Project 1：Monte Carlo 方法及概率的计算

课程名称：概率、数理统计与随机分析

目录

引言：Monte Carlo 方法是一种基于随机抽样的数值计算方法，广泛应用于解决复杂的概率、积分和优化问题。它通过**大量的随机试验模拟问题的解**，并通过统计这些试验结果来估算问题的答案。这种方法的核心思想是利用**随机性**来逼近问题的解，尤其适用于无法通过解析方法求解的高维或复杂问题。Monte Carlo 方法的应用领域非常广泛，包括金融风险评估、物理模拟、计算机图形学等多个领域。尽管其计算精度随着试验次数的增加而提高，但其主要缺点是计算量大且结果存在一定的不确定性。因此，蒙特卡罗方法在实际应用中通常需要通过增加试验次数来平衡精度和计算成本。

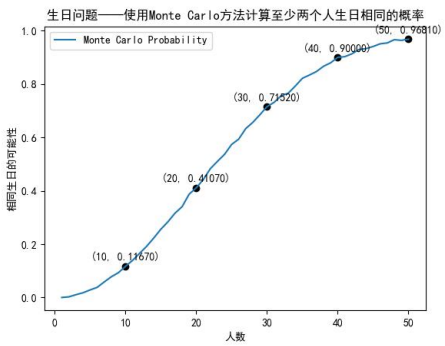
1. 用 Monte Carlo 方法计算生日问题

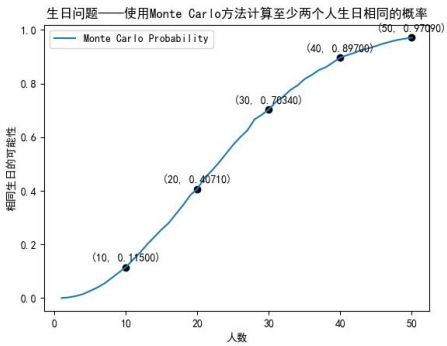
问题描述：即确定 n 个人中至少两个人生日相同的概率（以不同 n 值重复实验多次，记录实验结果并与理论值验证）

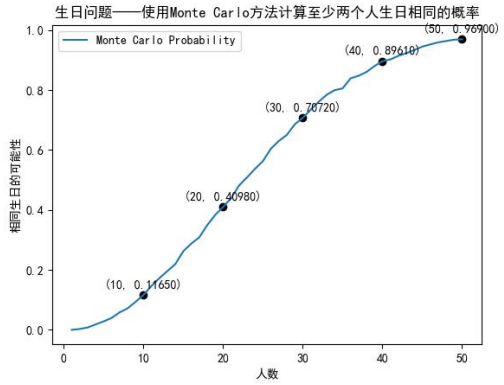
- **python 代码**

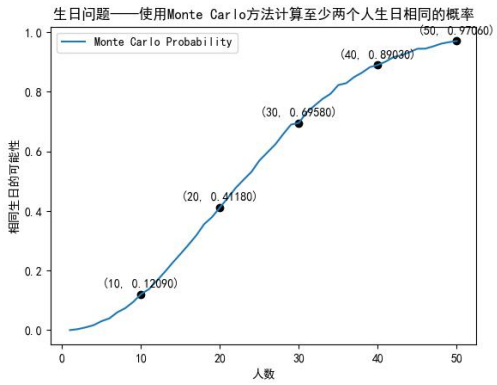
```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei'] # 使其表格能正常显示中文
def birthday_problem_simulation(trials=10000, max_n=50): # 模拟次数设为 10000，最大人数为 50
    probabilities = []
    for n in range(1, max_n + 1):
        matches = 0
        # 人数为 n 时具有相同生日的实验次数
        for _ in range(trials):
            birthdays = np.random.randint(1, 365, n)
            # 一年按 365 天来计算
            if len(birthdays) != len(set(birthdays)):
                # 判断是否存在相同的生日
                matches += 1
        probabilities.append(matches / trials)
    plt.plot(range(1, max_n + 1), probabilities, label="Monte Carlo Probability")
    key_points_x = [10, 20, 30, 40, 50] # 关键点
    key_points_y = [probabilities[x - 1] for x in key_points_x]
    for x, y in zip(key_points_x, key_points_y):
        plt.scatter(x,y,color='black') # 标记关键点
        plt.annotate(f"({x}, {y:.5f})", (x, y), textcoords="offset points", xytext=(0, 5), ha='center')

    plt.xlabel("人数")
    plt.ylabel("相同生日的可能性")
    plt.title("生日问题——使用 Monte Carlo 方法计算至少两个人生日相同的概率")
    plt.legend()
    plt.show()
birthday_problem_simulation()
```

