будут идентичными этим. Но теперь мы возьмем другую $\chi^2_{0.95,13} = 22.362$ и $\chi^2_{0.99,13} = 27.688$ Следовательно, гипотеза H_0 отвергается при $\alpha = 0.05$ и не отвергается при $\alpha = 0.01$ Проверим критерий на **смоделированных данных**. Пусть неизвестен параметр $\mu = \theta_1$, а параметр $\sigma = 1$ Тогда оценка данного параметра, найденная методом максимального правдоподобия, является выборочным средним.

Код:

```
# критерий хи-квадрат для сложной гипотезы для СМОДЕЛИРОВАННЫХ ДАННЫХ
# параметр мю неизвестный, sigma = 1
import numpy as np
import math
import scipy.integrate as integrate
for i in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(i))
    n = len(data)
    data.sort()
    mu = data.sum() / n
    k = int(math.log2(n) + 1)
    a = np.array_split(data, k)
    sigma = 1
    sum = 0
    for j in range(k):
        res = integrate.quad(lambda x: 1 /((2*np.pi)**(0.5) * sigma) * np.e **
                        (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                                                a[j][0], a[j][-1])[0]
        v_j = len(a[j])
        sum = sum + (v_j ** 2) / (n * res)
    sum = sum - n
    print('Статистика для {} выборки:'.format(i),sum)
    print('ст. свободы k - 1 - 1 = ', k-1-1) #  мы приняли, что нам неизвестен только один параметр - mu
```

Результат:

```
Статистика для 1 выборки: 44.39697407910717
ст. свободы k - 1 =  15
Статистика для 2 выборки: 25.386390365296393
ст. свободы k - 1 =  15
Статистика для 3 выборки: 31.851333727463498
ст. свободы k - 1 =  15
Статистика для 4 выборки: 26.70057968259789
ст. свободы k - 1 =  15
Статистика для 5 выборки: 29.790866191367968
ст. свободы k - 1 =  15
```

$\chi^2_{0.95,15} = 24.996$ и $\chi^2_{0.99,15} = 30.578$. При уровне значимости $\alpha = 0.95$ гипотеза H_0 отвергается на всех выборках, а при $\alpha = 0.01$ на 2,4,5 выборках не отвергается.

Пусть теперь неизвестен параметр $\sigma = \theta_2$, а $\mu = 0$.

Код:

```python
# критерий хи-квадрат для сложной гипотезы для СМОДЕЛИРОВАННЫХ ДАННЫХ
# параметр mu = 0, sigma неизвестен
import numpy as np
import math
import scipy.integrate as integrate
for i in range(1,6):
    data = np.loadtxt('filenormal100000_{}.txt'.format(i))
    n = len(data)
    data.sort()
    mu = 0
    k = int(math.log2(n) + 1)
    a = np.array_split(data, k)
    sigma = ((data - mu)**2).sum() / n
    sum = 0
    for j in range(k):
        res = integrate.quad(lambda x: 1 /((2*np.pi)**(0.5) * sigma) * np.e **
                        (- 1 / 2 * (x - mu)**2 / (sigma)**2),
                                                a[j][0], a[j][-1])[0]
        v_j = len(a[j])
        sum = sum + (v_j ** 2) / (n * res)
    sum = sum - n
    print('Статистика для {} выборки:'.format(i),sum)
    print('ст. свободы k - 1 - 1 = ', k-1-1) #  мы приняли, что нам неизвестен только
```

```
Статистика для 1 выборки: 44.93752889808093
ст. свободы k - 1 - 1 =  15
Статистика для 2 выборки: 25.56609849917004
ст. свободы k - 1 - 1 =  15
Статистика для 3 выборки: 32.717770450384705
ст. свободы k - 1 - 1 =  15
Статистика для 4 выборки: 27.589091446134262
ст. свободы k - 1 - 1 =  15
Статистика для 5 выборки: 34.96970932839031
ст. свободы k - 1 - 1 =  15
```

Таким образом, при $\alpha = 0.05$ гипотеза отвергается на всех выборках, а при $\alpha = 0.01$ гипотеза не отвергается на 2 и 4 выборках.

9 Домашнее задание 4.Часть 2.

