arXiv:2603.27276v1 [stat.AP] 28 Mar 2026

PyINLA：Python 中潜在高斯模型的快速贝叶斯推断

EsmailAbdulFattah*、EliasKrainski、Håvard Rue

阿卜杜拉国王科技大学（KAUST）CEMSE 部，
沙特阿拉伯麦加图瓦勒，邮编 23955

摘要

贝叶斯推断通常依赖马尔可夫链蒙特卡罗（MCMC）方法，尤其是在处理非高斯数据族时。当面对复杂的层次模型时，如果工作流程需要反复进行模型拟合，或模型维度较大而硬件资源有限，MCMC 方法在计算上可能负担沉重。积分嵌套拉普拉斯近似（INLA）是一种用于非高斯数据潜在高斯模型（LGM）的确定性替代方法，在许多应用场景中能够准确近似后验边缘分布。INLA 方法以独立程序的形式用 C 实现，名为 **inla**，并通过 R 中广泛使用的**INLA** 包调用。本文介绍 PyINLA，这是一个专门的 Python 包，可直接为 **inla**程序提供符合 Python 风格的接口。因此，PyINLA 能够让用户直接在 Python 中指定潜在高斯模型、运行基于 INLA 的推断并获取后验摘要，同时利用成熟的 INLA 实现。我们介绍该包的设计，并通过若干代表性模型展示其用法，包括广义线性混合模型、时间序列预测、疾病制图和地统计预测，说明如何在 Python 中使用 INLA 执行确定性贝叶斯推断，并使其自然融入常见的科学计算工作流程。

关键词：贝叶斯推断、INLA、潜在高斯模型、Python、层次模型、空间统计学

1 引言

贝叶斯层次模型广泛应用于应用统计推断，为纳入先验信息和量化不确定性提供了一个连贯的框架。在包括空间流行病学 [Lawson， 2018], 环境与生态建模[Cressie 和 Wikle、2011], 以及计量经济学 [Koop等领域 2003], 在内的许多应用领域中，贝叶斯建模的实际价值取决于计算工具是否快速、可靠，并易于集成到端到端分析流程中。

*通讯通讯作者：esmail.abdulfattah@kaust.ed u.sa

在当代软件生态系统中，通用贝叶斯推断通常采用马尔可夫链蒙特卡罗（MCMC）或变分推断（VI）进行。这些方法是成熟概率编程框架的基础，但各有众所周知的权衡：对于潜在结构维度较高的模型，MCMC 的计算开销很大，且需要仔细进行收敛性评估；而平均场变分推断会引入近似偏差，并且往往低估后验不确定性 [Brooks 等人，2011年，Blei 等人， 2017]。在实践中，对于具有大型潜在场的层次模型、涉及重复模型拟合的迭代工作流程（敏感性分析、模型选择、交叉验证），以及要求输出确定且可复现的生产环境，这些权衡会变得尤为突出。

积分嵌套拉普拉斯近似（INLA）为一类重要且具有广泛实际应用的模型——潜在高斯模型（LGM）——提供了一种确定性替代方法 [Rue 等人， 2009]。拉普拉斯近似两次利用了潜在场先验分布中的条件结构：一次是在搜索超参数的后验众数时，随后则结合关于超参数的数值积分，进行多次嵌套拉普拉斯近似。在 INLA 算法中，这些嵌套拉普拉斯近似可以替换为隐式低秩修正，正如Niekerk 和 Rue [2024], 所提出的方法；该方法已被证明比原始 INLA 更快，同时具有同等的准确性，Van Niekerk 等人 [2023]。当潜在精度矩阵是稀疏的（高斯马尔可夫随机场即是如此） [Rue 和 Held， 2005] 以及对于广泛的结构化随机效应而言，即使潜在维度很大，INLA 也能高效地计算后验边际分布的精确近似。这些特性使 INLA 成为应用统计学中空间和时空层级模型的有竞争力的方法 [Blangiardo 等人，2013年，Martins 等人，2013年，Heaton 等人， 2019]。

INLA 的参考实现是 R-INLA 项目（https://www.r-inla.org），该项目已发展成为 R 生态系统中成熟且广泛使用的综合框架 [Lindgren 和 Rue，2015年，Bivand 等人，2015年，Van Niekerk 等人， 2021]。多个 R 包扩展了其适用范围，涵盖更广泛的空间模型、非可分时空结构、非线性预测变量以及联合纵向-生存模型 [Bolin 和 Simas，2023年，Krainski 等人，2025年，Bachl 等人，2019年，Lindgren 等人，2024年，Rustand 等人， 2024], , 应用领域包括生物统计学、地球科学、流行病学和环境建模 [Alvares 等人，2024年，Fioravanti 等人，2023年，Myer 和 Johnston，2019年，Seaton 等人， 2024]。

对于使用 Python（如今数据科学和机器学习领域的主流语言）的从业者而言，当他们需要对具有空间、时间或分组结构的层次模型进行快速贝叶斯推断时，选择大多局限于基于马尔可夫链蒙特卡罗的工具，例如**PyMC**或**Stan**。对于包含大型潜在场的模型，这些工具可能计算开销很大，并且需要仔细进行收敛性评估。INLA 为这类模型提供了一种确定性替代方案，但迄今为止一直缺少原生 Python 接口。PyINLA 填补了这一空白：它是一个独立的 Python 包，无需 R 安装，即可将基于 INLA 的推断直接从独立的**inla** C 程序引入以**pandas**、**NumPy**和**SciPy**为基础的标准 Python 工作流程。

本文介绍 PyINLA，这是一个原生 Python 包，可将基于 INLA 的推断直接引入 Python 生态系统。PyINLA 通过 Python 风格 API 提供对已编译 INLA 计算引擎的访问，使用户能够使用标准 Python 数据结构（**pandas** DataFrame、**NumPy** 数组、**SciPy** 稀疏矩阵）指定潜在高斯模型、运行推断并获取完整的后验摘要。该软件包无需安装 R，也不需要跨语言互操作性

层，从而让广大且不断壮大的从业者群体能够使用 INLA；他们在 Python 中构建分析与部署流水线。

本工作的主要贡献包括：

- **原生 Python 接口。** PyINLA 使用标准的 Python 数据结构为潜在高斯模型提供简洁的 API，结果以**pandas** DataFrame 和**NumPy**数组的形式返回，可直接用于后续分析。无需安装 R 或互操作层；编译后的**inla**引擎由该软件包透明地管理。

- **广泛的潜在高斯模型覆盖范围。** 该软件包目前支持广泛使用的 INLA 似然族和潜在模型结构，包括组级随机效应、时间模型（随机游走、自回归过程）以及空间模型（BYM、随机偏微分方程），并通过惩罚复杂度先验实现有原则的先验设定 [Simpson 等人， 2017]。

- **确定性且可复现的推断。** PyINLA 返回后验边缘分布的确定性近似；因此不存在蒙特卡洛变异性，无需进行收敛诊断，也不需要设置随机种子。

- **无缝集成生态系统。** 结果以结构化 Python 对象的形式返回，可直接接入**pandas**、**matplotlib**、**scikit-learn**以及其他标准工具，用于后续分析、可视化和部署。

通过聚焦于核心潜在高斯模型功能，实现了广泛的潜在高斯模型覆盖范围，包括支持最常用的似然族和潜变量模型。计划在未来版本中加入更多模型组成部分。当前核心数值方法的最新成果也得到了支持。

本文其余部分安排如下。第 2 节回顾 Python 中的贝叶斯计算，并将 PyINLA与常用的替代方案进行比较。第 3 节概述潜在高斯模型框架和 INLA 近似策略，这些内容足以帮助读者理解该软件包的适用范围和输出。第 4 节介绍 PyINLA的设计、实现和用法。第 5节提供了具有代表性的示例，第 6 节则讨论其局限性和未来方向。

2 背景：贝叶斯计算及相关软件

应用贝叶斯工作流程涉及迭代式模型开发、在相关数据集上反复拟合，以及常规模型检查。在这些情境中，推断方法的选择不仅取决于统计目标，还受到运行时间、诊断负担，以及与数据处理和部署工作流程集成情况的影响。

在Python中，通用贝叶斯推断通常通过马尔可夫链蒙特卡罗或变分推断来实现，常见框架包括**PyMC**、**Stan (**通过**CmdStanPy)**、**TensorFlow Probability**和**Pyro** [Salvatier 等人，2016年、Carpenter 等人，2017年、Dillon 等人，2017年、Bingham 等人， 2019]。现代马尔可夫链蒙特卡罗方法，例如哈密顿蒙特卡罗和无转向采样器 [Hoffman 等人， 2014] 能够提供渐近精确的推断，但对于潜在结构庞大的层次模型而言，计算代价高昂，并且需要投入大量精力进行调优和收敛性评估。变分方法以优化取代采样，缩短运行时间，但会引入

表 1：Python 中可用的贝叶斯推断工具比较。

	PyINLA	PyMC	Stan	TFP
主要推断方法	INLA	马尔可夫链蒙特卡罗/变分推断	马尔可夫链蒙特卡罗/变分推断	马尔可夫链蒙特卡罗/变分推断
模型类别	仅限潜在高斯模型	常规	常规	常规
蒙特卡罗误差	No	Yes	Yes	Yes
收敛诊断	不需要	必需	必需	必需
空间模型（随机偏微分方程）	原生	手动	手动	手动
深度学习集成	No	间接[a]	No	原生

[a]可通过 PyTensor 后端实现深度学习集成，但这并非主要设计目标。

偏差，并在后验依赖较强时低估后验不确定性 [Rue 等人，2009年、Zhang 等人， 2019]。这些权衡促使人们为适用此类方法的模型类别探索确定性替代方案。

INLA 通过以解析近似取代抽样，为潜在高斯模型提供了一种确定性替代方案，概述见第 1 节，详见第 3 [Rue 等人，2009年，2017]。实证比较一再表明，在疾病制图、临床试验和金融建模等各种应用中，INLA 能达到与马尔可夫链蒙特卡罗相当的精度，同时运行速度快一到两个数量级 [De Smedt 等人，2015年，Chen 等人，2024年，Darkwah，2022年，Nacinben 和 Laurini， 2024]。

PyINLA 提供了原生 Python 接口，可连接已编译的 INLA 引擎，使潜在高斯模型的确定性贝叶斯推断能够直接融入 Python 工作流。

2.1 PyINLA 在现有 Python 工具中的定位

表 1 对比PyINLA 与 Python 中广泛使用的通用贝叶斯工具Python。

PyMC 提供灵活的概率编程接口，支持基于 NUTS 的马尔可夫链蒙特卡罗和变分推断。**Stan** 提供领域专用语言，并实现了先进的哈密顿蒙特卡罗/NUTS 算法。**TensorFlow Probability** 将概率建模与 TensorFlow 生态系统相结合，支持将贝叶斯推断与深度学习结合的工作流程。这些工具具有广泛的建模灵活性，但这种通用性也有代价：用户必须管理抽样效率、评估收敛情况，并承担具有复杂层级结构的模型所需的大量运行时间。

PyINLA 采用了根本不同的方法。它将关注范围限定于潜在高斯模型，并结合针对稀疏矩阵的数值方法，利用嵌套拉普拉斯近似，在无需任何抽样的情况下计算后验边缘分布。对于其适用范围内的大量模型，这种设计带来了切实的实践优势：

- **针对结构化潜在场的高效性**。 通过利用精度矩阵的稀疏性，可以高效拟合潜在分量维度较高的模型（例如空间效应和时空效应）。

- **内置结构化组件**。提供常见的时间和空间组件（例如随机

游走、自回归过程、CAR/BYM 类模型、基于随机偏微分方程的场），并配有经过合理设定的默认先验分布。

取舍在于适用范围：PyINLA 仅适用于潜在高斯模型。具有非高斯潜在结构的模型，或似然函数不受 INLA 支持的模型，需要使用通用工具。需要直接从完整联合后验中抽取样本的应用，也可能更适合使用马尔可夫链蒙特卡罗；不过，PyINLA 提供了后验抽样功能，可用于派生量推断。

3 INLA 方法

本节概述潜在高斯模型框架和 INLA 近似策略，介绍理解 PyINLA能够表示什么以及会返回什么所需的内容。更详细的方法论阐述见 Rue 等人 [2009], Abdul Fattah [2023], Niekerk 和 Rue [2024]。

在给出正式定义之前，我们可以先从数据出发逐步构建模型结构。假设我们观测到数据 y_i，这些数据可以是连续测量值（例如温度）、计数（例如病例数），也可以是二元结果（例如成功/失败）。我们希望将期望响应变量$\mu_i = \mathbb{E}[y_i]$ 建模为协变量和随机效应的函数。由于 μ_i通常受到限制（例如计数为正数，概率则在 (0,1) 内），因此使用链接函数 $g(\cdot)$将其映射到实数轴上。变换后的响应变量 $\eta_i = g(\mu_i)$称为线性预测子，其建模形式如下：

$$\underbrace{\eta_i = g(\mu_i)}_{\text{linear predictor}} = \underbrace{\beta_0 + \beta_1 x_{1i} + \cdots}_{\text{fixed effects}} + \underbrace{u^{(1)}_{k_1(i)} + u^{(2)}_{k_2(i)} + \cdots}_{\text{random effects}}, \qquad \underbrace{y_i \sim p(\cdot \mid \mu_i, \theta)}_{\text{likelihood}}. \tag{1}$$

