

概率分布与线性代数

Python 编程教材与教学方案

体系完善与融合教学版

概念与推导 手算与代码 建模与解读

依据原教案修订

参照麻省理工学院与哈佛大学公开课程

2026 年 10 月

目录

前言.....	7
从零搭建与运行 Python 环境	8
怎样把手算写成程序.....	10
符号与阅读约定.....	12
第 1 章 随机现象与概率语言.....	14
1.1 本章导读 入门 进阶与拓展.....	14
1.2 从一次试验到概率空间	14
1.3 进阶与拓展 事件族 无限样本空间和可测性.....	15
1.4 集合运算 计数与概率加法.....	17
1.5 条件概率 独立性与全概率.....	18
1.6 贝叶斯公式与检测结果的解释.....	20
1.7 相对频数预览与可复现随机数.....	25
1.8 本章要点 常见误区与延伸阅读.....	28
1.9 练习与参考解答.....	29
第 2 章 随机变量 期望与极限定理.....	33
2.1 本章导读 入门 进阶与拓展.....	33
2.2 随机变量与分布函数	33
2.3 期望 方差 指示变量与变换.....	34
2.4 联合 边缘与条件分布.....	36
2.5 条件期望 全期望与全方差.....	37
2.6 协方差 相关与平均的波动.....	40
2.7 大数定律 样本平均为何稳定.....	40
2.8 中心极限定理 标准化平均的分布.....	43
2.9 蒙特卡洛估计 从概率平均到数值精度.....	47
2.10 本章要点 常见误区与延伸阅读.....	50
2.11 练习与参考解答.....	50
第 3 章 离散概率分布.....	54

3.1 本章导读 入门 进阶与拓展	54
3.2 伯努利试验与二项计数	54
3.3 泊松计数与稀有事件极限	57
3.4 几何等待与负二项等待	63
3.5 不放回抽样与超几何分布	67
3.6 经验频率 模拟误差与模型检查	71
3.7 本章要点 常见误区与延伸阅读	73
3.8 练习与参考解答	74
第 4 章 连续概率分布	77
4.1 本章导读 入门 进阶与拓展	77
4.2 概率密度 累积概率与变量变换	77
4.3 均匀分布 长度比例成为概率	78
4.4 正态分布与标准化	81
4.5 泊松过程 从独立增量到指数间隔	85
4.6 指数分布 恒定速率下的等待	90
4.7 伽马分布 把多个等待相加	94
4.8 贝塔分布与有界比例的形状	98
4.9 本章要点 常见误区与延伸阅读	99
4.10 练习与参考解答	99
第 5 章 抽样 区间估计与模型检验	103
5.1 本章导读 入门 进阶与拓展	103
5.2 总体 样本 参数与统计量	103
5.3 频率学与贝叶斯视角 随机的对象是什么	104
5.4 似然与极大似然估计	105
5.5 置信区间与单样本 t 推断	106
5.6 假设检验 p 值与实际差异	107
5.7 贝塔 二项更新与后验可信区间	111
5.8 正态 正态共轭 按精度合并均值信息	115
5.9 分布拟合 分位数图与检验校准	121
5.10 从数据到结论的一份完整说明	126

5.11 本章要点 常见误区与延伸阅读.....	127
5.12 练习与参考解答.....	128
第 6 章 向量空间与矩阵映射.....	132
6.1 学习路线与符号约定.....	132
6.2 向量 线性组合与几何图像.....	132
6.3 矩阵乘法与线性映射.....	136
6.4 数学对象与数组形状.....	139
6.5 子空间 封闭性与反例.....	143
6.6 线性无关 基 维数与坐标.....	145
6.7 内积 正交与正交补.....	146
6.8 从行内积得到四个基本子空间.....	147
6.9 秩 零度定理与解集结构.....	148
6.10 练习与参考解答.....	150
6.11 核心要点与常见误区.....	152
6.12 延伸阅读 入门 进阶与拓展.....	152
第 7 章 线性方程组与正交最小二乘.....	154
7.1 三级学习路线.....	154
7.2 先看一般解与一致性.....	154
7.3 消元 行阶梯形与主元.....	154
7.4 行列式 可逆性与面积体积.....	156
7.5 范数 距离 残差与条件数.....	157
7.6 从一条直线推导正交投影.....	165
7.7 标准正交基与正交 三角分解.....	166
7.8 最小二乘与正规方程的两种推导.....	168
7.9 回归 秩亏与几何解释.....	169
7.10 矩阵与统计的桥梁 中心化及协方差.....	174
7.11 练习与参考解答.....	179
7.12 核心要点与常见误区.....	181
7.13 延伸阅读 入门 进阶与拓展.....	181
第 8 章 特征值 正定矩阵与马尔可夫链.....	183

8.1 三级学习路线.....	183
8.2 特征值是方向上的伸缩因子.....	183
8.3 对角化 相似与矩阵幂.....	187
8.4 实对称 正定与二次型.....	191
8.5 马尔可夫矩阵与行分布约定.....	192
8.6 稳态存在 唯一与趋近要分开判断.....	193
8.7 从概率守恒检查模型.....	197
8.8 练习与参考解答.....	198
8.9 核心要点与常见误区.....	200
8.10 延伸阅读 入门 进阶与拓展.....	200
第 9 章 奇异值分解 低秩近似与主成分分析.....	202
9.1 三级学习路线.....	202
9.2 二维几何 先换坐标 再伸缩 再换回.....	202
9.3 完整 约化与紧致形式的尺寸.....	206
9.4 从实对称谱定理完整构造矩形 SVD.....	207
9.5 四子空间 范数与敏感方向.....	212
9.6 最佳低秩近似 误差下界与达到条件.....	213
9.7 图像矩阵的误差 显示与存储.....	218
9.8 伪逆 最小范数与病态正则化.....	221
9.9 主成分分析与线性组合方差.....	223
9.10 同一结构中的不同任务.....	228
9.11 练习与参考解答.....	228
9.12 核心要点与常见误区.....	231
9.13 延伸阅读 入门 进阶与拓展.....	231
第 10 章 用矩阵工具完成概率统计分析.....	232
10.1 分层学习路线.....	232
10.2 从研究问题确定计算对象.....	232
10.3 浓度数据为什么要考虑分组.....	232
10.4 计数的额外波动从哪里来.....	238
10.5 状态转移怎样传播比例与不确定性.....	243

10.6 用户留存从一次转移到预计持续期 248

10.7 用户画像降维怎样把统计量写成矩阵分解 252

10.8 用真实标定数据贯通最小二乘与随机误差 256

10.9 本章核心要点与常见误区 260

10.10 练习与参考解答 261

10.11 继续阅读 263

配套工具的完整源码 264

附录一 教学安排与案例索引 269

附录二 完整注释代码 273

附录三 公式索引与参考资料 341

前言

概率论研究不确定现象中的数量规律，线性代数研究向量、线性关系与空间结构。它们在数据处理中经常同时出现：一批观测有随机波动，观测之间又存在可以用矩阵表达的关系。本书从这两个出发点讲起，逐步说明如何建立模型、完成计算，以及怎样读懂计算结果。

本书面向学过高中代数、了解函数并能阅读简单 Python 程序的读者。连续分布部分会用到定积分表示面积，推断部分会用到均值和标准差；相关概念均在使用时解释。学习线性代数不要求先掌握多元微积分。读者可以先沿着几何图形理解含义，再回到公式核对条件与符号。

全书前五章讨论概率、随机变量、常用分布和统计推断；第六至第九章依次建立向量空间、线性方程组、正交投影、特征值与矩阵分解的联系；第十章通过浓度、计数、用地转移、用户留存、用户画像和仪器标定六个问题，把概率模型与矩阵工具结合起来。学习时先辨认所讨论的对象，再辨认公式中每个量的含义，最后检查成立条件。

概率部分参考哈佛大学统计学 110 公开课程及布利茨坦、黄的《概率导论》第二版，结合麻省理工学院 18.05 的阅读材料。线性代数部分参考斯特朗的 18.06SC 课程和《线性代数导论》配套学习顺序。配套离线讲义可用于对照课程主题。各章末列出延伸阅读，书末给出具体链接和离线课包文件位置。

阅读可以分两遍。第一遍沿入门路线，先理解对象、图形和一个手算例子，再运行规定参数的案例；第二遍沿进阶路线补齐证明、编程解释和建模边界。标为拓展的测度、谱分解证明与正则化内容不作为首次运行程序的前提，但保留完整表述，供需要深入理解的读者查阅。每章末的十二道练习分为概念与手算、编程实践、场景应用三组，并给出参考解答或具体验收依据。

本书把完整源程序放在对应的知识点之后。每个案例先说明编程思路，再把手算步骤与具体语句对应，随后给出无行号代码框、运行输出、结果图和读图指南。代码框内只保留源

码，文件名写在框外；附录二另按文件名集中收录。绘图案例与共用的 `plot_style.py` 应保存在同一目录。涉及真实观测数据的案例给出来源，其余模拟或假设数据均明确说明用途。

本书的模拟数据用于学习模型，不代表实际调查。涉及概率近似、参数估计或长期预测时，正文同时交代其假设；数值输出展示一次规定环境下的运行结果。读者应把手算、图形和程序作为相互核对的三个渠道。能运行是第一步，能够说明计算对象和结果含义才表明理解了方法。

从零搭建与运行 Python 环境

Python 解释器负责执行程序，NumPy、SciPy 与 Matplotlib 是由解释器加载的库，编辑器则用于编写和保存文件。这三件事需要分别准备。请先解压配套源码，在自己的用户目录建立一个课程文件夹；所有 `.py` 文件、`requirements.txt` 与原有图片保持同级。不要把程序保存在稍后创建的 `.venv` 目录里。

第一步是安装 Python 3.13。进入 Python 官方下载页，查找 3.13 系列的系统安装方式，不要只按首页按钮安装另一主版本。Windows 初学者可使用完整安装器，保留 `pip` 与启动器组件；如勾选加入 `PATH`，安装后需重新打开终端。macOS 使用官方页面提供的系统安装包并按向导完成。某些较后的安全补丁只提供源码，不能假定每个补丁都有安装器；按发布页提供的方式选择可用的 3.13 安装。本文核验环境使用 3.13.16。

第二步是检查解释器并建立课程专用环境。Windows 打开 PowerShell，macOS 打开“终端”。先进入刚解压的源码文件夹：可在文件管理器复制文件夹完整路径，用 `cd` 加引号进入；下列命令从“已经进入源码文件夹”开始。命令是在终端执行的，不要粘贴到出现 `>>>` 的 Python 交互提示符后面。

Windows PowerShell 中建立并使用环境

```
py -3.13 --version
py -3.13 -m venv .venv
.\.venv\Scripts\python.exe -m pip install -r requirements.txt
.\.venv\Scripts\python.exe environment_check.py
.\.venv\Scripts\python.exe e01_rng.py
```


macOS 终端中建立并使用环境

```
python3.13 --version
python3.13 -m venv .venv
./venv/bin/python -m pip install -r requirements.txt
./venv/bin/python environment_check.py
./venv/bin/python e01_rng.py
```

`venv` 为这门课建立独立的库目录，避免把课程依赖和其他项目混在一起。上面的写法直接指定环境内的解释器，不需要额外激活环境。`requirements.txt` 固定 NumPy 2.4.6、SciPy 1.18.0、Matplotlib 3.11.1；使用 `python` 的 `-m pip` 形式，可以明确将库装给当前这个解释器。安装库需要网络，运行本书案例不需要联网下载数据。

第三步是配置编辑器。安装 VS Code 及 Microsoft 发布的 Python 扩展，用“打开文件夹”打开课程源码目录，再从命令面板选择“Python: Select Interpreter”（选择解释器），选中此目录 `.venv` 中的 Python。Windows 的命令面板快捷键通常为 `Ctrl+Shift+P`，macOS 为 `Shift+Command+P`。打开 `environment_check.py` 后使用“Run Python File in Terminal”（在终端运行 Python 文件），检查所打印的版本是否与刚才一致。编辑器只是一种运行入口，不能代替解释器或库。

第四步是读懂一次运行。打开 `e01_rng.py`，先阅读文件开头的数据与参数说明，再运行整份文件。终端显示数值，`e01_rng.png` 写入当前工作目录。绘图案例以保存 PNG 为结果，不要弹出窗口。接着将程序中的随机种子另存修改一份，比较输出变化，并保留未修改版本作基准。不要只选中脚本中间几行运行，否则前面定义的变量可能不存在。

第五步才是批量运行全部案例。终端中将上面的文件名换成 `run_all.py`，即会顺序执行教学案例和环境检查，将 `stdout.txt`、`stderr.txt` 写入 `results` 下对应子目录，PNG 统一保存在源码根目录。成功记录说明程序完成且数值断言通过；能运行仍不等于模型适合某个现实问题，建模假设需要结合正文判断。

表 0-1 常见环境问题与排查方法

出现的现象	先检查什么	可采取的操作
-------	-------	--------

出现的现象	先检查什么	可采取的操作
找不到 python 或 py	解释器是否已安装，终端是否重开	按官方安装指引检查启动器；在 VS Code 选择已安装的 3.13 解释器。
ModuleNotFoundError: numpy/scipy	装库与运行是否使用同一解释器	用 .venv 内的 Python 加 -m pip install -r requirements.txt，随后重跑环境检查。
ModuleNotFoundError: plot_style	共用本地文件是否与案例同级	把配套 plot_style.py 放回同级目录；它是本地文件，不是要另行安装的库。
图中文字变成方框	系统是否具有可用中文字体	按程序提示安装思源黑体或 Noto Sans CJK SC，再运行；字体不是教材分发的数据。
SyntaxError 或 IndentationError	是否复制了文件名、输出或漏掉续页	只复制一个代码框的全部内容，保存为 UTF-8；保留行首空格、普通英文引号与括号。
导入库后报属性不存在	当前目录是否有 numpy.py 等同名文件	将自己的文件改为有区分的名称，重新启动编辑器终端后检查解释器路径。

复制教材代码时，把一个代码框中的全部内容粘贴到编辑器，按框外文件名保存。框内没有行号或表头，跨页的代码仍属于同一文件；视觉上的自动折行不应被手工改成新的逻辑行。首次使用绘图案例时，先确认 plot_style.py 已在同级目录。附录中的代码与正文完全相同，二者任选一处复制即可。

本节延伸阅读：Python 3.13 官方文档“在 Windows 上使用 Python”4.1 与 4.8、“在 macOS 上使用 Python”第 5 章，以及标准库文档 venv 的“创建虚拟环境”与“环境的工作方式”；编辑器操作对照 VS Code 官方“Python 入门”和“Python 环境”页面。具体链接列于附录三。

怎样把手算写成程序

手算时写“先列出可能值，再分别求概率，最后相加”，程序也应保留这条顺序。不要先记函数名，而应先说明输入是什么、输出代表什么、数组中的一个位置对应什么。案例中的数

值断言把手算关系写成可以执行的核对，例如概率和接近 1、矩阵重建接近原矩阵；它们检查某个关系，不替代建模假设。

一个随机数生成器可以看作单独的一本抽样账本。`default_rng(seed)` 创建 Generator 对象，抽样状态保存在这个对象中；早期 `np.random.seed` 配合 `np.random.normal` 等接口通常操作共享的旧式随机状态。二者即使使用同一整数种子，也不应期待产生同一串数。使用局部 `rng` 可避免另一个案例或库调用悄悄消耗同一本账本。固定版本与种子用于复现课堂演示，更换种子则用于观察抽样波动；种子不会消除误差。

数组的 `shape` 先于 `axis`。若 `X` 的形状为 `(n,p)`，约定每行是一名对象、每列是一项特征，则 `axis=0` 指第一个轴，也就是行轴。对它求均值会把 `n` 行归约掉，留下 `p` 个列均值；`axis=1` 会归约列，留下 `n` 个行均值。`axis` 不是“结果沿哪个方向摆放”，而是“哪一个轴参与这次归约”。第七章将用相同约定构造样本协方差矩阵。

表 0-2 数组归约的含义与结果形状

手算对象	NumPy 表达式	得到的形状
每个特征的平均值	<code>X.mean(axis=0)</code>	<code>(p,)</code>
每个对象的特征平均值	<code>X.mean(axis=1)</code>	<code>(n,)</code>
用同一均值行中心化全部对象	<code>X - X.mean(axis=0)</code>	<code>(n,p)</code>
各列的样本标准差	<code>X.std(axis=0, ddof=1)</code>	<code>(p,)</code>

赋值符号 `=` 把计算结果存到变量中；`==` 比较是否相等，浮点数结果通常应按误差容限比较。数学中的 x^2 写作 `x**2`，矩阵乘法写作 `A @ B`，逐元素相乘写作 `A * B`。下标从 0 开始，因此 Python 中的第 0 列对应手算中常说的第一列。读取每个程序前，先在纸上写出重要数组的形状，往往比遇到报错后改参数更有效。

每份案例采用同一学习顺序：先手算一个小结果，再读“编程思路”和“代码对应”，运行完整脚本，最后根据“易错提示”和读图指南解释输出。编程练习中的“不得调用某函数”用于检验

原理理解，正式分析时仍应优先采用经过验证的数值接口。完整批运行工具与字体工具源码放在正文末“配套工具的完整源码”，首次学习可先运行附件而暂不逐行阅读工具实现。

符号与阅读约定

本书用大写 X 、 Y 表示随机变量，小写 x 、 y 表示它们可能取得的数值或已经观察到的结果； $P(A)$ 是事件 A 的概率， $E[X]$ 是随机变量 X 的期望。花括号表示集合，竖线 $|$ 在条件概率中表示给定条件；同一个符号在矩阵行列式或绝对值中有不同用途，应结合对象辨认。

Σ 表示按指标逐项相加， Π 表示逐项相乘；它们不是希腊字母 Σ 的另一种拼写。在线性代数中， Σ 常表示奇异值组成的对角形矩阵，在概率中 σ 通常表示标准差。下标 i 、 j 用于编号；上标 T 表示转置；可逆矩阵的上标 -1 表示逆矩阵，不是把每个元素分别取倒数。

实数集合记作 $\mathbb{R}$ ， $\mathbb{R}^n$ 表示有 n 个实数分量的向量空间， $\mathbb{R}^{m \times n}$ 表示 m 行 n 列的实矩阵集合。向量在代数推导中默认写成列向量，马尔可夫链章节为配合转移矩阵的行概率约定，会明确改用概率行向量。维数、形状和乘法方向必须一同检查。

$\approx$ 表示近似相等， $\propto$ 表示两边相差一个与所讨论变量无关的比例常数， $\sim$ 表示服从某一分布。概率密度可能大于 1，但事件概率只能在 0 与 1 之间。公式后的解释会说明参数范围；不满足范围的输入，即使某段程序没有立即报错，也没有相应模型含义。

符号会随对象改变含义。求和运算使用 Σ ；大写希腊字母 Σ 在一些概率文献中表示协方差矩阵，在本书第九章则表示奇异值组成的矩形对角矩阵。为了减少混淆，本书协方差矩阵通常写 C 。每次出现矩阵时均同时说明形状，不能仅凭字母外观判断它是标量、向量还是算子。

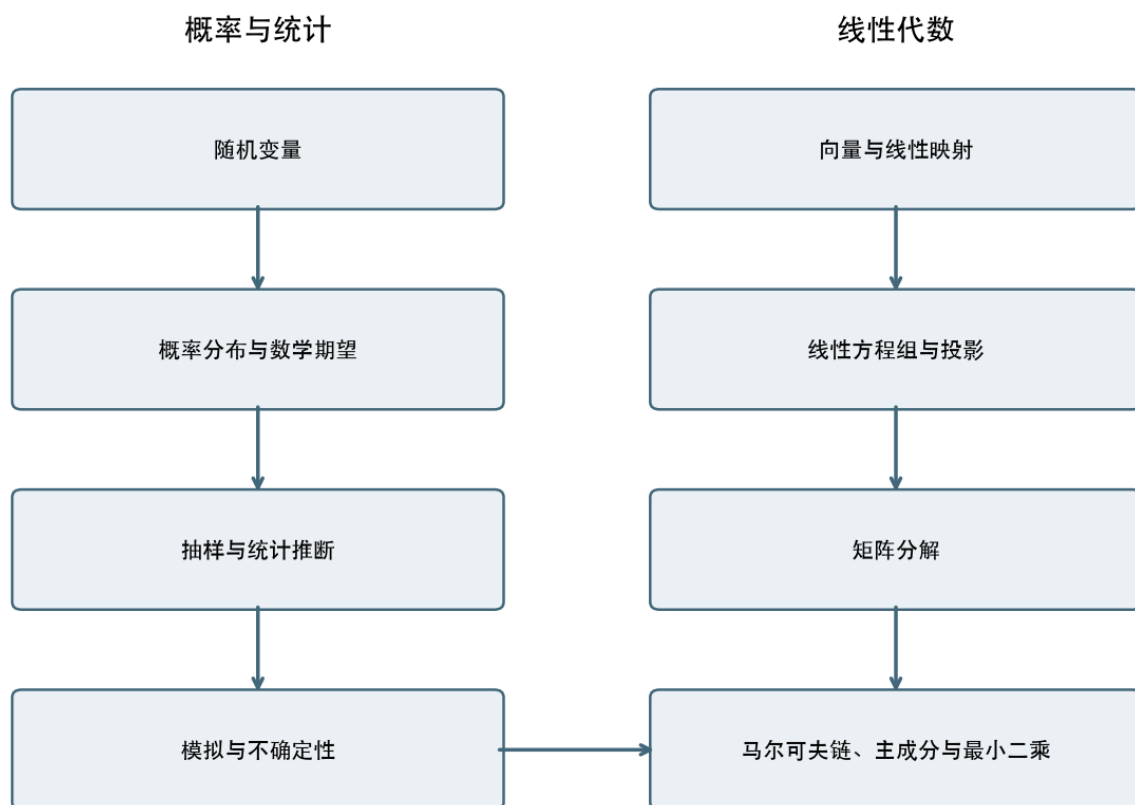


图 0-1 概率模型描述不确定性，矩阵运算组织数量关系。先分别掌握两条线索，再在综合应用中把它们连接起来。对应程序 `g00_knowledge_map.py`。

读图指南 先沿左列看“随机变量→分布→抽样推断”，再沿右列看“向量→方程→矩阵分解”。箭头表示学习和计算的联系，不代表统计因果；第七章的协方差矩阵与第十章的综合案例会把两条线索接起来。

第 1 章 随机现象与概率语言

1.1 本章导读 入门 进阶与拓展

本章先把“可能发生什么”与“怎样分配机会”说清楚。入门阅读关注样本空间、事件、条件概率与贝叶斯公式，能用有限集合和树状分解完成计算即可。进阶阅读解释事件族的封闭性、可数加性，以及从两类别到多类别的贝叶斯更新。拓展阅读给出无限样本空间、可测性的严格表述；这些概念说明模型为何自治，暂时不掌握测度论的构造也能继续学习。

1.2 从一次试验到概率空间

同一枚骰子重复投掷，点数可能不同；一条生产线重复抽检，次品数也可能不同。我们无法由一次结果消除这种不确定性，却可以明确规定哪些结果可能发生、哪些结果属于同一事件，并用概率比较它们的机会。概率模型由“试验怎样进行”和“怎样给结果赋权”共同决定。只说“随机抽一个”还不完整：抽取对象、抽取方式、是否放回以及观察内容都可能改变模型。

先以掷两枚可区分的公平骰子为例。一次结果可记为有序对 $\omega=(i,j)$ ，其中 i 是第一枚骰子的点数， j 是第二枚骰子的点数。样本空间 Ω 是所有 36 个有序对的集合。事件 A “点数和为 7”包含(1,6)、(2,5)等 6 个结果；事件 B “第一枚为 6”包含(6,1)至(6,6)。基本结果是一次试验最细的记录，事件是若干基本结果组成的集合。二者处于不同层次，不能把“一个事件”总理解成“一个点”。

一个概率模型可记作 $(\Omega, \mathcal{F}, P)$ 。 Ω 是所有允许基本结果组成的样本空间； $\mathcal{F}$ 是我们允许提问并计算概率的事件族； P 为这些事件分配概率。先想有限骰子即可： Ω 有 36 个点， $\mathcal{F}$ 可以取所有子集， P 由每个点的权重确定。接下来的公理并不依赖等可能性。它们也适用于抽样偏好不同或样本空间无限的模型。[P01、P02]

$$P(A) \geq 0, \quad P(\Omega) = 1, \quad P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i) \quad (A_i \cap A_j = \emptyset, i \neq j) \quad (1.1)$$

A 、 A_i 和 A_j 均为事件； $\cup$ 表示并集，即至少一个发生； $\cap$ 表示交集，即同时发生； $\emptyset$ 表示空集； $\sum$ 表示逐项求和； ∞ 表示项数可以没有上限。第三条要求事件两两互斥，称为可列可加性。 $P(A) \geq 0$ 说明概率非负， $P(\Omega)=1$ 说明必有某个允许结果发生。

公平且独立的两骰子模型给每个有序对概率 $1/36$ ，所以点数和为 7 的概率为 $6/36=1/6$ 。若改用点数和作样本空间，其可能值只有 2 至 12，但这 11 个值的概率不同。把结果写进同一个集合并不能推出等可能性。有限且等可能时才能使用“有利结果数除以总结果数”；一般情形必须把事件内各结果的概率相加。

概率是模型中的量，频率是一次或一组观测得到的量。公平骰子的理论概率 $1/6$ 并不保证每 6 次恰好各点数出现一次。建立模型时可以依据对称性、实验机制和已有信息；得到数据后再检查模型是否与实际相容。这样做把“假设是什么”和“观测到了什么”分开，也为后面的统计推断留下了清楚的位置。

1.3 进阶与拓展 事件族 无限样本空间和可测性

为什么事件要组成一个族？如果已经能够判断事件 A 是否发生，也应能够判断“ A 没有发生”；若能够判断 A_1 、 A_2 等一串事件，也希望能够提问“其中至少一个发生”。对这些操作保持封闭，意味着由已有合法问题构成的新问题仍然合法。这里的“封闭”不是概率值相同，而是操作后的集合仍属于允许的事件族。

严格地说， σ 代数 $\mathcal{F}$ 是 Ω 的子集所组成的一个集合族，满足三条要求： Ω 属于 $\mathcal{F}$ ；只要 A 属于 $\mathcal{F}$ ，其补集 $\Omega \setminus A$ 也属于 $\mathcal{F}$ ；只要 A_1 、 A_2 、...都属于 $\mathcal{F}$ ，它们的可数并也属于 $\mathcal{F}$ 。“可数”表示可按 1、2、3、...列出，既包括有限个，也包括可列无穷个。由补集与并集的关系，还能推出对可数交封闭。 σ 代数不是要求对任意不可数并或不可数交封闭。[P15]

$$\Omega \in \mathcal{F}, \quad A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}, \quad A_i \in \mathcal{F} (i \geq 1) \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{F} \quad (1.2)$$

$\mathcal{F}$ 是事件族； i 为正整数编号。第三条允许可数多个已有事件组成并事件。取 $A=\Omega$ 得 $\emptyset$ 属于 $\mathcal{F}$ ；有限并可在序列后补空集得到。

$$\bigcap_{i=1}^{\infty} A_i = (\bigcup_{i=1}^{\infty} A_i^c)^c \quad (1.3)$$

这是德摩根关系：所有 A_i 都发生，等价于“至少一个 A_i 不发生”的补事件。右侧依次使用补集、可数并和补集，所以仍属于 $\mathcal{F}$ 。

例题 有限信息也能形成事件族

掷骰子的真实结果仍是 1 至 6，但仪器只显示“奇”或“偶”。若只允许使用仪器提供的信息，可取 $\mathcal{F} = \{\emptyset, \Omega, \{1,3,5\}, \{2,4,6\}\}$ 。这个族对补集与可数并封闭。事件“点数恰为 1”并非不存在，而是当前信息结构没有把它作为可区分事件。若改为直接观察点数，就可取全部子集，信息更细。

在可数样本空间上，可给每个点 ω_j 非负权重 p_j ，并要求所有权重的和为 1，再把事件概率定义为事件内点权重之和。例如第一次掷到 6 所需的次数可取任意正整数；给第 k 次首次成功的权重 $(5/6)^{k-1}/6$ ，这些权重组成收敛级数。第 3 章会把它称为几何分布。可数无限没有“所有结果数”这个有限分母，仍然有一致的加权概率。

不可数样本空间更能说明为什么不能只靠点概率。把一根长度 1 的线段作为 $\Omega=[0,1]$ ，均匀选点的含义是区间概率等于区间长度。例如 $P(0.2 \leq U \leq 0.5) = 0.3$ ，任意指定单点 a 的概率为 0。可以在包含所有区间的波莱尔 σ 代数上定义这种长度概率；波莱尔 σ 代数是开区间生成的最小 σ 代数。普通区间、可数个区间的并交，以及本书使用的阈值事件都在其中。这里给出的是标准模型的含义，不要求读者自行完成长度测度的构造。

$$P(a \leq U \leq b) = b - a \quad (0 \leq a \leq b \leq 1), \quad P(U = a) = 0 \quad (1.4)$$

U 表示线段上选到的位置， a 、 b 为事先指定的端点。因为单点长度为 0，端点开闭不影响这个均匀模型的区间概率；这并不是离散模型的一般性质。

单点概率为 0 与总概率为 1 并不矛盾。概率公理允许相加的是可数个两两互斥事件，而 $[0,1]$ 含不可数多个单点；不能把“不可数多个 0 相加”当作普通级数。相反，任何可数集合，例

如 $[0,1]$ 中的有理点，确实是可数个零概率单点的并，所以概率仍为 0。在这个模型中选到无理点的概率为 1，但并没有某一个无理点具有正概率。

可数加性还使极限事件可以计算。设 $A_n=[0,1-1/n]$ ， n 从 1 递增，这些事件逐渐扩大，并集是 $[0,1)$ 。它们的概率 $1-1/n$ 趋于 1，恰等于并事件的概率。一般地，递增事件的并概率等于概率的极限；这一性质可把逐步逼近与概率计算联系起来。有限加性本身不能替代这种对可数极限的一致要求。

随机变量的通俗含义是把一次基本结果转换为一个数。可测性的要求是：转换后提出的普通数值问题，必须能拉回原样本空间中一个允许的事件。设 $X:\Omega\rightarrow\mathbb{R}$ ，对数轴上的集合 B ， $X^{-1}(B)$ 表示所有满足 $X(\omega)$ 属于 B 的基本结果；这里的 $^{-1}$ 表示原像，不要求 X 可逆，也不是 $1/X$ 。

$$X^{-1}(B) = \{\omega \in \Omega: X(\omega) \in B\}, \quad B \in \mathcal{B}(\mathbb{R}) \Rightarrow X^{-1}(B) \in \mathcal{F} \quad (1.5)$$

$\mathcal{B}(\mathbb{R})$ 为实数轴上的波莱尔 σ 代数。若对其中每个 B 都满足右侧条件，则 X 为实值可测随机变量。实际验证时，等价地只需检查每个实数 x 的阈值原像 $\{\omega: X(\omega) \leq x\}$ 属于 $\mathcal{F}$ 。[P16]

例题 从可观察信息检查可测性

沿用只显示奇偶的骰子仪器。 X =“偶数时记 1、奇数时记 0”可测，因为所有阈值原像都只能是 $\emptyset$ 、奇数集、偶数集或 Ω 。若 Y 直接等于骰子点数，则事件 $\{Y \leq 1\}=\{1\}$ 不在这个较粗的 $\mathcal{F}$ 中， Y 相对于当前信息不满足可测性。把 $\mathcal{F}$ 改成全部子集后， Y 立即成为可测随机变量。可测性总是相对于指定事件族而言。

第一次阅读到这里，只需保留三点：事件族保证集合操作后的问题仍可计算；概率测度用可数加性使无限模型自洽；可测随机变量保证数值问题有合法的原事件。第 2 章将在这套语言上定义分布、期望与方差。有限骰子计算不需要先学习更深的测度论，这层说明可以在理解连续分布后回读。

1.4 集合运算 计数与概率加法

事件的语言首先是集合语言。 A^c 表示 Ω 中不属于 A 的结果，读作 A 的补事件； $A \subseteq B$ 表示 A 发生必然伴随 B 发生。由概率公理可得 $P(\emptyset)=0$ 、 $P(A^c)=1-P(A)$ ，以及 $A \subseteq B$ 时 $P(A) \leq P(B)$ 。这些关系提供了最直接的结果检查：若算出的概率为负数或大于1，计算或模型一定有问题；若“小事件”的概率大于包含它的“大事件”，同样应回头检查。

两个事件可以重叠。例如一次公平骰子中， $A=\{2,4,6\}$ 表示偶数， $B=\{4,5,6\}$ 表示大于3。直接相加 $P(A)+P(B)$ 时，4与6各算了两遍，因此要减去一次交集。若 A 与 B 互斥，则交集为空，才可不加修正地相加。对于“至少一次”这种事件，直接列举常常繁琐，先算补事件“所有次都没有”通常更简洁。

$$P(A \cup B) = P(A) + P(B) - P(A \cap B), \quad \binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1.6)$$

左式是两事件加法公式。右式的组合数表示从 n 个不同对象中选出 k 个且不计顺序的选法数； n 和 k 为整数， $0 \leq k \leq n$ 。 $n!$ 读作 n 的阶乘，等于 $1 \times 2 \times \cdots \times n$ ，并约定 $0!=1$ 。组合数本身只是计数，转成概率还需说明哪些选法等可能。

例题 手算 1-1：顺序、放回与样本空间

袋中有2个红球和3个蓝球，各球可区分。不放回抽2个，且每个两球子集等可能。总选法为 $C(5,2)=10$ ；恰有1个红球的选法为 $C(2,1)C(3,1)=6$ ，所以概率为 $3/5$ 。

也可按有序抽取计算：先红后蓝概率为 $(2/5)(3/4)=3/10$ ，先蓝后红概率也为 $3/10$ ，两种顺序互斥，相加仍得 $3/5$ 。若放回，第二次的分母应恢复为5，答案改为 $2(2/5)(3/5)=12/25$ 。改变抽样机制会改变概率。

计数的关键是使分子与分母采用相同口径。若分母把先抽甲后抽乙和先抽乙后抽甲算成两种结果，分子也要计顺序；若分母使用组合数，分子也要按无序子集计数。把“排列”“组合”作为两条待套用的公式容易混淆，先写清楚一次结果的记录方式，往往就能决定该用哪一种。

1.5 条件概率 独立性与全概率

已知一个事件发生，会改变可讨论的结果范围。掷一枚公平骰子，令 A 表示偶数、B 表示大于 3。未获得信息时， $P(A)=1/2$ ；已知 B 发生后，结果只剩 4、5、6，其中两个是偶数，所以条件概率 $P(A|B)=2/3$ 。竖线左边是所问事件，右边是已经给出的条件。计算 $P(A|B)$ 时不是把 A 和 B 换个位置，而是先把参考范围限制在 B，再重新归一化。[P03]

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0; \quad P(A \cap B) = P(B)P(A|B) \quad (1.7)$$

$P(A|B)$ 读作“在 B 发生的条件下 A 发生的概率”。分母 $P(B)>0$ 是这个比值定义的前提；分子只保留 A、B 共同发生的部分。右式是乘法公式，它适用于有关联的事件，不要求独立。

若 $P(A \cap B)=P(A)P(B)$ ，称 A 与 B 独立。 $P(B)>0$ 时，这等价于 $P(A|B)=P(A)$ ：得知 B 不会改变 A 的概率。独立性是一条关于联合发生机会的模型性质，不能仅凭两个事件“看起来不相关”来断言。两个概率均为正的互斥事件不独立，因为一个发生后另一个的条件概率降为 0。讨论三个以上事件时，“两两独立”也弱于“相互独立”；后者还要求任意子组共同发生的概率都等于各自概率的乘积。

全概率公式适合分情景计算。若 $B_1, \dots, B_m$ 两两互斥、并集为 Ω ，且每个 $P(B_j)>0$ ，它们构成样本空间的一个划分。事件 A 可以拆为 $A \cap B_1$ 至 $A \cap B_m$ 这些不重叠的部分。先在每个情景内计算 A 的条件概率，再按情景发生机会加权，就能恢复总体概率。所谓加权并非技术修饰：不同情景占比不同，简单平均通常是错的。

$$P(A) = \sum_{j=1}^m P(A|B_j)P(B_j) \quad (1.8)$$

j 是情景编号，m 是情景总数， B_j 是第 j 个情景。条件概率 $P(A|B_j)$ 是情景内发生比例， $P(B_j)$ 是情景权重，各权重非负且总和为 1。公式来自对互斥交集 $A \cap B_j$ 逐项相加。

例题 手算 1-2：先分组，再合并

某批产品 60%来自甲线、40%来自乙线，甲线次品率为 1%，乙线次品率为 4%。随机抽取一件的次品概率为 $0.6 \times 0.01 + 0.4 \times 0.04 = 0.022$ ，即 2.2%。直接平均 1%与 4%得到 2.5%，忽略了两线产量不同。

已知抽到次品时，产品来自乙线的概率为 $0.4 \times 0.04 / 0.022 = 8/11$ 。总体次品率与次品中乙线占比回答不同问题，分母分别是全体产品和全部次品。

1.6 贝叶斯公式与检测结果的解释

许多问题容易知道“某种原因成立时会出现什么结果”，却真正想问“看到了这个结果，原因有多大可能成立”。贝叶斯公式把同一个交集写成两种乘法形式，由 $P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$ ，再除以 $P(B)$ 即可。它没有创造额外证据，而是把已知概率与新信息放在一致的分母下计算。[P03]

在检测情景中，D 表示产品确为次品，“+”表示检测为阳性。 $P(D)$ 是检测前的次品率，称为先验概率； $P(+|D)$ 是灵敏度； $P(-|D^c)$ 是特异度； $P(+|D^c) = 1 - \text{特异度}$ 是假阳性率。所问 $P(D|+)$ 称为阳性后的后验概率。灵敏度的分母是实际次品，后验的分母是全部阳性，两者不能互换。

$$P(D|+) = \frac{P(+|D)P(D)}{P(+|D)P(D) + P(+|D^c)P(D^c)} \quad (1.9)$$

D^c 表示非次品；分母用 D 与 D^c 的划分求出阳性总概率。分子是真阳性概率，分母还包括假阳性概率。该公式要求阳性总概率大于 0，并假定给定检测性能适用于当前产品总体。

例题 手算 1-3：低先验下为何有许多假阳性

取先验次品率 0.1%、灵敏度 0.99、特异度 0.95。假想检查 100000 件，模型预期有 100 件次品，其中 99 件阳性；99900 件非次品中预期有 4995 件假阳性。全部阳性约 5094 件，所以阳性中次品的比例为 $99/5094 \approx 1.9435\%$ 。

这些件数是按模型比例折算的数量，实际抽检数会波动；这里仅用比例计算，尚不需要期望符号。低后验并不表示检测“没有信息”：检测前次品概率为 0.1%，检测后约为 1.94%，确实提高了。只是非次品基数很大，5%的假阳性率仍带来了许多假阳性。

程序还用优势形式核验。若 $0 < p < 1$ ，概率 p 对应的优势为 $p/(1-p)$ ，比较的是事件与其补事件的机会之比；优势可以大于 1，不能直接当概率。阳性似然比为 $P(+|D)/P(+|D^c)$ ，表示同一阳性证据在两类产品下的相对可能性。由贝叶斯公式分别写出两类后验再相除，公共分母消去，便得下面的乘法。

$$\frac{P(D|+)}{P(D^c|+)} = \frac{P(D)}{P(D^c)} \frac{P(+|D)}{P(+|D^c)} \quad (1.10)$$

左边为后验优势，右边为先验优势乘阳性似然比。相关分母须为正；优势 o 转回概率时用 $o/(1+o)$ 。本例似然比 $0.99/0.05=19.8$ 。

程序 `e06_bayes.py` 程序固定灵敏度 0.99、特异度 0.95，分别使用 0.001、0.01、0.05、0.10 四个先验次品率。`prior`、`true_positive`、`false_positive`、`posterior` 都是长度 4 的数组，同一位置对应同一情景。逐元素除法实现四次贝叶斯更新；优势比核验使用“后验优势=先验优势×似然比”，其中阳性的似然比为 $0.99/0.05=19.8$ 。读输出时应保持先验与后验在同一行，不把四个情景混为一次抽样。

编程思路 先分别计算真阳性与假阳性的联合概率，再以所有阳性的概率作分母，求给定阳性的次品概率。另用先验赔率乘似然比复算，并用十万件产品的期望数量解释低基率效应。

代码对应

$P(D \cap +) = \text{灵敏度} \times \text{次品率} \rightarrow \text{true_positive} = \text{sensitivity} * \text{prior}$: `prior` 同时存四种次品率，向量计算保证检验性能相同而基率不同。

$P(D^c \cap +) = (1 - \text{特异度}) \times (1 - \text{次品率}) \rightarrow \text{false_positive} = (1 - \text{specificity}) * (1 - \text{prior})$: 假阳性来自合格品，不能使用次品率作其数量比例。

$P(D|+) = P(D \cap +) / P(+)$ $\rightarrow \text{posterior} = \text{true_positive} / (\text{true_positive} + \text{false_positive})$: 分母把真阳性与假阳性相加，是全概率公式。

后验赔率=先验赔率×阳性似然比 $\rightarrow \text{odds_post} = \text{odds_prior} * \text{ratio}$: 这是同一贝叶斯更新的赔率形式, 赔率还需换算才是概率。

易错提示 灵敏度是 $P(+|D)$, 后验是 $P(D|+)$, 条件方向不可交换。柱图期望件数来自概率乘产品总数, 不是一次随机抽样得到的实际件数。

正文代码 e06_bayes.py

```
# 教材位置: 第1章, 第1.6节; 图 1-1。
# 单位、变量与形状: prior、posterior 为 (4,); 概率无量纲, 图中乘 100 表示百分比, 期望数量
# 按件计。
# 学习目标: 理解阳性检测结果怎样更新次品概率, 以及为什么基准次品率会影响后验概率。
# 核心关系: 设 D 为次品、+ 为阳性, 则  $P(D|+) = P(+|D)P(D)/P(+)$ ;
#  $P(+)=P(+|D)P(D)+P(+|\text{非}D)P(\text{非}D)$ , 后验优势=先验优势×阳性似然比。
# 数据与输出: 对四种假定次品率使用同一组灵敏度、特异度, 逐行打印先验和阳性后的后验。
# 这些参数用于教学演示, 结果是给定模型下的理论概率, 不是从产品抽检数据估计出的产品质量结论。
# 运行方式: 安装 NumPy、Matplotlib 后执行 python e06_bayes.py; 输出终端结果和教材图
# e06_bayes.png。

import numpy as np

# prior 形状为 (4,), 各元素是不同情景的  $P(D)$ , 并非四个互斥类别的概率。
# 因此这四个数不需要相加为 1; 数组运算让四种情景一次完成相同的公式计算。
prior = np.array([0.001, 0.01, 0.05, 0.10])
# 灵敏度= $P(+|D)$ , 特异度= $P(-|\text{非}D)$ ; 二者都不能直接解释成  $P(D|+)$ 。
sensitivity, specificity = 0.99, 0.95
# 乘法公式得到联合概率  $P(D \text{ 且 } +)$ , 不是只针对次品群体内部的条件比例。
true_positive = sensitivity * prior
# 1-specificity 为假阳性率  $P(+|\text{非}D)$ ; 再乘合格品比例, 得到  $P(\text{非}D \text{ 且 } +)$ 。
false_positive = (1 - specificity) * (1 - prior)
# 分母把所有阳性来源相加, 得到  $P(+)$ ; 分子占其中的比例才是阳性后的次品概率。
# 即使检测参数相同, 先验很低时假阳性仍可能占阳性样本中的较大部分。
posterior = true_positive / (true_positive + false_positive)
# 优势= $p/(1-p)$  比较“发生”与“不发生”的概率, 不是概率本身, 允许大于 1。
odds_prior = prior / (1 - prior)
# 阳性似然比比较阳性结果在次品与合格品两种条件下的可能性, 不是后验概率。
ratio = sensitivity / (1 - specificity)
# 贝叶斯公式的优势写法将证据更新化为乘法; 标量 ratio 会广播到四个情景。
odds_post = odds_prior * ratio
# 用  $p=\text{优势}/(1+\text{优势})$  转回概率, 验证概率写法和优势写法逐元素一致。
# allclose 允许浮点舍入误差; 这里只检验两种代数形式, 没有进行随机模拟。
assert np.allclose(posterior, odds_post / (1 + odds_post))
# zip 按对应位置配对先验、后验; 百分比格式只影响显示, 不改变内部计算使用的小数。
for pre, post in zip(prior, posterior):
    print(f"先验={pre:.1%}, 后验={post:.2%}")

# 教材图 1-1 只采用上面已经计算的四种先验情景; 不增加新的检测参数或随机样本。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
```

```
# 左图比较同一检测规则下的先验和阳性后验；乘 100 只改变显示单位为百分数。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
positions = np.arange(prior.size)
axes[0].bar(positions - 0.18, 100 * prior, width=0.36, label='先验 P(D)')
axes[0].bar(positions + 0.18, 100 * posterior, width=0.36, label='后验 P(D | +)')
axes[0].set(xticks=positions, xticklabels=[f'{value:.1%}' for value in prior],
            xlabel='先验次品率', ylabel='概率 (%)',
            title='四种先验情景', ylim=(0, 85))
# 后验百分数直接来自 posterior；避免在图中硬写与计算结果分离的数字。
for position, value in zip(positions, posterior):
    axes[0].text(position + 0.18, 100 * value + 1.5, f'{value:.2%}',
                 ha='center', fontsize=7.8)
axes[0].legend(fontsize=7.8, loc='upper left')
# 右图只把第一种先验情景换算为 10 万件产品的期望数量，未把概率改成实际观测频数。
population_size = 100000
positive_counts = population_size * np.array([true_positive[0], false_positive[0]])
axes[1].bar(['真阳性', '假阳性'], positive_counts,
            color=['C1', 'C0'], width=0.55)
for position, value in enumerate(positive_counts):
    axes[1].text(position, value + 100, f'{value:.0f}', ha='center', fontsize=8)
axes[1].set(ylabel='产品期望件数', ylim=(0, 6600),
            title=f'{population_size:,} 件；先验={prior[0]:.1%}')
# 图中的阳性件数分母与上面 posterior[0]使用同一个全概率分母。
axes[1].text(0.5, 0.95,
            f'P(D | +) = {positive_counts[0]:.0f} / {positive_counts.sum():.0f}'
            f' = {posterior[0]:.2%}',
            transform=axes[1].transAxes, ha='center', fontsize=7.8)
fig.savefig('e06_bayes.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

先验=0.1%，后验=1.94%

先验=1.0%，后验=16.67%

先验=5.0%，后验=51.03%

先验=10.0%，后验=68.75%

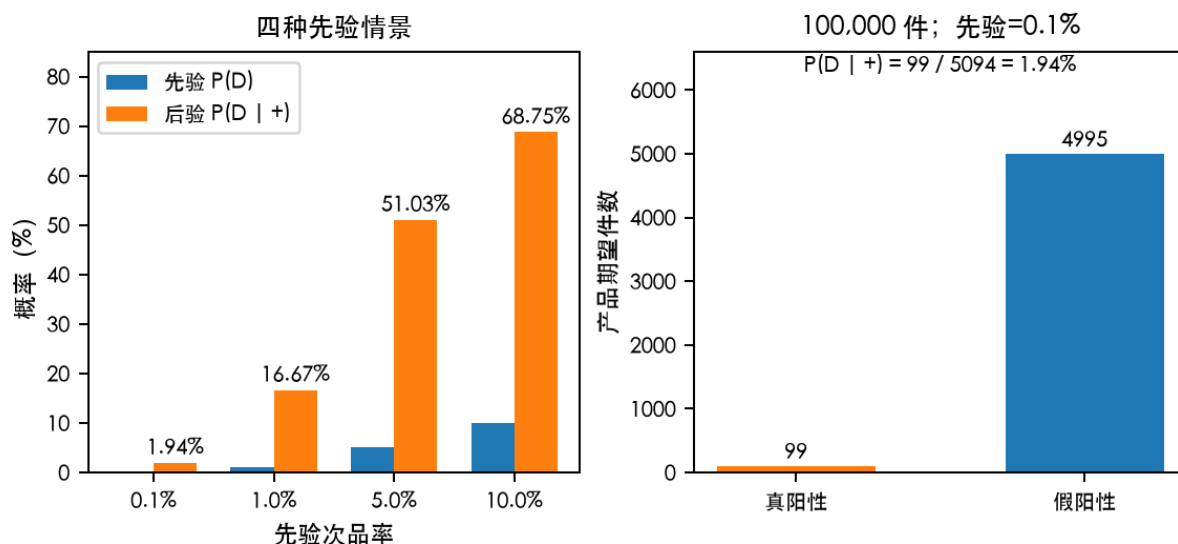


图 1-1 产品检测中的先验与阳性后验，由 `e06_bayes.py` 生成。灵敏度 0.99、特异度 0.95，左图比较四种先验次品率；右图只对先验 0.1%、产品总数 100000 的情景显示真阳性 99 件、假阳性 4995 件的按比例折算数量。

读图指南 左图每组两根柱必须使用同一个先验情景比较；右图只对应先验 0.1%。右图真阳性柱低而假阳性柱高，来自两类产品基数不同，不表示灵敏度只有 1.94%。

图中右面板只展示先验 0.1% 的情景；左面板比较四个不同先验。把右面板的件数换成其他情景，必须同步更换先验，不能沿用 99 与 4995。若同一产品重复检测，只有在给定真实类别后两次结果条件独立时，才可以把两次阳性似然直接相乘。共享的操作误差或设备偏差可能破坏这种独立性，单次性能指标不足以决定重复检测的后验。

两个类别并不是贝叶斯公式的限制。若 H_1 至 H_m 是互斥且穷尽的候选类别，观察到证据 E 后，每个类别先计算“先验×该类别下证据的似然”，再用所有类别所得数值的总和归一化。分母相同，所以后验和仍为 1；后验较大的类别，兼有先验上较常见与更容易产生当前证据这两方面的支持。

$$P(H_j|E) = \frac{P(E|H_j)P(H_j)}{\sum_{k=1}^m P(E|H_k)P(H_k)} \quad (j = 1, \dots, m) \quad (1.11)$$

H_j 为第 j 个类别， m 为类别总数； E 为已观察证据，要求分母 $P(E) > 0$ 。求和指标 k 只用于遍历全部类别，与正在计算的 j 区分。若某类别的先验为 0，这条更新本身不会把它变为正概率。

例题 手算：三类来源的完整更新

以下是教学假设，不是企业实测数据。某服务请求来自甲、乙、丙渠道的概率分别为 0.50、0.30、0.20。三个渠道出现“需二次处理”标记的条件概率依次为 0.02、0.04、0.10。先验和为 1；观察到标记后，三个未归一化权重为 0.010、0.012、0.020，证据总概率为 0.042。

因此后验依次为 $5/21 \approx 23.81\%$ 、 $2/7 \approx 28.57\%$ 、 $10/21 \approx 47.62\%$ ，总和为 1。丙原先只占 20%，却更容易产生这一标记，因而在标记请求中占比升高。不能把 10%的条件概率直接称为丙的后验，也不能只比较似然而丢掉渠道占比。

若再观察到一个新标记，须使用“新标记在给定类别及已有标记条件下的概率”继续更新。只有假设两个标记在给定类别后条件独立时，才可把同一类别下两个单标记似然相乘。类别之间互斥并不意味着同一类别内的证据自动独立。

1.7 相对频数预览与可复现随机数

本节只预览如何记录随机结果与相对频数，无须使用期望、标准误或显著性检验。若掷 8 次骰子，其中点数 6 出现 2 次，相对频数就是 $2/8$ ；它是当前记录中的比例，不等于理论概率已经改为 $1/4$ 。计算机模拟把指定模型转成具体记录，先检查取值是否合法、种子能否复现，以及计数的分母是否正确。为什么频数在大量重复中稳定，将在第 2 章用大数定律解释；怎样评估误差与检验模型，留到第 3 章和第 5 章。

常用随机数实际上由确定算法生成，称为伪随机数。种子提供初始化状态，发生器每调用一次就推进状态。相同发生器算法、软件环境、种子与调用顺序有助于复现同一序列；种子不控制理论概率，也不保证不同种子对应的有限样本一定不同。完整记录脚本与依赖版本，比单独记下一个种子更有用。

现代 NumPy 通常先用 `default_rng(seed)` 建立一个 Generator 对象，再由这个对象抽样。对象拥有自己的状态，便于让不同任务分别管理随机序列；传统 `np.random.seed` 与 `np.random.*` 主要操作共享的旧式全局状态。相同整数种子若配上不同发生器算法，并不应期待相同数列。

教学中固定对象、种子、版本与调用顺序，是为了复现实验记录；重新设置种子不能使模型更正确。

程序 e01_rng.py rng_a 和 rng_b 都由种子 2026 初始化，各调用 integers(1,7,size=8)产生 8 个骰子点数。下界 1 包括在内，上界 7 不包括在内，返回数组形状为(8,)。两对象首次抽取 a 与 b 应相同；rng_a 再次抽取 next_a 时从当前状态继续。array_equal 验证两次首次结果逐元素相同，all 验证点数范围。图使用 1 至 6 各 1/6 的完整理论概率，不把 8 次样本的频率画成理论分布。

编程思路 先建立分开保存状态、种子相同的两个 Generator，比较各自第一批整数，再继续使用其中一个生成器。区分八次随机样本与理论概率；图中展示骰子的理论质量函数与累积分布函数。

代码对应

以固定初始状态定义一次可复现的随机实验 $\rightarrow$ rng_a = np.random.default_rng(2026): default_rng(2026) 返回 Generator；生成器保存后续抽样状态，不是每次调用都重新播种。

从集合 {1,...,6} 抽取八次骰子结果 $\rightarrow$ a = rng_a.integers(1, 7, size=8): integers 的上界 7 不包含在结果中，size=8 指定一维样本形状。

继续同一条伪随机序列 $\rightarrow$ next_a = rng_a.integers(1, 7, size=8): 再次调用 rng_a 会消耗后续状态，结果不必等于首次样本。

写出六个等可能基本结果各自的概率 1/6 $\rightarrow$ dice_pmf = np.full(dice_values.size, 1 / 6): 该向量是理论质量函数；经验频数与它的差异属于抽样波动。

易错提示 相同种子只有在生成器类型、抽样方法及调用顺序一致时才保证复现。default_rng 与旧 np.random.seed/RandomState 使用的随机数机制不同，不能把它们的种子相同误当作逐项输出相同。

正文代码 e01_rng.py

教材位置：第 1 章，第 1.7 节；图 1-2。

```
# 单位、变量与形状: a、b、next_a 为 (8,) 的点数数组; 点数与概率均无量纲。
# 学习目标: 认识 NumPy 的随机数生成器、随机种子和离散均匀抽样, 理解怎样复现实验。
# 核心关系: 公平骰子满足  $P(X=k)=1/6$ ,  $k=1, \dots, 6$ ; 一次抽样得到的是随机样本,
# 不能因为本次某个点数出现较多, 就认定其理论概率较大。
# 数据与输出: 构造两个具有相同初始状态的生成器, 各取 8 个骰子点数, 再展示后续抽样。
# 运行方式: 安装 NumPy、Matplotlib 后执行 python e01_rng.py; 输出终端结果和教材图
# e01_rng.png。

# 使用 np 这个惯用别名, 便于调用数组运算和随机数工具。
import numpy as np

# default_rng 返回独立的生成器对象; 2026 是种子, 不是样本量或概率参数。
# 两个对象从相同种子出发, 并执行相同的调用时, 会产生相同的伪随机序列。
# 这便于排查代码差异; 跨 NumPy 版本或更换算法时, 不应无条件承诺序列仍相同。
rng_a = np.random.default_rng(2026)
rng_b = np.random.default_rng(2026)
# integers 的下界 1 包含在内, 上界 7 不包含在内, 因此可能值恰好是 1 到 6。
# size=8 生成形状为 (8,) 的一维整数数组, 而不是形状为 (8, 1) 的列向量。
a = rng_a.integers(1, 7, size=8)
# rng_b 也取自己的前 8 个结果, 因此 b 应与 a 逐元素相同。
b = rng_b.integers(1, 7, size=8)
# 生成器会记录并推进内部状态; 这里继续使用 rng_a, 不会自动从种子重新开始。
# 后续样本可能与前一批有重复点数, 这是随机抽样的正常现象。
next_a = rng_a.integers(1, 7, size=8)
# 三行输出依次对比首次抽样、相同种子的首次抽样、同一生成器的下一次抽样。
print("首次抽样: ", a)
print("相同种子的首次抽样: ", b)
print("同一生成器的下一次抽样: ", next_a)
# array_equal 同时检查形状及所有对应元素, 验证本例的可复现性。
assert np.array_equal(a, b)
# 数组比较得到布尔数组; & 逐元素合并两个条件, all 要求所有点数都在合法区间。
# 两边条件必须加括号, 避免 Python 的运算优先级改变原意。
assert np.all((1 <= a) & (a <= 6))

# 教材图 1-2: 绘制公平骰子的完整理论分布, 不把上面 8 次抽样的频率当作理论概率。
# pyplot 仅负责作图; 这里不调用随机数生成器, 因此不会改变原随机序列和终端输出。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# dice_values 是随机变量 X 的完整支持集; 理论 概率质量函数 为每个点数对应的 1/6。
dice_values = np.arange(1, 7)
dice_pmf = np.full(dice_values.size, 1 / 6)
# 在 0 与 7 之间画完整累积分布函数; where='post' 使每个整数右侧保持当前累计值, 体现右连续性。
cdf_x = np.arange(0, 8)
cdf_y = np.r_[0.0, np.cumsum(dice_pmf), 1.0]
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 左图的纵轴是单点概率; 六根柱高之和为 1。
axes[0].bar(dice_values, dice_pmf, width=0.55)
axes[0].set(title='公平骰子的概率质量函数', xlabel='骰子点数 k', ylabel='P(X = k)',
            xticks=dice_values, ylim=(0, 0.21))
# 右图实心点表示 F(k), 空心点表示从左侧到达 k 时尚未计入该点质量的极限。
axes[1].step(cdf_x, cdf_y, where='post')
axes[1].scatter(dice_values, np.cumsum(dice_pmf), s=18, zorder=3)
axes[1].scatter(dice_values, np.r_[0.0, np.cumsum(dice_pmf)[: -1]],
                s=18, facecolors='white', edgecolors='C0', zorder=4)
```

```
axes[1].set(title='右连续的累积分布函数', xlabel='阈值 x', ylabel='P(X <= x)',
            xticks=dice_values, xlim=(0, 7), ylim=(-0.03, 1.05))
# 此文件是教材实际嵌入图，运行脚本后直接采用，不另行重新抽样或独立重绘。
fig.savefig('e01_rng.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

首次抽样: [6 2 1 4 3 3 1 3]

相同种子的首次抽样: [6 2 1 4 3 3 1 3]

同一生成器的下一次抽样: [4 3 5 5 5 6 5 2]

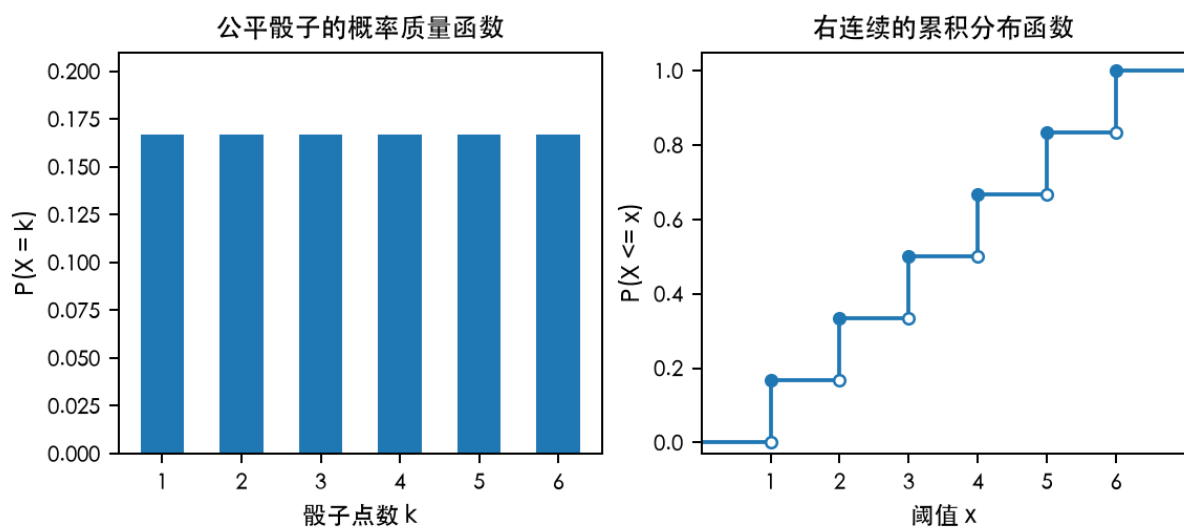


图 1-2 公平骰子的理论概率质量函数与右连续累积分布函数，由 `e01_rng.py` 生成。X 取 1 至 6 且各点概率为 1/6；累积分布函数实心点表示 $F(k)$ ，空心点表示左极限。图与该脚本打印的 8 次随机样本共用骰子模型，但不是用 8 次频率估计的分布。

读图指南 左图柱高为单点概率，六柱之和为 1；右图每次跳跃累积同一个点的概率，实心点表示已计入当前位置。阶梯水平处的长度不表示新增概率。

左图给出每个点数的概率质量；右图回答“点数不超过某个阈值”的概率。后者在整数处向上跳，跳跃高度恰为 1/6。图形读法将在下一章严格定义。现在可以先看出：阈值 3.2 与 3.9 都只包含 1、2、3，因此累积概率相同。理论概率表需要全部六个可能值；一次样本只需要记录实际出现的八个值，二者用途不同。

1.8 本章要点 常见误区与延伸阅读

核心要点：概率空间先规定结果、事件与赋权规则。有限等可能模型才可直接用计数比； σ 代数对补集和可数并封闭，使复合与极限事件仍合法。无限模型中，事件概率可以按区间长度定义，单点概率为 0 不等于该点不存在。

核心要点：条件概率先缩小参考范围再归一化；独立性是联合概率分解的性质，全概率按情景占比加权。贝叶斯更新对每个类别计算先验乘似然，再用共同证据概率归一化。记录好分母，通常比记住“正向”“反向”更可靠。

常见误区：把 $P(A|B)$ 当成 $P(B|A)$ ，把互斥当成独立，把多类似然强制加到 1，把某个零概率点排除出样本空间，或把 σ 代数说成对任意不可数并封闭。相对频数属于有限记录，不能用 8 次骰子的频率重新定义公平骰子的理论概率。

延伸阅读：哈佛《概率论导论》第二版第 1—2 章，重点对照计数口径、条件概率与全概率；官方免费教材入口为 <https://probabilitybook.net/> [P19]。哈佛统计 110 官方视频第 4—5 讲分别讨论独立、贝叶斯及全概率，目录与视频链接见 <https://stat110.hsites.harvard.edu/youtube> [P14]。

进阶阅读：麻省理工学院 18.05 第 3 课“条件概率、独立性与贝叶斯定理”，用于重画三渠道更新树[P03]。拓展阅读为 18.175 第 1 讲概率空间与 σ 代数、第 3 讲随机变量与分布，重点只取可数封闭与原像定义，真实官方链接见资料 P15、P16；这两讲为研究生材料，不作为继续本书的门槛。

1.9 练习与参考解答

练习 1-1（概念与手算，入门） 公平骰子中，A 为偶数、B 为大于 3。求 $P(A \cup B)$ 、 $P(A|B)$ ，判断二者是否独立；再求独立掷 3 次至少出现一次 6 的概率。

练习 1-2（概念与手算，进阶） 骰子只报告奇偶，取事件族 $\mathcal{F} = \{\emptyset, \Omega, \text{奇数集}, \text{偶数集}\}$ 。说明它为何是 σ 代数，并判断点数函数 $X(\omega) = \omega$ 及偶数指示函数是否可测。

练习 1-3（概念与手算，拓展） U 在 $[0,1]$ 均匀。求 $P(U=1/2)$ 、 $P(1/4 < U \leq 3/4)$ 、 $P(U \text{ 为有理数})$ ，并解释为什么不能用不可数个零概率单点相加算 $P([0,1])$ 。

练习 1-4（概念与手算，进阶） 三个互斥且穷尽类别的先验为 $(0.5, 0.3, 0.2)$ ，证据在各类别下的似然为 $(0.02, 0.04, 0.10)$ 。求证据总概率和三个后验，并指出哪个量需要归一化。

练习 1-5（编程实践，入门） 运行 `e01_rng.py`，记录两个同种子发生器的首次结果与同一发生器的第二次结果。用数组比较说明可复现性，不能只看屏幕上是否相似。

练习 1-6（编程实践，入门） 枚举两枚可区分骰子的 36 个有序对，用布尔条件统计点数和 7、第一枚 6 及二者交集；再用计数计算条件概率。

练习 1-7（编程实践，进阶） 把 `e06_bayes.py` 的四个先验同时放进数组，分别用概率公式和优势公式更新；检查两种结果一致与后验范围。

练习 1-8（编程实践，拓展） 为一个 4 点样本空间写集合程序，检验事件族 $\{\emptyset, \Omega, \{1,2\}, \{3,4\}\}$ 与 $\{\emptyset, \Omega, \{1\}\}$ 的补集及两两并封闭性。说明有限程序为何能检验这里的可数并条件。

练习 1-9（场景应用，入门） 教学设定：甲线占 70%、次品率 2%，乙线占 30%、次品率 5%。从混合产品随机抽到一件次品，求来自乙线的概率，并说明分母所代表的总体。

练习 1-10（场景应用，进阶） 教学设定：先验次品率 0.001，检测灵敏度 0.99、假阳性率 0.05。同一产品连续两次阳性，能否仅凭这三个数算后验？在给定真实类别后两次检测独立的附加假设下计算。

练习 1-11（场景应用，入门） 门店用“在营业时间随机抽一个时刻”调查拥挤程度。方案甲从 8 小时中按长度均匀取时刻；方案乙从当日实际顾客中随机抽一人并记录其到店时刻。两方案会得到同一时间分布吗？

练习 1-12（场景应用，进阶） 系统只保留成功交易日志。研究者在日志中看到渠道丙占 50%，便把它作为所有访问的渠道先验。说明错误，并给出可操作的修正信息。

参考解答与验收依据

1-1 $A \cup B = \{2, 4, 5, 6\}$ ，概率 $2/3$ ； $A \cap B = \{4, 6\}$ ，所以 $P(A|B) = (2/6)/(3/6) = 2/3$ ，不等于 $P(A) = 1/2$ ，故不独立。至少一次 6 取补事件，得 $1 - (5/6)^3 = 91/216 \approx 0.421296$ 。两个概率均正的互斥事件也不可能独立，因为交集概率 0 而边缘乘积为正。

1-2 补集把 $\emptyset$ 与 Ω 、奇数集与偶数集互换；任意可数并只能得到这 4 个集合之一。 X 的阈值原像 $\{X \leq 1\} = \{1\}$ 不属于 $\mathcal{F}$ ，故相对于这个 $\mathcal{F}$ 不可测。偶数指示函数仅取 0、1，其阈值原像是 $\emptyset$ 、奇数集或 Ω ，故可测。换成全部子集事件族后 X 也可测。

1-3 三个答案依次为 0、1/2、0。有理数可数，每个单点概率 0，可数加性给其并集概率 0。 $[0, 1]$ 不可数，概率公理没有规定用普通级数逐个相加不可数多项；总概率为 1 与单点概率 0 不矛盾。可数加性也不等于对任意无限并都存在逐项求和公式。

1-4 未归一化权重为 (0.010, 0.012, 0.020)，和为 0.042，正是证据总概率。逐项除以 0.042 得 (5/21, 2/7, 10/21)，约 (0.238095, 0.285714, 0.476190)。先验与似然的乘积需共同归一化；三个似然不必和为 1。

1-5 规定环境与调用顺序下，两个首次数组均为 [6, 2, 1, 4, 3, 3, 1, 3]，第二次为 [4, 3, 5, 5, 5, 6, 5, 2]。 $\text{np.array_equal}(a, b)$ 应为 True；所有元素满足 $1 \leq \text{值} \leq 6$ ，数组形状均为 (8,)。第二次从已有状态继续，不能以它与第一次不同作为跨算法的普遍保证。

1-6 可用双重循环构造 36 行的 (i, j) ，sum 条件得到 6 个、6 个、1 个。概率为 1/6、1/6、1/36； $P(\text{和 } 7 | \text{第一枚 } 6) = 1/6$ 。该特定事件对独立，因为 $1/36 = (1/6)(1/6)$ 。验收要同时确认分母 36、结果有序、联合事件用“且”。

1-7 灵敏度 0.99、假阳性率 0.05，未归一化真阳性为 $0.99 \times \text{prior}$ ，假阳性为 $0.05 \times (1 - \text{prior})$ 。后验约为 [0.01943463, 0.16666667, 0.51030928, 0.6875]；优势乘以 19.8 再转回概率应与它逐项 allclose。后验均在 [0, 1]，不能把 4 个情景的后验再强制加到 1。

1-8 第一族通过，第二族因 $\{1\}$ 的补集 $\{2, 3, 4\}$ 不在族中而失败。循环对每个集合取 $\Omega - A$ ，再遍历每对 $A|B$ ，检查是否仍在族中。有限事件族只有有限种不同集合，一个可数并等于其中

有限个不同集合之并，所以此处有限并封闭足以覆盖可数并；这个理由不能直接推广到无限事件族。

1-9 总体次品概率 $0.7 \times 0.02 + 0.3 \times 0.05 = 0.029$ ；乙线且次品概率 0.015，后验 $15/29 \approx 0.517241$ 。分母是混合总体中全部次品的概率，不是全部乙线产品的概率。需要随机抽自混合总体；若抽样故意过采乙线，须改用实际抽样先验。

1-10 不补充两次检测的联合机制就不能唯一计算。附加条件独立后，真次品两阳性权重 $0.001 \times 0.99^2 = 0.0009801$ ，合格两阳性权重 $0.999 \times 0.05^2 = 0.0024975$ ；后验为 $0.0009801 / 0.0034776 \approx 0.28183230$ 。若共用污染源导致相关，平方似然不成立。

1-11 一般不会。甲对等长时间段赋相同概率，1 小时占 1/8；乙按顾客量加权，客流高峰被抽到的概率更高。若前 4 小时有 80 名顾客、后 4 小时 20 名，甲两段各 1/2，乙为 0.8 与 0.2。只有客流在时间上恰好均匀等附加条件下才可能一致。

1-12 这是 $P(\text{丙}|\text{交易成功})$ ，不一定是 $P(\text{丙})$ 。应取得全部访问的渠道计数，或已知各渠道交易成功率按 $P(\text{渠道且成功}) = P(\text{成功}|\text{渠道})P(\text{渠道})$ 逆向校正并归一化。例如两渠道成功日志各 50%、成功率分别 0.5 与 0.1 时，先验正比 0.5/0.5 与 0.5/0.1，得 1/6 与 5/6。无成功率或全量日志时先验不能唯一恢复。

第 2 章 随机变量 期望与极限定理

2.1 本章导读 入门 进阶与拓展

本章围绕“数值记录怎样产生分布，平均为何稳定”展开。入门阅读先掌握分布函数、期望、方差和程序中的数组维度。进阶阅读学习联合与条件分布、全期望、全方差及协方差，并把大数定律与中心极限定理放在一起比较。拓展阅读关注收敛方式、条件期望作为随机变量、重尾与相关数据的边界；不要求第一次阅读就完成所有证明。

2.2 随机变量与分布函数

一次试验的结果可能是骰子点数、产品类别或一整条到达记录。为了计算平均、距离或总量，我们把结果转换成数。随机变量 X 就是定义在样本空间上的实值函数：对每个 ω ，规则 $X(\omega)$ 给出一个实数。函数规则在试验前已确定，随机性来自输入的基本结果尚未确定。大写 X 表示这个随机量，小写 x 通常表示一个可能值或已观测数值。

第 1 章已说明可测性：对每个阈值 x ，原事件 $\{\omega: X(\omega) \leq x\}$ 属于 $\mathcal{F}$ 。以下在骰子等有限模型中，默认 $\mathcal{F}$ 包含所有子集，因此任意给每个结果指定数值的函数都可测。若信息族较粗，则仍须检查原像，不能仅因样本空间有限就自动成立。用同一枚骰子结果可以定义 X 为点数、 $Y=X^2$ 为平方奖金、 I 为“是否偶数”的 0—1 编码；它们共享一次试验，却有不同取值和分布。

$$F_X(x) = P(X \leq x), \quad P(a < X \leq b) = F_X(b) - F_X(a) \quad (a < b) \quad (2.1)$$

F_X 是 X 的累积分布函数；下标 X 标明对应的随机变量，括号内 x 是可变阈值。 a 、 b 是实数边界。右式用“截至 b ”的概率减去“截至 a ”的概率，留下左开右闭区间。离散变量的边界点可能有正概率，因而不能随意把 $<$ 与 $\leq$ 互换。

累积分布函数单调不减、右连续，并在 x 趋于负无穷时趋于 0，在 x 趋于正无穷时趋于 1。离散变量的概率集中在有限或可数支持集 S 上，其概率质量函数为 $p_X(k)=P(X=k)$ 。每项非负且所有项之和为 1，累积分布函数等于不超过阈值的质量之和。对连续取值且具有密度的变量，累积分布函数由密度积分得到；密度不是点概率，这将在第 4 章展开。

如果分布在点 a 有跳跃，跳跃大小就是 $P(X=a)$ 。记 $F_x(a-)$ 为从 a 左侧逼近的累积分布函数极限，则 $P(X=a)=F_x(a)-F_x(a-)$ 。例如骰子 $P(X=4)=F_x(4)-F_x(4-)=4/6-3/6=1/6$ 。所有实值随机变量都有累积分布函数，所以它是贯穿离散与连续分布的一种统一语言。

2.3 期望 方差 指示变量与变换

期望是按概率加权的中心。它描述整个分布，不一定是某次试验能取得的值。公平骰子期望 3.5 并不意味着骰子能出现半点，也不意味着下一次会“接近 3.5”。若 X 的绝对值期望有限，则离散期望按取值乘概率求和；具有密度时按取值乘密度积分。可积条件排除正负无穷贡献互相抵消的未定义表达式。[P01、P04]

$$\mathbb{E}[X] = \sum_{x \in S} x p_X(x), \quad \mathbb{E}[g(X)] = \sum_{x \in S} g(x) p_X(x) \quad (2.2)$$

$\mathbb{E}$ 是期望算子； S 是离散支持集； g 是给定的确定函数。第二式直接在 X 的分布上计算 $g(X)$ 的期望，条件是相应绝对值加权和有限。若多个 x 映到同一个 $g(x)$ ，求新变量的概率质量函数时要合并质量，但求期望时可直接按原 x 逐项计算。

期望具有线性性：只要相关量可积，对确定常数 a 、 b ，有 $\mathbb{E}[aX+bY]=a\mathbb{E}[X]+b\mathbb{E}[Y]$ ，无须 X 与 Y 独立。相反， $\mathbb{E}[XY]=\mathbb{E}[X]\mathbb{E}[Y]$ 需要额外条件，独立且可积是常用的充分条件。将非线性函数移到期望外通常错误，特别是 $\mathbb{E}[X^2]$ 一般不等于 $(\mathbb{E}[X])^2$ 。

事件 A 的指示变量 $\mathbf{1}_A$ 在 A 发生时取 1，否则取 0。由两点加权可知 $\mathbb{E}[\mathbf{1}_A]=P(A)$ 。把次数写成指示量之和，是概率计算中非常有力的方法：统计 n 件产品中的次品数时，每件的指示量相加；即使各件不独立，总次数的期望仍等于各次品概率之和。独立性会影响方差，却不影响这条期望关系。

$$\mu = \mathbb{E}[X], \quad \text{Var}(X) = \mathbb{E}[(X - \mu)^2] = \mathbb{E}[X^2] - \mu^2, \quad \sigma = \sqrt{\text{Var}(X)} \quad (2.3)$$

μ 是总体均值， Var 表示方差， σ 是标准差。方差要求二阶矩有限，它对偏离中心的平方距离取平均；标准差取正平方根，与 X 使用相同单位。等式由展开 $(X-\mu)^2$ 并取期望得到。乘常数 a 会使方差乘 a^2 ，加常数 b 不改变方差。

例题 手算 2-1: 公平骰子的平方奖金

骰子点数 X 各取 1 至 6, 概率均为 $1/6$ 。设奖金 $Y=X^2$, 则 $E[Y]=(1+4+9+16+25+36)/6=91/6$ 。又 $E[Y^2]=(1+16+81+256+625+1296)/6=2275/6$, 所以 $\text{Var}(Y)=2275/6-(91/6)^2=5369/36$ 。

若只判断偶数, $I=\mathbf{1}_{\{X \text{ 为偶数}\}}$, 则 $E[I]=(0+1+0+1+0+1)/6=1/2$ 。 X 、 X^2 和 I 都是同一骰子结果的函数, 但各自的期望、方差和单位并不相同。

程序 `e02_expectation.py` x 为六个可能点数, `prob` 为六个 $1/6$, $y=x**2$ 。`sum(y*prob)`与 `sum((y-mean_y)**2*prob)`是在完整分布上精确加权; 这里不使用 $n-1$ 分母。随后用种子 11 抽 50000 次骰子并平方, 对照理论均值 $91/6$ 和方差 $5369/36$ 。`sample.var(ddof=1)`估计未知总体方差; 一次模拟接近理论值属于抽样现象, 不是两者恒等。

编程思路 把骰子点数平方看作随机变量的函数, 先在六点概率空间做加权求和, 再用五万次模拟核对。偶数事件则写成布尔指示量, 以同一套求和或均值操作计算事件概率。

代码对应

逐点计算 $Y=X^2 \rightarrow y = x ** 2$: NumPy 的 `**2` 对六个点数逐元素平方, 保留概率与取值的一一对应。

$E(Y)=\sum y p(y)=91/6 \rightarrow \text{mean_y} = \text{np.sum}(y * \text{prob})$: 数组乘法形成每一项的概率权重, `sum` 完成有限求和。

$\text{Var}(Y)=\sum [y-E(Y)]^2 p(y)=5369/36 \rightarrow \text{var_y} = \text{np.sum}((y - \text{mean_y}) ** 2 * \text{prob})$: 这里对理论分布求方差, 分母不是样本量减一。

将事件 X 为偶数变成 0/1 指示量 $\rightarrow \text{even} = (x \% 2 == 0).astype(float)$: 取模比较得到布尔数组, 乘概率并求和就是该事件的概率。

易错提示 先平方再求均值与先求均值再平方不同。模拟样本用 `ddof=1` 估计总体方差; 理论六点分布直接按概率加权, 不应再使用样本校正。

正文代码 `e02_expectation.py`

```
# 教材位置: 第 2 章, 第 2.3 节; 仅终端输出。
```

```
# 单位、变量与形状: x、prob、y 为 (6,); 期望按点数平方计, 方差按点数四次方计。
# 学习目标: 由离散分布精确计算变换后的期望和方差, 并用模拟结果作对照。
# 核心关系: 若 X 为公平骰子点数、 $Y=X^2$ , 则  $E[Y]=\sum x^2 P(X=x)$ ,
#  $Var(Y)=\sum (x^2-E[Y])^2 P(X=x)$ ; 事件 A 的指示变量满足  $E[1_A]=P(A)$ 。
# 数据与输出: 使用 6 个可能点数及其概率, 输出 Y 的理论矩和 50000 次模拟的样本矩,
# 再计算点数为偶数的概率; 理论计算不依赖模拟结果。
# 运行方式: 安装 NumPy 后执行 python e02_expectation.py, 所有结果输出到终端。

import numpy as np

# arange 的终点 7 不包含在内; x 是形状 (6,) 的浮点数组 [1,...,6]。
# 浮点类型便于随后进行加权计算, 数组下标与概率数组的位置一一对应。
x = np.arange(1, 7, dtype=float)
# full 创建 6 个相同的概率质量; 这些数之和为 1, 每个元素表示一个点数的概率。
prob = np.full(6, 1 / 6)
# ** 2 是逐元素平方, 得到 Y 的可能值 [1,4,9,16,25,36]。
# 这里先变换可能值再求期望; 一般来说  $E[X^2]$  并不等于  $(E[X])^2$ 。
y = x ** 2
# y * prob 是逐元素乘法; 求和实现有限离散分布的期望公式。
mean_y = np.sum(y * prob)
# 先以理论均值为中心, 再按概率加权平方偏差, 计算的是 Y 的总体方差。
# 由于枚举了完整分布, 这里不使用样本方差的 n-1 分母。
var_y = np.sum((y - mean_y) ** 2 * prob)
# x % 2 == 0 标出偶数点数; astype(float) 将 True/False 转成 1.0/0.0。
# 因此它表示事件“X 为偶数”的指示函数, 而不是事件“Y 等于 2”。
even = (x % 2 == 0).astype(float)
# 固定种子使这次模拟可复现; 伪随机数生成器的状态会随着抽样推进。
rng = np.random.default_rng(11)
# 每个骰子点数先抽取、再平方; sample 的形状为 (50000,), 对应 Y 的独立模拟观测。
sample = rng.integers(1, 7, size=50000) ** 2
# 理论值由完整概率分布得到; 模拟值会随随机样本波动, 通常只接近而不完全等于理论值。
print("理论期望 E[Y]、方差 Var(Y): ", mean_y, var_y)
# mean 用样本量作分母; var(ddof=1) 用样本量减 1 作分母, 得到总体方差的无偏估计。
# 无偏指重复抽样时估计量的期望等于真值, 不表示本次估计没有误差。
print("模拟样本的均值、方差: ", sample.mean(), sample.var(ddof=1))
# 对指示函数求加权平均, 得到偶数事件的理论概率 3/6。
print("偶数点数的概率: ", np.sum(even * prob))
# 浮点运算可能有舍入误差, 因此用 isclose 检查理论恒等式, 而不直接比较浮点数相等。
assert np.isclose(mean_y, 91 / 6)
assert np.isclose(var_y, 5369 / 36)
assert np.isclose(np.sum(even * prob), 0.5)
```

规定环境下的实际运行输出

理论期望 E[Y]、方差 Var(Y): 15.166666666666666 149.13888888888889

模拟样本的均值、方差: 15.21262 149.703246800536

偶数点数的概率: 0.5

2.4 联合 边缘与条件分布

只知道两个变量各自的分布, 通常不能知道它们共同变化的方式。假设 X 和 Y 分别表示同一产品的两个质检指标, 边缘分布描述“单看一个指标”, 联合分布描述“两个指标同时取何

值”。例如 X 、 Y 都各以 $1/2$ 概率取 0 或 1 ，既可以令 $Y=X$ ，也可以令 $Y=1-X$ ，还可以使二者独立；三个模型的边缘分布完全相同，联合结构却不同。[P05]

离散联合质量函数 $p_{X,Y}(x,y)=P(X=x,Y=y)$ 可以排成一张表，所有格子非负且总和为 1 。对固定 x 沿所有 y 求和，相当于把不关心的 Y 合并掉，得到 X 的边缘质量。已知 $Y=y$ 时，只取这一列，再除以列总概率，就得到 X 的条件分布。这与事件条件概率是同一个重新归一化的思想。

$$p_X(x) = \sum_y p_{X,Y}(x,y), \quad p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)} \quad (p_Y(y) > 0) \quad (2.4)$$

$p_{X,Y}$ 为联合概率质量函数， p_X 、 p_Y 为边缘概率质量函数， $p_{X|Y}$ 为条件概率质量函数。求和指标 y 遍历 Y 的全部可能值。连续变量具有联合密度时把边缘求和改为积分，并在 $f_Y(y)>0$ 处以联合密度除以边缘密度定义条件密度；这不是用 $P(Y=y)=0$ 作分母。

例题 手算 2-2：一张四格表如何产生三种分布

设 $P(X=0,Y=0)=0.4$ ， $P(X=1,Y=0)=0.1$ ， $P(X=0,Y=1)=0.2$ ， $P(X=1,Y=1)=0.3$ 。四项和为 1 。边缘有 $P(X=1)=0.1+0.3=0.4$ ， $P(Y=1)=0.2+0.3=0.5$ 。

已知 $Y=1$ 时， $X=1$ 的条件概率为 $0.3/0.5=0.6$ ；已知 $Y=0$ 时为 $0.1/0.5=0.2$ 。由于 $0.3 \neq 0.4 \times 0.5$ ， X 与 Y 不独立。边缘概率 0.4 是两种条件概率按 Y 的权重平均： $(0.2+0.6)/2=0.4$ 。

随机变量独立的统一定义是：对任意实数 a 、 b ，有 $P(X \leq a, Y \leq b) = F_X(a)F_Y(b)$ 。离散情形等价于每一格联合质量都等于对应边缘质量乘积；有联合密度时，等价于密度几乎处处可分解为 $f_X(x)f_Y(y)$ 。只核对某一格相等不足以证明独立。独立也意味着对 X 、 Y 分别作确定变换后仍然独立，但把同一个 X 变成 X 和 X^2 通常不会得到独立变量。

2.5 条件期望 全期望与全方差

条件期望就是在给定信息后的分布中取平均。离散情形中， $E[X|Y=y] = \sum_x x p_{X|Y}(x|y)$ ，是一个随 y 变化的数值函数 $m(y)$ 。在观察 Y 之前， $m(Y)$ 本身也是随机变量，写作 $E[X|Y]$ ；观察到

具体 $Y=y$ 之后才得到 $m(y)$ 。区分这两层含义，才能理解为什么对条件期望还可以再取一次期望。

全期望公式说：先在各组内求平均，再按组权重平均，等于直接对全部结果求平均。离散情况下把联合概率先按 x 求和再按 y 求和，就得到这一公式；它不是“平均的平均”都相等，而是必须保留正确的组权重。全方差则把总体波动分成组内波动与组间中心差异，特别适合解释混合总体的离散程度。哈佛大学统计 110 课程把这两条规律作为条件期望的重要用途。

[P01、P06]

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]], \quad \text{Var}(X) = \mathbb{E}[\text{Var}(X|Y)] + \text{Var}(\mathbb{E}[X|Y]) \quad (2.5)$$

外层期望按 Y 的边缘分布加权。 $\text{Var}(X|Y)$ 是各条件分布的方差所组成的随机变量；第一项为平均组内方差，第二项为各组均值的方差。全期望要求 X 可积，全方差通常在 $\mathbb{E}[X^2] < \infty$ 时使用。

下面把全方差公式逐步证明。假设 $\mathbb{E}[X^2] < \infty$ ，记 $\mu = \mathbb{E}[X]$ ， $m(Y) = \mathbb{E}[X|Y]$ 。 $m(Y)$ 是按当前组别给出的平均预测； $X - m(Y)$ 是组内残差， $m(Y) - \mu$ 是该组中心相对于总体中心的偏差。二者相加恰好恢复 $X - \mu$ ，这一步只是加上再减去同一个 $m(Y)$ 。

$$X - \mu = [X - m(Y)] + [m(Y) - \mu] \quad (2.6)$$

左侧是总体中心化偏差；右侧依次为组内偏差与组间中心偏差。这里 Y 可为离散组别，也可更为一般的观测信息。

$$(X - \mu)^2 = [X - m(Y)]^2 + [m(Y) - \mu]^2 + 2[X - m(Y)][m(Y) - \mu] \quad (2.7)$$

逐项平方必须保留交叉项。不能仅因把差分成两段，就直接把方差写成两项之和。

交叉项为什么消失？先固定 Y 。此时 $m(Y) - \mu$ 是已知数，可从条件平均中提出；而 X 的条件平均正是 $m(Y)$ ，所以组内残差的条件平均为 0。残差可以在某个观测上很大，却在同一条条件分布中正负抵消。

$$\mathbb{E}[X - m(Y)|Y] = \mathbb{E}[X|Y] - m(Y) = 0 \quad (2.8)$$

条件于 Y 时， $m(Y)$ 已知。等式没有要求残差为常数 0，也没有要求残差与 Y 独立。

$$\mathbb{E}\{[X - m(Y)][m(Y) - \mu]|Y\} = [m(Y) - \mu] \cdot 0 = 0 \quad (2.9)$$

先在每个条件分布内平均，交叉乘积的平均已为 0；随后对 Y 再平均，仍为 0。这使用的是全期望与“提出已知量”的性质。

再看两个平方项。按条件方差的定义，给定 Y 时第一项的条件平均为 $\text{Var}(X|Y)$ ，故其无条件平均为 $\mathbb{E}[\text{Var}(X|Y)]$ 。全期望又给 $\mathbb{E}[m(Y)] = \mu$ ，所以第二项的平均恰是 $\text{Var}(m(Y))$ 。把这两项及零交叉项代回平方展开，便得到全方差。

$$\mathbb{E}[(X - m(Y))^2] = \mathbb{E}[\text{Var}(X|Y)], \quad \mathbb{E}[(m(Y) - \mu)^2] = \text{Var}(m(Y)) \quad (2.10)$$

第一项平均组内散布，第二项衡量组中心的散布；二者均非负。因此知道组别并按组均值预测后，平均平方误差不会超过完全不知道组别时用 μ 预测的误差。

$$\text{Var}(X) = \mathbb{E}[\text{Var}(X|Y)] + \text{Var}(\mathbb{E}[X|Y]) \quad (2.11)$$

这就是逐步展开后得到的全方差公式：总体方差等于平均组内方差加组均值方差。两个来源都要按 Y 的分布计算，不能无权平均。

这不是在说每组方差都比总体小，个别组可以更分散；式 (2.11) 涉及按组概率加权的平均。也不能把交叉项为 0 误说成两部分独立。全方差说明总体波动的来源，正适合判断“单个模型的额外随机性”与“不同组别混在一起”的区别。

例题 手算 2-3：两种工作模式的混合

机器以各 1/2 概率处于低速、快速模式。给定模式后，每小时故障数分别服从泊松分布(2)与泊松分布(6)，条件均值和方差分别为 2 与 6。全期望给总体均值 $(2+6)/2=4$ 。

平均组内方差为 4；组均值 2 与 6 的方差为 $[(2-4)^2 + (6-4)^2]/2=4$ ；总方差为 8。总体计数均值为 4、方差为 8，不能再把它当作单一泊松分布(4)，尽管每个组内部都是泊松分布。

2.6 协方差 相关与平均的波动

方差描述一个变量的波动，协方差描述两个中心化变量共同变化的方向。若 X 偏大时 Y 也往往偏大，中心化乘积常为正，协方差可能为正；一大一小时常为负，协方差可能为负。由于它含有两种量纲的乘积，数值大小会受单位影响。把协方差除以两者标准差，得到无量纲的相关系数，便于比较。[P07]

$$\text{Cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] = \mathbb{E}[XY] - \mu_X\mu_Y, \quad \rho = \frac{\text{Cov}(X, Y)}{\sigma_X\sigma_Y} \quad (2.12)$$

μ_X 、 μ_Y 为各自均值， σ_X 、 σ_Y 为各自标准差。协方差在二阶矩有限时存在；相关系数 ρ 还要求两个标准差均大于 0，且 $-1 \leq \rho \leq 1$ 。常数变量标准差为 0，它与其他变量的相关系数通常不定义。

展开和的平方可以得到 $\text{Var}(X+Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$ 。独立且二阶矩有限会使协方差为 0，因此方差可加；反向推理通常不成立。一个清楚的反例是 X 等概率取 -1、0、1，令 $Y = X^2$ 。由对称性 $\mathbb{E}[X] = \mathbb{E}[X^3] = 0$ ，所以 $\text{Cov}(X, Y) = 0$ ；但知道 $Y=0$ 就知道 $X=0$ ，二者显然不独立。相关系数刻画线性关联，不能覆盖所有形式的依赖，更不能单独建立因果关系。

对独立同分布的 $X_1, \dots, X_n$ ，设共同均值为 μ 、方差为 σ^2 ，样本均值 $\bar{X}_n = (X_1 + \dots + X_n)/n$ 满足 $\mathbb{E}[\bar{X}_n] = \mu$ 、 $\text{Var}(\bar{X}_n) = \sigma^2/n$ 。其标准差 $\sigma/\sqrt{n}$ 称为标准误。标准差描述单个观测的散布，标准误描述重复抽样得到的均值的散布。两者都带原变量单位，却对应不同随机对象。

$$\text{Var}(\bar{X}_n) = \frac{1}{n^2} \left(\sum_{i=1}^n \text{Var}(X_i) + 2 \sum_{i < j} \text{Cov}(X_i, X_j) \right) \quad (2.13)$$

$\bar{X}_n$ 为样本均值； $i < j$ 表示每对不同观测只计一次。一般公式保留所有协方差。若观测独立同分布，协方差项消失， n 个 σ^2 除以 n^2 即得 σ^2/n 。正相关观测可能使均值比独立假设下更不稳定。

2.7 大数定律 样本平均为何稳定

大数定律说明平均在大量重复中趋于稳定。对独立同分布且 $\mathbb{E}[|X_i|] < \infty$ 的序列，弱大数定律断言：对任意固定 $\varepsilon > 0$ ， $P(|\bar{X}_n - \mu| \geq \varepsilon)$ 随 n 增加趋于 0。 ε 是事先给定的容许误差。这是概率收

敛，不是要求每一步误差都缩小，也不是承诺某个有限 n 绝无大误差。骰子先连续出现大点数，不会迫使下一次“补偿”出小点数。[P08]

$$P(|\bar{X}_n - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{n\varepsilon^2} \rightarrow 0 \quad (n \rightarrow \infty) \quad (2.14)$$

这是共同方差 σ^2 有限时由切比雪夫不等式得到的上界。箭头表示 n 增大时右侧趋于 0。证明先用独立性求 $\text{Var}(\bar{X}_n) = \sigma^2/n$ ，再把 $\bar{X}_n$ 代入 $P(|W - E[W]| \geq \varepsilon) \leq \text{Var}(W)/\varepsilon^2$ 。有限期望版本更一般，但不能仅用这个有限方差证明。

程序 e03_lln.py 程序用种子 7 产生 20000 个公平骰子点数，`cumsum(rolls)/n` 给出同一条长实验的累计均值。`n` 数组从 1 到 20000，`means` 的第 9、99、999、19999 号位置对应前 10、100、1000、20000 个观测。理论均值为 3.5，单个骰子方差 35/12， $n=20000$ 时标准误为 $\sqrt{(35/12)/20000} \approx 0.01208$ 。图中的局部反复是正常波动，不违背大数定律。

编程思路 一次生成两万次骰子，再用累计和同时得到所有样本量下的平均值。横轴是已使用的样本数；用若干检查点和理论标准误说明平均值趋近 3.5 仍会有起伏。

代码对应

列出样本量 $n=1, \dots, 20000 \rightarrow n = \text{np.arange}(1, 20001)$ ：从 1 开始避免累计均值除零，各位置与累计和长度一致。

生成独立同分布的公平骰子样本 $\rightarrow \text{rolls} = \text{rng.integers}(1, 7, \text{size}=n.\text{size})$ ：
`integers(1,7,...)` 的整数支持正是六个骰子点数。

计算每个前缀的平均数 $(X_1 + \dots + X_n)/n \rightarrow \text{means} = \text{np.cumsum}(\text{rolls}) / n$ ：`cumsum` 一次得到全部前缀和，再逐元素除以相应 n 。

易错提示 大数定律描述样本量增大时的收敛，不保证曲线单调靠近均值。固定种子只固定一条实现路径，不能把这条曲线当作所有实验都经过的轨迹。

正文代码 e03_lln.py

```
# 教材位置：第 2 章，第 2.7 节；图 2-1。
# 单位、变量与形状：rolls、means、n 均为 (20000,)；均值和标准误按骰子点数计。
```

```
# 学习目标：观察大数定律下的累计样本均值，并区分标准误与一次实验的实际误差。
# 核心关系：独立同分布且期望有限的样本满足样本均值趋向总体均值；公平骰子的
#  $E[X]=3.5$ 、 $\text{Var}(X)=35/12$ ，所以  $n$  次独立抽样均值的标准误为  $\sqrt{(35/12)/n}$ 。
# 数据与输出：模拟 20000 次骰子点数，打印若干累计均值和末尾的理论标准误，
# 并保存累计均值图 e03_11n.png；图展示的是一次实验的路径。
# 运行方式：安装 NumPy、Matplotlib 后执行 python e03_11n.py；图片写入当前工作目录。

import numpy as np
# pyplot 用于创建和保存图形；本例直接写文件，不依赖弹出绘图窗口。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 固定种子，让同一运行环境下的整条累计均值曲线可以复现。
rng = np.random.default_rng(7)
# n=[1,...,20000]，形状为 (20000,); n[i] 对应使用前 i+1 个观测的均值。
n = np.arange(1, 20001)
# 上界 7 不包含在内，生成与 n 等长的公平骰子样本。
rolls = rng.integers(1, 7, size=n.size)
# cumsum 保存前缀和；逐元素除以 n，同一次实验的前 1、2、...、20000 次样本被逐步复用。
# 因此这些累计均值彼此有关联，并不是每个 n 都重新做一轮独立实验。
means = np.cumsum(rolls) / n
# NumPy 从 0 开始编号，所以第 10 个累计均值的位置为 9，其他下标同理。
print("n=10、100、1000、20000 时的累计均值：", means[[9, 99, 999, 19999]])
# n[-1] 是最后一个样本量；这里计算的是均值在重复抽样下的标准差。
# 标准误不是本次 |样本均值-3.5|，也不是每条路径必然遵守的误差上界。
print("n=20000 时的理论标准误：", np.sqrt((35 / 12) / n[-1]))
# 枚举所有等概率点数，检查理论均值；该断言不要求随机样本均值恰好等于 3.5。
assert np.isclose(np.mean(np.arange(1, 7)), 3.5)
# 创建一幅 6×3 英寸的图；fig 管理整幅图，ax 管理坐标轴和曲线。
fig, ax = plt.subplots(figsize=(6.3, 3))
# 横轴使用对数刻度，让早期波动和后期稳定趋势都容易观察；样本量本身没有被改写。
ax.semilogx(n, means, label="累计样本均值")
# 用水平虚线标出理论均值，方便比较本次样本路径与极限目标。
ax.axhline(3.5, color="C1", linestyle="--", label="E[X]=3.5")
ax.set(xlabel="样本量 n (对数刻度)", ylabel="均值")
# 大数定律不保证误差随 n 单调下降；曲线短暂偏离目标并不与定律矛盾。
# 图例区分样本路径和理论值，tight_layout 调整边距以减少标签被裁切的可能。
ax.legend()
fig.tight_layout()
# dpi 控制保存图片的像素密度；关闭图形释放资源，不会删除已经保存的文件。
fig.savefig("e03_11n.png", dpi=160)
plt.close(fig)
```

规定环境下的实际运行输出

n=10、100、1000、20000 时的累计均值： [4.1 3.69 3.575 3.5011]

n=20000 时的理论标准误： 0.012076147288491199

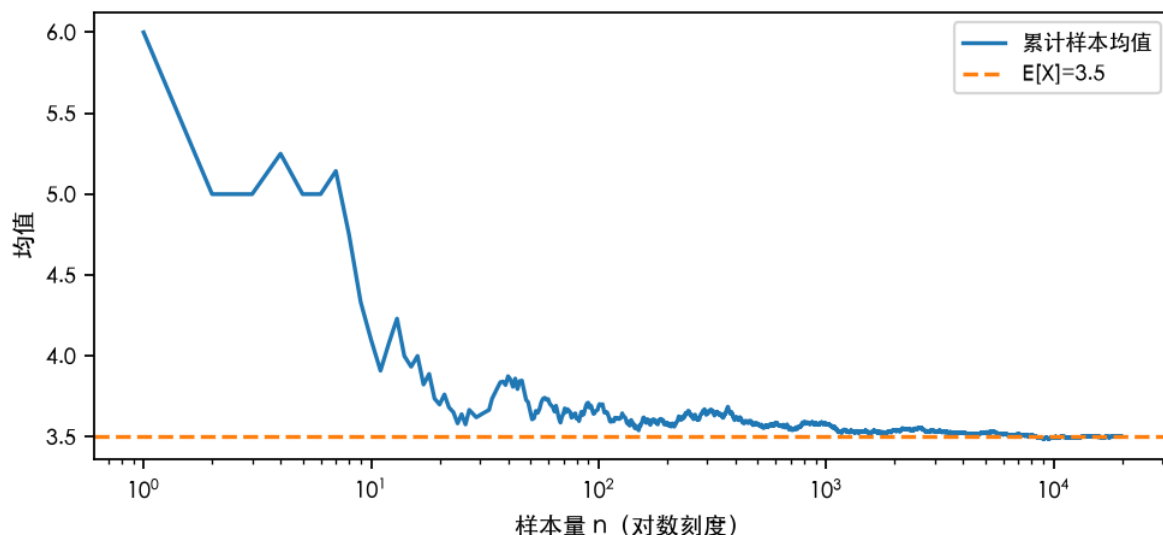


图 2-1 大数定律的一条累计均值路径，直接采用 `e03_lln.py` 运行图。种子=7 生成 20000 个公平骰子点数，横轴 n 为对数刻度，虚线为理论均值 3.5；`means` 是同一次长实验逐步累积得到的平均。

读图指南 横轴每向右一段代表更多观测，采用对数刻度便于看早期变化。蓝线是一条累计路径，不是许多独立实验的均值分布；局部远离虚线不违反大数定律。

2.8 中心极限定理 标准化平均的分布

大数定律告诉我们样本均值靠近 μ ，中心极限定理进一步描述其偏差在适当尺度下的形状。设 X_i 独立同分布，均值 μ 有限、方差 $0 < \sigma^2 < \infty$ 。把均值减去 μ ，再除以其标准误 $\sigma/\sqrt{n}$ ，所得标准化量的分布随着 n 增大趋于标准正态。这里趋近正态的是重复抽样的均值分布；原始观测的分布不会因此变成正态。[P08]

$$Z_n = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \rightarrow N(0,1) \quad \text{依分布收敛} \quad (2.15)$$

Z_n 为标准化样本均值，分母是均值标准误。依分布收敛表示对每个实数 z ， $P(Z_n \leq z)$ 趋于标准正态累积概率 $\Phi(z)$ ；不是同一条样本路径的 Z_n 趋于 0。 $\sigma^2=0$ 时变量为常数，不使用此标准化。

从两定理的尺度可以看出它们如何配合。 $\bar{X}_n - \mu$ 的典型尺度随 $1/\sqrt{n}$ 缩小，所以平均趋于稳定；乘上 $\sqrt{n}$ 之后，这种缩小被抵消，留下非退化的近似正态形状。若总体本来正态，独立样

本均值对每个 n 都精确正态，不需要等 n 很大。若总体严重偏斜或尾部很重，达到满意近似需要更多观测，不能把“ $n \geq 30$ ”当作普遍保证。

有限方差条件很关键。一些重尾分布没有有限方差，经典中心极限定理的条件不满足；也有相关数据需要不同的标准误与极限定理。实际使用正态近似时，除了看样本量，还要检查分布偏斜、异常值、数据依赖和估计目标。尾部很小的概率通常比中心区域更难近似，即使直方图中间已经接近钟形也应谨慎。

下面借一个右偏的连续模型观察式 (2.15)：指数分布的系统定义在第 4 章给出，此处只需接受它非负、尺度为 2 时均值与标准差均为 2 这几个设定。原始形状不对称，恰好能显示平均后的形状怎样变化。图的柱高按密度归一化，柱形面积之和为 1；它不是某个精确均值的点概率。

程序 e04_clt.py 本例总体为指数分布，尺度 $\theta=2$ ，故 $\mu=2$ 、 $\sigma=2$ 。种子 0 下，对 $n=1$ 、5、50 分别进行 4000 轮独立实验。values 形状为 (4000, n)，每行是一个样本，axis=1 取平均后得到 4000 个均值。standardized=(means-2)/(2/ $\sqrt{n}$) 使用已知总体标准误。图叠加标准正态密度； n 增加后标准化均值更接近该曲线，原始指数总体仍右偏。

编程思路 对偏斜的指数总体，分别重复模拟 $n=1$ 、5、50 的样本均值。每行是一组实验，沿列取均值，再按总体标准差 $\theta/\sqrt{n}$ 标准化；比较直方图与标准正态密度。

代码对应

生成 4000 组、每组 n 个指数等待时间 $\rightarrow$ values = rng.exponential(theta, size=(repetitions, n))：形状 (repetitions, n) 把重复实验放在行，单次实验的观测放在列。

逐组计算样本平均数 $\rightarrow$ means = values.mean(axis=1)：axis=1 消去每一行内部的 n 个观测，返回 4000 个均值。

计算 $Z=(\bar{X}-\theta)/(\theta/\sqrt{n}) \rightarrow$ standardized = (means - theta) / (theta / np.sqrt(n))：指数分布均值和标准差都为 θ ，标准化使用总体尺度而非每组估计尺度。

易错提示 中心极限定理用于独立同分布且方差有限的样本均值。增大 n 与增大 repetitions 的作用不同：前者改善近似，后者使模拟直方图更稳定。

正文代码 e04_clt.py

```
# 教材位置：第 2 章，第 2.8 节；图 2-2。
# 单位、变量与形状：values 为 (4000,n)，means 为 (4000,)；原等待时间用统一时间单位，标准
# 化量无量纲。
# 学习目标：用偏斜的指数分布演示中心极限定理，理解均值的标准化和标准误。
# 核心关系：独立同分布、方差有限且非零时， $\sqrt{n}(\text{样本均值}-\mu)/\sigma$  的分布随  $n$  增大
# 趋向标准正态分布；本例指数分布的尺度  $\theta=2$ ，均值和标准差均为  $\theta$ 。
# 数据与输出：对  $n=1, 5, 50$  各重复 4000 次抽样，打印理论标准误和均值的样本标准差，
# 保存三幅标准化均值直方图与标准正态密度的对照图 e04_clt.png。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e04_clt.py；图片写入当前工作目录
# 。

import numpy as np
# stats 提供理论分布的概率密度函数，用于绘制参照曲线。
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 生成器在后面的循环中持续推进；不会在每个样本量下重新设置种子。
rng = np.random.default_rng(0)
# theta 是尺度参数，不是速率参数；对应的指数分布速率为  $1/\theta$ 。
# repetitions 是重复实验次数，决定直方图的模拟精细程度； $n$  决定每次均值使用的样本量。
theta, repetitions = 2.0, 4000
# 生成一行三列子图，并统一坐标尺度，以便直接比较分布形状的变化。
fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.8), sharex=True, sharey=True)
# 标准正态参照曲线的横坐标网格；300 是画线点数，与模拟样本量无关。
z = np.linspace(-4, 4, 300)
# zip 将三个坐标轴分别与三个  $n$  配对；大  $n$  下的标准化均值通常更接近正态。
for ax, n in zip(axes, [1, 5, 50]):
    # values 形状为 (4000, n)：每行是一轮实验，每列是一轮中的一个独立观测。
    # NumPy 的 exponential 第一个参数为尺度，因此这里  $E[X]=\theta$ 、 $\text{Var}(X)=\theta^2$ 。
    values = rng.exponential(theta, size=(repetitions, n))
    # 沿 axis=1 对每行取平均，将 (4000, n) 压缩成 (4000,) 的均值数组。
    means = values.mean(axis=1)
    # 减去总体均值，再除以理论标准误  $\theta/\sqrt{n}$ ；不是除以原观测的标准差  $\theta$ 。
    # 本例使用已知总体参数，所以对应中心极限定理中的标准化量，而不是  $t$  统计量。
    standardized = (means - theta) / (theta / np.sqrt(n))
    # density=True 按“频数/(总数×组宽)”归一化，使所有柱形总面积为 1。
    # 柱高是密度而非点概率，区间概率要看该区间覆盖的面积；bins 控制分组数量。
    ax.hist(standardized, bins=40, density=True, alpha=0.6)
    # norm.pdf 给出标准正态密度，不是累积概率；连续随机变量在单点的概率为 0。
    ax.plot(z, stats.norm.pdf(z), color="C1")
    # xlim 只限制显示范围，不会丢弃上述模拟数据，也不会重新计算标准化结果。
    ax.set(title=f" $n={n}$ ", xlabel="标准化样本均值", xlim=(-4, 4))
    # 对 4000 个均值求样本标准差，估计“均值的抽样标准差”，应接近 理论标准误。
    # ddof=1 使用 repetitions-1 作分母；有限次模拟下二者不必完全相等。
    print(n, "理论标准误", theta / np.sqrt(n), "均值的样本标准差", means.std(ddof=1))
    # 检查每轮恰好产生一个均值，防止将求平均的轴写错而改变实验含义。
    assert means.shape == (repetitions,)
```

```
# 中心极限定理 描述的是均值经标准化后的分布；它不声称原始指数观测变成正态。
# 增加重复次数会使直方图更平滑，却不能代替增大 n 所带来的正态近似改善。
axes[0].set_ylabel("概率密度")
fig.tight_layout()
# 在当前工作目录保存图片后关闭图形，便于脚本在无交互环境下连续运行。
fig.savefig("e04_clt.png", dpi=160)
plt.close(fig)
```

规定环境下的实际运行输出

```
1 理论标准误 2.0 均值的样本标准差 2.06649152659734
5 理论标准误 0.8944271909999159 均值的样本标准差 0.8741213255436786
50 理论标准误 0.282842712474619 均值的样本标准差 0.2907973265116702
```

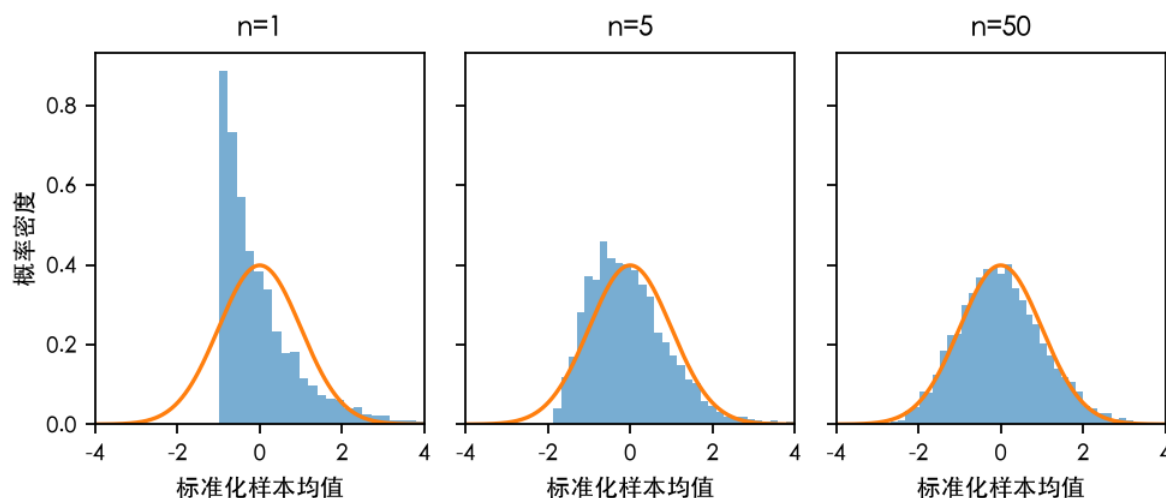


图 2-2 指数总体的标准化均值与中心极限定理，直接采用 `e04_clt.py` 运行图。指数尺度 $\theta=2$ 、种子=0；对 $n=1$ 、5、50 各重复 4000 次，三个面板使用共同横纵尺度，并叠加标准正态密度；柱高为密度。

读图指南 三个面板保持相同标准化横轴，先比较偏斜与两尾，再看柱形和标准正态曲线是否相容。每个面板 4000 个均值，单轮样本量分别为 1、5、50；不要把 4000 当成每个均值使用的观测数。

4000 是用来观察抽样分布的模拟轮数， n 是每轮样本量，两者作用不同。增加 n 改变均值分布与近似质量；增加轮数主要让我们把这个既定分布看得更清楚。直方图使用 `density=True`，使柱形总面积为 1；柱高是密度，不是某个精确均值取值的概率。这与第 4 章连续密度的读法保持一致。

把结论写在同一张纸上比较很有帮助：大数定律讨论未经放大的平均误差，问“超过固定容许误差的机会是否趋于 0”；中心极限定理先把误差按标准误放大，问“放大后的分布趋向什

么形状”。前者只需独立同分布与有限绝对一阶矩这一常用充分条件；本书使用的经典中心极限定理还要求正且有限的方差。有限方差模型中两者同时可用，但并不是一个定理的两种名称。

两者都没有承诺沿一条路径逐次单调变准。一次累计均值从真值左边跨到右边，或一段时间误差反而增加，都不违反大数定律；标准化误差 Z_n 也不会因为 n 增加就收缩到 0。中心极限定理提供近似分布，由它估计尾概率时还要判断近似质量。扩大模拟轮数主要让直方图更平滑，并不能替代扩大每轮样本量或修复有偏抽样。

例题 手算：把稳定性与波动形状分别使用

教学假设：观测独立同分布，均值 100、标准差 12，样本量 36。样本均值的均值仍为 100，标准误为 $12/6=2$ 。切比雪夫上界给 $P(|\bar{X}-100|\geq 6)\leq 1/9$ ，这是一条不依赖总体正态的有限方差保证。

若该样本量下正态近似足够好，中心极限定理给近似概率 $P(|\bar{X}-100|\geq 6)\approx 2[1-\Phi(3)]\approx 0.00270$ 。后一数值通常更精细，却是近似，不能把它当作切比雪夫那样对所有有限方差总体都成立的严格上界。若总体本来正态，此处正态计算才为精确值。

2.9 蒙特卡洛估计 从概率平均到数值精度

蒙特卡洛方法把待求量写成一个随机变量的期望，再用样本平均估计。例如在单位正方形内均匀抽点，令 I 表示点是否落在四分之一单位圆内，则 $P(I=1)=\pi/4$ ，故 $4I$ 的期望为 π 。重复独立抽 n 点， π 估计为 4 乘落圆频率。伯努利分布指示量的方差为 $q(1-q)$ ，因此估计标准误约为 $4\sqrt{(\hat{q}(1-\hat{q}))/n}$ 。要把标准误缩小一半，样本量通常需要增为四倍，这说明精度的代价。

程序 `e05_monte_carlo.py` 种子 3 生成形状(100000,2)的均匀点数组，每行是一点的两个坐标。`sum(points**2,axis=1)` 逐行求 x^2+y^2 ， ≤ 1 得到长度 100000 的布尔指示量 `inside`。`inside.mean()` 是 $\hat{q}$ ， $\pi_{\text{hat}}=4\hat{q}$ ；`se_hat` 是估计标准误。程序给出基于正态近似的 95% 区间及与已知 π 的实际绝对误差。一次实际误差和标准误不是同一概念：前者需知道真值，后者估计重复抽样的波动。

编程思路 在单位正方形均匀撒点，用是否落入四分之一单位圆的指示量估计面积，再乘四得到 π 。把几何计数转成伯努利平均值后，可直接使用二项抽样的标准误公式。

代码对应

在 $[0,1]^2$ 内产生十万个二维点 $\rightarrow$ `points = rng.random((n, 2))`：二维数组每行一个点，第二维的两个数是横纵坐标。

检验 $x^2+y^2 \leq 1 \rightarrow$ `inside = np.sum(points ** 2, axis=1) <= 1`：`axis=1` 对每个点的两个平方坐标求和，布尔结果就是圆内事件的指示量。

由圆内比例 $\hat{q}$ 计算 $\hat{\pi}=4\hat{q} \rightarrow$ `pi_hat = 4 * q`：平方区域面积为 1，四分之一圆面积为 $\pi/4$ 。

估计 π 的标准误 $4\sqrt{[\hat{q}(1-\hat{q})/n]} \rightarrow$ `se_hat = 4 * np.sqrt(q * (1 - q) / n)`：放大面积估计时，标准差同样乘以四。

易错提示 近似区间使用独立均匀抽样及大样本正态近似。模拟误差通常按 n 的平方根缩小，增加十倍样本不会自动增加十倍精度。

正文代码 `e05_monte_carlo.py`

```
# 教材位置：第 2 章，第 2.9 节；仅终端输出。
# 单位、变量与形状：points 为 (100000,2)，inside 为 (100000,)；坐标、圆周率估计和标准误
# 均无量纲。
# 学习目标：通过几何概率估计  $\pi$ ，并理解蒙特卡洛估计的标准误和近似置信区间。
# 核心关系：在单位正方形均匀抽点，落入第一象限单位圆内的概率  $q_0=\pi/4$ ；
# 若  $I$  是是否落入圆内的指示变量，则  $\pi$  估计= $4 \times$  样本均值，理论标准误为  $4\sqrt{q_0(1-q_0)/n}$ 。
# 数据与输出：抽取 100000 个二维点，输出  $\pi$  的估计、估计标准误、近似 95% 区间和实际绝对误差。
# 运行方式：安装 NumPy 后执行 python e05_monte_carlo.py；不保存外部数据或图片。

import numpy as np

# 固定种子用于复现实验；n 是独立随机点的个数，而不是每个坐标轴的网格点数。
```



```

rng = np.random.default_rng(3)
n = 100000
# random((n, 2)) 生成形状为 (100000, 2) 的数组，每行是一个 (x, y)。
# 两个坐标各自均匀分布于 [0,1)，因此这些点均匀覆盖单位正方形。
points = rng.random((n, 2))
# 沿 axis=1 对每行的 x2、y2 求和，得到到原点距离的平方，无需平方根。
# <=1 判断点是否在四分之一单位圆内，inside 是形状 (n,) 的布尔数组。
# 连续均匀模型下边界概率为 0，是否包含圆弧不改变理论面积比。
inside = np.sum(points ** 2, axis=1) <= 1
# mean 将 True 当 1、False 当 0，得到落入圆内的样本比例；q 是对 q0 的估计。
q = inside.mean()
# 四分之一圆的面积乘以 4 得到整圆面积，因此将样本比例乘以 4 来估计  $\pi$ 。
pi_hat = 4 * q
# 由独立伯努利样本的方差得到比例标准误，再利用常数倍的标准差缩放规则乘以 4。
# 用 q 代替未知 q0 得到代入估计；它描述重复抽样时估计量的波动，不等于本次实际误差。
se_hat = 4 * np.sqrt(q * (1 - q) / n)
print("圆周率估计值: ", pi_hat)
print("估计标准误: ", se_hat)
# 大样本下利用正态近似构造 Wald 区间，1.96 是标准正态约 97.5% 分位数。
# [-1,1] 生成左右两个端点；约 95% 指重复实验中这种区间的覆盖率，而非必然包含真值。
# 当命中比例非常接近 0 或 1、或样本过少时，这种近似可能不好；本例命中比例适中且 n 大。
print("近似 95% 置信区间: ", pi_hat + np.array([-1, 1]) * 1.96 * se_hat)
#  $\pi$  在本例中已知，所以能直接计算实现后的绝对误差；一般蒙特卡洛问题未必知道真值。
print("绝对误差: ", abs(pi_hat - np.pi))
# 确保每个二维点对应一个判断结果，防止求和方向错误。
assert inside.shape == (n,)
# 将 q0= $\pi$ /4 代入理论标准误并化简，检查两个等价公式；这里未要求模拟估计恰好等于  $\pi$ 。
assert np.isclose(4 * np.sqrt((np.pi / 4) * (1 - np.pi / 4) / n),
                  np.sqrt(np.pi * (4 - np.pi) / n))

```

规定环境下的实际运行输出

圆周率估计值: 3.13512

估计标准误: 0.0052072090274925585

近似 95% 置信区间: [3.12491387 3.14532613]

绝对误差: 0.006472653589792987

读蒙特卡洛区间时，先辨明它围绕哪一个量：本例目标为固定常数 π ，随机性来自计算机产生的独立点。每轮样本量决定标准误；若把同一批点反复复制，并没有获得更多独立信息。

第 5 章会区分置信覆盖、当前后验概率与未来观测的预测概率，这些名称不能只凭区间形式相似就互换。

2.10 本章要点 常见误区与延伸阅读

核心要点：随机变量是可测数值函数，分布函数把离散跳跃和连续累积统一起来。期望按概率加权，线性性不需要独立；方差是中心化平方的平均，单位为原单位的平方。对函数求期望时可在原分布上直接加权。

核心要点：联合分布决定共同变化；条件分布用条件概率重新归一化。全期望平均条件中心，全方差分解平均组内波动与组中心波动；交叉项为 0 来自条件残差均值 0，不能据此声称两部分独立。协方差为 0 也一般不推出独立。

核心要点：大数定律回答平均是否靠近真均值，中心极限定理回答按标准误放大后的分布形状。独立同分布、矩条件与近似质量都要逐项核对。蒙特卡洛轮数增加能降低计算波动，不能消除建模偏差或共享系统误差。

常见误区：把标准差与标准误当成同一对象，把 $E[g(X)]$ 一律写成 $g(E[X])$ ，把零相关理解成独立，把大数定律说成误差逐次下降，把 CLT 说成原数据变正态，或把“一轮样本量”与“重复轮数”混用。求和符号是 $\sum$ ；若后文用 Σ 表示协方差矩阵，它是另一个符号，不是求和运算。

延伸阅读：哈佛《概率论导论》第 3—4 章复习随机变量与期望，第 7 章读联合分布，第 9 章读条件期望，第 10 章读极限定理；作者免费入口 <https://probabilitybook.net/> [P19]。哈佛统计 110 第 27 讲对照全方差交叉项，第 29 讲对照大数定律与 CLT，官方目录 <https://stat110.hsites.harvard.edu/youtube> [P14]。

课程配读：麻省理工学院 18.05 第 4 课乙、第 6 课乙、第 7 课甲乙，分别用于加权期望、极限定理、联合分布与协方差；具体公开阅读链接见 P04、P08、P05、P07。复习时先写出所平均的对象与轴，再对照程序中的 sum、mean 和 axis。

2.11 练习与参考解答

练习 2-1（概念与手算，入门） X 以 1/4、1/2、1/4 概率取 0、1、2。求 $E[X]$ 、 $\text{Var}(X)$ 、 $E[3X+2]$ ，并判断 $E[X^2]$ 是否等于 $(E[X])^2$ 。

练习 2-2（概念与手算，进阶） 联合表 $P(0,0)=0.4$ 、 $P(1,0)=0.1$ 、 $P(0,1)=0.2$ 、 $P(1,1)=0.3$ 。求边缘、 $E[X|Y=0]$ 、 $E[X|Y=1]$ 与 $\text{Cov}(X,Y)$ 。

练习 2-3（概念与手算，进阶） 两组各占 1/2，条件均值 1、5，条件方差均为 2。求总体方差；用残差 $R=X-E[X|Y]$ 解释全方差交叉项为何为 0。

练习 2-4（概念与手算，进阶） 独立同分布观测的标准差 12，样本量 36 与 144 时均值标准误是多少？分别用一句准确的话陈述大数定律与经典中心极限定理，说明它们是否保证误差逐次缩小。

练习 2-5（编程实践，入门） 运行 `e02_expectation.py`，以完整六点分布计算平方奖金均值、方差，再与 50000 次模拟比较。指出哪一处使用 $n-1$ 分母。

练习 2-6（编程实践，入门） 运行 `e03_lln.py`，解释 `means[[9,99,999,19999]]` 对应的样本量，核对累计均值与理论标准误。

练习 2-7（编程实践，进阶） 用 `e04_clt.py` 的指数尺度 2，对 $n=1, 5, 50$ 各模拟 4000 轮。明确 `axis=1` 为何用于每轮均值，并给出标准化量的理论均值、方差及 $n=1$ 时落在 $[-1,1]$ 的精确概率。

练习 2-8（编程实践，进阶） 运行 `e05_monte_carlo.py`，解释 `axis=1` 与 `inside.mean()` 的含义，复算 π 估计、标准误及误差。

练习 2-9（场景应用，入门） 教学设定：某工序单件长度标准差 6 毫米，独立抽 100 件估计均值。要把均值标准误从 0.6 毫米缩到 0.3 毫米，样本量应如何调整？是否因此说明单件产品波动变小？

练习 2-10（场景应用，进阶） 教学设定：罕见事件概率 $q=10^{-5}$ ，独立模拟 10000 次且一次未发生。能否报告概率为 0？计算期望命中数和零命中概率，提出一个具体改进。

练习 2-11（场景应用，进阶） 教学设定：每个观测 $X_i=\mu+B+\varepsilon_i$ ，公共偏移 B 的方差为 4，各 ε_i 独立且方差 9，并与 B 独立。 $n=100$ 时均值方差是多少？为何不能用单个观测方差 13 除以 100？

练习 2-12（场景应用，拓展） 某资料声称“观测独立同分布且均值有限，所以标准化均值必趋于标准正态”。指出缺少的经典条件；若均值存在但方差无限，应如何改写结论。

参考解答与验收依据

2-1 $E[X]=1$, $E[X^2]=3/2$, 方差为 $1/2$; $E[3X+2]=5$. $E[X^2]=1.5$ 与 $(E[X])^2=1$ 不同, 差恰为方差。期望可以线性提出常数, 平方通常不能提出。

2-2 $P(X=1)=0.4$ 、 $P(Y=1)=0.5$; 两个条件均值为 0.2 与 0.6 。 $E[XY]=0.3$, 协方差 $0.3-0.4\times 0.5=0.1$ 。全期望给 $0.5\times 0.2+0.5\times 0.6=0.4$; 不独立, 因为联合概率 0.3 不等于边缘乘积 0.2 。

2-3 总体均值 3 , 平均组内方差 2 , 组中心方差 $[(1-3)^2+(5-3)^2]/2=4$, 总方差 6 。给定 Y , 组中心偏差是常数, 而 $E[R|Y]=0$, 所以 $E[R(E[X|Y]-3)|Y]=0$; 再用全期望得无条件交叉项 0 。不能仅写“两部分独立”, 因为本结论不要求它们独立。

2-4 标准误依次为 2 与 1 。有限绝对一阶矩下, 大数定律使均值超过固定误差范围的概率趋于 0 ; 有限正方差下, 中心极限定理使除以标准误后的均值偏差依分布趋于标准正态。二者都不保证每次增加一个观测后误差单调缩小; CLT 也不把原始总体变为正态。

2-5 理论结果为 $91/6\approx 15.16666667$ 、 $5369/36\approx 149.13888889$; 完整分布加权不使用 $n-1$ 。规定种子 11 的模拟均值 15.21262 、样本方差约 149.70324680 ; 只有 `sample.var(ddof=1)` 用样本量减 1 估计总体方差。验收还包括偶数指示量加权均值 0.5 。

2-6 下标从 0 开始, 所以依次对应 10 、 100 、 1000 、 20000 个观测。规定种子 7 得到 $[4.1, 3.69, 3.575, 3.5011]$; 末端理论标准误 $\sqrt{(35/12)/20000}\approx 0.01207615$ 。`cumsum` 沿同一条样本路径累计, 各个 n 的均值彼此依赖, 不应当作 4 个独立实验。

2-7 二维 `values` 有 4000 行、 n 列; 每行是一轮观测, 沿列方向 `axis=1` 压缩后得到 4000 个均值。 $Z=(\text{均值}-2)/(2/\sqrt{n})$ 对每个 n 理论均值 0 、方差 1 。 $n=1$ 时 $Z=X/2-1$, 故 $P(|Z|\leq 1)=P(X\leq 4)=1-\exp(-2)\approx 0.864665$, 不是标准正态的 0.682689 。验收看形状、标准化公式和这种差异, 不要求每幅经验直方图都精确同于理论密度。

2-8 `points` 形状 $(100000, 2)$, `axis=1` 逐行求两个坐标平方和, 得到每个点是否入圆的判断。`inside.mean()` 把 `True` 记 1 求比例; π 估计 3.13512 , 估计标准误约 0.00520721 , 近似 95% 区间

[3.12491387,3.14532613]，实际绝对误差约 0.00647265。标准误描述重复模拟散布，不等于当前绝对误差。

2-9 样本量须增到 400，因为 $6/\sqrt{400}=0.3$ 。单件标准差仍为 6 毫米，改进的是总体均值的估计精度。若 400 次是重复抄录同样 100 件，或测量存在强相关，不能直接获得这个四倍样本量效果。

2-10 期望命中数 $M_q=0.1$ ，零命中概率 $(1-10^{-5})^{10000} \approx 0.90484$ ，零命中很常见，不能据此把真概率判为 0。若希望平均命中约 100 次，需要 $M=10^7$ ；也可在明确权重和无偏性后采用专门稀有事件方法。此处用普通正态近似描述零频数的相对误差并不可靠。

2-11 均值为 $\mu+B+\bar{\epsilon}$ ，方差 $4+9/100=4.09$ ，标准误约 2.02237。各观测共享 B，任意两个不同观测协方差 4，故不独立。把 $13/100=0.13$ 作为均值方差遗漏公共偏移；增加同一批次观测只能缩小 ϵ 部分，不能消除 B。

2-12 经典独立同分布 CLT 还需 $0<\sigma^2<\infty$ ，并用 $\sigma/\sqrt{n}$ 标准化。只有有限绝对一阶矩时可援用常用大数定律说明样本均值趋于均值，不能自动得到正态波动或通常标准误。方差无限时可能需要其他归一化和稳定极限，或改用适合问题的稳健目标；不能继续给出未经论证的正态 95% 区间。