Гипотеза H_0 является нашей гипотезой, которую мы проверяем. Параллельно мы проверяем гипотезу H_1, которая является альтернативой гипотезе H_0, противоречащей ей. Составим следующую таблицу:

	$\mathscr{X}_0$	$\mathscr{X}_1$
H_0	правильно	отвергаем истину
H_1	принимаем ложь за истину	правильно

Ошибка **1 рода** заключается в том, что гипотеза H_0 будет отвергнута, хотя на самом деле она правильная:

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

Ошибка **2 рода** заключается в том, что гипотеза H_0 будет принята, но на самом деле она неправильная:

$$P(X \in \mathscr{X}_0 | H_1) = \beta$$

Функция мощности критерия - функционал на множестве допустимых распределений $\mathcal{F}$ и выборке X.

$$W(F) = W(F, \mathscr{X}_{1,\alpha}) = P(X \in \mathscr{X}_{1,\alpha}|F)$$

то есть это вероятность попасть в $\mathscr{X}_{1,\alpha}$, если F истинная гепотеза.Причем:

$$\alpha = \sup W(F)$$

$$\beta = \sup(1 - W(F))$$

При решении принять или отклонить гипотезу H_0 мы пользуемся критерием. Решение принимается на основе выборки наблюдений случайной величины ξ, необходимо выбрать подхоодящую статистику этого критерия. Критическая область - это такая область, при попадании в которую значения статистики критерия мы отклоняем гипотезу H_0, и задается вероятность попадания статистики критерия в эту область - α - уровень значимости. Критические области бывают 3 типов:

- левосторонняя
- правосторонняя
- двухсторонняя

Функция отношения правдоподобия - случайная величина - имеет вид:

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

Выберем границу для $l(\overline{X}) = c$. Если $l(\overline{x})$. Если $l(\overline{x}) \geq c$, то принимаем H_1, иначе H_0. Тогда необходимо взять все $X \in \mathscr{X}$ и упорядочить их по значению $l(\overline{x})$. Далее возьмем такую c, чтобы выполнялось следующее условие:

$$W(\theta_0, \mathscr{X}_{1,\alpha}) = \int\limits_{\mathscr{X}_{1,\alpha}} L(\overline{x}, \theta_0) d\overline{x} = \alpha$$

Рассмотрим функцию $\chi(c)$.

$$\chi(c) = P_0\{l(\overline{x}) \geq c\} = \int\limits_{\overline{x}: l(\overline{x}) \geq c} L(\overline{x}, \theta_0) d\overline{x}$$

Заметим, что чем больше c, тем меньше значение $\chi(c)$
Также

$$1 \geq P_1\{l(\overline{x}) \leq c\} = \int\limits_{\overline{x}: l(\overline{x}) \leq c} L(\overline{x}, \theta_1) d\overline{x} = c \cdot \int\limits_{\overline{x}: l(\overline{x}) \geq c} L(\overline{x}, \theta_0) d\overline{x} = c \cdot \chi(c)$$

Из этого следует, что при $c \to \infty$ $\chi(c) \to 0$.
Теорема **Леймана-Пирсона**:

Пусть $\forall\alpha\ \exists c_\alpha : \chi(c_\alpha) = \alpha$. Тогда критическая область:

$$\mathscr{X}^*_{1,\alpha} = \{\overline{x} : l(\overline{x}) \geq c_\alpha\}$$

задает наиболее мощный критерий для гипотезы H_0 с уровнем значимости α относительно альтернативы H_1 среди всех критериев с уровнем значимости α.
Так

10 Домашнее задание 5.

Регрессионный анализ - метод моделирования измеряемых данных и исследования их свойств.Данные обычно состоят из пар значений **зависимой** переменной и **независимой** переменной : (x_i, y_i).Суть регрессионного анализа заключается в нахождении наиболее важных факторов, которые влияют на зависимую переменную.
Примеры применения регрессионного анализа:

- Моделирование числа поступивших в университет для лучшего понимания факторов, удерживающих детей в том же учебной заведении;
- Моделирование потоков миграции в зависимости от таких факторов как средний уровень зарплат, наличие медицинских, школных учреждений, географическое положение и т.д...;

- Моделирование дорожных аварий как функции скорости, дорожныхм условий, погоды и т.д.;
- Моделирование потерь от пожаров как функции от таких переменных как количество пожарных станций, время обработки вызова или, цена собственности;

Термины и концепции регрессионного анализа:

- Зависимая переменная (Y) - это переменная, описывающая процесс, который мы пытаемся предсказать или понять
- Независимая переменная (X) - переменные, используемые для моделирования или прогнозирования значений зависимых переменных