这里，$\boldsymbol{\beta}$是固定回归系数，$u^{(k)}$是用于捕捉由空间邻近性、时间趋势或组成员关系造成的变异的结构化随机效应。它们共同构成潜在场 $\boldsymbol{x} = (\beta_0, \boldsymbol{\beta}, \boldsymbol{u}_1, \ldots\ldots, \boldsymbol{u}_K)^\top$，并被赋予联合高斯先验，详情如下。

3.1 潜在高斯模型

潜在高斯模型（LGM）可按照层级结构定义，其中包括：1）假定数据服从的族（或似然）；2）线性预测子（潜在场）中参数的高斯先验；3）超参数向量。设 $\boldsymbol{y} = (y_1, \ldots\ldots, y_n)^\top$ 表示观测数据，$\boldsymbol{x} \in \mathbb{R}^N$ 表示上文介绍的潜在场，$\boldsymbol{\theta} \in \mathbb{R}^p$ 表示超参数。简化示意如下；完整详情请参见Rue 等人 [2017]。

观测模型。给定 $\boldsymbol{x}$和 $\boldsymbol{\theta}$后，数据条件独立：

$$y_i \mid \boldsymbol{x}, \boldsymbol{\theta} \overset{\text{ind}}{\sim} p(y_i \mid \eta_i, \boldsymbol{\theta}), \qquad \eta_i = \boldsymbol{a}_i^\top \boldsymbol{x}, \tag{2}$$

其中 $\boldsymbol{a}_i^\top$通常是设计矩阵中的稀疏行，用于将潜在场映射到线性预测子。似然函数 $p(\cdot)$可以采用多种密度函数。唯一的技术要求是，它必须关于潜在场具有一阶和二阶导数。INLA 引擎支持种类繁多的似然族，也支持用户自定义似然函数；PyINLA 目前提供了其中最常用的子集（参见第4.4.3节）。

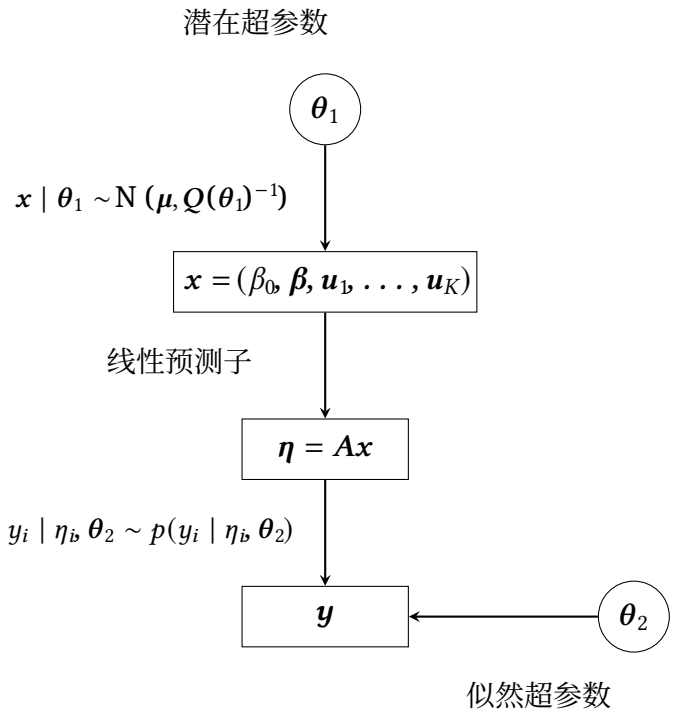

图 1：潜在高斯模型的层级结构。超参数分为潜在超参数 θ_1（控制潜在场 x 的精度结构）和似然超参数 θ_2（例如观测噪声方差），其中 $\theta = (\theta_1, \theta_2)$。线性预测子 η通过似然 $p(\cdot)$将 x与观测 y连接起来。

潜在场先验。在给定 θ 的条件下，

$$x \mid \theta \sim \mathcal{N}\big(\mu(\theta),\, Q(\theta)^{-1}\big), \tag{3}$$

其中 $Q(\theta)$是精度矩阵（协方差矩阵的逆矩阵）。在大多数结构化模型中，$Q(\theta)$是稀疏的，它编码了对计算效率至关重要的条件独立关系 [Rue 和 Held, 2005]。

超参数先验。 超参数是额外的似然参数（不在$\hat{\theta}$性预测子中），例如方差、离散度、零膨胀以及 $Q(\theta)$ 上的参数。为这些参数指定先验分布 $p(\theta)$，通常包括针对方差和相关参数的惩罚复杂度先验 [Simpson 等人, 2017]。PC先验通过直观的概率陈述而非抽象的分布参数来指定。例如， $P(\sigma > 1) = 0.01$ 表达了这样的信念：标准差超过 1 的可能性很低，并将估计收缩至不含随机效应的简单基准模型。

图 1 展示了潜在高斯模型的层级结构。

在PyINLA中，每个族都有默认的链接函数，也可以显式设置。固定效应通过`fixed` 指定，随机效应通过`random` 指定（参见第4节）。

3.2 推断目标与输出

INLA 旨在为潜在场的每个元素及超参数计算后验边际分布：

$$p(x_j \mid \boldsymbol{y}) = \int p(x_j \mid \boldsymbol{\theta}, \boldsymbol{y})\, p(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}, \qquad p(\theta_k \mid \boldsymbol{y}) = \int p(\boldsymbol{\theta} \mid \boldsymbol{y})\, d\boldsymbol{\theta}_{-k}. \tag{4}$$

这些边缘分布是非高斯的，拉普拉斯方法利用 $\boldsymbol{x}$ 上的高斯先验来准确且高效地近似它们。得到这些边缘分布后，均值、标准差和可信区间等后验摘要可直接根据这些边缘分布计算，并以 **pandas** DataFrame 的形式返回（见第 4 节）。INLA 关注边缘分布，而不是显式表示完整的联合后验；这种设计是其计算效率的重要原因。当需要完整的联合后验时，可在模型拟合 [Chiuchiolo 等人之后高效地抽取蒙特卡洛样本， 2021], 具体如第 4 节所述。

3.3 嵌套拉普拉斯近似与数值积分

INLA 通过结合以下两点开展推断： （i）针对高维高斯潜在结构的拉普拉斯近似；（ii）在超参数空间上进行数值积分 [Rue 等人， 2009, 2017]。概括而言：

1. 构造 $p(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{y})$ 的高斯近似，以计算 $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ 的拉普拉斯近似，并利用它搜索 $\boldsymbol{\theta}$ 的后验众数。2. 针对围绕其众数的一组支撑点 $\boldsymbol{\theta}$，近似条件边缘分布 $p(x_j \mid \boldsymbol{\theta}, \boldsymbol{y})$。3. 使用这组支撑点，并以与 $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ 成正比的权重对 $\boldsymbol{\theta}$ 进行积分，从而得到 $p(x_j \mid \boldsymbol{y})$ 和 $p(\theta_k \mid \boldsymbol{y})$。

注意：如果似然为高斯分布，则步骤 1 和 2 无需近似。

3.4 计算方面的考虑与局限

INLA 的第一步是通过优化确定超参数的众数，并使用自适应梯度估计技术 [Abdul Fattah 等, 2022]。 **inla** 程序通过乔列斯基分解和选择性求逆，利用潜在精度矩阵的稀疏性 [Rue 和 Held， 2005]。当精度矩阵稀疏时，这些操作速度很快；但如果精度矩阵稠密，它们就会成为瓶颈。先前的研究探索了针对特定情形的解决方案，包括稠密矩阵 [Abdul Fattah 等， 2025c] 以及大规模时空模型[Gaedke-Merzhäuser 等， 2024]。 **sTiles** 库 [Abdul Fattah 等, 2025年 b， a] 正在开发中，旨在为稀疏结构化矩阵提供高效分解，并计划在未来版本中加入 GPU 加速。 **inla** 源代码可在以下网址获取：`https://github.com/hrue/r-inla`。

4 接口说明

本节介绍 PyINLA 的设计、架构和用法。PyINLA 是编译后的 **inla** 程序的原生 Python 接口，可用于模型设定、执行和结果处理。

完全融入标准 Python 工作流。

4.1 安装与可用性

可以从 PyPI 安装 PyINLA：

```
pip install pyinla
```

对于需要额外功能的工作流程，可安装可选扩展，例如用于基于 SPDE 的空间模型的网格构建工具：

```
pip install pyinla[fmesher]
```

该 `[fmesher]` 扩展提供了Python 封装，以便调用**fmesher** 库 [Lindgren， 2023] 用于生成网格并进行 SPDE 预处理。我们建议在隔离环境中安装（例如，`conda`）。

PyINLA 是一个标准的 Python 包，不依赖 R。繁重的计算由**inla** 程序完成；这是一个编译后的 C 二进制程序，实现了核心数值算法、稀疏矩阵分解、拉普拉斯近似以及对超参数的数值积分。这与 R-INLA 使用的是同一套引擎，该引擎已运行 15 年有余，并在数千项已发表的分析中得到验证。PyINLA 会透明地管理此二进制程序：如果本地尚未安装，它会在首次使用时自动下载，因此无需手动配置。该软件包可直接在 Google Colab 等云端笔记本环境中运行，适用于教学、可重复演示和协作研究，无需在本地安装。完整文档、教程和交互式示例请访问`https://pyinla.org`。为支持可重复研究，代码库包含本文所报告全部结果的复现材料。本文介绍 PyINLA 0.2.0 版。

4.2 软件包架构

PyINLA 分为五层：

1. **面向用户的 API：**以 `pyinla()` 函数为核心的高级接口，该函数接收模型设定、数据和控制选项，并返回结构化的结果对象。2. **模型规范层：**将 Python 层级的模型定义转换为 INLA 引擎所需的内部表示，包括验证和默认值处理。3. **数据准备层：**将常见的 Python 对象、**pandas** DataFrame、**NumPy**数组和 **SciPy** 稀疏矩阵转换为引擎可处理的格式。4. **执行层：**负责协调 INLA 运行，包括管理工作目录、生成配置、调用编译后的引擎以及报告错误。5. **结果收集层：**解析引擎输出并构建 Python 结果对象，以提供后验边缘分布、摘要和诊断信息。

4.3 基本用法

一个拟合高斯线性模型的最简示例展示了主要工作流程。以下代码生成合成数据、设定模型、使用`pyinla()`进行拟合，并输出固定效应的后验摘要。

```
from pyinla import pyinla
import pandas as pd
import numpy as np

rng = np.random.default_rng(42)
n = 1000
x =rng.standard_normal(n)
y = -2.0 + 1.5 * x +rng.standard_normal(n) * 0.5
数据 = pd.DataFrame({"y": y,  "x": x})

模型 ={"response": "y", "fixed": ["1", "x"]}
结果 = pyinla(model=model,  族="高斯",  数据=数据)

print(result.summary_固定效应)
```

```
             mean     sd 0.025分位数 0.5分位数 0.975分位数   mode      kld
（截距）   -2.041  0.016    -2.072   -2.041    -2.009 -2.041 1.21e-11
x           1.496  0.016     1.464    1.496     1.528  1.496 1.05e-11
```

每行报告一个固定效应的后验均值、标准差、分位数和众数。对于非高斯似然，`kld`列很有用，因为它表示每个边缘分布的高斯近似与简化拉普拉斯近似之间的库尔贝克–莱布勒散度；数值接近零表示高斯近似足够准确。在此示例中，该值为零，因为当族为高斯时无需进行高斯近似。

```
import matplotlib.pyplot as plt marg_x = result.marginals_fixed["
x"]plt.figure(figsize=(5, 3)) plt.plot(marg_x[:, 0],marg_x[:, 1])
plt.axvline(1.5, color="red", linestyle="--", label="真实值") plt.xlabel("
x") plt.ylabel("密度") plt.legend() plt.tight_layout()
```

4.4 模型设定

PyINLA 中的统计模型以 Python 字典的形式指定。为了说明这种表示方式，我们从一个简单示例逐步扩展到一般情形，并在每一步展示字典如何映射到底层的数学模型。

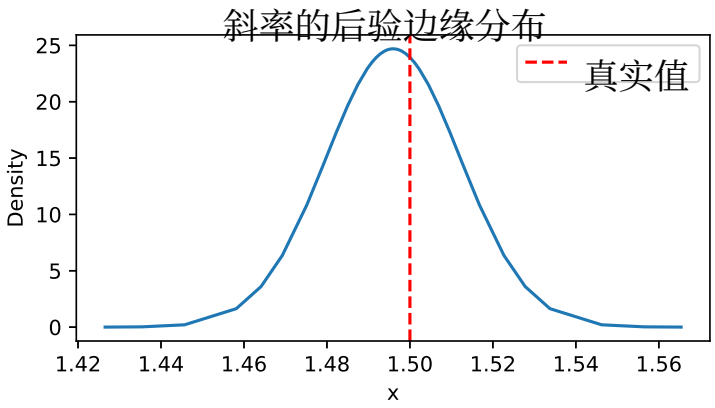

图2：斜率参数的后验边缘密度。红色虚线表示用于生成数据的真实值（1.5）。

4.4.1 从方程到字典：渐进式构建

步骤1：线性模型。 考虑最简单的情形：一个包含单个协变量的高斯线性模型，

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{5}$$

在 PyINLA 中，该模型指定如下：

```
模型 = {"response": "y","fixed": ["1", "x"]} 结果 =
pyinla(model=model, family="gaussian", data=data)
```

