Лабораторная работа №2

Методы оценки статистических характеристик, связанных с распределением пользователей на плоскости

Логинов Сергей НФИмд-01-22

Цель работы:

Ознакомление с методами оценки статистических характеристик

In [16]:
```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import math
```

Задание 1

Сгенерировать выборку случайных чисел размером 100 и 1000 для двух распределений - экспоненциального и нормального. Для созданных выборок сделать следующее:

1. Посчитать выборочное среднее и дисперсию, сравнить с математическим ожиданием соответствующих распределений;
2. Посчитать 0.5 и 0.99 квантили, сравнить с соответствующими теоретическими значениями;
3. Построить гистограмму распределения;
4. Построить функцию распределения случайной величины на основе выборки (на одном графике показать функции распределения, полученные из выборок разного размера и теоретическую);
5. Построить плотность распределения случайной величины на основе выборки (на одном графике показать плотности распределения, полученные из выборок разного размера и теоретическую).

1 Нормальное распределение

Генерируем выборку объема 100 из нормального распределения Norm(0, 1)

In [447…
```
a = stats.norm.rvs(0, 1, 100)
```

Выборочное среднее

In [448…
```
a.mean()
```

Out[448]:
```
0.04602430121065512
```

Выборочная дисперсия

In [449…
```
a.var()
```

Out[449]: 0.8359258002456597

Генерируем выборку объема 1000 из нормального распределения Norm(0, 1)

In [450…
```
b = stats.norm.rvs(0, 1, 1000)
```

Выборочное среднее

In [451…
```
b.mean()
```

Out[451]: -0.05410612845464068

Выборочная дисперсия

In [452…
```
b.var()
```

Out[452]: 0.9262816600106509

МО и дисперсия генеральной совокупности равны 0 и 1 соответственно (генерировали выборки из ГС с МО = 0 и СКО = 1, следовательно и дисперсия = 1). Выборка размера 1000 имеет более точные показатели характеристик

2

Находим 0.5 и 0.99 квантили каждой выборки

In [453…
```
print('Size 100: \n0.5 =',
      np.percentile(a, 50),
      '0.99 =',
      np.percentile(a, 99),
     '\nSize 1000: \n0.5 =',
      np.percentile(b, 50),
      '0.99 =',
      np.percentile(b, 99))
```

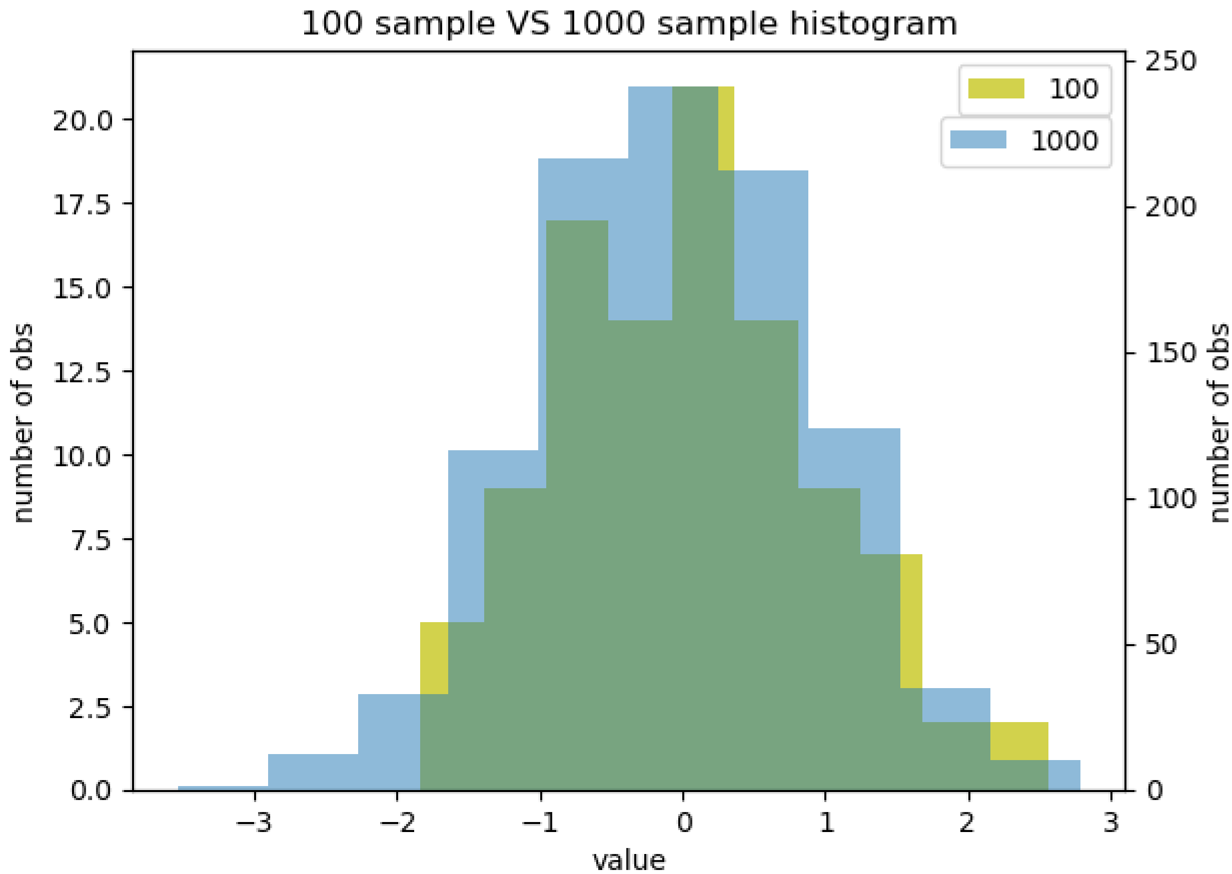
```
Size 100:
0.5 = 0.010745160621980731 0.99 = 2.283788419046739
Size 1000:
0.5 = -0.04212978804412616 0.99 = 2.071857806302335
```

