16 Monte Carlo Simulation 蒙特卡洛模拟

以概率统计为基础，基于伪随机数，进行数值模拟

任何考虑用算术手段来产生随机数的人当然都是有原罪的。

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

—— 约翰·冯·诺伊曼 (John von Neumann) | 美国籍数学家 | 1903 ~ 1957

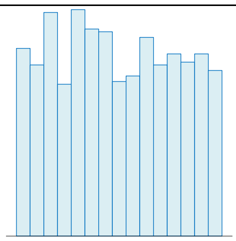
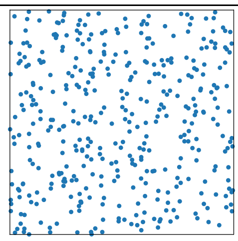
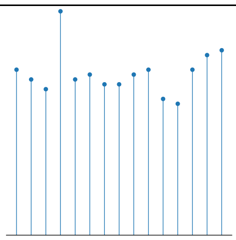
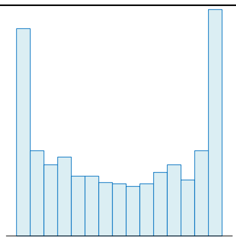
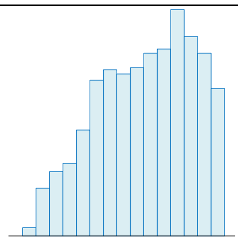
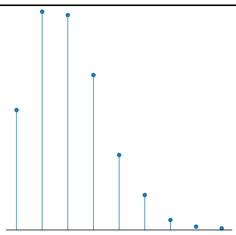
- ◀ matplotlib.patches.Circle() 绘制正圆
- ◀ matplotlib.pyplot.semilogx() 横轴设置为对数坐标
- ◀ numpy.empty() 产生全为 NaN 序列
- ◀ numpy.random.beta() 产生服从 Beta 分布的随机数
- ◀ numpy.random.binomial() 产生服从二项分布的随机数
- ◀ numpy.random.dirichlet() 产生服从 Dirichlet 分布的随机数
- ◀ numpy.random.exponential() 产生服从指数分布的随机数
- ◀ numpy.random.geometric() 产生服从几何分布的随机数
- ◀ numpy.random.lognormal() 产生服从对数正态分布的随机数
- ◀ numpy.random.multivariate_normal() 产生服从多项正态分布的随机数
- ◀ numpy.random.normal() 产生服从正态分布的随机数
- ◀ numpy.random.poisson() 产生服从泊松分布的随机数
- ◀ numpy.random.randint() 产生均匀整数随机数
- ◀ numpy.random.standard_t() 产生服从学生 t-分布的随机数
- ◀ numpy.random.uniform() 产生服从连续均匀分布的随机数
- ◀ numpy.where() 返回满足条件的元素序号
- ◀ scipy.integrate.dblquad() 求解双重定积分值
- ◀ scipy.integrate.quad() 求解定积分值
- ◀ scipy.linalg.cholesky() 对矩阵进行 Cholesky 分解
- ◀ seaborn.distplot() 绘制频率直方图和 KDE 曲线
- ◀ seaborn.heatmap() 绘制热图

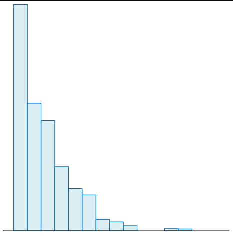
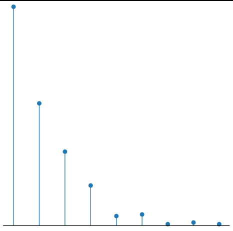
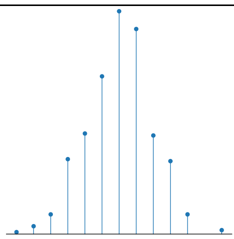
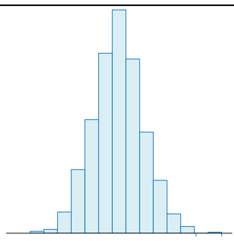
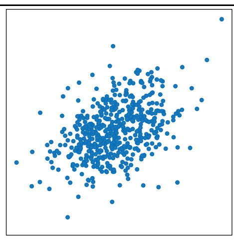
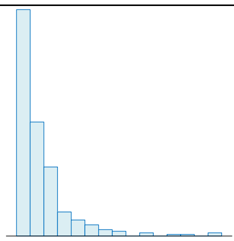
16.1 蒙特卡洛模拟：基于伪随机数发生器