该字典包含两个键。`"response"` 键指定了`data` 中包含观测值 y_i 的列。`"fixed"`键列出了线性预测子中的项：`"1"` 表示截距β_0，`"x"` 表示系数为 β_1 的协变量列。参数 `family="gaussian"`指定采用恒等连接函数的高斯观测，这意味着线性预测子直接等于条件均值： $\eta_i = \mathrm{E}[y_i] = \beta_0 + \beta_1 x_i$。

步骤2：更改似然和链接函数。 现在假设响应变量是计数变量，并且我们希望将其建模为泊松分布：

$$y_i \sim \mathrm{Poisson}(\mu_i), \qquad \log(\mu_i) = \beta_0 + \beta_1 x_i. \tag{6}$$

代码中唯一的变化是`family` 参数：

```
模型 = {"response": "y","fixed": ["1", "x"]} 结果 =
pyinla(model=model, family="poisson", data=data)
```

字典保持不变，因为线性预测子 $\eta_i = \beta_0 + \beta_1 x_i$ 没有变化。变化的是 η_i 与观测之间的关系：泊松族使用对数连接函数，因此 $\eta_i = \log(\mu_i)$，或等价地 $\mu_i = \exp(\eta_i)$。

表2：PyINLA 中常用的似然族。每个族都为观测数据定义一个分布，并设有默认链接函数 $g(\cdot)$，将线性预测子 η_i 与期望响应 μ_i相关联。

族	分布	默认链接 $g(\mu_i)$	逆函数：$\mu_i =$	μ_i 的定义域
“高斯”	N (μ_i, σ^2)	$\eta_i = \mu_i$ (恒等)	η_i	$(-\infty, \infty)$
“泊松”	泊松(μ_i)	η_i = 对数 μ_i	e^{η_i}	$(0, \infty)$
"二项"	Bin(n_i, p_i)	η_i =对数$\frac{p_i}{1-p_i}$ (逻辑斯蒂)	$\frac{e^{\eta_i}}{1+e^{\eta_i}}$	$(0, 1)$
"负二项"	负二项分布(μ_i, ϕ)	η_i = 对数 μ_i	e^{η_i}	$(0, \infty)$
"伽马"	伽马(μ_i, ϕ)	η_i = 对数 μ_i	e^{η_i}	$(0, \infty)$
"beta"	Beta分布(μ_i, ϕ)	η_i =对数$\frac{\mu_i}{1-\mu_i}$ （逻辑斯蒂）	$\frac{e^{\eta_i}}{1+e^{\eta_i}}$	（0, 1）

每个似然族都有一个默认的链接函数 $g(\cdot)$，用于将期望响应 $\mu_i = \mathrm{E}[y_i]$映射到线性预测子：

$$\eta_i = g(\mu_i). \tag{7}$$

链接函数可确保线性预测子（其取值范围为整个实数轴）映射到各分布对应的取值域（例如，计数对应正实数，比例对应单位区间）。表2列出了常用的分布族及其默认链接函数。

链接函数由PyINLA在内部处理：用户在线性预测子尺度上指定模型。拟合值的后验摘要以线性预测子尺度报告；若要在响应尺度上获得结果（例如期望计数或概率），可使用 `pyinla.tmarginal()`应用逆链接函数，如第4.5.2节所示。

步骤 3：添加随机效应。假设我们将泊松模型扩展为包含组级随机截距：

$$y_i \sim \mathrm{Poisson}(\mu_i), \qquad \log(\mu_i) = \beta_0 + \beta_1 x_i + u_{g(i)}, \qquad u_j \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_u^2), \tag{8}$$

其中，$g(i) \in \{1, \ldots\ldots, J\}$ 表示观测 i 的组成员关系。数据DataFrame 必须包含一列 `group`，其整数索引用于标识每个观测所属的组。字典中新增一个`"random"` 键：

```
模型 ={ "响应变量": "y", "固定效应": ["1", "x"],"随机效应": [
{"id": "组", "模型": "iid"}]} 结果 = pyinla(模型=model, 族="泊
松", 数据=data)
```

该`"id": "group"`条目会告知PyINLA 在数据中查找`group` 列；其值决定哪些观测共享同一随机效应水平 u_j。该`"model": "iid"`条目指定u_j为可交换（相互独立、同分布）的高斯随机效应，如公式（8）所示。

步骤 4：多个结构化组件。模型通常包含多个具有不同依赖结构的随机效应。例如，同时包含组效应和平滑时间趋势的模型：

$$\log(\mu_i) = \beta_0 + \beta_1 x_i + u_{g(i)} + f_{t(i)}, \tag{9}$$

where $u_j \overset{\text{IID}}{\sim} \mathrm{N}(0,\ \sigma_u^2)$，且 f_t 遵循一阶随机游走，$f_t - f_{t-1} \sim \mathrm{N}(0,\ \sigma_f^2)$ ：

```
模型 ={ "response": "y", "fixed": ["1", "x"],"random": [{"
id": "group", "model": "iid"}, {"id": "time", "model":"rw1", "
constr": True}]} 结果 = pyinla(model=model, family="
poisson", data=data)
```

`"random"` 列表中的每个条目都会向线性预测子添加一个潜在分量。`"constr":True` 选项施加和为零约束（$\sum_t f_t = 0$），使随机游走与截距并存时具有可识别性。

总结。总体模式遵循公式（1）：字典指定右侧（固定效应 + 随机效应），而 族 参数则指定链接函数 g 和观测分布 p。

与统计软件中常见的基于公式的接口（例如，Python 中的 **patsy** 或**statsmodels**，或 R 中的公式语法）不同，PyINLA 使用普通字典。这种设计支持以编程方式构建模型，例如在循环中迭代候选模型结构、从配置文件读取规范，或根据数据属性有条件地构建模型。

4.4.2 固定效应

固定效应通过指定 数据 DataFrame 中的列名列表来设定。指定`"1"`可包含截距；若从列表中省略，则不包含截距：

```
模型 = {"response": "y", "fixed": ["1", "x1", "x2", "x3"]}
```

这对应于 $\eta_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i}$。回归系数默认采用精度较低的高斯先验（较宽泛的弱信息先验）。

4.4.3 似然族

PyINLA 支持连续响应的似然族（"高斯"、"伽马"、"Beta分布"）、离散响应（"泊松"、"二项"、"负二项"）以及生存数据（"威布尔生存"、"指数生存"）。默认连接函数列于表 2。

常见的附加参数包括：

- `E`：带偏置项的泊松模型的期望计数，使得 $\log(\mu_i) = \log(E_i) + \eta_i$ and th e 模型估计相对风险 μ_i/E_i，
- `Ntrials`：试验次数 n_i（用于二项分布数据），因此 $y_i \sim \text{Bin}(n_i, p_i)$，

• 权重和尺度：观测权重和尺度向量。# 用于病例计数的泊松模型，期望计数为 E_i # 模型：病例数_i ~服从泊松分布(E_i * theta_i)，log(theta_i) =beta_0 +beta_1* x_i 结果 = pyinla(model={"response": "cases", "fixed": ["1", "x"]}, family="poisson", data= data, E=data["expected"].to_numpy()) # 成功次数（共 n_i 次试验）的二项分布模型 # 模型：成功次数_i ~服从二项分布(n_i, p_i)，logit(p_i) = beta_0 +beta_1* treatment_i 结果 = pyinla(model={"response": "successes", "fixed": ["1", "treatment"]}, fa mily="binomial", data=data, Ntrials=data["n_trials"].to_numpy())

4.4.4 随机效应（潜在成分）

随机效应在 “random” 键下以分量字典列表的形式指定。每个组件都需要：

- “id”：数据中的列名，其值用于索引随机效应水平，
- “model”：施加于该效应的先验依赖结构类型。

表 3su 概述了可用的潜在模型类型及其数学定义 .

“iid” 模型最简单：它为每个层级指定相互独立的高斯先验，适用于组级截距（例如混合模型中的受试者、区域或项目）。随机游走模型（“rw1”、“rw2”）通过惩罚相邻值之间的差异，在有序索引（时间步、年龄、有序类别）上生成平滑函数。对于平稳时间序列，若 u_t 与 u_{t+k}之间的相关系数按 $\rho^{|k|}$衰减，则适合使用“bym”和“spde”模型，它们分别用于处理格网上的空间依赖和连续区域中的空间依赖，详见第5.3 节和第5.4节。

In 除“id”和“model”外，每个组件字典还接受可选标志 s:

- “constr”：True施加和为零约束（$\sum_j u_j = 0$），当该组件与截距共线时，可确保其可辨识性。
- “scale.model”：True 缩放精度矩阵，使其广义方差为 1，

表 3：PyINLA 中的内置潜在模型类型。每个模型都为随机效应向量 $\boldsymbol{u} = (u_1, \ldots, u_m)$ 定义先验分布。

“模型”	Name	先验结构
“iid”	可交换	$u_j \text{iid} \sim \mathrm{N}(0, \tau^{-1})$
"rw1"	一阶随机游走	$u_t - u_{t-1} \sim \mathrm{N}(0, \tau^{-1})$
"rw2"	二阶随机游走	$u_{t-1} - 2u_t + u_{t+1} \sim \mathrm{N}(0, \tau^{-1})$
"ar1"	自回归（一阶）	$u_t = \rho\, u_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}(0, \sigma_\varepsilon^2)$
"bym"	BYM（区域空间）	$b_i = u_i + v_i$; u_i: ICAR(τ_u), v_i独立同分布$\sim \mathrm{N}(0, \tau_v^{-1})$
"bym2"	BYM2（区域空间）	$b_i = \sigma(\sqrt{\phi}\, u_i^* + \sqrt{1-\phi}\, v_i^*)$
"spde"	Matérn场（随机偏微分方程）	通过随机偏微分方程构建的连续空间场
"generic0"	自定义精度	用户提供的 Q: $\boldsymbol{u} \sim \mathrm{N}(\mathbf{0}, \tau^{-1} Q^{-1})$

从而使精度超参数在维度或图结构不同的模型之间具有可比性。

- `"cyclic": True` 会将随机游走首尾相接，使最后一个层级与第一个层级相邻，适用于星期或年内日期等周期效应的季节性。

4.4.5 先验设定

每个潜在分量都有超参数（例如，IID 效应的精度 τ，或一阶自回归过程的相关系数 ρ），这些参数需要先验分布。可在随机效应字典中使用 `"hyper"` 键来指定这些先验分布。

PyINLA 支持惩罚复杂度（PC）先验 [Simpson 等人， 2017], 这些先验分布通过易于理解的概率陈述来指定。例如，陈述

$$\Pr(\sigma > U) = \alpha \tag{10}$$

为标准差 $\sigma = 1/\sqrt{\tau}$ 设定先验分布，使 σ 超过阈值 U 的概率为 α。代码如下：

```
模型 ={ "响应变量": "y", "固定效应": ["1", "x"],"随机效应": [{ "id": "
group", "model": "iid", "hyper": { "prec": { "先验": "pc.prec", "参数": [1.
0,0.01] # P(随机效应标准差 > 1.0) = 0.01 } } }
```

表4：PyINLA 中常用的先验族。`"prior"` 字符串传入`"hyper"`字典中；`"param"` 提供相应的参数。

"prior"	参数	说明
"pc.prec"	$[U, \alpha]$	σ 的惩罚复杂度先验：$P(\sigma > U) = \alpha$
"pc.cor1"	$[U, \alpha]$	关于 ρ 的惩罚复杂度先验（基准模型 $\rho = 1$）
"pc.cor0"	$[U, \alpha]$	关于 ρ 的惩罚复杂度先验（基准模型 $\rho = 0$）
"pc"	$[U, \alpha]$	通用 PC 先验，例如 BYM2 混合 ϕ
"loggamma"	$[a, b]$	对数伽马分布（应用于 $\log \tau$，形状参数 a，速率参数 b）
"高斯"	$[\mu, \tau]$	高斯分布（应用于 θ，均值 μ，精度 τ）
"平坦"	—	非正规常数（非信息性）
"pc.dof"	$[U, \alpha]$	自由度的惩罚复杂度先验
"Beta分布相关系数"	$[a, b]$	相关系数的贝塔先验
"表："	grid	θ上的列表化对数密度

```
    ]
}
```

这里 `"参数": [1.0, 0.01]` 编码了公式 (10)，其中 $U = 1.0$ 和 $\alpha = 0.01$：该先验将随机效应标准差超过 1.0 的概率设为仅 1%，从而使其向零收缩（即不含随机效应的基准模型）。表 4 列出了最常用的先验族。