3

Строим гистограммы

In [454…
```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.hist(a,
         label='100',
         color='y',
         alpha=0.7)
ax2.hist(b,
         label='1000', alpha=0.5)
ax1.set_xlabel('value')
ax1.set_ylabel('number of obs')
ax2.set_ylabel('number of obs')
ax1.legend()
ax2.legend(bbox_to_anchor=(1, 0.93))
```

```
plt.title('100 sample VS 1000 sample histogram')
plt.show()
```



5

Функция для вычисления значений плотности распределения, внутри - формула плотности нормального распределения, в которую подставляем полученные при генерации выборок значения Рисуем графики двух выборок и график полноценного (теоретического) нормального распределения с параметрами (0, 1)

In [455…

```
def fr_norm(x):
    a = []
    for val in x:
        a.append(1 / (x.std() * math.sqrt((2 * 3.14))) * math.exp((val - x.mean())**2/(-
    return a
```

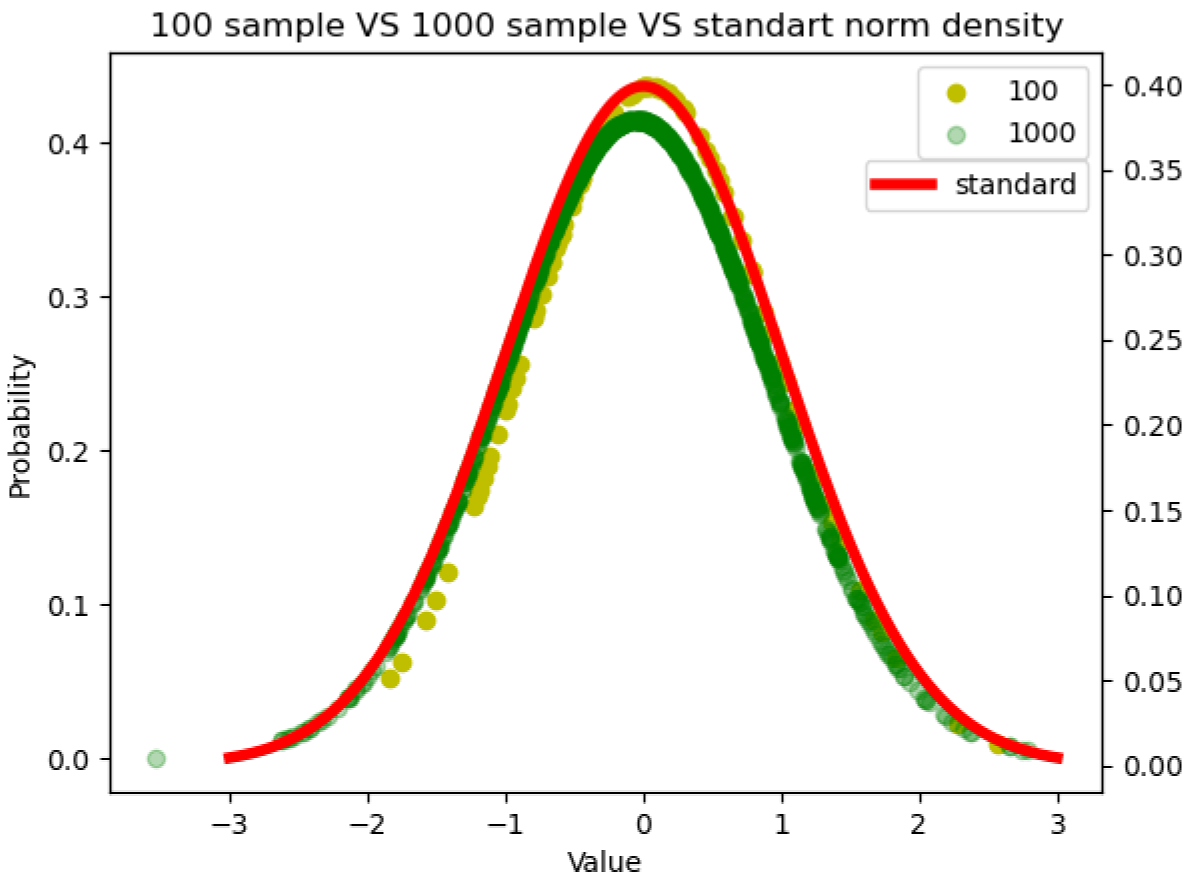
In [456…

```
y = fr_norm(a)
x = fr_norm(b)
x_st = np.arange(-3, 3, 0.001)
```

In [457…

```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.scatter(a, y,
            linewidth=1,
            c='y',
            label='100')
ax1.scatter(b, x,
            linewidth=1,
            alpha=0.3,
            c='g',
            label='1000')
ax2.plot(x_st, stats.norm.pdf(x_st, 0, 1),
         c='red',
```

```
            linewidth=4,
            label='standard')
ax1.set_xlabel('Value')
ax1.set_ylabel('Probability')
ax1.legend()
ax2.legend(loc='center right', bbox_to_anchor=(1, 0.82))
plt.title('100 sample VS 1000 sample VS standart norm density')
plt.show()
```



