Министерство науки и высшего образования Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчет по лабораторной работе №1

“ПОЛУЧЕНИЕ БАЗОВОЙ ПОСЛЕДОВАТЕЛЬНОСТИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ С ПОМОЩЬЮ ПРОГРАММНОГО ДАТЧИКА И ТЕСТОВЫЕ ПРОВЕРКИ ЕГО РАБОТЫ”

по дисциплине «Статистическое моделирование»

Выполнил
студент гр. 5130904/10102 Тампио И.С.

Преподаватель
Чуркин В.В.

Санкт-Петербург 2024

Цели работы

1. Получение на ЭВМ с помощью программного датчика базовой последовательности псевдослучайных чисел, имеющих равномерное распределение.

2. Освоение методов статистической оценки полученного распределения: вычисление эмпирических значений для математического ожидания и дисперсии.

3. Освоение методов оценки статистики связи: вычисление значений автокорреляционной функции и построение коррелограммы.

4. Освоение методов графического представления законов распределения: построение функции плотности распределения и интегральной функции распределения.

Ход работы

1. Вычисление эмпирических значений математического ожидания и дисперсии полученной последовательности псевдослучайных чисел; сравнение полученных результатов с соответствующими теоретическими значениями.

n	mean	mean_expected	mean_deviation	std	std_expected	std_deviation
i64	f64	f64	f64	f64	f64	f64
10	0.400669	0.5	-0.099331	0.091236	0.08333	0.007906
100	0.487628	0.5	-0.012372	0.083903	0.08333	0.000573
1000	0.511267	0.5	0.011267	0.082624	0.08333	-0.000706
10000	0.498967	0.5	-0.001033	0.082536	0.08333	-0.000794
100000	0.499757	0.5	-0.000243	0.083728	0.08333	0.000398
1000000	0.499978	0.5	-0.000022	0.083225	0.08333	-0.000105

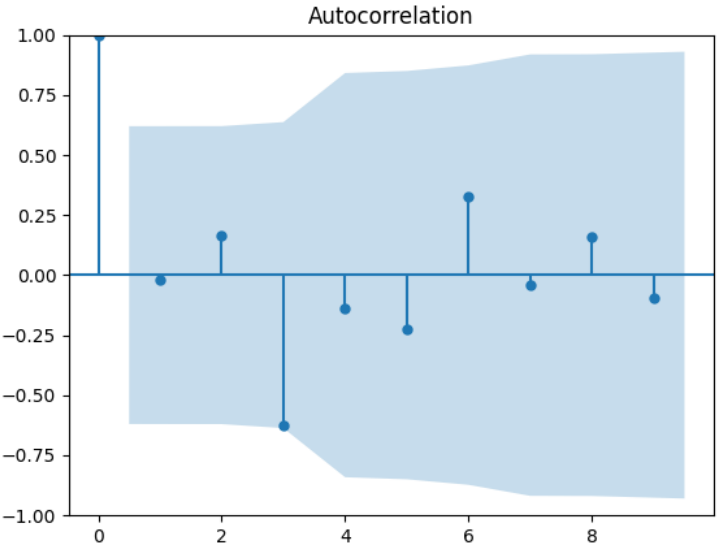
mean_deviation, std_deviation – отклонение экспериментального значения статистики от теоретического
mean_expected, std_expected – теоретические значения статистик

Как и ожидалось, даже на выборках малого размера программный генератор случайных чисел имеет небольшие отклонения от теоретических значений, что говорит о его корректности.