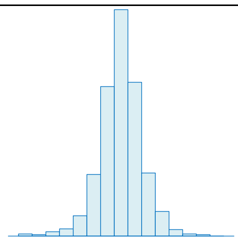
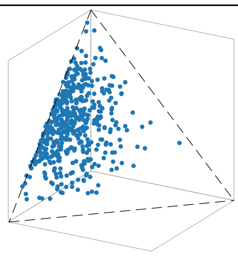
蒙特卡洛模拟 (Monte Carlo simulation)，也称统计模拟方法，是以概率统计理论为核心的数值计算方法。蒙特卡洛模拟将提供多种可能的结果以及通过大量随机数据样本得出的每种结果的概率。冯·诺伊曼 (John von Neumann) 等三名科学家在 20 世纪 40 年代发明了蒙特卡洛模拟。他们以摩纳哥著名的赌城——蒙特卡洛 (Monte Carlo)——为其命名。

本章介绍几个最基本的蒙特卡洛模拟实验。这些模拟实验都基于随机数发生器，表 1 总结常用随机数发生器函数和随机数分布图像。

表 1. 常用随机数发生器

随机数服从的分布	函数	随机数分布图像
连续均匀分布	numpy.random.uniform()	 
均匀整数	numpy.random.randint()	
Beat 分布	numpy.random.beta()	 
泊松分布	numpy.random.poisson()	

指数分布	numpy.random.exponential()	
几何分布	numpy.random.geometric()	
二项分布	numpy.random.binomial()	
正态分布	numpy.random.normal()	
多元正态分布	numpy.random.multivariate_normal()	
对数正态分布	numpy.random.lognormal()	

学生 t-分布	numpy.random.standard_t()	
Dirichlet 分布	numpy.random.dirichlet()	

16.2 估算平方根

本节用蒙特卡洛模拟估算 $\sqrt{2}$。如图 1 所示，为了估算 $\sqrt{2}$，可以在 0 ~ 2 的范围内产生大量服从均匀分布的随机数。在 0 ~ 2 范围内，随机数在 0 ~ $\sqrt{2}$ 出现的概率为 $\sqrt{2}/2$，$\sqrt{2}$ 则可以根据下式估计得到：

$$\sqrt{2} \approx 2 \times \frac{n\left(0 \le x \le \sqrt{2}\right)}{n\left(0 \le x \le 2\right)} \tag{1}$$

其中 $n()$计算频数。

由于，$\sqrt{2}$ 未知；所以，采用图 1 所示平方技巧，即 $\sqrt{2}$ 可以根据下式得到：

$$\sqrt{2} \approx 2 \times \frac{n\left(0 \le x^2 \le 2\right)}{n\left(0 \le x^2 \le 4\right)} \tag{2}$$

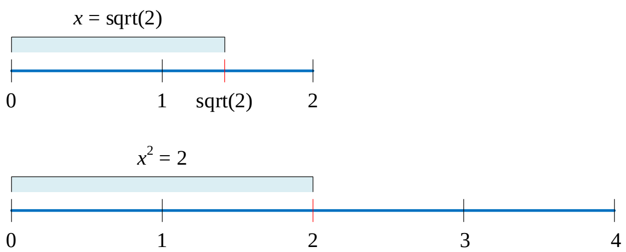


图 1. 估算 sqrt(2)

代码文件 Bk5_Ch15_01.py 估算 sqrt(2)。

16.3 估算积分

本节给出的例子用蒙特卡罗模拟方法估算积分。

给出如下函数 $f(x)$：

$$f(x)=\frac{x\cdot\sin(x)}{2}+8 \tag{3}$$

计算 $f(x)$ 在 [2, 10] 区间内定积分：

$$\int_{x=2}^{x=10}\left(\frac{x\cdot\sin(x)}{2}+8\right)\mathrm{d}x \tag{4}$$

如图 2 所示，在 [2, 10] 区间中，函数 $f(x)$ 的最大值为 12。在横轴取值从 2 ~ 10，纵轴取值从 0 ~ 12 的长方形空间里，产生满足均匀分布的 1000 个数据点。图 2 中蓝色 ● 在曲线之下，红色 × 在曲线之上。图 2 中整个长方形的面积为 96，定积分对应曲线之下的面积 A，可以通过下式估算得到：

