МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

ЛАБОРАТОРНАЯ РАБОТА №1

По курсу «Теория вероятностей и математическая статистика»

Студент группы Б9123-01.03.02ии
Моттуева Уруйдана Михайловна

г. Владивосток
2025

Ход работы:
1. Краткое математическое описание методов поиска среднего.

1. Формула выборочного среднего:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

Где x_i – элементы выборки, n - размер выборки.

В Python реализация без использования NumPy:

```
def sample_mean(data):
    return sum(data) / len(data)
```

В Python реализация с использованием NumPy:

```
numpy_mean = np.mean(data)
```

2. Формула выборочной дисперсии:

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

Где $\bar{x}$ – выборочное среднее.

В Python реализация без использования NumPy:

```
def sample_variance(data):
    mean = sample_mean(data)
    return sum((x - mean) ** 2 for x in data) / len(data)
```

В Python реализация с использованием NumPy:

```
numpy_var = np.var(data, ddof=0)
```

3. Формула выборочного стандартного отклонения:

$$\sigma = \sqrt{\sigma^2},$$

Где σ^2 – выборочная дисперсия.

В Python реализация без использования NumPy:

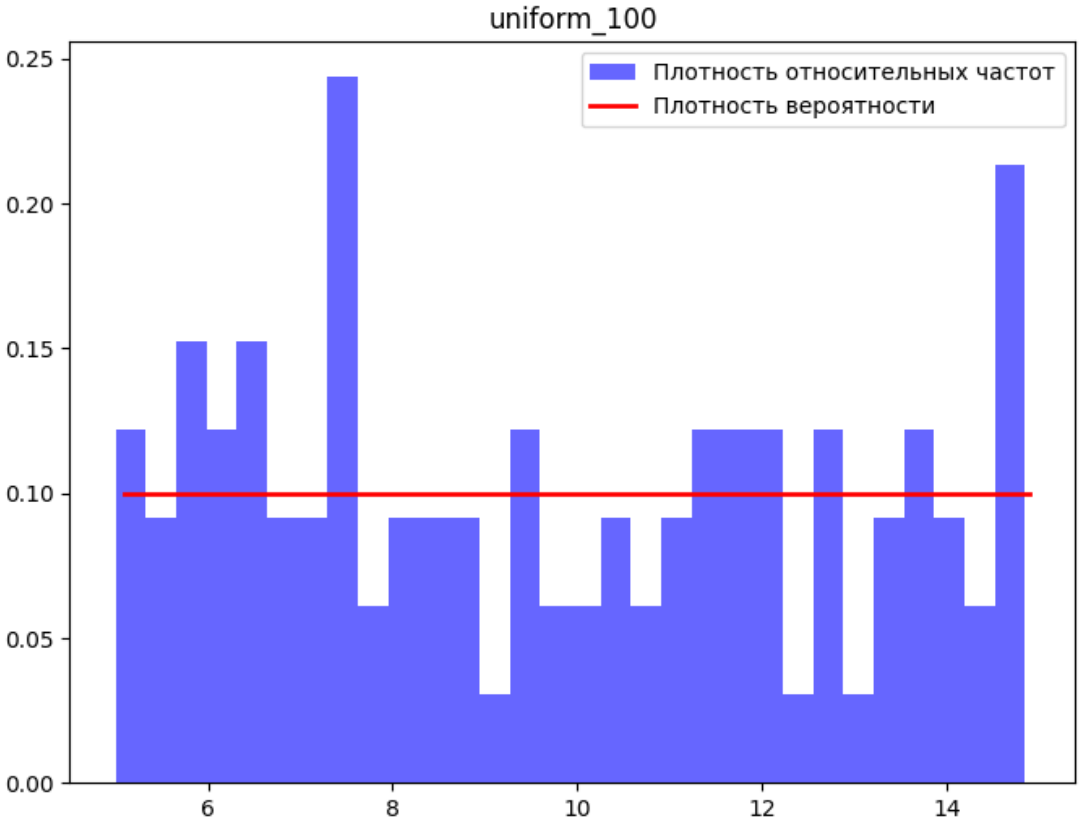
```
def sample_std_deviation(data):
    return sample_variance(data) ** 0.5
```