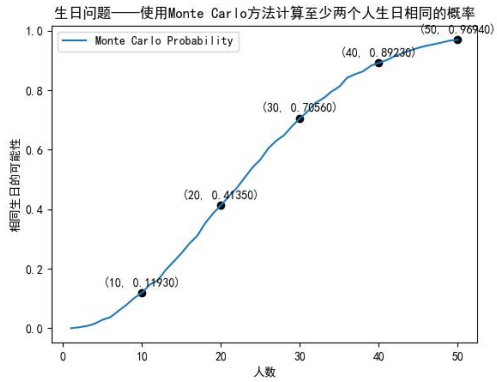
- **图表生成**

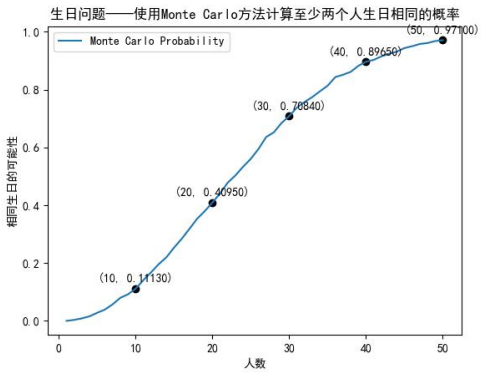










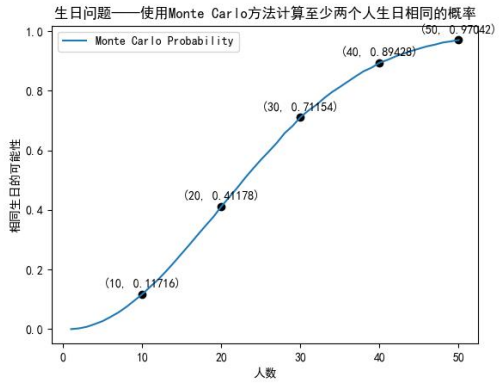


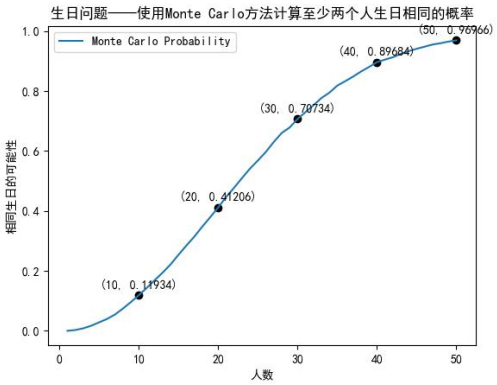
- **图表分析**

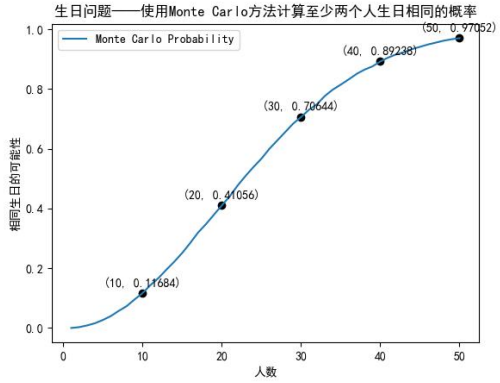
运行该代码六次生成六张不同的表格，下面我们仅研究五个关键点（n=10,20,30,40,50）处的概率。分析表格可知相同生日的可能性 P（在实验次数为 10000 的情况下）：

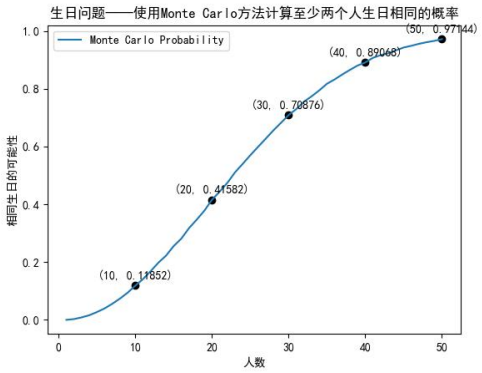
n 的值（实验次数为 10000）	P 的取值（大概范围或趋近值）
10	0.11~0.12
20	0.40~0.41
30	0.70
40	0.90
50	0.97

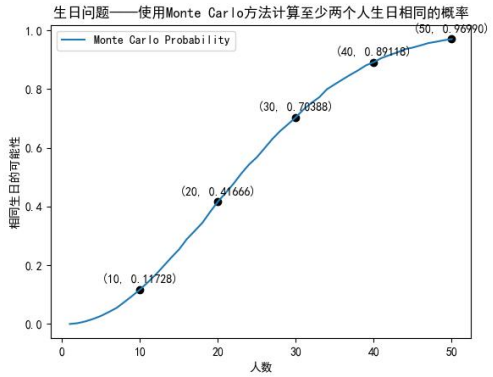
此时增加模拟实验次数到 **50000** 次，在进行进一步观测:

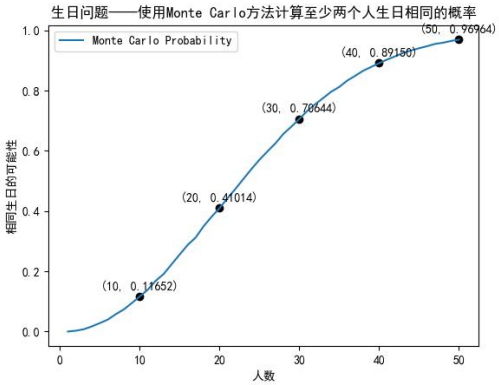












n 的值（实验次数为 50000）	P 的大概取值（范围或趋近值）
10	0.116~0.120
20	0.410~0.411
30	0.700
40	0.891~0.907
50	0.970

- **理论值计算**

至少有2人生日相同的概率 = 1- 所有人生日均不相同的概率

$$P(\text{不同生日}) = \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{365-n+1}{365}$$

n分别取 10, 20, 30, 40, 50

$n=10$时 $P_1 = 1 - \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{356}{365} \approx 0.11695$

$n=20$时 $P_2 = 1 - \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{346}{365} \approx 0.41144$

