

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:

Где – 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)

1. Формула выборочной дисперсии:

Где – 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)

1. Формула выборочного стандартного отклонения:

Где – выборочная дисперсия.

В Python реализация без использования NumPy:

def sample\_std\_deviation(data):

return sample\_variance(data) \*\* 0.5

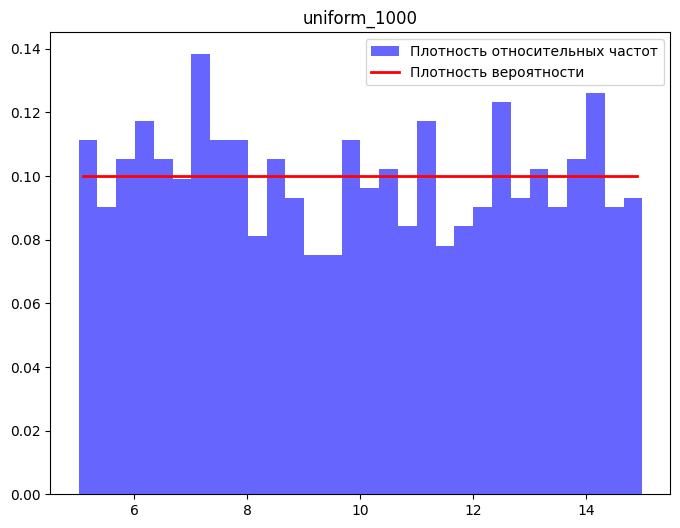
В Python реализация с использованием NumPy:

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 |

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, дисплей, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, График, линия

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, диаграмма, График, линия

Контент, сгенерированный ИИ, может содержать ошибки.

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

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)

for name, data in samples.items():

    plt.figure(figsize=(8, 6))

    counts, bins, \_ = plt.hist(data, bins=30, density=True, alpha=0.7, color='b', label="Плотность относительных частот")

    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