4

Функция распределения и график

Попробуем вычислить значения кумулятивной ФР по имеющимся выборкам

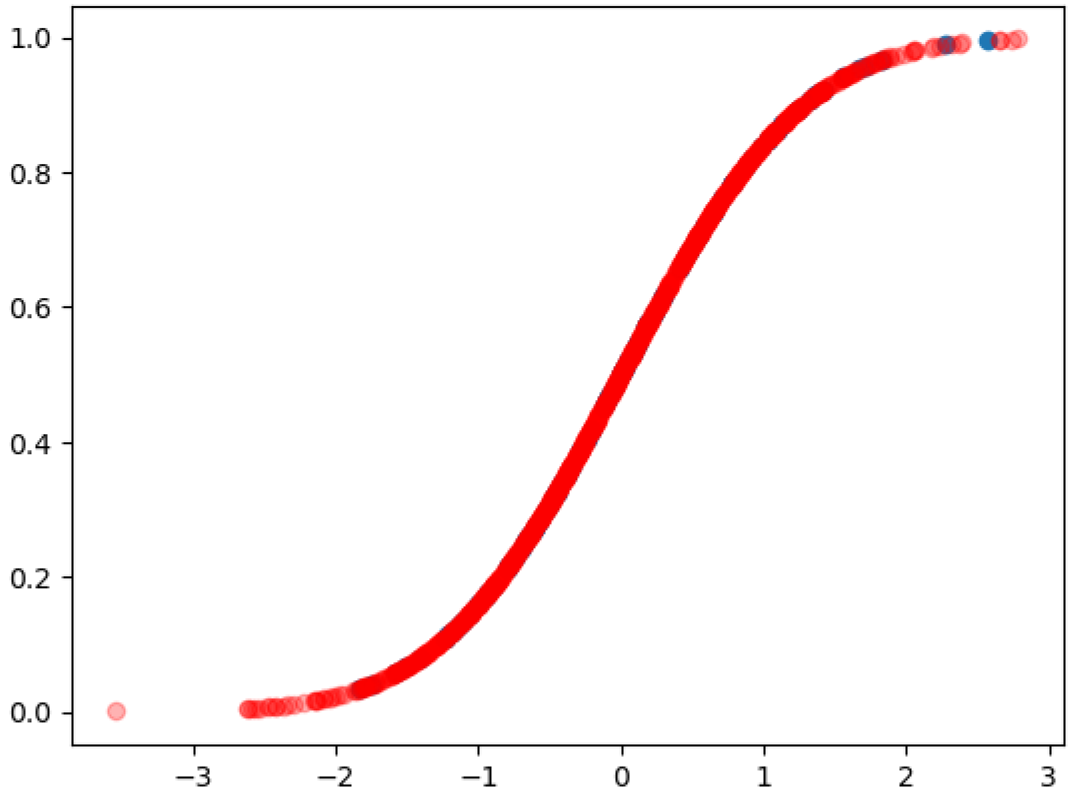
In [458…

```
c = stats.norm.cdf(a, 0, 1)
d = stats.norm.cdf(b, 0, 1)
```

Построим графики

In [459…

```
plt.scatter(a, c, linewidth=1)
plt.scatter(b, d, linewidth=1, alpha=0.3, color='r')
plt.show()
```



Получился слишком хороший результат (очень похоже на теоретическую ФР). Вероятно, такого быть не должно. Значит имеет место ошибка в подходе. Попробуем зайти с другой стороны.

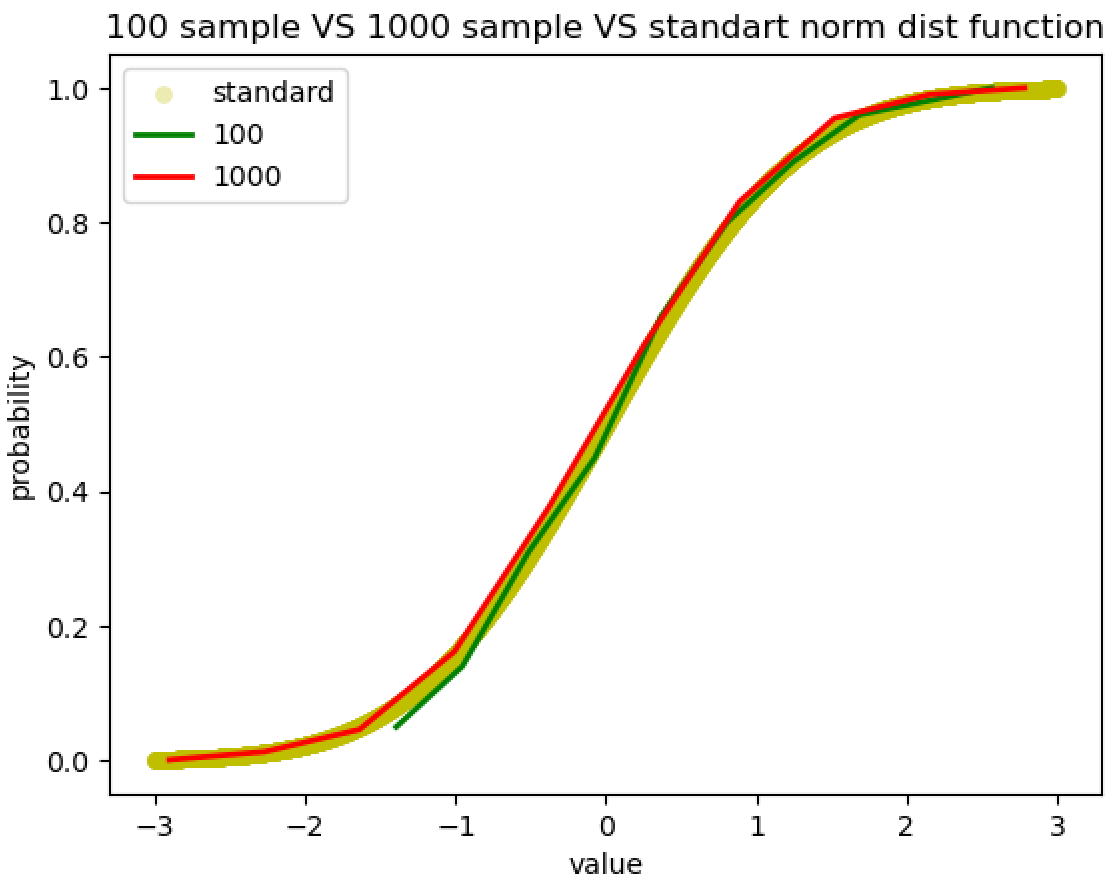
Сделаем следующие вещи:

- Построим гистограммы, сохранив значения бинов и их количество
- Далее найдем значения функции распределения как долю каждого бина
- Но даже после этого полученные значения не дают возможность построить нормальный график. Для этого используем сумму с накоплением
- А далее по оси X откладываем наши бины (с первого чтобы совпадала размерность), начиная с первого, по оси У - значения КФР и рисуем уже более точные графики.
- Также не забываем нарисовать график теоретической ФР