$n=30$时 $P_2 = 1 - \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{336}{365} \approx 0.70632$

$n=40$时 $P_2 = 1 - \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{326}{365} \approx 0.89123$

$n=50$时 $P_2 = 1 - \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{316}{365} \approx 0.97037$

实验结果在理论预测值的邻近微小范围内呈现波动，从而验证了理论的准确性。

2.每颗炮弹落在椭圆形区域的概率

炮弹射击的目标为一椭圆形区域，在 X 方向半轴长 120m，Y 方向半轴长 80m。当瞄准目标的中心发射炮弹时，在众多随机因素的影响下，弹着点服从以目标中心为均值的正态分布，设 X 方向和 Y 方向的均方差分别为 60m 和 40m，且 X 方向和 Y 方向相互独立，求每颗炮弹落在椭圆形区域的概率。

- **python 代码**

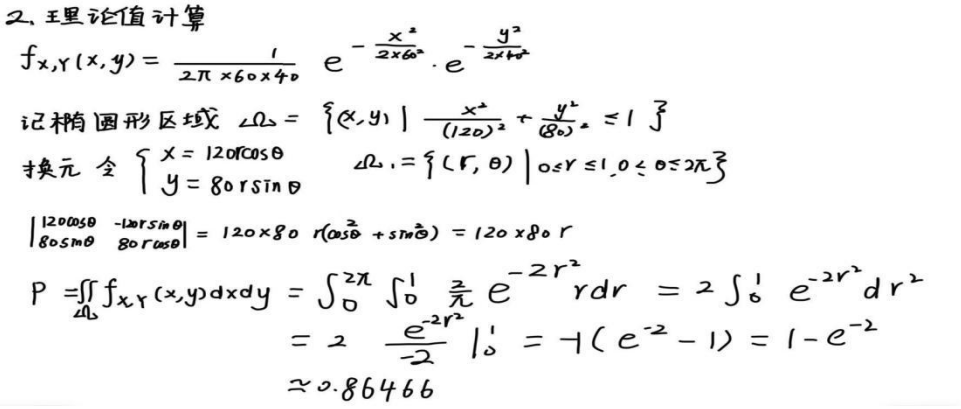
```python
import numpy as np

def bombing_probability(trials=10000, x_axis=120, y_axis=80):
    inside_count = 0
    for _ in range(trials):
        x = np.random.normal(0, 60)
        y = np.random.normal(0, 40)
        if (x ** 2 / x_axis ** 2 + y ** 2 / y_axis ** 2) <= 1:
            inside_count += 1

    probability = inside_count / trials
    print(f"每颗炮弹落在椭圆形区域的概率: {probability:.6f}")

bombing_probability()
```

- **生成结果（10 次为例）**

（1）每颗炮弹落在椭圆形区域的概率: 0.864700
（2）每颗炮弹落在椭圆形区域的概率: 0.862200
（3）每颗炮弹落在椭圆形区域的概率: 0.866400
（4）每颗炮弹落在椭圆形区域的概率: 0.868500
（5）每颗炮弹落在椭圆形区域的概率: 0.869800
（6）每颗炮弹落在椭圆形区域的概率: 0.862600
（7）每颗炮弹落在椭圆形区域的概率: 0.864100
（8）每颗炮弹落在椭圆形区域的概率: 0.858800
（9）每颗炮弹落在椭圆形区域的概率: 0.867600
（10）每颗炮弹落在椭圆形区域的概率: 0.863600

- **理论值计算**

根据所提供的数据，理论计算得出的概率值为 0.86466，而程序输出的概率值在该理论值附近呈现微小波动。这一现象进一步证实了程序计算结果的可靠性。



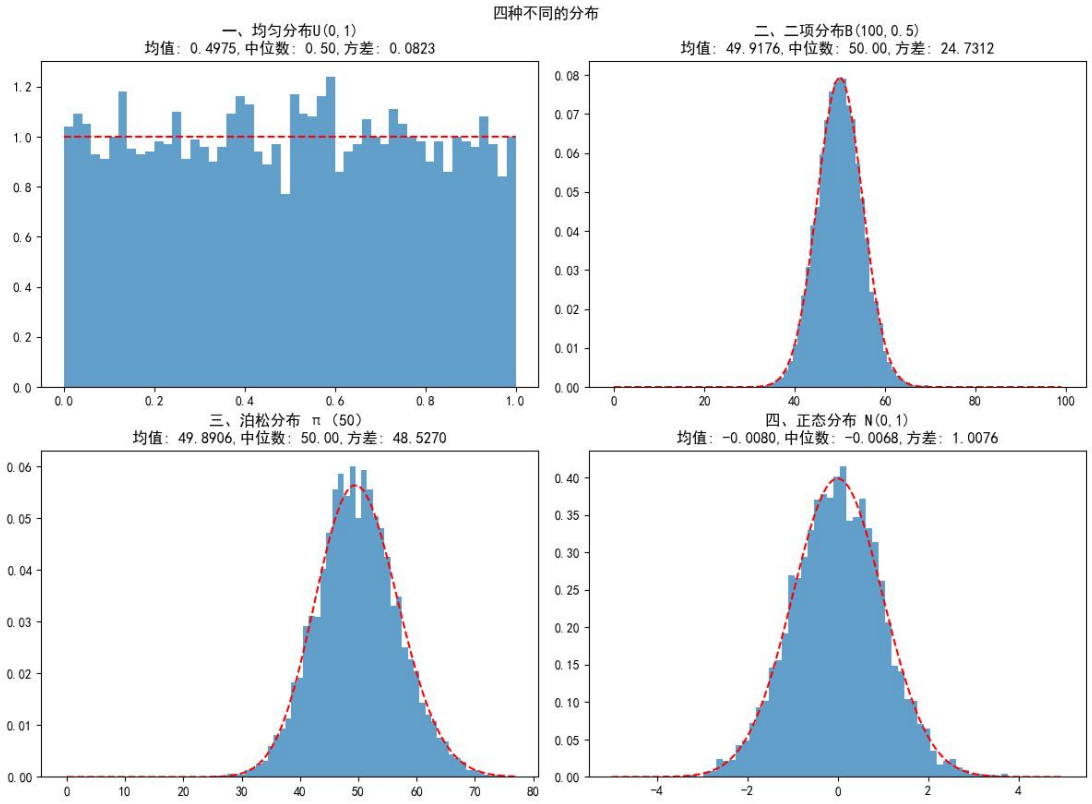
3.分别生成服从均匀分布、二项分布、泊松分布、正态分布的随机数，绘制其分布情况并与理论分布对比（绘图），且计算它们的均值、中位数、方差。