第 3 章 离散概率分布

3.1 本章导读 入门 进阶与拓展

本章选择分布的起点是试验机制。入门阅读区分固定次数的成功计数、固定时间内的到达计数和达到成功目标所需的次数。进阶阅读推导各模型的质量函数、矩与近似关系，特别留意抽样是否放回。拓展阅读讨论过度离散、模型混合与模拟误差。公式中的参数要与实际单位一起写出，不能只根据直方图外形挑分布。

3.2 伯努利试验与二项计数

选择分布之前，先说清变量在记录什么。一次“是否合格”的判断、固定次数中的成功数、等到首次成功的试验编号、固定时段内的到达数，都取整数，却对应不同的试验机制。支持集、停止规则与独立性通常比直方图外形更能帮助我们选模型。模型里的“成功”只是给某类结果起的名字，可以表示次品、拒绝或报警，并不带好坏含义。

伯努利随机变量 I 只取 0、1， $P(I=1)=p$ 、 $P(I=0)=1-p$ ，记为 $I \sim \text{伯努利分布}(p)$ ，其中 $0 \leq p \leq 1$ 。它就是某事件的指示量，所以 $E[I]=p$ 。由于 $I^2=I$ ， $\text{Var}(I)=E[I^2]-p^2=p(1-p)$ 。该方差在 $p=1/2$ 时最大，在 p 为 0 或 1 时为 0；结果越接近确定，波动越小。

固定做 n 次相互独立、成功概率都为 p 的伯努利试验，令 X 表示成功总次数，则 $X \sim \text{二项分布}(n, p)$ ，简称二项分布。它需要三个条件同时成立：次数预先固定、试验独立、成功概率相同。若前面结果改变后面概率，或各件产品来自不同次品率的设备，就不能只因“数成功次数”便直接使用二项分布。

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \dots, n; \quad E[X] = np, \quad \text{Var}(X) = np(1-p) \quad (3.1)$$

n 是非负整数试验次数， p 为单次成功概率， k 为成功次数。 $p^k(1-p)^{n-k}$ 是一种指定成功位置排列的概率；组合数 $C(n, k)$ 计算有多少种成功位置。均值与方差可把 X 写成 n 个独立指示量之和得到，方差相加使用了独立性。

公式的推导可从 $n=3$ 、 $k=2$ 看清：成功成功失败、成功失败成功、失败成功成功是三种互斥序列，每种概率 $p^2(1-p)$ ，相加为 $3p^2(1-p)$ 。一般情形只是把 3 推广为 $C(n,k)$ 。所有 k 的概率和为 1，也可由二项式展开 $[p+(1-p)]^n=1$ 验证。边界 $p=0$ 或 1 对应确定结果，计算公式中的 0^0 按组合分布的连续延拓处理，实际直接判断结果最清楚。

例题 手算 3-1：四次试验恰好两次成功

$n=4$ ， $p=0.3$ 。恰好两次成功概率为 $C(4,2) \times 0.3^2 \times 0.7^2 = 6 \times 0.09 \times 0.49 = 0.2646$ ；至少一次成功概率为 $1 - 0.7^4 = 0.7599$ 。前一个事件需要组合数，后一个事件取补集更方便。

X 的均值为 1.2，方差为 0.84。均值不是最可能值的同义词，也不必为整数。一个计数变量只能在整数上取值，不限制其概率加权中心也必须为整数。

程序 e07_binomial.py 程序保持 $n=20$ 、 $p=0.3$ ， k 遍历 0 至 20。manual 逐项实现组合公式， $rv.pmf(k)$ 由 SciPy 计算，两者用 allclose 核对；完整质量和应为 1。理论均值 6、方差 4.2。 $pmf(6)$ 给 $P(X=6)$ ， $cdf(4)$ 给 $P(X \leq 4)$ ， $sf(7)$ 给 $P(X > 7) = P(X \geq 8)$ 。尾事件的边界要先写成数学式再调用，避免把“至少 8”错算为 $sf(8)$ 。

编程思路 固定 $n=20$ 、 $p=0.3$ ，逐个枚举成功次数，分别用组合数公式和 SciPy 二项分布计算概率并断言一致。随后把单点概率、左尾累计与严格右尾分开调用。

代码对应

$P(X=j)=C(20,j)0.3^j0.7^{20-j} \rightarrow manual = np.array([comb(n, int(j)) * p ** j * (1 - p) ** (n - j) for j in k])$ ：math.comb 计算整数选取数，再乘每种成功失败排列的概率；列表遍历所有 j 。

定义 $X \sim \text{Binomial}(20, 0.3) \rightarrow rv = stats.binom(n=n, p=p)$ ：冻结分布后， $rv.pmf(k)$ 与 manual 在同一支持上逐点对应。

列出从 0 到 20 的全部可能成功次数 $\rightarrow k = np.arange(n + 1)$ ：arange($n+1$) 的右端不包含，因此需加一。

易错提示 pmf(6) 是恰好六次, cdf(4) 是不超过四次, sf(7) 是大于七次即至少八次。comb 是手算公式的组合项, stats.binom.pmf 则把组合项与幂次整体实现, 二者并不是两种不同模型。

正文代码 e07_binomial.py

```
# 教材位置: 第 3 章, 第 3.2 节; 图 3-1。
# 单位、变量与形状: k、manual 为 (21,); k 按成功次数计, 概率质量和累积概率均无量纲。
# 学习目标: 从组合计数推导二项分布的概率质量函数, 并核对 SciPy 的分布接口和尾概率端点。
# 核心关系: n 次相互独立、成功概率相同为 p 的伯努利试验中, 成功次数 X 满足
#  $P(X=k)=C(n,k)p^k(1-p)^{(n-k)}$ ,  $E[X]=np$ ,  $Var(X)=np(1-p)$ 。
# 数据与输出: 使用 n=20、p=0.3, 枚举全部 21 个可能成功次数, 输出理论均值、方差和三个事件
# 的概率。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后执行 python e07_binomial.py; 理论计算不做随
# 机抽样, 并保存 e07_binomial.png。

# comb 返回组合数  $C(n,k)$ , 表示从 n 个试验位置中选出 k 个成功位置的方式数。
from math import comb
import numpy as np
from scipy import stats

# n 为固定试验次数, p 为每次成功概率; 只规定这两个参数还不够, 二项模型也要求独立性和同一
# p。
n, p = 20, 0.3
# arange 不包含上界, 所以用 n+1 保留 k=n; k 的形状为 (21,), 覆盖完整支持集 0,...,20。
k = np.arange(n + 1)
# 对固定的 k,  $p^k(1-p)^{(n-k)}$  是一种成功/失败排列的概率, 再乘组合数汇总所有排列。
# 列表推导式为每个 j 计算一个概率, 最后转成数组; int(j) 将 NumPy 整数显式转成 Python 整
# 数。
manual = np.array([comb(n, int(j)) * p ** j * (1 - p) ** (n - j) for j in k])
# 将参数固定为一个“冻结分布”对象, 后面调用 pmf、cdf、sf 时不必再次传入 n、p。
rv = stats.binom(n=n, p=p)
# pmf(k) 是离散点概率  $P(X=k)$ , 不是连续分布的密度; 数组输入返回形状相同的概率数组。
# 两种实现逐项核对, 有助于发现组合数、指数或参数传递错误。
assert np.allclose(manual, rv.pmf(k))
# 完整支持集上的概率之和为 1; 使用 isclose 容纳浮点计算的舍入误差。
assert np.isclose(manual.sum(), 1)
# 将成功次数乘以对应概率再求和, 验证按定义计算的期望等于 np。
assert np.isclose(np.sum(k * manual), n * p)
# 这是分布自身的均值和方差, 并非某一组观测的样本均值、样本方差。
print("理论均值、方差: ", rv.mean(), rv.var())
# cdf(4)= $P(X\leq 4)$ , 包含端点 4; sf(7)= $P(X>7)=P(X\geq 8)$ , 所以必须传 7。
# 对离散分布,  $P(X\geq k)$  要用 sf(k-1); 直接用 sf(k) 会漏掉  $X=k$  的概率。
# sf 直接计算右尾, 通常也比先计算 cdf 再从 1 中相减更适合很小的尾概率。
print("P(X=6), P(X<=4), P(X>=8):", rv.pmf(6), rv.cdf(4), rv.sf(7))

# 教材图 3-1 使用本例唯一的主参数 n=20、p=0.3; 两种分布表示来自同一个冻结对象 rv。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# manual 是上面组合公式的结果; 折点与柱高使用相同 k, 显示完整支持集 0 到 n。
axes[0].bar(k, manual, width=0.7, alpha=0.6, label='组合公式')
axes[0].plot(k, rv.pmf(k), 'o', markersize=3, label='SciPy 概率质量函数')
```



```
axes[0].set(title=f'二项分布: n={n}, p={p}', xlabel='成功次数 k',
            ylabel='P(X = k)', xlim=(-0.6, n + 0.6))
axes[0].legend(fontsize=7.8)
# 在支持集左右各加一个点，画出累积分布函数的 0 与 1 平台；阶梯按右连续方式连接。
cdf_k = np.arange(-1, n + 2)
axes[1].step(cdf_k, rv.cdf(cdf_k), where='post')
axes[1].set(title='同一模型的累积分布函数', xlabel='阈值 k',
            ylabel='P(X ≤ k)', xlim=(-1, n + 1), ylim=(-0.03, 1.05))
fig.savefig('e07_binomial.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

理论均值、方差： 6.0 4.2

$P(X=6)$, $P(X \leq 4)$, $P(X \geq 8)$: 0.1916389827534427 0.23750777887760136
0.22772820258183926

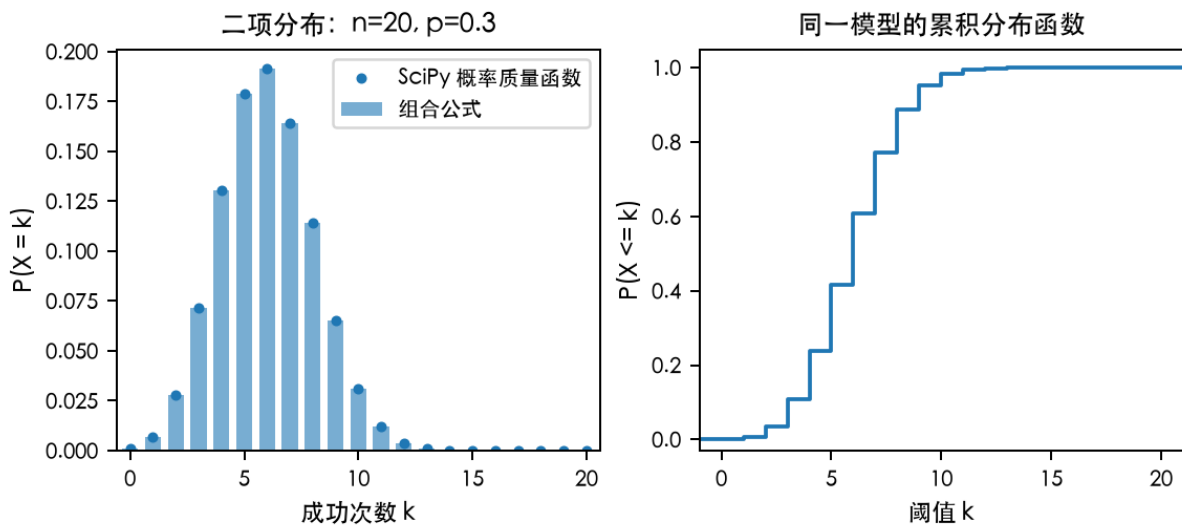


图 3-1 二项分布(20,0.3)的概率质量函数与累积分布函数，由 e07_binomial.py 生成。左图组合公式柱形与 SciPy 的点逐项对照，右图为同一模型的右连续累积分布函数；两图完整显示 0 至 20 的支持集。

读图指南 左图柱形与散点对照的是同一条二项质量公式的两种计算，差别应只在数值舍入；右图每级跳跃来自同一 k 的质量。sf(7)对应至少 8 次，不是至少 7 次。

3.3 泊松计数与稀有事件极限

泊松分布常用于记录固定时间、面积或暴露量内的事件次数，记作 $X \sim \text{泊松分布}(\lambda)$ ，参数 $\lambda > 0$ 表示该观察窗口的平均次数。它的支持集为 0、1、2、……，没有预设上限。把一个大窗口分成许多小窗口，若每小窗口出现事件的机会很小、不同小窗口近似独立，并且总平均次数保持稳定，就会得到泊松近似。真正的过程模型还需要说明速率是否稳定及事件是否存在聚集。

速率与平均计数必须分开。以下用 r 表示事件速率，单位为“次/小时”，用 $\lambda=rt$ 表示长度 t 小时窗口内的平均次数；泊松分布的参数 λ 是一个计数均值。若改用分钟，3 次/小时等于 0.05 次/分钟，同一 30 分钟窗口的参数无论怎样写都应为 1.5。把“3”不带单位地传入所有时长，会把不同问题混为一谈。第 4 章讨论整条过程时将惯用 λ 表示速率，并在当地明确单位；相同希腊字母在不同模型中的含义要重新核对。

设 $X_n \sim \text{二项分布}(n, \lambda/n)$ ，固定 k 后把二项质量展开：组合系数中的 $n(n-1)\cdots(n-k+1)$ 与 n^k 之比趋于 1； $(1-\lambda/n)^n$ 趋于 $\exp(-\lambda)$ ；余下的 $(1-\lambda/n)^{n-k}$ 趋于 1。因此每个固定 k 处的概率趋于 $\exp(-\lambda)\lambda^k/k!$ 。这是“机会很多、单次概率很小、平均数固定”的精确表达；只说 n 大或 p 小都不足以完整描述这个极限。

$$P(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}, \quad k = 0, 1, \dots; \quad \mathbb{E}[X] = \lambda, \quad \text{Var}(X) = \lambda \quad (3.2)$$

λ 是整个观察窗口的平均次数， e 是自然对数的底数， $k!$ 为阶乘。若事件速率为 r 次/小时，观察 t 小时，则参数是 $\lambda=rt$ ，而不是把每小时速率直接代入任意长度窗口。泊松均值等于方差，是后续检查模型的重要特征。

均值可以直接移位求和推导： $\sum_{k=1}^{\infty} k \exp(-\lambda) \lambda^k/k! = \lambda \sum_{j=0}^{\infty} \exp(-\lambda) \lambda^j/j! = \lambda$ 。类似地， $\mathbb{E}[X(X-1)] = \lambda^2$ 。由 $X^2 = X(X-1) + X$ 得到 $\mathbb{E}[X^2] = \lambda^2 + \lambda$ ，故方差等于 λ 。使用下降阶乘 $X(X-1)$ 能使阶乘相消，比直接处理 X^2 更简洁。

独立泊松计数之和仍为泊松，参数相加；但把不同速率组的数据混在一起并不等于把它们的计数相加。第 2 章的混合例子展示了前者可能有方差大于均值。现实数据若明显过度离散，应检查组间异质性、事件相关或暴露量差异，而非只把 λ 重新估大。均值与方差接近只是必要线索，不能独自证明泊松假设。

程序 `e08_poisson_limit.py` 固定 $\lambda=3$ ，分别取 $n=10, 50, 500$ ，并令 $p=3/n$ ；三个二项模型均值都为 3。程序在 $k=0$ 至 29 比较每一点概率质量函数与泊松分布(3)，报告这个有限网格上的最大点概率差。图只放大主要区域 0 至 12，最大差仍在 0 至 29 计算。该数值不是所有尾概率的最大误差，也不是总变差距离；理解所报误差对应的对象，是阅读数值实验的一部分。

编程思路 让二项分布保持 $np=3$ ，依次增大 n 、减小 p ，与 $\text{Poisson}(3)$ 在同一离散网格上比较。把概率曲线与逐点绝对误差并排画出，再检查绘图网格以外的泊松尾部。

代码对应

以 $e^{-3}3^k/k!$ 定义泊松极限概率 $\rightarrow \text{target} = \text{stats.poisson.pmf}(k, \text{mu}=\text{lam})$ ：这里用 SciPy 的 pmf 实现该公式， k 是共同对照的网格。

构造 $\text{Binomial}(n, 3/n) \rightarrow \text{approx} = \text{stats.binom.pmf}(k, n=n, p=\text{lam} / n)$ ：每次循环只改变 n 和相应 p ，均值 np 始终为 3。

计算每个整数点的二项—泊松绝对误差 $\rightarrow \text{delta} = \text{np.max}(\text{np.abs}(\text{approx} - \text{target}))$ ：逐元素差取绝对值，最大值只代表当前 k 网格内的最大逐点误差。

易错提示 泊松近似要求稀有独立事件、 n 大而 p 小；固定 np 并不使有限 n 下两个分布完全相同。图中有限网格的最大差不是全支持的总变差距离。

正文代码 `e08_poisson_limit.py`

```
# 教材位置：第 3 章，第 3.3 节；图 3-2。
# 单位、变量与形状：k、target、approx 为 (30,)；横轴为计数，概率质量和近似误差无量纲。
# 学习目标：观察二项分布在“试验次数增大、单次概率减小、均值固定”时的泊松极限。
# 核心关系：固定  $\lambda$ ，令  $p=\lambda/n$ ，当  $n \rightarrow \infty$  时二项分布  $(n, p)$  在每个固定整数  $k$  上的
# 概率趋向  $\text{Poisson}(\lambda)$  的概率；两者均值均为  $\lambda$ ，但二项分布方差是  $\lambda(1-\lambda/n)$ 。
# 数据与输出：取  $\lambda=3$ ，比较  $n=10, 50, 500$  时  $k=0, \dots, 29$  上的最大概率差，
# 再输出没有纳入该网格的泊松右尾概率；本例直接计算理论分布。
# 配套图件：运行后另保存 e08_poisson_limit.png，教材直接采用该文件。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e08_poisson_limit.py，结果输出到
# 终端。

import numpy as np
from scipy import stats

# lam 表示期望计数  $\lambda$ ；使用 lam 作为变量名，因为 lambda 是 Python 的保留字。
lam = 3.0
# k 是形状 (30,) 的整数数组，比较范围包括 0 和 29；泊松分布的支持集实际上没有有限上界。
k = np.arange(30)
```

```
# target[j] 对应  $P(\text{Poisson}(3)=k[j])$ ; 这里得到概率质量, 不是概率密度或累积概率。
target = stats.poisson.pmf(k, mu=lam)
# 每轮改变 n, 同时让 p 随 n 改变, 不能保持  $p=0.3$  不变后仍声称泊松参数固定为 3。
# 这些 n 均大于  $\lambda$ , 因此  $p=\lambda/n$  落在合法的概率区间内。
for n in [10, 50, 500]:
    # 在相同 k 网格计算二项分布; 当  $k>n$  时, 其概率为 0, 如  $n=10$  时的  $k=11, \dots, 29$ 。
    approx = stats.binom.pmf(k, n=n, p=lam / n)
    # 先逐项取概率差的绝对值, 再取最大值; delta 只衡量当前有限网格上的最大点概率差。
    # 它不是全部尾概率的误差, 也不是总变差距离, 更不是每个 k 上误差的总和。
    delta = np.max(np.abs(approx - target))
    # 格式符只控制对齐和保留小数位, 不改变计算所用的 p 或 delta。
    print(f"n={n:3d}, p={lam/n:.4f}, k=0..29 的最大绝对差={delta:.6f}")
    # 验证这三个二项分布的均值都精确对应  $n \times (\lambda/n) = \lambda$ , 仅允许浮点舍入误差。
    assert np.isclose(stats.binom.mean(n, lam / n), lam)
# k[-1]=29; 离散分布的  $\text{sf}(29)=P(X>29)=P(X\geq 30)$ , 恰好是网格之外的泊松概率质量。
# 输出它是为了交代截断范围; 网格内概率之和不足 1 并不代表分布没有正确归一化。
print("网格外遗漏的泊松尾概率: ", stats.poisson.sf(k[-1], mu=lam))

# 教材图 3-2 复用原诊断网格  $k=0, \dots, 29$ , 图上最大差与上面打印值使用完全相同的范围。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.8), constrained_layout=True,
                        sharex=True, sharey=True)
# 每个 n 对应  $p=\lambda/n$ ; 理论概率质量函数的重复计算不抽样, 不改变任何随机状态或终端结果。
for ax, plot_n in zip(axes, [10, 50, 500]):
    plot_p = lam / plot_n
    plot_binomial = stats.binom.pmf(k, n=plot_n, p=plot_p)
    plot_delta = np.max(np.abs(plot_binomial - target))
    ax.bar(k, plot_binomial, alpha=0.55, label='二项分布')
    ax.plot(k, target, 'o-', color='C1', markersize=2.6, linewidth=1, label='泊松分布')

    # 只把主要概率区域放大到 0..12; 最大差的计算范围仍明确写为 0..29。
    ax.set(title=f'n={plot_n}, p={plot_p:g}', xlabel='计数 k',
           xlim=(-0.5, 12.5), ylim=(0, 0.29), xticks=[0, 3, 6, 9, 12])
    ax.text(0.98, 0.95, f'最大差 (k=0..29) \n{plot_delta:.6f}',
           transform=ax.transAxes, ha='right', va='top', fontsize=7.8)
axes[0].set_ylabel('概率质量')
axes[0].legend(fontsize=7.8, loc='upper right', bbox_to_anchor=(1, 0.75))
# 文件名明确对应本例; 图注需说明显示到 12 与完整计算到 29 的区别。
fig.savefig('e08_poisson_limit.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

$n= 10$, $p=0.3000$, $k=0..29$ 的最大绝对差=0.042786

$n= 50$, $p=0.0600$, $k=0..29$ 的最大绝对差=0.007015

$n=500$, $p=0.0060$, $k=0..29$ 的最大绝对差=0.000675

网格外遗漏的泊松尾概率: 4.277047721426935e-20

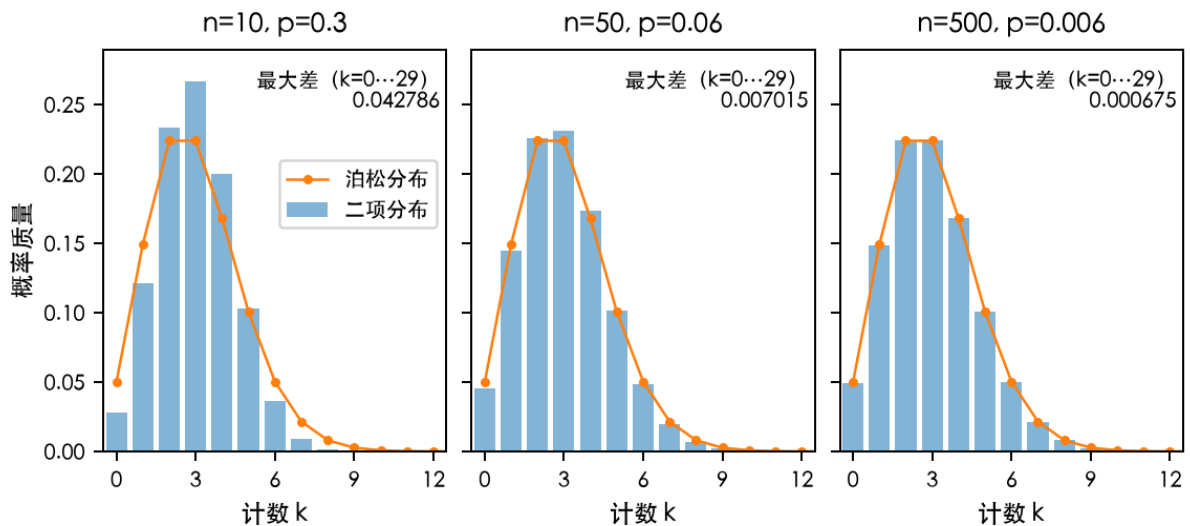


图 3-2 二项分布向泊松分布逼近，由 `e08_poisson_limit.py` 生成。固定 $\lambda=3$ ，对 $n=10、50、500$ 令 $p=\lambda/n$ ；概率质量函数与最大差均在 $k=0$ 至 29 计算，与终端输出一致。图中只放大显示 0 至 12 的主要概率区域，标注的最大差仍针对 0 至 29。

读图指南 从左到右 n 增大而 $p=3/n$ 减小，三组均值保持 3。比较橙线和蓝柱的逐点差；图上最大差在 0 至 29 计算，显示范围 0 至 12 只放大主要区域。

程序 e09_poisson.py 程序使用泊松分布(5)，理论均值与方差都为 5。upper 由 $1-10^{-12}$ 分位数选择，有限求和计算均值与 $E[X(X-1)]$ ，应接近 5 与 25；同时打印遗漏尾概率。种子 42 生成 50000 个样本并计算样本均值、样本方差。支持集无限，有限数组只能截断；“已累计质量+遗漏尾概率 ≈ 1 ”的核验比默认为数组已覆盖全部结果更严谨。

编程思路 用高分位数决定足够长的泊松求和区间，按概率权重计算一阶矩与二阶阶乘矩，再以固定种子的五万次模拟检查均值和方差。

代码对应

截取覆盖概率至少约 $1-10^{-12}$ 的整数支持 $\rightarrow$ `upper = int(rv.ppf(1 - 1e-12))`：ppf 给出上分位数，随后整数化；这一处理控制数值求和的遗漏尾部。

$E(X)=\sum kP(X=k) \rightarrow$ `mean_sum = np.sum(k * prob)`：逐元素 $k*\text{prob}$ 后求和，是离散期望定义。

$E[X(X-1)]=\lambda^2 \rightarrow \text{factorial_moment} = \text{np.sum}(k * (k - 1) * \text{prob})$: 泊松二阶阶乘矩与一阶矩合并可推出 $\text{Var}(X)=\lambda$ 。

模拟五万个 $\text{Poisson}(5)$ 观测 $\rightarrow \text{sample} = \text{rv.rvs}(\text{size}=50000, \text{random_state}=\text{rng})$: 把已建立的 Generator 传给 random_state , 使 SciPy 抽样与种子约定一致。

易错提示 有限区间求和会留下极小的尾部误差, 不能把浮点结果要求为逐位精确等于理论值。样本方差用 $\text{ddof}=1$, 理论方差直接等于 λ 。

正文代码 e09_poisson.py

```
# 教材位置: 第3章, 第3.3节; 仅终端输出。
# 单位、变量与形状: sample 为 (50000,); 观测为单位暴露量中的事件次数, 概率无量纲。
# 学习目标: 核对泊松分布的均值、方差和二阶阶乘矩, 理解无限求和的截断与模拟检验。
# 核心关系:  $X \sim \text{Poisson}(\lambda)$  时  $E[X]=\text{Var}(X)=\lambda$ ,  $E[X(X-1)]=\lambda^2$ ;
# 理论期望本应对  $k=0, 1, \dots$  的全部可能值加权求和, 代码使用高分位数进行有限截断。
# 数据与输出: 取  $\lambda=5$ , 输出精确理论矩、截断求和的矩、遗漏尾概率, 以及 50000 次模拟的样本矩。
# 运行方式: 安装 NumPy、SciPy 后执行 python e09_poisson.py; 结果输出到终端。

import numpy as np
from scipy import stats

# mu 是 SciPy 泊松分布的均值参数; 冻结分布对象保存参数, 供后续计算和抽样复用。
rv = stats.poisson(mu=5)
# ppf 是分位数函数: 寻找使  $P(X \leq \text{upper})$  至少达到给定概率的整数界限。
# 使用接近 1 的分位数是为了把遗漏概率压得很小; int 转成构造整数网格所需的 Python 整数。
upper = int(rv.ppf(1 - 1e-12))
# arange 不包含上界, 因此加 1 使网格覆盖  $0, \dots, \text{upper}$ ; 这是有限数组, 长度为  $\text{upper}+1$ 。
k = np.arange(upper + 1)
# prob 与 k 位置对应, 保存各整数的理论点概率; 截断后  $\text{prob.sum}()$  会略小于 1。
prob = rv.pmf(k)
# 这是  $\sum_{k=0}^{\text{upper}} kP(X=k)$ , 近似  $E[X]$ ; 没有对截断概率重新归一化。
mean_sum = np.sum(k * prob)
# 这里计算  $X(X-1)$  的期望, 而不是  $X^2$  的期望; 泊松分布的理论值为  $\lambda^2=25$ 。
# 因为  $X^2=X(X-1)+X$ , 也可以由这两个矩重建二阶原点矩与方差。
factorial_moment = np.sum(k * (k - 1) * prob)
# 固定随机种子, 以复现下面的样本; 理论矩与截断计算不使用随机数。
rng = np.random.default_rng(42)
# rvs 生成 (50000,) 形状的泊松整数观测; 通过 random_state 显式传入生成器。
sample = rv.rvs(size=50000, random_state=rng)
print("理论均值、方差: ", rv.mean(), rv.var())
# 有限和会遗漏尾部的加权贡献; 尾概率很小并不自动给出同样大小的矩误差上界,
# 因为均值和阶乘矩的尾项还分别乘上了 k 和  $k(k-1)$ 。
print("有限求和的均值、二阶阶乘矩  $E[X(X-1)]$ : ", mean_sum, factorial_moment)
#  $\text{sf}(\text{upper})=P(X>\text{upper})$ , 正好对应本次有限网格外的全部概率; 这里不是  $P(X \geq \text{upper})$ 。
print("遗漏的尾概率: ", rv.sf(upper))
# 模拟均值和方差会随机波动; ddof=1 用样本量减 1 为分母, 估计总体方差。
print("样本均值、样本方差: ", sample.mean(), sample.var(ddof=1))
# 首个断言核对理论均值和方差; 并未要求有限样本的均值、方差正好等于 5。
assert np.allclose([rv.mean(), rv.var()], [5, 5])
# 网格内的概率加上严格大于 upper 的尾概率应为 1, 检查截断端点没有漏算或重复计算。
```



```
assert np.isclose(prob.sum() + rv.sf(upper), 1)
```

规定环境下的实际运行输出

理论均值、方差： 5.0 5.0

有限求和的均值、二阶阶乘矩 $E[X(X-1)]$: 4.999999999971978 24.999999999237485

遗漏的尾概率: 9.933985021970706e-13

样本均值、样本方差: 5.00504 5.036235323106462

例题 教学假设：客服、服务器与故障计数

客服呼入：假设某个服务时段来电率稳定为 12 次/小时，互不重叠窗口的来电数独立。15 分钟内平均来电数为 3，因此无来电概率为 $\exp(-3) \approx 0.04979$ ，至少 2 次概率为 $1 - \exp(-3)(1+3) \approx 0.80085$ 。排班高峰与平峰若速率不同，应分段建模，不能把整天均值当作每个时段的稳定速率。

服务器请求：假设某接口在固定负载下平均每秒 2 次请求。半秒内平均 1 次，无请求概率为 $\exp(-1) \approx 0.36788$ 。如果重试会由一次故障触发多次紧邻请求，独立性便值得怀疑；只看计数均值不够。

设备故障：假设在稳定工况与充分维修后，每台设备平均每 100 小时发生 0.2 次故障。观察一台设备 500 小时，平均次数为 1，无故障概率为 $\exp(-1)$ 。若比较 10 台设备各 50 小时，还须说明设备间是否独立、工况速率是否相同；总暴露量相同只是计数参数可相加的一部分前提。

这些数字均为明确设置的教学参数，没有声称来自真实公司记录。真实应用须给出日志窗口、设备台时等暴露量，检查时间趋势、聚集性和不同对象的异质性。泊松模型便于计算，并不替代这些检验。

3.4 几何等待与负二项等待

二项分布先固定试验次数，再问成功多少次；等待分布先固定成功目标，再问需要多少次试验。设独立伯努利试验的成功概率为 p ， $0 < p \leq 1$ ，令 T 为首次成功发生的试验编号。 $T=1$

意味着第一次就成功； $T=k$ 意味着前 $k-1$ 次失败且第 k 次成功，故 $P(T=k)=(1-p)^{k-1}p$ 。这里的几何分布采用从 1 开始计总试验次数的约定。

另一些教材把首次成功前的失败次数 $G=T-1$ 称为几何变量，其支持集从 0 开始，均值也少 1。两种约定都合理，关键是先定义变量，再核对软件接口。SciPy 的 `geom` 使用 T 的从 1 开始约定；哈佛大学教材常在等待问题中强调“数失败”与“数总次数”的差别，阅读时应始终通过支持集确认。[P01]

$$P(T > m) = (1 - p)^m, \quad \mathbb{E}[T] = \frac{1}{p}, \quad \text{Var}(T) = \frac{1 - p}{p^2} \quad (3.3)$$

m 为非负整数， $T > m$ 表示前 m 次全失败。均值可由非负整数变量的尾和公式 $\mathbb{E}[T] = \sum_{m=0}^{\infty} P(T > m)$ 求得，几何级数之和为 $1/p$ 。方差可由对几何级数求导得到二阶矩，再减去均值平方。

几何分布具有无记忆性。当 $0 < p < 1$ 时，对非负整数 s, t ， $P(T > s+t | T > s) = (1-p)^{s+t} / (1-p)^s = (1-p)^t = P(T > t)$ 。已知前 s 次都未成功，后续独立试验的等待分布仍与从头开始相同。条件是“还没有成功”，不是“第 s 次刚成功”；这个性质也不适用于所有等待分布。每次成功率若随时间升高或下降，剩余等待会改变。

若目标改为第 r 次成功，令 T_r 为达到该目标的总试验数， r 是正整数。 $T_r = k$ 时，第 k 次必须成功，前 $k-1$ 次中恰有 $r-1$ 次成功，因此 $P(T_r = k) = C(k-1, r-1)p^r(1-p)^{k-r}$ ， $k=r, r+1, \dots$ 。它也可看作 r 段相互独立的几何等待相加，所以均值 r/p 、方差 $r(1-p)/p^2$ 。

SciPy 的 `nbinom(n=r, p=p)` 返回的是第 r 次成功之前的失败总数 $F_r = T_r - r$ ，支持集从 0 开始。其均值为 $r(1-p)/p$ ，方差仍为 $r(1-p)/p^2$ 。给定总次数 k 时，要调用失败数 $k-r$ 的质量；若把 k 原样传入，就会把事件错移 r 次。模型参数名 n 在这个接口中是目标成功数，与二项分布接口中的固定试验数含义不同。

例题 手算 3-2：第 3 次成功恰在第 5 次

$p=0.3$ ， $r=3$ 。第 5 次必须成功；前 4 次中选出 2 次成功有 $C(4,2)=6$ 种，概率为 $6 \times 0.3^3 \times 0.7^2 = 0.07938$ 。对应失败次数为 $5-3=2$ ，软件应计算 `nbinom(n=3, p=0.3)` 在 2 处的质量。

总次数均值为 $3/0.3=10$ ，失败次数均值为 7，二者只差确定常数 3，方差同为 $3 \times 0.7/0.3^2=70/3$ 。

程序 `e10_waiting.py` 程序固定 $p=0.3$ 、 $r=3$ ，用 $(s,t)=(3,2)$ 与 $(5,4)$ 检查 $\text{geom.sf}(s+t)/\text{geom.sf}(s)=\text{geom.sf}(t)$ 。打印总次数 3 至 8 时，调用 `failures.pmf(total-r)`。左图展示首次成功编号 1 至 20，右图展示第 3 次成功总次数 3 至 30，均值虚线为 10。这些是为显示而选的有限范围，等待分布在图外仍有尾概率。

编程思路 把第一次成功等待次数与第 r 次成功前的失败数放在同一成功概率 $p=0.3$ 下比较。先明确 SciPy 的支持约定，再把负二项失败数平移 r 得到总试验次数，并用生存概率验证几何无记忆性。

代码对应

第一次成功总试验数 T 的支持为 $1,2,\dots \rightarrow \text{geom} = \text{stats.geom}(p=p): \text{stats.geom}(p)$ 已包含成功的那一次，不能再人为加一。

第 $r=3$ 次成功前失败数 F 的支持为 $0,1,\dots \rightarrow \text{failures} = \text{stats.nbinom}(n=r, p=p): \text{stats.nbinom}$ 的变量是失败数，总试验数应写为 $F+r$ 。

$P(T>s+t|T>s)=P(T>s+t)/P(T>s) \rightarrow \text{conditional} = \text{geom.sf}(s+t) / \text{geom.sf}(s): \text{sf}$ 使用严格大于事件，与几何等待变量的整数支持一致。

易错提示 不同教材可能把几何分布定义为失败数；本例按 SciPy 的总试验数定义。负二项的 r 不是总试验次数，计算总次数概率时要把输入减去 r 。

正文代码 `e10_waiting.py`

```
# 教材位置：第 3 章，第 3.4 节；图 3-3。
# 单位、变量与形状：total 为 (6,)，图中总次数已把失败数平移 r；次数按试验次计，概率无量纲。
# 学习目标：理解离散等待时间的无记忆性，并区分负二项分布记录的失败数和总试验次数。
# 核心关系：独立伯努利试验的成功概率为 p，首次成功所需次数  $T \in \{1, 2, \dots\}$ ；
#  $P(T>s+t|T>s)=P(T>t)$ 。若等到第 r 次成功前有 F 次失败，则总次数  $N=F+r$ ，
#  $E[N]=r/p$ ， $\text{Var}(N)=\text{Var}(F)=r(1-p)/p^2$ 。
# 数据与输出：取  $p=0.3$ 、 $r=3$ ，验证两组无记忆关系，打印若干总次数的点概率及两种计数的均值。
# 配套图件：运行后另保存 e10_waiting.png，教材直接采用该文件。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e10_waiting.py；所有输出由理论分
```

```
# 布计算。

import numpy as np
from scipy import stats

# p 是每次独立试验成功的概率, r 是等待达到的成功总数; 本例中 r 为正整数。
p, r = 0.3, 3
# SciPy 的 geom 计入最后一次成功, 最小可能值是 1; 它不表示首次成功前的失败次数。
geom = stats.geom(p=p)
# SciPy 的 nbinom(n=r, p=p) 表示第 r 次成功前的失败次数, 支持集从 0 开始。
# 参数名 n 在这里代表目标成功次数 r, 与别处二项分布中“固定试验总数 n”的含义不同。
failures = stats.nbinom(n=r, p=p)
# s 是已经等待且仍未成功的次数, t 是后续比较的额外等待长度; 本例二者均取非负整数。
for s, t in [(3, 2), (5, 4)]:
    # sf(u)=P(T>u)=(1-p)^u; T>s+t 蕴含 T>s, 所以条件概率写成两个尾概率的比值。
    # 分母 P(T>s)>0; 条件是“前 s 次都未成功”, 不能换成“第 s 次已经成功”。
    conditional = geom.sf(s + t) / geom.sf(s)
    # 无记忆性说明剩余等待时间分布和从头等待相同, 来自后续伯努利试验与过去独立。
    # isclose 用于数值核对; 该关系不是通过这两组例子才成立, 而是几何分布的理论性质。
    assert np.isclose(conditional, geom.sf(t))
    print("无记忆性 (s、t、条件概率): ", s, t, conditional)
# 至少要做 r 次试验才能获得 r 次成功; 这里只展示 r, ..., r+5 这六种总次数。
# 上界 r+6 不包含在内, 这个展示网格并不包含负二项分布的全部无限支持集。
total = np.arange(r, r + 6)
print("总试验次数: ", total)
# 要计算 P(N=total), 先将总次数减去固定成功次数 r, 转换为失败数 F。
# 这里 概率质量函数 数组各项为离散点概率; 只打印前六项时, 它们之和通常小于 1。
print("概率质量函数: ", failures.pmf(total - r))
# F 的理论均值是 r(1-p)/p; 加上 r 次成功, 得到 N 的理论均值 r/p。
print("平均失败次数、平均总次数: ", failures.mean(), failures.mean() + r)
# 加常数使均值平移、方差不变, 因此同一方差公式同时适用于 F 和 N。
assert np.isclose(failures.mean() + r, r / p)
assert np.isclose(failures.var(), r * (1 - p) / p ** 2)

# 教材图 3-3 只使用原例 p=0.3、r=3, 分别显示首次成功与第 r 次成功两种等待口径。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 几何变量从 1 开始; 绘图显示前 20 个可能编号, 理论支持集仍为所有正整数。
geom_k = np.arange(1, 21)
axes[0].bar(geom_k, geom.pmf(geom_k), width=0.75)
axes[0].set(title=f'首次成功: p={p}', xlabel='首次成功所在试验次数',
            ylabel='概率质量')
# 总次数从 r 开始。输入 SciPy 时必须减 r, 因为 failures 记录的是失败次数。
# 绘图区间比终端列出的 3..8 更宽, 仅改变显示范围, 没有改变随机变量的定义。
total_plot = np.arange(r, 31)
axes[1].bar(total_plot, failures.pmf(total_plot - r), width=0.8)
axes[1].axvline(failures.mean() + r, color='C1', linestyle='--',
                label=f'平均总次数 = {failures.mean() + r:g}')
axes[1].set(title=f'第 {r} 次成功: p={p}', xlabel='总试验次数',
            ylabel='概率质量')
axes[1].legend(fontsize=7.8)
fig.savefig('e10_waiting.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

无记忆性 (s、t、条件概率): 3 2 0.48999999999999994

无记忆性 (s、t、条件概率): 5 4 0.24010000000000004

总试验次数: [3 4 5 6 7 8]

概率质量函数: [0.027 0.0567 0.07938 0.09261 0.0972405 0.09529569]

平均失败次数、平均总次数: 6.999999999999999 10.0

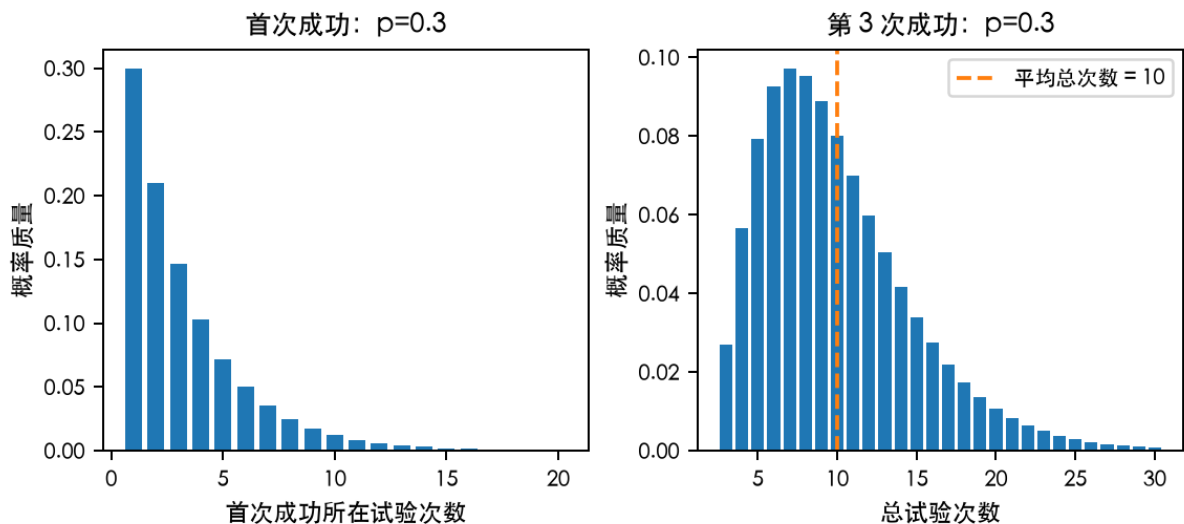


图 3-3 同一成功率 $p=0.3$ 下的两种等待口径, 由 `e10_waiting.py` 生成。左图为首次成功编号, 显示 1 至 20; 右图为第 $r=3$ 次成功的总试验次数, 显示 3 至 30, SciPy 评估传入总次数减 r , 虚线标示总次数均值 10。

读图指南 左图横轴从首次试验编号 1 起, 右图横轴从达到 3 次成功所需的最少总次数 3 起。右图调用失败数分布时先减去 3; 平均次数虚线无需位于最高柱上。

3.5 不放回抽样与超几何分布

从有限总体不放回抽样, 各次结果通常不独立。设总体有 N 个可区分对象, 其中 K 个为目标类, 均匀抽取 n 个不同对象, X 记样本中目标类个数。所有 $C(N,n)$ 个无序样本等可能; 若恰有 k 个目标, 需从 K 个目标中选 k 个, 再从 $N-K$ 个非目标中选 $n-k$ 个。两部分选法相乘, 除以总选法, 就得到超几何分布。

支持集不只是 0 至 n : 目标数不能超过 K , 非目标数 $n-k$ 不能超过 $N-K$ 。因此下界为 $\max(0, n-N+K)$, 上界为 $\min(n, K)$ 。先检查支持集可以立即发现一些错误概率, 例如总体仅 5 件次品时, 不放回抽样不可能抽出 6 件次品; 二项近似却可能给这件事一个小的正概率。

$$P(X = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}, \quad E[X] = nq, \quad \text{Var}(X) = nq(1-q) \frac{N-n}{N-1}, \quad q = \frac{K}{N} \quad (3.4)$$

N 为总体大小, K 为目标对象总数, n 为样本量, q 为总体目标比例。取 $0 \leq K \leq N$ 、 $0 \leq n \leq N$ 的整数且方差公式设 $N > 1$ 。有限总体修正 $(N-n)/(N-1)$ 小于等于 1, 表达不放回抽样的波动小于对应独立抽样。

修正因子可以从协方差推导。令 I_j 表示第 j 个抽到对象是否为目标, 则每个 $E[I_j] = q$ 。不同抽取位置的两次都为目标的概率为 $K(K-1)/[N(N-1)]$, 减去 q^2 得到 $\text{Cov}(I_i, I_j) = -q(1-q)/(N-1)$ 。把 $X = \sum I_j$ 代入和的方差, 得到 $nq(1-q) - n(n-1)q(1-q)/(N-1)$, 整理即为上述公式。负协方差反映抽到一个目标后, 余下目标比例下降。

若 n 远小于 N , 不放回对后续比例影响很弱, 超几何可由二项分布 $(n, K/N)$ 近似; 是否够好还取决于所关心的尾事件与误差要求。若 $n=N$, 样本包含总体全部对象, X 必等于 K , 修正因子为 0, 方差也为 0。此时继续套用二项方差会保留不存在的抽样不确定性。

程序 `ell_hypergeom.py` 本例 $N=50$ 、 $K=5$ 、 $n=10$, SciPy 对应参数名为 $M=50$ 、 $n=5$ 、 $N=10$ 。支持集为 0 至 5; 均值为 1, 修正因子 $40/49$; 超几何方差为 $0.9 \times 40/49 \approx 0.734694$, 而二项分布 $(10, 0.1)$ 方差为 0.9。图保留二项在 6 至 10 的尾部, 显示两模型支持集不同。接口参数名称容易混淆, 阅读时以“总体—目标—抽样”三种含义对应。

编程思路 从 50 件中含 5 件目标物的有限总体不放回抽取 10 件, 用超几何精确建模, 再与成功率 0.1 的二项近似比较。均值相同, 方差差异由有限总体校正系数给出。

代码对应

$X \sim \text{Hypergeom}(\text{总体 } 50, \text{目标 } 5, \text{抽取 } 10) \rightarrow \text{hyper} = \text{stats.hypergeom}(M=N, n=K, N=n):$

SciPy 的 M 、 n 、 N 分别是总体量、目标总量、样本量, 命名顺序需逐项对应。

独立有放回近似为 $\text{Binomial}(10, 5/50) \rightarrow \text{binomial} = \text{stats.binom}(n=n, p=K/N):$ 该模型保留每次的边际成功率, 却忽略不放回产生的负相关。

方差校正因子 $(50-10)/(50-1)=40/49 \rightarrow \text{correction} = (N - n) / (N - 1)$ ：用它乘二项方差可得到超几何方差。

易错提示 抽样比例 $10/50$ 并不很小，二项方差会明显偏大。超几何支持有上下界，图上超出支持的整数位置概率为零，并非数据遗漏。

正文代码 e11_hypergeom.py

```
# 教材位置：第3章，第3.5节；图3-4。
# 单位、变量与形状：N、K、n 是对象个数；k 是支持集网格，方差按目标个数平方计。
# 学习目标：比较不放回抽样的超几何分布与独立抽样的二项分布，理解有限总体修正。
# 核心关系：总体 N 个对象中有 K 个成功对象，不放回抽取 n 个时，成功数 X 的
#  $E[X]=nK/N$ ,  $\text{Var}(X)=n(K/N)(1-K/N)(N-n)/(N-1)$ ；最后一项是有限总体修正因子。
# 数据与输出：设  $N=50$ 、 $K=5$ 、 $n=10$ ，输出超几何支持集、两种模型的方差及至多一次成功的概率。
# 配套图件：运行后另保存 e11_hypergeom.png，教材直接采用该文件。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e11_hypergeom.py；本例只做理论计
# 算，无随机样本。

import numpy as np
from scipy import stats

# 这里的成功对象可以理解为带有某种标记的对象；每个大小为 n 的不放回样本等可能。
# 抽样后总体组成会变化，所以各次“是否成功”的结果不相互独立。
N, K, n = 50, 5, 10
# 特别注意 SciPy 参数命名：M 表示总体数，n 表示总体成功数，N 表示抽取数。
# 因此必须显式映射  $M=N$ 、 $n=K$ 、 $N=n$ ，不能只按照脚本变量名直接代入。
hyper = stats.hypergeom(M=N, n=K, N=n)
# 二项对照模型把每次试验当成相互独立、成功概率固定为  $K/N$ ，等价于有放回抽样的计数。
# 两种模型的均值相同，但一般概率和方差不同；抽样比例不很小时应谨慎使用二项近似。
binomial = stats.binom(n=n, p=K / N)
# 支持集下界为  $\max(0, n-(N-K))$ ，上界为  $\min(n, K)$ ，本例是  $0, \dots, 5$ 。
# 不放回时最多只能抽出总体已有的 5 个成功对象，而二项模型允许更多成功。
lower, upper = hyper.support()
# 转成整数后建立包含两端的支持集网格，形状为 (6,)，用于完整概率归一化检查。
k = np.arange(int(lower), int(upper) + 1)
# 本例修正因子为  $40/49$ ，小于 1；不放回抽样产生的负相关使计数方差较二项模型小。
# 当  $n/N$  很小时该因子接近 1；当抽完整个总体时，成功数固定为 K、方差为 0。
correction = (N - n) / (N - 1)
print("支持集端点: ", lower, upper)
# 两者都是给定模型的总体方差，不是样本方差或方差估计的误差。
print("超几何、二项模型的方差: ", hyper.var(), binomial.var())
#  $\text{cdf}(1)$  包含端点 1，即  $P(X=0)+P(X=1)$ ；二者差异反映抽样机制不同。
print("P(X<=1):", hyper.cdf(1), binomial.cdf(1))
#  $\text{hyper.pmf}(k)$  是完整支持集上的点概率，求和应为 1。
assert np.isclose(hyper.pmf(k).sum(), 1)
# 理论方差应等于二项方差乘有限总体修正因子，核对公式与参数映射是否一致。
assert np.isclose(hyper.var(), binomial.var() * correction)

# 教材图 3-4 保留本例  $N=50$ 、 $K=5$ 、 $n=10$ ，不另设总体规模或目标比例的演示情景。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
```

```
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 完整二项支持集为 0..n; 超几何在 k>5 时概率质量函数为 0, 保留这些位置能看见支持集差异。
# 此 plot_k 不替代上面用于检查超几何归一化的支持集数组 k。
plot_k = np.arange(n + 1)
axes[0].bar(plot_k - 0.18, hyper.pmf(plot_k), width=0.36, label='超几何分布')
axes[0].bar(plot_k + 0.18, binomial.pmf(plot_k), width=0.36, label='二项分布')
axes[0].set(title=f'N={N}, K={K}, n={n}', xlabel='目标个数 k',
            ylabel='概率质量', xlim=(-0.7, n + 0.7))
axes[0].legend(fontsize=7.8)
# 右图直接使用两个冻结对象的方差, 展示不放回抽样的有限总体修正。
variances = np.array([hyper.var(), binomial.var()])
axes[1].bar(['超几何分布', '二项分布'], variances, color=['C0', 'C1'], width=0.55)
for position, value in enumerate(variances):
    axes[1].text(position, value + 0.02, f'{value:.4f}', ha='center', fontsize=8)
axes[1].set(title=f'有限总体修正 = {correction:.4f}', ylabel='方差',
            ylim=(0, 1.05 * variances.max() + 0.1))
fig.savefig('e11_hypergeom.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

支持集端点: 0 5

超几何、二项模型的方差: 0.7346938775510204 0.9

$P(X \leq 1)$: 0.7418999792331364 0.7360989291

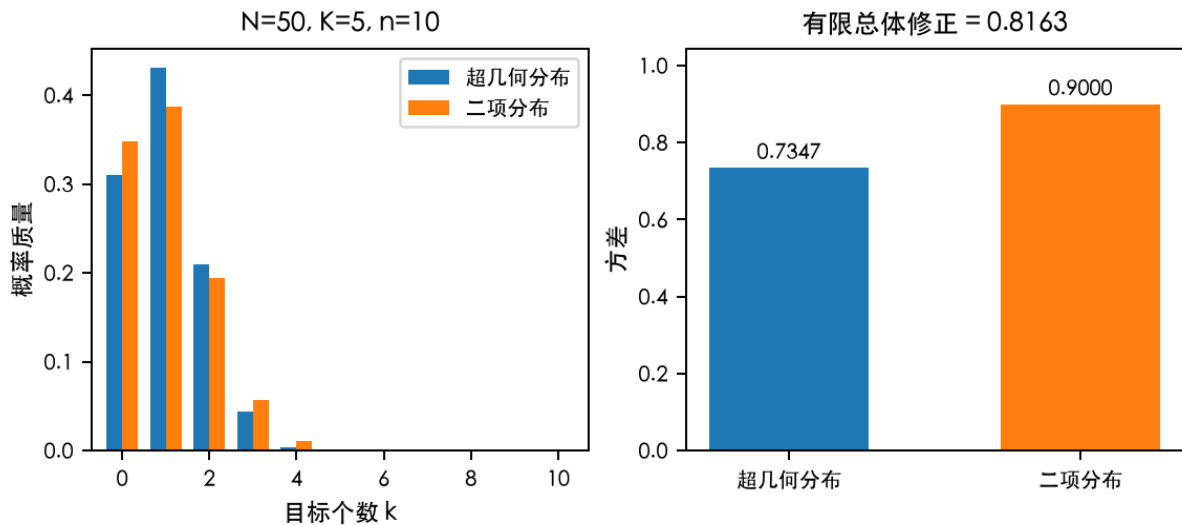


图 3-4 $N=50$ 、 $K=5$ 、 $n=10$ 的超几何模型与二项分布(10,0.1)对照, 由 `e11_hypergeom.py` 生成。左图保留二项在 $k>5$ 的尾部以显示支持集差异; 右图方差分别约 0.7347 与 0.9, 反映有限总体修正, 均值均为 1。

读图指南 两模型均值相同而散布不同, 超几何在目标数大于 5 时概率为 0, 二项仍有很小尾部。右图较小方差来自不放回产生的负关联, 不是画图归一化不同。

3.6 经验频率 模拟误差与模型检查

理论概率质量函数来自指定模型，经验频率来自有限样本。若独立生成 M 个 X ，每次判断是否等于 k ，便得到 M 个成功概率 $q=P(X=k)$ 的伯努利指示量。经验频率 $\hat{q}$ 的均值为 q ，方差为 $q(1-q)/M$ 。这把“模拟结果差多少算正常”转成了一个可计算问题。它同样说明罕见事件难以估计： q 很小时，预计命中数 Mq 可能很少，相对误差会很大。

标准化差 $(\hat{q}-q)/\sqrt{q(1-q)/M}$ 用标准误作为尺度。它不是必须落在某固定范围内的断言；在近似正态成立时，绝大多数重复实验会在常见范围内，也仍会有少数更极端的结果。不能因为一次标准化差较大就立即改动随机种子使它“好看”，也不能把某次差很小当作对所有输入都正确的保证。合适的检查包括支持集、质量归一化、解析矩、边界事件以及独立实现对照。

同一组经验频率各项之和为 1，因此不同 k 的频率存在约束，不是彼此独立的。若用全体频率评估模型，应采用适用于整体分布的统计量，并考虑参数是否由数据估计。逐个看柱形接近只能帮助理解，不能替代完整的模型诊断。

程序 `e12_simulation_check.py` 程序固定二项分布(20,0.3)，用种子 11 独立生成 50000 个计数。`bincount(...,minlength=21)` 保留 0 至 20 全部位置，即使某些值没出现也有 0 频数。目标事件为 $X=6$ ，理论 q 由 `pmf(6)` 给出，估计由 `mean(sample==6)` 给出。`se` 使用理论 q 计算，标准化差衡量本次模拟误差的尺度；同时核验频率和为 1，`frequency[6]` 等于同一事件的布尔平均。

编程思路 把 `Binomial(20,0.3)` 重复模拟五万次，逐个成功次数统计经验频率。针对恰好六次成功这一事件，把频率当作伯努利均值，计算理论标准误并量化一次模拟的偏差。

代码对应

重复抽取五万个二项计数 $\rightarrow$ `sample = rng.binomial(trials, p, size=count):`
`size=count` 是重复实验数量，每个元素自身已包含 `trials` 次伯努利试验。

经验概率=各整数结果出现次数/50000 $\rightarrow$ `frequency = np.bincount(sample,`
`minlength=trials + 1) / count:` `bincount` 与 `minlength=trials+1` 保留没有出现的支持位置，避免柱图错位。

恰好六次的理论事件概率 $q=P(X=6) \rightarrow q = \text{stats.binom.pmf}(6, \text{trials}, p)$: 二项 pmf

先确定事件概率, 再用它计算蒙特卡洛频率的波动尺度。

频率估计的标准误 $\sqrt{q(1-q)/\text{count}} \rightarrow \text{se} = \text{np.sqrt}(q * (1 - q) / \text{count})$: 分母是重复次数五万, 不是单个二项实验的 $\text{trials}=20$ 。

易错提示 一次模拟偏差除以标准误只是近似标准化量, 不构成算法正确性的确定证明。

固定种子用于复现; 应同时用理论概率与分布公式检查模拟。

正文代码 e12_simulation_check.py

```
# 教材位置: 第 3 章, 第 3.6 节; 仅终端输出。
# 单位、变量与形状: sample 为 (50000,), frequency 为 (21,); 事件频率和标准化差异无量纲。
# 学习目标: 用重复模拟核对二项分布的点概率, 并用标准误衡量模拟差异的通常尺度。
# 核心关系: 独立重复生成  $X \sim \text{二项分布}(20, 0.3)$ , 令  $I=1\{X=6\}$ , 则  $I \sim \text{Bernoulli}(q)$ ,
#  $q=P(X=6)$ 。count 次重复的样本比例标准误为  $\sqrt{q(1-q)/\text{count}}$ 。
# 数据与输出: 生成 50000 个成功次数, 打印理论/观察概率、概率估计的标准误与标准化差异,
# 以及理论/观察均值; 频数数组用于检查计数方式和归一化是否正确。
# 运行方式: 安装 NumPy、SciPy 后执行 python e12_simulation_check.py, 结果输出到终端。

import numpy as np
from scipy import stats

# 固定种子, 让同一环境下的模拟结果可复现; 不同种子产生不同的模拟误差。
rng = np.random.default_rng(11)
# trials=20 是一次二项实验内部的伯努利试验数, count=50000 是整个二项实验的重复次数。
# 概率估计的标准误使用 count; 不能误用 trials, 因为估计是对 count 个二项观测求比例。
trials, p, count = 20, 0.3, 50000
# sample 的形状为 (50000,), 每个元素是一次独立二项实验中的成功次数, 取值为 0, ..., 20。
sample = rng.binomial(trials, p, size=count)
# bincount 按整数值计数: 位置 k 保存 sample 中等于 k 的观测个数。
# minlength=21 保留可能未出现的高端类别; 除以 count 后得到经验点概率, 和为 1。
# 对离散类别而言这是相对频数, 不是连续直方图中的概率密度。
frequency = np.bincount(sample, minlength=trials + 1) / count
# q 由理论 概率质量函数 直接给出, 表示“一轮实验恰好有 6 次成功”的概率, 不是单次成功概率
# p。
q = stats.binom.pmf(6, trials, p)
# sample==6 形成布尔指示数组; 对它求均值就是相应事件的观察比例。
estimate = np.mean(sample == 6)
# 理论 q 在本例已知, 因此这里使用的是比例估计量的理论标准误, 而不是用 estimate 代入的估计值。
# 它是多次重新生成整个 50000 观测数据集时, estimate 的标准差, 不是原始成功次数的标准差。
se = np.sqrt(q * (1 - q) / count)
print("理论概率、模拟频率: ", q, estimate)
# (estimate-q)/se 说明本次有符号差异相当于多少个标准误; 其有限样本分布并非严格正态。
# 因指示变量独立同分布且方差有限, 大样本下可用 中心极限定理 将这个量近似看作标准正态变量
# 。
print("理论标准误、标准化差异: ", se, (estimate - q) / se)
# 原始成功次数的理论均值为 trials*p; sample.mean() 是对它的另一项模拟核对。
print("理论均值、样本均值: ", trials * p, sample.mean())
# 验证从 0 到 trials 的每个可能成功次数都有一个频率槽位, 包括零频数类别。
assert frequency.shape == (trials + 1,)
```



```
# 所有观测都恰好归入一个类别，因此经验概率和应为 1，只需允许浮点舍入误差。
assert np.isclose(frequency.sum(), 1)
# 两种统计同一事件的方法应一致；这检查统计实现，没有要求经验概率精确等于理论  $q$ 。
# 上述断言通过也不等同于完整的分布拟合检验，仍应结合模拟误差尺度解释理论与观察的差异。
assert np.isclose(frequency[6], estimate)
```

规定环境下的实际运行输出

理论概率、模拟频率： 0.1916389827534427 0.19172

理论标准误、标准化差异： 0.0017601902342796267 0.046027551442726966

理论均值、样本均值： 6.0 5.99784

实际建模时还要区分分布内的抽样误差与模型外的假设误差。把 50000 次模拟增加到 500000 次能减小数值估计误差，却不能修复“真实抽样不独立”“总体成功率在变化”这类模型偏差。理论推导、计算核验与实际数据检查各有作用；看清它们分别回答什么问题，比无限增加模拟次数更重要。

3.7 本章要点 常见误区与延伸阅读

核心要点：二项模型固定试验数，几何与负二项固定成功目标，泊松模型固定暴露窗口，超几何模型固定有限总体且不放回。都取整数不代表可互换；停止规则、支持集与独立性优先于公式外形。

核心要点：泊松窗口参数为速率乘暴露量，时间单位必须一致。均值等于方差是模型性质，实际偏离可能来自不同速率混合、依赖或暴露量不一致。二项到泊松极限同时要求试验数增大、单次成功概率缩小而均值保持。

常见误区：把 pmf 与 cdf 混用，把“至少 k ”直接调用 $\text{sf}(k)$ ，把 geom 的从 1 开始与 nbinom 的从 0 数失败混淆，把不放回抽样当作独立试验，或用无限次模拟掩盖错误支持集。完整有限分布的质量和应为 1；截断的无限支持网格还要说明遗漏尾概率。

延伸阅读：哈佛《概率论导论》第 3 章与第 4 章中的二项、超几何、几何、负二项和泊松例子，重点按试验故事区分计数口径；作者教材入口 <https://probabilitybook.net/> [P19]。统计 110 第 7—11 讲依次连接这些离散模型，官方讲次目录 <https://stat110.hsites.harvard.edu/youtube> [P14]。

课程配读：麻省理工学院 18.05 第 4 课甲乙及第 1 单元复习材料，从官方总目 <https://ocw.mit.edu/courses/18-05-introduction-to-probability-and-statistics-spring-2022/pages/classes-reading-and-in-class-materials/> 进入[P02、P04]。阅读时每次先注明随机变量数的是成功、失败、总次数还是给定时间内事件数。

3.8 练习与参考解答

练习 3-1（概念与手算，入门） 独立试验成功率 0.2，固定做 5 次。求恰成功 1 次、至少成功 1 次的概率及均值、方差，并说明 `comb(5,1)` 在质量公式中对应什么。

练习 3-2（概念与手算，入门） 教学设定：泊松速率 3 次/小时。半小时内计数的参数、均值、方差与无事件概率各是什么？如果时间写成 30 分钟，速率应怎样换算？

练习 3-3（概念与手算，进阶） 首次成功编号 T 服从成功率 0.25 的几何分布，已知前 4 次失败，再等超过 2 次的概率是多少？若等第 3 次成功，总次数为 5 的事件在 SciPy 负二项接口中传入哪个数？

练习 3-4（概念与手算，进阶） $N=10$ 、 $K=4$ ，不放回均匀抽 $n=3$ ，求恰有 2 个目标的概率、均值、方差与支持集，并与二项模型比较。

练习 3-5（编程实践，入门） 按 `e07_binomial.py` 的 $n=20$ 、 $p=0.3$ ，使用 `comb` 逐点计算完整质量，再与 `stats.binom.pmf` 核对。给出至少 8 次事件所需的正确尾接口。

练习 3-6（编程实践，进阶） 运行 `e08_poisson_limit.py`，记录 $n=10$ 、50、500 时的 p 与最大点概率差。解释为何这不是全部尾事件的最大误差。

练习 3-7（编程实践，进阶） 在 `e10_waiting.py` 中分别用组合公式与 `nbinom` 计算总次数 3 至 8 的质量，核验计数起点与均值平移。

练习 3-8（编程实践，进阶） 运行 `e12_simulation_check.py`，说明 `bincount` 的 `minlength` 和频数索引 6 的作用；计算单事件模拟差的标准量化。

练习 3-9（场景应用，入门） 教学设定：客服来电率 12 次/小时，按齐次泊松过程建模。15 分钟内至少 2 通的概率是多少？给出一个会破坏该模型的具体机制。

练习 3-10 (场景应用, 进阶) 教学设定: 服务器平均每秒 2 次外部请求。半秒计数用什么参数? 若每个外部请求都恰好触发一次立即重试, 记录总请求还适合相同泊松模型吗?

练习 3-11 (场景应用, 进阶) 教学设定: 独立设备在相同稳定工况下, 平均每台 100 小时 0.2 次故障。10 台各观察 50 小时, 总计故障的均值和无故障概率是多少? 若有两类设备速率不同, 直接合并有何前提?

练习 3-12 (场景应用, 进阶) 质检批量 50 件且明确只有 5 件次品, 不放回抽 10 件。研究者用二项分布(10,0.1)预测抽到 6 件次品。说明正确概率并比较两模型方差。

参考解答与验收依据

3-1 恰 1 次为 $C(5,1)0.2 \times 0.8^4 = 0.4096$; 至少 1 次为 $1 - 0.8^5 = 0.67232$; 均值 1、方差 0.8。组合数计算成功位置的选择数, 单个排列的概率由 0.2×0.8^4 给出。comb 不是概率本身。

3-2 参数 $3 \times 0.5 = 1.5$, 均值与方差均为 1.5, 无事件概率 $\exp(-1.5) \approx 0.223130$ 。分钟速率为 $3/60 = 0.05$ 次/分钟, 0.05×30 仍为 1.5。速率与窗口计数参数不可混用。

3-3 无记忆性给 $P(T > 6 | T > 4) = 0.75^2 = 0.5625$ 。第 3 次成功时总次数 5 意味着之前失败 2 次, 所以向 `stats.nbinom(n=3, p=0.25).pmf` 传入 2; n 参数在该接口是目标成功次数。几何接口 `geom` 从 1 数总次数, 负二项接口 `nbinom` 从 0 数失败次数。

3-4 概率 $C(4,2)C(6,1)/C(10,3) = 36/120 = 0.3$; 均值 $3 \times 0.4 = 1.2$; 方差 $3 \times 0.4 \times 0.6 \times 7/9 = 0.56$; 支持 0 至 3。相应二项方差 0.72, 缺少有限总体修正; 二者均值相同不能推出分布相同。

3-5 k 从 0 至 20, 共 21 项; `manual[k] = comb(20,k) × 0.3k × 0.720-k`。allclose 与质量和约 1 应通过, 均值 6、方差 4.2。 $P(X=6) \approx 0.19163898$ 、 $P(X \leq 4) \approx 0.23750778$, $P(X \geq 8) = sf(7) \approx 0.22772820$; `sf(8)` 会漏掉 8 次的质量。

3-6 固定 $\lambda=3$, p 依次为 0.3、0.06、0.006; 在 k=0 至 29 的最大绝对质量差约为 0.042786、0.007015、0.000675。它对有限网格逐点质量取最大, 不是对所有事件的概率差取最大, 也不是把各点差绝对值相加的总变差距离。图只显示 0 至 12, 计算仍用 0 至 29。

3-7 令 $\text{total}=[3,4,5,6,7,8]$ ，接口使用 $\text{failures.pmf}(\text{total}-3)$ 。组合公式 $C(\text{total}-1,2)0.3^30.7^{**}(\text{total}-3)$ 得到 $[0.027,0.0567,0.07938,0.09261,0.0972405,0.09529569]$ 。失败数均值 7、总次数均值 10，方差均为 70/3；只打印这 6 项时和不必等于 1，因为后面仍有尾部。

3-8 $\text{minlength}=21$ 保留 0 至 20 所有位置，缺失类别也占一个 0 频数槽。 $\text{frequency}[6]$ 与 $\text{mean}(\text{sample}==6)$ 应相同。规定种子 11 得到理论概率约 0.19163898、频率 0.19172、标准误约 0.00176019、标准化差约 0.04602755；理论均值 6、样本均值 5.99784。模拟差很小不能代替完整模型诊断。

3-9 窗口参数为 3， $P(N \geq 2) = 1 - \exp(-3)(1+3) \approx 0.80085173$ 。若集中发送通知后大量用户同时来电，事件可能聚集；若午间与晚间速率显著不同，跨时段平稳速率假设不成立。不能把这些机制当作已由题设真实观测证实。

3-10 原外部请求半秒计数参数 1。若每个事件成对记录，总请求 $C=2N$ 只取偶数，均值 2、方差 4，不可能是均值 2 的泊松分布（后者方差 2 且允许奇数）。应明确区分外部请求与含重试的总日志计数。

3-11 总暴露 500 台时，平均故障数 $0.2 \times 500 / 100 = 1$ ，无故障概率 $\exp(-1) \approx 0.367879$ 。不同确定速率的独立泊松计数仍可相加，参数按各自暴露量相加；但若把随机抽到的设备类别混合作为单台计数，或共享冲击造成依赖，不能直接援用独立泊松叠加。

3-12 实际事件不可能，正确概率 0；超几何支持 0 至 5。超几何方差 $0.9 \times 40 / 49 \approx 0.734694$ ，二项方差 0.9。抽样比例 20% 并不极小，独立抽样近似应评估误差；单看平均都是 1 不能说明尾概率合理。

第 4 章 连续概率分布

4.1 本章导读 入门 进阶与拓展

本章把区间概率、密度和等待过程连起来。入门阅读掌握面积、标准化、速率与尺度的换算。进阶阅读理解泊松过程的独立增量、平稳增量与短时间性质，并推导指数间隔和整数形状的伽马等待。拓展阅读讨论变量变换、非整数伽马形状及贝塔函数。过程模型是一组关于整条时间记录的假设，要求强于“某一个计数近似泊松”。

4.2 概率密度 累积概率与变量变换

测量时间、长度或浓度时，取值常用连续模型近似。对具有概率密度的 X ，存在非负函数 f_x ，使任意区间的概率等于曲线下相应面积。总面积必须为 1，但曲线高度可以超过 1。例如把全部概率均匀压在长度 0.1 的区间内，密度高度就是 10。高度表示单位长度上的概率密集程度，和“恰在某点”的概率不同。

连续变量在单点的概率为 0，因为单点区间没有长度；一个观测仍然会出现某个具体数值，这并不矛盾。概率 0 表示该特定点在模型下没有正的概率质量，并不表示样本空间中不存在这个点。实际测量有分辨率，记录“2.30 秒”常代表一个窄区间，计算记录值的机会时应使用相应区间概率。

累积分布函数仍按 $F_X(x)=P(X\leq x)$ 定义，因而离散与连续模型没有换一套累积概率语言。有密度时累积分布函数连续，并在适当位置满足 $F'_X(x)=f_X(x)$ 。反过来，累积分布函数连续本身并不足以保证存在通常的密度；本书讨论的连续分布都具有密度，不展开那些特殊的无密度连续分布。

$$F_X(x) = \int_{-\infty}^x f_X(t) dt, \quad P(a < X \leq b) = \int_a^b f_X(t) dt = F_X(b) - F_X(a) \quad (4.1)$$

$\int$ 是积分， t 为积分变量， dt 表示按 t 的长度累积； a 、 b 为区间端点， $-\infty$ 表示向左无限延伸。密度 $f_x \geq 0$ 且全实轴积分为 1。对有密度的变量，单点概率为 0，故端点处使用 $<$ 或 $\leq$ 不改变区间概率；离散变量则不能这样任意替换。

连续期望是 $E[g(X)] = \int g(x)f_x(x)dx$ ，只要绝对值积分有限。它仍然是按分布加权，只是把离散求和换为积分。计算方差仍可先求 $E[X]$ 、 $E[X^2]$ 再相减。分位数 q_p 是使累计概率达到 p 的位置，一般定义为 $\inf\{x:F_x(x)\geq p\}$ ；对严格递增连续累积分布函数，它就是反函数 $F_x^{-1}(p)$ 。概率到阈值的方向，与阈值到概率的累积分布函数方向正好相反。

变量变换最好先从事件与累积分布函数出发。例如 $Y=aX+b$ 且 $a>0$ 时， $Y\leq y$ 等价于 $X\leq(y-b)/a$ ，所以 $F_Y(y)=F_X((y-b)/a)$ 。求导得到 $f_Y(y)=f_X((y-b)/a)/a$ ； $a<0$ 时不等号方向改变，最终密度公式分母使用 $|a|$ 。这个因子补偿横轴伸缩，保证总面积仍为 1。若不乘伸缩因子，变换后的曲线通常不是合法密度。

$$f_Y(y) = f_X(g^{-1}(y)) \left| \frac{d}{dy} g^{-1}(y) \right| \quad (4.2)$$

此式适用于支持集上单调、可微且反函数具有适当可微性的变换 $Y=g(X)$ 。 g^{-1} 为反函数，导数绝对值是长度尺度的修正因子。非单调变换可能有多个原值映到同一个 y ，应把各分支贡献相加；不能未经检查直接套用单分支公式。

例题 手算 4-1：均匀变量平方后的分布

令 U 在 $(0,1)$ 均匀， $Y=U^2$ 。对 $0\leq y\leq 1$ ， $F_Y(y)=P(U\leq\sqrt{y})=\sqrt{y}$ ，故 $0<y<1$ 时 $f_Y(y)=1/(2\sqrt{y})$ 。它在 0 附近可大于 1 且趋于无穷，但从 0 到 1 的积分仍为 1。

$E[Y]$ 可直接按 U 的分布算 $\int_0^1 u^2 du=1/3$ ；也可按新密度算 $\int_0^1 y/(2\sqrt{y}) dy=1/3$ 。两种方法给同一答案，说明期望可在变换前后两个分布中计算。

4.3 均匀分布 长度比例成为概率

若 X 在区间 $[a,b]$ 上均匀， $a<b$ ，记 $X\sim$ 均匀分布 (a,b) 。均匀指同样长度的子区间具有相同概率，而不是给无限多个点各分配一个正概率。设密度恒为 c ，总面积 $c(b-a)=1$ 给出 $c=1/(b-a)$ 。区间外密度为 0，累积分布函数在 a 左边为 0，在 a 至 b 之间线性上升，在 b 右边为 1。

均匀分布适用于没有位置偏好的有限范围随机化机制，如在给定线段按长度均匀选点。仅知道最小值与最大值不足以推出中间均匀；实际误差可能更集中在中心。参数 a 、 b 应来自试验设计或明确假设，不能只因为数据都落在一个范围就认定其密度为常数。

$$f_X(x) = \frac{1}{b-a} \quad (a < x < b), \quad E[X] = \frac{a+b}{2}, \quad \text{Var}(X) = \frac{(b-a)^2}{12} \quad (4.3)$$

a 、 b 分别为左、右端点，区间外密度为 0。密度端点的单点取值不影响积分。均值由 $\int_a^b x/(b-a)dx$ 得到；把 X 写成 $a+(b-a)U$ ， $U \sim$ 均匀分布(0,1)，可由 $E[U]=1/2$ 、 $E[U^2]=1/3$ 推得方差。

例题 手算 4-2：在 2 至 6 之间均匀取值

$X \sim$ 均匀分布(2,6)，密度高度为 $1/4$ 。区间(3,5)宽 2，所以概率为 $2/4=1/2$ ； $F_X(3)=1/4$ 、 $F_X(5)=3/4$ ，两者相减也得 $1/2$ 。均值 4，方差 $16/12=4/3$ 。

第 90 百分位数满足 $(x-2)/4=0.9$ ，故为 5.6。这里 0.9 是左侧累计概率；若问上尾概率 0.9 对应的阈值，左侧累计概率应为 0.1，答案将为 2.4。

程序 e13_uniform.py `stats.uniform(loc=2,scale=4)` 中的 `loc` 是左端点、`scale` 是区间长度，所以支持区间为 [2,6]。`pdf(4)=0.25` 是密度；`cdf(5)-cdf(3)=0.5` 才是区间概率。程序绘制密度与累积分布函数，并核验方差 $16/12$ 。若把 `scale` 误传为右端点 6，就会得到 [2,8] 的另一个模型。

编程思路 用 `loc=2`、`scale=4` 表示区间 [2,6] 的均匀分布，分别查看区间外和端点附近的密度与分布函数。区间概率由 CDF 相减计算，并与长度比直接核对。

代码对应

$U(2,6)$ 的密度为 $1/(6-2)=1/4 \rightarrow rv = \text{stats.uniform}(loc=2, scale=4)$ ：SciPy 的第二个参数是区间长度 `scale=4`，不是右端点 6。

检查区间外、端点与区间内的若干位置 $\rightarrow x = \text{np.array}([0, 2, 3, 5, 6, 8], \text{dtype=float})$ ：离散数组用于比较 pdf 与 cdf 的不同含义，不能把密度值当成点概率。

把密度和累计分布分开画 → `fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8)):`

两列图分别对应分段常数密度和分段线性 CDF，纵轴的量纲与范围不同。

易错提示 连续分布在任意单点的概率为零； $\text{pdf}(4)=0.25$ 是密度。

$P(3 \leq X \leq 5) = \text{cdf}(5) - \text{cdf}(3) = 0.5$ ，是否包含端点不改变连续分布概率。

正文代码 **e13_uniform.py**

```
# 教材位置：第 4 章，第 4.3 节；图 4-1。
# 单位、变量与形状：x 为 (6,)；观测使用一般测量单位，密度单位是测量单位的倒数。
# 学习目标：用均匀分布把“概率密度、区间概率、累积分布函数”联系起来。
# 核心数学关系：若  $X \sim \text{Uniform}(a,b)$ ，则区间内密度为  $1/(b-a)$ ，
#  $F(x)$  在  $[a,b]$  内等于  $(x-a)/(b-a)$ ，方差为  $(b-a)^2/12$ 。
# 数据与输出：本例取区间  $[2,6]$ ，打印一点的密度和一段区间的概率，
# 并生成 e13_uniform.png；左图的阴影面积对应概率，右图展示 累积分布函数。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后，在终端运行
# python e13_uniform.py；图片写入运行命令时的当前工作目录。

# NumPy 负责数组和浮点比较，SciPy 提供分布对象，Matplotlib 负责作图。
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# SciPy 的 uniform 用 loc 表示左端点，用 scale 表示区间宽度，
# 因而这里右端点是  $\text{loc} + \text{scale} = 6$ ；scale 不是右端点本身。
# “冻结”后的 rv 保存这组参数，后续 pdf、cdf、var 都针对同一个分布。
rv = stats.uniform(loc=2, scale=4)
# 长度为 6 的一维浮点数组；特意覆盖支持区间左侧、内部、边界和右侧。
# rv.cdf 接受整个数组并逐元素返回结果，不必手写循环。
x = np.array([0, 2, 3, 5, 6, 8], dtype=float)
#  $\text{pdf}(4)=1/4$  是密度高度，而不是  $P(X=4)$ ；连续分布在单点的概率为 0。
print("x=4 处的概率密度：", rv.pdf(4))
# 区间概率是 累积分布函数 的差，也等于“宽度 2 × 密度 1/4”的面积。
# 连续分布的端点概率为零，因此这里开区间和闭区间的概率相同。
print("P(3<X<5):", rv.cdf(5) - rv.cdf(3))
# assert 将理论公式变成可执行的自检；浮点运算用 isclose 而非直接 ==。
assert np.isclose(rv.cdf(5) - rv.cdf(3), 0.5)
# 方差由区间宽度决定： $4^2/12$ ；平移区间不会改变方差。
assert np.isclose(rv.var(), 16 / 12)
# 创建一行两列子图；axes[0] 和 axes[1] 分别访问左右两个坐标轴。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8))
# 重复的横坐标 2、6 用于画密度函数的竖直边界，不代表边界处有概率质量。
axes[0].plot([0, 2, 2, 6, 6, 8], [0, 0, 0.25, 0.25, 0, 0])
# 在 3 到 5 之间填充至密度曲线的面积；alpha 是显示透明度。
axes[0].fill_between([3, 5], [0.25, 0.25], alpha=0.3)
# 累积分布函数 在区间左侧为 0，内部线性上升，区间右侧为 1。
# 这些采样点覆盖斜率变化处，所以连接后的折线能表达本例的完整 累积分布函数。
axes[1].plot(x, rv.cdf(x))
# 标注纵轴，帮助区分“密度高度”和“累计概率”这两种不同量。
axes[0].set(xlabel="x", ylabel="概率密度")
```



```
axes[1].set(xlabel="x", ylabel="累积分布函数")
# tight_layout 调整子图间距; savefig 按 160 dpi 保存图片。
fig.tight_layout()
fig.savefig("e13_uniform.png", dpi=160)
# 释放本图资源, 便于连续运行多个案例。
plt.close(fig)
```

规定环境下的实际运行输出

$x=4$ 处的概率密度: 0.25

$P(3 < X < 5)$: 0.5

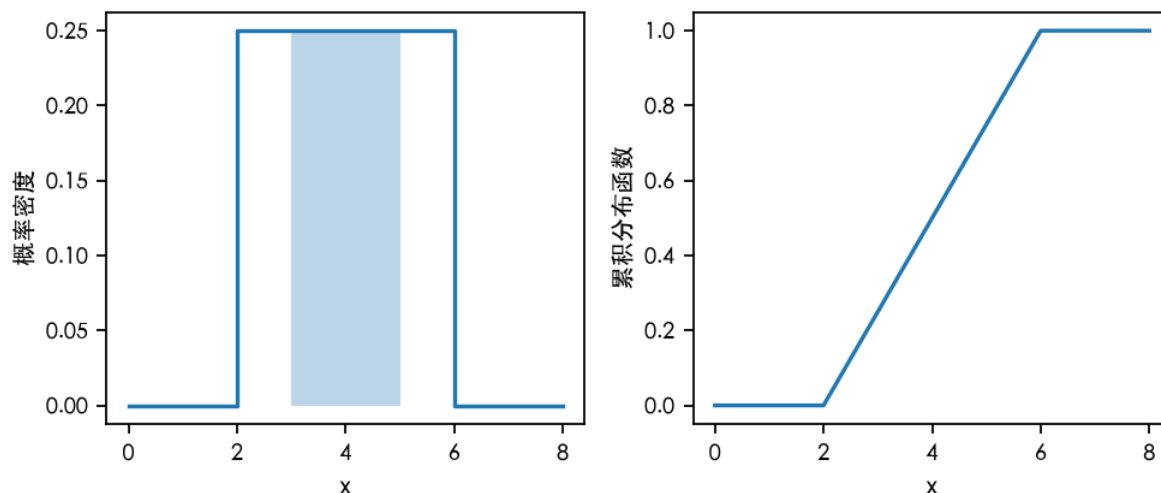


图 4-1 $U(2,6)$ 的密度与累积分布函数, 直接采用 `e13_uniform.py` 运行图。左图密度高度 $1/4$, 区间 $[3,5]$ 阴影面积为 $1/2$; 右图累积分布函数在 $[2,6]$ 线性上升, 两端连续。

读图指南 左图 $1/4$ 是密度高度, 阴影宽 2, 二者相乘才得概率 $1/2$ 。右图在区间外水平、内部线性上升, 两端连续; 单个点的密度高度不是该点概率。

4.4 正态分布与标准化

正态分布用一个对称的钟形密度描述围绕中心的连续波动, 记 $X \sim N(\mu, \sigma^2)$, μ 为均值, $\sigma > 0$ 为标准差。 μ 控制中心位置, σ 控制横向尺度。它的支持集是整个实数轴, 因此用于严格非负变量时, 要检查负值区域概率是否小到可以接受。外形近似钟形也只是模型线索, 不能代替生成机制和残差诊断。

将 X 减去 μ 并除以 σ , 得到标准分数 $Z = (X - \mu) / \sigma$ 。若 X 确为上述正态分布, $Z \sim N(0, 1)$ 。这个变换把不同单位、不同中心的正态概率统一到同一张标准正态表。记 ϕ 为标准正态密度, Φ

为标准正态累积分布函数，则 $P(X \leq x) = \Phi((x - \mu)/\sigma)$ 。数值标准化对任何变量都可以做，但只有正态变量标准化后才必为标准正态。

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right], \quad P(a < X < b) = \Phi\left(\frac{b - \mu}{\sigma}\right) - \Phi\left(\frac{a - \mu}{\sigma}\right) \quad (4.4)$$

π 是圆周率， $\exp(u) = e^u$ 。 $\mu \in \mathbb{R}$ ， $\sigma > 0$ ，密度在所有实数 x 上为正。 $\Phi(z) = P(Z \leq z)$ ，不是密度 $\phi(z)$ 。式中每个标准分数均无量纲。正态分布通常按均值和方差记为 $N(\mu, \sigma^2)$ ，而软件 `scale` 参数使用标准差 σ 。

标准正态对称，所以 $\Phi(-z) = 1 - \Phi(z)$ 。由此可得 $P(|Z| \leq c) = 2\Phi(c) - 1$ 。 $c=1, 2, 3$ 时覆盖率约为 68.27%、95.45%、99.73%。这些是特定分布的数值性质，并不是任意变量都满足的经验法则。单侧 95% 阈值是 $\Phi^{-1}(0.95) \approx 1.64485$ ，双侧中心 95% 区间要给两尾各留 2.5%，临界值是 $\Phi^{-1}(0.975) \approx 1.95996$ ，二者不可混用。

正态变量的线性组合也有重要性质：若 X, Y 独立且分别正态， $aX + bY$ 仍正态，均值为 $a\mu_x + b\mu_y$ ，方差为 $a^2\sigma_x^2 + b^2\sigma_y^2$ 。独立性使协方差消失；若只知道两个边缘都正态，尚不能无条件断言任意和都正态。这个区别将在多元统计和线性代数中的协方差矩阵里继续出现。

程序 `e14_normal.py` 程序取 $\mu=170$ 、 $\sigma=8$ 、阈值 178。标准化得到 $z=1$ ，所以 $P(X \leq 178) = \Phi(1) \approx 0.841345$ 。它还计算 $\pm 1, \pm 2, \pm 3$ 标准差的中心覆盖率，并分别输出单侧 95% 与双侧 95% 的标准正态临界值。原尺度上尾 5% 阈值为 $170 + 8\Phi^{-1}(0.95) \approx 183.159$ ；`stats.norm` 的 `scale=8` 表示标准差，不是方差 64。

编程思路 先把测量阈值 178 按 $\mu=170$ 、 $\sigma=8$ 转成 $z=1$ ，再用原分布与标准正态分布计算同一左侧累计概率。把中心覆盖区间、单侧分位数与双侧临界值分开解释。

代码对应

$X \sim N(170, 8^2) \rightarrow rv = \text{stats.norm}(loc=\mu, scale=\sigma):$ `scale` 接收标准差 8，而不是方差 64。

$z=(178-170)/8=1 \rightarrow z = (\text{threshold} - \mu) / \sigma$: 标准化保持事件方向, 使 $X \geq 178$

对应标准正态 $Z \geq 1$ 。

$P(\mu - m\sigma \leq X \leq \mu + m\sigma) \rightarrow \text{coverage} = \text{stats.norm.cdf}(\text{width}) - \text{stats.norm.cdf}(-\text{width})$: 循环 $m=1, 2, 3$ 时以两个 CDF 之差求覆盖率。

易错提示 右尾极小概率宜调用 `sf`, 避免 $1-\text{cdf}$ 的抵消误差。单侧 5% 上阈值用 `ppf(0.95)`, 双侧中间 95% 的上端用 `ppf(0.975)`, 两者不可混用。

正文代码 e14_normal.py

```
# 教材位置: 第 4 章, 第 4.4 节; 图 4-2。
# 单位、变量与形状:  $\mu$ 、 $\sigma$ 、threshold 采用同一般测量单位;  $z$  无量纲, 原密度单位为测量单位
# 的倒数。
# 学习目标: 理解正态变量的标准化、若干倍标准差的覆盖率及尾部分位点。
# 核心数学关系:  $X \sim N(\mu, \sigma^2)$  时,  $Z=(X-\mu)/\sigma \sim N(0,1)$ ,
# 因此  $P(X \leq x) = \Phi((x-\mu)/\sigma)$ , 分位数为  $\mu + \sigma * \Phi^{-1}(p)$ 。
# 数据与输出: 使用均值 170、标准差 8 的假想测量模型, 打印阈值 178
# 的累计概率、中心覆盖率, 以及单侧/双侧 95% 对应的临界值; 保存 e14_normal.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e14_normal.py。

# NumPy 用于浮点自检; SciPy 提供正态分布的 累积分布函数 和逆 累积分布函数。
import numpy as np
from scipy import stats

# sigma 表示标准差, 须为正; 方差为  $\sigma^2=64$ 。
# threshold 是要评估的原始量纲阈值, 与  $\mu$  使用相同单位。
mu, sigma, threshold = 170.0, 8.0, 178.0
# norm 的 loc 是均值, scale 是标准差, 不能把方差 64 填入 scale。
rv = stats.norm(loc=mu, scale=sigma)
# 标准化后  $z=1$ , 表示阈值位于均值上方一个标准差处。
z = (threshold - mu) / sigma
# 累积分布函数 返回左侧累计概率  $P(X \leq 178)$ , 不是阈值处的密度。
print("标准分数  $z$ 、累积概率: ", z, rv.cdf(threshold))
# stats.norm 不指定参数时是标准正态分布; 验证两种计算途径相同。
assert np.isclose(rv.cdf(threshold), stats.norm.cdf(z))
# width 以标准差为单位: 分别计算中心区间  $\mu \pm \sigma$ 、 $\mu \pm 2\sigma$ 、 $\mu \pm 3\sigma$ 。
for width in [1, 2, 3]:
    # 累积分布函数 相减给出  $P(-\text{width} \leq Z \leq \text{width})$ , 即中心覆盖率, 而非单侧概率。
    coverage = stats.norm.cdf(width) - stats.norm.cdf(-width)
    # :.6% 将 0 到 1 的概率按百分数输出, 并保留六位小数。
    print(f"均值两侧 {width} 个标准差内: {coverage:.6%}")
# ppf 是 累积分布函数 的反函数; 0.95 分位点右侧还剩 5% 的概率质量。
# 此处调用带参数的 rv, 所以结果仍使用原始测量单位。
print("右尾概率 5% 的阈值: ", rv.ppf(0.95))
# 单侧 95% 界把全部 5% 留在一侧, 临界值为  $\Phi^{-1}(0.95)$ 。
# 双侧中心 95% 区间把 5% 均分为两尾, 右临界值是  $\Phi^{-1}(0.975)$ ,
# 左临界值为其相反数; 两种临界值都处于标准化后的  $z$  尺度。
print("单侧 95%、双侧 95% 临界值: ",
      stats.norm.ppf(0.95), stats.norm.ppf(0.975))

# 教材图 4-2 把原尺度模型与标准化后的模型并列; 主参数仍是  $\mu=170$ 、 $\sigma=8$ 、 $\text{threshold}=178$ 。
```

```
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 原尺度密度网格覆盖均值左右四个标准差；网格只作显示，不改变总体参数。
normal_x = np.linspace(mu - 4 * sigma, mu + 4 * sigma, 600)
normal_pdf = rv.pdf(normal_x)
axes[0].plot(normal_x, normal_pdf)
# 阴影是阈值左侧的累计概率；阈值线与上面计算的 z=1 对应。
axes[0].fill_between(normal_x, normal_pdf, where=normal_x <= threshold,
                    alpha=0.25)
axes[0].axvline(threshold, color='C1', linestyle='--',
                label=f'P(X <= {threshold:g}) = {rv.cdf(threshold):.6f}')
axes[0].set(title=f'正态: 均值={mu:g}, 标准差={sigma:g}',
            xlabel='原尺度数值 x', ylabel='概率密度')
axes[0].legend(fontsize=7.8, loc='upper left')
# 右图的横轴是无量纲 z；同一幅标准正态密度标出三个中心覆盖区间。
standard_z = np.linspace(-4, 4, 801)
standard_pdf = stats.norm.pdf(standard_z)
for band_width, band_color in [(3, '#C9DFEB'), (2, '#86B5CE'), (1, '#E7A067')]:
    band_coverage = stats.norm.cdf(band_width) - stats.norm.cdf(-band_width)
    axes[1].fill_between(standard_z, standard_pdf,
                        where=np.abs(standard_z) <= band_width,
                        color=band_color, alpha=0.9,
                        label=f'+/{band_width} 个标准差内: {band_coverage:.2%}')
axes[1].plot(standard_z, standard_pdf, color='C0', linewidth=1)
axes[1].axvline(z, color='C1', linestyle='--', linewidth=1)
axes[1].set(title=r'标准化 $Z=(X-\mu)/\sigma$',
            xlabel='标准分数 z', ylabel='概率密度')
axes[1].legend(fontsize=7.8, loc='upper right')
fig.savefig('e14_normal.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

标准分数 z、累积概率: 1.0 0.8413447460685429

均值两侧 1 个标准差内: 68.268949%

均值两侧 2 个标准差内: 95.449974%

均值两侧 3 个标准差内: 99.730020%

右尾概率 5% 的阈值: 183.1588290156118

单侧 95%、双侧 95% 临界值: 1.644853626951472 1.959963984540054

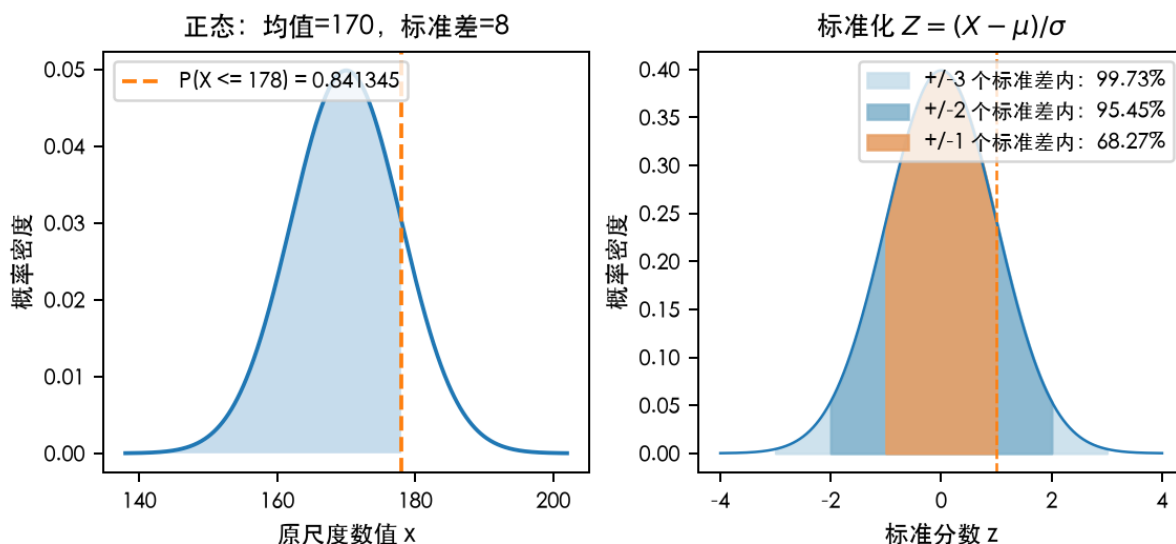


图 4-2 原尺度正态模型及其标准化，由 `e14_normal.py` 生成。左图 $X \sim N(170, 8^2)$ ，阈值 178 对应 $z=1$ ，阴影为 $P(X \leq 178) \approx 0.841345$ ；右图 $Z \sim N(0, 1)$ ，三层阴影分别对应 ± 1 、 ± 2 、 ± 3 标准差，覆盖率约 68.27%、95.45%、99.73%。

读图指南 两面板的横轴单位与密度高度不同，但阈值 178 与标准分数 1 描述同一累计事件。右图三层阴影嵌套，不能把三个覆盖率相加。

4.5 泊松过程 从独立增量到指数间隔

先把“到 t 时累计多少次”画成一条阶梯线。 $N(0)=0$ ，每次事件发生时计数增加 1，两个到达之间保持不变；同一条样本路径给出了每个时刻的计数。 $N(t)-N(s)$ 表示 $(s, t]$ 内新增的次数。只规定某个 t 的 $N(t)$ 服从泊松分布，不能确定不同时刻怎样关联，因此还不足以定义泊松过程。
[P14、P17]

齐次泊松过程的一种定义要求：从 0 开始、路径为非降且右连续的整数计数路径；互不重叠时间区间中的增量相互独立；任意长度 h 的增量服从均值 λh 的泊松分布。 $\lambda > 0$ 为恒定速率，单位为“次/单位时间”。最后一条同时表示平稳增量：分布取决于区间长度 h ，而不取决于区间从几点开始。平稳的是增量分布，累计计数 $N(t)$ 本身的均值仍随 t 增加。

$$N(t+h) - N(t) \sim \text{Poisson}(\lambda h), \quad \mathbb{E}[N(t)] = \lambda t, \quad \text{Var}(N(t)) = \lambda t \quad (4.5)$$

$t \geq 0, h \geq 0$ ； $N(t)$ 为累计计数， λ 为速率。不同区间的增量独立只适用于互不重叠区间； $N(1)$ 与 $N(2)$ 共享第一段计数，所以通常不独立。

短时间性质提供另一种直观刻画。对很短的 h ，一个事件的概率约为 λh ，两个及以上事件的概率比 h 还小一个量级，无事件概率约为 $1-\lambda h$ 。符号 $o(h)$ 表示某项除以 h 后，在 h 趋于 0 时趋于 0。它不是在任意一小时内禁止两次到达；它约束的是极短窗口内成团到达的机会。连同独立、平稳增量和适当的计数路径条件，这些局部性质导向相同的齐次泊松过程。

$$P(\Delta N = 0) = 1 - \lambda h + o(h), \quad P(\Delta N = 1) = \lambda h + o(h), \quad P(\Delta N \geq 2) = o(h) \quad (4.6)$$

$\Delta N = N(t+h) - N(t)$ 。 h 要使用与 λ 匹配的时间单位； $o(h)$ 不是固定数，而是随 h 变化的余项。由精确泊松概率 $\exp(-\lambda h)$ 及 $\lambda h \exp(-\lambda h)$ 展开即可验证这些关系。

现在推导等待时间。令 T_1 为首次到达时刻。“等到 t 仍未到达”与“ $(0, t]$ 内计数为 0”是同一个事件，因此生存概率为 $\exp(-\lambda t)$ 。求导得到指数密度 $\lambda \exp(-\lambda t)$ 。利用独立增量，在一个固定时刻之后的未来计数与过去独立，且服从相同速率；从 0 开始的连续到达间隔可表为独立同分布指数变量。这把离散的计数模型与连续的间隔模型连接起来。

$$P(T_1 > t) = P(N(t) = 0) = e^{-\lambda t}, \quad T_r = \sum_{j=1}^r W_j \quad (4.7)$$

W_j 为从起点 0 开始的第 j 个到达间隔，彼此独立，均服从速率 λ 、尺度 $1/\lambda$ 的指数分布； T_r 为第 r 次到达的累计时刻。 T_r 服从形状 r 、尺度 $1/\lambda$ 的伽马分布，后面的伽马分布一节会进一步计算。

反过来，用独立指数间隔逐次相加，也能构造同一个齐次泊松过程。模拟时常用这条路径：先抽等待间隔，再累加成到达时刻，最后用阈值计数。另一个办法是直接对互不重叠的一小时窗抽取独立泊松增量，再累加；这给出了网格上的过程。后者没有记录每个小时内部精确事件时刻，不能把一小时计数误当成一个间隔。

程序 `e37_poisson_process.py` 本例全部数据由模型模拟。速率为 3 次/小时，观察 24 小时，将时间分为 24 个互不重叠的一小时窗口，固定种子 2026，重复 2000 条独立路径。对每条路径，24 个增量服从独立泊松分布(3)，按时间轴累计得到 $N(t)$ ；理论 $E[N(t)] = \text{Var}(N(t)) = 3t$ ，24 小时终点的理论均值与方差均为 72。程序随后沿用同一发生器继续抽样，用尺度 $1/3$ 小时的指数间隔构造另一条连续时间路径，用于展示计数与等待的对应。读图时先区分“一条路径上

的 24 段时间”与“同一时刻 2000 条路径的分布”。相邻累计计数共享过去，不应据此要求零相关。本次首窗均值 3.045、样本方差约 2.937444，前两窗经验协方差约-0.032201；24 小时终点均值 71.459、样本方差约 71.772205。另生成的间隔法路径有 60 次到达。这些实现值与理论 3、0、72 不逐位相同，属于有限模拟波动，不能把另一路径事件数强制改成理论均值。

编程思路 以 $\lambda=3$ 次/小时在 24 个一小时窗内生成 2000 条独立泊松增量路径，累计得到计数过程。检查各窗均值、样本方差和相邻窗经验协方差，再以均值 1/3 小时的指数间隔生成另一条独立构造的路径。

代码对应

$\Delta N_j \sim \text{Poisson}(\lambda \Delta t)$ ，各不交叠窗相互独立 $\rightarrow \text{increments} = \text{rng.poisson}(\text{lam} * \text{dt}, \text{size}=(\text{repetitions}, \text{windows}))$ ：形状 (2000,24) 的每行是一条路径，列对应相邻的一小时窗。

$N(t_j) = \sum_{k \leq j} \Delta N_k$ ，且 $N(0)=0 \rightarrow \text{paths} = \text{np.column_stack}([\text{np.zeros}(\text{repetitions}), \text{increments.cumsum}(\text{axis}=1)])$ ：沿 axis=1 累计，再补上零起点；跨路径汇总使用 axis=0。

独立增量理论 $\text{Cov}(\Delta N_1, \Delta N_2)=0 \rightarrow \text{adjacent_cov} = \text{np.cov}(\text{increments[:, 0]}, \text{increments[:, 1]}, \text{ddof}=1)[0, 1]$ ：经验协方差以 2000 对观测计算，会因蒙特卡洛波动略偏离零。

第 t 小时累计数等于到达时刻 $\leq t$ 的事件数 $\rightarrow \text{interval_path} = \text{np.searchsorted}(\text{arrival_times}, \text{times}, \text{side}=\text{"right"})$ ：searchsorted(...,side='right') 把递增到达时刻映射到整数检查点的计数。

易错提示 指数间隔法与独立泊松增量法描述同一种过程分布，却不会在未耦合抽样时逐路径相同。24 小时终点均值和方差理论都为 72，有限 2000 路径的经验量无需恰好等于 72 或协方差恰好为零。

正文代码 e37_poisson_process.py

```
# 教材位置：第 4 章，第 4.5 节；图 4-3。
# 泊松过程的独立增量与等价间隔法，模拟客服呼入的理想模型。
```



```
# 前提是恒定呼入率、独立增量；真实数据的时段变化需另行检验。
# 时间按小时计，lam=3 次/小时，dt=1 小时，每条路径覆盖 24 小时。
# increments 形状 (2000,24)：行是独立路径，列是不重叠的时间窗。
# paths 形状 (2000,25)，包含第 0 时刻；均值、方差数组形状 (24,)。
# 输出理论/模拟矩、相邻增量经验协方差及另一条间隔法路径的事件数。
# 固定种子 2026；执行 python e37_poisson_process.py 保存中文四面板图。

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

rng = np.random.default_rng(2026)
lam, dt, windows, repetitions = 3.0, 1.0, 24, 2000
#  $\Delta N_j \sim \text{Poisson}(lam * dt)$ ，不同窗及不同路径的抽样相互独立。
increments = rng.poisson(lam * dt, size=(repetitions, windows))
# 沿 axis=1 累积各行的时间窗，最左补一个  $N(0)=0$ 。
paths = np.column_stack([np.zeros(repetitions), increments.cumsum(axis=1)])
times = np.arange(windows + 1) * dt
# 沿 axis=0 跨 2000 条独立路径求矩，保留 24 个时间窗。
means = increments.mean(axis=0)
variances = increments.var(axis=0, ddof=1)
adjacent_cov = np.cov(increments[:, 0], increments[:, 1], ddof=1)[0, 1]
# 理论独立增量的协方差为 0；有限模拟的经验协方差通常不恰为 0。
# 不对经验协方差断言等于 0，也不从一次接近 0 推断真实独立性。
assert paths.shape == (repetitions, windows + 1)
assert np.array_equal(paths[:, 1:] - paths[:, :-1], increments)
assert np.all(np.diff(paths, axis=1) >= 0)
print("每窗理论均值、方差：", lam * dt, lam * dt)
print("首窗模拟均值、样本方差：", means[0], variances[0])
print("前两窗增量的经验协方差（理论为 0）：", adjacent_cov)
print("24 小时计数的理论均值、方差：", lam * windows, lam * windows)
print("24 小时计数的模拟均值、样本方差：",
      paths[:, -1].mean(), paths[:, -1].var(ddof=1))

# 同一过程还可用独立 Exp(lam) 间隔生成；尺度参数是 1/lam 小时。
# 逐次累计直至首次越过 24 小时，避免预设个数不足而漏掉事件。
arrival_times = []
clock = 0.0
while True:
    clock += rng.exponential(scale=1 / lam)
    if clock > windows * dt:
        break
    arrival_times.append(clock)
arrival_times = np.asarray(arrival_times)
interval_path = np.searchsorted(arrival_times, times, side="right")
print("间隔法另一条路径的事件总数：", arrival_times.size)
# 两种构造同分布，但使用不同随机抽样，不要求逐点相同。
assert np.all(np.diff(arrival_times) > 0)
assert interval_path[-1] == arrival_times.size

configure_chinese_plot()
fig, axes = plt.subplots(2, 2, figsize=(6.3, 5.1), constrained_layout=True)
for index in range(3):
    axes[0, 0].step(times, paths[index], where="post", label=f"路径 {index + 1}")
    axes[0, 0].plot(times, lam * times, "k--", label="理论均值 3t")
    axes[0, 0].set(title="独立增量构造", xlabel="时刻 (小时)", ylabel="累计呼入次数")
    axes[0, 0].legend(fontsize=7)
```



```

window_index = np.arange(1, windows + 1)
axes[0, 1].plot(window_index, means, label="逐窗样本均值")
axes[0, 1].plot(window_index, variances, label="逐窗样本方差")
axes[0, 1].axhline(lam * dt, color="gray", ls="--", label="理论值 3")
axes[0, 1].set(title="跨路径估计每窗矩", xlabel="小时窗序号", ylabel="计数矩")
axes[0, 1].legend(fontsize=7)
k = np.arange(12)
frequency = np.bincount(increments[:, 0], minlength=12)[:12] / repetitions
axes[1, 0].bar(k, frequency, alpha=0.5, label="首窗模拟频率")
axes[1, 0].plot(k, stats.poisson.pmf(k, lam * dt), "o-", label="泊松理论概率")
axes[1, 0].set(title="一小时窗的计数分布", xlabel="呼入次数", ylabel="概率质量")
axes[1, 0].legend(fontsize=7)
event_grid = np.r_[0.0, arrival_times, windows * dt]
event_counts = np.r_[0, np.arange(1, arrival_times.size + 1), arrival_times.size]
axes[1, 1].step(event_grid, event_counts, where="post", label="指数间隔法")
axes[1, 1].plot(times, lam * times, "k--", label="理论均值 3t")
axes[1, 1].set(title="另一条连续时间路径", xlabel="时刻 (小时)", ylabel="累计呼入次数")
axes[1, 1].legend(fontsize=7)
for ax in axes.flat:
    ax.grid(alpha=0.2)
fig.savefig("poisson_process.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

规定环境下的实际运行输出

每窗理论均值、方差： 3.0 3.0

首窗模拟均值、样本方差： 3.045 2.937443721860898

前两窗增量的经验协方差（理论为 0）： -0.03220110055027511

24 小时计数的理论均值、方差： 72.0 72.0

24 小时计数的模拟均值、样本方差： 71.459 71.77220510255128

间隔法另一条路径的事件总数： 60

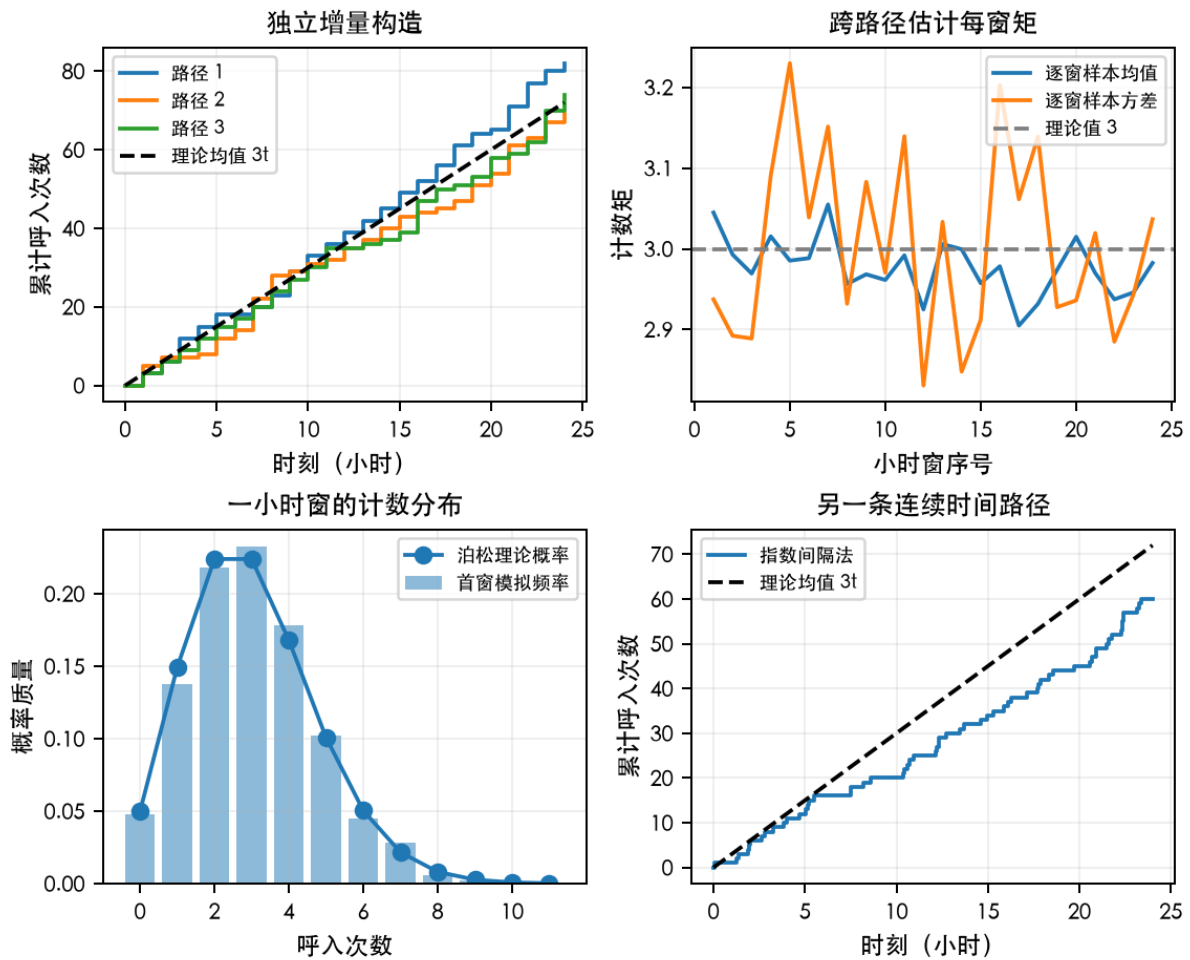


图 4-3 齐次泊松过程的四面板模拟：速率为每小时 3 次，24 个一小时观察窗，2000 条独立路径。左上为增量法的 3 条累计计数路径及 $3t$ ；右上为跨 2000 条路径估计的各窗均值、样本方差及理论值 3；左下为首窗频率与泊松分布(3)质量；右下为指数间隔法另生成的一条 24 小时路径。全部参数均为教学设定。

读图指南 上左仅按小时网格记录累计数，某次跃幅可大于 1，不能把网格跳跃解读为多次事件在同一瞬间发生；下右才按每次到达时刻画路径。上右比较的是每小时增量的跨路径散布，理论水平为 3，不能与累计方差 $3t$ 混淆。左下只显示计数 0 至 11，泊松支持集仍无上界。经验矩与零协方差均有有限模拟波动，图中未绘制终点计数分布。

例题 教学假设：等待的是到达，还是完成服务？

假设客服来电按每小时 3 次的齐次泊松过程到达，那么下一次呼入间隔均值为 $1/3$ 小时，即 20 分钟。这里推导的是“下一位顾客何时到达”，并非“当前顾客多久获得响应”。响应时间还取决于坐席数量、排队长度与服务时长，需要排队模型。把响应时间也设为指数分布可以作为另一个明确的简化假设，却不能由来电过程直接推出。

元件寿命采用指数模型，意味着尚未失效元件的瞬时失效率恒定，适合理想无老化阶段；磨损、早期缺陷、维修策略都可能破坏这一条件。门店客流受营业时间与促销影响时，平稳增量也可能失效。模型条件应从业务机制和记录中判断，而不是因为变量名称含“等待”或“故障”就自动成立。

4.6 指数分布 恒定速率下的等待

固定窗口的到达次数是离散计数，等到下一次到达的时间是连续等待。若事件构成速率 $\lambda > 0$ 的齐次泊松过程，则任意长度 t 内的计数 $N(t) \sim \text{泊松分布}(\lambda t)$ ，不重叠时间段的增量相互独立。令 T 为下一次到达前等待时间， $T > t$ 恰好等价于 $N(t) = 0$ ，因而 $P(T > t) = \exp(-\lambda t)$ ，得到指数分布。

速率 λ 的单位是“次数/时间”，尺度 $\theta = 1/\lambda$ 的单位是“时间”。同一指数分布可按速率或尺度参数化：均值为 $1/\lambda = \theta$ ，方差为 $1/\lambda^2 = \theta^2$ 。软件通常使用 $\text{scale} = \theta$ ，若把速率直接传给 scale ，除非恰巧 $\lambda = 1$ ，否则会得到错误的等待长度。使用带单位的参数能帮助立即发现这个错误。

$$f_T(t) = \lambda e^{-\lambda t} \quad (t \geq 0), \quad F_T(t) = 1 - e^{-\lambda t}, \quad S_T(t) = P(T > t) = e^{-\lambda t}, \quad \theta = 1/\lambda \quad (4.8)$$

T 为等待时间， t 为给定时长， $t < 0$ 时累积分布函数为 0。 S 称为生存函数或上尾函数，与累积分布函数满足 $S = 1 - F$ 。 λ 为速率， θ 为尺度；同一个模型只能在两种参数化之间换算，不能把两个数都当作独立可选的参数。

指数分布的无记忆性来自 $P(T > s+t | T > s) = \exp(-\lambda(s+t)) / \exp(-\lambda s) = \exp(-\lambda t)$ 。已经等待 s 时间仍未到达，不改变剩余等待的分布。等价地，其风险率 $h(t) = f(t)/S(t) = \lambda$ 为常数，可理解为“在尚未发生的条件下，每单位短时间发生的瞬时强度”。这适合无老化的理想机制；若零件随使用时间磨损，或人流具有高峰低谷，恒定风险与速率就需要重新评估。

连续等待与离散等待的“无记忆”十分相似，但事件边界不同。几何变量数试验次数，指数变量量时间；前者尾概率为 $(1-p)^m$ ，后者为 $\exp(-\lambda t)$ 。把概率 p 当作速率 λ 通常不正确。只有在很短时间 Δt 内，泊松事件发生至少一次的概率 $1 - \exp(-\lambda \Delta t)$ 才近似 $\lambda \Delta t$ 。

例题 手算 4-3: 速率 0.5 的等待模型

$\lambda=0.5$ 次/分钟, 尺度 $\theta=2$ 分钟。等候超过 3 分钟概率为 $\exp(-1.5)\approx 0.22313$ 。已等 2 分钟还没到达, 再等超过 1 分钟概率为 $P(T>3|T>2)=\exp(-1.5)/\exp(-1)=\exp(-0.5)\approx 0.60653$ 。

注意后一问题的答案不是 $P(T>3)$, 因为它以“已经等到 2 分钟”为条件。指数模型的中位数满足 $1-\exp(-\lambda m)=1/2$, 所以 $m=(\ln 2)/\lambda\approx 1.38629$ 分钟, 小于均值 2 分钟, 反映右长尾。

程序 `e15_exponential.py` 程序固定速率 0.5、尺度 2, 用种子 15 生成 5000 个独立间隔 `gaps`。`cumsum` 把间隔变为到达时刻; 首次到达时间等于首个间隔, 第 j 次到达时间是前 j 个间隔的和。图左只展示前 8 次到达与前 4 个间隔, 图右使用全部 5000 个原间隔。以 $s=2$ 、 $t=1$ 核验条件生存概率, 理论间隔均值为 2。

编程思路 以速率 $\lambda=0.5$ 生成指数间隔, 累计求和得到到达时刻。把间隔分布图与前若干到达事件的时间轴并列, 并用生存函数比值验证无记忆性。

代码对应

等待时间服从 $\text{Exp}(\lambda=0.5) \rightarrow \text{rv} = \text{stats.expon}(\text{scale}=1 / \text{rate})$: SciPy 用 $\text{scale}=1/\text{rate}=2$ 表达均值, 速率和尺度互为倒数。

第 j 次到达时刻 $S_j=W_1+\dots+W_j \rightarrow \text{arrival_times} = \text{np.cumsum}(\text{gaps})$: `cumsum` 把独立间隔转换为递增到达时刻; 到达时刻彼此不独立。

$P(W>s+t|W>s)=S(s+t)/S(s) \rightarrow \text{conditional} = \text{rv.sf}(s + t) / \text{rv.sf}(s)$: 这里 $s=2$ 、 $t=1$, 结果应等于 $S(1)$, 与已等待时长 s 无关。

易错提示 指数分布的无记忆性属于等待时间; 累计到达时刻不再是指数分布。本例展示单条事件时间轴, 不能仅凭一条路径断言经验速率必然等于理论速率。

正文代码 `e15_exponential.py`

```
# 教材位置: 第 4 章, 第 4.6 节; 图 4-4。
# 单位、变量与形状: gaps、arrival_times 为 (5000,); 间隔和时刻按时间计, rate 为每时间单
# 位的事件数。
# 学习目标: 从指数分布生成独立事件间隔, 累加得到到达时刻, 并验证无记忆性。
# 核心数学关系: 间隔 T 的速率为 lambda 时, E[T]=1/lambda,
# P(T>t)=exp(-lambda*t), 且 P(T>s+t | T>s)=P(T>t)。
```

```
# 数据与输出: 模拟 5000 个速率为 0.5 的间隔, 打印前五个到达时刻、
# 理论/样本平均间隔和条件生存概率; 数值自检查性质, 并保存 e15_exponential.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e15_exponential.py。

# NumPy 负责随机模拟和累加; SciPy 用于计算理论分布的性质。
import numpy as np
from scipy import stats

# 创建独立随机数生成器; 固定种子便于在相同软件环境下复现教学结果。
rng = np.random.default_rng(15)
# rate 的单位是“每单位时间的平均事件数”; 这里平均每单位时间 0.5 次。
rate = 0.5
# SciPy 和 NumPy 的指数分布参数都是 scale, 即平均等待时间 1/rate=2。
# rate 与 scale 互为倒数; 直接把 rate 传给 scale 会得到另一个分布。
rv = stats.expon(scale=1 / rate)
# size=5000 返回形状 (5000,) 的一维数组, 每个元素是一段独立等待时间。
gaps = rng.exponential(scale=1 / rate, size=5000)
# 从时刻 0 开始累计间隔: 第 i 个元素是第 i+1 次事件发生的绝对时刻。
# 区分“间隔 gaps”与“时刻 arrival_times”, 才能正确解释后续统计量。
arrival_times = np.cumsum(gaps)
# [:5] 切片取前五个元素; 样本到达时刻受随机间隔影响, 不会等距。
print("前五个到达时刻: ", arrival_times[:5])
# rv.mean() 是总体理论均值 2; gaps.mean() 是有限次模拟的样本均值,
# 两者接近但无需完全相等, 差异来自蒙特卡洛抽样误差。
print("理论平均间隔、样本平均间隔: ", rv.mean(), gaps.mean())
# 已等待 s 个时间单位, 考察还需等待超过 t 的条件概率。
s, t = 2.0, 1.0
# sf(u)=P(T>u) 是生存函数。因为 {T>s+t} 包含于 {T>s},
# 条件概率等于两者概率之比; sf 也避免了尾部用 1-cdf 的相减精度损失。
conditional = rv.sf(s + t) / rv.sf(s)
print("条件生存概率: ", conditional)
# 无记忆性说明“已等待多久”不会改变剩余等待时间的分布。
# 这是指数分布的性质, 并不适用于所有连续等待时间模型。
assert np.isclose(conditional, rv.sf(t))
# diff 计算相邻到达时刻之差, 长度从 5000 减为 4999。
# 指数间隔理论上几乎必然为正; 本断言检查本次样本的到达时刻严格递增。
assert np.all(np.diff(arrival_times) > 0)

# 教材图 4-4 使用本例已经抽出的 gaps 和 arrival_times, 绝不重新设种子或额外抽样。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图只显示 5000 次到达中的前 8 次; 在最前面补 0 仅用于画第一个等待间隔。
shown_count = 8
shown_times = np.r_[0.0, arrival_times[:shown_count]]
axes[0].hlines(1, 0, shown_times[-1] + 0.5, color='0.5', linewidth=1)
axes[0].vlines(shown_times[1:], 0.9, 1.1, color='C0', linewidth=1.4)
# 前四个带端点的水平线段展示“间隔”与“累计时刻”的区别, 端点完全来自已有数组。
for gap_index in range(4):
    # 很短的相邻间隔可能挤在一起, 错开线段和文字的高度即可避免标签重叠。
    arrow_height = 0.75 - 0.17 * (gap_index % 3)
    # 采用线段和端点短线, 极短间隔也不会因双箭头尺寸超过间隔而交叉。
    axes[0].hlines(arrow_height, shown_times[gap_index], shown_times[gap_index + 1],
                  color=f'C{gap_index % 3}', linewidth=1.3)
```

```

axes[0].vlines(shown_times[gap_index:gap_index + 2],
               arrow_height - 0.025, arrow_height + 0.025,
               color=f'C{gap_index % 3}', linewidth=1)
axes[0].text((shown_times[gap_index + 1] + shown_times[gap_index]) / 2,
             arrow_height - 0.11, rf'$T_{\{\{gap\_index + 1\}\}}$', ha='center',
             fontsize=7.8)
axes[0].set(title=f'{gaps.size:,} 次到达中的前 {shown_count} 次',
            xlabel='到达时刻', yticks=[], ylim=(0.15, 1.35),
            xlim=(0, shown_times[-1] + 0.5))
# 右图使用全部 5000 个原间隔, 与同一 scale=1/rate=2 的理论指数密度比较。
# bins=35 只控制显示分组; density=True 把柱形总面积归一化为 1。
axes[1].hist(gaps, bins=35, density=True, alpha=0.5, label='全部模拟间隔')
exponential_t = np.linspace(0, gaps.max(), 500)
axes[1].plot(exponential_t, rv.pdf(exponential_t), color='C1',
             label=f'指数: 速率={rate:g}, 尺度={1 / rate:g}')
axes[1].set(title='相同的等待时间模型', xlabel='间隔长度 t', ylabel='概率密度')
axes[1].legend(fontsize=7.8)
fig.savefig('e15_exponential.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

规定环境下的实际运行输出

前五个到达时刻: [4.99943026 8.04409158 8.6244241 8.70673041 11.83063819]

理论平均间隔、样本平均间隔: 2.0 2.009591879059883

条件生存概率: 0.6065306597126334

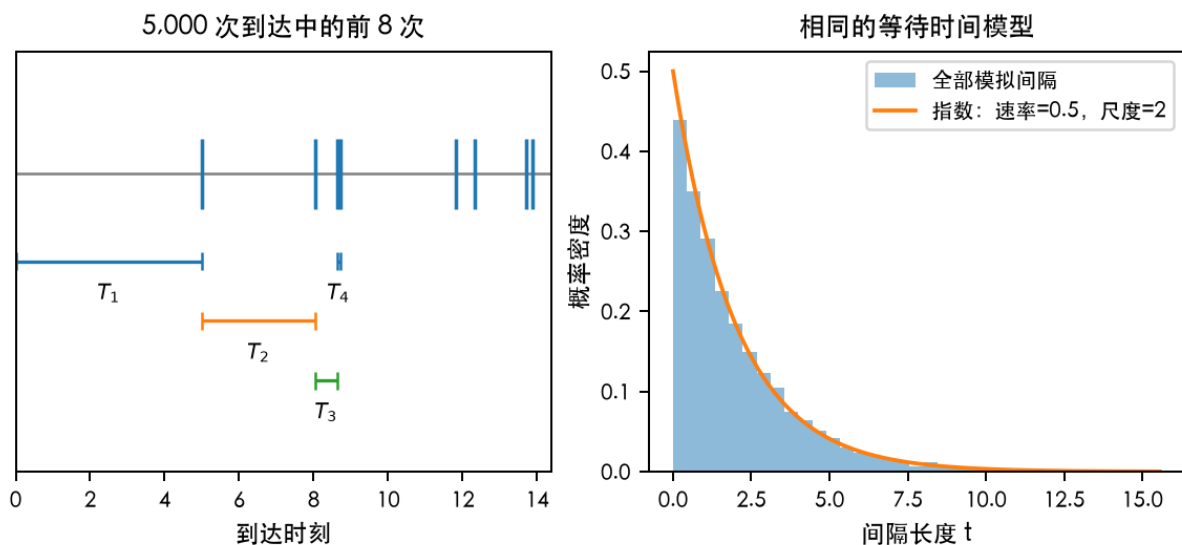


图 4-4 速率 $\lambda=0.5$ 、尺度 $\theta=2$ 的指数间隔与到达时刻, 由 `e15_exponential.py` 生成。种子=15 产生 5000 个 `gaps`, 左图只显示其中前 8 次到达及前 4 个间隔, 初始时刻 0 仅为绘图补入; 右图用全部 5000 个原间隔的直方图与同一尺度的理论密度比较, 没有额外抽样。

读图指南 左图端点标记为累计到达时刻, 彩色水平短段为相邻到达之间的间隔; 右图使用全部 5000 个间隔, 而非只用左图 8 次到达。第 4 个间隔很短仍是一段正等待。

4.7 伽马分布 把多个等待相加

若不是等下一次到达，而是等第 r 次到达，总等待时间 T_r 就是 r 个独立同尺度指数间隔之和。整数 r 时，这给出了伽马分布最直观的生成故事。第一次等待的指数密度在 0 附近最高；等多次到达往往需要一段累积时间，密度峰值可移到正数处。 r 越大，总等待的相对波动越小，因为均值与 r 成正比、标准差与 $\sqrt{r}$ 成正比。

伽马分布可推广到任意形状参数 $\alpha > 0$ ，尺度 $\theta > 0$ 。记伽马分布 (α, θ) ，其密度包含伽马函数 $\Gamma(\alpha) = \int_0^\infty u^{\alpha-1} \exp(-u) du$ 。对正整数 r ， $\Gamma(r) = (r-1)!$ ；分部积分给 $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$ 。这个递推使矩计算很简洁：把密度中 t 的幂提高一次或两次，积分即可换成相邻的伽马函数。

$$f_T(t) = \frac{t^{\alpha-1} e^{-t/\theta}}{\Gamma(\alpha) \theta^\alpha} \quad (t > 0), \quad E[T] = \alpha\theta, \quad \text{Var}(T) = \alpha\theta^2 \quad (4.9)$$

α 为无量纲形状， θ 为时间尺度。以 $u=t/\theta$ 换元可验证总积分为 1； $E[T] = \theta\Gamma(\alpha+1)/\Gamma(\alpha) = \alpha\theta$ ； $E[T^2] = \theta^2\Gamma(\alpha+2)/\Gamma(\alpha) = \alpha(\alpha+1)\theta^2$ ，减均值平方得方差。按速率参数化时使用 $\lambda=1/\theta$ 。

在泊松过程的整数形状情形，事件 $\{T_r \leq t\}$ 与 $\{N(t) \geq r\}$ 完全相同：第 r 次到达已经发生，等价于到 t 时至少有 r 次到达。因此伽马累积分布函数可由泊松上尾计算。它把连续等待与离散计数连接起来，也给软件计算提供了两条互相独立的核验路径。一般非整数 α 的伽马分布仍然合法，但“第 α 次事件”的直观故事不再具有字面含义。

$$P(T_r \leq t) = P(N(t) \geq r) = 1 - \sum_{k=0}^{r-1} e^{-t/\theta} \frac{(t/\theta)^k}{k!} \quad (4.10)$$

r 为正整数， T_r 为第 r 次到达时刻， $N(t)$ 为截至 t 的次数，过程速率为 $1/\theta$ 。右边先减去计数 0 至 $r-1$ 的概率。离散上尾接口 $\text{sf}(r-1)$ 恰好对应“至少 r ”，这是边界减 1 的原因。

程序 e16_gamma.py 程序取 $r=4$ 、 $\theta=0.5$ ，速率为 2，与上一案例速率 0.5 不同。种子 21 生成形状(30000,4)的指数间隔，每行求和得到 30000 个总等待。理论均值 $r\theta=2$ ，方差 $r\theta^2=1$ 。在 $t=0.5$ 、1、2、3 处，伽马分布(4,0.5)的累积分布函数与泊松分布(t/θ)的 sf(3)核对；两图复用同一组等待，未额外生成第二组数据。

编程思路 把四个独立、尺度为 0.5 的指数等待时间逐组相加，得到 Gamma(4,0.5)。另通过泊松计数至少四次的概率计算同一等待时间 CDF，展示计数与等待两种视角的等价。

代码对应

$T_4=W_1+W_2+W_3+W_4 \rightarrow$ waits = rng.exponential(scale=theta, size=(30000, r)).sum(axis=1): 形状 (30000,r) 的每行是一组四个间隔，sum(axis=1) 得到三万个总等待时间。

$T_4 \sim \text{Gamma}(\text{shape}=4, \text{scale}=0.5) \rightarrow$ rv = stats.gamma(a=r, scale=theta): 均值 $r\theta=2$ 、方差 $r\theta^2=1$ ， θ 是尺度而非速率。

$P(T_4 \leq t) = P[N(t) \geq 4] \rightarrow$ count_formula = stats.poisson.sf(r - 1, mu=t / theta): poisson.sf(r-1,mu=t/theta) 对应大于 3，即至少 4 次计数，速率是 $1/\theta$ 。

易错提示 Gamma 形状参数为正数即可定义，本例整数形状才具有四个指数间隔求和的解释。sf(4) 会漏掉恰好四次，正确输入是 $r-1$ 。

正文代码 e16_gamma.py

