

Моделирование бросков костей и анализ вероятности на python

Кокухина Софья Александровна

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Целью исследования является разработка консольной программы на языке Python для симуляции бросков кубика, расчета частот исходов и визуализации распределения вероятностей. В работе демонстрируются основы использования библиотек `numpy` и `matplotlib` для генерации случайных чисел, статистического анализа и построения графиков. Программа позволяет моделировать большое количество бросков, вычислять эмпирические вероятности и сравнивать их с теоретическими значениями. Работа иллюстрирует практическое применение Python в задачах моделирования случайных процессов и анализа вероятностей.

Ключевые слова Python, симуляция, вероятности, кубик, генератор случайных чисел, визуализация, программирование.

Simulation of dice rolls and probability analysis in python

Kokukhina Sofya Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the research is to develop a Python console program for simulating dice rolls, calculating outcome frequencies, and visualizing probability distributions. The paper demonstrates the basics of using the `numpy` and `matplotlib` libraries for random number generation, statistical analysis, and plotting. The program allows you to simulate a large number of throws, calculate empirical probabilities and compare them with theoretical values. This paper illustrates the practical application of Python in the tasks of modeling random processes and probability analysis.

Keywords Python, simulation, probabilities, dice, random number generator, visualization, programming.

1 Введение

1.1 Актуальность

Симуляция случайных процессов, таких как броски кубика, представляет собой один из наиболее доступных методов для освоения основ теории вероятностей и программирования. Проект по моделированию бросков кубика позволяет наглядно продемонстрировать элементы языка Python, включая генерацию случайных чисел, обработку массивов данных, расчет

статистических показателей и визуализацию результатов. Такой подход оказывается полезным для школьников и студентов, начинающих изучать Python, поскольку сочетает теоретические концепции вероятностных распределений с практическим программированием. В современном образовании акцент делается на использовании симуляций для иллюстрации закона больших чисел, где эмпирические частоты приближаются к теоретическим вероятностям при увеличении числа экспериментов. Кроме того, подобные программы применяются в анализе игр, статистике и моделировании реальных систем, где случайность играет ключевую роль.

1.2 Обзор исследований

В пособии М.С. Мизя и И.Н. Гореловой представлены возможности использования математических методов и моделей в современной экономике для изучения и прогнозирования, оценки вероятности тех или иных событий. Приведены основные формулы и примеры расчетов для практического применения [1]. В статье А.К. Микаеляна рассматриваются прикладные аспекты вероятностных задач. Демонстрируются основные принципы вероятностного моделирования и анализа, а также показываются примеры практических задач, где применение вероятностных методов имеет огромное значение [2]. К.С. Шаров рассмотрел принцип работы конгруэнтного генератора случайных чисел. Каков уровень истинной случайности в случайных числах, полученных с помощью участия человека и механических приёмов, а также без участия человека с помощью вычислительных машин. Показана связь случайных чисел и моделирования процессов и явлений [3]. А.С. Меркулов порассуждал о технологии компьютерного моделирования вероятности выпадения граней правильного игрального кубика с использованием *machinations* для обучения студентов вузов и школьников [4].

1.3 Цель исследования

Цель исследования заключается не только в разработке программы для симуляции бросков кубика на языке Python, но и в демонстрации базовых принципов работы с случайными процессами на современном скриптовом языке. В рамках работы ставятся задачи: изучить основы генерации случайных чисел с помощью библиотеки *numpy*, освоить расчет частот и вероятностей с использованием массивов, реализовать визуализацию распределений через *matplotlib*, а также закрепить навыки структурирования программы через функции. Дополнительно предусматривается сравнение эмпирических результатов с теоретическими вероятностями для подтверждения закона больших чисел.

2 Материалы и методы

Для реализации проекта используется язык программирования Python версии 3.10, обладающий простым синтаксисом и широкими возможностями для написания учебных приложений. В качестве среды разработки

применяется PyCharm Community Edition, обеспечивающая подсветку синтаксиса, автодополнение кода и удобный запуск программ.

На первом этапе подготовки создается новый скрипт в файле, например, `dice_simulator.py`. Далее подключаются необходимые библиотеки: `import numpy as np` и `import matplotlib.pyplot as plt`. Это позволяет использовать функции для генерации случайных чисел и построения графиков (Рис. 1).