$$A\approx 96\times\frac{n(\text{below } f(x))}{1000} \tag{5}$$

$n(\text{below } f(x))$ 为 1000 个数据点中位于 $f(x)$ 曲线之下的数量。

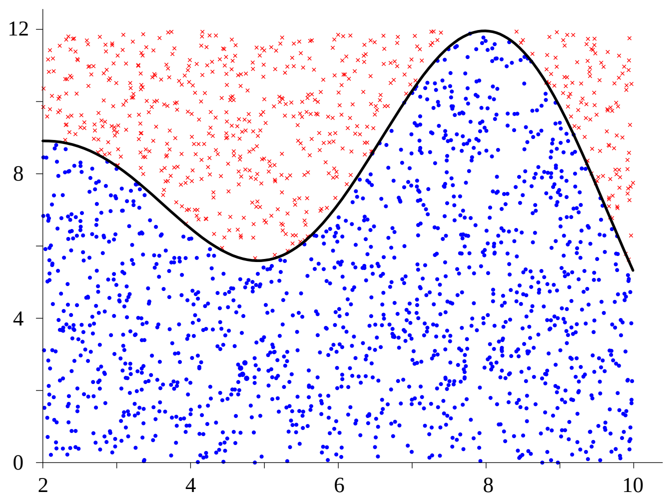


图 2. 用蒙特卡罗模拟法求积分

代码文件 Bk5_Ch15_02.py 估算积分。

16.4 估算体积

本节用蒙特卡洛模拟估算空间体积大小。图 3 (a) 所示二次曲面解析式如下：

$$z = 2 - x^2 - y^2 \tag{6}$$

当 x 和 y 均在 $[-1, 1]$ 范围内时，编写代码用蒙特卡洛模拟估算图 3 (a) 曲面和 $z = 0$ 平面 (蓝色) 构造的空间体积。这个体积相当于如下双重定积分：

$$\int_{x=-1}^{x=1} \int_{y=-1}^{y=1} \left(2 - x^2 - y^2\right) \mathrm{d}\,x\,\mathrm{d}\,y \tag{7}$$

整个立方体空间体积为 8，在这个空间均匀产生 5000 个随机点。如图 3 (b) 所示，二次曲面之上随机点为红色，曲面之下随机点为蓝色。类似上一节，根据随机点的比例，可以估算 (7) 定积分。