```
# 教材位置：第 4 章，第 4.7 节；图 4-5。
# 单位、变量与形状：原间隔数组为 (30000,4)，求和后 waits 为 (30000,); theta 按时间计。
# 学习目标：理解整数形状参数的 伽马分布为何是多个指数等待时间之和，
# 并用 Poisson 计数从另一角度计算第 r 次事件发生的概率。
# 核心数学关系：若 r 个独立间隔均为 Exponential(scale=theta)，
# 则总等待时间 T_r ~ 伽马分布(shape=r, scale=theta)，
# E[T_r]=r*theta、Var(T_r)=r*theta^2，P(T_r<=t)=P(N(t)>=r)。
# 数据与输出：模拟 30000 次“等到第 4 次事件”的过程，比较理论/样本矩，
# 并打印四个时刻前已经发生第 4 次事件的概率，保存 e16_gamma.png。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后，在终端运行 python e16_gamma.py。

import numpy as np
from scipy import stats

# 固定种子使当前环境中的样本可复现；每行模拟过程使用不同随机抽样。
rng = np.random.default_rng(21)
# r 是所等待的事件次数，也是 伽马分布 的形状参数；theta 是每段间隔的
```



```
# 平均时间（尺度），对应事件速率 lambda=1/theta=2，不能把尺度当作速率。
r, theta = 4, 0.5
# 先产生形状 (30000,4) 的数组：每行是一轮过程中的四个独立间隔。
# sum(axis=1) 沿列求和，把每行变成总等待时间，结果形状为 (30000,)。
# 形状参数为正整数的 伽马分布也称 Erlang 分布。
waits = rng.exponential(scale=theta, size=(30000, r)).sum(axis=1)
# SciPy 用 a 表示形状参数，用 scale 表示尺度；此处默认 loc=0。
rv = stats.gamma(a=r, scale=theta)
# 在四个非负时刻同时计算概率，数组形状为 (4,)。
t = np.array([0.5, 1.0, 2.0, 3.0])
# 对速率 1/theta 的齐次 Poisson 过程，N(t) 的均值为 t/theta。
# Poisson.sf(r-1)=P(N>r-1)=P(N>=r)，正好对应“第 r 次事件已到达”。
# 离散分布中必须注意这个 r-1；sf(r) 会漏掉 N=r 的情形。
count_formula = stats.poisson.sf(r - 1, mu=t / theta)
# allclose 逐元素比较后汇总为一个布尔值，验证四个时刻的等待/计数关系。
assert np.allclose(rv.cdf(t), count_formula)
# 独立间隔之和的均值、方差分别相加，得到 r*theta 和 r*theta**2。
assert np.allclose([rv.mean(), rv.var()], [r * theta, r * theta ** 2])
# 理论矩由分布参数确定，不受当前模拟样本影响。
print("理论均值、方差：", rv.mean(), rv.var())
# ddof=1 使用分母 30000-1，计算总体方差的无偏样本估计；
# 样本均值和方差存在随机误差，因此不要求它们与理论值完全相等。
print("样本均值、样本方差：", waits.mean(), waits.var(ddof=1))
# cdf(t) 对数组逐元素计算 P(T_r<=t)，输出顺序与 t 中时刻顺序一致。
print("第 r 次事件在 t 前到达的概率：", rv.cdf(t))

# 教材图 4-5 采用本例 r=4、theta=0.5，与前例 rate=0.5 的指数模型属于不同参数情景。
# 本图复用已有 30000 个 waits 及原 t 数组；新增理论曲线计算不进行任何随机抽样。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图把 r 段指数间隔之和的样本密度与对应伽马分布理论密度比较。
wait_grid = np.linspace(0, max(waits.max(), rv.ppf(0.999)), 600)
axes[0].hist(waits, bins=45, density=True, alpha=0.5, label=f'{waits.size:,} 组间隔和')
axes[0].plot(wait_grid, rv.pdf(wait_grid), color='C1',
              label=f'伽马：形状={r}，尺度={theta:g}')
axes[0].set(title='指数间隔之和', xlabel='总等待时间', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图两条理论曲线重合：第 r 次到达已发生，等价于时刻 t 前计数至少 r 次。
# count_curve 中的 Poisson 参数为时间/theta；SF 传 r-1 才能包含计数恰好等于 r 的情况。
cdf_grid = np.linspace(0, rv.ppf(0.999), 400)
count_curve = stats.poisson.sf(r - 1, mu=cdf_grid / theta)
axes[1].plot(cdf_grid, rv.cdf(cdf_grid), linewidth=2.5, alpha=0.6,
             label='伽马累积分布函数')
axes[1].plot(cdf_grid, count_curve, '--', color='C1', linewidth=1.2,
             label=f'泊松尾概率：次数 ≥ {r}')
# 四个实心点就是终端打印的 t 数组所对应的 count_formula，连接图形和数值核验。
axes[1].scatter(t, count_formula, color='C2', s=20, zorder=3, label='核对的时刻 t')
axes[1].set(title=r'$P(T_r \leq t) = P(N(t) \geq r)$',
            xlabel='时间 t', ylabel='累积概率', ylim=(-0.03, 1.05))
axes[1].legend(fontsize=7.8)
fig.savefig('e16_gamma.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

规定环境下的实际运行输出

理论均值、方差： 2.0 1.0

样本均值、样本方差： 2.016269153597875 1.0005834025211036

第 r 次事件在 t 前到达的概率： [0.01898816 0.14287654 0.56652988 0.84879612]

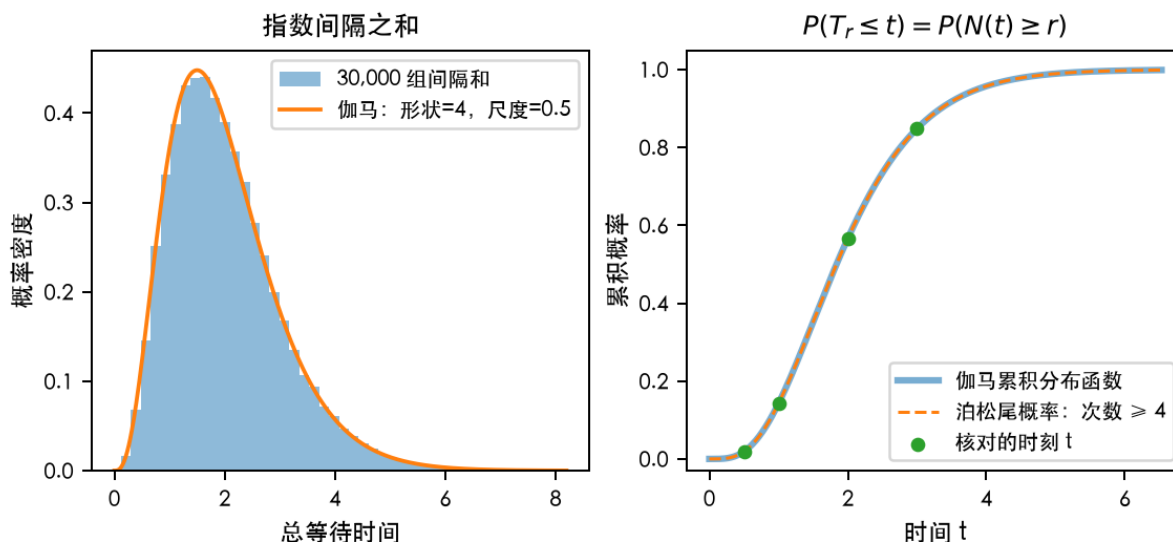


图 4-5 $r=4$ 、 $\theta=0.5$ 的伽马等待与泊松计数恒等式，由 `e16_gamma.py` 生成。左图复用种子=21 得到的 30000 个指数间隔行和 `waits`，与伽马分布(4,0.5)密度比较；右图伽马分布的累积分布函数与均值 t/θ 的泊松分布计数尾概率重合，四个点对应原 t 数组。这里事件速率 $\lambda=2$ ，与指数间隔与到达时刻图的 $\lambda=0.5$ 属于不同参数情景。

读图指南 左图每个等待是 4 个间隔的和，不能误读成单个间隔；右图两条曲线重合检验的是同一事件的计数与等待两种写法。此例尺度 0.5 与指数例尺度 2 不同。

4.8 贝塔分布与有界比例的形状

比例或成功概率位于 0 与 1 之间，正态和伽马的支持集都不能直接保证这一约束。贝塔分布用两个正形状参数 α 、 β 描述(0,1)上的密度。它可以对称、向左或向右偏斜，也可以在两端附近较高；当 $\alpha=\beta=1$ 时恰为均匀分布(0,1)。支持集符合参数含义是选择它的重要理由，而不是它总能拟合所有比例数据。

贝塔分布的归一化常数为 $B(\alpha, \beta) = \int_0^1 \theta^{\alpha-1} (1-\theta)^{\beta-1} d\theta$ ，称为贝塔函数。它与伽马函数满足 $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$ 。两个参数都为正时积分有限；当某参数小于 1，密度可能在对应端点发散，但总面积依然有限。这再次表明密度的高度不受 1 限制，也不等于端点的概率。

$$\begin{aligned}
 f(\theta) &= \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}, \quad 0 < \theta < 1 \\
 \mathbb{E}[\theta] &= \frac{\alpha}{\alpha + \beta} \\
 \text{Var}(\theta) &= \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}
 \end{aligned} \tag{4.11}$$

Θ 为取值在 0 至 1 的随机变量， θ 为其候选取值， α 、 β 均大于 0。此处 θ 表示比例变量，不是上一节伽马尺度；符号含义取决于当前模型。均值可由 $B(\alpha+1, \beta)/B(\alpha, \beta)$ 求出，二阶矩为 $\alpha(\alpha+1)/[(\alpha+\beta)(\alpha+\beta+1)]$ 。

令 $m=\alpha/(\alpha+\beta)$ 、 $\kappa=\alpha+\beta$ ，则方差为 $m(1-m)/(\kappa+1)$ 。固定均值 m 时， κ 越大，分布越集中。这种“中心加集中程度”的解释比单独记忆两个形状参数更容易使用。在贝叶斯估计中，贝塔分布常用于未知伯努利成功率的先验；后验更新只需把成功、失败次数分别加到两个形状参数，下一章将完整推导。

还应区分“某次试验结果”和“试验成功概率”。伯努利结果取 0 或 1，贝塔分布变量在 0 与 1 之间连续取值。将一次成功观测写成贝塔分布密度不合适；把未知成功率当作确定参数或随机参数，是频率学与贝叶斯建模中需要明确说明的选择。实际比例若来自 k/n ，分母 n 携带了精度信息，仅对比例数值拟合连续分布可能丢掉这一信息。

4.9 本章要点 常见误区与延伸阅读

核心要点：连续密度是单位长度上的概率密集程度，概率来自面积，密度可大于 1。累积分布函数与分位数函数方向相反；标准化改变位置和尺度，却不自动把任意数据变成正态。变量变换须保留尺度修正因子。

核心要点：齐次泊松过程规定整条时间记录，独立增量针对互不重叠区间，平稳增量针对相同长度窗口。计数 $N(t)$ 服从泊松分布 (λt) ，间隔为独立指数变量，第 r 次到达为整数形状伽马变量；累计计数彼此共享过去，通常不独立。

核心要点：速率 λ 与尺度 θ 互为倒数，单位相反。指数无记忆对应恒定风险，不等于所有现实等待都无记忆。贝塔变量表示 0 至 1 内的比例或参数，伯努利变量表示一次 0 或 1 结果；两者不能只因都与成功有关就混用。

常见误区：把密度高度当点概率，把正态 scale 填成方差，把指数 scale 填成速率，把相同均值当作相同分布，把呼入间隔当成响应时间，或把元件无老化假设当作事实。伽马函数与贝塔函数用于归一化，二者都不是累积分布函数。

延伸阅读：哈佛《概率论导论》第 5 章的连续变量、第 8 章的变换；统计 110 第 16 讲指数与无记忆、第 23—24 讲读贝塔、伽马和泊松过程，官方入口 <https://stat110.hsites.harvard.edu/youtube> [P14、P19]。

课程配读：麻省理工学院 18.05 第 5—6 课阅读密度、连续期望与方差，第 15 课读贝塔形状[P02、P12]；6.041SC 第 14—15 讲从过程角度补充独立增量与指数间隔，官方目录 <https://ocw.mit.edu/courses/6-041sc-probabilistic-systems-analysis-and-applied-probability-fall-2013/resources/lecture-videos/> [P17]。

4.10 练习与参考解答

练习 4-1（概念与手算，入门） X 在(2,6)均匀。求 $P(X>5)$ 、中位数、方差及 $x=4$ 的密度；再令 U 在(0,1)均匀、 $Y=U^2$ ，写出 Y 的累积分布函数与内部密度。

练习 4-2（概念与手算，入门） X 服从 $N(170,8^2)$ 。求 $P(162<X<178)$ ，说明标准化和正态性有什么关系；再比较单侧 95%与双侧中心 95%的标准正态临界值。

练习 4-3（概念与手算，进阶） 速率 3 次/小时的齐次泊松过程中，求 $\text{Cov}(N(1),N(2))$ ，并判断第一、第二小时增量是否独立。等待下一次超过半小时的概率是多少？

练习 4-4（概念与手算，进阶） T 服从伽马分布(3,2)，第二参数为尺度。求均值、方差与对应整数到达过程速率； Θ 服从贝塔分布(2,2)，求均值、方差和 1/2 处密度。

练习 4-5（编程实践，入门） 运行 e13_uniform.py，核对 loc、scale、pdf 与 cdf 的含义。如果把 scale 写成 6，支持集和 $P(3<X<5)$ 会怎样变？

练习 4-6（编程实践，入门） 运行 `e14_normal.py`，核对阈值 178 的累计概率和原尺度上尾 5% 阈值；指出 `scale` 应填什么。

练习 4-7（编程实践，进阶） 比较 `e15_exponential.py` 与 `e16_gamma.py`：解释 `cumsum` 和 `sum(axis=1)` 各在构造什么，给出可复核的理论量与事件恒等式。

练习 4-8（编程实践，进阶） 用 `e37_poisson_process.py` 核对 24 窗、2000 条路径的轴方向。说明理论增量均值方差、终点均值方差，以及为什么不要求经验协方差精确为 0。

练习 4-9（场景应用，入门） 教学设定：某无老化元件寿命为指数分布，均值 4 分钟。写速率与尺度，求寿命超过 8 分钟的概率；已工作 4 分钟未坏，再工作超过 4 分钟的条件概率是多少？

练习 4-10（场景应用，进阶） 教学设定：客服呼入为每小时 3 次的泊松过程。有读者据此声称顾客响应时间均值为 20 分钟。指出缺失机制，写出能由题设计算的正确 20 分钟含义。

练习 4-11（场景应用，进阶） 教学设定：门店上午稳定客流 6 人/小时，下午 2 人/小时，两个时段内可各用独立增量计数。上午半小时与下午半小时无客流概率各是多少？整天是否为同一恒定速率的齐次泊松过程？

练习 4-12（场景应用，拓展） 教学设定：某任务要等同速率 2 次/小时的独立到达累计 4 次才启动。求启动时间均值、标准差及 1 小时内启动的概率；说明为何“平均等 2 小时”不代表 2 小时前不能启动。

参考解答与验收依据

4-1 均匀模型答案为 1/4、4、4/3、1/4；密度 1/4 不是 $P(X=4)$ ，后者为 0。Y 在 $0 \leq y \leq 1$ 时累积概率 $\sqrt{y}$ ，内部密度 $1/(2\sqrt{y})$ ，区间外密度 0； $y < 0$ 时累积概率 0、 $y \geq 1$ 时为 1。

4-2 两边界标准化为 -1 与 1，概率 $2\Phi(1)-1 \approx 0.682689$ 。任意非退化变量都能减均值除标准差，但只有已正态的变量标准化后必为标准正态。两个临界值约为 1.64485 与 1.95996，不能混用。

4-3 $N(2)=N(1)+[N(2)-N(1)]$ ，后一增量与 $N(1)$ 独立，所以协方差为 $\text{Var}(N(1))=3$ 。互不重叠的一小时增量独立，且均服从泊松分布(3)。等待超过半小时等价于该窗口无到达，概率 $\exp(-1.5)\approx 0.223130$ 。累计计数不因独立增量而彼此独立。

4-4 T 均值 6、方差 12，若解释为第 3 次到达时刻，则速率 1/2。 Θ 均值 1/2、方差 1/20；密度为 $6\theta(1-\theta)$ ，在 1/2 处为 1.5，可超过 1 但总面积为 1。贝塔函数 $B(2,2)=1/6$ 是归一化常数，不是累积分布函数。

4-5 原 $\text{loc}=2$ 、 $\text{scale}=4$ 给 $[2,6]$ ， $\text{pdf}(4)=0.25$ 、区间概率 0.5。改成 $\text{scale}=6$ 后支持 $[2,8]$ ，密度 1/6，区间概率 $2/6=1/3$ 。验收时同时检查 $\text{rv.support}()$ 、两次 cdf 相减与密度高度，不能把 scale 当作右端点。

4-6 $z=1$ ，累计概率约 0.84134475；原尺度上尾 5% 阈值 $170+8\times 1.64485363\approx 183.15882902$ 。 scale 填标准差 8；若填方差 64，相当于另一个更分散的模型。 ± 1 、 ± 2 、 ± 3 个标准差中心覆盖约 68.268949%、95.449974%、99.730020%。

4-7 $e15$ 以尺度 2 抽 5000 个间隔， cumsum 把它们累加成同一过程的到达时刻，理论间隔均值 2； $s=2$ 、 $t=1$ 的条件生存概率 $\exp(-0.5)\approx 0.60653066$ 。 $e16$ 生成(30000,4)的尺度 0.5 间隔，每行求和得独立的第 4 次到达等待，均值 2、方差 1。对 $t=0.5$ 、1、2、3， $\text{gamma.cdf}(t,a=4,\text{scale}=.5)$ 应与 $\text{poisson.sf}(3,\mu=2t)$ 逐项接近。两个程序虽然均值有 2，都在记录不同随机对象。

4-8 increments 应为(2000,24)， $\text{axis}=1$ 沿时间累计， $\text{axis}=0$ 在同一窗口跨路径求统计量。每小时增量的理论均值方差均 3， $N(24)$ 的理论均值方差均 72。第 1 和第 2 小时增量独立，理论协方差 0；2000 条有限样本的经验值会有波动。验收还要检查累计路径非降、首时刻补 0、终点等于 24 个增量之和，以及指数构造间隔尺度为 1/3 小时。

4-9 尺度 4 分钟，速率 0.25 次/分钟； $P(T>8)=\exp(-2)\approx 0.135335$ 。条件概率 $P(T>8|T>4)=\exp(-1)\approx 0.367879$ 。这个结论来自恒定风险模型，不能直接套到明显磨损的真实元件。

4-10 20 分钟是下一次呼入的平均间隔 $1/3$ 小时。响应时间还需坐席数、服务时间分布、排队规则与当前队列；例如坐席空闲且立即接通时响应可近 0，全部忙时则可能很长，同样呼入率不能唯一决定。应把到达间隔与服务/排队等待分别定义。

4-11 两个窗口均值分别 3 与 1，无客流概率 $\exp(-3) \approx 0.049787$ 与 $\exp(-1) \approx 0.367879$ 。由于同长度窗口的分布取决于时间位置，整天不满足平稳增量的恒定速率模型。可以明确分段速率或考虑非齐次过程；不能把整天平均率用于每个时段并仍保留原假设。

4-12 启动时间为形状 4、尺度 0.5 的伽马分布，均值 2 小时、标准差 1 小时。
 $P(T_4 \leq 1) = P(N(1) \geq 4) = 1 - \exp(-2)(1 + 2 + 2 + 8/6) \approx 0.14287654$ 。2 小时是加权中心而非最短等待；任意正时长都有正概率出现 4 次到达。

第 5 章 抽样 区间估计与模型检验

5.1 本章导读 入门 进阶与拓展

本章从有限观测讨论未知参数。入门阅读先分清总体、样本、参数、统计量、置信区间与后验可信区间。进阶阅读推导似然、t 推断、贝塔—二项与正态—正态共轭，并学习拟合诊断。拓展阅读关注后验预测、先验敏感性与拟合参数后的检验校准。所有结论都应说明抽样方式和模型前提；程序输出的数值只是这份说明的一部分。

5.2 总体 样本 参数与统计量

前几章的概率问题从已知模型出发，计算可能数据的机会；统计推断从有限数据出发，讨论未知模型特征。总体是研究对象的分布或有限对象集合，样本是按某种机制实际获得的观测。参数是总体模型中的固定特征，如均值 μ 、方差 σ^2 或成功率 p ；统计量是只由样本计算、而不含未知参数的函数，如样本均值、样本方差与样本最大值。这些词区分的是对象层次，而非数值大小。[P09]

抽样之前， $X_1, \dots, X_n$ 是随机变量，样本均值 $\bar{X}$ 也是随机变量；抽样之后， $x_1, \dots, x_n$ 与 $\bar{x}$ 是确定的观测值。估计量是为估计参数而选择的统计量，估计值则是它在当前样本上的数值。参数固定不等于已知，统计量随机不等于没有规律：给定总体与抽样机制后，统计量具有抽样分布，区间与检验都在这个分布上建立。

独立同分布抽样是常用起点，简称 i.i.d.，表示各观测相互独立并来自同一分布。时间序列中的相邻测量、同一家族的成员、同一设备上的重复读数都可能有关联；总体不均匀且抽样有选择时也可能不具有同分布性质。样本量大只能减小适当抽样模型下的随机误差，不能自动消除选择偏差或系统测量偏差。

样本均值 $\bar{X} = (\sum X_i)/n$ 在可积时满足 $E[\bar{X}] = \mu$ ，称为总体均值的无偏估计量。无偏是重复抽样平均意义上的性质，不保证当前估计刚好等于 μ 。样本方差 $S^2 = \sum (X_i - \bar{X})^2 / (n-1)$ 在独立同分布

且有限二阶矩条件下对 σ^2 无偏；用 n 作分母则是另一种估计量。样本标准差 $S=\sqrt{S^2}$ 一般不对 σ 无偏，开平方并不保持期望。

$$\sum_{i=1}^n (X_i - \bar{X})^2 = \sum_{i=1}^n (X_i - \mu)^2 - n(\bar{X} - \mu)^2, \quad \mathbb{E}[S^2] = \sigma^2 \quad (5.1)$$

μ 为未知总体均值， $\bar{X}$ 为样本均值。左边用一个由数据估计的中心，平方偏差总和平均少了一个 σ^2 ：右边取期望得 $n\sigma^2 - n(\sigma^2/n) = (n-1)\sigma^2$ ，故除以 $n-1$ 恢复无偏。这里 $n \geq 2$ ，观测独立同分布且方差有限。

估计量还可用偏差、方差与均方误差评价。均方误差 $\mathbb{E}[(\theta - \hat{\theta})^2] = \text{Var}(\hat{\theta}) + [\mathbb{E}(\hat{\theta}) - \theta]^2$ ，把波动与系统偏差放在同一尺度。无偏并非唯一目标：某个稍有偏差但波动明显更小的估计量，均方误差可能更低。报告估计值时说明对象、单位、抽样机制和精度，通常比只说“这个估计无偏”更能帮助读者判断结论。

5.3 频率学与贝叶斯视角 随机的对象是什么

同一组观测可以支持不同的推断问题。频率学通常把参数 θ 视为固定未知值，把重复抽取的数据与由它构成的估计量、区间和检验结果视为随机对象。它问：在某个真实参数下，方法会多常覆盖真值、误报或漏报？贝叶斯方法再为未知参数规定先验分布，并在观察当前数据后求条件分布；它问：给定这份数据、先验和模型，参数及未来观测还有多大不确定性？这两种视角的区别不在于是否会算样本平均，而在于概率分配给什么对象。[P18]

频率学的“参数固定”不表示参数已经知道；贝叶斯的“参数有分布”也不必声称每次生产的真实参数都在物理上重新抽一次。对于同一设备的固定未知均值，先验可以表述分析者在数据之前的不确定性。若实际设备均值确有组间变化，则还需要明确层次总体，而不能只靠“随机参数”四个字混合两种解释。

两种方法共享似然与数据模型，也都要面对选择偏差、依赖、误测和模型不合适。频率学结果不依赖参数先验，却依赖抽样分布与规则；贝叶斯结果说明当前条件概率，却依赖先验及其合理性。报告时可以并列估计值、置信区间、后验区间和预测，但须各自标明含义，不能把同一个 95% 在这些对象间直接搬用。

例题 用同一成功计数比较两个问题

教学假设：固定做 20 次、给定未知成功率后各次独立，观察 15 次成功。频率学极大似然估计为 $15/20=0.75$ ；若要区间，需要指定二项区间构造并评价它的重复抽样覆盖。这个估计值自身不是“成功率等于 0.75 的概率”。

贝叶斯若事前取贝塔分布(2,2)，后验为贝塔分布(17,7)，均值 $17/24 \approx 0.70833$ 。当前后验可以回答成功率落在某段区间的概率；同时还能对下一次成功作预测。先验中心 0.5 带来的权重解释了两个点估计不同。若更换先验，须说明依据并检查敏感性。

本章的抽样推断将直接调用第 2 章中心极限定理，不再重复其模拟。使用均值的正态近似前，应回到第 2 章核对独立同分布、有限方差与近似质量；小样本正态总体的精确 t 推断则有更具体的分布前提。把极限定理放在前面，是为了让区间公式有概率依据，而不是以“样本够大”替代论证。

5.4 似然与极大似然估计

给定参数 θ 时，概率模型把数据 x 看成变量；数据已观察后，似然把 x 固定，比较不同 θ 对这份数据的相对支持。离散模型中 $L(\theta; x) = P_\theta(X=x)$ ，连续模型中使用样本密度。似然作为 θ 的函数通常不归一化，因此不是“ θ 取某值的概率”。要对 θ 本身赋概率，还需要先验并应用贝叶斯公式。

独立观测的联合质量或联合密度是各项乘积，取对数后变为求和。最大化似然与最大化对数似然等价，因为对数函数严格递增。对数形式既便于求导，也避免许多小密度连乘造成数值下溢。若观测不独立，则不能直接相乘边缘密度，应使用适当的联合分布。[P09]

$$L(\theta; x_1, \dots, x_n) = \prod_{i=1}^n f(x_i; \theta), \quad \ell(\theta) = \log L(\theta) = \sum_{i=1}^n \log f(x_i; \theta) \quad (5.2)$$

θ 表示待估参数， x_i 为已固定的观测， f 可以是概率质量函数或概率密度函数。 Π 表示逐项相乘， ℓ 为对数似然， $\log$ 为自然对数。极大似然估计 $\hat{\theta}$ 是在允许参数范围内使 L 或 ℓ 最大的值；应检查参数约束、边界及最大值是否存在。

例题 手算 5-1：二项成功率的极大似然

固定 n 次条件独立伯努利试验，观察到 k 次成功。略去与 p 无关的组合系数， $L(p)$ 正比于 $p^k(1-p)^{n-k}$ 。 $0 < k < n$ 时，对数似然导数为 $k/p - (n-k)/(1-p)$ ，令其为 0 得到 $\hat{p} = k/n$ ，二阶导数为负。

若 $n=20$ 、 $k=15$ ，极大似然估计为 0.75。若 $k=0$ 或 $k=n$ ，最大值分别在边界 $p=0$ 或 $p=1$ ，不能机械地只在开区间求驻点。0.75 表示最佳拟合参数，并非当前“参数等于 0.75 的概率”。

对正态独立样本，均值的极大似然估计是 $\bar{x}$ ，方差的极大似然估计为 $\sum(x_i - \bar{x})^2/n$ 。它与无偏样本方差的分子不同，说明“极大似然”与“无偏”是两种评价标准。选择估计方式时应说明方法，不能只把软件返回值都称为唯一正确的参数。

若正值数据 X 的对数 $Y = \log X$ 服从 $N(\mu, \sigma^2)$ ，则 X 为对数正态分布。其密度由变量变换得到 $f_X(x) = f_Y(\log x)/x$ ， $x > 0$ ； μ 是对数尺度均值，原尺度中位数为 $\exp(\mu)$ 而均值为 $\exp(\mu + \sigma^2/2)$ 。在原尺度比较候选密度时，这个 $1/x$ 变换因子必须保留。只比较 $\log X$ 和 X 上的对数似然总和，若未校正变换与数据尺度，就不再是同一比较问题。

5.5 置信区间与单样本 t 推断

点估计给出一个中心值，区间估计还要表达抽样不确定性。频率学置信区间是由样本决定的随机区间 $[L(X), U(X)]$ 。若在指定模型与重复抽样方式下，它以至少 $1-\alpha$ 的概率覆盖固定参数 μ ，就称置信水平为 $1-\alpha$ 。区间计算完成后，两个端点已固定， μ 也仍为固定未知值；“95%”描述构造方法的长期覆盖率。[P10]

若总体正态且 σ 已知，标准化均值 $(\bar{X} - \mu)/(\sigma/\sqrt{n})$ 为标准正态，取两尾各 $\alpha/2$ 的临界值即可解出 μ 的区间。 σ 未知时，用 S 代替并不能让统计量继续精确正态。正态独立样本中， $T = (\bar{X} - \mu)/(S/\sqrt{n})$ 精确服从自由度 $n-1$ 的 t 分布，它比标准正态具有更厚的尾，以反映估计尺度带来的额外波动。

$$\bar{X} \pm t_{n-1, 1-\alpha/2} \frac{S}{\sqrt{n}} \quad (5.3)$$

此式是正态独立同分布样本、总体方差未知时 μ 的双侧 $1-\alpha$ 置信区间； $\pm$ 表示分别取减号、加号得到两端。 $t_{v,q}$ 为自由度 v 的 t 分布第 q 分位数， S 为用 $n-1$ 分母的样本标准差。样本量 $n \geq 2$ ， $S > 0$ 时通常按此式计算。

例题 手算 5-2：九个给定观测的均值区间

观测为 168、172、175、169、171、173、170、174、167，共 9 个。总和 1539，均值 171；与 171 的平方偏差和为 60，所以 $S^2=60/8=7.5$ ， $S \approx 2.73861$ ，标准误 $S/\sqrt{9} \approx 0.912871$ 。

95% 双侧区间使用自由度 8 的 t 临界值约 2.30600，误差边界约 2.10508，区间为 [168.89492, 173.10508]。这是总体均值区间；单个未来观测的波动还包含个体差异，不能把这个区间当作 95% 个体覆盖区间。

小样本 t 区间的精确性依赖总体正态。大样本在适当条件下可采用近似，但严重偏斜、异常值或相关性仍可能影响覆盖率。 t 分布并不是自动修复这些问题的工具。区间宽度还由置信水平、变异程度与样本量共同决定：同一数据提高置信水平会使区间变宽；在其他条件相同时增加样本量会缩小标准误。

置信区间与预测区间还应明确区分。总体均值是一个稳定中心，未来单个观测同时包含中心估计误差和个体波动，预测区间通常更宽。在正态模型下，未来一个独立观测的预测区间使用 $S\sqrt{(1+1/n)}$ ，而均值区间使用 $S/\sqrt{n}$ 。即使均值估得很精确，单个观测也未必靠近均值。

5.6 假设检验 p 值与实际差异

假设检验先设定原假设 H_0 、备择假设 H_1 、检验统计量和判断规则，再衡量数据与 H_0 的相容程度。例如 $H_0: \mu=170$ ，双侧 $H_1: \mu \neq 170$ 。单样本 t 统计量比较“观测均值偏离原假设多少”与“均值的估计标准误多大”，同样的差 1 个单位，在精度不同的样本中会有不同证据强度。

p 值是在 H_0 及其抽样条件成立时，得到至少像当前结果这样极端的检验统计量的概率。双侧 t 检验采用 $P(|T| \geq |t_{\text{obs}}|)$ ，单侧检验则按事先规定方向取尾概率。 p 值不是 H_0 成立的概率，也不是数据由偶然造成的概率。看到数据后再挑选显著的方向会改变错误率，不能仍按原单次检验解释。[P11]

$$t_{obs} = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}, \quad p_{two} = 2P(T_{n-1} \geq |t_{obs}|) \quad (5.4)$$

$\bar{x}$ 与 s 是当前样本均值、标准差， μ_0 是 H_0 规定的均值， t_{obs} 为观测统计量。 T_{n-1} 为 H_0 下自由度 $n-1$ 的参考 t 变量。对称性使双侧概率可用一个上尾乘 2；不应把有符号 t 直接传入上尾再乘 2。

显著性水平 α 是检验规则在原假设成立时拒绝它的概率上限，称为第一类错误控制。备择成立却未拒绝是第二类错误，检验功效是某个具体备择下正确拒绝的概率。 α 并不告诉我们某次拒绝是否错了，也不决定所有备择的功效。样本量、效应大小和噪声水平都会影响发现差异的能力。

当 $p > \alpha$ 时说“未拒绝 H_0 ”，其含义是当前证据不足以按该规则拒绝，而不是证明均值恰好相等。 p 很小也不说明差异一定大；样本极大时很小的实际差异也可能显著。报告时应把估计差值、区间、单位与研究背景一起给出。若同时尝试许多检验并只报告最小 p 值，单次 α 不再代表整个分析的误报水平。

程序 `e18_t_inference.py` 程序使用上节九个原始观测， $\mu_0=170$ 、 $\alpha=0.05$ 。样本均值 171， $t \approx 1.095445$ ，双侧 $p \approx 0.305201$ ，95% 均值区间约 [168.894916, 173.105084]。在正态独立样本假设下，0.05 水平未拒绝 H_0 ，区间也包含 170；这两者对应同一双侧检验。图左显示自由度 8 的参考 t 密度与标准正态，图右显示当前均值及其区间。

编程思路 对九个观测检验总体均值是否为 170；先用样本标准差得到均值标准误，再构造 t 统计量与双侧 p 值。用同一临界值建立置信区间，并与 SciPy 的单样本 t 检验相互核对。

代码对应

$SE = s/\sqrt{9}$, $s^2 = \sum (x_i - \bar{x})^2 / (9-1) \rightarrow se = x.std(ddof=1) / np.sqrt(n)$: `std(ddof=1)` 使用八个自由度，随后除以样本量平方根。

$t = (\bar{x} - 170) / SE \rightarrow t_value = (x.mean() - mu0) / se$: 分子比较样本均值与原假设值，分母为估计标准误。

双侧 $p=2P(T \geq |t|) \rightarrow p_value = 2 * stats.t.sf(abs(t_value), df=n - 1): sf$ 输入绝对统计量，并乘二覆盖两个对称尾部。

$\bar{x} \pm t_{\alpha/2} SE \rightarrow interval = x.mean() + np.array([-1, 1]) * critical * se$: 置信区间与显著性水平 0.05 的双侧检验采用同一临界值。

易错提示 小样本精确 t 推断依赖独立正态观测；程序给出数值不等于已验证这些条件。未拒绝 $\mu=170$ 也不能证明总体均值恰好等于 170。

正文代码 e18_t_inference.py

```
# 教材位置: 第 5 章, 第 5.6 节; 图 5-1。
# 单位、变量与形状: 样本按同一般测量单位计; 均值区间保留该单位,  $t$  统计量与  $p$  值无量纲
# 。
# 学习目标: 手算单样本双侧  $t$  检验和均值置信区间, 并与 SciPy 的实现核对。
# 核心数学关系: 样本标准差为  $s$  时,  $SE=s/\sqrt{n}$ ,
#  $t=(\text{样本均值}-\mu_0)/SE$ ; 对正态总体的独立同分布样本,
# 该统计量在原假设下服从  $t_{(n-1)}$ 。
# 前提: 总体标准差未知, 观测应独立;  $n=9$  很小, 精确  $t$  推断依赖总体正态性,
# 数据若严重偏斜或有异常值, 不能仅因使用  $t$  分布就认为推断可靠。
# 数据与输出: 九个假想测量值, 检验总体均值是否为 170; 打印均值、 $t$  值、
# 双侧  $p$  值和 95% 置信区间, 并保存 e18_t_inference.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e18_t_inference.py。

import numpy as np
from scipy import stats

# 一维数组中每个元素代表一个观测, dtype=float 便于后续连续量计算。
x = np.array([168, 172, 175, 169, 171, 173, 170, 174, 167], dtype=float)
#  $\mu_0$  是原假设  $H_0: \mu=170$  的均值;  $\alpha=0.05$  是预先选定的显著性水平。
# 此处备择假设是  $\mu \neq 170$ , 因此需要考虑正负两个方向的偏离。
mu0, alpha = 170.0, 0.05
# size 返回观测总数; 此处  $x$  是一维, 所以也等于数组长度。
n = x.size
# ddof=1 使样本方差分母为  $n-1$ ; std 再开平方得到样本标准差  $s$ 。
# SE 描述样本均值的抽样波动, 和描述个体离散程度的  $s$  是不同的量。
se = x.std(ddof=1) / np.sqrt(n)
# 标准化均值与原假设的差异; 正  $t$  值表示样本均值大于  $\mu_0$ 。
# 用样本估计标准差后有  $n-1=8$  个自由度, 所以采用  $t$  分布。
t_value = (x.mean() - mu0) / se
#  $t$  分布关于 0 对称; 先算  $|t|$  右侧尾概率, 再乘 2 得到双侧  $p$  值。
#  $p$  值是在  $H_0$  和模型前提下, 得到至少这么极端统计量的概率,
# 并不是“原假设为真”的概率; sf 用于直接计算右尾概率。
p_value = 2 * stats.t.sf(abs(t_value), df=n - 1)
# 双侧中心 95% 对应每尾  $\alpha/2$ ; ppf 返回右侧正临界值。
critical = stats.t.ppf(1 - alpha / 2, df=n - 1)
#  $[-1, 1]$  与标量相乘, 一次生成“均值-误差界、均值+误差界”两个端点。
# 95% 置信度是重复抽样时该构造方法覆盖固定总体均值的长期比例,
# 不能把它直接解释成已得到的固定区间对固定参数的后验概率。
interval = x.mean() + np.array([-1, 1]) * critical * se
# SciPy 默认做双侧单样本  $t$  检验; popmean 指定原假设的总体均值。
result = stats.ttest_1samp(x, popmean=mu0)
```

```

print("样本均值、t 统计量、p 值: ", x.mean(), t_value, p_value)
print("95% 置信区间: ", interval)
# 分别比较统计量、双侧 p 值和区间端点, 防止手工推导出出现自由度或单双侧错误。
# 浮点运算存在舍入误差, 用 isclose/allclose 进行近似比较。
assert np.isclose(t_value, result.statistic)
assert np.isclose(p_value, result.pvalue)
# confidence_interval() 默认置信水平为 0.95, 恰好与本例的 1-alpha 相同。
assert np.allclose(interval, result.confidence_interval())

# 教材图 5-1 使用原九个观测及本次计算的置信区间; 总体均值 mu 和总体标准差均未知。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图只画本例检验实际采用的 n-1 自由度 t 密度, 并用标准正态密度作尾部参照。
t_grid = np.linspace(-5, 5, 500)
axes[0].plot(t_grid, stats.t.pdf(t_grid, df=n - 1), label=f't 分布: 自由度={n - 1}')
axes[0].plot(t_grid, stats.norm.pdf(t_grid), '--', label='标准正态')
axes[0].axvline(t_value, color='C2', linestyle=':', label=f'观察 t={t_value:.3f}')
axes[0].set(title='原假设下的参考分布', xlabel='t / z', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图误差条的端点直接取 interval, 不另写自由度 8 或分母 3, 避免数据变化后两处失配。
mean_x = x.mean()
ci_errors = np.array([mean_x - interval[0], interval[1] - mean_x])
axes[1].errorbar(mean_x, 0.6, xerr=ci_errors, fmt='o', capsize=5)
axes[1].axvline(mu0, color='C1', linestyle='--', label=rf'$H_0: \mu=\mu_0$')
axes[1].text(mean_x, 0.82, f'样本均值 = {mean_x:.2f}', ha='center', fontsize=8)
axes[1].text(mean_x, 0.34,
              f'{1 - alpha:.0%} 置信区间 = [{interval[0]:.2f}, {interval[1]:.2f}]',
              ha='center', fontsize=7.8)
axes[1].set(title=f'模拟样本: n={n}', xlabel='总体均值尺度',
            yticks=[], ylim=(0.15, 1.05),
            xlim=(min(interval[0], mu0) - 0.7, max(interval[1], mu0) + 0.7))
axes[1].legend(fontsize=7.8, loc='upper right')
fig.savefig('e18_t_inference.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

规定环境下的实际运行输出

样本均值、t 统计量、p 值: 171.0 1.0954451150103321 0.3052013652062302

95% 置信区间: [168.89491586 173.10508414]

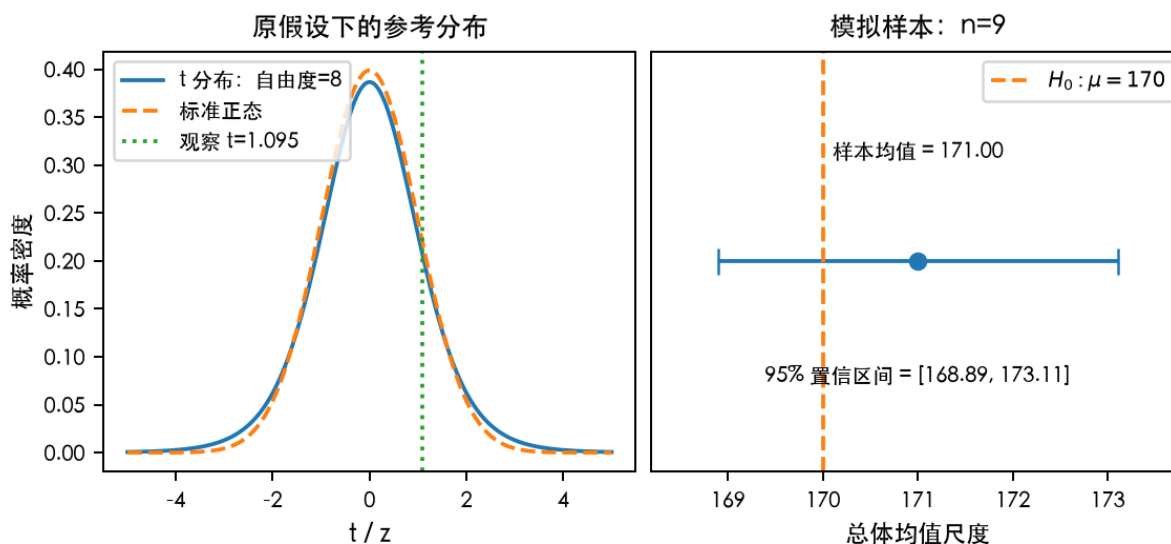


图 5-1 九个给定观测的 t 推断，由 `e18_t_inference.py` 生成。左图只比较本例自由度 8 的 t 密度与标准正态密度，竖线为观测 t 值；右图以本例 `x.mean()`、`interval` 和 `mu0=170` 显示均值点、95%置信区间及原假设均值，参数均与终端输出同源。

读图指南 左图横轴是无量纲 t 值，右图横轴是原观测单位。右侧横线估计总体均值的不确定性，不是 9 个观测的最小最大范围；原假设 170 位于区间内。

对同一个正态模型、同一 α 及同一种双侧 t 方法， μ_0 落在置信区间外等价于拒绝 $H_0: \mu = \mu_0$ 。这种检验与区间的对应来自同一个不等式的两种读法。更换检验方向、采用不同方差模型或改用其他区间构造后，需要重新核对，不能把这种对应当成所有方法之间自动成立的规则。

5.7 贝塔 二项更新与后验可信区间

贝叶斯推断为未知参数规定先验分布，再用观察到的数据更新。设 Θ 为未知成功率，先验 $\Theta \sim \text{贝塔分布}(\alpha, \beta)$ ；给定 $\Theta = \theta$ 时， $K \sim \text{二项分布}(n, \theta)$ 。这里“给定”非常重要：各次伯努利试验在给定共同成功率后独立，先验不确定性则描述我们对这个共同率的认识。先验、似然与后验分别对应数据前的信息、参数对数据的解释、数据后的信息。

贝叶斯公式给后验密度正比于“先验密度 $\times$ 似然”。先验核为 $\theta^{\alpha-1}(1-\theta)^{\beta-1}$ ，二项似然中与 θ 有关的部分为 $\theta^k(1-\theta)^{n-k}$ ，相乘后指数分别变成 $\alpha+k-1$ 与 $\beta+n-k-1$ 。因此后验仍为贝塔分布，称为共轭更新。这是由函数形式推出来的，不应只把它当作成功失败各加一次的记忆法。[P12]

$$\theta|K = k \sim \text{Beta}(\alpha + k, \beta + n - k), \quad \mathbb{E}[\theta|K = k] = \frac{\alpha + k}{\alpha + \beta + n} \quad (5.5)$$

α 、 β 为先验形状， n 为试验数， k 为观测成功数。 $\alpha+k$ 与 $\beta+n-k$ 为后验形状；该后验建立在同一个固定未知成功率、给定参数后的独立二项模型之上。后验均值也是下一次伯努利试验成功的后验预测概率，可由全期望公式得到。

后验均值还可写成 $[\kappa/(\kappa+n)]m+[n/(\kappa+n)](k/n)$ ，其中 $m=\alpha/\kappa$ 、 $\kappa=\alpha+\beta$ ，且 $n>0$ 。它把先验均值与样本比例按集中程度和样本量加权；数据越多，样本比例权重越大。 κ 常被解释为先验强度，但具体“等价观测次数”的说法依赖参数约定，不应把任意先验都当成真实收集过的观测。

后验等尾 95%可信区间取后验的 2.5%与 97.5%分位数。给定当前数据、先验与模型，参数落在这个区间的后验概率为 95%。这句话与置信区间的重复抽样覆盖率有不同的概率对象。二者有时数值接近，却不能仅靠换名字互相替代；贝叶斯区间还应说明先验，频率学区间应说明抽样与覆盖方法。

程序 `e17_beta_update.py` 程序固定先验贝塔分布(2,2)， $n=20$ 、 $k=15$ ，后验为贝塔分布(17,7)，均值 $17/24 \approx 0.708333$ ，等尾 95%可信区间约[0.515948,0.867897]。网格含 0 至 1 共 4001 个候选值，`prior.pdf(grid)`与 `binom.pmf(k,n,grid)`逐点相乘，再用梯形积分归一化。`grid_mean` 与解析均值核对，区别在于网格积分为数值近似；它不是另一次采集的样本。

编程思路 以 Beta(2,2) 为先验，观察 20 次中 15 次成功，得到 Beta(17,7) 后验。既用共轭参数更新，也在网格上把先验密度乘二项似然并归一化，以数值积分独立复核。

代码对应

$\alpha'=2+15=17$ ， $\beta'=2+(20-15)=7 \rightarrow \text{posterior} = \text{stats.beta}(\alpha + k, \beta + n - k)$ ：
成功数加到 α ，失败数加到 β ，不能把总试验数同时加到两个参数。

后验核 $\propto$ 先验密度 $\times P(K=15|\theta) \rightarrow \text{unnormalized} = \text{prior.pdf}(\text{grid}) * \text{stats.binom.pmf}(k, n, \text{grid})$ ：
`grid` 是未知成功概率 θ 的候选值，`binom.pmf` 的观察计数 k 与它的概率参数角色不同。

用积分常数归一化后验核 $\rightarrow \text{density} = \text{unnormalized} / \text{np.trapezoid}(\text{unnormalized}, \text{grid})$: 连续密度在 θ 网格上应按 trapezoid 积分归一化, 而不是只把数组元素相加。

$E(\theta|\text{数据}) = \int \theta p(\theta|\text{数据}) d\theta = 17/24 \rightarrow \text{grid_mean} = \text{np.trapezoid}(\text{grid} * \text{density}, \text{grid})$: 网格积分结果与共轭分布解析均值比较。

易错提示 可信区间表述的是给定数据后的参数后验概率, 不等同于重复抽样置信区间。网格密度数值会依赖网格精度, 解析 Beta 更新不依赖该离散化。

正文代码 e17_beta_update.py

```
# 教材位置: 第 5 章, 第 5.7 节; 图 5-2。
# 单位、变量与形状: grid、density 为 (4001,); 成功概率与参数空间无量纲, 区间是未知概率的
# 后验区间。
# 学习目标: 用 贝塔-二项 共轭更新估计未知成功概率, 并用数值积分核对。
# 核心数学关系:  $p \sim$  贝塔分布(alpha,beta)、 $K|p \sim$  二项分布(n,p) 时,
#  $p|K=k \sim$  贝塔分布(alpha+k,beta+n-k), 后验均值为 (alpha+k)/(alpha+beta+n)。
# 数据与输出: 先验 贝塔分布(2,2), 20 次条件独立的同概率试验中成功 15 次;
# 打印解析/积分得到的后验均值和等尾 95% 可信区间, 并保存 e17_beta_update.png。
# 运行方式: 安装支持 np.trapezoid 的 NumPy、SciPy 及 Matplotlib 后,
# 在终端运行 python e17_beta_update.py。

import numpy as np
from scipy import stats

# alpha、beta 为正的先验形状参数; n 是试验数, k 是成功数, 失败数为 n-k。
# 贝塔分布 密度含  $p^{alpha-1}(1-p)^{beta-1}$ , 体现先验对成功率的倾向。
alpha, beta, n, k = 2.0, 2.0, 20, 15
# 对称的 贝塔分布(2,2) 以 0.5 为中心, 相比均匀先验更偏向中间成功率。
prior = stats.beta(alpha, beta)
# 似然含  $p^k(1-p)^{n-k}$ , 与先验相乘后指数相加, 得到 贝塔分布(17,7)。
# 共轭更新计算的是未知参数 p 的分布, 而不是未来成功次数的分布。
posterior = stats.beta(alpha + k, beta + n - k)
# 用 4001 个等距点覆盖参数空间 [0,1]; 一维数组形状为 (4001,)。
# 这里积分较平滑且端点密度为零, 细网格能给出很精确的数值核对。
grid = np.linspace(0, 1, 4001)
# 逐个候选 p 计算“先验密度 × 观测到恰好 k 次成功的似然”。
# binom.pmf 的第三个参数在这里是整条 grid, 所以输出与网格形状相同。
# 得到的函数尚未归一化, 其关于 p 的积分是观测数据的边际概率。
unnormalized = prior.pdf(grid) * stats.binom.pmf(k, n, grid)
# trapezoid(y,x) 用梯形法近似积分; 除以总面积后得到积分约为 1 的后验密度。
# 连续密度必须按网格间距积分, 不能直接把高度的和当作精确的积分。
density = unnormalized / np.trapezoid(unnormalized, grid)
# 连续后验的期望是  $\int p f(p|data) dp$ ; 这里使用同一网格数值近似。
grid_mean = np.trapezoid(grid * density, grid)
print("解析后验均值、积分后验均值: ", posterior.mean(), grid_mean)
# ppf 接收两个左侧累计概率, 给出各留 2.5% 后验质量的等尾可信区间。
# 在本先验和模型下, 参数落在该区间内的后验概率为 95%。
print("95% 可信区间: ", posterior.ppf([0.025, 0.975]))
# 解析均值  $17/(17+7)=17/24$ ; 后验均值同时融合先验与样本成功率 15/20。
assert np.isclose(posterior.mean(), 17 / 24)
# 积分存在离散化误差, 用绝对容差 atol=1e-6 检查数值结果足够接近解析值。
```

```

assert np.isclose(grid_mean, posterior.mean(), atol=1e-6)

# 教材图 5-2 中的未知成功率用数学符号  $\theta$  表示; SciPy 的 binom 接口把同一参数称为 p。
# 原数组 grid 的每个元素都是一个候选  $\theta$ , 既不是观测次数 k, 也不是新的随机样本。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 原积分网格在本例端点密度为 0, 可直接用于画先验及解析后验, 不另设隐藏网格参数。
prior_label = f'先验贝塔({alpha:g}, {beta:g})'
posterior_label = f'后验贝塔({alpha + k:g}, {beta + n - k:g})'
axes[0].plot(grid, prior.pdf(grid), label=prior_label)
axes[0].plot(grid, posterior.pdf(grid), color='C1', label=posterior_label)
# 等尾可信区间直接采用本例 posterior 的分位数, 阴影面积约为 95% 的后验质量。
credible_lower, credible_upper = posterior.ppf([0.025, 0.975])
axes[0].fill_between(grid, posterior.pdf(grid),
                    where=(grid >= credible_lower) & (grid <= credible_upper),
                    color='C1', alpha=0.2, label='95% 可信区间')
axes[0].axvline(posterior.mean(), color='C1', linestyle='--', linewidth=1)
axes[0].set(title=f'共轭更新: n={n}, k={k}',
            xlabel=r'未知成功概率  $\theta$ ', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图对照解析共轭后验与原数值积分的归一化密度; 二者接近重合是实现核查。
axes[1].plot(grid, posterior.pdf(grid), linewidth=2.5, alpha=0.6,
            label='解析后验')
axes[1].plot(grid, density, '--', color='C2', linewidth=1.2,
            label='归一化网格密度')
axes[1].set(title='数值积分核对',
            xlabel=r'候选  $\theta$  (接口参数 p)', ylabel='后验密度')
axes[1].legend(fontsize=7.8)
fig.savefig('e17_beta_update.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

规定环境下的实际运行输出

解析后验均值、积分后验均值: 0.7083333333333334 0.7083333333333334

95% 可信区间: [0.51594803 0.86789712]

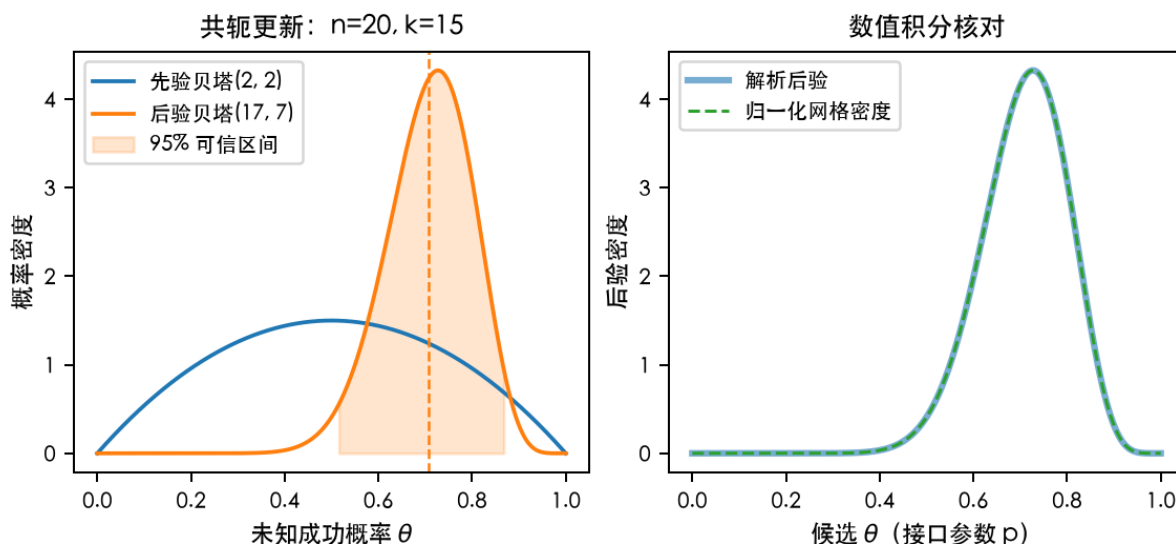


图 5-2 同一组贝塔分布共轭更新与数值积分核验，由 `e17_beta_update.py` 生成。左图先验贝塔分布(2,2)、观测 $n=20$ 且 $k=15$ 后的后验贝塔分布(17,7)，阴影为后验等尾 95%可信区间；右图比较解析后验与 4001 点网格归一化的数值密度。横轴 θ 为未知成功率，对应 SciPy 参数 p 和代码 `grid` 中的候选值。

读图指南 横轴是未知成功率，不是一次试验的 0 或 1 结果。后验阴影是对当前参数的不确定性；右图两条曲线核对解析式与网格积分，不代表额外采集了一组数据。

该例的样本比例 0.75 与后验均值约 0.7083 不同，是因为先验中心 0.5 仍有权重。若换用不同先验，后验数值也可能改变，应检查结论对此是否敏感。把连续参数后验密度某一点的高度解读为“该点的后验概率”仍然错误；区间后验概率需要积分。分布越来越集中时密度峰值可以升高，同时每个单点概率依旧为 0。

5.8 正态 正态共轭 按精度合并均值信息

连续均值也可以作共轭更新。设同一稳定过程的观测满足：给定未知均值 μ 后， X_1 至 X_n 条件独立，且各服从正态分布 $N(\mu, \sigma^2)$ ； $\sigma^2 > 0$ 是已知的观测方差。对 μ 指定正态先验 $N(\mu_0, \tau_0^2)$ ， $\tau_0^2 > 0$ 。这里 σ 描述单个观测的噪声， τ_0 描述数据前对共同均值的不确定性，两者单位相同、角色不同。若 σ^2 也是未知的，下面两参数公式便不能原样作为精确更新。[P12]

$$\mu \sim N(\mu_0, \tau_0^2), \quad X_i | \mu \sim N(\mu, \sigma^2) \quad (i = 1, \dots, n) \quad (5.6)$$

n 为观测数， μ_0 为先验中心， τ_0^2 为先验方差， σ^2 为已知观测方差。条件独立使似然能够写成各观测密度的乘积。先验参数必须在看到当前数据前选定，或明确说明其外部来源。

推导从“先验密度×似然”开始。略去与 μ 无关的正因子，后验对 μ 的依赖是两个平方指数相乘。先验贡献 $(\mu - \mu_0)^2 / \tau_0^2$ ，数据贡献各个 $(x_i - \mu)^2 / \sigma^2$ 。关键恒等式是把每个观测偏差分为 $x_i - \bar{x}$ 与 $\bar{x} - \mu$ ；由于 $\sum (x_i - \bar{x}) = 0$ ，交叉项相消，观测内部散布成为与 μ 无关的常数。

$$\sum_{i=1}^n (x_i - \mu)^2 = \sum_{i=1}^n (x_i - \bar{x})^2 + n(\bar{x} - \mu)^2 \quad (5.7)$$

$\bar{x}$ 为当前样本均值。第一项在 μ 变化时不变，因此对已知 σ^2 下的均值后验，只需 $\bar{x}$ 和 n ，而不需保存每个观测在先验更新式中的单独位置。

$$\pi(\mu|x) \propto \exp \left[-\frac{1}{2} \left(\frac{(\mu - \mu_0)^2}{\tau_0^2} + \frac{n(\mu - \bar{x})^2}{\sigma^2} \right) \right] \quad (5.8)$$

$\pi(\mu|x)$ 为后验密度； $\propto$ 表示只差一个与 μ 无关的归一化因子。此处小写 π 是密度记号，不是圆周率，圆周率已在略去的正态归一化常数中。

为看清配方，定义先验精度 $a_0 = 1/\tau_0^2$ 、数据对均值提供的精度 $a_D = n/\sigma^2$ 。精度是方差的倒数，单位为原测量单位的平方倒数。注意数据精度对应均值的方差 σ^2/n ，而不是单个观测方差 σ^2 。展开括号中的二次式， μ^2 系数是 $a_0 + a_D$ ， μ 的一次项系数是 $-2(a_0\mu_0 + a_D\bar{x})$ 。

$$a_0(\mu - \mu_0)^2 + a_D(\mu - \bar{x})^2 = (a_0 + a_D)\mu^2 - 2(a_0\mu_0 + a_D\bar{x})\mu + C_0 \quad (5.9)$$

$C_0 = a_0\mu_0^2 + a_D\bar{x}^2$ ，不含 μ 。把一次项系数除以二次项系数，就得到配方后的中心。

$$\tau_n^2 = \frac{1}{a_0 + a_D}, \quad \mu_n = \frac{a_0\mu_0 + a_D\bar{x}}{a_0 + a_D} \quad (5.10)$$

τ_n^2 为后验方差， μ_n 为后验均值。 μ_n 是两个中心的精度加权平均，两个权重均非负且和为 1。数据越精确或 n 越大，样本均值的权重越高。

$$a_0(\mu - \mu_0)^2 + a_D(\mu - \bar{x})^2 = \frac{(\mu - \mu_n)^2}{\tau_n^2} + C_1 \quad (5.11)$$

$C_1 = C_0 - \mu_n^2 / \tau_n^2$ ，也不含 μ 。归一化只需识别正态核，所以 μ 在当前数据下的后验为 $N(\mu_n, \tau_n^2)$ 。前面的后验参数公式给出了中心与方差。

由这个二次核可知，后验仍为正态，故称正态—正态共轭。后验方差只取决于先验精度、观测噪声和样本量；本模型的 σ^2 已知，所以当前观测之间的散布不会另行改变它。若观测有明显异常值或实际噪声远大于指定 σ ，不能只因公式算出了很窄后验就认定精度可靠，应先检查已知噪声假设。

例题 手算：四个给定观测的更新

全部数字为教学设定。取先验均值 $\mu_0=50$ 、先验标准差 $\tau_0=10$ ，观测标准差已知为 $\sigma=8$ 。四个观测为 52、48、55、51，样本均值为 51.5。先验精度为 0.01，数据精度为 $4/64=0.0625$ ，总精度为 0.0725。

后验方差为 $1/0.0725=400/29 \approx 13.79310$ ，标准差约 3.71391；后验均值为 $(0.01 \times 50 + 0.0625 \times 51.5) / 0.0725 \approx 51.29310$ 。样本中心的权重为 25/29，先验中心的权重为 4/29。后验中心在 50 与 51.5 之间，且更靠近数据。

后验等尾 95% 可信区间为 $\mu_n \pm 1.95996\tau_n$ ，约 [44.0140, 58.5722]。这表达对共同均值的不确定性，不是新观测的个体波动范围。先验很宽时 μ_n 接近 $\bar{x}$ 、 τ_n 接近 $\sigma/\sqrt{n}$ ，但有限宽先验的区间不能因此改名为频率学置信区间。

后验预测把“均值还不确定”与“新观测仍有噪声”合在一起。设未来 $X_{\text{新}}$ 在给定 μ 后与已有数据条件独立，且同为 $N(\mu, \sigma^2)$ 。全期望给预测中心 $E[X_{\text{新}}|\text{数据}] = \mu_n$ 。全方差给预测方差 $\sigma^2 + \tau_n^2$ ：第一项是给定均值后的个体噪声，第二项是当前均值的不确定性。在正态模型下，正态与正态相加仍正态，所以这同时确定了整个预测分布。

$$X_{\text{new}}|x \sim N(\mu_n, \sigma^2 + \tau_n^2) \quad (5.12)$$

$X_{\text{新}}$ 为将来的一次独立观测。后验预测标准差大于参数后验标准差；本例约为 8.82004。不能用 μ 的窄后验区间替代未来单个读数的预测区间。

程序 e36_normal_normal.py 先验 $N(50, 10^2)$ ，已知观测标准差 8，数据固定为 [52, 48, 55, 51]，不另作随机抽样。编程先求样本均值 51.5，再把 $1/10^2$ 与 $4/8^2$ 相加，得到后验方差 400/29 及后验均值约 51.293103。左图比较参数 μ 的先验与后验密度；右图显示下一次观测的后验预测密

度及区间。后验预测使用方差 $64+400/29$ 。也可自行作顺序更新，每次只加入一条观测的精度 $1/64$ ，最终结果须与批量更新一致。stats.norm 的 scale 接收标准差，须分别传入先验标准差、后验方差开平方与预测方差开平方；误把方差直接传入会显著改错图形宽度。

编程思路 在观测标准差 $\sigma=8$ 已知时，将 μ 的 $N(50,10^2)$ 先验与四个观测 $[52,48,55,51]$ 合并。以精度相加得到后验，再把观测噪声方差加回后验方差，分别计算参数可信区间与新观测预测区间。

代码对应

$\tau_n^2=[1/10^2+4/8^2]^{-1}=400/29 \rightarrow \text{posterior_var} = 1 / (\text{prior_precision} + \text{data_precision})$ ：先验与样本均值信息按精度相加，不能直接把两个方差相加。

$\mu_n=\tau_n^2(50/10^2+4\times 51.5/8^2)=2975/58 \rightarrow \text{posterior_mean} = \text{posterior_var} * (\text{prior_precision} * \mu_0 + \text{data_precision} * \text{sample_mean})$ ：它是先验均值与样本均值的精度加权平均，数据权重为 $25/29$ 。

参数 μ 的 95% 后验可信区间 $\rightarrow \text{credible} = \text{posterior.ppf}([0.025, 0.975])$ ：用 posterior 的 2.5% 与 97.5% 分位数，约为 $[44.01397995, 58.57222695]$ 。

新观测的预测方差 $=\sigma^2+\tau_n^2=64+400/29 \rightarrow \text{predictive_var} = \text{sigma}^2 + \text{posterior_var}$ ：未知均值的不确定性与未来独立观测噪声相加，预测区间因而比参数可信区间宽。

易错提示 共轭公式假设各观测条件独立、共享均值且 σ 已知；若 σ 也未知，不能照搬本例的正态后验。可信区间讨论 μ ，预测区间讨论一个新观测，两种对象和方差不可混淆。

正文代码 e36_normal_normal.py

```
# 教材位置：第 5 章，第 5.8 节；图 5-3。
# 已知观测标准差时的正态-正态共轭更新；本例完全解析，不随机抽样。
# 学习目标：把先验精度、样本信息和后验预测对应到可核对的计算。
# 观测、 $\mu$ 、 $\mu_0$ 、 $\tau_0$ 、sigma 使用同一测量单位；方差使用该单位的平方。
# data 形状为 (4,)，区间端点形状为 (2,)，所有密度共用原测量尺度。
# 参数： $\mu_0=50$ 、 $\tau_0=10$ 、sigma=8，四次观测为 52、48、55、51。
# 输出后验均值/方差/可信区间及下一次观测的预测区间，保存对应图片。
# 运行：安装 requirements.txt 中依赖后执行 python e36_normal_normal.py。

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot
```

```
# 假设给定  $\mu$  后四个观测条件独立, 同服从  $N(\mu, \sigma^2)$ 。
#  $\sigma=8$  是已知观测标准差, 不用这四个观测重新估计它。
mu0, tau0, sigma = 50.0, 10.0, 8.0
data = np.array([52.0, 48.0, 55.0, 51.0])
n = data.size
sample_mean = data.mean()
# 精度是方差的倒数: 独立正态信息在精度尺度上相加。
prior_precision = 1 / tau0**2
data_precision = n / sigma**2
posterior_var = 1 / (prior_precision + data_precision)
posterior_mean = posterior_var * (
    prior_precision * mu0 + data_precision * sample_mean
)
posterior_sd = np.sqrt(posterior_var)
# 后验均值是先验均值与样本均值的加权平均, 权重和为 1。
data_weight = data_precision / (prior_precision + data_precision)
assert np.isclose(data_weight, 25 / 29)
assert np.isclose(posterior_mean, 2975 / 58)
assert np.isclose(posterior_var, 400 / 29)
prior = stats.norm(loc=mu0, scale=tau0)
posterior = stats.norm(loc=posterior_mean, scale=posterior_sd)
# 等尾可信区间按后验分布分位数给出, 是未知  $\mu$  的后验概率范围。
credible = posterior.ppf([0.025, 0.975])
# 预测还含一次新观测自身的噪声, 不能只用 posterior_sd 作预测标准差。
predictive_var = sigma**2 + posterior_var
predictive = stats.norm(loc=posterior_mean, scale=np.sqrt(predictive_var))
predictive_interval = predictive.ppf([0.025, 0.975])
assert predictive_var > posterior_var
print("样本量、样本均值: ", n, sample_mean)
print("先验精度、数据精度: ", prior_precision, data_precision)
print("后验均值、后验方差: ", posterior_mean, posterior_var)
print("后验标准差: ", posterior_sd)
print(" $\mu$  的 95% 等尾可信区间: ", credible)
print("下一次观测的预测均值、方差: ", predictive.mean(), predictive_var)
print("下一次观测的 95% 预测区间: ", predictive_interval)

# 同一后验在两个面板中回答不同问题: 参数  $\mu$  与未来观测。
# 图仅呈现以上解析分布, 不再生成一批样本以替代精确计算。
configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0), constrained_layout=True)
grid = np.linspace(15, 85, 700)
axes[0].plot(grid, prior.pdf(grid), label=" $\mu$  的先验")
axes[0].plot(grid, posterior.pdf(grid), label=" $\mu$  的后验")
axes[0].fill_between(
    grid, posterior.pdf(grid),
    where=(grid >= credible[0]) & (grid <= credible[1]), alpha=0.2,
)
axes[0].axvline(sample_mean, color="gray", ls=":", label="样本均值")
axes[0].set(title="未知均值的共轭更新", xlabel="未知均值  $\mu$ ", ylabel="概率密度")
axes[1].plot(grid, predictive.pdf(grid), label="下一次观测的预测密度")
axes[1].plot(grid, posterior.pdf(grid), ls="--", label=" $\mu$  的后验密度")
axes[1].fill_between(
    grid, predictive.pdf(grid),
    where=(grid >= predictive_interval[0]) & (grid <= predictive_interval[1]),
    alpha=0.2,
)
axes[1].set(title="预测需保留观测噪声", xlabel="下一次观测值", ylabel="概率密度")
for ax in axes:
```



```
ax.legend(fontsize=7.5)
ax.grid(alpha=0.2)
fig.savefig("normal_normal.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

样本量、样本均值: 4 51.5

先验精度、数据精度: 0.01 0.0625

后验均值、后验方差: 51.293103448275865 13.793103448275863

后验标准差: 3.713906763541037

μ 的 95% 等尾可信区间: [44.01397995 58.57222695]

下一次观测的预测均值、方差: 51.293103448275865 77.79310344827586

下一次观测的 95% 预测区间: [34.00614295 68.58006395]

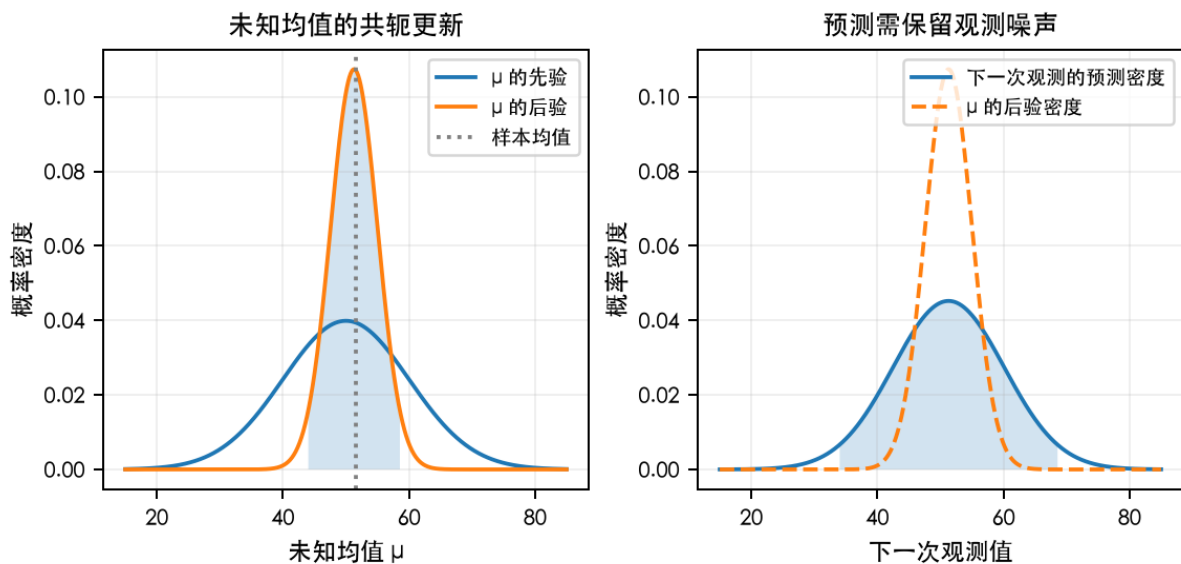


图 5-3 已知观测方差的正态—正态更新。先验中心 50、标准差 10，四个固定观测 52、48、55、51，观测标准差 8。左图比较参数 μ 的先验与后验；右图显示未来单个观测的后验预测密度及区间。场景及观测均为教学设定。

读图指南 左图横轴为共同均值 μ ，后验比先验更集中，中心从 50 移到约 51.2931；右图横轴为未来观测值，预测分布还包含标准差 8 的个体噪声，因而更宽。两图虽使用相同测量单位，却是不同随机对象，不可把参数区间当作个体覆盖区间。

敏感性检查可以分别改变先验中心、先验宽度与假定噪声。宽先验减少先验中心的影响，较小 σ 提高数据的权重；两者不是同一个操作。若当前样本不是独立同方差正态，或 σ 来自

很少数据而非确知外部校准，应改用相应模型，不能把估出的 σ 当作毫无误差的常数后仍宣称本公式精确。

5.9 分布拟合 分位数图与检验校准

选择候选分布时，先检查数据支持集、生成机制、暴露量与分组，再拟合参数。正态、伽马、对数正态即使都能画出平滑曲线，也可能对尾部、负值或尺度给出不同预测。直方图依赖分箱，曲线视觉上贴合不能代替对累积概率与分位数的检查。若最终用途是极端尾部预测，中心拟合好尤其不能保证尾部可靠。

经验累积分布函数把每个观测以 $1/n$ 质量累加。分位数—分位数图，常简称 Q—Q 图，则先把数据排序，横轴放候选参考分布在相应累计概率处的理论分位数，纵轴放样本有序值。判断弯曲前，必须先说清横纵轴、参考分布和数据是否经过对数变换；交换坐标轴会改变视觉方向。

本例程序实际调用 `stats.probplot`，SciPy 官方将其称为“概率图”：横轴采用指定理论分布的顺序统计量中位数估计，纵轴为排好序的样本值，并可拟合一条最小二乘参考线。它与常见理论分位数对样本分位数的 Q—Q 诊断思路接近，但绘图位置约定有别；更不能与比较累计概率的 P—P 图混用。下面讨论的位置尺度、两尾与样本波动，均应放回这个具体坐标约定理解。[P20]

如果样本来自参考家族的某个位置与尺度变换，点会大致沿一条直线。直线的截距反映位置差异，斜率反映尺度差异；仅仅偏离 45 度线不一定说明分布形状不对。图中的拟合线可能是最小二乘线或通过选定分位数的线，应结合绘图方法解释。样本有限时，最外侧几个顺序统计量尤其不稳定。

在“横轴标准正态理论分位数、纵轴当前数据”这一约定下，右端点明显高于参考线，表示相应的样本高分位数更大；左端点明显低于线，表示样本低分位数更小。若两端向外张开而中间较直，常提示相对于正态更重的两尾；若两端向内收，可能提示较轻尾。只看一端不能把位置、尺度、偏态和尾重完全分开，也不能把任意向上弯直接等同“右偏”。

若整段正态分位数图呈有规律的不对称弯曲，偏态是可调查的原因之一，但仍应同时查看两尾、原始尺度、离群点和样本量。对本例，右图使用 $\log(\text{data})$ 而非 data ；因此接近直线支持的是“对数数据与某正态分布相容”，与“原数据正态”是两个不同命题。对数正态原数据在正态参考下会出现形状偏离，换成正确的对数正态参考后则可能近似直线。

读图应寻找稳定、系统的模式，而不是把每个点必须在线上作为验收标准。模拟同样样本量的参考样本或绘制模拟包络，可以帮助判断尾点的自然波动；参数拟合会影响这种参照。Q—Q 图是诊断工具，不能单凭图形宣判独立性、因果关系或极端尾部外推可靠。

极大似然偏好能够更好适应当前数据的模型，增加参数常会提高拟合程度。 $AIC=2d-2\ell(\theta)$ 在最大对数似然之外加入自由参数数目 d 的惩罚。它用于同一数据、同一尺度、可比似然下候选模型的相对比较；较小表示这一准则更偏好，但不等于模型为真的概率，也不等于绝对拟合已经通过。预先固定的参数不计入自由参数数目。

$$D_n = \sup_x |F_n(x) - F(x; \hat{\theta})| \quad (5.13)$$

D_n 为 Kolmogorov—Smirnov 距离， F_n 为经验累积分布函数， $F(x; \theta)$ 为拟合模型累积分布函数， $\sup$ 表示在所有阈值 x 上取最大可能差。统计量越大表示累计分布偏离越明显；把它转换为 p 值还需要匹配参数是否已知、样本量与抽样机制的参考分布。

若模型参数事先完全指定，经典 KS 检验使用相应的固定分布参考。若参数从同一数据估计，拟合过程已经使曲线更接近数据，原来的参考分布通常不再适用。合理的参数自助校准先拟合原数据，再从拟合模型反复生成同样大小的新样本；每个新样本都重新拟合未知参数，再计算距离，最后用这些距离比较当前距离。重新拟合这一步必须与原分析一致。[P13]

模拟校准的 p 值也有蒙特卡洛波动。1999 次参考模拟时，常用 $(b+1)/(1999+1)$ 计数，其中 b 为参考统计量至少与观测一样大的次数；不能把没有更大参考值简单报为概率 0。校准解决的是检验参考分布的问题，并不会让模型假设自动成为事实。未拒绝意味着该检验没有发现足够偏离，仍可能存在它不敏感的结构差异。

程序 `e19_distribution_fit.py` 程序以种子 30 生成 150 个正值， $\log X$ 的均值为 $\log 5$ 、标准差 0.9，所以原尺度中位数为 5。候选为正态、伽马分布和对数正态；后两者固定 $\text{loc}=0$ ，各模型均有 2 个自由参数。图左在同一原尺度比较密度并打印 AIC；图右画 $\log(\text{data})$ 的正态概率图。`goodness_of_fit` 对 $\log(\text{data})$ 进行正态家族 KS 校准，使用种子 31 与 1999 个参考样本，并在各样本重新拟合未知均值和尺度。

编程思路 先生成固定种子的对数正态教学数据，再拟合正态、固定 $\text{loc}=0$ 的 Gamma 和对数正态三个候选模型。以对数似然和自由参数数目计算 AIC，并对拟合后正态化的对数数据使用蒙特卡洛校准的拟合优度检验。

代码对应

在每个模型约束下最大化样本似然 $\rightarrow \text{parameters} = \text{family.fit}(\text{data}, **\text{fixed})$ ：
`family.fit(data,**fixed)` 返回包含固定位置参数的完整参数元组，分布构造时需按原顺序传回。

$\ell = \sum \log f(x_i; \theta) \rightarrow \text{loglike} = \text{rv.logpdf}(\text{data}).\text{sum}()$ ：直接求 `logpdf` 的和比先连乘密度再取对数更稳定。

$\text{AIC} = 2q - 2\ell$ 中 q 是自由估计参数数 $\rightarrow \text{free_count} = \text{len}(\text{parameters}) - \text{len}(\text{fixed})$ ：
 固定 $\text{floc}=0$ 的参数不能再次计入惩罚项；参数元组长度本身不是 q 。

在每次模拟中重估参数，校准复合原假设检验 $\rightarrow \text{result} =$
`stats.goodness_of_fit(stats.norm, np.log(data), statistic="ks",`
`n_mc_samples=1999,`
`rng=np.random.default_rng(31))`：`goodness_of_fit` 对数据驱动的拟合进行蒙特卡洛校准，
 避免套用参数完全已知的 KS 表。

易错提示 AIC 只比较当前候选模型的相对拟合与复杂度，不证明获胜模型真实。这里生成机制已知只为教学；实际数据仍需考察独立性、支持、异常值与模型用途。

正文代码 `e19_distribution_fit.py`

```
# 教材位置：第 5 章，第 5.9 节；图 5-4。
# 单位、变量与形状：各候选分布用相同正值数据；观测单位任意但统一，对数表示采用该单位后的
# 数值。
# 学习目标：用极大似然拟合多个连续分布，用 赤池信息准则 比较候选模型，
```

```

# 并理解“参数来自同一份数据”时为何需要校准拟合优度检验。
# 核心数学关系:  $\log L = \sum(\log f(x_i; \theta_{\text{hat}}))$ , 赤池信息准则  $= 2 * k - 2 * \log L$ ,
# 其中  $k$  只计算自由估计的参数; 同一数据/同一观测尺度下通常偏好较小的 赤池信息准则。
# 拟合优度检验 统计量衡量经验 累积分布函数 与拟合 累积分布函数 的最大差异; 估计参数会改
# 变其零假设分布。
# 数据与输出: 生成 150 个正值, 打印正态/伽马分布/对数正态的拟合参数和 赤池信息准则,
# 打印对数数据的正态族 拟合优度检验 统计量及校准  $p$  值, 保存 e19_fit.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 运行 python e19_distribution_fit.py;
# 图片写入运行命令时的当前工作目录, 蒙特卡洛校准需要比简单拟合更多计算。

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 固定模拟种子, 使同一环境中的教学数据可重复; 校准稍后使用独立的种子。
rng = np.random.default_rng(30)
# lognormal 的 mean 和 sigma 是  $\log(X)$  的均值与标准差,
# 不是  $X$  本身的均值和标准差; 此处原尺度中位数为  $\exp(\log(5))=5$ 。
# 返回形状 (150,) 的正值数组, 模拟独立同分布观测。
data = rng.lognormal(mean=np.log(5), sigma=0.9, size=150)
# 每项包括显示名称、SciPy 分布族、拟合时固定的参数字典。
# floc=0 固定位置参数, 令 伽马分布 和对数正态的支持区间从 0 开始;
# 正态分布允许负值, 能否接受这种支持范围还需结合实际判断。
models = [("正态分布", stats.norm, {}), ("伽马分布", stats.gamma, {"floc": 0}),
          ("对数正态分布", stats.lognorm, {"floc": 0})]
# 左图叠加样本直方图和拟合密度, 右图检查对数数据的正态概率图关系。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3))
# density=True 把直方图归一化为总面积 1, 才能与连续密度曲线比较;
# bins=25 控制分箱数, alpha=0.4 让重叠曲线仍然可见。
axes[0].hist(data, bins=25, density=True, alpha=0.4)
# 在正数网格上画密度, 避免在 0 处触及某些分布的边界行为。
# 网格只用于显示, 不改变拟合时使用的完整原始数据。
x = np.linspace(0.01, data.max(), 300)
# Python 循环同时解包每个候选模型的三项配置。
for name, family, fixed in models:
    # **fixed 将字典展开为关键字参数; 本例各分布按极大似然拟合。
    # 返回元组依次含形状参数 (若有)、loc、scale, 固定参数也包含在元组中。
    parameters = family.fit(data, **fixed)
    # *parameters 按位置展开拟合结果, 生成可直接计算 pdf/logpdf 的分布对象。
    rv = family(*parameters)
    # 独立观测的联合密度为各密度乘积; 取对数后可求和。
    # 相比直接连乘大量很小的密度值, logpdf 能降低数值下溢风险。
    loglike = rv.logpdf(data).sum()
    # 本例 fixed 仅包含 floc, 因此减去字典长度恰好去掉固定位置参数。
    # 正态为均值/尺度两个参数; 伽马分布 和对数正态各自自由估计形状/尺度两个。
    # 这种计数写法依赖当前配置, 若以后添加非固定参数的拟合选项须重新判断。
    free_count = len(parameters) - len(fixed)
    # 赤池信息准则 在拟合优度与参数复杂度之间权衡; 它是候选模型的相对比较指标,
    # 不是“模型正确概率”, 也不能仅据最小 赤池信息准则 证明分布假设成立。
    print(name, "参数", parameters, "赤池信息准则", 2 * free_count - 2 * loglike)
    # 确保总对数似然既不是无穷也不是 NaN, 避免拿无效值作模型比较。
    assert np.isfinite(loglike)
    # 在同一原始数据尺度绘制各模型密度, label 供后续图例识别。
    axes[0].plot(x, rv.pdf(x), label=name)

```

```
# 对数正态模型等价于  $\log(X)$  服从正态分布；数据均为正才能安全取对数。
# 检验的是“某个均值/标准差的正态分布族”，这两个参数由当前  $\log$  数据估计。
# 若把估计参数当成预先已知参数，直接套用经典 拟合优度检验  $p$  值，校准会不正确。
# goodness_of_fit 从拟合零假设生成样本、每次重新估计未知参数并重算统计量，
# 用这套参数自助法模拟的参考分布进行校准；这里用 1999 次蒙特卡洛样本。
# 固定独立种子 31 便于复现；有限模拟次数使  $p$  值本身仍有蒙特卡洛误差。
result = stats.goodness_of_fit(stats.norm, np.log(data), statistic="ks",
                               n_mc_samples=1999, rng=np.random.default_rng(31))

# 校准  $p$  值针对上述复合原假设，不应与各模型 赤池信息准则 的数值大小直接比较。
# 较大  $p$  值只表示当前检验未发现充分偏离证据，不能证明正态性。
print("对数数据正态族的检验统计量、校准  $p$  值：",
      result.statistic, result.pvalue)

# 正态概率图比较排序后的  $\log$  数据与理论顺序统计量中位数；近似直线支持该形状，
# 两端偏离可提示尾部差异；图形诊断和统计检验提供互补信息。
stats.probplot(np.log(data), dist="norm", plot=axes[1])
# 添加坐标标签和图例；右图标题明确使用的是对数尺度数据。
axes[0].set(xlabel="正值观测", ylabel="概率密度")
axes[0].legend()
axes[1].set(title="对数数据的正态概率图",
            xlabel="理论顺序统计量中位数", ylabel="排序后的对数观测")

# 调整布局、保存 160 dpi 图片并关闭图，便于批量运行案例时释放内存。
fig.tight_layout()
fig.savefig("e19_fit.png", dpi=160)
plt.close(fig)
```

规定环境下的实际运行输出

正态分布 参数 (np.float64(8.041507445695657), np.float64(8.671861551957466)) 赤池信息
准则 1077.7066038474904

伽马分布 参数 (1.3230785731727093, 0, np.float64(6.0778759544206995)) 赤池信息准则
922.6780873812808

对数正态分布 参数 (np.float64(0.9060096580892277), 0, np.float64(5.2662947035559355))

赤池信息准则 898.4680729944746

对数数据正态族的检验统计量、校准 p 值： 0.05251397237986777 0.403

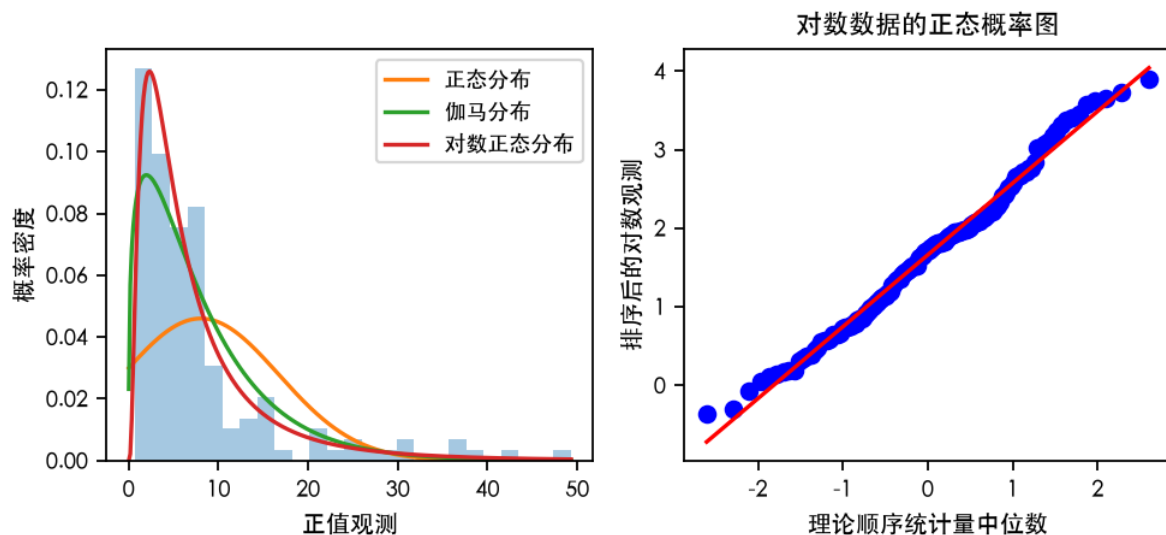


图 5-4 合成正值样本的拟合诊断，直接采用 `e19_distribution_fit.py` 运行图。种子=30 生成 150 个 $\log(X)$ 均值为 $\log(5)$ 、标准差 0.9 的观测；左图为同一原尺度数据的直方图及正态、伽马分布、对数正态拟合密度，右图为 $\log(\text{data})$ 的正态概率图。同脚本另在终端报告 AIC 及 1999 次重新拟合的 KS 校准结果，图中未绘制校准参考分布。

读图指南 左图在原尺度比较不同候选密度，右图用对数数据与正态理论分位数比较。横轴参考分布、纵轴样本尺度不同于左图；终端的校准 p 值专用于右图所对应的对数正态家族问题。

该检验针对“对数数据属于某个正态分布”这个复合原假设，不能把它的 p 值直接安到原尺度正态或伽马候选上。图左的 AIC 比较与终端的校准检验也回答不同问题：前者在候选间作相对取舍，后者衡量指定家族与数据的相容程度。把目标、数据尺度、候选参数约束和检验假设一并写进报告，才能使数值结论可复核。

5.10 从数据到结论的一份完整说明

一个完整统计结论至少要让读者知道：观察的单位是什么、样本如何取得、目标参数是什么、采用什么分布或抽样假设、估计与区间如何计算，以及数据对哪些假设提供了何种证据。这些内容不必写成繁复术语，清楚说明九个测量值来自同一正态总体的独立样本，已比仅写“运行 t 检验得到 $p=0.305$ ”传达了更多必要信息。

模拟教材中真参数通常由程序预设，因此可以把估计与真值比较；现实分析往往没有这个参照。此时要靠设计、校准、诊断、外部知识和重复研究共同判断。置信区间描述指定方法下的抽样误差，后验区间描述给定先验模型后的参数不确定性，情景区间描述改变设定后得到的结果分布。三者都可用，但概率对象不同，应使用对应名称并交代来源。

学习这些方法的目标，是能从一个具体问题返回它的模型：随机的是什么、给定的是什么、需要平均或累积的是什么、哪些独立性真正成立。公式、图与代码应围绕同一个随机对象解释。只要这个对象明确，二项计数、指数等待、均值区间与参数更新就能组织为连贯的方法，而不再是一组互不相关的命令。

5.11 本章要点 常见误区与延伸阅读

核心要点：样本是实际获得的观测，统计量是样本函数，参数描述总体。似然固定数据并比较参数支持，通常不是参数概率。极大似然、无偏性与均方误差是不同评价标准，应与目标及采样设计一起说明。

核心要点：置信区间的概率指向重复抽样方法的覆盖，后验可信区间指向给定数据后的参数概率，预测区间还包含未来观测的个体波动。t 推断、贝塔—二项与正态—正态更新各有明确模型前提；共轭方便计算，不会自动保证假设合理。

核心要点：正态均值更新把先验精度与数据精度相加。全期望给预测中心，全方差把预测方差分成已知观测噪声与后验均值不确定性。先验敏感性、已知噪声是否可信及条件独立性，都是数值结果的组成条件。

常见误区：把 p 值当原假设真概率，把未拒绝写成证明相等，把 AIC 当绝对真值评分，把 Q—Q 任一弯曲唯一解释成偏态，或在同一数据拟合参数后仍使用完全指定分布的 KS 参照。先说明参考家族与数据尺度，再解释图形和校准结果。

延伸阅读：麻省理工学院 18.05 第 10 课读总体、样本、似然与极大似然，第 15 课读贝塔与已知方差的正态共轭，第 20 课比较频率学与贝叶斯，第 22 课读正态样本均值区间；具体真

实阅读链接见 P09、P12、P18、P10。第 15 课官方文件为 https://ocw.mit.edu/courses/18-05-introduction-to-probability-and-statistics-spring-2022/mit18_05_s22_class15-prep.pdf。

诊断配读：SciPy 官方 `goodness_of_fit` 说明拟合未知参数后每份参考样本重新拟合的流程，网址 https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.goodness_of_fit.html [P13]。中心极限定理回读本书第 2 章及麻省理工学院 18.05 第 6 课乙[P08]；不要在推断章另学一套与前文无关的“大样本规则”。

5.12 练习与参考解答

练习 5-1（概念与手算，入门） 九个观测 168、172、175、169、171、173、170、174、167，用正态独立样本模型估计均值并检验 $\mu_0=170$ 。给出样本方差、t 值、95%区间和 0.05 水平结论。

练习 5-2（概念与手算，进阶） 先验贝塔分布(1,1)，固定 10 次条件独立试验成功 7 次。推导后验和后验均值，再求下一次成功预测概率。

练习 5-3（概念与手算，进阶） 先验 $\mu \sim N(50, 10^2)$ ，已知观测标准差 8，数据[52,48,55,51]。计算后验均值方差和下一次观测的预测方差，说明精度权重。

练习 5-4（概念与手算，进阶） 判断并改写三句话：“95%置信区间表示固定均值有 95%概率在当前区间”；“ $p=0.03$ 表示原假设成立概率 3%”；“任何向上弯的 Q—Q 图都表示右偏”。

练习 5-5（编程实践，入门） 运行 `e18_t_inference.py`，用手工公式与 SciPy 结果核对，并检查双侧 p 值为什么使用 `abs(t)`。

练习 5-6（编程实践，进阶） 运行 `e17_beta_update.py`，用 4001 点网格归一化先验乘似然，核对解析后验，并说明梯形积分为何不能直接用密度数组求和代替。

练习 5-7（编程实践，进阶） 用 `e36_normal_normal.py` 分别进行批量与逐条更新。给出后验区间和预测区间，作为数值验收依据。

练习 5-8（编程实践，进阶） 运行 `e19_distribution_fit.py`，记录三个候选模型的自由参数数、对数似然与 AIC，并解释校准的 KS 检验具体检验哪个家族。若 1999 次参考模拟中有 $b=1199$ 个统计量不小于观测值，模拟 p 值是多少？

练习 5-9（场景应用，入门） 教学设定：稳定测量过程噪声标准差已知为 8，独立正态观测为 $[52, 48, 55, 51]$ 。给出频率学 95% 均值区间，再与先验 $N(50, 10^2)$ 的后验区间比较。两者回答什么问题？

练习 5-10（场景应用，进阶） 教学设定：20 名独立受邀用户中 15 名完成激活，先验成功率为贝塔分布 $(2, 2)$ 。求下一名受邀用户激活概率，以及未来 10 名中激活人数的预测均值。能否把 10 次预测当成边缘独立、概率 $17/24$ 的二项试验？

练习 5-11（场景应用，进阶） 某小样本等待时间明显右偏，有一个极大值。分析者仍称“用了 t 分布就已解决非正态”，并给出很窄区间。提出两项具体检查与一个可报告的限度。

练习 5-12（场景应用，拓展） 两个候选模型 AIC 分别为 410 与 414；某拟合检验 p 值为 0.30。研究者声称前者真概率 70%、后者真概率 30%。写一段合格的模型结论，交代 Q—Q 图与外推的限度。

参考解答与验收依据

5-1 $\bar{x}=171$ ，偏差平方和 60， $S^2=7.5$ ，标准误约 0.912871。 $t \approx 1.095445$ 、自由度 8、双侧 $p \approx 0.305201$ ；95% 区间约 $[168.894916, 173.105084]$ ，包含 170，故未拒绝原假设。该结论不等于证明 μ 恰为 170。

5-2 先验核为 1，似然中与 θ 有关的部分为 $\theta^7(1-\theta)^3$ ，故后验贝塔分布 $(8, 4)$ 。后验均值 $8/12=2/3$ ；未来一次成功的预测概率 $E[\theta|\text{数据}]$ 也为 $2/3$ 。这不是说参数每次重抽，也不是把后验密度某一点当作点概率。

5-3 $\bar{x}=51.5$ ，先验精度 0.01、数据精度 $4/64=0.0625$ 。后验方差 $400/29 \approx 13.793103$ ，均值 $2975/58 \approx 51.293103$ ，先验与数据权重为 $4/29$ 与 $25/29$ 。预测方差 $64+400/29 \approx 77.793103$ ，必须包含新观测噪声。

5-4 频率学 95%描述指定方法在重复抽样中的覆盖率；给定数据后若要给参数区间概率，须说明贝叶斯先验和后验。p 值是在原假设及抽样模型下统计量至少同样极端的概率，不是原假设后验概率。Q—Q 形状须先给参考分布、坐标、位置尺度及两尾，有限样本波动也会造成弯曲，不能仅凭向上弯唯一判右偏。

5-5 均值 171、 $S \approx 2.738613$ 、 $t \approx 1.095445$ ， $2 \times \text{stats.t.sf}(\text{abs}(t), \text{df}=8)$ 约 0.305201，与程序一致。95%区间约[168.894916, 173.105084]。负 t 时若直接 sf(t)再乘 2 可能大于 1，双侧极端程度应看绝对值；n-1 自由度与 ddof=1 必须一致。

5-6 后验为贝塔分布(17,7)，均值 $17/24 \approx 0.708333$ ，95%可信区间约[0.515948, 0.867897]。网格积分应计入网格间隔；直接 sum 把密度高度当概率，量纲与总量都不对。数值后验积分应接近 1，网格均值接近解析均值；误差随网格细化通常减小，但不要求每点浮点数完全相等。

5-7 每次观测增加精度 1/64，并增加精度加权和 $x_i/64$ ；最终与批量更新 allclose。后验均值约 51.293103、标准差 3.713907，95%可信区间[44.013980, 58.572227]；预测标准差 8.820040，95%预测区间[34.006143, 68.580064]。参数区间明显更窄，不能使用同一 scale 画两者。

5-8 正态、固定 loc=0 的伽马与固定 loc=0 的对数正态均估 2 个参数，AIC 均按 $4-2\ell$ 在同一原尺度比较。右图与 KS 校准针对 $\log(\text{data})$ 的正态家族，每份参考样本都重新拟合均值和尺度。题设计数给 $(1199+1)/(1999+1)=0.6$ ；这只是计数验算，不声称程序实际输出必为 0.6。不得把这个 p 值移给原尺度正态或伽马候选。

5-9 频率区间为 $51.5 \pm 1.959964 \times 4$ ，即[43.660144, 59.339856]，构造方法在模型下有 95%覆盖。后验区间约[44.013980, 58.572227]，给定先验与数据后参数落入它的后验概率 95%。后者中心与宽度受先验影响，不能只比较区间长短就宣布方法更准确；未来单次测量区间还应更宽。

5-10 后验为贝塔分布(17,7)，下一名激活概率 17/24，未来 10 名预测均值 $10 \times 17/24 = 85/12 \approx 7.08333$ 。给定共同 Θ 后独立，积分掉 Θ 后未来结果共享参数不确定性，一般

并非边缘独立二项，而为贝塔—二项预测。其方差为 $10m(1-m)(24+10)/(24+1) \approx 2.809722$ ，较代入二项方差 $10m(1-m) \approx 2.06597$ 大。

5-11 先核对极大值是录入/单位错误还是真实长等待；再检查采样独立性、是否只记录已完成任务从而漏掉仍在等待者。小样本 t 区间的精确性依赖正态， t 分布不能自动修复偏斜、删失或相关。可说明该区间是正态独立模型下的结果，并展示包含/不包含疑似错误值的敏感性；若有右删失，应明确删失信息后用相应等待模型，不能只删掉长等待。

5-12 在同一数据尺度、可比似然及参数计数下，AIC 较小的模型按该准则更受支持，差为 4；这不是模型真概率。 $p=0.30$ 表示当前校准检验未发现足够偏离，不能证明模型真实。应同时报告支持集、估计参数、参考家族及 Q—Q 两尾偏离；若任务是极端分位数预测，还需独立验证或尾部敏感性分析，中心直线良好并不足以保证尾部外推。

第 6 章 向量空间与矩阵映射

6.1 学习路线与符号约定

入门路线：先把向量看成位移或一组测量，从线性组合和矩阵映射理解“哪些输出能达到”。进阶路线：学习子空间、基与维数，再从行内积推导四个基本子空间。拓展路线：完整读懂秩—零度证明，尝试用核解释参数能否从观测中识别。第 7 章再把这些结构变成可操作的消元和拟合方法。

本章讨论实数。 $\mathbb{R}^n$ 表示含 n 个实数分量的列向量空间； $A \in \mathbb{R}^{m \times n}$ 表示 m 行 n 列矩阵，输入在 $\mathbb{R}^n$ 、输出在 $\mathbb{R}^m$ 。上标 T 表示转置。数学推导采用列向量；NumPy 一维数组的行列角色由运算决定。概率转移的行分布约定到第 8 章另行说明。

6.2 向量 线性组合与几何图像

写 $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ 时， x 是整个向量， x_i 是第 i 个分量，下标 i 从 1 到 n 。坐标向量的相等要求各分量逐一相等。向量相加、数乘也逐分量定义： $x+y$ 的第 i 个分量是 x_i+y_i ， cx 的第 i 个分量是 cx_i 。这里 c 是一个实数，称为标量。二维和三维中可把向量画成箭头；箭头的起点可以平移，只要位移不变，它所表示的向量就不变。

向量加法可用首尾相接解释。先走 x ，再走 y ，总位移就是 $x+y$ 。数乘 $c>0$ 时保留方向并把长度乘 c ； $c<0$ 时同时反向； $c=0$ 时得到零向量。零向量记作 0 ，它的维数由上下文确定。不要把“每个分量都是零的向量”与数字零混淆，尽管它们共用同一符号。向量的分量依赖坐标系；几何对象与它的一组坐标表示也要区分。

给定 $v_1, \dots, v_k$ 和实数系数 $c_1, \dots, c_k$ ，表达式 $c_1v_1 + \dots + c_kv_k$ 称为线性组合。符号 $\sum$ 是求和号， $\sum_{i=1}^k c_i v_i$ 就是这串和的简写。线性组合允许负系数，也不要求系数相加等于 1，因此与概率混合或加权平均的限制不同。后面方程 $Ax=b$ 中的未知量 x_j ，正是用矩阵各列组合出 b 时需要寻找的系数。

在坐标中，两个同维向量的内积（也叫点积）是对应分量乘积之和： $a^T b = \sum_i a_i b_i$ ，结果是一个数。它把“沿同一方向的程度”变成可计算的量；二维直角坐标下，内积为零的非零箭头相互垂直。长度由 $\|a\|_2 = \sqrt{a^T a}$ 给出。这里先用它核对几何，后面再完整讨论内积和正交补。求和符号是 Σ ；大写希腊字母 Σ 将在第 9 章专门表示奇异值矩阵，两者不同。

$$c = 1.5a - 0.5b = 1.5 \begin{pmatrix} 2 \\ 1 \end{pmatrix} - 0.5 \begin{pmatrix} -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3.5 \\ 0.5 \end{pmatrix} \quad (6.1)$$

a 、 b 是给定二维向量， c 是它们的线性组合；等号逐坐标成立。系数 1.5 与 -0.5 没有概率含义。

例题 手算：先看组合，再核对长度

取 $a=(2,1)^T$ 、 $b=(-1,2)^T$ 。先算 $a+b=(1,3)^T$ ，再算 $1.5a=(3,1.5)^T$ 、 $-0.5b=(0.5,-1)^T$ ，因此 $c=(3.5,0.5)^T$ 。若把第二个箭头接在第一个箭头末端，终点就是 c 。

这两个已知方向的内积为 $2 \times (-1) + 1 \times 2 = 0$ ，它们相互垂直。故组合后的长度平方为 $1.5^2 \times 5 + (-0.5)^2 \times 5 = 12.5$ ，与 $3.5^2 + 0.5^2$ 相同。这里用的是勾股定理；一般非正交方向的组合还会出现交叉项，不能照搬这个平方和。

程序 `e20_vector.py` 本例取 $a=[2,1]$ 、 $b=[-1,2]$ 、 $c=1.5a-0.5b$ 。输出 $a+b=[1,3]$ 、 $c=[3.5,0.5]$ 、 $\text{dot}(a,b)=0$ 和 $\text{norm}(a) \approx 2.236068$ 。投影 $\text{projection}=[3,1.5]$ ，残差 $\text{residual}=[0.5,-1]$ ；第 7 章将从最短距离条件推导它们。程序断言既检查坐标结果，也检查 a 与残差的内积为零。

编程思路 在二维中先做线性组合，再把向量 c 分解为沿 a 的正交投影和垂直残差。用内积为零及勾股关系把图形直觉变成可以断言的数值等式。

代码对应

$c=1.5a-0.5b \rightarrow c = 1.5 * a - 0.5 * b$ ：数组按分量相乘、相减，结果仍是长度为二的坐标向量。

$\text{proj}_a(c) = [(c \cdot a)/(a \cdot a)]a \rightarrow \text{projection} = (c @ a) / (a @ a) * a$ ：内积先给出沿 a 的系数，再乘整个向量 a 。

$r = c - \text{proj}_a(c) \rightarrow \text{residual} = c - \text{projection}$: 残差与 a 正交；用内积与平方范数断言核对分解而不只看箭头。

易错提示 投影公式要求 a 非零。坐标箭头需采用等比例坐标轴，否则图上直角和长度关系会被显示比例扭曲。

正文代码 e20_vector.py

```
# 教材位置: 第 6 章, 第 6.2 节; 图 6-1。
# 单位、变量与形状: a、b、c、p、r 均为 (2,); 坐标使用统一长度单位, 内积按长度平方计。
# 学习目标: 计算二维向量的线性组合、内积、欧氏长度和正交投影,
# 并用图形与勾股关系理解投影残差。
# 核心数学关系:  $\text{proj}_a(c) = ((c \cdot a) / (a \cdot a)) * a$ ,  $\text{residual} = c - \text{proj}_a(c)$ ,
# 且  $a \cdot \text{residual} = 0$ ,  $\|c\|^2 = \|\text{proj}_a(c)\|^2 + \|\text{residual}\|^2$  (要求  $a$  非零)。
# 数据与输出: 使用两个二维向量及其线性组合, 打印和、内积、长度、
# 投影向量和残差, 保存 e20_vector.png。
# 运行方式: 安装 NumPy、Matplotlib 后, 运行 python e20_vector.py;
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# a、b 都是形状 (2,) 的一维数组, 按两个坐标表示向量;
# 它们没有显式的“行向量/列向量”第二维, @ 在这里执行向量内积。
a = np.array([2.0, 1.0])
b = np.array([-1.0, 2.0])
# 线性组合逐坐标进行:  $c = 1.5 * a - 0.5 * b = [3.5, 0.5]$ 。
# 系数的正负决定沿该向量同向或反向移动, 绝对值决定伸缩比例。
c = 1.5 * a - 0.5 * b
#  $c @ a$  是内积,  $a @ a = \|a\|^2$ ; 这个比值是投影沿  $a$  的伸缩系数。
# projection 是二维投影向量, 并不是一个“投影长度”标量;
# 沿单位方向  $a / \|a\|$  的有符号标量分量为  $(c @ a) / \|a\|$ ,
# 投影向量的长度则为  $\text{abs}(c @ a) / \|a\|$ 。此处  $a$  非零, 分母有定义。
projection = (c @ a) / (a @ a) * a
# residual 是从投影点指向  $c$  的向量; 正交投影使这部分与  $a$  垂直。
residual = c - projection
print("向量和  $a+b$  =", a + b, "; 线性组合  $c$  =", c)
# norm(a) 默认求欧氏长度  $\sqrt{a @ a}$ ; round 仅调整显示精度。
print("内积  $a^T b$  =", a @ b, "; 二范数  $\|a\|_2$  =", round(np.linalg.norm(a), 6))
print("投影向量 =", projection, "; 残差向量 =", residual)
# allclose 用来比较向量各分量, isclose 用来比较标量, 允许浮点舍入误差。
assert np.allclose(c, [3.5, 0.5])
# a 与 b 正交; 同时验证残差确实与投影所在直线的方向  $a$  正交。
assert np.isclose(a @ b, 0) and np.isclose(a @ residual, 0)
# 向量与自身的内积是长度平方; 此式就是正交分解后的勾股定理。
assert np.isclose(c @ c, projection @ projection + residual @ residual)
# 同一批已计算向量分成两个面板: 左侧解释首尾相接, 右侧解释正交分解。
# 两图直接使用 a、b、c、projection、residual, 不另外输入一套向量数值。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1))
for ax in axes:
    # equal 让两个坐标轴单位长度一致, 否则图上的直角与长度比例可能失真。
```

```

ax.set_aspect("equal")
ax.grid(alpha=0.3)
ax.axhline(0, color="gray", lw=0.5)
ax.axvline(0, color="gray", lw=0.5)
ax.set(xlim=(-1.6, 4.4), ylim=(-0.5, 3), xlabel="$x_1$", ylabel="$x_2$")
# xy 是箭头终点, xytext 是起点; 空字符串表示只画箭头, 不附加正文。
# 向量平移不改变其分量; 第二条组合箭头从 projection 走到 c, 位移为 residual。
ax = axes[0]
for start, end, label, color in [
    (np.zeros(2), a, "$a$", "gray"),
    (np.zeros(2), b, "$b$", "gray"),
    (np.zeros(2), projection, "$1.5a$", "tab:green"),
    (projection, c, "$-0.5b$", "tab:orange"),
    (np.zeros(2), c, "$c$", "tab:blue"),
]:
    ax.annotate("", xy=end, xytext=start, arrowprops={"arrowstyle": "->",
        "color": color})
    # end[0]、end[1] 分别是横纵坐标; 稍微偏移标签, 避免遮住箭头尖端。
    if label == "$-0.5b$":
        ax.text(end[0] + 0.08, end[1] - 0.35, label, color=color)
    else:
        ax.text(end[0] + 0.08, end[1] + 0.08, label, color=color)
ax.set_title("向量的线性组合")
ax = axes[1]
# 每次取一个向量和对应标签, 统一从原点画箭头, 便于比较方向和长度。
for v, label, color in [(c, "$c$", "tab:blue"), (projection, "$p$", "tab:green")]:
    ax.annotate("", xy=v, xytext=(0, 0), arrowprops={"arrowstyle": "->",
        "color": color})
    ax.text(v[0] + 0.08, v[1] + 0.08, label, color=color)
# 红色虚线连接 projection 与 c, 其位移正好是 residual;
# 残差箭头虽未从原点画出, 但向量只取决于位移, 平移不会改变它。
ax.plot([projection[0], c[0]], [projection[1], c[1]], "r--", label="残差 $r$")
ax.annotate("", xy=c, xytext=projection, arrowprops={"arrowstyle": "->",
    "color": "red"})
# 沿 a 和 residual 的单位方向画直角记号; 本例两方向均非零且正交。
u = a / np.linalg.norm(a)
w = residual / np.linalg.norm(residual)
right_angle = np.array([projection - 0.18 * u,
    projection - 0.18 * u + 0.18 * w,
    projection + 0.18 * w])
ax.plot(*right_angle.T, color="gray", lw=0.8)
ax.set_title("向量的正交投影")
ax.text(0.02, 0.04, "正交投影 p; $a^T r=0$", transform=ax.transAxes, fontsize=8)
ax.legend(fontsize=8, loc="upper left")
# 指定显示范围, 保证所有箭头和标签位于图内。
# 调整布局后保存并关闭, 避免批量作图时保留多余图对象。
fig.tight_layout()
fig.savefig("e20_vector.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

规定环境下的实际运行输出

向量和 $a+b = [1. \ 3.]$; 线性组合 $c = [3.5 \ 0.5]$

内积 $a^T b = 0.0$; 二范数 $\|a\|_2 = 2.236068$

投影向量 = $[3. \ 1.5]$; 残差向量 = $[0.5 \ -1.]$

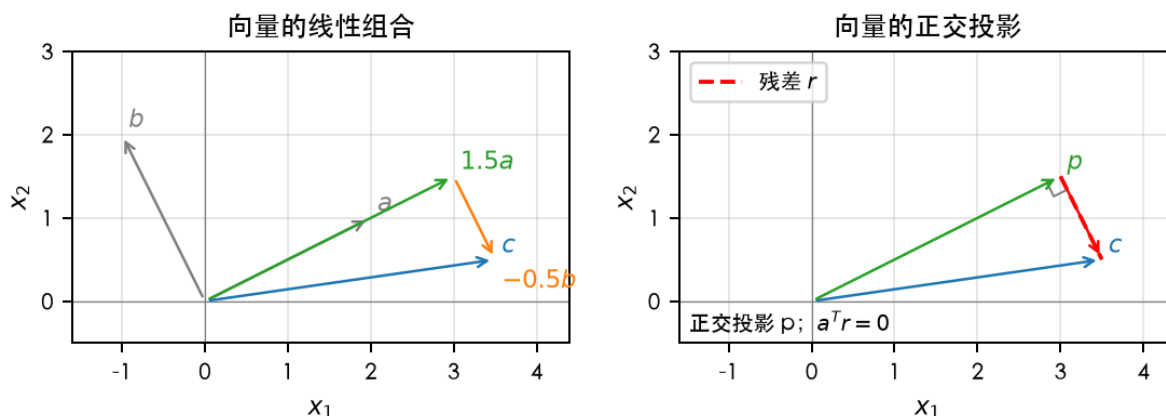


图 6-1 e20_vector.py 的同一组向量。左图按首尾相接画出 $1.5a$ 与 $-0.5b$ ；右图把 c 分解为沿 a 的投影 $p=(3,1.5)^T$ 与正交残差 $r=(0.5,-1)^T$ 。两图中 c 始终是 $(3.5,0.5)^T$ 。

读图指南 先看左图从原点到 $1.5a$ 再接 $-0.5b$ 如何到达同一个 c ；再看右图 p 与 r 首尾相接，直角记号表示 $a^T r = 0$ 。两图中的 c 坐标相同，投影不是另换了目标向量。

6.3 矩阵乘法与线性映射

矩阵 $A=(a_{ij}) \in \mathbb{R}^{m \times n}$ 有 m 行、 n 列。元素 a_{ij} 位于第 i 行第 j 列，先报行、再报列。把 A 的第 j 列记作 a_j ，则 $A=[a_1 \cdots a_n]$ 。对输入 $x \in \mathbb{R}^n$ ，输出 Ax 是 $\mathbb{R}^m$ 中的向量。它的第 i 个分量等于第 i 行与 x 的内积，而整个输出也等于各列按 x 的坐标进行线性组合。这两个看法使用同一个公式，分别强调“每条约束”和“所有可达到的输出”。

映射 $T: \mathbb{R}^n \rightarrow \mathbb{R}^m$ 若满足 $T(u+v)=T(u)+T(v)$ 与 $T(cu)=cT(u)$ ，就称为线性映射。箭头 $\rightarrow$ 表示从输入集合到输出集合； u, v 是任意输入向量， c 是任意实数。矩阵映射 $T(x)=Ax$ 一定满足这两条。反过来，取标准基 e_j ，即第 j 个分量为 1、其余为 0 的向量，任何 x 都可写成 $\sum x_j e_j$ ，于是 $T(x)=\sum x_j T(e_j)$ 。因此把 $T(e_j)$ 依次排成矩阵各列，就得到 T 的矩阵表示。

如果 B 先把 $\mathbb{R}^q$ 映到 $\mathbb{R}^n$ ， A 再把 $\mathbb{R}^n$ 映到 $\mathbb{R}^m$ ，那么复合结果是 $A(Bx)=(AB)x$ 。因此 A 为 $m \times n$ 、 B 为 $n \times q$ 时， AB 为 $m \times q$ ，内侧尺寸 n 必须相同。矩阵乘法通常不交换， AB 与 BA 即使都有定义，也未必相等。转置则把行列互换： $(A^T)_{ji}=a_{ij}$ ，并且 $(AB)^T=B^T A^T$ 。顺序颠倒是由复合顺序产生的，不是记忆口诀的偶然安排。

$$Ax = \sum_{j=1}^n x_j a_j, \quad (AB)_{ij} = \sum_{k=1}^n a_{ik} b_{kj} \quad (6.2)$$

a_j 是 A 的第 j 列, x_j 是输入的第 j 个坐标; $(AB)_{ij}$ 是乘积第 i 行第 j 列的元素。第二式中的 k 是求和用的临时下标, 不是矩阵尺寸。

单位矩阵 I_n 的主对角线上为 1, 其余元素为 0, 作用是保持每个输入不变。若 A 为 $n \times n$ 且存在 B 使 $AB=BA=I_n$, 则 B 称为 A 的逆, 记作 A^{-1} 。逆表示把变换撤回去, 而不是对每个元素取倒数。例如剪切矩阵的逆是沿相反方向剪切。只有方阵才使用这里的双侧逆定义; 矩形矩阵可能有左逆或右逆, 第 9 章还会用伪逆统一处理信息缺失与冗余。

例题 手算: 变换的次序有实际含义

令 R 为逆时针 90° 旋转, $R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$; 令 $S = \text{diag}(2, 1)$, 表示横向伸长两倍、纵向保持不变。对 $e_1 = (1, 0)^T$, 先旋转再伸缩得到 $SR e_1 = (0, 1)^T$; 先伸缩再旋转得到 $RS e_1 = (0, 2)^T$, 所以 $SR \neq RS$ 。

这里 $\text{diag}(2, 1)$ 表示以 2 和 1 作为对角元素的矩阵。两种变换都保持原点, 但“加上一个固定平移向量”的映射 $T(x) = Ax + d$ 在 $d \neq 0$ 时不线性, 因为 $T(0) = d \neq 0$ 。这是检查线性最迅速的必要条件, 满足 $T(0) = 0$ 本身却还不足以证明线性。

程序 `e21_transforms.py` 程序把同一个闭合正方形分别交给 I 、 30° 旋转 R 、 $S = \text{diag}(2, 0.6)$ 及 $H = \begin{bmatrix} 1 & 0.8 \\ 0 & 1 \end{bmatrix}$ 。四个行列式 (第 7 章解释) 为 $[1, 1, 1.2, 1]$ 。四种变换各自作用于原正方形。输入 `points` 每行保存一个顶点, 所以用 `points@M.T` 对应数学中的列向量 Mx 。旋转满足 $R.T@R=I$, 剪切虽然面积不变却通常改变长度和角度。

编程思路 把正方形的顶点写成行向量数组, 分别施加旋转、缩放和剪切, 借行列式解释面积变化。再比较先旋转后缩放与相反顺序, 以同一组顶点说明矩阵乘法一般不交换。

代码对应

构造旋转角 30° 的二维旋转矩阵 $\rightarrow R = \text{np.array}(\begin{bmatrix} \text{np.cos}(\text{theta}), & -\text{np.sin}(\text{theta}) \\ \text{np.sin}(\text{theta}), & \text{np.cos}(\text{theta}) \end{bmatrix})$: `cos`、`sin` 的位置与符号按列向量变换约定填写, `theta` 以弧度传入。

分别把横纵方向伸缩 2 和 0.6 倍 $\rightarrow S = \text{np.diag}([2.0, 0.6])$: 对角矩阵的行列式 1.2 是有向面积倍率。

对每个顶点执行列向量公式 $p' = Mp \rightarrow \text{transformed} = \text{points} @ M.T$: 顶点在数组中按行保存, 故批量实现写成 $\text{points}@M.T$ 。

易错提示 数组按行存储不改变数学中列向量公式的含义, 转置负责对齐两种记法。 $S @ R$ 表示先 R 后 S, 读乘法顺序时应从右往左。

正文代码 e21_transforms.py

```
# 教材位置: 第 6 章, 第 6.3 节; 图 6-2。
# 单位、变量与形状: 各变换矩阵均为 (2,2); 坐标使用统一长度单位, 变换系数无量纲。
# 学习目标: 把二维矩阵看作旋转、缩放、剪切等线性变换,
# 理解行列式的面积意义, 以及矩阵复合时顺序为何重要。
# 核心数学关系: 列向量变换为  $y=Mx$ ; 若多个点按行存放, 统一变换写作
#  $\text{points} @ M.T$ ; 面积放大倍数为  $\text{abs}(\text{det}(M))$ , 一般有  $S @ R \neq R @ S$ 。
# 数据与输出: 对中心位于原点的正方形应用四种变换, 打印行列式,
# 保存 e21_transforms.png; 所有矩阵都是形状 (2,2)。
# 运行方式: 安装 NumPy、Matplotlib 后, 运行 python e21_transforms.py;
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 三角函数使用弧度, 因此先把 30 度转换为  $\pi/6$ 。
theta = np.deg2rad(30)
# 标准二维旋转矩阵把列向量逆时针旋转 theta, 保持长度和夹角不变。
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
# diag 从一维元素生成对角矩阵: 横坐标放大 2 倍, 纵坐标缩小为 0.6 倍。
# 两方向伸缩不同, 因此一般不保持角度, 也不会与旋转矩阵交换。
S = np.diag([2.0, 0.6])
# 剪切给出  $x'=x+0.8*y$ 、 $y'=y$ ; 水平移动量随原来的高度 y 变化。
H = np.array([[1.0, 0.8], [0.0, 1.0]])
# eye(2) 是单位矩阵, 对应“保持原图”; 两个列表按相同顺序配对。
matrices = [np.eye(2), R, S, H]
names = ["恒等变换", "旋转变换", "缩放变换", "剪切变换"]
# 五行坐标包含四个顶点和重复的起点, 确保轮廓闭合, 形状为 (5,2)。
# 标量 0.5 广播到所有坐标, 使方形中心从 (0.5,0.5) 平移到原点。
# 这一步是构造输入图形; 后续 M 本身仍是关于原点的线性变换。
points = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]]) - 0.5
# 列表推导对每个矩阵求行列式; round 仅用于让打印结果更易读。
# det 的绝对值是面积倍数, 符号表示是否翻转方向; 本例行列式都为正。
print("各变换矩阵的行列式 =", np.round([np.linalg.det(M) for M in matrices], 6))
# R.T 表示转置;  $R.T @ R = I$  验证旋转矩阵的列是单位正交向量。
assert np.allclose(R.T @ R, np.eye(2))
# 单位变换/旋转/剪切保持面积; 缩放的面积倍数为  $2*0.6=1.2$ 。
assert np.allclose([np.linalg.det(M) for M in matrices], [1, 1, 1.2, 1])
# 在列向量约定下  $S @ R$  表示先旋转后缩放;  $R @ S$  表示先缩放后旋转。
```

```
# 这两个结果不同，说明矩阵乘法通常不满足交换律。
assert not np.allclose(S @ R, R @ S)
fig, axes = plt.subplots(1, 4, figsize=(6.3, 2.8))
# zip 按位置把坐标轴、矩阵、名称组合起来，逐一绘制四种变换。
for ax, M, name in zip(axes, matrices, names):
    # 每个点存为一行 p，列向量公式  $M @ p$  对应的行形式是  $p @ M.T$ 。
    # (5,2) @ (2,2) 得到 (5,2)，一次完成五个点的同一变换。
    transformed = points @ M.T
    # [:,0] 选所有点的横坐标，[:,1] 选纵坐标；黑色虚线作原图参照。
    ax.plot(points[:, 0], points[:, 1], "k--", alpha=0.4)
    # fill 把变换后的顶点连成填充多边形，透明度便于观察和原图的重叠。
    ax.fill(transformed[:, 0], transformed[:, 1], alpha=0.4)
    # 四个子图使用同样的范围，避免自动缩放掩盖真实伸缩差异。
    ax.set(xlim=(-1.5, 1.5), ylim=(-1.5, 1.5), title=name)
    # 坐标轴等比例显示，才能直观看出旋转的角度与形状是否保持。
    ax.set_aspect("equal")
    ax.grid(alpha=0.2)
# 保存 180 dpi 图片并关闭图，释放资源。
fig.tight_layout()
fig.savefig("e21_transforms.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

各变换矩阵的行列式 = $[1. \quad 1. \quad 1.2 \quad 1.]$

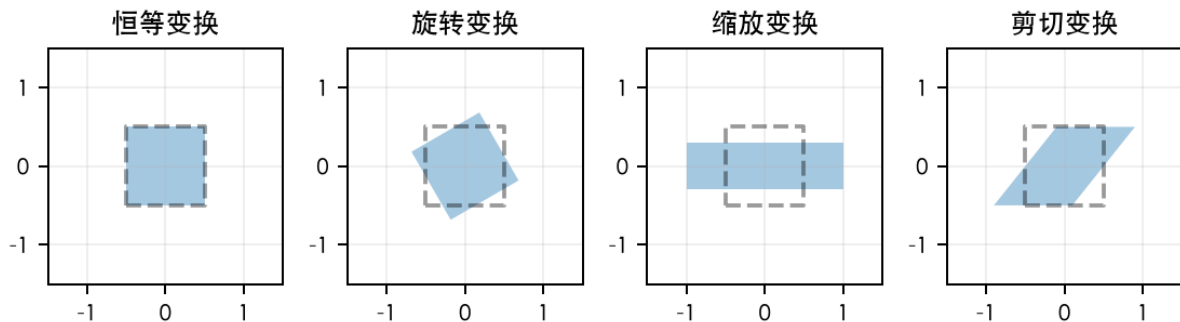


图 6-2 同一正方形分别经恒等、 30° 旋转、沿轴缩放(2,0.6)和剪切 0.8。虚线是原形状，填色是各自输出；行列式绝对值只报告面积倍率，不能概括边长和角度。

读图指南 把每个填色形状与本格灰色虚线原正方形比较：旋转保持边长和角，缩放分别改变两轴，剪切使直角改变。四格分别施加各自矩阵，并不是连续四步。

6.4 数学对象与数组形状

数学里的列向量有 $n \times 1$ 个位置；NumPy 的一维数组形状(n,)只有一条轴，因此 $v.T$ 仍是一维，不会自动变成列。显式列数组使用 $v[:,None]$ ，形状(n,1)；显式行数组使用 $v[None,:]$ ，形状(1,n)。圆括号中的数字报告各轴长度，末尾逗号表示只有一项的形状元组。转置二维数组会

交换两轴，一维数组只有一轴，无从交换。把这一区别写清，可以避免一段运算虽然能运行，所得对象却并非预期的内积或外积。

矩阵乘法使用`@`，逐元素乘法使用`*`。若 A 、 B 均为 2×2 ， $A @ B$ 按“行乘列求和”， $A * B$ 则只是对应位置相乘。两个形状 $(n,)$ 的数组 $v @ w$ 得到标量内积；形状 $(n,1)$ 乘形状 $(1,n)$ 得到 $n \times n$ 外积。内积压缩一个坐标方向，外积保留两组下标，它们回答不同的问题。不能仅凭变量名写着 `vector` 或 `matrix` 判断运算，必须查看 `shape`。

广播是逐元素运算的尺寸规则：从最后一轴向前比较，每一对尺寸必须相同，或其中一个等于 1；缺少的前轴按 1 处理。例如 X 形状 $(3,2)$ ，`row` 形状 $(2,)$ ， $X + \text{row}$ 给每一行加同一组两项。形状 $(3,)$ 的数组不能直接加到 X 上，因为尾轴 3 与 2 不兼容。若它表示逐行校正量，应写成 $(3,1)$ 。广播通常不需要先完整复制输入，但结果数组仍占相应内存。

程序 `e22_arrays.py` 原矩阵 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ 、 $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 给出 $A @ B = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$ ， $A * B = \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}$ 。当 $v = [1, 2]$ 时， $A @ v = [5, 11]$ ， $v @ A = [7, 10]$ ，次序不同体现行列角色。 $X = \text{ones}((3,2))$ 加 `row = [10, 20]`得到三行 $[11, 21]$ 。程序还展示切片视图会修改原数组、`copy` 得到独立副本；这属于数据存储行为，不是新的线性代数定理。

编程思路 先以两个小矩阵直接比较矩阵乘积与逐元素乘积，再用向量形状、广播以及切片共享数据说明 NumPy 的数组规则。每一步都显示形状或实际值，让运算语义由可见结果核对。

代码对应

用 2×2 数组作为乘法示例 $\rightarrow A = \text{np.array}([[1, 2], [3, 4]])$ ： $A @ B$ 对应行乘列， $A * B$ 则是同位置元素相乘，二者的数学对象不同。

给 3×2 矩阵的每行加上同一二维偏移 $\rightarrow \text{row} = \text{np.array}([10, 20])$ ：长度为二的 `row` 从末尾维对齐并广播，等价于手算三行分别加 $[10, 20]$ 。

取 `base` 的第 2 至 4 号索引形成视图 $\rightarrow \text{view} = \text{base}[2:5]$ ：基本切片共享底层存储，随后修改 `view` 会改变 `base`。

建立与 base 脱离存储关系的副本 → `separate = base[2:5].copy()`：copy 之后修改 separate 不再改变原数组。

易错提示 一维 `v` 的形状是 `(2,)`，`v.T` 仍是一维；列向量要显式写 `v[:,None]`。广播只从末尾维匹配且各维须相等或有一方为一，不能靠结果看似合理猜测方向。

正文代码 e22_arrays.py

```
# 教材位置：第 6 章，第 6.4 节；图 6-3。
# 单位、变量与形状：v 为 (2,)，X 为 (3,2)，row 为 (2,)；数组元素是无量纲教学数字。
# 学习目标：区分矩阵乘法与逐元素运算，理解一维数组的形状、广播规则，
# 并通过切片视图和 copy 学会判断数组是否共享底层数据。
# 核心规则：@ 按线性代数的求和规则相乘，* 按对应元素相乘；
# 广播从最右侧维度对齐，每一对维度须相等或其中之一为 1。
# 数据与输出：使用 2×2 矩阵、长度为 2 的向量和 3×2 数组，打印各种运算，
# 展示不兼容广播被拒绝，以及修改视图如何改变原数组；保存 e22_arrays.png。
# 运行方式：安装 NumPy 后，在终端运行 python e22_arrays.py。

import numpy as np
# A、B 的形状都是 (2,2)；B 是交换两个坐标的矩阵。
A = np.array([[1, 2], [3, 4]])
B = np.array([[0, 1], [1, 0]])
# v 的 shape 为 (2,)，只有一个轴，并不是形状 (1,2) 或 (2,1) 的二维数组。
v = np.array([1.0, 2.0])
# A@B 的 (i,j) 元素是 A 的第 i 行与 B 的第 j 列的内积；
# A*B 只把相同位置的数字相乘。本例右乘 B 会交换 A 的两列。
# sep="\n" 让本次 print 的多个参数分行显示，便于比较两种结果。
print("A@B =", A @ B, "; A*B =", A * B, sep="\n")
# A@v 计算各行与 v 的内积，等价于把 v 当列向量；
# v@A 计算 v 与各列的内积，等价于把 v 当行向量；结果均为形状 (2,)。
print("A@v =", A @ v, "; v@A =", v @ A)
# 一维数组转置仍为 (2,)；[:,None] 在第二轴新增长度为 1 的维度，
# 才得到真正的二维列形数组 (2,1)，其中 None 等价于 np.newaxis。
print("数组形状：", v.shape, v.T.shape, v[:, None].shape)
# array_equal 检查形状与元素完全一致，适合此处确定的整数计算。
assert np.array_equal(A @ B, [[2, 1], [4, 3]])
assert not np.array_equal(A @ B, A * B)
assert v.T.shape == (2,) and v[:, None].shape == (2, 1)
# X 形状 (3,2)，row 形状 (2,)；广播时将 row 视作 (1,2)，
# 第一轴的 1 可扩展为 3，相同的两项偏移便应用到每一行。
X = np.ones((3, 2))
row = np.array([10, 20])
print("X+row =", X + row)
# [[11,21]]*3 在这里是 Python 列表重复，构造三行相同的预期结果，
# 并不是 NumPy 数组的数值乘法；allclose 将其与计算结果逐元素比较。
assert np.allclose(X + row, [[11, 21]] * 3)
# 用一个预期失败的例子认识广播边界；try/except 捕获错误后程序可继续。
try:
    # arange(3) 的形状是 (3,)。从右对齐后遇到 X 的末维 2 与 3，
    # 二者不相等且都不是 1，所以无法相加；它不会自动猜测你想按列加。
    X + np.arange(3)
except ValueError:
    # NumPy 用 ValueError 报告形状不兼容；捕获到它说明此项自检符合预期。
    print("已拒绝不兼容的数组形状")
```

```

else:
    # try 没有抛错才会执行 else; 若不兼容运算竟然成功, 就主动报告自检失败。
    raise AssertionError("此形状的广播应失败")
# arange(6.0) 创建浮点数组 [0,1,2,3,4,5]。基本切片 [2:5] 包含下标 2、3、4,
# 得到与 base 共享内存的视图; view[0] 对应 base[2], 因此赋值会改动 base。
base = np.arange(6.0)
view = base[2:5]
view[0] = 99
# copy() 显式复制数据; separate 修改自己的第一个元素, 不再影响 base。
separate = base[2:5].copy()
separate[0] = -1
# 同时确认原数组保留视图写入的 99, 而副本可独立改成 -1。
assert base[2] == 99 and separate[0] == -1
print("修改视图、复制后, 原数组 =", base)

# 配套图直接画上面实际使用的 v、X、row; 这里不改 v 的长度, 也不把加法换成减法。
# Matplotlib 只负责画图, 不改变 NumPy 运算或上面已经打印的数值输出。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0))
ax = axes[0]
ax.axis("off")
# 用文本方框展示同一份 v 的三种数组形状; None 只插轴, 不改变元素顺序。
for y_position, label, text in [
    (0.82, "$v$: (2,)", str(v)),
    (0.52, "$v[None,:]: (1,2)", str(v[None, :])),
    (0.16, "$v[:,None]: (2,1)", str(v[:, None])),
]:
    ax.text(0.02, y_position, label, transform=ax.transAxes, fontsize=9, va="center")
    ax.text(0.57, y_position, text, transform=ax.transAxes, fontsize=10,
           family="monospace", va="center", bbox={"facecolor": "#e7f0f7",
           "edgecolor": "#52758c"})
ax.set_title("同一数组的三种形状")
ax = axes[1]
ax.axis("off")
# X+row 中 row 形状为(2,), 广播到 X 的每一行; 此处数值直接读取实际运算。
assert X.shape == (3, 2) and row.shape == (2,)
broadcast_result = X + row
ax.text(0.03, 0.78, "$X$: (3,2)\n" + str(X), family="monospace", fontsize=9,
       transform=ax.transAxes, va="center")
ax.text(0.03, 0.42, "$row$: (2,)\n" + str(row), family="monospace", fontsize=9,
       transform=ax.transAxes, va="center")
ax.text(0.56, 0.62, "$X+row$: (3,2)\n" + str(broadcast_result), family="monospace",
       fontsize=9,
       transform=ax.transAxes, va="center", bbox={"facecolor": "#e7f0f7",
       "edgecolor": "#52758c"})
# arange(3) 的尾轴 3 与 X 的尾轴 2 不兼容, 图中保留已验证的异常规则。
ax.text(0.03, 0.12, "(3,2) + (3,) -> ValueError", transform=ax.transAxes,
       fontsize=8)
ax.set_title("广播加法: 从末轴对齐")
fig.tight_layout()
fig.savefig("e22_arrays.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```


规定环境下的实际运行输出

```
A@B =
[[2 1]
 [4 3]]
; A*B =
[[0 2]
 [3 0]]
A@v = [ 5. 11.] ; v@A = [ 7. 10.]
数组形状: (2,) (2,) (2, 1)
X+row = [[11. 21.]
 [11. 21.]
 [11. 21.]]
```

已拒绝不兼容的数组形状

修改视图、复制后, 原数组 = [0. 1. 99. 3. 4. 5.]