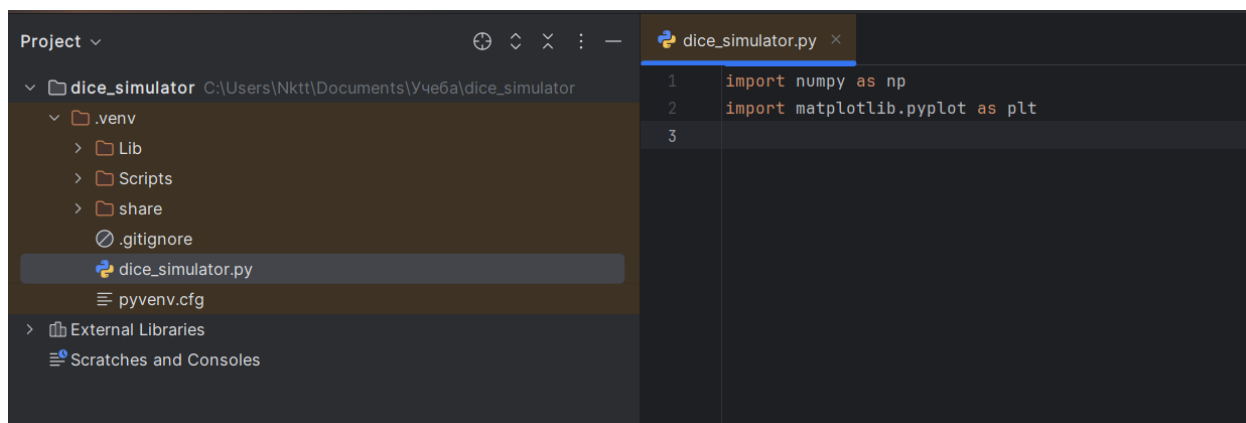


Рисунок 1 – Импорт библиотек в проекте

Далее определяются функции для симуляции бросков, расчета частот и построения гистограммы. Параметры, такие как количество бросков и граней кубика, запрашиваются у пользователя через `input()` для обеспечения интерактивности.

3 Результаты и обсуждения

После подготовки скрипта переходится к реализации логики симуляции. Программа состоит из нескольких ключевых этапов: ввод параметров от пользователя, генерация случайных бросков, расчет частот исходов, сравнение с теоретическими вероятностями и визуализация распределения.

Сначала в основной функции `main` организуется ввод данных с клавиатуры. Пользователю предлагается ввести количество бросков (например, 10000 для точности) и количество граней кубика (по умолчанию 6). Введенные значения преобразуются в целые числа с помощью `int()`. Для обеспечения корректности ввода применяется цикл `while` с конструкцией `try-except`, который перезапрашивает количество граней, если оно меньше 2 или не является целым числом. Это ограничение вводится для моделирования физически возможных сценариев: хотя математическая симуляция работает для любого положительного целого, в реальности игровые кости (`dice`) требуют минимум 2 грани (например, как монета для `d2`), а для полиэдральных форм — минимум 3 или 4. Если ввод некорректен, выводится сообщение об ошибке, что повышает устойчивость программы (Рис. 2).

```
while True:
    try:
        num_sides_input = input("Введите количество граней кубика (по умолчанию 6, минимум 2): ") or "6"
        num_sides = int(num_sides_input)
        if num_sides >= 2:
            break
        else:
            print("Ошибка: Количество граней должно быть минимум 2 (физически возможное для кости). Попробуйте снова.")
    except ValueError:
        print("Ошибка: Введите целое число. Попробуйте снова.")
```

Рисунок 2 – Организация ввода параметров

В контексте физических ограничений и возможных вариантов кубиков отмечается, что программа симулирует равномерные кости (fair dice), где каждая грань имеет одинаковую вероятность. Физически такие кости реализуются как многогранники (polyhedral dice), основанные на платоновых телах или их производных, обеспечивающих симметрию и равновесие. Стандартные варианты включают: 4 грани (тетраэдр, d4), 6 граней (куб, d6), 8 граней (октаэдр, d8), 10 граней (пентагональный трапецоэдр, d10), 12 граней (додекаэдр, d12) и 20 граней (икосаэдр, d20). Нечётные числа граней, такие как 3 (треугольная призма, d3), 5, 7 или 9, также возможны с использованием нерегулярных форм, хотя они могут быть менее идеально симметричными и реже используются в играх.

Теоретически количество граней может быть любым (даже до 120 и более, как в случае disdyakis triacontahedron для d120), но на практике ограничивается сложностью изготовления и обеспечения fairness (равномерности выпадения). В программе минимум 2 грани позволяет моделировать простые случаи, такие как бросок монеты, но для реалистичных полиэдральных кубиков рекомендуется вводить значения от 4 и выше.