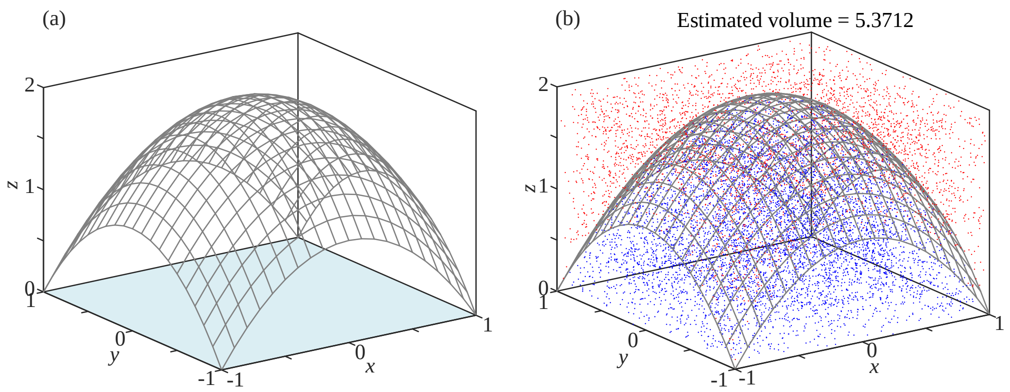


图 3. 利用蒙特卡洛估算体积

Bk5_Ch15_03.py 估算本节体积。

16.5 估算圆周率

《数学要素》一本已经介绍几种方法估算圆周率 π，本节介绍采用蒙特卡罗模拟法估算圆周率。

圆面积和正方形面积之间的比例关系为：

$$\frac{A_{circle}}{A_{square}} = \frac{\pi}{4} \tag{8}$$

可以推导得到：

$$\pi = 4 \times \frac{A_{circle}}{A_{square}} \tag{9}$$

图 4 所示为一次随机数数量为 500 条件下，圆周率估算结果。图 5 所示为不断增大随机数数量，圆周率估算精确度不断提高。

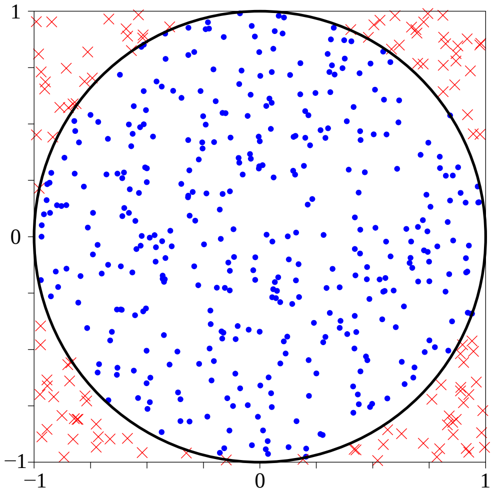


图 4. 蒙特卡罗模拟估算圆周率

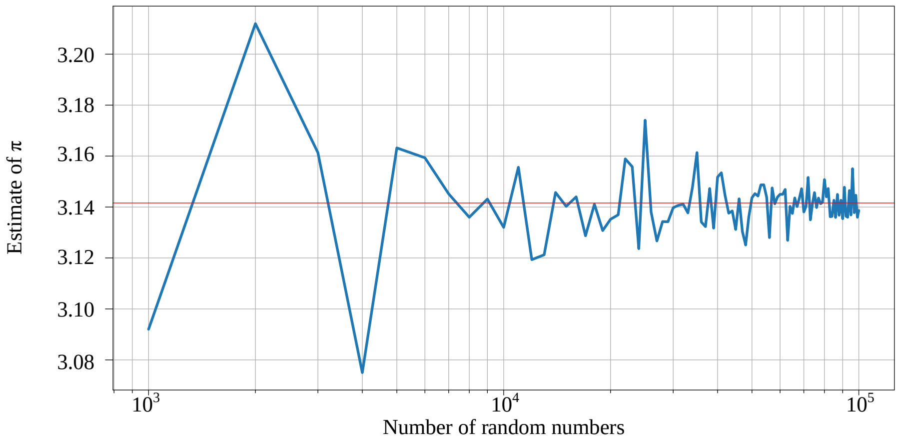


图 5. 不断增大随机数数量，圆周率估算精确度不断提高

Bk5_Ch15_04.py 利用蒙特卡洛模拟估算圆周率。

16.6 布丰投针估算圆周率

布丰投针 (Buffon's needle problem) 也可以用来估算圆周率。

十八世纪，法国博物学家布丰 (Comte de Buffon) 提出著名的布丰投针问题。一个用平行且等距木纹铺成的地板，随意投掷一支长度比木纹间距略小的针，求针和其中一条木纹相交的概率。

如图 6 所示，和平行线相交的针颜色为红色，不和平行线相交的针颜色为蓝色。设平行线距离为 t，针的长度为 l。本节布丰投针问题，我们仅仅考虑“短针”情况，即 $l < t$。