同一数组的三种形状

广播加法: 从未轴对齐

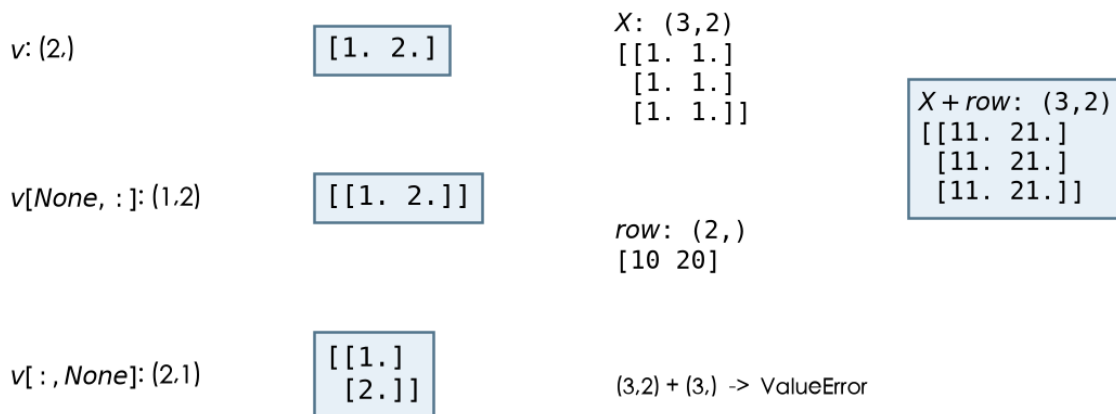


图 6-3 $v=[1,2]$ 的一维、行与列形状分别为 $(2,)$ 、 $(1,2)$ 、 $(2,1)$; 右图显示 3×2 的 X 与长度 2 的 row 相加。图中的实际广播结果为三行 $[11,21]$ 。

读图指南 按左列自上而下对照同一数组的一维、行与列形状, 数值顺序没有改变; 右列从 X 与 row 的尾轴匹配读到三行相同的广播结果。最下方不相容形状提示广播的边界。

6.5 子空间 封闭性与反例

空间首先表示允许我们进行运算的全部对象。 $\mathbb{R}^2$ 中的一个向量可以是平面位移, 也可以是两项测量组成的数据; 只要采用相同的加法与数乘, 就具有同一种线性结构。所谓子空间, 是在大空间中选出一部分向量后, 仍能不受限制地进行这两种运算。它不要求“比原空间小一维”: $\{0\}$ 和整个 $\mathbb{R}^n$ 也都是子空间。

非空集合 $W \subseteq \mathbb{R}^n$ 称为线性子空间，如果任取 $u, v \in W$ 和实数 α, β ，都有 $\alpha u + \beta v \in W$ 。这叫对线性组合封闭。它等价于分别检查加法封闭与实数数乘封闭：取 $\alpha = \beta = 1$ 得到加法，取 $\beta = 0$ 得到数乘；反过来，分别数乘后再相加即可。非空不能省略，否则空集会使“任取”形式的陈述空真。

判定时可先找零向量：若 $u \in W$ ，封闭性给出 $0u = 0 \in W$ 。找不到零向量即可否定，但含零并不足够；还要检查所有向量、所有实数系数。检验有限个数值只能反驳一般命题，通常不能证明无限集合封闭。证明要把任意 u, v 代入集合定义，推出 $\alpha u + \beta v$ 仍满足该定义。

例题 反例：三种常见图形为什么不是子空间

直线 $L = \{(x, y)^T : x + y = 1\}$ 不含 $(0, 0)^T$ ，所以不是子空间。它可以写成 $(1, 0)^T + t(-1, 1)^T$ ， t 为任意实数；这种“子空间平移后得到的集合”称仿射集合。平移后的直线仍是一条直线，但不再保持数乘封闭。

单位圆 $C = \{(x, y)^T : x^2 + y^2 = 1\}$ 不含零，且 $(1, 0)^T \in C$ 而 $2(1, 0)^T \notin C$ 。它的线性张成是整个平面，但圆周本身不是子空间；一个集合和它的张成空间必须分开。

闭第一象限 $Q = \{(x, y)^T : x \geq 0, y \geq 0\}$ 包含零，也对相加封闭；取 $u = (1, 1)^T$ ，乘 -1 却离开 Q ，所以它不是实线性子空间。若只检查正数倍，就会把锥形区域误当子空间。

例题 证明：过原点的直线与齐次约束平面

$W = \{(x, y)^T : x + y = 0\}$ 中的向量都可写成 $t(1, -1)^T$ 。两个这样的向量任意线性组合仍是该方向的倍数，所以是子空间。一般地，任意固定非零 v 的所有实数倍组成一维子空间。

$H = \{(x, y, z)^T : z = x + y\}$ 也是子空间。若 $u_3 = u_1 + u_2$ 、 $v_3 = v_1 + v_2$ ，则 $(\alpha u + \beta v)_3 = \alpha u_3 + \beta v_3 = (\alpha u + \beta v)_1 + (\alpha u + \beta v)_2$ 。非齐次条件 $z = x + y + 1$ 则不含零。齐次与非齐次的区别由此直接体现在线性结构上。

给定 $v_1, \dots, v_k$ ，它们的张成空间 $\text{span}\{v_1, \dots, v_k\}$ 是所有有限线性组合的集合。两组线性组合相加或数乘后仍能整理成同一组向量的线性组合，因此 span 一定是子空间。它还是包含这组

向量的最小子空间：任何包含各 v_i 并对线性组合封闭的子空间，都必须包含这些组合。这里的“最小”按集合包含关系理解。

6.6 线性无关 基 维数与坐标

张成回答“够不够用”，线性无关回答“有没有重复方向”。向量组 $v_1, \dots, v_k$ 线性无关，是说 $c_1 v_1 + \dots + c_k v_k = 0$ 只能由全部 $c_i = 0$ 实现。若存在一组不全为零的系数使组合为零，就称线性相关。相关式中某个非零系数对应的向量，可以用其余向量表示，所以它没有增加新的方向。

空间 W 的一组基必须同时满足无关与张成。无关保证表示唯一，张成保证表示存在：若 w 有两组系数表示，两式相减就是零组合，无关迫使对应系数相等。任意基的向量数相同，称为维数 $\dim(W)$ 。零空间 $\{0\}$ 的维数是 0，其基是空组；不能把“零向量这一项”当作基。维数计的是独立方向数，而不是向量写了几个坐标。三维空间里的一条线仍只有 1 维。

为什么所有基都含相同数量的向量？一组无关向量 $u_1, \dots, u_k$ 若位于 r 个向量 $v_1, \dots, v_r$ 的张成中，可依次用 u 替换某个有非零表示系数的 v 而保持张成：先放入 u_1 ，再放入不在 u_1 张成中的 u_2 ，如此继续。无关性保证每一步仍能找到尚未替换的 v ，最多能替换 r 次，所以 $k \leq r$ 。对两组基互换角色便得两向量数彼此不超过，因而相等。这使维数不依赖所选基。

$$[w]_B = \begin{pmatrix} c_1 \\ \vdots \\ c_r \end{pmatrix}, \quad w = c_1 v_1 + \dots + c_r v_r \quad (6.3)$$

$B=(v_1, \dots, v_r)$ 是 W 的一组有序基； $[w]_B$ 表示 w 在该基下的坐标列， c_i 为唯一组合系数。坐标列属于 $\mathbb{R}^r$ ，而 w 本身可能属于 $\mathbb{R}^n$ ， r 不必等于 n 。

例题 手算：一条平面内的基与坐标

在 $\mathbb{R}^3$ 中取 $v_1=(1,0,1)^T$ 、 $v_2=(0,1,1)^T$ 。若 $av_1+bv_2=0$ ，前两坐标直接给出 $a=b=0$ ，因此无关。它们张成所有 $(a,b,a+b)^T$ ，恰是平面 $z=x+y$ ，因此构成该平面的一组基，维数为 2。

对 $w=(2,-1,1)^T$ ，有 $w=2v_1-v_2$ ，所以 $[w]_B=(2,-1)^T$ 。 w 写在三个标准坐标中，基坐标却只有两个；这不矛盾，因为第三个标准坐标被平面方程约束。若换一组平面基， w 的位置不变，坐标数字会变。

基也可以从一组候选向量中筛选：依次加入不在已有张成空间中的向量，直到所有候选都能表示。第 7 章会把这个判断化为消元计算；保留原矩阵的主元列可得原列空间的基。行化简虽保持各行张成的集合，却会改变列向量的具体位置，不能直接拿化简后矩阵的主元列充当原列空间的基。

6.7 内积 正交与正交补

内积在几何中衡量方向关系，在数据中可衡量两列中心化变化的共同起伏。实向量 $u, v \in \mathbb{R}^n$ 的欧氏内积为 $\langle u, v \rangle = u^T v = \sum u_i v_i$ 。内积对每个参数线性，满足对称性 $u^T v = v^T u$ ，且 $u^T u \geq 0$ ，等号仅在 $u=0$ 时成立。于是 $\|u\|_2 = \sqrt{u^T u}$ 可作为长度。

对两个非零向量，夹角 θ 满足 $\cos \theta = (u^T v) / (\|u\|_2 \|v\|_2)$ 。柯西—施瓦茨不等式 $|u^T v| \leq \|u\|_2 \|v\|_2$ 保证比值在 $[-1, 1]$ 内。它可从任意实数 t 都使 $\|u - tv\|_2^2 \geq 0$ 推出：当 $v \neq 0$ 时取 $t = (u^T v) / (v^T v)$ ，整理得到 $(u^T v)^2 \leq (u^T u)(v^T v)$ 。零向量没有确定夹角，但与每个向量的内积都为零。

$u^T v = 0$ 时称 u 与 v 正交。直角坐标中的两个非零箭头正交就垂直；在高维中，内积为零仍是可计算的定义。若 $u \perp v$ ，则 $\|u+v\|_2^2 = \|u\|_2^2 + \|v\|_2^2$ ，因为展开中的交叉项 $2u^T v$ 为零。这是勾股定理的代数形式，也是之后投影、最小二乘和 SVD 误差分解的共同依据。

例题 手算：方向相反、正交与相关变化

$u = (1, 2, -1)^T$ 、 $v = (2, 0, 2)^T$ 时， $u^T v = 2 - 2 = 0$ ， $\|u\|_2^2 = 6$ 、 $\|v\|_2^2 = 8$ ，而 $\|u+v\|_2^2 = 14$ 。把 v 改为 $-u$ 时内积为 -6 ，表示反向，不能把负内积说成“负长度”。

若 u, v 是两个变量在同一批样本中的中心化观测列，它们的样本协方差为 $u^T v / (N-1)$ 。内积为零在此表示样本不相关。这个代数事实不能自动证明随机变量独立，需与第 2 章的概率定义区别。

给定子空间 $W \subseteq \mathbb{R}^n$ ，正交补 $W^\perp$ 定义为所有与 W 内每个向量都正交的向量组成的集合。若 $w_1, \dots, w_r$ 是 W 的一组基，只需检查 $x^T w_i = 0$ 对每个 i 成立，因为 x 与任意线性组合的内积也为零。只与 W 中一个任意向量垂直则通常不够。 $W^\perp$ 对线性组合封闭，因此也是子空间。

有限维欧氏空间里，可以把 W 的基逐个减去已选单位方向上的分量，再归一化，得到标准正交基；第 7 章详细给出格拉姆—施密特算法。若尚未张成整个 $\mathbb{R}^n$ ，就选一个不在已有张成空间中的向量，用同样方法补入。直到得到 n 个标准正交方向。前 r 个张成 W ，其余 $n-r$ 个恰张成 $W^\perp$ ：与前 r 个方向的坐标全为零，正是属于正交补的条件。因此 $\dim(W^\perp)=n-\dim(W)$ 。

$$\mathbb{R}^n = W \oplus W^\perp, \quad x = w + z, \quad w \in W, z \in W^\perp \quad (6.4)$$

$\oplus$ 表示直和：每个 x 都有且只有一种上述分解；这里两部分还互相正交。 n 是母空间维数， $\dim(W)$ 与 $\dim(W^\perp)$ 之和为 n 。

存在性来自把 x 在这 n 个标准正交方向上的坐标分成前后两组。唯一性可以直接证明：若 $x=w_1+z_1=w_2+z_2$ ，则 $w_1-w_2=z_2-z_1$ 同时属于 W 与 $W^\perp$ ；它与自己正交，长度平方为零，只能是零。所以 $w_1=w_2$ 、 $z_1=z_2$ 。几何上，直和把输入的自由度分成互不重叠的两部分；不是把同一个方向重复算两次。

6.8 从行内积得到四个基本子空间

把 A 看成从 $\mathbb{R}^n$ 到 $\mathbb{R}^m$ 的线性映射，有四个自然集合。列空间 $\text{Col}(A)=\{Ax:x\in\mathbb{R}^n\}$ 位于输出侧 $\mathbb{R}^m$ ，回答“哪些目标可以达到”；核 $\ker(A)=\{x\in\mathbb{R}^n:Ax=0\}$ 位于输入侧 $\mathbb{R}^n$ ，回答“哪些输入完全不可见”。核也记作 $N(A)$ 。行空间 $\text{Row}(A)=\text{Col}(A^T)$ 位于 $\mathbb{R}^n$ ，左核 $\ker(A^T)$ 位于 $\mathbb{R}^m$ 。这些集合都对线性组合封闭。[MIT-L4]

先看输入侧。把 A 的第 i 行转写成列向量 r_i ，则 $(Ax)_i=r_i^T x$ 。 $Ax=0$ 等价于 x 与每一行都正交，也等价于 x 与各行的所有线性组合正交。 $\text{Row}(A)$ 正由这些行张成，所以核与行空间正交补完全相同。证明包含两个方向，因而不只是“核中的向量与行正交”的单向结论。

$$\ker A = \text{Row}(A)^\perp \quad (6.5)$$

A 为 $m \times n$ ；等式两侧均为 $\mathbb{R}^n$ 的子空间。左侧要求 m 个行内积全为零，右侧要求与行空间的每个向量正交，这两项要求等价。

再把同一论证用于 A^T 。 A^T 的行就是 A 的列，因此 $A^T y = 0$ 等价于 y 与 A 的所有列、进而与整个列空间正交。这里 y 有 m 个分量，不能拿输入侧的 n 维向量代替。左核记录的是输出约束间的线性冗余。

$$\ker A^T = \text{Col}(A)^\perp \quad (6.6)$$

两侧均位于 $\mathbb{R}^m$ ； $\text{Col}(A)$ 与 $\ker(A^T)$ 构成输出侧的一对正交补。

于是每个输入 x 唯一分成行空间分量 x_r 和核分量 z ，且 $Ax = Ax_r$ ；核分量在映射后消失。每个输出目标 b 唯一分成列空间分量 p 和左核分量 e ，后者不能由 Ax 产生。第 7 章会证明 p 是离 b 最近的可达输出，而 e 正是最小二乘的正交残差。在这一章，先把这两组母空间和角色区分清楚。

例题 手算：不用消元也能核对四个空间

取 $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 1 & 2 \end{bmatrix}$ 。三列满足 $a_3 = a_1 + a_2$ ，前两列 $a_1 = (1, 2, 1)^T$ 、 $a_2 = (2, 4, 1)^T$ 不成比例，所以列空间维数为 2。第三行与第一行无关，第二行恰为第一行两倍，所以行空间也有两个独立方向。

向量 $z = (-1, -1, 1)^T$ 与每一行内积为零，因此 $Az = 0$ ；所有核向量都是 z 的倍数。向量 $y = (-2, 1, 0)^T$ 与每列内积为零，因此 $A^T y = 0$ ；所有左核向量都是 y 的倍数。输出 b 有解必须满足 $y^T b = -2b_1 + b_2 = 0$ 。下一节的维数定理会说明，为什么这里恰好各剩一个核方向。

6.9 秩 零度定理与解集结构

维数是独立方向的个数。长度为 n 的输入有 n 个可独立指定的坐标，但映射可能把部分方向压成零。秩 r 定义为列空间的维数，即可被输出分辨的独立变化数量；零度 k 定义为核的维数，即完全不可分辨的输入自由度。秩—零度定理说明 $r + k = n$ 。这不是巧合的计数，而是线性映射本身的结构。本章的维数关系见式 (6.7)。

例题 证明：从核的一组基扩充到整个输入空间

取核的一组基 $z_1, \dots, z_k$ ，并把它扩充为 $\mathbb{R}^n$ 的一组基 $z_1, \dots, z_k, w_1, \dots, w_{n-k}$ 。扩充可逐次完成：只要已有方向尚未张成 $\mathbb{R}^n$ ，就加入一个不在已有张成空间中的向量；最多加入到 n 个，既保持无关又最终张成。

证明 $Aw_1, \dots, Aw_{n-k}$ 张成 $\text{Col}(A)$ 。任意输入可按完整基写成 $x = \sum_i \alpha_i z_i + \sum_j \beta_j w_j$ 。因 $Az_i = 0$ ， $Ax = \sum_j \beta_j Aw_j$ 。所有可达输出都在这些像的张成中，反过来这些像当然也在列空间中。

证明这些像无关。若 $\sum_j \beta_j Aw_j = 0$ ，则 $w = \sum_j \beta_j w_j$ 属于核，所以又能写成 $\sum_i \alpha_i z_i$ 。把两种表示相减，是完整输入基的一条零组合；其无关性迫使所有 $\beta_j = 0$ 。因此这些像是一组基， $\text{rank}(A) = n - k$ ，得到 $r + k = n$ 。

$$\dim \text{Col}(A) + \dim \ker A = n \quad (6.7)$$

n 是输入空间维数；每个输入方向或用于区分输出，或进入核中不可辨认的自由度。这里的分配是维数计数，不是声称每个原坐标轴天然属于其中一类。

结合 $\ker(A) = \text{Row}(A)^\perp$ 与正交补维数公式， $\dim \text{Row}(A) = n - \dim \ker(A) = r$ ，所以行秩等于列秩。再对 A^T 用同一定理， $\dim \ker(A^T) = m - r$ 。至此四子空间维数分别为：列空间 r 、行空间 r 、核 $n - r$ 、左核 $m - r$ 。 m 和 n 不同会使两侧剩余方向数不同；即使 $m = n$ ，也须保留输入侧与输出侧的角色区别。

若 $r = n$ ，核只有零向量，映射是单射： $Ax = Ay$ 推出 $A(x - y) = 0$ ，故 $x = y$ 。若 $r = m$ ，列空间为整个 $\mathbb{R}^m$ ，映射是满射：每个 b 都能达到。方阵 $m = n$ 时，单射、满射及可逆彼此等价；矩形矩阵则可能只有其中一种。比如 3×2 满列秩映射不会混淆输入，却不能覆盖所有三维目标。

解集结构也由核解释。若已找到一个特解 x_p 满足 $Ax_p = b$ ，则任意解 x 都满足 $A(x - x_p) = 0$ ，所以 $x = x_p + z$ ，其中 $z \in \ker(A)$ ；反过来这种形式必有 $A(x_p + z) = b$ 。于是有解集合是一份核的平移，维数为 $n - r$ 。 $b = 0$ 时它就是子空间； $b \neq 0$ 时通常是不过原点的仿射集合。若 $b \notin \text{Col}(A)$ ，连特解也不存在。

例题 预览：可辨认输入与自由变量

$A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ 把 $(x_1, x_2)^T$ 映成 $(x_1 + x_2, 0)^T$ 。列空间是横轴，核由 $(1, -1)^T$ 张成，行空间由 $(1, 1)^T$ 张成；秩 1、零度 1。对 $b = (2, 0)^T$ ，可取特解 $(1, 1)^T$ ，全部解为 $(1, 1)^T + t(1, -1)^T$ 。

这些解沿一条直线分布，因为改变 x_1 、 x_2 的差值不影响它们的和。对 $b = (2, 1)^T$ 没有解，因为第二输出永远是 0。这个例子同时说明：未知量多不必然无解，约束多也不必然唯一，决定因素是秩、核以及目标是否在列空间中。

6.10 练习与参考解答

练习 6-1（概念与手算，入门） 判断 $W_1 = \{(x, y)^T : x + y = 0\}$ 、 $W_2 = \{(x, y)^T : x + y = 1\}$ 和闭第一象限是否为 $\mathbb{R}^2$ 的子空间，分别给出证明或反例。

练习 6-2（概念与手算，入门） 在平面 $z = x + y$ 中， $B = ((1, 0, 1)^T, (0, 1, 1)^T)$ 。求 $w = (3, -2, 1)^T$ 的 B 坐标，并解释平面维数与坐标分量数的区别。

练习 6-3（概念与手算，进阶） A 是 4×6 、秩 3。求四个基本子空间的维数和各自母空间。

练习 6-4（概念与手算，进阶） 证明若 $Ax_p = b$ ，则全部解恰为 $x_p + \ker(A)$ 。为什么这并不意味着无条件有解？

练习 6-5（编程实践，入门） 运行 `e20_vector.py`，独立用 NumPy 核对 $c = 1.5a - 0.5b$ 和 a 与残差的内积。列出应验收的数值，不把投影系数当概率。

练习 6-6（编程实践，入门） 令 $v = \text{np.array}([1., 2.])$ 。检查 v 、 $v.T$ 、 $v[:, \text{None}]$ 、 $v[\text{None}, :]$ 的 shape，并比较 $v @ v$ 与 $v[:, \text{None}] @ v[\text{None}, :]$ 。

练习 6-7（编程实践，进阶） 编程检查 $T(x) = (x_1^2, x_2^2)^T$ 是否线性。它满足 $T(0) = 0$ ；选择 $u = (1, 0)^T$ 、 $v = (1, 0)^T$ 计算加法条件。

练习 6-8（编程实践，进阶） 对 $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 1 & 2 \end{bmatrix}$ ，用矩阵乘法验收 $z = (-1, -1, 1)^T$ 和 $y = (-2, 1, 0)^T$ 分别属于核与左核，并验证两个原列无关。

练习 6-9 (场景应用, 入门) 一个模拟配方问题中, 原料份额 w_1 、 w_2 满足 $w_1, w_2 \geq 0$ 且 $w_1 + w_2 = 1$ 。这个可行集合是不是子空间? 若只研究份额变化 δw , 应采用什么空间?

练习 6-10 (场景应用, 进阶) 两个未知负载 x_1 、 x_2 由模拟传感器只报告它们的和与两倍和, $A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ 。观测 $b = (6, 12)^T$ 能否唯一恢复两负载? 增加哪类测量才可能分开?

练习 6-11 (场景应用, 进阶) 用两个参数描述三项模拟响应: $(x_1, x_2, x_1 + x_2)^T$ 。写出映射矩阵、列空间约束, 并判断目标 $(2, 3, 6)^T$ 是否可达。

练习 6-12 (场景应用, 拓展) 某模拟模型的参数有 6 个, 所有观测只能区分 4 个独立输入方向。若观测相容, 解集有多少自由度? 能否通过更换坐标直接消除不可辨认性?

参考解答与验收依据

6-1 $W_1 = \text{span}\{(1, -1)^T\}$, 对任意线性组合封闭, 是一维子空间。 W_2 不含零, 故不是。第一象限含零且加法封闭, 但 $(1, 1)^T$ 乘 -1 离开集合, 故不是。检验只允许正倍数会漏掉最后这个反例。

6-2 $w = 3(1, 0, 1)^T - 2(0, 1, 1)^T$, 所以 $[w]_B = (3, -2)^T$ 。两基向量无关且张成该平面, 维数 2; w 虽用三个标准坐标表示, 第三项由前两项约束, 不是额外自由度。

6-3 $\text{Col}(A) \subseteq \mathbb{R}^4$, 维 3; $\ker(A) \subseteq \mathbb{R}^6$, 维 3; $\text{Row}(A) \subseteq \mathbb{R}^6$, 维 3; $\ker(A^T) \subseteq \mathbb{R}^4$, 维 1。输入侧维数 $3+3=6$, 输出侧 $3+1=4$ 。

6-4 若 $Ax=b$, 则 $A(x-x_p)=0$, 所以 $x-x_p$ 属于核; 反过来 $Az=0$ 给出 $A(x_p+z)=b$ 。式子的前提是先存在特解, 等价于 $b \in \text{Col}(A)$; 若 b 不在列空间, 右侧不能凭空选择 x_p 。

6-5 应有 $a+b=[1, 3]$ 、 $c=[3.5, 0.5]$ 、 $a @ b=0$ 、 $\text{norm}(a) \approx 2.236068$; $\text{projection}=[3, 1.5]$ 、 $\text{residual}=[0.5, -1]$, $a @ \text{residual}=0$ 。用 `np.allclose` 比较向量。投影系数 1.5 大于 1 仍合法。

6-6 形状依次为 $(2,)$ 、 $(2,)$ 、 $(2, 1)$ 、 $(1, 2)$ 。 $v @ v$ 为标量 5; 外积为 $\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$, 形状 $(2, 2)$ 。一维转置不增加轴, 不能仅凭 $\cdot T$ 推断列向量。

6-7 $T(u+v)=[4,0]$, $T(u)+T(v)=[2,0]$, 二者不同, 所以不线性。可用 `np.array_equal` 得到 `False`。 $T(0)=0$ 只是必要条件, 几个随机样本通过也不能证明一般线性。

6-8 $A@z$ 与 $A.T@y$ 均为三维零向量。前两列组成 3×2 矩阵 C , 其前两列的第 1、3 行形成 $\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$, 行列式 -1 非零, 所以两列无关; `np.linalg.matrix_rank(C)` 应为 2。由第三列为前两列之和, A 的秩也为 2。

6-9 份额可行集合是线段, 不含零, 且不允许任意负数倍, 因此不是子空间。保持总份额的微小变化满足 $\delta w_1 + \delta w_2 = 0$, 组成 $\text{span}\{(1, -1)^T\}$; 靠近边界时还需另外检查原份额加变化后非负。线性变化空间不能替代完整业务约束。

6-10 秩 1, 核 $\text{span}\{(1, -1)^T\}$, 全部解 $(3, 3)^T + t(1, -1)^T$, 不能唯一。新增一条不与 $(1, 1)$ 成比例的测量, 如直接测 x_1 或测 $x_1 - x_2$, 可使两列独立。重复测相同总和只能增加精度信息, 不能在无噪代数上消除核。

6-11 $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, 输出满足 $b_3 = b_1 + b_2$, 列空间为该过原点平面。目标 $6 \neq 2 + 3$, 故不可达。左核向量 $(-1, -1, 1)^T$ 与目标内积 1, 直接给出不相容证据。

6-12 $\text{rank}(A)=4$, 零度 $6-4=2$, 解集为二维核的平移。可逆换坐标只改变核的表示, 不改变维数; 要识别全部参数, 需要新增独立信息、合理约束或改变参数化。额外约束必须有问题依据, 不能只因想得到唯一数值而添加。

6.11 核心要点与常见误区

核心要点: 线性组合把方向、坐标和矩阵列联系起来。子空间要求对全部实数线性组合封闭; 基同时保证表示存在与唯一, 维数统计独立自由度。

核心要点: Ax 的每个分量是一个行内积, 所以 $\ker(A) = \text{Row}(A)^\perp$; 对 A^T 重复论证得到 $\ker(A^T) = \text{Col}(A)^\perp$ 。秩—零度由核基扩充证明, 两侧母空间各有自己的维数。

常见误区: 含零不等于子空间; 单位圆也不等于它的张成空间。向量写有 n 个坐标不意味着所在子空间维数 n 。一维数组转置不会自动变列; 逆矩阵也不是逐元素取倒数。

6.12 延伸阅读 入门 进阶与拓展

入门阅读：看 18.06SC 第一单元总览“ $Ax=b$ 与四子空间”，先用图形区分输入、输出和可达集合（来源 MIT-L1；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/ax-b-and-the-four-subspaces/>）。

进阶阅读：读“线性无关、基与维数”及其 Ses1.9sum 摘要；课程指定第五版 3.4 节，重点完成基坐标练习（来源 MIT-L3；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/ax-b-and-the-four-subspaces/independence-basis-and-dimension/>）。

拓展阅读：读第六版公开样章 3.5“各子空间的维数”，与本章秩—零度证明相互核对，尤其分清 $n-r$ 与 $m-r$ （来源 LA-B2；公开链接：https://math.mit.edu/~gs/linearalgebra/ila6/ila6_3_5.pdf）。

第 7 章 线性方程组与正交最小二乘

7.1 三级学习路线

入门路线：从列空间相容条件和特解加核的一般解开始，手算一遍消元、回代与投影。

进阶路线：理解行阶梯形、秩、行列式和条件数，再分别从残差正交与平方求导推出正规方程。

拓展路线：用 QR 解释稳定计算，用协方差二次型把矩阵几何与统计方差联系起来。

7.2 先看一般解与一致性

求方程组前先问两个结构问题。第一， b 是否属于 $\text{Col}(A)$ ？属于才可能精确求解。第二， $\ker(A)$ 里有多少自由方向？一旦找到一个特解 x_p ，所有答案都是 $x_p + \ker(A)$ 。这把“有没有解”和“解是否唯一”分开。高斯消元接下来负责实际找出特解、核的基以及不相容的约束。

$$\{x: Ax = b\} = x_p + \ker A \quad (Ax_p = b) \quad (7.1)$$

此式以存在特解 x_p 为前提。右侧表示所有 $x_p + z$ ， z 遍历核；它不是向量与集合的逐元素数值相加，而是集合的平移。

由第 6 章 $\text{Col}(A)^\perp = \ker(A^T)$ ，有 $b \in \text{Col}(A)$ 当且仅当每个 $y \in \ker(A^T)$ 都满足 $y^T b = 0$ 。必要性来自 $y^T Ax = 0$ ；充分性来自 $\mathbb{R}^m = \text{Col}(A) \oplus \ker(A^T)$ 的唯一分解。如果 b 还含非零左核分量，就不可能是 Ax 。一个非零 y 若同时满足 $A^T y = 0$ 、 $y^T b \neq 0$ ，就给出无解的可核验证据。

例题 手算：先用相容条件判断，再写一般解

$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix}$ 的列空间为 $\text{span}\{(1, 2)^T\}$ ，核由 $(-2, 1, 0)^T$ 与 $(-3, 0, 1)^T$ 张成。对 $b = (4, 8)^T$ ，可取 $x_p = (4, 0, 0)^T$ ，全部解为 $x = x_p + s(-2, 1, 0)^T + t(-3, 0, 1)^T$ 。两个实数 s 、 t 自由变化，对应零度 2。

对 $b = (4, 9)^T$ ，取 $y = (-2, 1)^T$ 便有 $A^T y = 0$ 、 $y^T b = 1$ ，因此无解。行操作会把这个事实显示成 $0=1$ ，但相容性的原因已经能从列空间和左核解释。

7.3 消元 行阶梯形与主元

考虑 $Ax=b$ 。按行看，每一行是一条方程；按列看，是寻找列向量的组合系数。高斯消元的目标不是改变答案，而是把方程改写为等价、易回代的形式。允许三种初等行变换：交换两条方程；把一条方程乘非零常数；把一条方程的倍数加到另一条上。每种操作都有逆操作，所以解集保持不变。操作必须同时作用于系数和右端，增广矩阵 $[A|b]$ 用竖线把二者放在一起，防止漏改右端。[MIT-L2]

消元从左向右寻找主元。主元是当前未处理行列区域中，用来消去其下方元素的非零数；主元所在列称为主元列。如果预定位置是零，可以与下方非零行交换；若该列下方也全为零，就跳过这一列并继续寻找。跳过一列不等于整个方程无解，它表示对应未知量可能成为自由变量。行阶梯形的各非零行，其首个非零元素的位置逐行右移，零行排在最下方。主元数量就是矩阵的秩。

把主元缩放到 1，再消去其上方元素，会得到简化行阶梯形 R 。主元变量由其他变量决定；非主元列对应的变量可以自由选择，称为自由变量。每给自由变量赋值一次，就能回代得到一个解。用计算机求解时通常不必把所有上方元素消成零，三角分解加回代更经济；简化行阶梯形主要帮助我们看清解集结构。

消元后的某行如果是 $[0 \cdots 0|d]$ 且 $d \neq 0$ ，它表示 $0=d$ ，系统无解。如果没有矛盾行，就有解；此时自由变量数为 $n-r$ ， r 是 A 的秩。 $r=n$ 时不存在自由变量，解唯一； $r < n$ 时至少一个自由变量可任意取实数，所以有无穷多解。等价的统一条件是 $\text{rank}(A)=\text{rank}([A|b])$ 表示有解，二者不等表示无解；增广列只能使秩至多增加 1。

例题 手算：完整做一次消元与回代

方程为 $x+2y=9$ 、 $3x-y=5$ 。增广矩阵写成 $[[1,2|9],[3,-1|5]]$ 。执行第二行减 3 倍第一行，得到 $[[1,2|9],[0,-7|-22]]$ 。第二个主元是 -7 ，因此 $y=22/7$ ，再由 $x=9-2y$ 得到 $x=19/7$ 。回代两条原方程，分别得到 9 和 5。

消元矩阵 $E = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}$ 满足 $EA=U$ ，其中 $U = \begin{bmatrix} 1 & 2 \\ 0 & -7 \end{bmatrix}$ ； $E^{-1} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}$ 记作 L ，故 $A=LU$ 。 L 记录消元时用过的倍数 3。求解可以先解 $Lz=b$ ，再解 $Ux=z$ 。若消元发生换行，准确关系应写作 $PA=LU$ ，其中 P 是置换矩阵，不能无条件删去 P 。

浮点消元会受到舍入影响。部分选主元在当前列的候选行中选择绝对值最大的元素，再换到主元位置，以控制消元倍数。它常能显著改善计算稳定性，但无法消除问题本身对数据误差的敏感性。数学上的非零主元与数值上足够可靠的主元也有区别：数据带噪时，“这个元素算不算零”需要结合尺度和容差。标准数值库使用成熟分解，而手写消元主要用于理解结构。

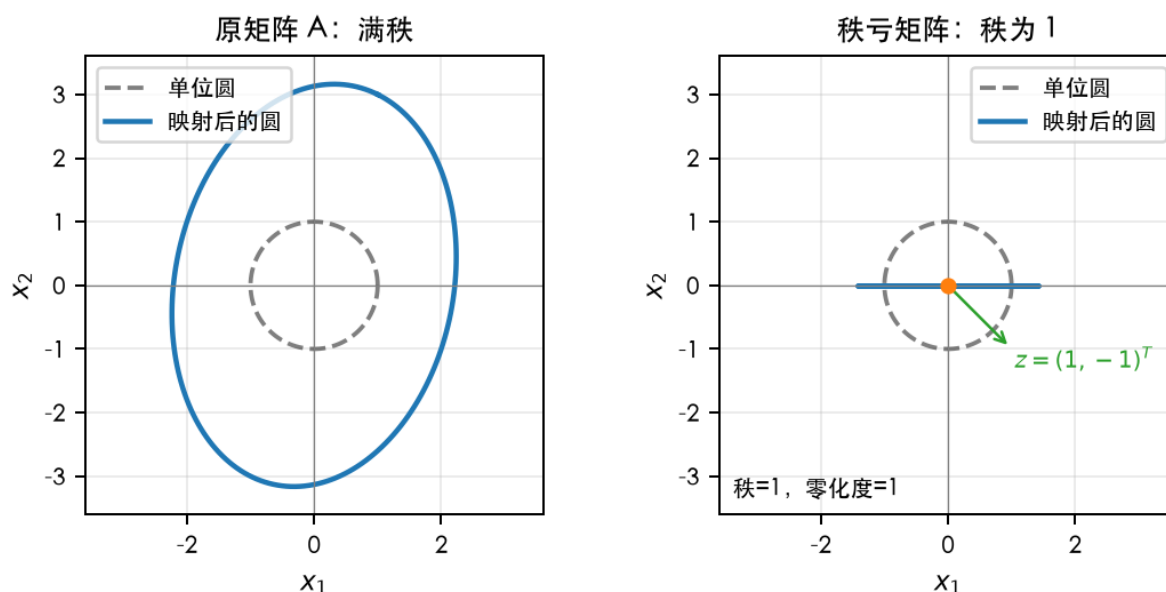


图 7-1 `e23_solve.py` 的秩图：左侧原矩阵 $A = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$ 把单位圆映成椭圆；右侧另设 $\text{rank_deficient} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ 把平面压成一条直线， $\text{null_vector} = (1, -1)^T$ 映到零。右图矩阵与后面的病态矩阵 C 不是同一对象。

读图指南 左图跟踪单位圆经满秩 A 成为椭圆，仍保留两个独立方向；右图秩 1 矩阵把圆压到横轴，绿色核方向映到原点。右图矩阵是明确的秩亏示例，不是后面病态但可逆的 C 。

7.4 行列式 可逆性与面积体积

行列式 $\det(A)$ 是为方阵定义的标量。二维 $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 的行列式为 $ad-bc$ ，它等于两个列向量张成平行四边形的有向面积。绝对值给面积，正负号给定向是否反转。三维对应有向体

积， n 维对应体积倍率。矩形矩阵没有这里的行列式，判断它的信息维数应使用秩和奇异值，不能强行套入方阵公式。

行列式的基本规则与消元一致：交换两行会变号；一行乘 c ，行列式也乘 c ；一行加上另一行的倍数，行列式不变。三角矩阵的行列式是对角元素之积。因此只用换行和倍加消元时， $\det(A)$ 等于主元乘积再乘 $(-1)^s$ ， s 是换行次数。主元的数值本身可能受行缩放影响，若消元中把某行另乘常数，必须把该因子计入，不能直接相乘了事。

对 $n \times n$ 方阵，下列说法等价： $\det(A) \neq 0$ ； $\text{rank}(A) = n$ ；齐次方程 $Ax = 0$ 只有零解； A 有逆；对每一个 $b \in \mathbb{R}^n$ ， $Ax = b$ 都有且只有一个解。它们分别从体积、独立方向、信息损失、可撤回性和求解能力描述同一事实。这个等价列表中的“对每一个 b ”和“方阵”不可省略。矩形矩阵列满秩只保证解若存在就唯一；行满秩只保证每个右端都有解。

$$\det(AB) = \det(A)\det(B), \quad \det(A^{-1}) = \frac{1}{\det(A)} \quad (7.2)$$

第一式要求 A 、 B 是相同阶数的方阵；第二式还要求 A 可逆。行列式乘积对应先后变换的体积倍率相乘，逆变换则撤销原来的倍率。

例题 手算：同样面积倍率，长度变化可以不同

I_2 的行列式是 1，剪切 $H = \begin{bmatrix} 1 & 0.8 \\ 0 & 1 \end{bmatrix}$ 的行列式也是 1，但 $H(0,1)^T = (0.8, 1)^T$ 的长度大于 1。所以 $\det(A) = 1$ 不能推出保持长度，更不能推出 A 为单位矩阵。

对程序方程矩阵 $A = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$ ， $\det(A) = -1 - 6 = -7$ 。绝对值 7 表示面积放大七倍，负号表示定向翻转；非零同时保证方程 $Ax = b$ 对任何二维右端唯一可解。这里的负号并不表示“面积为负”或长度为负。

7.5 范数 距离 残差与条件数

范数为向量定义大小。二范数 $\|x\|_2 = \sqrt{\sum_i x_i^2}$ 是欧氏长度，一范数 $\|x\|_1 = \sum_i |x_i|$ 把各分量绝对值相加，无穷范数 $\|x\|_\infty = \max_i |x_i|$ 取最大分量绝对值。双竖线表示范数，绝对值用于标量。任何一种

范数都满足：非负且仅零向量取零； $\|cx\|=|c|\|x\|$ ； $\|x+y\|\leq\|x\|+\|y\|$ 。最后一条是三角不等式，是“绕路不会比直达更短”的抽象表达。

距离定义为 $d(x,y)=\|x-y\|$ 。它把两点的差向量转换成一个非负数，不同范数对应不同的几何衡量。例如城市街区步行可用一范数的折线路程，平面直线距离用二范数，多指标中最大的偏差可用无穷范数。量纲不一致时应先说明缩放依据，否则一个取值数百的变量可能压过另一个取值仅零点几的变量；选择距离也在选择“差异”的含义。

计算 $Ax=b$ 的近似解 $\hat{x}$ 后，残差 $r=b-A\hat{x}$ 衡量原方程未被满足的程度。帽子表示近似估计， r 是右端空间中的向量；真实解误差 $\hat{x}-x$ 则在输入空间。二者不是同一个量。若 A 在某一输入方向压缩得很厉害，较大的解误差也会产生很小的残差，因此“小残差”不自动等于“解很准确”。因此应结合问题的敏感程度解释残差。

程序 e24_distances.py 三个点固定为 $[[1,1],[3,3],[5,-0.5]]$ ，参考点固定为 $[4,3]$ 。先广播相减，再沿 $\text{axis}=1$ 按每行坐标求距离。三组输出为 $L1=[5,1,4.5]$ 、 $L2\approx[3.605551,1,3.640055]$ 、 $Linf=[3,1,3.5]$ 。图只画第一点 $P=(1,1)$ 和 $Q=(4,3)$ ，不能误读为同时画出了三行观测。

编程思路 先让三个点同时减去同一个参考点，得到三个差向量，再沿坐标维分别计算 $L1$ 、 $L2$ 、 $L\infty$ 距离。把数值与二维等距集合的菱形、圆和正方形联系起来。

代码对应

逐点计算 $p-q \rightarrow \text{difference} = \text{points} - \text{reference}$: reference 的二维坐标广播到 points 的每行，得到形状 $(3,2)$ 的差。

$\|p-q\|_1=\sum|p_j-q_j| \rightarrow l1 = \text{np.linalg.norm}(\text{difference}, \text{ord}=1, \text{axis}=1)$: $\text{axis}=1$ 沿每个点的坐标求范数，避免把三个点合成一个矩阵范数。

$\|p-q\|_2=\sqrt{\sum(p_j-q_j)^2} \rightarrow l2 = \text{np.linalg.norm}(\text{difference}, \text{ord}=2, \text{axis}=1)$: 与 $L1$ 和 $L\infty$ 比较时必须保持相同的差向量和相同的计算轴。

$\|p-q\|_\infty=\max_j|p_j-q_j| \rightarrow \text{linf} = \text{np.linalg.norm}(\text{difference}, \text{ord}=\text{np.inf}, \text{axis}=1)$: 每个点的最大坐标差给出切比雪夫距离，并满足 $L\infty\leq L2\leq L1$ 。

易错提示 距离选择取决于特征单位和用途；量纲不同的特征直接相加可能被大尺度特征支配。二维图需等比例坐标轴，才能正确阅读不同范数的单位球。

正文代码 e24_distances.py

```
# 教材位置：第 7 章，第 7.5 节；图 7-2。
# 单位、变量与形状：两端点为 (2,) 坐标；一范数、二范数、无穷范数均沿用坐标的长度单位。
# 学习目标：将点的距离写成“坐标差的范数”，比较  $L_1$ 、 $L_2$ 、 $L_\infty$  三种度量。
# 核心数学关系：对位移  $d=(dx, dy)$ ， $L_1=|dx|+|dy|$ ，
#  $L_2=\sqrt{dx^2+dy^2}$ ， $L_\infty=\max(|dx|, |dy|)$ ，且  $L_\infty \leq L_2 \leq L_1$ 。
# 数据与输出：计算三个二维点到参考点 (4,3) 的三种距离，打印数值，
# 并保存 e24_distances.png；图中只展示第一个点到参考点的路线。
# 运行方式：安装 NumPy、Matplotlib 后，运行 python e24_distances.py；
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# points 形状 (3,2)，每行是一个点；第一列是横坐标，第二列是纵坐标。
points = np.array([[1.0, 1.0], [3.0, 3.0], [5.0, -0.5]])
# reference 是形状 (2,) 的一维数组，表示所有距离计算共同使用的参考点。
reference = np.array([4.0, 3.0])
# 广播将 reference 应用于每一行，得到形状仍为 (3,2) 的位移矩阵。
# 正负号反映方向，而范数取绝对值或平方，所以两点间距离不依赖减法方向。
difference = points - reference
# axis=1 表示每行独立计算一个向量范数，输出形状为 (3,);
# 若省略 axis 对二维数组调用 norm，含义可能变为矩阵范数，不能混用。
# ord=1 是绝对值之和，即只能沿坐标方向移动时的曼哈顿距离。
l1 = np.linalg.norm(difference, ord=1, axis=1)
# ord=2 是每行的欧氏长度，对应平面中的直线距离。
l2 = np.linalg.norm(difference, ord=2, axis=1)
# ord=np.inf 是每行绝对值最大的坐标差，也称切比雪夫距离；
# 若每一步可同时在两个坐标方向移动相同的最大幅度，它体现所需步数尺度。
linf = np.linalg.norm(difference, ord=np.inf, axis=1)
# 只给 l2 的显示结果四舍五入；保存的 l2 数组仍保留后续断言所需精度。
print("L1 =", l1, "; L2 =", np.round(l2, 6), "; Linf =", linf)
# 手算三行的绝对坐标差之和，验证广播和计算轴选择正确。
assert np.allclose(l1, [5, 1, 4.5])
# 欧氏距离的平方分别为  $3^2+2^2$ 、 $1^2+0^2$ 、 $1^2+3.5^2$ 。
assert np.allclose(l2, np.sqrt([13, 1, 13.25]))
# np.all 要求三个点均满足范数之间的不等式；这里比较的是同一位移的不同范数。
assert np.all(linf <= l2) and np.all(l2 <= l1)
# 取第一个点 P=(1,1) 与参考点 Q=(4,3)，两者各为形状 (2,) 的数组。
P, Q = points[0], reference
fig, ax = plt.subplots(figsize=(4.8, 3.6))
# 红色直线直接连接 P、Q，长度  $\sqrt{3^2+2^2}=\sqrt{13}$ ，对应  $L_2$ 。
ax.plot([P[0], Q[0]], [P[1], Q[1]], "r-", label="二范数：直线路径")
# 蓝色折线先横移 3 再竖移 2，总路径长度为 5，对应  $L_1$ 。
#  $L_\infty=3$  是最大单轴位移；本图用标题给出数值，没有另画一条  $L_\infty$  路线。
ax.plot([P[0], Q[0], Q[0]], [P[1], P[1], Q[1]], "b--", label="一范数：网格路径")
# 标出端点以便确认两种路线连接的是相同的一对位置。
ax.scatter([P[0], Q[0]], [P[1], Q[1]], color="black")
# 显示范围围绕本图的 P、Q；第三个点仅参与上面的数值计算，没有画在这里。
```



```
ax.set(xlim=(0, 5), ylim=(0, 4), xlabel="x", ylabel="y",
       title=r"距离:  $L_1=5$ ,  $L_2=\sqrt{13}$ ,  $L_\infty=3$ ")
# equal 让横纵单位长度一致, 保证图上的直线长度比例有几何意义。
ax.set_aspect("equal")
ax.legend()
ax.grid(alpha=0.3)
# 调整布局、保存 180 dpi 图片, 然后关闭图以释放资源。
fig.tight_layout()
fig.savefig("e24_distances.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

```
L1 = [5. 1. 4.5] ; L2 = [3.605551 1. 3.640055] ; Linf = [3. 1. 3.5]
```

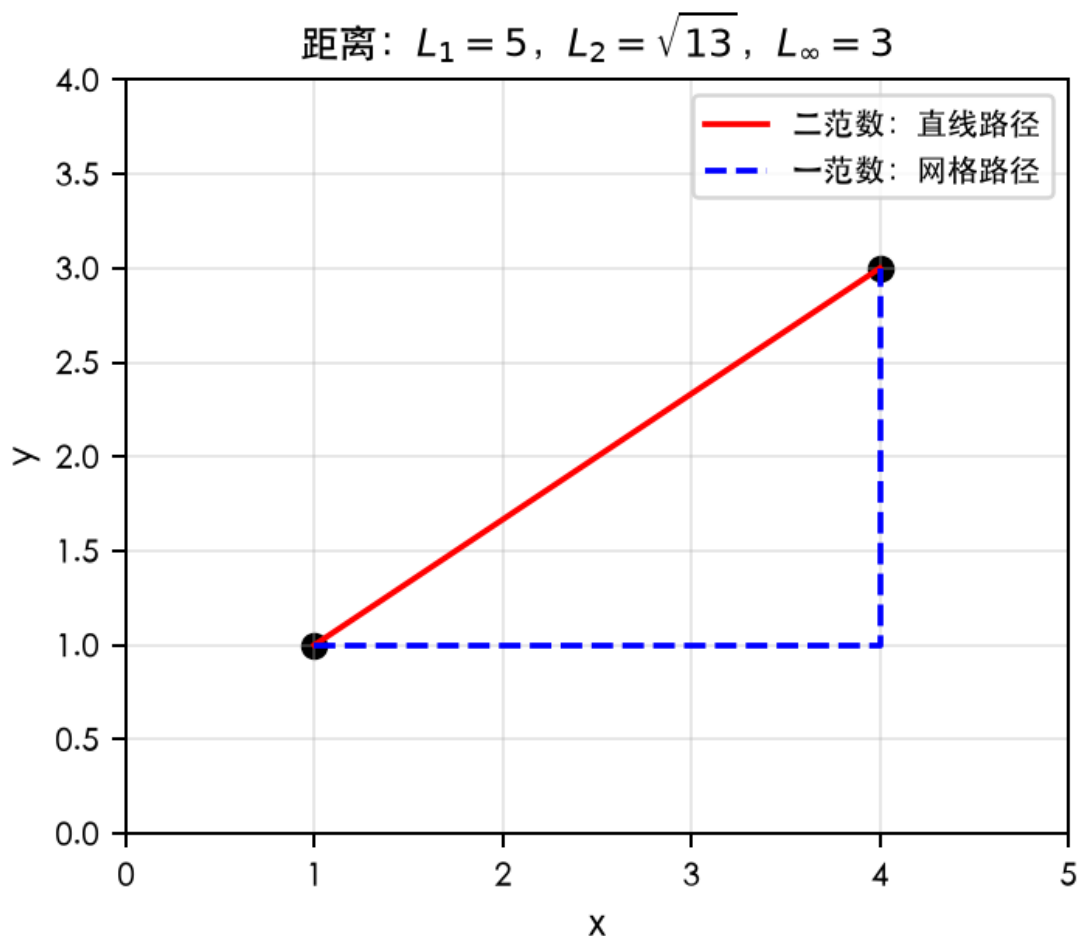


图 7-2 $P=(1,1)$ 、 $Q=(4,3)$ 的差为 $(3,2)$ 。直线路程 $L_2=\sqrt{13}$, 先横后纵的折线路程 $L_1=5$, 最大单轴位移 $L_\infty=3$ 。图中的折线仅帮助解释一范数, 不是最小二乘残差图。

读图指南 先认出同一对端点 $P=(1,1)$ 、 $Q=(4,3)$, 再比较红色直线与蓝色网格折线。标题给出的 $L_\infty=3$ 是最大单轴位移, 图中没有另画第三条路径。

对可逆方阵, 矩阵二范数 $\|A\|_2$ 定义为单位输入的最大输出长度, $\|A^{-1}\|_2$ 则反映逆向最大放大。二范数条件数 $\kappa_2(A)=\|A\|_2\|A^{-1}\|_2$, 希腊字母 κ 读作“卡帕”, 表示输入数据相对误差在最不利方向

可能被放大的量级。第 9 章将证明它等于最大与最小奇异值之比。条件数属于问题与所选范数，不是程序运行时间，也不是某个求解算法的准确率。

当 A 保持不变，右端从 b 变为 $b+\Delta b$ ，解从 x 变为 $x+\Delta x$ ，则 $A\Delta x=\Delta b$ ，所以 $\Delta x=A^{-1}\Delta b$ 。由范数次乘性得 $\|\Delta x\|_2\leq\|A^{-1}\|_2\|\Delta b\|_2$ ，再由 $\|b\|_2=\|Ax\|_2\leq\|A\|_2\|x\|_2$ 整理出相对误差界。 Δ 表示扰动而非微分，此处不要求扰动无穷小。因为不等式取最坏方向，实际误差放大不必达到条件数。若 A 本身也变动，则需要另加入 ΔA ，不能只用这条界。

行列式大小不适合代替条件数。 $10^{-6}I_2$ 的行列式只有 10^{-12} ，条件数却是 1； $\text{diag}(10^8,10^{-8})$ 的行列式为 1，条件数却是 10^{16} 。统一缩放会改变体积而保持相对敏感度，方向间伸缩差异才造成病态。双精度机器精度约为 2.22×10^{-16} ，但没有对所有数据、用途和精度目标都适用的固定条件数分界。

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \kappa_2(A) \frac{\|\Delta b\|_2}{\|b\|_2} \quad (7.3)$$

A 为固定可逆方阵， $b\neq 0$ ， $x=A^{-1}b$ 。 Δb 是右端扰动， Δx 是相应解变化。该式说明最坏方向的放大上界，不说明每次扰动都恰好放大 $\kappa_2(A)$ 倍。

程序 `e23_solve.py` 对 $A=[[1,2],[3,-1]]$ 与 $b=[9,5]$ ，`solve` 给出 $[2.714286,3.142857]$ ，等于 $[19/7,22/7]$ ；`rank=2`、`det=-7`、`condition≈1.456083`。病态例另外固定 $C=[[1,1],[1,1.0001]]$ ，右端从 $b_1=[2,2]$ 改为 $b_2=[2,2.0001]$ ，解从 $[2,0]$ 变成 $[1,1]$ 。相对右端变化约 3.536×10^{-5} ，相对解变化约 0.707107，条件数约 40002。两组解各自回代都正确，差异来自两条约束几乎平行。

编程思路 先求一个可逆 2×2 系统的唯一解，再用近乎平行的两条约束展示病态系统对右端微扰的敏感性。最后用秩亏矩阵与零空间区分不唯一和无解，并以两张图分别解释秩与条件数。

代码对应

解 $Ax=b$ ， $A=[[1,2],[3,-1]]$ ， $b=[9,5]\rightarrow x = \text{np.linalg.solve}(A, b)$ ：`solve` 直接解线性系统；结果为 $[19/7,22/7]$ ，再代回检验残差。

将近奇异系统右端 [2,2] 改为 [2,2.0001] $\rightarrow x_2 = \text{np.linalg.solve}(C, b_2)$: 同一系数矩阵 C 下解从 [2,0] 变为 [1,1], 展示微小输入变化可能被放大。

计算相对解变化 $\|x_2 - x_1\|/\|x_1\| \rightarrow \text{relative_x} = \text{np.linalg.norm}(x_2 - x_1) / \text{np.linalg.norm}(x_1)$: 与 relative_b 及 $\text{cond}(C)$ 的上界比较, 区分绝对变化和相对变化。

验证秩亏系统存在非零零空间向量 [1,-1] $\rightarrow \text{null_vector} = \text{np.array}([1.0, -1.0])$: A 的零空间方向解释一族解; 不一致右端则没有精确解。

易错提示 可逆不等于数值稳定; 很小的残差也不保证病态问题的解接近真实值。不要先显式求逆再乘 b, solve 更直接; 秩与条件数回答的不是同一个问题。

正文代码 e23_solve.py

```
# 教材位置: 第 7 章, 第 7.5 节; 图 7-1、图 7-3。
# 单位、变量与形状: 矩阵 A 为 (2,2), 解与右端为 (2,); 本例数值无量纲, 条件数衡量相对敏感
# 度。
# 学习目标: 求解线性方程组 Ax=b, 并用近奇异矩阵观察输入扰动的放大。
# 核心数学关系: 可逆方阵给出唯一解; 条件数 kappa(A)=||A||_2*||A^{-1}||_2
# 描述最不利方向上的相对误差放大上界, 固定 A 时有
# ||delta x||/||x|| <= kappa(A)*||delta b||/||b|| (精确算术下)。
# 数据与输出: 先解一个 2x2 方程组, 再保持病态矩阵 C 不变而轻微改变右端,
# 打印两组解、矩阵性质和相对变化, 保存 e23_solve.png, 并用 e23_rank.png 解释秩与零空间。
# 运行方式: 安装 NumPy、Matplotlib 后, 运行 python e23_solve.py;
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# A 形状 (2,2), 每一行是一条方程的系数; b 形状 (2,) 是对应右端项。
# 方程为 x+2y=9、3x-y=5, 待求解向量包含未知量 x、y。
A = np.array([[1.0, 2.0], [3.0, -1.0]])
b = np.array([9.0, 5.0])
# solve 直接求解 Ax=b, 返回形状 (2,); 无需显式计算 inv(A)@b,
# 直接求解通常更高效、数值上也更合适。本接口要求 A 为方阵且非奇异。
x = np.linalg.solve(A, b)
# round 仅控制显示的小数位, x 本身仍保留完整浮点结果供后续检验。
print("方程组的解 =", np.round(x, 6))
# rank 是数值秩, det 是行列式; cond 默认给出 2-范数条件数。
# 满秩说明在精确模型下可唯一求解, 条件数则进一步描述解对扰动有多敏感;
# 行列式的绝对大小还受缩放影响, 不能单独用它衡量数值稳定性。
print("秩、行列式、条件数 =", np.linalg.matrix_rank(A),
      round(np.linalg.det(A), 6), round(np.linalg.cond(A), 6))
# 同时检查解析解和代回方程的残差; allclose 允许浮点舍入误差。
assert np.allclose(x, [19 / 7, 22 / 7]) and np.allclose(A @ x, b)
# C 的两行几乎相同, 仍可逆但接近奇异; 两条约束几乎平行, 交点很敏感。
# “病态”描述问题本身的敏感性, 不能简单归因于求解算法出错。
```

```

C = np.array([[1.0, 1.0], [1.0, 1.0001]])
# 只让第二个右端分量增加 0.0001, 系数矩阵 C 完全不变。
b1 = np.array([2.0, 2.0])
b2 = np.array([2.0, 2.0001])
# 相同的近奇异矩阵配上略有不同的右端, 解分别接近 [2,0] 与 [1,1]。
x1 = np.linalg.solve(C, b1)
x2 = np.linalg.solve(C, b2)
# 使用欧氏范数衡量整体变化, 并分别以原右端和原解的范数归一化,
# 得到无单位的相对变化, 便于比较不同量纲/规模的问题。
relative_b = np.linalg.norm(b2 - b1) / np.linalg.norm(b1)
relative_x = np.linalg.norm(x2 - x1) / np.linalg.norm(x1)
# 条件数很大意味着存在可被显著放大的扰动方向, 但并非每个扰动都取到上界。
print("扰动前后的解 =", x1, x2, "; 条件数 =",
      round(np.linalg.cond(C), 2))
print("右端 b、解 x 的相对变化 =", round(relative_b, 8), round(relative_x, 6))
# 检查两次解确实相距明显; 这与右端变化很小并不矛盾。
assert np.allclose(x1, [2, 0]) and np.allclose(x2, [1, 1])
# 对“仅扰动 b”检查条件数上界, 额外的 1e-10 容差吸收浮点舍入影响。
# 上界是最坏情形的保证, 不要求相对解变化恰好等于该上界。
assert relative_x <= np.linalg.cond(C) * relative_b + 1e-10
# 为秩与零空间的概念配套作图: 满秩矩阵直接复用本例开头的 A。
# rank_deficient 是另一个明确命名的秩亏矩阵; 它与病态但可逆的 C 不是同一对象。
rank_deficient = np.array([[1.0, 1.0], [0.0, 0.0]])
null_vector = np.array([1.0, -1.0])
assert np.linalg.matrix_rank(A) == 2
assert np.linalg.matrix_rank(rank_deficient) == 1
assert np.allclose(rank_deficient @ null_vector, 0)
# 200 个单位圆列向量由同一矩阵统一映射; 满秩保留面积, 秩亏只留下一个方向。
angles = np.linspace(0, 2 * np.pi, 200)
unit_circle = np.vstack([np.cos(angles), np.sin(angles)])
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0))
for ax, matrix, title in zip(axes, [A, rank_deficient],
                              ["原矩阵 A: 满秩", "秩亏矩阵: 秩为 1"]):
    ax.plot(*unit_circle, "k--", alpha=0.5, label="单位圆")
    ax.plot(*(matrix @ unit_circle), lw=1.8, label="映射后的圆")
    ax.axhline(0, color="gray", lw=0.5)
    ax.axvline(0, color="gray", lw=0.5)
    ax.set_aspect("equal")
    ax.grid(alpha=0.25)
    ax.set_title(title, fontsize=10)
    ax.set(xlim=(-3.6, 3.6), ylim=(-3.6, 3.6), xlabel="$x_1$", ylabel="$x_2$")
    ax.legend(fontsize=7.8)
# z=(1,-1)落在 rank_deficient 的零空间; 图中原点就是它变换后的像。
axes[1].annotate("", xy=null_vector, xytext=(0, 0),
                 arrowprops={"arrowstyle": "->", "color": "tab:green"})
axes[1].text(1.05, -1.3, "$z=(1,-1)^T$", fontsize=8, color="tab:green")
axes[1].scatter([0], [0], color="tab:orange", s=22, zorder=4)
axes[1].text(0.03, 0.04, "秩=1, 零化度=1",
             transform=axes[1].transAxes, fontsize=8)
fig.tight_layout()
fig.savefig("e23_rank.png", dpi=180, bbox_inches="tight")
plt.close(fig)

# 生成横坐标网格, 用二维几何图展示两个很接近的约束方向。
xx = np.linspace(-1, 3, 200)
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1))
# 第一条方程 x+y=2 是两个系统共用的约束。
# 另外同时画出原系统与扰动系统的第二约束, 避免把原解误当成另外两线的交点。

```

```
def constraint_lines(horizontal):
    # 每条线均从实际 C 与 b1/b2 求出; 没有重新输入另一套近奇异系数。
    common = (b1[0] - C[0, 0] * horizontal) / C[0, 1]
    original = (b1[1] - C[1, 0] * horizontal) / C[1, 1]
    perturbed = (b2[1] - C[1, 0] * horizontal) / C[1, 1]
    return common, original, perturbed
labels = ["共同约束: $x+y=2$", "原约束: $x+1.0001y=2$",
          "扰动约束: $x+1.0001y=2.0001$"]
colors = ["tab:blue", "tab:green", "tab:orange"]
styles = ["-", ":", "--"]
ax = axes[0]
for values, label, color, style in zip(constraint_lines(xx), labels, colors, styles):
    ax.plot(xx, values, linestyle=style, color=color, label=label)
# 两个点分别代表两套系统的解, zorder=3 使点显示在曲线之上。
ax.scatter([x1[0], x2[0]], [x1[1], x2[1]], color=["tab:green", "tab:orange"],
           zorder=3)
ax.annotate("原解 (2,0)", xy=x1, xytext=(1.5, 0.65), fontsize=8, color="tab:green")
ax.annotate("扰动解 (1,1)", xy=x2, xytext=(-0.75, 1.45), fontsize=8,
           color="tab:orange")
# 标注坐标、图例和网格, 帮助把代数解读作两条直线的交点。
ax.set(xlabel="$x$", ylabel="$y$", xlim=(-1, 3), ylim=(-1, 3),
      title="约束相近, 解相距较远")
ax.legend(fontsize=7.8)
ax.grid(alpha=0.3)
# 围绕 (1.5, 0.5) 按相同倍数放大横纵坐标, 三条几乎重合的约束才能看清。
# 这是显示坐标的变换; 求解所用 C、b1、b2 均保持原样。
zoom_factor = 1e4
zoom_center = np.array([1.5, 0.5])
zoom_x = np.linspace(-0.6, 0.6, 150)
local_x = zoom_center[0] + zoom_x / zoom_factor
ax = axes[1]
for values, label, color, style in zip(constraint_lines(local_x), labels, colors,
                                       styles):
    ax.plot(zoom_x, (values - zoom_center[1]) * zoom_factor,
            linestyle=style, color=color, label=label.split(":")[0])
ax.set(xlabel="$10^4(x-1.5)$", ylabel="$10^4(y-0.5)$", title="局部坐标放大 $10^4$")
ax.legend(fontsize=8)
ax.grid(alpha=0.3)
# 调整布局、保存图片并关闭图, 便于批量运行时控制资源占用。
fig.tight_layout()
fig.savefig("e23_solve.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

方程组的解 = [2.714286 3.142857]

秩、行列式、条件数 = 2 -7.0 1.456083

扰动前后的解 = [2. 0.] [1. 1.] ; 条件数 = 40002.0

右端 b、解 x 的相对变化 = 3.536e-05 0.707107

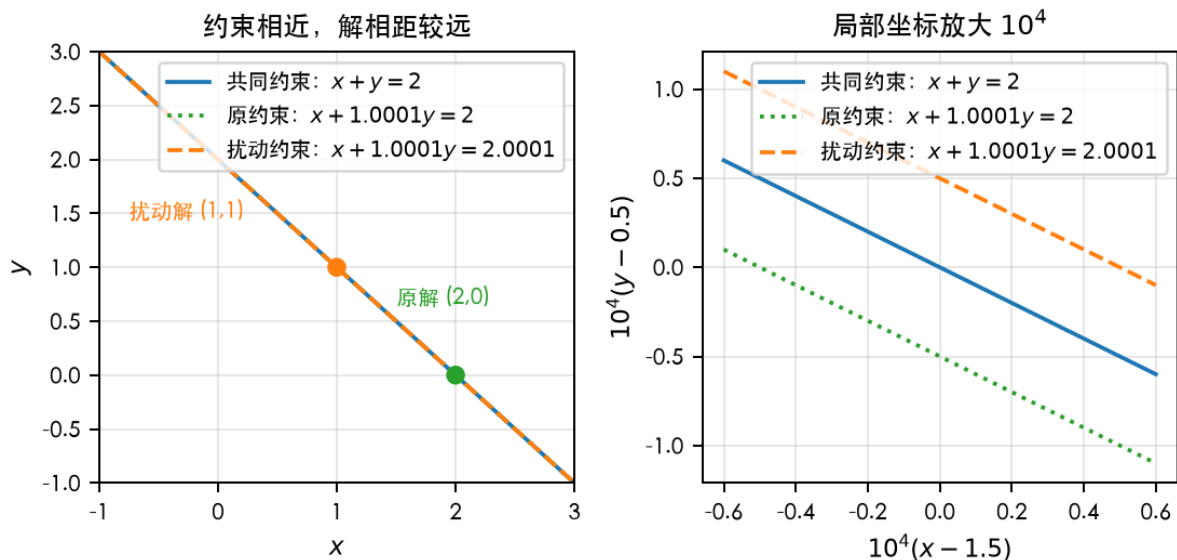


图 7-3 同一个病态系统 C 的原始与扰动约束。第二条约束右端仅增加 0.0001 ，交点由 $(2,0)$ 移到 $(1,1)$ 。右图围绕 $(1.5,0.5)$ 把横纵坐标共同放大 10^4 倍，用于看见原图中几乎重合的三条直线。

读图指南 左图中三条约束几乎重合，但原解 $(2,0)$ 与扰动解 $(1,1)$ 相距明显；右图同时放大两坐标一万倍后看清线间差异。横纵放大用于观察，没有改变求解时的 C 和右端。

把计算残差 $r=b-A \hat{x}$ 视为右端扰动，可以把近似解解释成另一个系统的精确解： $A \hat{x}=b-r$ 。若原 A 可逆且 $b \neq 0$ ，前面的扰动界给出 $\|\hat{x}-x\|_2/\|x\|_2 \leq \kappa_2(A) \|r\|_2/\|b\|_2$ 。这个式子把可直接计算的残差，与通常无法直接知道真实解误差联系起来。它仍然是上界，条件数很大时，界可能很宽；条件数适中且相对残差很小时，才能得到有力的准确性证据。

若数据本身只能测到三位有效数字，即使计算机给出十五位小数，后面的小数也不自动获得物理意义。另一方面，遇到病态问题不必立刻放弃，可以重新审视变量尺度、增加独立观测、改用更可辨认的参数组合，或引入有依据的正则化。算法稳定性、模型可辨认性与测量精度各处在不同环节；把它们区分开，才知道该改哪一部分。

7.6 从一条直线推导正交投影

精确方程无解时，最自然的替代问题是：在可达输出中，哪一个离 b 最近？先看二维的一条直线，再推广到任意列空间。这里“最近”必须说明距离，本节采用欧氏距离；更换范数可能改变最优点与相应算法。

给定非零方向 a 和目标 b ，沿 a 的直线上所有候选向量都写作 ca 。希望 ca 最接近 b ，即最小化 $\|b-ca\|_2$ 。几何上最近点应使残差与直线正交，因此 $a^T(b-ca)=0$ ，解得 $c=(a^Tb)/(a^Ta)$ 。分母 $a^Ta=\|a\|_2^2>0$ ，所以唯一。投影向量 $p=ca$ 是向量，系数 c 是标量，沿单位方向 $a/\|a\|_2$ 的带符号长度为 $(a^Tb)/\|a\|_2$ ；这三个对象不能混写。

为什么正交条件确实给全局最短距离，而不只是图上看起来合理？取残差 $e=b-p$ ，并取任意其他直线候选 w 。因为 $p-w$ 仍沿 a ， e 与 $p-w$ 正交，展开平方得到 $\|b-w\|_2^2=\|e\|_2^2+\|p-w\|_2^2$ 。第二项非负，所以 $w=p$ 时达到唯一最小值。这是勾股定理，也是投影与最小二乘最核心的证明；它不依赖二维图像或求导，在任意有限维子空间同样成立。

写成矩阵时， $p=Pb$ ，其中 $P=aa^T/(a^Ta)$ 。分子 aa^T 是外积矩阵，不是数字 a^Ta ；尺寸要先核对。 P 满足 $P^T=P$ ，称对称； $P^2=P$ ，称幂等，表示已经落在直线上的向量再次投影不会改变。任意幂等矩阵都是某种投影，但只有同时对称才是本书欧氏内积下的正交投影；不能把“投影”二字一律当作垂直投影。

例题 手算：投影系数可以为负，也可以大于 1

沿 $a=(2,1)^T$ 投影 $c=(3.5,0.5)^T$ ，有 $a^Tc=7.5$ 、 $a^Ta=5$ ，所以系数为 1.5，投影 $p=(3,1.5)^T$ ，残差 $e=(0.5,-1)^T$ 。内积 $a^Te=1-1=0$ ，长度平方 $12.5=11.25+1.25$ 。系数超过 1 完全正常，因为投影不是概率。

若目标改为 $-c$ ，系数与投影同时反向，分别为 -1.5 与 $(-3,-1.5)^T$ ；投影长度仍非负。若把方向 a 改为 $10a$ ，投影结果不变，系数改为原来的十分之一，说明方向向量的尺度不影响那条直线。

7.7 标准正交基与正交三角分解

非零向量两两正交时一定线性无关：把零组合与任意一个组内向量作内积，其余项全部消失，留下该向量长度平方乘对应系数，因此系数只能为零。若每个向量长度还等于 1，就称标准正交组。把这些向量排成 Q 的列，得到 $Q^TQ=I$ 。若 Q 为方阵，它还满足 $QQ^T=I$ ，因而 $Q^{-1}=Q^T$ ，称正交矩阵；若 Q 只有 r 列且 $r<m$ ，则 QQ^T 是投影矩阵，通常不是 I_m 。

格拉姆—施密特过程把无关向量变成张成同一空间的标准正交组。先把 a_1 归一化得到 q_1 ；处理 a_2 时减掉它沿 q_1 的投影，剩下的非零部分归一化得到 q_2 ；处理 a_3 时减掉沿 q_1 、 q_2 的两部分，如此继续。每一步只减去已张成空间中的向量，所以新的向量仍在原来的张成空间中；而被减去的部分已保存在之前的方向里，故没有丢失整体空间。若某步剩余恰为零，就说明原列有线性相关，不能继续除以零。

令 $r_{ij}=q_i^T a_j$ ($i \leq j$)，令 r_{jj} 为第 j 步残余长度，则 $a_j = \sum_{i=1}^j r_{ij} q_i$ 。把这些等式并排，得到正交—三角分解 (QR 分解) $A=QR$ ，其中 Q 有标准正交列， R 为上三角。 R 记录原列在新基中的坐标；三角结构来自第 j 列只需要前 j 个新方向。符号 R 此处是 QR 分解因子，不是前文简化行阶梯形，二者由上下文区分。[MIT-L7]

手算格拉姆—施密特适合学习投影如何逐步抽取方向；数值程序通常使用豪斯霍尔德反射等更稳定的方法计算 QR。经典格拉姆—施密特在接近相关的列上会因舍入失去正交性，修正形式可改善这一点。理解理论分解不等于必须在实际求解中逐字照写理论算法；调用成熟的正交—三角分解与奇异值分解 (SVD) 实现能够把数学结构和数值可靠性结合起来。

$$w_j = a_j - \sum_{i=1}^{j-1} (q_i^T a_j) q_i, \quad q_j = \frac{w_j}{\|w_j\|_2} \quad (7.4)$$

a_j 是第 j 个原始列， q_i 是此前构造的单位正交方向， w_j 是去除已有分量后的残余。仅在 $w_j \neq 0$ 时可归一化；首步求和为空， $w_1 = a_1$ 。

例题 手算：用两列构造一组正交平面坐标

取 $a_1=(1,1,0)^T$ 、 $a_2=(1,0,1)^T$ 。首先 $q_1=(1,1,0)^T/\sqrt{2}$ 。 $q_1^T a_2=1/\sqrt{2}$ ，故 $w_2=a_2-(1/2)(1,1,0)^T=(1/2,-1/2,1)^T$ ，长度为 $\sqrt{(3/2)}$ 。于是 $q_2=(1,-1,2)^T/\sqrt{6}$ 。可逐项核验 $q_1^T q_2=0$ 、各自长度为 1。

因此 $Q=[q_1 \ q_2]$ ， $R=[[\sqrt{2}, 1/\sqrt{2}], [0, \sqrt{(3/2)}]]$ 。先乘 QR 重建原两列，再比较 QQ^T 与 I_3 ：前者投影到这两列张成的平面，只保留两个独立方向，所以不可能等于秩为 3 的 I_3 。

7.8 最小二乘与正规方程的两种推导

设 A 为 $m \times n$ ，目标 $b \in \mathbb{R}^m$ 。候选输出 Ax 都在列空间中。最小二乘选择系数 x ，使 $F(x) = \|b - Ax\|_2^2$ 最小；“二乘”指逐分量误差平方相加。把最优系数记作 $\hat{x}$ ，拟合向量记 $p = A\hat{x}$ ，残差记 $e = b - p$ 。它们的尺寸分别为 n 、 m 、 m ；残差的正交性发生在 m 维输出空间。正规方程的最终形式见式 (7.5)。

先从几何推导必要条件。若 p 已是最近点，则沿 A 的任意一列 a_j 微移 $p + t a_j$ ，都不能使距离平方变小。展开 $\|e - t a_j\|_2^2 = \|e\|_2^2 - 2t a_j^T e + t^2 \|a_j\|_2^2$ 。如果 $a_j^T e \neq 0$ ，选择符号相同且足够小的 t 会降低目标，出现矛盾。因此每列都满足 $a_j^T e = 0$ ，合并起来就是 $A^T e = 0$ 。

$$A^T(b - A\hat{x}) = 0, \quad A^T A \hat{x} = A^T b \quad (7.5)$$

把残差 $e = b - A\hat{x}$ 与每列正交的条件展开、移项，就得到正规方程。 $A^T A$ 为 $n \times n$ ， $A^T b$ 为 n 维列；“正规”在此与正交法向有关，与正态分布无关。

同一个结论也可从微分得到。按条目展开 $F(x) = \sum_i (b_i - \sum_j a_{ij} x_j)^2$ ，对第 k 个未知量求偏导：每个平方项的导数为 $-2a_{ik}(b_i - \sum_j a_{ij} x_j)$ 。对 i 求和后，第 k 个偏导就是 $-2[A^T(b - Ax)]_k$ 。因此梯度 $\nabla F(x) = 2A^T Ax - 2A^T b$ ，令各偏导为零仍得到正规方程。

$$F(x) = b^T b - 2x^T A^T b + x^T A^T A x, \quad \nabla F(x) = 2A^T A x - 2A^T b \quad (7.6)$$

x 为 n 维系数， b 为 m 维目标。交叉项 $b^T A x$ 与 $x^T A^T b$ 是同一个标量，所以合成 $-2x^T A^T b$ ； $A^T A$ 对称使二次项梯度为 $2A^T A x$ 。

驻点为什么一定是全局最小，而非局部最大？若 $\hat{x}$ 满足正规方程，对任意 x 写 $b - Ax = e + A(\hat{x} - x)$ 。 e 与 A 的每一列正交，因而与 $A(\hat{x} - x)$ 正交。平方展开中的交叉项为零，得到下式。右侧第二项总非负，故任何别的系数都不可能取得更小误差。证明不需要先假设误差正态。

$$\|b - Ax\|_2^2 = \|b - A\hat{x}\|_2^2 + \|A(\hat{x} - x)\|_2^2 \quad (7.7)$$

等式以 $A^T(b-A\hat{x})=0$ 为前提。最后一项是离开最优拟合向量带来的额外平方损失；列相关时，它可能在 $x \neq \hat{x}$ 时仍为零。

有限维子空间的正交分解保证拟合向量 p 总存在且唯一。如果 A 列无关，对非零 z 有 $z^T A^T A z = \|Az\|_2^2 > 0$ ，故 $A^T A$ 正定可逆，系数也唯一。于是 $\hat{x} = (A^T A)^{-1} A^T b$ ，投影矩阵 $P = A(A^T A)^{-1} A^T$ 。若 A 列相关，正规方程仍有解，但系数可能有无穷多组；这些系数的差都在 $\ker(A)$ ，不能继续使用带普通逆的公式。[MIT-L6]

对列满秩 A ，若已有约化正交—三角分解 $A=QR$ ，则 $p=QQ^T b$ 。由 $QR\hat{x}=p$ 左乘 Q^T 得 $R\hat{x}=Q^T b$ ，再作上三角回代即可。形成 $A^T A$ 会把二范数条件数平方： $\kappa_2(A^T A) = \kappa_2(A)^2$ 。因此正规方程适合推导，实际求解通常用 QR 或 SVD；NumPy 的 `lstsq` 直接解决最小二乘并返回数值秩与奇异值。

例题 手算：三个点怎样成为一个投影问题

观测 $(t,y)=(0,1)$ 、 $(1,2)$ 、 $(2,2)$ ，模型 $y \approx \beta_0 + \beta_1 t$ 。令 $\beta = (\beta_0, \beta_1)^T$ 、 $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$ 、 $b = (1, 2, 2)^T$ 。逐项计算得 $A^T A = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$ 、 $A^T b = (5, 6)^T$ 。第一正规方程是 $3\beta_0 + 3\beta_1 = 5$ ，第二是 $3\beta_0 + 5\beta_1 = 6$ ，相减得 $\beta_1 = 1/2$ ，代回得 $\beta_0 = 7/6$ 。

拟合 $p = (7/6, 5/3, 13/6)^T$ ，残差 $e = (-1/6, 1/3, -1/6)^T$ 。与常数列的内积为 $-1/6 + 1/3 - 1/6 = 0$ ，与时间列的内积为 $0 + 1/3 - 2/6 = 0$ 。残差平方和为 $1/6$ 。图中的三个竖直误差段不是点到拟合线的最短垂线：它们代表三维观测向量 b 与列空间投影 p 之差。

7.9 回归 秩亏与几何解释

对 N 个观测 (t_i, y_i) ，若模型是 $y_i \approx \beta_1 t_i + \beta_0$ ，设计矩阵可按列 $[t, 1]$ 排列，也可按 $[1, t]$ 排列，但未知系数顺序必须同步改变。e32 采用前一种，因此 `coef` 的第一项是斜率、第二项是截距。设计矩阵 A 位于 $\mathbb{R}^{N \times 2}$ ，拟合向量 $\hat{y}$ 位于 $\mathbb{R}^N$ ，而不是二维图上的单个点。每一条候选直线都生成一个 N 维拟合向量，所有这些向量组成 A 的列空间。

含截距的普通最小二乘在列满秩时使残差和为零，因为常数列 1 与残差正交。决定系数 $R^2 = 1 - \text{SSE}/\text{SST}$ ，其中 $\text{SSE} = \sum_i (y_i - \hat{y}_i)^2$ ， $\text{SST} = \sum_i (y_i - \bar{y})^2$ ， $\bar{y}$ 是观测均值；SST 必须大于零。

$RMSE=\sqrt{(SSE/N)}$ 衡量每个观测的均方根拟合误差，单位与 y 相同。它不等于用于统计推断的残差标准差 $\sqrt{(SSE/(N-r))}$ ，后者扣除了估计 r 个独立系数消耗的自由度。

最小二乘本身是几何优化，不需要先假设误差正态。要把系数解释为因果作用，或给出标准误、置信区间，还需要额外的数据生成与误差假设。高 R^2 也不会自动证明模型能外推、变量有因果关系或误差相互独立。几何结论明确告诉我们拟合做了什么；统计结论则要结合前面概率与推断章节进行判断。

若设计列重复，例如 $D=[[1,1],[2,2],[3,3]]$ ， $b=(1,2,3)^T$ ，那么所有 $x_1+x_2=1$ 的系数都实现零残差。写成 $x=(1/2,1/2)^T+t(1,-1)^T$ ，第一部分在行空间，第二部分在零空间，二者正交，长度平方为 $1/2+2t^2$ 。所以最小长度的解是 $(1/2,1/2)^T$ 。这说明最小范数选择为什么排除不可辨认的零空间分量；第 9 章用伪逆把它推广。

程序 `e32_lstsq.py` 固定随机种子 21，30 个 t 从区间 $[1,12)$ 均匀生成， $y=8.5-0.55t$ 加标准差 0.8 的高斯噪声。设计 $A=\text{column_stack}([t,\text{ones}])$ ，输出 $\text{coef}\approx[-0.548847,8.408000]$ 、 $\text{rank}=2$ 、 $R^2\approx0.871989$ 、 $RMSE\approx0.664997$ 。估计值接近生成参数但不要求完全相同，因为样本包含随机噪声。程序核验 $A.T@residual\approx0$ ；另一个独立秩亏问题 D 、 b 给出最小范数解 $[0.5,0.5]$ 。

编程思路 把 $y\approx kt+b$ 写成设计矩阵 $A=[t,1]$ 与系数向量 $[k,b]$ ，用 `lstsq` 求最小二乘解并检验残差正交。另用两列相同的秩亏设计说明最小范数解，三维图则展示真实观测空间中拟合与残差的正交分解。

代码对应

第 i 行写成 $[t,1]$ ，系数顺序为斜率、截距 $\rightarrow A = \text{np.column_stack}([t, \text{np.ones_like}(t)])$ ：列顺序决定 $\text{coef}[0]$ 和 $\text{coef}[1]$ 的含义，与 Norris 案例的 $[1,x]$ 顺序不同。

求使 $\|A\beta - y\|_2$ 最小的 $\beta \rightarrow \text{coef}, \text{sums}, \text{rank}, \text{singular} = \text{np.linalg.lstsq}(A, y, \text{rcond=None})$ ：`lstsq` 同时返回残差平方和信息、秩和奇异值，不需要显式求 $(A.T@A)$ 的逆。

$r=y-A\beta$ ，并检验 $A.T@r\approx0 \rightarrow \text{residual} = y - \text{yhat}$ ：残差与设计矩阵列空间正交，是最小二乘的一阶条件。

秩亏系统的多个最小二乘解中取欧氏范数最小者 $\rightarrow \text{minimum, _, deficient_rank, _} = \text{np.linalg.lstsq(D, b, rcond=None)}$: 相同两列的设计得到 $[0.5, 0.5]$, 再与伪逆结果核对。

易错提示 `lstsq` 返回的 `sums` 在秩亏或非超定条件下可能为空, 不能一律读取 `sums[0]`。几何图使用三维正交坐标表示拟合与残差, 并非把原始 30 维观测随意截成前三个数。

正文代码 e32_lstsq.py

```
# 教材位置: 第 7 章, 第 7.9 节; 图 7-4。
# 单位、变量与形状: t、y、yhat、residual 为 (30,); t 和 y 使用各自一般单位, 均方根误差沿用 y 的单位。
# 案例 32: 线性最小二乘拟合、残差正交与秩亏时的最小范数解。
# 学习目标: 把拟合直线  $y \approx at + b$  写成  $A\beta \approx y$ , 理解最小化平方误差的投影几何。
# 核心关系: 满列秩时  $\beta^*$  为唯一最小二乘解,  $r = y - A\beta^*$  满足  $A^T r = 0$ ;
# 秩亏时最小二乘解可能不唯一, lstsq 返回其中欧氏范数最小的一组系数。
# 数据与形状: 30 个模拟观测, A 为 (30, 2)、coef 为 (2, ), 列顺序是斜率项和截距项。
# 输出: 打印斜率/截距、秩、样本内  $R^2$ /均方根误差、秩亏解; 生成 e32_lstsq.png。
# 运行方式: 安装依赖后在本目录执行 python e32_lstsq.py。
# 图中灰色线是竖直观测残差;  $A^T r = 0$  的正交发生在 30 维观测向量空间。

# NumPy 提供最小二乘、伪逆及用于验证残差几何的矩阵运算。
import numpy as np
# Matplotlib 把观测、拟合直线以及各点的竖直残差画在一起。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 21, 使数据与拟合结果可以重复核对。
rng = np.random.default_rng(21)
# 随机生成 30 个自变量位置; 它们未排序, 拟合本身不要求观测按 t 排序。
t = rng.uniform(1, 12, 30)
# 真实模拟直线截距 8.5、斜率 -0.55, 叠加均值 0、标准差 0.8 的独立高斯噪声。
y = 8.5 - 0.55 * t + rng.normal(0, 0.8, 30)
# 设计矩阵第 1 列为 t, 第 2 列全为 1; 若 coef=[a,b], 则  $A @ \text{coef} = a * t + b$ 。
# 加常数项使模型能拟合非零截距; 不加该列会强制直线通过原点。
A = np.column_stack([t, np.ones_like(t)])
# 直接用数值最小二乘求解, 避免显式构造  $(A^T A)^{-1} A^T y$  而放大条件数问题。
# 返回系数、残差平方和数组、数值秩、奇异值; rcond=None 使用默认奇异值截断准则判断秩。
# 本例是观测数多于参数数且满列秩, 故 sums 含一个残差平方和; 秩亏时该数组可为空。
coef, sums, rank, singular = np.linalg.lstsq(A, y, rcond=None)
# yhat 是 30 个拟合值; residual=r 是原观测与投影的差, 正号表示观测高于拟合线。
yhat = A @ coef
residual = y - yhat
#  $R^2 = 1 - \text{SSE} / \text{SST}$ , 将拟合误差与“所有观测均用样本均值预测”的误差比较。
# 这里计算的是带截距模型的样本内拟合指标, 不代表新数据预测表现。
r2 = 1 - (residual @ residual) / np.sum((y - y.mean()) ** 2)
# 按设计矩阵列顺序打印斜率、截距, 数值秩为 2 表示两列提供独立信息。
print("斜率、截距 =", np.round(coef, 6), "; 秩 =", rank)
# 均方根误差 =  $\sqrt{\text{SSE} / 30}$ , 是每个观测的均方误差开方。
# 它不同于回归噪声标准差估计  $\sqrt{\text{SSE} / (30 - 2)}$ , 后者要扣除拟合的两个参数自由度。
print("决定系数  $R^2$ 、均方根误差 =", round(r2, 6),
      round(np.sqrt(np.mean(residual ** 2)), 6))
# 验证满列秩及残差与设计矩阵每一列都正交, 即最小二乘的一阶条件  $A^T r = 0$ 。
```

```
# 由于含全 1 的截距列, 这也意味着残差和接近零; 不要求每个残差都为零。
assert rank == 2 and np.allclose(A.T @ residual, 0, atol=1e-9)
# 将 lstsq 返回的残差平方和与自行计算的  $r^T r$  比较, 核对返回值的含义。
assert np.isclose(sums[0], residual @ residual)
# 构造两列完全相同的 (3,2) 秩亏矩阵; 只能识别两个系数的和, 无法分别识别。
D = np.array([[1.0, 1.0], [2.0, 2.0], [3.0, 3.0]])
# 右端恰为 D 的任意一列, 因此所有满足  $\beta_1 + \beta_2 = 1$  的系数组合都能实现零残差。
b = np.array([1.0, 2.0, 3.0])
# lstsq 在无穷多个零残差解中选择  $\|\beta\|_2$  最小者; 下划线表示这里不使用对应返回项。
# 在  $\beta_1 + \beta_2 = 1$  约束下, 平均分配为 (0.5, 0.5) 时平方范数最小。
minimum, _, deficient_rank, _ = np.linalg.lstsq(D, b, rcond=None)
# 检查秩为 1, 并确认结果是对称分配的最小范数解。
assert deficient_rank == 1 and np.allclose(minimum, [0.5, 0.5])
# 穆尔-彭罗斯 伪逆  $D^+$  同样给出最小范数最小二乘解, 即  $\beta = D^+ b$ 。
assert np.allclose(minimum, np.linalg.pinv(D) @ b)
# 单独打印秩亏示例, 帮助区分“拟合值唯一”和“系数解唯一”这两个问题。
print("秩亏矩阵的最小范数解 =", minimum)
# 观测 y、拟合 yhat 与残差 residual 都在  $R^{30}$ ; 但它们共同落在一个至多三维的子空间。
# 先把 A 的两列与实际 residual 并列, 再做约化 QR, 得到这个子空间的标准正交基。
# 本例 A 满秩且残差非零, basis 的形状为 (30, 3); 不是随意指定的三维示意坐标。
basis, triangular = np.linalg.qr(np.column_stack([A, residual]), mode="reduced")
observed_coordinates = basis.T @ y
fitted_coordinates = basis.T @ yhat
residual_coordinates = basis.T @ residual
# 在标准正交基中取坐标保持长度、内积与正交关系; 重建应恢复实际 30 维向量。
assert basis.shape == (30, 3) and np.allclose(basis.T @ basis, np.eye(3))
assert np.allclose(basis @ observed_coordinates, y)
assert np.allclose(basis @ fitted_coordinates, yhat)
assert np.allclose(basis @ residual_coordinates, residual)
assert np.allclose(observed_coordinates, fitted_coordinates + residual_coordinates)
assert np.allclose(fitted_coordinates[2], 0, atol=1e-10)
assert np.allclose(residual_coordinates[:2], 0, atol=1e-10)
assert np.isclose(fitted_coordinates @ residual_coordinates, 0, atol=1e-9)
assert np.isclose(np.linalg.norm(residual_coordinates), np.linalg.norm(residual))
# 创建拟合图, 左图横轴为自变量 t, 纵轴为响应 y; 右图是实际观测子空间的 QR 坐标。
fig = plt.figure(figsize=(6.3, 3.2))
ax = fig.add_subplot(1, 2, 1)
# 散点不必排序; 另外取得按 t 升序的索引, 以便画线时从左到右连接。
ax.scatter(t, y, label="模拟观测", s=18)
order = np.argsort(t)
# 按同一索引排列 t 与拟合值, 避免原始随机顺序造成来回折返的线段。
ax.plot(t[order], yhat[order], "r-", label="最小二乘拟合")
# 在固定 t 上连接 y 与 yhat, 画的是竖直方向的回归残差。
# 最小二乘的残差正交是 r 与 A 的列空间在观测空间中正交, 并非图上灰线垂直于斜拟合线。
ax.vlines(t, y, yhat, color="gray", alpha=0.5)
# 标题用 A.T@residual=0 提醒读者区分散点图中的竖直距离与向量空间中的正交。
ax.set(xlabel="t", ylabel="y", title="观测值的竖直残差")
# 图例区分观测与拟合线, 淡网格便于读出纵向残差大小。
ax.legend(fontsize=7.8)
ax.grid(alpha=0.3)
ax = fig.add_subplot(1, 2, 2, projection="3d")
# 前两条 QR 基张成 Col(A), 所以拟合平面在第三坐标为 0 处; 平面范围由实际拟合坐标决定。
# 以下只是设置显示范围, 没有对坐标向量重标度, 也没有改动原观测数据。
extent = 1.2 * max(np.max(np.abs(observed_coordinates)), 1.0)
plane_axis = np.linspace(-extent, 0.2 * extent, 2)
plane_x, plane_y = np.meshgrid(plane_axis, plane_axis)
ax.plot_surface(plane_x, plane_y, np.zeros_like(plane_x), color="tab:green",
alpha=0.18)
```

```
# y 是从原点到实际观测坐标的向量, yhat 是其平面投影, r 从投影端点指向观测端点。
ax.quiver(0, 0, 0, *observed_coordinates, color="tab:blue", arrow_length_ratio=0.07)
ax.quiver(0, 0, 0, *fitted_coordinates, color="tab:green", arrow_length_ratio=0.07)
ax.quiver(*fitted_coordinates, *residual_coordinates, color="tab:orange",
          arrow_length_ratio=0.18)
ax.text(*observed_coordinates, " $y$", color="tab:blue", fontsize=9)
# 标签做纯显示偏移, 避免与拟合向量端点和直角记号重叠; 向量坐标仍保持原值。
fitted_label = fitted_coordinates + np.array([0.16 * extent, 0.03 * extent,
        -0.15 * np.linalg.norm(residual_coordinates)])
ax.text(*fitted_label, " $\hat{y}$", color="tab:green", fontsize=9)
ax.text(*(fitted_coordinates + residual_coordinates / 2), " $r$",
        color="tab:orange", fontsize=9)
# 直角记号沿拟合平面的一条单位坐标轴与实际残差方向绘制, 表示  $A^T r = 0$ 。
marker_size = np.linalg.norm(residual_coordinates) * 0.18
normal = residual_coordinates / np.linalg.norm(residual_coordinates)
in_plane = np.array([1.0, 0.0, 0.0])
right_angle = np.array([fitted_coordinates + marker_size * in_plane,
        fitted_coordinates + marker_size * (in_plane + normal),
        fitted_coordinates + marker_size * normal])
ax.plot(*right_angle.T, color="gray", lw=0.8)
ax.set_title("实际观测向量的子空间", fontsize=10)
ax.set(xlim=(-extent, 0.2 * extent), ylim=(-extent, 0.2 * extent),
        zlim=(-1.0, 1.5 * abs(residual_coordinates[2]) + 0.5),
        xlabel="$q_1$", ylabel="$q_2$", zlabel="$q_3$")
ax.set_box_aspect((1, 1, 0.5))
ax.set_proj_type("ortho")
ax.view_init(elev=24, azim=-54)
ax.tick_params(labelsize=6, pad=0)
# 一般最小二乘公式中的 b 在本回归中就是 y; 代码较早的 b=[1,2,3] 属于另一秩亏例子。
ax.text2D(0.02, 0.01, "实际正交基坐标;  $A^T r = 0$ ", transform=ax.transAxes, fontsize=7.8)
# 自动调整边距、保存图片并关闭图对象; tight 边框防止三维轴标签被裁切。
fig.tight_layout()
fig.savefig("e32_lstsq.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

斜率、截距 = [-0.548847 8.408] ; 秩 = 2

决定系数 R^2 、均方根误差 = 0.871989 0.664997

秩亏矩阵的最小范数解 = [0.5 0.5]

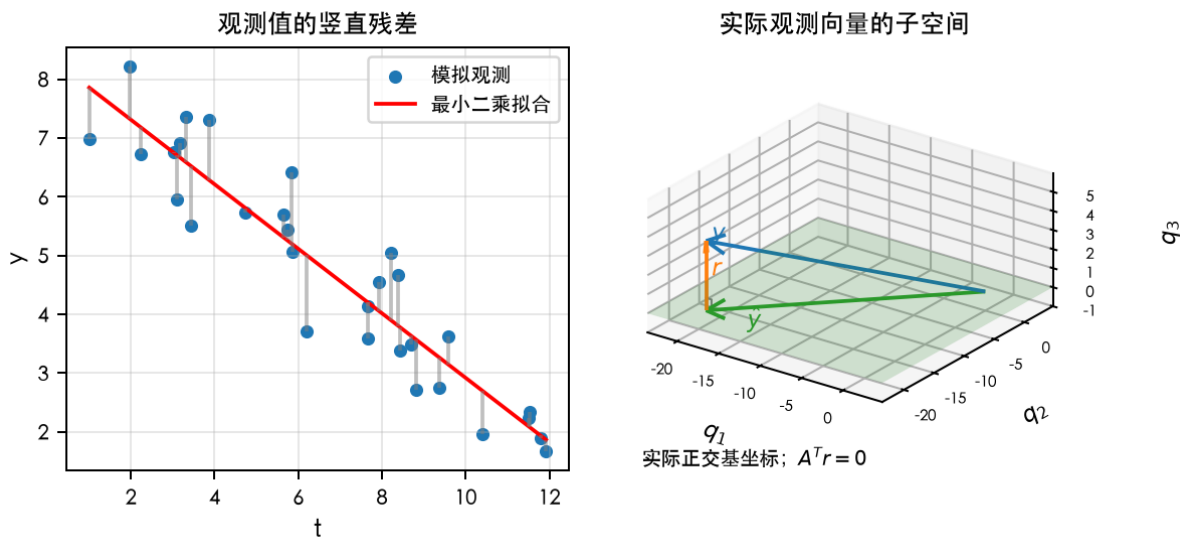


图 7-4 e32_lstsq.py。左：30 个观测与拟合直线，竖线是数据平面的残差。右：以 QR 生成的 30×3 标准正交基表示 y 、 $\hat{y}$ 与 residual 这三个 30 维向量；拟合向量在列空间平面内，残差与该平面垂直，三维图精确表达它们所在的子空间。

读图指南 左图按同一个 t 比较观测点与拟合线，灰色竖段是响应差；右图用 QR 坐标呈现同一 30 维观测、拟合与残差，橙色残差垂直绿色列空间平面。两图分别位于数据平面与观测向量空间。

散点图的一条拟合直线位于“自变量 t —响应 y ”的二维平面，横轴 t 在普通最小二乘中视为给定，优化只调整各点的预测响应。因此左图线段沿纵轴，长度是 $y_i - \hat{y}_i$ 。第 7 章投影理论所在的观测空间，却把 30 个响应依次排成一个向量 $y \in \mathbb{R}^{30}$ ；设计列 t 与 1 张成其中一个二维平面，拟合是在这个高维平面里选一个向量。两幅图使用不同的点和不同的轴。

若观测自变量也有明显误差，并希望最小化二维数据点到直线的垂直距离，那是总最小二乘或正交回归等另一类问题，通常需要 SVD 分析。不能因为“垂直残差”听起来更几何，就擅自替换普通回归的目标。先说明误差发生在哪些量上、距离按什么尺度衡量，再选择拟合方法，几何才能真正服务模型。

7.10 矩阵与统计的桥梁 中心化及协方差

一系列数据围绕均值变化，多个变量则会一起变化。设 X 为 $N \times d$ 数据矩阵，每行一条观测、每列一个变量， $N \geq 2$ 。列均值 $\mu_j = (1/N) \sum_i X_{ij}$ ；把 μ 记成 d 维列， $\mathbf{1}$ 记成 N 维全 1 列，则 $X_c = X - \mathbf{1}\mu^T$ 为中心化矩阵。它只减去平均位置，既不改变变量的单位，也不自动使方差相等。

$$S = \frac{X_c^T X_c}{N-1}, \quad S_{jk} = \frac{1}{N-1} \sum_{i=1}^N (X_{ij} - \mu_j)(X_{ik} - \mu_k) \quad (7.8)$$

S 是 $d \times d$ 样本协方差矩阵； j, k 标识变量， i 标识观测， $N-1$ 是通常的样本协方差分母。对角 S_{jj} 为第 j 变量样本方差，非对角 S_{jk} 为两变量样本协方差。

S 对称，因为交换两个变量不改变乘积。对任意系数列 $a \in \mathbb{R}^d$ ， $a^T S a = \|X_c a\|_2^2 / (N-1) \geq 0$ ，所以 S 半正定；如果中心化各列线性相关，它就可能有零特征值。样本协方差是由这一份样本算出的统计量，不能与总体协方差矩阵 Σ 混同。总体 $\Sigma_{jk} = E[(Y_j - E Y_j)(Y_k - E Y_k)]$ 要求随机向量 Y 各分量有有限二阶矩。这里大写 Σ 是协方差矩阵名称；到 SVD 中同一字母改表示矩形奇异值矩阵，需根据定义区分。

线性组合的方差正是这个二次型。令随机变量 $Z = a^T Y$ ，均值为 $a^T E Y$ ，则 $Z - E Z = a^T (Y - E Y)$ 。把平方写为 $a^T (Y - E Y)(Y - E Y)^T a$ 并取期望，得到 $\text{Var}(Z) = a^T \Sigma a$ 。样本版本中， $z = X a$ 的中心化列为 $X_c a$ ，故样本方差 $s^2(z) = a^T S a$ 。非对角协方差贡献交叉项，不能只把各变量方差按权重平方相加，除非相关协方差为零。

$$\text{Var}(a^T Y) = a^T \Sigma a, \quad s^2(X a) = a^T S a \quad (7.9)$$

a 为固定 d 维权重； Y 为随机向量， Σ 为总体协方差， X 为 $N \times d$ 观测表， S 为样本协方差。两个式子结构一致，但左边一个是总体分布量，一个是样本计算量。

例题 手算：同向变化怎样影响组合波动

取四条模拟观测 $X = [[1, 2], [2, 1], [3, 5], [4, 4]]$ 。均值 $\mu = (2.5, 3)^T$ ，中心化各行 $(-1.5, -1)$ 、 $(-0.5, -2)$ 、 $(0.5, 2)$ 、 $(1.5, 1)$ 。第一列平方和 5，第二列平方和 10，两列乘积和 5，所以 $S = [[5/3, 5/3], [5/3, 10/3]]$ 。

组合权重 $a=(0.6,0.4)^T$ 产生 $z=(1.4,1.6,3.8,4.0)^T$ ，均值 2.7。离均差平方和为 $1.69+1.21+1.21+1.69=5.8$ ，样本方差 $5.8/3=1.933333\dots$ 。二次型也给出 $0.6^2 \times 5/3 + 2 \times 0.6 \times 0.4 \times 5/3 + 0.4^2 \times 10/3 = 1.933333\dots$ 。中间交叉项 0.8 反映两变量的共同波动。

程序 `e38_covariance_matrix.py` 输入固定为四行两列模拟数据 $X=[[1,2],[2,1],[3,5],[4,4]]$ ，权重 $a=[0.6,0.4]$ 。沿 `axis=0` 计算列均值 $[2.5,3]$ ， $S=Xc.T@Xc/3$ 得到 $[[1.666667,1.666667],[1.666667,3.333333]]$ 。`np.var(X@a,ddof=1)` 与 $a@S@a$ 均约 1.9333333333；对称特征值约 $[0.63661002,4.36338998]$ ，均非负。左图显示观测、样本中心和权重方向，右图显示程序记作 C 的协方差矩阵；组合方差见运行输出。正文用 S 表示该样本矩阵。

编程思路 对四行二维观测逐列中心化，手算散布矩阵再除以 $n-1$ 得到样本协方差。把组合变量 $0.6X_1+0.4X_2$ 的直接样本方差与 $a.T C a$ 及展开式三路核对，并检查协方差矩阵半正定。

代码对应

$Xc.T Xc=[[5,5],[5,10]] \rightarrow cross_products = Xc.T @ Xc$ ：每个元素都是两列中心化偏差乘积的总和，非对角项体现共同变化。

$C=Xc.T Xc/3=[[5/3,5/3],[5/3,10/3]] \rightarrow C = cross_products / (n - 1)$ ：四个样本估计均值后使用 $n-1=3$ ，与 `np.cov` 的 `ddof=1` 一致。

对四个组合值 $[1.4,1.6,3.8,4]$ 求样本方差 $\rightarrow direct_var = combined.var(ddof=1)$ ：直接以组合观测算得约 1.9333333333，与二次型路径独立对应。

$Var(0.6X_1+0.4X_2)=a.T C a \rightarrow quadratic_var = a @ C @ a$ ：一维 a 的转置不改变形状，但 $a@C@a$ 仍按乘法顺序产生标量二次型。

易错提示 `np.cov` 默认将每行当变量；本例变量在列，需 `rowvar=False`。半正定允许特征值为零，不能误写为所有协方差矩阵都正定；浮点检查也需允许极小负舍入误差。

正文代码 `e38_covariance_matrix.py`

```
# 教材位置：第 7 章，第 7.10 节；图 7-5。
# 从四个二维观测构造样本协方差，再核对线性组合的样本方差。
# X 形状 (4,2)：行是观测，列是特征；这里两列均用无量纲教学数值。
# mean 形状 (2,)，Xc 形状 (4,2)，C 形状 (2,2)，a 形状 (2,)
```

```
# 核心关系:  $C = X_c.T @ X_c / (n-1)$ ,  $\text{var}(X@a, \text{ddof}=1) = a.T @ C @ a$ .
# 输入固定, 不随机抽样; 半正定性来自任何方向方差均非负。
# 输出中心、中心化平方和、样本协方差、特征值及两条方差计算路径。
# 执行 python e38_covariance_matrix.py, 图片写入当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

X = np.array([[1, 2], [2, 1], [3, 5], [4, 4]], dtype=float)
n = X.shape[0]
# axis=0 消去观测这一维, 每列保留一个均值。
mean = X.mean(axis=0)
# 广播把两个列均值复制到四行, 不对行向量逐行分别中心化。
Xc = X - mean
cross_products = Xc.T @ Xc
C = cross_products / (n - 1)
# 手算: 列均值 (2.5, 3), 平方和为 5、10, 交叉乘积和为 5。
assert np.allclose(mean, [2.5, 3.0])
assert np.allclose(cross_products, [[5, 5], [5, 10]])
manual_cov = np.array([[5 / 3, 5 / 3], [5 / 3, 10 / 3]])
assert np.allclose(C, manual_cov)
# np.cov 默认把行视为变量; rowvar=False 才与本例“列为特征”一致。
assert np.allclose(C, np.cov(X, rowvar=False, ddof=1))

a = np.array([0.6, 0.4])
combined = X @ a
# combined=[1.4, 1.6, 3.8, 4]; 它的样本均值为 2.7。
direct_var = combined.var(ddof=1)
quadratic_var = a @ C @ a
manual_var = 0.6**2 * (5 / 3) + 2 * 0.6 * 0.4 * (5 / 3)
manual_var += 0.4**2 * (10 / 3)
assert np.allclose(combined, [1.4, 1.6, 3.8, 4.0])
assert np.isclose(direct_var, quadratic_var)
assert np.isclose(quadratic_var, manual_var)
# C 对称, 可用 eigvalsh; 浮点核验允许很小的负舍入误差。
eigenvalues = np.linalg.eigvalsh(C)
assert np.allclose(C, C.T)
assert eigenvalues.min() >= -1e-12
# 这里特征值均正, 故四个观测在二维中并非完全落在同一条直线上。
print("列均值: ", mean)
print("中心化后的矩阵: \n", Xc)
print("交叉乘积和矩阵: \n", cross_products)
print("样本协方差矩阵: \n", C)
print("协方差矩阵的特征值: ", eigenvalues)
print("线性组合观测及均值: ", combined, combined.mean())
print("直接样本方差、二次型方差、手算方差: ",
      direct_var, quadratic_var, manual_var)

# 左图保留原始观测和中心, 右图逐元素显示 C, 帮助对应三个手算和。
configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1), constrained_layout=True)
axes[0].scatter(X[:, 0], X[:, 1], label="四个观测")
axes[0].scatter(*mean, marker="x", s=70, color="red", label="样本中心")
for index, point in enumerate(X, 1):
    axes[0].annotate(str(index), point, xytext=(4, 4), textcoords="offset points")
axes[0].quiver(*mean, *a, angles="xy", scale_units="xy", scale=1, color="gray")
axes[0].set(title="观测与线性组合方向 a", xlabel="第一特征", ylabel="第二特征")
```

```
axes[0].set_aspect("equal")
axes[0].legend(fontsize=7.5)
axes[0].grid(alpha=0.2)
axes[1].imshow(C, cmap="Blues", vmin=0, vmax=C.max() * 1.3)
for i in range(2):
    for j in range(2):
        axes[1].text(j, i, f"{C[i, j]:.4f}", ha="center", va="center")
axes[1].set(xticks=[0, 1], yticks=[0, 1],
            xticklabels=["第一特征", "第二特征"],
            yticklabels=["第一特征", "第二特征"], title="样本协方差 C")
fig.savefig("covariance_matrix.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

列均值: [2.5 3.]

中心化后的矩阵:

```
[[-1.5 -1. ]
 [-0.5 -2. ]
 [ 0.5  2. ]
 [ 1.5  1. ]]
```

交叉乘积和矩阵:

```
[[ 5.  5.]
 [ 5. 10.]
```

样本协方差矩阵:

```
[[1.66666667 1.66666667]
 [1.66666667 3.33333333]]
```

协方差矩阵的特征值: [0.63661002 4.36338998]

线性组合观测及均值: [1.4 1.6 3.8 4.] 2.7

直接样本方差、二次型方差、手算方差: 1.9333333333333333 1.9333333333333336
1.9333333333333333

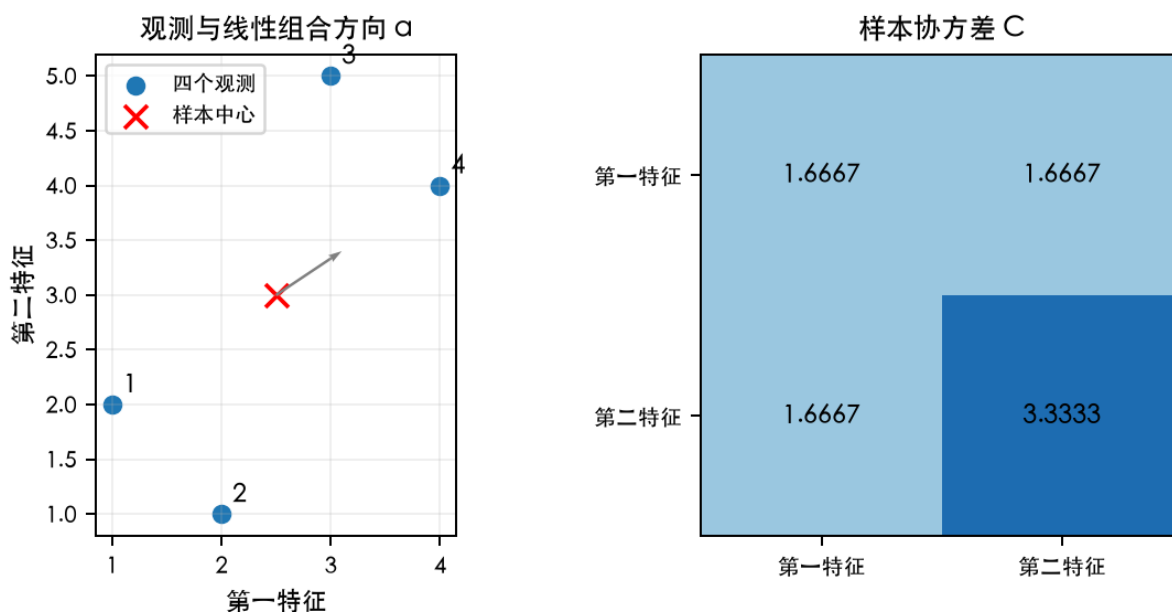


图 7-5 四条模拟观测的样本协方差与线性组合方向。图中 C 即正文的样本协方差 S； $\alpha=(0.6,0.4)^T$ 的组合样本方差约 1.933333。

读图指南 先核对左图四个点、红色均值中心和灰色权重方向，再读右图对角方差与非对角协方差。结合运行输出对照组合样本方差与 $\alpha^T S \alpha$ ；两种计算使用同一份 X 与分母 $N-1$ 。

矩阵形式把第 2 章的协方差公式与后续 PCA 连接起来。寻找“变化最大的变量组合”，就是在单位方向 α 上最大化 $\alpha^T S \alpha$ ；第 8 章用实对称谱定理描述这种二次型，第 9 章再证明主成分方向就是中心化数据的右奇异方向。方差大的方向是样本中变化显著的方向，仍需结合变量单位和问题目标解释。

7.11 练习与参考解答

练习 7-1（概念与手算，入门） 分别解 $x+2y=3$ 、 $2x+4y=6$ 与 $x+2y=3$ 、 $2x+4y=7$ ，给出秩与解集。

练习 7-2（概念与手算，入门） 把 $b=(2,0,1)^T$ 投影到 $a=(1,1,1)^T$ 张成的直线。求投影、残差及最短距离。

练习 7-3（概念与手算，进阶） Q 为 5×2 且 $Q^T Q = I_2$ 。 $Q Q^T$ 是否为 I_5 ？其作用是什么？

练习 7-4（概念与手算，进阶） $D=\text{diag}(1000,0.001)$ 的行列式和二范数条件数各为多少？为何行列式为 1 仍可敏感？

练习 7-5 (编程实践, 入门) 运行 e23_solve.py, 核对方程 $A = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$ 、 $b = [9, 5]$ 的解, 并验收病态例 C 两个右端对应的解。

练习 7-6 (编程实践, 入门) 对 $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$ 、 $b = [1, 2, 2]$ 调用 `np.linalg.lstsq`。核对系数、秩、残差平方和以及 $A.T @ \text{residual}$ 。

练习 7-7 (编程实践, 进阶) 复现 e38 中的四行数据与 $a = [0.6, 4]$, 用矩阵乘法和 `np.cov` 分别求 S , 并验收组合方差。`np.cov` 需要怎样指定变量方向?

练习 7-8 (编程实践, 进阶) 令 $D = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$ 、 $b = [1, 2, 3]$ 。比较 `lstsq` 返回系数与该系数加 $[1, -1]$ 后的拟合、残差和系数范数。

练习 7-9 (场景应用, 入门) 在一项明确为教学模拟的标定中, 输入 $t = 0, 1, 2$, 对应输出 $y = 1, 3, 4$ 。拟合 $y \approx \beta_0 + \beta_1 t$, 给出设计矩阵、系数与残差, 并说明为什么误差段是竖直的。

练习 7-10 (场景应用, 进阶) 两个模拟指标的样本协方差为 $S = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$ 。组合 $a = (1/2, 1/2)^T$ 的方差是多少? 把两指标误当不相关会少算多少?

练习 7-11 (场景应用, 进阶) 一个模拟模型同时使用“日活用户数”和其两倍作为两列解释变量, 并含截距。样本量再多是否保证两个斜率可分别识别? 如何报告?

练习 7-12 (场景应用, 拓展) 固定可逆 A 、非零 b , 已算相对残差为 10^{-6} , $\kappa_2(A) = 10^4$ 。能得到怎样的相对解误差保证? 如果真实输入数据本身有误差, 这个保证还覆盖什么?