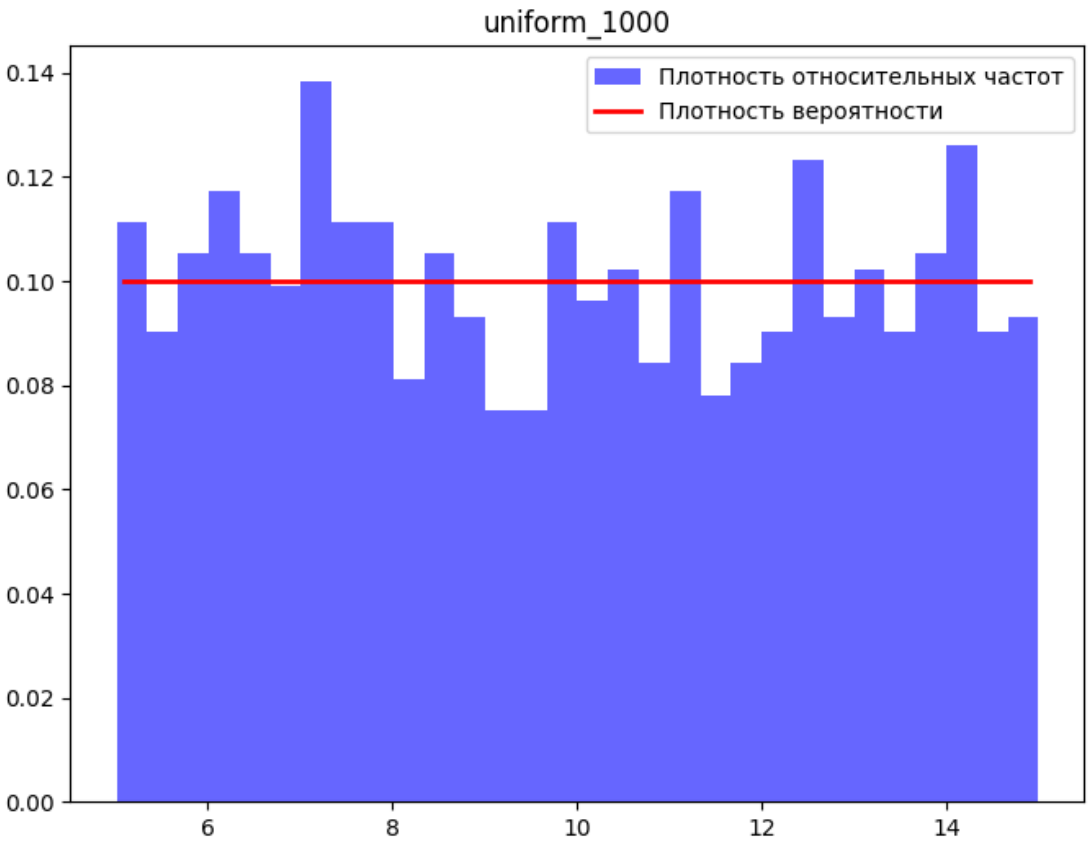
В Python реализация с использованием NumPy:

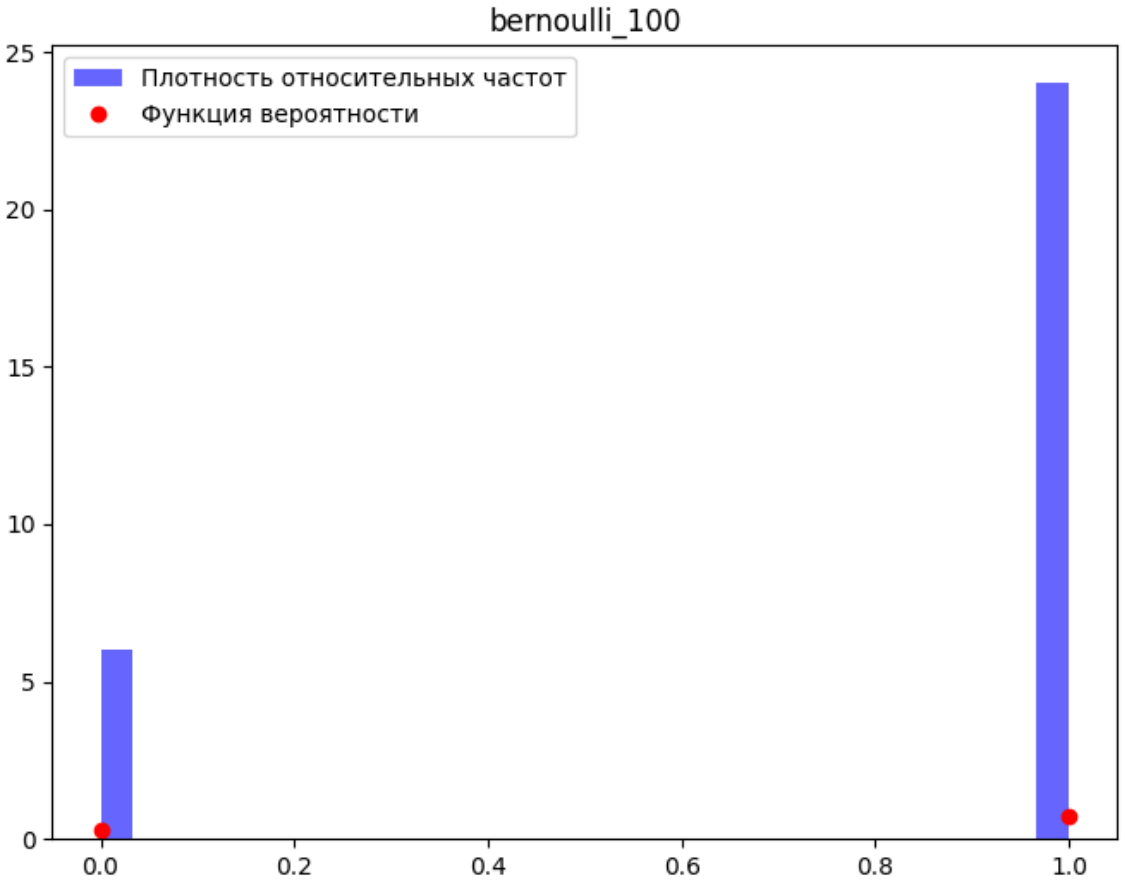
```
numpy_std = np.std(data, ddof=0)
```

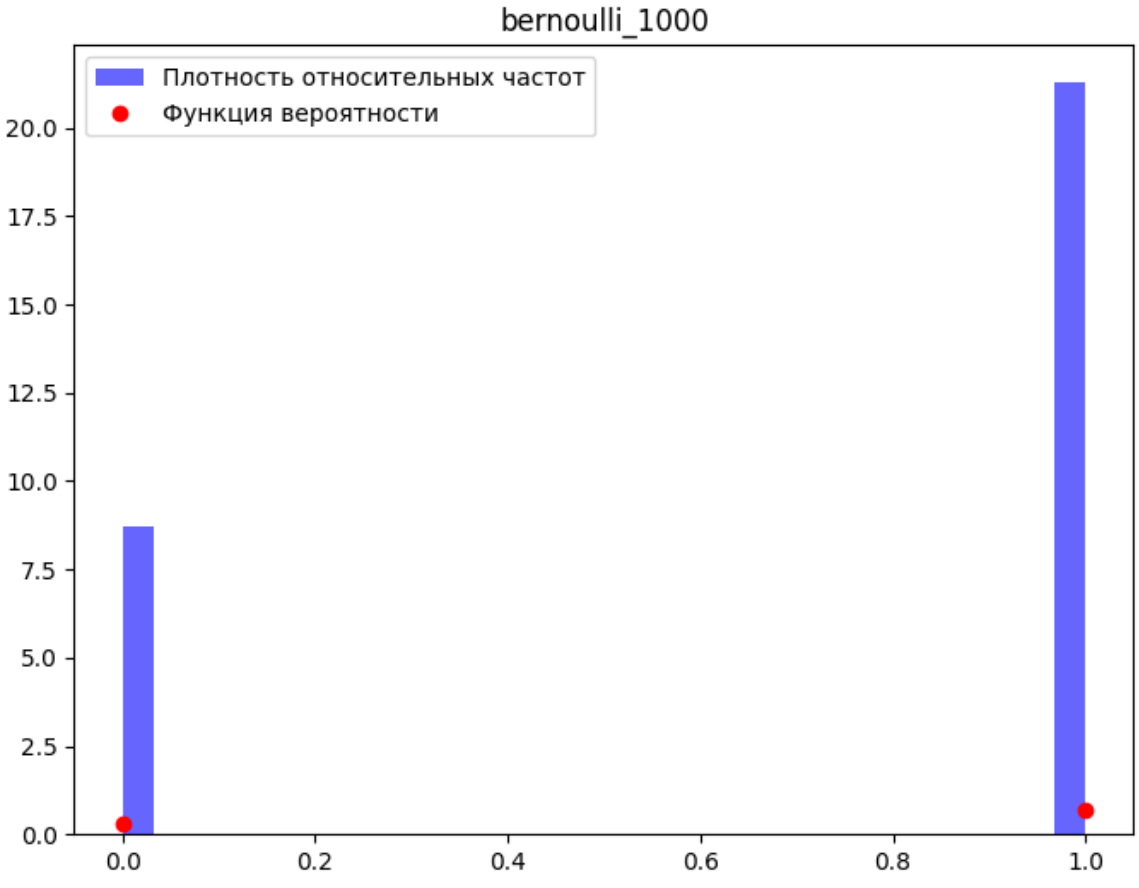
2. Таблица результатов:

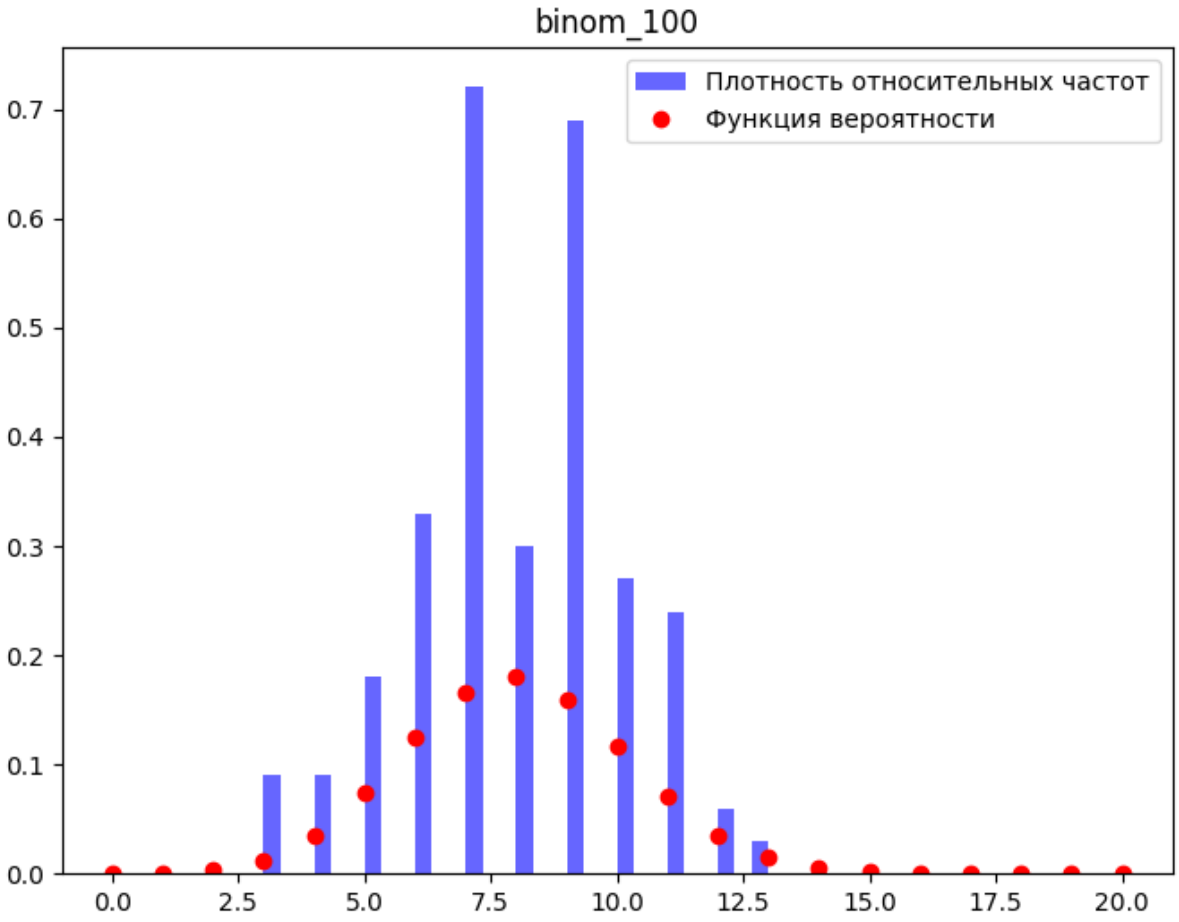
Выборка	Среднее	Дисперсия	Стандартное отклонение	Среднее numpy	Дисперсия numpy	Станд-ое отклонение numpy
uniform_100	9.751311	9.241274	3.039946	9.751311	9.241274	3.039946
uniform_1000	9.930836	8.580982	2.929331	9.930836	8.580982	2.929331
bernoulli_100	0.800000	0.160000	0.400000	0.800000	0.160000	0.400000
bernoulli_1000	0.709000	0.206319	0.454224	0.709000	0.206319	0.454224
binom_100	7.870000	4.313100	2.076800	7.870000	4.313100	2.076800
binom_1000	8.125000	4.605375	2.146014	8.125000	4.605375	2.146014
normal_100	14.675226	9.239656	3.039680	14.675226	9.239656	3.039680
normal_1000	14.967388	8.857275	2.976117	14.967388	8.857275	2.976117

