# Лабораторная работа № 2

## Задание 1.

import numpy as np  
import matplotlib.pyplot as plt

## Задание 1.1

#### Выборка случайных чисел размером 100 и 1000 для экспоненциального распределения.

exp\_100 = np.random.exponential(1, size=100) #первый парметр лямбда  
exp\_1000 = np.random.exponential(1, size=1000)

#### Выборка случайных чисел размером 100 и 1000 для нормального распределения.

normal\_100 = np.random.normal(0, 1, size=100) #первый парметр мат. ожидание, второй среднее отклонение  
normal\_1000 = np.random.normal(0, 1, size=1000)

Считаем выборочное среднее и дисперсию для экспоненциального и нормального распределения случайных чисел размером 100.

print('Mean for exp\_100:', np.mean(exp\_100))  
print('Var for exp\_100:', np.var(exp\_100))  
print('Mean for normal\_100:', np.mean(normal\_100))  
print('Var for normal\_100:', np.var(normal\_100))

Mean for exp\_100: 0.9679209762001996  
Var for exp\_100: 0.744037698122693  
Mean for normal\_100: -0.09554952833351896  
Var for normal\_100: 1.1564401268512627

Считаем выборочное среднее и дисперсию для экспоненциального и нормального распределения случайных чисел размером 1000.

print('Mean for exp\_1000:', np.mean(exp\_1000))  
print('Var for exp\_1000:', np.var(exp\_1000))  
print('Mean for normal\_1000:', np.mean(normal\_1000))  
print('Var for normal\_1000:', np.var(normal\_1000))

Mean for exp\_1000: 1.0200550787381597  
Var for exp\_1000: 1.035875411883424  
Mean for normal\_1000: 0.003997072227004246  
Var for normal\_1000: 1.0075419229095148

### Сравнительная таблица

Из теоретической части:

* мат.ожидание для экспоненциального распределения = $ ^-1 $
* дисперсия для экспоненциального распределения = $ ^-2 $
* мат.ожидание для нормального распределения = $ $
* дисперсия для нормального распределения = $ ^2 $

|  | 100 | 1000 | Theoretic |
| --- | --- | --- | --- |
| Mean exp | 0.9679209762001996 | 1.0200550787381597 | 1 |
| Var exp | 0.744037698122693 | 1.035875411883424 | 1 |
| Mean normal | -0.09554952833351896 | 0.003997072227004246 | 0 |
| Var normal | 1.1564401268512627 | 1.0075419229095148 | 1 |

## Задание 1.2

Считаем 0.5 квантиль для экспоненциального и нормального распределения случайных чисел размером 100 и 1000.

print('Quantile 0.5 for exp\_100 =', np.quantile(exp\_100, 0.5))  
print('Quantile 0.5 for exp\_1000 =', np.quantile(exp\_1000, 0.5))  
print('Quantile 0.5 for normal\_100 =', np.quantile(normal\_100, 0.5))  
print('Quantile 0.5 for normal\_1000 =', np.quantile(normal\_1000, 0.5))

Quantile 0.5 for exp\_100 = 0.7912717455148244  
Quantile 0.5 for exp\_1000 = 0.7141470337906621  
Quantile 0.5 for normal\_100 = -0.15160365461878086  
Quantile 0.5 for normal\_1000 = -0.025540874962847

Считаем 0.99 квантиль для экспоненциального и нормального распределения случайных чисел размером 100 и 1000.

print('Quantile 0.99 for exp\_100 =', np.quantile(exp\_100, 0.99))  
print('Quantile 0.99 for exp\_1000 =', np.quantile(exp\_1000, 0.99))  
print('Quantile 0.99 for normal\_100 =', np.quantile(normal\_100, 0.99))  
print('Quantile 0.99 for normal\_1000 =', np.quantile(normal\_1000, 0.99))

Quantile 0.99 for exp\_100 = 3.0312923371733755  
Quantile 0.99 for exp\_1000 = 4.473178308315431  
Quantile 0.99 for normal\_100 = 2.353149251874975  
Quantile 0.99 for normal\_1000 = 2.465417814914604