参考解答与验收依据

7-1 第二行减 2 倍第一行。第一组得到 $0=0$, $\text{rank}(A) = \text{rank}([A|b]) = 1$, 全部解 $(3, 0)^T + t(-2, 1)^T$; 第二组得到 $0=1$, $\text{rank}(A) = 1$ 、 $\text{rank}([A|b]) = 2$, 无解。

7-2 $a^T b = 3$ 、 $a^T a = 3$, 系数 1, $p = (1, 1, 1)^T$, $e = (1, -1, 0)^T$, $a^T e = 0$, 距离 $\sqrt{2}$ 。对任意候选 $w = ta$, $\|b - w\|_2^2 = 2 + \|p - w\|_2^2$, 所以 p 为唯一最近点。

7-3 不是。 QQ^T 为 5×5 对称幂等投影矩阵, 秩 2, 投到 Q 两列张成的二维子空间; I_5 的秩是 5。只有 Q 为方阵且正交时才有 $QQ^T = I$ 。

7-4 $\det(D)=1$ ，最大与最小伸缩之比为 $1000/0.001=10^6$ ，所以 $\kappa_2(D)=10^6$ 。行列式把各方向伸缩相乘，条件数比较最大与最小。条件数给最坏相对误差放大的上界，不保证每次扰动恰放大 10^6 倍。

7-5 主例解应约 $[2.714286, 3.142857]=[19/7, 22/7]$ ， $A @ x$ 与 b 近似相同，秩 2、行列式 -7 、条件数约 1.456083。 $C=[[1,1],[1,1.0001]]$ 对 $b_1=[2,2]$ 与 $b_2=[2,2.0001]$ 分别给 $[2,0]$ 与 $[1,1]$ ，条件数约 40002。

7-6 系数 $[7/6, 1/2]$ ，秩 2；拟合 $[7/6, 5/3, 13/6]$ ，残差 $[-1/6, 1/3, -1/6]$ ，平方和 $1/6$ 。 $A.T @ residual$ 应在舍入容差内为零。建议分别核对这些数学量，不要求浮点输出逐字符相等。

7-7 X 每列是变量，所以用 $np.cov(X, rowvar=False, ddof=1)$ ，结果应等于 $Xc.T @ Xc/3$ ，约 $[[1.666667, 1.666667], [1.666667, 3.333333]]$ 。 $np.var(X@a, ddof=1)$ 与 $a @ S @ a$ 均约 1.9333333333。若不改默认 $rowvar$ ，会误把四行当四个变量，尺寸也错。

7-8 $lstsq$ 返回最小范数解约 $[.5, .5]$ ，秩 1、拟合 b 、残差近零。加 $[1, -1]$ 得到 $[1.5, -.5]$ ，拟合仍为 b ；系数范数平方从 .5 增至 2.5。相同拟合不意味着系数唯一。

7-9 $A=[[1,0],[1,1],[1,2]]$ ， $A^T b=(8,11)^T$ ，解 $\beta_0=7/6$ 、 $\beta_1=3/2$ 。拟合 $[7/6, 8/3, 25/6]$ ，残差 $[-1/6, 1/3, -1/6]$ ，平方和 $1/6$ 。普通最小二乘把 t 视为给定，只优化 y 方向误差；若 t 也有测量误差，需要另建模型。

7-10 $a^T S a=1+1+1=3$ 。只算对角权重平方得到 2，漏掉 $2 \times (1/2) \times (1/2) \times 2=1$ 的协方差交叉项。正共同波动会提高这份同向组合的方差。

7-11 两列始终线性相关，增加同结构样本不会恢复列满秩。只能识别 $\beta_1+2\beta_2$ 等组合，两个斜率可沿核方向改变而不影响拟合。应删除冗余列、重新参数化，或提供真正独立的外部信息；可报告拟合向量与可识别组合，不能把最小范数系数包装成唯一机制。

7-12 对精确模型中固定 A 、右端 b 的近似解，相对解误差至多 $10^4 \times 10^{-6}=10^{-2}$ ，即 1%。这只是上界，实际可能更小。它不自动覆盖 A 本身的测量误差、模型偏差或使用错误单位；这些误差需另外分析。

7.12 核心要点与常见误区

核心要点：精确有解等价于 b 在列空间，全部解是一个特解加核。行操作作用于整个增广矩阵，并保持解集；主元与自由变量把结构变成计算。

核心要点：投影残差与可达子空间正交。正规方程既可由每列方向的距离变化推出，也可展开平方求导；全局最优来自正交勾股分解。协方差矩阵把线性组合方差写成二次型。

常见误区：小残差须结合条件数解释；条件数是最坏上界，不是每次必然放大倍数。拟合向量唯一不保证系数唯一。普通回归图中的竖直误差段不是数据平面里点到斜线的最短距离。

7.13 延伸阅读 入门 进阶与拓展

入门阅读：读 18.06SC“矩阵消元”的 Ses1.2sum，手写增广矩阵并完成回代（来源 MIT-L2；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/ax-b-and-the-four-subspaces/elimination-with-matrices/>）。

进阶阅读：看第 16 讲“投影矩阵与最小二乘”并读第五版 4.3 节，逐步解释每个正规方程为何对应残差正交（来源 MIT-L6；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/least-squares-determinants-and-eigenvalues/projection-matrices-and-least-squares/>）。

拓展阅读：读官方公开“范数与条件数”补充，比较相对残差、相对解误差和最坏界；再结合 QR 理解稳定求解（来源 LA-B3；公开链接：<https://math.mit.edu/~gs/linearalgebra/ila6/ila5conditionnumbers.pdf>）。

第 8 章 特征值 正定矩阵与马尔可夫链

8.1 三级学习路线

入门路线：先手算一个二阶特征问题，再用对角化计算重复变换。进阶路线：学习实对称谱定理与正定二次型，并把特征值 1 连接到概率稳态。拓展路线：辨别重复根、缺少特征向量、周期链与可约链的反例，理解结论各自需要的条件。

第 6 章的矩阵表达输入与输出的关系；在本章，特征问题要求同一个向量能与它的像比较，因此先要求 A 为方阵。实矩阵的特征值也可能是复数；主线案例均能在实数范围解释，涉及一般结论时会明确说明适用范围。MIT18.06SC 将特征值、对角化与矩阵幂连续安排，并用马尔可夫矩阵展示结构如何控制长期行为。[MIT-L8、MIT-L9]

8.2 特征值是方向上的伸缩因子

设 $A \in \mathbb{R}^{n \times n}$ 。若存在非零向量 v 和标量 λ 使 $Av = \lambda v$ ，则 λ 称为 A 的特征值， v 称为对应特征向量。希腊字母 λ 读作“拉姆达”。 v 必须非零，否则 $A0 = \lambda 0$ 对任何 λ 都成立，没有区分作用。 λ 可以为零，此时 v 属于零空间； $\lambda < 0$ 时输出反向； $0 < \lambda < 1$ 时沿该方向收缩； $\lambda > 1$ 时放大。复特征值没有直接的实数伸缩解释，需要复向量空间或二维旋转块理解。

如果 v 是特征向量，任意非零 cv 也都是同一特征值的特征向量，因此方向比某一组具体坐标更重要。求特征向量时常把长度规范成 1，但 v 与 $-v$ 仍都正确。 λ 对应的全部特征向量加上零向量组成特征空间 $E_\lambda = N(A - \lambda I)$ 。它是子空间，而“所有特征向量的集合”本身不含零向量；不同特征值的特征向量相加一般也不再是特征向量。

由 $Av = \lambda v$ 移项得到 $(A - \lambda I)v = 0$ 。要存在非零解， $A - \lambda I$ 必须不可逆，因此 $\det(A - \lambda I) = 0$ 。左边是 λ 的 n 次多项式，称为特征多项式，根是特征值，重复次数称代数重数。找到 λ 后仍须解对应齐次方程求 E_λ ；仅解出多项式并未完成特征分解。特征空间维数称几何重数，它至少为 1 且不超过相应代数重数。

$$Av = \lambda v \quad (v \neq 0), \quad E_\lambda = N(A - \lambda I), \quad \det(A - \lambda I) = 0 \quad (8.1)$$

A 是 $n \times n$ 方阵, I 为同阶单位矩阵, λ 为标量, v 为非零 n 维列向量。 $E\lambda$ 包括零向量, 因而是一个空间; 特征向量定义则排除零向量。

例题 手算: 对称矩阵的两个特征方向

取 $A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ 。行列式 $\det(A - \lambda I) = (3 - \lambda)(2 - \lambda) - 1 = \lambda^2 - 5\lambda + 5$ 。解二次方程得 $\lambda_- = (5 - \sqrt{5})/2 \approx 1.381966$, $\lambda_+ = (5 + \sqrt{5})/2 \approx 3.618034$ 。对任意一个 λ , 第一行给出 $(3 - \lambda)v_1 + v_2 = 0$, 可取 $v = (1, \lambda - 3)^T$, 再除以自身长度得到单位特征向量。

这两条方向正交, 因为 $(1, \lambda_- - 3)^T$ 与 $(1, \lambda_+ - 3)^T$ 的内积为 $1 + (\lambda_- - 3)(\lambda_+ - 3) = 0$, 其中利用 $\lambda_- + \lambda_+ = 5$, $\lambda_- \lambda_+ = 5$ 。一般非对称矩阵的不同特征方向不保证正交, 不能把这个算例的图形经验直接推广。

矩阵迹 $\text{tr}(A) = \sum_i a_{ii}$ 是主对角元素之和。若按代数重数列出全部复特征值 $\lambda_1, \dots, \lambda_n$, 则 $\sum_i \lambda_i = \text{tr}(A)$, $\prod_i \lambda_i = \det(A)$ 。 $\prod$ 表示连乘。这两式是核对计算结果的有效办法, 但核对通过并不能代替 $Av = \lambda v$ 的逐列验证。对本例, 两特征值之和为 5, 与迹相同; 乘积为 5, 与行列式相同。

需要特别区分特征值与奇异值: 特征值描述同一空间中不偏转的方向, 可能为负或复数; 奇异值描述输入输出之间的非负伸缩倍率, 总是实数。只有对称正定矩阵等特殊情形, 按顺序排列后两者可以相同。把某矩阵特征值的绝对值直接叫作奇异值, 一般会得到错误结果。

程序 `e25_eigen.py` 固定 $A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$, 用适合实对称矩阵的 `eigh` 得到升序特征值 $[1.381966, 3.618034]$ 。`vectors[:, j]` 是第 j 个单位特征向量, 程序按列验证 $A @ \text{vectors} \approx \text{vectors} * \text{values}$ 并核验正交性。`trace=5`、`determinant=5`。图中单位圆经 A 映成椭圆, 由于本例对称正定, 半轴长恰为两个正特征值; 这个相等关系依赖本例条件。

编程思路 对称矩阵 $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ 先由特征多项式得到两个解析特征值, 再用 `eigh` 求正交特征向量。检查 $Av = \lambda v$ 、正交性、迹与行列式, 最后把单位圆变换成椭圆展示主方向。

代码对应

解 $\lambda^2 - 5\lambda + 5 = 0$, 得到 $(5 \pm \sqrt{5})/2 \rightarrow \text{values}$, `vectors = np.linalg.eigh(A)`: `eigh` 专用于实对称矩阵, 返回升序特征值及按列排列的单位特征向量。

把手算的两个根写成数组 $\rightarrow \text{manual} = \text{np.array}([(5 - \text{np.sqrt}(5)) / 2, (5 + \text{np.sqrt}(5)) / 2])$: 与 `values` 按相同升序比较, 避免因排列不同误判数值不一致。

将单位圆上的每点乘以 $A \rightarrow \text{ellipse} = A @ \text{circle}$: `circle` 的每列是一个点, 左乘 A 符合列向量约定, 特征方向对应伸缩轴。

易错提示 特征向量可整体变号且仍正确, 不能以符号不同判断算法错误。对称矩阵可用正交特征基; 任意非对称矩阵不具有这项保证。

正文代码 e25_eigen.py

```
# 教材位置: 第 8 章, 第 8.2 节; 图 8-1。
# 单位、变量与形状: A 为 (2,2), 特征向量按列组成 (2,2); 坐标单位统一, 特征值是轴向缩放倍数。
# 案例 25: 用对称正定矩阵理解特征值、特征向量与几何变换。
# 学习目标: 把  $Av=\lambda v$  的代数关系与“沿特征方向只发生伸缩”的图像联系起来,
# 并检验实对称矩阵的特征向量可以构成一组标准正交基。
# 核心关系:  $A=Q \text{diag}(\lambda) Q^T$ ,  $\text{tr}(A)=\sum \lambda$ ,  $\det(A)=\prod \lambda$ ; 本例两个特征值均为正。
# 数据与形状: A 为 2×2 矩阵; vectors 的每一列是一个特征向量;
# 单位圆用一个 2×200 数组表示, 每一列都是平面上的一个点。
# 输出: 打印特征值、迹和行列式; 断言核对恒等式; 生成 e25_eigen.png。
# 运行方式: 安装本目录 requirements.txt 中的依赖, 在本目录执行
# python e25_eigen.py。图片保存到运行时的当前工作目录, 不弹出交互窗口。

# NumPy 负责矩阵计算; 约定 @ 表示矩阵乘法, * 表示逐元素运算或广播。
import numpy as np
# Matplotlib 负责把单位圆及其线性变换后的椭圆画在同一坐标系内。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
#  $A=A^T$ , 因此适用实对称矩阵专用的 eigh; 浮点数便于后续数值线性代数计算。
A = np.array([[3.0, 1.0], [1.0, 2.0]])
# eigh 按特征值从小到大返回 values, 形状为 (2,); vectors 形状为 (2,2)。
# 第 j 列 vectors[:,j] 对应 values[j], 不是第 j 行; 各列长度为 1 且两两正交。
# 特征向量的正负号不唯一, 换成相反方向仍是同一个特征方向。
values, vectors = np.linalg.eigh(A)
# 手算  $\det(A-\lambda I)=\lambda^2-5\lambda+5=0$ , 得到  $(5\pm\sqrt{5})/2$ ; 按升序排好以与 eigh 对齐。
# 两根均为正, 所以 A 在任意非零方向的二次型  $x^T A x$  都为正。
manual = np.array([(5 - np.sqrt(5)) / 2, (5 + np.sqrt(5)) / 2])
# 只在显示时保留 6 位小数; 后续检验继续使用未舍入的原始数值。
print("特征值 =", np.round(values, 6))
# 迹为主对角线元素之和 5, 行列式为  $3\times 2-1\times 1=5$ , 可独立核对特征值结果。
print("迹、行列式 =", np.trace(A), round(np.linalg.det(A), 6))
# 浮点计算可能有舍入误差, 使用 allclose 的容差比较数值, 不要求逐位相同。
assert np.allclose(values, manual)
# 同时检验两条  $Av_j=\lambda_j v_j$ : 右侧 (2,2)*(2,) 把第 j 列乘以  $\lambda_j$ 。
# 这条广播规则十分关键; 如果误把特征向量当作行, 关系一般不成立。
assert np.allclose(A @ vectors, vectors * values)
#  $Q^T Q=I$  检验列向量标准正交; 因此在这个实对称案例中  $Q^{-1}=Q^T$ 。
assert np.allclose(vectors.T @ vectors, np.eye(2))
```

```
# 核对“特征值之和等于迹”；迹在换基后保持不变。
assert np.isclose(values.sum(), np.trace(A))
# 核对“特征值之积等于行列式”；在二维中该乘积也是面积缩放倍数。
assert np.isclose(values.prod(), np.linalg.det(A))
# 用从 0 到  $2\pi$  的 200 个角度描出单位圆；含两个端点，曲线自然闭合。
theta = np.linspace(0, 2 * np.pi, 200)
# vstack 把横、纵坐标叠成 (2,200), A@(2,200) 同时变换 200 个列向量。
# 由于 A 对称且正定，圆变为沿特征向量方向、半轴长度为特征值的椭圆。
circle = np.vstack([np.cos(theta), np.sin(theta)])
ellipse = A @ circle
# 创建一张图及一个坐标轴；figsize 的单位是英寸。
fig, ax = plt.subplots(figsize=(4.5, 4))
# 星号把两行坐标展开为 plot(x,y)；黑色虚线给出变换前的单位圆。
ax.plot(*circle, "k--", label="单位圆")
# 将全部变换后的点连成曲线，与单位圆比较伸缩方向和幅度。
ax.plot(*ellipse, label="A 映射后的圆")
# vectors.T 的每一行恰好是原矩阵的一列，因此可逐对取出 ( $\lambda_j, v_j$ )。
# 这里转置是为了 Python 的逐行迭代习惯，并未改变特征向量按列存储的约定。
for value, vector in zip(values, vectors.T):
    # 把  $-\lambda v$  和  $+\lambda v$  作为两列，组成 (2,2) 的线段端点数组，画出完整的椭圆轴。
    endpoints = value * np.column_stack([-vector, vector])
    # 图例显示该方向的伸缩倍数；3 位小数只影响图例文本。
    ax.plot(*endpoints, label=f"主轴 {value:.3f}")
# 横纵坐标使用相同长度单位，避免把圆因画布比例画成椭圆；淡网格辅助读图。
ax.set_aspect("equal")
ax.grid(alpha=0.3)
ax.legend()
# 标题说明本图展示的是对称正定线性变换。
ax.set_title("对称正定矩阵的变换")
# 自动调整边距，以 180 dpi 保存，再关闭图对象以释放内存。
fig.tight_layout()
fig.savefig("e25_eigen.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

特征值 = [1.381966 3.618034]

迹、行列式 = 5.0 5.0

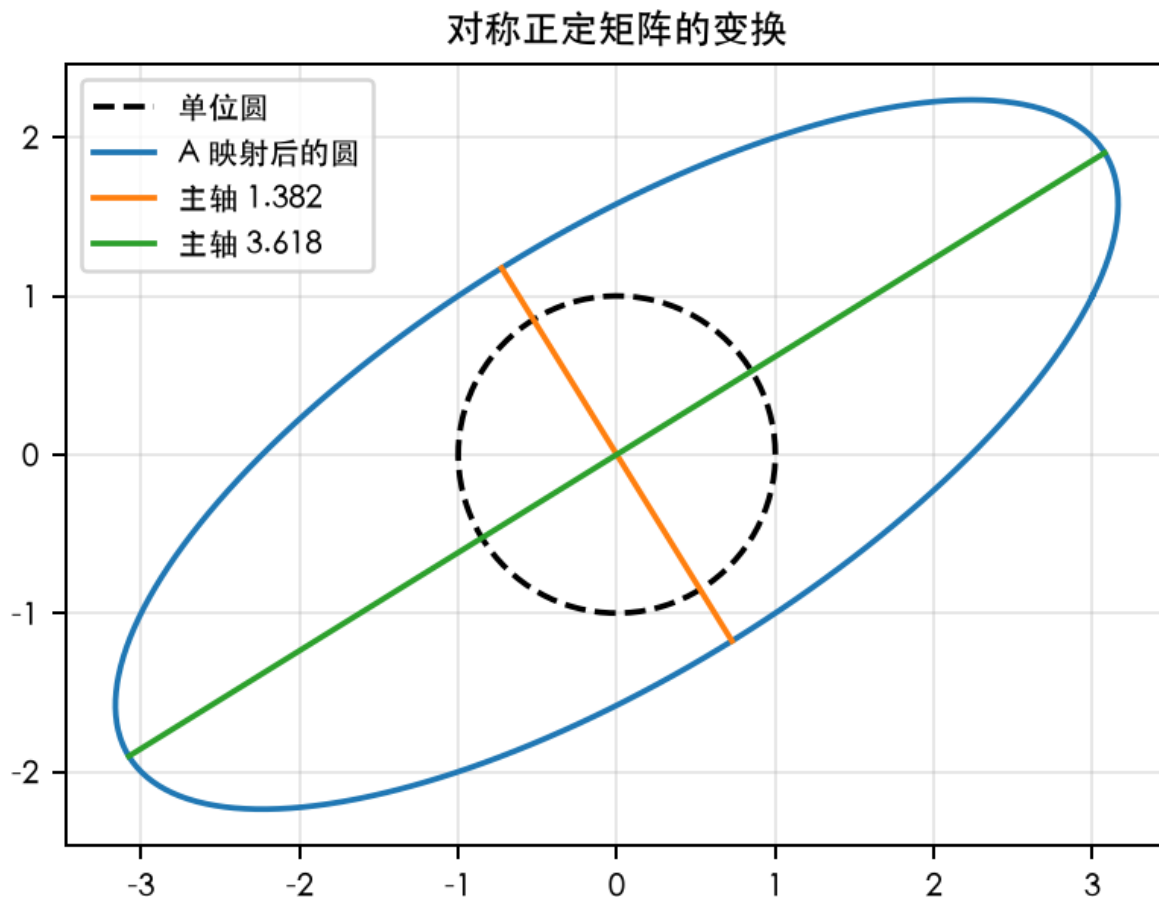


图 8-1 `e25_eigen.py` 的 $A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ 把单位圆映成椭圆。两条特征方向为椭圆轴线，半轴长为 $(5 \pm \sqrt{5})/2$ 。虚线圆和输出椭圆使用同一坐标尺度。

读图指南 先用虚线单位圆认出输入长度尺度，再沿两条彩色轴线读输出椭圆的半轴。这里 A 对称正定，所以半轴长恰等于两个正特征值；一般矩阵应改用奇异值解释半轴。

8.3 对角化 相似与矩阵幂

若 $n \times n$ 矩阵 A 拥有 n 个线性无关特征向量，把它们按列组成 $Q = [v_1 \cdots v_n]$ ，把相应特征值排成 $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ ，则 $AQ = Q\Lambda$ 。由于列无关， Q 可逆，于是 $A = Q\Lambda Q^{-1}$ ，称为对角化。这里 Q 只是特征向量矩阵，未必正交；不能无条件把 Q^{-1} 写成 Q^T 。 Q 的列顺序改变时， Λ 的对角顺序必须同步改变。

对角化可以读作换坐标： Q^{-1} 先把标准坐标改写成特征基坐标， Λ 让各坐标分别乘对应特征值， Q 再换回标准坐标。在同一线性映射的不同基下，矩阵可能不同，却满足 $B = S^{-1}AS$ ，称

为相似。S 为可逆换基矩阵。相似矩阵具有相同特征多项式、迹和行列式，但元素、长度尺度和数值条件数未必相同。换基是在改写描述，不是在额外改变原映射。

n 个互不相同的特征值必能提供 n 个无关特征向量，因此保证可对角化。反向不成立： I_n 只有一个不同特征值 1，却拥有整个 $\mathbb{R}^n$ 作为特征空间，当然可以对角化。更一般地，A 可对角化当且仅当各特征空间维数之和等于 n，即每个特征值的几何重数都达到其代数重数。重根本身不是障碍，缺少独立特征向量才是障碍。

对整数 $k \geq 0$ ，重复相乘中的 $Q^{-1}Q$ 相消，给出 $A^k = Q\Lambda^k Q^{-1}$ 。第 i 个特征方向上的分量经过 k 次演化乘 λ_i^k 。若所有 $|\lambda_i| < 1$ ，则 $A^k \rightarrow 0$ ；若某个特征值为 1，可能保留对应稳态方向；若存在 $|\lambda_i| > 1$ ，一般会有增长方向。对角化公式要求矩阵确实可对角化；不可对角化时，约当块还会产生 k、 k^2 等多项式因子。

$$A^k = Q\Lambda^k Q^{-1}, \quad \Lambda^k = \text{diag}(\lambda_1^k, \dots, \lambda_n^k) \quad (8.2)$$

Q 由 n 个无关特征向量组成， Λ 按相同顺序放置特征值；k 为非负整数。k=0 时 $A^0 = I$ ；若要用负整数幂，还需 A 可逆，即所有 λ_i 非零。

例题 反例：重复特征值不保证可对角化

取 $J = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ 。特征多项式为 $(2-\lambda)^2$ ，唯一特征值 2 的代数重数为 2。 $J - 2I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ，齐次方程要求 $v_2 = 0$ ，所以特征空间只有 $\text{span}\{(1,0)^T\}$ ，几何重数 1，无法凑齐两个无关特征向量。

写 $J = 2I + N$ ，其中 $N = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ 且 $N^2 = 0$ 。二项展开得 $J^k = 2^k I + k 2^{k-1} N$ ($k \geq 1$)。非对角元素中的 k 说明仅看“有两个相同特征值”并不能把 J 当成 2I；特征值相同不意味着矩阵演化相同。

程序 `e26_diagonalize.py` 取 $A = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$ 的特征值为 0.7 和 1, $k=10$ 。程序按特征分解重建 A^{10} , 并与 `matrix_power` 逐项比较, 结果约 $\begin{bmatrix} 0.676083 & 0.323917 \\ 0.647835 & 0.352165 \end{bmatrix}$, 两行之和均为 1。末尾独立反例 $J = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ 报告代数重数 2、几何重数 1, 说明对角化必须检查完整条件。这里用数值矩阵直接核验两种幂运算。

编程思路 在可对角化的 2×2 矩阵上, 用 $A = Q\Lambda Q^{-1}$ 计算十次幂, 再与直接矩阵幂比较。另构造只有一个线性无关特征向量的若尔当块, 明确特征值存在不保证可对角化。

代码对应

求特征向量矩阵 Q 的逆作用 $\rightarrow Q_inverse = \text{np.linalg.solve}(Q, \text{np.eye}(2))$: 以 `solve(Q,I)` 求解, 而不是把 $Q.T$ 当成逆; 此处 Q 不保证正交。

$A^{10} = Q \text{diag}(\lambda_1^{10}, \lambda_2^{10}) Q^{-1} \rightarrow \text{power_from_eigen} = Q @ \text{np.diag}(\text{values} ** k) @ Q_inverse$: 只在特征向量构成可逆基时才能使用该表达式。

直接相乘意义下的矩阵十次幂 $\rightarrow \text{power_direct} = \text{np.linalg.matrix_power}(A, k)$: `matrix_power` 与 $A**k$ 的逐元素幂不同, 作为独立计算路径核对。

$\dim \ker(J-2I)=2-\text{rank}(J-2I)=1 \rightarrow \text{geometric_multiplicity} = 2 - \text{np.linalg.matrix_rank}(J - 2 * \text{np.eye}(2))$: 几何重数小于特征值 2 的代数重数 2, 所以 J 不能对角化。

易错提示 重复特征值本身不阻止对角化, 关键是是否有足够多的线性无关特征向量。使用特征分解算高次幂前, 还要留意特征向量矩阵的条件数。

正文代码 `e26_diagonalize.py`

```
# 教材位置: 第 8 章, 第 8.3 节; 仅终端输出。
# 单位、变量与形状: 随机矩阵 A 为 (2,2), 矩阵幂为十步转移概率; 全部元素和特征值无量纲。
# 案例 26: 用对角化计算矩阵幂, 并认识不能对角化的若尔当块。
# 学习目标: 理解  $A = Q\Lambda Q^{-1}$  时  $A^k = Q\Lambda^k Q^{-1}$ , 及“有足够多线性无关特征向量”
# 才是可对角化的关键; 重复特征值本身并不自动意味着不能对角化。
# 核心关系: 几何重数  $\dim \ker(A - \lambda I) = n - \text{rank}(A - \lambda I)$ , 且不超过代数重数。
# 数据: A 为非对称的  $2 \times 2$  行随机矩阵; J 为特征值 2 的二阶若尔当块。
# 输出: 打印 A 的特征值、A 的 10 次幂、行和以及 J 的几何重数; 不生成图片。
# 运行方式: 安装依赖后在本目录执行 python e26_diagonalize.py。
# 所有 assert 都是数值或结构检查, 正常通过时不会额外打印信息。
```

```
# NumPy 提供特征分解、线性方程求解、矩阵幂和秩计算。
import numpy as np
# 各行非负且和为 1，因此也可把 A 理解为两状态转移矩阵。
# A 不对称，不能沿用对称矩阵  $Q^{-1}=Q^T$  的结论。
A = np.array([[0.9, 0.1], [0.2, 0.8]])
# 一般矩阵用 eig; Q 的第 j 列是与 values[j] 对应的右特征向量。
# 返回次序不保证升序。本例特征值 1 和 0.7 不同，所以两列线性无关、Q 可逆。
values, Q = np.linalg.eig(A)
# 求解  $QX=I$  得到  $Q^{-1}$ ；使用线性方程求解接口表达这个计算。
# 这里显式构造逆矩阵是为了展示对角化公式，一般求解  $Ax=b$  时通常直接 solve(A,b)。
Q_inverse = np.linalg.solve(Q, np.eye(2))
# 将矩阵幂指数设为 10，用重复转移的例子观察对角化如何简化计算。
k = 10
# 先把两个特征值分别取 k 次幂，再构成 2x2 对角矩阵。
# 由于相邻的  $Q^{-1}Q=I$  会抵消，A 连乘 k 次就化为  $Q\Lambda^kQ^{-1}$ 。
power_from_eigen = Q @ np.diag(values ** k) @ Q_inverse
# 用独立的矩阵幂算法作参照，验证上面的特征分解计算。
power_direct = np.linalg.matrix_power(A, k)
# 只为显示而排序特征值；不能单独重排 values 后仍与原 Q 配对使用。
print("特征值 =", np.round(np.sort(values), 6))
# 打印十次转移矩阵；对行概率向量 x，十步分布为  $x@A^{10}$ 。
print("A^10 =\n", np.round(power_direct, 6))
# axis=1 沿每行的列求和，得到形状 (2,) 的行和数组。
print("各行元素之和 =", A.sum(axis=1))
# 检验每行和为 1；这里只检查行和，矩阵非负性由给定数据直接可见。
assert np.allclose(A.sum(axis=1), 1)
# 重建 A，以容差检验本例的对角化确实成立。
assert np.allclose(A, Q @ np.diag(values) @ Q_inverse)
# 两个独立算法应给出相同的  $A^{10}$ ，允许浮点舍入误差。
assert np.allclose(power_direct, power_from_eigen)
# J 的特征多项式为  $(2-\lambda)^2$ ，所以  $\lambda=2$  的代数重数是 2。
# 右上角的 1 使 J 不是简单的  $2I$ ，接下来检查其特征空间维数。
J = np.array([[2.0, 1.0], [0.0, 2.0]])
# 特征向量满足  $(J-2I)v=0$ ；由秩-零化度定理，核空间维数为  $2-\text{rank}$ 。
#  $J-2I$  的秩为 1，故只有一个线性无关特征向量，无法组成可逆的  $2\times 2$  的 Q。
geometric_multiplicity = 2 - np.linalg.matrix_rank(J - 2 * np.eye(2))
# 把代数重数与几何重数并列显示，突出“根出现几次”和“独立方向有几个”的区别。
print("矩阵 J: 代数重数=2, 几何重数=",
      geometric_multiplicity)
# 几何重数 1 小于代数重数 2，说明这个 J 不能对角化。
assert geometric_multiplicity == 1
# 写  $J=2I+N$ ，且  $N^2=0$ ，因此  $J^3=8I+12N$ ；右上角为 12。
# 这说明不能对角化的矩阵仍然可以求幂，只是不能直接套用  $Q\Lambda^kQ^{-1}$ 。
assert np.allclose(np.linalg.matrix_power(J, 3), [[8, 12], [0, 8]])
```

规定环境下的实际运行输出

特征值 = [0.7 1.]

A^10 =
[[0.676083 0.323917]
[0.647835 0.352165]]

各行元素之和 = [1. 1.]

矩阵 J: 代数重数=2, 几何重数= 1

8.4 实对称 正定与二次型

实对称矩阵满足 $A^T=A$ 。谱定理说明它总有实特征值，且可选一整组标准正交特征向量，因此 $A=Q\Lambda Q^T$ 。不同特征值的特征向量为什么正交？若 $Av=\lambda v$ 、 $Aw=\mu w$ ，则 $v^T Aw=\mu v^T w$ ；又因 A 对称， $v^T Aw=(Av)^T w=\lambda v^T w$ 。相减得 $(\lambda-\mu)v^T w=0$ ， $\lambda\neq\mu$ 时只能 $v^T w=0$ 。重特征空间内也可以用格拉姆—施密特选择正交基。

谱定理可这样理解其完整性。实对称 A 的二次型在单位球面上有最大值；取最大点 v ，对球面内任意垂直于 v 的微小变化求导可得 $w^T Av=0$ ，因此 Av 只能沿 v ， v 是实特征向量。对任何 $x\perp v$ ，有 $v^T Ax=(Av)^T x=\lambda v^T x=0$ ，所以 v 的正交补在 A 作用下保持不变。把 A 限制到这个少一维的空间仍然对称，递归选择单位特征向量，最终得到整个 $\mathbb{R}^n$ 的标准正交特征基。这个论证也说明重特征值不会妨碍实对称矩阵的正交对角化。[MIT-L13]

二次型 $x^T Ax$ 把一个向量转成标量。对实对称 A ，若每个非零 x 都满足 $x^T Ax>0$ ，称正定；若总是 ≥ 0 ，称半正定。设 $z=Q^T x$ ，则 $x^T Ax=\sum_i \lambda_i z_i^2$ 。所以正定等价于全部特征值为正，半正定等价于全部非负。这里的判据要求对称性；对于非对称 A ，二次型只看见其对称部分 $(A+A^T)/2$ 。

对称正定矩阵的二次型等值面是椭球，矩阵的逆也正定。最小二乘中 $A^T A$ 总是半正定，因为 $x^T A^T Ax=\|Ax\|_2^2$ ；当 A 列无关时才正定。若存在非零零空间，则某个非零 x 使这一平方为零，所以不能说任意 $A^T A$ 都正定。这把第 7 章正规方程的唯一性条件与谱结构连接起来。

正定还控制二次优化。对 $f(x)=1/2 x^T Hx-g^T x$ ，其中 H 为实对称正定矩阵， g 为给定向量，设 $x^*=H^{-1}g$ 。把 x 写成 x^*+z 并展开，交叉项因 $Hx^*=g$ 消去，得到 $f(x)=f(x^*)+1/2 z^T Hz$ 。后项仅在 $z=0$ 时取零，所以 x^* 为唯一全局最小点。这一证明和最小二乘的正交平方分解同源，都通过把“离开最优点的代价”写成非负项建立结论。

例题 手算：正定与非正定的边界

对 $H=[[2,1],[1,2]]$ ，二次型为 $2x_1^2+2x_1x_2+2x_2^2=(x_1+x_2)^2+x_1^2+x_2^2$ 。非零 x 时严格为正，因此正定。特征值是 3 和 1，也给出同一判断。

对 $K = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ ，二次型为 $(x_1 + x_2)^2$ ，虽然非负，却在 $x = (1, -1)^T$ 时为零，所以只是半正定。
 K 的特征值为 2 与 0，行列式为 0，零空间记录了二次型完全不惩罚的方向。

例题 反例：正对角元素和正行列式为什么还不够

对称矩阵 $H = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ 的对角元素全正，但在 $x = (1, -1)^T$ 上， $x^T H x = -2 < 0$ ，所以不是半正定。对称矩阵 $-I_2$ 的行列式为 1，却在每个非零方向上给负二次型。因此单看对角或行列式都不足以判断正定。

对二阶实对称矩阵 $H = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ ，若 $a > 0$ ，可配方为 $a(x_1 + (b/a)x_2)^2 + [(ac - b^2)/a]x_2^2$ ，所以 H 正定恰等价于 $a > 0$ 且 $ac - b^2 > 0$ 。一般 n 阶可用全部特征值或适用的顺序主子式判据；不能把二阶的两个数检查机械推广到所有矩阵。

协方差矩阵的半正定性由 $\text{Var}(a^T Y) = a^T \Sigma a \geq 0$ 保证。正定意味着每个非零线性组合都有正方差；若某些变量存在精确线性关系，就有非零 a 使该组合恒定，中心化后为零，因此 Σ 奇异。例如 $Y_2 = 2Y_1$ 时，组合 $2Y_1 - Y_2$ 的方差为 0。样本协方差还有秩至多 $N-1$ 的限制：中心化的 N 行之和为零，所以变量数 $d \geq N$ 时不可能样本正定。

8.5 马尔可夫矩阵与行分布约定

有限状态马尔可夫链用一步转移概率描述离散时间演化。设状态为 $1, \dots, s$ ， $P_{ij} = P(\text{下一步状态 } j | \text{当前状态 } i)$ 。矩阵 P 元素非负，每行之和等于 1，称行随机矩阵。这里“随机矩阵”是概率转移矩阵的固定名称，并不要求每次重新生成矩阵。马尔可夫假设指已知当前状态后，下一步分布不再依赖更早历史；时齐性还要求同一 P 适用于各时间步。

转移公式正是全概率公式。令 Z_t 为当前状态， $q_{ti} = P(Z_t = i)$ ，则 $P(Z_{t+1} = j) = \sum_i P(Z_{t+1} = j | Z_t = i) P(Z_t = i)$ 。马尔可夫性让这一步条件分布只依赖当前状态；时齐性才让相同的 P 在每步复用。并不是说相邻状态独立，通常它们正因为转移而相关。若存在更长的记忆、季节效应或人群差异，单一固定转移矩阵可能只是近似模型。

令 $q_t=(q_{t,1},\dots,q_{t,s})$ 为第 t 步行分布，分量非负且和为 1。转移后的第 j 项为 $\sum_i q_{t,i}P_{ij}$ ，写成 $q_{t+1}=q_tP$ 。因此 $q_t=q_0P^t$ ，矩阵幂把多条路径概率相加。若改用列分布 $p_t=q_t^T$ ，公式应同步改成 $p_{t+1}=P^Tp_t$ 。行列约定可以任选，但不能把同一个行随机 P 与列分布直接写成 Pp_t 。

行和为 1 意味着 $P1=1$ ， 1 是全 1 列向量，因此 1 总是 P 的右特征值所对应的特征向量。概率稳态 π 满足 $\pi P=\pi$ ，位于左侧，是左特征向量；等价地 $P^T\pi^T=\pi^T$ 。求 π 时还要加归一化 $\sum_j\pi_j=1$ 与非负条件，单求特征方程中的任意倍数还不是概率分布。第 6 章的一般列向量约定在这里特意改为行分布，程序也保持这一约定。

$$q_{t+1} = q_t P, \quad q_t = q_0 P^t, \quad \pi P = \pi, \quad \sum_{j=1}^s \pi_j = 1 \quad (8.3)$$

P 为 $s \times s$ 行随机矩阵； t 是非负整数时间， q_t 与 π 均是长度 s 的行分布。上标 t 在 P^t 中表示整数乘方，与转置 T 不同。

例题 手算：两状态稳态与收敛速度

对 $P=[[0.9,0.1],[0.2,0.8]]$ ，令 $\pi=(u,1-u)$ 。稳态第一分量满足 $u=0.9u+0.2(1-u)$ ，即 $0.3u=0.2$ ，得 $u=2/3$ 。因此稳态为 $(2/3,1/3)$ 。它描述长期概率比例，不意味着单个系统在稳态时停止转移。

若第一状态的概率为 u_t ，则 $u_{t+1}=0.7u_t+0.2$ 。减去 $2/3$ 得到 $u_{t+1}-2/3=0.7(u_t-2/3)$ ，所以偏差每步乘 0.7 。这正是第二特征值 0.7 在长期演化中的作用。第 10 步两种纯状态初值给出 e26 结果的两行。

8.6 稳态存在 唯一与趋近要分开判断

有限状态链至少存在一个稳态，但它可能不唯一。不可约表示任意两个状态之间都能经某个有限步数以正概率到达；非周期表示返回状态的可能步数没有大于 1 的共同周期。对有限不可约链，稳态唯一且各项为正；再有非周期性，任意初始分布才收敛到该稳态。若 P 所有元素严格为正，这两个条件同时满足。链的图结构比单看一次迭代是否平滑更能保证结论。

周期反例是 $C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 。它有唯一稳态 $(1/2, 1/2)$ ，但从 $(1, 0)$ 出发会在 $(1, 0)$ 与 $(0, 1)$ 之间交替，永不收敛。 C 的特征值为 1 与 -1 ，第二个方向每一步改变符号，幅度不衰减。若初始恰为稳态则一直不动；这不推翻反例，因为“任意初始都收敛”要求覆盖所有初始分布。

另一个反例是 I_2 ，每个分布都是稳态，链不能从一个状态转到另一个状态，所以不具不可约性，也没有唯一长期比例。还有可约但仍唯一稳态、并从所有初值收敛的链，例如存在单一吸收类且其余状态最终离开；所以不可约加非周期是常用充分条件，不能误称为唯一可能的情形。模型检验应依据具体转移结构。

例题 手算：可约链也可能有唯一长期分布

$P = \begin{bmatrix} 1 & 0 \\ 0.2 & 0.8 \end{bmatrix}$ 中，状态 1 是吸收态；状态 2 每步以 0.2 概率转到状态 1。它可约，但从 $q_0 = (u_0, 1 - u_0)$ 出发，第二状态概率为 $(1 - u_0)0.8^t$ ，最终收敛到 $(1, 0)$ 。唯一稳态含一个零分量。

因此“唯一稳态”不必意味着每类都有正份额；“不可约、非周期”是保证任意初值收敛到唯一正稳态的常用充分条件。这里的零份额反映吸收结构，不能解释成每个人在有限步内都已确定进入状态 1。

对可对角化的收敛链，除了特征值 1，其余特征值的模均小于 1，非稳态部分由这些幂衰减。最大非主特征值模越接近 1，往往趋近越慢；常把 1 减这个模称为谱间隙的一个指标。但精确误差界还受特征向量、矩阵是否可对角化及所用距离影响，不能仅凭“第二特征值是 0.9”就承诺固定步数内达到某项精度。

程序 `e27_markov.py` 取转移矩阵 $P = \begin{bmatrix} 0.90 & 0.08 & 0.02 \\ 0.05 & 0.92 & 0.03 \\ 0.01 & 0.04 & 0.95 \end{bmatrix}$ ，初始行分布 $[0.40, 0.35, 0.25]$ 。程序按 $X[t+1] = X[t]@P$ 生成 301×3 数组，图展示前 81 行，第 300 步只作高精度核验。稳态约 $[0.241379, 0.413793, 0.344828]$ ，第二特征值模约 0.925311。另用 $\text{cycle} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 与初始 $[1, 0]$ 生成 11×2 周期序列，对照“存在稳态却不收敛”。

编程思路 用行向量表示三类状态比例，逐步右乘转移矩阵。稳态由 $P.T$ 的特征值 1 的特征向量归一化得到，并与长期迭代核对；再以二状态周期链说明稳态存在与逐步收敛的区别。

代码对应

为第 0 至 300 步的三状态分布建立表 $\rightarrow x = \text{np.empty}((301, 3))$: 每行存一步、每列存一状态, 数组形状 (301,3) 与递推方向相符。

把 $\pi P = \pi$ 转为 $P.T \pi^T = \pi^T \rightarrow \text{values, vectors} = \text{np.linalg.eig}(P.T)$: $\text{eig}(P.T)$ 的右特征向量才能对应行分布的稳态。

将特征值 1 对应向量归一化为总和 1 $\rightarrow \text{pi} = \text{vector.real} / \text{vector.real.sum}()$: 得到理论稳态 [7/29,12/29,10/29], 再检查非负性及 $\text{pi}@P=\text{pi}$ 。

以两状态必然交替构造周期为二的链 $\rightarrow \text{cycle} = \text{np.array}([[0, 1], [1, 0]])$: 该链仍有 [1/2,1/2] 稳态, 但从单一状态出发的分布会振荡。

易错提示 本例按行给出条件转移概率, 各行和为一; 若改成列约定, 乘法与求特征向量方向也必须同时改。第二特征值模小于一与非周期结构支撑本例的收敛, 不能推广到所有链。

正文代码 e27_markov.py

```
# 教材位置: 第 8 章, 第 8.6 节; 图 8-2。
# 单位、变量与形状: P 为 (3,3), x 每行是一步的 (3,) 状态概率; 概率无量纲, 步数按转移次数
# 计。
# 案例 27: 行随机矩阵、概率分布迭代、稳态及周期性反例。
# 学习目标: 分清行概率向量 x 的更新  $x_{t+1}=x_t P$  与列向量特征方程的关系,
# 理解稳态  $\pi P=\pi$ , 并观察长期分布如何靠近稳态。
# 数据与形状: P 为 3x3, P[i,j] 为从状态 i 到状态 j 的一步概率;
# X 为 301x3, 第 t 行保存第 t 步的三种状态概率, 不是随机抽取的一条状态路径。
# 本例 P 的元素全为正, 有限状态链不可约且非周期, 因而有唯一稳态并收敛。
# 输出: 打印稳态和第二大特征值模; 验证周期反例; 生成 e27_markov.png。
# 运行方式: 安装依赖后在本目录执行 python e27_markov.py。
# 图片保存到当前工作目录。计算完全确定, 本例没有随机抽样。

# NumPy 用于概率向量更新、特征分解及收敛性的数值检查。
import numpy as np
# Matplotlib 画各状态概率的时间演化及其稳态参考线。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 每一行给出一个当前状态的下一步去向; 对角线接近 1 表示一步内较可能保持原状态。
P = np.array([[0.90, 0.08, 0.02], [0.05, 0.92, 0.03], [0.01, 0.04, 0.95]])
# 转移概率必须非负且每行和为 1, 这样  $x@P$  才能保持非负并使总概率仍为 1。
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
# 包含初始第 0 步与之后 300 步, 所以预分配 301 行。
# empty 的未赋值位置没有有效数据; 循环按顺序写入各行后再使用。
X = np.empty((301, 3))
X[0] = [0.40, 0.35, 0.25]
# t 依次为 0,...,299, 从已知的第 t 行计算第 t+1 行。
for t in range(300):
```

```

# 行向量形状 (3,) 与 (3,3) 相乘; 第 j 项为  $\sum_i x_t[i] P[i, j]$ 。
X[t + 1] = X[t] @ P
# 稳态是 P 的左特征向量:  $\pi P = \pi$ 。转置后写成  $P^T \pi^T = \pi^T$ ,
# 所以对 P.T 求右特征向量, 再把所得列向量的数值当作稳态各分量。
values, vectors = np.linalg.eig(P.T)
# 浮点计算中的特征值未必逐位等于 1; 选择距离 1 最近的特征值对应的列。
vector = vectors[:, np.argmax(np.abs(values - 1))]
# 一般 eig 可能返回复数类型; 先确认虚部只在舍入误差范围, 再取实部。
assert np.max(np.abs(vector.imag)) < 1e-12
# 特征向量可以乘任意非零常数, 除以分量总和和把它归一化成概率向量。
# 即使求出的向量整体为负, 这种归一化也会恢复本例的非负稳态。
pi = vector.real / vector.real.sum()
# 显示稳态概率; 稳态表示总体分布不再改变, 不表示每个个体停止转移。
print("稳态概率向量 =", np.round(pi, 6))
# 去掉模为 1 的主特征值后, 第二大模刻画本例误差长期衰减的主要速率。
# 模越接近 1, 通常需要越多步才能接近稳态; 这里取模也覆盖负或复特征值。
print("次大特征值模 =", round(np.sort(np.abs(values))[-2], 6))
# 同时验证概率非负和稳态方程; pi.sum()=1 已由归一化构造保证。
assert np.all(pi >= 0) and np.allclose(pi @ P, pi)
# 与手工解线性方程  $\pi P = \pi$ 、 $\sum \pi_i = 1$  得到的分数形式核对。
assert np.allclose(pi, [7 / 29, 12 / 29, 10 / 29])
# 第 300 步分布应已接近稳态; rtol=0 使这里采用纯绝对误差  $10^{-10}$ 。
assert np.allclose(X[-1], pi, atol=1e-10, rtol=0)
# 两状态链每步必定换到另一状态, 周期为 2, 特征值为 1 和 -1。
# 它有稳态  $[1/2, 1/2]$ , 但从  $[1, 0]$  出发的分布会来回振荡, 说明有稳态不保证收敛。
cycle = np.array([[0, 1], [1, 0]])
# 三步为奇数, 从状态 1 必定到状态 2; 这里整数结果可用 array_equal 精确比较。
assert np.array_equal(np.array([1, 0]) @ np.linalg.matrix_power(cycle, 3),
                      [0, 1])
# 用两个坐标轴展示三状态收敛过程与周期二状态反例, 画布大小以英寸指定。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.6))
ax = axes[0]
# enumerate 同时提供状态列索引 j 与图例文字。
for j, name in enumerate(["状态 1", "状态 2", "状态 3"]):
    # 只绘制第 0 至 80 步以突出过渡过程; line, 从返回列表中取出唯一的曲线对象。
    line, = ax.plot(np.arange(81), X[:81, j], label=name)
    # 用与该状态曲线相同的颜色画稳态水平虚线, 便于观察趋近关系。
    ax.axhline(pi[j], color=line.get_color(), linestyle=":")
# 横轴为离散时间步, 纵轴为总体状态概率; 三条曲线在任一步的值相加为 1。
ax.set(xlabel="步数", ylabel="概率", title="正转移链: 概率收敛")
# 图例置于坐标轴外侧下方, 避免遮挡任一条状态轨迹; 网格辅助读数。
ax.legend(fontsize=7.8, loc="upper center", bbox_to_anchor=(0.5, -0.25), ncol=3)
ax.grid(alpha=0.3)
# 周期曲线从已定义的 cycle 实际迭代; 不把交替结果硬编码成奇偶公式。
periodic = np.empty((11, 2))
periodic[0] = [1, 0]
for step in range(10):
    periodic[step + 1] = periodic[step] @ cycle
periodic_stationary = np.array([0.5, 0.5])
assert np.allclose(periodic.sum(axis=1), 1)
assert np.allclose(periodic_stationary @ cycle, periodic_stationary)
assert np.allclose(periodic[:, 2], [1, 0]) and np.allclose(periodic[1::2], [0, 1])
ax = axes[1]
ax.plot(np.arange(11), periodic[:, 0], "-.", label="状态 1, 初始 (1,0)")
ax.axhline(periodic_stationary[0], color="tab:orange", linestyle=":",
           label="稳态概率 = 1/2")
ax.set(xlabel="步数", ylabel="状态 1 的概率", ylim=(-0.08, 1.08),
       title="周期链: 概率不收敛")

```

```
ax.legend(fontsize=7.8, loc="upper center", bbox_to_anchor=(0.5, -0.25))
ax.grid(alpha=0.3)
# 300 步只用于上面的高精度数值核验：左图仍使用同一个 x 的前 81 行。
# 调整边距并保存 180 dpi 图片，关闭图对象以避免批量运行时积累内存。
fig.tight_layout()
fig.savefig("e27_markov.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

稳态概率向量 = [0.241379 0.413793 0.344828]

次大特征值模 = 0.925311

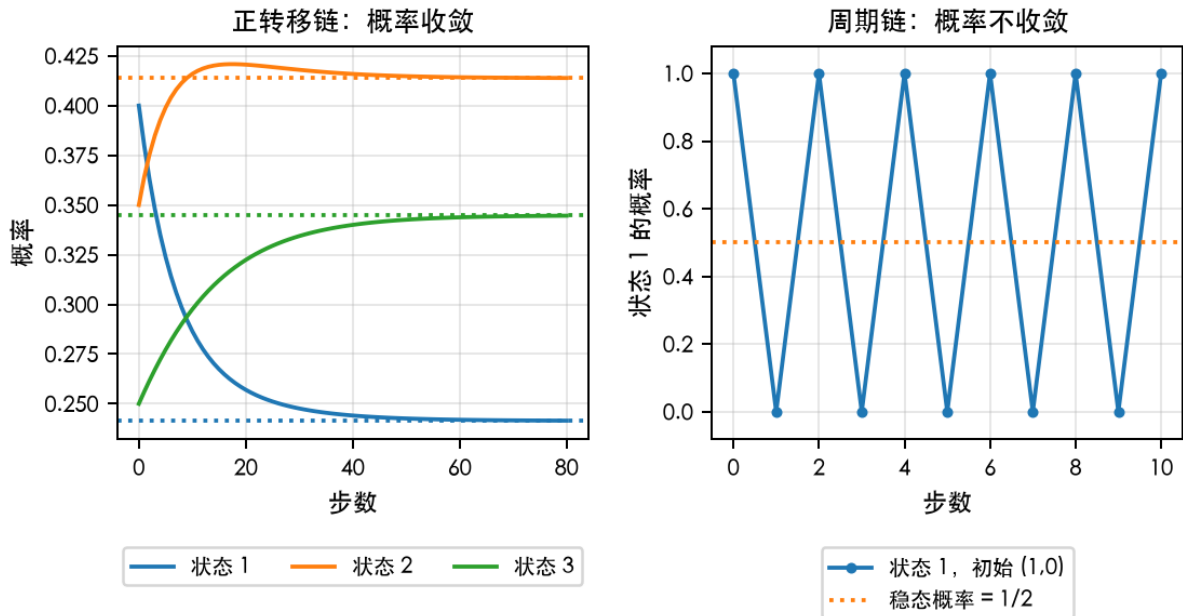


图 8-2 左侧为 e27 三状态分布的前 80 步，水平虚线标出同一 P 算得的稳态 π ；右侧是两状态周期反例，第一状态概率在 0 与 1 间交替。两图共同区分稳态方程与长期收敛。

读图指南 左图沿时间看三条分布曲线接近各自水平稳态线，纵轴是概率比例；右图第一状态概率持续在 0 和 1 间交替。并排对照可以区分稳态存在与从给定初值收敛。

8.7 从概率守恒检查模型

行随机性本身有可核验的代数后果。若 q_i 分量非负， P 元素非负，则 $q_i P$ 也分量非负；又因 $P1=1$ ，有 $(q_i P)1=q_i 1=1$ ，所以总概率守恒。这一证明不依赖具体三状态案例。程序应在开始时检查元素与行和，并在迭代中核对分布，避免把方向约定或索引错误当成新的系统规律。若只满足行和为 1 却含负元素，代数守恒仍成立，但不再是概率转移矩阵。

稳态比例也不同于单个个体的确定命运。 π_j 描述在稳态抽取一次观察时处于状态 j 的概率；在合适的遍历条件下，它还与长时间访问频率相联系。它不说明某个对象会永远留在某状态，也不表示每条轨迹在每一步的实际比例都恰好等于 π 。把分布演化、随机轨迹和样本频率分开，才能把概率章节与矩阵演化准确衔接。

8.8 练习与参考解答

练习 8-1（概念与手算，入门） $A=\text{diag}(2,-1)$, $x=(3,4)^T$ 。求 A^3x 并解释两个特征方向。

练习 8-2（概念与手算，入门） $J=\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ 能否对角化？与 I_2 相比， J^k 有何不同？

练习 8-3（概念与手算，进阶） 判断 $H=\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ 和 $K=\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ 是否正定、半正定。

练习 8-4（概念与手算，进阶） 行随机 P 满足 $P1=1$ ，能否据此断定均匀行分布是稳态？给出反例。

练习 8-5（编程实践，入门） 运行 `e25_eigen.py`，用 `eigh` 核对特征值，并用矩阵等式验收特征向量；不要要求固定符号。

练习 8-6（编程实践，入门） 运行 `e26_diagonalize.py`，比较 A^{10} 的特征分解法与 `matrix_power`，给出数值验收。

练习 8-7（编程实践，进阶） $P=\begin{bmatrix} .8 & .2 \\ .3 & .7 \end{bmatrix}$, $q_0=[1,0]$ 。计算 q_1 、 q_2 、 q_6 并与稳态比较。

练习 8-8（编程实践，进阶） 为周期链 $C=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ 生成从 $[1,0]$ 出发的前六步行分布。另验收 $[\cdot 5, \cdot 5]$ 是否稳态。

练习 8-9（场景应用，入门） 一个教学模拟的“活跃/暂不活跃”模型采用 $P=\begin{bmatrix} .8 & .2 \\ .3 & .7 \end{bmatrix}$ 。从全部活跃出发，至少多少步可使活跃比例距稳态小于.01？这种比例能否断言每个人的状态？

练习 8-10（场景应用，进阶） 一个模拟反馈系统沿两个独立模态每步分别乘.9 和 1.1。是否所有初始状态都趋近零？如何解释某次模拟看似稳定？

练习 8-11（场景应用，进阶） 某模拟转移模型 $P=\begin{bmatrix} 1 & 0 \\ .2 & .8 \end{bmatrix}$ 含吸收状态。求第 t 步第二状态概率和稳态，说明能否声称长期两类都有正份额。

练习 8-12 (场景应用, 拓展) 两个模拟变量满足 $Y_2=2Y_1$, 且 $\text{Var}(Y_1)=3$ 。写出总体协方差矩阵, 判断正定性, 并找一个零方差组合。

参考解答与验收依据

8-1 e_1 、 e_2 为特征方向, 特征值 2、-1。 $A^3x=(24,-4)^T$, 第一分量每步放大 2 倍, 第二分量每步反号但长度保持。

8-2 特征值 1 的代数重数 2, 但 $\ker(J-I)=\text{span}\{(1,0)^T\}$ 只有 1 维, 不能对角化。 $J=I+N$ 且 $N^2=0$, $J^k=I+kN=[[1,k],[0,1]]$; L^k 始终不变。

8-3 H 特征值 3、-1, 在 $(1,-1)^T$ 上二次型为 -2, 非半正定。K 特征值 2、0, 二次型 $(x_1+x_2)^2 \geq 0$, 半正定但不正定、不可逆。

8-4 不能, $P1=1$ 是右特征关系。取 $P=[[.9,.1],[.2,.8]]$, 均匀行分布乘 P 得 $[.55,.45]$, 不是 $[.5,.5]$ 。它的稳态是 $[2/3,1/3]$ 。若还满足每列和为 1, 均匀行分布才一定稳态。

8-5 应得升序特征值约 $[1.381966,3.618034]$, 其和 5、积 5。用 `np.allclose(A@V,V*w)` 与 `np.allclose(V.T@V,np.eye(2))` 应均为 True。任一系列及对应结果同时变号仍满足定义, 不构成错误。

8-6 $A=[[.9,.1],[.2,.8]]$ 时, 应约为 $[[.676083,.323917],[.647835,.352165]]$, 两种算法 `np.allclose` 为 True、每行和约 1。必须用特征向量矩阵的逆; 它一般不正交, 不能无条件用转置代替。

8-7 $q_1=[.8,.2]$, $q_2=[.7,.3]$, 稳态 $[.6,.4]$; $u_t-.6=.4 \times .5^t$, 故 $q_6=[.60625,.39375]$ 。每步按 $q=q@P$, 检查分量非负和总和 1; 若采用列向量需整体转为 $P.T$ 。

8-8 $t=0$ 至 5 依次为 $[1,0]$ 、 $[0,1]$ 、 $[1,0]$ 、 $[0,1]$ 、 $[1,0]$ 、 $[0,1]$ 。均匀分布满足 $\pi@C=\pi$, 但该初值序列不收敛。特征值 -1 导致非稳态分量反号而不衰减。

8-9 稳态活跃比例 .6, 偏差为 $.4 \times .5^t$ 。 $t=5$ 偏差 .0125 仍超标, $t=6$ 为 .00625, 首次小于 .01。这里是模型下分布比例, 不是保证每个个体何时转态; 实际转移概率是否时齐需数据另验。

8-10 不是。含第二模态的初始分量会按 1.1^t 增长；只有第二分量精确为零时才只见 0.9^t 衰减。一次特殊初值稳定不能证明系统对所有初值稳定。模型误差或噪声还可能激发增长方向。

8-11 若初始第二状态概率为 v_0 ，则 $v_t = v_0 \cdot 0.8^t$ ，稳态 $[1, 0]$ 。它可约但唯一稳态且从所有初值收敛；第二类长期份额为零，所以不能声称各类都有正份额，也不能把极限陈述误读为所有个体在某个有限时刻必然已吸收。

8-12 $\Sigma = \begin{bmatrix} 3 & 6 \\ 6 & 12 \end{bmatrix}$ ，特征值 15 与 0，半正定但不正定。 $a = (2, -1)^T$ 给 $a^T Y = 0$ ，方差 0，且 $\Sigma a = 0$ 。零特征方向反映精确线性冗余，不代表两个变量分别都没有波动。

8.9 核心要点与常见误区

核心要点：特征向量要求非零；可对角化要求足够多独立特征方向，重复特征值可以对角化也可以不行。实对称矩阵有完整标准正交特征基，二次型的符号由特征值控制。

核心要点：行随机矩阵把行分布按 $q_{t+1} = q_t P$ 传播，稳态满足左特征方程与概率归一化。稳态存在、唯一、任意初值收敛是三个不同命题；周期和吸收结构必须分别检查。

常见误区：正对角元素、正行列式都不足以一般断言正定； $A^T A$ 总半正定，列无关才正定。 $P1=1$ 不代表均匀行分布稳态，概率分布稳定也不意味着个体停止转移。

8.10 延伸阅读 入门 进阶与拓展

入门阅读：看第 22 讲“对角化与矩阵幂”的 Ses2.9sum，分别核对特征方程、因子重建和整数幂（来源 MIT-L8；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/least-squares-determinants-and-eigenvalues/diagonalization-and-powers-of-a/>）。

进阶阅读：看第 25 讲“实对称矩阵与正定性”；配套第四/五版 6.4—6.5 节，重点读二次型及谱判据（来源 MIT-L13；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/positive-definite-matrices-and-applications/symmetric-matrices-and-positive-definiteness/>）。

拓展阅读：读第 24 讲的马尔可夫矩阵部分及 Ses2.11sum；把课程采用的概率方向与本章行分布整体对应，再检查周期反例（来源 MIT-L9；公开链接：<https://ocw.mit.edu/courses/18->

06sc-linear-algebra-fall-2011/pages/least-squares-determinants-and-eigenvalues/markov-matrices-fourier-series/)

第 9 章 奇异值分解 低秩近似与主成分分析

9.1 三级学习路线

入门路线：先追踪同一条二维单位圆经过 V^T 、 Σ 、 U 的三步变化，分清输入方向和输出方向。进阶路线：用实对称谱定理完整构造一般矩形 SVD，再证明低秩近似的最优误差。拓展路线：把小奇异值与噪声放大、伪逆与正则化、中心化数据与 PCA 连接起来。每一步都先报告矩阵尺寸和目标，再选择保留方向。

9.2 二维几何 先换坐标 再伸缩 再换回

一般矩阵未必存在足够的实特征方向，矩形矩阵也不能直接做同空间的特征分解。但仍可分别在输入侧、输出侧选择标准正交坐标，使变换在中间只按轴伸缩。这就是奇异值分解 (SVD)。它允许输入方向 v_i 和输出方向 u_i 不同，条件 $Av_i = \sigma_i u_i$ 中的 $\sigma_i \geq 0$ 记录长度倍率，不求 $u_i = v_i$ 。

例题 手算：从二维剪切与伸缩看 SVD

取 $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ 。第一输入基 e_1 映到 $(2,0)^T$ ，第二基 e_2 映到 $(1,1)^T$ ，两输出并不正交。因此直接拿原坐标轴描述“独立伸缩”并不合适。计算 $A^T A = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$ ，特征多项式为 $\lambda^2 - 6\lambda + 4$ ，特征值为 $3 \pm \sqrt{5}$ 。

取 $\sigma_1 = \sqrt{3 + \sqrt{5}} \approx 2.28824561$ 、 $\sigma_2 = \sqrt{3 - \sqrt{5}} \approx 0.87403205$ 。可选 $v_1 \approx (0.850651, 0.525731)^T$ 、 $v_2 \approx (-0.525731, 0.850651)^T$ ，再由 $u_i = Av_i / \sigma_i$ 得到 $u_1 \approx (0.973249, 0.229753)^T$ 、 $u_2 \approx (-0.229753, 0.973249)^T$ 。它们在各自一侧都是标准正交基。

把 $V = [v_1 \ v_2]$ 、 $U = [u_1 \ u_2]$ 、 $\Sigma = \text{diag}(\sigma_1, \sigma_2)$ 相乘，得到 $A = U\Sigma V^T$ 。 V^T 先改写输入坐标，单位圆作为点集仍是圆； Σ 把圆变成半轴长 σ_1 、 σ_2 的轴对齐椭圆； U 再改变输出方向。若只盯圆周形状，看不出第一步的坐标变化，必须同时追踪一个着色点或基向量。

$$A = U\Sigma V^T, \quad Av_i = \sigma_i u_i \quad (i = 1, 2) \quad (9.1)$$

此处 A 、 U 、 Σ 、 V 均为 2×2 ； V 和 U 正交， Σ 为非负对角矩阵。 v_i 位于输入平面， u_i 位于输出平面。不要把 Σ 误写成求和号 Σ 。

正交矩阵保持长度和夹角，但可能包含反射。二维正交矩阵行列式为 1 时是绕原点旋转，行列式为 -1 时含反射；高维中同样不能把所有正交变换统称旋转。SVD 程序可对一对奇异向量同时变号，使 U 、 V 的朝向看起来不同，重建 A 仍相同。解释图形时应追踪同一点的计算链，避免根据向量符号猜测算法错误。

程序 `e39_svd2d.py` 固定二维矩阵 $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ 。程序用同一组单位圆点和两条输入基，依次计算 $Vh@points$ 、 $diag(s)@$ 上述坐标、 $U@$ 上述坐标，并核对最后结果等于 $A@points$ 。奇异值约为 $[2.28824561, 0.87403205]$ ，平方为 $3 \pm \sqrt{5}$ ； Vh 就是 V^T 。输出同时核对两侧正交性与重建误差。图的四个阶段均使用同一对应点。

编程思路 对非对称矩阵 $\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ 先从 $A^T A$ 手算奇异值，再用 SVD 获得正交因子。把同一单位圆和基向量按输入、 V^T 、 Σ 、 U 四阶段画出，逐步验证最终轮廓等于直接乘 A 。

代码对应

$A^T A$ 的特征值为 $3 \pm \sqrt{5}$ ，奇异值为其非负平方根 $\rightarrow manual_s = np.sqrt([3 + np.sqrt(5), 3 - np.sqrt(5)])$ ：按从大到小排列，与 `np.linalg.svd` 的返回顺序一致。

第一步用 V^T 变换输入圆 $\rightarrow after_v = Vh @ circle$ ： Vh 就是 V^T ，正交变换保持圆的半径但会改变标记点和基方向。

第二步沿坐标轴伸缩 s_1 、 $s_2 \rightarrow after_sigma = Sigma @ after_v$ ：`diag(s)` 将圆变成椭圆，伸缩倍率是奇异值。

第三步施加 U ，把椭圆移到输出主方向 $\rightarrow after_u = U @ after_sigma$ ：与 $A@circle$ 比较可同时核对因子顺序与矩阵乘法方向。

易错提示 SVD 适用于非对称矩阵，不能把 U 和 V 当成同一特征向量矩阵。一般正交因子可能包含反射，但本例实际返回的两个正交因子行列式都约为 +1，图中不声称本例发生反射。

正文代码 e39_svd2d.py

```
# 教材位置：第 9 章，第 9.2 节；图 9-1。
# 用同一个二维矩阵依次展示  $V^T$ 、 $\Sigma$ 、 $U$  的几何作用。
#  $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$  不是对称矩阵；不可把特征分解直接当作 SVD。
#  $A$ 、 $U$ 、 $V_h$ 、 $\Sigma$  均为  $(2,2)$ ， $s$  为  $(2,)$ ，circle 为  $(2,361)$ 。
# 坐标、矩阵元素均无量纲； $\Sigma$  是奇异值对角矩阵， $\Sigma$  才是求和符号。
# 正交矩阵保持长度和夹角，也可能包含反射，不一定是旋转。
# 固定矩阵无随机抽样；输出奇异值、正交因子行列式及重建误差。
# 执行 python e39_svd2d.py 生成四阶段中文图 svd2d.png。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

A = np.array([[2.0, 1.0], [0.0, 1.0]])
U, s, Vh = np.linalg.svd(A, full_matrices=False)
Sigma = np.diag(s)
# 手算  $A^T A = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$ ，特征方程  $\lambda^2 - 6\lambda + 4 = 0$ 。
# 特征值是  $3 \pm \sqrt{5}$ ，取非负平方根得到降序奇异值。
manual_s = np.sqrt([3 + np.sqrt(5), 3 - np.sqrt(5)])
assert np.allclose(s, manual_s)
assert np.allclose(A.T @ A, [[4, 2], [2, 2]])
assert np.allclose(U.T @ U, np.eye(2))
assert np.allclose(Vh @ Vh.T, np.eye(2))
assert np.allclose(U @ Sigma @ Vh, A)
# NumPy 的 Vh 存放  $V$  的共轭转置；本例全为实数，所以  $V_h = V^T$ 。
# 奇异向量正负可同时翻转；只核对重建和对应方向关系。
assert np.allclose(A @ Vh.T, U * s)
print("手算、数值奇异值：", manual_s, s)
print("正交因子 U 与  $V^T$  的行列式：", np.linalg.det(U), np.linalg.det(Vh))
print("矩阵重建误差：", np.linalg.norm(U @ Sigma @ Vh - A))

# 所有圆周点作为列向量存放，左乘矩阵同时映射全部点。
theta = np.linspace(0, 2 * np.pi, 361)
circle = np.vstack([np.cos(theta), np.sin(theta)])
after_v = Vh @ circle
after_sigma = Sigma @ after_v
after_u = U @ after_sigma
assert np.allclose(after_u, A @ circle)
assert np.allclose(np.linalg.norm(after_v, axis=0), 1)
# 追踪同一批输入基向量的像，而非每幅图重新画标准基。
stage_matrices = [np.eye(2), Vh, Sigma @ Vh, A]
stage_points = [circle, after_v, after_sigma, after_u]
titles = ["输入单位圆", "施加  $V^T$ ：正交变换", "施加  $\Sigma$ ：伸缩", "施加  $U$ ：正交变换"]
# 四个面板使用同样的坐标范围和等比例单位，圆与椭圆形状可直接比较。
# 坐标基不同于奇异向量；基向量箭头只是追踪映射的辅助标记。

configure_chinese_plot()
fig, axes = plt.subplots(2, 2, figsize=(6.3, 5.6), constrained_layout=True)
for ax, points, matrix, title in zip(
    axes.flat, stage_points, stage_matrices, titles
):
    ax.plot(*points, color="tab:blue", lw=1.5)
    # 橙点和短箭头追踪输入圆上的同一个起点及前进方向。
    ax.scatter(*points[:, 0], color="tab:orange", zorder=4, s=20)
    ax.annotate("", xy=points[:, 30], xytext=points[:, 15],
                arrowprops={"arrowstyle": "->", "color": "tab:orange"})
```

```
for column, color in zip(matrix.T, ["tab:green", "tab:red"]):
    ax.annotate("", xy=column, xytext=(0, 0),
                arrowprops={"arrowstyle": "->", "color": color})
ax.axhline(0, color="gray", lw=0.5)
ax.axvline(0, color="gray", lw=0.5)
ax.set(xlim=(-2.7, 2.7), ylim=(-2.7, 2.7),
       xlabel="第一坐标", ylabel="第二坐标", title=title)
ax.set_aspect("equal")
ax.set_xticks([-2, 0, 2])
ax.set_yticks([-2, 0, 2])
ax.grid(alpha=0.2)
# 颜色约定在源码中明确：绿、红箭头分别为第一、第二输入基向量的像。
# 末图应与直接计算 A@circle 相同，不能另换矩阵来追求视觉效果。
fig.savefig("svd2d.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

手算、数值奇异值： [2.28824561 0.87403205] [2.28824561 0.87403205]

正交因子 U 与 V^T 的行列式： 0.9999999999999997 1.0

矩阵重建误差： 5.732038665767851e-16

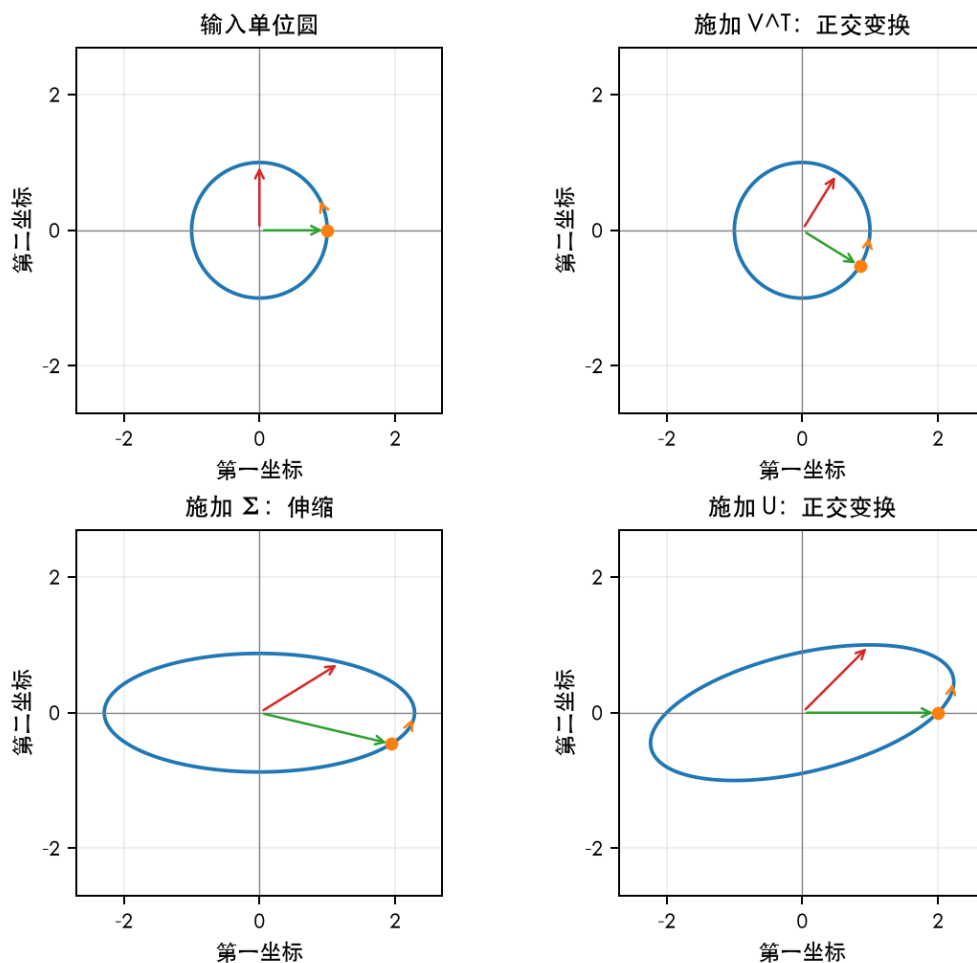


图 9-1 二维矩阵 $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ 的奇异值分解。单位圆依次经过输入正交变换 V^T 、非负轴伸缩 Σ 和输出正交变换 U ，最终与直接作用 A 一致。

读图指南 按输入 $\rightarrow V^T \rightarrow \Sigma \rightarrow U$ 的顺序追踪同色标记点和基向量；前两幅圆形相同并不表示坐标未变。中间椭圆两半轴为 2.28824561 和 0.87403205，最后只改变朝向。

9.3 完整 约化与紧致形式的尺寸

推广到任意 $A \in \mathbb{R}^{m \times n}$ ，都存在 $A = U \Sigma V^T$ 。完整形式中 U 为 $m \times m$ 正交矩阵， V 为 $n \times n$ 正交矩阵， Σ 为 $m \times n$ 矩形对角矩阵，只有主对角线上可能非零。令 $q = \min(m, n)$ ，其对角元素按 $\sigma_1 \geq \dots \geq \sigma_q \geq 0$ 排列，称奇异值。希腊字母 σ 读作“西格玛”；此处是矩阵伸缩量，不是概率章节的标准差。若秩为 r ，就恰好有 r 个正奇异值，其余精确为零。

V 的第 i 列 v_i 称右奇异向量，位于输入侧 $\mathbb{R}^n$ ； U 的第 i 列 u_i 称左奇异向量，位于输出侧 $\mathbb{R}^m$ 。正奇异值对应 $Av_i = \sigma_i u_i$ 。也就是说，先用 V^T 把输入换到一组标准正交坐标，再经 Σ 按各轴伸缩、消去零方向并衔接维数，最后经 U 换到输出坐标。 U 和 V 可以包含反射，统称正交变换比统称旋转更准确。

完整形式保留输入、输出两侧的全部正交方向，适合解释四子空间。常用约化形式令 U_q 为 $m \times q$ ， Σ_q 为 $q \times q$ ， V_q 为 $n \times q$ ，仍有 $A = U_q \Sigma_q V_q^T$ ；它去除矩形 Σ 多余的全零行列。还有紧致秩 r 形式 $U_r \Sigma_r V_r^T$ ，只保留正奇异值项。这三种形式都可能被简称为“约化”，阅读教材或接口时应直接核对尺寸。NumPy `full_matrices=False` 返回 q 项形式，不会自动删去数值接近零的项。

当 $m < n$ 时，输入方向比输出方向多，至少 $n - m$ 个输入方向会被压成零；当 $m > n$ 时，输出空间中至少 $m - n$ 个方向没有对应的输入像。 Σ 的矩形形状正好记录这种维数衔接。例如 $m=3$ 、 $n=2$ 、 $r=1$ 时，完整 U 有 3 列、 V 有 2 列， Σ 为 3×2 且只有一个正对角项； q 项形式 U_q 为 3×2 ，仍含一个零奇异方向，不能把其列数 2 当成秩。

SVD 因子并不总唯一。某个奇异向量对 (u_i, v_i) 同时变号，乘积 $\sigma_i u_i v_i^T$ 不变。重复奇异值对应的子空间可以内部正交换基；零奇异空间的基也有多种选择。因此比较两次分解时，应先比较奇异值、重建和相关子空间，而不是要求每个向量坐标逐字相同。

$$A = U \Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T, \quad Av_i = \sigma_i u_i \quad (9.2)$$

A 为 $m \times n$, r 为秩。 u_i 是 m 维单位列, v_i 是 n 维单位列, $u_i v_i^T$ 为 $m \times n$ 秩 1 外积; 第二式针对 $i=1, \dots, r$ 的正奇异值方向。

9.4 从实对称谱定理完整构造矩形 SVD

现在证明每个实矩阵都具有这种结构。令 $A \in \mathbb{R}^{m \times n}$, 先形成理论上的 $n \times n$ 矩阵 $G = A^T A$ 。它对称, 且 $x^T G x = \|Ax\|_2^2 \geq 0$, 故由第 8 章实对称谱定理, 存在 $\mathbb{R}^n$ 的一组标准正交特征向量 $v_1, \dots, v_n$, 对应特征值 $\lambda_1 \geq \dots \geq \lambda_n \geq 0$ 。把正值个数记为 r , 定义 $\sigma_i = \sqrt{\lambda_i}$, $i=1, \dots, r$ 。[MIT-L10]

核为什么相同? $Ax=0$ 显然推出 $Gx=0$; 反过来 $Gx=0$ 给出 $\|Ax\|_2^2 = x^T G x = 0$, 故 $Ax=0$ 。因此 $\ker(G) = \ker(A)$, 秩—零度定理又说明 $r = \text{rank}(A)$ 。正特征方向负责所有可辨认输入, 其余 $n-r$ 个 v_i 恰张成 A 的核。不能对这些零方向使用除以 σ_i 的公式。

$$u_i = \frac{A v_i}{\sigma_i}, \quad u_i^T u_j = \frac{v_i^T A^T A v_j}{\sigma_i \sigma_j} = \frac{\lambda_j v_i^T v_j}{\sigma_i \sigma_j} \quad (9.3)$$

此式只用于 $i, j \leq r$ 的正奇异值。 $i \neq j$ 时内积为 0; $i=j$ 时 $\lambda_i = \sigma_i^2$ 使其为 1, 因此 $u_1, \dots, u_r$ 是输出侧的一组标准正交向量。

这些 u_i 张成 $\text{Col}(A)$: 把任意 x 按 $v_1, \dots, v_n$ 展开, 后 $n-r$ 项经过 A 全为零, 前 r 项则分别变成 $\sigma_i u_i$ 。将 $u_1, \dots, u_r$ 补为 $\mathbb{R}^m$ 的完整标准正交基 $u_1, \dots, u_m$; 补充的 $m-r$ 个方向位于 $\text{Col}(A)^\perp = \ker(A^T)$ 。这里扩充的是输出空间的基, 维数是 m , 不能把输入侧的 n 个 v 直接搬来充当 U 。

令 $U = [u_1 \dots u_m]$, $V = [v_1 \dots v_n]$ 。在 $m \times n$ 矩形矩阵 Σ 的前 r 个对角位置填入 $\sigma_1, \dots, \sigma_r$, 其他位置全填 0。逐列核对 $AV = U\Sigma$: 前 r 列由 u_i 定义成立, 其余列两侧都为零。右乘 V^T , 因 $VV^T = I_n$, 得到 $A = U\Sigma V^T$, 完整构造结束。零矩阵也包括在内: $r=0$ 、 $\Sigma=0$, 任意合适尺寸的正交 U 、 V 都可。

$$AV = U\Sigma \quad \Rightarrow \quad A = U\Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T \quad (9.4)$$

AV 与 $U\Sigma$ 均为 $m \times n$ 。 $u_i v_i^T$ 是 $m \times n$ 秩 1 外积, 所有正奇异层相加恢复 A 。矩形 Σ 的零行列由尺寸决定。

重复正奇异值不会破坏存在性。对应特征子空间内可任意选标准正交基，只要在两侧用相同的正交换基，乘积保持不变。单个向量的符号也不唯一。 AA^T 的非零特征值同样为 σ_i^2 ，因为 $AA^T u_i = \sigma_i^2 u_i$ ；其余 $m-r$ 个为零。 $A^T A$ 有 $n-r$ 个零值，两个零值个数不必相同。

这个构造用于证明和手算。数值计算通常不显式形成 $A^T A$ 再求特征值，因为条件数平方可能掩盖小奇异值。成熟 SVD 实现直接对 A 进行稳定分解。理解存在性证明与选择数值算法是两个环节：公式能够推导，并不意味着应逐字照写成计算流程。

例题 手算：一个可完整核对的矩形 SVD

取 $B = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ 。输入侧 $V = I_2$ ，完整 $U = I_3$ ， $\Sigma = B$ ，奇异值为 2 和 1，已经得到完整 SVD。约化 U_q 仅保留 I_3 前两列， $\Sigma_q = \text{diag}(2, 1)$ ， $V_q = I_2$ ，乘积仍为同一 B 。左零空间由 $(0, 0, 1)^T$ 张成，但它不会出现在约化 U_q 中。

若再把 B 左乘一个三维正交矩阵 W ，得到 $A = WB$ ，则 $U = W$ 、 Σ 仍为 B 、 V 仍为 I_2 。奇异值不变，输出平面改变。这展示了为什么输入方向与输出方向应分别用 V 与 U 描述，不能强求 $U = V$ ，甚至它们的尺寸都可能不同。

程序 e28_svd.py 本例实际矩阵 $A = \begin{bmatrix} 3 & 1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}$ ，形状 3×2 。约化输出 $U(3, 2)$ 、 $s(2,)$ 、 $V_h(2, 2)$ ，奇异值约 $[3.377147, 1.896016]$ ； V_h 就是 V^T ，不是 V 。完整形式 $U_f(3, 3)$ 、 $\text{Sigma}(3, 2)$ 、 $V_h f(2, 2)$ 。程序逐阶段跟踪同一个二维单位圆，先 $V_h f$ 、再 Sigma 进入三维、再 U_f ，最终逐项等于 $A @ \text{input_circle}$ 。

编程思路 对 3×2 矩阵同时计算紧致与完整 SVD，检查重构和正交性。把二维单位圆依次经过 $V.T$ 、矩形 Σ 、 U ，展示从二维到三维的变换，并解释完整 U 多出的一列是左零空间方向。

代码对应

$A = U \text{ diag}(s) V.T$ 的紧致分解 $\rightarrow U, s, V_h = \text{np.linalg.svd}(A, \text{full_matrices}=\text{False})$ ：
 $\text{full_matrices}=\text{False}$ 给出 $U(3, 2)$ 、 $s(2,)$ 、 $V_h(2, 2)$ ，足够重构 A 。

完整 Σ 是 3×2 的矩形矩阵 $\rightarrow \text{Sigma} = \text{np.zeros}(A.\text{shape})$ ：先建零矩阵再把两个奇异值填入左上对角块，匹配完整 $U_f(3, 3)$ 。

先把输入圆施加 $V.T \rightarrow \text{after_v} = \text{Vhf} @ \text{input_circle}$: V_h 已经是转置后的矩阵，不能再转置一次。

沿两个正交方向按奇异值伸缩并嵌入三维 $\rightarrow \text{after_sigma} = \text{Sigma} @ \text{after_v}$: 矩形 Σ 把二维坐标变成第三坐标为零的三维坐标，随后 U_f 旋转到像空间。

易错提示 NumPy 的 s 是奇异值向量，不是矩阵； $(U*s)$ 利用列广播等价于 $U@diag(s)$ 。完整 U 的额外列与 A 的列空间正交，不应误当成第三个非零奇异方向。

正文代码 e28_svd.py

```
# 教材位置：第 9 章，第 9.4 节；图 9-2。
# 单位、变量与形状：A 为 (3,2)，输入圆为 (2,200)，输出为 (3,200)；示意坐标和矩阵元素无量纲。
# 案例 28：比较精简 奇异值分解 与完整 奇异值分解，逐项核对形状和重建关系。
# 学习目标：理解任意实矩阵  $A=U\Sigma V^T$ ，及奇异值与  $A^TA$  特征值、弗罗贝尼乌斯 范数的联系。
# 本例 A 为 3×2、秩为 2；精简分解 U 为 3×2、s 为长度 2、Vh 为 2×2。
# NumPy 的 Vh 存储 V 的共轭转置；本例为实数，故  $V_h=V^T$ 。
# 完整分解 U 为 3×3，矩形  $\Sigma$  为 3×2，Vh 仍为 2×2；额外的 U 列补齐正交基。
# 输出：打印精简与完整分解的形状、奇异值；用 assert 验证各项关系，生成四阶段几何图
# e28_svd.png。
# 运行方式：安装依赖后在本目录执行 python e28_svd.py。
# 本例没有随机数据；allclose/isclose 允许线性代数算法的浮点误差。

# 用 NumPy 的 奇异值分解 与对称特征值算法作相互独立的数值核对。
import numpy as np
# 两列线性无关，故 rank(A)=2；非方阵也可以进行奇异值分解。
A = np.array([[3.0, 1.0], [0.0, 2.0], [1.0, 0.0]])
# full_matrices=False 返回精简形式，内维度 k=min(3,2)=2；
# s 按从大到小排列且非负，U 的列与 Vh 的行按同一顺序一一对应。
# 精简形式保留 k 个奇异值（必要时也包含零），不等同于自动删去全部零奇异值。
U, s, Vh = np.linalg.svd(A, full_matrices=False)
# 先读形状有助于确认 (3,2)@(2,2)@(2,2) 最后重建成 (3,2)。
print("精简分解各因子的形状 =", U.shape, s.shape, Vh.shape)
# 奇异值描述沿右奇异向量方向的伸缩量，打印时舍入并不修改 s。
print("奇异值 =", np.round(s, 6))
# 用形状断言防止把完整分解与精简分解混用。
assert U.shape == (3, 2) and Vh.shape == (2, 2)
# U*s 将 U 的第 j 列乘以 s[j]，等价于 U@diag(s)，但无需显式构造对角矩阵。
# 于是 (U*s)@Vh 就是  $\sum s_j u_j v_j^T$ ，应该重建原矩阵。
assert np.allclose((U * s) @ Vh, A)
# 检验 U 的列与 Vh 的行标准正交。精简 U 不是方阵，
#  $U^T U = I_2$ ，但  $U U^T$  是投影到 A 列空间的 3×3 矩阵，并不等于  $I_3$ 。
assert np.allclose(U.T @ U, np.eye(2)) and np.allclose(Vh @ Vh.T, np.eye(2))
# Vh.T 的列是右奇异向量  $v_j$ ； $A v_j = s_j u_j$  表示“先选方向，再伸缩到左奇异方向”。
assert np.allclose(A @ Vh.T, U * s)
#  $A^T A$  为 2×2 对称半正定矩阵，特征值等于奇异值的平方。
# eigvalsh 按升序返回，s 按降序返回，因此用 s[::-1] 反向后再逐项比较。
assert np.allclose(np.linalg.eigvalsh(A.T @ A), s[::-1] ** 2)
# 完整形式把左奇异向量扩充为三维空间的整组正交基；奇异值仍只返回两个。
Uf, sf, Vhf = np.linalg.svd(A, full_matrices=True)
# 构造 (3,2) 的矩形  $\Sigma$ ：前两个对角位置放奇异值，第三行补零。
```

```

Sigma = np.zeros(A.shape)
Sigma[:2, :2] = np.diag(sf)
# 完整形式的矩阵乘法形状为 (3,3)@(3,2)@(2,2)=(3,2)。
print("完整分解各因子的形状 =", Uf.shape, Sigma.shape, Vhf.shape)
# 完整与精简形式虽形状不同,但都必须正确重建 A。
assert np.allclose(Uf @ Sigma @ Vhf, A)
# Uf 的额外第三列正交于 A 的全部列,所以属于  $\ker(A^T)$ ,即左零空间。
# 它在完整重建中乘以  $\Sigma$  的零行,因此不改变 A。
assert np.allclose(A.T @ Uf[:, 2], 0)
# 弗罗贝尼乌斯范数平方为所有元素平方和,也等于奇异值平方和  $\sum s_j^2$ 。
assert np.isclose(np.linalg.norm(A, "fro") ** 2, s @ s)

# 用同一个实际 3×2 矩阵展示完整奇异值分解的几何阶段:输入是二维,输出是三维。
# 完整因子 Uf(3,3)、Sigma(3,2)、Vhf(2,2)来自上面的实际分解,不另外换成 2×2 矩阵。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
angles = np.linspace(0, 2 * np.pi, 200)
input_circle = np.vstack([np.cos(angles), np.sin(angles)])
# 每一列是一个输入点;分别施加  $V^T$ 、矩形  $\Sigma$  和  $U$ ,阶段形状清楚显示维数衔接。
after_v = Vhf @ input_circle
after_sigma = Sigma @ after_v
after_u = Uf @ after_sigma
assert input_circle.shape == after_v.shape == (2, 200)
assert after_sigma.shape == after_u.shape == (3, 200)
assert np.allclose(after_u, A @ input_circle)
#  $\Sigma$  的第三行是零,因此第三阶段在三维空间的坐标平面内;Uf 将该平面映到 A 的列空间。
assert np.allclose(after_sigma[2], 0)
assert np.allclose(Uf.T @ after_u, after_sigma)
assert np.allclose(Uf @ Sigma @ Vhf, A)
# 标准基向量也随阶段同步变换,能看出正交因子含旋转或反射,而不只看到无标记圆。
stage_matrices = [np.eye(2), Vhf, Sigma @ Vhf, Uf @ Sigma @ Vhf]
stage_points = [input_circle, after_v, after_sigma, after_u]
fig = plt.figure(figsize=(6.3, 6.4))
titles = ["1 输入: 二维单位圆", "2  $V^T$ : 二维到二维", "3  $\Sigma$ : 二维到三维",
          "4  $U$ : 三维到三维"]
for index, (points, matrix, title) in enumerate(zip(stage_points, stage_matrices,
                                                    titles)):
    # 所有面板共用相同数值范围;三维图使用正交投影和相同坐标单位比例。
    if index < 2:
        ax = fig.add_subplot(2, 2, index + 1)
        ax.plot(*points, color="tab:blue", lw=1.3)
        ax.scatter(*points[:, 0], color="tab:orange", s=15, zorder=4)
        ax.annotate("", xy=points[:, 15], xytext=points[:, 7],
                    arrowprops={"arrowstyle": "->", "color": "tab:orange", "lw":
1.0})
        for column, color in zip(matrix.T, ["tab:green", "gray"]):
            ax.annotate("", xy=column, xytext=(0, 0),
                        arrowprops={"arrowstyle": "->", "color": color})
        ax.set(xlim=(-3.8, 3.8), ylim=(-3.8, 3.8), xlabel="$x_1$", ylabel="$x_2$")
        ax.set_aspect("equal")
        ax.set_xticks([-3, 0, 3])
        ax.set_yticks([-3, 0, 3])
        ax.grid(alpha=0.2)
    else:
        ax = fig.add_subplot(2, 2, index + 1, projection="3d")

```

```

ax.plot(*points, color="tab:blue", lw=1.3)
ax.scatter(*points[:, 0], color="tab:orange", s=15)
# 沿圆周前进方向的小箭头与橙点共同标出方向; 基向量箭头从原点出发。
direction = points[:, 15] - points[:, 7]
ax.quiver(*points[:, 7], *direction, color="tab:orange",
          arrow_length_ratio=0.45)
for column, color in zip(matrix.T, ["tab:green", "gray"]):
    ax.quiver(0, 0, 0, *column, color=color, arrow_length_ratio=0.12)
ax.set(xlim=(-3.8, 3.8), ylim=(-3.8, 3.8), zlim=(-3.8, 3.8))
ax.set_xticks([-3, 0, 3])
ax.set_yticks([-3, 0, 3])
ax.set_zticks([-3, 0, 3])
ax.set_box_aspect((1, 1, 1))
ax.set_proj_type("ortho")
ax.view_init(elev=32, azim=-125)
# 三维轴标签放在本面板内部; 两行两列的布局让方向标记与刻度在教材页宽下仍可辨认。
ax.set_xlabel("$x_1$", fontsize=8, labelpad=-5)
ax.set_ylabel("$x_2$", fontsize=8, labelpad=-5)
ax.text2D(0.91, 0.58, "$x_3$", transform=ax.transAxes, fontsize=8)
ax.tick_params(labelsize=8, pad=0)
ax.set_title(title, fontsize=9.5, pad=5)
ax.tick_params(labelsize=8)
# 因输入输出维度不同, 第三、四面板必须是三维显示, 不能沿用二维圆→椭圆四图。
fig.text(0.5, 0.015, "同一矩阵 A (3×2) ; 橙色标记显示方向变化", ha="center", fontsize=8)
fig.tight_layout(rect=(0, 0.07, 1, 1), pad=0.5)
fig.savefig("e28_svd.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

规定环境下的实际运行输出

精简分解各因子的形状 = (3, 2) (2,) (2, 2)

奇异值 = [3.377147 1.896016]

完整分解各因子的形状 = (3, 3) (3, 2) (2, 2)

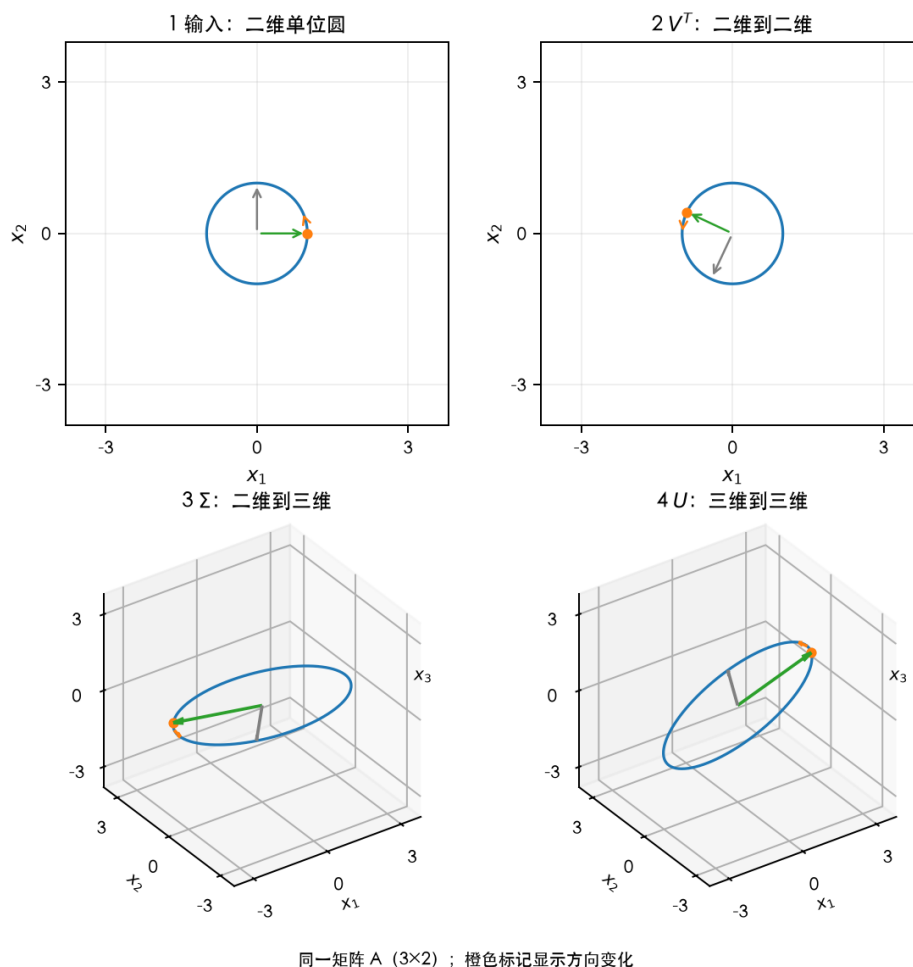


图 9-2 e28 的实际 3×2 矩阵 SVD。顺序为左上输入圆、右上 V^T 后的圆、左下 Σ 伸缩并嵌入三维坐标平面、右下 U 把结果映到 A 的列空间。橙色点在四图中对应同一输入，不能把后两阶段误读为二维方阵变换。

读图指南 按输入圆、 V^T 后的圆、 Σ 伸缩嵌入三维、 U 映入三维列空间的顺序追踪橙色标记。后两幅是三维输出阶段，不能把整幅图当作四个二维方阵变换。

9.5 四子空间 范数与敏感方向

完整 SVD 给出四空间的标准正交基： $u_1, \dots, u_r$ 张成列空间， $v_1, \dots, v_r$ 张成行空间； $v_{r+1}, \dots, v_n$ 张成零空间， $u_{r+1}, \dots, u_m$ 张成左零空间。若某组没有向量，对应空间就是零空间 $\{0\}$ 。它们的维数分别为 r 、 r 、 $n-r$ 、 $m-r$ ，与第 7 章消元得到的结果相同；SVD 额外把这些基整理成正交单位方向。

对输入 $x = \sum_i c_i v_i$ ，由正交性有 $\|x\|_2^2 = \sum_i c_i^2$ ，而 $\|Ax\|_2^2 = \sum_{i=1}^r \sigma_i^2 c_i^2$ 。单位输入的最大输出长度由最大 σ_1 达到，因此矩阵二范数 $\|A\|_2 = \sigma_1$ 。弗罗贝尼乌斯范数定义为 $\|A\|_F = \sqrt{\sum_{ij} a_{ij}^2}$ ，由标准正交变

换不改变平方和，可得 $\|A\|_F^2 = \sum_i \sigma_i^2$ 。前者看最强方向，后者看所有条目或所有正交方向的总能量。

对可逆方阵，最小输出长度为 σ_n ，逆变换在该方向放大 $1/\sigma_n$ ，所以 $\kappa_2(A) = \sigma_1/\sigma_n$ 。如果 σ_n 为零，矩阵已不可逆；如果 σ_n 只是很小，问题可逆却敏感。数据噪声和浮点误差可能使精确零变成小非零，数值秩据阈值 τ 统计 $\sigma_i > \tau$ 的项数。 τ 必须结合数据尺度、精度和用途理解，不能把数值秩当作脱离容差的永恒真值。

对满列秩矩形 A ， σ_1/σ_n 也衡量可辨认输入方向之间的相对伸缩，并控制最小二乘敏感度的重要部分；但最小二乘误差还会受 b 到列空间的夹角等因素影响，不能把方阵右端扰动界直接原样套给所有回归问题。统一结构的价值是看清哪些方向接近丢失，同时保留各问题自己的条件。

$$\|A\|_2 = \sigma_1, \quad \|A\|_F^2 = \sum_{i=1}^q \sigma_i^2, \quad \kappa_2(A) = \frac{\sigma_1}{\sigma_n} \quad (9.5)$$

$q = \min(m, n)$ ；前两式适用于任意实矩阵，零矩阵的 σ_1 为零。第三式这里要求 A 是 $n \times n$ 可逆矩阵，因而 $\sigma_n > 0$ 。

9.6 最佳低秩近似 误差下界与达到条件

SVD 把矩阵写成 r 个秩 1 层的叠加。第 i 层 $\sigma_i u_i v_i^T$ 的每一列都是 u_i 的倍数，每一行都是 v_i^T 的倍数，因此只提供一个独立方向。保留前 k 层得到 $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$ ，秩至多 k 。 $k=0$ 时约定 A_0 为零矩阵， $k \geq r$ 时可重建 A 。低秩不是把小元素逐个删掉，而是保留能组织全矩阵的少数方向。

这些秩 1 层在弗罗贝尼乌斯内积 $\langle B, C \rangle_F = \text{tr}(B^T C)$ 下正交，所以删除后部层的误差平方是被删奇异值平方之和：其表达式见下方的尾部平方和公式。矩阵二范数误差是下一项 σ_{k+1} 。这里“正交”作用在整个矩阵空间，和第 7 章对向量正交分解使用同一逻辑。

最佳低秩近似定理断言：对 $0 \leq k < r$ ， A_k 在全部秩至多 k 的 B 中同时达到最小弗罗贝尼乌斯误差与最小二范数误差。先证明总平方误差。给任意 B ，令 W 为其列空间， $\ell = \dim W \leq k$ ， P 为

到 W 的正交投影。对 A 的每一列 a_j 与 B 的对应列 b_j ，分解 $a_j - b_j = (I - P)a_j + (Pa_j - b_j)$ ，两项正交。逐列平方求和，得到 $\|A - B\|_F^2 \geq \|(I - P)A\|_F^2 = \|A\|_F^2 - \|PA\|_F^2$ 。

$$\|A - B\|_F^2 \geq \|A\|_F^2 - \|PA\|_F^2 \quad (9.6)$$

P 投影到 B 的列空间。每一列无法投进 W 的部分都是任何 B 都必须付出的误差；调整 W 内的列只能再增加非负误差。

还需证明一个 ℓ 维投影最多保留最大的 ℓ 个奇异方向能量。把完整 U 基用于计算，记 $d_i = \|Pu_i\|_2^2$ ， $i=1, \dots, m$ 。正交投影不增加长度，所以 $0 \leq d_i \leq 1$ ；若 W 的标准正交基为 $w_1, \dots, w_\ell$ ，交换有限求和得 $\sum_i d_i = \sum_h \sum_i (w_h^T u_i)^2 = \ell$ 。SVD 层的右向量正交又给出 $\|PA\|_F^2 = \sum_{i=1}^r \sigma_i^2 d_i$ 。将 $\sigma_i = 0$ 补到 $i=m$ ，使式子覆盖全部输出方向。

为何这份加权能量不超过前 k 项平方和？若 d 在较小权重 σ_j^2 处有正质量，而更大权重 σ_i^2 处尚未达到 1，把一小部分质量从 j 移到 i 不会减小总和。反复转移，固定总质量 ℓ 的最大值在前 ℓ 个 d 全为 1 时达到。因此 $\sum_i \sigma_i^2 d_i \leq \sum_{i=1}^{\ell} \sigma_i^2 \leq \sum_{i=1}^k \sigma_i^2$ 。结合上一不等式，任何秩 $\leq k$ 的 B 都满足 $\|A - B\|_F^2 \geq \sum_{i=k+1}^r \sigma_i^2$ 。 A_k 的列空间选前 k 个 u ，且在空间内取精确投影，恰达到这个下界。

$$\|A - B\|_F^2 \geq \sum_{i=k+1}^r \sigma_i^2 = \|A - A_k\|_F^2 \quad (\text{rank } B \leq k) \quad (9.7)$$

这给出弗罗贝尼乌斯范数最优性的完整下界与达到条件。 $k \geq r$ 时取 A 本身即可使误差为零。

二范数结论也要单独证明。在前 $k+1$ 个右奇异向量张成的空间里， B 至多有 k 个独立输出，因此 B 在该空间内必有一个单位核向量 z 。写 $z = \sum_{i=1}^{k+1} c_i v_i$ ，则 $\sum c_i^2 = 1$ ，且 $\|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum \sigma_i^2 c_i^2 \geq \sigma_{k+1}^2$ 。所以 $\|A - B\|_2 \geq \sigma_{k+1}$ ，而截断余项 $A - A_k$ 的最大伸缩正是 σ_{k+1} 。这证明 A_k 也达到最小最坏方向误差，且没有循环使用要证明的结论。

若 $\sigma_k = \sigma_{k+1}$ ，最优子空间可能不唯一，但最小误差值仍明确。最优也只针对所说的范数和秩约束，不保证人眼最满意、分类最准或异常最容易发现。像素平方误差可能偏好保留大片平滑区域，却牺牲小而关键的细节；实践中应把数学最优与任务目标联系起来。

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T, \quad \|A - A_k\|_F^2 = \sum_{i=k+1}^q \sigma_i^2 \quad (9.8)$$

A_k 为截断 SVD, k 为保留方向数, $0 \leq k \leq q$; 当 k 超过秩时剩余项为零。所有奇异值必须按从大到小排列。

累计能量 E_k 为前 k 个奇异值平方和除以全部奇异值平方和, 报告保留的平方范数比例, 要求 A 不全为零。它满足相对弗罗贝尼乌斯误差 $\|A - A_k\|_F / \|A\|_F = \sqrt{1 - E_k}$ 。所以“保留 99%能量”对应约 10%的相对误差, 而不是 1%; 原因是能量报告平方量。选择 k 可依误差预算、谱拐点或后续验证, 但数据有噪时不存在自动等于“真实维数”的万能拐点。

例题 手算: 能量百分比和误差百分比

设奇异值为 4、3、1。总平方能量 $16+9+1=26$ 。保留前两项时能量为 $25/26 \approx 96.15\%$, 弗罗贝尼乌斯绝对误差为 1, 相对误差为 $1/\sqrt{26} \approx 19.61\%$ 。能量损失只有 3.85%, 长度型误差却约 19.61%, 两者不能混为同一百分比。

若只保留第一项, 平方误差为 $3^2+1^2=10$, 二范数误差为 3。这两个数分别回答总平方损失与最坏方向损失; 报告“误差是 3”时, 必须先说明所用范数。

程序 `e29_lowrank.py` 固定种子 9, 以 $(60,4)@(4,80)$ 生成秩至多 4 的信号, 再加标准差 0.25 的 60×80 高斯噪声。观测矩阵一般不再精确秩 4。前四项能量约 0.987269; $k=2, 4, 10$ 时弗罗贝尼乌斯误差分别约 87.947679、16.560696、13.866765, 均与尾部奇异值的二范数一致。保留四项的相对误差约 $\sqrt{1-0.987269}$, 不能说成 1.2731%。

编程思路 先构造低秩信号加噪声的 60×80 矩阵, 再用截断 SVD 比较秩 2、4、10 的重构。把累计平方奇异值占比与 弗罗贝尼乌斯 相对误差联系起来, 并用被舍弃奇异值的平方和复核误差。

代码对应

以 60×4 与 4×80 相乘生成秩不超过四的信号, 再加噪声 $\rightarrow A = \text{rng.normal(size=(60, 4))} @ \text{rng.normal(size=(4, 80))}$: 噪声会使实际矩阵一般不再严格秩四, 因此截断并非无损。

前 k 个奇异方向的平方能量占比 $\sum_1^k s_j^2 / \sum s_j^2 \rightarrow \text{cumulative} = \text{np.cumsum}(s ** 2) / (s$

@ s): 这里度量 弗罗贝尼乌斯 平方范数的保留比例, 不是直接对奇异值求和。

$A_k = U_k \text{diag}(s_1, \dots, s_k) V_k^T \rightarrow \text{approximation} = (U[:, :k] * s[:k]) @ Vh[:, :k]$: U 的前 k

列乘长度 k 的 s , 按列广播后再乘 Vh 的前 k 行。

最优秩 k 弗罗贝尼乌斯 误差为 $\sqrt{\sum_{j>k} s_j^2} \rightarrow \text{tail} = \text{np.linalg.norm}(s[k:])$: 与直接

$\text{norm}(A - \text{approximation}, 'fro')$ 比较, 核对 埃卡特—杨 公式。

易错提示 能量保留比例高不保证每个元素误差都小。 $k=4$ 对应生成信号的秩上界, 但加入噪声后仍有残余, 不能声称完整还原了无噪声信号。

正文代码 e29_lowrank.py

```
# 教材位置: 第 9 章, 第 9.6 节; 图 9-3。
# 单位、变量与形状: A 为 (60,80), s 为 (60,); 矩阵元素用一般测量单位, 能量比例无量纲。
# 案例 29: 截断 奇异值分解、最佳低秩近似与保留能量。
# 学习目标: 用低秩信号加噪声的数据观察奇异谱, 并用数学恒等式解释重建误差。
# 核心关系:  $A_k = \sum_{j=1}^k s_j u_j v_j^T$  在  $\text{rank} \leq k$  的矩阵中使 弗罗贝尼乌斯 误差最小;
#  $\|A - A_k\|_F^2 = \sum_{\{j>k\}} s_j^2$ ; 保留能量为前  $k$  个奇异值平方和占比。
# 数据与形状: 两个高斯矩阵 (60,4) @ (4,80) 产生秩至多为 4 的信号,
# 再加独立高斯噪声得到 A(60,80); 精简 奇异值分解 的 U、s、Vh 形状为 (60,60)、(60,)、(60,
# 80)。
# 输出: 打印前四项能量、 $k=2/4/10$  的误差和谐尾范数; 生成 e29_lowrank.png。
# 运行方式: 安装依赖后在本目录执行 python e29_lowrank.py, 图保存在当前工作目录。

# NumPy 生成模拟矩阵并计算 奇异值分解、累计和以及矩阵/向量范数。
import numpy as np
# 两幅图分别展示奇异值衰减和累计能量, 帮助选择近似秩。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定随机种子 9, 便于在相同软件环境中重复生成数据、核对结果。
rng = np.random.default_rng(9)
# 标准正态矩阵的乘积最多只有 4 个独立列方向, 故无噪声信号秩不超过 4。
# 低秩结构来自因子维度, 而不是简单地把矩阵的大部分元素设为零。
A = rng.normal(size=(60, 4)) @ rng.normal(size=(4, 80))
# 在每个元素上加入均值 0、标准差 0.25 的独立噪声, scale 不是方差。
# 加噪后矩阵通常满秩, 因此第 5 项以后的奇异值一般非零。
A += rng.normal(scale=0.25, size=A.shape)
# 精简 奇异值分解 按奇异值从大到小排列; 最大的几个分量通常承载主要的低秩信号。
U, s, Vh = np.linalg.svd(A, full_matrices=False)
# cumsum 得到前 1、2、...项奇异值平方和, 分母为  $\|A\|_F^2$ 。
# 这里“能量”特指原矩阵的平方和; 没有中心化, 所以不直接等同于 主成分分析 的解释方差。
cumulative = np.cumsum(s ** 2) / (s @ s)
# Python 从 0 编号, 索引 3 表示前 4 项累计保留能量, 呼应信号的因子维度 4。
print("前四个分量的能量保留比例 =", round(cumulative[3], 6))
# 比较低于、等于、高于原信号秩的三种截断选择, 观察精度随 k 的变化。
```

```

for k in [2, 4, 10]:
    # U[:, :k] 为 (60,k), s[:k] 按列广播缩放, Vh[:k] 为 (k,80)。
    # 只保留前 k 个外积项, 仍重建为 (60,80), 秩至多为 k。
    approximation = (U[:, :k] * s[:k]) @ Vh[:k]
    # 弗罗贝尼乌斯 误差把全部元素的重建差平方相加后开方, 衡量整张矩阵的总体误差。
    error = np.linalg.norm(A - approximation, "fro")
    # s[k:] 是被舍弃的奇异值; 向量默认的二范数即  $\sqrt{\sum_{j>k} s_j^2}$ 。
    tail = np.linalg.norm(s[k:])
    # 将直接计算的矩阵误差与谱尾范数并排打印, 展示截断 奇异值分解 的误差公式。
    print(f"k={k}: 误差={error:.6f}, 舍弃奇异值的二范数={tail:.6f}")
    # 核对最佳低秩近似定理的误差等式, 而不只是凭图判断近似是否好。
    assert np.isclose(error, tail)
    # 相对误差= $\sqrt{1-\text{保留能量}}$ , 可把无单位的精度要求转成累计能量要求。
    assert np.isclose(error / np.linalg.norm(A, "fro"),
                       np.sqrt(1 - cumulative[k - 1]))
# 创建左右两图, axes 为包含两个坐标轴的一维数组。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3))
# 横轴从 1 开始表示第几个奇异值; 纵轴取对数, 使大信号与小噪声尺度都可见。
axes[0].semilogy(np.arange(1, len(s) + 1), s, "-")
# 奇异谱的明显落差常提示有效低秩结构; 具体选 k 仍应结合应用容许的误差。
axes[0].set(xlabel="序号", ylabel="奇异值", title="奇异值谱")
# 累计能量随 k 单调不减, 保留全部分量时应接近 1。
axes[1].plot(np.arange(1, len(s) + 1), cumulative)
# 该图读取的是保留平方和比例; 例如接近 1 不代表每个元素都有同样的小误差。
axes[1].set(xlabel="k", ylabel="能量保留比例", title="累计能量比例")
# 为两个坐标轴添加淡网格, 便于比较谱的尺度及累计比例。
for ax in axes: ax.grid(alpha=0.3)
# 整理布局、保存图片并释放绘图资源, 方便 run_all.py 连续执行。
fig.tight_layout()
fig.savefig("e29_lowrank.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

规定环境下的实际运行输出

前四个分量的能量保留比例 = 0.987269

k=2: 误差=87.947679, 舍弃奇异值的二范数=87.947679

k=4: 误差=16.560696, 舍弃奇异值的二范数=16.560696

k=10: 误差=13.866765, 舍弃奇异值的二范数=13.866765

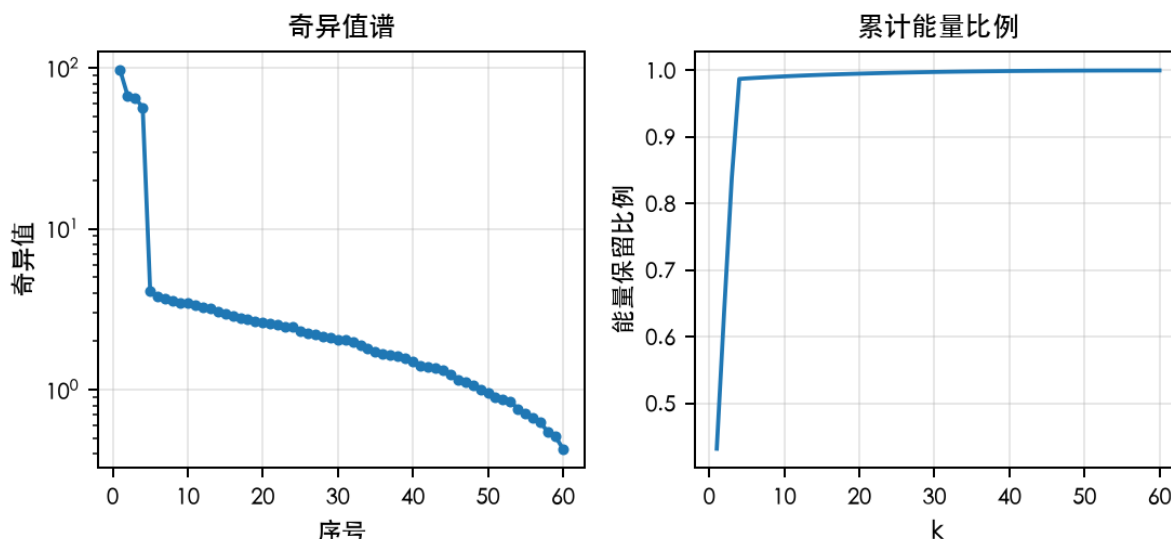


图 9-3 e29 同一 60×80 含噪矩阵的奇异值谱与累计能量。信号由 4 个因子方向生成，加入噪声后后部奇异值仍非零；前四项累计能量约 98.7269%。

读图指南 左图先观察前几项与后部奇异值的尺度差异，再在右图读取同一 k 对应的累计平方能量。加噪后尾部仍非零，累计 98.7269% 能量也不等于只剩 1.2731% 的范数误差。

9.7 图像矩阵的误差 显示与存储

灰度图像可看作 $m \times n$ 矩阵，每个条目表示一个像素亮度。截断 SVD 近似保留 k 个左向量、 k 个奇异值及 k 个右向量，需 $k(m+n+1)$ 个实数，而原矩阵需 mn 个实数。理论标量存储比为 $k(m+n+1)/(mn)$ ；只有该比例小于 1 才在这个模型下节省标量数。这里没有计入文件头、索引、数值精度、熵编码等，因此不是某种 JPEG 文件大小的预测。

重建矩阵的元素可能略低于 0 或高于 1。显示软件可以为可视化把范围裁到 $[0,1]$ ，但若误差是按未裁剪重建计算，必须保持这一口径；不能看见显示后的图更清楚就把原误差也随意改写。多幅图比较时要固定相同亮度范围，否则自动拉伸对比度会掩盖真实差别。压缩效果也需同时报告 k 、误差与存储账，不能只挑一张看起来不错的图。

低秩常能抓住平滑背景和全局规律，却需要更多项表达尖锐边缘、文字和局部纹理。噪声也会占据部分后部方向，因此增加 k 总会减小重建训练矩阵的平方误差，却未必让图像更符合“无噪真实场景”。如果目标是去噪，应根据噪声结构和独立信息选择截断；如果目标是原数据无损存储，则截断本身就是有损的。

程序 e30_image.py 固定种子 12，原图为 100×140 合成矩阵： $0.5 + 0.2\sin(8x)\cos(5y)$ + 一个中心在 $(0.25, -0.1)$ 的高斯形亮斑，再加标准差 0.035 的噪声。 $k=2, 8, 25$ 时，理论标量存储比分别约 0.0344、0.1377、0.4304，相对弗罗贝尼乌斯误差约 0.071007、0.059798、0.044888。误差按未裁剪矩阵计算；四幅图统一显示范围 $[0,1]$ 。

编程思路 在固定网格上生成有纹理、局部亮斑与噪声的灰度教学图，按秩 2、8、25 截断 SVD。比较视觉效果、弗罗贝尼乌斯 误差与保存 U、s、Vh 所需的浮点数个数，并检查误差随秩增大不增。

代码对应

把平滑纹理、局部结构和随机噪声相加 $\rightarrow$ `image = 0.5 + 0.2 * np.sin(8 * x) * np.cos(5 * y)`：这是可离线复现的 100×140 数值矩阵，不依赖外部图片。

用前 k 个奇异三元组重构图像矩阵 $\rightarrow$ `approximation = (U[:, :k] * s[:k]) @ Vh[:k]`：此处先得到未经显示裁切的数值重构，误差以该矩阵计算。

相对误差 $= \|A - A_k\|_F / \|A\|_F \rightarrow$ `error = (np.linalg.norm(image - approximation, "fro") / np.linalg.norm(image, "fro"))`：归一化使不同 k 的重构误差可直接比较。

存储比例 $= k(m+n+1)/(mn) \rightarrow$ `storage = k * (sum(image.shape) + 1) / image.size`：分别统计 U 的 mk 、Vh 的 kn 和奇异值的 k 个数，不包含文件编码开销。

易错提示 显示时把灰度裁到 $[0,1]$ 只是可视化处理，不应偷偷改用裁切后的矩阵计算 SVD 理论误差。此处存储比例不是实际 PNG 文件大小或压缩率。

正文代码 e30_image.py

```
# 教材位置：第 9 章，第 9.7 节；图 9-4。
# 单位、变量与形状：image 为 (100,140)；灰度使用无量纲相对亮度，存储比和相对误差均无量纲。
# 。
# 案例 30：把灰度图看作矩阵，用截断 奇异值分解 比较压缩量与失真。
# 学习目标：理解图像的每个像素是矩阵元素；只保留 k 个奇异分量可减少需存储的标量数。
# 核心关系： $I_k = (U[:, :k] * s[:k]) @ Vh[:k]$ ；相对误差  $= \|I - I_k\|_F / \|I\|_F$ 。
# 对  $m \times n$  图，保存三组因子的标量数为  $mk + k + kn = k(m+n+1)$ ，原图为  $mn$ 。
# 数据与形状：用函数生成  $100 \times 140$  灰度矩阵并添加噪声，不读取外部图片；
# 精简 U、s、Vh 的形状分别为 (100,100)、(100,)、(100,140)。
# 输出：打印  $k=2/8/25$  的标量存储比和相对误差；生成四幅并排比较图 e30_image.png。
```

```
# 运行方式：安装依赖后在本目录执行 python e30_image.py, 图片保存到当前工作目录。

# NumPy 负责网格、模拟灰度图、奇异值分解及误差计算。
import numpy as np
# imshow 把二维数组显示为灰度图；各子图共用颜色范围以便公平比较。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 12 使噪声可复现；这里只影响模拟图像，不改变 奇异值分解 的定义。
rng = np.random.default_rng(12)
# mgrid 切片中的 100j/140j 表示在闭区间上取相应个数的网格点，
# 不是虚数坐标；y、x 都是 (100,140)，依次对应行方向与列方向坐标。
y, x = np.mgrid[-1:1:100j, -1:1:140j]
# 基底亮度 0.5 加上平滑的二维波纹；乘积  $\sin(8x)\cos(5y)$  有可分离的低秩结构。
image = 0.5 + 0.2 * np.sin(8 * x) * np.cos(5 * y)
# 加入中心在 (0.25,-0.1) 的平滑亮斑；系数 25 控制亮斑衰减的快慢。
image += 0.3 * np.exp(-25 * ((x - 0.25) ** 2 + (y + 0.1) ** 2))
# 每个像素叠加标准差 0.035 的高斯噪声，使图像具有许多较小的奇异分量。
image += rng.normal(scale=0.035, size=image.shape)
# 精简 奇异值分解 保留  $\min(100,140)=100$  个奇异值，按降序排列供后续截断。
U, s, Vh = np.linalg.svd(image, full_matrices=False)
# 一行四个子图：原图和三个不同 k 的重建图，方便比较细节损失。
fig, axes = plt.subplots(1, 4, figsize=(6.3, 2.8))
# 灰度颜色固定映射到 0~1；噪声可能使数组少数值超出此范围，
# 显示时颜色会饱和，但这里没有修改或截断用于 奇异值分解 与误差计算的原始数值。
axes[0].imshow(image, cmap="gray", vmin=0, vmax=1)
# 标出含噪声的原始模拟图，作为所有重建的共同参照。
axes[0].set_title("原始模拟图")
# 用正无穷初始化“上一次误差”，使第一次迭代也能统一执行单调性断言。
previous = np.inf
# 逐个将后三个子图与截断秩 2、8、25 配对，k 越大保存的细节和噪声也越多。
for ax, k in zip(axes[1:], [2, 8, 25]):
    # (100,k) 的左因子按列乘奇异值，再乘 (k,140) 的右因子，输出仍为 (100,140)。
    # 这是给定秩上限 k 时的最佳 弗罗贝尼乌斯 近似，并不保证每个像素都更接近无噪声真图。
    approximation = (U[:, :k] * s[:k]) @ Vh[:k]
    # 用整图 弗罗贝尼乌斯 误差除以原图范数，得到不依赖灰度整体缩放的相对误差。
    error = (np.linalg.norm(image - approximation, "fro") /
             np.linalg.norm(image, "fro"))
    # 保存  $U_k$ 、 $s_k$ 、 $Vh_k$  需要  $100k+k+140k$  个标量，除以原图 14000 个标量。
    # 这是相同标量精度下的理论数量比，没有计入文件头、编码、量化等实际存储开销。
    storage = k * (sum(image.shape) + 1) / image.size
    # 同时报告成本与精度，避免只根据重建图片的主观观感选择压缩秩。
    print(f"k={k}: 标量存储比={storage:.4f}, 相对误差={error:.6f}")
    # 本例按递增 k 检查相对误差不增加，且所选三个 k 的因子标量数均少于原图。
    assert error <= previous and storage < 1
    # 核对截断 奇异值分解 的误差=被舍弃奇异值的二范数，验证重建实现与理论一致。
    assert np.isclose(np.linalg.norm(image - approximation, "fro"),
                      np.linalg.norm(s[k:]))
    # 记录本次误差，供下一个更大的 k 检查单调性。
    previous = error
    # 重建图使用与原图完全相同的灰度范围，避免自动调色掩盖亮度误差。
    ax.imshow(approximation, cmap="gray", vmin=0, vmax=1)
    # 在图题中标明截断秩和百分比相对误差，便于图像与数字对应。
    ax.set_title(f"k={k}, 误差={error:.1%}")
# 隐藏像素坐标和边框，让读者集中比较图像内容。
for ax in axes: ax.axis("off")
```



```
# 调整子图间距后保存图片并关闭图对象。
fig.tight_layout()
fig.savefig("e30_image.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

k=2: 标量存储比=0.0344, 相对误差=0.071007

k=8: 标量存储比=0.1377, 相对误差=0.059798

k=25: 标量存储比=0.4304, 相对误差=0.044888

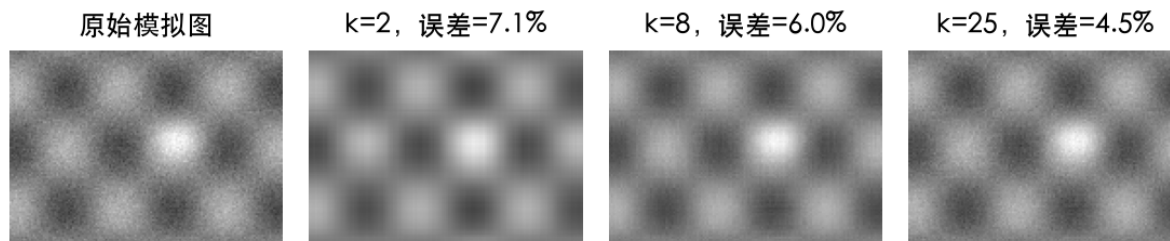


图 9-4 e30 同一 100×140 合成灰度图及 k=2、8、25 的近似。图题中的误差按一位百分数显示，精确核对值分别约 7.10%、5.98%、4.49%；增加 k 保留更多细节，同时增加标量存储。

读图指南 先比较固定亮度范围下原图与不同 k 的细节，再联合读取图题中的误差和存储比。k 增加使当前矩阵重建误差下降，但是否更接近无噪信号须有额外依据。

9.8 伪逆 最小范数与病态正则化

对一般矩阵 A，完整 SVD 中的 Σ 可能为矩形或包含零，不能取普通逆。伪逆 A^+ 定义为 $V\Sigma^+U^T$ ，其中 Σ^+ 为 $n \times m$ 矩阵：只把正奇异值 σ_i 换成 $1/\sigma_i$ ，零仍保持零，再交换形状。上标+是伪逆符号，不表示把负值删掉。它把可辨认输出方向逆向映回输入，对不在列空间的部分作零处理。[MIT-L12]

给定 b ，先分解成列空间投影 p 与左零空间残差 e 。 A^+b 得到 $x^+=\sum_{i=1}^r(u_i^Tb/\sigma_i)v_i$ ，它恰好产生 p 。其他最小二乘解都可写成 x^++z ，其中 $z \in N(A)$ ，因为增加零空间分量不改变输出。 x^+ 属于行空间，与 z 正交，因此 $\|x^++z\|_2^2 = \|x^+\|_2^2 + \|z\|_2^2$ 。由此得到统一结论：伪逆给出所有最小二乘解中唯一的最小二范数解，既包括有解也包括无解情形。

例题 手算：噪声怎样被小奇异值放大

$A=\text{diag}(1,0.01)$ ，真实输入 $x=(1,1)^T$ ，理想输出 $b=(1,0.01)^T$ 。若第二输出仅增加 0.01，观测变成 $\tilde{b}=(1,0.02)^T$ ，直接求逆得到 $(1,2)^T$ ：输出绝对误差 0.01 在该方向被放大 100 倍。

若用阈值 $\tau=0.05$ 截断，只保留奇异值 1，得到 $x_\tau=(1,0)^T$ ；噪声没有被放大，但真实第二分量也丢掉。岭式目标取 $\alpha=0.01$ 时，两系数为 $1/(1.01)\approx 0.990099$ 和 $0.01\times 0.02/(0.0001+0.01)\approx 0.019802$ 。它们仍有偏差，展示的是偏差与不稳定性的交换，不能把截断后的小范数当成真值已恢复的证明。

对于可逆方阵， $A^+=A^{-1}$ ；对于满列秩矩阵， $A^+=(A^T A)^{-1} A^T$ ；对于满行秩矩阵， $A^+=A^T (A A^T)^{-1}$ 。后两个公式各有自己的秩条件，不能因尺寸看似能相乘就省略。投影矩阵 $A A^+$ 把输出投到列空间， $A^+ A$ 把输入投到行空间；它们分别为 $m\times m$ 与 $n\times n$ ，矩形情况下通常不是同一个矩阵。

很小的正奇异值会使 $1/\sigma_i$ 很大，从而放大 b 中微小噪声。截断伪逆只保留 σ_i 大于阈值的方向，主动放弃不可靠信息，以较大偏差换取较小噪声放大。这属于正则化的思想，不等于恢复了被丢掉的真实信息。截断阈值应结合误差预算、数据噪声或独立验证确定；为展示稳定而任意删项，可能同时删掉重要信号。

也可用岭式目标 $\|Ax-b\|_2^2 + \alpha \|x\|_2^2$ ， $\alpha>0$ 控制惩罚强度。沿奇异方向的系数不再乘 $1/\sigma_i$ ，而乘 $\sigma_i/(\sigma_i^2+\alpha)$ ，所以小奇异方向被平滑压低。这里的 α 是模型选择参数，大小与数据尺度相关；它不是让原方程突然精确可解的魔法常数。这一联系说明奇异值结构既能诊断病态，也能为控制噪声放大提供明确依据。

岭式滤波因子也可以直接推出。把 x 写成 $\sum c_i v_i$ ，目标分成 $\sum_{i=1}^r (\sigma_i c_i - u_i^T b)^2$ 加上与 x 无关的左核残差平方，再加 $\alpha \sum c_i^2$ 。每个 c_i 的导数为 $2(\sigma_i^2+\alpha)c_i - 2\sigma_i u_i^T b$ ，故最优 $c_i = \sigma_i u_i^T b / (\sigma_i^2 + \alpha)$ ；核方向因只受 αc_i^2 惩罚而取零。 $\alpha>0$ 使所有方向目标严格凸，解唯一。

$$A^+ = V \Sigma^+ U^T, \quad x^+ = A^+ b = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} v_i \quad (9.9)$$

A^+ 为 $n \times m$, b 为 m 维列, x^+ 为 n 维最小范数最小二乘解; 仅对正奇异值求倒数。零空间分量无法由 b 识别, 因此最小范数规则把它设为零。

9.9 主成分分析与线性组合方差

第 7 章已把线性组合方差写成 $a^T S a$ 。主成分分析 (PCA) 进一步问: 在长度为 1 的变量权重中, 哪种组合保留最多样本变化? 它首先是描述数据和降维的方法。总体参数估计、置信区间、预测验证或因果识别都需要额外的抽样与建模假设, 不能由一次 PCA 图自动获得。样本协方差的定义见式 (7.8)。

主成分分析 (PCA) 处理 N 个样本、 d 个变量的数据表 $X \in \mathbb{R}^{N \times d}$, 约定每行是样本, 每列是变量。先算列均值 $\mu_j = (1/N) \sum_i X_{ij}$, 并中心化 $X_c = X - 1\mu^T$ 。这里 μ 为 d 维均值列, 1 为 N 维全 1 列, $1\mu^T$ 把均值复制到各行。中心化使各变量均值为零, 保留的是围绕平均状态的变化; 不中心化得到的主要方向可能主要指向均值位置。

对单位变量方向 $v \in \mathbb{R}^d$, 将每个样本投影到该方向, 得到得分 $z = X_c v \in \mathbb{R}^N$ 。样本方差为 $z^T z / (N-1) = v^T C v$, 其中 $C = X_c^T X_c / (N-1)$ 为样本协方差矩阵, 要求 $N \geq 2$ 。第一主方向是在 $\|v\|_2 = 1$ 约束下使投影方差最大的 v 。后续方向还要与前面方向正交, 依次保留剩余变化。主方向位于变量空间, 得分则是一列样本坐标, 它们不是同一对象。

因为 C 对称半正定, 写 $C = V \Lambda V^T$ 。把单位 v 展开为 $v = \sum_j c_j v_j$, 有 $\sum_j c_j^2 = 1$ 、 $v^T C v = \sum_j \lambda_j c_j^2$, 所以最大值为最大特征值, 方向为其特征向量。这是瑞利商最优性的直接证明。对后续方向约束 $c_1 = 0$, 再选次大特征值即可。若首两特征值相等, 第一方向在对应平面中不唯一, 但保留整个重特征子空间有明确意义。

$$C = \frac{X_c^T X_c}{N-1}, \quad \max_{\|v\|_2=1} v^T C v = \lambda_1 \quad (9.10)$$

X_c 为中心化 $N \times d$ 数据, C 为 $d \times d$ 样本协方差矩阵, v 为 d 维单位列; λ_1 表示 C 的最大特征值, 此节按从大到小排列。

若 $Xc=U\Sigma V^T$ ，则 $C=V[\Sigma^T\Sigma/(N-1)]V^T$ ，因此 PCA 方向就是右奇异向量，协方差特征值 $\lambda_j=\sigma_j^2/(N-1)$ 。所有得分组成 $Z=XcV=U\Sigma$ 。第 j 个方向解释率为 $\lambda_j/\sum \ell \lambda \ell$ ，也等于 $\sigma_j^2/\sum \ell \sigma \ell^2$ ；分母为总样本方差，要求数据存在非零变化。解释率是一组分配总变化的比例，不是预测准确率，也不是因果贡献率。

保留前 k 个方向 V_k 时，得分 $Z_k=XcV_k$ ，中心化重建 $\hat{X}_{\text{hatc}}=Z_kV_k^T$ ，再加回均值。每个样本都被正交投影到同一个 k 维变量子空间。它在所有 k 维线性子空间中最小化样本总重建平方误差，与截断 SVD 低秩近似一致：最大化保留方差，等价于最小化丢失方差，因为中心化总平方能量固定。

中心化与标准化不同。标准化还把第 j 列除以该列标准差 s_j ，使变量统一成无量纲尺度，对应分析相关矩阵。如果一个变量单位从米改为毫米，未经标准化的方差会扩大 10^6 倍，主方向可能明显改变。是否标准化要依据问题：同单位且方差大小本来有意义的数据，保留原尺度可能合理；不同量纲且希望公平比较时，标准化常有帮助。不能机械地说 PCA 永远必须或永远不应标准化。

PCA 轴的符号任意：把 v_j 和对应得分列同时乘 -1 ，重建和解释率不变。轴方向也只说明样本中最强的线性变化，不保证发现独立来源；不同主成分得分样本协方差为零，通常不能由此推出统计独立。非线性结构、异常值以及样本量不足也可能影响解释，因此仍需看原变量、散点图和问题背景。

例题 手算：协方差方向与投影

取协方差矩阵 $C=[[4,2],[2,4]]$ 。特征方向 $v_1=(1,1)^T/\sqrt{2}$ 、 $v_2=(1,-1)^T/\sqrt{2}$ ，对应特征值 6 与 2。因此第一方向解释 75% 样本总方差，第二方向解释 25%。一个中心化样本 $x=(3,1)^T$ 的第一得分为 $v_1^T x=2\sqrt{2}$ ，沿第一方向重建为 $(2,2)^T$ 。

残差为 $(1,-1)^T$ ，恰沿第二主方向。单个样本的误差不要求等于总方差的 25%；解释率汇总全部样本的平方变化，不能逐点套用。若两变量量纲改变，新的协方差矩阵和 PCA 方向需要重新计算。

程序 e31_pca.py 固定种子 5, N=300。base 服从均值 0.5、标准差 0.18 的正态分布；第一列为 base 加标准差 0.05 噪声，第二列为 0.8base+0.25 加标准差 0.06 噪声。X 为 300×2。本例只中心化，不标准化。C=Xc.T@Xc/299，主方向 V=Vh.T，scores=Xc@V，解释率约 [0.940281,0.059719]。程序比较协方差特征分解与 SVD 的同一主子空间，不能因轴符号相反判定错误。

编程思路 对具有同类量纲的二维模拟点云先逐列中心化，计算样本协方差，再以特征分解和中心化数据的 SVD 两路求主方向。把投影得分、解释率以及得分协方差的对角化相互核对。

