МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

МАТЕМАТИЧЕСКАЯ СТАТИСТИКА И СЛУЧАЙНЫЕ ПРОЦЕССЫ

ЛАБОРАТОРНАЯ РАБОТА 1

Тема: Выборка

Студент
Гузовская Александра Чеславовна
группы Б9123-01.03.02сп

Преподаватель Деревягин А. А.

Регистрационный № _____ Оценка_____

_____ _____ _____ _____

(подпись) (И. О. Фамилия) (подпись) (И. О. Фамилия)

«___» _____ 2025 г. «___» _____ 2025 г.

г. Владивосток
2025

Лабораторная работа 1

Гузовская Александра Чеславовна

март 2025

1 Описание используемых методов

1.1 Равномерное распределение

Если случайная величина обладает постоянной плотностью распределения вероятностей на данном отрезке и нулевой плотностью вне него, то она распределена равномерно с плотностью:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{если } x \in [a, b] \\ 0 & \text{если } x \notin [a, b] \end{cases}$$

Реализация:

1) задаём начальные значения: начало и конец интервала, требуемое количество значений, после создаём массив возможных значений:

import numpy as np
a = 0
b = 10
n = 100
x = np.linspace(a, b, n)

2) считаем вероятности с помощью .uniform.pdf():

import scipy.stats as stats probabilities = stats.uniform.pdf(x, loc=a, scale=b-a)
где x массив значений, для которых считаем плотность вероятности, loc

задает нижний предел распределения, scale задает ширину распределения, что соответствует длине интервала, на котором распределение равномерно

1.2 Распределение Бернулли

Частный случай биномиального распределения, где n=1
Если случайная величина принимает всего два значения 1, 0 с вероятностями p и $q = n - p$ соответственно, иными словами успех или неудача, то говорят о распределении Бернулли
Плотности соответственно $P(\xi = 1) = p$ и $P(\xi = 0) = q$

Реализация:

1) задаём вероятность успеха, создаём массив x = [0, 1] всех возможных значений распределения, считаем вероятности с помощью .bernoulli.pmf():

import numpy as np
import scipy.stats as stats
p = 0.7
x = [0, 1] (возможные исходы, где 0 - неудача, 1 - успех)
probabilities = stats.bernoulli.pmf(x, p)

1.3 Биномиальное распределение

Это распределение вероятностей для фиксированного числа независимых испытаний Бернулли (n), где каждый раз вероятность успеха (p) остается постоянной
Наиболее распространенный вид дискретного распределения, вероятности определяются формулой Бернулли и представляют собой члены Бинома Ньютона

$$p_i = C_n^{x_i} \cdot p^{x_i} \cdot q^{n-x_i}$$

где n независимых событий, вероятность появления события p и вероятность непоявления q
В сумме вероятности равны единице:

$$\sum_{x_i=0}^{n} C_n^{x_i} \cdot p^{x_i} \cdot q^{n-x_i} = (q+p)^n = 1^n = 1$$

Выходит, плотность вероятности:

$$p_\xi(k) = P(\xi = k) = C_n^k \cdot p^k \cdot q^{n-k}$$

Реализация:

1) создаём массив целых чисел от 1 до n:

import numpy as np
n = 100 (задаём количество испытаний)
$x = np.arange(0, n + 1)$

2) после вычисляем вероятность для каждого значения в массиве x с использованием функции вероятности массы (PMF) биномиального распределения:

import scipy.stats as stats
p = 0.5 (задаём вероятность для каждого испытания)
$probabilities = stats.binom.pmf(x, n, p)$

1.4 Нормальное распределение

Нормально распределённые величины встречаются в природе чаще равномерно распределённых, функция плотности вероятности имеет вид кривой, резко возрастающей и резко убывающей около средних значений признака. В формульном представлении имеет вид:

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{(x-a)^2}{2\sigma^2}}$$

где σ есть среднеквадратическое отклонение (или корень из дисперсии), a - параметр