- **python 代码**

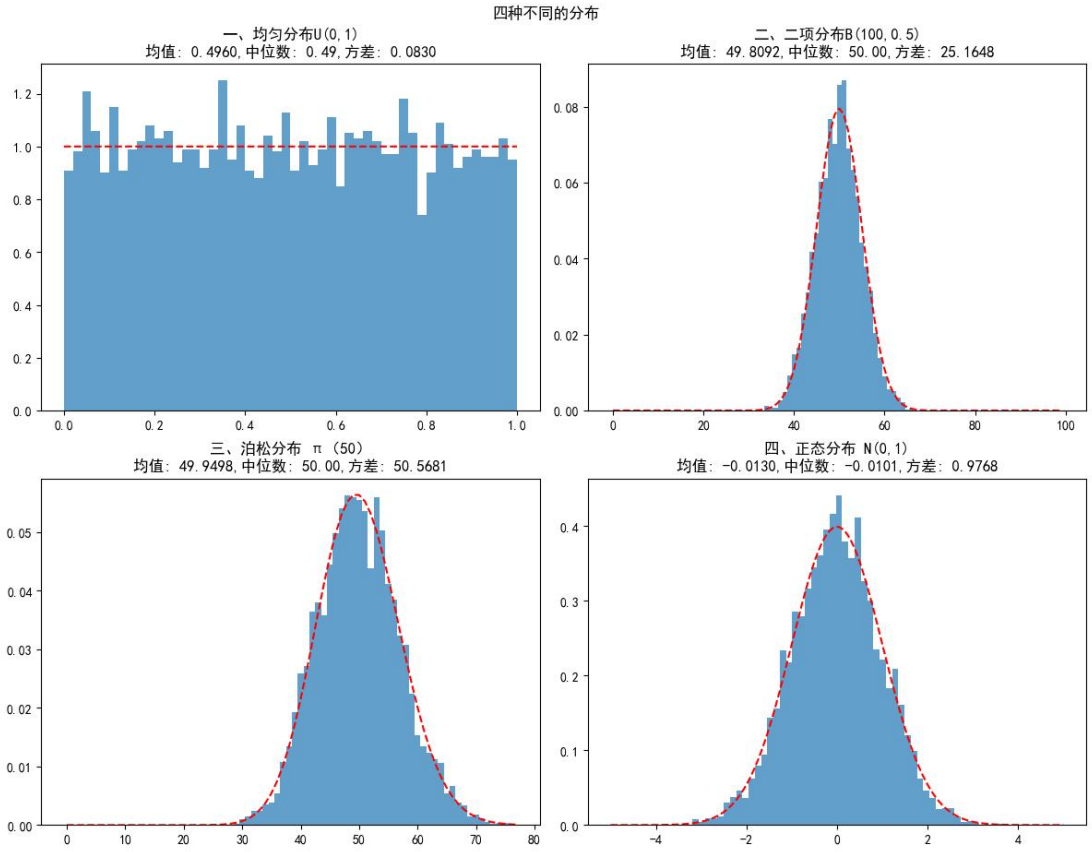
```
import numpy as np

import matplotlib.pyplot as plt
from scipy.stats import binom, poisson, norm

plt.rcParams['font.sans-serif'] = ['SimHei']  # 使其表格能正常显示中文
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题



def plot_distributions(sample_size=5000):  # 样本容量设为 5000
    uniform_data = np.random.uniform(0, 1, sample_size)  # 均匀分布 U(0,1)
    binomial_data = np.random.binomial(100, 0.5, sample_size)  # 二项分布
B(100,0.5)
    poisson_data = np.random.poisson(50, sample_size)  # 泊松分布 π（50）
    normal_data = np.random.normal(0, 1, sample_size)  # 标准正态分布 N(0,1)

    fig, axs = plt.subplots(2, 2, figsize=(12, 10))

    # 均匀分布 U(0,1)
    axs[0, 0].hist(uniform_data, bins=50, density=True, alpha=0.7)
    axs[0, 0].set_title(
        f"一、均匀分布 U(0,1)\n 均值: {np.mean(uniform_data):.4f},中位数:
{np.median(uniform_data):.2f},方差: {np.var(uniform_data):.4f}")
    x = np.linspace(0, 1, 5000)
    axs[0, 0].plot(x, np.ones_like(x), 'r--', label="理论分布")

    # 二项分布 B(100,0.5)
    axs[0, 1].hist(binomial_data, bins=np.arange(0, 101)-0.5, density=True,
alpha=0.7)
    axs[0, 1].set_title(
        f"二、二项分布 B(100,0.5)\n 均值: {np.mean(binomial_data):.4f},中位数:
{np.median(binomial_data):.2f},方差: {np.var(binomial_data):.4f}")
    x = np.arange(0, 100)
    axs[0, 1].plot(x, binom.pmf(x, 100, 0.5), 'r--', label="理论分布")

    # 泊松分布 π（50）
    axs[1, 0].hist(poisson_data, bins=np.arange(0, max(poisson_data) + 1)-0.5,
density=True, alpha=0.7)
    axs[1, 0].set_title(
```

```
        f"三、泊松分布 π（50）\n均值：{np.mean(poisson_data):.4f},中位数：
{np.median(poisson_data):.2f},方差：{np.var(poisson_data):.4f}")
    x = np.arange(0, max(poisson_data) + 1)
    axs[1, 0].plot(x, poisson.pmf(x, 50), 'r--', label="理论分布")

    # 标准正态分布 N(0,1)
    axs[1, 1].hist(normal_data, bins=50, density=True, alpha=0.7)
    axs[1, 1].set_title(
        f"四、正态分布 N(0,1)\n均值：{np.mean(normal_data):.4f},中位数：
{np.median(normal_data):.4f},方差：{np.var(normal_data):.4f}")
    x = np.linspace(-5, 5, 5000)
    axs[1, 1].plot(x, norm.pdf(x, 0, 1), 'r--', label="理论分布")

    plt.suptitle("四种不同的分布")
    plt.tight_layout()
    plt.show()


plot_distributions()
```