4.5 结果对象

函数 `pyinla()` 会返回一个 `PyINLAresult` 对象，其中提供了后验摘要、边际密度和模型评估量，格式均为原生 Python 格式。

4.5.1 后验摘要

固定效应、潜在成分、超参数和拟合值的摘要以 **pandas** DataFrame 的形式返回：

```
result.summary_固定效应
result.summary_随机效应
result.summary_超参数
result.summary_拟合_值
```

4.5.2 边缘后验分布和实用函数

边际后验密度以数组表示，数组由$(x,$ 密度) 对组成。实用函数可用于评估、变换和汇总（参见附录A）：

```
导入 pyinla 导入 numpy 作为 np 边
缘分布 = result.marginals_固定效
应["x"]
```

```
概率_为正 = 1.0 -pyinla.pmarginal(0.0, marg) # 累积分布函数 (p) 最高
后验密度区间95 = pyinla.hpdmarginal(0.95, marg) # 最高后验密度区间
边缘分布 _ 或 = pyinla.tmarginal(np.exp, marg) # 变换 (t) 边际摘要 =
pyinla.zmarginal(marg) # 汇总统计量 (z)
```

4.6 控制选项

控制参数是一个可选字典，用于调整默认设置以外的计算方式和先验设置。它由多个命名分组组成，每个分组控制拟合过程的不同方面。最常用的分组包括：

- “compute”：用于切换是否生成额外输出。“dic”、“waic”和“cpo”用于启用相应的模型比较准则；“config”用于存储posterior_sample()所需的内部配置；“return_marginals”用于请求拟合值的完整边际密度。
- “fixed”：固定效应的默认先验精度。“prec”用于设置回归系数零均值高斯先验的精度（方差的倒数），而“prec.intercept”则专门设置截距的精度。较低的值对应更宽、信息量更少的先验分布。

所有键都是可选的；未指定的设置将保留默认值。

4.7 模型诊断

通过control请求时，PyINLA 会计算 INLA 工作流程中常用的诊断指标，包括偏差信息准则、广泛适用信息准则、CPO，以及边际似然的近似值：

```
结果 = pyinla( 模型=model, 族="gaussian", 数据=data, control={"
compute": {"dic": True, "waic": True, "cpo": True}} ) 偏差信息准则 =
result.dic 广泛适用信息准则 = result.waic mlik = result.mlik
```

4.8 后验抽样

尽管 INLA 侧重于边缘后验分布，但对于依赖多个组件的联合派生量，后验抽样很有用。PyINLA 提供了三种抽样函数：

- 后验_sample(n, result, seed) 抽样n 从潜在场和超参数的近似后验分布中抽取联合样本。config 标志必须在control 中启用（参见第4.6节）。

- 后验_样本_eval(fun, samples) 用于提取或转换样本。当fun 是字符串（例如，"x"）时，它会返回指定固定效应的样本；当fun 是可调用对象时，则会将其应用于每个样本以计算派生量。

- 超参数_sample(n, result) 从仅针对超参数的近似边际后验中抽取n 个样本。

```
结果 = pyinla( 模型=model, 族="gaussian",
数据=data, 控制参数={"compute": {"config":
True}} )




样本 = pyinla.posterior_sample(n=1000, result=result, seed=42) x_样本 =
pyinla.posterior_sample_eval("x", samples) exp_x = pyinla.posterior_
sample_eval(lambda x, **_: np.exp(x), samples) 超参数_样本 =
pyinla.hyperpar_sample(n=1000, result=result)
```

4.9 稳健执行

PyINLA 针对数值困难提供了自动回退策略。如果默认设置遇到问题（例如优化过程不稳定或不收敛），软件包会使用更保守的设置重试，例如采用替代近似模式、调整积分策略，或为病态系统启用稳定化选项。此行为由 safe参数控制（默认值：True）。

4.10 支持的功能与持续开发

PyINLA 支持常用的似然族、潜在成分（可交换、时间、空间和通用成分）、后验摘要和边际分布、关键诊断以及后验抽样。随着更多组件经过测试并趋于稳定，支持范围将持续扩大。本软件包使用与第 **inla** 节所述相同的已编译3引擎，确保底层计算的数值准确性。

支持的功能的完整文档可在软件包网站上查阅。

5 个示例

本节通过四个结构复杂度逐步递增的应用来展示PyINLA。第一个应用将采用可交换随机效应的层级泊松模型用于足球比赛预测，展示广义线性混合模型的核心工作流程。第二个应用使用缩放 BYM 模型 [Riebler 等人， 2016] 对苏格兰唇癌数据集进行区域疾病制图，并引入格网上的空间结构化随机效应。第三个应用使用随机偏微分方程方法 [Lindgren 等人， 2011] 利用连续空间坐标进行地统计温度建模。第四个应用展示带有不确定性量化的时间序列预测，

并将PyINLA 与基于神经网络的**NeuralProphet** 工具 [Triebe等人， 2021]进行比较。这些示例共同展示了从标准广义线性混合模型到格网和地统计空间模型，再到结构化时序模型的递进过程。更多示例请访问https://pyinla.org。

5.1 体育分析：足球比赛结果预测

此示例展示了一个带有交叉随机效应的泊松广义线性混合模型（GLMM），用于预测足球比赛结果。我们将 PyINLA 与 PyMC [Salvatier 等人， 2016] 以及采用无转向采样器（NUTS）的方案进行比较，以验证后验推断的准确性并量化计算速度的提升。

5.1.1 数据

该数据集包含 2018–2019 英格兰足球超级联赛赛季中 20 支球队参加的 313 场已进行比赛（原定共 380 场）。每场比赛会重新整理为两条观测记录（每支球队一条），得到长格式的 $2 \times 313 = 626$ 行：

```
import pandasas pd from pyinlaimport pyinla # 加载并重新整理数据行 = []
for_, match in played_df.iterrows(): rows.append({'goals': match['主场_进
球'],'attack': team_to_id[match['主场_球队']],'defense': team_to_id[match['
客场_球队']], 'home': 1}) rows.append({'goals': match['客场_进球'],'
attack': team_to_id[match['客场_球队']],'defense': team_to_id[match['主场_
球队']], 'home': 0}) data = pd.DataFrame(rows)
```

5.1.2 实现

该模型设定使用独立同分布随机效应来表示进攻和防守能力：

```
模型 ={ 'response': 'goals', 'fixed': ['1', 'home'],'random': [{'id': 'attack',
'model':'iid', 'hyper': {'prec':{'prior': 'loggamma', 'param': [1, 0.01]}},
{'id': 'defense','model': 'iid', 'hyper': {'prec':{'prior': 'loggamma', 'param':
[1, 0.01]}}}]} 结果 = pyinla(model=model, family='poisson', data=data,
control={'compute': {'dic': True, 'mlik': True}})
```

表5：足球预测模型中 PyINLA 与马尔可夫链蒙特卡罗（PyMC/NUTS）的计时和参数比较。PyINLA 实现了 ~90× 的加速，同时生成了几乎完全相同的后验估计。

	PyINLA		MCMC		
参数	Mean	SD	Mean	SD	|Δ|
计时					
CPU 时间（秒）	0.24		21.74		92×
固定效应					
截距	0.150	0.101	0.150	0.100	0.001
Home	0.237	0.068	0.237	0.068	0.000
超参数					
τ_{attack}	12.41	5.10	12.41	5.06	0.00
τ_{defense}	22.75	11.57	22.54	11.07	0.21

pyINLA 拟合时间：0.24 秒

固定效应：

	mean	sd	0.025分位数	0.975分位数
（截距）	0.150466	0.100775	-0.050613	0.346319
home	0.237082	0.067770	0.104192	0.369971

超参数：

	mean	sd	0.025分位数	0.975分位数
进攻精确度	12.408061	5.102970	5.216360	24.937430
防守精确度	22.751203	11.570014	8.076030	52.307311

ˆ 主场优势系数 $\beta = 0.237$ 对应于预期进球数的乘数增幅为 $e^{0.237} \approx 1.27$，这表明主队的进球数比在中立场地时约高 27%。95%可信区间 $[0.104, 0.370]$ 完全高于零，为主场效应提供了有力证据。

5.1.3 与马尔可夫链蒙特卡罗的比较

为验证 PyINLA 的准确性，我们使用 PyMC 和 NUTS 拟合了相同的模型（4 条链，每条链在 5,000 次调优迭代后抽取 25,000 个样本）：

表5 报告了模型拟合的实际运行时间。[1] PyINLA 完成推断用时 0.24 秒，而马尔可夫链蒙特卡罗用时 21.74 秒（4 条链 × 25,000 个样本，按顺序运行），CPU 时间约提升 92× 倍。固定效应估计值精确到小数点后三位均一致，而超参数估计值的差异最多为 0.21，远在后验不确定性范围内。全部 40 个随机效应（20 个攻击 + 20 防守）之间的皮尔逊相关系数为 $r = 1.0000$。

图 3叠加显示 PyINLA 对主场优势参数的 MCMC 直方图所绘制的边际密度

[1]本文中的所有计时均在 Intel Core Ultra 7 155H（4.8 GHz）单核上进行，配备 64 GB 内存，运行 Ubuntu 24.04 和 Python 3.12。马尔可夫链蒙特卡罗的计时反映了所有链按顺序运行时的总采样时间；并行运行可缩短实际运行时间，但不会减少总 CPU 时间。

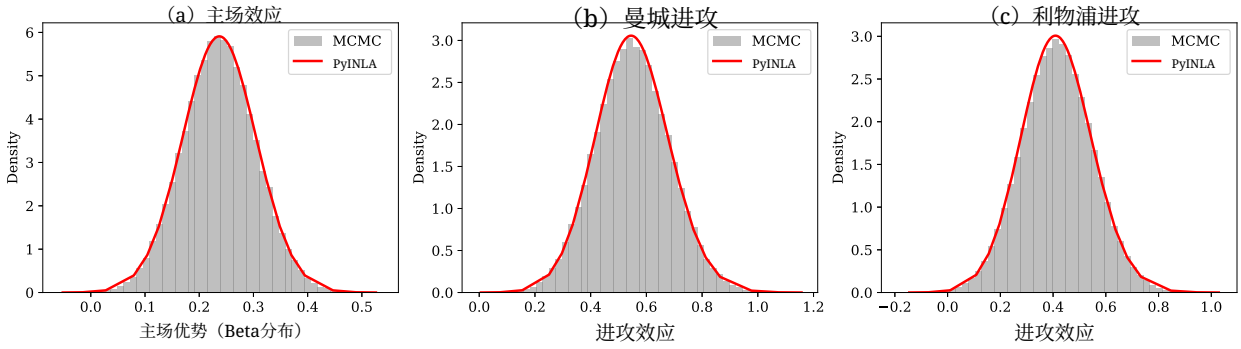

图 3：PyINLA 与马尔可夫链蒙特卡罗的验证。左：主场优势参数的 INLA 边缘密度（红色曲线）与马尔可夫链蒙特卡罗直方图（灰色）叠加对比。中/右：曼彻斯特城和利物浦的进攻效应。两者高度一致，证实 INLA 近似对于此类模型是准确的。

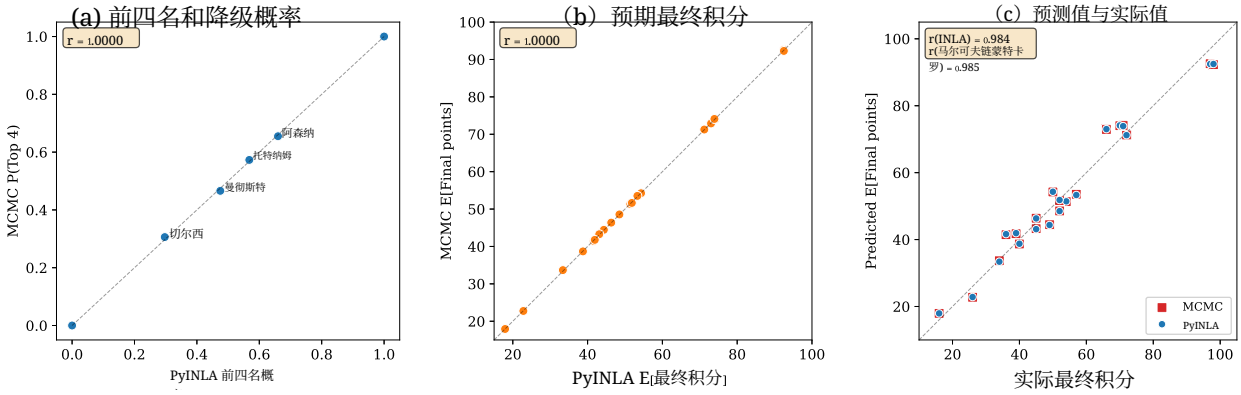

图 4：赛季模拟结果。(a) 前四名晋级概率：PyINLA 与马尔可夫链蒙特卡罗（$r = 0.9999$）。(b) 预期最终积分（$r = 0.9998$）。(c) 两种方法预测的最终积分与实际最终积分对比。两种方法都正确识别出利物浦和曼彻斯特城是冠军争夺者，哈德斯菲尔德、富勒姆和卡迪夫是降级候选球队。