# Расчёт квантиля для экспоненциального распределения  
from math import log  
quantile\_exp1 = - log(1-0.5) # квантиль 0.5, так как лямбда = 1  
quantile\_exp2 = - log(1-0.99) # квантиль 0.99, так как лямбда = 1  
print(f'Quantile 0.5 = {quantile\_exp1}\nQuantile 0.99 = {quantile\_exp2}')

Quantile 0.5 = 0.6931471805599453  
Quantile 0.99 = 4.605170185988091

### Сравнительная таблица

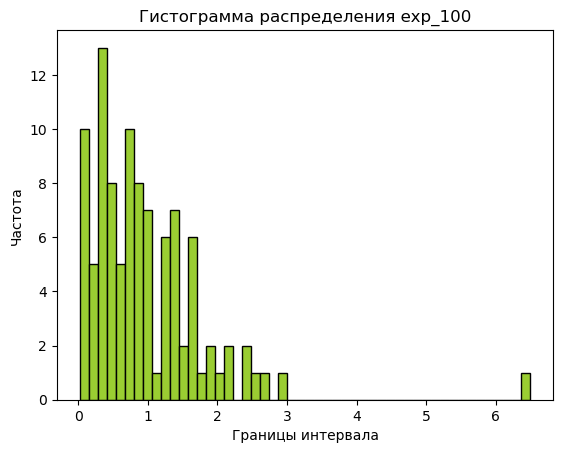
Теоретические значения берём из таблицы.

|  | 100 | 1000 | Theoretic |
| --- | --- | --- | --- |
| Quantile 0.5 exp | 0.7912717455148244 | 0.7141470337906621 | 0.6931471805599453 |
| Quantile 0.99 exp | 3.0312923371733755 | 4.473178308315431 | 4.605170185988091 |
| Quantile 0.5 normal | -0.15160365461878086 | -0.025540874962847 | 0 |
| Quantile 0.99 normal | 2.353149251874975 | 2.465417814914604 | 2.326 |

## Задание 1.3

Построим гистограмму экспоненциального распределения случайных чисел размером 100.

# Построить базовую гистограмму  
plt.hist(exp\_100, bins=50, color='yellowgreen', edgecolor='black')  
  
# Добавить метки и заголовок  
plt.xlabel('Границы интервала')  
plt.ylabel('Частота')  
plt.title('Гистограмма распределения exp\_100')  
  
# Вывести график  
plt.show()



png

Построим гистограмму экспоненциального распределения случайных чисел размером 1000.

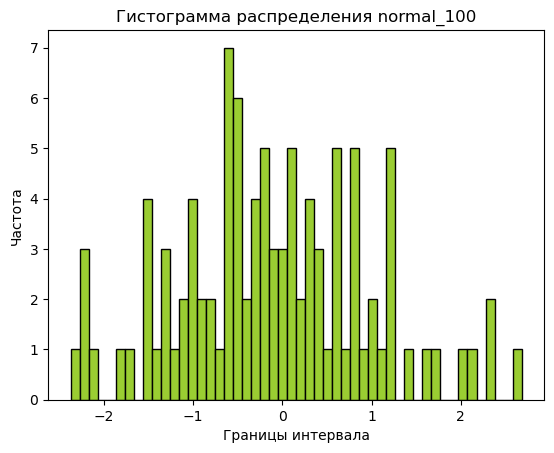
# Построить базовую гистограмму  
plt.hist(exp\_1000, bins=50, color='yellowgreen', edgecolor='black')  
  
# Добавить метки и заголовок  
plt.xlabel('Границы интервала')  
plt.ylabel('Частота')  
plt.title('Гистограмма распределения exp\_1000')  
  
# Вывести график  
plt.show()



png

Построим гистограмму нормального распределения случайных чисел размером 100.