如放大视图所示，x 为针的中心和最近平行线的距离，θ 为针和平行线之间的锐角夹角。

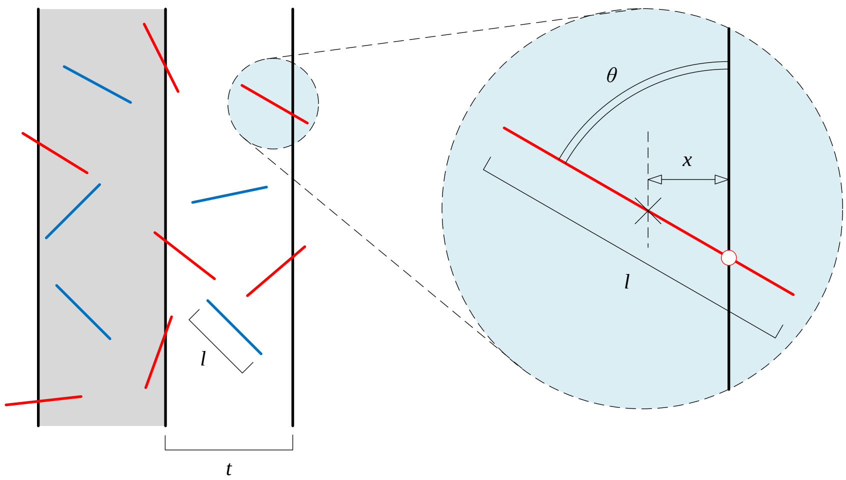


图 6. 布丰投针原理

不难理解，X 作为一个随机变量是在 $[0, t/2]$ 区间的连续均匀分布，概率密度函数为：

$$f_X(x) = \frac{2}{t} \quad x \in [0, t/2] \tag{10}$$

同理，Θ 作为一个随机变量是在 $[0, \pi/2]$ 区间的均匀分布，概率密度函数为：

$$f_{\Theta}(\theta)=\frac{2}{\pi}\quad \theta\in[0,\pi/2] \tag{11}$$

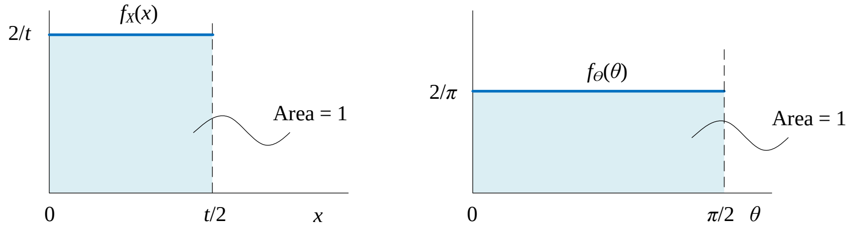


图 7. X 和 Θ 的概率密度函数

显然，X 和 Θ 这两个随机变量相互独立；因此，它们的联合概率密度函数是两者之积，即：

$$f_{\Theta,X}(\theta,x)=\frac{2}{\pi}\frac{2}{t}=\frac{4}{\pi t}\quad \theta\in[0,\pi/2],\quad x\in[0,t/2] \tag{12}$$

给定夹角 θ，满足如下条件，针和平行线相交：

$$x\le\frac{l}{2}\sin\theta,\quad \theta\in[0,\pi/2],\quad x\in[0,t/2] \tag{13}$$

因此，针线相交的概率为如下双重定积分：

$$\Pr(\text{cross})=\int_{\theta=0}^{\theta=\frac{\pi}{2}}\int_{x=0}^{x=\frac{l}{2}\sin\theta}\frac{4}{\pi t}\,\mathrm{d}x\,\mathrm{d}\theta=\frac{2l}{\pi t} \tag{14}$$

假设抛 n 根针，其中有 c 根和平行线相交，概率值 Pr(cross) 可以通过下式估算：

$$\mathrm{P}(\text{cross})\approx\frac{c}{n} \tag{15}$$

联立 (14) 和 (15)，可以得到：

$$\frac{2l}{\pi t}\approx\frac{c}{n} \tag{16}$$

从而推导得到，圆周率的估算值：

$$\pi\approx\frac{2l}{t}\frac{n}{c} \tag{17}$$

图 8 所示为某次实验投掷 2000 根针，612 根和平行线相交；式样中，针的长度 $l = 1$，平行线间隔 $t = 2$。

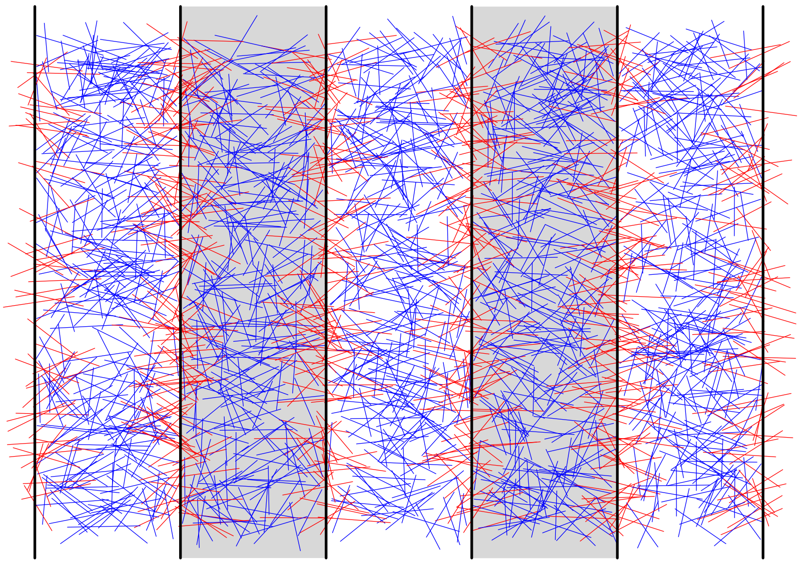


图 8. 投掷 2000 根针，612 根和平行线相交，针的长度 $l = 1$，平行线间隔 $t = 2$

实际上，根据 (13)，我们知道针和平行线相交的概率 Pr(cross) 可以进一步简化。在图 9 阴影区域产生满足均匀分布的随机数，随机数落入蓝色区域的概率就是 Pr(cross)。这样，我们可以根据这一思路编程解决这个简化版的布丰投针估算圆周率问题。