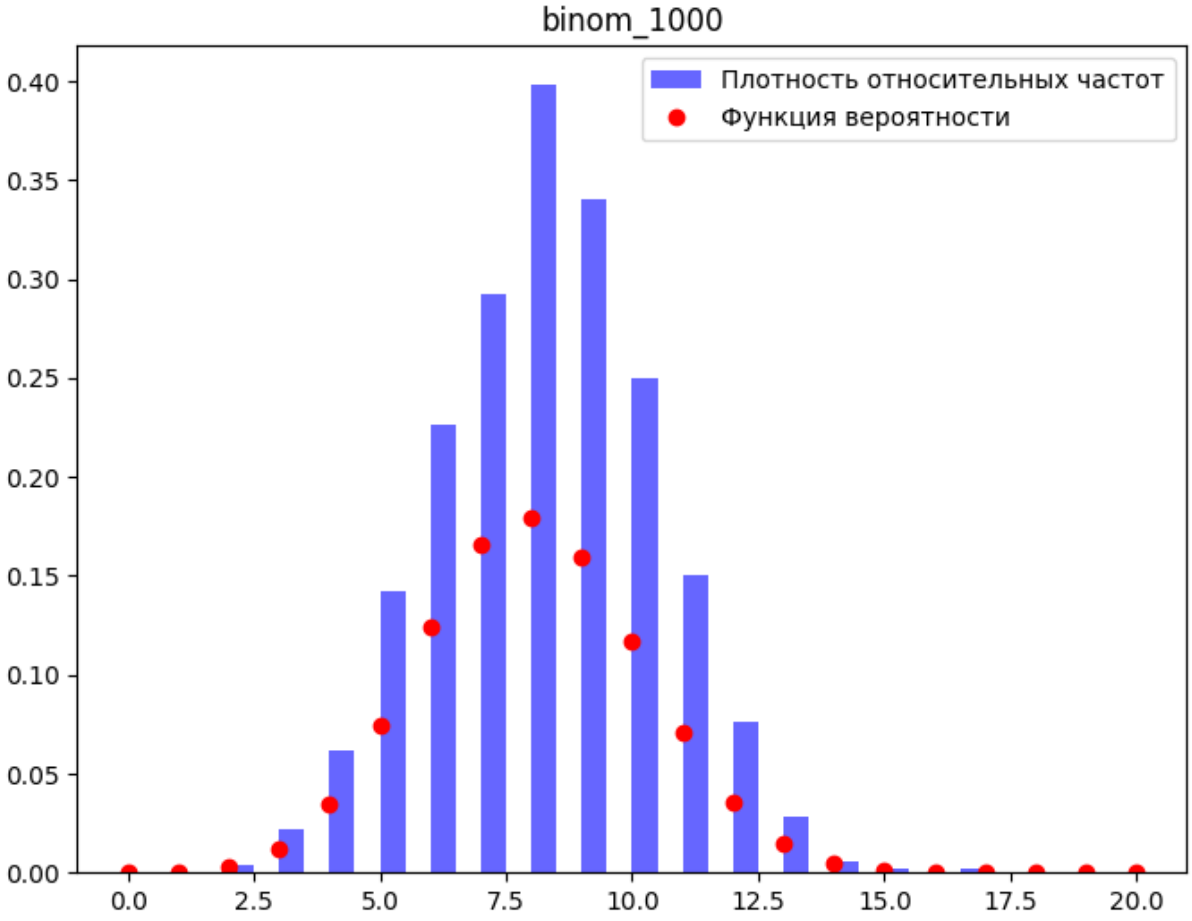

uniform_100
Плотность относительных частот
Плотность вероятности
0.25
0.20
0.15
0.10
0.05
0.00
6
8
10
12
14