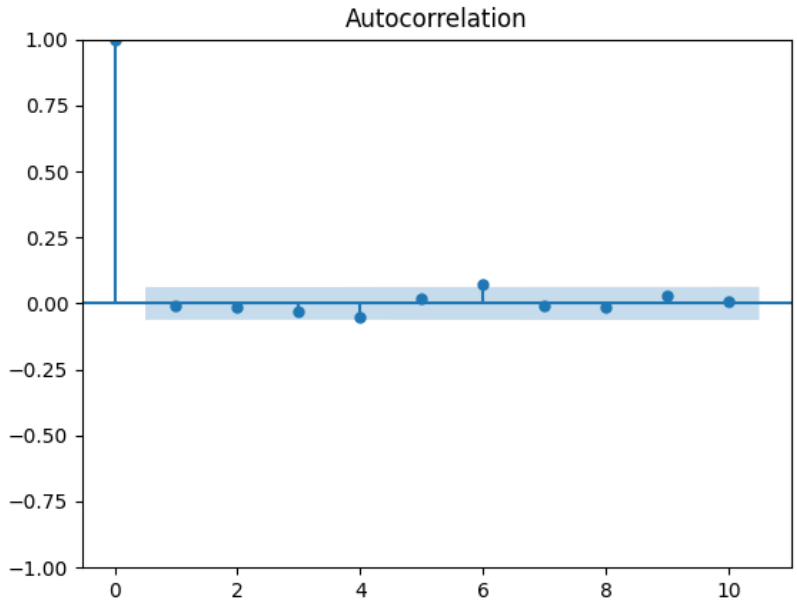
2. Вычисление значений автокорреляционной функции и построение коррелограммы.

Рассмотрим построенные для n = 10, 1000, 1000000 корреллограммы.

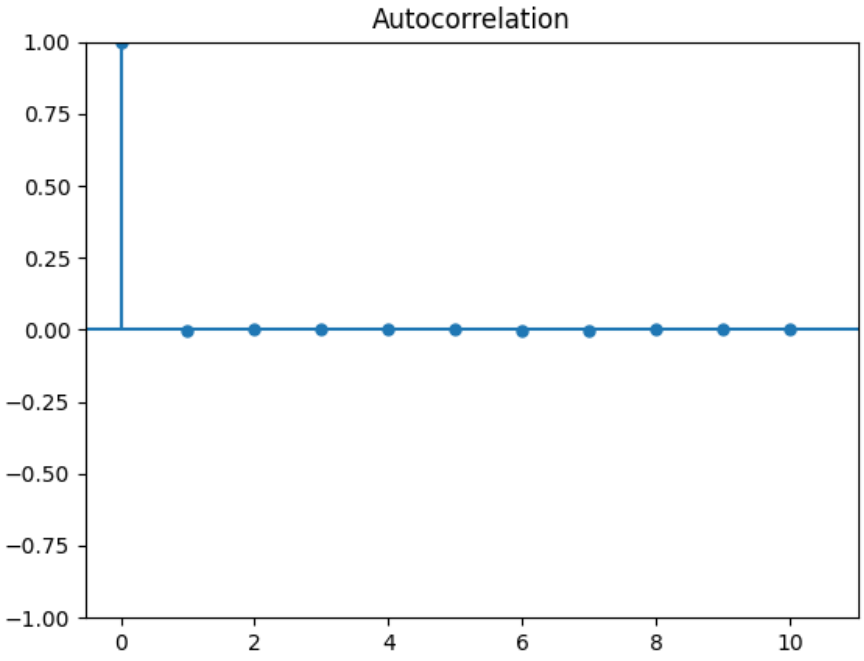
N = 10: Полученные значения корреллограммы лежат в пределах доверительного интервала, значительной автокоррелляции нет.



N = 1000: Полученные значения корреллограммы лежат в пределах доверительного интервала, возможно незначительное отклонение, но значительной автокоррелляции нет.