以及选定球队效应的边际密度，展示了两种方法之间的高度一致性。

5.1.4 预测模拟

使用`后验_sample()`抽取 1,000 个联合后验样本后，我们对尚未进行的 67 场比赛进行模拟，得到完整赛季的结果。图4 比较了 PyINLA 和马尔可夫链蒙特卡罗的预测分布，包括前四名晋级概率和预期最终积分。

两种方法都正确地将利物浦和曼彻斯特城识别为冠军争夺者（P(前四名)≈ 1.0），并将哈德斯菲尔德、富勒姆和卡迪夫识别为降级候选球队。对于所有派生量，PyINLA 与马尔可夫链蒙特卡罗预测值之间的相关系数均超过 0.999，这证实了 INLA 近似保留了下游决策所需的信息。

总而言之，对于所有固定效应和随机效应，PyINLA 与马尔可夫链蒙特卡罗的后验均值差异均在 10^{-3} 以内，皮尔逊相关系数为 $r = 1.0$。PyINLA 在不到一

秒内即可完成推断；与进行 100,000 次后验抽样的马尔可夫链蒙特卡罗相比，速度提升了 92× 。两种方法模拟的赛季结果几乎完全相同，包括晋级概率和预期积分。

复现材料。 完整的复现脚本可在补充材料中获取（`football.py`）。

5.2 时间序列预测：网站流量预测

本例展示如何进行时间序列预测并量化不确定性，同时将 PyINLA 与 NeuralProphet [Triebe等人, 2021], 这一广泛使用的基于神经网络的预测工具进行比较。我们使用佩顿·曼宁的维基百科页面浏览量数据集（2007–2016年）。该数据集是预测文献中的标准基准数据集，包含 2964 个对数变换后的页面浏览量每日观测值。

5.2.1 模型设定

设 y_t 表示时刻 t 的对数页面浏览量。我们将该序列分解为：

$$y_t = \mu + f_{\text{trend}}(t) + f_{\text{week}}(w_t) + f_{\text{year}}(d_t) + u_t + \varepsilon_t, \tag{11}$$

其中， μ 为截距， f_{trend} 为采用二阶随机游走（RW2）建模的平滑趋势， f_{week} 采用周期为 7 的循环二阶随机游走捕捉星期效应， f_{year} 采用周期为 365 的循环二阶随机游走捕捉年度季节性， u_t 服从一阶自回归过程，以建模残差自相关，而 $\varepsilon_t \sim \mathrm{N}(0, \sigma_\varepsilon^2)$ 为观测噪声。

日历对齐的季节性指标直接根据日期计算，以确保正确对齐：

```
周_idx =dates.dt.dayofweek.values + 1 # 星期一=1, ..., 星期日=7年_idx =
np.clip(dates.dt.dayofyear.values, 1, 365) #处理闰年
```

5.2.2 实现

我们将数据划分为训练期（前 2599 天）和测试期（剩余 365 天）：

```
import numpy as np import pandas as pd from pyinla
import pyinla # 加载数据 df =pd.read_csv("peyton_
manning.csv", parse_dates=['ds']) y =df['y'].values dates =
df['ds']# 训练集/测试集划分 N_TRAIN = 2599y_train = y[:N_
TRAIN]n_total = len(y)
```

表6：佩顿·曼宁基准数据集上的预测准确率比较。

方法	RMSE	平均绝对误差 95%置信区间覆盖率		平均宽度
NeuralProphet	0.575	0.466	—	—
PyINLA（无 AR1）	1.107	0.977	94.0%	—
PyINLA（含 AR1）	**0.486**	**0.308**	98.4%	2.75

```
# 为测试期（预测值）准备含缺失值的数据 y_full = np.concatenate([y_
train, np.full(n_total - N_TRAIN, np.nan)]) data = pd.DataFrame({ 'y': y_
full, 't': np.arange(1, n_total + 1), 'week': (dates.dt.dayofweek.values + 1).
astype(int), 'year_day': np.clip(dates.dt.dayofyear.values, 1,
365).astype(int), 't_ar': np.arange(1,n_total + 1) })
```

模型设定包含 PC先验，用于表达对分量平滑性和自相关强度的先验判断：

```
模型 ={ 'response': 'y', 'fixed': ['1'],'random': [{'id': 't', 'model': 'rw2',
'scale.model': True, 'hyper': {'prec':{'prior': 'pc.prec', 'param': [0.5, 0.01]}}},
{'id': 'week', 'model': 'rw2', 'cyclic': True, 'hyper': {'prec':{'prior': 'pc.prec',
'param': [1, 0.01]}}}, {'id': 'year_day', 'model': 'rw2', 'cyclic': True, 'hyper':
{'prec':{'prior': 'pc.prec', 'param': [1, 0.01]}}}, {'id': 't_ar', 'model': 'ar1',
'hyper': {'prec':{'prior': 'pc.prec', 'param': [1, 0.01]}, 'rho':
{'prior':'pc.cor1', 'param': [0.9, 0.9]}}}] 结果 = pyinla(model=model,
family='gaussian', data=data)
```

5.2.3 结果

表 6 报告了 365 天测试期内的预测准确率。NeuralProphet 分别采用默认和调优后的超参数运行，报告效果最佳的配置（默认设置）。这项比较仅作说明，并非正式基准测试：NeuralProphet 是一种点预测工具（不确定性区间需要额外配置），因此主要目的是展示，作为贝叶斯推断的自然副产品，PyINLA的结构化加性模型在提供具有竞争力的点预测的同时，还能提供经过校准的不确定性量化。

采用 AR(1) 的 PyINLA 的均方根误差比 NeuralProphet 低 15.5%。AR(1) 组件至关重要：若不使用该组件，PyINLA 的表现会明显更差（均方根误差 = 1.107），这表明对该数据集中的残差自相关进行建模十分重要。

95%可信预测区间覆盖率为 98.4%，这表明不确定性量化略显保守，但校准良好。估计的自相关 $\hat{\rho} \approx 0.65$表明，在考虑趋势和季节性后，日与日之间仍存在中等程度的持续性。

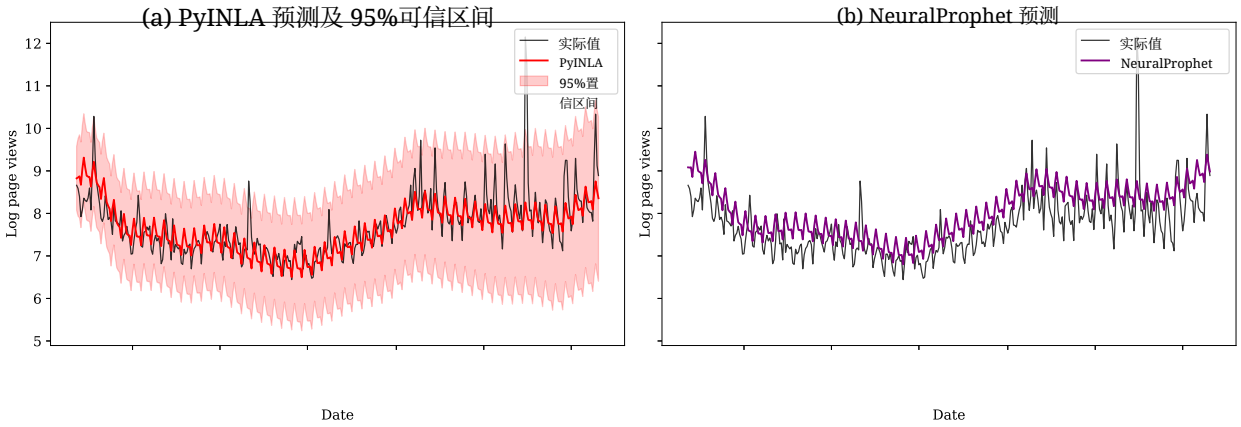

图5：佩顿·曼宁维基百科页面浏览量的预测对比。左：PyINLA 预测（红色），带有 95%可信区间（阴影部分）。右：NeuralProphet 预测（紫色）。黑线表示实际测试数据。

图5 展示了预测对比。PyINLA 提供经过校准的 95%可信区间，在预测期内适当扩大；而 NeuralProphet 默认只提供点预测。

PyINLA 的一项关键优势是能够以可解释的方式分解组件并量化不确定性。图6展示了各组件的后验均值和 95%可信区间：平滑趋势捕捉长期人气变化，每周效应揭示工作日/周末模式，年度效应则体现 NFL 赛季的动态变化（高峰期为 9 月至次年 2 月）。

局限性。 本次比较仅使用一个数据集，并采用一次训练集/测试集划分。结果仅供说明，并非定论；在其他数据集上或进行超参数调优时，性能可能会有所不同。全面比较需要使用多个数据集并进行交叉验证。

复现材料。 完整的复现脚本作为补充材料提供（`timeseries.py`）。

5.3 疾病制图：苏格兰唇癌

本示例展示了如何使用经典的 BYM 模型进行区域疾病制图 [?]，并使用缩放精度矩阵（`scale.model=True`）应用于苏格兰唇癌数据集 [Clayton和Kaldor，1987]。尺度调整可确保 ICAR 和 IID 组件的精度超参数具有可比性

，[?]。我们将 PyINLA 与使用 NUTS 的 PyMC 进行比较，以验证空间结构化随机效应模型的后验推断。

请注意，BYM2 重新参数化 [Riebler 等人， 2016], 使用单个总标准差和一个混合参数，也可通过 `"model": "bym2"` 在 PyINLA 中使用，并且推荐用于新的分析；这里我们使用经典形式来演示分别进行超参数估计。

5.3.1 模型设定

设 Y_i 表示区 i 中观察到的唇癌病例数，其预期病例数为 E_i，该数值基于年龄-性别标准化得出。我们将相对风险 θ_i 建模为：

$$Y_i \sim \text{Poisson}(E_i \cdot \theta_i), \qquad \log \theta_i = \beta_0 + \beta_1 \cdot x_i + b_i, \tag{12}$$

其中，β_0 为截距，x_i 为从事农业、渔业和林业（AFF）的人口比例，b_i 为空间结构化随机效应，遵循缩放 BYM 分解

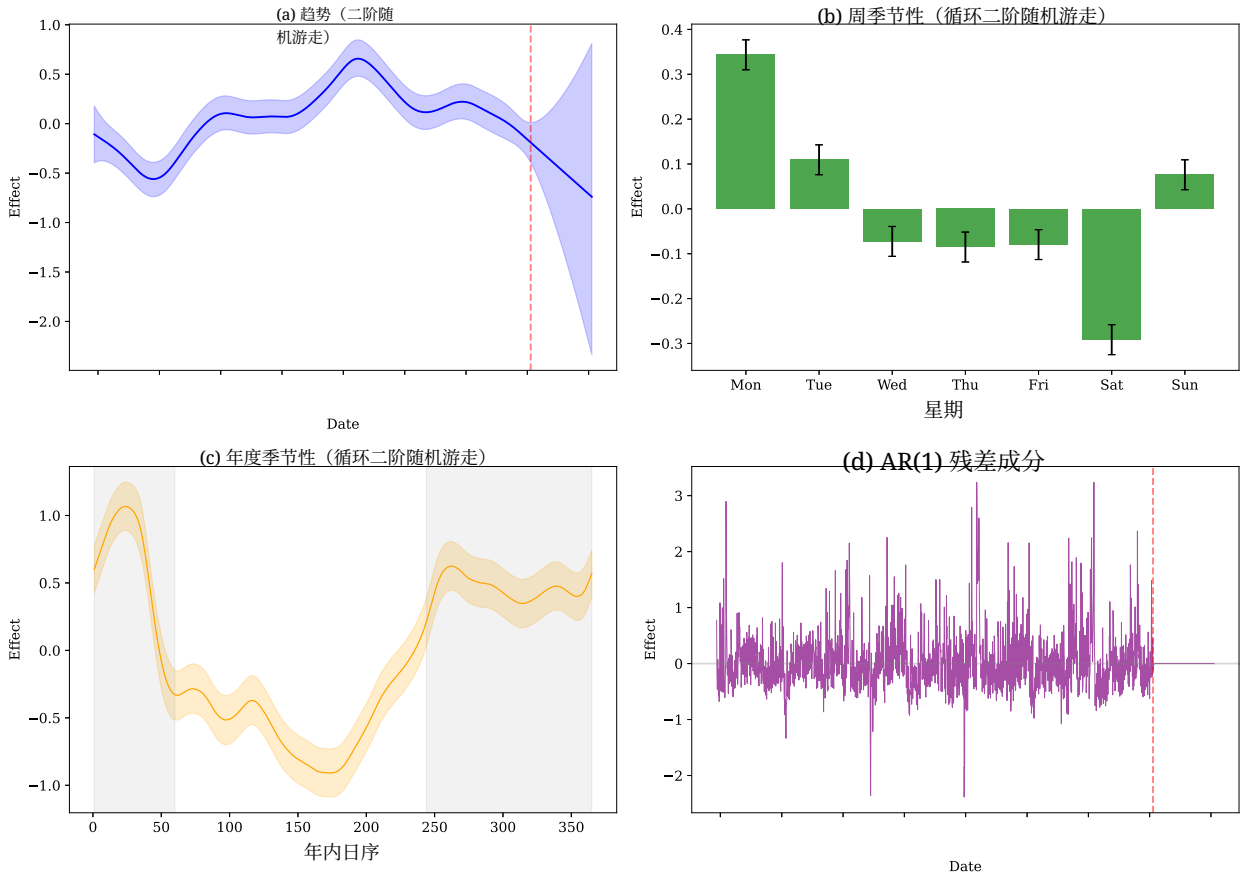

图 6：时间序列成分的后验分解。左上：趋势（二阶随机游走）。右上：周季节性（循环二阶随机游走，周期为 7）。左下：年度季节性（循环二阶随机游走，周期为 365）。右下：AR(1) 残差。阴影区域表示 95% 可信区间。

