数据科学与大数据技术

Python 贝叶斯建模与计算

[阿根廷] 奥斯瓦尔多 • A. 马丁(Osvaldo A. Martin)
[美] 拉万 • 库马尔(Ravin Kumar) 著
劳俊鹏(Junpeng Lao)

郭 涛 译

清華大學出版社
北 京

北京市版权局著作权合同登记号　图字：01-2022-5478
Bayesian Modeling and Computation in Python/by Osvaldo A. Martin, Ravin Kumar and Junpeng Lao/
ISBN: 9780367894368

图书在版编目(CIP)数据

Python贝叶斯建模与计算 / (阿根廷) 奥斯瓦尔多 • A. 马丁 (Osvaldo A. Martin)，(美) 拉万 • 库马尔 (Ravin Kumar)，(美) 劳俊鹏著；郭涛译. —北京：清华大学出版社，2024.3
(数据科学与大数据技术)
书名原文：Bayesian Modeling and Computation in Python
ISBN 978-7-302-65485-8

I. ①P…　II. ①奥…②拉…③劳…④郭…　III. ①软件工具—程序设计　IV. ①TP311.561

中国国家版本馆CIP数据核字(2024)第036434号

责任编辑：王　军
装帧设计：孔祥峰
责任校对：成风进
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京联兴盛业印刷股份有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：21.75　　字　　数：555 千字
版　　次：2024 年 3 月第 1 版　　印　　次：2024 年 3 月第 1 次印刷
定　　价：98.00 元

产品编号：097315-01

译 者 序

贝叶斯是概率图模型(Probabilistic Graphical Model，PGM)技术框架体系的核心研究内容，旨在构建多个随机变量的联合概率分布，为刻画数据和模型中的不确定性提供一种严谨、系统和科学的方法。PGM 是一种将概率论与图论有机结合的机器学习方法，是概率机器学习和人工智能可解释性领域的集大成者。PGM 通过将概率论与图论结合，建立了 PGM 框架。可以从不同的视角描述分布的两类图表示，一类用有向图表示(即边有起点和终点)，称为贝叶斯网络；另一类用无向图表示，称为马尔可夫网络，可将其看作定义独立分布的一组断言，也可以将其看作一组紧凑的因子分解。PGM 有 3 个组成部分：表示、推理和学习。这 3 个部分都是构建智能系统的关键。表示是不确定性的有效表示，主要涉及贝叶斯表示、无向 PGM、局部 PGM 等；推理是概率分布的推理，主要涉及精确推理(变量消除、团树)、近似推理(基于粒子)、最大后验概率推理等；学习是根据给定数据集估计合适的模型，主要涉及参数估计、贝叶斯网络结构学习、部分可观测数据等。此外，近年来 PGM 在行为和决策方面也取得了进展，主要是在因果关系、效用和决策以及结构化决策问题等方面。

贝叶斯属数理统计和概率机器学习的研究范畴，历史悠久。得名贝叶斯，是为了纪念托马斯·贝叶斯(Thomas Bayes)在贝叶斯理论和贝叶斯概率方面做出的重要贡献及对后世的影响。1763 年 12 月 23 日，理查德·普莱斯(Richard Price)在伦敦皇家学会会议上宣读了贝叶斯的遗世之作《机遇理论中一个问题的解》(“An essay towards solving a problem in the doctrine of chances”)。这篇论文提出了一种归纳推理的理论，从此贝叶斯定理诞生于世。虽然拉普拉斯(Pierre-Simon Laplace)等基于贝叶斯推导出一些有价值的成果，但贝叶斯的理论还不够完善，在应用中出现了一些问题，致使贝叶斯方法未被接受。在 20 世纪，一批数学家和统计学家不断完善贝叶斯观点、方法和理论，使其能够应用于工业、物理学和经济学领域。后期也出现了研究论文和著作，通过众多学者的不断完善，贝叶斯的理论最终发展为一种系统的统计推理方法——贝叶斯方法。

统计学在发展过程中出现了两个主要的学派：频率学派和贝叶斯学派。两派之间一直存在着争论，出现了两学派间的争鸣。虽然说两个学派在哲理和思想上存在对立的一面，但从长期发展来看，两个学派之间相互补充和促进，都是统计学花园中的鲜花，缺一不可。要解释清楚它们之间的争议，需要从统计推断使用的三种信息说起：**总体信息**、**样本信息**和**先验信息**。**总体信息**即总体分布或总体所属分布族给出的信息，如常见的正态分布。**样本信息**是从总体抽取的样本所提供的信息。基于以这两种信息进行的统计推断称为**经典统计学**，它的基本观点是把数据(样本)看作来自具有一定概率分布的总体，所研究的对象是这个总体，而不局限于数据本身。**先验信息**即在抽样之前有关统计问题的一些信息，一般来说，先验信息主要来源于经验和历史资料。基于上述三种信息(总体信息、样本信息和先验信息)进行的统计推断称为**贝叶斯统**

计。它与经典统计学的差别在于是否利用先验信息。其与经典统计学在使用样本上也存在差异。贝叶斯学派重视已出现的样本观察值，而对尚未发生的样本观察值不予考虑，贝叶斯学派很重视先验信息的收集、挖掘和加工，以达数量化，形成先验分布，参与到统计推断中，以提升统计推断的质量。但先验信息的表示和量化是一个难题，先验的质量参差会导致推断出现不合理的结论。

贝叶斯学派的基本观点是：任一未知量 θ 都可被看作一个随机变量，应该用概率分布描述 θ 的未知状态。此概率分布是关于 θ 先验信息的概率陈述(这个概率分布称为先验分布或先验)，在抽样前就已存在。贝叶斯学派受到了以下两点批评：一是将参数 θ 看作随机变量是否合适；二是先验是否存在，如何确定和量化先验信息。贝叶斯学派主要集中在小样本问题和区间估计的解释，在似然原理的认识等问题上对古典统计学派进行了批评。由于这两个学派的研究方法不同，因而产生的基础理论和方法也不同。在估计理论方法方面，贝叶斯学派估计的总体分布为后验分布，而古典统计学派认为，极大似然估计是贝叶斯最大后验估计的特殊情况。由此可知，古典统计学派的统计推断是从无到有的过程，常常在对总体分布一无所知，或是已知含有未知参数的总体分布族的情况下讨论的，它的推断主要依赖于所使用的统计量的针对性和实际问题与总体样本的近似性，这在大样本情况下可以产生很好的结果。贝叶斯理论则相反，它是从有到有的过程，遵循由浅入深、由表及里的认识思维，这也完全符合人的认知，也是人类认识世界所遵循的普遍规律。为了充分利用贝叶斯推断，必须考虑所有的先验都是主观的，要么来源于信念，要么来源于自己专业领域内的专家知识经验。需要应用贝叶斯定理修订对参数的信念，由此产生后验分布(关于这个主题，可阅读本书译者的另一本译作《概率图模型及计算机视觉应用》)。两个学派正是在相互批评中相互借鉴，不断完善自己的建设思想和理论体系。基于以上诸多不同和假设，统计学家、数学家和计算机科学家从不同角度、不同层面开展了一系列研究，并取得了丰富的成果。

传统的贝叶斯方法往往局限于条件概率分布，属于参数较少的浅层表征模型，很难拟合高维数据，这也是统计学理论的瓶颈。深度学习方法能够有效针对高维数据降维，提供有效特征，因而被广泛应用于人工智能诸多领域。深度学习的可解释性仍然是这一领域的世界难题(具体可阅读本书译者翻译并出版的《AI 可解释性(Python 语言版)》)，深度学习可被看作深度可表征的学习，但依然很难利用先验知识提取可解释性的表征。因此，如何有效且高效地提取海量多源异构数据的高维特征，并对数据提取可解释性特征成为人工智能领域的一个主要热点方向。贝叶斯深度学习(Bayesian Deep Learning，BDL)为解决以上问题提供了契机，将深度学习概率化作为“感知模块”，将 PGM 作为“任务模块”，统一在同一个原则性概率框架内进行学习和推断。用深度学习对文本、图像的感知能力来提高进一步推断的性能，反过来，通过推断过程的反馈增强文本或图像的感知能力。Zhang Hao 等在 *A Survey on Bayesian Deep Learning* 中提出了原则性的概率框架。**主要优势**体现在：①感知任务和推断任务之间的信息交换；②对高维数据的条件依赖；③对不确定性的有效建模。具体需要解决的**不确定性问题包括**：①神经网络参数的不确定性；②任务相关参数的不确定性；③感知部分和任务部分信息传递的不确定性。这也带来了**挑战**：①设计一个具有合理时间复杂度的有效贝叶斯神经网络模型；②确保感知组件和任务组件之间的高效信息交换。**解决思路**：感知组件应是贝叶斯(或概率)神经网络，以便与任务组件(任务组件天生是概率的)兼容，确保感知组件具备处理参数及输出不确定性的能力。感知组

件可采用受限玻尔兹曼机(RBM)、概率广义堆叠去噪自动编码器(pSDAE)、变分自编码器(VAE)、概率反向传播(PBP)等。任务组件的目的是将概率先验知识合并到贝叶斯深度学习中。概率先验知识可以用 PGM 自然地表示。具体地说，它可以是典型的(或浅层)贝叶斯网络、双向推断网络或随机过程。

此外，纽约大学 Wilson 教授发表的学术论文“The Case for Bayesian Deep Learning”，科学地对贝叶斯深度学习存在的误区进行了回应，主要观点为：①贝叶斯的核心特征是边际化，即贝叶斯模型平均；②传统方法是贝叶斯边际化的一种特例；③深度集成方法本质上也是贝叶斯的，但贝叶斯模型平均和深度集成的使用场景有所不同；④先验的重要性主要体现在函数空间中，而不是贝叶斯神经网络的参数空间中；⑤贝叶斯深度学习的研究在方法和实用性上都在蓬勃发展。因此，现代深度学习(特别是大模型和多模态模型)与 PGM 深度相结合，在理论和应用方面将会带来概率机器学习的大发展。

在贝叶斯(贝叶斯深度学习)实现方面，现已实现了编程框架的概率编程语言(Probabilistic Programming Languages，PPL)。概率编程语言旨在描述概率分布，同时以类似 PGM 的方式描述变量之间的关系，以加快推断速度。大多数概率编程语言包括许多不同的优化算法，如 MCMC、变分推断和最大似然估计等。Stan 和 PyMC3 是目前普遍使用的概率编程语言。Stan 是一种灵活的概率建模语言，可以使用贝叶斯技术直接估计多种类型的概率模型，使用一套自己的语言(包括 Python、R、MATLAB、Julia 和一个命令行接口)定义概率模型。PyMC3 用 Python 编写，基于 Theano 实现自动微分。随着深度贝叶斯模型的出现，深度学习框架下的概率编程成为当下流行的趋势，其最主要的要求是自动微分和大规模数据的优化程序。这些概率编程语言与各自的深度学习框架无缝集成，进而使其成为设计、训练和使用贝叶斯神经网络的理想工具。这些工具主要包括 Pyro(构建在 PyTorch 之上)和 Edward(构建在 TensorFlow 之上)，Edward 已经被扩展并集成到了 TensorFlow 的 TensorFlow Probability(TFP)子模块中。本书主要采用 PyMC3 和 TFP 进行建模和计算。

Pyro 的开发重点是变分推断。它基于函数式编程范式，借助上下文管理器来轻松地修改随机函数。Pyro 包括一个封装类，可以将 PyTorch 的网络层转换为概率层，并允许用运行中(on-the-fly)的样本替换网络参数。Edward 和 TFP 包括一些更高层次的结构，特别是可单独使用的概率层，以及一些建立概率网络的低层次结构。Pyro 可能更适合动态 PGM，因为 PyTorch 使用了动态计算图技术。

本书内容翔实，推导过程简洁，代码优雅，将理论与实践结合进行贝叶斯建模与计算。本书主要由三部分组成，第一部分为贝叶斯推断和探索性分析(第 1 章和第 2 章)；第二部分为常见贝叶斯建模(第 3~8 章)，主要包括线性模型、样条、时间序列、贝叶斯加性回归树、近似贝叶斯推理；第三部分为贝叶斯工作流和概率编程语言(第 9 章和第 10 章)。此外，第 11 章和词汇表介绍了本书所涉及的数理统计基础知识、编程常识及专业术语。尤其对熵、Kullback Leibler 散度、边际似然、推断方法等基础知识进行了补充说明。本书可作为统计学、计算机科学、人工智能等专业本科生和研究生的课程教材，也可作为知识图谱、贝叶斯深度学习等方面的工程师和科学家的参考书。

在本书的翻译过程中，我查阅了大量的经典著(译)作，也得到了很多人的帮助。感谢本书的审校者——吉林大学外国语学院吴禹林和吉林财经大学外国语学院张煜琪分别完成了整本书的审校，感谢她们所做的工作。最后，感谢清华大学出版社的编辑，他们做了大量的编辑与校对工作，保证了本书的质量，使本书符合出版要求。在此深表谢意。

由于本书涉及的内容广度和深度较大，加上译者翻译水平有限，在翻译过程中难免有不足之处。若各位读者在阅读过程中发现问题，欢迎批评指正。

译者

译 者 简 介

郭涛，主要从事人工智能、现代软件工程、智能空间信息处理以及时空大数据挖掘与分析等前沿交叉研究，已翻译并出版《深度强化学习图解》《AI 可解释性(Python 语言版)》《概率图模型原理与应用(第 2 版)》等多部畅销作品。

致　谢

感谢 Romina 和 Abril 对我无微不至的爱。感谢所有帮助过我的人。

——Osvaldo Martin

感谢向我传授知识、为我启迪思维的师长亲朋，是他们无条件的分享成就了如今的我。感谢 Tim Pegg、Michael Collins 先生、Sara LaFramboise Saadeh 夫人、Mehrdad Haghi 教授、Winny Dong 教授、Dixon Davis 教授、Jason Errington、Chris Lopez、John Norman、Ananth Krishnamurthy 教授和 Kurt Campbell(按与我相识的时间顺序排列)。

谢谢你们!