Реализация:

1) генерация случайных чисел из нормального распределения .norm.rvs()

import scipy.stats as stats

$data = stats.norm.rvs(loc = \mu, scale = \sigma, size = 1000, random\_state = 10)$
где loc= среднее, scale= стандартное отклонение, size= требуемый размер выборки, random_state= начальное значение для псевдослучайных чисел

2) вычисление плотности вероятности нормального распределения .norm.pdf()

import numpy as np
$x = np.linspace(\mu - 4 \cdot \sigma, mu + 4 \cdot \sigma, 1000)$
$pdf = stats.norm.pdf(x, loc = \mu, scale = \sigma)$
где $\mu - 4 \cdot \sigma, mu + 4 \cdot \sigma$ есть диапазон, на котором ищем плотность

2 Результаты: графики, вывод в терминале

Получили вывод вида

```
Results:
        Sample     Mean     Variance     Standart Deviation
```

для кастомной реализации и библиотечной реализации
при заданных выборках
из распределений равномерного, Бернулли, биномиального, нормального

ddof (delta degrees of freedom) используется для смещённой и несмещённой оценки в дисперсии и стандартного отклонения
Степени свободы — это количество независимых значений, которые могут варьироваться в расчетах статистических показателей.

Results:

	Sample	Mean (Custom)	Variance (Custom)	Standart Dev (Custom)	Mean (Numpy)	Variance (Numpy)	Standart Deviation (Numpy)
0	U_100	6.355189	3.902969	1.975593	6.355189	3.902969	1.975593
1	U_1000	6.521198	3.833480	1.957927	6.521198	3.833480	1.957927
2	Bernoulli_100	0.170000	0.142525	0.377525	0.170000	0.142525	0.377525
3	Bernoulli_1000	0.191000	0.154674	0.393286	0.191000	0.154674	0.393286
4	Binom_100	5.880000	3.642020	1.908408	5.880000	3.642020	1.908408
5	Binom_1000	5.996000	3.745730	1.935389	5.996000	3.745730	1.935389
6	Norm_100	14.948583	17.656730	4.201991	14.948583	17.656730	4.201991
7	Norm_1000	15.126295	16.642552	4.079528	15.126295	16.642552	4.079528