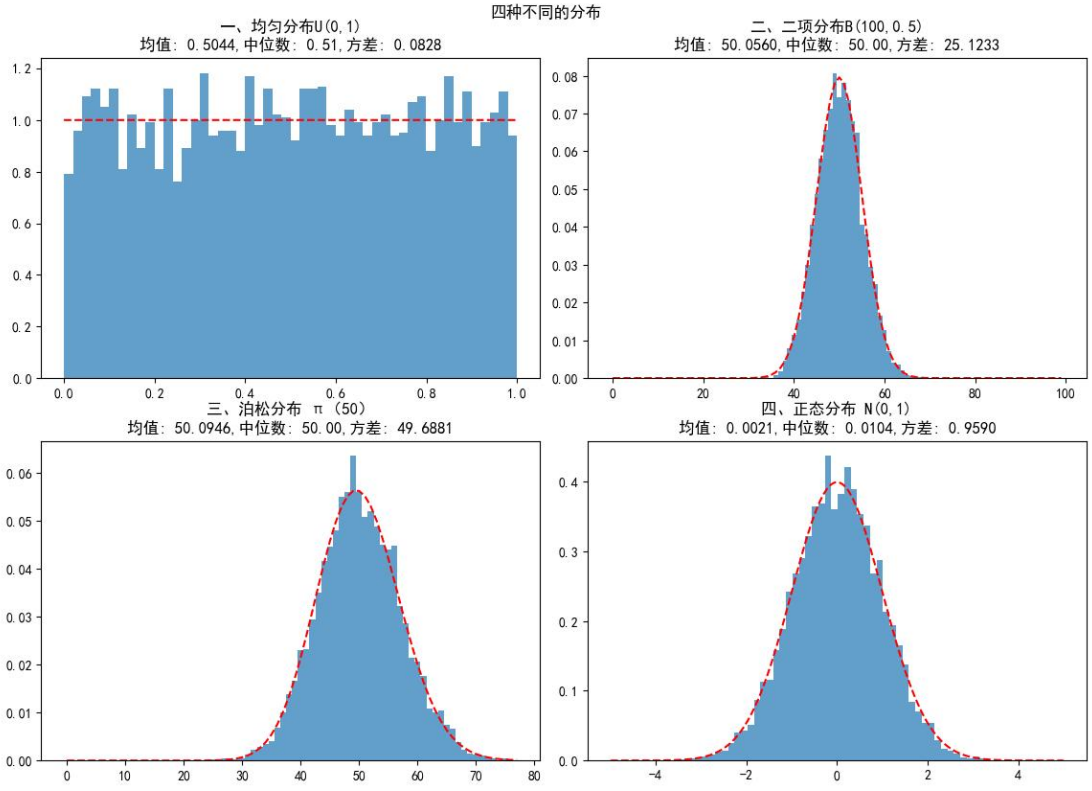
- **图表生成（五张图表）**



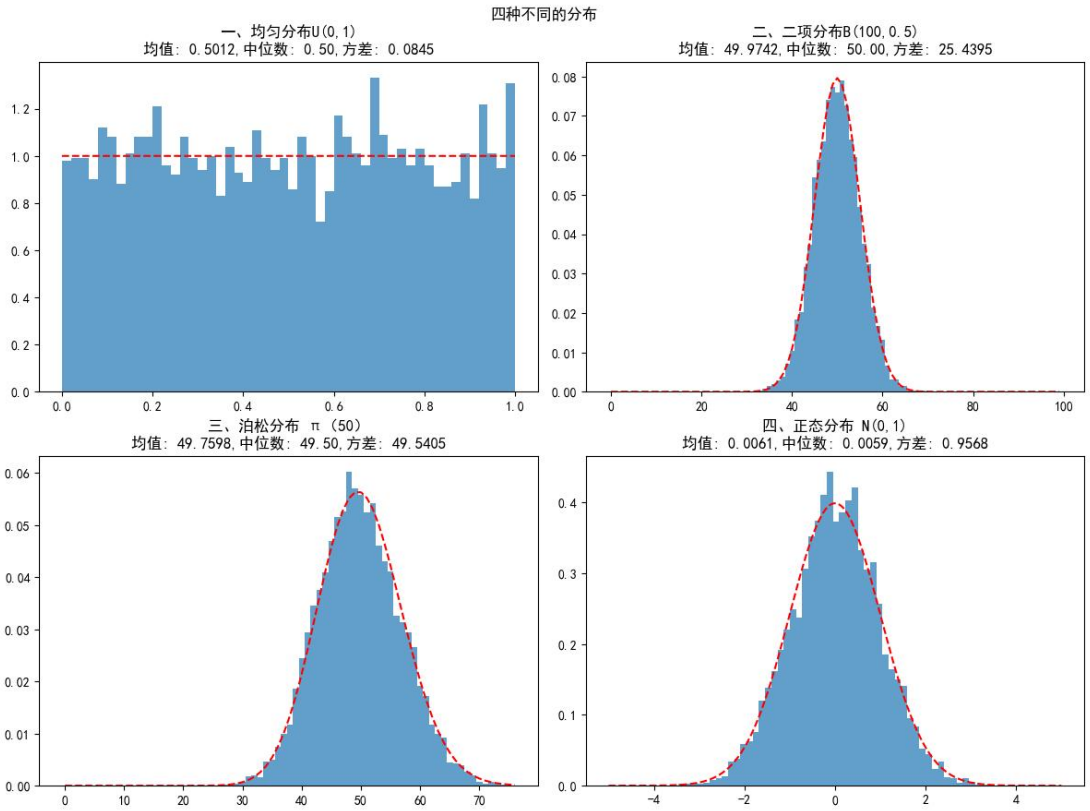
图一


四种不同的分布
一、均匀分布U(0,1)
均值：0.4960,中位数：0.49,方差：0.0830
二、二项分布B(100,0.5)
均值：49.8092,中位数：50.00,方差：25.1648
三、泊松分布 π（50）
均值：49.9498,中位数：50.00,方差：50.5681
四、正态分布 N(0,1)