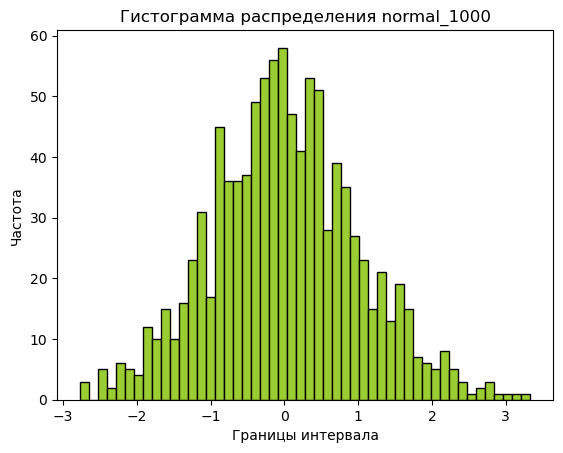
# Построить базовую гистограмму  
plt.hist(normal\_100, bins=50, color='yellowgreen', edgecolor='black')  
  
# Добавить метки и заголовок  
plt.xlabel('Границы интервала')  
plt.ylabel('Частота')  
plt.title('Гистограмма распределения normal\_100')  
  
# Вывести график  
plt.show()



png

Построим гистограмму нормального распределения случайных чисел размером 1000.

# Построить базовую гистограмму  
plt.hist(normal\_1000, bins=50, color='yellowgreen', edgecolor='black')  
  
# Добавить метки и заголовок  
plt.xlabel('Границы интервала')  
plt.ylabel('Частота')  
plt.title('Гистограмма распределения normal\_1000')  
  
# Вывести график  
plt.show()



png

## Задание 1.4

Построим функцию распределения случайной величины на основе выборки (на одном графике покажем функции распределения, полученные из выборок разного размера и теоретическую)

Функции распределения для экспоненциального распределения.

def fun\_exp(l, x):  
 return 1-np.exp(-l\*x)

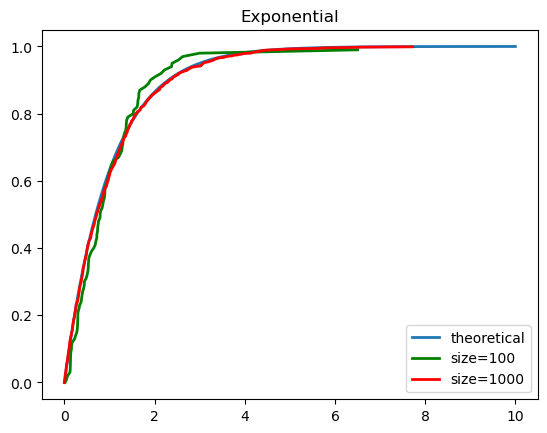
Задаем значения для х и с ними строим функцию.

theory\_exp\_x = np.linspace(0, 10, 100)  
theory\_exp\_y = fun\_exp(1, theory\_exp\_x)  
sort\_exp\_100 = sorted(exp\_100) # сортируем и преобразовываем рандомные значения экспоненциального распределения в список  
sort\_exp\_1000 = sorted(exp\_1000)

На одном графике покажем функции экспоненциального распределения, полученные из выборок разного размера и теоретическую.

plt.figure(dpi=100)   
plt.plot(theory\_exp\_x, theory\_exp\_y, lw=2, label='theoretical')  
plt.plot(sort\_exp\_100, np.arange(0.,1.,0.01), lw=2, label='size=100', color='green')   
plt.plot(sort\_exp\_1000, np.arange(0.,1.,0.001), lw=2, label='size=1000', color='red')   
plt.title('Exponential')   
plt.legend()

<matplotlib.legend.Legend at 0x2e5631ced90>



png

Функции распределения для нормального распределения.

from scipy import special  
def fun\_normal(m, sigma, x):  
 return 0.5\*(1 + special.erf(x-m / sigma\*np.sqrt(2)))

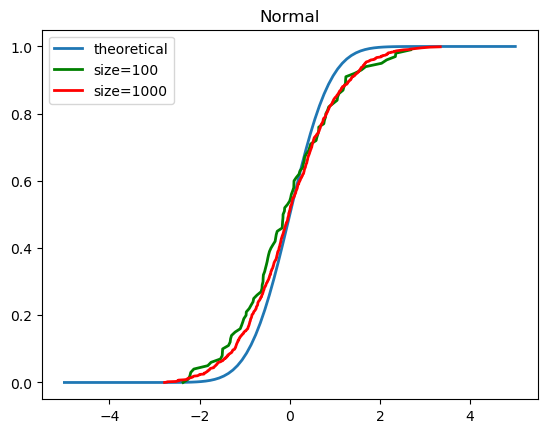
Задаем значения для х и с ними строим функцию.