Рис. 1: Вывод в терминале

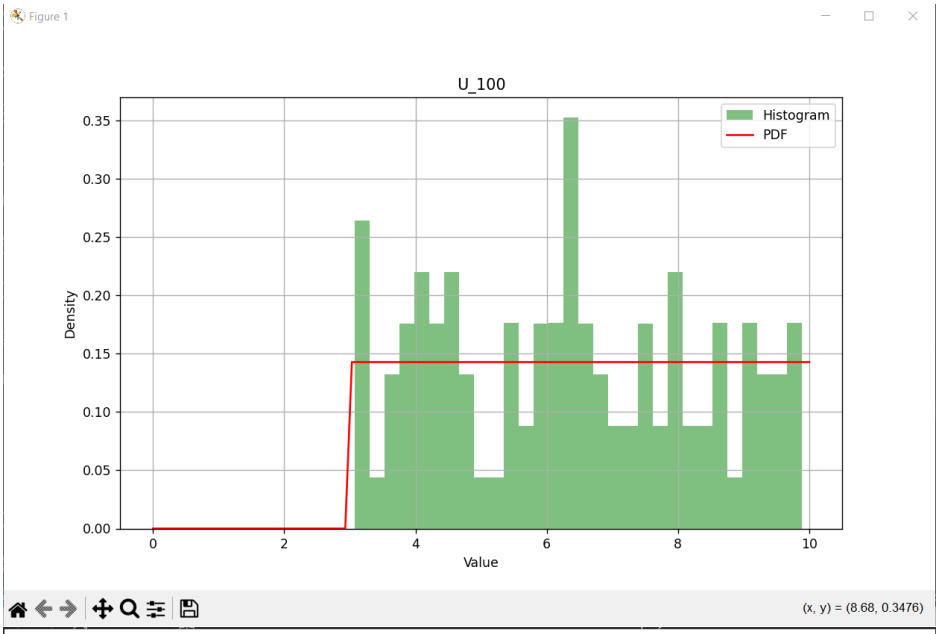


Рис. 2: Равномерное распределение