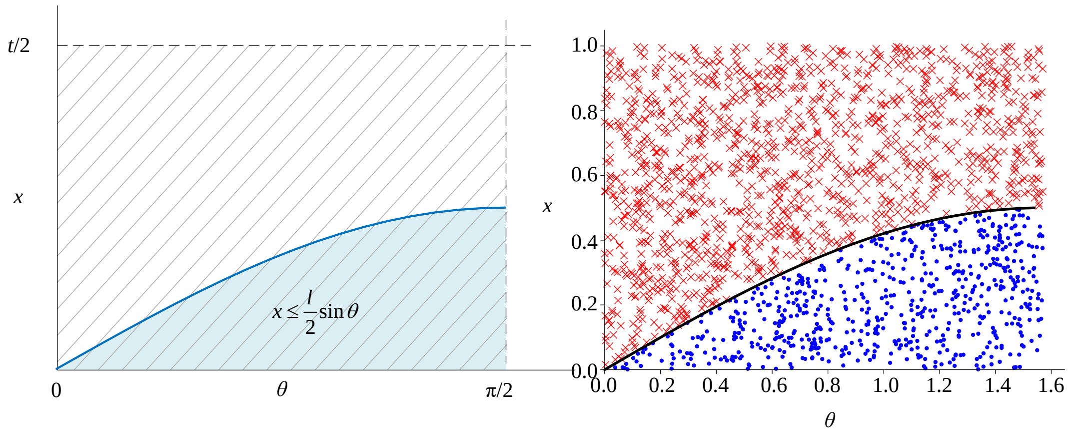


图 9. 针和平行线相交的概率，蒙特卡洛模拟实验结果

Bk5_Ch15_05.py 完成简化版布丰投针蒙特卡洛模拟实验。

16.7 二项分布随机漫步

丛书《数学要素》第 20 章讲过在二叉树规定的网格行走的例子。如图 10 所示，登山者在二叉树始点或中间节点时，他都会面临“向上”或“向下”抉择。如果登山者，通过抛硬币来决定每一步的行走路径——正面，向右上走；反面，向右下走。

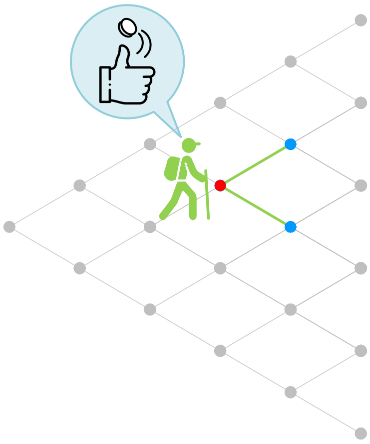


图 10. 二叉树路径与可能性，图片来自《数学要素》

图 11 所示为若干条二叉树随机行走路径，模拟时向上行走的概率 $p = 0.5$。乍一看图 11 很难发现任何规律。但是不断增大随机行走的路径数 n，如图 12 所示，我们发现登山者到达终点的位置呈现类似二项分布规律。观察图 12 (c)，我们发现当 $p = 0.5$ 时，登山者大概率会到达二叉树网格终点中部。