均值：-0.0130,中位数：-0.0101,方差：0.9768

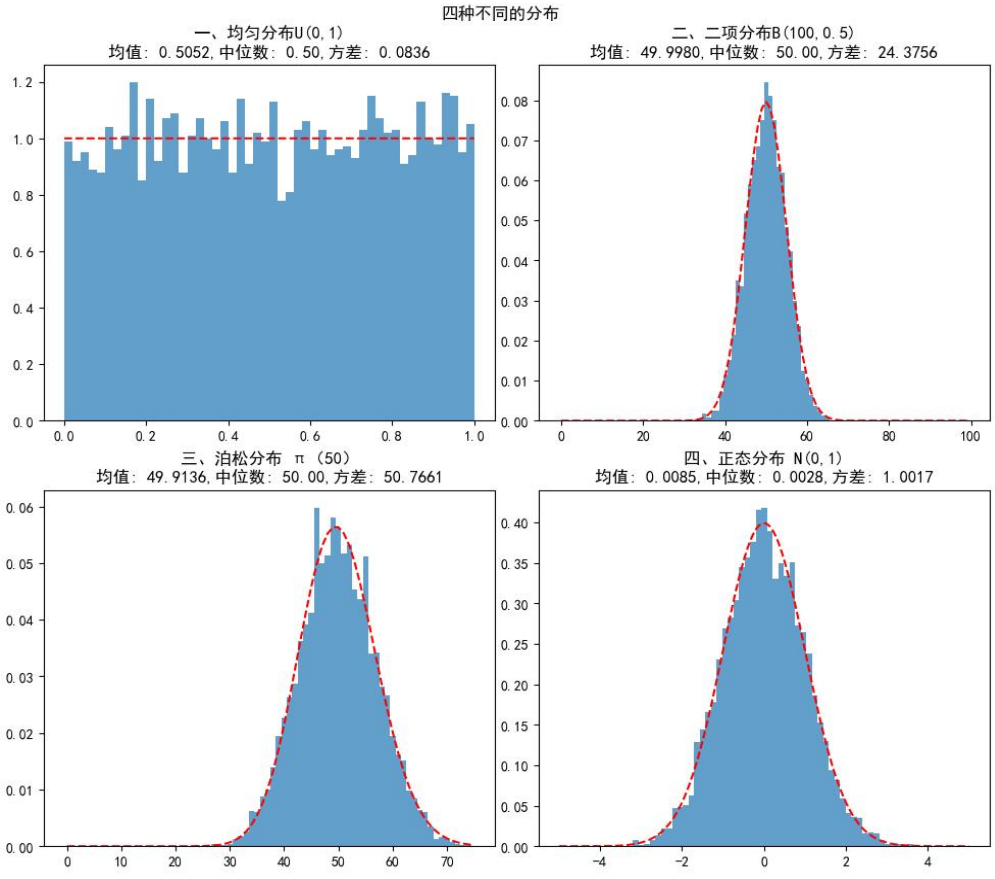
图二


四种不同的分布
一、均匀分布U(0,1)
均值：0.5044,中位数：0.51,方差：0.0828
二、二项分布B(100,0.5)
均值：50.0560,中位数：50.00,方差：25.1233
三、泊松分布 π（50）
均值：50.0946,中位数：50.00,方差：49.6881
四、正态分布 N(0,1)
均值：0.0021,中位数：0.0104,方差：0.9590