theory\_normal\_x = np.linspace(-5, 5, 100)  
theory\_normal\_y = fun\_normal(0, 1, theory\_normal\_x)  
sort\_normal\_100 = sorted(normal\_100) # сортируем и преобразовываем рандомные значения нормального распределения в список  
sort\_normal\_1000 = sorted(normal\_1000)

На одном графике покажем функции нормального распределения, полученные из выборок разного размера и теоретическую.

plt.figure(dpi=100)   
plt.plot(theory\_normal\_x, theory\_normal\_y, lw=2, label='theoretical')  
plt.plot(sort\_normal\_100, np.arange(0.,1.,0.01), lw=2, label='size=100', color='green')   
plt.plot(sort\_normal\_1000, np.arange(0.,1.,0.001), lw=2, label='size=1000', color='red')   
plt.title('Normal')   
plt.legend()

<matplotlib.legend.Legend at 0x2e5676d4790>



png

## Задание 1.5

Построим плотность распределения случайной величины на основе выборки (наодном графике покажем плотности распределения, полученные из выборок разного размера и теоретическую)

Функции плотности распределения для экспоненциального распределения.

def plot\_exp(l, x):  
 return l\*np.exp(-l\*x)

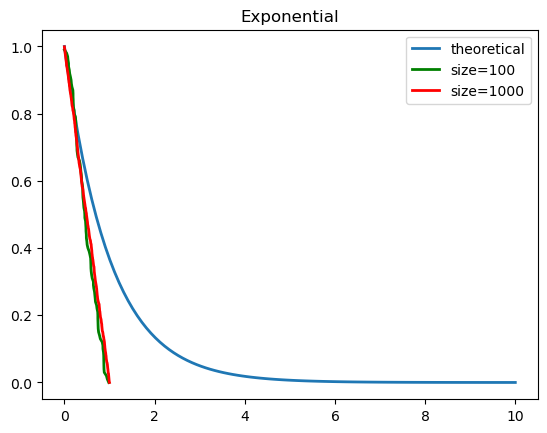
Задаем значения для х и с ними строим плотность распределения.

from scipy.stats import expon  
theoryPL\_exp\_x = np.linspace(0, 10, 100)  
theoryPL\_exp\_y = plot\_exp(1, theoryPL\_exp\_x)  
PL\_exp\_100 = expon.pdf(exp\_100, loc=0, scale=1)  
PL\_exp\_1000 = expon.pdf(exp\_1000, loc=0, scale=1)

На одном графике покажем плотность экспоненциального распределения, полученные из выборок разного размера и теоретическую.

plt.figure(dpi=100)   
plt.plot(theoryPL\_exp\_x, theoryPL\_exp\_y, lw=2, label='theoretical')  
plt.plot(PL\_exp\_100, np.arange(0.,1.,0.01), lw=2, label='size=100', color='green')   
plt.plot(PL\_exp\_1000, np.arange(0.,1.,0.001), lw=2, label='size=1000', color='red')   
plt.title('Exponential')   
plt.legend()

<matplotlib.legend.Legend at 0x2e56ab2d7d0>



png

Функции плотности распределения для нормального распределения.

def plot\_normal(m, sigma, x):  
 return (1 / (sigma\*np.sqrt(2\*np.pi)))\*np.exp(-((x-m)\*\*2)/2\*sigma\*\*2)

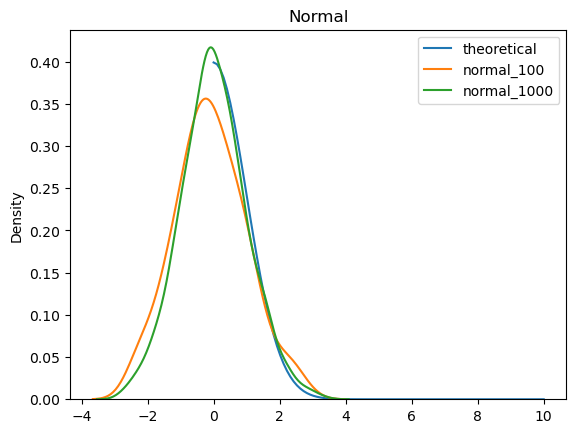
Задаем значения для х и с ними строим плотность распределения.