图 13 和图 14 对应登山者向上行走的概率 $p = 0.6$。图 15 图 16 对应登山者向上行走的概率 $p = 0.4$。请大家自行分析这四幅图。

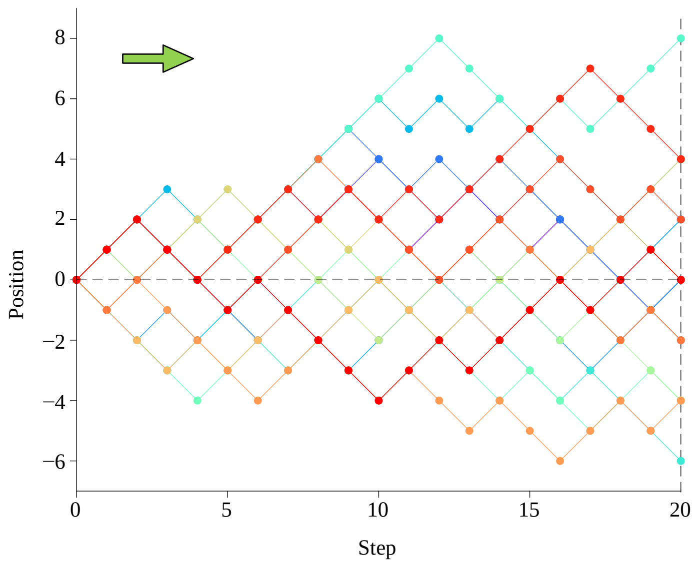


图 11. 二叉树随机行走路径，向上行走的概率 $p = 0.5$

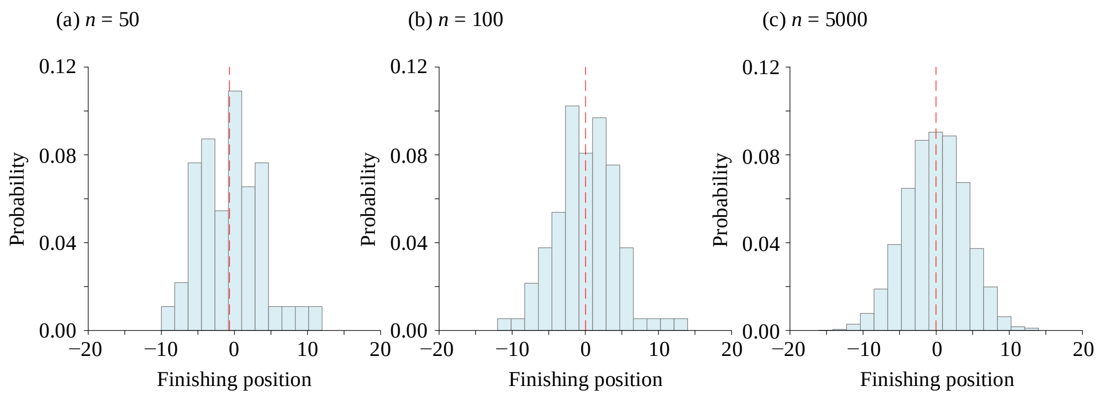


图 12. 第 20 步时随机漫步位置分布，$p = 0.5$

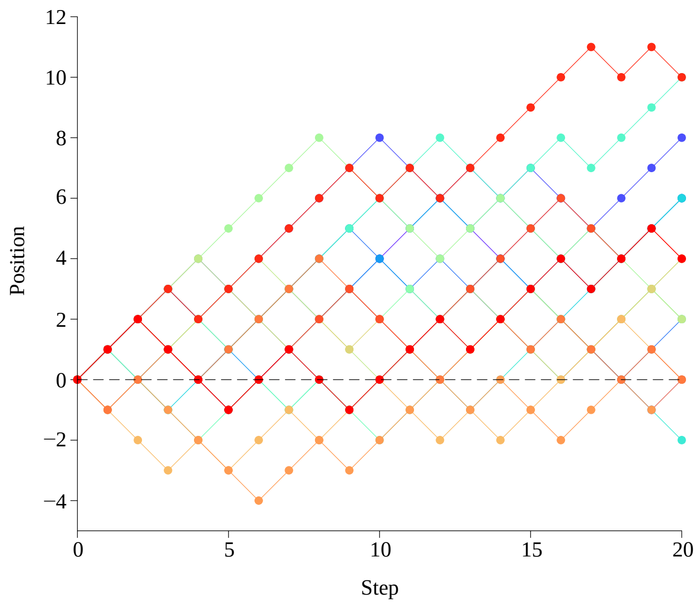