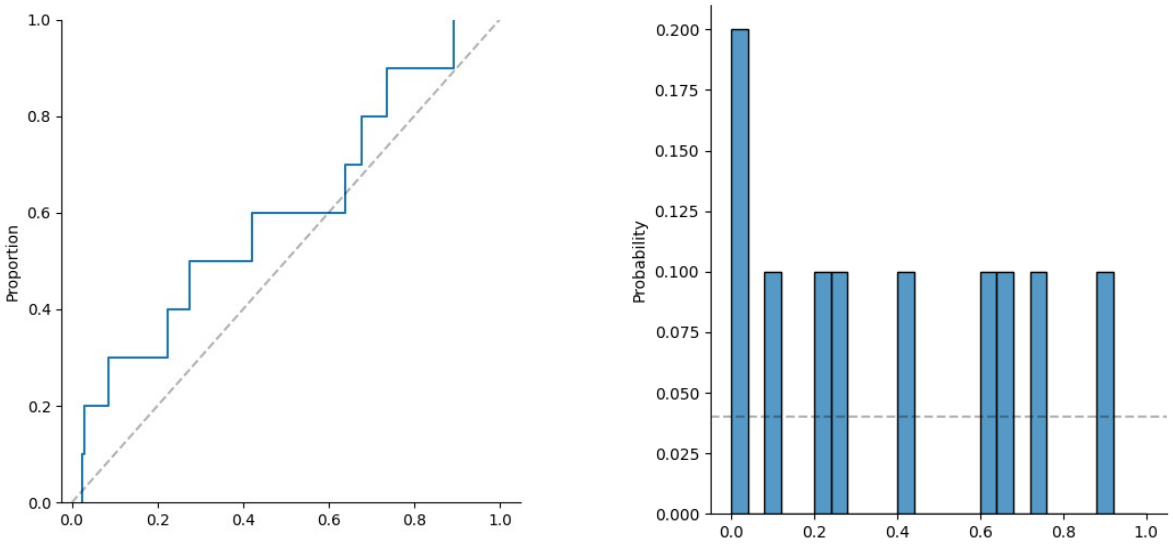


N = 1000000: Полученные значения корреллограммы лежат в пределах доверительного интервала, автокорреллляция отсутствует.

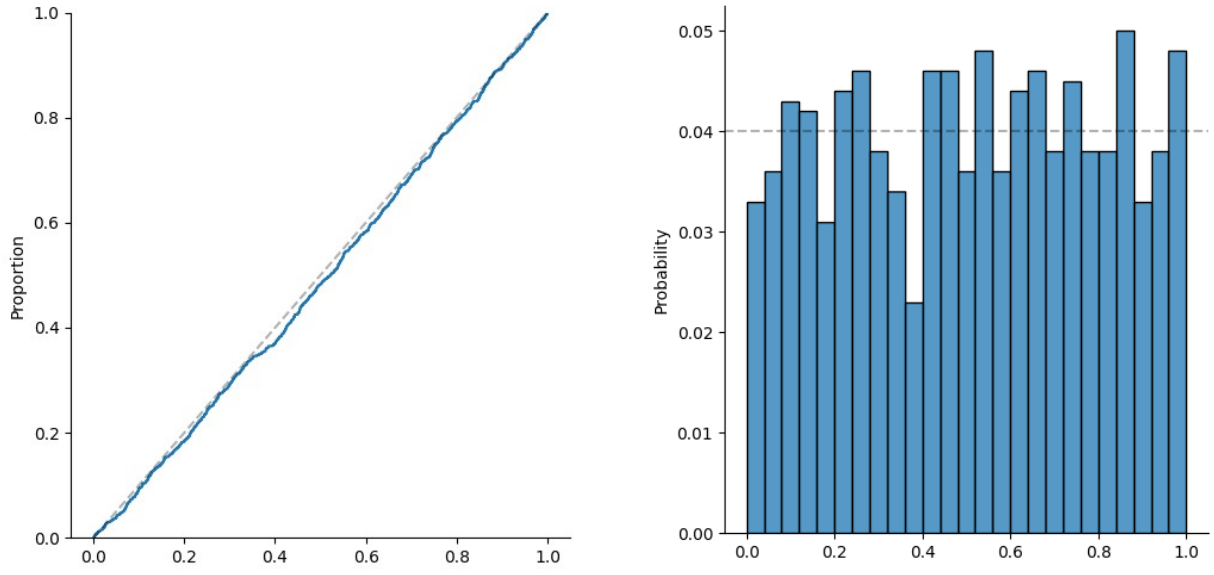


3. Графическое представление законов распределения: построение эмпирической функции плотности распределения и эмпирической интегральной распределения; сравнение с соответствующими кривыми.

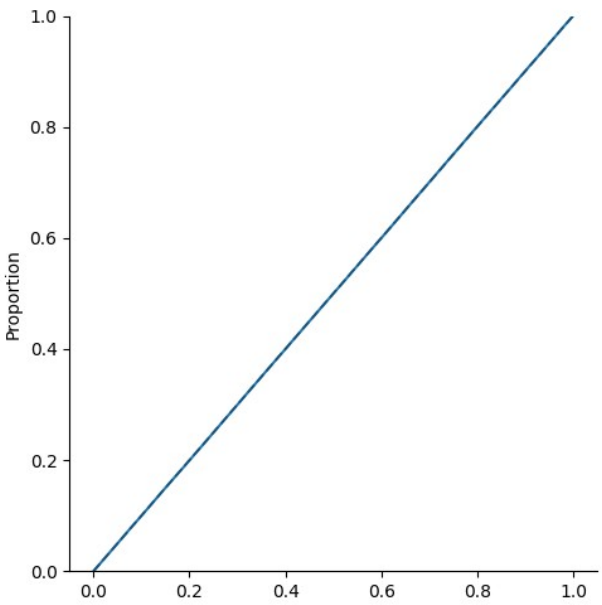
Рассмотрим построенные для n = 10, 1000, 1000000 графики.