图三


四种不同的分布
一、均匀分布U(0, 1)
均值：0.5012, 中位数：0.50, 方差：0.0845
二、二项分布B(100, 0.5)
均值：49.9742, 中位数：50.00, 方差：25.4395
三、泊松分布 π（50）
均值：49.7598, 中位数：49.50, 方差：49.5405
四、正态分布 N(0, 1)
均值：0.0061, 中位数：0.0059, 方差：0.9568

图四


四种不同的分布
一、均匀分布U(0, 1)
均值：0.5052, 中位数：0.50, 方差：0.0836
二、二项分布B(100, 0.5)
均值：49.9980, 中位数：50.00, 方差：24.3756
三、泊松分布 π（50）
均值：49.9136, 中位数：50.00, 方差：50.7661
四、正态分布 N(0, 1)
均值：0.0085, 中位数：0.0028, 方差：1.0017

图五

• 图表分析

（1）均匀分布 U（0,1）：均值理论值为 0.5，实验均值在 0.5 附近

（2）二项分布 B（100，0.5）：均值 μ=100×0.5=50，实验均值在其附近

（3）泊松分布 π （50）：理论均值=方差= λ=50，均值和方差的实验结果均在该值附近

（4）正态分布 N（0,1）： μ=0，方差=1，实验所得结果也在其附近

从而的得到了验证，证明了蒙特卡洛方法进行估算的正确性。

4. 编程构造 100 个在(0,1)服从均匀分布的向量，绘制它们和的概率分布函数图，并与正态分布的密度函数对比。目的：验证中心极限定理（大量独立随机变量的和近似服从正态分布）

• 中心极限定理的说明

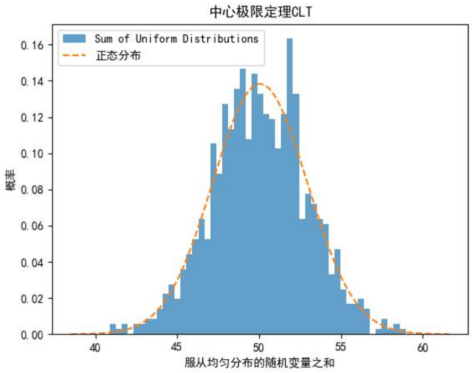
独立同分布的中心极限定理（Central Limit Theorem, CLT）指出，对于一组独立同分布（i.i.d.）的随机变量 X1,X2,…,Xn，且具有相同的数学期望与方差，那么随着 n 增大，它们的标准化和（即和的均值减去期望后除以标准差）将近似服从标准正态分布。

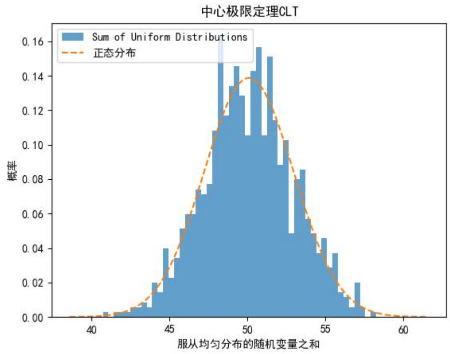
• python 代码

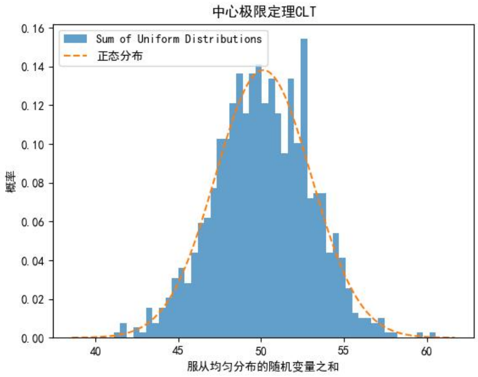
一个向量含有 100 个在（0,1）上服从均匀分布的随机变量，下面我们进行 1000 次实验

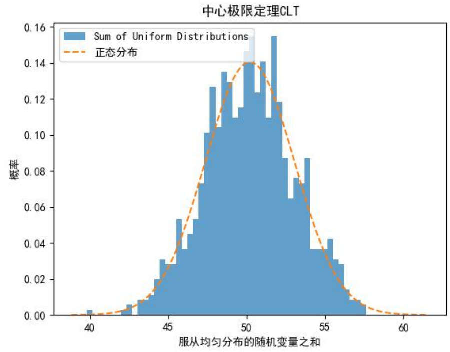
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
plt.rcParams['font.sans-serif'] = ['SimHei'] # 使其表格能正常显示中文
def central_limit_theorem(sample_size=100, vector_count=1000):
    # 生成多个均匀分布的随机数并求和
    uniform_sums = np.sum(np.random.uniform(0, 1, (vector_count, sample_size)),
axis=1)
    # 生成一个 1000×100 的矩阵，对每一行进行求和，即得到样本数
    plt.hist(uniform_sums, bins=50, density=True, alpha=0.7, label="Sum of Uniform
Distributions")
    #bins=50 设置直方图的柱子数量，density=True 使得直方图的总面积为 1，alpha=0.7
设置透明度，
    mean_sum = np.mean(uniform_sums)
    std_sum = np.std(uniform_sums)
    x = np.linspace(mean_sum - 4 * std_sum, mean_sum + 4 * std_sum, 1000)
    plt.plot(x, norm.pdf(x, mean_sum, std_sum), label="正态分布",
linestyle="dashed")