In [460…

```
count, bins = np.histogram(a, bins=10)
pdf = count / sum(count)
cdf = np.cumsum(pdf)
count1, bins1 = np.histogram(b, bins=10)
pdf1 = count1 / sum(count1)
cdf1 = np.cumsum(pdf1)
y_st = stats.norm.cdf(x_st, 0, 1)
plt.scatter(x_st, y_st,
            linewidth=0.2,
            alpha=0.3,
            c='y',
            label='standard')
plt.plot(bins[1:], cdf,
         color='g',
         linewidth=2,
         label='100')
plt.plot(bins1[1:], cdf1,
         color='r',
         linewidth=2,
         label='1000')
plt.legend()
plt.title('100 sample VS 1000 sample VS standart norm dist function')
```

```
plt.xlabel('value')
plt.ylabel('probability')
plt.show()
```



Экспоненциальное

Генерируем выборки

In [464…
```
a = stats.expon.rvs(size=100)
b = stats.expon.rvs(size=1000)
```

Характеристики

In [418…
```
print(a.mean(), a.var())
```

```
0.9928851409126355 1.0422398732696196
```

In [419…
```
print(b.mean(), b.var())
```

```
0.9755787696272542 0.9357925427718293
```

Квантили

In [416…
```
print('Size 100: \n0.5 =',
      np.percentile(a, 50),
      '0.99 =',
      np.percentile(a, 99),
     '\nSize 1000: \n0.5 =',
      np.percentile(b, 50),
      '0.99 =',
      np.percentile(b, 99))
```

```
Size 100:
0.5 = 0.6151332979526305 0.99 = 5.37347762753218
```

```
Size 1000:
0.5 = 0.6874681141242929 0.99 = 4.456870288477133
```

Плотность распределения, где лямбда = 1

In [465…

```python
def exp_dd(x):
    a = []
    for x in x:
        a.append(math.exp(-x))
    return a
```

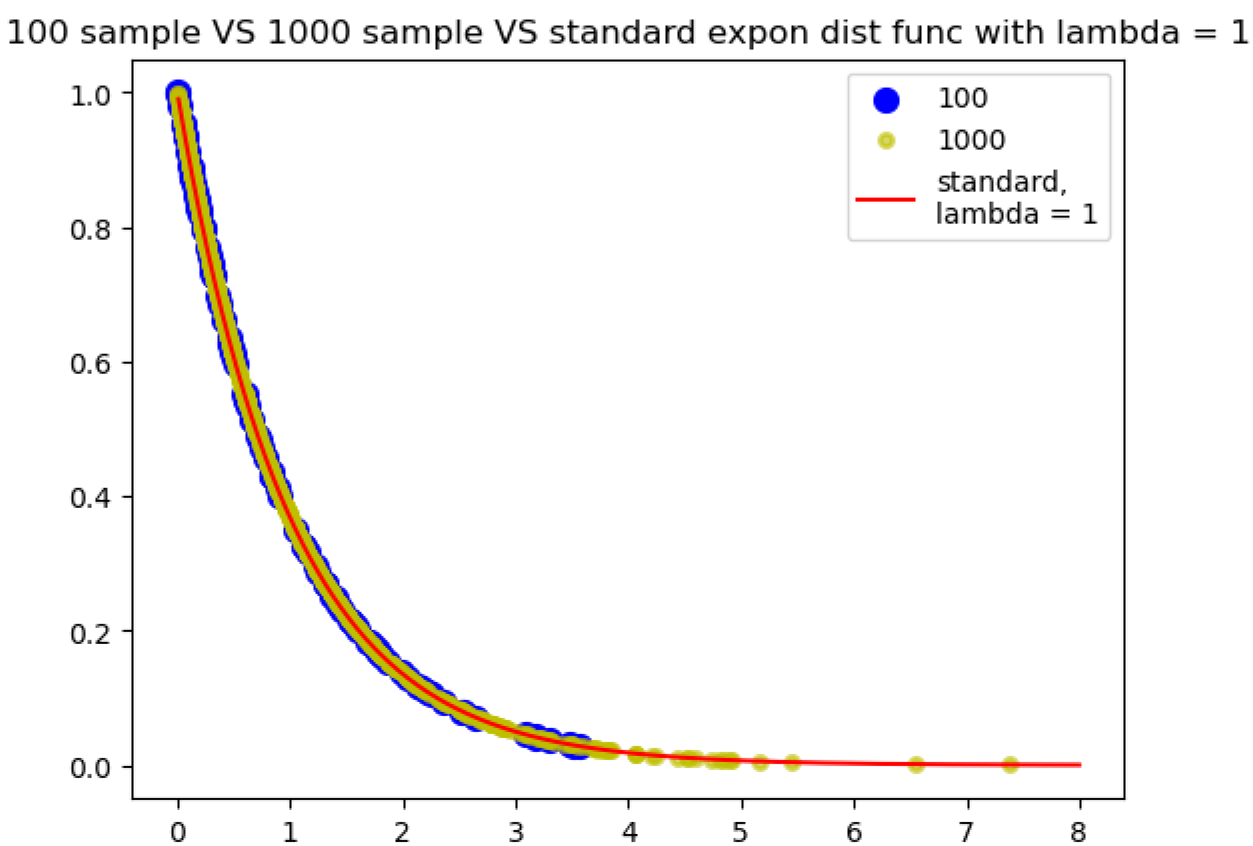
In [466…

```python
dd1 = exp_dd(a)
dd2 = exp_dd(b)
x_st = np.arange(0.01, 8, 0.001)
```