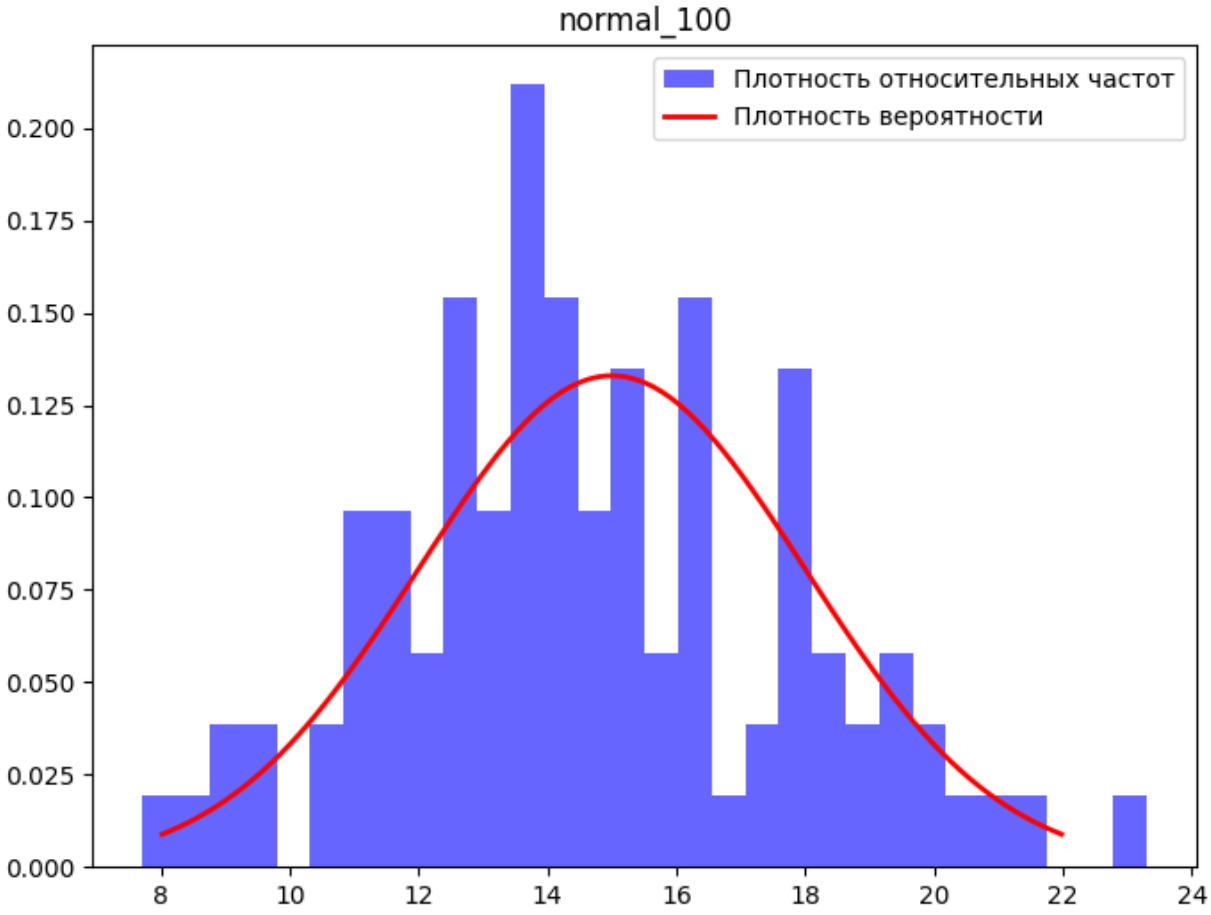

uniform_1000
Плотность относительных частот
Плотность вероятности
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
6
8
10