Задачи **линейного** регрессинного анализа состоит в восстановлении функциональной зависимости $y(x) = a_0 + a_1 \cdot x$. Уравнение $\widehat{y} = \widehat{a}_0 + \widehat{a}_1 \cdot x$ определяет прямую линию, которая является оценкой истинной линейной регрессии. Данные коэффициенты можно вычислить различными способами:

- минимизируя сумму квадратов отклонений

$$E(\widehat{a}_0, \widehat{a}_1) = \sum_{i=1}^{n} (\widehat{a}_0 + \widehat{a}_1 x_i - y_i)^2$$

- численно решаю систему уравнений

$$\frac{\partial E(\widehat{a}_0, \widehat{a}_1)}{\partial \widehat{a}_i} = 0, i \in \widehat{0,1}$$

- и др.

Свойства коэффциента регрессии ($\widehat{a}_1$):

- коэффициент регрессии может принимать любые значения
- коэффициент регрессии не симметричен, то есть изменяется, если X и Y поменять местами
- коэффициент регрессии изменяется при изменении единиц измерения X и Y

Найдем $\widehat{a}_0$ и $\widehat{a}_1$.

$$\sum_{i=1}^{n}(\widehat{a}_0+\widehat{a}_1x_i-y_i)^2 = \sum_{i=1}^{n}\widehat{a}_0^2+\widehat{a}_1^2x_i^2+y_i^2+2\widehat{a}_0\widehat{a}_1x_i-2\widehat{a}_0y_i-2\widehat{a}_1x_iy_i =$$

$$= n\widehat{a}_0^2+\widehat{a}_1^2\sum_{i=1}^{n}x_i^2+\sum_{i=1}^{n}y_i^2+2\widehat{a}_0\widehat{a}_1\sum_{i=1}^{n}x_i-2\widehat{a}_0\sum_{i=1}^{n}y_i-2\widehat{a}_1\sum_{i=1}^{n}x_iy_i =$$

$$= n\cdot\widehat{a}_0^2+n\widehat{a}_1^2\cdot\overline{X^2}+n\overline{Y^2}+2n\widehat{a}_0\widehat{a}_1\overline{X}-2n\widehat{a}_0\overline{Y}-2\widehat{a}_1\sum_{i=1}^{n}x_iy_i$$

Возьмем частную производную по $\widehat{a}_0$.

$$\frac{\partial E(\widehat{a}_0,\widehat{a}_1)}{\partial\widehat{a}_0} = 2n\widehat{a}_0 + 2n\widehat{a}_1\overline{X} - 2n\overline{Y} = 0$$

$$\widehat{a}_0 + \widehat{a}_1\overline{X} = \overline{Y}$$

Возьмем частную производную по $\widehat{a}_1$.

$$\frac{\partial E(\widehat{a}_0,\widehat{a}_1)}{\partial\widehat{a}_0} = 2n\widehat{a}_1\overline{X^2} + 2n\widehat{a}_0\overline{X} - 2\sum_{i=1}^{n}x_iy_i = 0$$

$$\widehat{a}_1\overline{X^2} + \widehat{a}_0\overline{X} - \frac{1}{n}\sum_{i=1}^{n} x_i y_i = 0$$

Решив систему уравнений:

$$\begin{cases} \widehat{a}_0 + \widehat{a}_1\overline{X} = \overline{Y} & = 0 \\ \widehat{a}_1\overline{X^2} + \widehat{a}_0\overline{X} - \frac{1}{n}\sum_{i=1}^{n} x_i y_i = 0 & = 0 \end{cases}$$

Получим:

$$\widehat{a}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \overline{Y}\cdot\overline{X}}{\overline{X^2} - (\overline{X})^2}$$

$$\widehat{a}_0 = \frac{\overline{Y}\cdot\overline{X^2} - \overline{X}\sum_{i=1}^{n} x_i y_i}{\overline{X^2} - (\overline{X})^2}$$

Также рекомендуется вычислять коэффициент корреляции Пирсона:

$$r_{xy} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \sum_{i=1}^{m}(y_i - \bar{y})^2}} = \frac{cov(x,y)}{\sqrt{s_x^2 s_y^2}},$$

где $\overline{x}$, $\overline{y}$ - выборочный средние X,Y соответственно, s_x^2, s_y^2 - выборочные дисперсии. $r_{xy} \in [-1,1]$. Между значением r_{xy} и зависимостью данных есть некоторые соотношения (таблица Чеддока).