Строим графики а также теоретическую плотность

In [405…

```python
plt.scatter(a, dd1,
            linewidth=5,
            color='b',
            s=20,
            label='100')
plt.scatter(b, dd2,
            linewidth=2,
            alpha=0.6,
            s=20,
            c='y',
            label='1000')
plt.plot(x_st, stats.expon.pdf(x_st, 0, 1),
         c='r',
         label='standard,\nlambda = 1')
plt.title('100 sample VS 1000 sample VS standard expon dist func with lambda = 1')
plt.legend()
plt.show()
```

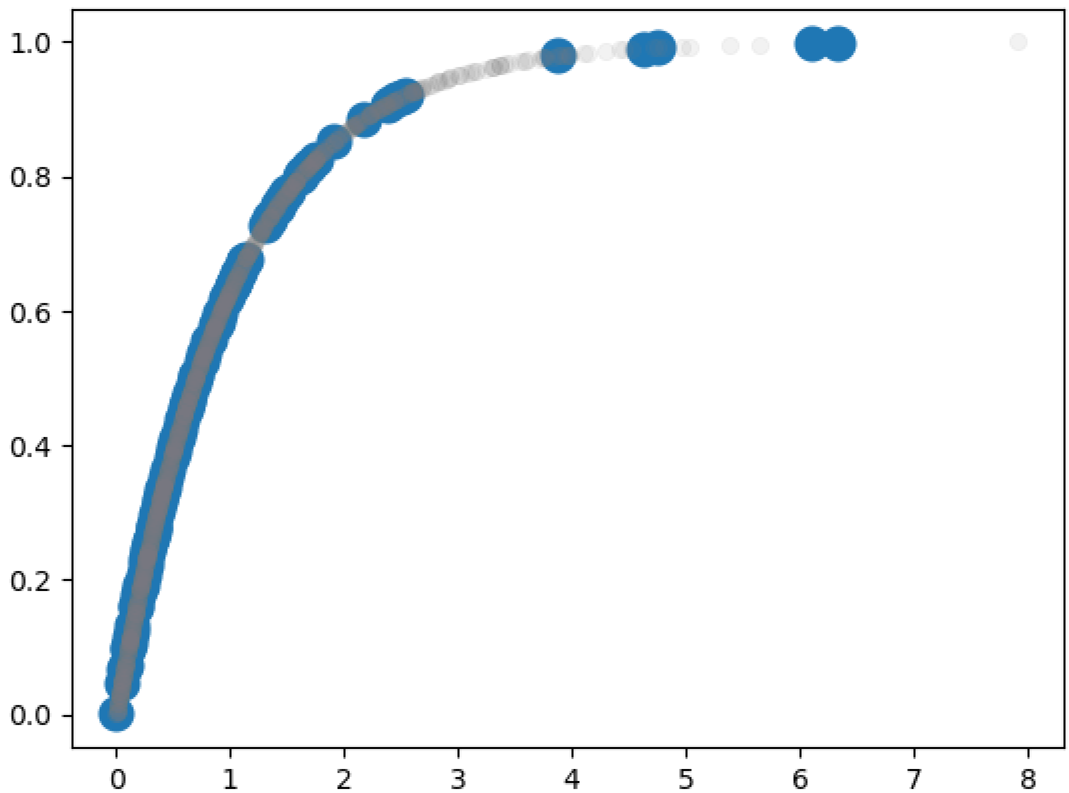


Функция распределения

In [467…
```
def exp_fd(x):
    a = []
    for x in x:
        a.append(1 - math.exp(-x))
    return a
```

In [468…
```
fd1 = exp_fd(a)
fd2 = exp_fd(b)
```

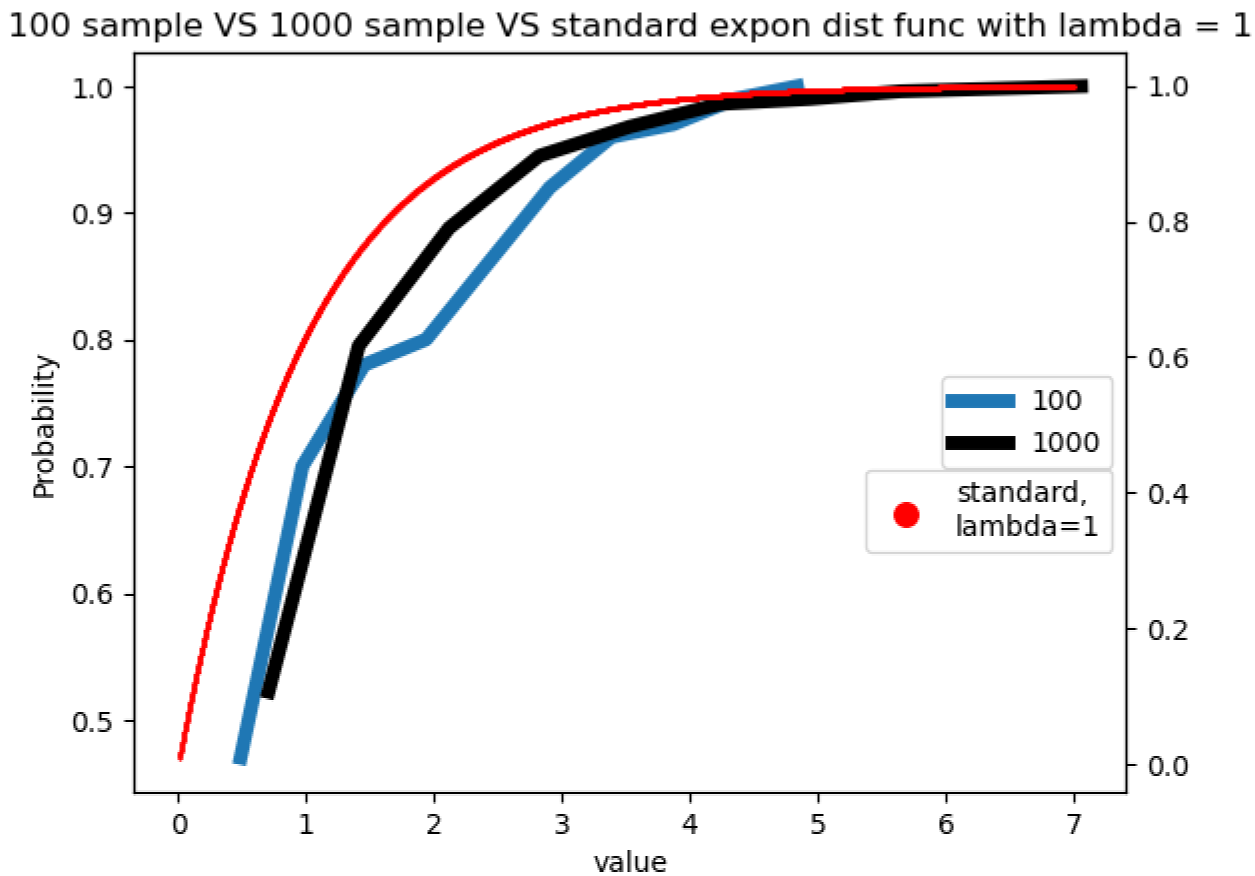
In [469…
```
plt.scatter(a, fd1, linewidth=7)
plt.scatter(b, fd2, alpha=0.1, color='grey', linewidth=0.5)
plt.show()
```