——Ravin Kumar

感谢 Yuli。

——Junpeng Lao

推 荐 序

贝叶斯建模为许多数据科学和决策问题提供了一种简明的方法，但在实践中很难充分发挥其作用。尽管有许多软件库可以轻松指定复杂的分层模型，如 Stan、PYMC3、TensorFlow Probability(TFP)和 Pyro，但用户仍然需要额外的工具来诊断其计算结果是否正确。应对问题时，他们也需要一些实践建议。

本书重点介绍 ArviZ 软件库。该库使用户能够对贝叶斯模型进行探索性分析(例如对任何推断方法生成的后验样本进行诊断)，并可用于诊断贝叶斯推断中的各种故障模式。本书还讨论了多种建模策略(如中心化处理)，可用于消除许多常见问题。本书的大多数示例使用了 PYMC3，另外一些使用了 TFP。书中还包括对其他概率编程语言的简要比较。

本书的作者都是贝叶斯软件领域的专家，并且是 PYMC3、ArviZ 和 TFP 软件库的主要贡献者。他们在实际应用贝叶斯数据分析方面也拥有丰富经验，这在本书采用的各种实用方法中有所体现。总体来说，本书丰富了现有文献，有望进一步推动贝叶斯方法的应用。

——Kevin P. Murphy

前　言

贝叶斯统计这个名字取自长老会牧师兼业余数学家托马斯·贝叶斯(Thomas Bayes，1702—1761)，他最先推导出了贝叶斯定理，该定理于其逝世后的 1763 年发表。但真正开发贝叶斯方法的第一人是 Pierre-Simon Laplace(1749—1827)，因此将其称为拉普拉斯统计也许更合理。尽管如此，我们将遵循斯蒂格勒的同名法则，在本书的其余部分使用传统的贝叶斯方法命名。从贝叶斯和拉普拉斯(以及许多其他人)的开创性时代至今，发生了很多事情，特别是开发了很多新想法，其中大部分是由计算机技术推动和/或实现的。本书旨在提供一个关于此主题的现代视角，涵盖从基础知识到使用现代贝叶斯工作流和工具等各方面内容。

本书旨在帮助贝叶斯初学者成为中级从业者。这并不代表你读完本书后会自动达到中等水平，但希望本书能够引导你朝富有成效的方向发展。如果你通读这本书，认真做练习，把书中的想法应用于自己的问题，并继续向他人学习，那么将更容易进步。

要特别指出，本书面向对应用贝叶斯模型解决数据分析问题感兴趣的贝叶斯从业者。通常，学术界和工业界是有区别的。但本书没有做这样的区分，因为无论是大学生还是就职于公司的机器学习工程师，都能从本书中受益。

我们的目标是：阅读本书后，你不仅能够熟悉**贝叶斯推断**，而且能轻松地对贝叶斯模型进行**探索性分析**，包括模型比较、模型诊断、模型评估和结果交流等。我们计划从现代计算的角度讲授这些内容。对我们来说，如果采用**计算方法**，贝叶斯统计会更易于理解和应用。例如，我们更关注实证检查假设被推翻的原因，而不试图从理论上证明假设是正确的。这也意味着我们会使用许多可视化的表达手段。通读后，建模方法的其他含义将会逐步变得清晰。

如本书标题所表明的，书中使用 Python 编程语言。更具体地说，本书将主要使用 PYMC3[138] 和 TensorFlow Probability(TFP)[47]作为模型构建和推断的主要概率编程语言(PPL)，并使用 ArviZ 作为探索性分析贝叶斯模型的主要软件库[91]。本书并未对所有 Python PPL 进行详尽评述和比较，因为选择较多而且发展迅速。反之，我们专注于贝叶斯分析的实践方面。编程语言和软件库只是用于达到目的的手段。

虽然本书选择的编程语言是 Python 及少量软件库，但书中涵盖的统计和建模概念基本与编程语言和软件库无关，可以应用于许多计算机编程语言，如 R、Julia 和 Scala 等。因此，虽不了解 Python 但掌握上述编程语言的读者，也可以从本书中受益。当然，如果能够在自身熟悉的编程语言中找到等效的软件库或代码进行实践则最好。此外，我们鼓励将本书中的 Python 示例代码转换为其他编程语言或框架。有意者请与我们联系。

知识准备

为使本书帮助初学者向中级从业者转变，希望读者能事先接触乃至掌握贝叶斯统计的基本概念(如先验、似然和后验)，以及一些基本统计概念(如随机变量、概率分布、期望等)。对于技艺生疏的读者，第 11 章回顾了基本统计概念。有关这些概念的更深入解释，参见 *Understanding Advanced Statistical Methods*[158]和 *Introduction to Probability*[21]。后者更具理论性，但两者都比较重视应用。

如果你因实践或训练对统计学有很好的理解，但从未接触过贝叶斯统计学，也可以将本书作为对该主题的入门读物，只是开始几章(主要是前两章)的节奏会有点快，可能需要通读数次。

我们希望你能够熟悉一些数学概念，如积分、导数和对数的性质等，写作水平最好能够达到技术高中或者科学、技术、工程和数学专业的大学第一学年以上的水平。若需要复习这些数学概念，推荐 3Blue1Brown 的系列视频。这里不要求做过多数学练习，但要求使用代码和交互式计算环境来理解和解决问题。本书中出现数学公式是为了帮助你更好地理解贝叶斯统计建模。

本书假定读者具备一定的计算机编程能力。使用 Python 语言时，还会使用一些专门的软件库，特别是概率编程语言。在阅读本书之前，至少利用概率编程语言拟合一个模型，对你会有帮助，但也不是必须的。关于如何设置本书所需要的计算环境或 Python 参考，可以阅读 GitHub 中的 README.md，了解如何设置编码环境。

阅读方法

我们将使用模拟模型来解释一些重要概念，而不会让数据模糊了主要概念；然后使用真实数据集来近似一些实践中会面临的真实问题，如采样问题、重参数化、先验/后验校准等。鼓励你阅读本书时，在交互式编程环境中运行这些模型。

强烈建议你阅读并使用各种软件库的在线文档。尽管我们已经尽最大努力使本书涵盖海量信息，从而自成一体，但在网上还有大量关于这些工具的文档，参考这些文档有助于学习本书，并帮助你独立使用这些工具。

第 1 章回顾、简介贝叶斯推断中的基本和核心概念。该章中的概念将在本书其余部分被反复提及和应用。

第 2 章介绍了贝叶斯探索性分析(Exploratory Analysis of Bayesian)模型。介绍了许多属于贝叶斯工作流但并非推断本身的概念。该章中的概念将在本书其余部分被反复应用和提及。

第 3 章开始介绍特定模型架构。介绍了线性回归(Linear Regression)模型，并为接下来的 5 章奠定了基础。第 3 章还全面介绍了本书使用的主要概率编程语言：PyMC3 和 TFP。

第 4 章扩展了线性回归模型，并讨论了更高级的主题，如鲁棒回归、分层模型和模型重参数化。本章使用 PyMC3 和 TFP。

第 5 章介绍了基函数，并着重介绍了线性模型的扩展——样条，使我们能够构建更灵活的模型。本章使用 PyMC3。

第 6 章侧重于时间序列模型，包括从时间序列建模为回归模型，以及更复杂的模型[如 ARIMA 和线性高斯状态空间(Gaussian State Space)模型]等内容。本章使用 TFP。

第 7 章介绍了名为贝叶斯加性回归树的非参数模型。本章讨论了这个模型的可解释性和变量的重要性。本章使用 PyMC3。

第 8 章聚焦于逼近贝叶斯计算(Approximate Bayesian Computation，ABC)框架，该框架有助于解决没有明确似然函数的问题。本章使用 PyMC3。

第 9 章概述了端到端的贝叶斯工作流。本章展示了商业应用中的观测性研究和科研环境中的试验性研究。本章使用 PyMC3。

第 10 章深入探讨了概率编程语言，展示了各种不同的概率编程语言。

第 11 章为阅读其他章节提供辅助，各主题之间相关度不高，因此可以有选择地阅读。

强调内容

本书对文本突出强调的方式是用粗体。**粗体文本**表示强调新概念或概念的重点。当提到特定代码时，也会突出显示，如 pymc3.sample。

代码

书中的代码块用阴影框标记，左侧带有行号，并使用章节编号后跟代码块编号进行引用，如代码清单 0.1 所示。

代码清单 0.1

```
1 for i in range(3):
2     print(i**2)
```

```
0
1
4
```

每次看到代码块时都会想查看运行结果。结果通常体现为一张图、一个数字、一份代码输出或一个表格。反之，书中大部分图都有相关的代码，有时会省略一些代码以节省篇幅，但你可以在 GitHub 库(https://github.com/BayesianModelingandComputationInPython)中访问完整代码。该库还包括一些用于练习的附加材料。其中的笔记还可能包含其他图、代码或输出，这些内容未出现在书中但用于开发书中所见模型。GitHub 中还包含说明，指导如何根据已有设备创建标准计算环境。

方框

本书使用方框简要提及重要的统计、数学或(Python)编程概念。书中还会提供参考资料，供你继续学习相应主题。

中心极限定理(Central Limit Theorem)

在概率论中，中心极限定理规定：在某些情况下，添加独立随机变量时，即使原始变量本身不呈正态分布，但它们的适当归一化总和也会趋于正态分布。

设 X_1，X_2，X_3，…独立同分布，平均值为 μ，标准差为 σ。当 $n \to \infty$ 时，有：

$$\sqrt{n}\left(\frac{\bar{X}-\mu}{\sigma}\right) \xrightarrow{\mathrm{d}} \mathcal{N}(0,1)$$

Introduction to Probability [21]一书介绍了许多概率基础理论，可用于实践。

代码导入

在本书中，导入 Python 包时使用代码清单 0.2 所示的约定。

代码清单 0.2

```
# 基本的
import numpy as np
from scipy import stats
import pandas as pd
from patsy import bs, dmatrix
import matplotlib.pyplot as plt

# 贝叶斯模型探索性分析
import arviz as az

# 概率编程语言
import bambi as bmb
import pymc3 as pm
import tensorflow_probability as tfp

tfd = tfp.distributions

# 计算后端
import theano
import theano.tensor as tt
import tensorflow as tf
```

本书还会使用 ArviZ 样式：az.style.use("arviz-grayscale")。

由于本书是黑白印刷，本书中的彩图为方便读者阅读，以彩插形式放在封底二维码，读者可自行下载。

与本书互动

本书的受众不是贝叶斯读者，而是贝叶斯从业者。我们将提供材料，帮助练习贝叶斯推断和贝叶斯模型探索性分析。由于利用计算和代码是现代贝叶斯从业者所需要的核心技能，因此将提供示例，以供你在多次尝试中建立思维。对于本书代码，我们期望你阅读、执行、修改，并再次执行多次。我们只能在本书中展示有限示例，但你可以使用计算机自己制作无数的示例。通过这种方式，你不仅可以学习统计概念，还可以学习如何使用计算机将这些概念应用于实践。

计算机还将使你摆脱印刷文本的限制，例如缺乏颜色、缺乏动画和并排比较。现代贝叶斯从业者利用监视器和快速可计算“双重检查”提供的灵活性，本书专门创建了示例以允许相同级别的交互性。每章的末尾都设有练习，用于测试学习和实践成果。练习按难易程度标记为简单(E)、中等(M)和困难(H)，可根据需要酌情解答。

本书的参考文献可下载封底二维码获取。

致谢

感谢我们的朋友和同事，他们牺牲了大量时间和精力来阅读早期书稿，提出了建设性的反馈，帮助我们改进了本书，也帮助我们修复了书中的许多错误。非常感谢：

Oriol Abril-Pla、Alex Andorra、Paul Anzel、Dan Becker、Tomás Capretto、Allen Downey、Christopher Fonnesbeck、Meenal Jhajharia、Will Kurt、Asael Matamoros、Kevin Murphy 以及 Aki Vehtari。