Для симуляции бросков кубика используется библиотека numpy. Функция `np.random.randint` генерирует массив целых чисел от 1 до числа граней +1, размером равным количеству бросков. Это обеспечивает равномерное распределение и непредсказуемость результатов, моделируя реальные броски (Рис. 3).

```
def simulate_dice_rolls(num_rolls, num_sides=6):
    """Симулирует броски кубика и возвращает массив результатов."""
    return np.random.randint(1, num_sides + 1, size=num_rolls)
```

Рисунок 3 - Генерация случайных бросков

После получения массива бросков рассчитываются частоты каждого исхода. Функция `np.unique` с параметром `return_counts=True` возвращает уникальные значения и их количества, на основе которых вычисляются относительные частоты путем деления на общее число бросков. Результаты хранятся в словаре для удобства вывода (Рис. 4).

```
def calculate_frequencies(rolls):
    """Рассчитывает частоты каждого исхода."""
    unique, counts = np.unique(rolls, return_counts=True)
    frequencies = counts / len(rolls)
    return dict(zip(unique, frequencies))
```

Рисунок 4 – Расчет частот исходов

Далее выводятся частоты на экран в отсортированном виде, с сравнением каждой эмпирической вероятности с теоретической ($1 / \text{число граней}$). Формат вывода включает четыре знака после запятой для точности, что позволяет наблюдать приближение к теоретическому значению при большом числе бросков (Рис. 5).

```
print("Частоты исходов:")
for outcome, freq in sorted(frequencies.items()):
    print(f"Исход {outcome}: {freq:.4f} (ожидаемая вероятность: {1 / num_sides:.4f})")
```

Рисунок 5 – Вывод частот и сравнение с теорией

Визуализация реализуется с помощью matplotlib. Создается фигура с гистограммой, где bins определяются как $\text{np.arange}(1, \text{num_sides} + 2) - 0.5$ для центрирования столбцов. Добавляются заголовок, метки осей, сетка и тики для граней. График сохраняется в файл `dice_histogram.png` и отображается, если среда интерактивная (Рис. 6).

```
def plot_histogram(rolls, num_sides):
    """Строит гистограмму распределения исходов."""
    plt.figure(figsize=(8, 6))
    plt.hist(rolls, bins=np.arange(1, num_sides + 2) - 0.5, edgecolor='black', alpha=0.7)
    plt.title(f'Распределение исходов для {len(rolls)} бросков кубика')
    plt.xlabel('Исход')
    plt.ylabel('Частота')
    plt.xticks(range(1, num_sides + 1))
    plt.grid(True)
    plt.savefig('dice_histogram.png') # Сохраняет график в файл
    plt.show() # Показывает график (если в интерактивной среде)
```

Рисунок 6 – Построение гистограммы

После завершения кода программа запускается командой `python dice_simulator.py` в консоли (Рис.7).

```
Введите количество бросков кубика: 1000
Введите количество граней кубика (по умолчанию 6, минимум 2): 8
Частоты исходов:
Исход 1: 0.1270 (ожидаемая вероятность: 0.1250)
Исход 2: 0.1070 (ожидаемая вероятность: 0.1250)
Исход 3: 0.1240 (ожидаемая вероятность: 0.1250)
Исход 4: 0.1440 (ожидаемая вероятность: 0.1250)
Исход 5: 0.1240 (ожидаемая вероятность: 0.1250)
Исход 6: 0.1200 (ожидаемая вероятность: 0.1250)
Исход 7: 0.1160 (ожидаемая вероятность: 0.1250)
Исход 8: 0.1380 (ожидаемая вероятность: 0.1250)
```

Рисунок 7 – Отображение вероятностей в консоли

Графический пример работы программы для 10000 бросков шестигранного кубика представлен на рисунке 8, где частоты близки к $1/8 \approx 0.1250$, а гистограмма показывает равномерное распределение (Рис. 8).



Рисунок 7 – Пример работы программы

4

Выводы. В ходе работы разработана программа для симуляции бросков кубика на языке Python. Проект позволяет освоить основы работы с numpy для генерации

случайных данных, расчета статистик и matplotlib для визуализации. Реализация подобной симуляции демонстрирует доступность Python для обучения школьников и студентов теории вероятностей, а также может служить базой для расширения (например, добавления симуляции нескольких кубиков или статистических тестов).

Разработанная программа успешно моделирует броски кубика, рассчитывает вероятности и визуализирует результаты. Она корректно обрабатывает ввод, использует случайную генерацию и предоставляет наглядное сравнение эмпирических и теоретических значений.

Библиографический список

1. Мизя М. С., Горелова И. Н. Математические методы и модели в современной экономике. Учебное текстовое электронное издание локального распространения. Омск: Омский государственный технический университет. 2022. 78 с.
2. Микаелян А. К. Теория вероятностей в практическом использовании // Теоретико-методологические аспекты преподавания математики в современных условиях: материалы Всероссийской научно-практической конференции. Луганск: ФГБОУ ВО «Луганский государственный педагогический университет». 2024. С. 93-96.
3. Шаров К. С. Случайны ли случайности? // Энергия: экономика, техника, экология. 2016. № 6. С. 62-67.
4. Меркулов А. С. Технология компьютерного моделирования вероятности выпадения граней правильного игрального кубика с использованием Machinations для обучения студентов вузов и школьников // Информатика: проблемы, методология, технологии. Информатика в образовании: материалы XVIII Международной школы-конференции. Том 7. Воронеж: Общество с ограниченной ответственностью "Вэлборн". 2018. С. 127-131.