для 100 и 1000

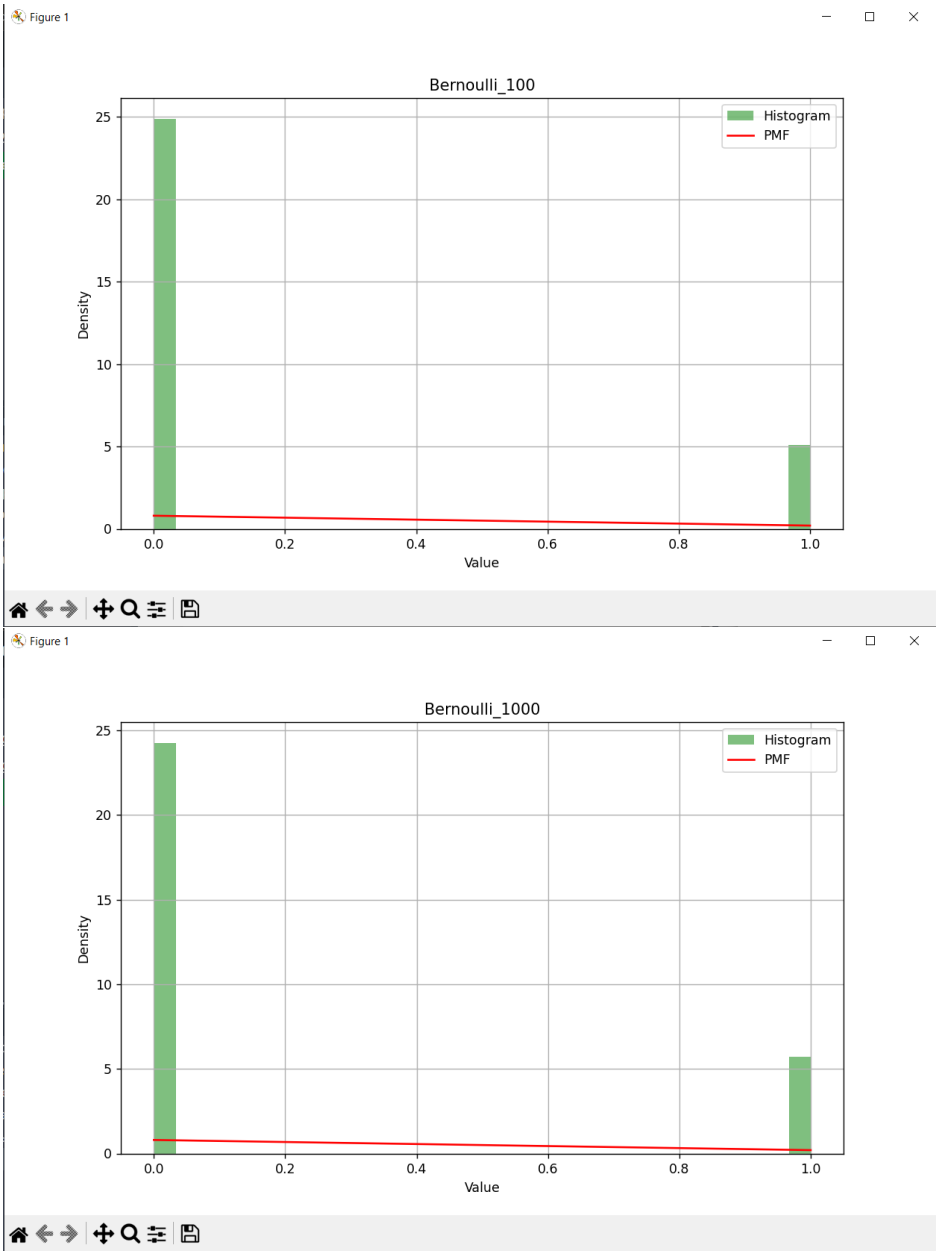
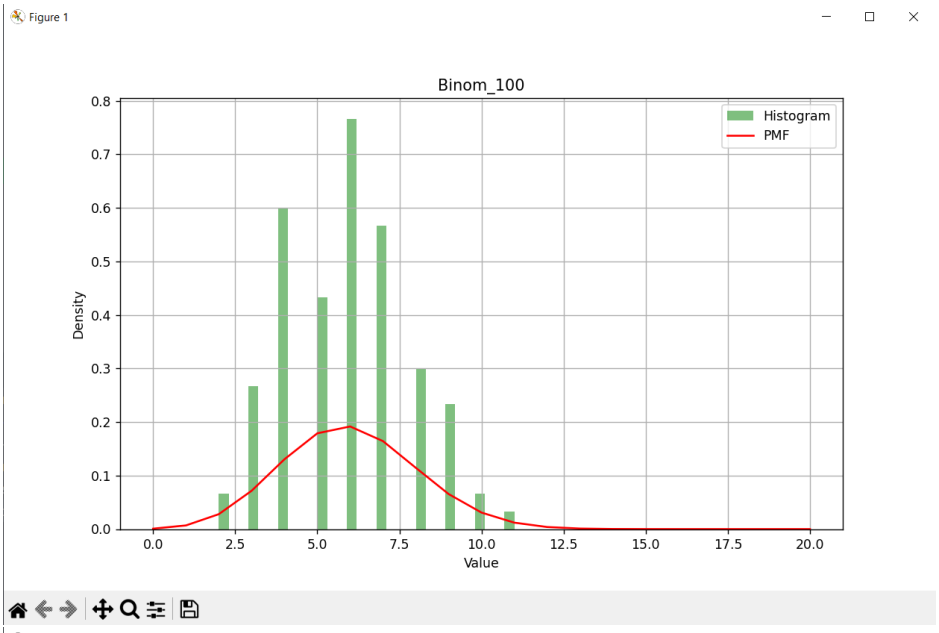