符 号 表

符号	解释
$\log(x)$	x 的自然对数
$\mathbb{R}$	实数
$\mathbb{R}^n$	实数的 n 维向量空间
A, S	集合
$x \in A$	集合成员。x 是集合 A 的一个元素
$\mathbb{1}_A$	指示函数。当 $x \in A$ 时返回 1，否则返回 0
$a \propto b$	a 与 b 成比例
$a \propto\!\!\!\!\sim\ b$	a 与 b 近似成比例
$a \approx b$	a 约等于 b
a, c, α, γ	标量用小写字母表示
$\mathbf{x}, \mathbf{y}$	向量用粗体小写字母表示，因此将列向量写为 $\mathbf{x} = [x_1, ..., x_n]^T$
$\mathbf{X}, \mathbf{Y}$	矩阵用粗体大写字母表示
X, Y	随机变量用大写罗马字母表示
x, y	随机变量的结果用小写罗马字母表示
$\boldsymbol{X}, \boldsymbol{Y}$	随机向量采用粗斜体的大写字母表示，$\boldsymbol{X} = [X_1, ..., X_n]^T$
θ	模型参数用小写的希腊字母表示。需要注意的是，在贝叶斯统计中，参数通常被视为随机变量
$\hat{\theta}$	θ 的点估计
$\mathbb{E}_X[X]$	随机变量 X 关于 X 的期望，大多时候被简写为 $\mathbb{E}[X]$
$\mathbb{V}_X[X]$	随机变量 X 关于 X 的方差，大多时候被简写为 $\mathbb{V}[X]$
$X \sim p$	随机变量 X 服从分布 p
$p(\cdot)$	概率密度函数或概率质量函数
$p(y \mid \boldsymbol{x})$	在给定 $\boldsymbol{x}$ 时，y 的概率(密度)。这是 $p(Y = y \mid \boldsymbol{X} = \boldsymbol{x})$ 的简写。
$f(x)$	关于 x 的任意函数
$f(\boldsymbol{X}; \theta, \gamma)$	f 是 $\boldsymbol{X}$ 的函数，其参数为 θ 和 γ。使用这个符号强调 $\boldsymbol{X}$ 是传递给函数(或模型)的数据，而 θ 和 γ 是函数的参数

$\mathcal{N}(\mu, \sigma)$	平均值为 μ、标准差为 σ 的高斯(或正态)分布
$\mathcal{HN}(\sigma)$	标准差为 σ 的半高斯(或半正态)分布
Beta(α, β)	形状参数为 α 和 β 的贝塔分布
Expo(λ)	速率参数为 λ 的指数分布
$\mathcal{U}(a, b)$	下界为 a、上界为 b 的均匀分布
$T(\nu, \mu, \sigma)$	高斯等级(也称自由度)为 ν、位置参数为 μ(当 $\nu>1$ 时的平均值)、缩放参数为 σ(当 $\lim_{\nu\to\infty}$时的标准差)的学生 t 分布
$\mathcal{H}$T(ν, σ)	高斯等级(也称自由度)为 ν、尺度参数为 σ 的半学生 t 分布
Cauchy(α, β)	位置参数为 α、尺度参数为 β 的柯西分布
$\mathcal{H}$C(β)	尺度参数为 β 的半柯西分布
Laplace(μ, τ)	平均值为 μ、尺度为τ的拉普拉斯分布
Bin(n, p)	总试验次数为 n，成功次数为 p 的二项分布
Pois(μ)	平均值(和方差)为 μ 的泊松分布
NB(μ, α)	泊松参数为 μ、伽马分布参数为 α 的负二项分布
$gRW(\mu, \sigma)$	新偏移为 μ、新标准差为 σ 的高斯随机游走分布
$\mathbb{KL}(p \parallel q)$	p 到 q 的 Kullback-Leibler(KL)散度

目　录

第 1 章

贝叶斯推断

现代贝叶斯统计主要使用计算机代码执行。从几十年前开始，这就极大地改变了贝叶斯统计的执行方式。我们能够构建越来越复杂的模型，必要的数学和计算技能的障碍逐步减少。而且迭代式建模过程也在多个方面变得比以往更容易实施、更有价值。计算机方法的普及和流行有很大益处，但也需要承担更多责任。现在表达统计方法比以往任何时候都容易，但再强大的计算方法也无法替代统计的微妙之处。因此，具有良好理论知识背景(尤其是与实践相关的知识)是有效应用统计方法的基础。本章先介绍一些基础概念和方法，还有很多内容将在本书的其余部分进一步探索和扩展。

1.1 贝叶斯建模

概念模型是对一个系统的表征，它由若干概念组成，用于帮助人们了解、理解或模拟该模型所代表的对象或过程[39]。此外，模型是人为设计的表征，具有特定的目标。因此，讨论某个(些)模型对于指定问题的充分性，通常比讨论模型内在的正确性更方便。模型的存在仅仅是为了帮助人们实现进一步的目标。

在设计新车时，汽车公司会制作实体模型，以帮助人们理解产品在制造时的外观。此时，有一位雕刻家具有汽车先验知识并且善于估计模型的用处。他会寻找所需要的黏土等原材料，并使用手工工具雕刻实体模型。此实体模型能够帮助其他人了解设计，例如外观是否美观、形状是否符合空气动力学等。这样的模型需要同时结合汽车设计专业知识和雕刻知识才能得到想要的结果。此外，建模过程通常需要构建多个模型，以探索不同的选择，或者是因为需要与其他汽车设计团队交流，以获得迭代式的改进和扩展。如今，除了上述实体汽车模型外，用计算机辅助设计软件制作数字模型也很常见。计算机模型与实体模型相比有自身的优势。例如，与在实体汽车模型上进行测试相比，使用数字模型进行碰撞模拟更简单、成本更低，与团队内不同领域的同事共享模型也更加容易。

贝叶斯建模的理念与上述汽车建模非常相似。要构建一个贝叶斯模型，需要结合领域专业知识和统计技能，从而将知识整合到一些可计算的目标中，并确定结果的可用性。在贝叶斯建模场景中，所使用的“原材料”是数据，而“雕刻”统计模型的主要数学工具是统计分布。人们需要结合领域专业知识和统计知识，才能获得有用的结果。此外，贝叶斯从业者同样会以迭

代方式构建多个模型，往往其中的第一个为基础模型，主要用于帮助从业者识别自身思维的差距或其模型存在的缺陷。然后这些早期模型用于构建后续的改进模型和扩展模型。此外，使用一种推断机制并不会阻碍其他推断机制发挥作用，就像汽车的实体模型不会阻碍数字模型发挥作用一样。现代贝叶斯从业者也有很多方式来表达想法、生成结果和分享输出，从而使从业者及其同行能够更广泛地推广其积极成果。

1.1.1 贝叶斯模型

贝叶斯模型，无论其是否可计算，都有两个基本特征：

- 使用概率分布描述未知量[1]，通常称这些未知量为参数[2]。
- 采用贝叶斯定理，根据数据更新参数的值，此过程也可以被视为概率的重新分配。

在高层次上，可以将贝叶斯模型的构建过程分为以下 3 个步骤。

(1) 给定一些数据，以及关于如何生成这些数据的假设，通过组合(combing)和转换(transforming)随机变量来设计模型。

(2) 利用贝叶斯定理，根据现有数据调整模型，我们称此过程为**推断(inference)**。推断的结果是获得了参数的后验分布。我们希望已有的数据能够减少可能参数值的不确定性，但并非所有贝叶斯模型都能够保证做到。

(3) 根据不同标准检查模型是否有意义，进而对模型进行评判。这些标准包括数据及专业领域知识等。由于模型具有不确定性，因此有时会比较多个模型。

如果你熟悉其他形式的建模，就会认识到模型评判的重要性，以及迭代式执行上述 3 个步骤的必要性。例如，我们可能需要在任何给定点回溯历史步骤。这也许是因为引入了一个低级编程错误，或者在经历一些挑战之后找到了改进模型的方法，或者发现数据不像最初想象的那样可用，以至于需要收集更多数据甚至是不同类型的数据。

本书会详细讨论每一步的实施方法，并学习如何将上述简单流程扩展到更为复杂的**贝叶斯工作流(Bayesian Workflow)**。贝叶斯工作流非常重要，因此将用完整的一章(第 9 章)阐释该主题。

1.1.2 贝叶斯推断介绍

通俗地说，推断是指“根据证据和原因得出结论”。贝叶斯推断是一种特殊形式的统计推断，通过组合概率分布来获得其他概率分布。贝叶斯定理提供了一种通用方法，用于在观测到一些数据 $\boldsymbol{Y}$ 时，估计参数 $\boldsymbol{\theta}$：

$$\underbrace{p(\boldsymbol{\theta} \mid \boldsymbol{Y})}_{\text{后验}} = \frac{\overbrace{p(\boldsymbol{Y} \mid \boldsymbol{\theta})}^{\text{似然}}\ \overbrace{p(\boldsymbol{\theta})}^{\text{先验}}}{\underbrace{p(\boldsymbol{Y})}_{\text{边际似然}}} \tag{1.1}$$

1 更宽泛一些说，甚至可以说一切都是一个概率分布，作为一个你假设知道的量，任意精度由 Dirac δ 函数描述。

2 一些著作将这些量称为潜在变量，并保留名称参数，以识别固定但未知的量。

式(1.1)中，似然函数(简称似然，likelihood)将观测数据($\boldsymbol{Y}$)与未知参数($\boldsymbol{\theta}$)连接起来；先验分布(简称先验，prior)表示在观测到数据 $\boldsymbol{Y}$ 之前参数的不确定性[1]；通过将两者相乘，可以得到后验分布(简称后验，posterior)，即给定观测数据的条件下，模型中所有未知参数的联合分布。图 1.1 展示了一个任意的先验分布、似然函数，以及两者产生的后验分布[2]。

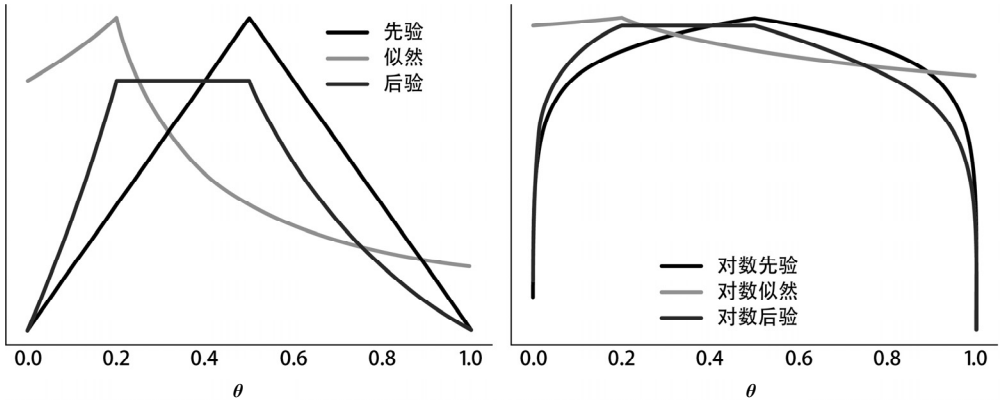

图 1.1　左图为一个假想的先验分布(黑色曲线)，其中 $\boldsymbol{\theta}=0.5$ 的可能性最大，而其余值则呈现出线性对称的下降趋势；似然函数(灰色曲线)则表明，$\boldsymbol{\theta}=0.2$ 能更好地解释数据；而两者相乘后的后验分布(蓝色曲线)，则是先验和似然之间的折中。图中省略了 y 轴的值和刻度，因为我们只关注相对值。右图与左图类似，但其 y 轴采用了对数尺度。可以发现对数尺度能够保留相对值，例如，两个图中最大值和最小值所在位置并没有改变。由于对数尺度的数值更稳定，计算时往往被作为首选

注意，虽然 $\boldsymbol{Y}$ 是观测数据，但会被视为随机向量，因为其值取决于特定试验结果[3]。为了获得后验分布，会将数据 $\boldsymbol{Y}$ 的值固定在实际观测值上不变，因此一个常见的替代表示符号是 y_{obs}。

正如式(1.1)和图 1.1 所示，在某个特定点上计算后验的值，从概念层面来说非常简单，只需要将一个先验值乘以一个似然值即可。但后验概率并非仅限于此，因为不仅需要特定点的绝对后验值，还需要其与周围点的相对后验值。后验分布的这种全局信息由**边际归一化常数** $p(\boldsymbol{Y})$ 表示。然而，这个常数难以计算。把边际似然写成式(1.2)可能更容易理解这一点。

$$p(\boldsymbol{Y}) = \int_{\Theta} p(\boldsymbol{Y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \tag{1.2}$$

其中Θ表示需要对 $\boldsymbol{\theta}$ 的所有可能值做积分。

计算这样的积分实际上非常难。对于大多数问题而言，可能根本无法给出边际似然的解析表达式。好在有一些数值方法可以应对这一挑战。另外，实践中的很多问题并不需要计算边际似然，此时将贝叶斯定理表示为比率形式比较常见。

$$\underbrace{p(\boldsymbol{\theta} \mid \boldsymbol{Y})}_{\text{后验}} \propto \overbrace{p(\boldsymbol{Y} \mid \boldsymbol{\theta})}^{\text{似然}} \overbrace{p(\boldsymbol{\theta})}^{\text{先验}} \tag{1.3}$$

1 也可以从确定性或信息的角度考虑这一点，这取决于你是悲观的人还是乐观的人。

2 有时单词的分布是隐含的，讨论这些主题时通常如此。

3 这里指广义的试验，即用于收集或生成数据的任意程序。

关于符号的说明

在本书中，使用了相同的符号 $p(\cdot)$表示不同的量，如似然函数和先验概率分布。这其实是对符号的轻微滥用，但也有一定好处：一是为贝叶斯公式中的所有量都提供了相同的认识论地位；二是反映出即便似然不是严格意义上的概率密度函数，也能接受，因为我们只关注先验背景下的似然，反之亦然。换句话说，为了计算后验分布，将似然和先验分布视为模型中同等必要的元素。