Абсолютное значение r_{xy}	Теснота (сила) корреляционной связи
менее 0.3	слабая
от 0.3 до 0.5	умеренная
от 0.5 до 0.7	заметная
от 0.7 до 0.9	высокая
более 0.9	весьма высокая

В задании рекомендуется выбирать данные, имеющие $|r_{xy}| \geq 0.7$
Возьмем следующие данные: в файле height.txt есть рост и вес 25 000 людей. Посмотрим, как рост влияет на вес.
Результат:

```
import numpy as np
import matplotlib.pyplot as plt
#Весь людей дан в фунтах 1 pounds = 0.45359237 килограмм (кг)
# рост людей дан в дюймах 1 дюйм = 2.54 см
x_dt = np.loadtxt('height.txt',usecols=1,skiprows=1) # рост
y_dt = np.loadtxt('height.txt',usecols=2,skiprows=1) # вес
#x_dt = x_dt * 2.54
#y_dt = y_dt * 0.45359237
x_mean = np.mean(x_dt)
y_mean = np.mean(y_dt)
r_xy = ((x_dt - x_mean)*(y_dt - y_mean)).sum() / (((x_dt - x_mean)**2).sum() *
                                                  ((y_dt - y_mean)**2).sum() )**(0.5)
print(r_xy)
```

```
0.5028585206028441
```

Получается, что зависимость заметная и мы "не выбираем"данные.
Рассмотрим другие данные: зависимость ширины листка от его длины. Данные взяты отсюда: https://www.kaggle.com/sharmistha96/linear-model-on-iris-dataset/data

Код:

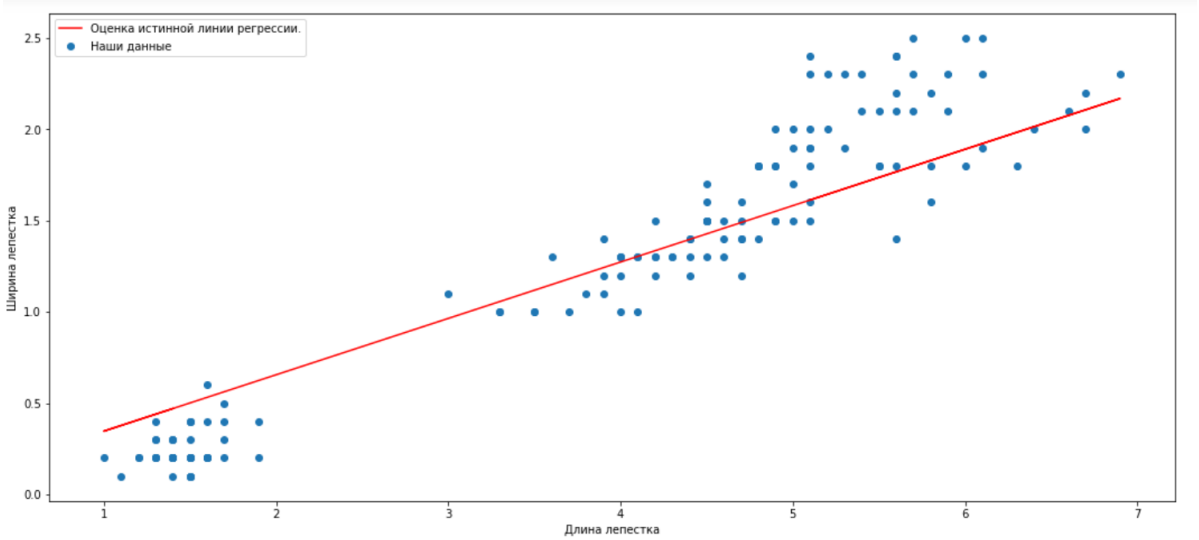
```
# зависимость ширины лепестка от его длины
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dt = pd.read_csv('IRIS.csv', usecols=['petal_length', 'petal_width'])
x_dt = np.array(dt['petal_length'].tolist())
y_dt = np.array(dt['petal_width'].tolist())
x_mean = np.mean(x_dt)
y_mean = np.mean(y_dt)
n = 500
r_xy = ((x_dt - x_mean)*(y_dt - y_mean)).sum() / (((x_dt - x_mean)**2).sum() *
                                                  ((y_dt - y_mean)**2).sum() )**(0.5)
print('r_xy =', r_xy)
a_0 = (y_mean  * (x_dt**2).sum() / n - x_mean*(x_dt * y_dt).sum() / n) / ((x_dt**2).sum()
                                                                          / n  - x_mean**2)
print('a_0 =',a_0)
a_1 = ((x_dt * y_dt).sum() / n  - y_mean * x_mean ) / ((x_dt**2).sum() / n  - x_mean**2)
print('a_1 = ',a_1)
y = x_dt * a_1 + a_0
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x_dt,y,label='Оценка истинной линии регрессии.',color = 'r')
plt.scatter(x_dt,y_dt,label='Наши данные')
ax.legend()
plt.show()
```