Опять получен слишком хороший результат, пробуем подход, аналогичный ФР нормального

In [359…
```
count, bins = np.histogram(a, bins=10)
pdf = count / sum(count)
cdf = np.cumsum(pdf)
count1, bins1 = np.histogram(b, bins=10)
pdf1 = count1 / sum(count1)
cdf1 = np.cumsum(pdf1)
x = np.arange(0.01, 7, 0.001)
y = stats.expon.cdf(x, 0, 1)
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(bins[1:], cdf,
         label='100',
         linewidth=5)
ax1.plot(bins1[1:], cdf1,
         label='1000',
         c='black',
         linewidth=5,
         alpha=1)
ax2.scatter(x, y,
            c='r',
            label='standard,\nlambda=1',
```

```
             linewidths=0.001,
             s=2)
ax1.legend(loc='right')
ax2.legend(loc='center right',
           bbox_to_anchor=(1, 0.38),
           markerscale=6.6)
plt.title('100 sample VS 1000 sample VS standard expon dist func with lambda = 1')
ax1.set_xlabel('value')
ax1.set_ylabel('Probability')
plt.show()
```



Результат выглядит более точным и реальным.

Вывод

Точность аппроксимации увеличивается вместе с увеличением объема экспериментальных выборок

Задание 2

- Сгенерировать три выборки размера 100, 1000 и 10000 для случайных расстояний между двумя точками, равномерно распределенные в прямоугольнике со сторонами 10 и 30.
- Получить среднее значение расстояния между точками
- Построить функцию распределения вероятностей
- Построить плотность вероятностей
- Показать разницу между соответствующими функциями на одном графике.

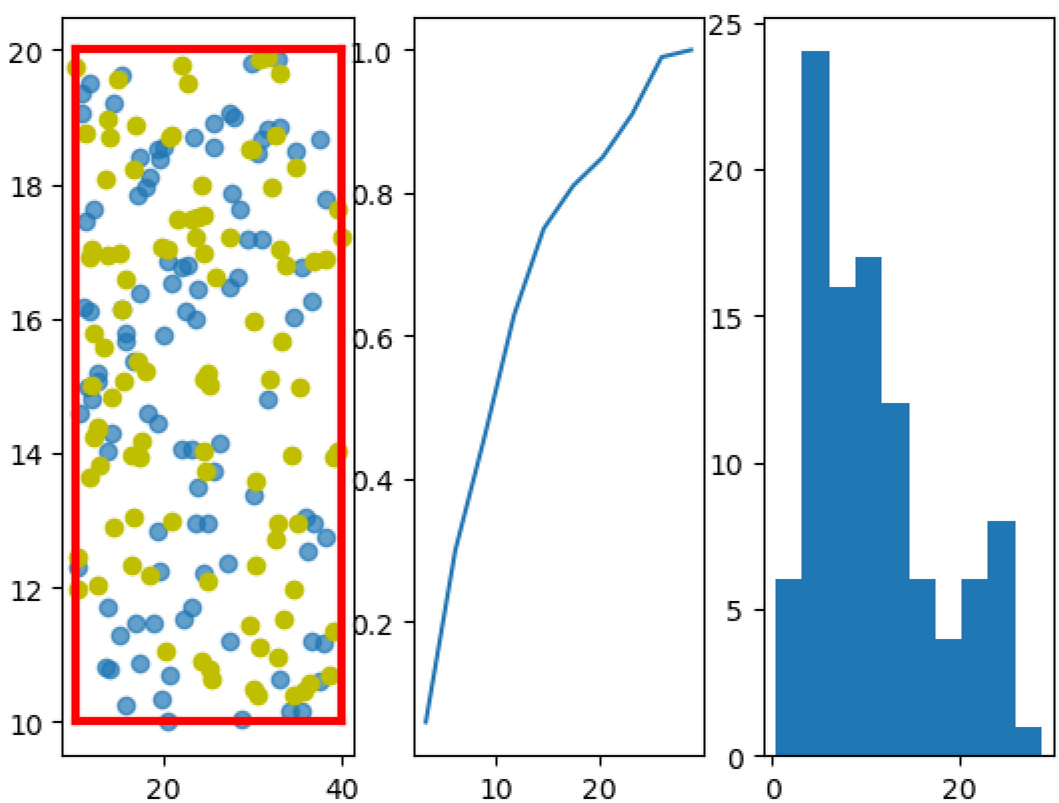
Тестовый вариант пишем для выборки размера 100.