Рис. 3: Распределение Бернулли для 100 и 1000



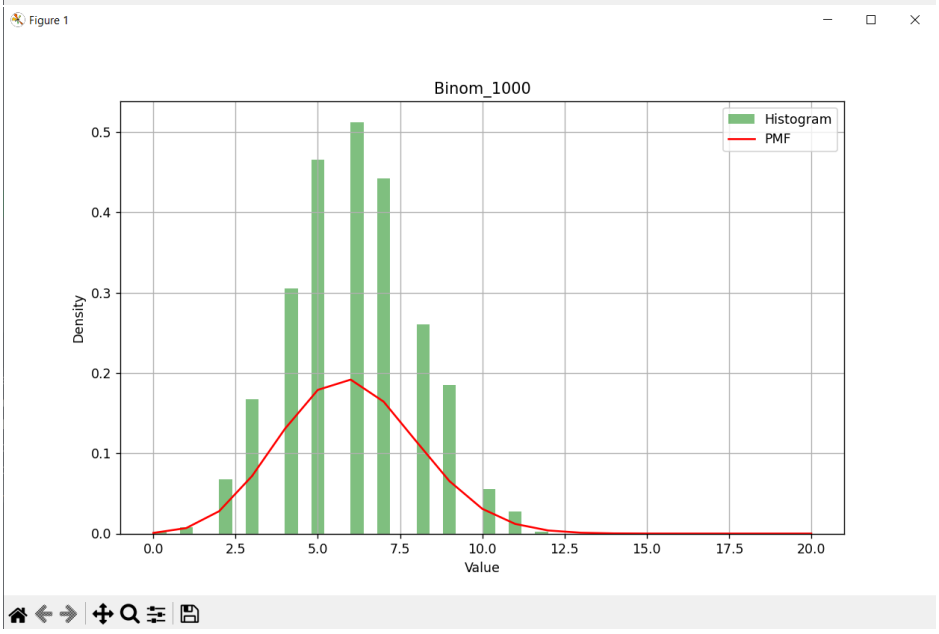


Рис. 4: Биномиальное распределение для 100

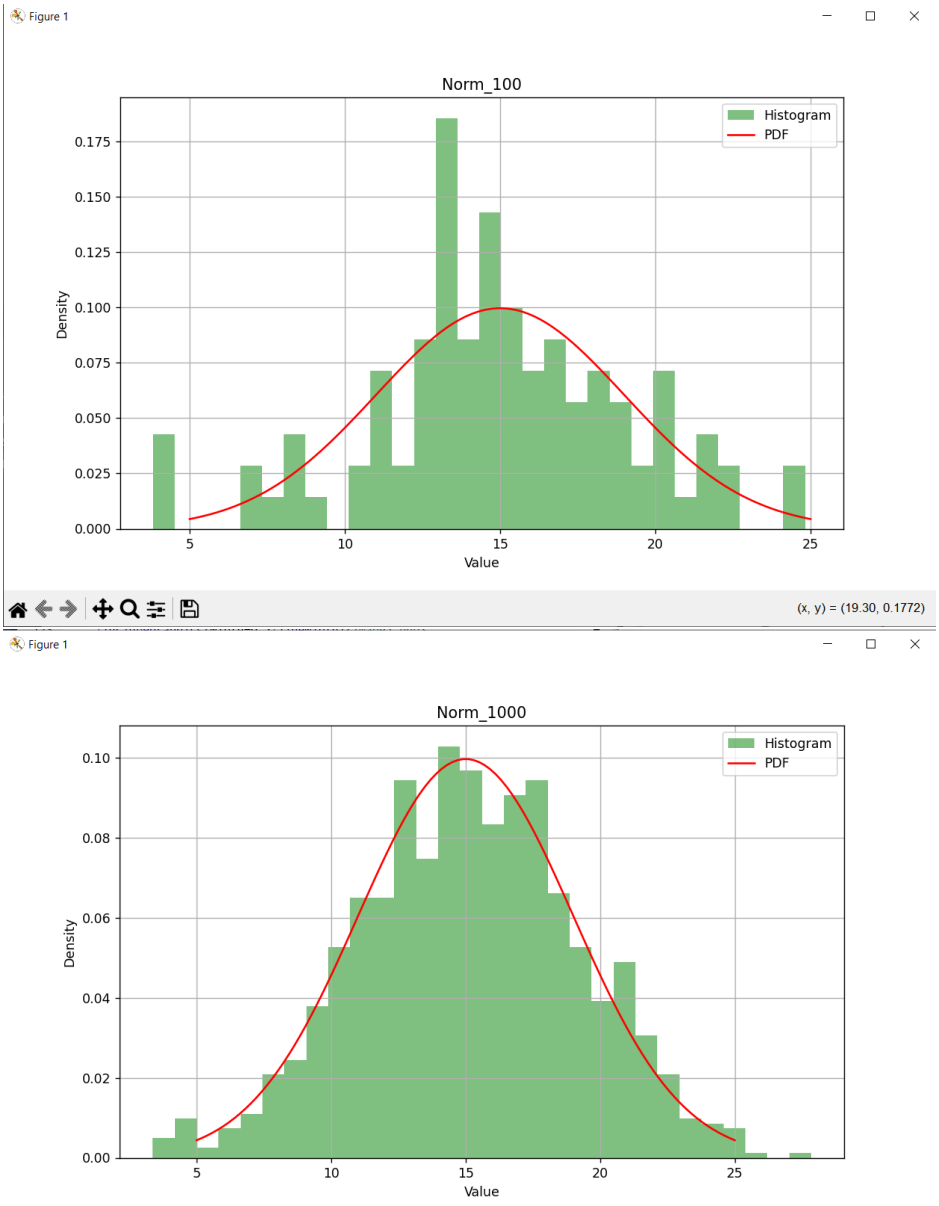


Рис. 5: Нормальное распределение для 100 и 1000

3 Код программы

```
import numpy as np
import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt

random_state = 9

# Выборки из требуемых распределений
samples_ = {
    'U_100': stats.uniform.rvs(loc=3, scale=7, size=100, random_state=random_state
    'U_1000': stats.uniform.rvs(loc=3, scale=7, size=1000, random_state=random_sta
    'Bernoulli_100': stats.bernoulli.rvs(p=0.2, size=100, random_state=random_stat
    'Bernoulli_1000': stats.bernoulli.rvs(p=0.2, size=1000, random_state=random_st
    'Binom_100': stats.binom.rvs(n=20, p=0.3, size=100, random_state=random_state)
    'Binom_1000': stats.binom.rvs(n=20, p=0.3, size=1000, random_state=random_stat
    'Norm_100': stats.norm.rvs(loc=15, scale=4, size=100, random_state=random_stat
    'Norm_1000': stats.norm.rvs(loc=15, scale=4, size=1000, random_state=random_st
}

# Функция для выборочной средней
def sampleAverage(data):
    return sum(data) / len(data)

# Функция для дисперсии для несмещённой оценки
def varianceDef(data):
    avg_ = sampleAverage(data)
    variance_ = sum((x - avg_) ** 2 for x in data) / (len(data) - 1)
    return variance_

# Функция для дисперсии для смещённой оценки
def varianceDef2(data):
    avg_ = sampleAverage(data)
    variance_ = sum((x - avg_) ** 2 for x in data) / (len(data))
    return variance_

# Функция для стандартного отклонения
```