Результаты:

```
r_xy = 0.9627570970509662
a_0 = -0.22744378233056575
a_1 =  3.3254536560043646
```

Таким образом, зависимость между двумя этими величинами весьма высокая.

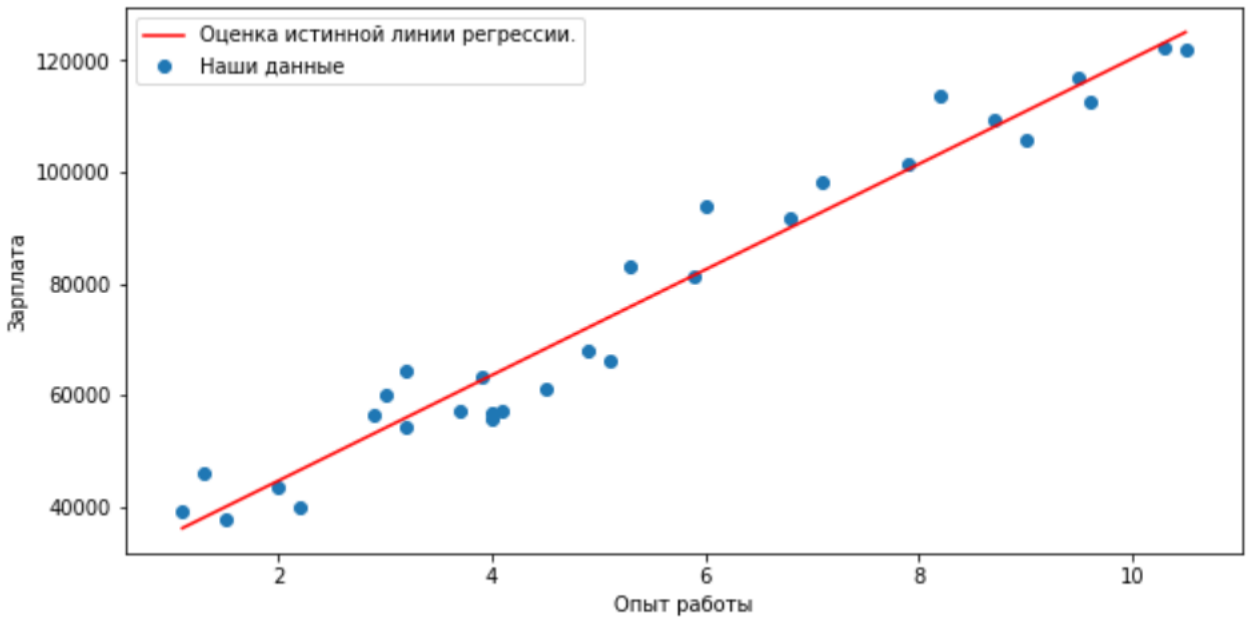
График:



Рассмотрим другие данные, взятые по ссылке: https://www.kaggle.com/karthickveerakumar/salary-data-simple-li
Это зависимость зарплаты от опыта работы.
Значения:

```
r_xy = 0.97824161848876
a_0 = 25792.200198668714
a_1 =  9449.962321455081
```

График:



Расмотрим баскетбольную статистику с 1980 по 2005 год по всем клубам. Данные взяты с https://www.kaggle.com/gantalaswetha/usa-housing-dataset-linear-regre data. Посмотрим, как согласуется число побед за сезон с разницей между забитыми и пропущенными очками за сезон.

Полученные значения:

```
r_xy = 0.9707432626446404
a_0 = 41.0
a_1 =  0.032586332821132886
```

График:


Оценка истинной линии регрессии.
Наши данные
Число побед за сезон
Разница между забитыми и пропущенными очками за сезон
0
10
20
30
40
50
60
70
−1000
−500
0
500
1000