12
14

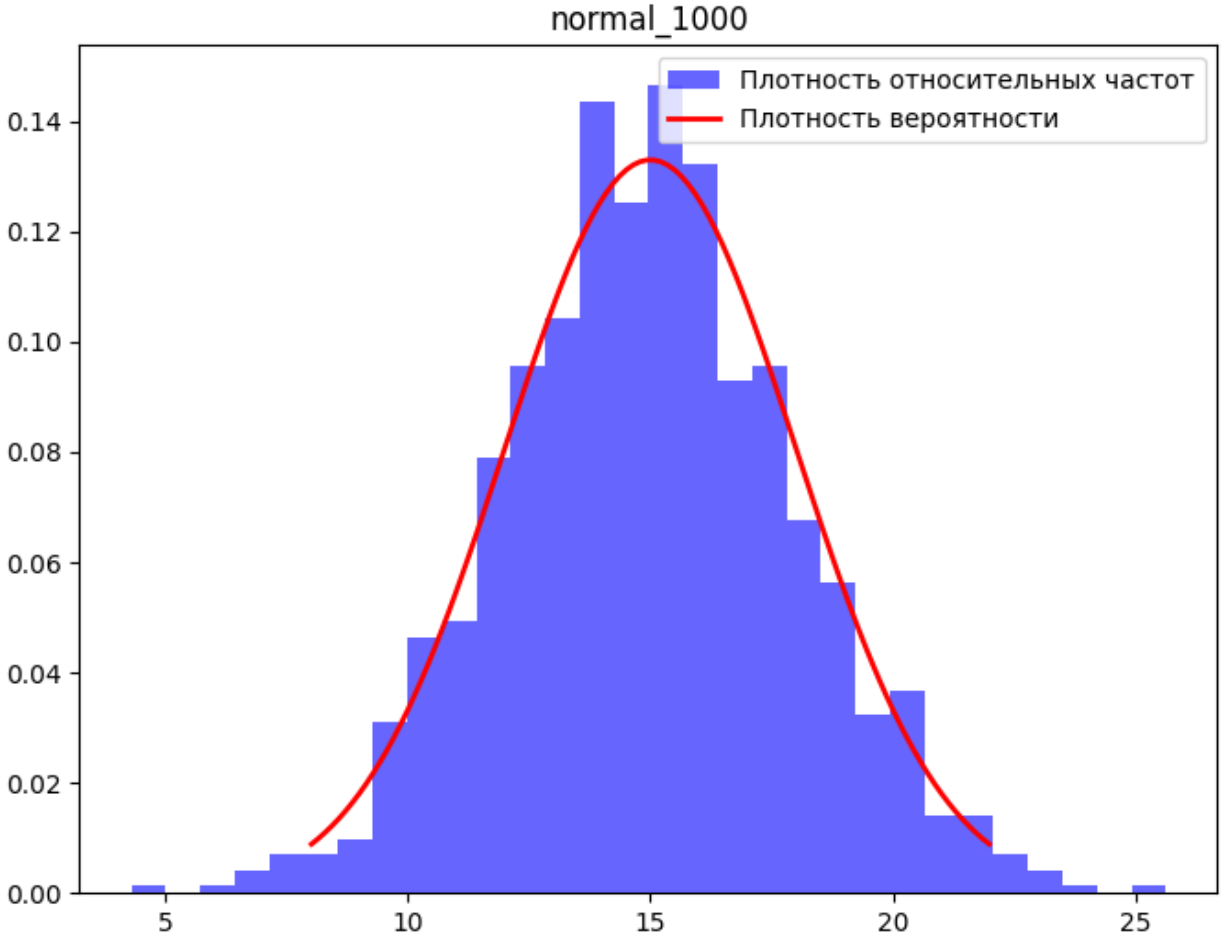

bernoulli_100
Плотность относительных частот
Функция вероятности
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0