贝叶斯统计的特点之一是：后验(总)是一个概率分布。这使我们能够对参数做出概率性的表示。例如参数τ为正的概率是 0.35。或者 Φ 介于 10 和 15 之间的概率为 50%，最可能的值是 12。此外，还可以将后验分布视为将模型与数据相结合所得的逻辑结果，因此保证了由其得出的概率陈述在数学上的一致性。我们只需要记住，实现所有这些好的数学性质的前提是，存在球体、高斯和马尔可夫链等数学对象，即其只在理想情况下有效。当从抽象的数学理论转向现实世界中复杂的数学应用时，必须始终牢记：结果不仅取决于数据，还取决于模型。在实际问题中，有时候即便具有数学一致性，不良数据和/或不良模型也有可能导致毫无意义的结果。因此，必须始终辩证地看待数据、模型和结果。为了更明确这一点，可以更准确地表示贝叶斯定理如下：

$$p(\boldsymbol{\theta} \mid \boldsymbol{Y}, M) \propto p(\boldsymbol{Y} \mid \boldsymbol{\theta}, M)\, p(\boldsymbol{\theta}, M) \tag{1.4}$$

式(1.4)显式地强调了：推断总是依赖于模型 M 的假设。

得到后验分布后，就可以基于它推导出有关参数的其他量。而这通常以计算期望的方式实现，例如：

$$J = \int f(\boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \boldsymbol{Y})\, \mathrm{d}\boldsymbol{\theta} \tag{1.5}$$

当f为恒等函数时，积分 J 就是参数 $\boldsymbol{\theta}$ 的期望平均值[1]：

$$\bar{\boldsymbol{\theta}} = \int_{\Theta} \boldsymbol{\theta} p(\boldsymbol{\theta} \mid \boldsymbol{Y})\, \mathrm{d}\boldsymbol{\theta} \tag{1.6}$$

后验分布是贝叶斯统计的核心对象之一。除了要对参数值进行推断外，也需要对数据做出推断。这可以通过计算**先验预测分布**来完成：

$$p(\boldsymbol{Y}^*) = \int_{\Theta} p(\boldsymbol{Y}^* \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \tag{1.7}$$

式(1.7)根据模型(即先验和似然)得出了预期的数据概率分布。这是在得到任何实际观测数据 $\boldsymbol{Y}^*$之前，根据给定模型所得的预期数据。但要注意：式(1.2)(边际似然)和式(1.7)(先验预测分布)看起来很相似，但也存在不同。在边际似然公式中，有已知的观测数据 $\boldsymbol{Y}$ 作为前提条件，最终积分结果是一个数字；而在先验预测分布的公式中，并没有观测数据作为已知前提，最终结果表现为概率分布。

可以使用先验预测分布的样本作为评估和校准模型的一种手段。例如，我们可能会问：“人

1 严格来讲，应该讨论随机变量的期望值。详见 11.1.8 节。

类身高的模型能否将人类身高预测为－1.5 米？”对于此类问题，通常在实际观测之前，就能认识到其荒谬性。在本书后面，会介绍许多在实践中使用先验预测分布进行模型评估的案例，以及使用先验预测分布如何为后续建模选择提供有效或无效信息。

作为生成式模型的贝叶斯模型

采用概率视角建模可以得到一个准则：模型生成数据[158]。我们认为此概念至关重要。理解掌握后，所有统计模型都会变得更加清晰，甚至包括非贝叶斯模型。此准则可以指导我们创建新的模型。如果数据是由模型生成的，那么仅通过思考如何生成数据，就能为数据创建适合的模型！此外，此准则并非抽象概念，我们可以将先验预测分布作为其具体表现形式。如果重新审视贝叶斯建模的 3 个步骤，可以将它们重新调整为：编写先验预测分布，添加数据以对其进行约束，检查结果是否有意义。当然，必要时同样需要进行迭代。

另一个需要计算的量是**后验预测分布**：

$$p(\tilde{\boldsymbol{Y}} \mid \boldsymbol{Y}) = \int_{\Theta} p(\tilde{\boldsymbol{Y}} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \boldsymbol{Y})\, \mathrm{d}\boldsymbol{\theta} \tag{1.8}$$

这是根据后验 $p(\boldsymbol{\theta} \mid \boldsymbol{Y})$预测的未来数据 $\tilde{\boldsymbol{Y}}$ 的分布，而分布是模型(先验和似然)和观测数据的结果。因此，后验预测分布是模型在得到数据集 $\boldsymbol{Y}$ 后预期得到的未来数据，即该分布是模型的预测结果。从式(1.8)可知，通过对参数的后验分布进行积分(边际化)来计算预测。因此，该预测包含了估计的不确定性。

频率主义者眼中的贝叶斯后验

因为后验仅来自模型和观测数据，所以并不是基于未观测到的数据做出陈述，而是基于内在数据生成过程得到的可能观测做出陈述。对未观测到的数据做出推断通常是频率主义者常用的方法。但在使用后验预测样本检查模型时，贝叶斯主义者其实(部分)接受了频率主义者关于“未观测但可能可观测的数据”的思想。我们不仅能够接受该想法，而且将在本书中涵盖多个示例。这真是绝妙的概念，值得进一步探究！

1.2　一个自制采样器，不要随意尝试

式(1.2)中的积分有时没有解析表达式，因此如今大多使用**通用推断引擎(Universal Inference Engines)**这一数值方法进行贝叶斯推断(参见 11.9 节)。有许多通过测试的 Python 软件库能够提供此类数值方法，因此一般来说，贝叶斯从业者不太可能需要编写自己的通用推断引擎。

当前编写自己的推断引擎通常只有两个理由：一是为了设计一个能够改进旧引擎的新引擎；二是为了学习某个引擎的工作原理。本章出于学习目的，将编写一个简易的引擎，但本书其余部分主要使用 Python 库中的可用推断引擎。

可用于通用推断引擎的算法很多，其中使用最广泛、功能最强的算法是马尔可夫链蒙特卡罗(Markov Chain Monte Carlo，MCMC)方法。所有 MCMC 方法几乎都使用样本来逼近后验分

布，而这些样本大多通过接受或拒绝来自某个提议分布的样本生成。我们有理论上的保证，即通过遵循某些规则[1]和假设能够获得非常逼近后验分布的样本。因此，MCMC 方法也称为采样器(Sampler)。所有这些 MCMC 方法都需要具备在给定参数值时估算先验和似然的能力。也就是说，即使不知道完整的后验，通过逐点计算，也能够获取其概率密度。

此类算法之一是 Metropolis-Hastings[103,78,135]。这并不是一个非常现代且有效的算法，但很容易理解，并为理解更复杂、更强大的其他方法奠定了基础。[2]

Metropolis-Hasting 算法定义如下：

(1) 在 x_i 处初始化参数 $\boldsymbol{X}$ 的值。

(2) 使用提议分布[3] $q(x_{i+1}|x_i)$ 从旧值 x_i 生成新值 x_{i+1}。

(3) 计算新值被接受的概率：

$$p_a(x_{i+1} \mid x_i) = \min\left(1, \frac{p(x_{i+1})\ q(x_i \mid x_{i+1})}{p(x_i)\ q(x_{i+1} \mid x_i)}\right) \tag{1.9}$$

(4) 如果 $p_a > R$ 其中 $R \sim \mathcal{U}(0, 1)$，则保留新值，否则保留旧值。

(5) 迭代步骤(2)～(4)，直到生成足够大的样本。

Metropolis-Hasting 算法非常通用，而且可以用于非贝叶斯应用。但对于本书内容，$p(x_i)$ 是参数值为 x_i 的后验密度。如果 q 是对称分布，则式(1.9)中的 $q(x_i|x_{i+1})$ 和 $q(x_{i+1}|x_i)$ 将被消掉(这在概念上意味着从 x_{i+1} 转移到 x_i 与从 x_i 转移到 x_{i+1} 具有相同的可能性)，只留下在两个点处估计的后验之比。从式(1.9)可知，该算法始终接受从低概率区到高概率区的转移，接受从高概率区到低概率区的转移则需要一定概率。

需要说明的是：Metropolis-Hastings 算法并不是一种优化方法！我们不关注概率密度最大的参数值，而是想探索整个 p 分布(后验分布)。如果深入洞察和分析，就会发现此方法在达到最大概率区域后并不会停止，而是在后续步骤中继续转移到概率较低的区域。

下面尝试求解一个贝塔二项式(Beta-Binomial)模型。这可能是贝叶斯统计中最常见的示例，常用于对二值的、互斥的事件进行建模，例如 0 或 1、正或负、正面或反面、垃圾邮件或非垃圾邮件、热狗或非热狗、健康或不健康等。贝塔二项式模型经常被用作介绍贝叶斯统计基础知识的第一个示例，因为它足够简单，可以轻松求解和计算。在统计符号系统中，贝塔二项式模型记为：

$$\begin{aligned} \boldsymbol{\theta} &\sim \text{Beta}(\alpha, \beta) \\ Y &\sim \text{Bin}(n = 1, p = \boldsymbol{\theta}) \end{aligned} \tag{1.10}$$

在式(1.10)中，未知参数为 $\boldsymbol{\theta}$，其先验分布为 Beta(α, β)；假设数据的似然函数为二项分布 Bin($n=1$, $p=\boldsymbol{\theta}$)。在此模型中，成功的次数 $\boldsymbol{\theta}$ 可以代表抛硬币得到正面的比例、病亡率(统计有时也涉及消极信息)等量。贝塔二项式模型实际上存在封闭形式的解(详见 1.4.1 节)，但这里为了方便讲解示例，假设不知道如何计算该模型的后验。因此需要在 Python 代码中实现 Metropolis-Hastings 算法，以获得逼近的数值解。在 SciPy 软件库的统计函数支持下，可以实现

1 详见 11.1.11 节和 11.9.2 节。

2 关于推断方法的进一步讨论，参见 11.9 节和其附带的参考文献。

3 在其他通用推断引擎中，有时被称为内核。

为代码清单 1.1。

代码清单 1.1

```
def post(θ, Y, α=1, β=1):
    if 0 <=θ <= 1:
        prior = stats.beta(α, β).pdf(θ)
        like = stats.bernoulli(θ).pmf(Y).prod()
        prob = like * prior
    else:
        prob = -np.inf
    return prob
```

推断后验分布需要引入观测数据，为此随机生成了一些伪数据，见代码清单 1.2。

代码清单 1.2

```
Y = stats.bernoulli(0.7).rvs(20)
```

最后，运行 Metropolis-Hastings 算法的实现，见代码清单 1.3。

代码清单 1.3

```
1 n_iters = 1000
2 can_sd = 0.05
3 α = β = 1
4 θ = 0.5
5 trace = {"θ":np.zeros(n_iters)}
6 p2 = post(θ, Y, α, β)
7
8 for iter in range(n_iters):
9   θ_can = stats.norm(θ, can_sd).rvs(1)
10  p1 = post(θ_can, Y, α, β)
11  pa = p1 / p2
12
13  if pa > stats.uniform(0, 1).rvs(1):
14      θ=θ_can
15      p2 = p1
16
17  trace["θ"][iter] =θ
```

在代码清单 1.3 的第 9 行，从标准差为 can_sd 的正态分布中采样以生成提议分布。第 10 行在新生成的参数值 θ_can 处估计后验，第 11 行计算接受概率。第 17 行在 trace 数组中保存了 θ 的值。该值是一个新值还是重复上一次的值，取决于第 13 行的比较结果。

模糊的 MCMC 术语

当使用 MCMC 方法进行贝叶斯推断时，通常将其称为 MCMC 采样器。在每次迭代中，从采样器中抽取一个随机样本，因此很自然地将 MCMC 的输出结果称为样本(sample)或抽取(draw)。有人将样本视为由一组抽取组成，而有人则认为两个概念可以互换。

由于 MCMC 是按迭代顺序抽取样本的，因此也会说：我们得到了一个采样结果的链(chain)，或者简称为 MCMC 链。为进行计算和诊断，通常需要抽取许多链(详见第 2 章)。所有输出的链，无论是单条还是多条，通常都称为轨迹、迹(trace)或直接称为后验。无论怎么定义这些名词，口语中总是不精确的。如果需要精确，最好的方法还是查看代码以准确了解具体内容。

注意，代码清单 1.3 中的实现方式并非旨在提高效率，实际上生产级代码中会出现许多优化，例如在对数级别上计算概率，以避免过低/过高的对数概率转换(见 10.4.1 节)，或预先计算提议分布和均匀分布值。进行这些优化都需要调整抽象数学理论以适应计算机实现，而构建推断引擎的原因最好由专家解释。同样，can_sd 的值是 Metropolis-Hastings 算法的参数，而不是贝叶斯模型的参数。理论上该参数不应该影响算法的正确行为，但在实践中它又非常重要，因为方法的效率肯定会受到该值的影响(详见 11.9 节)。

回到示例，现在有了 MCMC 样本，我们想了解其形态。检查贝叶斯推断结果的一种常用方法是：将每次迭代得到的采样值通过直方图或其他可视化工具绘制出来，以表示分布。例如，可以使用代码清单 1.4 中的代码绘制图 1.2[1]。

代码清单 1.4

```
1 _, axes = plt.subplots(1,2, sharey=True)
2 axes[1].hist(trace["θ"], color="0.5", orientation="horizontal", density=True)
```