:

$$b_i = \frac{\sigma_{\text{spatial}}}{\sqrt{s}} \cdot u_i + \sigma_{\text{iid}} \cdot v_i, \tag{13}$$

其中， u_i 是经过因子 s （ICAR 精度矩阵边际方差的几何平均数）尺度调整的内在条件自回归（ICAR）分量， $v_i \sim \text{N}(0,1)$ 是非结构化独立同分布分量，而 σ_{spatial} 和 σ_{iid} 是两个标准差超参数。尺度调整确保两个精度超参数具有可比性，空间占比则根据后验样本事后推导得出：

$$\phi = \frac{\sigma^2_{\text{spatial}}}{\sigma^2_{\text{spatial}} + \sigma^2_{\text{iid}}}. \tag{14}$$

5.3.2 数据

该数据集包含苏格兰的 56 个行政区，记录了 1975–1980 年间的唇癌病例。AFF 协变量反映职业性日晒暴露，这是唇癌的已知风险因素：

```
import pandas as pd import numpy as np from pyinla import pyinla
data = pd.read_csv("scotland_data.csv") data['idarea'] =np.arange(1,
len(data) + 1) data['SIR'] = data['Y'] /data['E']print(f"行政区数量:
{len(data)}") print(f"病例总数: {data['Y'].sum()}") print(f"AFF 范围: {data[
'AFF'].min():.2f} 至 {data['AFF'].max():.2f}")
```

```
行政区: 56 总病例数:
536 AFF 范围: 0.00 至
0.24
```

邻接图定义了每个行政区的空间邻居。三个孤立岛屿（奥克尼群岛、设得兰群岛、西部群岛）在原始图中没有邻居，因此将它们与最近的大陆行政区相连，形成一个连通分量，共有 56 个节点和 120 条边。

5.3.3 实现

缩放 BYM 模型使用惩罚复杂度（PC）先验 [Simpson 等人， 2017]用于两个精度超参数。每个 PC 先验都编码了 $P(\sigma > 1) = 0.01$，将每个标准差收缩至零：

```
模型 ={ '响应变量': 'Y', '固定
效应': ['1', 'AFF'],'随机效应
': [{ 'id': 'idarea',
```

'模型': 'bym', '图': 'output/scotland_connected.adj', '缩放模型': True, '超参数': { 'theta1': {'先验': '惩罚复杂度精度先验', '参数': [1, 0.01]}, 'theta2': {'先验': '惩罚复杂度精度先验', '参数': [1, 0.01]} } }]} 结果 = pyinla(模型=模型, 族='泊松', 数据=数据, E=数据['E'].values, 控制={'计算': {'偏差信息准则': True, '广泛适用信息准则': True, '配置': True, '返回_边际分布': True}, '固定效应': {'prec.intercept': 0.001, 'prec': 0.001}}) PyINLA 拟合时间：0.42秒 固定效应：均值 标准差 0.025分位数 0.975分位数 （截距） -0.264932 0.123754 -0.507552 -0.020795 AFF 4.217436 1.279918 1.662672 6.697430 超参数：均值 标准差 0.025分位数 0.975分位数 精度。idarea (iid) 874.329651 6125.35 7.99932 5891.90199 精度。idarea (空间) 4.762095 1.86 2.04940 9.23697

ˆ AFF 系数 $\beta_1 = 4.22$ 表明，AFF 就业增加 10% 与风险乘以 $e^{0.1\times4.22} \approx 1.52$相关。IID 精度极高（$\hat{\tau}_{\text{iid}} \approx 874$，对应于 $\hat{\sigma}_{\text{iid}} \approx 0.12$），这表明几乎所有残差变异都具有空间结构，而非独立噪声。

空间占比 nϕ是根据后验超参数样本使用 inla_hyperpar_sampl e:

from pyinla.sampling import inla_hyperpar_sample hyp_samples =inla_hyperpar_sample(n=100000, result=result, intern=True).values vv =np.exp(-hyp_samples) # 列: 1/tau_iid, 1/tau_spatial phi_samples =vv[:, 1]/ vv.sum(axis=1) print(f"空间占比 (phi): {phi_samples.mean():.3f}") print(f"95%置信区间: [{np.percentile(phi_samples, 2.5):.3f}, " f"{np.percentile(phi_samples, 97.5):.3f}]") 空间占比 (phi): 0.920 95%置信区间: [0.644, 0.999]

ˆ派生的 $\phi = 0.92$ 表明，随机效应方差中约有 92% 具有空间

表 7：缩放 BYM 疾病制图模型中 PyINLA 与马尔可夫链蒙特卡罗（PyMC/NUTS）的计时和参数比较。

	PyINLA		MCMC		
参数	Mean	SD	Mean	SD	|Δ|
计时					
CPU 时间（秒）	0.42		117.2		278×
固定效应					
截距	−0.265	0.124	−0.261	0.122	0.004
AFF	4.217	1.280	4.170	1.269	0.047
超参数（标准差尺度）					
σ_{spatial}	0.450	—	0.474	0.091	0.024
σ_{iid}	0.120	—	0.119	0.090	0.001
派生的					
ϕ	0.920	—	0.907	—	0.013

结构化特征，这与风险因素的地理聚集相一致。

5.3.4 与马尔可夫链蒙特卡罗的比较

为验证 PyINLA，我们使用 PyMC 和 NUTS 拟合了相同的缩放 BYM 模型（4 条链，每条链在 3,000 次调优迭代后抽取 5,000 个样本，`目标_接受率=0.95`）。惩罚复杂度先验 $P(\sigma > 1) = 0.01$可转换为标准差上的指数先验，其速率为 $\lambda = -\log(0.01) \approx 4.605$:$\sigma_{\text{spatial}} \sim \text{Exp}(4.605)$, $\sigma_{\text{iid}} \sim \text{Exp}(4.605)$。

表 7 显示，PyINLA 完成模型拟合仅需 0.42 秒，而马尔可夫链蒙特卡罗需要 117.2 秒，速度提升约 278×。固定效应非常接近（AFF 系数仅相差 0.04 个后验标准差），且全部 56 个区的地区级相对风险之间的皮尔逊相关系数为 $r = 0.9999$，这表明在两个精度超参数采用相同先验分布的情况下，INLA 与马尔可夫链蒙特卡罗的结果高度一致。

5.3.5 疾病风险制图

图 7 展示了苏格兰 56 个行政区的后验平均相对风险和超越概率分级设色地图。红色表示风险升高（RR > 1），蓝色表示风险降低。高风险区域集中在北部农业区，尤其是斯凯-洛哈什、班夫-布肯和凯斯内斯，这些地区的 AFF 就业人数最多。

PyINLA 的一项主要优势是可以直接获取超越概率。对于每个区，我们计算 $P(\theta_i > 1 \mid \boldsymbol{y})$，即真实相对风险超过 1 的后验概率：

```
import pyinla as pinla for i in range(len(数据)): 超越 = 1.0- pinla.pmarginal(1.0,
result.marginals_fitted_values[i]) data.loc[i, '超越_概率'] = 超越
```

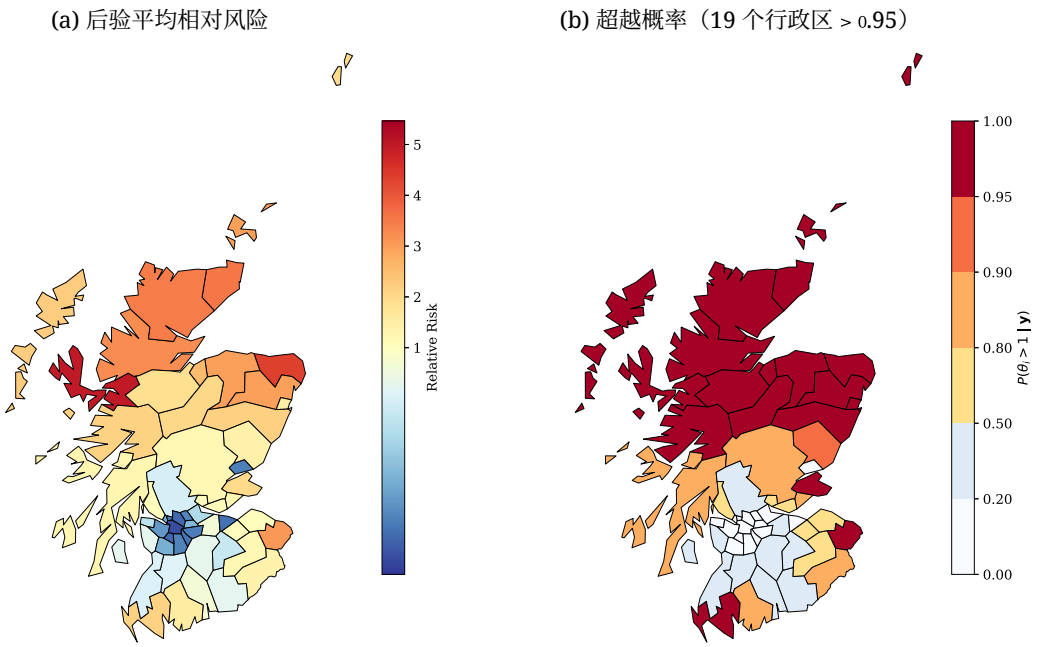

图 7：左：缩放 BYM 模型中苏格兰 56 个行政区唇癌的后验平均相对风险。颜色以 RR $= 1$ 为中心渐变，红色表示风险升高，蓝色表示风险降低。右：超越概率 $P(\theta_i > 1 \mid \boldsymbol{y})$。空间模式反映了 AFF 协变量效应和残差空间相关性的共同作用。

56 个行政区中，有 31 个的超越概率为 > 0.95，这有力地表明其风险升高。

5.3.6 模型拟合

缩放 BYM 模型的偏差信息准则为 $= 297.63$，广泛适用信息准则为 $= 294.64$。空间平滑显著降低了粗略标准化发病比中的噪声：例如，斯凯-洛哈什的粗略标准化发病比为 6.43，但平滑后的相对风险为 4.66，这是借助邻近行政区的信息得到的。

PyINLA 和马尔可夫链蒙特卡罗得到的估计结果一致，地区级相对风险的相关性为 $r = 0.9999$，AFF 系数仅相差 0.04 个后验标准差。PyINLA 在 0.42 秒内完成拟合，而马尔可夫链蒙特卡罗耗时 117 秒，速度提升了 $278\times$ 倍。推导出的ˆ空间占比 $\phi = 0.92$ 表明，92% 的残差变异具有空间结构化特征，这与苏格兰户外工作暴露的地理聚集情况一致。

复现材料。 完整的复现笔记本可作为补充材料获取（`scottish.py`）。

5.4 地统计建模：区域温度插值

本示例演示了使用随机偏微分方程（SPDE）方法进行连续空间预测 [Lindgren 等人, 2011], 该方法应用于来自黎巴嫩、叙利亚、约旦和沙特阿拉伯 55 个气象站的 2024年1月温度数据。数据取自全球历史气候网络 [Menne 等人, 2012]。

5.4.1 模型设定

令 $T(s_i)$ 表示气象站位置 s_i 处的平均气温。我们设定：

$$T(s_i) = \beta_0 + \beta_1\, z_i + u(s_i) + \varepsilon_i, \tag{15}$$

其中， β_0 为截距（海平面处的基准气温）， z_i = 海拔$_i$/1000 表示以千米为单位的海拔， $u(s)$ 是一个具有 Matérn 协方差、通过随机偏微分方程表示近似的高斯随机场，而 $\varepsilon_i \sim \mathrm{N}\,(0,\sigma_\varepsilon^2)$ 是测量误差。根据标准大气直减率（约为每 1000 米 6° 摄氏度），我们预期 $\beta_1 \approx -6$。

空间场 $u(s)$ 采用惩罚复杂度（PC）先验 [Simpson 等人, 2017] 用于实际范围 ρ 和边际标准差 σ_u：

$$\Pr(\rho < 30\ \mathrm{km}) = 0.05, \qquad \Pr(\sigma_u > 1^\circ\mathrm{C}) = 0.05. \tag{16}$$