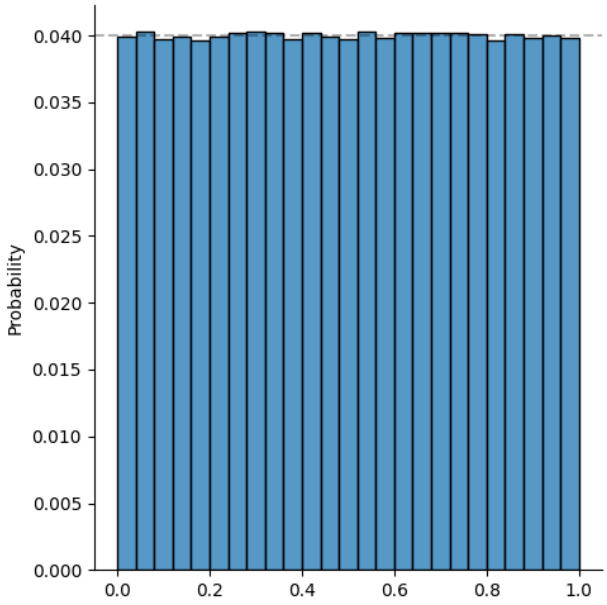


N=10: Эмпирическая интегральная функция распределения чем-то напоминает прямую линию, а эмпирическая функция плотности распределения выглядит не наилучшим образом в меру большого кол-ва корзин (25) для визуализации. Тем не менее, можно заметить что большая часть значений более-менее равномерно распределена по отрезку 0 – 1.



N=1000: Эмпирическая интегральная функция распределения имеет небольшие отклонения от теоретической прямой для равномерного распределения, а функция плотности распределения начинает больше напоминать приближенность к теоретической, хотя и кажется видимое значительное отклонение, вызавнное большим кол-вом корзин для визуализации (25).





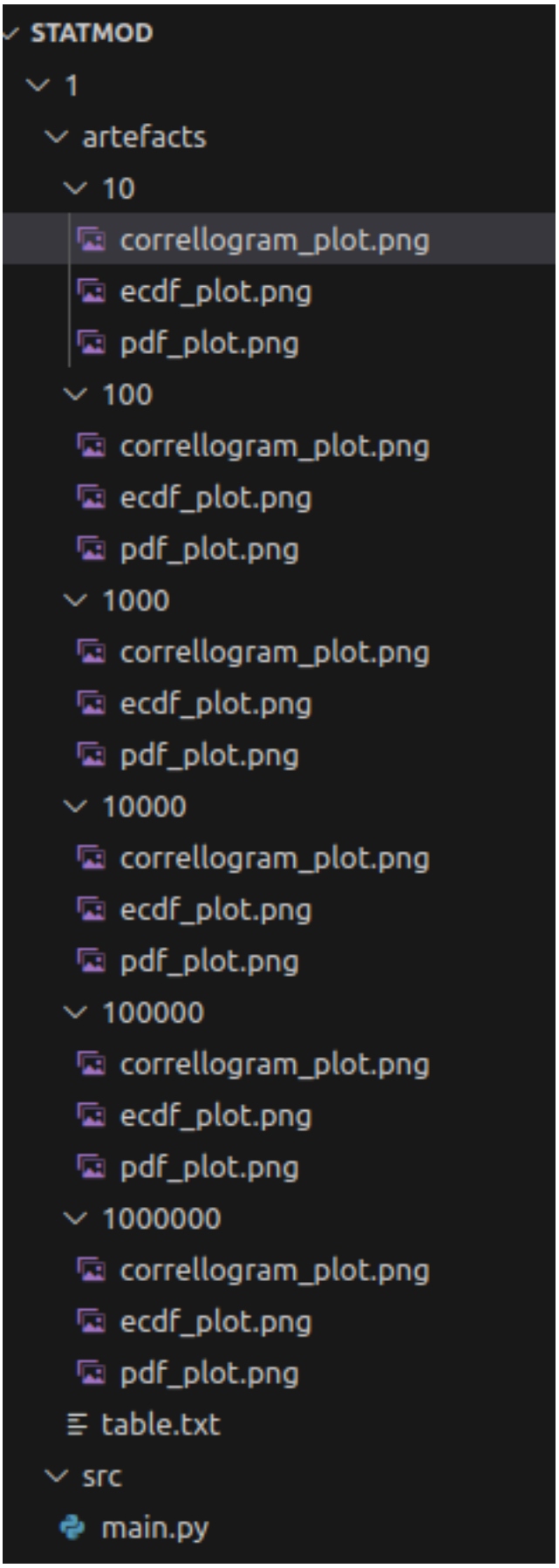
N=10000: Эмпирическая интегральная функция распределения полностью совпадает с предполагаемой прямой для равномерного распределения, а функция плотности распределения подтверждает этот факт, соответствуя теоретическим значениям.