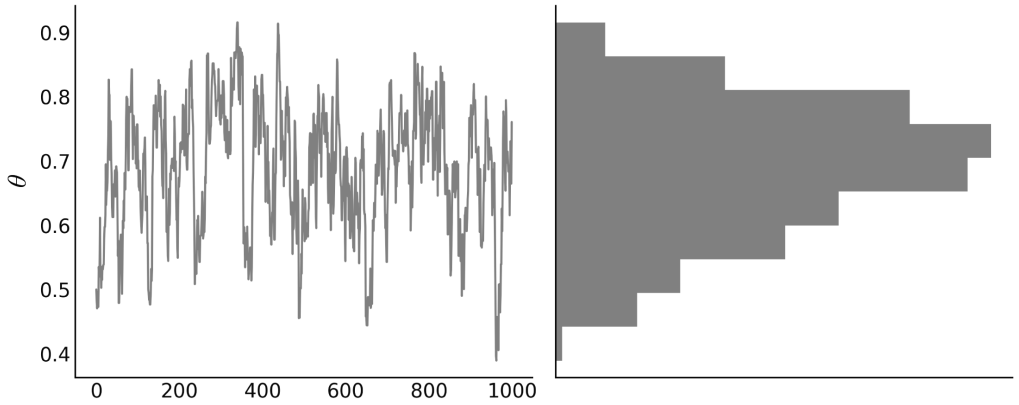

图 1.2 左图为每次迭代产生的参数 θ 的采样值。右图为 θ 采样值的直方图。该直方图经过了旋转，以便更容易看出两幅图之间的密切关系。左图显示了采样值的序列，该序列其实就是马尔可夫链，而右图则显示了采样值的分布情况

通常，计算一些数字汇总信息也很有用。我们将使用名为 ArviZ 的 Python 软件库[91]计算这些统计信息，如代码清单 1.5 所示。

代码清单 1.5

```
az.summary(trace, kind="stats", round_to=2))
```

相关统计信息如表 1.1 所示。

表 1.1 代码清单 1.5 得出的统计信息

	平均值	标准差	hdi_3%	hdi_97%
θ	0.69	0.01	0.51	0.87

ArviZ 的 summary 函数计算参数 θ 的平均值、标准差和 94%最高密度区间(Highest Density

1 可以使用 ArviZ 的 plot_trace 函数获得类似的绘图。本书其余部分将使用这一方法。

Interval，HDI)。HDI 是包含给定概率密度(此例为 94%)的最短区间[1]。图 1.3 是由 az.plot_posterior (trace)生成的，与图 1.2 中的汇总信息非常相似。可以在代表整个后验分布的曲线顶部看到平均值和 HDI。该曲线使用**核密度估计器(Kernel Density Estimator, KDE)**计算，其类似于直方图的平滑版本。ArviZ 在其许多绘图函数中都使用 KDE，甚至用于在内部进行一些计算。

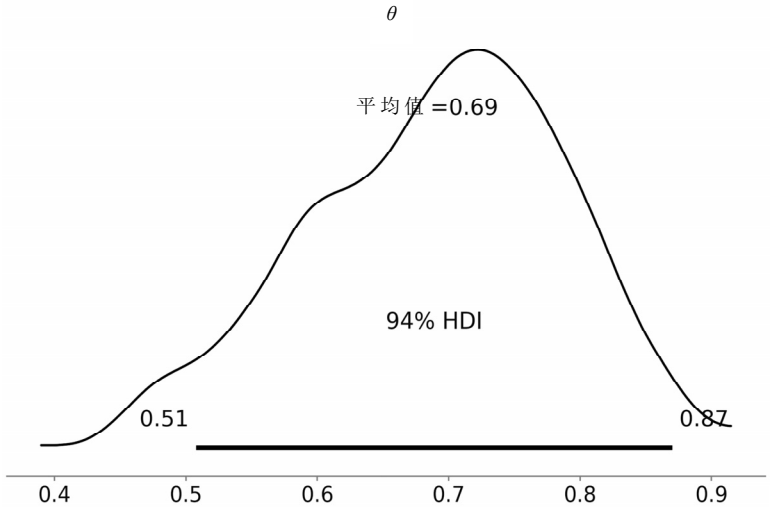

图 1.3　此后验图对代码清单 1.3 生成的样本进行可视化。后验分布使用 KDE 表示，平均值和 94% HDI 均在图中有所展示

HDI 常用于贝叶斯统计中，像 50%或 95%这样的整数边界值也很常见。但 ArviZ 的默认值为 94%(或 0.94)，如表 1.1 和图 1.3 所示。这样选择的原因是，94 接近广泛使用的 95，而同时这种微小的区别可以提醒受众，边界值为整数并没有特殊之处[101]。理想情况下，应该选择一个满足需要的值[92]，或者至少表明你使用的是默认值。

1.3　支持自动推断，反对自动建模

我们可以在概率编程语言(Probabilistic Programming Languages, PPL)的帮助下定义和推断模型，而不是编写自己的采样器并使用 scipy.stats 方法定义自己的模型。概率编程语言允许用户使用代码表达贝叶斯模型，然后借助通用推断引擎以自动化的方式执行贝叶斯推断。简而言之，概率编程语言能够帮助贝叶斯从业者更专注于模型构建，而不是数学和计算细节。在过去的几十年中，此类工具的可用性大大提升了贝叶斯方法的普及度和实用性。遗憾的是，这些通用推断引擎方法并不是真正通用的(但名称仍然很酷)，因为它们无法有效地求解所有贝叶斯模型。现代贝叶斯从业者职责的一部分是理解这些局限性并给出解决方法。

在本书中，我们使用的概率编程语言是 PyMC3[138]和 TensorFlow Probability(TFP)[47]。例如用 PyMC3 为式(1.10)编写模型，如代码清单 1.6 所示。

1 注意，原则上，包含给定比例总密度的区间的数量是无限的。

代码清单 1.6

```
# 在 PyMC3 中声明模型
with pm.Model() as model:
  # 指定未知参数的先验分布
  θ= pm.Beta("θ", alpha=1, beta=1)

  # 指定似然分布并以观测数据为条件
  y_obs = pm.Binomial("y_obs", n=1, p=θ, observed=Y)

  # 从后验分布中取采样
  idata = pm.sample(1000, return_inferencedata=True)
```

你可以自己检查这段代码的结果是否与之前使用的自制采样器的结果一致，并且工作量要少得多。如果不熟悉 PyMC3 语法，现阶段只需要关注代码注释中表明的每一行的意图。用 PyMC3 语法定义模型后，可以利用 pm.model_to_graphviz(model)在代码清单 1.6 中生成模型的概率图表征(见图 1.4)。

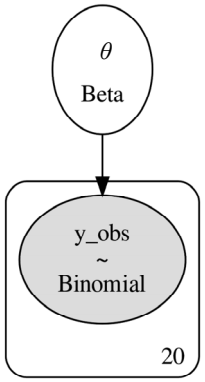

图 1.4 式(1.10)和代码清单 1.6 中定义的贝叶斯模型的概率图表征。椭圆代表先验和似然，而 20 表示观测次数

概率编程语言不仅可以计算随机变量的对数概率以获得后验分布，还可以模拟前面提到的两种预测分布：先验预测分布和后验预测分布。例如，代码清单 1.7 展示了如何使用 PyMC3 分别获得先验预测分布以及后验预测分布的 1000 个样本。注意，第一个函数仅有 model 参数，而第二个函数必须同时传递 model 和 trace 参数，这反映了先验预测分布仅由模型计算，而后验预测分布不仅需要模型还需要后验分布。两种预测分布的样本分别在绘制在图 1.5 的上下图中。

代码清单 1.7

```
pred_dists = (pm.sample_prior_predictive(1000, model)["y_obs"],
              pm.sample_posterior_predictive(idata, 1000, model)["y_obs"])
```

式(1.1)、式(1.7)和式(1.8)清楚地将后验分布、先验预测分布和后验预测分布定义为 3 个不同的数学对象。显然后面两个是数据的概率分布，而第一个是参数的概率分布。图 1.5 可视化了这种区别，图中还包含了先验分布，从而更加完整。

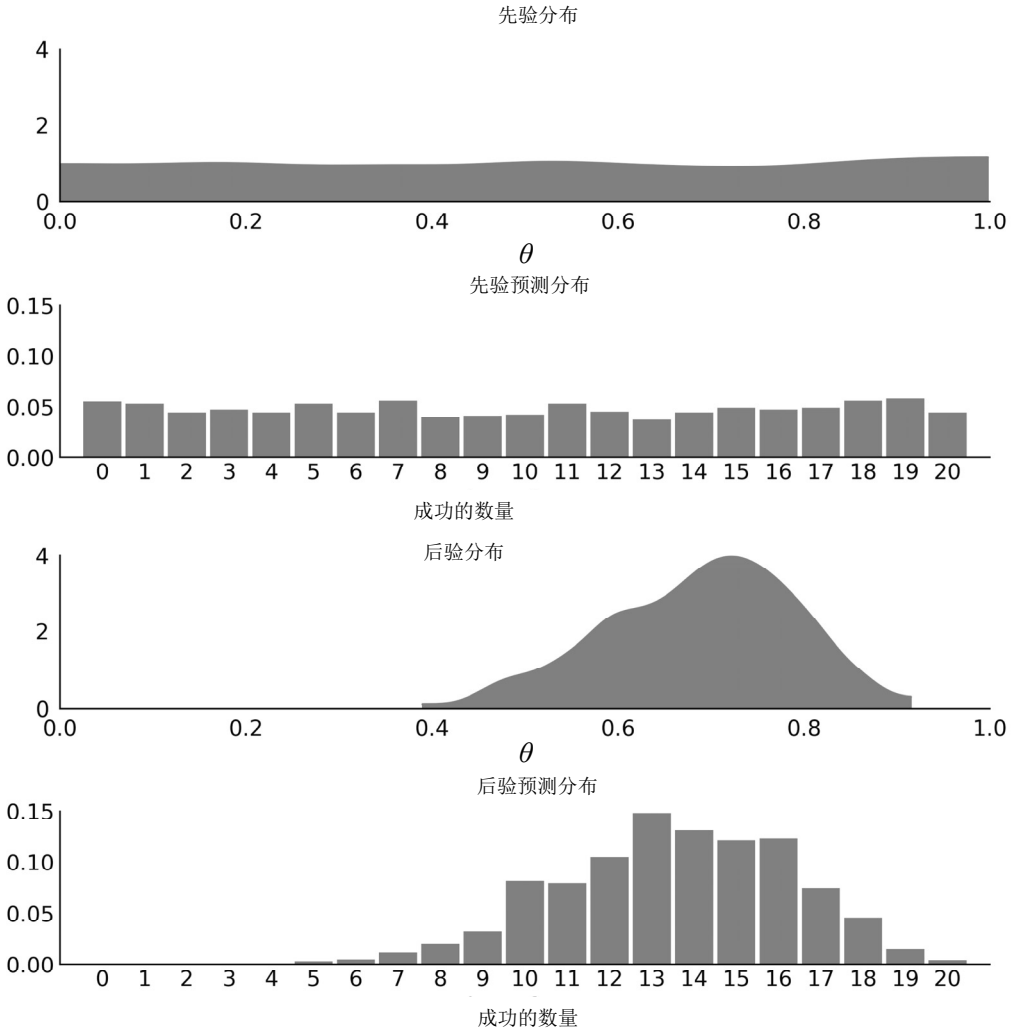

图 1.5　自上至下，展示了：(1)参数 θ 的先验分布样本；(2)先验预测分布样本，绘制成功的数量的概率分布；(3)参数 θ 的后验分布样本；(4)成功的数量的后验预测分布。在第一幅和第三幅图、第二幅和第四幅图之间分别共享 x 轴和 y 轴的坐标尺度

几种统计模型表达方式

有许多方法可以表示统计模型的架构，现列示如下(没有特定的顺序):

- 口语和书面语言。
- 概念图，如图 1.4。
- 数学符号，如式(1.10)。
- 代码，如代码清单 1.6。

对于现代贝叶斯从业者来说，所有这些方法都很有用。常见于演讲、科学论文、与同事讨论时的手绘草图、互联网上的代码示例中。熟练地使用这些方法，能够更好地理解以某种形式

呈现的概念，然后以其他形式进行应用。例如，阅读一篇论文然后实现一个模型，或者在演讲中听到一种技术，然后能够为其写一篇博客。对个人而言，熟练掌握这些会加快学习速度并提高与他人交流的能力。最终，这有助于实现统计界一直追求的目标——对世界的共识。

如前所述，后验预测分布考虑了估计结果的不确定性。图 1.6 表明：根据后验平均值计算得到的预测结果，比后验预测分布中的预测结果范围更窄。此现象不仅对平均值有效，对于其他任何点估计，都会得到类似的图。

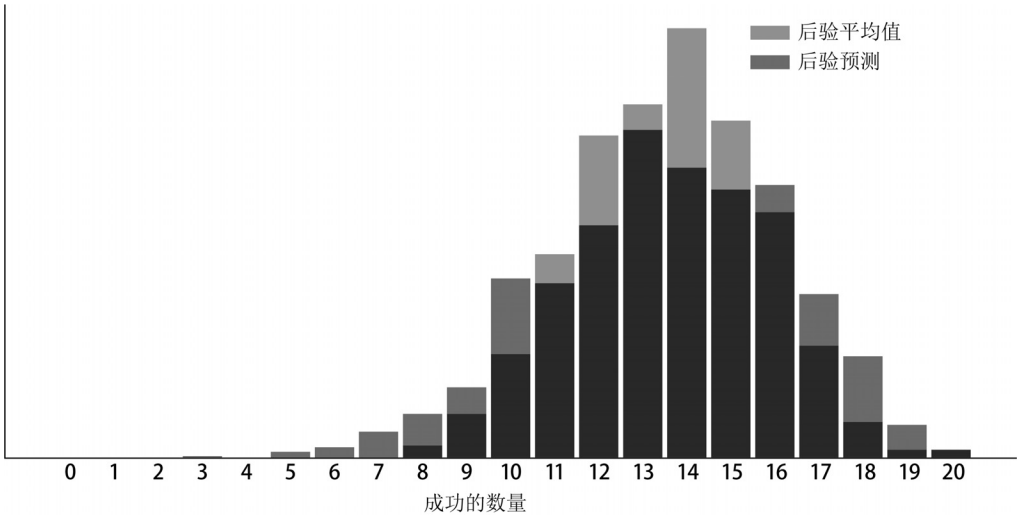

图 1.6　贝塔二项式模型的预测结果对比，使用后验平均值所做的预测表示(灰色直方图)，使用完整后验(即后验预测分布)所做的预测表示(蓝色直方图)