代码对应

每个特征减去该列样本均值 $\rightarrow X_c = X - X.\text{mean}(\text{axis}=0)$ ：mean(axis=0) 沿观测行求每列均值，本例仅中心化而未按标准差缩放。

样本协方差 $C=X_c.T@X_c/(N-1) \rightarrow C = X_c.T @ X_c / (N - 1)$ ：中心化矩阵形状 N×2，转置相乘得到 2×2 协方差矩阵。

主成分坐标=中心化观测×右奇异方向 $\rightarrow \text{scores} = X_c @ V$ ：V 的每列是一条原特征空间方向，scores 的列是对应投影得分。

解释率= $s_j^2/\sum s^2 \rightarrow \text{ratio} = s ** 2 / (s @ s)$ ：特征值等于 $s_j^2/(N-1)$ ，所以归一化时公共分母抵消。

易错提示 特征向量与奇异向量可能整体反号，应比较方向子空间或外积，不能逐项要求符号相同。若特征单位明显不同，需先决定是否标准化；另一个用户画像案例专门处理该情形。

正文代码 e31_pca.py

```
# 教材位置：第 9 章，第 9.9 节；图 9-5。
# 单位、变量与形状：X、Xc、scores 为 (300,2)；特征单位统一，得分沿用该单位，方差比例无量纲。
# 案例 31：从中心化、样本协方差和 奇异值分解 三个角度理解 主成分分析。
# 学习目标：提取二维相关数据中方差最大的正交方向，理解主成分得分与解释方差比例。
# 核心关系：Xc=X-列均值，C=Xc.T@Xc/(N-1)；若 Xc=USV.T，
```

```
# 则  $C = V \text{diag}(s^2 / (N-1)) V^T$ ,  $\text{scores} = XcV = US$ , 不同得分列之间的样本协方差为零。
# 数据与形状:  $N=300$  个样本、2 个特征;  $X/Xc/\text{scores}$  均为  $(300, 2)$ ,  $C$  为  $(2, 2)$ 。
# 本例只中心化, 没有按标准差标准化; 主成分分析 方向因此依赖原特征的量纲与尺度。
# 输出: 打印两个主成分的方差比例, 验证等价关系, 生成 e31_pca.png。
# 运行方式: 安装依赖后在本目录执行 python e31_pca.py, 图片保存到当前工作目录。

# NumPy 用于模拟相关变量、样本协方差、对称特征分解与 奇异值分解。
import numpy as np

# 左图展示中心化数据和主轴, 中图展示旋转到主轴坐标系后的得分, 右图展示解释率。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 5, 便于复现相同的点云和解释方差比例。
rng = np.random.default_rng(5)
# 生成 300 个共同潜在值, 均值 0.5、标准差 0.18; 两个特征共用它来制造相关性。
base = rng.normal(0.5, 0.18, 300)
# 第一特征为 base 加标准差 0.05 的独立噪声, 第二特征为  $0.8 * \text{base} + 0.25$  加噪声。
# column_stack 把两个长度 300 的数组按列拼成  $(300, 2)$ : 行是样本, 列是特征。
X = np.column_stack([base + rng.normal(0, 0.05, 300),
                     0.8 * base + 0.25 + rng.normal(0, 0.06, 300)])
# axis=0 沿样本行求各列均值, 形状  $(2,)$  自动广播到 300 行。
# 先移除均值才能让 主成分分析 描述围绕样本中心的变化; 未中心化的 奇异值分解 会混入均值位置。
Xc = X - X.mean(axis=0)
N = len(X)
#  $(2, 300) @ (300, 2)$  得到  $2 \times 2$  散布矩阵; 除以  $N-1$  采用样本协方差口径。
# 对角线是各特征样本方差, 非对角线是两特征样本协方差。
C = Xc.T @ Xc / (N - 1)
# C 为实对称矩阵, eigh 返回升序特征值, directions 的各列为相应的标准正交方向。
values, directions = np.linalg.eigh(C)
# 同时反转特征值和特征向量列, 使第一列对应最大方差方向; 保持配对关系不能只反转一方。
values = values[::-1]
directions = directions[:, ::-1]
# 中心化矩阵的精简 奇异值分解 形状为  $U(300, 2)$ 、 $s(2,)$ 、 $Vh(2, 2)$ , 奇异值已按降序排列。
U, s, Vh = np.linalg.svd(Xc, full_matrices=False)
#  $V$  的列为特征空间里的主轴;  $Xc @ V$  把每个样本投影到这些轴上, 得到主成分得分。
# 每轴方差为  $s_j^2 / (N-1)$ , 求比例时公因子  $N-1$  抵消, 因此  $\text{ratio} = s^2 / \sum s^2$ 。
V = Vh.T
scores = Xc @ V
ratio = s ** 2 / (s @ s)
# 两个比例之和为 1, 表示保留各主成分能解释多少中心化数据的总样本方差。
print("主成分方差比例 =", np.round(ratio, 6))
# 核对协方差特征值与中心化数据奇异值平方之间的关系, 分母保持  $N-1$  一致。
assert np.allclose(values, s ** 2 / (N - 1))
# 得分均值仍为零, 因此该式正是得分样本协方差矩阵。
# 非对角元接近零说明主成分线性不相关; 这通常不意味着统计独立。
assert np.allclose(scores.T @ scores / (N - 1), np.diag(values))
# 本例存在明显的最大方差主方向; 特征值不等也使该方向除正负号外唯一。
assert values[0] > values[1]
# 特征分解与 奇异值分解 可能为同一条主轴返回相反方向, 直接比较向量会误判。
# 比较  $vv^T$  投影矩阵可以消除符号差异, 因为  $(-v)(-v)^T = vv^T$ 。
assert np.allclose(np.outer(directions[:, 0], directions[:, 0]),
                    np.outer(V[:, 0], V[:, 0]))
#  $XcV=US$ ;  $U*s$  用广播把每列  $U$  乘以对应奇异值, 形状仍为  $(300, 2)$ 。
assert np.allclose(scores, U * s)
# 创建中心化原坐标、主成分得分与解释率三幅图; 全都使用上面同一批 300 个样本。
```

```
fig, axes = plt.subplots(1, 3, figsize=(6.3, 3.0))
# Xc.T 的两行被展开为散点横纵坐标；小点和半透明可减轻重叠遮挡。
axes[0].scatter(*Xc.T, s=8, alpha=0.4)
# 分别绘制 PC1 与 PC2；两条方向线应正交。
for j in range(2):
    # 特征值是该轴方差，其平方根为标准差；取两倍标准差作为每一侧示意轴长度。
    v = 2 * np.sqrt(values[j]) * V[:, j]
    # 从 -v 画到 +v，线段经过中心化后的原点；它只是主轴示意，并非置信区间。
    axes[0].plot([-v[0], v[0]], [-v[1], v[1]], label=f"主成分 {j + 1}")
# 左图横纵单位等长，保证图上的主轴角度忠实反映几何正交关系。
axes[0].set_aspect("equal")
axes[0].legend(fontsize=7.8)
# 左图数据中心在原点，展示中心化后的原特征坐标。
axes[0].set(xlabel="中心化特征 1", ylabel="中心化特征 2", title="中心化数据")
# 中图以 PC1、PC2 得分为横纵坐标，点云的主要变化沿 PC1 方向展开。
axes[1].scatter(*scores.T, s=8, alpha=0.4)
axes[1].set(xlabel="主成分 1 得分", ylabel="主成分 2 得分", title="主成分得分")
# 得分图也采用等比例坐标，避免纵横自动缩放改变读者对主次变化的观感。
axes[1].set_aspect("equal")
# 右图直接读取已算出的 ratio；不重新生成点云或用另一套特征值。
axes[2].bar([1, 2], ratio, width=0.55, color=["tab:blue", "tab:orange"])
axes[2].set_xticks([1, 2], ["主成分 1", "主成分 2"])
axes[2].set(ylabel="方差比例", ylim=(0, 1.1), title="解释方差比例")
for index, value in enumerate(ratio, 1):
    axes[2].text(index, value + 0.035, f"{value:.2%}", ha="center", fontsize=8)
# 为各幅图添加淡网格，帮助比较点云位置、主轴方向与方差份额。
for ax in axes:
    ax.grid(alpha=0.3)
    ax.tick_params(labelsize=7)
# 调整布局、保存图片并关闭图对象，以支持脚本批量执行。
fig.tight_layout()
fig.savefig("e31_pca.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

主成分方差比例 = [0.940281 0.059719]

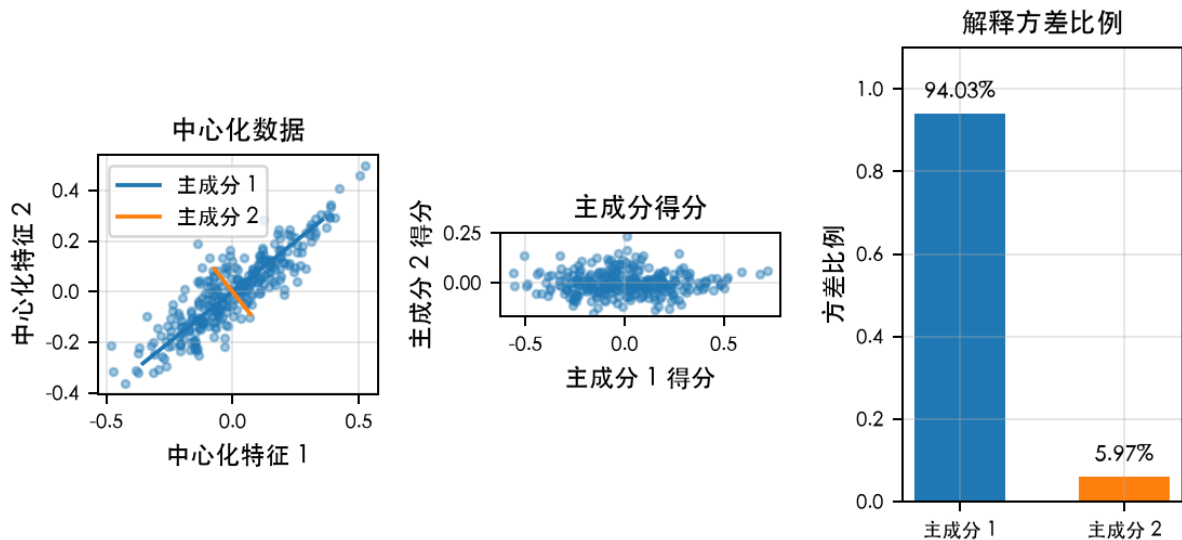


图 9-5 e31 的同一批 300 个样本。左：中心化点云及两个主方向；中：得分坐标 $\mathbf{Xc@V}$ ；右：奇异值平方得到的解释率，约 94.03%与 5.97%。两幅散点图等比例显示，轴符号翻转不改变保留子空间。

读图指南 左图在原变量中心化坐标中看点云与主方向，中图读同一批点在主轴上的得分，右图看两方向解释率。散点轴等比例，符号翻转不改变重建或解释率。

用 e38 的 $S = \begin{bmatrix} 5/3, 5/3 \\ 5/3, 10/3 \end{bmatrix}$ 核对这一联系，总样本方差 $\text{tr}(S)=5$ ，特征值为 $(15 \pm 5\sqrt{5})/6$ ，约 4.36338998 与 0.63661002，解释率约 87.2678%与 12.7322%。四条观测中心化矩阵的奇异值平方等于 3 倍这些特征值。这里没有新增数据，也没有把原始两变量改成独立随机源；只是换一组正交坐标描述同一份样本变化。

9.10 同一结构中的不同任务

当任务是求解，重点看小奇异值是否放大噪声；当任务是压缩，重点看尾部平方能量是否足够小；当任务是 PCA，重点看中心化后变量方向的方差及重建；当任务是求空间基，重点看零与非零方向的分界。这些任务共用 SVD，却不应共享一个未经说明的“最佳 k”。例如只占很小方差的方向，可能恰好包含识别罕见异常所需的信号；压缩时愿意舍弃它，不代表诊断时也该舍弃。

每次解释结果，可以依次报出矩阵尺寸、预处理、所用分解形式、保留或截断规则、误差口径，以及返回原问题后的含义。对矩形变换、低秩近似、图像压缩与主成分分析，分别

核对矩形几何、尾谱误差、标量存储和样本方差。这种核验方式比孤立记住一个函数名更可靠，因为即使换了编程语言，数学对象和判断标准仍然保持。

9.11 练习与参考解答

练习 9-1（概念与手算，入门） A 为 7×4 、秩 3。写出完整 SVD、 q 项约化形式的尺寸与四子空间维数。

练习 9-2（概念与手算，入门） 奇异值 5、2、1，保留第一层时，二范数误差、弗罗贝尼乌斯误差、保留能量分别多少？

练习 9-3（概念与手算，进阶） $A = \text{diag}(4, 0)$ 、 $b = (8, 3)^T$ 。求伪逆、最小范数最小二乘解和全部最小二乘解。

练习 9-4（概念与手算，进阶） 中心化样本数 $N=101$ ，样本协方差特征值 9、3、0。求 X_c 的奇异值及解释率。

练习 9-5（编程实践，入门） 复现 e39 二维 $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ ，核对奇异值平方、乘积与三阶段重建。

练习 9-6（编程实践，进阶） 运行 e29_lowrank.py，分别用尾谱平方和与直接矩阵差验收 $k=2$ 、4、10 的误差。

练习 9-7（编程实践，进阶） 在 e31_pca.py 中验证协方差特征分解与 SVD 对应。给出不依赖单轴符号的核验方法。

练习 9-8（编程实践，拓展） 编程比较 $A = \text{diag}(1, 0.1)$ 、观测 $b = [1, 0.2]$ 的直接解、阈值 0.05 截断解与 $\alpha = 0.1$ 岭解。

练习 9-9（场景应用，入门） 一幅 80×120 灰度矩阵以秩 10 保存 U 、 s 、 V 。理论标量存储比是多少？可否直接当成实际压缩文件大小比？

练习 9-10（场景应用，进阶） 两变量样本协方差 $S = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$ 。若第一变量从米换成毫米，PCA 会保持方向吗？若先按各自样本标准差标准化，相关矩阵和解释率为何？

练习 9-11（场景应用，拓展） 四个明确模拟的样本为 $(-10,-1)$ 、 $(10,-1)$ 、 $(-10,1)$ 、 $(10,1)$ ，标签只由第二坐标正负决定。保留最大方差的一维 PCA 能保留分类信息吗？

练习 9-12（场景应用，进阶） 模拟传感器只报告两个分量之和： $A=[1,1]$ 、 $b=2$ 。伪逆给出的两个分量是什么？若外部物理信息要求第二分量为 0，应怎样解释不同答案？

参考解答与验收依据

9-1 完整 U 7×7 、 Σ 7×4 、 V 4×4 ； $q=\min(7,4)=4$ ， U_q 7×4 、 Σ_q 4×4 、 V_q 4×4 。列空间 3、行空间 3、核 1、左核 4。约化 U_q 没有保留全部左核基，且列数 4 不等于秩 3。

9-2 依次为 2、 $\sqrt{5}$ 、 $25/30=5/6$ 。相对弗罗贝尼乌斯误差为 $\sqrt{(1/6)} \approx .408248$ ，不是 $1/6$ ；保留能量是平方量的比例。

9-3 $A^+=\text{diag}(1/4,0)$ ， $x^+=(2,0)^T$ ；拟合 $(8,0)^T$ ，残差 $(0,3)^T$ 。全部最小二乘解为 $(2,t)^T$ ， t 任意；范数平方 $4+t^2$ 在 $t=0$ 唯一最小。

9-4 $\sigma_j=\sqrt{(N-1)\lambda_j}$ ，故奇异值 30、 $10\sqrt{3}$ 、0；解释率 75%、25%、0。变化位于二维子空间，第三变量仍存在，只是这些变量有一条中心化线性约束。

9-5 s^2 应约 $[5.23606798, .76393202]=3 \pm \sqrt{5}$ ； s 的乘积 $2=|\det A|$ 。用 $U @ \text{np.diag}(s) @ V^H$ 重建 A ，逐阶段终点与 $A @ \text{points}$ 近似相同。 U 、 $V^H.T$ 均正交；不要求奇异向量的符号与另一台机器相同。

9-6 对同一 60×80 观测矩阵，应分别约 87.947679、16.560696、13.866765； $\text{np.linalg.norm}(A-A_k, \text{"fro"})$ 应等于 $\text{np.linalg.norm}(s[k:])$ 。前 4 项能量约 .987269。后一公式是尾部奇异值的欧氏范数，不能只取下一奇异值代替总误差。

9-7 X_c 形状 300×2 ， $C=X_c.T @ X_c/299$ 。将 eigh 所得特征值降序后，与 $s^2/299$ 比较；解释率应约 $[.940281, .059719]$ 。分别取协方差特征法与 SVD 的第一主方向为一维数组 v_{cov} 、 v_{svd} ，用 $\text{np.outer}(v_{\text{cov}}, v_{\text{cov}})$ 和 $\text{np.outer}(v_{\text{svd}}, v_{\text{svd}})$ 形成两个 2×2 投影矩阵，二者应在浮点容差内相同。也可把方向保留成二维列 $V[:, [0]]$ 后计算 $V[:, [0]] @ V[:, [0]].T$ 。对重特征值应比

较整个相应子空间。完整正交基满足 $Xc @ V @ V.T \approx Xc$ 只验证重建恒等式，不能单独验证两算法主方向对应。

9-8 直接解 [1,2]；截断解 [1,0]；岭解约 [.990099,.019802]。可由 `solve(A.T @ A + alpha * I, A.T @ b)` 核对岭式滤波结果。若真实输入 [1,1]，截断与岭都存在偏差，不能以范数更小就断言更接近真值。

9-9 保存 $10(80+120+1)=2010$ 个实数，原矩阵 9600 个，比为 .209375。它未计浮点精度、元数据、编码与文件头，不等于实际图像文件体积比。应同时报告重建误差和显示范围，避免只看一幅图。

9-10 原尺度协方差变为 $[[4 \times 10^6, 2000], [2000, 4]]$ ，未经标准化 PCA 一般改变。标准化后相关矩阵仍为 $[[1, .5], [.5, 1]]$ ，特征值 1.5、.5，解释率 75%、25%；这表示分析目标已改为标准差单位下的相对变化，是否适合须由问题决定。

9-11 两列已中心化且协方差为 0，样本方差分别 400/3 和 .04/3。第一 PCA 方向为横轴，解释率 $10000/10001 \approx 99.9900\%$ ，却把两类投影到相同的 -10 和 10，丢失全部标签区分。说明最大解释率只优化无监督平方重建，不保证分类或因果目标。

9-12 伪逆为 $(1/2, 1/2)^T$ ，最小范数解 [1,1]；所有精确解为 $[1+t, 1-t]$ 。物理约束 $x_2=0$ 给 [2,0]，仍满足观测。伪逆按最小范数规则选解，并不替代物理知识；有依据的约束可选择另一解。

9.12 核心要点与常见误区

核心要点：SVD 为输入和输出分别选择标准正交基，中间用非负矩形对角 Σ 连接两侧；正交可含反射。实对称谱定理给出 V ，正值方向定义 u_i ，正交补扩充得到完整 U 。

核心要点：截断 SVD 的低秩最优性来自投影误差下界和有限能量分配界。伪逆消除不可识别核分量，给唯一最小范数最小二乘解；截断和岭式正则化控制小奇异方向的噪声放大。

常见误区： q 项约化 SVD 不一定只含正值方向。保留能量比例与相对范数误差不是同一个百分比。PCA 最大化样本方差，不自动保证预测、独立来源、统计推断或因果解释；是否标准化须符合变量单位和目标。

9.13 延伸阅读 入门 进阶与拓展

入门阅读：看第 29 讲“奇异值分解”与 Ses3.5sum，先手算二维，再核对矩形因子尺寸（来源 MIT-L10；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/positive-definite-matrices-and-applications/singular-value-decomposition/>）。

进阶阅读：读第 31 讲“换基与图像压缩”的 Ses3.7sum，联合报告秩、尾谱误差与标量存储（来源 MIT-L11；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/positive-definite-matrices-and-applications/change-of-basis-image-compression/>）。

拓展阅读：读第 33 讲“左逆、右逆与伪逆”的 Ses3.8sum，用核与行空间重做最小范数证明；PCA 同时回顾本书第 7 章协方差二次型（来源 MIT-L12；公开链接：<https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/pages/positive-definite-matrices-and-applications/left-and-right-inverses-pseudoinverse/>）。

第 10 章 用矩阵工具完成概率统计分析

10.1 分层学习路线

入门：先说明一行数据和一个状态表示什么，再手算一次加权平均或一步转移。进阶：完成浓度、计数、土地、留存、用户画像与仪器标定六个案例，解释每个矩阵运算服务的统计问题。拓展：比较模型假设、参数不确定性、标准化和外推风险，能够说明计算正确与结论可靠之间还缺少哪些证据。

10.2 从研究问题确定计算对象

同一份数据可以计算很多数字，但有用的分析应从一个明确问题开始。研究浓度时，可能关心超过某个水平的概率；研究计数时，可能关心怎样设置复核门槛；研究状态转移时，可能关心若规则保持不变，未来各类的期望比例是多少。问题决定需要计算的量，也决定哪些假设必须检查。

先说明一行数据代表什么。浓度案例的一行是一份读数，计数案例的一行是相同观察长度与暴露量下的事件数，土地利用案例的一个对象是等面积单元。连续读数、整数计数和类别状态的取值空间不同，不能只因为直方图形状相似就套用同一个分布。

还应区分三层不确定性。有限样本造成参数估计的抽样波动；未来事件即使在已知参数下仍有过程波动；转移规则或分布假设不合适会造成模型误差。增加模拟次数可以减小模拟计算自身的波动，却不能自动消除观测偏差或模型误差。下列案例依次展示这些区别。

除 Norris 标定案例外，本章其余数据均为教学模拟或假设模型。浓度单位为微克每立方米，75 是案例自定的演示门槛；计数中的基准率和误报要求为教学设定；土地利用转移矩阵没有对应具体地区。分析的目的在于学会提出可计算的问题和解释模型输出。

10.3 浓度数据为什么要考虑分组

设 X 是一份严格为正的浓度读数。有些读数离中心很远，右侧尾部较长，正态分布可能把一部分概率放到不可能的负数上。一个可以检查的候选模型是对数正态：先对读数的数值

取自然对数，若所得变量近似正态，则原变量近似对数正态。这里取对数针对按规定单位记录的数值；换单位会改变对数位置参数。

$$\ln X \sim N(\mu, \sigma^2), \quad f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right], \quad x > 0. \quad (10.1)$$

μ 是对数数值的均值， $\sigma > 0$ 是对数数值的标准差。密度中的 $1/x$ 来自变换 $y = \ln x$ 的导数。它保证在横轴由 y 换成 x 时，同一事件的面积不变。

在 SciPy 中，lognorm 的形状参数 s 等于 σ ，scale 等于 $\exp(\mu)$ 。scale 是中位数，不是均值；均值为 $\exp(\mu + \sigma^2/2)$ 。本例固定 loc=0，表示没有再加一个位置平移。右偏只说明应比较正值候选模型，不能单凭右偏认定数据服从对数正态。

例题 一个正值模型的手算检查

若 $\mu = \ln 32$ 、 $\sigma = 0.22$ ，中位数为 32，均值为 $32 \times \exp(0.22^2/2)$ ，约等于 32.78。均值略大于中位数，是右尾拉高平均值的结果。超过 75 的概率可以先变成 $P(Z > (\ln 75 - \ln 32)/0.22)$ ，再用标准正态右尾计算。

程序 e33_pm25.py 生成三组读数，样本量依次为 120、125、120，中位数依次为 55、32、22，对数标准差依次为 0.25、0.22、0.20，种子为 2024。分组在生成数据时已经确定。合并数据以后，整体密度是组密度的加权平均；每组都是对数正态，并不意味着合并后仍是同一个对数正态。

$$f(x) = \sum_{g=1}^3 w_g f_g(x), \quad w_g = \frac{n_g}{n_1 + n_2 + n_3}, \quad \sum_{g=1}^3 w_g = 1. \quad (10.2)$$

g 是组编号，带下标 g 的 n 、 f 、 w 分别表示该组的观测数、密度与权重。随机抽取一个观测时，先以概率 w_g 抽到第 g 组，再按该组分布取得数值；对全部可能组相加就得到整体密度。

程序同时拟合单一对数正态和已知分组下的加权模型。检查整体拟合时，用 KS 统计量比较经验累积分布与拟合累积分布。因为参数由这批数据估计，参考模拟也必须重复估参步骤；若把估计参数当作事先已知参数，再套用固定参数的普通 KS 参考分布，检验校准就会失准。

$$D = \sup_x |F_n(x) - F_{\hat{\theta}}(x)|, \quad \hat{p} = \frac{b+1}{B+1}. \quad (10.3)$$

F_n 是经验累积分布, $F_{\hat{\theta}}$ 是拟合累积分布, $\sup$ 表示取上确界。B 是参考模拟次数, b 是模拟统计量不小于观测统计量的次数。加 1 的写法使有限模拟不会报告零概率。本例 B=499, 因此最小可报告数值为 0.002。

程序 e33_pm25.py 运行时先观察三组数据的合并直方图, 再比较单一模型与分组模型。右图只画第一组对数读数的正态概率图, 用来检查组内形状。注意整体模型的检验与单个组的图形回答不同问题。完整代码与附件同名文件见下方。

编程思路 按三个已知教学分组模拟正值浓度, 比较全体单一对数正态拟合与按标签拟合的混合密度。把超阈值期望次数与观测次数对照, 并用蒙特卡洛校准检验说明混合总体未必适合单一分布。

代码对应

合并规模 120、125、120 的三组模拟浓度 $\rightarrow$ pm =
`rng.lognormal(np.log(np.array([55, 32, 22])[group])),`
`np.array([0.25, 0.22, 0.20])[group]):` 合并后共有 365 个观测,
 分组标签保留在 group 中用于条件模型。

分别拟合每个已知组的对数正态分布 $\rightarrow$ parts =
`[stats.lognorm(*stats.lognorm.fit(pm[group == j], floc=0))`
`for j in range(3)]:` 各组拟合固定 loc=0; 这利用已知标签, 不是从无标签数据
 自动识别混合成分。

组权重=各组观测数/365 $\rightarrow$ weights = `np.array([(group == j).mean() for j in`
`range(3)])`: 混合密度与尾概率均按相同权重相加。

$P(X>75)=\sum w_j S_j(75) \rightarrow$ tail = `sum(w * d.sf(limit) for w, d in zip(weights,`
`parts))`: 用各组生存函数求严格右尾概率, 再乘样本量得到模型期望超阈次数。

易错提示 本例浓度是教学模拟, 不能当作真实监测结论。全体模型拟合不佳与分组混合的解释相关, 但不能仅凭显著性检验确定真实机制或把期望超标次数当作实际计数。

正文代码 e33_pm25.py

```
# 教材位置: 第 10 章, 第 10.3 节; 图 10-1。
# 单位、变量与形状: pm、group 为 (365,); pm 按  $\mu\text{g}/\text{m}^3$  计, 图中概率密度按  $(\mu\text{g}/\text{m}^3)^{-1}$  计。
# 案例 33: 模拟 PM2.5 浓度的分组对数正态分布与混合模型。
# 学习目标: 认识偏态、分组异质性和尾部概率, 并区分经验超阈值次数与模型期望次数。
# 本例全部数据由随机数模拟, 不代表真实监测结论; 75 只是教学阈值, 不是法律标准。
# 核心关系: 若  $\log(Y) \sim N(\mu, \sigma^2)$ , 则 Y 服从对数正态; 混合生存函数为  $\sum w_j S_j(Y)$ 。
# 数据: 三组分别有 120、125、120 个值, 共 365 个; 组内模拟中位数为 55、32、22。
# 输出: 单一对数正态模型经模拟校准的 拟合优度检验、均值/中位数、超阈值计数,
# 并生成 application_pm.png, 比较整体拟合、分组混合密度与组 0 的对数 分位数对比 图。
# 运行方式: 安装本目录依赖后执行 python e33_pm25.py; 拟合优度模拟可能需要一些时间。
# 所有图保存到当前工作目录; 固定随机种子便于在相同环境下复现模拟结果。

# NumPy 负责生成模拟分组与随机浓度, 并做数组级统计。
import numpy as np
# SciPy 提供对数正态拟合、分布尾概率、拟合优度模拟校准与 分位数对比 图。
from scipy import stats
# Matplotlib 用于绘制密度比较图和 分位数对比 图。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 同一个随机数生成器依次用于生成数据和检验模拟; 固定种子使整个流程可复现。
rng = np.random.default_rng(2024)
# 构造长度 365 的组标签, 0/1/2 分别重复 120/125/120 次。
# 分组可理解为三种教学情境: 没有真实日期信息, 不应直接据此解释季节变化。
group = np.repeat(np.arange(3), [120, 125, 120])
# 按每个样本的组标签选取参数; lognormal 的 mean、sigma 指对数尺度的  $\mu$ 、 $\sigma$ 。
# 因此 55、32、22 是各组在原始浓度尺度的中位数, 而不是算术均值;
# 算术均值为  $\exp(\mu + \sigma^2/2)$ , 通常高于中位数, 体现右偏分布。
pm = rng.lognormal(np.log(np.array([55, 32, 22])[group]),
                  np.array([0.25, 0.22, 0.20])[group])
# 对数正态样本应是有限正数; 检查正值也保证后面 np.log(pm) 有定义。
assert np.all(np.isfinite(pm)) and np.all(pm > 0)
# 先对全部 365 个值拟合一个对数正态, 再把参数传回构造冻结分布对象。
# floc=0 固定位置参数为零; fit 返回 (shape, loc, scale), 其中 shape= $\sigma$ 、scale= $\exp(\mu)$ 。
# 把异质分组合并拟合只是一个对照模型, 并不保证合并分布仍服从单一对数正态。
whole = stats.lognorm(*stats.lognorm.fit(pm, floc=0))
# 按已知组标签分别拟合 3 个对数正态模型; 这是利用分组信息, 不是无标签混合模型的自动识别
# 。
parts = [stats.lognorm(*stats.lognorm.fit(pm[group == j], floc=0))
         for j in range(3)]
# 布尔数组的均值等于该组样本比例, 得到 120/365、125/365、120/365, 权重和为 1。
weights = np.array([(group == j).mean() for j in range(3)])
# 检验原假设是“总体样本可由一个 loc=0 的对数正态描述”, 而不是检验分组混合模型。
# 拟合优度检验 统计量比较经验 累积分布函数 与拟合 累积分布函数 的最大差; 参数由同一数据
# 估计时,
# 不能直接把普通已知参数 拟合优度检验 的参考分布当作正确校准。
# goodness_of_fit 进行 499 次蒙特卡洛/参数自助模拟, 对每份模拟样本重新估计未知参数,
# 同时始终把 loc 固定为 0; 返回经这种过程校准的模拟 p 值。模拟次数有限, p 值有随机误差。
test = stats.goodness_of_fit(stats.lognorm, pm, known_params={"loc": 0},
                             statistic="ks", n_mc_samples=499, rng=rng)
# 仅设定课堂示例的浓度阈值 75.0; 此数值不用于声称满足任何地区或时间口径的标准。
limit = 75.0 # 教学演示阈值, 并非法定限值。
# sf(limit)=P(Y>limit) 为生存函数, 直接计算通常比 1-cdf 更稳定。
# 按组权重求和得到混合模型的总体超阈值概率; 连续分布中 > 与 ≥ 的概率相同。
```

```

tail = sum(w * d.sf(limit) for w, d in zip(weights, parts))
# D 越大表示偏离拟合 累积分布函数 越大; p 值衡量该单一分布假设下统计量的尾部程度, 不是原
# 假设为真的概率。
print(f"拟合优度统计量 D={test.statistic:.4f}, 校准 p 值={test.pvalue:.4f}")
# 经验算术均值和经验中位数从同一批 365 个模拟值计算, 比较它们可观察右偏与组间差异。
print(f"均值={pm.mean():.2f}, 中位数={np.median(pm):.2f}")
# 这是当前模拟样本中实际严格超过阈值的个数, 为整数, 并不等于模型期望次数。
print(f"观察到的超阈值次数={np.count_nonzero(pm > limit)}")
# 以 365 个观测为口径, 将单次超阈值概率乘以 365 得到期望计数。
# 分别使用整体模型和分组混合模型; 期望值可为小数, 也不保证与本次观察计数完全相等。
print(f"期望超阈值次数: 整体模型={365 * whole.sf(limit):.2f}, "
      f"分组模型={365 * tail:.2f}")
# 在正浓度范围上取 500 个点画连续密度曲线, 右端稍超过样本最大值。
x = np.linspace(1, 1.1 * pm.max(), 500)
# 左图检查整体分布形状, 右图单独检查组内对数正态假设。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# density=True 将柱形面积归一到 1, 纵轴为密度而非频数, 才能与连续 pdf 直接叠加。
axes[0].hist(pm, bins=24, density=True, alpha=0.4, label="模拟数据")
# 绘制合并数据得到的单一对数正态拟合密度, 作为基准模型。
axes[0].plot(x, whole.pdf(x), label="单一对数正态模型")
# 混合密度  $f(x) = \sum w_j f_j(x)$ , 与超阈值概率的加权方式保持一致。
axes[0].plot(x, sum(w * d.pdf(x) for w, d in zip(weights, parts)),
             label="分组混合模型")
# 横轴标明浓度单位  $\mu\text{g}/\text{m}^3$ , 纵轴密度的积分才是概率。
axes[0].set(title="整体拟合与分组拟合",
            xlabel="浓度 ( $\mu\text{g}/\text{m}^3$ )", ylabel="概率密度")
# 较小字号的图例区分数据直方图、单一模型及分组混合模型。
axes[0].legend(fontsize=8)
# 若组 0 的浓度对数正态, 则其对数应近似正态, 因此用正态概率图检查  $\log(pm)$ 。
# 点接近直线支持该组对数分布的形状; 该图不是对总体混合分布的正态性检验。
stats.probplot(np.log(pm[group == 0]), dist="norm", plot=axes[1])
# 明确右图只展示组 0, 且纵轴为取对数后的排序数值, 避免与原浓度单位混淆。
axes[1].set(title="第 0 组: 对数浓度概率图", xlabel="理论顺序统计量中位数",
            ylabel="排序后的对数浓度")
# 自动调整左右图边距, 避免轴标签与图题拥挤。
fig.tight_layout(pad=0.8, w_pad=1.2)
# 以 180 dpi 保存应用案例图片到当前工作目录。
# bbox_inches="tight" 使面板标题和坐标说明完整保存在 PNG 中; 教材直接使用此文件。
fig.savefig("application_pm.png", dpi=180, bbox_inches="tight")
# 保存后关闭图对象, 避免批量执行多个案例时持续占用绘图资源。
plt.close(fig)

```

规定环境下的实际运行输出

拟合优度统计量 $D=0.0733$, 校准 p 值 $=0.0020$

均值 $=37.02$, 中位数 $=32.23$

观察到的超阈值次数 $=10$

期望超阈值次数: 整体模型 $=12.37$, 分组模型 $=10.95$

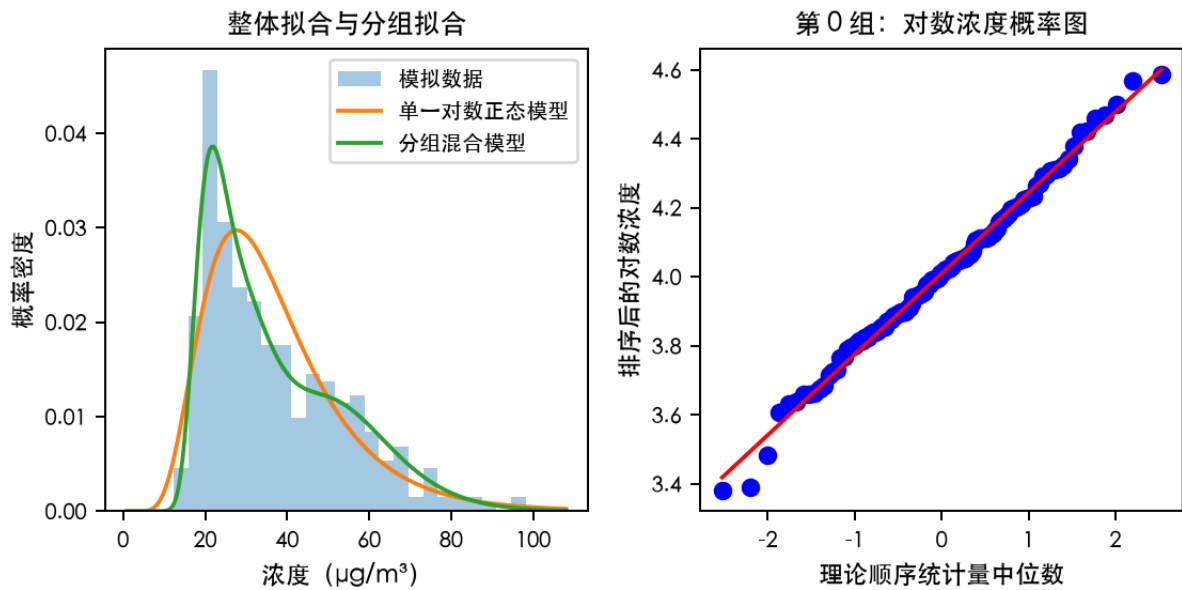


图 10-1 左图比较模拟浓度的整体拟合与已知分组拟合，右图检查第一组对数读数的正态分位数关系。参数与 e33_pm25.py 一致，图片为 application_pm.png。

读图指南 先看左图：柱状图是全部模拟观测，单一拟合曲线与分组混合曲线使用同一浓度单位。再看右图：只检查第 0 组的对数读数与正态理论分位数的关系；它不直接检验全部混合数据是否正态。参考直线容许位置和尺度变化，不要求原单位下斜率恰为 1。

规定环境中，整体 KS 统计量约为 0.0733，模拟校准 p 值为 0.0020。这个结果提示单一整体模型难以解释当前样本。分组依据来自生成机制中已知的组别；实际研究也应使用事先记录的季节、地点或批次，而不能在看完一张图后任意分组，再把分组当作独立证据。

正态概率图把参考分位数放在横轴，把有序的对数读数放在纵轴。若这些点近似沿一条直线排列，说明组内数据与某个正态位置、尺度组合相容。直线的斜率和截距允许变化，所以不要求点落在 $y=x$ 上。尾部系统弯曲需要进一步检查，少量散点偏离也可能只是抽样波动。

若问题转为下一批读数中有多少次超过门槛，应把每个位置是否超过门槛写成指示变量，再用期望的线性性质相加。设下一批各组的观测数仍与原来相同，超过门槛 u 的期望次数如下。

$$N_u = \sum_i \mathbf{1}_{\{x_i > u\}}, \quad E[N_u] = \sum_{g=1}^3 n_g [1 - F_g(u)]. \quad (10.4)$$

指示变量在 $X_i > u$ 时取 1，否则取 0。求期望时可以逐项相加，不需要假设各读数独立；但求次数方差或预测区间时，需要处理读数之间的依赖。

本次生成样本有 10 个读数超过 75。整体模型给出的代入期望约为 12.37，分组模型约为 10.95。这三个数分别是观测次数和两种模型下的期望，不能互换。一个期望可以不是整数，也没有承诺下一批恰好出现该次数。

10.4 计数的额外波动从哪里来

本例的同率泊松模型假定，各个等暴露观察单位具有相同的平均发生率。若窗口长度或暴露量不同，即使发生机制相同，计数的均值也会不同。先把观测单位统一，再讨论单一泊松是否足以解释样本波动。本例把所有单位设为等暴露量，60 个计数中的前 45 个按率 6 生成，其余 15 个按率 13 生成。

泊松总体的方差等于均值，但有限样本的方差和均值不会每次恰好相等。程序比较样本方差与样本均值的比值，并在拟合的同率泊松零假设下模拟 2000 批同样大小的样本。每一批都重新计算均值、方差和比值。参考分布给出了这个比值在零假设下正常波动的尺度。

为什么把不同发生率混在一起会增加波动？设潜在发生率 Λ 自身也变化。在给定 Λ 的条件下， X 服从参数为 Λ 的泊松分布。先在条件分布中求均值、方差，再对 Λ 平均，得到下列分解。

$$E[X] = E[\Lambda], \quad \text{Var}(X) = E[\text{Var}(X|\Lambda)] + \text{Var}(E[X|\Lambda]) = E[\Lambda] + \text{Var}(\Lambda). \quad (10.5)$$

第一项是各组自身随机波动的平均；第二项是各组平均水平不同带来的波动。只要发生率有变异，第二项就为正，边际计数方差便超过其均值。这是全方差公式的一次具体应用。

例题 不依赖程序的混合方差计算

假设以 3/4 的概率选到率为 6 的组，以 1/4 的概率选到率为 13 的组。平均率为 7.75。率的方差为 $(3/4)(6-7.75)^2 + (1/4)(13-7.75)^2 = 9.1875$ ，因此混合计数的理论方差为 $7.75 + 9.1875 = 16.9375$ 。实际 60 个观测的样本均值和方差会在这些总体量附近波动。

若 Λ 进一步服从伽马分布，积分掉 Λ 可得到负二项分布。它提供一种描述额外波动的计数模型。本例只是用负二项近似合并数据，不能把两种固定率的混合说成严格的伽马混合。采用 SciPy 的计数参数约定， $r>0$ ， $0<p\leq 1$ ，其均值 μ 和方差 v 满足下式。

$$\mu = \frac{r(1-p)}{p}, \quad v = \frac{r(1-p)}{p^2} = \mu + \frac{\mu^2}{r}, \quad \hat{p} = \frac{\bar{x}}{s^2}, \quad \hat{r} = \frac{\bar{x}^2}{s^2 - \bar{x}}. \quad (10.6)$$

$\bar{x}$ 和 s^2 分别是样本均值、使用分母 $n-1$ 的样本方差。最后两个式子由前两个关系解出，再用样本矩代替总体矩；必须有 $\bar{x}>0$ 且 $s^2>\bar{x}$ 。此处 r 可为正实数，不必解释为整数次成功。

程序 `e34_count.py` 先查看样本均值和方差比，再观察同率泊松参考模拟中的比值分布。左图比较泊松和负二项概率，右图比较诊断统计量，不把右图误读为事件计数的分布。完整源代码见下方。

编程思路 将两段不同泊松强度的教学计数合并，先检查样本方差与均值比，再在同均值泊松原假设下模拟离散程度。以矩估计拟合负二项作描述对照，并在单独的基准 `Poisson(6)` 模型上求报警阈值。

代码对应

合并 45 个 `Poisson(6)` 与 15 个 `Poisson(13)` 计数 $\rightarrow$ `counts = rng.poisson(np.r_[np.full(45, 6.0), np.full(15, 13.0)])`：强度异质性可以使合并后的方差大于均值，即便组内仍为泊松。

对 2000 组模拟样本各算方差/均值 $\rightarrow$ `null_ratio = null.var(axis=1, ddof=1) / null.mean(axis=1)`：`axis=1` 在每组 60 个计数内汇总，`ddof=1` 与观测样本方差约定一致。

以均值 m 、方差 v 求负二项 $r=m^2/(v-m)$ 、 $p=m/v \rightarrow r, p = \text{mean}^2 / (\text{var} - \text{mean})$ ，`mean / var`：该矩估计需 $v>m$ ；SciPy `nbinom` 的计数是成功 r 次前的失败数。

最小整数 h 使 $P[\text{Poisson}(6)\geq h]\leq 0.02 \rightarrow \text{threshold} = \text{int}(\text{baseline.ppf}(1 - \alpha)) + 1$ ：`ppf(1-alpha)+1` 将累计分位数转为含端点的上尾报警事件，本例 $h=13$ 。

易错提示 过度离散可能来自异质性、相关或模型遗漏，不单独证明负二项机制。报警阈值使用的是外定基准 $\lambda=6$ ，不能换成合并样本均值；比较 5% 时相应阈值为 11。

正文代码 e34_count.py

```
# 教材位置：第 10 章，第 10.4 节；图 10-2。
# 单位、变量与形状：counts 为 (60,)，null 为 (2000,60)；计数按次计，报警阈值按次计。
# 案例 34：计数数据的过度离散、负二项矩估计与离散报警阈值。
# 学习目标：理解泊松模型要求均值等于方差，及率的异质性如何使合并数据方差大于均值。
# 核心关系：SciPy 负二项参数  $r, p$  下  $E[X]=r(1-p)/p$ 、 $\text{Var}(X)=r(1-p)/p^2$ ；
# 用样本均值  $m$ 、样本方差  $v>m$  匹配矩可得  $r=m^2/(v-m)$ 、 $p=m/v$ 。
# 数据：共 60 个暴露量单位，前 45 个泊松率 6、后 15 个泊松率 13，全部为模拟计数。
# 报警基准另外指定为  $\text{Poisson}(6)$ ，阈值控制该已知基准下  $P(X \geq \text{threshold}) \leq 0.02$ 。
# 输出：统计量、模拟尾部比例、负二项参数及阈值；生成 application_count.png。
# 运行方式：安装依赖后在本目录执行 python e34_count.py，图保存到当前工作目录。

# NumPy 用于泊松模拟、样本统计、矩估计和离散横坐标。
import numpy as np
# SciPy 的冻结分布对象提供均值、方差、概率质量函数、分位数与生存函数。
from scipy import stats
# Matplotlib 比较经验计数分布与模型，并画出零假设模拟统计量分布。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 99；同一生成器依次产生观察样本及零假设模拟样本。
rng = np.random.default_rng(99)
# np.r_ 拼接 45 个 6 与 15 个 13，作为 60 个单位各自的泊松均值。
# 每组内部是泊松，但合并后的异质性会产生额外变异；该生成机制并非直接从负二项抽样。
counts = rng.poisson(np.r_[np.full(45, 6.0), np.full(15, 13.0)])
# mean 为样本均值，var 采用分母  $n-1$  的样本方差，ddof=1 明确这一统计口径。
mean, var = counts.mean(), counts.var(ddof=1)
# 离散指数=样本方差/样本均值；接近 1 符合泊松的量级，大于 1 提示过度离散。
ratio = var / mean
# 在“60 个单位共享同一泊松率”的零假设下模拟 2000 份样本，每份长度 60。
# 这里用观察样本的 mean 代入未知率，属于拟合零假设下的参数模拟诊断，非精确有限样本检验。
null = rng.poisson(mean, size=(2000, counts.size))
# 每行是一份模拟样本；沿 axis=1 算各自样本方差和均值，得到 2000 个离散指数。
null_ratio = null.var(axis=1, ddof=1) / null.mean(axis=1)
# 计算模拟统计量至少与观察值一样大的比例，选择右尾来检查过度离散。
# 分子与分母都加 1 是常用蒙特卡洛校正，避免有限次模拟给出零 p 值；该值仍有模拟误差。
p_sim = (1 + np.count_nonzero(null_ratio >= ratio)) / (len(null) + 1)
# 负二项的这种矩估计要求  $v>m$ ；不满足时公式会给非正或无穷的  $r$ ，不能继续套用。
if var <= mean:
    # 明确报错可防止把不满足参数条件的数据悄悄传入分布对象。
    raise ValueError("负二项矩估计要求样本方差大于样本均值。")
# 把样本均值和 ddof=1 样本方差代入负二项矩方程； $r$  可以是正实数，不必为整数。
#  $r$  为正整数时， $X$  可解释为成功  $r$  次之前的失败数， $p$  为单次成功概率；
# 本例  $r$  为矩估计得到的正实数，应按泊松率的伽马分布混合解释，不能说完成了非整数次成功
# 。
r, p = mean**2 / (var - mean), mean / var
# 冻结负二项分布，后面通过 nb.mean()/var()/pmf() 重复使用同一组参数。
nb = stats.nbinom(r, p)
# 矩估计应使模型理论均值和方差与采用的两个样本矩完全匹配到数值容差。
assert np.allclose([nb.mean(), nb.var()], [mean, var])
```

```
# 报警基准独立设为已知均值 6 的泊松分布, 允许的单次右尾概率为 2%。
# 它与前面的整体泊松拟合或负二项拟合用途不同, 阈值不由拟合的负二项模型产生。
baseline, alpha = stats.poisson(6), 0.02
# 离散 ppf(q) 返回使  $F(k) \geq q$  的最小整数 k; 选择 threshold=k+1,
# 使报警事件  $X \geq \text{threshold}$  等价于  $X > k$ , 并令其概率  $1 - F(k) \leq \alpha$ 。
threshold = int(baseline.ppf(1 - alpha)) + 1
# 离散  $sf(a) = P(X > a)$ , 所以  $P(X \geq \text{threshold})$  必须写  $sf(\text{threshold}-1)$ , 不能写  $sf(\text{threshold})$ 。
assert baseline.sf(threshold - 1) <= alpha
# 再把报警阈值降低 1 就会使右尾概率超过 alpha, 检验当前阈值确为满足条件的最小整数。
assert baseline.sf(threshold - 2) > alpha
# 并列显示均值、样本方差、离散指数及单侧模拟尾部比例, 联系效应大小与参考分布。
print(f"均值={mean:.3f}, 方差={var:.3f}, 离散指数={ratio:.3f}, 模拟 p 值={p_sim:.4f}")
# 显示负二项参数与独立基准阈值; 两个结果服务于分布描述和报警两个不同问题。
print(f"负二项参数 r={r:.3f}, p={p:.3f}; 报警阈值={threshold}")
# 报告基准模型下的实际单次报警概率; 离散跳跃使其一般小于而非恰好等于 0.02。
print(f"P(X >= threshold)={baseline.sf(threshold - 1):.6f}")
# 建立从 0 开始的整数网格, 并在观察最大值之后延伸几个取值, 便于展示模型尾部。
k = np.arange(counts.max() + 8)
# 左图比较计数概率质量, 右图比较离散指数在零假设下的模拟分布。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# 半整数边界使每个宽度为 1 的柱正好包含一个整数计数。
# 在单位箱宽下, 归一化密度的柱高可与对应计数的相对频率、离散 概率质量函数 直接比较。
axes[0].hist(counts, bins=np.arange(-0.5, k[-1] + 1.5), density=True, alpha=0.4)
# 用全部数据的样本均值拟合单一泊松, 展示其相对较窄的计数分布。
axes[0].plot(k, stats.poisson.pmf(k, mean), "o-", ms=3, label="泊松分布")
# 叠加负二项 概率质量函数, 观察允许方差大于均值后分布宽度与尾部的变化。
axes[0].plot(k, nb.pmf(k), "s-", ms=3, label="负二项分布")
# 强调暴露量相同; 若观察时长或面积不同, 原始计数通常不能直接以同一泊松率比较。
axes[0].set(title="计数模型比较",
            xlabel="每个等暴露单元的计数",
            ylabel="概率质量 / 相对频率")
# 图例对应两个模型曲线, 灰蓝色半透明柱对应观察频率。
axes[0].legend(fontsize=8)
# 右图纵轴为 2000 次模拟的频数, 没有 density=True, 不要把柱高直接当作概率密度。
axes[1].hist(null_ratio, bins=30, alpha=0.6)
# 红线标出观察离散指数; 位于模拟分布右侧越远, 零假设下出现的机会越小。
axes[1].axvline(ratio, color="red", label="观察到的离散指数")
# 横轴明确是均一泊松零假设下的方差/均值统计量, 不是原始计数。
axes[1].set(title="泊松原假设下的离散指数",
            xlabel="泊松原假设下的方差 / 均值", ylabel="频数")
# 图例标识红色观察值参考线, 便于与模拟频数直方图比较。
axes[1].legend(fontsize=8, framealpha=1)
# 调整布局并以 180 dpi 保存应用图。
fig.tight_layout(pad=0.8, w_pad=1.2)
# 保存完整标题与轴标签; 教材直接嵌入此 PNG, 并与教材的中文说明保持对应。
fig.savefig("application_count.png", dpi=180, bbox_inches="tight")
# 保存后释放图对象, 便于连续运行其他案例。
plt.close(fig)
```

规定环境下的实际运行输出

均值=8.383, 方差=16.274, 离散指数=1.941, 模拟 p 值=0.0005

负二项参数 r=8.906, p=0.515; 报警阈值=13

$P(X \geq \text{threshold}) = 0.008827$

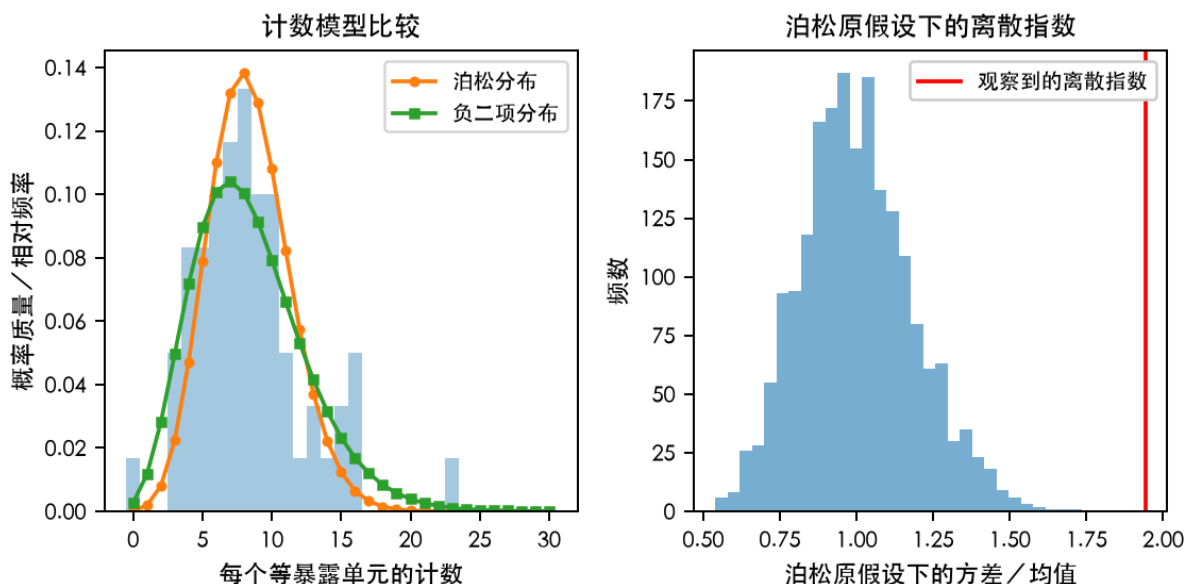


图 10-2 左图为等暴露单元的计数与两个候选模型，右图为同率泊松零假设下方差比的参考模拟。红线是实际模拟样本的比值。对应 `e34_count.py` 和 `application_count.png`。

读图指南 左图比较同一批计数的相对频率与两种模型；右图的横轴却是“样本方差/样本均值”这一诊断统计量，红线是观测统计量，不能把它读成原始事件数。红线位于模拟分布很右侧时提示同率泊松假设不足，仍需检查暴露量等前提。

本次样本均值约为 8.383，样本方差约为 16.274，比值约为 1.941。参考模拟 p 值约为 0.0005，提示同率、独立、等暴露的泊松零假设与当前样本不相容。负二项矩估计为 $r \approx 8.906$ 、 $p \approx 0.515$ ；模型均值与样本均值、模型方差与样本方差相等，是矩估计公式的构造性质；还需要检查整个分布及观测过程。

设置复核门槛是另一个问题。假定基准率 ϕ 已由外部信息给定，要求单次误报概率不超过 2%，选择最小整数 T 使 $P(X \geq T) \leq 0.02$ 。离散变量在端点处有正概率，所以 $X \geq T$ 对应生存函数 $\text{sf}(T-1)$ ，而不是 $\text{sf}(T)$ 。

$$T = \min\{t \in \mathbb{Z}_{\geq 0} : P(X \geq t) \leq 0.02\}, \quad P(X \geq T) = \text{sf}(T-1). \quad (10.7)$$

$\min$ 表示在满足约束的非负整数中取最小值。生存函数 $\text{sf}(a)$ 定义为 $P(X > a)$ ，因此这里必须减 1。

在 Poisson(6) 下, $P(X \geq 12) \approx 0.020092$, 略高于约束; $P(X \geq 13) \approx 0.008827$, 满足约束, 所以最小门槛为 13。门槛对应复核规则, 不是事件成因的证据。若基准率也由有限样本估计, 或同时监测许多单位, 还需要相应处理参数不确定性或多重比较问题。

10.5 状态转移怎样传播比例与不确定性

把一个固定区域划分成等面积单元, 每个单元在同一时刻恰属于农地、建设用地、其他用地三类之一。类别需要互斥且穷尽, 区域边界和单元面积保持不变。这时单元个数比例等于面积比例。若单元面积不等, 就应按面积加权, 不能直接把个数比例当成面积比例。

$$P = \begin{bmatrix} 0.88 & 0.09 & 0.03 \\ 0.04 & 0.93 & 0.03 \\ 0.02 & 0.05 & 0.93 \end{bmatrix}, \quad \mathbf{x}_0 = [0.42 \quad 0.33 \quad 0.25]. \quad (10.8)$$

P 的第 i 行第 j 列表示当前第 i 类的单元下一步进入第 j 类的条件概率。 $\mathbf{x}_0$ 是初始比例行向量, 分量次序始终是农地、建设用地、其他用地。每行概率之和为 1。

以第一行为例, 当前农地单元下一步仍是农地的概率为 0.88, 进入建设用地的概率为 0.09。这不表示建设用地中有 9% 来自农地, 后一问题把条件方向反了。计算建设用地下一步的比例时, 要分别加上三种来源的转入贡献。

例题 第一步比例的手算

下一步农地比例为 $0.42 \times 0.88 + 0.33 \times 0.04 + 0.25 \times 0.02 = 0.3878$ 。建设用地比例为 $0.42 \times 0.09 + 0.33 \times 0.93 + 0.25 \times 0.05 = 0.3572$ 。其他用地比例为 0.2550。三个分量之和仍为 1, 与矩阵乘法 $\mathbf{x}_0 P$ 一致。

$$\mathbf{x}_{t+1} = \mathbf{x}_t P, \quad \mathbf{x}_{10} = \mathbf{x}_0 P^{10}. \quad (10.9)$$

t 表示等长时间步。马尔可夫假设是给定当前状态后, 较早历史不再提供下一步转移的信息; 时间齐次假设是每一步使用同一个 P 。十步公式同时使用了这两个约定。

固定 P 时，矩阵公式给出各类的期望比例，实际单元去向仍有过程波动。例如 20000 个当前农地单元若相互独立地按第一行转移，其三个去向计数可用多项分布模拟，计数和严格等于 20000。20000 是单元个数，没有另给单元面积就不能写成公顷。

转移概率本身若不确定，就需要让一整行概率一起变化。给三个元素各加一个独立正态噪声可能产生负数或破坏行和。狄利克雷分布直接生成非负、和为 1 的概率向量，因此可以用来构造合法的转移行。它把贝塔分布的两类概率推广到多类概率。

$$Q_i \sim \text{Dirichlet}(CP_{i1}, CP_{i2}, CP_{i3}), \quad E[Q_{ij}] = P_{ij}, \quad \text{Var}(Q_{ij}) = \frac{P_{ij}(1 - P_{ij})}{C + 1}. \quad (10.10)$$

Q_i 是第 i 行随机概率向量， $C > 0$ 为浓度参数。本例 $C=300$ 。 C 增大时，各行围绕给定 P 的变化通常减小；同一行分量还存在负协方差，因为总和必须保持 1。

本例的 C 是人为设定的敏感性参数，不代表已经观察了 300 个实际单元。若有真实转移计数，需要先明确先验，再按狄利克雷与多项分布的共轭关系更新后验，才能把结果解释为相应的后验不确定性。这里只讨论指定概率扰动下的情景变化。

程序 `e35_land.py` 程序先用固定 P 计算十步期望，再抽取 2000 个合法转移矩阵；每个情景抽一次矩阵，并在该情景的十步中保持不变。种子为 2026，先执行的多项分布演示也会消耗随机数，因此核对图片时应运行完整文件。完整代码见下方。

编程思路 先由行转移矩阵计算土地三状态的确定性十步分布与稳态，再把每行概率当作狄利克雷情景抽样。每个抽到的矩阵在十步中保持不变，比较 2000 个情景的第 5、50、95 百分位。

代码对应

用第一行转移概率模拟 20000 个初始同类单元的去向 $\rightarrow$ `cell_counts = rng.multinomial(20000, P[0])`：这次 `multinomial` 调用发生在狄利克雷抽样之前，是随机序列复现所必需的调用顺序。

十步比例 $x_{10}=x_0P^{10} \rightarrow \text{deterministic} = x_0 @ \text{np.linalg.matrix_power}(P, 10):$

`matrix_power` 保持同一固定 P ，并由初始行分布左乘。

每行独立抽取 狄利克雷($300P_i$) 转移概率 $\rightarrow Q = \text{np.vstack}([\text{rng.dirichlet}(300 * \text{row})$

`for row in P]):` 300 是浓度而非样本数；每次抽样得到一个行和为一的完整情景矩阵。

对 2000 个十步情景按状态求 5%、50%、95% 分位 $\rightarrow lo, mid, hi =$

`np.quantile(pred, [0.05, 0.5, 0.95], axis=0):` `axis=0` 跨情景汇总，保留三个状态列。

易错提示 情景分位带取决于人为设定的 狄利克雷 浓度 300，不能直接称作由真实观测推得的置信区间或后验预测区间。每个情景十步共用 Q ，与每步重抽矩阵是不同模型。

正文代码 e35_land.py

```
# 教材位置：第 10 章，第 10.5 节；图 10-3。
# 单位、变量与形状：P 为 (3,3)，x0 为 (3,)，pred 为 (2000,3)；面积占比无量纲，时间步按年
# 计。
# 案例 35：土地类型马尔可夫转移、期望占比与转移矩阵的情景不确定性。
# 学习目标：区分等面积单元的多项抽样、固定矩阵的十步期望，以及扰动转移矩阵后的情景分布。
# 核心关系：行向量  $x_{t+1}=x_tP$ ，固定矩阵十步期望为  $x_0P^{10}$ ；稳态满足  $\pi P=\pi$ 。
# 数据与形状：三类土地，P 为 3×3，x0 为长度 3 的初始面积比例；
# 各行分别抽 狄利克雷(300*P[i]) 得到情景矩阵 Q，每个情景内固定 Q 连续推演 10 步。
# 2000 份情景结果 pred 为 (2000,3)，报告各分量的 5%、50%、95% 边际分位数。
# 这些是给定扰动设定的情景分位区间，不是用观测样本推导的频率学置信区间；
# 也未指定后验数据模型，因此不能自动称为后验可信区间或真实土地变化预测区间。
# 输出：抽样计数、稳态、十步点值和情景分位数；生成 application_land.png。
# 运行方式：安装依赖后在本目录执行 python e35_land.py，图片保存到当前工作目录。

# NumPy 用于多项/狄利克雷 抽样、矩阵幂、特征分解以及情景分位数。
import numpy as np
# 左图比较点预测和情景区间，右图展示每类土地的边际情景直方图。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 2026，先抽一次多项计数，再依次抽取 2000 个转移矩阵情景。
rng = np.random.default_rng(2026)
# 行与列的类型顺序均为农地、建设用地、其他用地；P[i,j] 是从类型 i 到 j 的一步概率。
# 所有元素都为正，使本例固定矩阵具有唯一稳态且长期期望分布收敛。
P = np.array([[0.88, 0.09, 0.03], [0.04, 0.93, 0.03], [0.02, 0.05, 0.93]])
# 初始三类土地面积占比分别为 42%、33%、25%；作为行概率向量右乘转移矩阵。
x0 = np.array([0.42, 0.33, 0.25])
# 验证每行是有效概率向量，保证乘法更新保持总面积比例为 1。
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
# 初始占比也必须非负且和为 1，否则不能解释为三类土地的完整组成。
assert np.all(x0 >= 0) and np.isclose(x0.sum(), 1)
# 单独示范：20000 个当前都属于第 0 类的等面积单元，下一步去向按 P[0] 多项抽样。
# 它不是按 x0 配置的全区域单元样本，也没有被用来拟合后面的 狄利克雷 参数。
cell_counts = rng.multinomial(20000, P[0])
```



```
# 打印三种去向的实现计数；其期望为 20000*P[0]，本次抽样一般有随机波动。
print("等面积单元下一年的去向计数：", cell_counts)
# 多项抽样把所有单元分到三个互斥且完备的去向，因此计数总和必须恰为 20000。
assert cell_counts.sum() == 20000
#  $\Pi P = \Pi$  转置成  $P^T \Pi^T = \Pi^T$ ，所以用  $P.T$  的右特征向量计算稳态。
vals, vecs = np.linalg.eig(P.T)
# 选择最接近特征值 1 的那一列；本例理论稳态为实数，取实部去除数值虚部。
# 特征向量尺度任意，除以分量和后才能解释为概率占比；steady 只对应固定的 P。
steady = np.real(vecs[:, np.argmin(np.abs(vals - 1))])
steady /= steady.sum()
# 检查归一化结果非负且满足固定矩阵的稳态方程，防止特征向量选错或方向用错。
assert np.all(steady >= 0) and np.allclose(steady @ P, steady)
# 计算固定 P 下第 10 步的期望面积占比，不是抽样得到的实际面积计数。
# 有限步点值与长期稳态概念不同，十步时不必已经达到 steady。
deterministic = x0 @ np.linalg.matrix_power(P, 10)
# 先用列表逐份收集情景结果，循环结束再转换为便于按列统计的二维数组。
pred = []
# 构造 2000 个独立的转移矩阵情景；下划线说明无需使用循环编号。
for _ in range(2000):
    # 狄利克雷 每次产生一行和为 1 的非负向量；vstack 把三行拼成 3×3 行随机矩阵。
    # 各行参数总和为 300，均值正好等于相应 P 行，单个分量方差为  $p_j(1-p_j)/301$ 。
    # 300 是人为设定的集中程度，越大扰动通常越小；没有在本例中从真实转移计数估计它。
    # 同一行的分量因总和固定而相关，不能视为三个独立概率。
    Q = np.vstack([rng.dirichlet(300 * row) for row in P])
    # 每份情景先抽一个 Q，再把这个 Q 固定使用 10 步，表达时间齐次转移参数的情景变化。
    # 此处没有逐年重新抽 Q，也没有模拟有限单元数的逐步状态抽样，结果仍是给定 Q 的期望占
    # 比。
    pred.append(x0 @ np.linalg.matrix_power(Q, 10))
# 转换为 (2000,3) 数组：行是情景，列是第 10 步的三种土地类型占比。
pred = np.asarray(pred)
# 验证每一份情景结果都在概率单纯形上，即非负且三类比例合计为 1。
assert np.all(pred >= 0) and np.allclose(pred.sum(axis=1), 1)
# 沿情景行求每一类的 5%、中位数、95% 分位数，三个输出各为长度 3 的数组。
# 每一类的 lo~hi 是中央约 90% 的边际情景区间，不是均值置信区间。
# 三类边际中位数一般不保证相加为 1，各列分位端点也不代表同一份可共同实现的情景，
# 因此不能把三条边际区间解释成联合覆盖率 90% 的整体区域。
lo, mid, hi = np.quantile(pred, [0.05, 0.5, 0.95], axis=0)
# 报告固定 P 的长期稳态比例，显示到 4 位小数。
print("固定转移矩阵的稳态占比：", np.round(steady, 4))
# 报告固定 P 的有限十步点值，便于与稳态和情景中位数分别比较。
print("固定转移矩阵的十步期望占比：", np.round(deterministic, 4))
# 按三行输出 5%、50%、95% 分位数；每列对应一种土地类型。
print("边际情景分位数 (5%、50%、95%)：\n",
      np.round([lo, mid, hi], 4))
# 创建情景分位区间图与情景分布图，两图使用同一份 pred 结果。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# 明确类型名称及顺序，使转移矩阵、初始向量、输出各列和图例保持对应。
names = ["农地", "建设用地", "其他用地"]
# errorbar 的 yerr 接收相对中心点的下/上偏移量，而不是直接接收上下端点。
# 误差棒以中位数为中心，允许上下不对称，表示边际情景分位区间而非标准误差置信区间。
axes[0].errorbar(np.arange(3), mid, yerr=[mid - lo, hi - mid],
                 fmt="o", capsize=5)
# 叉号为固定 P 的十步期望，占比经非线性的  $Q^{10}$  变换后，情景中位数不必与叉号重合。
axes[0].scatter(np.arange(3), deterministic, marker="x", label="固定矩阵 P")
# 横轴只表示三类土地标签；纵轴是第十步期望占比，不是面积计数或抽样频率。
axes[0].set(title="第十步的边际情景区间",
            xticks=range(3), xticklabels=names,
            ylabel="期望面积占比", ylim=(0, 0.7))
```



```
# 图例标明固定矩阵点值，其余圆点与误差棒来自情景的中位数和分位数。
axes[0].legend(fontsize=8)
# 依次读取 pred 的三列，各列都是跨 2000 份情景的一维边际结果。
for j, name in enumerate(names):
    # 叠加三个半透明直方图；纵轴为情景数，未做概率密度归一化。
    axes[1].hist(pred[:, j], bins=30, alpha=0.4, label=name)
# 横轴强调是条件期望占比，右图宽度由人为设定的 狄利克雷 情景扰动决定。
axes[1].set(title="情景中的期望占比",
            xlabel="第十步期望面积占比", ylabel="情景频数")
# 添加类型图例并调整整体边距，避免长类型名称挤压坐标标签。
axes[1].legend(fontsize=8)
fig.tight_layout(pad=0.8, w_pad=1.2)
# 以 180 dpi 保存图并关闭对象，结束这个不依赖外部数据的应用示例。
# bbox_inches="tight" 防止面板标题和类别名称被裁切：教材直接嵌入此 PNG。
fig.savefig("application_land.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

等面积单元下一年的去向计数： [17638 1760 602]

固定转移矩阵的稳态占比： [0.2125 0.4875 0.3]

固定转移矩阵的十步期望占比： [0.2517 0.4657 0.2826]

边际情景分位数（5%、50%、95%）：

```
[[0.1926 0.3866 0.217 ]
 [0.2495 0.4645 0.2801]
 [0.321 0.5431 0.3546]]
```

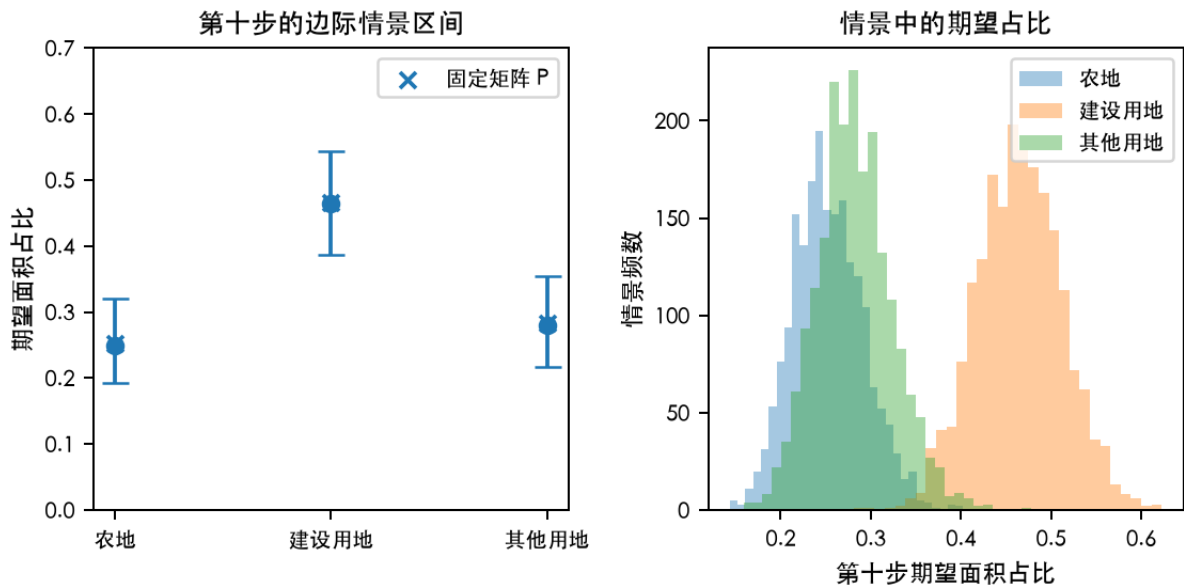


图 10-3 左图比较固定矩阵的十步期望与随机矩阵情景的中位数及 5% 至 95% 边际分位范围，右图展示三类情景期望比例的分布。对应 e35_land.py 和 application_land.png。

读图指南 左图的叉号是固定转移矩阵给出的十步期望，圆点和误差棒来自人为设定的转移矩阵情景分布；右图展示同一批情景的各类别边际结果。区间不是由真实样本估计的置信区间，三类边际端点也不必对应同一个共同情景。

每个情景中三类比例总和为 1，但分别计算的三个 5% 分位数不一定相加为 1，三个 95% 分位数也不一定相加为 1。这些是逐类的边际范围，端点通常来自不同情景，不能把三个下端点拼成一种区域布局。

图中范围也不是未来观测比例的预测区间，因为程序只传播转移矩阵的不确定性，没有在每个情景内继续模拟有限单元的随机去向。它也没有真实转移样本支持，所以不应称为统计置信区间或后验可信区间。把范围的来源说清楚，才能判断它回答了哪个问题。

若概率行向量 π 满足 $\pi P = \pi$ ，就称为平稳分布。它对应 P 的特征值 1 的左特征向量。本例每个元素均为正，满足保证唯一平稳分布并从任意初始比例收敛的一组充分条件。这个长期数学性质仍以规则持续不变为条件，不能直接写成现实用地必然达到的结果。

10.6 用户留存从一次转移到预计持续期

业务问题是：已有用户在接下来的十二个观察周期中，有多大比例仍未流失。先规定状态为活跃、沉默、流失，一个周期的长度由业务自行规定，本例不把它硬说成天或月。下列转移概率是教学假设，不来自任何真实平台。活跃表示本周期发生规定行为，沉默表示未发生但账号尚未流失，二者均计为“未流失”。

$$P = \begin{bmatrix} 0.75 & 0.20 & 0.05 \\ 0.30 & 0.60 & 0.10 \\ 0 & 0 & 1 \end{bmatrix}, \quad x_0 = (0.8, 0.2, 0). \quad (10.11)$$

P 的行对应当前状态，列对应下一状态；每行概率和为 1。 x_0 为三类用户的初始占比，按行向量更新。

第一步先手算。下一期活跃占比为 $0.8 \times 0.75 + 0.2 \times 0.30 = 0.66$ ；沉默为 $0.8 \times 0.20 + 0.2 \times 0.60 = 0.28$ ；流失为 0.06。因此活跃率是 0.66，未流失率却是 0.94。若把二者混作“留存率”，即使矩阵乘法毫无错误，业务解释也已经改变。

$$x_{t+1} = x_t P, \quad x_t = x_0 P^t, \quad R_t = x_{t,1} + x_{t,2}. \quad (10.12)$$

$t=0,1,2,\dots$ 为周期编号； R_t 是约定的未流失率，数学下标 1、2 分别指活跃与沉默。Python 对应索引 0、1。

本例“流失”是吸收态：一旦进入就不再返回。活跃与沉默之间可以往返，但各期均可能流失，长期分布最终集中在流失态。这个结论由模型假设决定，不能包装成乐观的稳定用户构成。对真实平台，首先应检查“流失后永不返回”和“每期转移规则相同”是否合理。

$$Q = \begin{bmatrix} 0.75 & 0.20 \\ 0.30 & 0.60 \end{bmatrix}, \quad F = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}. \quad (10.13)$$

Q 只保留未流失的两个状态，不是行和为 1 的完整转移矩阵。由于其所有特征值的模小于 1，几何级数收敛。 F 的第 i 、 j 项是在初始状态 i 下，吸收前处于状态 j 的期望期数，包含第 0 期。

$$t = F \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 15 \\ 13.75 \end{bmatrix}, \quad (0.8, 0.2)t = 14.75. \quad (10.14)$$

列向量 t 给出首次进入流失态所需步数的期望。等式也可从“本期计 1 次，再按 Q 进入下一状态”推出 $t=1+Qt$ 。14.75 是本模型的期望值，不是每个用户都在这一时点流失。

程序 `e40_customer_retention.py` 依照先手算一步、再迭代十二步、最后求吸收前期望持续期的顺序计算。区分活跃率与未流失率，并核对每一步三类概率之和为 1。

编程思路 用活跃、沉默、流失三个状态的每期转移链，从 $[0.8, 0.2, 0]$ 递推 12 步。把活跃率与未流失率分开绘制，再截出非吸收子矩阵 Q ，计算基本矩阵及吸收前期数，说明长期全部进入流失。

代码对应

保存第 0 至 12 步三类比例 $\rightarrow$ `distribution = np.empty((steps + 1, 3))`：行向量右乘 P 递推，每行仍为非负且和为一的概率分布。

未流失率=活跃比例+沉默比例 $\rightarrow$ retained_rate =

distribution[:, :2].sum(axis=1): 沉默仍属于未流失, 不能以活跃率冒充留存率。

$F=(I-Q)^{-1}=\sum_{k \geq 0} Q^k \rightarrow$ fundamental = np.linalg.solve(np.eye(2) - Q, np.eye(2)):

solve(I-Q,I) 得到 [[10,5],[7.5,6.25]], 谱半径小于一保证级数收敛。

初始混合的预计未流失期数=[0.8,0.2]F1=14.75 $\rightarrow$ initial_expected = x0[:2] @

expected_periods: 此约定包含第 0 期, 只统计吸收前活跃或沉默状态的占用期数。

易错提示 流失是吸收状态, 本模型不含回流, 所以长期分布为 [0,0,1], 不能画出积极的长期留存平台。转移概率是教学设定, 不是由真实客户日志估计; 预计期数也依赖模型在各期保持不变。

正文代码 e40_customer_retention.py

```
# 教材位置: 第 10 章, 第 10.6 节; 图 10-4。
# 三状态用户留存的教学模型; P 为人为设定, 不代表真实企业用户。
# 状态顺序: 活跃、沉默、流失; 流失为吸收态, 模型不允许流失后回流。
# P 形状 (3,3), x0 为 (3,), distribution 为 (13,3), 含当前第 0 期。
# 时间按期计, 比例无量纲; 基本矩阵给出流失前非流失期数的期望。
# 学习目标: 区分“活跃率”与“尚未流失率”, 解释有限期及长期结果。
# 本例无随机抽样; 执行 python e40_customer_retention.py 保存两面板图。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

P = np.array([[0.75, 0.20, 0.05], [0.30, 0.60, 0.10], [0.0, 0.0, 1.0]])
x0 = np.array([0.8, 0.2, 0.0])
steps = 12
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
assert np.array_equal(P[2], [0, 0, 1])
distribution = np.empty((steps + 1, 3))
distribution[0] = x0
# 行概率向量右乘 P; 同一矩阵用于每一步是时间齐次假设。
for step in range(steps):
    distribution[step + 1] = distribution[step] @ P
active_rate = distribution[:, 0]
# 沉默者尚未流失, 但不计为活跃者; 二者不能用同一个业务指标代替。
retained_rate = distribution[:, :2].sum(axis=1)
# 分布总和是守恒检查; 未流失率不增来自本例没有吸收后的回流。
assert np.allclose(distribution.sum(axis=1), 1)
assert np.all(active_rate <= retained_rate)
assert np.all(np.diff(retained_rate) <= 1e-12)
assert np.allclose(distribution[-1], x0 @ np.linalg.matrix_power(P, steps))

# Q 只保留两个暂态: 活跃与沉默。其谱半径小于 1, 最终必然吸收。
Q = P[:2, :2]
```

```
spectral_radius = np.max(np.abs(np.linalg.eigvals(Q)))
assert spectral_radius < 1
#  $F = I + Q + Q^2 + \dots = (I - Q)^{-1}$ ; 用求解线性方程避免显式求逆。
fundamental = np.linalg.solve(np.eye(2) - Q, np.eye(2))
expected_periods = fundamental @ np.ones(2)
initial_expected = x0[:2] @ expected_periods
#  $F_{ij}$  为从暂态  $i$  出发、吸收前访问暂态  $j$  的期望次数, 含当前期。
# 因每个未流失期后走一步, expected_periods 也等于首次流失的步数期望。
assert np.allclose(fundamental, [[10, 5], [7.5, 6.25]])
assert np.allclose(expected_periods, [15, 13.75])
assert np.isclose(initial_expected, 14.75)
long_run = x0 @ np.linalg.matrix_power(P, 1000)
assert np.allclose(long_run, [0, 0, 1], atol=1e-12)
print("第 12 期三状态比例: ", distribution[-1])
print("第 12 期活跃率、未流失率: ", active_rate[-1], retained_rate[-1])
print("暂态 Q 的谱半径: ", spectral_radius)
print("基本矩阵: \n", fundamental)
print("从活跃、沉默出发的流失前非流失期数期望 (含当前期): ", expected_periods)
print("初始混合人群的期望非流失期数: ", initial_expected)
print("本吸收模型的长期比例: ", long_run)

# 有限期图仍可能有较高留存, 不能据此宣称长期有稳定活跃用户。
# 长期全流失是吸收规则的数学后果, 不是某企业必然发生的现实预测。
figure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2), constrained_layout=True)
times = np.arange(steps + 1)
for j, name in enumerate(["活跃", "沉默", "流失"]):
    axes[0].plot(times, distribution[:, j], marker="o", ms=3, label=name)
axes[0].set(title="三状态每期分布", xlabel="观察周期数", ylabel="人群比例", ylim=(0, 1))
axes[1].plot(times, active_rate, marker="o", ms=3, label="活跃率")
axes[1].plot(times, retained_rate, marker="s", ms=3, label="未流失率")
axes[1].set(title="沉默用户仍属未流失", xlabel="观察周期数",
            ylabel="人群比例", ylim=(0, 1))
for ax in axes:
    ax.legend(fontsize=8)
    ax.grid(alpha=0.2)
fig.savefig("customer_retention.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

第 12 期三状态比例: [0.26953415 0.16276681 0.56769905]

第 12 期活跃率、未流失率: 0.269534145784369 0.43230095397291

暂态 Q 的谱半径: 0.93117376914899

基本矩阵:

```
[[10.    5.   ]
 [ 7.5   6.25]]
```

从活跃、沉默出发的流失前非流失期数期望 (含当前期): [15. 13.75]

初始混合人群的期望非流失期数: 14.75

本吸收模型的长期比例: [6.80723518e-32 4.11097485e-32 1.00000000e+00]

实际计算得到第 12 期活跃率约 0.269534，未流失率约 0.432301，流失率约 0.567699。前两项的差约 0.162767，就是尚未流失但处于沉默状态的占比。这些数值给出指定转移规则下的状态概率，不是对某个平台的实测判断。

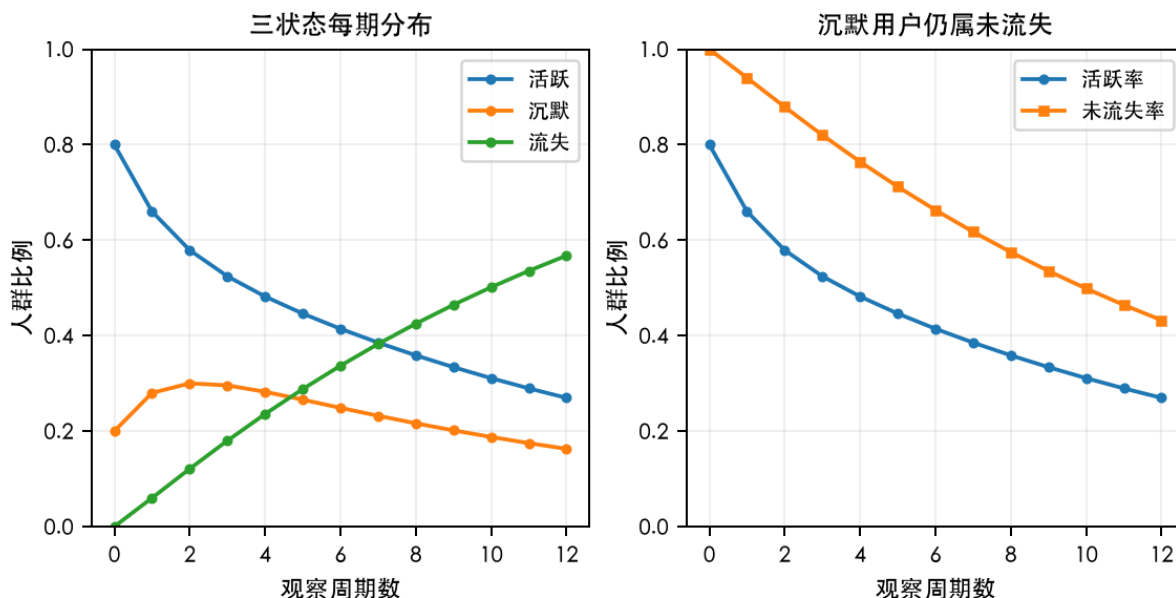


图 10-4 教学转移模型中的用户状态、未流失率与预计持续期，由 `e40_customer_retention.py` 实际生成。

读图指南 先从第 0 期读初始占比，再检查第 1 期活跃 0.66、未流失 0.94。中长期曲线应与吸收态设定一致：未流失率下降、流失率上升。预计持续期是对随机吸收时间的平均，不是在曲线上寻找一条人人共同的截止线。

10.7 用户画像降维怎样把统计量写成矩阵分解

考虑每名用户的月订单量、消费金额与浏览时长。三列单位不同，消费金额的数值可能远大于订单量。直接按原尺度寻找最大方差方向，往往首先反映单位大小。因此本例先按各列样本均值与样本标准差进行标准化，再比较相关结构。样本为 120 名教学模拟用户，不能据此推断任何真实客群。

$$Z = (X - \mathbf{1}\bar{x}^T)D^{-1}, \quad D = \text{diag}(d_1, d_2, d_3). \quad (10.15)$$

X 为 120×3 数据矩阵，每行为用户。 $\bar{x}$ 为三维列均值向量， $\mathbf{1}$ 为 120 维全 1 列向量； d_j 是第 j 列以 $\text{ddof}=1$ 计算的非零样本标准差。 Z 的三列均值为 0、样本方差为 1；下文 s_j 另指奇异值。

$$Z = USV^T, \quad C_Z = \frac{Z^T Z}{n-1} = V \frac{S^T S}{n-1} V^T. \quad (10.16)$$

这里用 S 表示本例奇异值矩阵，避免与其他章节的协方差字母混用。 V 的列是变量空间中的正交主方向， C_Z 为 3×3 样本协方差矩阵； $n=120$ 。该式将第七章的样本协方差式 (7.8) 与第九章的奇异值分解式 (9.2) 连接起来。

$$T = ZV, \quad \lambda_j = \frac{s_j^2}{n-1}, \quad \eta_j = \frac{s_j^2}{\sum_i s_i^2}. \quad (10.17)$$

T 为主成分得分，每行对应原用户。 s_j 为第 j 个奇异值， λ_j 为第 j 个主成分的样本方差， η_j 为方差解释率。比例可以约去共同的 $n-1$ 。

第一主成分不是“最重要的业务标签”，而是在指定标准化方式下方差最大的线性组合。若订单量与消费金额在同一方向上有较大载荷，可以结合原变量解释该方向，但符号可整体反转，不改变轴或得分之间的相对关系。比较两次分解应比较同一子空间或投影结果，不能只比较载荷向量正负。

保留前两列得分得到二维画像，重建时先回到标准化特征空间，再按 D 与均值还原原单位。丢弃部分方差可能同时丢弃有业务价值的少数行为，是否适合分群或预测还需单独验证。PCA 本身是描述性变换，不自动给出参数置信区间、用户类别真相或因果关系。

程序 `e41_customer_pca.py` 从带不同单位的三列模拟特征出发，先逐列标准化，再用 SVD 计算得分与解释率；另用协方差矩阵特征分解检查同一结果。

编程思路 由共享潜变量生成 120 个教学用户的订单数、消费金额和浏览时长，三列单位不同。先用各列样本标准差标准化，再以 SVD 做 PCA，并用标准化协方差的 `eigh` 核对特征值、解释率和投影子空间。

代码对应

以共同潜变量改变各用户的泊松订单强度 $\rightarrow$ `orders = rng.poisson(np.exp(2.2 + 0.35 * latent))`: `rng.poisson(np.exp(2.2+0.35*latent))` 产生非负整数订单，同时使三类特征具有共同变化来源。

$Z_{ij}=(X_{ij}-\text{列均值})/\text{列样本标准差} \rightarrow Z = (X - \text{mean}) / \text{sd}$: `axis=0` 沿用户求各特征统计量，`ddof=1` 与后续协方差分母 $n-1$ 一致。

标准化数据投影到右奇异方向 $\rightarrow$ `scores = Z @ directions`: `directions=Vh.T`, `scores=Z@directions`，使主成分坐标是标准化特征的线性组合。

各标准化主成分的方差占比= $s_j^2/\sum s^2 \rightarrow \text{ratio} = s^{**2} / \text{np.sum}(s^{**2})$: 本次约为 `[0.96005204,0.03620924,0.00373871]`; 协方差 `eigh` 提供另一条核对路径。

易错提示 这是有意构造相关性的教学模拟，不代表真实用户群，也不构成因果或统计推断。标准化后的载荷属于无量纲空间；重构回元、分钟等原单位时必须乘回 `sd` 再加 `mean`。

正文代码 `e41_customer_pca.py`

```
# 教材位置：第 10 章，第 10.7 节；图 10-5。
# 用户画像的教学模拟：120 个用户、月订单数、月消费金额、月浏览时长。
# 数据由明确给出的潜在变量和噪声生成，不代表真实用户或任何经营结论。
# 三列单位依次为次、元、分钟；X 形状 (120,3)，每行一个模拟用户。
# 先按样本均值和样本标准差标准化，再用 SVD 求描述性主成分。
# Z 形状 (120,3)，C 为 (3,3)，scores 为 (120,3)，ratio 为 (3,)。
# 固定种子 2026；输出列尺度、解释率、主成分方向与二维重建误差。
# 执行 python e41_customer_pca.py，图片写入当前目录。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

rng = np.random.default_rng(2026)
n = 120
latent = rng.normal(size=n)
# 潜在活跃度影响订单强度，订单为整数；后两列仍有各自独立噪声。
orders = rng.poisson(np.exp(2.2 + 0.35 * latent))
amount = 100 + 45 * orders + 30 * latent + rng.normal(0, 30, n)
minutes = 40 + 8 * latent + 2 * orders + rng.normal(0, 4, n)
X = np.column_stack([orders, amount, minutes])
# axis=0 跨用户计算每列统计量；不能沿 axis=1 把每个用户的三种单位混合。
mean = X.mean(axis=0)
sd = X.std(axis=0, ddof=1)
assert np.all(sd > 0)
Z = (X - mean) / sd
assert np.allclose(Z.mean(axis=0), 0, atol=1e-12)
assert np.allclose(Z.std(axis=0, ddof=1), 1)
```



```
# C 是标准化样本的协方差，也就是原三特征的样本相关矩阵。
C = Z.T @ Z / (n - 1)
U, s, Vh = np.linalg.svd(Z, full_matrices=False)
directions = Vh.T
scores = Z @ directions
ratio = s**2 / np.sum(s**2)
# 用独立的对称特征值算法核对，eigh 默认升序，须同时重排特征向量。
eigenvalues, eigenvectors = np.linalg.eigh(C)
order = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[order]
eigenvectors = eigenvectors[:, order]
assert np.allclose(eigenvalues, s**2 / (n - 1))
assert np.allclose(ratio, eigenvalues / eigenvalues.sum())
# 向量可能整体异号；外积给出同一一维子空间的正交投影矩阵。
for j in range(3):
    assert np.allclose(np.outer(directions[:, j], directions[:, j]),
                        np.outer(eigenvectors[:, j], eigenvectors[:, j]))
assert np.allclose(scores, U * s)
assert np.allclose(scores.T @ scores / (n - 1), np.diag(eigenvalues))
# 保留前两成分重建标准化数据，再还原单位；误差先在无量纲 Z 上度量。
reconstructed_z = scores[:, :2] @ directions[:, :2].T
reconstructed_x = reconstructed_z * sd + mean
relative_error = np.linalg.norm(Z - reconstructed_z) / np.linalg.norm(Z)
assert np.isclose(relative_error, np.sqrt(ratio[2]))
print("三列样本均值（次、元、分钟）：", mean)
print("三列样本标准差（次、元、分钟）：", sd)
print("标准化样本协方差：\n", C)
print("主成分解释率：", ratio)
print("主成分方向（每列一个方向）：\n", directions)
print("前两成分的标准化重建相对误差：", relative_error)
# 主成分得分不相关不等于独立；这里不进行检验、置信区间或因果推断。
# 未标准化时金额的大数值会主导平方损失，这是另一种分析问题。

figure_chinese_plot()
fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.9), constrained_layout=True)
axes[0].scatter(orders, amount, alpha=0.6, s=12)
axes[0].set(title="明确模拟的原始画像", xlabel="月订单数（次）",
            ylabel="月消费金额（元）")
axes[1].scatter(scores[:, 0], scores[:, 1], alpha=0.6, s=12)
axes[1].set(title="标准化后的二维得分", xlabel="第一主成分得分",
            ylabel="第二主成分得分")
axes[2].bar([1, 2, 3], ratio)
for index, value in enumerate(ratio, 1):
    axes[2].text(index, value + 0.025, f"{value:.1%}", ha="center", fontsize=7)
axes[2].set(title="标准化方差的解释率", xlabel="主成分序号",
            ylabel="解释率", ylim=(0, 1.1))
axes[2].set_xticks([1, 2, 3])
for ax in axes:
    ax.grid(alpha=0.2)
fig.savefig("customer_pca.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

三列样本均值（次、元、分钟）： [10.3 560.48149102 60.78780578]

三列样本标准差（次、元、分钟）： [4.6120048 231.24277108 16.45141845]

标准化样本协方差:

```
[[1.          0.98726539 0.90588024]
 [0.98726539 1.          0.92644515]
 [0.90588024 0.92644515 1.          ]]
```

主成分解释率: [0.96005204 0.03620924 0.00373871]

主成分方向 (每列一个方向):

```
[[-0.58020084  0.48595477  0.65361682]
 [-0.58427051  0.31077273 -0.74969879]
 [-0.56744599 -0.8168649   0.10361844]]
```

前两成分的标准化重建相对误差: 0.06114499927370813

本次运行的三个解释率约为 96.0052%、3.6209%与 0.3739%，前两项累计约 99.6261%。

保留两个成分时，标准化重建的相对弗罗贝尼乌斯误差约 6.1145%，正是被舍弃解释率 0.00373871 的平方根；“舍弃 0.3739%方差”与“相对误差 6.1145%”使用的量不同，不应混称为同一个百分比。

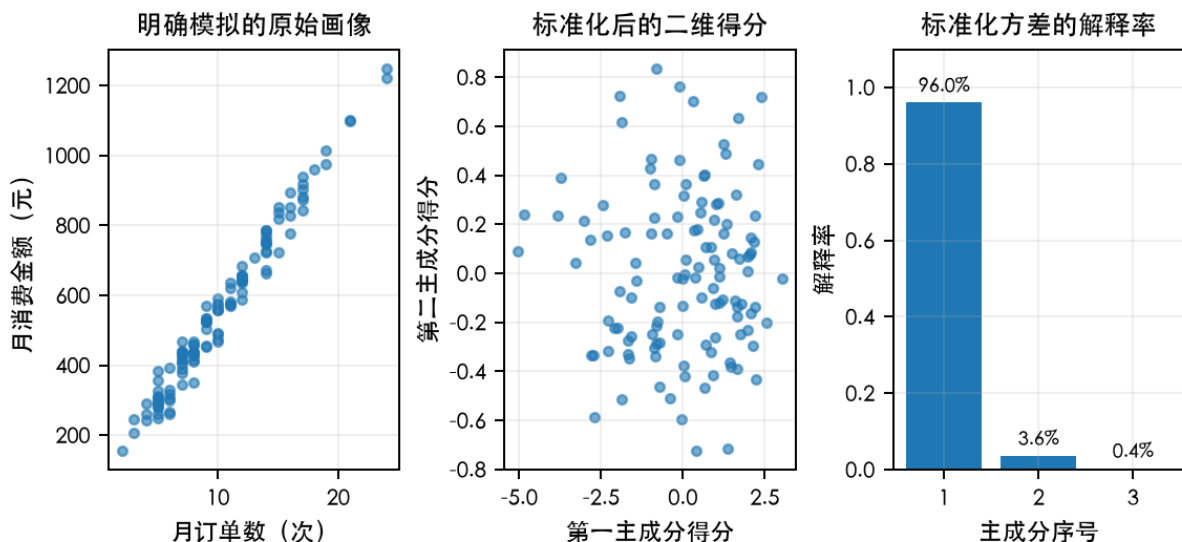


图 10-5 模拟用户画像的标准化、主成分得分与方差解释率，由 e41_customer_pca.py 实际生成。

读图指南 先辨认图中的原变量与主成分坐标，不能把得分轴当作消费金额单位。解释率图中的比例来自同一标准化数据的奇异值平方；前两项之和表示保留的样本方差比例。散点中的接近只说明所显示二维表示下接近，不证明用户属于同一真实类别。

10.8 用真实标定数据贯通最小二乘与随机误差

NIST 的 Norris 数据用于臭氧监测仪的标定，是 36 对公开观测，原始文件按 y、x 两列排列。与前面假设的商业转移数据不同，这里采用真实公开数值。原始 StRD 文件没有列出数值

单位，也没有给出足以把 x 或 y 认定为某一具体基准设备的全部说明，因此图表只使用原始变量名，不另补单位或设备角色。配套程序直接嵌入这 36 对数据，离线即可运行。

要解决的问题是：用一条含截距的直线描述 y 随 x 的变化，并检验数值实现是否重现 NIST 认证结果。模型写成 $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ 。截距允许系统偏移，斜率允许比例偏移；“误差”在模型中是随机变量，拟合后的“残差”则是观测减拟合值，二者不应混称为同一组已知数。

$$y = A\beta + \varepsilon, \quad A = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_{36} \end{bmatrix}, \quad \hat{\beta} = \operatorname{argmin}_{b \in \mathbb{R}^2} \|y - Ab\|_2^2. \quad (10.18)$$

y 为 36 维观测列向量， A 为 36×2 设计矩阵， $\beta = (\beta_0, \beta_1)^T$ 。 A 满列秩时最小二乘系数唯一；程序使用 `lstsq`，不为求系数显式构造逆矩阵。

NIST 给出的认证截距约为 -0.2623230738 ，斜率约为 1.0021168180 ，残差标准差约为 0.8847963961 。本书用它们核对实现精度；认证值不是声称直线模型在所有使用条件下都正确。把 $x=500$ 代入拟合式，得到约 500.796086 ，这是条件均值的点估计，不是某个新观测必定取得的结果。

$$\hat{\sigma}^2 = \frac{\|y - A\hat{\beta}\|_2^2}{n - 2}, \quad n = 36. \quad (10.19)$$

估计了截距和斜率两个参数，残差自由度为 34。该方差估计的无偏性要求模型正确且误差同方差、互不相关；单靠较大的 R^2 不能保证这些条件。

$$E[\hat{\beta}|A] = \beta, \quad \operatorname{Cov}(\hat{\beta}|A) = \sigma^2(A^T A)^{-1}. \quad (10.20)$$

条件是 A 固定且满列秩， $E[\varepsilon|A]=0$ ， $\operatorname{Cov}(\varepsilon|A)=\sigma^2 I$ 。先由 $\hat{\beta} = \beta + (A^T A)^{-1} A^T \varepsilon$ ，再按线性变换的期望与协方差规则推出此式；矩阵逆在这里用于推导，并非建议用它替代数值求解。

这里把第七章的正规方程式 (7.5) 与线性变换的协方差规则连接起来：最小二乘给出投影和计算方法，随机误差模型决定估计量如何波动。若还假设误差联合正态，则可进一步建立精确的小样本 t 区间；若 x 也存在不能忽略的测量误差，当前条件模型需要重新审视。

程序 e42_norris_calibration.py 读取源码中按原始 y、x 顺序保存的 36 对数据，构造含常数列的设计矩阵，计算拟合、残差与残差标准差，并以 NIST 认证结果作数值核对。

编程思路 嵌入 NIST Norris 的 36 对官方数据以便离线重现，严格保持原文件 y、x 列顺序。以 $A=[1,x]$ 做最小二乘，手算中心化斜率与截距复核，再对照认证系数、残差平方和、残差标准差与决定系数。

代码对应

每行写成 $[1, x_i]$ ，系数为 $[\text{截距}, \text{斜率}] \rightarrow A = \text{np.column_stack}([\text{np.ones}(n), x])$ ：此处顺序与前面的 $[t, 1]$ 案例相反，coef 的解释必须随设计矩阵列顺序改变。

$b_1 = \sum(x_i - \bar{x})(y_i - \bar{y}) / \sum(x_i - \bar{x})^2 \rightarrow \text{manual_slope} = (xc @ yc) / (xc @ xc)$ ：两个中心化向量的内积直接实现手算求和，截距再由 $\bar{y} - b_1 \bar{x}$ 得到。

残差标准差 $= \sqrt{[SSE/(36-2)]} \rightarrow \text{residual_sd} = \text{np.sqrt}(sse / (n - \text{rank}))$ ：分母减去设计矩阵秩，即两个拟合参数，不是 36 或 35。

在 $x=500$ 处代入 $\hat{y} = b_0 + 500b_1 \rightarrow \text{prediction} = \text{np.array}([1.0, x_new]) @ \text{coef}$ ：
 $[1, x_new]$ 与设计矩阵保持同一列顺序，结果约 500.796085936453。

易错提示 NIST 说明这是臭氧监测仪标定数据，但原数据未标明计量单位，也未提供足以给 x、y 擅自命名具体仪器角色的信息。 R^2 极高仍须单独查看残差；单点拟合值不是预测区间。

正文代码 e42_norris_calibration.py

```
# 教材位置：第10章，第10.8节；图 10-6。
# NIST StRD Norris 的真实臭氧监测仪标定数据，36 对观测原始列顺序为 y,x。
# 来源：https://www.itl.nist.gov/div898/strd/lls/data/Norris.shtml
# 数据：https://www.itl.nist.gov/div898/strd/lls/data/LINKS/DATA/Norris.dat
# 原始资料未声明数值单位；这里不补设单位，也不指定两列为某具体仪器。
# data 形状 (36,2)，A 为 (36,2)，coef 为 (2,)，residual 为 (36,)。
# 设计列顺序 [1,x]，所以 coef[0] 是截距，coef[1] 是斜率。
# 输出手算/最小二乘系数、残差标准差、平方和及 x=500 处拟合值。
# 数据直接嵌入程序，离线可运行；执行 python e42_norris_calibration.py。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot
```

```
# 每行严格保留官方文件的 [y,x] 顺序; 不能交换两列后仍沿用认证系数。
data = np.array([
    [0.1, 0.2], [338.8, 337.4], [118.1, 118.2],
    [888.0, 884.6], [9.2, 10.1], [228.1, 226.5],
    [668.5, 666.3], [998.5, 996.3], [449.1, 448.6],
    [778.9, 777.0], [559.2, 558.2], [0.3, 0.4],
    [0.1, 0.6], [778.1, 775.5], [668.8, 666.9],
    [339.3, 338.0], [448.9, 447.5], [10.8, 11.6],
    [557.7, 556.0], [228.3, 228.1], [998.0, 995.8],
    [888.8, 887.6], [119.6, 120.2], [0.3, 0.3],
    [0.6, 0.3], [557.6, 556.8], [339.3, 339.1],
    [888.0, 887.2], [998.5, 999.0], [778.9, 779.0],
    [10.2, 11.1], [117.6, 118.3], [228.9, 229.2],
    [668.4, 669.1], [449.2, 448.9], [0.2, 0.5],
], dtype=float)
y, x = data[:, 0], data[:, 1]
n = x.size
A = np.column_stack([np.ones(n), x])
coef, sums, rank, singular = np.linalg.lstsq(A, y, rcond=None)
yhat = A @ coef
residual = y - yhat
# 用中心化求和公式作独立手算核对, 而非对 A.T@A 显式求逆。
xc, yc = x - x.mean(), y - y.mean()
manual_slope = (xc @ yc) / (xc @ xc)
manual_intercept = y.mean() - manual_slope * x.mean()
assert n == 36 and rank == 2
assert np.allclose(coef, [manual_intercept, manual_slope])
# 官方认证截距、斜率; 只用作数值核验, 不反向覆盖本次实际拟合结果。
certified_coef = np.array([-0.262323073774029, 1.00211681802045])
assert np.allclose(coef, certified_coef, rtol=1e-12, atol=1e-11)
sse = residual @ residual
residual_sd = np.sqrt(sse / (n - rank))
r_squared = 1 - sse / (yc @ yc)
# 残差标准差分母为 36-2=34; 它不同于直接除以 36 的均方根误差。
assert np.isclose(sse, 26.6173985294224, rtol=1e-12)
assert np.isclose(residual_sd, 0.884796396144373, rtol=1e-12)
assert np.isclose(r_squared, 0.999993745883712, rtol=1e-12)
assert np.isclose(sums[0], sse)
assert np.allclose(A.T @ residual, 0, atol=1e-8)
# x=500 在已观测范围内; 这是拟合均值, 不是下一次仪器读数的保证。
x_new = 500.0
prediction = np.array([1.0, x_new]) @ coef
print("数据来源: NIST StRD Norris, 36 对臭氧监测仪标定观测")
print("截距、斜率: ", coef)
print("手算截距、斜率: ", manual_intercept, manual_slope)
print("残差平方和、残差自由度: ", sse, n - rank)
print("残差标准差、决定系数: ", residual_sd, r_squared)
print("x=500 处的拟合均值: ", prediction)
# 高决定系数还受 x 的覆盖范围影响, 不能代替残差诊断或外部验证。
# 若要置信/预测区间, 还需说明噪声模型和推断条件; 本例只做拟合核验。

configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0), constrained_layout=True)
order = np.argsort(x)
axes[0].scatter(x, y, s=14, alpha=0.7, label="官方观测")
axes[0].plot(x[order], yhat[order], color="tab:red", label="最小二乘拟合")
axes[0].set(title="臭氧监测仪标定数据", xlabel="官方数据列 x", ylabel="官方数据列 y")
```

```
axes[0].legend(fontsize=8)
axes[1].scatter(x, residual, s=14, alpha=0.7)
axes[1].axhline(0, color="gray", ls="--")
axes[1].set(title="残差需单独检查", xlabel="官方数据列 x", ylabel="观测减拟合值")
for ax in axes:
    ax.grid(alpha=0.2)
fig.savefig("norris_calibration.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

规定环境下的实际运行输出

数据来源: NIST StRD Norris, 36 对臭氧监测仪标定观测

截距、斜率: $[-0.26232307 \quad 1.00211682]$

手算截距、斜率: $-0.26232307377398456 \quad 1.0021168180204543$

残差平方和、残差自由度: $26.617398529422942 \quad 34$

残差标准差、决定系数: $0.8847963961443822 \quad 0.9999937458837117$

$x=500$ 处的拟合均值: 500.796085936453

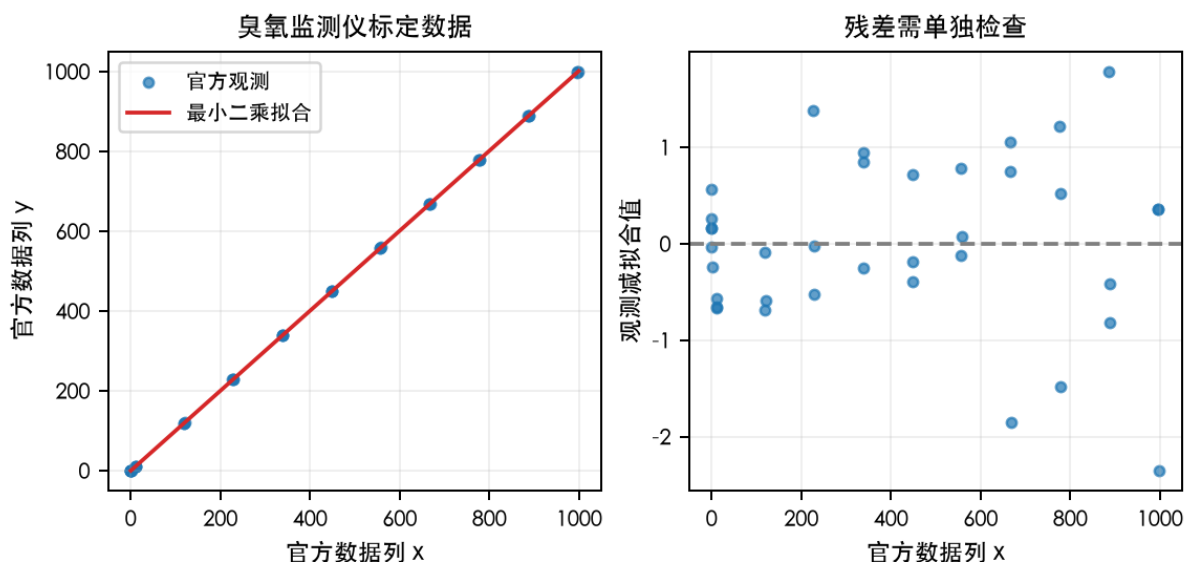


图 10-6 NIST Norris 观测数据的线性标定拟合与残差诊断，由 `e42_norris_calibration.py` 实际生成。

读图指南 左图先检查观测与拟合线的关系，再看右图残差是否随 x 呈现结构或幅度变化。宽范围的 x 可以使 R^2 非常接近 1，因此仍需读残差；没有明显图形结构也不能单凭一张图证明误差独立或同方差。来源未注明数值单位，图中不另设单位。

10.9 本章核心要点与常见误区

核心要点：分组混合的总体概率是组内概率的加权和；计数比较须统一暴露量；状态概率按规定方向乘转移矩阵；吸收态决定长期结构；PCA 依赖中心化与尺度约定；最小二乘的几何结论与估计量的概率性质需要分别说明条件。

误区一是把模拟数据称为真实调查。本章仅 Norris 案例使用所引来源中的真实观测，其余案例均为教学模拟或假设模型。误区二是把矩阵情景区间、频率学置信区间和贝叶斯可信区间混为一谈：它们的随机对象和解释对象不同。

误区三是把模型计算值当作决策承诺。留存模型中的长期流失、PCA 中的高解释率或回归中的大 R^2 ，都不能替代对转移规则、业务价值和测量机制的检查。误区四是只核对小数，不核对变量顺序：调换数据列、状态名称或设计矩阵列时，图注与解释也必须同步改变。

10.10 练习与参考解答

练习 10-1（概念与手算，入门） 两组观测数为 100 与 300，组内超限概率为 0.1 与 0.3。随机选一份观测，其超限概率与 400 份观测的期望超限数各为多少？

练习 10-2（概念与手算，进阶） 随机抽到的计数先以 $3/4$ 概率使用泊松率 6，以 $1/4$ 概率使用泊松率 13。求总体均值和方差，解释为什么仍不能当作同率泊松。

练习 10-3（概念与手算，入门） 用户留存案例从 $x_0=(0.8,0.2,0)$ 出发。手算一步活跃率、未流失率和流失率，并说明三者关系。

练习 10-4（概念与手算，进阶） Norris 回归有 36 个观测，估计截距与斜率。残差平方和认证值为 26.6173985294224，求残差方差与标准差。

练习 10-5（编程实践，进阶） 运行 e33_pm25.py，对每组分别打印 n_g、w_g 与组内模型超限概率。用这三项重算总体超限概率，核对原脚本 tail。

练习 10-6（编程实践，进阶） 在 e40 模型下模拟 20000 名用户的十二期状态，比较经验未流失率与矩阵计算。说明合理的误差量级。

练习 10-7（编程实践，进阶） 把 e41 中的消费金额统一乘以 100，再逐列标准化并重做 PCA。哪些结果应保持？怎样规避特征向量符号差异？

练习 10-8（编程实践，进阶） 在 e42 中用正规方程数值解作教学对照，核对 lstsq 所得系数与残差，并解释为何正式计算仍采用后者。

练习 10-9（场景应用，进阶） 两家门店日计数不同，但营业时长分别为 4 小时与 12 小时。能否直接套同率泊松比较原始计数并判第二家异常？

练习 10-10（场景应用，进阶） Norris 观测 x 范围约为 0.2 至 999。有人用这条线对 $x=1200$ 作精确保证，问题在哪里？

练习 10-11（场景应用，拓展） 把用户转移模型流失行改为(0.05,0.05,0.90)，仍称第三类“永久流失”，再使用原吸收时间公式，是否妥当？

练习 10-12（场景应用，进阶） 某少数用户行为只贡献 1%的总方差，却与重要投诉相关。能否仅因 PCA 前两维解释率 99%就丢弃该方向？

参考解答与验收依据

10-1 权重为 $1/4$ 与 $3/4$ ，总概率 $= (1/4) \times 0.1 + (3/4) \times 0.3 = 0.25$ ，期望数 $= 400 \times 0.25 = 100$ 。直接把两组概率平均成 0.2 会忽略组大小。

10-2 $E\lambda = 7.75$ 。 $\text{Var}\lambda = (3/4)(1/4)(13-6)^2 = 9.1875$ 。全方差给出 $\text{Var}X = E\lambda + \text{Var}\lambda = 16.9375$ ，大于均值。条件于组内为泊松，不代表合并后也是同一泊松分布。

10-3 $x_1 = (0.66, 0.28, 0.06)$ ；活跃率 0.66，未流失率 0.94，流失率 0.06。活跃只是未流失的一部分；未流失加流失等于 1。

10-4 自由度 $36-2=34$ ，方差 $= 26.6173985294224/34 \approx 0.782864662630069$ ，标准差 $\approx 0.884796396144373$ 。除以 36 得到的是平均残差平方，不能与本题的无偏方差估计混用。

10-5 使用 `group==g` 选取各组，`n_g` 为布尔值求和、`w_g=n_g/365`；对 `parts[g]`调用 `sf(limit)`。验收为三组 `n_g=120、125、120`，`w` 和为 1，重算 `sum(w_g*sf_g)`与 `tail` 在 $1e-12$ 绝对容差内相同。不要重新生成另一份样本。

10-6 按 `xo`初始化类别，再每期按该类别所在 `P` 行抽下一状态。经验比例 $\hat{q}$ 与理论 q 比较；单次比例的标准误约为 $\sqrt{q(1-q)/20000}$ ，应报告差与标准误之比，而非要求逐位相同。流失用户一直停在吸收态；循环抽样不得每次重置同一个种子。

10-7 正比例换单位后，标准化 `Z` 在浮点容差内相同，因此奇异值、解释率及所选主子空间应相同。比较 `V_kV_k^T`或重建误差，不能强求每一列 `V` 的符号完全相同；可按 $1e-10$ 数量级容差核对本例。

10-8 构造 `A=[1,x]`，对照 `solve(A.T@A,A.T@y)`。本例应与认证系数相近，残差范数一致；`lstsq` 避免把条件数平方放大到正规矩阵，适用于更一般形状和秩情形。数值相同不能推出任何数据上两种实现都同样稳定。

10-9 不能。应先统一暴露量，按每小时率或带时长的 `Poisson( $\lambda t$ )`建模；还要考虑时段客流、独立性与率稳定性。比较条件不同的原始计数会把营业时间差当成异常。

10-10 `x=1200` 位于观测范围外，是外推；认证数值检验的是所给数据上的计算，不保证测量机制在更大范围仍线性。新观测还有随机误差，拟合均值也有估计误差，不能把一条点估计直线当成精确承诺。

10-11 不妥。新第三类能够返回，语义已变，应改称可回流或暂停等符合业务定义的状态；原“首次永久吸收”与基本矩阵解释不再适用。新矩阵需重新分析不可约性、稳态与所关心的时间指标。

10-12 不能仅据解释率决定。PCA 优化的是无监督重建方差，没有使用投诉损失或业务目标；需在独立评价方案中检查该方向对目标和少数群体的价值，必要时保留原特征或采用适合目标的方法。

10.11 继续阅读

真实数据阅读：NIST StRD 的 Norris 数据页面和 ASCII 数据文件，先确认 36 条观测、 y/x 列次序与两参数模型，再对照认证系数和残差统计量；不要把“认证计算结果”扩写成“认证现实模型”。矩阵方法阅读：麻省理工学院 18.06SC 第二单元“马尔可夫矩阵与傅里叶级数”与第三单元“奇异值分解”，分别复习行概率约定和正交分解。概率阅读：哈佛统计 110 所配《概率导论》中马尔可夫链主题，先检查自己的行列向量约定再与原书对照。

配套工具的完整源码

本节为拓展阅读，集中解释课程程序怎样运行和保存结果。首次学习可以直接运行附件中的工具文件；想了解程序组织方式时，再阅读下列完整实现。四份工具与附录中的同名源码完全一致，代码框均无行号。

正文代码 `environment_check.py`

```
# 环境检查：确认当前解释器及三个教学依赖的实际版本。
# 运行方式：在本目录的终端执行 python environment_check.py。
# 目标环境：Python 3.13、NumPy 2.4.6、SciPy 1.18.0、Matplotlib 3.11.1。
# 本脚本只打印版本，不自动安装或升级软件，也不将版本不一致判为异常。
# 若某个依赖尚未安装，对应 import 会报 ModuleNotFoundError；
# 请先在同一个解释器环境中执行 python -m pip install -r requirements.txt。

# sys 属于 Python 标准库；sys.version 记录正在执行本文件的解释器版本。
import sys
# NumPy 负责数组与线性代数；np 是本案例包统一采用的简称。
import numpy as np
# SciPy 提供概率分布、统计推断及相关数值计算接口。
import scipy
# Matplotlib 负责绘图；这里只读版本，不创建图形窗口。
import matplotlib

# sys.version 还含编译器等信息；split()[0]只取开头的版本号。
# 应使用这里显示的版本判断当前环境，而不是另一个终端里 pip 指向的环境。
print("Python 解释器版本：", sys.version.split()[0])
# 各库通过 __version__ 公开当前已加载版本；这三行可用于课堂排查环境差异。
print("NumPy 版本：", np.__version__)
print("SciPy 版本：", scipy.__version__)
print("Matplotlib 版本：", matplotlib.__version__)
```

正文代码 `plot_style.py`

```
"""共用绘图样式：使用当前系统已有的中文字体，不随案例包分发字体。"""

from pathlib import Path

import matplotlib as mpl
from matplotlib import font_manager
from matplotlib.ft2font import FT2Font

# Windows、macOS 和 Linux 的常见字体均列入候选。
# 顺序只影响显示外观，不参与任何随机抽样或数值计算。
CHINESE_FONT_CANDIDATES = (
    "Microsoft YaHei",
    "SimHei",
    "Noto Sans CJK SC",
    "Source Han Sans SC",
    "PingFang SC",
    "Heiti SC",
    "Songti SC",
)
```

```

    "WenQuanYi Zen Hei",
    "Arial Unicode MS",
)

def configure_chinese_plot():
    """在首次创建图形前选择中文字体，并返回实际使用的字体信息。

    变量 available 把字体名称映射到本机文件；选用第一个可用候选。
    返回的字典只供运行报告使用，不改变案例的数据或算法。
    """
    available = {font.name: font.fname for font in font_manager.fontManager.ttflist}
    selected = next(
        (name for name in CHINESE_FONT_CANDIDATES if name in available), None
    )
    if selected is None:
        raise RuntimeError(
            "未找到中文字体。请安装思源黑体或 Noto Sans CJK SC, "
            "然后重新运行案例。案例包不附带系统字体。"
        )
    mpl.rcParams.update({
        "font.family": "sans-serif",
        "font.sans-serif": [selected, "DejaVu Sans"],
        "font.size": 9,
        "axes.titlesize": 10,
        "axes.labelsize": 9,
        "xtick.labelsize": 8,
        "ytick.labelsize": 8,
        "legend.fontsize": 8,
        "axes.unicode_minus": False,
        "savefig.facecolor": "white",
    })
    return {"name": selected, "path": str(Path(available[selected]).resolve())}

def check_font_characters(text):
    """核对所选字体覆盖的中文字符；返回缺字列表，空列表表示全部覆盖。"""
    selected = configure_chinese_plot()
    covered = FT2Font(selected["path"]).get_charmap()
    characters = {char for char in text if "\u3400" <= char <= "\u9fff"}
    return sorted(char for char in characters if ord(char) not in covered)

```

正文代码 g00_knowledge_map.py

```

# 学习路径图：对应教材前言的图 0-1，连接概率与线性代数两条学习路径。
# 本图没有模拟数据或待估计参数；节点坐标只控制版面，不表示概率或数量。
# 运行：python g00_knowledge_map.py；PNG 写入当前工作目录。
# 图内文字采用中文；字体由共用样式模块按当前系统选择。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
from matplotlib.patches import FancyBboxPatch

# 左右列节点依次对应两条课程知识链；每项为中心横坐标、纵坐标和名称。
nodes = {

```

```

    "random": (0.23, 0.86, "随机变量"),
    "distributions": (0.23, 0.65, "概率分布与数学期望"),
    "inference": (0.23, 0.44, "抽样与统计推断"),
    "simulation": (0.23, 0.23, "模拟与不确定性"),
    "vectors": (0.77, 0.86, "向量与线性映射"),
    "equations": (0.77, 0.65, "线性方程组与投影"),
    "decomposition": (0.77, 0.44, "矩阵分解"),
    "applications": (0.77, 0.23, "马尔可夫链、主成分与最小二乘"),
}
# 箭头表示学习先后或应用联系；不表示随机变量之间的因果关系。
edges = [
    ("random", "distributions"), ("distributions", "inference"),
    ("inference", "simulation"), ("vectors", "equations"),
    ("equations", "decomposition"), ("decomposition", "applications"),
    ("simulation", "applications"),
]
# 相对坐标范围固定为 0 到 1；关闭数值坐标轴，突出概念间的联系。
fig, ax = plt.subplots(figsize=(6.3, 4.7), constrained_layout=True)
ax.set(xlim=(0, 1), ylim=(0.08, 1.0))
ax.axis("off")
# 先画箭头，再画节点，避免连线遮住框中文字。
for start, end in edges:
    x1, y1, _ = nodes[start]
    x2, y2, _ = nodes[end]
    # 同列连接上下边缘，跨列连接左右边缘；这些偏移只与框尺寸有关。
    if x1 == x2:
        beginning, ending = (x1, y1 - 0.055), (x2, y2 + 0.055)
    else:
        beginning, ending = (x1 + 0.20, y1), (x2 - 0.20, y2)
    ax.annotate("", xy=ending, xytext=beginning,
                arrowprops={"arrowstyle": "→", "color": "#42677a", "lw": 1.4})
for x, y, label in nodes.values():
    # 圆角框是概念节点；颜色不编码任何统计量。
    box = FancyBboxPatch((x - 0.20, y - 0.055), 0.40, 0.11,
                        boxstyle="round,pad=0.008", facecolor="#eaf0f3",
                        edgecolor="#42677a", linewidth=1.0)
    ax.add_patch(box)
    ax.text(x, y, label, ha="center", va="center", fontsize=8.5)
ax.text(0.23, 0.975, "概率与统计", ha="center", fontsize=11, weight="normal")
ax.text(0.77, 0.975, "线性代数", ha="center", fontsize=11, weight="normal")
# 教材直接嵌入本 PNG；bbox_inches 给标签保留完整边界。
fig.savefig("g00_knowledge_map.png", dpi=180, bbox_inches="tight", pad_inches=0.08)
plt.close(fig)

```

正文代码 run_all.py

"""发现并逐例运行配套程序，把一套图片和逐例日志写入本案例包。

运行: python run_all.py。

本脚本依次执行环境检查、学习路径图和所有 e*.py 案例。

每个子程序使用独立进程和工作目录，彼此不共享变量或随机数状态。

图片最终移至本目录；results 只保存标准输出、错误输出和运行汇总。

"""

```

from datetime import datetime, timezone
from pathlib import Path
import hashlib

```

```

import json
import os
import shutil
import subprocess
import sys
import time

def sha256(path):
    """按文件实际字节计算摘要，用于核对代码和图片的具体版本。"""
    return hashlib.sha256(path.read_bytes()).hexdigest()

def main():
    # __file__ 锚定案例包位置，允许从任意目录启动本工具。
    root = Path(__file__).resolve().parent
    results = root / "results"
    results.mkdir(exist_ok=True)
    env = os.environ.copy()
    env["MPLBACKEND"] = "Agg"
    env["MPLCONFIGDIR"] = str(results / "matplotlib_cache")

    # 文件名决定发现范围；run_all.py 和 plot_style.py 均不在此范围。
    # g00 学习路径图与 e 开头案例保持排序，新增案例无需修改此列表。
    discovered = sorted(root.glob("e*.py"))
    cases = [path for path in discovered if path.name != "environment_check.py"]
    maps = sorted(root.glob("g00*.py"))
    environment = root / "environment_check.py"
    scripts = ([environment] if environment.is_file() else []) + maps + cases
    records = {}
    failed = []
    for script in scripts:
        destination = results / script.stem
        destination.mkdir(exist_ok=True)
        # 仅清理本工具生成的单例临时图，不触碰源目录或其他案例。
        for old_png in destination.glob("*.png"):
            old_png.unlink()
        started = datetime.now(timezone.utc).isoformat()
        clock_start = time.perf_counter()
        try:
            process = subprocess.run(
                [sys.executable, str(script)],
                cwd=destination,
                env=env,
                capture_output=True,
                text=True,
                timeout=180,
                check=False,
            )
            code, stdout, stderr = (
                process.returncode, process.stdout, process.stderr
            )
        except subprocess.TimeoutExpired as exc:
            code = 124
            stdout = exc.stdout or ""
            stderr = exc.stderr or ""
            if isinstance(stdout, bytes):
                stdout = stdout.decode("utf-8", errors="replace")

```

```

        if isinstance(stderr, bytes):
            stderr = stderr.decode("utf-8", errors="replace")
        stderr += "\n 程序运行超过 180 秒。"
    elapsed = time.perf_counter() - clock_start
    (destination / "stdout.txt").write_text(stdout, encoding="utf-8")
    (destination / "stderr.txt").write_text(stderr, encoding="utf-8")
    figures = {}
    for png in sorted(destination.glob("*.png")):
        # 移动而非复制，使案例包只保留根目录这一套可嵌入教材的 PNG。
        final_png = root / png.name
        shutil.move(str(png), str(final_png))
        figures[png.name] = sha256(final_png)
    records[script.name] = {
        "returncode": code,
        "source_sha256": sha256(script),
        "started_at_utc": started,
        "elapsed_seconds": round(elapsed, 4),
        "stdout": stdout,
        "stderr": stderr,
        "figures": figures,
    }
    if code:
        failed.append(script.name)
    print(script.name, "通过" if code == 0 else "失败", flush=True)
    if code:
        print(stderr, flush=True)

report = results / "运行汇总.json"
report.write_text(
    json.dumps(records, ensure_ascii=False, indent=2), encoding="utf-8"
)
passed = sum(records[path.name]["returncode"] == 0 for path in cases)
print(f"教学案例通过 {passed}/{len(cases)}; 失败程序 {len(failed)} 个。")
return int(bool(failed))

if __name__ == "__main__":
    raise SystemExit(main())

```

附录一 教学安排与案例索引

完整课程可安排十八次课，每次九十分钟。一次典型课按问题导入 10 分钟、概念与手算 25 分钟、代码映射与演示 15 分钟、上机练习 30 分钟、解释与反馈 10 分钟组织。需要完整讨论测度与谱分解证明时，另安排拓展研讨；不能把仅展示图像等同于已完成整章教学。

表 0-3 完整课程的教学安排

次序	阅读范围	课堂重点	应提交的学习证据
1	第 1 章	事件、条件与多分类更新	画树状图并解释每个分母
2	第 2 章	期望方差与联合关系	用指示变量和条件期望拆问题
3	第 2 章	LLN 与 CLT	比较均值集中与标准化形状
4	第 3 章	离散分布	按生成机制、支持集选模型
5	第 4 章	连续分布与等待	用面积、单位与独立增量解释
6	第 5 章	抽样区间与检验	区分统计量、参数、p 值
7	第 5 章	贝叶斯更新	完成 Beta 二项与正态共轭
8	第 6 章	向量与子空间	用反例检验封闭性
9	第 6 章	四子空间与秩零度	对照定义、正交关系和维数
10	第 7 章	解结构、消元与敏感性	区分模型条件与数值误差
11	第 7 章	投影、QR、最小二乘	把残差正交写成正规方程
12	第 7 至 8 章	协方差与特征值	连接随机变量方差与矩阵
13	第 8 章	矩阵幂与状态转移	解释稳态与收敛的不同
14	第 9 章	二维到一般 SVD	核对每步空间和因子形状
15	第 9 章	低秩、伪逆、PCA	解释目标函数与误差

次序	阅读范围	课堂重点	应提交的学习证据
16	第 10 章	浓度与计数	区分异质性和过程波动
17	第 10 章	用户留存与画像	完成业务问题到模型解释
18	第 10 章	真实标定综合实践	对照来源、复核计算并说明边界

较短的演示课可以只选入门路线，但应明确未覆盖的证明与拓展范围。每章从概念与手算、编程实践、场景应用三组各选一题作形成性评价；完整上机报告另包含来源、模型假设、变量形状、参数与种子、实际图、解释与限制。不得只提交截图而不说明计算对象。

案例与附件查找表

表 0-4 源文件和正文代码的对应位置

附件文件名	正文位置	正文／附录页
e01_rng.py	第 1 章	26 / 273
e02_expectation.py	第 2 章	35 / 274
e03_lln.py	第 2 章	41 / 275
e04_clt.py	第 2 章	45 / 276
e05_monte_carlo.py	第 2 章	48 / 277
e06_bayes.py	第 1 章	21 / 278
e07_binomial.py	第 3 章	56 / 279
e08_poisson_limit.py	第 3 章	59 / 281
e09_poisson.py	第 3 章	62 / 282
e10_waiting.py	第 3 章	65 / 283
e11_hypergeom.py	第 3 章	69 / 284

附件文件名	正文位置	正文／附录页
e12_simulation_check.py	第 3 章	72 / 285
e13_uniform.py	第 4 章	80 / 286
e14_normal.py	第 4 章	83 / 287
e15_exponential.py	第 4 章	92 / 289
e16_gamma.py	第 4 章	96 / 290
e17_beta_update.py	第 5 章	113 / 292
e18_t_inference.py	第 5 章	109 / 293
e19_distribution_fit.py	第 5 章	123 / 295
e20_vector.py	第 6 章	134 / 297
e21_transforms.py	第 6 章	138 / 298
e22_arrays.py	第 6 章	141 / 300
e23_solve.py	第 7 章	162 / 302
e24_distances.py	第 7 章	158 / 304
e25_eigen.py	第 8 章	185 / 305
e26_diagonalize.py	第 8 章	189 / 307
e27_markov.py	第 8 章	195 / 308
e28_svd.py	第 9 章	209 / 310
e29_lowrank.py	第 9 章	216 / 312
e30_image.py	第 9 章	219 / 313
e31_pca.py	第 9 章	225 / 315

附件文件名	正文位置	正文／附录页
e32_lstsq.py	第 7 章	171 / 317
e33_pm25.py	第 10 章	235 / 319
e34_count.py	第 10 章	240 / 321
e35_land.py	第 10 章	245 / 323
e36_normal_normal.py	第 5 章	118 / 325
e37_poisson_process.py	第 4 章	87 / 326
e38_covariance_matrix.py	第 7 章	176 / 328
e39_svd2d.py	第 9 章	204 / 329
e40_customer_retention.py	第 10 章	250 / 331
e41_customer_pca.py	第 10 章	254 / 332
e42_norris_calibration.py	第 10 章	258 / 334
environment_check.py	工具源码	264 / 335
g00_knowledge_map.py	工具源码	265 / 336
plot_style.py	工具源码	264 / 337
run_all.py	工具源码	266 / 338

附录二 完整注释代码

本附录按文件名集中刊出全部 Python 源码，与正文中的对应代码逐字一致。文件名位于框外，框内没有行号、表头或结果输出，保留原始空行和缩进，可整段复制到编辑器。跨页仍是同一源文件，须连同续页复制；绘图程序与 plot_style.py 保持同级。

程序 e01_rng.py

附件文件 e01_rng.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e01_rng.py