4. Выводы

Стандартный датчик в языке Python модуля random random.random() может быть использован в качестве базового для получения случайных величин с другими законами распределения, поскольку его распределение соответствует нормальному закону, что было проверено с помощью вычисления статистик и сравнения их с эталонными для равномерного распределения на отрезке [0, 1].

Программа

Структура проекта:



main.py

```
import random
import os

from tqdm import tqdm
import polars as pl
import numpy as np
from statsmodels.graphics.tsaplots import plot_acf
import matplotlib.pyplot as plt
import seaborn as sns

def rand():
    return random.uniform(0, 1)

def pseudo_randoms(n):
    return [rand() for _ in tqdm(range(n))]

def experiment(n):
    random_values = pseudo_randoms(n)
    return np.mean(random_values), np.std(random_values) ** 2, random_values

def deviations(a, b):
    return list(np.array(a) - np.array(b))

def autocorrelation_f(u, f):
    n = len(u)
    u_mean = np.mean(u)
    upper = ((u[i] - u_mean)*(u[i+f] - u_mean) for i in range(n-f))
    upper = np.sum(upper)
    lower = ((u[i] - u_mean) ** 2 for i in range(n))
    lower = np.sum(lower)
    return upper / lower

def main(seed, ns, lags):
    random.seed(seed)

    means, disps = [], []
    for n in ns:
        mean, disp, random_values = experiment(n)

        disps.append(disp)
        means.append(mean)

        # Построение графика корреллограммы
        plot_acf(random_values, lags=min(lags, n-1))
        os.makedirs(f'1/artefacts/{str(n)}', exist_ok=True)
        plt.savefig(f"1/artefacts/{str(n)}/correllogram_plot.png")
        plt.close()

        # Построение графика эмпирической интегральной функции распределения
        sns.displot(random_values, kind="ecdf")
        plt.axline((1, 1), slope=1, color='black', linestyle='--', alpha=0.3)
        plt.savefig(f"1/artefacts/{str(n)}/ecdf_plot.png")
        plt.close()

        # Построение графика эмпирической функции плотности распределения
        bin_count = 25
        sns.displot(random_values, bins=[i / bin_count for i in range(0,
bin_count + 1)], stat="probability")
        plt.axhline(y=1/bin_count, color='black', linestyle='--', alpha=0.3)
```

```
        plt.savefig(f"1/artefacts/{str(n)}/pdf_plot.png")
        plt.close()


    means_awaited = [0.5 for m in means]
    disps_awaited = [0.08333 for disp in disps]

    means_deviations = deviations(means, means_awaited)
    disps_deviations = deviations(disps, disps_awaited)

    df = pl.DataFrame(
        {
            "n": ns,
            "mean": means,
            "mean_expected": means_awaited,
            "mean_deviation": means_deviations,
            "disp": disps,
            "disp_expected": disps_awaited,
            "disp_deviation": disps_deviations,
        }
    )
    print(df)

    # Сохранить таблицу в файл
    with open("1/artefacts/table.txt", "w") as f:
        print(df, file=f)


if __name__=='__main__':
    params = {
        "seed": 42,
        "ns": [10 ** i for i in range(1, 7)],
        "lags": 10,
    }
    main(**params)
```