5.4.2 实现

我们加载温度数据，并将坐标投影到 UTM 第40带（单位为千米）：

```
import numpy as np import pandas as pd import geopandas as gpd
from scipy importsparse from shapely.geometryimport Point from
pyinla import pyinla from pyinla.fmesher import (fm_mesh_2d, fm_
hexagon_lattice, spde2_pcmatern, spde_make_A, spde_grid_projector) #
坐标参考系（UTM 第40带，单位为千米） CRS_KM = "+proj=utm +zone
=40 +units=km +no_defs" # 加载温度数据 tavg = pd.read_csv("tavg_
month.csv") # 转换到以千米为单位的坐标参考系 geometry = [Point(lon,
lat) for lon, lat in zip(tavg["longitude"], tavg["latitude"])]tavg_gdf =
gpd.GeoDataFrame(tavg, geometry=geometry, crs="EPSG:4326") tavg_
km = tavg_gdf.to_crs(CRS_KM) coords_km =np.column_stack([tavg_
km.geometry.x, tavg_km.geometry.y])
```

```
print(f"气象站：{len(tavg)}") print(f"温度范围：{tavg[
'X202401'].min():.1f} 至 " f"{tavg['X202401'].max():.1f} 摄氏度")
```

```
气象站：55 温度范围：9.0 至 26.1
摄氏度
```

我们在缓冲研究区域内使用六边形格点构建三角网格。网格向数据范围外延伸 500 km，以避免边界效应：

```
# 在缓冲边界内创建六边形格网 boundary_buffered =
boundary.buffer(100) boundary_coords =np.array(boundary_
buffered.exterior.coords) hexpoints =fm_hexagon_
lattice(boundary_coords, edge_len=50)

# 构建三角网格 mesh =fm_mesh_2d(loc=hexpoints, offset=500,
max_edge=200) print(f"网格：{mesh.n} 个顶点，{mesh.n_
triangle} 个三角形")
```

```
网格：1792 个顶点，3499 个三角形
```

图 8 展示了生成的网格以及 55 个气象站位置。

带有 PC 先验和观测投影矩阵的 SPDE 模型定义如下：

```
# 使用 PC 先验定义 SPDE 模型 spde = spde2_pcmatern( mesh=
mesh, prior_range=(30,0.05), # P(range < 30 km) = 0.05 prior_
sigma=(1, 0.05) # P(sigma > 1 摄氏度) = 0.05 ) # 创建从网格到观
测位置的投影矩阵 A = spde_make_A(mesh=mesh, loc=coords_
km)
```

模型拟合采用基于字典的模型设定：

```
# 准备数据 n_观测数 =len(tavg) dataf =pd.DataFrame({ 'X202401': t
avg['X202401'].values, 'elevation_km':tavg['elevation'].values /1000,
'spatial': np.arange(n_观测数) }) # 定义模型：温度 ~ 截距 + 海拔 + 空
间效应模型 ={ 'response': 'X202401', 'fixed': ['1', 'elevation_km'],
'random': [{'id':'spatial','model': spde, 'A.local':A}]}
```

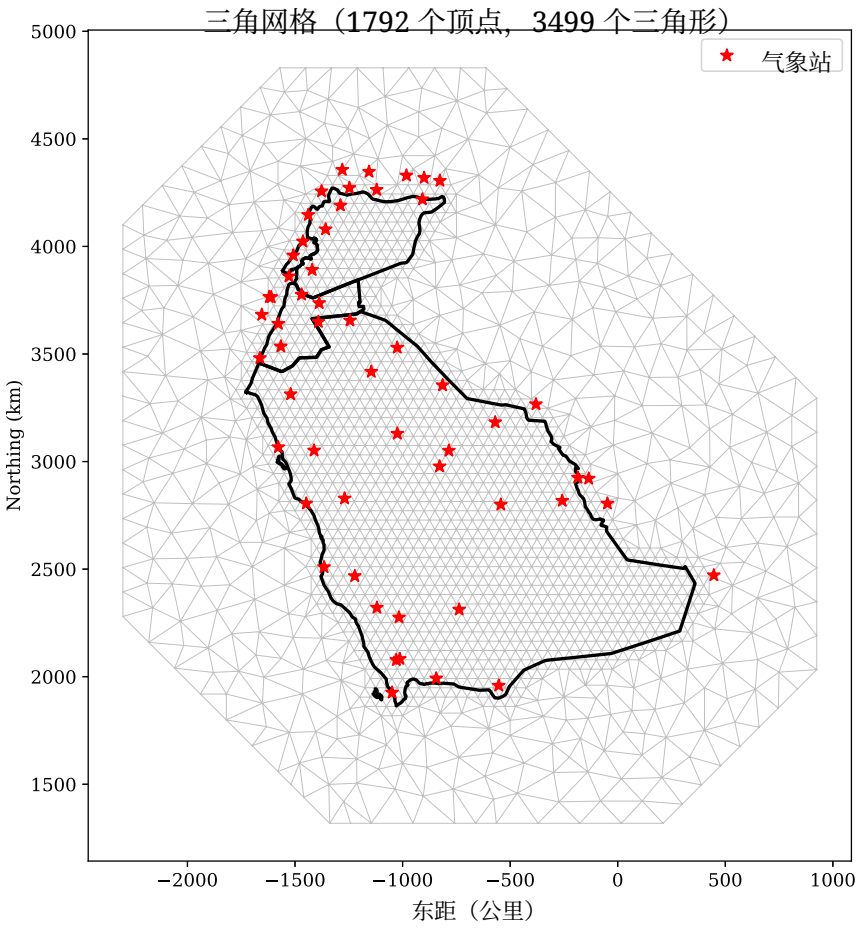

图8：用于逼近 Matérn 场的随机偏微分方程三角网格。该网格包含 1792 个顶点和 3499 个三角形，在研究区域边界附近分辨率较高，在外部扩展区域则采用较粗的三角形。红色星号表示气象站位置。

```
# 拟合模型结果 = pyinla(model=model,family='高斯', data=dataf)
print(result.summary_固定效应[['均值', '标准差', '0.025分位数','0.975分位数'
]])
```

```
                mean     sd  0.025分位数  0.975分位数
（截距）        20.12   3.87       12.10       28.14
海拔    _km    -5.74   0.37       -6.47       -5.01
```

估计的直减率为每千米 $-5.74°$ 摄氏度（95%置信区间：$[-6.47, -5.01]$），与大气预期一致。截距 $20.12°$ 摄氏度表示海平面处一月气温的预测值；其较大的不确定性（标准差 $= 3.87$）反映了空间相关结构吸收了大部分截距变异。

5.4.3 空间预测

我们在规则的 10公里网格（230 × 280 个点）上生成预测值，并纳入 ETOPO 2022数字高程数据 [美国国家海洋和大气管理局国家环境信息中心， 2022]:

```
# 创建预测网格投影器 projGrid =spde_网格_投影器( mesh=网格, xlim=(-1800,
500), ylim=(1700, 4500), dims=(230, 280)) # 预测数据：缺失值响应变量，来自
ETOPO 2022 的网格海拔 n_预测 = len(projGrid.lattice) 预测数据框 =
pd.DataFrame({ 'X202401': np.full(n_预测, np.nan), '海拔_千米': etopoll / 1000, '空
间': np.arange(n_观测, n_观测 + n_预测) }) 数据_合并 = pd.concat([数据框, 预测数
据框]，忽略_索引=True) # 合并观测和预测投影矩阵 A_合并 = sparse.vstack([A,
projGrid.proj]) # 使用合并投影矩阵的相同模型结构 模型_预测 = { '响应变量':
'X202401', '固定效应': ['1', '海拔_千米'],'随机效应': [{'id': '空间', '模型': spde,
'A.local': A_合并}]} 结果_预测 =pyinla(模型=模型_预测, 族='高斯', 数据=数据_合并) #
提取预测值（观测值之后的行） i_预测 = range(n_观测,n_观测 + n_预测) 预测_均值
 = 结果 _ 预测 .summary_fitted_values.iloc[i_ 预测 ][' 均值 '].values 预测 _ 标准差 =
结果_预测.summary_fitted_values.iloc[i_预测]['标准差'].values 温度范围：-2.5 至
27.7 摄氏度 不确定性范围：0.38 至 1.46 摄氏度
```

图9 展示了预测温度面及其相关不确定性。观测位置附近的预测不确定性最低，而在数据稀疏区域则会增加，尤其是在沙特阿拉伯内陆地区，最近的气象站相距数百千米。

该模型在拟合温度与观测温度方面达到 $R^2 = 0.979$ （图 10），预测值范围从 −2.5°摄氏度（山区）到 27.7°摄氏度（低海拔沙漠地区）。

复现材料。完整的复现脚本包含从全球历史气候网络获取数据以及所有中间处理步骤，可作为补充材料获取（`spde.py`）。

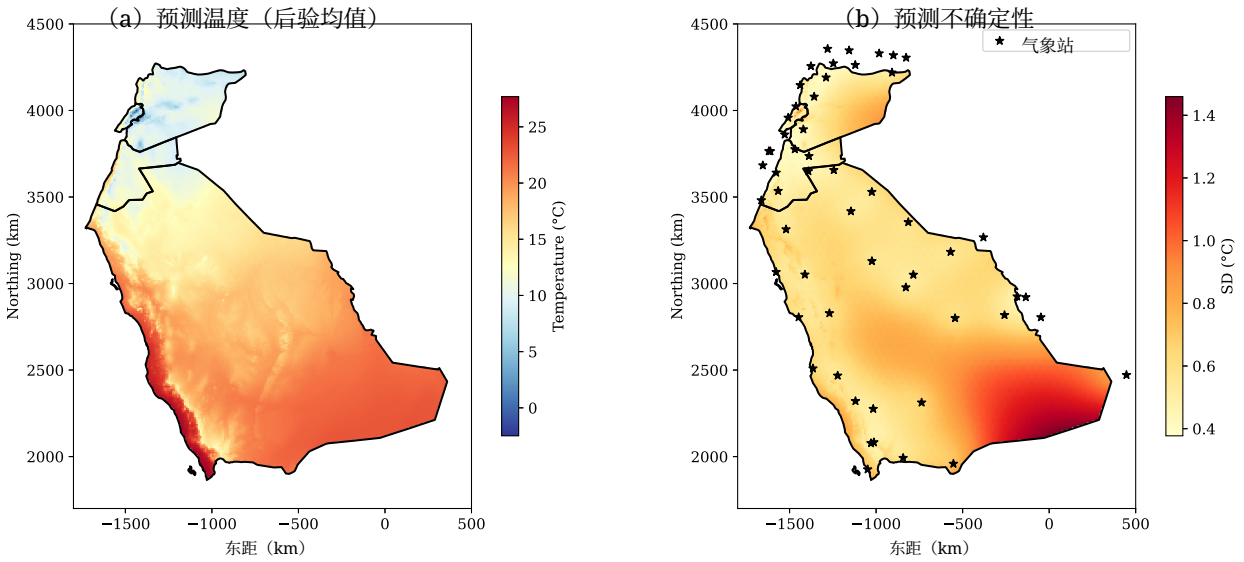

图9：2024年1月的空间温度预测。左：温度后验均值，显示了从温暖低地（25–28°摄氏度）到凉爽高地（低于10°摄氏度）的海拔驱动梯度。右：预测不确定性（后验标准差），在气象站（黑色星号）附近最低，在数据稀疏区域最高。

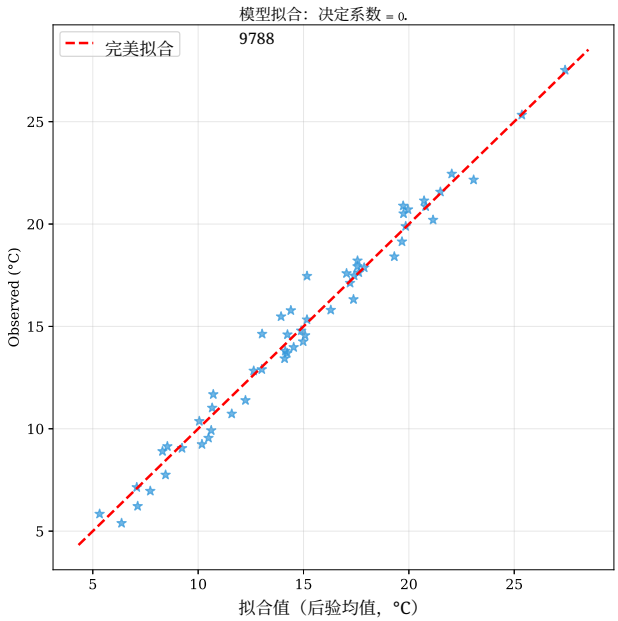

图10：观测温度与拟合温度对比，显示模型拟合效果极佳（R^2 = 0.979）。虚线表示完美一致。

6 讨论与结论

本文介绍了 PyINLA，这是一个原生 Python 包，可将基于INLA的贝叶斯推断引入 Python 生态系统。通过直接访问编译后的 INLA 计算引擎

通过 Python 风格 API，PyINLA 消除了跨语言工作流程带来的阻碍，同时保留了 INLA 在潜在高斯模型领域成为标准工具所依赖的计算效率和统计准确性。