1.4　量化先验信息的方法

在贝叶斯统计中，选择先验分布，这有利也有弊，而我们认为这是必要的。如果没有选择先验分布，那么最大可能是别人已经代为完成。当然，让别人替你做决定并不总是坏事。如果在正确的场景中应用并且意识到其局限性，许多非贝叶斯方法会非常有用和有效。然而，我们坚信：了解模型假设并能够灵活调整这些假设是从业者的优势，而先验即为一种假设。

我们也明白，对于许多从业者来说，先验选择可能导致怀疑、焦虑甚至沮丧等情绪，对于新手来说更是如此。寻找给定问题的最佳先验，是一个常见且有效的问题。但是除了“没有最佳先验”这个结论，很难给出令人满意的答案。好在有一些默认值，可以作为迭代建模工作流的起点。

本节将讨论一些用于选择先验的一般性方法。此讨论所涵盖的信息是递增的，从不包含任何信息的“空白”先验到信息丰富的先验。本章关于先验的讨论更多是从理论角度出发。在后续章节中，将讨论如何在更实际的环境中选择先验。

1.4.1　共轭先验

如果后验与先验属于同一分布族，则先验与似然共轭。例如，如果似然呈泊松分布并且先验呈伽马分布，那么后验也呈伽马分布。

从纯数学角度来看，**共轭先验**是最有利的选择，因为我们仅用纸和笔就可以解析计算后验分布，不需要复杂的计算[1]。但从现代计算角度来看，共轭先验通常并不优于其他方法，主要原因是现代计算方法允许使用几乎任何先验进行推断，而不仅仅是便于数学计算的有限选择。尽管如此，在学习贝叶斯推断时，以及在某些需要对后验使用解析表达式的情况下，共轭先验仍然非常有用(参阅 10.2.2 节的示例)。因此，使用贝塔二项式模型简要讨论解析形式的共轭先验。

顾名思义，该模型的似然为二项分布，而其共轭先验呈贝塔分布：

$$p(\theta \mid Y) \propto \overbrace{\frac{N!}{y!(N-y)!}\theta^{y}(1-\theta)^{N-y}}^{\text{二项似然}} \overbrace{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}}^{\text{贝塔先验}} \tag{1.11}$$

式中所有不包含 θ 的项都为常数，可以省略它们，进而得到：

$$p(\theta \mid Y) \propto \overbrace{\theta^{y}(1-\theta)^{N-y}}^{\text{二项似然}} \overbrace{\theta^{\alpha-1}(1-\theta)^{\beta-1}}^{\text{贝塔先验}} \tag{1.12}$$

重新组织公式，得到：

$$p(\theta \mid Y) \propto \theta^{\alpha-1+y}(1-\theta)^{\beta-1+N-y} \tag{1.13}$$

如果想确保后验是一个正确的概率分布函数，还需要添加一个归一化常数，以确保 PDF 的积分为 1(参见 11.1.5 节)。注意，式(1.13)看起来与贝塔分布的核一致，由此，在添加贝塔分布的归一化常数后，得出贝塔二项式模型的后验分布为：

$$p(\theta \mid Y) \propto \frac{\Gamma(\alpha_{\text{post}}+\beta_{\text{post}})}{\Gamma(\alpha_{\text{post}})\Gamma(\beta_{\text{post}})}\theta^{\alpha_{\text{post}}-1}(1-\theta)^{\beta_{\text{post}}-1} = \text{Beta}(\alpha_{\text{post}}, \beta_{\text{post}}) \tag{1.14}$$

其中，$\alpha_{\text{post}}=\alpha+y$，$\beta_{\text{post}}=\beta+N-y$。

贝塔二项式模型的后验也呈贝塔分布，因此可以使用贝塔后验作为下一步贝叶斯分析的先验。这意味着，一次使用完整的数据集和一次更新一个数据点将获得相同的结果。例如，图 1.7 中的前 4 个子图显示了从 0 次到 1 次、2 次和 3 次试验时，不同的先验更新情况。遵循此顺序，或者直接从 0 次试验跳到 3 次试验(或 n 次试验)，最终会得到一致的结果。

1 你头脑中的想法除外。

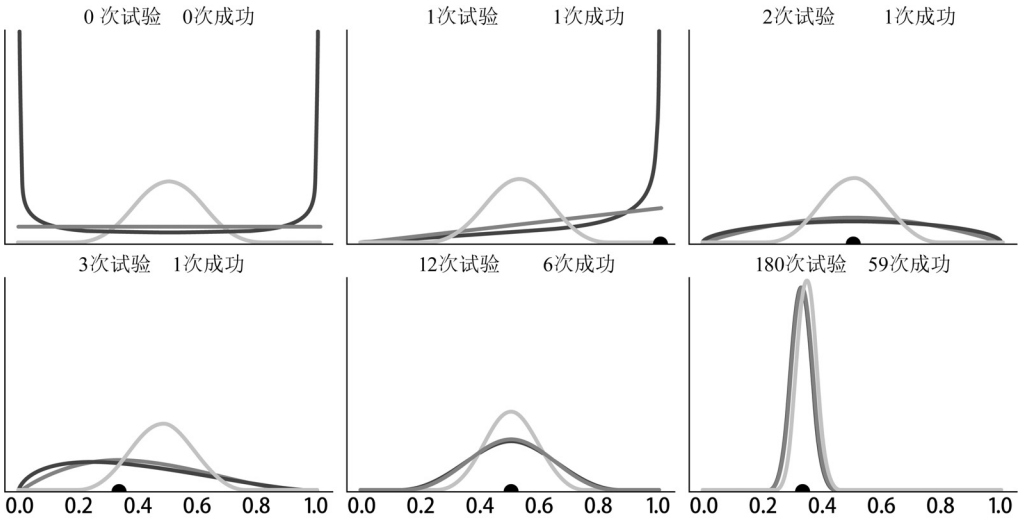

图 1.7 从 3 种不同的先验开始连续更新先验并增加试验次数(可能还有成功的次数)。

黑点代表根据样本得到的成功比例估计值 $\hat{\theta}=\frac{y}{n}$

从图 1.7 中还可以得到其他结论。例如，随着试验次数增加，后验的宽度越来越小，即不确定性越来越低。子图 3 和子图 5 分别显示了“2 次试验 1 次成功”和“12 次试验 6 次成功”的结果。两种情况根据样本得到的成功比例估计值 $\hat{\theta}=\frac{y}{n}$ (黑点)相同，均为 0.5(后验分布的众数也是 0.5)，不过子图 5 中的后验相对更加集中，反映出观测数量更大，不确定性更低。最后可以观察到：随着观测次数增加，不同先验最终可以收敛到同样的后验分布。在无限数据的条件下，后验与先验的选择无关，根据不同的先验推断得到的后验，最终将在点 $\hat{\theta}=\frac{y}{n}$ 处具有几乎所有密度，如代码清单 1.8 所示。

代码清单 1.8

```
1 _, axes = plt.subplots(2,3, sharey=True, sharex=True)
2 axes = np.ravel(axes)
3
4 n_trials = [0, 1, 2, 3, 12, 180]
5 success = [0, 1, 1, 1, 6, 59]
6 data = zip(n_trials, success)
7
8 beta_params = [(0.5, 0.5), (1, 1), (10, 10)]
9 θ= np.linspace(0, 1, 1500)
10 for idx, (N, y) in enumerate(data):
11     s_n = ("s" if (N > 1) else "")
12     for jdx, (a_prior, b_prior) in enumerate(beta_params):
13         p_theta_given_y = stats.beta.pdf(θ, a_prior + y, b_prior + N - y)
14
15         axes[idx].plot(θ, p_theta_given_y, lw=4, color=viridish[jdx])
16         axes[idx].set_yticks([])
17         axes[idx].set_ylim(0, 12)
18         axes[idx].plot(np.divide(y, N), 0, color="k", marker="o", ms=12)
19         axes[idx].set_title(f"{N:4d} trial{s_n} {y:4d} success")
```

贝塔分布的平均值为$\frac{\alpha}{\alpha+\beta}$，因此先验平均值为：

$$\mathbb{E}[\theta]=\frac{\alpha}{\alpha+\beta} \tag{1.15}$$

后验平均值为：

$$\mathbb{E}[\theta \mid Y]=\frac{\alpha+y}{\alpha+\beta+n} \tag{1.16}$$

可以看到，如果n的值相对于α和β的值较小，那么后验平均值更接近于先验平均值。也就是说，先验对结果的贡献大于数据。如果n的值相对于α和β的值较大，则后验平均值将更接近成功比例的估计值$\hat{\theta}=\frac{y}{n}$，实际上在$n\to\infty$的情况下，后验平均值将与α和β的先验无关，最终都会完美地匹配根据样本得到的成功比例。

对于贝塔二项式模型，后验众数为：

$$\underset{\theta}{\operatorname{argmax}}[\theta \mid Y]=\frac{\alpha+y-1}{\alpha+\beta+n-2} \tag{1.17}$$

可以看到，当先验为Beta(α=1，β=1，即均匀分布)时，后验众数在数值上等于根据样本计算的成功比例的估计值$\hat{\theta}=\frac{y}{n}$。后验众数通常被称为**最大后验(Maximum a Posteriori，MAP)**值。此结果并非贝塔二项式模型独有。事实上，许多非贝叶斯方法的结果都可以理解为贝叶斯方法在特定先验条件下的MAP[1]。

将式(1.16)与样本得到的成功比例$\frac{y}{n}$进行比较。贝叶斯估计器将成功次数增加了α，将试验次数增加了$\alpha+\beta$。这使β成为失败的次数。从此意义上说，可以将先验参数视为伪计数，或者作为先验数据。先验Beta(1,1)等价于进行两次试验，一次成功，一次失败。从概念上讲，贝塔分布的形状由参数α和β控制，观测数据会更新先验，从而使贝塔分布的形状更接近且更窄地移向大多数观测值。对于$\alpha<1$和/或$\beta<1$的值，先验的解释变得有点奇怪，因为字面解释会得到先验Beta(0.5,0.5)对应于一次试验中半次失败，半次成功，或者可能是一次结果未定的试验。

1.4.2　客观先验

在没有先验信息的情况下，遵循无差别原则(也称为理由不充分原则)听起来似乎更合理。此原则基本上是说：如果没有关于某个问题的信息，就没有任何理由相信一个结果会比任何其他结果更有可能发生。在贝叶斯统计背景下，这一原则推动了**客观先验(Objective Priors)**的研究和应用。这是一种“生成对给定分析影响最小的先验”的系统方法。有些统计学者偏爱客观先验，因为他们认为此类先验消除了先验选择的主观性。当然，并没有消除其他来源的主观性，如似然的选择、数据的选择、建模或研究问题的选择等。

1 例如，具有L2正则化的正则化线性回归与在系数上使用高斯先验相同。

一种获得客观先验的方法是著名的 Jeffreys 先验(Jeffreys' Prior, JP)。此类先验虽然总是以某种方式提供了信息，但通常被称为无信息先验。Jeffreys 先验的特点是，在重参数化(reparametrization)时保持不变(**重参数化**指以形式不同但在数学上等效的方式重写表达式)。下面通过实例具体说明。假设 Alice 具有未知参数为 θ 的二项似然，她选择了某种先验并计算得到后验。而她的朋友 Bob，也对同一问题感兴趣，但他并不关心成功次数 θ，而是关注另外一个参数：成功的**赔率**(odds)，即参数 k，$k=\frac{\theta}{1-\theta}$。此时 Bob 有两种选择：一是使用 Alice 在 θ 上的后验计算 k[1]，二是在 k 上选择先验来自己计算后验。如果 Alice 模型和 Bob 模型都为各自的参数设置了 Jeffreys 先验，那么无论 Bob 用哪种方法计算后验，最终都会得到相同的结果。也就是说，最终的后验推断结果对于所选择的参数而言具有不变性。此解释的一个推论是，只有使用 Jeffreys 先验才能保证模型的两种或多种参数化必然得到一致的后验。

对于一维 θ 的情况，Jeffreys 先验为：

$$p(\theta) \propto \sqrt{I(\theta)} \tag{1.18}$$

其中，$I(\theta)$为期望的 Fisher 信息：

$$I(\theta) = -\mathbb{E}_{\mathbb{Y}}\left[\frac{d^2}{d\theta^2}\log p(Y \mid \theta)\right] \tag{1.19}$$

一旦从业者确定了似然函数 $p(Y \mid \theta)$，Jeffreys 先验就自动被确定。也就是说，省去了对先验选择的任何讨论。除非高层管理人员从一开始就否定了你选择的 Jeffreys 先验。