```
# 教材位置：第 1 章，第 1.7 节；图 1-2。
# 单位、变量与形状：a、b、next_a 为 (8,) 的点数数组；点数与概率均无量纲。
# 学习目标：认识 NumPy 的随机数生成器、随机种子和离散均匀抽样，理解怎样复现实验。
# 核心关系：公平骰子满足  $P(X=k)=1/6$ ,  $k=1, \dots, 6$ ；一次抽样得到的是随机样本，
# 不能因为本次某个点数出现较多，就认定其理论概率较大。
# 数据与输出：构造两个具有相同初始状态的生成器，各取 8 个骰子点数，再展示后续抽样。
# 运行方式：安装 NumPy、Matplotlib 后执行 python e01_rng.py；输出终端结果和教材图
# e01_rng.png。

# 使用 np 这个惯用别名，便于调用数组运算和随机数工具。
import numpy as np

# default_rng 返回独立的生成器对象；2026 是种子，不是样本量或概率参数。
# 两个对象从相同种子出发，并执行相同的调用时，会产生相同的伪随机序列。
# 这便于排查代码差异；跨 NumPy 版本或更换算法时，不应无条件承诺序列仍相同。
rng_a = np.random.default_rng(2026)
rng_b = np.random.default_rng(2026)
# integers 的下界 1 包含在内，上界 7 不包含在内，因此可能值恰好是 1 到 6。
# size=8 生成形状为 (8,) 的一维整数数组，而不是形状为 (8, 1) 的列向量。
a = rng_a.integers(1, 7, size=8)
# rng_b 也取自己的前 8 个结果，因此 b 应与 a 逐元素相同。
b = rng_b.integers(1, 7, size=8)
# 生成器会记录并推进内部状态；这里继续使用 rng_a，不会自动从种子重新开始。
# 后续样本可能与前一批有重复点数，这是随机抽样的正常现象。
next_a = rng_a.integers(1, 7, size=8)
# 三行输出依次对比首次抽样、相同种子的首次抽样、同一生成器的下一次抽样。
print("首次抽样: ", a)
print("相同种子的首次抽样: ", b)
print("同一生成器的下一次抽样: ", next_a)
# array_equal 同时检查形状及所有对应元素，验证本例的可复现性。
assert np.array_equal(a, b)
# 数组比较得到布尔数组；& 逐元素合并两个条件，all 要求所有点数都在合法区间。
# 两边条件必须加括号，避免 Python 的运算优先级改变原意。
assert np.all((1 <= a) & (a <= 6))

# 教材图 1-2：绘制公平骰子的完整理论分布，不把上面 8 次抽样的频率当作理论概率。
# pyplot 仅负责作图；这里不调用随机数生成器，因此不会改变原随机序列和终端输出。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
```

```
from plot_style import configure_chinese_plot

configure_chinese_plot()

# dice_values 是随机变量 X 的完整支持集; 理论 概率质量函数 为每个点数对应的 1/6。
dice_values = np.arange(1, 7)
dice_pmf = np.full(dice_values.size, 1 / 6)
# 在 0 与 7 之间画完整累积分布函数; where='post'使每个整数右侧保持当前累计值, 体现右连续性。
cdf_x = np.arange(0, 8)
cdf_y = np.r_[0.0, np.cumsum(dice_pmf), 1.0]
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 左图的纵轴是单点概率; 六根柱高之和为 1。
axes[0].bar(dice_values, dice_pmf, width=0.55)
axes[0].set(title='公平骰子的概率质量函数', xlabel='骰子点数 k', ylabel='P(X = k)',
            xticks=dice_values, ylim=(0, 0.21))
# 右图实心点表示 F(k), 空心点表示从左侧到达 k 时尚未计入该点质量的极限。
axes[1].step(cdf_x, cdf_y, where='post')
axes[1].scatter(dice_values, np.cumsum(dice_pmf), s=18, zorder=3)
axes[1].scatter(dice_values, np.r_[0.0, np.cumsum(dice_pmf)[:-1]],
                s=18, facecolors='white', edgecolors='C0', zorder=4)
axes[1].set(title='右连续的累积分布函数', xlabel='阈值 x', ylabel='P(X <= x)',
            xticks=dice_values, xlim=(0, 7), ylim=(-0.03, 1.05))
# 此文件是教材实际嵌入图, 运行脚本后直接采用, 不另行重新抽样或独立重绘。
fig.savefig('e01_rng.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e02_expectation.py

附件文件 e02_expectation.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e02_expectation.py

```
# 教材位置: 第 2 章, 第 2.3 节; 仅终端输出。
# 单位、变量与形状: x、prob、y 为 (6,); 期望按点数平方计, 方差按点数四次方计。
# 学习目标: 由离散分布精确计算变换后的期望和方差, 并用模拟结果作对照。
# 核心关系: 若 X 为公平骰子点数、Y=X2, 则 E[Y]= $\sum x^2 P(X=x)$ ,
# Var(Y)= $\sum (x^2 - E[Y])^2 P(X=x)$ ; 事件 A 的指示变量满足 E[1_A]=P(A)。
# 数据与输出: 使用 6 个可能点数及其概率, 输出 Y 的理论矩和 50000 次模拟的样本矩,
# 再计算点数为偶数的概率; 理论计算不依赖模拟结果。
# 运行方式: 安装 NumPy 后执行 python e02_expectation.py, 所有结果输出到终端。

import numpy as np

# arange 的终点 7 不包含在内; x 是形状 (6,) 的浮点数组 [1,...,6]。
# 浮点类型便于随后进行加权计算, 数组下标与概率数组的位置一一对应。
x = np.arange(1, 7, dtype=float)
# full 创建 6 个相同的概率质量; 这些数之和为 1, 每个元素表示一个点数的概率。
prob = np.full(6, 1 / 6)
# ** 2 是逐元素平方, 得到 Y 的可能值 [1,4,9,16,25,36]。
# 这里先变换可能值再求期望; 一般来说 E[X2] 并不等于 (E[X])2。
y = x ** 2
# y * prob 是逐元素乘法; 求和实现有限离散分布的期望公式。
mean_y = np.sum(y * prob)
# 先以理论均值为中心, 再按概率加权平方偏差, 计算的是 Y 的总体方差。
# 由于枚举了完整分布, 这里不使用样本方差的 n-1 分母。
var_y = np.sum((y - mean_y) ** 2 * prob)
```

```
# x % 2 == 0 标出偶数点数; astype(float) 将 True/False 转成 1.0/0.0。
# 因此它表示事件“x 为偶数”的指示函数, 而不是事件“y 等于 2”。
even = (x % 2 == 0).astype(float)
# 固定种子使这次模拟可复现; 伪随机数生成器的状态会随着抽样推进。
rng = np.random.default_rng(11)
# 每个骰子点数先抽取、再平方; sample 的形状为 (50000,), 对应 Y 的独立模拟观测。
sample = rng.integers(1, 7, size=50000) ** 2
# 理论值由完整概率分布得到; 模拟值会随随机样本波动, 通常只接近而不完全等于理论值。
print("理论期望 E[Y]、方差 Var(Y): ", mean_y, var_y)
# mean 用样本量作分母; var(ddof=1) 用样本量减 1 作分母, 得到总体方差的无偏估计。
# 无偏指重复抽样时估计量的期望等于真值, 不表示本次估计没有误差。
print("模拟样本的均值、方差: ", sample.mean(), sample.var(ddof=1))
# 对指示函数求加权平均, 得到偶数事件的理论概率 3/6。
print("偶数点数的概率: ", np.sum(even * prob))
# 浮点运算可能有舍入误差, 因此用 isclose 检查理论恒等式, 而不直接比较浮点数相等。
assert np.isclose(mean_y, 91 / 6)
assert np.isclose(var_y, 5369 / 36)
assert np.isclose(np.sum(even * prob), 0.5)
```

程序 e03_lln.py

附件文件 e03_lln.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e03_lln.py

```
# 教材位置: 第 2 章, 第 2.7 节; 图 2-1。
# 单位、变量与形状: rolls、means、n 均为 (20000,); 均值和标准误按骰子点数计。
# 学习目标: 观察大数定律下的累计样本均值, 并区分标准误与一次实验的实际误差。
# 核心关系: 独立同分布且期望有限的样本满足样本均值趋向总体均值; 公平骰子的
#  $E[X]=3.5$ 、 $Var(X)=35/12$ , 所以  $n$  次独立抽样均值的标准误为  $\sqrt{(35/12)/n}$ 。
# 数据与输出: 模拟 20000 次骰子点数, 打印若干累计均值和末尾的理论标准误,
# 并保存累计均值图 e03_lln.png; 图展示的是一次实验的路径。
# 运行方式: 安装 NumPy、Matplotlib 后执行 python e03_lln.py; 图片写入当前工作目录。

import numpy as np
# pyplot 用于创建和保存图形; 本例直接写文件, 不依赖弹出绘图窗口。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 固定种子, 让同一运行环境下的整条累计均值曲线可以复现。
rng = np.random.default_rng(7)
# n=[1,...,20000], 形状为 (20000,); n[i] 对应使用前 i+1 个观测的均值。
n = np.arange(1, 20001)
# 上界 7 不包含在内, 生成与 n 等长的公平骰子样本。
rolls = rng.integers(1, 7, size=n.size)
# cumsum 保存前缀和; 逐元素除以 n, 同一次实验的前 1、2、...、20000 次样本被逐步复用。
# 因此这些累计均值彼此有关联, 并不是每个 n 都重新做一轮独立实验。
means = np.cumsum(rolls) / n
# NumPy 从 0 开始编号, 所以第 10 个累计均值的位置为 9, 其他下标同理。
print("n=10、100、1000、20000 时的累计均值: ", means[[9, 99, 999, 19999]])
# n[-1] 是最后一个样本量; 这里计算的是均值在重复抽样下的标准差。
# 标准误不是本次 |样本均值-3.5|, 也不是每条路径必然遵守的误差上界。
print("n=20000 时的理论标准误: ", np.sqrt((35 / 12) / n[-1]))
```

```
# 枚举所有等概率点数，检查理论均值；该断言不要求随机样本均值恰好等于 3.5。
assert np.isclose(np.mean(np.arange(1, 7)), 3.5)
# 创建一幅 6×3 英寸的图；fig 管理整幅图，ax 管理坐标轴和曲线。
fig, ax = plt.subplots(figsize=(6.3, 3))
# 横轴使用对数刻度，让早期波动和后期稳定趋势都容易观察；样本量本身没有被改写。
ax.semilogx(n, means, label="累计样本均值")
# 用水平虚线标出理论均值，方便比较本次样本路径与极限目标。
ax.axhline(3.5, color="C1", linestyle="--", label="E[X]=3.5")
ax.set(xlabel="样本量 n (对数刻度)", ylabel="均值")
# 大数定律不保证误差随 n 单调下降；曲线短暂偏离目标并不与定律矛盾。
# 图例区分样本路径和理论值，tight_layout 调整边距以减少标签被裁切的可能。
ax.legend()
fig.tight_layout()
# dpi 控制保存图片的像素密度；关闭图形释放资源，不会删除已经保存的文件。
fig.savefig("e03_lln.png", dpi=160)
plt.close(fig)
```

程序 e04_clt.py

附件文件 e04_clt.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e04_clt.py

```
# 教材位置：第 2 章，第 2.8 节；图 2-2。
# 单位、变量与形状：values 为 (4000,n)，means 为 (4000,)；原等待时间用统一时间单位，标准
# 化量无量纲。
# 学习目标：用偏斜的指数分布演示中心极限定理，理解均值的标准化和标准误。
# 核心关系：独立同分布、方差有限且非零时， $\sqrt{n}(\text{样本均值}-\mu)/\sigma$  的分布随 n 增大
# 趋向标准正态分布；本例指数分布的尺度  $\theta=2$ ，均值和标准差均为  $\theta$ 。
# 数据与输出：对 n=1、5、50 各重复 4000 次抽样，打印理论标准误和均值的样本标准差，
# 保存三幅标准化均值直方图与标准正态密度的对照图 e04_clt.png。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e04_clt.py；图片写入当前工作目录
# 。

import numpy as np
# stats 提供理论分布的概率密度函数，用于绘制参照曲线。
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 生成器在后面的循环中持续推进；不会在每个样本量下重新设置种子。
rng = np.random.default_rng(0)
# theta 是尺度参数，不是速率参数；对应的指数分布速率为 1/theta。
# repetitions 是重复实验次数，决定直方图的模拟精细程度；n 决定每次均值使用的样本量。
theta, repetitions = 2.0, 4000
# 生成一行三列子图，并统一坐标尺度，以便直接比较分布形状的变化。
fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.8), sharex=True, sharey=True)
# 标准正态参照曲线的横坐标网格：300 是画线点数，与模拟样本量无关。
z = np.linspace(-4, 4, 300)
# zip 将三个坐标轴分别与三个 n 配对；大 n 下的标准化均值通常更接近正态。
for ax, n in zip(axes, [1, 5, 50]):
    # values 形状为 (4000, n)：每行是一轮实验，每列是一轮中的一个独立观测。
    # NumPy 的 exponential 第一个参数为尺度，因此这里  $E[X]=\theta$ 、 $\text{Var}(X)=\theta^2$ 。
```



```

values = rng.exponential(theta, size=(repetitions, n))
# 沿 axis=1 对每行取平均, 将 (4000, n) 压缩成 (4000,) 的均值数组。
means = values.mean(axis=1)
# 减去总体均值, 再除以理论标准误 theta/sqrt(n); 不是除以原观测的标准差 theta。
# 本例使用已知总体参数, 所以对应中心极限定理中的标准化量, 而不是 t 统计量。
standardized = (means - theta) / (theta / np.sqrt(n))
# density=True 按“频数/(总数×组宽)”归一化, 使所有柱形总面积为 1。
# 柱高是密度而非点概率, 区间概率要看该区间覆盖的面积; bins 控制分组数量。
ax.hist(standardized, bins=40, density=True, alpha=0.6)
# norm.pdf 给出标准正态密度, 不是累积概率; 连续随机变量在单点的概率为 0。
ax.plot(z, stats.norm.pdf(z), color="C1")
# xlim 只限制显示范围, 不会丢弃上述模拟数据, 也不会重新计算标准化结果。
ax.set(title=f"n={n}", xlabel="标准化样本均值", xlim=(-4, 4))
# 对 4000 个均值求样本标准差, 估计“均值的抽样标准差”, 应接近 理论标准误。
# ddof=1 使用 repetitions-1 作分母; 有限次模拟下二者不必完全相等。
print(n, "理论标准误", theta / np.sqrt(n), "均值的样本标准差", means.std(ddof=1))
# 检查每轮恰好产生一个均值, 防止将求平均的轴写错而改变实验含义。
assert means.shape == (repetitions,)
# 中心极限定理 描述的是均值经标准化后的分布; 它不声称原始指数观测变成正态。
# 增加重复次数会使直方图更平滑, 却不能代替增大 n 所带来的正态近似改善。
axes[0].set_ylabel("概率密度")
fig.tight_layout()
# 在当前工作目录保存图片后关闭图形, 便于脚本在无交互环境下连续运行。
fig.savefig("e04_clt.png", dpi=160)
plt.close(fig)

```

程序 e05_monte_carlo.py

附件文件 e05_monte_carlo.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e05_monte_carlo.py

```

# 教材位置: 第 2 章, 第 2.9 节; 仅终端输出。
# 单位、变量与形状: points 为 (100000,2), inside 为 (100000,); 坐标、圆周率估计和标准误
# 均无量纲。
# 学习目标: 通过几何概率估计  $\pi$ , 并理解蒙特卡洛估计的标准误和近似置信区间。
# 核心关系: 在单位正方形均匀抽点, 落入第一象限单位圆内的概率  $q_0=\pi/4$ ;
# 若  $I$  是是否落入圆内的指示变量, 则  $\pi$  估计= $4\times$ 样本均值, 理论标准误为  $4\sqrt{q_0(1-q_0)/n}$ 。
# 数据与输出: 抽取 100000 个二维点, 输出  $\pi$  的估计、估计标准误、近似 95% 区间和实际绝对误
# 差。
# 运行方式: 安装 NumPy 后执行 python e05_monte_carlo.py; 不保存外部数据或图片。

import numpy as np

# 固定种子用于复现实验; n 是独立随机点的个数, 而不是每个坐标轴的网格点数。
rng = np.random.default_rng(3)
n = 100000
# random((n, 2)) 生成形状为 (100000, 2) 的数组, 每行是一个 (x, y)。
# 两个坐标各自均匀分布于 [0,1), 因此这些点均匀覆盖单位正方形。
points = rng.random((n, 2))
# 沿 axis=1 对每行的  $x^2$ 、 $y^2$  求和, 得到到原点距离的平方, 无需平方根。
#  $\leq 1$  判断点是否在四分之一单位圆内, inside 是形状 (n,) 的布尔数组。
# 连续均匀模型下边界概率为 0, 是否包含圆弧不改变理论面积比。
inside = np.sum(points ** 2, axis=1) <= 1
# mean 将 True 当 1、False 当 0, 得到落入圆内的样本比例; q 是对  $q_0$  的估计。
q = inside.mean()

```



```
# 四分之一圆的面积乘以 4 得到整圆面积，因此将样本比例乘以 4 来估计  $\pi$ 。
pi_hat = 4 * q
# 由独立伯努利样本的方差得到比例标准误，再利用常数倍的标准差缩放规则乘以 4。
# 用  $q$  代替未知  $q_0$  得到代入估计；它描述重复抽样时估计量的波动，不等于本次实际误差。
se_hat = 4 * np.sqrt(q * (1 - q) / n)
print("圆周率估计值: ", pi_hat)
print("估计标准误: ", se_hat)
# 大样本下利用正态近似构造 Wald 区间，1.96 是标准正态约 97.5% 分位数。
# [-1,1] 生成左右两个端点；约 95% 指重复实验中这种区间的覆盖率，而非必然包含真值。
# 当命中比例非常接近 0 或 1、或样本过少时，这种近似可能不好；本例命中比例适中且  $n$  大。
print("近似 95% 置信区间: ", pi_hat + np.array([-1, 1]) * 1.96 * se_hat)
#  $\pi$  在本例中已知，所以能直接计算实现后的绝对误差；一般蒙特卡洛问题未必知道真值。
print("绝对误差: ", abs(pi_hat - np.pi))
# 确保每个二维点对应一个判断结果，防止求和方向错误。
assert inside.shape == (n,)
# 将  $q_0=\pi/4$  代入理论标准误并化简，检查两个等价公式；这里未要求模拟估计恰好等于  $\pi$ 。
assert np.isclose(4 * np.sqrt((np.pi / 4) * (1 - np.pi / 4) / n),
                  np.sqrt(np.pi * (4 - np.pi) / n))
```

程序 e06_bayes.py

附件文件 e06_bayes.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e06_bayes.py

```
# 教材位置：第 1 章，第 1.6 节；图 1-1。
# 单位、变量与形状：prior、posterior 为 (4,)；概率无量纲，图中乘 100 表示百分比，期望数量
# 按件计。
# 学习目标：理解阳性检测结果怎样更新次品概率，以及为什么基准次品率会影响后验概率。
# 核心关系：设  $D$  为次品、 $+$  为阳性，则  $P(D|+) = P(+|D)P(D)/P(+)$ ；
#  $P(+)=P(+|D)P(D)+P(+|\text{非}D)P(\text{非}D)$ ，后验优势=先验优势 $\times$ 阳性似然比。
# 数据与输出：对四种假定次品率使用同一组灵敏度、特异度，逐行打印先验和阳性后的后验。
# 这些参数用于教学演示，结果是给定模型下的理论概率，不是从产品抽检数据估计出的产品质量结
# 论。
# 运行方式：安装 NumPy、Matplotlib 后执行 python e06_bayes.py；输出终端结果和教材图
# e06_bayes.png。

import numpy as np

# prior 形状为 (4,)，各元素是不同情景的  $P(D)$ ，并非四个互斥类别的概率。
# 因此这四个数不需要相加为 1；数组运算让四种情景一次完成相同的公式计算。
prior = np.array([0.001, 0.01, 0.05, 0.10])
# 灵敏度= $P(+|D)$ ，特异度= $P(-|\text{非}D)$ ；二者都不能直接解释成  $P(D|+)$ 。
sensitivity, specificity = 0.99, 0.95
# 乘法公式得到联合概率  $P(D \text{ 且 } +)$ ，不是只针对次品群体内部的条件比例。
true_positive = sensitivity * prior
#  $1-\text{specificity}$  为假阳性率  $P(+|\text{非}D)$ ；再乘合格品比例，得到  $P(\text{非}D \text{ 且 } +)$ 。
false_positive = (1 - specificity) * (1 - prior)
# 分母把所有阳性来源相加，得到  $P(+)$ ；分子占其中的比例才是阳性后的次品概率。
# 即使检测参数相同，先验很低时假阳性仍可能占阳性样本中的较大部分。
posterior = true_positive / (true_positive + false_positive)
# 优势= $p/(1-p)$  比较“发生”与“不发生”的概率，不是概率本身，允许大于 1。
odds_prior = prior / (1 - prior)
# 阳性似然比比较阳性结果在次品与合格品两种条件下的可能性，不是后验概率。
ratio = sensitivity / (1 - specificity)
# 贝叶斯公式的优势写法将证据更新化为乘法；标量 ratio 会广播到四个情景。
```

```

odds_post = odds_prior * ratio
# 用 p=优势/(1+优势) 转回概率, 验证概率写法和优势写法逐元素一致。
# allclose 允许浮点舍入误差; 这里只检验两种代数形式, 没有进行随机模拟。
assert np.allclose(posterior, odds_post / (1 + odds_post))
# zip 按对应位置配对先验、后验; 百分比格式只影响显示, 不改变内部计算使用的小数。
for pre, post in zip(prior, posterior):
    print(f"先验={pre:.1%}, 后验={post:.2%}")

# 教材图 1-1 只采用上面已经计算的四种先验情景; 不增加新的检测参数或随机样本。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 左图比较同一检测规则下的先验和阳性后验; 乘 100 只改变显示单位为百分数。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
positions = np.arange(prior.size)
axes[0].bar(positions - 0.18, 100 * prior, width=0.36, label='先验 P(D)')
axes[0].bar(positions + 0.18, 100 * posterior, width=0.36, label='后验 P(D | +)')
axes[0].set(xticks=positions, xticklabels=[f'{value:.1%}' for value in prior],
            xlabel='先验次品率', ylabel='概率 (%)',
            title='四种先验情景', ylim=(0, 85))
# 后验百分数直接来自 posterior; 避免在图中硬写与计算结果分离的数字。
for position, value in zip(positions, posterior):
    axes[0].text(position + 0.18, 100 * value + 1.5, f'{value:.2%}',
                 ha='center', fontsize=7.8)
axes[0].legend(fontsize=7.8, loc='upper left')
# 右图只把第一种先验情景换算为 10 万件产品的期望数量。未把概率改成实际观测频数。
population_size = 100000
positive_counts = population_size * np.array([true_positive[0], false_positive[0]])
axes[1].bar(['真阳性', '假阳性'], positive_counts,
            color=['C1', 'C0'], width=0.55)
for position, value in enumerate(positive_counts):
    axes[1].text(position, value + 100, f'{value:.0f}', ha='center', fontsize=8)
axes[1].set(ylabel='产品期望件数', ylim=(0, 6600),
            title=f'{population_size:,} 件; 先验={prior[0]:.1%}')
# 图中的阳性件数分母与上面 posterior[0] 使用同一个全概率分母。
axes[1].text(0.5, 0.95,
            f'P(D | +) = {positive_counts[0]:.0f} / {positive_counts.sum():.0f}'
            f' = {posterior[0]:.2%}',
            transform=axes[1].transAxes, ha='center', fontsize=7.8)
fig.savefig('e06_bayes.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

程序 e07_binomial.py

附件文件 e07_binomial.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e07_binomial.py

```

# 教材位置: 第 3 章, 第 3.2 节; 图 3-1。
# 单位、变量与形状: k、manual 为 (21,); k 按成功次数计, 概率质量和累积概率均无量纲。
# 学习目标: 从组合计数推导二项分布的概率质量函数, 并核对 SciPy 的分布接口和尾概率端点。
# 核心关系: n 次相互独立、成功概率相同为 p 的伯努利试验中, 成功次数 X 满足
#  $P(X=k) = C(n, k)p^k(1-p)^{(n-k)}$ ,  $E[X] = np$ ,  $Var(X) = np(1-p)$ 。

```

```
# 数据与输出：使用 n=20、p=0.3，枚举全部 21 个可能成功次数，输出理论均值、方差和三个事件
# 的概率。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e07_binomial.py；理论计算不做随
# 机抽样，并保存 e07_binomial.png。

# comb 返回组合数  $C(n, k)$ ，表示从  $n$  个试验位置中选出  $k$  个成功位置的方式数。
from math import comb
import numpy as np
from scipy import stats

# n 为固定试验次数，p 为每次成功概率；只规定这两个参数还不够，二项模型也要求独立性和同一
# p。
n, p = 20, 0.3
# arange 不包含上界，所以用 n+1 保留  $k=n$ ； $k$  的形状为 (21,)，覆盖完整支持集  $0, \dots, 20$ 。
k = np.arange(n + 1)
# 对固定的  $k$ ， $p^k(1-p)^{(n-k)}$  是一种成功/失败排列的概率，再乘组合数汇总所有排列。
# 列表推导式为每个  $j$  计算一个概率，最后转成数组；int(j) 将 NumPy 整数显式转成 Python 整
# 数。
manual = np.array([comb(n, int(j)) * p ** j * (1 - p) ** (n - j) for j in k])
# 将参数固定为一个“冻结分布”对象，后面调用 pmf、cdf、sf 时不必再次传入 n、p。
rv = stats.binom(n=n, p=p)
# pmf(k) 是离散点概率  $P(X=k)$ ，不是连续分布的密度；数组输入返回形状相同的概率数组。
# 两种实现逐项核对，有助于发现组合数、指数或参数传递错误。
assert np.allclose(manual, rv.pmf(k))
# 完整支持集上的概率之和为 1；使用 isclose 容纳浮点计算的舍入误差。
assert np.isclose(manual.sum(), 1)
# 将成功次数乘以对应概率再求和，验证按定义计算的期望等于 np。
assert np.isclose(np.sum(k * manual), n * p)
# 这是分布自身的均值和方差，并非某一组观测的样本均值、样本方差。
print("理论均值、方差：", rv.mean(), rv.var())
# cdf(4)= $P(X \leq 4)$ ，包含端点 4；sf(7)= $P(X > 7)$ = $P(X \geq 8)$ ，所以必须传 7。
# 对离散分布， $P(X \geq k)$  要用 sf(k-1)；直接用 sf(k) 会漏掉  $X=k$  的概率。
# sf 直接计算右尾，通常也比先计算 cdf 再从 1 中相减更适合很小的尾概率。
print("P(X=6), P(X<=4), P(X>=8):", rv.pmf(6), rv.cdf(4), rv.sf(7))

# 教材图 3-1 使用本例唯一的主参数 n=20、p=0.3；两种分布表示来自同一个冻结对象 rv。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# manual 是上面组合公式的结果；折点与柱高使用相同 k，显示完整支持集 0 到 n。
axes[0].bar(k, manual, width=0.7, alpha=0.6, label='组合公式')
axes[0].plot(k, rv.pmf(k), 'o', markersize=3, label='SciPy 概率质量函数')
axes[0].set(title=f'二项分布: n={n}, p={p}', xlabel='成功次数 k',
            ylabel='P(X = k)', xlim=(-0.6, n + 0.6))
axes[0].legend(fontsize=7.8)
# 在支持集左右各加一个点，画出累积分布函数的 0 与 1 平台；阶梯按右连续方式连接。
cdf_k = np.arange(-1, n + 2)
axes[1].step(cdf_k, rv.cdf(cdf_k), where='post')
axes[1].set(title='同一模型的累积分布函数', xlabel='阈值 k',
            ylabel='P(X <= k)', xlim=(-1, n + 1), ylim=(-0.03, 1.05))
fig.savefig('e07_binomial.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e08_poisson_limit.py

附件文件 e08_poisson_limit.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e08_poisson_limit.py

```
# 教材位置：第3章，第3.3节；图3-2。
# 单位、变量与形状：k、target、approx 为 (30,)；横轴为计数，概率质量和近似误差无量纲。
# 学习目标：观察二项分布在“试验次数增大、单次概率减小、均值固定”时的泊松极限。
# 核心关系：固定  $\lambda$ ，令  $p=\lambda/n$ ，当  $n \rightarrow \infty$  时二项分布  $(n, p)$  在每个固定整数  $k$  上的
# 概率趋向  $\text{Poisson}(\lambda)$  的概率；两者均值均为  $\lambda$ ，但二项分布方差是  $\lambda(1-\lambda/n)$ 。
# 数据与输出：取  $\lambda=3$ ，比较  $n=10, 50, 500$  时  $k=0, \dots, 29$  上的最大概率差，
# 再输出没有纳入该网格的泊松右尾概率；本例直接计算理论分布。
# 配套图件：运行后另保存 e08_poisson_limit.png，教材直接采用该文件。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e08_poisson_limit.py，结果输出到
# 终端。

import numpy as np
from scipy import stats

# lam 表示期望计数  $\lambda$ ；使用 lam 作为变量名，因为 lambda 是 Python 的保留字。
lam = 3.0
# k 是形状 (30,) 的整数数组，比较范围包括 0 和 29；泊松分布的支持集实际上没有有限上界。
k = np.arange(30)
# target[j] 对应  $P(\text{Poisson}(3)=k[j])$ ；这里得到概率质量，不是概率密度或累积概率。
target = stats.poisson.pmf(k, mu=lam)
# 每轮改变 n，同时让 p 随 n 改变，不能保持  $p=0.3$  不变后仍声称泊松参数固定为 3。
# 这些 n 均大于  $\lambda$ ，因此  $p=\lambda/n$  落在合法的概率区间内。
for n in [10, 50, 500]:
    # 在相同 k 网格计算二项分布；当  $k>n$  时，其概率为 0，如  $n=10$  时的  $k=11, \dots, 29$ 。
    approx = stats.binom.pmf(k, n=n, p=lam / n)
    # 先逐项取概率差的绝对值，再取最大值；delta 只衡量当前有限网格上的最大点概率差。
    # 它不是全部尾概率的误差，也不是总变差距离，更不是每个 k 上误差的总和。
    delta = np.max(np.abs(approx - target))
    # 格式符只控制对齐和保留小数位，不改变计算所用的 p 或 delta。
    print(f"n={n:3d}, p={lam/n:.4f}, k=0..29 的最大绝对差={delta:.6f}")
    # 验证这三个二项分布的均值都精确对应  $n \times (\lambda/n) = \lambda$ ，仅允许浮点舍入误差。
    assert np.isclose(stats.binom.mean(n, lam / n), lam)
# k[-1]=29；离散分布的  $\text{sf}(29)=P(X>29)=P(X \geq 30)$ ，恰好是网格之外的泊松概率质量。
# 输出它是为了交代截断范围；网格内概率之和不足 1 并不代表分布没有正确归一化。
print("网格外遗漏的泊松尾概率：", stats.poisson.sf(k[-1], mu=lam))

# 教材图 3-2 复用原诊断网格  $k=0, \dots, 29$ ，图上最大差与上面打印值使用完全相同的范围。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.8), constrained_layout=True,
                        sharex=True, sharey=True)
# 每个 n 对应  $p=\lambda/n$ ；理论概率质量函数的重复计算不抽样，不改变任何随机状态或终端结果。
for ax, plot_n in zip(axes, [10, 50, 500]):
    plot_p = lam / plot_n
    plot_binomial = stats.binom.pmf(k, n=plot_n, p=plot_p)
    plot_delta = np.max(np.abs(plot_binomial - target))
    ax.bar(k, plot_binomial, alpha=0.55, label='二项分布')
```

```
ax.plot(k, target, 'o-', color='C1', markersize=2.6, linewidth=1, label='泊松分布')
# 只把主要概率区域放大到 0..12; 最大差的计算范围仍明确写为 0..29。
ax.set(title=f'n={plot_n}, p={plot_p:g}', xlabel='计数 k',
       xlim=(-0.5, 12.5), ylim=(0, 0.29), xticks=[0, 3, 6, 9, 12])
ax.text(0.98, 0.95, f'最大差 (k=0..29) \n{plot_delta:.6f}',
       transform=ax.transAxes, ha='right', va='top', fontsize=7.8)
axes[0].set_ylabel('概率质量')
axes[0].legend(fontsize=7.8, loc='upper right', bbox_to_anchor=(1, 0.75))
# 文件名明确对应本例; 图注需说明显示到 12 与完整计算到 29 的区别。
fig.savefig('e08_poisson_limit.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e09_poisson.py

附件文件 **e09_poisson.py**。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e09_poisson.py

```
# 教材位置: 第 3 章, 第 3.3 节; 仅终端输出。
# 单位、变量与形状: sample 为 (50000,); 观测为单位暴露量中的事件次数, 概率无量纲。
# 学习目标: 核对泊松分布的均值、方差和二阶阶乘矩, 理解无限求和的截断与模拟检验。
# 核心关系:  $X \sim \text{Poisson}(\lambda)$  时  $E[X] = \text{Var}(X) = \lambda$ ,  $E[X(X-1)] = \lambda^2$ ;
# 理论期望本应对  $k=0, 1, \dots$  的全部可能值加权求和, 代码使用高分位数进行有限截断。
# 数据与输出: 取  $\lambda=5$ , 输出精确理论矩、截断求和的矩、遗漏尾概率, 以及 50000 次模拟的样本矩。
# 运行方式: 安装 NumPy、SciPy 后执行 python e09_poisson.py; 结果输出到终端。

import numpy as np
from scipy import stats

# mu 是 SciPy 泊松分布的均值参数; 冻结分布对象保存参数, 供后续计算和抽样复用。
rv = stats.poisson(mu=5)
# ppf 是分位数函数: 寻找使  $P(X \leq \text{upper})$  至少达到给定概率的整数界限。
# 使用接近 1 的分位数是为了把遗漏概率压得很小; int 转成构造整数网格所需的 Python 整数。
upper = int(rv.ppf(1 - 1e-12))
# arange 不包含上界, 因此加 1 使网格覆盖  $0, \dots, \text{upper}$ ; 这是有限数组, 长度为 upper+1。
k = np.arange(upper + 1)
# prob 与 k 位置对应, 保存各整数的理论点概率; 截断后 prob.sum() 会略小于 1。
prob = rv.pmf(k)
# 这是  $\sum_{k=0}^{\text{upper}} kP(X=k)$ , 近似  $E[X]$ ; 没有对截断概率重新归一化。
mean_sum = np.sum(k * prob)
# 这里计算  $X(X-1)$  的期望, 而不是  $X^2$  的期望; 泊松分布的理论值为  $\lambda^2=25$ 。
# 因为  $X^2 = X(X-1) + X$ , 也可以由这两个矩重建二阶原点矩与方差。
factorial_moment = np.sum(k * (k - 1) * prob)
# 固定随机种子, 以复现下面的样本; 理论矩与截断计算不使用随机数。
rng = np.random.default_rng(42)
# rvs 生成 (50000,) 形状的泊松整数观测; 通过 random_state 显式传入生成器。
sample = rv.rvs(size=50000, random_state=rng)
print("理论均值、方差: ", rv.mean(), rv.var())
# 有限和会遗漏尾部的加权贡献; 尾概率很小并不自动给出同样大小的矩误差上界,
# 因为均值和阶乘矩的尾项还分别乘上了 k 和  $k(k-1)$ 。
print("有限求和的均值、二阶阶乘矩  $E[X(X-1)]$ : ", mean_sum, factorial_moment)
# sf(upper)= $P(X > \text{upper})$ , 正好对应本次有限网格外的全部概率; 这里不是  $P(X \geq \text{upper})$ 。
print("遗漏的尾概率: ", rv.sf(upper))
# 模拟均值和方差会随机波动; ddof=1 用样本量减 1 为分母, 估计总体方差。
```



```
print("样本均值、样本方差: ", sample.mean(), sample.var(ddof=1))
# 首个断言核对理论均值和方差; 并未要求有限样本的均值、方差正好等于 5。
assert np.allclose([rv.mean(), rv.var()], [5, 5])
# 网格内的概率加上严格大于 upper 的尾概率应为 1, 检查截断端点没有漏算或重复计算。
assert np.isclose(prob.sum() + rv.sf(upper), 1)
```

程序 e10_waiting.py

附件文件 e10_waiting.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e10_waiting.py

```
# 教材位置: 第 3 章, 第 3.4 节; 图 3-3。
# 单位、变量与形状: total 为 (6, ), 图中总次数已把失败数平移 r; 次数按试验次计, 概率无量
# 纲。
# 学习目标: 理解离散等待时间的无记忆性, 并区分负二项分布记录的失败数和总试验次数。
# 核心关系: 独立伯努利试验的成功概率为 p, 首次成功所需次数  $T \in \{1, 2, \dots\}$ ;
#  $P(T > s+t | T > s) = P(T > t)$ 。若等到第 r 次成功前有 F 次失败, 则总次数  $N = F + r$ ,
#  $E[N] = r/p$ ,  $\text{Var}(N) = \text{Var}(F) = r(1-p)/p^2$ 。
# 数据与输出: 取  $p = 0.3$ ,  $r = 3$ , 验证两组无记忆关系, 打印若干总次数的点概率及两种计数的均值
# 。
# 配套图件: 运行后另保存 e10_waiting.png, 教材直接采用该文件。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后执行 python e10_waiting.py; 所有输出由理论分
# 布计算。

import numpy as np
from scipy import stats

# p 是每次独立试验成功的概率, r 是等待达到的成功总数; 本例中 r 为正整数。
p, r = 0.3, 3
# SciPy 的 geom 计入最后一次成功, 最小可能值是 1; 它不表示首次成功前的失败次数。
geom = stats.geom(p=p)
# SciPy 的 nbinom( $n=r, p=p$ ) 表示第 r 次成功前的失败次数, 支持集从 0 开始。
# 参数名 n 在这里代表目标成功次数 r, 与别处二项分布中“固定试验总数 n”的含义不同。
failures = stats.nbinom(n=r, p=p)
# s 是已经等待且仍未成功的次数, t 是后续比较的额外等待长度; 本例二者均取非负整数。
for s, t in [(3, 2), (5, 4)]:
    #  $\text{sf}(u) = P(T > u) = (1-p)^u$ ;  $T > s+t$  蕴含  $T > s$ , 所以条件概率写成两个尾概率的比值。
    # 分母  $P(T > s) > 0$ ; 条件是“前 s 次都未成功”, 不能换成“第 s 次已经成功”。
    conditional = geom.sf(s + t) / geom.sf(s)
    # 无记忆性说明剩余等待时间分布和从头等待相同, 来自后续伯努利试验与过去独立。
    # isclose 用于数值核对; 该关系不是通过这两组例子才成立, 而是几何分布的理论性质。
    assert np.isclose(conditional, geom.sf(t))
    print("无记忆性 (s、t、条件概率): ", s, t, conditional)
# 至少要做 r 次试验才能获得 r 次成功; 这里只展示 r, ..., r+5 这六种总次数。
# 上界 r+6 不包含在内, 这个展示网格并不包含负二项分布的全部无限支持集。
total = np.arange(r, r + 6)
print("总试验次数: ", total)
# 要计算  $P(N=\text{total})$ , 先将总次数减去固定成功次数 r, 转换为失败数 F。
# 这里 概率质量函数 数组各项为离散点概率; 只打印前六项时, 它们之和通常小于 1。
print("概率质量函数: ", failures.pmf(total - r))
# F 的理论均值是  $r(1-p)/p$ ; 加上 r 次成功, 得到 N 的理论均值  $r/p$ 。
print("平均失败次数、平均总次数: ", failures.mean(), failures.mean() + r)
# 加常数使均值平移、方差不变, 因此同一方差公式同时适用于 F 和 N。
assert np.isclose(failures.mean() + r, r / p)
assert np.isclose(failures.var(), r * (1 - p) / p ** 2)
```

```
# 教材图 3-3 只使用原例 p=0.3、r=3，分别显示首次成功与第 r 次成功两种等待口径。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 几何变量从 1 开始；绘图显示前 20 个可能编号，理论支持集仍为所有正整数。
geom_k = np.arange(1, 21)
axes[0].bar(geom_k, geom.pmf(geom_k), width=0.75)
axes[0].set(title=f'首次成功: p={p}', xlabel='首次成功所在试验次数',
            ylabel='概率质量')
# 总次数从 r 开始。输入 SciPy 时必须减 r，因为 failures 记录的是失败次数。
# 绘图区间比终端列出的 3..8 更宽，仅改变显示范围，没有改变随机变量的定义。
total_plot = np.arange(r, 31)
axes[1].bar(total_plot, failures.pmf(total_plot - r), width=0.8)
axes[1].axvline(failures.mean() + r, color='C1', linestyle='--',
                label=f'平均总次数 = {failures.mean() + r:g}')
axes[1].set(title=f'第 {r} 次成功: p={p}', xlabel='总试验次数',
            ylabel='概率质量')
axes[1].legend(fontsize=7.8)
fig.savefig('e10_waiting.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e11_hypergeom.py

附件文件 e11_hypergeom.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e11_hypergeom.py

```
# 教材位置：第 3 章，第 3.5 节；图 3-4。
# 单位、变量与形状：N、K、n 是对象个数；k 是支持集网格，方差按目标个数平方计。
# 学习目标：比较不放回抽样的超几何分布与独立抽样的二项分布，理解有限总体修正。
# 核心关系：总体 N 个对象中有 K 个成功对象，不放回抽取 n 个时，成功数 X 的
#  $E[X]=nK/N$ ， $\text{Var}(X)=n(K/N)(1-K/N)(N-n)/(N-1)$ ；最后一项是有限总体修正因子。
# 数据与输出：设  $N=50$ 、 $K=5$ 、 $n=10$ ，输出超几何支持集、两种模型的方差及至多一次成功的概率。
# 配套图件：运行后另保存 e11_hypergeom.png，教材直接采用该文件。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后执行 python e11_hypergeom.py；本例只做理论计
# 算，无随机样本。

import numpy as np
from scipy import stats

# 这里的成功对象可以理解为带有某种标记的对象；每个大小为 n 的不放回样本等可能。
# 抽样后总体组成会变化，所以各次“是否成功”的结果不相互独立。
N, K, n = 50, 5, 10
# 特别注意 SciPy 参数命名：M 表示总体数，n 表示总体成功数，N 表示抽取数。
# 因此必须显式映射  $M=N$ 、 $n=K$ 、 $N=n$ ，不能只按照本脚本变量名直接代入。
hyper = stats.hypergeom(M=N, n=K, N=n)
# 二项对照模型把每次试验当成相互独立、成功概率固定为  $K/N$ ，等价于有放回抽样的计数。
# 两种模型的均值相同，但一般概率和方差不同；抽样比例不很小时应谨慎使用二项近似。
binomial = stats.binom(n=n, p=K / N)
# 支持集下界为  $\max(0, n-(N-K))$ ，上界为  $\min(n, K)$ ，本例是  $0, \dots, 5$ 。
# 不放回时最多只能抽出总体已有的 5 个成功对象，而二项模型允许更多成功。
```

```

lower, upper = hyper.support()
# 转成整数后建立包含两端的支持集网格, 形状为 (6,), 用于完整概率归一化检查。
k = np.arange(int(lower), int(upper) + 1)
# 本例修正因子为 40/49, 小于 1; 不放回抽样产生的负相关使计数方差较二项模型小。
# 当 n/N 很小时该因子接近 1; 当抽完整个总体时, 成功数固定为 K、方差为 0。
correction = (N - n) / (N - 1)
print("支持集端点: ", lower, upper)
# 两者都是给定模型的总体方差, 不是样本方差或方差估计的误差。
print("超几何、二项模型的方差: ", hyper.var(), binomial.var())
# cdf(1) 包含端点 1, 即 P(X=0)+P(X=1); 二者差异反映抽样机制不同。
print("P(X<=1):", hyper.cdf(1), binomial.cdf(1))
# hyper.pmf(k) 是完整支持集上的点概率, 求和应为 1。
assert np.isclose(hyper.pmf(k).sum(), 1)
# 理论方差应等于二项方差乘有限总体修正因子, 核对公式与参数映射是否一致。
assert np.isclose(hyper.var(), binomial.var() * correction)

# 教材图 3-4 保留本例 N=50、K=5、n=10, 不另设总体规模或目标比例的演示情景。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8), constrained_layout=True)
# 完整二项支持集为 0..n; 超几何在 k>5 时概率质量函数为 0, 保留这些位置能看见支持集差异。
# 此 plot_k 不替代上面用于检查超几何归一化的支持集数组 k。
plot_k = np.arange(n + 1)
axes[0].bar(plot_k - 0.18, hyper.pmf(plot_k), width=0.36, label='超几何分布')
axes[0].bar(plot_k + 0.18, binomial.pmf(plot_k), width=0.36, label='二项分布')
axes[0].set(title=f'N={N}, K={K}, n={n}', xlabel='目标个数 k',
            ylabel='概率质量', xlim=(-0.7, n + 0.7))
axes[0].legend(fontsize=7.8)
# 右图直接使用两个冻结对象的方差, 展示不放回抽样的有限总体修正。
variances = np.array([hyper.var(), binomial.var()])
axes[1].bar(['超几何分布', '二项分布'], variances, color=['C0', 'C1'], width=0.55)
for position, value in enumerate(variances):
    axes[1].text(position, value + 0.02, f'{value:.4f}', ha='center', fontsize=8)
axes[1].set(title=f'有限总体修正 = {correction:.4f}', ylabel='方差',
            ylim=(0, 1.05 * variances.max() + 0.1))
fig.savefig('e11_hypergeom.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

程序 e12_simulation_check.py

附件文件 e12_simulation_check.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e12_simulation_check.py

```

# 教材位置: 第 3 章, 第 3.6 节; 仅终端输出。
# 单位、变量与形状: sample 为 (50000,), frequency 为 (21,); 事件频率和标准化差异无量纲。
# 学习目标: 用重复模拟核对二项分布的点概率, 并用标准误差衡量模拟差异的通常尺度。
# 核心关系: 独立重复生成 X~二项分布(20,0.3), 令 I=1{X=6}, 则 I~Bernoulli(q),
# q=P(X=6)。count 次重复的样本比例标准误为 sqrt(q(1-q)/count)。
# 数据与输出: 生成 50000 个成功次数, 打印理论/观察概率、概率估计的标准误与标准化差异,
# 以及理论/观察均值; 频数数组用于检查计数方式和归一化是否正确。
# 运行方式: 安装 NumPy、SciPy 后执行 python e12_simulation_check.py, 结果输出到终端。

```



```
import numpy as np
from scipy import stats

# 固定种子，让同一环境下的模拟结果可复现；不同种子产生不同的模拟误差。
rng = np.random.default_rng(11)
# trials=20 是一次二项实验内部的伯努利试验数，count=50000 是整个二项实验的重复次数。
# 概率估计的标准误使用 count；不能误用 trials，因为估计是对 count 个二项观测求比例。
trials, p, count = 20, 0.3, 50000
# sample 的形状为 (50000,)，每个元素是一次独立二项实验中的成功次数，取值为 0,...,20。
sample = rng.binomial(trials, p, size=count)
# bincount 按整数值计数：位置 k 保存 sample 中等于 k 的观测个数。
# minlength=21 保留可能未出现的高端类别；除以 count 后得到经验点概率，和为 1。
# 对离散类别而言这是相对频数，不是连续直方图中的概率密度。
frequency = np.bincount(sample, minlength=trials + 1) / count
# q 由理论 概率质量函数 直接给出，表示“一轮实验恰好有 6 次成功”的概率，不是单次成功概率
# p。
q = stats.binom.pmf(6, trials, p)
# sample==6 形成布尔指示数组；对它求均值就是相应事件的观察比例。
estimate = np.mean(sample == 6)
# 理论 q 在本例已知，因此这里使用的是比例估计量的理论标准误，而不是用 estimate 代入的估计值。
# 它是多次重新生成整个 50000 观测数据集时，estimate 的标准差，不是原始成功次数的标准差。
se = np.sqrt(q * (1 - q) / count)
print("理论概率、模拟频率: ", q, estimate)
# (estimate-q)/se 说明本次有符号差异相当于多少个标准误；其有限样本分布并非严格正态。
# 因指示变量独立同分布且方差有限，大样本下可用 中心极限定理 将这个量近似看作标准正态变量
# 。
print("理论标准误、标准化差异: ", se, (estimate - q) / se)
# 原始成功次数的理论均值为 trials*p；sample.mean() 是对它的另一项模拟核对。
print("理论均值、样本均值: ", trials * p, sample.mean())
# 验证从 0 到 trials 的每个可能成功次数都有一个频率槽位，包括零频数类别。
assert frequency.shape == (trials + 1,)
# 所有观测都恰好归入一个类别，因此经验概率和应为 1，只需允许浮点舍入误差。
assert np.isclose(frequency.sum(), 1)
# 两种统计同一事件的方法应一致；这检查统计实现，没有要求经验概率精确等于理论 q。
# 上述断言通过也不等同于完整的分布拟合检验，仍应结合模拟误差尺度解释理论与观察的差异。
assert np.isclose(frequency[6], estimate)
```

程序 e13_uniform.py

附件文件 e13_uniform.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e13_uniform.py

```
# 教材位置：第 4 章，第 4.3 节；图 4-1。
# 单位、变量与形状：x 为 (6,)；观测使用一般测量单位，密度单位是测量单位的倒数。
# 学习目标：用均匀分布把“概率密度、区间概率、累积分布函数”联系起来。
# 核心数学关系：若  $X \sim \text{Uniform}(a,b)$ ，则区间内密度为  $1/(b-a)$ ，
#  $F(x)$  在  $[a,b]$  内等于  $(x-a)/(b-a)$ ，方差为  $(b-a)^2/12$ 。
# 数据与输出：本例取区间  $[2,6]$ ，打印一点的密度和一段区间的概率，
# 并生成 e13_uniform.png；左图的阴影面积对应概率，右图展示 累积分布函数。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后，在终端运行
# python e13_uniform.py；图片写入运行命令时的当前工作目录。

# NumPy 负责数组和浮点比较，SciPy 提供分布对象，Matplotlib 负责作图。
```

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# SciPy 的 uniform 用 loc 表示左端点, 用 scale 表示区间宽度,
# 因而这里右端点是 loc + scale = 6; scale 不是右端点本身。
# “冻结”后的 rv 保存这组参数, 后续 pdf、cdf、var 都针对同一个分布。
rv = stats.uniform(loc=2, scale=4)
# 长度为 6 的一维浮点数组; 特意覆盖支持区间左侧、内部、边界和右侧。
# rv.cdf 接受整个数组并逐元素返回结果, 不必手写循环。
x = np.array([0, 2, 3, 5, 6, 8], dtype=float)
# pdf(4)=1/4 是密度高度, 而不是 P(X=4); 连续分布在单点的概率为 0。
print("x=4 处的概率密度: ", rv.pdf(4))
# 区间概率是 累积分布函数 的差, 也等于“宽度 2 × 密度 1/4”的面积。
# 连续分布的端点概率为零, 因此这里开区间和闭区间的概率相同。
print("P(3<X<5):", rv.cdf(5) - rv.cdf(3))
# assert 将理论公式变成可执行的自检; 浮点运算用 isclose 而非直接 ==。
assert np.isclose(rv.cdf(5) - rv.cdf(3), 0.5)
# 方差由区间宽度决定: 4^2/12; 平移区间不会改变方差。
assert np.isclose(rv.var(), 16 / 12)
# 创建一行两列子图; axes[0] 和 axes[1] 分别访问左右两个坐标轴。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.8))
# 重复的横坐标 2、6 用于画密度函数的竖直边界, 不代表边界处有概率质量。
axes[0].plot([0, 2, 2, 6, 6, 8], [0, 0, 0.25, 0.25, 0, 0])
# 在 3 到 5 之间填充至密度曲线的面积; alpha 是显示透明度。
axes[0].fill_between([3, 5], [0.25, 0.25], alpha=0.3)
# 累积分布函数 在区间左侧为 0, 内部线性上升, 区间右侧为 1。
# 这些采样点覆盖斜率变化处, 所以连接后的折线能表达本例的完整 累积分布函数。
axes[1].plot(x, rv.cdf(x))
# 标注纵轴, 帮助区分“密度高度”和“累计概率”这两种不同量。
axes[0].set(xlabel="x", ylabel="概率密度")
axes[1].set(xlabel="x", ylabel="累积分布函数")
# tight_layout 调整子图间距; savefig 按 160 dpi 保存图片。
fig.tight_layout()
fig.savefig("e13_uniform.png", dpi=160)
# 释放本图资源, 便于连续运行多个案例。
plt.close(fig)
```

程序 e14_normal.py

附件文件 e14_normal.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e14_normal.py

```
# 教材位置: 第 4 章, 第 4.4 节; 图 4-2。
# 单位、变量与形状:  $\mu$ 、 $\sigma$ 、threshold 采用同一般测量单位; z 无量纲, 原密度单位为测量单位
# 的倒数。
# 学习目标: 理解正态变量的标准化、若干倍标准差的覆盖率及尾部分位点。
# 核心数学关系:  $X \sim N(\mu, \sigma^2)$  时,  $Z = (X - \mu) / \sigma \sim N(0, 1)$ ,
# 因此  $P(X \leq x) = \Phi((x - \mu) / \sigma)$ , 分位数为  $\mu + \sigma * \Phi^{-1}(p)$ 。
# 数据与输出: 使用均值 170、标准差 8 的假想测量模型, 打印阈值 178
# 的累计概率、中心覆盖率, 以及单侧/双侧 95% 对应的临界值; 保存 e14_normal.png。
```

```
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e14_normal.py.

# NumPy 用于浮点自检; SciPy 提供正态分布的 累积分布函数 和逆 累积分布函数。
import numpy as np
from scipy import stats

# sigma 表示标准差, 须为正; 方差为 sigma**2=64。
# threshold 是要评估的原始量纲阈值, 与 mu 使用相同单位。
mu, sigma, threshold = 170.0, 8.0, 178.0
# norm 的 loc 是均值, scale 是标准差, 不能把方差 64 填入 scale。
rv = stats.norm(loc=mu, scale=sigma)
# 标准化后 z=1, 表示阈值位于均值上方一个标准差处。
z = (threshold - mu) / sigma
# 累积分布函数 返回左侧累计概率 P(X<=178), 不是阈值处的密度。
print("标准分数 z、累积概率: ", z, rv.cdf(threshold))
# stats.norm 不指定参数时是标准正态分布; 验证两种计算途径相同。
assert np.isclose(rv.cdf(threshold), stats.norm.cdf(z))
# width 以标准差为单位; 分别计算中心区间 mu±sigma、mu±2sigma、mu±3sigma。
for width in [1, 2, 3]:
    # 累积分布函数 相减给出 P(-width<=Z<=width), 即中心覆盖率, 而非单侧概率。
    coverage = stats.norm.cdf(width) - stats.norm.cdf(-width)
    # :.6% 将 0 到 1 的概率按百分数输出, 并保留六位小数。
    print(f"均值两侧 {width} 个标准差内: {coverage:.6%}")
# ppf 是 累积分布函数 的反函数; 0.95 分位点右侧还剩 5% 的概率质量。
# 此处调用带参数的 rv, 所以结果仍使用原始测量单位。
print("右尾概率 5% 的阈值: ", rv.ppf(0.95))
# 单侧 95% 界把全部 5% 留在一侧, 临界值为 Phi^{-1}(0.95)。
# 双侧中心 95% 区间把 5% 均分为两尾, 右临界值是 Phi^{-1}(0.975),
# 左临界值为其相反数; 两种临界值都处于标准化后的 z 尺度。
print("单侧 95%、双侧 95% 临界值: ",
      stats.norm.ppf(0.95), stats.norm.ppf(0.975))

# 教材图 4-2 把原尺度模型与标准化后的模型并列; 主参数仍是 mu=170、sigma=8、threshold=178。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 原尺度密度网格覆盖均值左右四个标准差; 网格只作显示, 不改变总体参数。
normal_x = np.linspace(mu - 4 * sigma, mu + 4 * sigma, 600)
normal_pdf = rv.pdf(normal_x)
axes[0].plot(normal_x, normal_pdf)
# 阴影是阈值左侧的累计概率; 阈值线与上面计算的 z=1 对应。
axes[0].fill_between(normal_x, normal_pdf, where=normal_x <= threshold,
                    alpha=0.25)
axes[0].axvline(threshold, color='C1', linestyle='--',
                label=f'P(X <= {threshold:g}) = {rv.cdf(threshold):.6f}')
axes[0].set(title=f'正态: 均值={mu:g}, 标准差={sigma:g}',
            xlabel='原尺度数值 x', ylabel='概率密度')
axes[0].legend(fontsize=7.8, loc='upper left')
# 右图的横轴是无量纲 z; 同一幅标准正态密度标出三个中心覆盖区间。
standard_z = np.linspace(-4, 4, 801)
standard_pdf = stats.norm.pdf(standard_z)
for band_width, band_color in [(3, '#C9DFEB'), (2, '#86B5CE'), (1, '#E7A067')]:
    band_coverage = stats.norm.cdf(band_width) - stats.norm.cdf(-band_width)
    axes[1].fill_between(standard_z, standard_pdf,
```

```

        where=np.abs(standard_z) <= band_width,
        color=band_color, alpha=0.9,
        label=f'+/{band_width} 个标准差内: {band_coverage:.2%}')
axes[1].plot(standard_z, standard_pdf, color='C0', linewidth=1)
axes[1].axvline(z, color='C1', linestyle='--', linewidth=1)
axes[1].set(title=r'标准化 $Z=(X-\mu)/\sigma$',
            xlabel='标准分数 z', ylabel='概率密度')
axes[1].legend(fontsize=7.8, loc='upper right')
fig.savefig('e14_normal.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

程序 e15_exponential.py

附件文件 e15_exponential.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e15_exponential.py

```

# 教材位置：第 4 章，第 4.6 节；图 4-4。
# 单位、变量与形状：gaps、arrival_times 为 (5000,)；间隔和时刻按时间计，rate 为每时间单
# 位的事件数。
# 学习目标：从指数分布生成独立事件间隔，累加得到到达时刻，并验证无记忆性。
# 核心数学关系：间隔 T 的速率为 lambda 时， $E[T]=1/\lambda$ ，
#  $P(T>t)=\exp(-\lambda t)$ ，且  $P(T>s+t \mid T>s)=P(T>t)$ 。
# 数据与输出：模拟 5000 个速率为 0.5 的间隔，打印前五个到达时刻、
# 理论/样本平均间隔和条件生存概率；数值自检检查性质，并保存 e15_exponential.png。
# 运行方式：安装 NumPy、SciPy、Matplotlib 后，在终端运行 python e15_exponential.py。

# NumPy 负责随机模拟和累加；SciPy 用于计算理论分布的性质。
import numpy as np
from scipy import stats

# 创建独立随机数生成器；固定种子便于在相同软件环境下复现教学结果。
rng = np.random.default_rng(15)
# rate 的单位是“每单位时间的平均事件数”；这里平均每单位时间 0.5 次。
rate = 0.5
# SciPy 和 NumPy 的指数分布参数都是 scale，即平均等待时间 1/rate=2。
# rate 与 scale 互为倒数；直接把 rate 传给 scale 会得到另一个分布。
rv = stats.expon(scale=1 / rate)
# size=5000 返回形状 (5000,) 的一维数组，每个元素是一段独立等待时间。
gaps = rng.exponential(scale=1 / rate, size=5000)
# 从时刻 0 开始累计间隔：第 i 个元素是第 i+1 次事件发生的绝对时刻。
# 区分“间隔 gaps”与“时刻 arrival_times”，才能正确解释后续统计量。
arrival_times = np.cumsum(gaps)
# [:5] 切片取前五个元素；样本到达时刻受随机间隔影响，不会等距。
print("前五个到达时刻：", arrival_times[:5])
# rv.mean() 是总体理论均值 2；gaps.mean() 是有限次模拟的样本均值，
# 两者接近但无需完全相等，差异来自蒙特卡洛抽样误差。
print("理论平均间隔、样本平均间隔：", rv.mean(), gaps.mean())
# 已等待 s 个时间单位，考察还需等待超过 t 的条件概率。
s, t = 2.0, 1.0
# sf(u)=P(T>u) 是生存函数。因为 {T>s+t} 包含于 {T>s}，
# 条件概率等于两者概率之比；sf 也避免了尾部用 1-cdf 的相减精度损失。
conditional = rv.sf(s + t) / rv.sf(s)
print("条件生存概率：", conditional)
# 无记忆性说明“已等待多久”不会改变剩余等待时间的分布。
# 这是指数分布的性质，并不适用于所有连续等待时间模型。

```

```

assert np.isclose(conditional, rv.sf(t))
# diff 计算相邻到达时刻之差, 长度从 5000 减为 4999。
# 指数间隔理论上几乎必然为正; 本断言检查本次样本的到达时刻严格递增。
assert np.all(np.diff(arrival_times) > 0)

# 教材图 4-4 使用本例已经抽出的 gaps 和 arrival_times, 绝不重新设种子或额外抽样。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图只显示 5000 次到达中的前 8 次; 在最前面补 0 仅用于画第一个等待间隔。
shown_count = 8
shown_times = np.r_[0.0, arrival_times[:shown_count]]
axes[0].hlines(1, 0, shown_times[-1] + 0.5, color='0.5', linewidth=1)
axes[0].vlines(shown_times[1:], 0.9, 1.1, color='C0', linewidth=1.4)
# 前四个带端点的水平线段展示“间隔”与“累计时刻”的区别, 端点完全来自已有数组。
for gap_index in range(4):
    # 很短的相邻间隔可能挤在一起, 错开线段和文字的高度即可避免标签重叠。
    arrow_height = 0.75 - 0.17 * (gap_index % 3)
    # 采用线段和端点短线, 极短间隔也不会因双箭头尺寸超过间隔而交叉。
    axes[0].hlines(arrow_height, shown_times[gap_index], shown_times[gap_index + 1],
                  color=f'C{gap_index % 3}', linewidth=1.3)
    axes[0].vlines(shown_times[gap_index:gap_index + 2],
                  arrow_height - 0.025, arrow_height + 0.025,
                  color=f'C{gap_index % 3}', linewidth=1)
    axes[0].text((shown_times[gap_index + 1] + shown_times[gap_index]) / 2,
                arrow_height - 0.11, rf'$T_{{{{gap_index + 1}}}}$', ha='center',
                fontsize=7.8)
axes[0].set(title=f'{gaps.size:,} 次到达中的前 {shown_count} 次',
            xlabel='到达时刻', yticks=[], ylim=(0.15, 1.35),
            xlim=(0, shown_times[-1] + 0.5))
# 右图使用全部 5000 个原间隔, 与同一 scale=1/rate=2 的理论指数密度比较。
# bins=35 只控制显示分组; density=True 把柱形总面积归一化为 1。
axes[1].hist(gaps, bins=35, density=True, alpha=0.5, label='全部模拟间隔')
exponential_t = np.linspace(0, gaps.max(), 500)
axes[1].plot(exponential_t, rv.pdf(exponential_t), color='C1',
            label=f'指数: 速率={rate:g}, 尺度={1 / rate:g}')
axes[1].set(title='相同的等待时间模型', xlabel='间隔长度 t', ylabel='概率密度')
axes[1].legend(fontsize=7.8)
fig.savefig('e15_exponential.png', dpi=180, bbox_inches='tight')
plt.close(fig)

```

程序 e16_gamma.py

附件文件 e16_gamma.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e16_gamma.py

```

# 教材位置: 第 4 章, 第 4.7 节; 图 4-5。
# 单位、变量与形状: 原间隔数组为 (30000,4), 求和后 waits 为 (30000,); theta 按时间计。
# 学习目标: 理解整数形状参数的伽马分布为何是多个指数等待时间之和,
# 并用 Poisson 计数从另一角度计算第 r 次事件发生的概率。
# 核心数学关系: 若 r 个独立间隔均为 Exponential(scale=theta),

```



```
# 则总等待时间  $T_r \sim$  伽马分布(shape=r, scale=theta),
#  $E[T_r]=r*\theta$ 、 $Var(T_r)=r*\theta^2$ ,  $P(T_r \leq t)=P(N(t) \geq r)$ 。
# 数据与输出: 模拟 30000 次“等到第 4 次事件”的过程, 比较理论/样本矩,
# 并打印四个时刻前已经发生第 4 次事件的概率, 保存 e16_gamma.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e16_gamma.py。

import numpy as np
from scipy import stats

# 固定种子使当前环境中的样本可复现; 每行模拟过程使用不同随机抽样。
rng = np.random.default_rng(21)
# r 是所等待的事件次数, 也是 伽马分布 的形状参数; theta 是每段间隔的
# 平均时间(尺度), 对应事件速率  $\lambda=1/\theta=2$ , 不能把尺度当作速率。
r, theta = 4, 0.5
# 先产生形状 (30000, 4) 的数组: 每行是一轮过程中的四个独立间隔。
# sum(axis=1) 沿列求和, 把每行变成总等待时间, 结果形状为 (30000,)。
# 形状参数为正整数的 伽马分布也称 Erlang 分布。
waits = rng.exponential(scale=theta, size=(30000, r)).sum(axis=1)
# SciPy 用 a 表示形状参数, 用 scale 表示尺度; 此处默认 loc=0。
rv = stats.gamma(a=r, scale=theta)
# 在四个非负时刻同时计算概率, 数组形状为 (4,)。
t = np.array([0.5, 1.0, 2.0, 3.0])
# 对速率  $1/\theta$  的齐次 Poisson 过程,  $N(t)$  的均值为  $t/\theta$ 。
#  $Poisson.sf(r-1)=P(N>r-1)=P(N\geq r)$ , 正好对应“第 r 次事件已到达”。
# 离散分布中必须注意这个  $r-1$ ;  $sf(r)$  会漏掉  $N=r$  的情形。
count_formula = stats.poisson.sf(r - 1, mu=t / theta)
# allclose 逐元素比较后汇总为一个布尔值, 验证四个时刻的等待/计数关系。
assert np.allclose(rv.cdf(t), count_formula)
# 独立间隔之和的均值、方差分别相加, 得到  $r*\theta$  和  $r*\theta^2$ 。
assert np.allclose([rv.mean(), rv.var()], [r * theta, r * theta ** 2])
# 理论矩由分布参数确定, 不受当前模拟样本影响。
print("理论均值、方差: ", rv.mean(), rv.var())
# ddof=1 使用分母 30000-1, 计算总体方差的无偏样本估计;
# 样本均值和方差存在随机误差, 因此不要求它们与理论值完全相等。
print("样本均值、样本方差: ", waits.mean(), waits.var(ddof=1))
# cdf(t) 对数组逐元素计算  $P(T_r \leq t)$ , 输出顺序与 t 中时刻顺序一致。
print("第 r 次事件在 t 前到达的概率: ", rv.cdf(t))

# 教材图 4-5 采用本例  $r=4$ 、 $\theta=0.5$ , 与前例  $\text{rate}=0.5$  的指数模型属于不同参数情景。
# 本图复用已有 30000 个 waits 及原 t 数组; 新增理论曲线计算不进行任何随机抽样。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图把 r 段指数间隔之和的样本密度与对应伽马分布理论密度比较。
wait_grid = np.linspace(0, max(waits.max(), rv.ppf(0.999)), 600)
axes[0].hist(waits, bins=45, density=True, alpha=0.5, label=f'{waits.size:,} 组间隔和')
axes[0].plot(wait_grid, rv.pdf(wait_grid), color='C1',
              label=f'伽马: 形状={r}, 尺度={theta:g}')
axes[0].set(title='指数间隔之和', xlabel='总等待时间', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图两条理论曲线重合: 第 r 次到达已发生, 等价于时刻 t 前计数至少 r 次。
# count_curve 中的 Poisson 参数为时间/ $\theta$ ; SF 传  $r-1$  才能包含计数恰好等于 r 的情况。
cdf_grid = np.linspace(0, rv.ppf(0.999), 400)
```

```
count_curve = stats.poisson.sf(r - 1, mu=cdf_grid / theta)
axes[1].plot(cdf_grid, rv.cdf(cdf_grid), linewidth=2.5, alpha=0.6,
             label='伽马累积分布函数')
axes[1].plot(cdf_grid, count_curve, '--', color='C1', linewidth=1.2,
             label=f'泊松尾概率: 次数 ≥ {r}')
# 四个实心点就是终端打印的 t 数组所对应的 count_formula, 连接图形和数值核。
axes[1].scatter(t, count_formula, color='C2', s=20, zorder=3, label='核对的时刻 t')
axes[1].set(title=r'$P(T_r \leq t) = P(N(t) \geq r)$',
            xlabel='时间 t', ylabel='累积概率', ylim=(-0.03, 1.05))
axes[1].legend(fontsize=7.8)
fig.savefig('e16_gamma.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e17_beta_update.py

附件文件 e17_beta_update.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e17_beta_update.py

```
# 教材位置: 第 5 章, 第 5.7 节; 图 5-2。
# 单位、变量与形状: grid、density 为 (4001,); 成功概率与参数空间无量纲, 区间是未知概率的
# 后验区间。
# 学习目标: 用 贝塔-二项 共轭更新估计未知成功概率, 并用数值积分核。
# 核心数学关系:  $p \sim \text{贝塔分布}(\alpha, \beta)$ 、 $K|p \sim \text{二项分布}(n, p)$  时,
#  $p|K=k \sim \text{贝塔分布}(\alpha+k, \beta+n-k)$ , 后验均值为  $(\alpha+k)/(\alpha+\beta+n)$ 。
# 数据与输出: 先验 贝塔分布 (2,2), 20 次条件独立的同概率试验中成功 15 次;
# 打印解析/积分得到的后验均值和等尾 95% 可信区间, 并保存 e17_beta_update.png。
# 运行方式: 安装支持 np.trapezoid 的 NumPy、SciPy 及 Matplotlib 后,
# 在终端运行 python e17_beta_update.py。

import numpy as np
from scipy import stats

# alpha、beta 为正的先验形状参数; n 是试验数, k 是成功数, 失败数为 n-k。
# 贝塔分布 密度含  $p^{(\alpha-1)}(1-p)^{(\beta-1)}$ , 体现先验对成功率的倾向。
alpha, beta, n, k = 2.0, 2.0, 20, 15
# 对称的 贝塔分布 (2,2) 以 0.5 为中心, 相比均匀先验更偏向中间成功率。
prior = stats.beta(alpha, beta)
# 似然含  $p^k(1-p)^{(n-k)}$ , 与先验相乘后指数相加, 得到 贝塔分布 (17,7)。
# 共轭更新计算的是未知参数 p 的分布, 而不是未来成功次数的分布。
posterior = stats.beta(alpha + k, beta + n - k)
# 用 4001 个等距点覆盖参数空间 [0,1]; 一维数组形状为 (4001,)。
# 这里积分较平滑且端点密度为零, 细网格能给出很精确的数值核。
grid = np.linspace(0, 1, 4001)
# 逐个候选 p 计算“先验密度 × 观测到恰好 k 次成功的似然”。
# binom.pmf 的第三个参数在这里是整条 grid, 所以输出与网格形状相同。
# 得到的函数尚未归一化, 其关于 p 的积分是观测数据的边际概率。
unnormalized = prior.pdf(grid) * stats.binom.pmf(k, n, grid)
# trapezoid(y,x) 用梯形法近似积分; 除以总面积后得到积分约为 1 的后验密度。
# 连续密度必须按网格间距积分, 不能直接把高度的和当作精确的积分。
density = unnormalized / np.trapezoid(unnormalized, grid)
# 连续后验的期望是  $\int p f(p|data) dp$ ; 这里使用同一网格数值近似。
grid_mean = np.trapezoid(grid * density, grid)
print("解析后验均值、积分后验均值: ", posterior.mean(), grid_mean)
# ppf 接收两个左侧累计概率, 给出各留 2.5% 后验质量的等尾可信区间。
# 在本先验和模型下, 参数落在该区间内的后验概率为 95%。
```

```
print("95% 可信区间: ", posterior.ppf([0.025, 0.975]))
# 解析均值 17/(17+7)=17/24; 后验均值同时融合先验与样本成功率 15/20。
assert np.isclose(posterior.mean(), 17 / 24)
# 积分存在离散化误差, 用绝对容差 atol=1e-6 检查数值结果足够接近解析值。
assert np.isclose(grid_mean, posterior.mean(), atol=1e-6)

# 教材图 5-2 中的未知成功率用数学符号 theta 表示; SciPy 的 binom 接口把同一参数称为 p。
# 原数组 grid 的每个元素都是一个候选 theta, 既不是观测次数 k, 也不是新的随机样本。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 原积分网格在本例端点密度为 0, 可直接用于画先验及解析后验, 不另设隐藏网格参数。
prior_label = f'先验贝塔({alpha:g}, {beta:g})'
posterior_label = f'后验贝塔({alpha + k:g}, {beta + n - k:g})'
axes[0].plot(grid, prior.pdf(grid), label=prior_label)
axes[0].plot(grid, posterior.pdf(grid), color='C1', label=posterior_label)
# 等尾可信区间直接采用本例 posterior 的分位数, 阴影面积约为 95% 的后验质量。
credible_lower, credible_upper = posterior.ppf([0.025, 0.975])
axes[0].fill_between(grid, posterior.pdf(grid),
                     where=(grid >= credible_lower) & (grid <= credible_upper),
                     color='C1', alpha=0.2, label='95% 可信区间')
axes[0].axvline(posterior.mean(), color='C1', linestyle='--', linewidth=1)
axes[0].set(title=f'共轭更新: n={n}, k={k}',
            xlabel=r'未知成功概率 $\theta$', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图对照解析共轭后验与原数值积分的归一化密度; 二者接近重合是实现核查。
axes[1].plot(grid, posterior.pdf(grid), linewidth=2.5, alpha=0.6,
            label='解析后验')
axes[1].plot(grid, density, '--', color='C2', linewidth=1.2,
            label='归一化网格密度')
axes[1].set(title='数值积分核对',
            xlabel=r'候选 $\theta$ (接口参数 p)', ylabel='后验密度')
axes[1].legend(fontsize=7.8)
fig.savefig('e17_beta_update.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e18_t_inference.py

附件文件 e18_t_inference.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e18_t_inference.py

```
# 教材位置: 第 5 章, 第 5.6 节; 图 5-1。
# 单位、变量与形状: 样本按同一般测量单位计; 均值区间保留该单位, t 统计量与 p 值无量纲
# 。
# 学习目标: 手算单样本双侧 t 检验和均值置信区间, 并与 SciPy 的实现核对。
# 核心数学关系: 样本标准差为 s 时, SE=s/sqrt(n),
# t=(样本均值-mu0)/SE; 对正态总体的独立同分布样本,
# 该统计量在原假设下服从 t_(n-1)。
# 前提: 总体标准差未知, 观测应独立; n=9 很小, 精确 t 推断依赖总体正态性,
# 数据若严重偏斜或有异常值, 不能仅因使用 t 分布就认为推断可靠。
# 数据与输出: 九个假想测量值, 检验总体均值是否为 170; 打印均值、t 值、
```



```
# 双侧 p 值和 95% 置信区间, 并保存 e18_t_inference.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 在终端运行 python e18_t_inference.py。

import numpy as np
from scipy import stats

# 一维数组中每个元素代表一个观测, dtype=float 便于后续连续量计算。
x = np.array([168, 172, 175, 169, 171, 173, 170, 174, 167], dtype=float)
# mu0 是原假设 H0: mu=170 的均值; alpha=0.05 是预先选定的显著性水平。
# 此处备择假设是 mu!=170, 因此需要考虑正负两个方向的偏离。
mu0, alpha = 170.0, 0.05
# size 返回观测总数; 此处 x 是一维, 所以也等于数组长度。
n = x.size
# ddof=1 使样本方差分母为 n-1; std 再开平方得到样本标准差 s。
# SE 描述样本均值的抽样波动, 和描述个体离散程度的 s 是不同的量。
se = x.std(ddof=1) / np.sqrt(n)
# 标准化均值与原假设的差异; 正 t 值表示样本均值大于 mu0。
# 用样本估计标准差后有 n-1=8 个自由度, 所以采用 t 分布。
t_value = (x.mean() - mu0) / se
# t 分布关于 0 对称; 先算 |t| 右侧尾概率, 再乘 2 得到双侧 p 值。
# p 值是在 H0 和模型前提下, 得到至少这么极端统计量的概率,
# 并不是“原假设为真”的概率; sf 用于直接计算右尾概率。
p_value = 2 * stats.t.sf(abs(t_value), df=n - 1)
# 双侧中心 95% 对应每尾 alpha/2; ppf 返回右侧正临界值。
critical = stats.t.ppf(1 - alpha / 2, df=n - 1)
# [-1, 1] 与标量相乘, 一次生成“均值-误差界、均值+误差界”两个端点。
# 95% 置信度是重复抽样时该构造方法覆盖固定总体均值的长期比例,
# 不能把它直接解释成已得到的固定区间对固定参数的后验概率。
interval = x.mean() + np.array([-1, 1]) * critical * se
# SciPy 默认做双侧单样本 t 检验; popmean 指定原假设的总体均值。
result = stats.ttest_1samp(x, popmean=mu0)
print("样本均值、t 统计量、p 值: ", x.mean(), t_value, p_value)
print("95% 置信区间: ", interval)
# 分别比较统计量、双侧 p 值和区间端点, 防止手工推导出现自由度或单双侧错误。
# 浮点运算存在舍入误差, 用 isclose/allclose 进行近似比较。
assert np.isclose(t_value, result.statistic)
assert np.isclose(p_value, result.pvalue)
# confidence_interval() 默认置信水平为 0.95, 恰好与本例的 1-alpha 相同。
assert np.allclose(interval, result.confidence_interval())

# 教材图 5-1 使用原九个观测及本次计算的置信区间; 总体均值 mu 和总体标准差均未知。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

fig, axes = plt.subplots(1, 2, figsize=(6.3, 2.9), constrained_layout=True)
# 左图只画本例检验实际采用的 n-1 自由度 t 密度, 并用标准正态密度作尾部参照。
t_grid = np.linspace(-5, 5, 500)
axes[0].plot(t_grid, stats.t.pdf(t_grid, df=n - 1), label=f't 分布: 自由度={n - 1}')
axes[0].plot(t_grid, stats.norm.pdf(t_grid), '--', label='标准正态')
axes[0].axvline(t_value, color='C2', linestyle=':', label=f'观察 t={t_value:.3f}')
axes[0].set(title='原假设下的参考分布', xlabel='t / z', ylabel='概率密度')
axes[0].legend(fontsize=7.8)
# 右图误差条的端点直接取 interval, 不另写自由度 8 或分母 3, 避免数据变化后两处失配。
mean_x = x.mean()
ci_errors = np.array([mean_x - interval[0], [interval[1] - mean_x]])
```

```
axes[1].errorbar(mean_x, 0.6, xerr=ci_errors, fmt='o', capsize=5)
axes[1].axvline(mu0, color='C1', linestyle='--', label=rf'$H_0: \mu=\mu_0$')
axes[1].text(mean_x, 0.82, f'样本均值 = {mean_x:.2f}', ha='center', fontsize=8)
axes[1].text(mean_x, 0.34,
             f'{1 - alpha:.0%} 置信区间 = [{interval[0]:.2f}, {interval[1]:.2f}]',
             ha='center', fontsize=7.8)
axes[1].set(title=f'模拟样本: n={n}', xlabel='总体均值尺度',
           yticks=[], ylim=(0.15, 1.05),
           xlim=(min(interval[0], mu0) - 0.7, max(interval[1], mu0) + 0.7))
axes[1].legend(fontsize=7.8, loc='upper right')
fig.savefig('e18_t_inference.png', dpi=180, bbox_inches='tight')
plt.close(fig)
```

程序 e19_distribution_fit.py

附件文件 e19_distribution_fit.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e19_distribution_fit.py

```
# 教材位置: 第 5 章, 第 5.9 节; 图 5-4。
# 单位、变量与形状: 各候选分布用相同正值数据; 观测单位任意但统一, 对数表示采用该单位后的
# 数值。
# 学习目标: 用极大似然拟合多个连续分布, 用 赤池信息准则 比较候选模型,
# 并理解“参数来自同一份数据”时为何需要校准拟合优度检验。
# 核心数学关系:  $\log L = \sum (\log f(x_i; \theta_{\text{hat}}))$ , 赤池信息准则  $= 2 * k - 2 * \log L$ ,
# 其中  $k$  只计算自由估计的参数; 同一数据/同一观测尺度下通常偏好较小的 赤池信息准则。
# 拟合优度检验 统计量衡量经验 累积分布函数 与拟合 累积分布函数 的最大差异; 估计参数会改
# 变其零假设分布。
# 数据与输出: 生成 150 个正值, 打印正态/伽马分布/对数正态的拟合参数和 赤池信息准则,
# 打印对数数据的正态族 拟合优度检验 统计量及校准 p 值, 保存 e19_fit.png。
# 运行方式: 安装 NumPy、SciPy、Matplotlib 后, 运行 python e19_distribution_fit.py;
# 图片写入运行命令时的当前工作目录, 蒙特卡洛校准需要比简单拟合更多计算。

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()

# 固定模拟种子, 使同一环境中的教学数据可重复; 校准稍后使用独立的种子。
rng = np.random.default_rng(30)
# lognormal 的 mean 和 sigma 是  $\log(X)$  的均值与标准差,
# 不是  $X$  本身的均值和标准差; 此处原尺度中位数为  $\exp(\log(5))=5$ 。
# 返回形状 (150,) 的正值数组, 模拟独立同分布观测。
data = rng.lognormal(mean=np.log(5), sigma=0.9, size=150)
# 每项包括显示名称、SciPy 分布族、拟合时固定的参数字典。
# floc=0 固定位置参数, 令 伽马分布 和对数正态的支持区间从 0 开始;
# 正态分布允许负值, 能否接受这种支持范围还需结合实际判断。
models = [("正态分布", stats.norm, {}), ("伽马分布", stats.gamma, {"floc": 0}),
          ("对数正态分布", stats.lognorm, {"floc": 0})]
# 左图叠加样本直方图和拟合密度, 右图检查对数数据的正态概率图关系。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3))
# density=True 把直方图归一化为总面积 1, 才能与连续密度曲线比较;
# bins=25 控制分箱数, alpha=0.4 让重叠曲线仍然可见。
```

```

axes[0].hist(data, bins=25, density=True, alpha=0.4)
# 在正数网格上画密度，避免在 0 处触及某些分布的边界行为。
# 网格只用于显示，不改变拟合时使用的完整原始数据。
x = np.linspace(0.01, data.max(), 300)
# Python 循环同时解包每个候选模型的三项配置。
for name, family, fixed in models:
    # **fixed 将字典展开为关键字参数；本例各分布按极大似然拟合。
    # 返回元组依次含形状参数（若有）、loc、scale，固定参数也包含在元组中。
    parameters = family.fit(data, **fixed)
    # *parameters 按位置展开拟合结果，生成可直接计算 pdf/logpdf 的分布对象。
    rv = family(*parameters)
    # 独立观测的联合密度为各密度乘积；取对数后可求和，
    # 相比直接连乘大量很小的密度值，logpdf 能降低数值下溢风险。
    loglike = rv.logpdf(data).sum()
    # 本例 fixed 仅包含 floc，因此减去字典长度恰好去掉固定位置参数。
    # 正态为均值/尺度两个参数；伽马分布 和对数正态各自自由估计形状/尺度两个。
    # 这种计数写法依赖当前配置，若以后添加非固定参数的拟合选项须重新判断。
    free_count = len(parameters) - len(fixed)
    # 赤池信息准则 在拟合优度与参数复杂度之间权衡；它是候选模型的相对比较指标，
    # 不是“模型正确概率”，也不能仅据最小 赤池信息准则 证明分布假设成立。
    print(name, "参数", parameters, "赤池信息准则", 2 * free_count - 2 * loglike)
    # 确保总对数似然既不是无穷也不是 NaN，避免拿无效值作模型比较。
    assert np.isfinite(loglike)
    # 在同一原始数据尺度绘制各模型密度，label 供后续图例识别。
    axes[0].plot(x, rv.pdf(x), label=name)
# 对数正态模型等价于 log(X) 服从正态分布；数据均为正才能安全取对数。
# 检验的是“某个均值/标准差的正态分布族”，这两个参数由当前 log 数据估计。
# 若把估计参数当成预先已知参数，直接套用经典 拟合优度检验 p 值，校准会不正确。
# goodness_of_fit 从拟合零假设生成样本、每次重新估计未知参数并重算统计量，
# 用这套参数自助法模拟的参考分布进行校准；这里用 1999 次蒙特卡洛样本。
# 固定独立种子 31 便于复现；有限模拟次数使 p 值本身仍有蒙特卡洛误差。
result = stats.goodness_of_fit(stats.norm, np.log(data), statistic="ks",
                                n_mc_samples=1999, rng=np.random.default_rng(31))
# 校准 p 值针对上述复合原假设，不应与各模型 赤池信息准则 的数值大小直接比较。
# 较大 p 值只表示当前检验未发现充分偏离证据，不能证明正态性。
print("对数数据正态族的检验统计量、校准 p 值：",
      result.statistic, result.pvalue)
# 正态概率图比较排序后的 log 数据与理论顺序统计量中位数；近似直线支持该形状，
# 两端偏离可提示尾部差异；图形诊断和统计检验提供互补信息。
stats.probplot(np.log(data), dist="norm", plot=axes[1])
# 添加坐标标签和图例；右图标题明确使用的是对数尺度数据。
axes[0].set(xlabel="正值观测", ylabel="概率密度")
axes[0].legend()
axes[1].set(title="对数数据的正态概率图",
            xlabel="理论顺序统计量中位数", ylabel="排序后的对数观测")
# 调整布局、保存 160 dpi 图片并关闭图，便于批量运行案例时释放内存。
fig.tight_layout()
fig.savefig("e19_fit.png", dpi=160)
plt.close(fig)

```

程序 e20_vector.py

附件文件 e20_vector.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e20_vector.py

```
# 教材位置：第 6 章，第 6.2 节；图 6-1。
# 单位、变量与形状：a、b、c、p、r 均为 (2,); 坐标使用统一长度单位，内积按长度平方计。
# 学习目标：计算二维向量的线性组合、内积、欧氏长度和正交投影，
# 并用图形与勾股关系理解投影残差。
# 核心数学关系：proj_a(c)=((c·a)/(a·a))*a, residual=c-proj_a(c),
# 且 a·residual=0, ||c||^2=||proj_a(c)||^2+||residual||^2 (要求 a 非零)。
# 数据与输出：使用两个二维向量及其线性组合，打印和、内积、长度、
# 投影向量和残差，保存 e20_vector.png。
# 运行方式：安装 NumPy、Matplotlib 后，运行 python e20_vector.py;
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# a、b 都是形状 (2,) 的一维数组，按两个坐标表示向量；
# 它们没有显式的“行向量/列向量”第二维，@ 在这里执行向量内积。
a = np.array([2.0, 1.0])
b = np.array([-1.0, 2.0])
# 线性组合逐坐标进行：c=1.5*a-0.5*b=[3.5,0.5]。
# 系数的正负决定沿该向量同向或反向移动，绝对值决定伸缩比例。
c = 1.5 * a - 0.5 * b
# c@a 是内积，a@a=||a||^2；这个比值是投影沿 a 的伸缩系数。
# projection 是二维投影向量，并不是一个“投影长度”标量；
# 沿单位方向 a/||a|| 的有符号标量分量为 (c@a)/||a||，
# 投影向量的长度则为 abs(c@a)/||a||。此处 a 非零，分母有定义。
projection = (c @ a) / (a @ a) * a
# residual 是从投影点指向 c 的向量；正交投影使这部分与 a 垂直。
residual = c - projection
print("向量和 a+b =", a + b, "; 线性组合 c =", c)
# norm(a) 默认求欧氏长度 sqrt(a@a); round 仅调整显示精度。
print("内积 a^T b =", a @ b, "; 二范数 ||a||_2 =", round(np.linalg.norm(a), 6))
print("投影向量 =", projection, "; 残差向量 =", residual)
# allclose 用来比较向量各分量，isclose 用来比较标量，允许浮点舍入误差。
assert np.allclose(c, [3.5, 0.5])
# a 与 b 正交；同时验证残差确实与投影所在直线的方向 a 正交。
assert np.isclose(a @ b, 0) and np.isclose(a @ residual, 0)
# 向量与自身的内积是长度平方；此式就是正交分解后的勾股定理。
assert np.isclose(c @ c, projection @ projection + residual @ residual)
# 同一批已计算向量分成两个面板：左侧解释首尾相接，右侧解释正交分解。
# 两图直接使用 a、b、c、projection、residual，不另外输入一套向量数值。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1))
for ax in axes:
    # equal 让两个坐标轴单位长度一致，否则图上的直角与长度比例可能失真。
    ax.set_aspect("equal")
    ax.grid(alpha=0.3)
    ax.axhline(0, color="gray", lw=0.5)
    ax.axvline(0, color="gray", lw=0.5)
    ax.set(xlim=(-1.6, 4.4), ylim=(-0.5, 3), xlabel="$x_1$", ylabel="$x_2$")
```

```
# xy 是箭头终点, xytext 是起点; 空字符串表示只画箭头, 不附加正文。
# 向量平移不改变其分量; 第二条组合箭头从 projection 走到 c, 位移为 residual。
ax = axes[0]
for start, end, label, color in [
    (np.zeros(2), a, "$a$", "gray"),
    (np.zeros(2), b, "$b$", "gray"),
    (np.zeros(2), projection, "$1.5a$", "tab:green"),
    (projection, c, "$-0.5b$", "tab:orange"),
    (np.zeros(2), c, "$c$", "tab:blue"),
]:
    ax.annotate("", xy=end, xytext=start, arrowprops={"arrowstyle": "->",
        "color": color})
    # end[0]、end[1] 分别是横纵坐标; 稍微偏移标签, 避免遮住箭头尖端。
    if label == "$-0.5b$":
        ax.text(end[0] + 0.08, end[1] - 0.35, label, color=color)
    else:
        ax.text(end[0] + 0.08, end[1] + 0.08, label, color=color)
ax.set_title("向量的线性组合")
ax = axes[1]
# 每次取一个向量和对应标签, 统一从原点画箭头, 便于比较方向和长度。
for v, label, color in [(c, "$c$", "tab:blue"), (projection, "$p$", "tab:green")]:
    ax.annotate("", xy=v, xytext=(0, 0), arrowprops={"arrowstyle": "->",
        "color": color})
    ax.text(v[0] + 0.08, v[1] + 0.08, label, color=color)
# 红色虚线连接 projection 与 c, 其位移正好是 residual;
# 残差箭头虽未从原点画出, 但向量只取决于位移, 平移不会改变它。
ax.plot([projection[0], c[0]], [projection[1], c[1]], "r--", label="残差 $r$")
ax.annotate("", xy=c, xytext=projection, arrowprops={"arrowstyle": "->",
    "color": "red"})
# 沿 a 和 residual 的单位方向画直角记号; 本例两方向均非零且正交。
u = a / np.linalg.norm(a)
w = residual / np.linalg.norm(residual)
right_angle = np.array([projection - 0.18 * u,
    projection - 0.18 * u + 0.18 * w,
    projection + 0.18 * w])
ax.plot(*right_angle.T, color="gray", lw=0.8)
ax.set_title("向量的正交投影")
ax.text(0.02, 0.04, "正交投影 p; $a^T r=0$", transform=ax.transAxes, fontsize=8)
ax.legend(fontsize=8, loc="upper left")
# 指定显示范围, 保证所有箭头和标签位于图内。
# 调整布局后保存并关闭, 避免批量作图时保留多余图对象。
fig.tight_layout()
fig.savefig("e20_vector.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e21_transforms.py

附件文件 e21_transforms.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e21_transforms.py

```
# 教材位置: 第 6 章, 第 6.3 节; 图 6-2。
# 单位、变量与形状: 各变换矩阵均为 (2,2); 坐标使用统一长度单位, 变换系数无量纲。
# 学习目标: 把二维矩阵看作旋转、缩放、剪切等线性变换,
# 理解行列式的面积意义, 以及矩阵复合时顺序为何重要。
# 核心数学关系: 列向量变换为  $y=MX$ ; 若多个点按行存放, 统一变换写作
```



```
# points @ M.T; 面积放大倍数为 abs(det(M)), 一般有  $S@R \neq R@S$ 。
# 数据与输出: 对中心位于原点的正方形应用四种变换, 打印行列式,
# 保存 e21_transforms.png; 所有矩阵都是形状 (2,2)。
# 运行方式: 安装 NumPy、Matplotlib 后, 运行 python e21_transforms.py;
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 三角函数使用弧度, 因此先把 30 度转换为  $\pi/6$ 。
theta = np.deg2rad(30)
# 标准二维旋转矩阵把列向量逆时针旋转 theta, 保持长度和夹角不变。
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
# diag 从一维元素生成对角矩阵: 横坐标放大 2 倍, 纵坐标缩小为 0.6 倍。
# 两方向伸缩不同, 因此一般不保持角度, 也不会与旋转矩阵交换。
S = np.diag([2.0, 0.6])
# 剪切给出  $x'=x+0.8*y$ 、 $y'=y$ ; 水平移动量随原来的高度 y 变化。
H = np.array([[1.0, 0.8], [0.0, 1.0]])
# eye(2) 是单位矩阵, 对应“保持原图”; 两个列表按相同顺序配对。
matrices = [np.eye(2), R, S, H]
names = ["恒等变换", "旋转变换", "缩放变换", "剪切变换"]
# 五行坐标包含四个顶点和重复的起点, 确保轮廓闭合, 形状为 (5,2)。
# 标量 0.5 广播到所有坐标, 使方形中心从 (0.5,0.5) 平移到原点。
# 这一步是构造输入图形; 后续 M 本身仍是关于原点的线性变换。
points = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]]) - 0.5
# 列表推导对每个矩阵求行列式; round 仅用于让打印结果更易读。
# det 的绝对值是面积倍数, 符号表示是否翻转方向; 本例行列式都为正。
print("各变换矩阵的行列式 =", np.round([np.linalg.det(M) for M in matrices], 6))
# R.T 表示转置;  $R.T@R=I$  验证旋转矩阵的列是单位正交向量。
assert np.allclose(R.T @ R, np.eye(2))
# 单位变换/旋转/剪切保持面积; 缩放面积倍数为  $2*0.6=1.2$ 。
assert np.allclose([np.linalg.det(M) for M in matrices], [1, 1, 1.2, 1])
# 在列向量约定下  $S@R$  表示先旋转后缩放;  $R@S$  表示先缩放后旋转。
# 这两个结果不同, 说明矩阵乘法通常不满足交换律。
assert not np.allclose(S @ R, R @ S)
fig, axes = plt.subplots(1, 4, figsize=(6.3, 2.8))
# zip 按位置把坐标轴、矩阵、名称组合起来, 逐一绘制四种变换。
for ax, M, name in zip(axes, matrices, names):
    # 每个点存为一行 p, 列向量公式  $M@p$  对应的行形式是  $p@M.T$ 。
    # (5,2)@(2,2) 得到 (5,2), 一次完成五个点的同一变换。
    transformed = points @ M.T
    #[:,0] 选所有点的横坐标,[:,1] 选纵坐标; 黑色虚线作原图参照。
    ax.plot(points[:, 0], points[:, 1], "k--", alpha=0.4)
    # fill 把变换后的顶点连成填充多边形, 透明度便于观察和原图的重叠。
    ax.fill(transformed[:, 0], transformed[:, 1], alpha=0.4)
    # 四个子图使用同样的范围, 避免自动缩放掩盖真实伸缩差异。
    ax.set(xlim=(-1.5, 1.5), ylim=(-1.5, 1.5), title=name)
    # 坐标轴等比例显示, 才能直观看出旋转的角度与形状是否保持。
    ax.set_aspect("equal")
    ax.grid(alpha=0.2)
# 保存 180 dpi 图片并关闭图, 释放资源。
fig.tight_layout()
fig.savefig("e21_transforms.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e22_arrays.py

附件文件 e22_arrays.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e22_arrays.py

```
# 教材位置：第 6 章，第 6.4 节；图 6-3。
# 单位、变量与形状：v 为 (2,)，X 为 (3,2)，row 为 (2,)；数组元素是无限精度浮点数字。
# 学习目标：区分矩阵乘法与逐元素运算，理解一维数组的形状、广播规则，
# 并通过切片视图和 copy 学会判断数组是否共享底层数据。
# 核心规则：@ 按线性代数的求和规则相乘，* 按对应元素相乘；
# 广播从最右侧维度对齐，每一对维度须相等或其中之一为 1。
# 数据与输出：使用 2×2 矩阵、长度为 2 的向量和 3×2 数组，打印各种运算，
# 展示不兼容广播被拒绝，以及修改视图如何改变原数组；保存 e22_arrays.png。
# 运行方式：安装 NumPy 后，在终端运行 python e22_arrays.py。

import numpy as np
# A、B 的形状都是 (2,2)；B 是交换两个坐标的矩阵。
A = np.array([[1, 2], [3, 4]])
B = np.array([[0, 1], [1, 0]])
# v 的 shape 为 (2,)，只有一个轴，并不是形状 (1,2) 或 (2,1) 的二维数组。
v = np.array([1.0, 2.0])
# A@B 的 (i,j) 元素是 A 的第 i 行与 B 的第 j 列的内积；
# A*B 只把相同位置的数字相乘。本例右乘 B 会交换 A 的两列。
# sep="\n" 让本次 print 的多个参数分行显示，便于比较两种结果。
print("A@B =", A @ B, "; A*B =", A * B, sep="\n")
# A@v 计算各行与 v 的内积，等价于把 v 当列向量；
# v@A 计算 v 与各列的内积，等价于把 v 当行向量；结果均为形状 (2,)。
print("A@v =", A @ v, "; v@A =", v @ A)
# 一维数组转置仍为 (2,)；[:,None] 在第二轴新增长度为 1 的维度，
# 才得到真正的二维列形数组 (2,1)，其中 None 等价于 np.newaxis。
print("数组形状：", v.shape, v.T.shape, v[:, None].shape)
# array_equal 检查形状与元素完全一致，适合此处确定的整数计算。
assert np.array_equal(A @ B, [[2, 1], [4, 3]])
assert not np.array_equal(A @ B, A * B)
assert v.T.shape == (2,) and v[:, None].shape == (2, 1)
# X 形状 (3,2)，row 形状 (2,)；广播时将 row 视作 (1,2)，
# 第一轴的 1 可扩展为 3，相同的两项偏移便应用到每一行。
X = np.ones((3, 2))
row = np.array([10, 20])
print("X+row =", X + row)
# [[11,21]]*3 在这里是 Python 列表重复，构造三行相同的预期结果，
# 并不是 NumPy 数组的数值乘法；allclose 将其与计算结果逐元素比较。
assert np.allclose(X + row, [[11, 21]] * 3)
# 用一个预期失败的例子认识广播边界；try/except 捕获错误后程序可继续。
try:
    # arange(3) 的形状是 (3,)。从右对齐后遇到 X 的末维 2 与 3，
    # 二者不相等且都不是 1，所以无法相加；它不会自动猜测你想按列加。
    X + np.arange(3)
except ValueError:
    # NumPy 用 ValueError 报告形状不兼容；捕获到它说明此项自检符合预期。
    print("已拒绝不兼容的数组形状")
else:
    # try 没有抛错才会执行 else；若不兼容运算竟然成功，就主动报告自检失败。
    raise AssertionError("此形状的广播应失败")
# arange(6.0) 创建浮点数组 [0,1,2,3,4,5]。基本切片 [2:5] 包含下标 2、3、4，
# 得到与 base 共享内存的视图；view[0] 对应 base[2]，因此赋值会改动 base。
```

```

base = np.arange(6.0)
view = base[2:5]
view[0] = 99
# copy() 显式复制数据; separate 修改自己的第一个元素, 不再影响 base。
separate = base[2:5].copy()
separate[0] = -1
# 同时确认原数组保留视图写入的 99, 而副本可独立改成 -1。
assert base[2] == 99 and separate[0] == -1
print("修改视图、复制后, 原数组 =", base)

# 配套图直接画上面实际使用的 v、X、row; 这里不改 v 的长度, 也不把加法换成减法。
# Matplotlib 只负责画图, 不改变 NumPy 运算或上面已经打印的数值输出。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0))
ax = axes[0]
ax.axis("off")
# 用文本方框展示同一份 v 的三种数组形状; None 只插轴, 不改变元素顺序。
for y_position, label, text in [
    (0.82, "$v$: (2,)", str(v)),
    (0.52, "$v[None,:]: (1,2)", str(v[None, :])),
    (0.16, "$v[:,None]: (2,1)", str(v[:, None])),
]:
    ax.text(0.02, y_position, label, transform=ax.transAxes, fontsize=9, va="center")
    ax.text(0.57, y_position, text, transform=ax.transAxes, fontsize=10,
           family="monospace", va="center", bbox={"facecolor": "#e7f0f7",
           "edgecolor": "#52758c"})
ax.set_title("同一数组的三种形状")
ax = axes[1]
ax.axis("off")
# X+row 中 row 形状为(2,), 广播到 X 的每一行; 此处数值直接读取实际运算。
assert X.shape == (3, 2) and row.shape == (2,)
broadcast_result = X + row
ax.text(0.03, 0.78, "$X$: (3,2)\n" + str(X), family="monospace", fontsize=9,
       transform=ax.transAxes, va="center")
ax.text(0.03, 0.42, "$row$: (2,)\n" + str(row), family="monospace", fontsize=9,
       transform=ax.transAxes, va="center")
ax.text(0.56, 0.62, "$X+row$: (3,2)\n" + str(broadcast_result), family="monospace",
       fontsize=9,
       transform=ax.transAxes, va="center", bbox={"facecolor": "#e7f0f7",
       "edgecolor": "#52758c"})
# arange(3) 的尾轴 3 与 X 的尾轴 2 不兼容, 图中保留已验证的异常规则。
ax.text(0.03, 0.12, "(3,2) + (3,) -> ValueError", transform=ax.transAxes,
       fontsize=8)
ax.set_title("广播加法: 从末轴对齐")
fig.tight_layout()
fig.savefig("e22_arrays.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```


程序 e23_solve.py

附件文件 e23_solve.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e23_solve.py

```
# 教材位置：第 7 章，第 7.5 节；图 7-1、图 7-3。
# 单位、变量与形状：矩阵 A 为 (2,2)，解与右端为 (2,)；本例数值无量纲，条件数衡量相对敏感
# 度。
# 学习目标：求解线性方程组 Ax=b，并用近奇异矩阵观察输入扰动的放大。
# 核心数学关系：可逆方阵给出唯一解；条件数 kappa(A)=||A||_2*||A^{-1}||_2
# 描述最不利方向上的相对误差放大上界，固定 A 时有
# ||delta x||/||x|| <= kappa(A)*||delta b||/||b||（精确算术下）。
# 数据与输出：先解一个 2x2 方程组，再保持病态矩阵 C 不变而轻微改变右端，
# 打印两组解、矩阵性质和相对变化，保存 e23_solve.png，并用 e23_rank.png 解释秩与零空间。
# 运行方式：安装 NumPy、Matplotlib 后，运行 python e23_solve.py；
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# A 形状 (2,2)，每一行是一条方程的系数；b 形状 (2,) 是对应右端项。
# 方程为 x+2y=9、3x-y=5，待求解向量包含未知量 x、y。
A = np.array([[1.0, 2.0], [3.0, -1.0]])
b = np.array([9.0, 5.0])
# solve 直接求解 Ax=b，返回形状 (2,)；无需显式计算 inv(A)@b，
# 直接求解通常更高效、数值上也更合适。本接口要求 A 为方阵且非奇异。
x = np.linalg.solve(A, b)
# round 仅控制显示的小数位，x 本身仍保留完整浮点结果供后续检验。
print("方程组的解 =", np.round(x, 6))
# rank 是数值秩，det 是行列式；cond 默认给出 2-范数条件数。
# 满秩说明在精确模型下可唯一求解，条件数则进一步描述解对扰动有多敏感；
# 行列式的绝对大小还受缩放影响，不能单独用它衡量数值稳定性。
print("秩、行列式、条件数 =", np.linalg.matrix_rank(A),
      round(np.linalg.det(A), 6), round(np.linalg.cond(A), 6))
# 同时检查解析解和代回方程的残差；allclose 允许浮点舍入误差。
assert np.allclose(x, [19 / 7, 22 / 7]) and np.allclose(A @ x, b)
# C 的两行几乎相同，仍可逆但接近奇异；两条约束几乎平行，交点很敏感。
# “病态”描述问题本身的敏感性，不能简单归因于求解算法出错。
C = np.array([[1.0, 1.0], [1.0, 1.0001]])
# 只让第二个右端分量增加 0.0001，系数矩阵 C 完全不变。
b1 = np.array([2.0, 2.0])
b2 = np.array([2.0, 2.0001])
# 相同的近奇异矩阵配上略有不同的右端，解分别接近 [2,0] 与 [1,1]。
x1 = np.linalg.solve(C, b1)
x2 = np.linalg.solve(C, b2)
# 使用欧氏范数衡量整体变化，并分别以原右端和原解的范数归一化，
# 得到无单位的相对变化，便于比较不同量纲/规模的问题。
relative_b = np.linalg.norm(b2 - b1) / np.linalg.norm(b1)
relative_x = np.linalg.norm(x2 - x1) / np.linalg.norm(x1)
# 条件数很大意味着存在可被显著放大的扰动方向，但并非每个扰动都取到上界。
print("扰动前后的解 =", x1, x2, "; 条件数 =",
      round(np.linalg.cond(C), 2))
print("右端 b、解 x 的相对变化 =", round(relative_b, 8), round(relative_x, 6))
```

```
# 检查两次解确实相距明显；这与右端变化很小并不矛盾。
assert np.allclose(x1, [2, 0]) and np.allclose(x2, [1, 1])
# 对“仅扰动 b”检查条件数上界，额外的 1e-10 容差吸收浮点舍入影响。
# 上界是最坏情形的保证，不要求相对解变化恰好等于该上界。
assert relative_x <= np.linalg.cond(C) * relative_b + 1e-10
# 为秩与零空间的概念配套作图：满秩矩阵直接复用本例开头的 A。
# rank_deficient 是另一个明确命名的秩亏矩阵；它与病态但可逆的 C 不是同一对象。
rank_deficient = np.array([[1.0, 1.0], [0.0, 0.0]])
null_vector = np.array([1.0, -1.0])
assert np.linalg.matrix_rank(A) == 2
assert np.linalg.matrix_rank(rank_deficient) == 1
assert np.allclose(rank_deficient @ null_vector, 0)
# 200 个单位圆列向量由同一矩阵统一映射：满秩保留面积，秩亏只留下一个方向。
angles = np.linspace(0, 2 * np.pi, 200)
unit_circle = np.vstack([np.cos(angles), np.sin(angles)])
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0))
for ax, matrix, title in zip(axes, [A, rank_deficient],
                              ["原矩阵 A: 满秩", "秩亏矩阵: 秩为 1"]):
    ax.plot(*unit_circle, "k--", alpha=0.5, label="单位圆")
    ax.plot(*(matrix @ unit_circle), lw=1.8, label="映射后的圆")
    ax.axhline(0, color="gray", lw=0.5)
    ax.axvline(0, color="gray", lw=0.5)
    ax.set_aspect("equal")
    ax.grid(alpha=0.25)
    ax.set_title(title, fontsize=10)
    ax.set(xlim=(-3.6, 3.6), ylim=(-3.6, 3.6), xlabel="$x_1$", ylabel="$x_2$")
    ax.legend(fontsize=7.8)
# z=(1,-1)落在 rank_deficient 的零空间；图中原点就是它变换后的像。
axes[1].annotate("", xy=null_vector, xytext=(0, 0),
                 arrowprops={"arrowstyle": "->", "color": "tab:green"})
axes[1].text(1.05, -1.3, "$z=(1,-1)^T$", fontsize=8, color="tab:green")
axes[1].scatter([0], [0], color="tab:orange", s=22, zorder=4)
axes[1].text(0.03, 0.04, "秩=1, 零化度=1",
             transform=axes[1].transAxes, fontsize=8)
fig.tight_layout()
fig.savefig("e23_rank.png", dpi=180, bbox_inches="tight")
plt.close(fig)

# 生成横坐标网格，用二维几何图展示两个很接近的约束方向。
xx = np.linspace(-1, 3, 200)
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1))
# 第一条方程 x+y=2 是两个系统共用的约束。
# 另外同时画出原系统与扰动系统的第二约束，避免把原解误当成另外两线的交点。
def constraint_lines(horizontal):
    # 每条线均从实际 C 与 b1/b2 求出；没有重新输入另一套近奇异系数。
    common = (b1[0] - C[0, 0] * horizontal) / C[0, 1]
    original = (b1[1] - C[1, 0] * horizontal) / C[1, 1]
    perturbed = (b2[1] - C[1, 0] * horizontal) / C[1, 1]
    return common, original, perturbed
labels = ["共同约束: $x+y=2$", "原约束: $x+1.0001y=2$",
          "扰动约束: $x+1.0001y=2.0001$"]
colors = ["tab:blue", "tab:green", "tab:orange"]
styles = ["-", ":", "--"]
ax = axes[0]
for values, label, color, style in zip(constraint_lines(xx), labels, colors, styles):
    ax.plot(xx, values, linestyle=style, color=color, label=label)
# 两个点分别代表两套系统的解，zorder=3 使点显示在曲线之上。
ax.scatter([x1[0], x2[0]], [x1[1], x2[1]], color=["tab:green", "tab:orange"],
```

```

zorder=3)
ax.annotate("原解 (2,0)", xy=x1, xytext=(1.5, 0.65), fontsize=8, color="tab:green")
ax.annotate("扰动解 (1,1)", xy=x2, xytext=(-0.75, 1.45), fontsize=8,
color="tab:orange")
# 标注坐标、图例和网格，帮助把代数解读作两条直线的交点。
ax.set(xlabel="$x$", ylabel="$y$", xlim=(-1, 3), ylim=(-1, 3),
title="约束相近，解相距较远")
ax.legend(fontsize=7.8)
ax.grid(alpha=0.3)
# 围绕(1.5,0.5)按相同倍数放大纵横坐标，三条几乎重合的约束才能看清。
# 这是显示坐标的变换；求解所用 C、b1、b2 均保持原样。
zoom_factor = 1e4
zoom_center = np.array([1.5, 0.5])
zoom_x = np.linspace(-0.6, 0.6, 150)
local_x = zoom_center[0] + zoom_x / zoom_factor
ax = axes[1]
for values, label, color, style in zip(constraint_lines(local_x), labels, colors,
styles):
    ax.plot(zoom_x, (values - zoom_center[1]) * zoom_factor,
linestyle=style, color=color, label=label.split(":")[0])
ax.set(xlabel="$10^4(x-1.5)$", ylabel="$10^4(y-0.5)$", title="局部坐标放大 $10^4$")
ax.legend(fontsize=8)
ax.grid(alpha=0.3)
# 调整布局、保存图片并关闭图，便于批量运行时控制资源占用。
fig.tight_layout()
fig.savefig("e23_solve.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e24_distances.py

附件文件 e24_distances.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e24_distances.py

```

# 教材位置：第 7 章，第 7.5 节；图 7-2。
# 单位、变量与形状：两端点为 (2,) 坐标；一范数、二范数、无穷范数均沿用坐标的长度单位。
# 学习目标：将点的距离写成“坐标差的范数”，比较 L1、L2、L $\infty$  三种度量。
# 核心数学关系：对位移  $d=(dx,dy)$ ， $L1=|dx|+|dy|$ ，
#  $L2=\sqrt{dx^2+dy^2}$ ， $L\infty=\max(|dx|,|dy|)$ ，且  $L\infty\leq L2\leq L1$ 。
# 数据与输出：计算三个二维点到参考点 (4,3) 的三种距离，打印数值，
# 并保存 e24_distances.png；图中只展示第一个点到参考点的路线。
# 运行方式：安装 NumPy、Matplotlib 后，运行 python e24_distances.py；
# 图片写入运行命令时的当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# points 形状 (3,2)，每行是一个点；第一列是横坐标，第二列是纵坐标。
points = np.array([[1.0, 1.0], [3.0, 3.0], [5.0, -0.5]])
# reference 是形状 (2,) 的一维数组，表示所有距离计算共同使用的参考点。
reference = np.array([4.0, 3.0])
# 广播将 reference 应用于每一行，得到形状仍为 (3,2) 的位移矩阵。
# 正负号反映方向，而范数取绝对值或平方，所以两点间距离不依赖减法方向。

```

```

difference = points - reference
# axis=1 表示每行独立计算一个向量范数，输出形状为 (3,);
# 若省略 axis 对二维数组调用 norm，含义可能变为矩阵范数，不能混用。
# ord=1 是绝对值之和，即只能沿坐标方向移动时的曼哈顿距离。
l1 = np.linalg.norm(difference, ord=1, axis=1)
# ord=2 是每行的欧氏长度，对应平面中的直线距离。
l2 = np.linalg.norm(difference, ord=2, axis=1)
# ord=np.inf 是每行绝对值最大的坐标差，也称切比雪夫距离；
# 若每一步可同时在两个坐标方向移动相同的最大幅度，它体现所需步数尺度。
linf = np.linalg.norm(difference, ord=np.inf, axis=1)
# 只给 l2 的显示结果四舍五入；保存的 l2 数组仍保留后续断言所需精度。
print("L1 =", l1, "; L2 =", np.round(l2, 6), "; Linf =", linf)
# 手算三行的绝对坐标差之和，验证广播和计算轴选择正确。
assert np.allclose(l1, [5, 1, 4.5])
# 欧氏距离的平方分别为 3^2+2^2、1^2+0^2、1^2+3.5^2。
assert np.allclose(l2, np.sqrt([13, 1, 13.25]))
# np.all 要求三个点均满足范数之间的不等式；这里比较的是同一位移的不同范数。
assert np.all(linf <= l2) and np.all(l2 <= l1)
# 取第一个点 P=(1,1) 与参考点 Q=(4,3)，两者各为形状 (2,) 的数组。
P, Q = points[0], reference
fig, ax = plt.subplots(figsize=(4.8, 3.6))
# 红色直线直接连接 P、Q，长度 sqrt(3^2+2^2)=sqrt(13)，对应 L2。
ax.plot([P[0], Q[0]], [P[1], Q[1]], "r-", label="二范数：直线路径")
# 蓝色折线先横移 3 再竖移 2，总路径长度为 5，对应 L1。
# L∞=3 是最大单轴位移；本图用标题给出数值，没有另画一条 L∞ 路线。
ax.plot([P[0], Q[0], Q[0]], [P[1], P[1], Q[1]], "b--", label="一范数：网格路径")
# 标出端点以便确认两种路线连接的是相同的一对位置。
ax.scatter([P[0], Q[0]], [P[1], Q[1]], color="black")
# 显示范围围绕本图的 P、Q；第三个点仅参与上面的数值计算，没有画在这里。
ax.set(xlim=(0, 5), ylim=(0, 4), xlabel="x", ylabel="y",
       title=r"距离：$L_1=5$, $L_2=\sqrt{13}$, $L_\infty=3$")
# equal 让横纵单位长度一致，保证图上的直线长度比例有几何意义。
ax.set_aspect("equal")
ax.legend()
ax.grid(alpha=0.3)
# 调整布局、保存 180 dpi 图片，然后关闭图以释放资源。
fig.tight_layout()
fig.savefig("e24_distances.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e25_eigen.py

附件文件 e25_eigen.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e25_eigen.py

```

# 教材位置：第 8 章，第 8.2 节；图 8-1。
# 单位、变量与形状：A 为 (2,2)，特征向量按列组成 (2,2)；坐标单位统一，特征值是轴向缩放倍
# 数。
# 案例 25：用对称正定矩阵理解特征值、特征向量与几何变换。
# 学习目标：把  $Av=\lambda v$  的代数关系与“沿特征方向只发生伸缩”的图像联系起来，
# 并检验实对称矩阵的特征向量可以构成一组标准正交基。
# 核心关系： $A=Q \operatorname{diag}(\lambda) Q^T$ ， $\operatorname{tr}(A)=\sum \lambda$ ， $\det(A)=\prod \lambda$ ；本例两个特征值均为正。
# 数据与形状：A 为 2×2 矩阵；vectors 的每一列是一个特征向量；
# 单位圆用一个 2×200 数组表示，每一列都是平面上的一个点。
# 输出：打印特征值、迹和行列式；断言核对恒等式；生成 e25_eigen.png。

```

```
# 运行方式：安装本目录 requirements.txt 中的依赖，在本目录执行
# python e25_eigen.py。图片保存到运行时的当前工作目录，不弹出交互窗口。

# NumPy 负责矩阵计算；约定 @ 表示矩阵乘法，* 表示逐元素运算或广播。
import numpy as np
# Matplotlib 负责把单位圆及其线性变换后的椭圆画在同一坐标系内。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
#  $A=A^T$ ，因此适用实对称矩阵专用的 eigh；浮点数便于后续数值线性代数计算。
A = np.array([[3.0, 1.0], [1.0, 2.0]])
# eigh 按特征值从小到大返回 values，形状为 (2,); vectors 形状为 (2,2)。
# 第 j 列 vectors[:,j] 对应 values[j]，不是第 j 行；各列长度为 1 且两两正交。
# 特征向量的正负号不唯一，换成相反方向仍是同一个特征方向。
values, vectors = np.linalg.eigh(A)
# 手算  $\det(A-\lambda I)=\lambda^2-5\lambda+5=0$ ，得到  $(5\pm\sqrt{5})/2$ ；按升序排好以与 eigh 对齐。
# 两根均为正，所以 A 在任意非零方向的二次型  $x^T A x$  都为正。
manual = np.array([(5 - np.sqrt(5)) / 2, (5 + np.sqrt(5)) / 2])
# 只在显示时保留 6 位小数；后续检验继续使用未舍入的原始数值。
print("特征值 =", np.round(values, 6))
# 迹为主对角线元素之和 5，行列式为  $3\times 2-1\times 1=5$ ，可独立核对特征值结果。
print("迹、行列式 =", np.trace(A), round(np.linalg.det(A), 6))
# 浮点计算可能有舍入误差，使用 allclose 的容差比较数值，不要求逐位相同。
assert np.allclose(values, manual)
# 同时检验两条  $A v_j = \lambda_j v_j$ ：右侧  $(2,2)\times(2,)$  把第 j 列乘以  $\lambda_j$ 。
# 这条广播规则十分关键；如果误把特征向量当作行，关系一般不成立。
assert np.allclose(A @ vectors, vectors * values)
#  $Q^T Q = I$  检验列向量标准正交；因此在这个实对称案例中  $Q^{-1}=Q^T$ 。
assert np.allclose(vectors.T @ vectors, np.eye(2))
# 核对“特征值之和等于迹”；迹在换基后保持不变。
assert np.isclose(values.sum(), np.trace(A))
# 核对“特征值之积等于行列式”；在二维中该乘积也是面积缩放倍数。
assert np.isclose(values.prod(), np.linalg.det(A))
# 用从 0 到  $2\pi$  的 200 个角度描出单位圆；含两个端点，曲线自然闭合。
theta = np.linspace(0, 2 * np.pi, 200)
# vstack 把横、纵坐标叠成  $(2,200)$ ， $A@ (2,200)$  同时变换 200 个列向量。
# 由于 A 对称且正定，圆变为沿特征向量方向、半轴长度为特征值的椭圆。
circle = np.vstack([np.cos(theta), np.sin(theta)])
ellipse = A @ circle
# 创建一张图及一个坐标轴；figsize 的单位是英寸。
fig, ax = plt.subplots(figsize=(4.5, 4))
# 星号把两行坐标展开为 plot(x,y)；黑色虚线给出变换前的单位圆。
ax.plot(*circle, "k--", label="单位圆")
# 将全部变换后的点连成曲线，与单位圆比较伸缩方向和幅度。
ax.plot(*ellipse, label="A 映射后的圆")
# vectors.T 的每一行恰好是原矩阵的一列，因此可逐对取出  $(\lambda_j, v_j)$ 。
# 这里转置是为了 Python 的逐行迭代习惯，并未改变特征向量按列存储的约定。
for value, vector in zip(values, vectors.T):
    # 把  $-\lambda v$  和  $+\lambda v$  作为两列，组成  $(2,2)$  的线段端点数组，画出完整的椭圆轴。
    endpoints = value * np.column_stack([-vector, vector])
    # 图例显示该方向的伸缩倍数；3 位小数只影响图例文本。
    ax.plot(*endpoints, label=f"主轴 {value:.3f}")
# 横纵坐标使用相同长度单位，避免把圆因画布比例画成椭圆；淡网格辅助读图。
ax.set_aspect("equal")
ax.grid(alpha=0.3)
ax.legend()
```



```
# 标题说明本图展示的是对称正定线性变换。
ax.set_title("对称正定矩阵的变换")
# 自动调整边距, 以 180 dpi 保存, 再关闭图对象以释放内存。
fig.tight_layout()
fig.savefig("e25_eigen.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e26_diagonalize.py

附件文件 `e26_diagonalize.py`。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e26_diagonalize.py

```
# 教材位置: 第 8 章, 第 8.3 节; 仅终端输出。
# 单位、变量与形状: 随机矩阵 A 为 (2,2), 矩阵幂为十步转移概率; 全部元素和特征值无量纲。
# 案例 26: 用对角化计算矩阵幂, 并认识不能对角化的 若尔当 块。
# 学习目标: 理解  $A=Q\Lambda Q^{-1}$  时  $A^k=Q\Lambda^k Q^{-1}$ , 及“有足够多线性无关特征向量”
# 才是可对角化的关键; 重复特征值本身并不自动意味着不能对角化。
# 核心关系: 几何重数  $\dim \ker(A-\lambda I)=n-\text{rank}(A-\lambda I)$ , 且不超过代数重数。
# 数据: A 为非对称的 2x2 行随机矩阵; J 为特征值 2 的二阶 若尔当 块。
# 输出: 打印 A 的特征值、A 的 10 次幂、行和以及 J 的几何重数; 不生成图片。
# 运行方式: 安装依赖后在本目录执行 python e26_diagonalize.py。
# 所有 assert 都是数值或结构检查, 正常通过时不会额外打印信息。

# NumPy 提供特征分解、线性方程求解、矩阵幂和秩计算。
import numpy as np
# 各行非负且和为 1, 因此也可把 A 理解为两状态转移矩阵。
# A 不对称, 不能沿用对称矩阵  $Q^{-1}=Q^T$  的结论。
A = np.array([[0.9, 0.1], [0.2, 0.8]])
# 一般矩阵用 eig; Q 的第 j 列是与 values[j] 对应的右特征向量。
# 返回次序不保证升序。本例特征值 1 和 0.7 不同, 所以两列线性无关、Q 可逆。
values, Q = np.linalg.eig(A)
# 求解  $QX=I$  得到  $Q^{-1}$ ; 使用线性方程求解接口表达这个计算。
# 这里显式构造逆矩阵是为了展示对角化公式, 一般求解  $Ax=b$  时通常直接 solve(A,b)。
Q_inverse = np.linalg.solve(Q, np.eye(2))
# 将矩阵幂指数设为 10, 用重复转移的例子观察对角化如何简化计算。
k = 10
# 先把两个特征值分别取 k 次幂, 再构成 2x2 对角矩阵。
# 由于相邻的  $Q^{-1}Q=I$  会抵消, A 连乘 k 次就化为  $Q\Lambda^k Q^{-1}$ 。
power_from_eigen = Q @ np.diag(values ** k) @ Q_inverse
# 用独立的矩阵幂算法作参照, 验证上面的特征分解计算。
power_direct = np.linalg.matrix_power(A, k)
# 只为显示而排序特征值; 不能单独重排 values 后仍与原 Q 配对使用。
print("特征值 =", np.round(np.sort(values), 6))
# 打印十次转移矩阵; 对行概率向量 x, 十步分布为  $x@A^{10}$ 。
print("A^10 =\n", np.round(power_direct, 6))
# axis=1 沿每行的列求和, 得到形状 (2,) 的行和数组。
print("各行元素之和 =", A.sum(axis=1))
# 检验每行和为 1; 这里只检查行和, 矩阵非负性由给定数据直接可见。
assert np.allclose(A.sum(axis=1), 1)
# 重建 A, 以容差检验本例的对角化确实成立。
assert np.allclose(A, Q @ np.diag(values) @ Q_inverse)
# 两个独立算法应给出相同的  $A^{10}$ , 允许浮点舍入误差。
assert np.allclose(power_direct, power_from_eigen)
# J 的特征多项式为  $(2-\lambda)^2$ , 所以  $\lambda=2$  的代数重数是 2。
# 右上角的 1 使 J 不是简单的  $2I$ , 接下来检查其特征空间维数。
```

```
J = np.array([[2.0, 1.0], [0.0, 2.0]])
# 特征向量满足  $(J-2I)v=0$ ; 由秩-零化度定理, 核空间维数为  $2-\text{rank}$ 。
#  $J-2I$  的秩为 1, 故只有一个线性无关特征向量, 无法组成可逆的  $2\times 2$  的  $Q$ 。
geometric_multiplicity = 2 - np.linalg.matrix_rank(J - 2 * np.eye(2))
# 把代数重数与几何重数并列显示, 突出“根出现几次”和“独立方向有几个”的区别。
print("矩阵 J: 代数重数=2, 几何重数=",
      geometric_multiplicity)
# 几何重数 1 小于代数重数 2, 说明这个 J 不能对角化。
assert geometric_multiplicity == 1
# 写  $J=2I+N$ , 且  $N^2=0$ , 因此  $J^3=8I+12N$ ; 右上角为 12。
# 这说明不能对角化的矩阵仍然可以求幂, 只是不能直接套用  $Q\Lambda^kQ^{-1}$ 。
assert np.allclose(np.linalg.matrix_power(J, 3), [[8, 12], [0, 8]])
```

程序 e27_markov.py

附件文件 e27_markov.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e27_markov.py

```
# 教材位置: 第 8 章, 第 8.6 节; 图 8-2。
# 单位、变量与形状: P 为  $(3, 3)$ , X 每行是一步的  $(3,)$  状态概率; 概率无量纲, 步数按转移次数
# 计。
# 案例 27: 行随机矩阵、概率分布迭代、稳态及周期性反例。
# 学习目标: 分清行概率向量  $x$  的更新  $x_{t+1}=x_tP$  与列向量特征方程的关系,
# 理解稳态  $\pi P=\pi$ , 并观察长期分布如何靠近稳态。
# 数据与形状: P 为  $3\times 3$ ,  $P[i, j]$  为从状态  $i$  到状态  $j$  的一步概率;
# X 为  $301\times 3$ , 第  $t$  行保存第  $t$  步的三种状态概率, 不是随机抽取的一条状态路径。
# 本例 P 的元素全为正, 有限状态链不可约且非周期, 因而有唯一稳态并收敛。
# 输出: 打印稳态和第二大特征值模; 验证周期反例; 生成 e27_markov.png。
# 运行方式: 安装依赖后在本目录执行 python e27_markov.py。
# 图片保存到当前工作目录。计算完全确定, 本例没有随机抽样。