- Генерируем координаты 200 точек (в таком случае будет 100 значений расстояния). Они будут равномерно распределены в прямоугольнике 30 на 10, начало отрисовки прямоугольника - точка (10, 10), следовательно пределы генерации увеличиваем на 10
- Находим расстояния между точками по известной формуле для координатных вычислений
- Находим среднее расстояние
- На третьем графике рисуем прямоугольник и точки в нем
- По уже известной методике находим значения ФР и КФР для графика
- Плотность строим в виде гистограммы

In [470…

```
x = np.random.uniform(10, 40, 100)
y = np.random.uniform(10, 20, 100)
x1 = np.random.uniform(10, 40, 100)
y1 = np.random.uniform(10, 20, 100)
dist = [math.sqrt((x1[i] - x[i])**2 + (y1[i] - y[i])**2) for i in range(100)]
dist = np.array(dist)
print('Mean distance: ', dist.mean())
count, bins = np.histogram(dist, bins=10)
pdf = count / sum(count)
cdf = np.cumsum(pdf)
fig, ax = plt.subplots(1, 3)
ax[2].hist(dist)
ax[1].plot(bins[1:], cdf)
ax[0].scatter(x, y, alpha=0.7)
ax[0].scatter(x1, y1, c='y')
ax[0].plot([10, 40], [20, 20],
         [10, 40], [10, 10],
         [10, 10], [20, 10],
         [40, 40], [20, 10],
        color='r', linewidth=3)
plt.show()
```

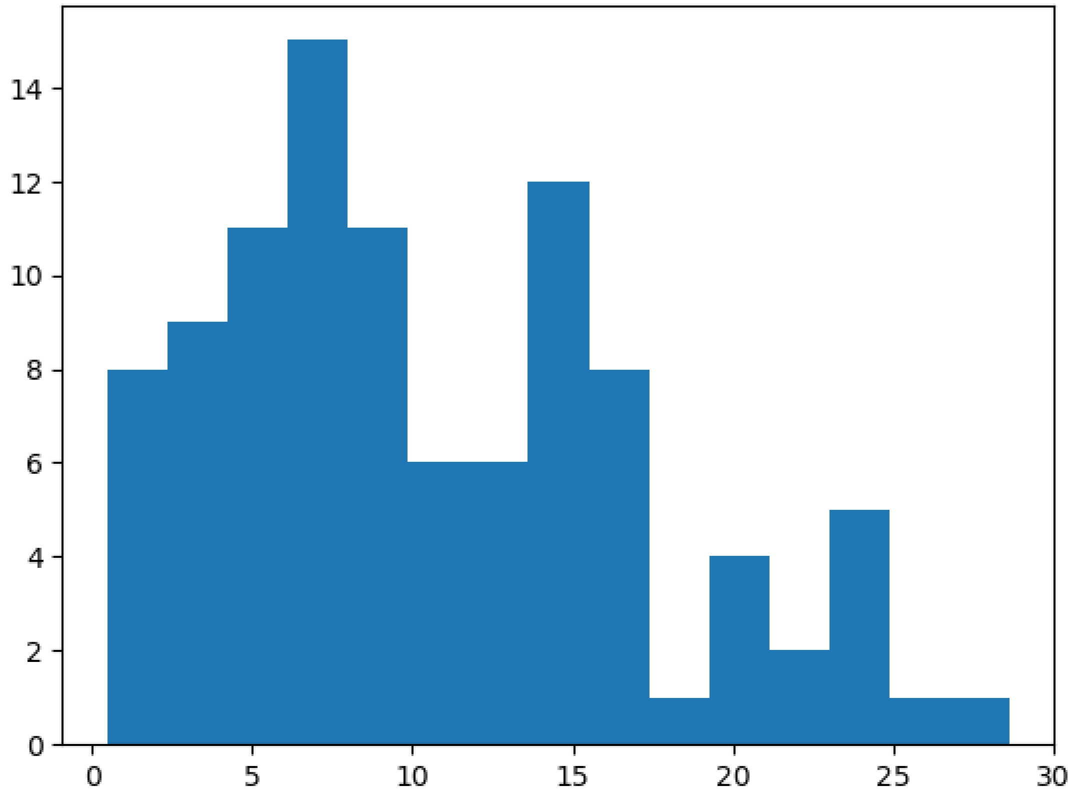
```
Mean distance:  11.078048580010464
```



Эта часть была написана до обобщения задачи в ячейке выше, просто тестовые материалы.
Плотность расстояний:

In [195…

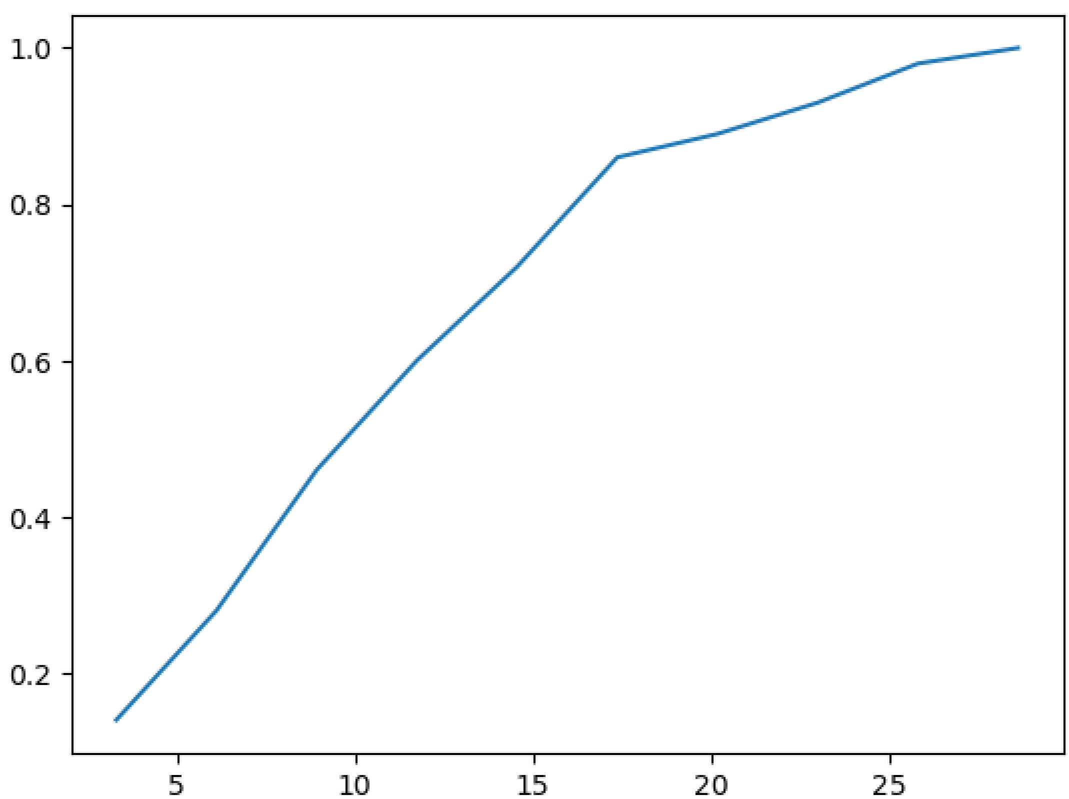
```
plt.hist(dist, bins=15)
plt.show()
```