图 13. 二叉树随机行走路径，向上行走的概率 $p = 0.6$

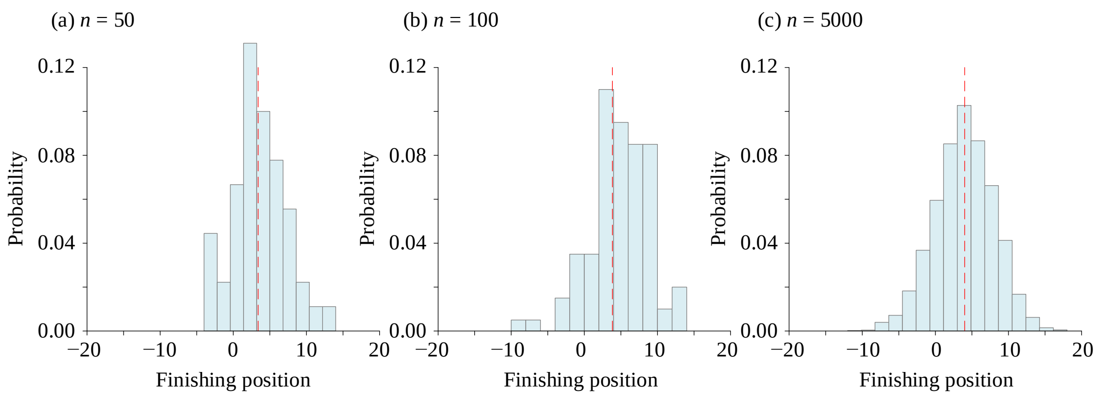


图 14. 第 20 步时随机漫步位置分布，$p = 0.6$

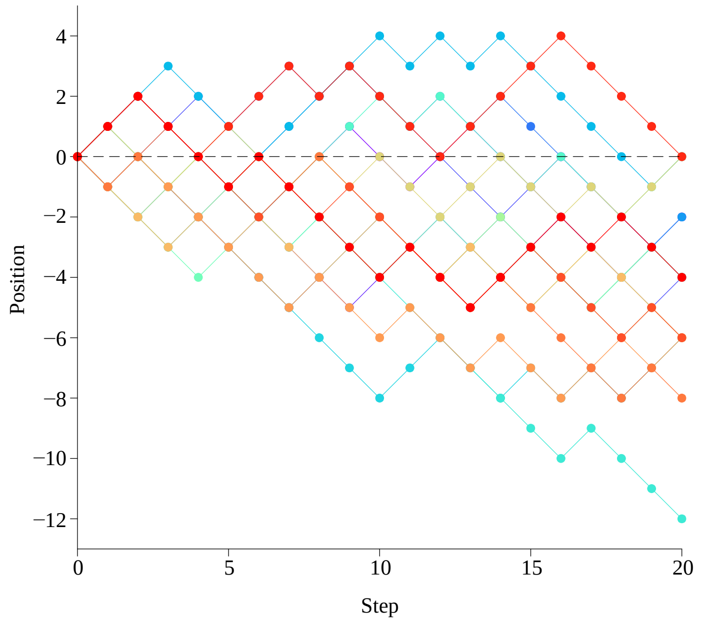


图 15. 二叉树随机行走路径，向上行走的概率 $p = 0.4$

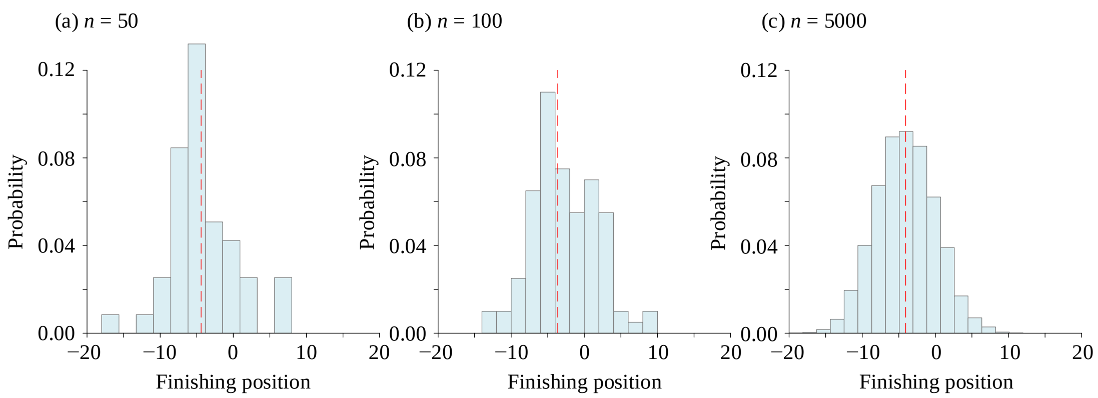


图 16. 向上行走的概率 $p = 0.4$

Bk5_Ch15_06.py 完成本节二叉树随机漫步实验。