bernoulli_1000
Плотность относительных частот
Функция вероятности
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


binom_100
Плотность относительных частот
Функция вероятности
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


binom_1000
Плотность относительных частот
Функция вероятности
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


normal_100
Плотность относительных частот
Плотность вероятности
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
8
10
12
14
16
18
20
22
24


normal_1000
Плотность относительных частот
Плотность вероятности
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
5
10
15
20
25

4. Полный код программы

```
from scipy.stats import uniform, bernoulli, binom, norm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(13)

samples = {
    "uniform_100": uniform(loc=5, scale=10).rvs(100),
    "uniform_1000": uniform(loc=5, scale=10).rvs(1000),
    "bernoulli_100": bernoulli(p=0.7).rvs(100),
    "bernoulli_1000": bernoulli(p=0.7).rvs(1000),
    "binom_100": binom(n=20, p=0.4).rvs(100),
    "binom_1000": binom(n=20, p=0.4).rvs(1000),
    "normal_100": norm(loc=15, scale=3).rvs(100),
    "normal_1000": norm(loc=15, scale=3).rvs(1000),
}

def sample_mean(data):
    return sum(data) / len(data)

def sample_variance(data):
    mean = sample_mean(data)
    return sum((x - mean) ** 2 for x in data) / len(data)

def sample_std_deviation(data):
    return sample_variance(data) ** 0.5

# Вычисление статистик для каждой выборки
results = []
for name, data in samples.items():
    custom_mean = sample_mean(data)
    custom_var = sample_variance(data)
    custom_std = sample_std_deviation(data)

    numpy_mean = np.mean(data)
    numpy_var = np.var(data, ddof=0)
    numpy_std = np.std(data, ddof=0)

    results.append([name, custom_mean, custom_var, custom_std,
numpy_mean, numpy_var, numpy_std])

columns = ["№ выборки", "Среднее своё", "Дисперсия своя", "Стандартное
отклонение своё",
           "Среднее numpy", "Дисперсия numpy", "Стандартное отклонение
numpy"]
df = pd.DataFrame(results, columns=columns)

print(df)
```

```
# Построение графиков для каждой выборки
for name, data in samples.items():
    plt.figure(figsize=(8, 6))

    # Гистограмма плотностей относительных частот
    counts, bins, _ = plt.hist(data, bins=30, density=True, alpha=0.6,
color='b', label="Плотность относительных частот")

    # Плотность вероятности/функция вероятности
    if "uniform" in name:
        dist = uniform(loc=5, scale=10)
        x = np.linspace(dist.ppf(0.01), dist.ppf(0.99), 100)
        plt.plot(x, dist.pdf(x), 'r-', lw=2, label="Плотность
вероятности")
    elif "bernoulli" in name:
        dist = bernoulli(p=0.7)
        x = [0, 1]
        plt.plot(x, dist.pmf(x), 'ro', lw=2, label="Функция вероятности")
    elif "binom" in name:
        dist = binom(n=20, p=0.4)
        x = np.arange(0, 21)
        plt.plot(x, dist.pmf(x), 'ro', lw=2, label="Функция вероятности")
    elif "normal" in name:
        dist = norm(loc=15, scale=3)
        x = np.linspace(dist.ppf(0.01), dist.ppf(0.99), 100)
        plt.plot(x, dist.pdf(x), 'r-', lw=2, label="Плотность
вероятности")

    plt.title(name)
    plt.legend()
    plt.show()
```

https://colab.research.google.com/drive/1uyNtb7ice-Y5EmwiXLrZqfE62t-BAUrv?usp=sharing – ссылка на колаб с полным кодом программы