# NumPy 用于概率向量更新、特征分解及收敛性的数值检查。
import numpy as np
# Matplotlib 画各状态概率的时间演化及其稳态参考线。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 每一行给出一个当前状态的下一步去向; 对角线接近 1 表示一步内较可能保持原状态。
P = np.array([[0.90, 0.08, 0.02], [0.05, 0.92, 0.03], [0.01, 0.04, 0.95]])
# 转移概率必须非负且每行和为 1, 这样  $x@P$  才能保持非负并使总概率仍为 1。
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
# 包含初始第 0 步与之后 300 步, 所以预分配 301 行。
# empty 的未赋值位置没有有效数据; 循环按顺序写入各行后再使用。
X = np.empty((301, 3))
X[0] = [0.40, 0.35, 0.25]
# t 依次为 0, ..., 299, 从已知的第 t 行计算第 t+1 行。
for t in range(300):
    # 行向量形状  $(3,)$  与  $(3, 3)$  相乘; 第 j 项为  $\sum_i x_t[i]P[i, j]$ 。
    X[t + 1] = X[t] @ P
# 稳态是 P 的左特征向量:  $\pi P=\pi$ 。转置后写成  $P^T\pi^T=\pi^T$ ,
# 所以对 P.T 求右特征向量, 再把所得列向量的数值当作稳态各分量。
values, vectors = np.linalg.eig(P.T)
# 浮点计算中的特征值未必逐位等于 1; 选择距离 1 最近的特征值对应的列。
```

```

vector = vectors[:, np.argmin(np.abs(values - 1))]
# 一般 eig 可能返回复数类型; 先确认虚部只在舍入误差范围, 再取实部。
assert np.max(np.abs(vector.imag)) < 1e-12
# 特征向量可以乘任意非零常数, 除以分量总和把它归一化成概率向量。
# 即使求出的向量整体为负, 这种归一化也会恢复本例的非负稳态。
pi = vector.real / vector.real.sum()
# 显示稳态概率; 稳态表示总体分布不再改变, 不表示每个个体停止转移。
print("稳态概率向量 =", np.round(pi, 6))
# 去掉模为 1 的主特征值后, 第二大模刻画本例误差长期衰减的主要速率。
# 模越接近 1, 通常需要越多步才能接近稳态; 这里取模也覆盖负或复特征值。
print("次大特征值模 =", round(np.sort(np.abs(values))[-2], 6))
# 同时验证概率非负和稳态方程; pi.sum()=1 已由归一化构造保证。
assert np.all(pi >= 0) and np.allclose(pi @ P, pi)
# 与手工解线性方程  $\pi P = \pi$ 、 $\sum \pi_i = 1$  得到的分数形式核对。
assert np.allclose(pi, [7 / 29, 12 / 29, 10 / 29])
# 第 300 步分布应已接近稳态; rtol=0 使这里采用纯绝对误差  $10^{-10}$ 。
assert np.allclose(X[-1], pi, atol=1e-10, rtol=0)
# 两状态链每步必定换到另一状态, 周期为 2, 特征值为 1 和 -1。
# 它有稳态  $[1/2, 1/2]$ , 但从  $[1, 0]$  出发的分布会来回振荡, 说明有稳态不保证收敛。
cycle = np.array([[0, 1], [1, 0]])
# 三步为奇数, 从状态 1 必定到状态 2; 这里整数结果可用 array_equal 精确比较。
assert np.array_equal(np.array([1, 0]) @ np.linalg.matrix_power(cycle, 3),
                      [0, 1])
# 用两个坐标轴展示三状态收敛过程与周期二状态反例, 画布大小以英寸指定。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.6))
ax = axes[0]
# enumerate 同时提供状态列索引 j 与图例文字。
for j, name in enumerate(["状态 1", "状态 2", "状态 3"]):
    # 只绘制第 0 至 80 步以突出过渡过程; line, 从返回列表中取出唯一的曲线对象。
    line, = ax.plot(np.arange(81), X[:81, j], label=name)
    # 用与该状态曲线相同的颜色画稳态水平虚线, 便于观察趋近关系。
    ax.axhline(pi[j], color=line.get_color(), linestyle=":")
# 横轴为离散时间步, 纵轴为总体状态概率; 三条曲线在任一步的值相加为 1。
ax.set(xlabel="步数", ylabel="概率", title="正转移链: 概率收敛")
# 图例置于坐标轴外侧下方, 避免遮挡任一条状态轨迹; 网格辅助读数。
ax.legend(fontsize=7.8, loc="upper center", bbox_to_anchor=(0.5, -0.25), ncol=3)
ax.grid(alpha=0.3)
# 周期曲线从已定义的 cycle 实际迭代; 不把交替结果硬编码成奇偶公式。
periodic = np.empty((11, 2))
periodic[0] = [1, 0]
for step in range(10):
    periodic[step + 1] = periodic[step] @ cycle
periodic_stationary = np.array([0.5, 0.5])
assert np.allclose(periodic.sum(axis=1), 1)
assert np.allclose(periodic_stationary @ cycle, periodic_stationary)
assert np.allclose(periodic[:, 2], [1, 0]) and np.allclose(periodic[1::2], [0, 1])
ax = axes[1]
ax.plot(np.arange(11), periodic[:, 0], "-.", label="状态 1, 初始 (1,0)")
ax.axhline(periodic_stationary[0], color="tab:orange", linestyle=":",
           label="稳态概率 = 1/2")
ax.set(xlabel="步数", ylabel="状态 1 的概率", ylim=(-0.08, 1.08),
       title="周期链: 概率不收敛")
ax.legend(fontsize=7.8, loc="upper center", bbox_to_anchor=(0.5, -0.25))
ax.grid(alpha=0.3)
# 300 步只用于上面的高精度数值核验; 左图仍使用同一个 x 的前 81 行。
# 调整边距并保存 180 dpi 图片, 关闭图对象以避免批量运行时积累内存。
fig.tight_layout()
fig.savefig("e27_markov.png", dpi=180, bbox_inches="tight")
    
```



```
plt.close(fig)
```

程序 e28_svd.py

附件文件 e28_svd.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e28_svd.py

```
# 教材位置：第 9 章，第 9.4 节；图 9-2。
# 单位、变量与形状：A 为 (3,2)，输入圆为 (2,200)，输出为 (3,200)；示意坐标和矩阵元素无量纲。
# 案例 28：比较精简 奇异值分解 与完整 奇异值分解，逐项核对形状和重建关系。
# 学习目标：理解任意实矩阵  $A=U\Sigma V^T$ ，及奇异值与  $A^T A$  特征值、弗罗贝尼乌斯 范数的联系。
# 本例 A 为 3×2、秩为 2；精简分解 U 为 3×2、s 为长度 2、Vh 为 2×2。
# NumPy 的 Vh 存储 V 的共轭转置；本例为实数，故  $Vh=V^T$ 。
# 完整分解 U 为 3×3，矩形  $\Sigma$  为 3×2，Vh 仍为 2×2；额外的 U 列补齐正交基。
# 输出：打印精简与完整分解的形状、奇异值；用 assert 验证各项关系，生成四阶段几何图 e28_svd.png。
# 运行方式：安装依赖后在本目录执行 python e28_svd.py。
# 本例没有随机数据；allclose/isclose 允许线性代数算法的浮点误差。

# 用 NumPy 的 奇异值分解 与对称特征值算法作相互独立的数值核对。
import numpy as np
# 两列线性无关，故 rank(A)=2；非方阵也可以进行奇异值分解。
A = np.array([[3.0, 1.0], [0.0, 2.0], [1.0, 0.0]])
# full_matrices=False 返回精简形式，内维度 k=min(3,2)=2；
# s 按从大到小排列且非负，U 的列与 Vh 的行按同一顺序一一对应。
# 精简形式保留 k 个奇异值（必要时也包含零），不等同于自动删去全部零奇异值。
U, s, Vh = np.linalg.svd(A, full_matrices=False)
# 先读形状有助于确认 (3,2)@(2,2)@(2,2) 最后重建成 (3,2)。
print("精简分解各因子的形状 =", U.shape, s.shape, Vh.shape)
# 奇异值描述沿右奇异向量方向的伸缩量，打印时舍入并不修改 s。
print("奇异值 =", np.round(s, 6))
# 用形状断言防止把完整分解与精简分解混用。
assert U.shape == (3, 2) and Vh.shape == (2, 2)
# U*s 将 U 的第 j 列乘以 s[j]，等价于 U@diag(s)，但无需显式构造对角矩阵。
# 于是 (U*s)@Vh 就是  $\sum s_j u_j v_j^T$ ，应该重建原矩阵。
assert np.allclose((U * s) @ Vh, A)
# 检验 U 的列与 Vh 的行标准正交。精简 U 不是方阵，
#  $U^T U = I_2$ ，但  $U U^T$  是投影到 A 列空间的 3×3 矩阵，并不等于  $I_3$ 。
assert np.allclose(U.T @ U, np.eye(2)) and np.allclose(Vh @ Vh.T, np.eye(2))
# Vh.T 的列是右奇异向量  $v_j$ ； $A v_j = s_j u_j$  表示“先选方向，再伸缩到左奇异方向”。
assert np.allclose(A @ Vh.T, U * s)
#  $A^T A$  为 2×2 对称半正定矩阵，特征值等于奇异值的平方。
# eigvalsh 按升序返回，s 按降序返回，因此用 s[::-1] 反向后再逐项比较。
assert np.allclose(np.linalg.eigvalsh(A.T @ A), s[::-1] ** 2)
# 完整形式把左奇异向量扩充为三维空间的整组正交基；奇异值仍只返回两个。
Uf, sf, Vhf = np.linalg.svd(A, full_matrices=True)
# 构造 (3,2) 的矩形  $\Sigma$ ：前两个对角位置放奇异值，第三行补零。
Sigma = np.zeros(A.shape)
Sigma[:2, :2] = np.diag(sf)
# 完整形式的矩阵乘法形状为 (3,3)@(3,2)@(2,2)=(3,2)。
print("完整分解各因子的形状 =", Uf.shape, Sigma.shape, Vhf.shape)
# 完整与精简形式虽形状不同，但都必须正确重建 A。
assert np.allclose(Uf @ Sigma @ Vhf, A)
# Uf 的额外第三列正交于 A 的全部列，所以属于  $\ker(A^T)$ ，即左零空间。
```

```

# 它在完整重建中乘以  $\Sigma$  的零行，因此不改变 A。
assert np.allclose(A.T @ Uf[:, 2], 0)
# 弗罗贝尼乌斯 范数平方为所有元素平方和，也等于奇异值平方和  $\sum s_j^2$ 。
assert np.isclose(np.linalg.norm(A, "fro") ** 2, s @ s)

# 用同一个实际 3×2 矩阵展示完整奇异值分解的几何阶段：输入是二维，输出是三维。
# 完整因子 Uf(3,3)、Sigma(3,2)、Vhf(2,2) 来自上面的实际分解，不另外换成 2×2 矩阵。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
angles = np.linspace(0, 2 * np.pi, 200)
input_circle = np.vstack([np.cos(angles), np.sin(angles)])
# 每一列是一个输入点；分别施加  $V^T$ 、矩形  $\Sigma$  和 U，阶段形状清楚显示维数衔接。
after_v = Vhf @ input_circle
after_sigma = Sigma @ after_v
after_u = Uf @ after_sigma
assert input_circle.shape == after_v.shape == (2, 200)
assert after_sigma.shape == after_u.shape == (3, 200)
assert np.allclose(after_u, A @ input_circle)
#  $\Sigma$  的第三行是零，因此第三阶段在三维空间的坐标平面内；Uf 把该平面映到 A 的列空间。
assert np.allclose(after_sigma[2], 0)
assert np.allclose(Uf.T @ after_u, after_sigma)
assert np.allclose(Uf @ Sigma @ Vhf, A)
# 标准基向量也随阶段同步变换，能看出正交因子含旋转或反射，而不只看到无标记圆。
stage_matrices = [np.eye(2), Vhf, Sigma @ Vhf, Uf @ Sigma @ Vhf]
stage_points = [input_circle, after_v, after_sigma, after_u]
fig = plt.figure(figsize=(6.3, 6.4))
titles = ["1 输入：二维单位圆", "2  $V^T$ ：二维到二维", "3  $\Sigma$ ：二维到三维",
          "4  $U$ ：三维到三维"]
for index, (points, matrix, title) in enumerate(zip(stage_points, stage_matrices,
                                                    titles)):
    # 所有面板共用相同数值范围；三维图使用正交投影和相同坐标单位比例。
    if index < 2:
        ax = fig.add_subplot(2, 2, index + 1)
        ax.plot(*points, color="tab:blue", lw=1.3)
        ax.scatter(*points[:, 0], color="tab:orange", s=15, zorder=4)
        ax.annotate("", xy=points[:, 15], xytext=points[:, 7],
                    arrowprops={"arrowstyle": "->", "color": "tab:orange", "lw":
1.0})
        for column, color in zip(matrix.T, ["tab:green", "gray"]):
            ax.annotate("", xy=column, xytext=(0, 0),
                        arrowprops={"arrowstyle": "->", "color": color})
        ax.set(xlim=(-3.8, 3.8), ylim=(-3.8, 3.8), xlabel="$x_1$", ylabel="$x_2$")
        ax.set_aspect("equal")
        ax.set_xticks([-3, 0, 3])
        ax.set_yticks([-3, 0, 3])
        ax.grid(alpha=0.2)
    else:
        ax = fig.add_subplot(2, 2, index + 1, projection="3d")
        ax.plot(*points, color="tab:blue", lw=1.3)
        ax.scatter(*points[:, 0], color="tab:orange", s=15)
        # 沿圆周前进方向的小箭头与橙点共同标出方向；基向量箭头从原点出发。
        direction = points[:, 15] - points[:, 7]
        ax.quiver(*points[:, 7], *direction, color="tab:orange",
                  arrow_length_ratio=0.45)
        for column, color in zip(matrix.T, ["tab:green", "gray"]):

```

```

        ax.quiver(0, 0, 0, *column, color=color, arrow_length_ratio=0.12)
    ax.set(xlim=(-3.8, 3.8), ylim=(-3.8, 3.8), zlim=(-3.8, 3.8))
    ax.set_xticks([-3, 0, 3])
    ax.set_yticks([-3, 0, 3])
    ax.set_zticks([-3, 0, 3])
    ax.set_box_aspect((1, 1, 1))
    ax.set_proj_type("ortho")
    ax.view_init(elev=32, azim=-125)
    # 三维轴标签放在本面板内部：两行两列的布局让方向标记与刻度在教材页宽下仍可辨认。
    ax.set_xlabel("$x_1$", fontsize=8, labelpad=-5)
    ax.set_ylabel("$x_2$", fontsize=8, labelpad=-5)
    ax.text2D(0.91, 0.58, "$x_3$", transform=ax.transAxes, fontsize=8)
    ax.tick_params(labelsize=8, pad=0)
    ax.set_title(title, fontsize=9.5, pad=5)
    ax.tick_params(labelsize=8)
# 因输入输出维度不同，第三、四面板必须是三维显示，不能沿用二维圆→椭圆四图。
fig.text(0.5, 0.015, "同一矩阵 A (3×2)；橙色标记显示方向变化", ha="center", fontsize=8)
fig.tight_layout(rect=(0, 0.07, 1, 1), pad=0.5)
fig.savefig("e28_svd.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e29_lowrank.py

附件文件 e29_lowrank.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e29_lowrank.py

```

# 教材位置：第 9 章，第 9.6 节；图 9-3。
# 单位、变量与形状：A 为 (60,80)，s 为 (60,)；矩阵元素用一般测量单位，能量比例无量纲。
# 案例 29：截断 奇异值分解、最佳低秩近似与保留能量。
# 学习目标：用低秩信号加噪声的数据观察奇异谱，并用数学恒等式解释重建误差。
# 核心关系： $A_k = \sum_{j=1}^k s_j u_j v_j^T$  在  $\text{rank} \leq k$  的矩阵中使 弗罗贝尼乌斯 误差最小；
#  $\|A - A_k\|_F^2 = \sum_{j>k} s_j^2$ ；保留能量为前 k 个奇异值平方和占比。
# 数据与形状：两个高斯矩阵 (60,4) @ (4,80) 产生秩至多为 4 的信号，
# 再加独立高斯噪声得到 A(60,80)；精简 奇异值分解 的 U、s、Vh 形状为 (60,60)、(60,)、(60,
# 80)。
# 输出：打印前四项能量、 $k=2/4/10$  的误差和谐尾范数；生成 e29_lowrank.png。
# 运行方式：安装依赖后在本目录执行 python e29_lowrank.py，图保存在当前工作目录。

# NumPy 生成模拟矩阵并计算 奇异值分解、累计和以及矩阵/向量范数。
import numpy as np
# 两幅图分别展示奇异值衰减和累计能量，帮助选择近似秩。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定随机种子 9，便于在相同软件环境中重复生成数据、核对结果。
rng = np.random.default_rng(9)
# 标准正态矩阵的乘积最多只有 4 个独立列方向，故无噪声信号秩不超过 4。
# 低秩结构来自因子维度，而不是简单地把矩阵的大部分元素设为零。
A = rng.normal(size=(60, 4)) @ rng.normal(size=(4, 80))
# 在每个元素上加入均值 0、标准差 0.25 的独立噪声，scale 不是方差。
# 加噪后矩阵通常满秩，因此第 5 项以后的奇异值一般非零。
A += rng.normal(scale=0.25, size=A.shape)
# 精简 奇异值分解 按奇异值从大到小排列；最大的几个分量通常承载主要的低秩信号。

```

```

U, s, Vh = np.linalg.svd(A, full_matrices=False)
# cumsum 得到前 1、2、...项奇异值平方和，分母为  $\|A\|_F^2$ 。
# 这里“能量”特指原矩阵的平方和；没有中心化，所以不直接等同于 主成分分析 的解释方差。
cumulative = np.cumsum(s ** 2) / (s @ s)
# Python 从 0 编号，索引 3 表示前 4 项累计保留能量，响应信号的因子维度 4。
print("前四个分量的能量保留比例 =", round(cumulative[3], 6))
# 比较低于、等于、高于原信号秩的三种截断选择，观察精度随 k 的变化。
for k in [2, 4, 10]:
    # U[:, :k] 为 (60, k)，s[:k] 按列广播缩放，Vh[:k] 为 (k, 80)。
    # 只保留前 k 个外积项，仍重建为 (60, 80)，秩至多为 k。
    approximation = (U[:, :k] * s[:k]) @ Vh[:k]
    # 弗罗贝尼乌斯 误差把全部元素的重建差平方相加后开方，衡量整张矩阵的总体误差。
    error = np.linalg.norm(A - approximation, "fro")
    # s[k:] 是被舍弃的奇异值；向量默认的二范数即  $\sqrt{\sum_{j>k} s_j^2}$ 。
    tail = np.linalg.norm(s[k:])
    # 将直接计算的矩阵误差与谱尾范数并排打印，展示截断 奇异值分解 的误差公式。
    print(f"k={k}: 误差={error:.6f}, 舍弃奇异值的二范数={tail:.6f}")
    # 核对最佳低秩近似定理的误差等式，而不只是凭图判断近似是否好。
    assert np.isclose(error, tail)
    # 相对误差= $\sqrt{1-\text{保留能量}}$ ，可把无单位的精度要求转成累计能量要求。
    assert np.isclose(error / np.linalg.norm(A, "fro"),
                      np.sqrt(1 - cumulative[k - 1]))
# 创建左右两图，axes 为包含两个坐标轴的一维数组。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3))
# 横轴从 1 开始表示第几个奇异值；纵轴取对数，使大信号与小噪声尺度都可见。
axes[0].semilogy(np.arange(1, len(s) + 1), s, "-")
# 奇异谱的明显落差常提示有效低秩结构；具体选 k 仍应结合应用容许的误差。
axes[0].set(xlabel="序号", ylabel="奇异值", title="奇异值谱")
# 累计能量随 k 单调不减，保留全部分量时应接近 1。
axes[1].plot(np.arange(1, len(s) + 1), cumulative)
# 该图读取的是保留平方和比例；例如接近 1 不代表每个元素都有同样的小误差。
axes[1].set(xlabel="k", ylabel="能量保留比例", title="累计能量比例")
# 为两个坐标轴添加淡网格，便于比较谱的尺度及累计比例。
for ax in axes: ax.grid(alpha=0.3)
# 整理布局、保存图片并释放绘图资源，方便 run_all.py 连续执行。
fig.tight_layout()
fig.savefig("e29_lowrank.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e30_image.py

附件文件 e30_image.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e30_image.py

```

# 教材位置：第 9 章，第 9.7 节；图 9-4。
# 单位、变量与形状：image 为 (100, 140)；灰度使用无量纲相对亮度，存储比和相对误差均无量纲
# 。
# 案例 30：把灰度图看作矩阵，用截断 奇异值分解 比较压缩量与失真。
# 学习目标：理解图像的每个像素是矩阵元素；只保留 k 个奇异分量可减少需存储的标量数。
# 核心关系： $I_k = (U[:, :k] * s[:k]) @ Vh[:k]$ ；相对误差= $\|I - I_k\|_F / \|I\|_F$ 。
# 对  $m \times n$  图，保存三组因子的标量数为  $mk + k + kn = k(m + n + 1)$ ，原图为  $mn$ 。
# 数据与形状：用函数生成  $100 \times 140$  灰度矩阵并添加噪声，不读取外部图片；
# 精简 U、s、Vh 的形状分别为 (100, 100)、(100,)、(100, 140)。
# 输出：打印  $k=2/8/25$  的标量存储比和相对误差；生成四幅并排比较图 e30_image.png。
# 运行方式：安装依赖后在本目录执行 python e30_image.py，图片保存到当前工作目录。

```

```
# NumPy 负责网格、模拟灰度图、奇异值分解及误差计算。
import numpy as np
# imshow 把二维数组显示为灰度图；各子图共用颜色范围以便公平比较。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 12 使噪声可复现；这里只影响模拟图像，不改变 奇异值分解 的定义。
rng = np.random.default_rng(12)
# mgrid 切片中的 100j/140j 表示在闭区间上取相应个数的网格点，
# 不是虚数坐标；y、x 都是 (100,140)，依次对应行方向与列方向坐标。
y, x = np.mgrid[-1:1:100j, -1:1:140j]
# 基底亮度 0.5 加上平滑的二维波纹；乘积  $\sin(8x)\cos(5y)$  有可分离的低秩结构。
image = 0.5 + 0.2 * np.sin(8 * x) * np.cos(5 * y)
# 加入中心在 (0.25,-0.1) 的平滑亮斑；系数 25 控制亮斑衰减的快慢。
image += 0.3 * np.exp(-25 * ((x - 0.25) ** 2 + (y + 0.1) ** 2))
# 每个像素叠加标准差 0.035 的高斯噪声，使图像具有许多较小的奇异分量。
image += rng.normal(scale=0.035, size=image.shape)
# 精简 奇异值分解 保留  $\min(100,140)=100$  个奇异值，按降序排列供后续截断。
U, s, Vh = np.linalg.svd(image, full_matrices=False)
# 一行四个子图：原图和三个不同 k 的重建图，方便比较细节损失。
fig, axes = plt.subplots(1, 4, figsize=(6.3, 2.8))
# 灰度颜色固定映射到 0~1；噪声可能使数组少数值超出此范围，
# 显示时颜色会饱和，但这里没有修改或截断用于 奇异值分解 与误差计算的原始数值。
axes[0].imshow(image, cmap="gray", vmin=0, vmax=1)
# 标出含噪声的原始模拟图，作为所有重建的共同参照。
axes[0].set_title("原始模拟图")
# 用正无穷初始化“上一次误差”，使第一次迭代也能统一执行单调性断言。
previous = np.inf
# 逐个将后三个子图与截断秩 2、8、25 配对，k 越大保存的细节和噪声也越多。
for ax, k in zip(axes[1:], [2, 8, 25]):
    # (100,k) 的左因子按列乘奇异值，再乘 (k,140) 的右因子，输出仍为 (100,140)。
    # 这是给定秩上限 k 时的最佳 弗罗贝尼乌斯 近似，并不保证每个像素都更接近无噪声真图。
    approximation = (U[:, :k] * s[:k]) @ Vh[:k]
    # 用整图 弗罗贝尼乌斯 误差除以原图范数，得到不依赖灰度整体缩放的相对误差。
    error = (np.linalg.norm(image - approximation, "fro") /
             np.linalg.norm(image, "fro"))
    # 保存  $U_k$ 、 $s_k$ 、 $Vh_k$  需要  $100k+k+140k$  个标量，除以原图 14000 个标量。
    # 这是相同标量精度下的理论数量比，没有计入文件头、编码、量化等实际存储开销。
    storage = k * (sum(image.shape) + 1) / image.size
    # 同时报告成本与精度，避免只根据重建图片的主观观感选择压缩秩。
    print(f"k={k}: 标量存储比={storage:.4f}, 相对误差={error:.6f}")
    # 本例按递增 k 检查相对误差不增加，且所选三个 k 的因子标量数均少于原图。
    assert error <= previous and storage < 1
    # 核对截断 奇异值分解 的误差=被舍弃奇异值的二范数，验证重建实现与理论一致。
    assert np.isclose(np.linalg.norm(image - approximation, "fro"),
                      np.linalg.norm(s[k:]))
    # 记录本次误差，供下一个更大的 k 检查单调性。
    previous = error
    # 重建图使用与原图完全相同的灰度范围，避免自动调色掩盖亮度误差。
    ax.imshow(approximation, cmap="gray", vmin=0, vmax=1)
    # 在图题中标明截断秩和百分比相对误差，便于图像与数字对应。
    ax.set_title(f"k={k}, 误差={error:.1%}")
# 隐藏像素坐标和边框，让读者集中比较图像内容。
for ax in axes: ax.axis("off")
# 调整子图间距后保存图片并关闭图对象。
```



```
fig.tight_layout()
fig.savefig("e30_image.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e31_pca.py

附件文件 e31_pca.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e31_pca.py

```
# 教材位置：第 9 章，第 9.9 节；图 9-5。
# 单位、变量与形状：X、Xc、scores 为 (300,2)；特征单位统一，得分沿用该单位，方差比例无量纲。
# 案例 31：从中心化、样本协方差和 奇异值分解 三个角度理解 主成分分析。
# 学习目标：提取二维相关数据中方差最大的正交方向，理解主成分得分与解释方差比例。
# 核心关系：Xc=X-列均值，C=XcTXc/(N-1)；若 Xc=USVT，
# 则 C=V diag(s2/(N-1)) VT，scores=XcV=US，不同得分列之间的样本协方差为零。
# 数据与形状：N=300 个样本、2 个特征；X/Xc/scores 均为 (300,2)，C 为 (2,2)。
# 本例只中心化，没有按标准差标准化；主成分分析 方向因此依赖原特征的量纲与尺度。
# 输出：打印两个主成分的方差比例，验证等价关系，生成 e31_pca.png。
# 运行方式：安装依赖后在本目录执行 python e31_pca.py，图片保存到当前工作目录。

# NumPy 用于模拟相关变量、样本协方差、对称特征分解与 奇异值分解。
import numpy as np
# 左图展示中心化数据和主轴，中图展示旋转到主轴坐标系后的得分，右图展示解释率。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 5，便于复现相同的点云和解释方差比例。
rng = np.random.default_rng(5)
# 生成 300 个共同潜在值，均值 0.5、标准差 0.18；两个特征共用它来制造相关性。
base = rng.normal(0.5, 0.18, 300)
# 第一特征为 base 加标准差 0.05 的独立噪声，第二特征为 0.8*base+0.25 加噪声。
# column_stack 把两个长度 300 的数组按列拼成 (300,2)：行是样本，列是特征。
X = np.column_stack([base + rng.normal(0, 0.05, 300),
                     0.8 * base + 0.25 + rng.normal(0, 0.06, 300)])
# axis=0 沿样本行求各列均值，形状 (2,) 自动广播到 300 行。
# 先移除均值才能让 主成分分析 描述围绕样本中心的变化；未中心化的 奇异值分解 会混入均值位置。
Xc = X - X.mean(axis=0)
N = len(X)
# (2,300)@(300,2) 得到 2x2 散布矩阵；除以 N-1 采用样本协方差口径。
# 对角线是各特征样本方差，非对角线是两特征样本协方差。
C = Xc.T @ Xc / (N - 1)
# C 为实对称矩阵，eigh 返回升序特征值，directions 的各列为相应的标准正交方向。
values, directions = np.linalg.eigh(C)
# 同时反转特征值和特征向量列，使第一列对应最大方差方向；保持配对关系不能只反转一方。
values = values[::-1]
directions = directions[:, ::-1]
# 中心化矩阵的精简 奇异值分解 形状为 U(300,2)、s(2,)、Vh(2,2)，奇异值已按降序排列。
U, s, Vh = np.linalg.svd(Xc, full_matrices=False)
# V 的列为特征空间里的主轴；Xc@V 把每个样本投影到这些轴上，得到主成分得分。
# 每轴方差为 sj2/(N-1)，求比例时公因子 N-1 抵消，因此 ratio=s2/Σs2。
V = Vh.T
```

```

scores = Xc @ V
ratio = s ** 2 / (s @ s)
# 两个比例之和为 1, 表示保留各主成分能解释多少中心化数据的总样本方差。
print("主成分方差比例 =", np.round(ratio, 6))
# 核对协方差特征值与中心化数据奇异值平方之间的关系, 分母保持 N-1 一致。
assert np.allclose(values, s ** 2 / (N - 1))
# 得分均值仍为零, 因此该式正是得分样本协方差矩阵。
# 非对角元接近零说明主成分线性不相关; 这通常不意味着统计独立。
assert np.allclose(scores.T @ scores / (N - 1), np.diag(values))
# 本例存在明显的最大方差主方向; 特征值不等也使该方向除正负号外唯一。
assert values[0] > values[1]
# 特征分解与 奇异值分解 可能为同一条主轴返回相反方向, 直接比较向量会误判。
# 比较  $\mathbf{v}\mathbf{v}^T$  投影矩阵可以消除符号差异, 因为  $(-\mathbf{v})(-\mathbf{v})^T = \mathbf{v}\mathbf{v}^T$ 。
assert np.allclose(np.outer(directions[:, 0], directions[:, 0]),
                    np.outer(V[:, 0], V[:, 0]))
#  $\mathbf{XcV} = \mathbf{US}$ ;  $\mathbf{U}^*\mathbf{s}$  用广播把每列  $\mathbf{U}$  乘以对应奇异值, 形状仍为 (300,2)。
assert np.allclose(scores, U * s)
# 创建中心化原坐标、主成分得分与解释率三幅图; 全都使用上面同一批 300 个样本。
fig, axes = plt.subplots(1, 3, figsize=(6.3, 3.0))
#  $\mathbf{Xc.T}$  的两行被展开为散点纵横坐标; 小点和半透明可减轻重叠遮挡。
axes[0].scatter(*Xc.T, s=8, alpha=0.4)
# 分别绘制 PC1 与 PC2; 两条方向线应正交。
for j in range(2):
    # 特征值是该轴方差, 其平方根为标准差; 取两倍标准差作为每一侧示意轴长度。
    v = 2 * np.sqrt(values[j]) * V[:, j]
    # 从  $-\mathbf{v}$  画到  $+\mathbf{v}$ , 线段经过中心化后的原点; 它只是主轴示意, 并非置信区间。
    axes[0].plot([-v[0], v[0]], [-v[1], v[1]], label=f"主成分 {j + 1}")
# 左图纵横单位等长, 保证图上的主轴角度忠实反映几何正交关系。
axes[0].set_aspect("equal")
axes[0].legend(fontsize=7.8)
# 左图数据中心在原点, 展示中心化后的原特征坐标。
axes[0].set(xlabel="中心化特征 1", ylabel="中心化特征 2", title="中心化数据")
# 中图以 PC1、PC2 得分为纵横坐标, 点云的主要变化沿 PC1 方向展开。
axes[1].scatter(*scores.T, s=8, alpha=0.4)
axes[1].set(xlabel="主成分 1 得分", ylabel="主成分 2 得分", title="主成分得分")
# 得分图也采用等比例坐标, 避免纵横自动缩放改变读者对主次变化的观感。
axes[1].set_aspect("equal")
# 右图直接读取已算出的 ratio; 不重新生成点云或用另一套特征值。
axes[2].bar([1, 2], ratio, width=0.55, color=["tab:blue", "tab:orange"])
axes[2].set_xticks([1, 2], ["主成分 1", "主成分 2"])
axes[2].set(ylabel="方差比例", ylim=(0, 1.1), title="解释方差比例")
for index, value in enumerate(ratio, 1):
    axes[2].text(index, value + 0.035, f"{value:.2%}", ha="center", fontsize=8)
# 为各幅图添加淡网格, 帮助比较点云位置、主轴方向与方差份额。
for ax in axes:
    ax.grid(alpha=0.3)
    ax.tick_params(labelsize=7)
# 调整布局、保存图片并关闭图对象, 以支持脚本批量执行。
fig.tight_layout()
fig.savefig("e31_pca.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e32_lstsq.py

附件文件 e32_lstsq.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e32_lstsq.py

```
# 教材位置：第 7 章，第 7.9 节；图 7-4。
# 单位、变量与形状：t、y、yhat、residual 为 (30,)；t 和 y 使用各自一般单位，均方根误差沿
# 用 y 的单位。
# 案例 32：线性最小二乘拟合、残差正交与秩亏时的最小范数解。
# 学习目标：把拟合直线  $y \approx at + b$  写成  $A\beta \approx y$ ，理解最小化平方误差的投影几何。
# 核心关系：满列秩时  $\beta^*$  为唯一最小二乘解， $r = y - A\beta^*$  满足  $A^T r = 0$ ；
# 秩亏时最小二乘解可能不唯一，lstsq 返回其中欧氏范数最小的一组系数。
# 数据与形状：30 个模拟观测，A 为 (30,2)、coef 为 (2,)，列顺序是斜率项和截距项。
# 输出：打印斜率/截距、秩、样本内  $R^2$ /均方根误差、秩亏解；生成 e32_lstsq.png。
# 运行方式：安装依赖后在本目录执行 python e32_lstsq.py。
# 图中灰色线是竖直观测残差； $A^T r = 0$  的正交发生在 30 维观测向量空间。

# NumPy 提供最小二乘、伪逆及用于验证残差几何的矩阵运算。
import numpy as np
# Matplotlib 把观测、拟合直线以及各点的竖直残差画在一起。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 21，使数据与拟合结果可以重复核对。
rng = np.random.default_rng(21)
# 随机生成 30 个自变量位置；它们未排序，拟合本身不要求观测按 t 排序。
t = rng.uniform(1, 12, 30)
# 真实模拟直线截距 8.5、斜率 -0.55，叠加均值 0、标准差 0.8 的独立高斯噪声。
y = 8.5 - 0.55 * t + rng.normal(0, 0.8, 30)
# 设计矩阵第 1 列为 t，第 2 列全为 1；若 coef=[a,b]，则 A@coef=a*t+b。
# 加常数列使模型能拟合非零截距；不加该列会强制直线通过原点。
A = np.column_stack([t, np.ones_like(t)])
# 直接用数值最小二乘求解，避免显式构造  $(A^T A)^{-1} A^T y$  而放大条件数问题。
# 返回系数、残差平方和数组、数值秩、奇异值；rcond=None 使用默认奇异值截断准则判断秩。
# 本例是观测数多于参数数且满列秩，故 sums 含一个残差平方和；秩亏时该数组可为空。
coef, sums, rank, singular = np.linalg.lstsq(A, y, rcond=None)
# yhat 是 30 个拟合值；residual=r 是原观测与投影的差，正号表示观测高于拟合线。
yhat = A @ coef
residual = y - yhat
#  $R^2 = 1 - \text{SSE} / \text{SST}$ ，将拟合误差与“所有观测均用样本均值预测”的误差比较。
# 这里计算的是带截距模型的样本内拟合指标，不代表新数据预测表现。
r2 = 1 - (residual @ residual) / np.sum((y - y.mean()) ** 2)
# 按设计矩阵列顺序打印斜率、截距，数值秩为 2 表示两列提供独立信息。
print("斜率、截距 =", np.round(coef, 6), "; 秩 =", rank)
# 均方根误差= $\sqrt{\text{SSE}/30}$ ，是每个观测的均方误差开方。
# 它不同于回归噪声标准差估计  $\sqrt{\text{SSE}/(30-2)}$ ，后者要扣除拟合的两个参数自由度。
print("决定系数  $R^2$ 、均方根误差 =", round(r2, 6),
      round(np.sqrt(np.mean(residual ** 2)), 6))
# 验证满列秩及残差与设计矩阵每一列都正交，即最小二乘的一阶条件  $A^T r = 0$ 。
# 由于含全 1 的截距列，这也意味着残差和接近零；不要求每个残差都为零。
assert rank == 2 and np.allclose(A.T @ residual, 0, atol=1e-9)
# 将 lstsq 返回的残差平方和与自行计算的  $r^T r$  比较，核对返回值的含义。
assert np.isclose(sums[0], residual @ residual)
# 构造两列完全相同的 (3,2) 秩亏矩阵；只能识别两个系数的和，无法分别识别。
```



```

D = np.array([[1.0, 1.0], [2.0, 2.0], [3.0, 3.0]])
# 右端恰为 D 的任意一列, 因此所有满足  $\beta_1 + \beta_2 = 1$  的系数组合都能实现零残差。
b = np.array([1.0, 2.0, 3.0])
# lstsq 在无穷多个零残差解中选择  $\|\beta\|_2$  最小者; 下划线表示这里不使用对应返回项。
# 在  $\beta_1 + \beta_2 = 1$  约束下, 平均分配为 (0.5, 0.5) 时平方范数最小。
minimum, _, deficient_rank, _ = np.linalg.lstsq(D, b, rcond=None)
# 检查秩为 1, 并确认结果是对称分配的最小范数解。
assert deficient_rank == 1 and np.allclose(minimum, [0.5, 0.5])
# 穆尔-彭罗斯 伪逆  $D^+$  同样给出最小范数最小二乘解, 即  $\beta = D^+ b$ 。
assert np.allclose(minimum, np.linalg.pinv(D) @ b)
# 单独打印秩亏示例, 帮助区分“拟合值唯一”和“系数解唯一”这两个问题。
print("秩亏矩阵的最小范数解 =", minimum)
# 观测 y、拟合 yhat 与残差 residual 都在  $\mathbb{R}^{30}$ ; 但它们共同落在一个至多三维的子空间。
# 先把 A 的两列与实际 residual 并列, 再做约化 QR, 得到这个子空间的标准正交基。
# 本例 A 满列秩且残差非零, basis 的形状为 (30, 3); 不是随意指定的三维示意坐标。
basis, triangular = np.linalg.qr(np.column_stack([A, residual]), mode="reduced")
observed_coordinates = basis.T @ y
fitted_coordinates = basis.T @ yhat
residual_coordinates = basis.T @ residual
# 在标准正交基中取坐标保持长度、内积与正交关系; 重建应恢复实际 30 维向量。
assert basis.shape == (30, 3) and np.allclose(basis.T @ basis, np.eye(3))
assert np.allclose(basis @ observed_coordinates, y)
assert np.allclose(basis @ fitted_coordinates, yhat)
assert np.allclose(basis @ residual_coordinates, residual)
assert np.allclose(observed_coordinates, fitted_coordinates + residual_coordinates)
assert np.allclose(fitted_coordinates[2], 0, atol=1e-10)
assert np.allclose(residual_coordinates[:2], 0, atol=1e-10)
assert np.isclose(fitted_coordinates @ residual_coordinates, 0, atol=1e-9)
assert np.isclose(np.linalg.norm(residual_coordinates), np.linalg.norm(residual))
# 创建拟合图, 左图横轴为自变量 t, 纵轴为响应 y; 右图是实际观测子空间的 QR 坐标。
fig = plt.figure(figsize=(6.3, 3.2))
ax = fig.add_subplot(1, 2, 1)
# 散点不必排序; 另外取得按 t 升序的索引, 以便画线时从左到右连接。
ax.scatter(t, y, label="模拟观测", s=18)
order = np.argsort(t)
# 按同一索引排列 t 与拟合值, 避免原始随机顺序造成来回折返的线段。
ax.plot(t[order], yhat[order], "r-", label="最小二乘拟合")
# 在固定 t 上连接 y 与 yhat, 画的是竖直方向的回归残差。
# 最小二乘的残差正交是 r 与 A 的列空间在观测空间中正交, 并非图上灰线垂直于斜拟合线。
ax.vlines(t, y, yhat, color="gray", alpha=0.5)
# 标题用 A.T@residual=0 提醒读者区分散点图中的竖直距离与向量空间中的正交。
ax.set(xlabel="t", ylabel="y", title="观测值的竖直残差")
# 图例区分观测与拟合线, 淡网格便于读出纵向残差大小。
ax.legend(fontsize=7.8)
ax.grid(alpha=0.3)
ax = fig.add_subplot(1, 2, 2, projection="3d")
# 前两条 QR 基张成 Col(A), 所以拟合平面在第三坐标为 0 处; 平面范围由实际拟合坐标决定。
# 以下只是设置显示范围, 没有对坐标向量重标度, 也没有改动原观测数据。
extent = 1.2 * max(np.max(np.abs(observed_coordinates)), 1.0)
plane_axis = np.linspace(-extent, 0.2 * extent, 2)
plane_x, plane_y = np.meshgrid(plane_axis, plane_axis)
ax.plot_surface(plane_x, plane_y, np.zeros_like(plane_x), color="tab:green",
alpha=0.18)
# y 是从原点到实际观测坐标的向量, yhat 是其平面投影, r 从投影端点指向观测端点。
ax.quiver(0, 0, 0, *observed_coordinates, color="tab:blue", arrow_length_ratio=0.07)
ax.quiver(0, 0, 0, *fitted_coordinates, color="tab:green", arrow_length_ratio=0.07)
ax.quiver(*fitted_coordinates, *residual_coordinates, color="tab:orange",
arrow_length_ratio=0.18)

```

```
ax.text(*observed_coordinates, " $y$", color="tab:blue", fontsize=9)
# 标签做纯显示偏移, 避免与拟合向量端点和直角记号重叠; 向量坐标仍保持原值。
fitted_label = fitted_coordinates + np.array([0.16 * extent, 0.03 * extent,
        -0.15 * np.linalg.norm(residual_coordinates)])
ax.text(*fitted_label, " $\hat{y}$", color="tab:green", fontsize=9)
ax.text(*(fitted_coordinates + residual_coordinates / 2), " $r$",
        color="tab:orange", fontsize=9)
# 直角记号沿拟合平面的一条单位坐标轴与实际残差方向绘制, 表示  $A^T r = 0$ 。
marker_size = np.linalg.norm(residual_coordinates) * 0.18
normal = residual_coordinates / np.linalg.norm(residual_coordinates)
in_plane = np.array([1.0, 0.0, 0.0])
right_angle = np.array([fitted_coordinates + marker_size * in_plane,
        fitted_coordinates + marker_size * (in_plane + normal),
        fitted_coordinates + marker_size * normal])
ax.plot(*right_angle.T, color="gray", lw=0.8)
ax.set_title("实际观测向量的子空间", fontsize=10)
ax.set(xlim=(-extent, 0.2 * extent), ylim=(-extent, 0.2 * extent),
        zlim=(-1.0, 1.5 * abs(residual_coordinates[2]) + 0.5),
        xlabel="$q_1$", ylabel="$q_2$", zlabel="$q_3$")
ax.set_box_aspect((1, 1, 0.5))
ax.set_proj_type("ortho")
ax.view_init(elev=24, azim=-54)
ax.tick_params(labelsize=6, pad=0)
# 一般最小二乘公式中的  $b$  在本回归中就是  $y$ ; 代码较早的  $b=[1, 2, 3]$  属于另一秩亏例子。
ax.text2D(0.02, 0.01, "实际正交基坐标;  $A^T r = 0$ ", transform=ax.transAxes, fontsize=7.8)
# 自动调整边距、保存图片并关闭图对象; tight 边框防止三维轴标签被裁切。
fig.tight_layout()
fig.savefig("e32_lstsq.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e33_pm25.py

附件文件 e33_pm25.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e33_pm25.py

```
# 教材位置: 第 10 章, 第 10.3 节; 图 10-1。
# 单位、变量与形状: pm、group 为 (365,); pm 按  $\mu\text{g}/\text{m}^3$  计, 图中概率密度按  $(\mu\text{g}/\text{m}^3)^{-1}$  计。
# 案例 33: 模拟 PM2.5 浓度的分组对数正态分布与混合模型。
# 学习目标: 认识偏态、分组异质性和尾部概率, 并区分经验超阈值次数与模型期望次数。
# 本例全部数据由随机数模拟, 不代表真实监测结论; 75 只是教学阈值, 不是法律标准。
# 核心关系: 若  $\log(Y) \sim N(\mu, \sigma^2)$ , 则  $Y$  服从对数正态; 混合生存函数为  $\sum w_j S_j(Y)$ 。
# 数据: 三组分别有 120、125、120 个值, 共 365 个; 组内模拟中位数为 55、32、22。
# 输出: 单一对数正态模型经模拟校准的 拟合优度检验、均值/中位数、超阈值计数,
# 并生成 application_pm.png, 比较整体拟合、分组混合密度与组 0 的对数 分位数对比 图。
# 运行方式: 安装本目录依赖后执行 python e33_pm25.py; 拟合优度模拟可能需要一些时间。
# 所有图保存到当前工作目录; 固定随机种子便于在相同环境下复现模拟结果。

# NumPy 负责生成模拟分组与随机浓度, 并做数组级统计。
import numpy as np
# SciPy 提供对数正态拟合、分布尾概率、拟合优度模拟校准与 分位数对比 图。
from scipy import stats
# Matplotlib 用于绘制密度比较图和 分位数对比 图。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot
```

```

configure_chinese_plot()
# 同一个随机数生成器依次用于生成数据和检验模拟；固定种子使整个流程可复现。
rng = np.random.default_rng(2024)
# 构造长度 365 的组标签，0/1/2 分别重复 120/125/120 次。
# 分组可理解为三种教学情境；没有真实日期信息，不应直接据此解释季节变化。
group = np.repeat(np.arange(3), [120, 125, 120])
# 按每个样本的组标签选取参数；lognormal 的 mean、sigma 指对数尺度的  $\mu$ 、 $\sigma$ 。
# 因此 55、32、22 是各组在原始浓度尺度的中位数，而不是算术均值；
# 算术均值为  $\exp(\mu + \sigma^2/2)$ ，通常高于中位数，体现右偏分布。
pm = rng.lognormal(np.log(np.array([55, 32, 22])[group]),
                  np.array([0.25, 0.22, 0.20])[group])
# 对数正态样本应是有限正数；检查正值也保证后面 np.log(pm) 有定义。
assert np.all(np.isfinite(pm)) and np.all(pm > 0)
# 先对全部 365 个值拟合一个对数正态，再把参数传回构造冻结分布对象。
# floc=0 固定位置参数为零；fit 返回 (shape, loc, scale)，其中 shape= $\sigma$ 、scale= $\exp(\mu)$ 。
# 把异质分组合并拟合只是一个对照模型，并不保证合并分布仍服从单一对数正态。
whole = stats.lognorm(*stats.lognorm.fit(pm, floc=0))
# 按已知组标签分别拟合 3 个对数正态模型；这是利用分组信息，不是无标签混合模型的自动识别
# 。
parts = [stats.lognorm(*stats.lognorm.fit(pm[group == j], floc=0))
         for j in range(3)]
# 布尔数组的均值等于该组样本比例，得到 120/365、125/365、120/365，权重和为 1。
weights = np.array([(group == j).mean() for j in range(3)])
# 检验原假设是“总体样本可由一个 loc=0 的对数正态描述”，而不是检验分组混合模型。
# 拟合优度检验 统计量比较经验 累积分布函数 与拟合 累积分布函数 的最大差；参数由同一数据
# 估计时，
# 不能直接把普通已知参数 拟合优度检验 的参考分布当作正确校准。
# goodness_of_fit 进行 499 次蒙特卡洛/参数自助模拟，对每份模拟样本重新估计未知参数，
# 同时始终把 loc 固定为 0；返回经这种过程校准的模拟 p 值。模拟次数有限，p 值有随机误差。
test = stats.goodness_of_fit(stats.lognorm, pm, known_params={"loc": 0},
                             statistic="ks", n_mc_samples=499, rng=rng)
# 仅设定课堂示例的浓度阈值 75.0；此数值不用于声称满足任何地区或时间口径的标准。
limit = 75.0 # 教学演示阈值，并非法定限值。
# sf(limit)=P(Y>limit) 为生存函数，直接计算通常比 1-cdf 更稳定。
# 按组权重求和得到混合模型的总体超阈值概率；连续分布中 > 与  $\geq$  的概率相同。
tail = sum(w * d.sf(limit) for w, d in zip(weights, parts))
# D 越大表示偏离拟合 累积分布函数 越大；p 值衡量该单一分布假设下统计量的尾部程度，不是原
# 假设为真的概率。
print(f"拟合优度统计量 D={test.statistic:.4f}, 校准 p 值={test.pvalue:.4f}")
# 经验算术均值和经验中位数从同一批 365 个模拟值计算，比较它们可观察右偏与组间差异。
print(f"均值={pm.mean():.2f}, 中位数={np.median(pm):.2f}")
# 这是当前模拟样本中实际严格超过阈值的个数，为整数，并不等于模型期望次数。
print(f"观察到的超阈值次数={np.count_nonzero(pm > limit)}")
# 以 365 个观测为口径，将单次超阈值概率乘以 365 得到期望计数。
# 分别使用整体模型和分组混合模型；期望值可为小数，也不保证与本次观察计数完全相等。
print(f"期望超阈值次数：整体模型={365 * whole.sf(limit):.2f}, "
      f"分组模型={365 * tail:.2f}")
# 在正浓度范围上取 500 个点画连续密度曲线，右端稍超过样本最大值。
x = np.linspace(1, 1.1 * pm.max(), 500)
# 左图检查整体分布形状，右图单独检查组内对数正态假设。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# density=True 将柱形面积归一到 1，纵轴为密度而非频数，才能与连续 pdf 直接叠加。
axes[0].hist(pm, bins=24, density=True, alpha=0.4, label="模拟数据")
# 绘制合并数据得到的单一对数正态拟合密度，作为基准模型。
axes[0].plot(x, whole.pdf(x), label="单一对数正态模型")
# 混合密度  $f(x) = \sum w_j f_j(x)$ ，与超阈值概率的加权方式保持一致。
axes[0].plot(x, sum(w * d.pdf(x) for w, d in zip(weights, parts)),

```

```

        label="分组混合模型")
# 横轴标明浓度单位  $\mu\text{g}/\text{m}^3$ ，纵轴密度的积分才是概率。
axes[0].set(title="整体拟合与分组拟合",
            xlabel="浓度 ( $\mu\text{g}/\text{m}^3$ )", ylabel="概率密度")
# 较小字号的图例区分数据直方图、单一模型及分组混合模型。
axes[0].legend(fontsize=8)
# 若组 0 的浓度对数正态，则其对数应近似正态，因此用正态概率图检查  $\log(\text{pm})$ 。
# 点接近直线支持该组对数分布的形状；该图不是对总体混合分布的正态性检验。
stats.probplot(np.log(pm[group == 0]), dist="norm", plot=axes[1])
# 明确右图只展示组 0，且纵轴为取对数后的排序数值，避免与原浓度单位混淆。
axes[1].set(title="第 0 组：对数浓度概率图", xlabel="理论顺序统计量中位数",
            ylabel="排序后的对数浓度")
# 自动调整左右图边距，避免轴标签与图题拥挤。
fig.tight_layout(pad=0.8, w_pad=1.2)
# 以 180 dpi 保存应用案例图片到当前工作目录。
# bbox_inches="tight" 使面板标题和坐标说明完整保存在 PNG 中；教材直接使用此文件。
fig.savefig("application_pm.png", dpi=180, bbox_inches="tight")
# 保存后关闭图对象，避免批量执行多个案例时持续占用绘图资源。
plt.close(fig)

```

程序 e34_count.py

附件文件 e34_count.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e34_count.py

```

# 教材位置：第 10 章，第 10.4 节；图 10-2。
# 单位、变量与形状：counts 为 (60,)，null 为 (2000,60)；计数按次计，报警阈值按次计。
# 案例 34：计数数据的过度离散、负二项矩估计与离散报警阈值。
# 学习目标：理解泊松模型要求均值等于方差，及率的异质性如何使合并数据方差大于均值。
# 核心关系：SciPy 负二项参数  $r, p$  下  $E[X]=r(1-p)/p$ 、 $\text{Var}(X)=r(1-p)/p^2$ ；
# 用样本均值  $m$ 、样本方差  $v>m$  匹配矩可得  $r=m^2/(v-m)$ 、 $p=m/v$ 。
# 数据：共 60 个等暴露量单位，前 45 个泊松率 6、后 15 个泊松率 13，全部为模拟计数。
# 报警基准另外指定为  $\text{Poisson}(6)$ ，阈值控制该已知基准下  $P(X \geq \text{threshold}) \leq 0.02$ 。
# 输出：统计量、模拟尾部比例、负二项参数及阈值；生成 application_count.png。
# 运行方式：安装依赖后在本目录执行 python e34_count.py，图保存到当前工作目录。

# NumPy 用于泊松模拟、样本统计、矩估计和离散横坐标。
import numpy as np
# SciPy 的冻结分布对象提供均值、方差、概率质量函数、分位数与生存函数。
from scipy import stats
# Matplotlib 比较经验计数分布与模型，并画出零假设模拟统计量分布。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 99；同一生成器依次产生观察样本及零假设模拟样本。
rng = np.random.default_rng(99)
# np.r_ 拼接 45 个 6 与 15 个 13，作为 60 个单位各自的泊松均值。
# 每组内部是泊松，但合并后的异质性会产生额外变异；该生成机制并非直接从负二项抽样。
counts = rng.poisson(np.r_[np.full(45, 6.0), np.full(15, 13.0)])
# mean 为样本均值，var 采用分母  $n-1$  的样本方差，ddof=1 明确这一统计口径。
mean, var = counts.mean(), counts.var(ddof=1)
# 离散指数=样本方差/样本均值；接近 1 符合泊松的量级，大于 1 提示过度离散。
ratio = var / mean

```



```

# 在“60 个单位共享同一泊松率”的零假设下模拟 2000 份样本，每份长度 60。
# 这里用观察样本的 mean 代入未知率，属于拟合零假设下的参数模拟诊断，非精确有限样本检验。
null = rng.poisson(mean, size=(2000, counts.size))
# 每行是一份模拟样本；沿 axis=1 算各自样本方差和均值，得到 2000 个离散指数。
null_ratio = null.var(axis=1, ddof=1) / null.mean(axis=1)
# 计算模拟统计量至少与观察值一样大的比例，选择右尾来检查过度离散。
# 分子与分母都加 1 是常用蒙特卡洛校正，避免有限次模拟给出零 p 值；该值仍有模拟误差。
p_sim = (1 + np.count_nonzero(null_ratio >= ratio)) / (len(null) + 1)
# 负二项的这种矩估计要求  $v > m$ ；不满足时公式会给非正或无穷的  $r$ ，不能继续套用。
if var <= mean:
    # 明确报错可防止把不满足参数条件的数据悄悄传入分布对象。
    raise ValueError("负二项矩估计要求样本方差大于样本均值。")
# 把样本均值和 ddof=1 样本方差代入负二项矩方程； $r$  可以是正实数，不必为整数。
#  $r$  为正整数时， $X$  可解释为成功  $r$  次之前的失败数， $p$  为单次成功概率；
# 本例  $r$  为矩估计得到的正实数，应按泊松率的 伽马分布 混合解释，不能说完成了非整数次成功
# 。
r, p = mean**2 / (var - mean), mean / var
# 冻结负二项分布，后面通过 nb.mean()/var()/pmf() 重复使用同一组参数。
nb = stats.nbinom(r, p)
# 矩估计应使模型理论均值和方差与采用的两个样本矩完全匹配到数值容差。
assert np.allclose([nb.mean(), nb.var()], [mean, var])
# 报警基准独立设为已知均值 6 的泊松分布，允许的单次右尾概率为 2%。
# 它与前面的整体泊松拟合或负二项拟合用途不同，阈值不由拟合的负二项模型产生。
baseline, alpha = stats.poisson(6), 0.02
# 离散 ppf(q) 返回使  $F(k) \geq q$  的最小整数  $k$ ；选择 threshold=k+1，
# 使报警事件  $X \geq \text{threshold}$  等价于  $X > k$ ，并令其概率  $1 - F(k) \leq \alpha$ 。
threshold = int(baseline.ppf(1 - alpha)) + 1
# 离散 sf(a)= $P(X > a)$ ，所以  $P(X \geq \text{threshold})$  必须写 sf(threshold-1)，不能写 sf(threshold)。
assert baseline.sf(threshold - 1) <= alpha
# 再把报警阈值降低 1 就会使右尾概率超过 alpha，检验当前阈值确为满足条件的最小整数。
assert baseline.sf(threshold - 2) > alpha
# 并列显示均值、样本方差、离散指数及单侧模拟尾部比例，联系效应大小与参考分布。
print(f"均值={mean:.3f}，方差={var:.3f}，离散指数={ratio:.3f}，模拟 p 值={p_sim:.4f}")
# 显示负二项矩参数与独立基准阈值；两个结果服务于分布描述和报警两个不同问题。
print(f"负二项参数 r={r:.3f}，p={p:.3f}；报警阈值={threshold}")
# 报告基准模型下的实际单次报警概率；离散跳跃使其一般小于而非恰好等于 0.02。
print(f" $P(X \geq \text{threshold})$ ={baseline.sf(threshold - 1):.6f}")
# 建立从 0 开始的整数网格，并在观察最大值之后延伸几个取值，便于展示模型尾部。
k = np.arange(counts.max() + 8)
# 左图比较计数概率质量，右图比较离散指数在零假设下的模拟分布。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# 半整数边界使每个宽度为 1 的柱正好包含一个整数计数。
# 在单位箱宽下，归一化密度的柱高可与对应计数的相对频率、离散 概率质量函数 直接比较。
axes[0].hist(counts, bins=np.arange(-0.5, k[-1] + 1.5), density=True, alpha=0.4)
# 用全部数据的样本均值拟合单一泊松，展示其相对较窄的计数分布。
axes[0].plot(k, stats.poisson.pmf(k, mean), "o-", ms=3, label="泊松分布")
# 叠加负二项 概率质量函数，观察允许方差大于均值后分布宽度与尾部的变化。
axes[0].plot(k, nb.pmf(k), "s-", ms=3, label="负二项分布")
# 强调暴露量相同；若观察时长或面积不同，原始计数通常不能直接以同一泊松率比较。
axes[0].set(title="计数模型比较",
            xlabel="每个等暴露单元的计数",
            ylabel="概率质量 / 相对频率")
# 图例对应两个模型曲线，灰蓝色半透明柱对应观察频率。
axes[0].legend(fontsize=8)
# 右图纵轴为 2000 次模拟的频数，没有 density=True，不要把柱高直接当作概率密度。
axes[1].hist(null_ratio, bins=30, alpha=0.6)
# 红线标出观察离散指数；位于模拟分布右侧越远，零假设下出现的机会越小。
axes[1].axvline(ratio, color="red", label="观察到的离散指数")

```

```
# 横轴明确是均一泊松零假设下的方差/均值统计量，不是原始计数。
axes[1].set(title="泊松原假设下的离散指数",
            xlabel="泊松原假设下的方差 / 均值", ylabel="频数")
# 图例标识红色观察值参考线，便于与模拟频数直方图比较。
axes[1].legend(fontsize=8, framealpha=1)
# 调整布局并以 180 dpi 保存应用图。
fig.tight_layout(pad=0.8, w_pad=1.2)
# 保存完整标题与轴标签：教材直接嵌入此 PNG，并与教材的中文说明保持对应。
fig.savefig("application_count.png", dpi=180, bbox_inches="tight")
# 保存后释放图对象，便于连续运行其他案例。
plt.close(fig)
```

程序 e35_land.py

附件文件 e35_land.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e35_land.py

```
# 教材位置：第 10 章，第 10.5 节；图 10-3。
# 单位、变量与形状：P 为 (3,3)，x0 为 (3,)，pred 为 (2000,3)；面积占比无量纲，时间步按年
# 计。
# 案例 35：土地类型马尔可夫转移、期望占比与转移矩阵的情景不确定性。
# 学习目标：区分等面积单元的多项抽样、固定矩阵的十步期望，以及扰动转移矩阵后的情景分布。
# 核心关系：行向量  $x_{t+1}=x_tP$ ，固定矩阵十步期望为  $x_0P^{10}$ ；稳态满足  $\pi P=\pi$ 。
# 数据与形状：三类土地，P 为 3×3，x0 为长度 3 的初始面积比例；
# 各行分别抽 狄利克雷(300*P[i]) 得到情景矩阵 Q，每个情景内固定 Q 连续推演 10 步。
# 2000 份情景结果 pred 为 (2000,3)，报告各分量的 5%、50%、95% 边际分位数。
# 这些是给定扰动设定的情景分位区间，不是用观测样本推导的频率学置信区间；
# 也未指定后验数据模型，因此不能自动称为后验可信区间或真实土地变化预测区间。
# 输出：抽样计数、稳态、十步点值和情景分位数；生成 application_land.png。
# 运行方式：安装依赖后在本目录执行 python e35_land.py，图片保存到当前工作目录。

# NumPy 用于多项/狄利克雷 抽样、矩阵幂、特征分解以及情景分位数。
import numpy as np
# 左图比较点预测和情景区间，右图展示每类土地的边际情景直方图。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
# 固定种子 2026，先抽一次多项计数，再依次抽取 2000 个转移矩阵情景。
rng = np.random.default_rng(2026)
# 行与列的类型顺序均为农地、建设用地、其他用地；P[i,j] 是从类型 i 到 j 的一步概率。
# 所有元素都为正，使本例固定矩阵具有唯一稳态且长期期望分布收敛。
P = np.array([[0.88, 0.09, 0.03], [0.04, 0.93, 0.03], [0.02, 0.05, 0.93]])
# 初始三类土地面积占比分别为 42%、33%、25%；作为行概率向量右乘转移矩阵。
x0 = np.array([0.42, 0.33, 0.25])
# 验证每行是有效概率向量，保证乘法更新保持总面积比例为 1。
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
# 初始占比也必须非负且和为 1，否则不能解释为三类土地的完整组成。
assert np.all(x0 >= 0) and np.isclose(x0.sum(), 1)
# 单独示范：20000 个当前都属于第 0 类的等面积单元，下一步去向按 P[0] 多项抽样。
# 它不是按 x0 配置的全区域单元样本，也没有被用来拟合后面的 狄利克雷 参数。
cell_counts = rng.multinomial(20000, P[0])
# 打印三种去向的实现计数；其期望为 20000*P[0]，本次抽样一般有随机波动。
print("等面积单元下一年的去向计数：", cell_counts)
```

```
# 多项抽样把所有单元分到三个互斥且完备的去向，因此计数总和必须恰为 20000。
assert cell_counts.sum() == 20000
#  $\pi P = \pi$  转置成  $P^T \pi^T = \pi^T$ ，所以用  $P.T$  的右特征向量计算稳态。
vals, vecs = np.linalg.eig(P.T)
# 选择最接近特征值 1 的那一列；本例理论稳态为实数，取实部去除数值虚部。
# 特征向量尺度任意，除以分量和后才能解释为概率占比；steady 只对应固定的 P。
steady = np.real(vecs[:, np.argmax(np.abs(vals - 1))])
steady /= steady.sum()
# 检查归一化结果非负且满足固定矩阵的稳态方程，防止特征向量选错或方向用错。
assert np.all(steady >= 0) and np.allclose(steady @ P, steady)
# 计算固定 P 下第 10 步的期望面积占比，不是抽样得到的实际面积计数。
# 有限步点值与长期稳态概念不同，十步时不必已经达到 steady。
deterministic = x0 @ np.linalg.matrix_power(P, 10)
# 先用列表逐份收集情景结果，循环结束再转换为便于按列统计的二维数组。
pred = []
# 构造 2000 个独立的转移矩阵情景；下划线说明无需使用循环编号。
for _ in range(2000):
    # 狄利克雷 每次产生一行和为 1 的非负向量；vstack 把三行拼成 3×3 行随机矩阵。
    # 各行参数总和为 300，均值正好等于相应 P 行，单个分量方差为  $p_j(1-p_j)/301$ 。
    # 300 是人为设定的集中程度，越大扰动通常越小；没有在本例中从真实转移计数估计它。
    # 同一行的分量因总和固定而相关，不能视为三个独立概率。
    Q = np.vstack([rng.dirichlet(300 * row) for row in P])
    # 每份情景先抽一个 Q，再把这个 Q 固定使用 10 步，表达时间齐次转移参数的情景变化。
    # 此处没有逐年重新抽 Q，也没有模拟有限单元数的逐步状态抽样，结果仍是给定 Q 的期望占
    # 比。
    pred.append(x0 @ np.linalg.matrix_power(Q, 10))
# 转换为 (2000, 3) 数组：行是情景，列是第 10 步的三种土地类型占比。
pred = np.asarray(pred)
# 验证每一份情景结果都在概率单纯形上，即非负且三类比例合计为 1。
assert np.all(pred >= 0) and np.allclose(pred.sum(axis=1), 1)
# 沿情景行求每一类的 5%、中位数、95% 分位数，三个输出各为长度 3 的数组。
# 每一类的 lo~hi 是中央约 90% 的边际情景区间，不是均值置信区间。
# 三类边际中位数一般不保证相加为 1，各列分位端点也不代表同一份可共同实现的情景，
# 因此不能把三条边际区间解释成联合覆盖率 90% 的整体区域。
lo, mid, hi = np.quantile(pred, [0.05, 0.5, 0.95], axis=0)
# 报告固定 P 的长期稳态比例，显示到 4 位小数。
print("固定转移矩阵的稳态占比: ", np.round(steady, 4))
# 报告固定 P 的有限十步点值，便于与稳态和情景中位数分别比较。
print("固定转移矩阵的十步期望占比: ", np.round(deterministic, 4))
# 按三行输出 5%、50%、95% 分位数；每列对应一种土地类型。
print("边际情景分位数 (5%、50%、95%): \n",
      np.round([lo, mid, hi], 4))
# 创建情景分位区间图与情景分布图，两图使用同一份 pred 结果。
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2))
# 明确类型名称及顺序，使转移矩阵、初始向量、输出各列和图例保持对应。
names = ["农地", "建设用地", "其他用地"]
# errorbar 的 yerr 接收相对中心点的下/上偏移量，而不是直接接收上下端点。
# 误差棒以中位数为中心，允许上下不对称，表示边际情景分位区间而非标准误差或置信区间。
axes[0].errorbar(np.arange(3), mid, yerr=[mid - lo, hi - mid],
                 fmt="o", capsize=5)
# 叉号为固定 P 的十步期望，占比经非线性的  $Q^{10}$  变换后，情景中位数不必与叉号重合。
axes[0].scatter(np.arange(3), deterministic, marker="x", label="固定矩阵 P")
# 横轴只表示三类土地标签；纵轴是第十步期望占比，不是面积计数或抽样频率。
axes[0].set(title="第十步的边际情景区间",
            xticks=range(3), xticklabels=names,
            ylabel="期望面积占比", ylim=(0, 0.7))
# 图例标明固定矩阵点值，其余圆点与误差棒来自情景的中位数和分位数。
axes[0].legend(fontsize=8)
```

```
# 依次读取 pred 的三列，各列都是跨 2000 份情景的一维边际结果。
for j, name in enumerate(names):
    # 叠加三个半透明直方图；纵轴为情景数，未做概率密度归一化。
    axes[1].hist(pred[:, j], bins=30, alpha=0.4, label=name)
# 横轴强调是条件期望占比，右图宽度由人为设定的 狄利克雷 情景扰动决定。
axes[1].set(title="情景中的期望占比",
            xlabel="第十步期望面积占比", ylabel="情景频数")
# 添加类型图例并调整整体边距，避免长类型名称挤压坐标标签。
axes[1].legend(fontsize=8)
fig.tight_layout(pad=0.8, w_pad=1.2)
# 以 180 dpi 保存图并关闭对象，结束这个不依赖外部数据的应用示例。
# bbox_inches="tight" 防止面板标题和类别名称被裁切；教材直接嵌入此 PNG。
fig.savefig("application_land.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e36_normal_normal.py

附件文件 e36_normal_normal.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e36_normal_normal.py

```
# 教材位置：第 5 章，第 5.8 节；图 5-3。
# 已知观测标准差时的正态-正态共轭更新；本例完全解析，不随机抽样。
# 学习目标：把先验精度、样本信息和后验预测对应到可核对的计算。
# 观测、 $\mu$ 、 $\mu_0$ 、 $\tau_0$ 、 $\sigma$  使用同一测量单位；方差使用该单位的平方。
# data 形状为 (4,)，区间端点形状为 (2,)，所有密度共用原测量尺度。
# 参数： $\mu_0=50$ 、 $\tau_0=10$ 、 $\sigma=8$ ，四次观测为 52、48、55、51。
# 输出后验均值/方差/可信区间及下一次观测的预测区间，保存对应图片。
# 运行：安装 requirements.txt 中依赖后执行 python e36_normal_normal.py。

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

# 假设给定  $\mu$  后四个观测条件独立，同服从  $N(\mu, \sigma^2)$ 。
#  $\sigma=8$  是已知观测标准差，不用这四个观测重新估计它。
mu0, tau0, sigma = 50.0, 10.0, 8.0
data = np.array([52.0, 48.0, 55.0, 51.0])
n = data.size
sample_mean = data.mean()
# 精度是方差的倒数；独立正态信息在精度尺度上相加。
prior_precision = 1 / tau0**2
data_precision = n / sigma**2
posterior_var = 1 / (prior_precision + data_precision)
posterior_mean = posterior_var * (
    prior_precision * mu0 + data_precision * sample_mean
)
posterior_sd = np.sqrt(posterior_var)
# 后验均值是先验均值与样本均值的加权平均，权重和为 1。
data_weight = data_precision / (prior_precision + data_precision)
assert np.isclose(data_weight, 25 / 29)
assert np.isclose(posterior_mean, 2975 / 58)
assert np.isclose(posterior_var, 400 / 29)
prior = stats.norm(loc=mu0, scale=tau0)
posterior = stats.norm(loc=posterior_mean, scale=posterior_sd)
```



```
# 等尾可信区间按后验分布分位数给出, 是未知  $\mu$  的后验概率范围。
credible = posterior.ppf([0.025, 0.975])
# 预测还含一次新观测自身的噪声, 不能只用 posterior_sd 作预测标准差。
predictive_var = sigma**2 + posterior_var
predictive = stats.norm(loc=posterior_mean, scale=np.sqrt(predictive_var))
predictive_interval = predictive.ppf([0.025, 0.975])
assert predictive_var > posterior_var
print("样本量、样本均值: ", n, sample_mean)
print("先验精度、数据精度: ", prior_precision, data_precision)
print("后验均值、后验方差: ", posterior_mean, posterior_var)
print("后验标准差: ", posterior_sd)
print(" $\mu$  的 95% 等尾可信区间: ", credible)
print("下一次观测的预测均值、方差: ", predictive.mean(), predictive_var)
print("下一次观测的 95% 预测区间: ", predictive_interval)

# 同一后验在两个面板中回答不同问题: 参数  $\mu$  与未来观测。
# 图仅呈现以上解析分布, 不再生成一批样本以替代精确计算。
configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0), constrained_layout=True)
grid = np.linspace(15, 85, 700)
axes[0].plot(grid, prior.pdf(grid), label=" $\mu$  的先验")
axes[0].plot(grid, posterior.pdf(grid), label=" $\mu$  的后验")
axes[0].fill_between(
    grid, posterior.pdf(grid),
    where=(grid >= credible[0]) & (grid <= credible[1]), alpha=0.2,
)
axes[0].axvline(sample_mean, color="gray", ls=":", label="样本均值")
axes[0].set(title="未知均值的共轭更新", xlabel="未知均值  $\mu$ ", ylabel="概率密度")
axes[1].plot(grid, predictive.pdf(grid), label="下一次观测的预测密度")
axes[1].plot(grid, posterior.pdf(grid), ls="--", label=" $\mu$  的后验密度")
axes[1].fill_between(
    grid, predictive.pdf(grid),
    where=(grid >= predictive_interval[0]) & (grid <= predictive_interval[1]),
    alpha=0.2,
)
axes[1].set(title="预测需保留观测噪声", xlabel="下一次观测值", ylabel="概率密度")
for ax in axes:
    ax.legend(fontsize=7.5)
    ax.grid(alpha=0.2)
fig.savefig("normal_normal.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e37_poisson_process.py

附件文件 e37_poisson_process.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e37_poisson_process.py

```
# 教材位置: 第 4 章, 第 4.5 节; 图 4-3。
# 泊松过程的独立增量与等价间隔法, 模拟客服呼入的理想模型。
# 前提是恒定呼入率、独立增量; 真实数据的时段变化需另行检验。
# 时间按小时计, lam=3 次/小时, dt=1 小时, 每条路径覆盖 24 小时。
# increments 形状 (2000,24): 行是独立路径, 列是不重叠的时间窗。
# paths 形状 (2000,25), 包含第 0 时刻; 均值、方差数组形状 (24,)。
# 输出理论/模拟矩、相邻增量经验协方差及另一条间隔法路径的事件数。
# 固定种子 2026; 执行 python e37_poisson_process.py 保存中文四面板图。
```

```

import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

rng = np.random.default_rng(2026)
lam, dt, windows, repetitions = 3.0, 1.0, 24, 2000
#  $\Delta N_j \sim \text{Poisson}(\text{lam} * dt)$ , 不同窗及不同路径的抽样相互独立。
increments = rng.poisson(lam * dt, size=(repetitions, windows))
# 沿 axis=1 累积各行的时间窗, 最左补一个  $N(0)=0$ 。
paths = np.column_stack([np.zeros(repetitions), increments.cumsum(axis=1)])
times = np.arange(windows + 1) * dt
# 沿 axis=0 跨 2000 条独立路径求矩, 保留 24 个时间窗。
means = increments.mean(axis=0)
variances = increments.var(axis=0, ddof=1)
adjacent_cov = np.cov(increments[:, 0], increments[:, 1], ddof=1)[0, 1]
# 理论独立增量的协方差为 0; 有限模拟的经验协方差通常不恰为 0。
# 不对经验协方差断言等于 0, 也不从一次接近 0 推断真实独立性。
assert paths.shape == (repetitions, windows + 1)
assert np.array_equal(paths[:, 1:] - paths[:, :-1], increments)
assert np.all(np.diff(paths, axis=1) >= 0)
print("每窗理论均值、方差: ", lam * dt, lam * dt)
print("首窗模拟均值、样本方差: ", means[0], variances[0])
print("前两窗增量的经验协方差 (理论为 0): ", adjacent_cov)
print("24 小时计数的理论均值、方差: ", lam * windows, lam * windows)
print("24 小时计数的模拟均值、样本方差: ",
      paths[:, -1].mean(), paths[:, -1].var(ddof=1))

# 同一过程还可独立  $\text{Exp}(\text{lam})$  间隔生成; 尺度参数是  $1/\text{lam}$  小时。
# 逐次累计直至首次越过 24 小时, 避免预设个数不足而漏掉事件。
arrival_times = []
clock = 0.0
while True:
    clock += rng.exponential(scale=1 / lam)
    if clock > windows * dt:
        break
    arrival_times.append(clock)
arrival_times = np.asarray(arrival_times)
interval_path = np.searchsorted(arrival_times, times, side="right")
print("间隔法另一条路径的事件总数: ", arrival_times.size)
# 两种构造同分布, 但使用不同随机抽样, 不要求逐点相同。
assert np.all(np.diff(arrival_times) > 0)
assert interval_path[-1] == arrival_times.size

configure_chinese_plot()
fig, axes = plt.subplots(2, 2, figsize=(6.3, 5.1), constrained_layout=True)
for index in range(3):
    axes[0, 0].step(times, paths[index], where="post", label=f"路径 {index + 1}")
    axes[0, 0].plot(times, lam * times, "k--", label="理论均值  $3t$ ")
    axes[0, 0].set(title="独立增量构造", xlabel="时刻 (小时)", ylabel="累计呼入次数")
    axes[0, 0].legend(fontsize=7)
    window_index = np.arange(1, windows + 1)
    axes[0, 1].plot(window_index, means, label="逐窗样本均值")
    axes[0, 1].plot(window_index, variances, label="逐窗样本方差")
    axes[0, 1].axhline(lam * dt, color="gray", ls="--", label="理论值  $3$ ")
    axes[0, 1].set(title="跨路径估计每窗矩", xlabel="小时窗序号", ylabel="计数矩")
    axes[0, 1].legend(fontsize=7)

```

```
k = np.arange(12)
frequency = np.bincount(increments[:, 0], minlength=12)[:12] / repetitions
axes[1, 0].bar(k, frequency, alpha=0.5, label="首窗模拟频率")
axes[1, 0].plot(k, stats.poisson.pmf(k, lam * dt), "o-", label="泊松理论概率")
axes[1, 0].set(title="一小时窗的计数分布", xlabel="呼入次数", ylabel="概率质量")
axes[1, 0].legend(fontsize=7)
event_grid = np.r_[0.0, arrival_times, windows * dt]
event_counts = np.r_[0, np.arange(1, arrival_times.size + 1), arrival_times.size]
axes[1, 1].step(event_grid, event_counts, where="post", label="指数间隔法")
axes[1, 1].plot(times, lam * times, "k--", label="理论均值 3t")
axes[1, 1].set(title="另一条连续时间路径", xlabel="时刻 (小时)", ylabel="累计呼入次数")
axes[1, 1].legend(fontsize=7)
for ax in axes.flat:
    ax.grid(alpha=0.2)
fig.savefig("poisson_process.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e38_covariance_matrix.py

附件文件 e38_covariance_matrix.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e38_covariance_matrix.py

```
# 教材位置：第 7 章，第 7.10 节；图 7-5。
# 从四个二维观测构造样本协方差，再核对线性组合的样本方差。
# X 形状 (4,2)：行是观测，列是特征；这里两列均用无量纲教学数值。
# mean 形状 (2,)，Xc 形状 (4,2)，C 形状 (2,2)，a 形状 (2,)。
# 核心关系：C=Xc.T@Xc/(n-1)，var(X@a,ddof=1)=a.T@C@a。
# 输入固定，不随机抽样；半正定性来自任何方向方差均非负。
# 输出中心、中心化平方和、样本协方差、特征值及两条方差计算路径。
# 执行 python e38_covariance_matrix.py，图片写入当前工作目录。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

X = np.array([[1, 2], [2, 1], [3, 5], [4, 4]], dtype=float)
n = X.shape[0]
# axis=0 消去观测这一维，每列保留一个均值。
mean = X.mean(axis=0)
# 广播把两个列均值复制到四行，不对行向量逐行分别中心化。
Xc = X - mean
cross_products = Xc.T @ Xc
C = cross_products / (n - 1)
# 手算：列均值 (2.5,3)，平方和为 5、10，交叉乘积和为 5。
assert np.allclose(mean, [2.5, 3.0])
assert np.allclose(cross_products, [[5, 5], [5, 10]])
manual_cov = np.array([[5 / 3, 5 / 3], [5 / 3, 10 / 3]])
assert np.allclose(C, manual_cov)
# np.cov 默认把行视为变量；rowvar=False 才与本例“列为特征”一致。
assert np.allclose(C, np.cov(X, rowvar=False, ddof=1))

a = np.array([0.6, 0.4])
combined = X @ a
# combined=[1.4,1.6,3.8,4]；它的样本均值为 2.7。
direct_var = combined.var(ddof=1)
```

```

quadratic_var = a @ C @ a
manual_var = 0.6**2 * (5 / 3) + 2 * 0.6 * 0.4 * (5 / 3)
manual_var += 0.4**2 * (10 / 3)
assert np.allclose(combined, [1.4, 1.6, 3.8, 4.0])
assert np.isclose(direct_var, quadratic_var)
assert np.isclose(quadratic_var, manual_var)
# C 对称, 可用 eigvalsh; 浮点核验允许很小的负舍入误差。
eigenvalues = np.linalg.eigvalsh(C)
assert np.allclose(C, C.T)
assert eigenvalues.min() >= -1e-12
# 这里特征值均正, 故四个观测在二维中并非完全落在同一条直线上。
print("列均值: ", mean)
print("中心化后的矩阵: \n", Xc)
print("交叉乘积和矩阵: \n", cross_products)
print("样本协方差矩阵: \n", C)
print("协方差矩阵的特征值: ", eigenvalues)
print("线性组合观测及均值: ", combined, combined.mean())
print("直接样本方差、二次型方差、手算方差: ",
      direct_var, quadratic_var, manual_var)

# 左图保留原始观测和中心, 右图逐元素显示 C, 帮助对应三个手算和。
configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.1), constrained_layout=True)
axes[0].scatter(X[:, 0], X[:, 1], label="四个观测")
axes[0].scatter(*mean, marker="x", s=70, color="red", label="样本中心")
for index, point in enumerate(X, 1):
    axes[0].annotate(str(index), point, xytext=(4, 4), textcoords="offset points")
axes[0].quiver(*mean, *a, angles="xy", scale_units="xy", scale=1, color="gray")
axes[0].set(title="观测与线性组合方向 a", xlabel="第一特征", ylabel="第二特征")
axes[0].set_aspect("equal")
axes[0].legend(fontsize=7.5)
axes[0].grid(alpha=0.2)
axes[1].imshow(C, cmap="Blues", vmin=0, vmax=C.max() * 1.3)
for i in range(2):
    for j in range(2):
        axes[1].text(j, i, f"{C[i, j]:.4f}", ha="center", va="center")
axes[1].set(xticks=[0, 1], yticks=[0, 1],
            xticklabels=["第一特征", "第二特征"],
            yticklabels=["第一特征", "第二特征"], title="样本协方差 C")
fig.savefig("covariance_matrix.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e39_svd2d.py

附件文件 e39_svd2d.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e39_svd2d.py

```

# 教材位置: 第 9 章, 第 9.2 节; 图 9-1。
# 用同一个二维矩阵依次展示  $V^T$ 、 $\Sigma$ 、 $U$  的几何作用。
#  $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$  不是对称矩阵; 不可把特征分解直接当作 SVD。
#  $A$ 、 $U$ 、 $V_h$ 、 $\Sigma$  均为  $(2, 2)$ ,  $s$  为  $(2,)$ , circle 为  $(2, 361)$ 。
# 坐标、矩阵元素均无量纲;  $\Sigma$  是奇异值对角矩阵,  $\Sigma$  才是求和符号。
# 正交矩阵保持长度和夹角, 也可能包含反射, 不一定是旋转。
# 固定矩阵无随机抽样; 输出奇异值、正交因子行列式及重建误差。
# 执行 python e39_svd2d.py 生成四阶段中文图 svd2d.png。

```

```

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

A = np.array([[2.0, 1.0], [0.0, 1.0]])
U, s, Vh = np.linalg.svd(A, full_matrices=False)
Sigma = np.diag(s)
# 手算  $A^T A = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$ , 特征方程  $\lambda^2 - 6\lambda + 4 = 0$ 。
# 特征值是  $3 \pm \sqrt{5}$ , 取非负平方根得到降序奇异值。
manual_s = np.sqrt([3 + np.sqrt(5), 3 - np.sqrt(5)])
assert np.allclose(s, manual_s)
assert np.allclose(A.T @ A, [[4, 2], [2, 2]])
assert np.allclose(U.T @ U, np.eye(2))
assert np.allclose(Vh @ Vh.T, np.eye(2))
assert np.allclose(U @ Sigma @ Vh, A)
# NumPy 的 Vh 存放 V 的共轭转置; 本例全为实数, 所以  $Vh = V^T$ 。
# 奇异向量正负可同时翻转; 只核对重建和对应方向关系。
assert np.allclose(A @ Vh.T, U * s)
print("手算、数值奇异值: ", manual_s, s)
print("正交因子 U 与  $V^T$  的行列式: ", np.linalg.det(U), np.linalg.det(Vh))
print("矩阵重建误差: ", np.linalg.norm(U @ Sigma @ Vh - A))

# 所有圆周点作为列向量存放, 左乘矩阵同时映射全部点。
theta = np.linspace(0, 2 * np.pi, 361)
circle = np.vstack([np.cos(theta), np.sin(theta)])
after_v = Vh @ circle
after_sigma = Sigma @ after_v
after_u = U @ after_sigma
assert np.allclose(after_u, A @ circle)
assert np.allclose(np.linalg.norm(after_v, axis=0), 1)
# 追踪同一批输入基向量的像, 而非每幅图重新画标准基。
stage_matrices = [np.eye(2), Vh, Sigma @ Vh, A]
stage_points = [circle, after_v, after_sigma, after_u]
titles = ["输入单位圆", "施加  $V^T$ : 正交变换", "施加  $\Sigma$ : 伸缩", "施加 U: 正交变换"]
# 四个面板使用同样的坐标范围和等比例单位, 圆与椭圆形状可直接比较。
# 坐标基不同于奇异向量; 基向量箭头只是追踪映射的辅助标记。

configure_chinese_plot()
fig, axes = plt.subplots(2, 2, figsize=(6.3, 5.6), constrained_layout=True)
for ax, points, matrix, title in zip(
    axes.flat, stage_points, stage_matrices, titles
):
    ax.plot(*points, color="tab:blue", lw=1.5)
    # 橙点和短箭头追踪输入圆上的同一个起点及前进方向。
    ax.scatter(*points[:, 0], color="tab:orange", zorder=4, s=20)
    ax.annotate("", xy=points[:, 30], xytext=points[:, 15],
                arrowprops={"arrowstyle": "->", "color": "tab:orange"})
    for column, color in zip(matrix.T, ["tab:green", "tab:red"]):
        ax.annotate("", xy=column, xytext=(0, 0),
                    arrowprops={"arrowstyle": "->", "color": color})
    ax.axhline(0, color="gray", lw=0.5)
    ax.axvline(0, color="gray", lw=0.5)
    ax.set(xlim=(-2.7, 2.7), ylim=(-2.7, 2.7),
           xlabel="第一坐标", ylabel="第二坐标", title=title)
    ax.set_aspect("equal")
    ax.set_xticks([-2, 0, 2])
    ax.set_yticks([-2, 0, 2])

```

```
ax.grid(alpha=0.2)
# 颜色约定在源码中明确: 绿、红箭头分别为第一、第二输入基向量的像。
# 末图应与直接计算  $A@circle$  相同, 不能另换矩阵来追求视觉效果。
fig.savefig("svd2d.png", dpi=180, bbox_inches="tight")
plt.close(fig)
```

程序 e40_customer_retention.py

附件文件 e40_customer_retention.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e40_customer_retention.py

```
# 教材位置: 第 10 章, 第 10.6 节; 图 10-4。
# 三状态用户留存的教学模型; P 为人为设定, 不代表真实企业用户。
# 状态顺序: 活跃、沉默、流失; 流失为吸收态, 模型不允许流失后回流。
# P 形状 (3,3), x0 为 (3,), distribution 为 (13,3), 含当前第 0 期。
# 时间按期计, 比例无量纲; 基本矩阵给出流失前非流失期数的期望。
# 学习目标: 区分“活跃率”与“尚未流失率”, 解释有限期及长期结果。
# 本例无随机抽样; 执行 python e40_customer_retention.py 保存两面板图。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

P = np.array([[0.75, 0.20, 0.05], [0.30, 0.60, 0.10], [0.0, 0.0, 1.0]])
x0 = np.array([0.8, 0.2, 0.0])
steps = 12
assert np.all(P >= 0) and np.allclose(P.sum(axis=1), 1)
assert np.array_equal(P[2], [0, 0, 1])
distribution = np.empty((steps + 1, 3))
distribution[0] = x0
# 行概率向量右乘 P; 同一矩阵用于每一步是时间齐次假设。
for step in range(steps):
    distribution[step + 1] = distribution[step] @ P
active_rate = distribution[:, 0]
# 沉默者尚未流失, 但不计为活跃者; 二者不能用同一个业务指标代替。
retained_rate = distribution[:, :2].sum(axis=1)
# 分布总和是守恒检查; 未流失率不增来自本例没有吸收后的回流。
assert np.allclose(distribution.sum(axis=1), 1)
assert np.all(active_rate <= retained_rate)
assert np.all(np.diff(retained_rate) <= 1e-12)
assert np.allclose(distribution[-1], x0 @ np.linalg.matrix_power(P, steps))

# Q 只保留两个暂态: 活跃与沉默。其谱半径小于 1, 最终必然吸收。
Q = P[:2, :2]
spectral_radius = np.max(np.abs(np.linalg.eigvals(Q)))
assert spectral_radius < 1
#  $F=I+Q+Q^2+\dots=(I-Q)^{-1}$ ; 用求解线性方程避免显式求逆。
fundamental = np.linalg.solve(np.eye(2) - Q, np.eye(2))
expected_periods = fundamental @ np.ones(2)
initial_expected = x0[:2] @ expected_periods
#  $F_{ij}$  为从暂态 i 出发、吸收前访问暂态 j 的期望次数, 含当前期。
# 因每个未流失期后走一步, expected_periods 也等于首次流失的步数期望。
assert np.allclose(fundamental, [[10, 5], [7.5, 6.25]])
assert np.allclose(expected_periods, [15, 13.75])
assert np.isclose(initial_expected, 14.75)
```



```

long_run = x0 @ np.linalg.matrix_power(P, 1000)
assert np.allclose(long_run, [0, 0, 1], atol=1e-12)
print("第 12 期三状态比例: ", distribution[-1])
print("第 12 期活跃率、未流失率: ", active_rate[-1], retained_rate[-1])
print("暂态 Q 的谱半径: ", spectral_radius)
print("基本矩阵: \n", fundamental)
print("从活跃、沉默出发的流失前非流失期数期望 (含当前期): ", expected_periods)
print("初始混合人群的期望非流失期数: ", initial_expected)
print("本吸收模型的长期比例: ", long_run)

# 有限期图仍可能有较高留存, 不能据此宣称长期有稳定活跃用户。
# 长期全流失是吸收规则的数学后果, 不是某企业必然发生的现实预测。
configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.2), constrained_layout=True)
times = np.arange(steps + 1)
for j, name in enumerate(["活跃", "沉默", "流失"]):
    axes[0].plot(times, distribution[:, j], marker="o", ms=3, label=name)
axes[0].set(title="三状态每期分布", xlabel="观察周期数", ylabel="人群比例", ylim=(0, 1))
axes[1].plot(times, active_rate, marker="o", ms=3, label="活跃率")
axes[1].plot(times, retained_rate, marker="s", ms=3, label="未流失率")
axes[1].set(title="沉默用户仍属未流失", xlabel="观察周期数",
            ylabel="人群比例", ylim=(0, 1))
for ax in axes:
    ax.legend(fontsize=8)
    ax.grid(alpha=0.2)
fig.savefig("customer_retention.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 e41_customer_pca.py

附件文件 e41_customer_pca.py。框内仅为完整源码, 保留缩进与空行, 可整段复制。

附录代码 e41_customer_pca.py

```

# 教材位置: 第 10 章, 第 10.7 节; 图 10-5。
# 用户画像的教学模拟: 120 个用户、月订单数、月消费金额、月浏览时长。
# 数据由明确给出的潜在变量和噪声生成, 不代表真实用户或任何经营结论。
# 三列单位依次为次、元、分钟; X 形状 (120, 3), 每行一个模拟用户。
# 先按样本均值和样本标准差标准化, 再用 SVD 求描述性主成分。
# Z 形状 (120, 3), C 为 (3, 3), scores 为 (120, 3), ratio 为 (3,)。
# 固定种子 2026; 输出列尺度、解释率、主成分方向与二维重建误差。
# 执行 python e41_customer_pca.py, 图片写入当前目录。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

rng = np.random.default_rng(2026)
n = 120
latent = rng.normal(size=n)
# 潜在活跃度影响订单强度, 订单为整数; 后两列仍有各自独立噪声。
orders = rng.poisson(np.exp(2.2 + 0.35 * latent))
amount = 100 + 45 * orders + 30 * latent + rng.normal(0, 30, n)
minutes = 40 + 8 * latent + 2 * orders + rng.normal(0, 4, n)
X = np.column_stack([orders, amount, minutes])
# axis=0 跨用户计算每列统计量; 不能沿 axis=1 把每个用户的三种单位混合。

```

```

mean = X.mean(axis=0)
sd = X.std(axis=0, ddof=1)
assert np.all(sd > 0)
Z = (X - mean) / sd
assert np.allclose(Z.mean(axis=0), 0, atol=1e-12)
assert np.allclose(Z.std(axis=0, ddof=1), 1)
# C 是标准化样本的协方差, 也就是原三特征的样本相关矩阵。
C = Z.T @ Z / (n - 1)
U, s, Vh = np.linalg.svd(Z, full_matrices=False)
directions = Vh.T
scores = Z @ directions
ratio = s**2 / np.sum(s**2)
# 用独立的对称特征值算法核对, eigh 默认升序, 须同时重排特征向量。
eigenvalues, eigenvectors = np.linalg.eigh(C)
order = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[order]
eigenvectors = eigenvectors[:, order]
assert np.allclose(eigenvalues, s**2 / (n - 1))
assert np.allclose(ratio, eigenvalues / eigenvalues.sum())
# 向量可能整体异号; 外积给出同一维子空间的正交投影矩阵。
for j in range(3):
    assert np.allclose(np.outer(directions[:, j], directions[:, j]),
                        np.outer(eigenvectors[:, j], eigenvectors[:, j]))
assert np.allclose(scores, U * s)
assert np.allclose(scores.T @ scores / (n - 1), np.diag(eigenvalues))
# 保留前两成分重建标准化数据, 再还原单位; 误差先在无量纲 z 上度量。
reconstructed_z = scores[:, :2] @ directions[:, :2].T
reconstructed_x = reconstructed_z * sd + mean
relative_error = np.linalg.norm(Z - reconstructed_z) / np.linalg.norm(Z)
assert np.isclose(relative_error, np.sqrt(ratio[2]))
print("三列样本均值 (次、元、分钟): ", mean)
print("三列样本标准差 (次、元、分钟): ", sd)
print("标准化样本协方差: \n", C)
print("主成分解释率: ", ratio)
print("主成分方向 (每列一个方向): \n", directions)
print("前两成分的标准化重建相对误差: ", relative_error)
# 主成分得分不相关不等于独立; 这里不进行检验、置信区间或因果推断。
# 未标准化时金额的大数值会主导平方损失, 这是另一种分析问题。

configure_chinese_plot()
fig, axes = plt.subplots(1, 3, figsize=(6.3, 2.9), constrained_layout=True)
axes[0].scatter(orders, amount, alpha=0.6, s=12)
axes[0].set(title="明确模拟的原始画像", xlabel="月订单数 (次)",
            ylabel="月消费金额 (元)")
axes[1].scatter(scores[:, 0], scores[:, 1], alpha=0.6, s=12)
axes[1].set(title="标准化后的二维得分", xlabel="第一主成分得分",
            ylabel="第二主成分得分")
axes[2].bar([1, 2, 3], ratio)
for index, value in enumerate(ratio, 1):
    axes[2].text(index, value + 0.025, f"{value:.1%}", ha="center", fontsize=7)
axes[2].set(title="标准化方差的解释率", xlabel="主成分序号",
            ylabel="解释率", ylim=(0, 1.1))
axes[2].set_xticks([1, 2, 3])
for ax in axes:
    ax.grid(alpha=0.2)
fig.savefig("customer_pca.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```


程序 e42_norris_calibration.py

附件文件 e42_norris_calibration.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 e42_norris_calibration.py

```
# 教材位置: 第 10 章, 第 10.8 节; 图 10-6。
# NIST StRD Norris 的真实臭氧监测仪标定数据, 36 对观测原始列顺序为 y,x。
# 来源: https://www.itl.nist.gov/div898/strd/lis/data/Norris.shtml
# 数据: https://www.itl.nist.gov/div898/strd/lis/data/LINKS/DATA/Norris.dat
# 原始资料未声明数值单位; 这里不补设单位, 也不指定两列为某具体仪器。
# data 形状 (36,2), A 为 (36,2), coef 为 (2,), residual 为 (36,)。
# 设计列顺序 [1,x], 所以 coef[0] 是截距, coef[1] 是斜率。
# 输出手算/最小二乘系数、残差标准差、平方和及 x=500 处拟合值。
# 数据直接嵌入程序, 离线可运行; 执行 python e42_norris_calibration.py。

import numpy as np
import matplotlib.pyplot as plt
from plot_style import configure_chinese_plot

# 每行严格保留官方文件的 [y,x] 顺序; 不能交换两列后仍沿用认证系数。
data = np.array([
    [0.1, 0.2], [338.8, 337.4], [118.1, 118.2],
    [888.0, 884.6], [9.2, 10.1], [228.1, 226.5],
    [668.5, 666.3], [998.5, 996.3], [449.1, 448.6],
    [778.9, 777.0], [559.2, 558.2], [0.3, 0.4],
    [0.1, 0.6], [778.1, 775.5], [668.8, 666.9],
    [339.3, 338.0], [448.9, 447.5], [10.8, 11.6],
    [557.7, 556.0], [228.3, 228.1], [998.0, 995.8],
    [888.8, 887.6], [119.6, 120.2], [0.3, 0.3],
    [0.6, 0.3], [557.6, 556.8], [339.3, 339.1],
    [888.0, 887.2], [998.5, 999.0], [778.9, 779.0],
    [10.2, 11.1], [117.6, 118.3], [228.9, 229.2],
    [668.4, 669.1], [449.2, 448.9], [0.2, 0.5],
], dtype=float)
y, x = data[:, 0], data[:, 1]
n = x.size
A = np.column_stack([np.ones(n), x])
coef, sums, rank, singular = np.linalg.lstsq(A, y, rcond=None)
yhat = A @ coef
residual = y - yhat
# 用中心化求和公式作独立手算核对, 而非对 A.T@A 显式求逆。
xc, yc = x - x.mean(), y - y.mean()
manual_slope = (xc @ yc) / (xc @ xc)
manual_intercept = y.mean() - manual_slope * x.mean()
assert n == 36 and rank == 2
assert np.allclose(coef, [manual_intercept, manual_slope])
# 官方认证截距、斜率; 只用作数值核验, 不反向覆盖本次实际拟合结果。
certified_coef = np.array([-0.262323073774029, 1.00211681802045])
assert np.allclose(coef, certified_coef, rtol=1e-12, atol=1e-11)
sse = residual @ residual
residual_sd = np.sqrt(sse / (n - rank))
r_squared = 1 - sse / (yc @ yc)
# 残差标准差分母为 36-2=34; 它不同于直接除以 36 的均方根误差。
assert np.isclose(sse, 26.6173985294224, rtol=1e-12)
assert np.isclose(residual_sd, 0.884796396144373, rtol=1e-12)
assert np.isclose(r_squared, 0.999993745883712, rtol=1e-12)
```

```

assert np.isclose(sums[0], sse)
assert np.allclose(A.T @ residual, 0, atol=1e-8)
# x=500 在已观测范围内；这是拟合均值，不是下一次仪器读数的保证。
x_new = 500.0
prediction = np.array([1.0, x_new]) @ coef
print("数据来源: NIST StRD Norris, 36 对臭氧监测仪标定观测")
print("截距、斜率: ", coef)
print("手算截距、斜率: ", manual_intercept, manual_slope)
print("残差平方和、残差自由度: ", sse, n - rank)
print("残差标准差、决定系数: ", residual_sd, r_squared)
print("x=500 处的拟合均值: ", prediction)
# 高决定系数还受 x 的覆盖范围影响，不能代替残差诊断或外部验证。
# 若要置信/预测区间，还需说明噪声模型和推断条件；本例只做拟合核验。

configure_chinese_plot()
fig, axes = plt.subplots(1, 2, figsize=(6.3, 3.0), constrained_layout=True)
order = np.argsort(x)
axes[0].scatter(x, y, s=14, alpha=0.7, label="官方观测")
axes[0].plot(x[order], yhat[order], color="tab:red", label="最小二乘拟合")
axes[0].set(title="臭氧监测仪标定数据", xlabel="官方数据列 x", ylabel="官方数据列 y")
axes[0].legend(fontsize=8)
axes[1].scatter(x, residual, s=14, alpha=0.7)
axes[1].axhline(0, color="gray", ls="--")
axes[1].set(title="残差需单独检查", xlabel="官方数据列 x", ylabel="观测减拟合值")
for ax in axes:
    ax.grid(alpha=0.2)
fig.savefig("norris_calibration.png", dpi=180, bbox_inches="tight")
plt.close(fig)

```

程序 environment_check.py

附件文件 `environment_check.py`。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 environment_check.py

```

# 环境检查：确认当前解释器及三个教学依赖的实际版本。
# 运行方式：在本目录的终端执行 python environment_check.py。
# 目标环境：Python 3.13、NumPy 2.4.6、SciPy 1.18.0、Matplotlib 3.11.1。
# 本脚本只打印版本，不自动安装或升级软件，也不将版本不一致判为异常。
# 若某个依赖尚未安装，对应 import 会报 ModuleNotFoundError；
# 请先在同一个解释器环境中执行 python -m pip install -r requirements.txt。

# sys 属于 Python 标准库；sys.version 记录正在执行本文件的解释器版本。
import sys
# NumPy 负责数组与线性代数；np 是本案例包统一采用的简称。
import numpy as np
# SciPy 提供概率分布、统计推断及相关数值计算接口。
import scipy
# Matplotlib 负责绘图；这里只读版本，不创建图形窗口。
import matplotlib

# sys.version 还含编译器等信息；split()[0]只取开头的版本号。
# 应使用这里显示的版本判断当前环境，而不是另一个终端里 pip 指向的环境。
print("Python 解释器版本: ", sys.version.split()[0])
# 各库通过__version__公开当前已加载版本；这三行可用于课堂排查环境差异。
print("NumPy 版本: ", np.__version__)

```

```
print("SciPy 版本: ", scipy.__version__)
print("Matplotlib 版本: ", matplotlib.__version__)
```

程序 g00_knowledge_map.py

附件文件 g00_knowledge_map.py。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 g00_knowledge_map.py

```
# 学习路径图：对应教材前言的图 0-1，连接概率与线性代数两条学习路径。
# 本图没有模拟数据或待估计参数；节点坐标只控制版面，不表示概率或数量。
# 运行：python g00_knowledge_map.py；PNG 写入当前工作目录。
# 图内文字采用中文；字体由共用样式模块按当前系统选择。
import matplotlib.pyplot as plt
# 在创建第一幅图之前配置可跨系统选择的中文字体。
from plot_style import configure_chinese_plot

configure_chinese_plot()
from matplotlib.patches import FancyBboxPatch

# 左右列节点依次对应两条课程知识链；每项为中心横坐标、纵坐标和名称。
nodes = {
    "random": (0.23, 0.86, "随机变量"),
    "distributions": (0.23, 0.65, "概率分布与数学期望"),
    "inference": (0.23, 0.44, "抽样与统计推断"),
    "simulation": (0.23, 0.23, "模拟与不确定性"),
    "vectors": (0.77, 0.86, "向量与线性映射"),
    "equations": (0.77, 0.65, "线性方程组与投影"),
    "decomposition": (0.77, 0.44, "矩阵分解"),
    "applications": (0.77, 0.23, "马尔可夫链、主成分与最小二乘"),
}

# 箭头表示学习先后或应用联系；不表示随机变量之间的因果关系。
edges = [
    ("random", "distributions"), ("distributions", "inference"),
    ("inference", "simulation"), ("vectors", "equations"),
    ("equations", "decomposition"), ("decomposition", "applications"),
    ("simulation", "applications"),
]

# 相对坐标范围固定为 0 到 1；关闭数值坐标轴，突出概念间的联系。
fig, ax = plt.subplots(figsize=(6.3, 4.7), constrained_layout=True)
ax.set(xlim=(0, 1), ylim=(0.08, 1.0))
ax.axis("off")
# 先画箭头，再画节点，避免连线遮住框中文字。
for start, end in edges:
    x1, y1, _ = nodes[start]
    x2, y2, _ = nodes[end]
    # 同列连接上下边缘，跨列连接左右边缘；这些偏移只与框尺寸有关。
    if x1 == x2:
        beginning, ending = (x1, y1 - 0.055), (x2, y2 + 0.055)
    else:
        beginning, ending = (x1 + 0.20, y1), (x2 - 0.20, y2)
    ax.annotate("", xy=ending, xytext=beginning,
               arrowprops={"arrowstyle": "->", "color": "#42677a", "lw": 1.4})
for x, y, label in nodes.values():
    # 圆角框是概念节点；颜色不编码任何统计量。
    box = FancyBboxPatch((x - 0.20, y - 0.055), 0.40, 0.11,
```

```

        boxstyle="round,pad=0.008", facecolor="#eaf0f3",
        edgecolor="#42677a", linewidth=1.0)

    ax.add_patch(box)
    ax.text(x, y, label, ha="center", va="center", fontsize=8.5)
ax.text(0.23, 0.975, "概率与统计", ha="center", fontsize=11, weight="normal")
ax.text(0.77, 0.975, "线性代数", ha="center", fontsize=11, weight="normal")
# 教材直接嵌入本 PNG; bbox_inches 给标签保留完整边界。
fig.savefig("g00_knowledge_map.png", dpi=180, bbox_inches="tight", pad_inches=0.08)
plt.close(fig)

```

程序 plot_style.py

附件文件 `plot_style.py`。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 plot_style.py

```

"""共用绘图样式：使用当前系统已有的中文字体，不随案例包分发字体。"""

from pathlib import Path

import matplotlib as mpl
from matplotlib import font_manager
from matplotlib.ft2font import FT2Font

# Windows、macOS 和 Linux 的常见字体均列入候选。
# 顺序只影响显示外观，不参与任何随机抽样或数值计算。
CHINESE_FONT_CANDIDATES = (
    "Microsoft YaHei",
    "SimHei",
    "Noto Sans CJK SC",
    "Source Han Sans SC",
    "PingFang SC",
    "Heiti SC",
    "Songti SC",
    "WenQuanYi Zen Hei",
    "Arial Unicode MS",
)

def configure_chinese_plot():
    """在首次创建图形前选择中文字体，并返回实际使用的字体信息。

    变量 available 把字体名称映射到本机文件；选用第一个可用候选。
    返回的字典只供运行报告使用，不改变案例的数据或算法。
    """
    available = {font.name: font.fname for font in font_manager.fontManager.ttflist}
    selected = next(
        (name for name in CHINESE_FONT_CANDIDATES if name in available), None
    )
    if selected is None:
        raise RuntimeError(
            "未找到中文字体。请安装思源黑体或 Noto Sans CJK SC, "
            "然后重新运行案例。案例包不附带系统字体。"
        )
    mpl.rcParams.update({
        "font.family": "sans-serif",

```

```

        "font.sans-serif": [selected, "DejaVu Sans"],
        "font.size": 9,
        "axes.titlesize": 10,
        "axes.labelsize": 9,
        "xtick.labelsize": 8,
        "ytick.labelsize": 8,
        "legend.fontsize": 8,
        "axes.unicode_minus": False,
        "savefig.facecolor": "white",
    })
    return {"name": selected, "path": str(Path(available[selected]).resolve())}

def check_font_characters(text):
    """核对所选字体覆盖的中文字符；返回缺字列表，空列表表示全部覆盖。"""
    selected = configure_chinese_plot()
    covered = FT2Font(selected["path"]).get_charmap()
    characters = {char for char in text if "\u3400" <= char <= "\u9fff"}
    return sorted(char for char in characters if ord(char) not in covered)

```

程序 run_all.py

附件文件 `run_all.py`。框内仅为完整源码，保留缩进与空行，可整段复制。

附录代码 run_all.py

```

"""发现并逐例运行配套程序，把一套图片和逐例日志写入本案例包。

运行: python run_all.py。
本脚本依次执行环境检查、学习路径图和所有 e*.py 案例。
每个子程序使用独立进程和工作目录，彼此不共享变量或随机数状态。
图片最终移至本目录；results 只保存标准输出、错误输出和运行汇总。
"""

from datetime import datetime, timezone
from pathlib import Path
import hashlib
import json
import os
import shutil
import subprocess
import sys
import time

def sha256(path):
    """按文件实际字节计算摘要，用于核对代码和图片的具体版本。"""
    return hashlib.sha256(path.read_bytes()).hexdigest()

def main():
    # __file__ 锚定案例包位置，允许从任意目录启动本工具。
    root = Path(__file__).resolve().parent
    results = root / "results"
    results.mkdir(exist_ok=True)
    env = os.environ.copy()

```

```

env["MPLBACKEND"] = "Agg"
env["MPLCONFIGDIR"] = str(results / "matplotliblib_cache")

# 文件名决定发现范围; run_all.py 和 plot_style.py 均不在此范围。
# g00 学习路径图与 e 开头案例保持排序, 新增案例无需修改此列表。
discovered = sorted(root.glob("e*.py"))
cases = [path for path in discovered if path.name != "environment_check.py"]
maps = sorted(root.glob("g00*.py"))
environment = root / "environment_check.py"
scripts = ([environment] if environment.is_file() else []) + maps + cases
records = {}
failed = []
for script in scripts:
    destination = results / script.stem
    destination.mkdir(exist_ok=True)
    # 仅清理本工具生成的单例临时图, 不触碰源目录或其他案例。
    for old_png in destination.glob("*.png"):
        old_png.unlink()
    started = datetime.now(timezone.utc).isoformat()
    clock_start = time.perf_counter()
    try:
        process = subprocess.run(
            [sys.executable, str(script)],
            cwd=destination,
            env=env,
            capture_output=True,
            text=True,
            timeout=180,
            check=False,
        )
        code, stdout, stderr = (
            process.returncode, process.stdout, process.stderr
        )
    except subprocess.TimeoutExpired as exc:
        code = 124
        stdout = exc.stdout or ""
        stderr = exc.stderr or ""
        if isinstance(stdout, bytes):
            stdout = stdout.decode("utf-8", errors="replace")
        if isinstance(stderr, bytes):
            stderr = stderr.decode("utf-8", errors="replace")
        stderr += "\n 程序运行超过 180 秒。"
    elapsed = time.perf_counter() - clock_start
    (destination / "stdout.txt").write_text(stdout, encoding="utf-8")
    (destination / "stderr.txt").write_text(stderr, encoding="utf-8")
    figures = {}
    for png in sorted(destination.glob("*.png")):
        # 移动而非复制, 使案例包只保留根目录这一套可嵌入教材的 PNG。
        final_png = root / png.name
        shutil.move(str(png), str(final_png))
        figures[png.name] = sha256(final_png)
    records[script.name] = {
        "returncode": code,
        "source_sha256": sha256(script),
        "started_at_utc": started,
        "elapsed_seconds": round(elapsed, 4),
        "stdout": stdout,
        "stderr": stderr,
    }

```

```

        "figures": figures,
    }
    if code:
        failed.append(script.name)
    print(script.name, "通过" if code == 0 else "失败", flush=True)
    if code:
        print(stderr, flush=True)

report = results / "运行汇总.json"
report.write_text(
    json.dumps(records, ensure_ascii=False, indent=2), encoding="utf-8"
)
passed = sum(records[path.name]["returncode"] == 0 for path in cases)
print(f"教学案例通过 {passed}/{len(cases)}; 失败程序 {len(failed)} 个。")
return int(bool(failed))

if __name__ == "__main__":
    raise SystemExit(main())

```

附录三 公式索引与参考资料

正文的显示公式按章编号。公式只在所说明的条件下成立，不能脱离相邻文字单独套用。
下面列出关键公式的编号与主题，便于跨章查找。

表 0-5 关键公式查找表

主题	公式编号
条件分解与全方差	式 (2.11)
中心极限定理	式 (2.15)
正态均值的共轭更新	式 (5.10)
秩与零度	式 (6.7)
最小二乘正规方程	式 (7.5)
样本协方差矩阵	式 (7.8)
一般奇异值分解	式 (9.2)
伪逆的构造	式 (9.9)
用户状态转移矩阵	式 (10.11)
用户状态与未流失率	式 (10.12)
PCA 得分与解释率	式 (10.17)
真实标定的最小二乘目标	式 (10.18)
最小二乘估计量的协方差	式 (10.20)

以下列出实际查核的课程、教材、接口和数据资料。各章“继续阅读”给出具体主题与阅读重点；这里补充可以打开的原始链接。中文推导与练习按教学顺序重新组织，不把课程举例替代本书实际运行结果。核对日期为 2026 年 10 月 19 日。

[APP-S1] SciPy 官方文档 对数正态分布

用于核对形状与尺度参数、中位数约定及密度。

[打开原始资料](#)

[APP-S2] SciPy 官方文档 拟合优度的参数模拟检验

用于核对估参后必须在每份参考样本中重新拟合及模拟 p 值公式。

[打开原始资料](#)

[APP-S3] SciPy 官方文档 负二项分布

用于核对均值、方差和参数化；本书的 r 对应接口中的 n 。

[打开原始资料](#)

[APP-S4] NumPy 官方文档 多项分布与狄利克雷随机数

用于核对正参数与归一化概率向量的生成。

[打开原始资料](#)

[ENV-PYWIN] Python 3.13 官方文档 Windows 安装与启动器

第 4 章 4.1 完整安装器与 4.8 启动器；用于按 3.13 解释器创建课程环境，不依赖首页默认主版本。

[打开原始资料](#)

[ENV-PYMAC] Python 3.13 官方文档 在 macOS 上使用 Python

第 5 章系统安装与命令入口；选择发布页实际提供的安装方式。

[打开原始资料](#)

[ENV-VENV] Python 3.13 官方文档 venv

阅读“Creating virtual environments”和“How venvs work”，核对通过环境内解释器直接运行而不必激活的方式。

[打开原始资料](#)

[ENV-PIP] Python 3.13 官方文档 安装 Python 模块

阅读基本用法，区分解释器与包安装位置，使用 `python -m pip` 形式。

[打开原始资料](#)

[ENV-VSC] VS Code 官方 Python 入门教程

选择解释器、打开程序与在终端运行文件；配合官方 Python 环境页面查找 `.venv`。

[打开原始资料](#)

[API-RNG] NumPy 2.4 官方文档 Generator

阅读 default_rng 与 Generator 对象；固定种子不意味着新旧随机接口生成相同数列。

[打开原始资料](#)

[API-QQ] SciPy 1.18 官方文档 probplot

核对横轴理论分位数与纵轴排序样本、参考线拟合约定；读图不以无条件 45 度线或单一弯曲方向判断一切。

[打开原始资料](#)

[DATA-NORRIS] NIST StRD Norris 真实观测数据与标定任务

数据说明列出 36 个观测、一个预测变量、一个响应和两参数线性模型，来源为臭氧监测仪标定。

[打开原始资料](#)

[DATA-NORRIS-RAW] NIST Norris 原始 ASCII 数据与认证统计量

原文件数据列顺序 y、x；认证截距、斜率、残差标准差用于 e42 数值校验。文件未声明数值单位，本书不补设。

[打开原始资料](#)

[P01] 哈佛大学统计 110 《概率论》与《概率论导论》第二版官方入口

课程教师为约瑟夫·布利茨斯坦，教材由他与杰西卡·黄合著。教材英文检索名为 Introduction to Probability，第二版，2019 年，作者提供免费在线版本；官方教材入口 <https://probabilitybook.net/> 由课程首页链接。正文按第 1—10 章的主题关系组织为中文教学叙述，并保留本案例数据与程序，不是逐段翻译。核验日期：2026 年 10 月 19 日。

[打开原始资料](#)

[P02] 麻省理工学院 18.05 《概率与统计导论》2022 年春季课程阅读材料总目

课程公开材料按概率、贝叶斯推断、频率学检验、区间估计与回归组织，提供阅读、课堂题与解答。用于核对本篇从概率模型到统计推断的知识顺序；课程示例软件为 R，本书案例仍使用给定 Python 程序。核验日期：2026 年 10 月 19 日。

[打开原始资料](#)

[P03] 麻省理工学院 18.05 第 3 课：条件概率、独立性与贝叶斯定理

官方阅读，涉及条件概率定义、乘法与全概率、独立性、贝叶斯公式及基率问题。对应第 1 章条件与检测解释；产品数据沿用案例 e06。

[打开原始资料](#)

[P04] 麻省理工学院 18.05 第 4 课：离散随机变量的期望

官方阅读资源，期望是按概率加权、期望线性性等概念的继续阅读入口。对应第 2 章期望、指示变量与第 3 章离散模型。

[打开原始资料](#)

[P05] 麻省理工学院 18.05 第 7 课甲：联合分布与独立性

官方阅读，讨论联合、边缘、条件分布及变量独立。对应第 2 章的联合表、边缘求和与条件归一化，四格表为本书另设手算例。

[打开原始资料](#)

[P06] 哈佛大学统计 110 第 10 组专题练习：条件期望与条件方差

课程官方练习及解答，供全期望与全方差继续练习。正文两模式泊松混合例为自设数字，用于解释组内与组间方差。

[打开原始资料](#)

[P07] 麻省理工学院 18.05 第 7 课乙：协方差与相关系数

官方阅读，讨论协方差、相关、和的方差及独立性的关系。对应第 2 章标准误、协方差项与零相关反例。

[打开原始资料](#)

[P08] 麻省理工学院 18.05 第 6 课乙：中心极限定理与大数定律

官方阅读，将均值稳定性与标准化均值的近似分布分别讨论。对应第 2 章大数定律、第 2 章中心极限定理；骰子与指数模拟参数保持原 e03、e04 设定。

[打开原始资料](#)

[P09] 麻省理工学院 18.05 第 10 课：极大似然估计

官方阅读资源，供概率、似然与参数估计的区别以及二项极大似然推导继续阅读。总体、样本、参数与统计量另见同课甲阅读，通过 P02 总目进入。

[打开原始资料](#)

[P10] 麻省理工学院 18.05 第 22 课：基于正态数据的置信区间

官方阅读，讨论正态样本的均值区间与 t 分布等内容。对应第 5 章频率学覆盖率、未知方差的 t 区间和区间解释；九个观测保持原 e18 数据。

[打开原始资料](#)

[P11] 麻省理工学院 18.05 第 19 课：原假设显著性检验专题

官方阅读，与第 17—18 课共同构成检验单元，涵盖统计量、p 值与参考分布。对应第 5 章双侧检验与 t 推断；具体值由原 e18 程序核对。

[打开原始资料](#)

[P12] 麻省理工学院 18.05 第 15 课：贝塔与正态共轭先验

已核对官方第 15 课阅读。阅读重点是贝塔—二项核相乘及已知观测方差时正态先验的更新；本书为多观测均值推导配方与精度权重，再由全期望、全方差得到后验预测。e17 沿用先验(2,2)、20 次成功 15 次；e36 的四个数值为另设教学样本。

[打开原始资料](#)

[P13] SciPy 官方统计文档：拟合优度的蒙特卡洛校准

官方统计工具说明拟合未知参数后的蒙特卡洛拟合优度检验，强调每份参考样本重新拟合与加一计数。用于核验 e19，不能把对数数据正态家族的校准 p 值套到原尺度其他候选家族。

[打开原始资料](#)

[P14] 哈佛大学统计 110 官方视频目录：泊松过程、条件期望与极限定理

已核对官方讲次：第 24 讲为伽马分布与泊松过程；第 27 讲为条件期望、全期望和全方差；第 29 讲为大数定律与中心极限定理。阅读重点分别是计数—等待的同事事件表达、条件中心化的交叉项及两种极限定理的尺度。它们是本书中文论述的主题依据，教学参数与练习另设。

[打开原始资料](#)

[P15] 麻省理工学院 18.175 第 1 讲：概率空间与 σ 代数（拓展）

2014 年春季斯科特·谢菲尔德课程官方幻灯。用于核验 σ 代数对补集、可数并的封闭，概率测度的可数加性，以及波莱尔事件族。该课程为研究生层次，只作为第 1 章严格定义的拓展依据，不是本教材入门前置要求。

[打开原始资料](#)

[P16] 麻省理工学院 18.175 第 3 讲：随机变量与分布（拓展）

第 3 讲“随机变量与分布”与第 1 讲配读，用于可测随机变量的原像定义。读者只需理解数轴上的波莱尔事件须拉回合法原事件，无须在本书掌握测度延拓或勒贝格积分构造。

[打开原始资料](#)

[P17] 麻省理工学院 6.041SC 第 14—15 讲：泊松过程

官方视频目录列出第 14 讲泊松过程第一部分和第 15 讲第二部分。用于补充独立增量、平稳增量、指数间隔与累计计数的过程视角；本书 e37 采用每小时 3 次、24 小时、2000 条独立路径的教学参数。

[打开原始资料](#)

[P18] 麻省理工学院 18.05 第 20 课：频率学与贝叶斯推断的比较

官方阅读已核验。重点是固定未知参数与参数先验的角色、两种区间和检验所回答问题的差异。配合第 17 课甲“频率学派”、第 15 课共轭更新阅读；本书不把两种视角写成谁总是更正确的竞赛。

[打开原始资料](#)

[P19] 《概率论导论》作者提供的第二版免费教材入口

哈佛统计 110 官网 Book 链接指向本页；核验时转至作者提供的教材 PDF。作者为约瑟夫·布利茨斯坦与杰西卡·黄。按全书章节检索：第 1—2 章计数与条件概率，第 3—4 章随机变量与期望，第 5 章连续随机变量，第 7 章联合分布，第 8 章变换，第 9 章条件期望，第 10 章不等式与极限定理。测度论严格定义另用 P15、P16，不暗示该入门书全面讲授测度论。

[打开原始资料](#)

[P20] SciPy 官方统计文档：正态概率图与理论分位数

已核验 1.18.0 文档。probplot 默认采用正态参考分布，返回理论顺序统计量中位数与有序样本，可拟合最小二乘直线；官方明确区分概率图与一般 Q—Q 或 P—P 图。本书据此说明 e19 右图的具体坐标与函数约定，不凭任意弯曲唯一断言偏态。

[打开原始资料](#)

[MIT-L1] 麻省理工学院 18.06SC：课程总览与线性方程及四子空间

麻省理工学院 18.06SC 第一单元“ $Ax=b$ 与四子空间”总览。阅读重点：线性方程、核与像如何组织成学习主线。

[打开原始资料](#)

[MIT-L2] 麻省理工学院 18.06SC：矩阵消元

18.06SC“矩阵消元”，课包摘要 Ses1.2sum。阅读重点：消元矩阵、可逆行操作、主元与回代。

[打开原始资料](#)

[MIT-L3] 麻省理工学院 18.06SC: 主元变量、特殊解、基与维数

18.06SC“线性无关、基与维数”，摘要 Ses1.9sum。课程指定《线性代数导论》第五版 3.4 节；重点核对张成、无关与维数。

[打开原始资料](#)

[MIT-L4] 麻省理工学院 18.06SC: 四个基本子空间

18.06SC“四个基本子空间”，摘要 Ses1.10sum。课程指定第五版 3.5 节；重点核对母空间、行列秩和零度。

[打开原始资料](#)

[MIT-L5] 麻省理工学院 18.06SC: 子空间上的投影

18.06SC“子空间上的投影”，摘要 Ses2.2sum。阅读重点：残差与子空间正交、投影的几何作用。

[打开原始资料](#)

[MIT-L6] 麻省理工学院 18.06SC: 投影矩阵与最小二乘

18.06SC 第 16 讲“投影矩阵与最小二乘”，摘要 Ses2.3sum；课程指定第四/五版 4.3 节。重点把正规方程与线性回归对应。

[打开原始资料](#)

[MIT-L7] 麻省理工学院 18.06SC: 正交矩阵与格拉姆—施密特

18.06SC“正交矩阵与格拉姆—施密特”，摘要 Ses2.4sum。阅读重点：正交化、单位化与 QR 因子。

[打开原始资料](#)

[MIT-L8] 麻省理工学院 18.06SC: 对角化与矩阵幂

18.06SC 第 22 讲“对角化与矩阵幂”，摘要 Ses2.9sum。阅读重点：独立特征向量条件、换基与幂的计算。

[打开原始资料](#)

[MIT-L9] 麻省理工学院 18.06SC: 马尔可夫矩阵与傅里叶级数

18.06SC 第 24 讲“马尔可夫矩阵与傅里叶级数”，摘要 Ses2.11sum。重点读马尔可夫矩阵部分，并整体核对行随机与列随机约定。

[打开原始资料](#)

[MIT-L10] 麻省理工学院 18.06SC: 奇异值分解

18.06SC 第 29 讲“奇异值分解”，摘要 Ses3.5sum。课程指定第五版 7.1—7.2 节；重点读 $A^T A$ 、奇异方向与四子空间。

[打开原始资料](#)

[MIT-L11] 麻省理工学院 18.06SC: 换基与图像压缩

18.06SC 第 31 讲“换基与图像压缩”，摘要 Ses3.7sum。阅读重点：换基、秩 1 层与图像压缩的联系。

[打开原始资料](#)

[MIT-L12] 麻省理工学院 18.06SC: 左逆、右逆与伪逆

18.06SC 第 33 讲“左逆、右逆与伪逆”，摘要 Ses3.8sum。阅读重点：左右逆的秩条件与伪逆的几何角色。

[打开原始资料](#)

[MIT-L13] 麻省理工学院 18.06SC: 实对称矩阵与正定性

第 25 讲；课程指定《线性代数导论》第四/五版 6.4—6.5 节。重点读对称矩阵的正交特征基、二次型与正定判据。

[打开原始资料](#)

[MIT-L14] 麻省理工学院 18.06SC: 正交向量与子空间

第二单元“正交向量与子空间”；课程指定第四/五版 4.1 节。重点核对四子空间的两组正交补。

[打开原始资料](#)

[LA-B1] 斯特朗《线性代数导论》第六版：官方资源与样章

吉尔伯特·斯特朗，2023 年第六版，ISBN 978-17331466-7-8。官方公开样章 1.4 矩阵乘法、2.2 消元与逆、3.5 四子空间维数、6.1 特征值；版次不同，勿混用课程第五版小节编号。

[打开原始资料](#)

[LA-B2] 斯特朗《线性代数导论》第六版样章：四子空间的维数

第 3.5 节官方公开样章。阅读重点：秩、四空间维数与可解性；可与 18.06SC 第一单元相互核对。

[打开原始资料](#)

[LA-B3] 斯特朗《线性代数导论》第五版公开补充：范数与条件数

第六版官网列出的第五版主题“Norms and Condition Numbers”。阅读重点：矩阵范数、相对扰动与最坏情形误差界。

[打开原始资料](#)

麻省理工学院离线课程包定位

以下是用户提供的 18.06sc-fall-2011.zip 内部真实相对路径，用于按主题找到讲义。视频可能通过课程外链提供；本书配套源码不需要运行此课程包中的程序。

课程大纲：三单元组织与独立学习说明 <pages/syllabus/index.html>

麻省理工学院 18.06SC：课程总览与线性方程及四子空间 <pages/ax-b-and-the-four-subspaces/index.html>

麻省理工学院 18.06SC：矩阵消元

static_resources/0903b4b404284cd14b66ecccea103fd4_MIT18_06SCF11_Ses1.2sum.pdf

麻省理工学院 18.06SC：主元变量、特殊解、基与维数

static_resources/0bbc30e3f1d7933ea07a2d2e9ab050d9_MIT18_06SCF11_Ses1.9sum.pdf

齐次方程：主元变量与特殊解

static_resources/dddb31dfe72d2e2efd09e74713b7775_MIT18_06SCF11_Ses1.7sum.pdf

麻省理工学院 18.06SC：四个基本子空间

static_resources/62a9db9eeab190694d40afe4734068ca_MIT18_06SCF11_Ses1.10sum.pdf

麻省理工学院 18.06SC：子空间上的投影

static_resources/00e9c8f0eafedeab21a3d079a17ed3d8_MIT18_06SCF11_Ses2.2sum.pdf

麻省理工学院 18.06SC：投影矩阵与最小二乘

static_resources/198bde3a6536b62989b6cc09495575c1_MIT18_06SCF11_Ses2.3sum.pdf

麻省理工学院 18.06SC：正交矩阵与格拉姆—施密特

static_resources/7ac32be444c25e48590f47573833ccc6_MIT18_06SCF11_Ses2.4sum.pdf

麻省理工学院 18.06SC：对角化与矩阵幂

static_resources/d05795355f5cbe35356a4a1b39658336_MIT18_06SCF11_Ses2.9sum.pdf

麻省理工学院 18.06SC：马尔可夫矩阵与傅里叶级数

static_resources/c8a22883b5e0cc402b3df9c1d754ef5b_MIT18_06SCF11_Ses2.11sum.pdf

麻省理工学院 18.06SC：奇异值分解

static_resources/d273f75ee2552a5c3c35ccab37e5edce_MIT18_06SCF11_Ses3.5sum.pdf

麻省理工学院 18.06SC：换基与图像压缩

static_resources/7ce6815ffe9ec37129e581485494acde_MIT18_06SCF11_Ses3.7sum.pdf

麻省理工学院 18.06SC：左逆、右逆与伪逆

static_resources/0550c89b69c99e97dcbf52074e293308_MIT18_06SCF11_Ses3.8sum.pdf