theoryPL\_normal\_x = np.linspace(0, 10, 100)  
theoryPL\_normal\_y = plot\_normal(0, 1, theoryPL\_normal\_x)

На одном графике покажем плотность нормального распределения, полученные из выборок разного размера и теоретическую.

import seaborn as sb  
plt.figure(dpi=100)   
plt.plot(theoryPL\_normal\_x, theoryPL\_normal\_y, label='theoretical')  
sb.kdeplot(normal\_100, label = 'normal\_100')   
sb.kdeplot(normal\_1000, label = 'normal\_1000')   
plt.title('Normal')   
plt.legend()

<matplotlib.legend.Legend at 0x2e56aaa76d0>



png

## Задание 2.

Напишем формулу для нахождения Евклидового расстаяния, для дальнейшего анализа распределения случайных расстояний между точками.

def dist(x1,y1,x2,y2):   
 return np.sqrt((x2-x1)\*\*2+(y2-y1)\*\*2)

Создаем функцию в которой создаем пустой список, куда записываем значения рандомно сгенерированных расстояний.

a = 10   
b = 30   
  
def lenght(num):   
 d = []   
 for i in range(num):   
 x1 = np.random.uniform(0,a)  
 y1 = np.random.uniform(0,a)  
   
 x2 = np.random.uniform(0,b)   
 y2 = np.random.uniform(0,b)   
   
 d.append(dist (x1,y1, x2,y2))   
   
 return sorted(d)

lenght\_100 = lenght(100)   
lenght\_1000 = lenght(1000)   
lenght\_10000 = lenght(10000)

Находим среднее в сгенерированных значениях.

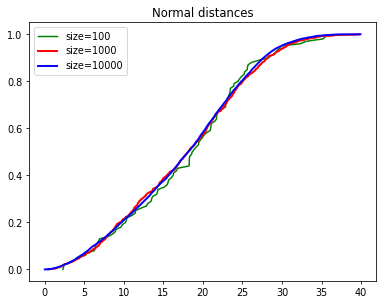
print('average for 100 =', np.average(lenght\_100))   
print('average for 1000 =' , np.average (lenght\_1000))   
print('average for 10000 =', np.average (lenght\_10000))

average for 100 = 17.762045179444247  
average for 1000 = 17.684931262488668  
average for 10000 = 17.565799729945937

Строим 3 графика с разным колическтвом значений.

plt.figure(dpi=70)  
plt.plot(lenght\_100,np.arange(0,1,0.01), lw=1.5,label='size=100',color="green")   
plt.plot(lenght\_1000,np.arange(0.,1.,0.001), lw=2, label='size=1000',color='red')   
plt.plot(lenght\_10000,np.arange(0.,1.,0.0001), lw=2, label='size=10000',color='blue')   
plt.title('Normal distances')   
plt.legend()

<matplotlib.legend.Legend at 0x2e56a9d29d0>

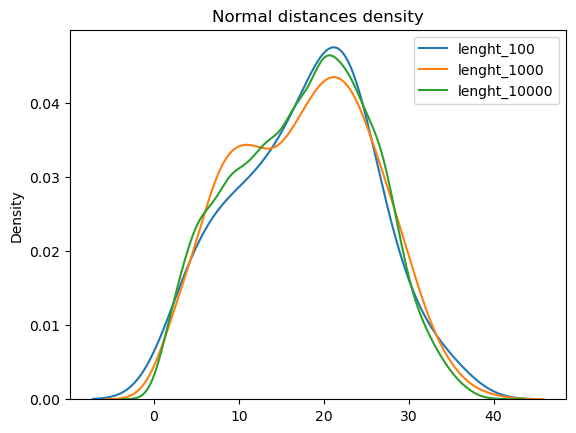


png

Строим график плотности вероятностей случайных расстояний.

plt.figure(dpi=100)   
sb.kdeplot(lenght\_100, label = 'lenght\_100')   
sb.kdeplot(lenght\_1000, label = 'lenght\_1000')  
sb.kdeplot(lenght\_10000, label = 'lenght\_10000')  
plt.title('Normal distances density')   
plt.legend()

<matplotlib.legend.Legend at 0x2e56c0e95d0>



png