有关 Alice 和 Bob 问题的 Jeffreys 先验详细推导，参阅 11.6 节。Alice 的 Jeffreys 先验为：

$$p(\theta) \propto \theta^{-0.5}(1-\theta)^{-0.5} \tag{1.20}$$

这就是 Beta(0.5,0.5)分布的核，是一个 U 形分布，如图 1.8 的左上子图所示。

而对于 Bob 来说，Jeffreys 先验为：

$$p(\kappa) \propto \kappa^{-0.5}(1+\kappa)^{-1} \tag{1.21}$$

这是一个半 U 形分布，定义在[0, ∞]区间，如图 1.8 的右上子图所示。称其为半 U 形似乎有些奇怪，但实际上它是 Beta-prime 分布(贝塔分布的一个分支)在参数 $\alpha=\beta=0.5$ 时的核。

注意，式(1.19)中的期望是关于观测 $Y|\theta$ 的，这是在样本空间上的期望。这意味着为了获得 Jeffreys 先验，需要对所有可能的试验结果求平均值，这违反了似然原则，因为关于 θ 的推断不仅取决于当前数据，还取决于可能(但尚未)观测到的数据集。

Jeffreys 先验可以是不恰当的先验，即它的积分可能不为 1。例如，已知方差的高斯分布，其平均值参数的 Jeffreys 先验在整个实数轴上均匀分布。但只要能够验证，这些不恰当的先验与似然组合后，能够产生恰当(即积分为 1)的后验分布，则这些不恰当的先验就是可用的。另外还要注意，我们不能从不恰当的先验中抽取随机样本(即此类先验是非生成式的)，否则会造成许多模型推断工具失效。

1 例如，如果有来自后验的样本，那么可以将这些 θ 样本插入 $k=\frac{\theta}{1-\theta}$。

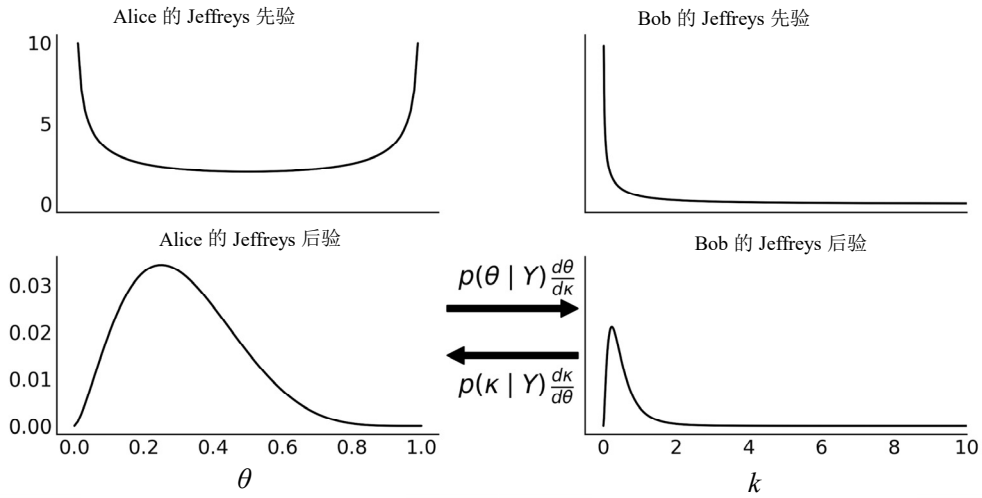

图 1.8　上图：根据成功次数 θ(左)或成功赔率 k(右)参数化的二项似然的 Jeffreys 先验(未归一化)。下图：根据成功次数 θ(左)或成功赔率 k(右)参数化的二项似然的 Jeffreys 后验(未归一化)。后验之间的箭头表示，通过调整一些变量规则，两个后验之间可相互转换(详细信息参阅 11.1.9 节)

Jeffreys 先验并不是获得客观先验的唯一方法。另一种途径是通过最大化先验和后验之间 KL 散度的期望来获得先验(参见 11.3 节)。此类先验被称为 Bernardo 参考先验。之所以是客观先验，是因为它们允许数据将最大量的信息带入后验分布。另外，Bernardo 参考先验和 Jeffreys 先验不必一致。此外，对于某些复杂模型，可能不存在客观先验或难以推导出客观先验。

1.4.3　最大熵先验

另一种证明先验选择合理性的方法是选择具有最大熵的先验。如果可取值为任意值，那么此先验就在可取值范围内呈均匀分布[1]。但若可取值有限制呢？例如，我们可能知道参数必须位于[0,∞]区间，那么能得到既有最大熵又能满足约束的先验吗？是的，可以，这正是最大熵先验的原理。在文献中谈论最大熵原理时，通常会提到 MaxEnt 这个词。

为了获得最大熵先验，需要求解包含一组约束条件的优化问题。在数学上，这可以使用拉格朗日乘数实现。但这里不会采用形式化方法推导和证明最大熵先验，而将通过几个代码案例获得灵感。

图 1.9 展示了通过最大化熵获得的 3 个分布。紫色分布是在没有约束条件下获得的，这确实是 11.2 节中讨论熵时所预期的均匀分布。如果我们对问题一无所知，那么所有事件都是同等先验的。第二个青色分布是在知道分布平均值(此例为 1.5)的约束下获得的，这是一个类似指数的分布。最后一个黄绿色的分布是在已知出现 3 和 4 的概率为 0.8 的约束下获得的。注意：如果检查代码清单 1.9，会看到所有的分布都是在两个约束条件下计算的，一是概率只能在[0,1]区间中取值，二是总概率必须为 1。它们是有效概率分布的共同约束，因此可以被视为内在的或本体论约束。为此，经常称图 1.9 中的紫色分布是在无约束条件下获得的。

1 详见 11.2 节。

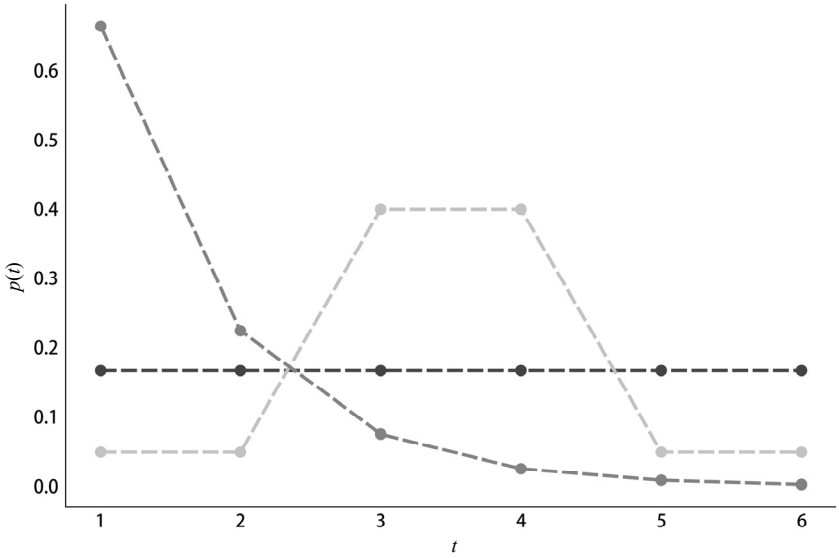

图 1.9 在不同约束下通过最大化熵获得的离散分布。我们使用了 scipy.stats 的 entropy 函数来估计这些分布。注意，在添加约束后分布发生了较大变化

代码清单 1.9

```
cons = [[{"type": "eq", "fun": lambda x: np.sum(x) - 1}],
        [{"type": "eq", "fun": lambda x: np.sum(x) - 1},
        {"type": "eq", "fun": lambda x: 1.5 - np.sum(x * np.arange(1, 7))}],
        [{"type": "eq", "fun": lambda x: np.sum(x) - 1},
        {"type": "eq", "fun": lambda x: np.sum(x[[2, 3]]) - 0.8}]]

max_ent = []
for i, c in enumerate(cons):
    val = minimize(lambda x: -entropy(x), x0=[1/6]*6, bounds=[(0., 1.)] * 6,
                   constraints=c)['x']
   max_ent.append(entropy(val))
   plt.plot(np.arange(1, 7), val, 'o--', color=viridish[i], lw=2.5)
plt.xlabel("$t$")
plt.ylabel("$p(t)$")
```

我们可以将最大熵原理理解为在给定约束下选择最平坦分布(可延伸为最平坦先验分布)的过程。在图 1.9 中，均匀分布是最平坦分布，但注意，一旦引入“3 和 4 的出现概率为 80%”的约束，黄绿色分布就变成了最平坦的分布。注意：要想使 3 和 4 的出现概率之和为 0.8，有很多种选择，如 0 + 0.8、0.7 + 0.1、0.312 + 0.488 等，但本例中二者概率都为 0.4。同理，值 1、2、5 和 6 也有类似情况，它们的总概率值为 0.2，而本例中也采用了均匀分布(每个值的概率均为 0.05)。现在观察类似指数的曲线，它看起来肯定不是很平坦，但应注意到，其他选择会更不平坦且更集中。例如，分别以 50%的概率获得 1 和 2(因此 3 至 6 的概率不变)，这也满足平均值为 1.5 的约束，但更不平坦，如代码清单 1.10 所示。

代码清单 1.10

```
ite = 100_000
entropies = np.zeros((3, ite))
for idx in range(ite):
    rnds = np.zeros(6)
    total = 0
    x_ = np.random.choice(np.arange(1, 7), size=6, replace=False)
    for i in x_[:-1]:
      rnd = np.random.uniform(0, 1-total)
       rnds[i-1] = rnd
       total = rnds.sum()
    rnds[-1] = 1 - rnds[:-1].sum()
    H = entropy(rnds)
    entropies[0, idx] = H
    if abs(1.5 - np.sum(rnds * x_)) < 0.01:
         entropies[1, idx] = H
    prob_34 = sum(rnds[np.argwhere((x_ == 3) | (x_ == 4)).ravel()])
    if abs(0.8 - prob_34) < 0.01:
        entropies[2, idx] = H
```

图 1.10 显示了在满足与图 1.9 中 3 个分布完全相同的约束时，随机生成样本的熵的分布。垂直虚线表示图 1.10 中曲线的熵。虽然并未证明，但试验似乎表明没有分布会比图 1.10 中的分布具有更高的熵，这与理论完全一致。

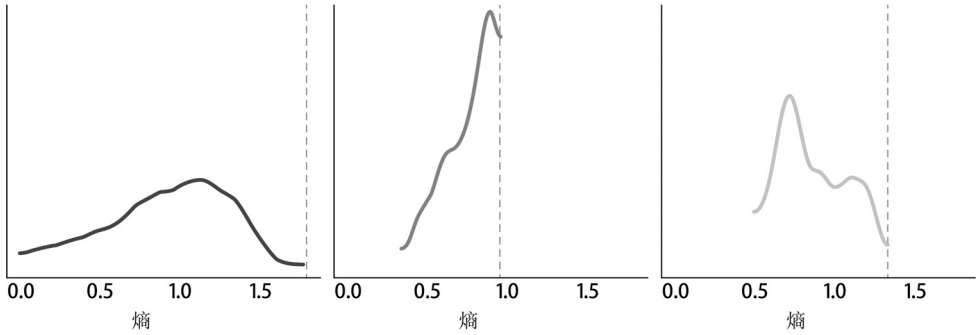

图 1.10　一组随机生成的分布的熵的分布。垂直虚线表示具有最大熵分布的值，使用代码清单 1.9 计算。可以看到，没有一个随机生成的分布的熵大于具有最大熵分布的熵，尽管这不是正式证明，但结果足够令人信服

下面给出相应约束下，对应的最大熵分布。[1]

- 无约束时：均匀分布(连续的或离散的，取决于变量类型)。
- 具有正平均值，值域为[0,∞]时：指数分布。
- 具有绝对值平均值，值域为(－∞, ∞)时：拉普拉斯(Laplace，也称为双指数)分布。
- 具有指定的平均值和方差，值域为(－∞, ∞)时：正态分布。
- 具有指定的平均值和方差，值域为[－π, π]时：冯・米塞斯(Von Mises)分布。
- 只有两个无序值，且平均值为常数：二项分布，或者泊松(Poission)分布(泊松可以看作特殊的二项分布)。

值得注意的是，考虑到模型约束的情况下，许多广义线性模型(如第 3 章中的模型)仍使用最大熵分布来定义。与客观先验类似，MaxEnt 先验有可能并不存在或难以推导。

1 详见 https://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution#Other_examples。

1.4.4 弱信息先验与正则化先验

在前几节中，使用一般性的过程生成了无信息先验，旨在不将太多信息纳入分析中。这些过程甚至还提供了“以某种方式”自动生成先验的方法。这两个特征听起来非常有吸引力，而且实际上被大量贝叶斯从业者和理论家所采用。

但在本书中，不会过分依赖这些先验。我们认为先验的选择与其他建模选择一样，应该取决于上下文。这意味着特定问题的细节、给定科学领域的特性等，都可以为先验的选择提供信息。虽然 MaxEnt 先验能够包含其中一些约束，但似乎还可以更加靠近信息先验频谱的信息端。用弱信息先验可以实现这一点。