```
def standartDeviation(data):
    return varianceDef(data) ** 0.5

results_ = []

for name, data in samples_.items():
    mean_cust_ = sampleAverage(data)
    var_cust_ = varianceDef(data)
    std_cust_ = standartDeviation(data)

    mean_numpy = np.mean(data)
    var_numpy = np.var(data, ddof=1)  # Используем ddof=1 для несмещенной оценки
    std_numpy = np.std(data, ddof=1)

    results_.append({
        'Sample': name,
        'Mean (Custom)': mean_cust_,
        'Variance (Custom)': var_cust_,
        'Standart Dev (Custom)': std_cust_,
        'Mean (Numpy)': mean_numpy,
        'Variance (Numpy)': var_numpy,
        'Standart Deviation (Numpy)': std_numpy,
    })

results_dataframe_ = pd.DataFrame(results_)

print("Results:")
print(results_dataframe_)

# Построение графиков
for name, data in samples_.items():
    plt.figure(figsize=(10, 6))

    plt.hist(data, bins=30, density=True, alpha=0.5, color='g', label='Histogram')

    # Плотность вероятности
    if 'U' in name:
        x = np.linspace(0, 10, 100)
```

```
    plt.plot(x, stats.uniform.pdf(x, loc=3, scale=7), 'r-', label='PDF')
elif 'Bernoulli' in name:
    x = [0, 1]
    plt.plot(x, stats.bernoulli.pmf(x, p=0.2), 'r-', label='PMF')
elif 'Binom' in name:
    x = np.arange(0, 21)  # n=20
    plt.plot(x, stats.binom.pmf(x, n=20, p=0.3), 'r-', label='PMF')
elif 'Norm' in name:
    x = np.linspace(5, 25, 100)
    plt.plot(x, stats.norm.pdf(x, loc=15, scale=4), 'r-', label='PDF')

plt.title(name)
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.grid()
plt.show()
```