ФР расстояний

In [196…

```
count, bins = np.histogram(dist, bins=10)
pdf = count / sum(count)
cdf = np.cumsum(pdf)
plt.plot(bins[1:], cdf)
plt.show()
```



- Для более удобного решения напишем функцию. Она позволит выполнить все операции из ячеек выше для заданного прямоугольника и количества выборок. Для лучшего отображения задаем параметры прозрачности и цветов
- Делаем то же самое, но на одном графике для всех трех выборок Получаем
- Среднее значение расстояний
- График плотностей
- График ФР

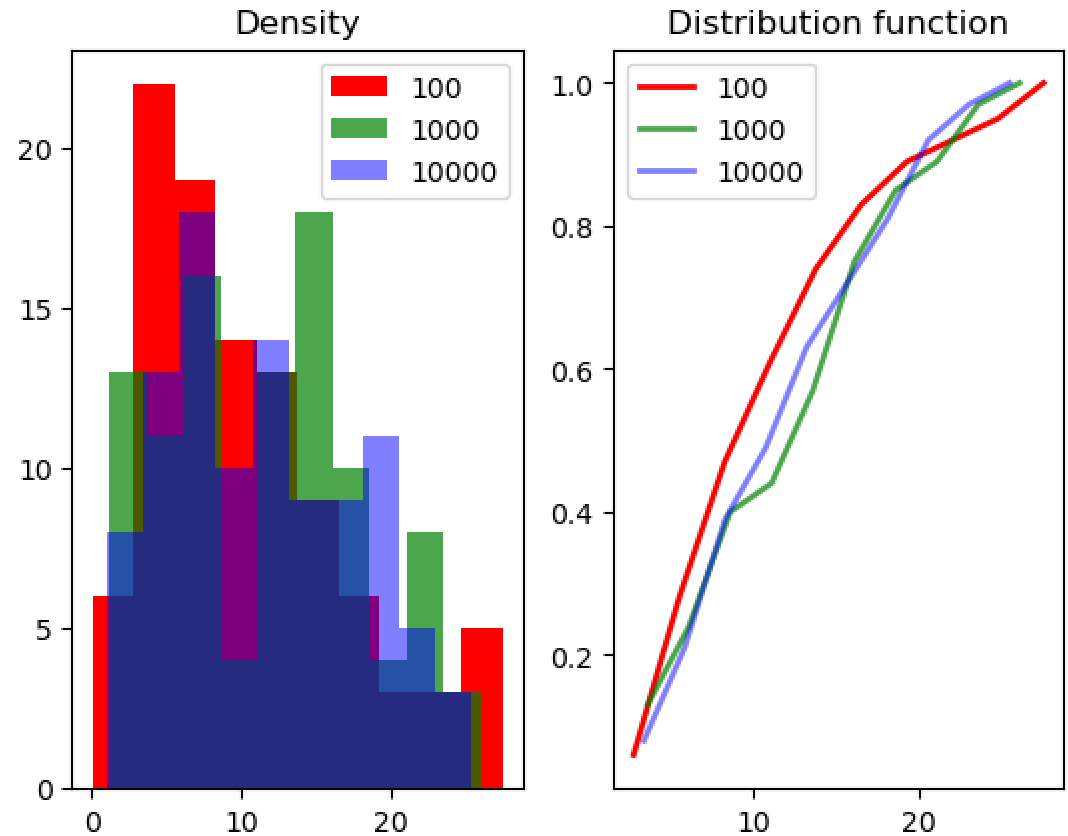
In [471…
```
def second(length, height, size, alpha, colors):
    fig, ax = plt.subplots(1, 2)
    for i in range(len(size)):
        x = np.random.uniform(10, 10 + length, size[i])
        y = np.random.uniform(10, 10 + height, size[i])
        x1 = np.random.uniform(10, 10 + length, size[i])
        y1 = np.random.uniform(10, 10 + height, size[i])
        dist = [math.sqrt((x1[i] - x[i])**2 + (y1[i] - y[i])**2) for i in range(100)]
        dist = np.array(dist)
        print('\nSize: ', str(size[i]), '\nMean distance: ', dist.mean())
        count, bins = np.histogram(dist, bins=10)
        pdf = count / sum(count)
        cdf = np.cumsum(pdf)
        ax[0].hist(dist,
                   bins=10,
                   alpha=alpha[i],
                   label=str(size[i]),
                   color=colors[i])
        ax[1].plot(bins[1:],
                   cdf,
                   linewidth=2,
                   alpha=alpha[i],
                   color=colors[i],
                   label=str(size[i]))
        ax[0].set_title('Density')
        ax[1].set_title('Distribution function')
        ax[0].legend()
        ax[1].legend()
```

In [472…
```
second(30, 10, [100, 1000, 10000], [1, 0.7, 0.5], ['r', 'g', 'b'])
```

```
Size:  100
Mean distance:  10.28311460779964

Size:  1000
Mean distance:  11.642522443775114

Size:  10000
Mean distance:  11.403122025874998
```



Распределение имеет сходство с нормальным