构造弱信息先验的方法通常没有 Jeffreys 先验或 MaxEnt 先验那样完备的数学定义。相反，弱信息先验更多是经验主义和模型驱动的。也就是说，弱信息先验大多通过结合领域知识和模型本身来定义。对于很多问题，经常有关于参数可取值的信息，这些信息往往来自参数的物理意义，例如身高必须是正数。我们甚至可以从之前的试验或观测中得到取值范围。我们或许可以充分证明一个值应该接近 0 或高于某个预定义的下限。上述这些信息都能够为选择先验提供弱信息，同时保持一定程度的未知。

这里再次使用贝塔二项式模型作为示例，图 1.11 显示了 4 个可选的先验。前两个分别是 Jeffreys 先验和最大熵先验；第三个是弱信息先验，它优先考虑 $\theta = 0.5$，同时对其他值保持宽泛或相对模糊；最后一个是信息先验，以 $\theta = 0.8$ 为中心[1]。如果从理论、之前的试验、观测数据中能够获得高质量的信息，则信息先验是最有效的选择。信息先验可以传达大量信息，因此相较于其他先验通常需要更有力的理由。正如 Carl Sagan 说的“非凡主张需要非凡的证据”[45]。重要的是：先验的信息量取决于模型及其上下文。某个先验可能在某种情境中无信息，但在另一个情境中的信息量很大[63]。例如，如果以米为单位对成年人的平均身高进行建模，则 $N(2, 1)$的先验被认为是无信息的；但如果用于估计长颈鹿的高度，则该先验涵盖的信息量非常大。因为现实中长颈鹿的高度与人类身高相差很大。

弱信息先验可以将后验分布保持在一定的合理范围内，所以它们也被称为正则化先验。正则化是一种添加信息的过程，目的是解决病态问题或减少过拟合的概率，先验提供了一种执行正则化的规范方法。

在本书中，经常使用弱信息先验。有时会在没有太多理由的情况下在模型中使用先验，仅仅是因为示例的重点可能与贝叶斯建模工作流的其他方面有关。但我们也会展示一些使用先验预测检查来校准先验分布的示例。

过拟合(overfitting)

出现过拟合的情况为：某模型生成的预测非常接近用于拟合它的有限数据集，但这样的模型无法拟合新数据且/或不能很好地预测未来的观测。也就是说，模型未能将其预测推广到更广泛的可观测结果。过拟合的反义词是欠拟合，即模型未能充分捕获数据的内在结构。我们会在 2.5 节和 11.4 节进一步讨论。

1 即使这种先验的定义需要的背景信息比已有的先验所需要的更多，我们仍然认为该示例传达了一种有用的理念，这将在后面进一步讨论。

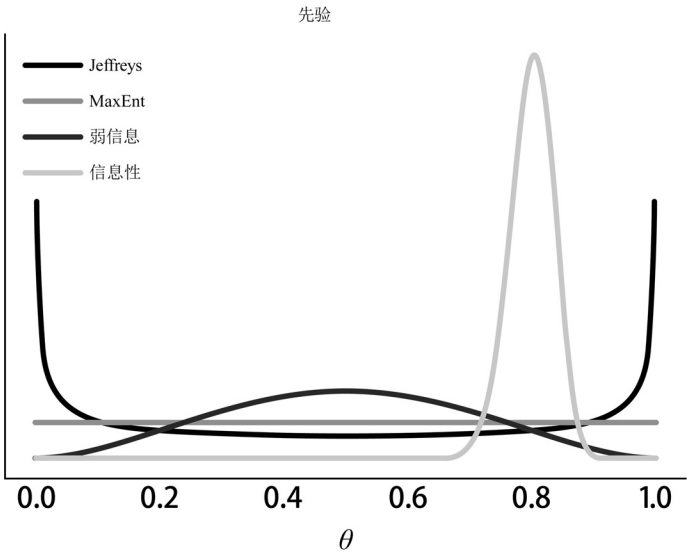

图 1.11　先验信息谱：Jeffrey 先验和 MaxEnt 先验仅为二项似然定义，但弱信息先验和信息先验则不同，二者取决于之前的信息和从业者的建模决策

1.4.5　先验预测分布用于评估先验选择

在评估先验选择时，1.3 节中介绍的先验预测分布是一个方便的工具。通过从先验预测分布中采样，计算机会将在参数空间中的选择转换为在观测变量空间中的样本。考虑观测值通常会比考虑模型参数更直观，这使模型评估更为容易。以贝塔二项式模型为例，先验预测分布并不判断 θ 的特定值是否合理，而是允许判断特定的成功次数是否合理。这对复杂模型更有用，因为需要通过许多数学操作转换参数或是有多个先验相交互。最后，计算先验预测分布可以帮助我们确保模型已经被正确地编码，并且能够用我们的概率编程语言运行，甚至可以帮助调试模型。接下来将介绍更具体的示例，了解如何推断先验预测样本并使用它们选择合理的先验。

1.5　练习

问题按难易程度标记为简单(Easy, E)、中等(Medium, M)和困难(Hard, H)。

1E1. 如前文所述，模型是一个人工表示，用于定义和理解某对象或过程。然而，没有任何模型能够完美地复制它所代表的事物，因此模型在某种程度上是有缺陷的。在本书中，将重点介绍一种特殊模型，即统计模型。你还能想到哪些其他类型的模型？它们如何帮助理解所建模的事物？它们在哪些方面还有不足？

1E2. 按照语言描述，匹配相应的数学表达式：

(a) 给定观测数据的参数概率

(b) 查看任何数据之前的参数分布

(c) 给定参数值的观测数据的合理性

(d) 给定观测数据的不可见观测的概率

(e) 查看任何数据之前，不可见观测的概率

1E3. 下面的表达式中，哪一个对应于“1816 年 7 月 9 日为晴天的概率”？

(a) p(晴天)

(b) p(晴天 | 7 月)

(c) p(晴天 | 1816 年 7 月 9 日)

(d) p(1816 年 7 月 9 日 | 晴天)

(e) p(晴天，1816 年 7 月 9 日)/p(1816 年 7 月 9 日)

1E4. 证明随机选择一个人并选择 Pope 的概率与 Pope 是人类的概率不同。注：在美国系列动画片《飞出个未来》中，(太空)Pope 是一种爬行动物。这如何改变你以前的计算？

1E5. 对于以下情况，绘制可能观测值的分布：

(a) 假设呈泊松分布，光顾当地咖啡馆的人数

(b) 假设呈均匀分布，成年犬的体重(千克)

(c) 假设呈正态分布，成年大象的体重(千克)

(d) 假设呈偏正态分布，成年人类的体重(磅)

1E6. 对于练习 1E5 中的每个示例，使用 Python 中的 SciPy 指定分布。选择你认为合理的参数，随机抽取 1000 个样本，并绘制结果分布。根据你的领域知识判断此分布是否合理？如果不合理，则调整参数并重复该过程，直到合理为止。

1E7. 比较先验 Beta(0.5, 0.5)、Beta(1, 1)和 Beta(1, 4)。先验的形状有何不同？

1E8. 重新运行代码清单 1.8，但要使用你选择的两个 Beta 先验。提示：你可以尝试使用 $\alpha \neq \beta$ 的先验，类似 Beta(2, 5)。

1E9. 尝试提出新的约束条件，以获得新的 Max-Ent 分布(代码清单 1.9)。

1E10. 在代码清单 1.3 中，更改 can_sd 的值并运行 Metropolis-Hastings 采样器。尝试 0.001 和 1 这样的值。

(a) 计算平均值、SD 和高密度区间(HDI)，并将这些值与书中的值进行比较(使用 can_sd = 0.05 计算所得)。估计值有何不同？

(b) 使用函数 az.plot_posterior。

1E11. 你需要估计蓝鲸、人类和老鼠的重量。假设它们呈正态分布，并且为方差设置了相同的先验 $\mathcal{HN}$(200kg)。

对于成年蓝鲸来说，这是什么类型的先验？强信息、弱信息还是无信息？对于老鼠和人类来说，情况如何呢？先验的信息如何对应于我们对这些动物的真实感知？

1E12. 使用代码清单 1.11 所示的函数探索先验(更改参数 a 和 b)和数据(更改头部和试验)的不同组合。总结观察结果。

代码清单 1.11

```
1 def posterior_grid(grid=10, a=1, b=1, heads=6, trials=9):
2     grid = np.linspace(0, 1, grid)
3     prior = stats.beta(a, b).pdf(grid)
```

```
    likelihood = stats.binom.pmf(heads, trials, grid)
    posterior = likelihood * prior
    posterior /= posterior.sum()
    _, ax = plt.subplots(1, 3, sharex=True, figsize=(16, 4))
    ax[0].set_title(f"heads = {heads}\ntrials = {trials}")
    for i, (e, e_n) in enumerate(zip(
            [prior, likelihood, posterior],
            ["prior", "likelihood", "posterior"])):
      ax[i].set_yticks([])
      ax[i].plot(grid, e, "o-", label=e_n)
      ax[i].legend(fontsize=14)


interact(posterior_grid,
     grid=ipyw.IntSlider(min=2, max=100, step=1, value=15),
     a=ipyw.FloatSlider(min=1, max=7, step=1, value=1),
     b=ipyw.FloatSlider(min=1, max=7, step=1, value=1),
     heads=ipyw.IntSlider(min=0, max=20, step=1, value=6),
     trials=ipyw.IntSlider(min=0, max=20, step=1, value=9))
```

1E13. 在先验、先验预测、后验和后验预测分布之间，哪个分布有助于回答以下各个问题？某些问题可能有多个答案。

(a) 在看到任何数据之前，如何看待参数值的分布？

(b) 在看到任何数据之前，我们认为可以看到哪些观测值？

(c) 在使用模型估计参数之后，我们预测接下来会观察到什么？

(d) 哪些参数值解释了调整数据后观察到的数据？

(e) 哪个(些)分布可以用于计算参数的数值汇总信息(如平均值)？

(f) 哪个(些)分布可以用于可视化最高密度区间(HDI)？

1M14. 式(1.1)的分母中包含边际似然，这很难计算。在式(1.3)中，表明知道比例常数的后验值足以进行推断。请说明为什么 Metropolis-Hasting 方法不需要边际似然。提示：用纸和笔完成此练习，尝试展开式(1.9)。

1M15. 在概率模型的以下定义中，确定先验、似然和后验：

$$
\begin{aligned}
Y &\sim \mathcal{N}(\mu, \sigma) \\
\mu &\sim \mathcal{N}(0, 1) \\
\sigma &\sim \mathcal{HN}(1)
\end{aligned}
$$

1M16. 在之前的模型中，后验有多少个参数？将你的答案与式(1.10)中掷硬币问题模型的答案进行比较。

1M17. 假设有两枚硬币；投掷第一枚硬币时，它落在反面与正面的概率同为 1/2。另一枚硬币不均衡，总是落在正面。如果随机选择其中一枚硬币，观察发现其落在正面，那么这枚硬币是不均衡硬币的概率是多少？

1M18. 修改代码清单 1.2，使用你选择的参数从泊松分布生成随机样本。然后修改代码清单 1.1 和代码清单 1.3 以生成 MCMC 样本，估计所选参数。测试样本数、MCMC 迭代次数和初始起点对收敛到你真正选择的参数的影响。

1M19. 假设正在建立一个模型以估计成人身高(厘米)的平均值和标准差。建立一个模型进行上述评估。从代码清单 1.6 开始，根据需要更改似然和先验。在此之后

(a) 从先验预测中取样。生成先验预测分布的可视化和数值汇总。

(b) 使用(a)中的输出来证明你选择的先验和似然的合理性。

1M20. 根据领域知识可得，给定的参数不可能是负值，其平均值大致在 3 到 10 个单位之间，标准差约为 2。使用 Python 确定满足这些约束的两个先验分布。为此，可能需要通过抽取样本并使用绘图和数字汇总来验证是否满足这些标准，以进行反复试验。

1M21. 一家商店在某一天有 n 位顾客光顾。进行购买的顾客数量 Y 分布为 Bin(n，θ)，其中 θ 是顾客进行购买的概率。假设知道 θ 的值，且 n 的先验是 Pois(4.5)。

(a) 使用 PyMC3 计算 $Y \in 0,5,10$ 和 $\theta \in 0.2,0.5$ 的所有组合的 n 的后验分布。使用 az.plot_posterior 在单个图中绘制结果。

(b) 总结 Y 和 θ 对后验的影响。

1H22. 修改代码清单 1.2 以从正态分布生成样本，注意你选择的平均值和标准差参数。然后修改代码清单 1.1 和代码清单 1.3 以从正态模型中取样，检查是否可以得到所选参数。

1H23. 制作一个模型，估计你所在地区晴天与阴天的比例。使用过去 5 天的个人观察数据。仔细考虑数据收集过程。要记住过去的 5 天有多难？如果需要过去 30 天的数据，怎么办？需要过去一年的数据呢？请证明你选择的先验是正确的。获得后验分布，估计晴天与阴天的比例。生成对未来 10 天天气的预测。使用数字汇总和可视化给出你的答案。

1H24. 你种了 12 棵幼苗，有 3 棵发芽了。设 θ 为幼苗发芽的概率。假设 θ 的先验分布为 $\beta(1, 1)$。

(a) 用笔和纸计算后验平均值和标准差。使用 SciPy 验证你的计算。

(b) 使用 SciPy 计算等尾且 HD 为 94%的后验区间。

(c) 使用 SciPy 计算后验预测概率，如果再种植 12 棵幼苗，则至少有 1 棵幼苗会发芽。

使用 SciPy 获得结果后，使用 PyMC3 和 ArviZ 重复此练习。