    plt.title("中心极限定理 CLT")
    plt.xlabel("服从均匀分布的随机变量之和")
    plt.ylabel("概率")
    plt.legend()
    plt.show()
```

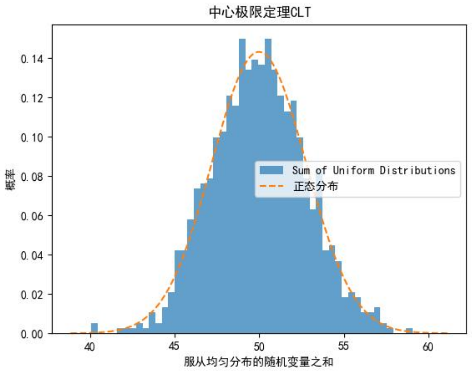
• 图表生成

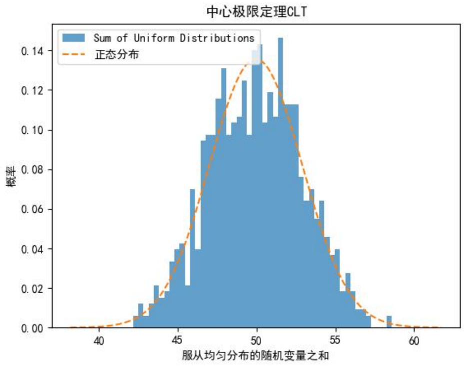

中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和

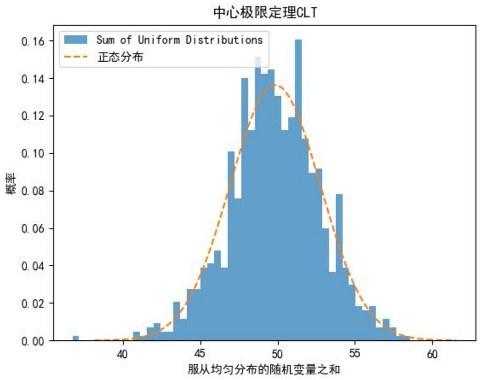

中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和

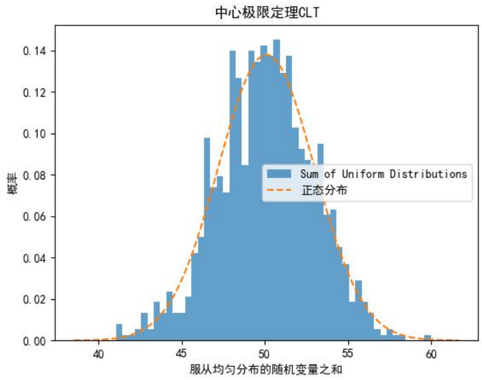

中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和

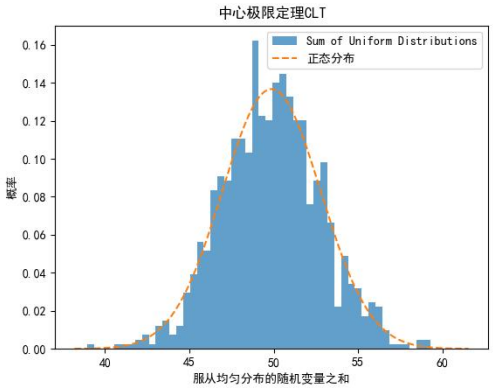

中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和

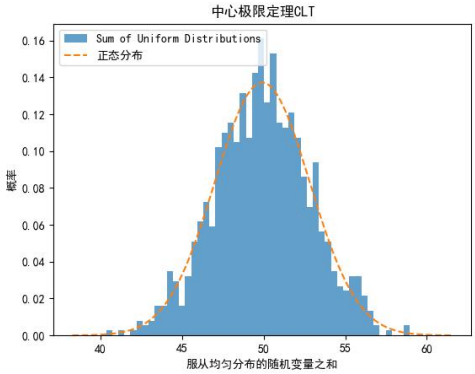

中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和


中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和


中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和


中心极限定理CLT
Sum of Uniform Distributions
正态分布
概率
服从均匀分布的随机变量之和





• 图表分析

根据图表分析结果，我们可以得出以下结论：

1. 柱状图与正态分布的拟合度：

观察到的柱状图显示，概率分布的形状近似接近于虚线表示的正态分布的概率分布函数（PDF）。这一现象表明，在一定条件下，大量独立随机变量的和的值的分布趋向于正态分布，即使这些变量本身不是正态分布的。

2. 中心极限定理的验证：

实验结果支持中心极限定理（Central Limit Theorem, CLT）的理论预测。中心极限定理指出，iid 且具有相同的数学期望和方差的一组数据，当样本数据足够大时，样本均值的分布将趋近于正态分布。通过实验数据与正态分布的拟合程度，我们验证了中心极限定理在实际应用中的有效性。

综上所述，图表中的概率分布与正态分布的高拟合度不仅展示了蒙特卡洛模拟方法的准确性，也进一步证实了中心极限定理在统计学中的普遍适用性和重要性。这一发现对于理解和应用统计学原理具有重要意义。