第 5 节的示例展示了 PyINLA 在多种模型结构中的应用：从用于体育预测的可交换随机效应，到用于疾病制图的空间结构化效应、用于地统计插值的连续空间场，以及用于时间序列预测的时间分量。在这些应用中，后验估计与马尔可夫链蒙特卡罗基准结果高度吻合，同时计算速度提升了 100–200×倍，且确定性输出简化了测试与部署。

需要承认若干局限。PyINLA 仅适用于潜在高斯模型；需要非高斯潜在结构或 INLA 支持范围之外的似然函数的应用，则需要使用通用工具。需要直接访问完整联合后验的应用也可能受益于马尔可夫链蒙特卡罗方法，不过 PyINLA 可为派生量提供后验抽样。

开发工作正沿着多个方向持续推进，包括扩展模型支持、通过 sTiles 库实现 GPU 加速 [Abdul Fattah 等、2025年 b、以及], 更深入地融入 Python 机器学习工作流，并在 https://pyinla.org扩充文档。未来版本还将允许用户定义自定义潜在模型和似然函数，使 PyINLA 的功能超越内置组件。

PyINLA 使基于INLA的贝叶斯推断成为 Python 科学计算生态系统的原生组成部分。通过直接提供针对潜在高斯模型的快速、确定性推断，PyINLA 为与现代机器学习和人工智能工作流更紧密地集成打开了大门；在这些工作流中，严谨的不确定性量化正日益受到重视。我们预计，原生支持 Python 将使 INLA 方法惠及新的应用领域和更大规模的问题。

参考文献

Esmail Abdul Fattah。《基于稠密矩阵的近似贝叶斯推断及使用 INLA 的新功能》。博士论文，阿卜杜拉国王科技大学（KAUST），2023年。

Esmail Abdul Fattah、Janet Van Niekerk 和 Håvard Rue。智能梯度——一种改进梯度估计的自适应技术。《数据科学基础》，4(1):123–136，2022年。

Esmail Abdul Fattah、Hatem Ltaief、Havard Rue 和 David Keyes。使用 sTiles 对结构化矩阵进行 GPU 加速并行选定求逆。《arXiv 预印本 arXiv:2504.19171》，2025年 a。

Esmail Abdul Fattah、Hatem Ltaief、Håvard Rue 和 David Keyes。sTiles：用于结构化矩阵稀疏分解的加速计算框架。载于 *ISC* 高性能计算 *2025* 研究论文集（第 *40* 届国际会议），第 1–14 页。Prometeus GmbH，2025年 b。

Esmail Abdul Fattah、Janet Van Niekerk 和 Håvard Rue。Inla+：使用高性能计算对非稀疏模型进行近似贝叶斯推断。《统计与计算》，35(1):17，2025年 c。Danilo Alvares、Janet Van Niekerk、Elias Teixeira Krainski、Håvard Rue 和 Denis Rustand。使用 inla 进行贝叶斯生存分析。《医学统计》，43(20):3975–4010，2024年。doi：10.1002/sim.10160。Fabian E Bachl、Finn Lindgren、David L Borchers 和 Janine B Illian。inlabru：用于基于生态调查数据进行贝叶斯空间建模的 R软件包。《生态学与进化方法》，10(6):760–766，2019年。Eli Bingham、Jonathan P Chen、Martin Jankowiak、Fritz Obermeyer、Neeraj Pradhan、Theofanis Karaletsos、Rohit Singh、Paul Szerlip、Paul Horsfall 和 Noah D Goodman。Pyro：深度通用概率编程。《机器学习研究杂志》，20(28):1–6，2019年。Roger Bivand、Virgilio Gómez-Rubio 和 Håvard Rue。使用 r-inla 进行空间数据分析及其若干扩展。《统计软件杂志》，63:1–31，2015年。Marta Blangiardo、Michela Cameletti、Gianluca Baio 和 Håvard Rue。使用 r-inla 的空间及时空模型。《空间与时空流行病学》，4:33–49，2013年。

David M Blei、Alp Kucukelbir 和 Jon DMcAuliffe。《变分推断：面向统计学家的综述》。*Journal of the American Statistical Association*，112(518):859–877，2017年。

David Bolin 和 Alexandre B. Simas。*rSPDE*：分数阶随机偏微分方程的有理逼近，2023年。网址：https://CRAN.R-project.org/package=rSPDE。R软件包版本 2.3.3。

Steve Brooks、Andrew Gelman、Galin Jones 和 Xiao-Li Meng。《马尔可夫链蒙特卡洛手册》。CRC出版社，2011年。Bob Carpenter、Andrew Gelman、Matthew D Hoffman、Daniel Lee、Ben Goodrich、Michael Be- tancourt、Marcus Brubaker、Jiqiang Guo、Peter Li 和 Allen Riddell。Stan：一种概率编程语言。《统计软件杂志》，76：1–32，2017年。Ziming Chen、Jeffrey S Berger、Lana A Castellucci、Michael Farkouh、Ewan C Goligher、Erinn M Hade、Beverley J Hunt、Lucy Z Kornblith、Patrick R Lawler、Eric S Leifer 等。临床试验贝叶斯分析计算算法的比较。《临床试验》，21(6)：689–700，2024年。Cristian Chiuchiolo、Janet van Niekerk 和 Håvard Rue。使用 R-INLA 对潜在高斯模型进行联合后验推断。Journal of Statistical Computation and Simulation，2021年。doi：https://doi.org/10.1080/00949655.2022.2117813。David Clayton 和 John Kaldor。用于疾病制图的年龄标化相对风险经验贝叶斯估计。《Biometrics》，第671–681页，1987年。Noel Cressie 和 Christopher K Wikle。《时空数据统计学》。John Wiley & Sons，2011年。

J Darkwah。简单线性模型和广义线性模型的贝叶斯推断：比较 INLA 与马尔可夫链蒙特卡罗。载于《植物化学、纺织与可再生能源研究促进产业增长》，第62–67页。CRC出版社，2022年。Tom De Smedt、Koen Simons、An Van Nieuwenhuyse 和 Geert Molenberghs。使用贝叶斯层次模型进行疾病制图时比较马尔可夫链蒙特卡罗与 INLA。《公共卫生档案》，73：1，2015年。Joshua V Dillon、Ian Langmore、Dustin Tran、Eugene Brevdo、Srinivas Vasudevan、Dave Moore、Brian Patton、Alex Alemi、Matt Hoffman 和 Rif A Saurous。TensorFlow概率分布。arXiv 预印本 arXiv:1711.10604，2017年。Guido Fioravanti、Sara Martino、Michela Cameletti 和 Andrea Toreti。使用 INLA 和 SPDE 方法插值气候变量。《国际气候学杂志》，43(14)：6866–6886，2023年。Lisa Gaedke-Merzhäuser、Elias Krainski、Radim Janalik、Håvard Rue 和 Olaf Schenk。面向大规模时空贝叶斯建模的积分嵌套拉普拉斯近似。《SIAM科学计算杂志》，46(4)：B448–B473，2024年。Matthew J. Heaton、Abhirup Datta、Andrew O. Finley、Reinhard Furrer、Joseph Guinness、Rajarshi Guhaniyogi、Florian Gerber、Robert B. Gramacy、Dorit Hammerling、Matthias Katzfuss、Finn Lindgren、Douglas W. Nychka、Furong Sun 和 Andrew Zammit-Mangion。分析大型空间数据方法之间的案例研究竞赛。《农业、生物与环境统计学杂志》，(24)：398–425，2019年。Matthew D Hoffman、Andrew Gelman 等。无转向采样器：自适应设定哈密顿蒙特卡洛中的路径长度。《J. Mach. Learn. Res.》，15(1)：1593–1623，2014年。

Gary M Koop。《贝叶斯计量经济学》。约翰·威立父子公司，2003年。

ET Krainski、F Lindgren 和 H Rue。Inlaspacetime：使用“inla”的空间和时空模型，2025年。

Andrew B Lawson。《贝叶斯疾病制图：空间流行病学中的层次建模》。Chapman and Hall/CRC，2018年。Finn Lindgren。fmesher：三角网格及相关几何工具。*R*软件包版本 *0.1*，5，2023年。Finn Lindgren 和 Håvard Rue。使用 r-inla 进行贝叶斯空间建模。《统计软件杂志》，63：1–25，2015年。Finn Lindgren、Håvard Rue 和 Johan Lindström。高斯场与高斯马尔可夫随机场之间的明确联系：随机偏微分方程方法。《英国皇家统计学会杂志*B*辑：统计方法学》，73(4)：423–498，2011年。

Finn Lindgren、Fabian Bachl、Janine Illian、Man Ho Suen、Håvard Rue 和 Andrew E Seaton。inlabru：用于拟合带非线性预测变量的潜在高斯模型的软件。*arXiv*预印本 *arXiv:2407.00791*，2024年。

Thiago G Martins、Daniel Simpson、Finn Lindgren 和 Håvard Rue。INLA 贝叶斯计算：新功能。《计算统计与数据分析》，67:68–83，2013年。Matthew J Menne、Imke Durre、Russell S Vose、Byron E Gleason 和 Tamara G Houston。全球历史气候网络日值数据库概述。《大气与海洋技术杂志》，29(7):897–910，2012年。Mark H Myer 和 John M Johnston。西尼罗河病毒的时空贝叶斯建模：利用局部尺度预测变量识别蚊子的感染风险。《整体环境科学》，650:2818–2829，2019年。João Pedro Coli de Souza Monteneri Nacinben 和 Márcio Laurini。通过积分嵌套拉普拉斯近似进行多变量随机波动率建模：一种多因子扩展。《计量经济学》，12(1):5，2024年。Van Niekerk 和 Håvard Rue。对拉普拉斯方法的低秩变分贝叶斯校正。《机器学习研究杂志》，(25):1–25，2024年。美国国家海洋和大气管理局国家环境信息中心。ETOPO 2022 15角秒全球地形起伏模型，2022年。Andrea Riebler、Sigrunn H Sørbye、Daniel Simpson 和 Håvard Rue。一种直观的贝叶斯空间模型，用于考虑尺度调整的疾病制图。《医学研究统计方法》，25 (4):1145–1165，2016年。Havard Rue 和 Leonhard Held。高斯马尔可夫随机场：理论与应用。Chapman and Hall/CRC，2005年。Havard Rue、Sara Martino 和 Nicolas Chopin。使用积分嵌套拉普拉斯近似对潜在高斯模型进行近似贝叶斯推断。《英国皇家统计学会杂志：*B*辑（统计方法学）》，71(2):319–392，2009年。Havard Rue、Andrea Riebler、Sigrunn H. Sørbye、Janine B. Illian、Daniel P. Simpson 和 Finn K. Lindgren。INLA 贝叶斯计算：综述。《统计学及其应用年度评论》，4:395–421，2017年。Denis Rustand、Janet van Niekerk、Elias Teixeira Krainski 和 Håvard Rue。使用 R 包 inlajoint 对多变量纵向结局和生存结局进行联合建模。*arXiv*预印本 *arXiv:2402.08335*，2024年。John Salvatier、Thomas V Wiecki 和 Christopher Fonnesbeck。使用 PyMC3 在 Python 中进行概率编程。《*PeerJ*计算机科学》，2:e55，2016年。Fiona M Seaton、Susan G Jarvis 和 Peter A Henrys。使用 R-INLA 进行物种分布建模的时空数据整合。*Methods inEcologyand Evolution*，15(7):1221–1232，2024年。Daniel Simpson、Håvard Rue、Andrea Riebler、Thiago G Martins 和 Sigrunn H Sørbye。惩罚模型组件复杂度：构建先验分布的一种原则性、实用方法。《统计科学》，32(1):1–28，2017年。

Oskar Triebe、Hansika Hewamalage、Polina Pilyugina、Nikolay Laptev、Christoph Bergmeir 和 Ram Rajagopal。《Neuralprophet：大规模可解释预测》。*arXiv* 预印本 *arXiv:2111.15397*，2021。Janet Van Niekerk、Haakon Bakka、Håvard Rue 和 Olaf Schenk。《使用 R 中的 INLA 软件包进行贝叶斯建模的新前沿》。《统计软件杂志》，100:1–28，2021。Janet Van Niekerk、Elias Krainski、Denis Rustand 和 Håvard Rue。《使用 INLA 进行贝叶斯推断的新途径》。《计算统计与数据分析》，181:107692，2023。Cheng Zhang、Judith Bütepage、Hedvig Kjellström 和 Stephan Mandt。《变分推断的进展》。*IEEE* 模式分析与机器智能汇刊，41(8):2008–2026，2019。

A 边际后验实用函数

模型拟合后，PyINLA 会为每个模型组成部分（固定效应、超参数、随机效应和线性预测值）返回边际后验分布 $p(\theta_i \mid \mathbf{y})$。每个边缘分布都存储为由 $(n \times 2)$ **NumPy** 数组表示的 $(x,$ 密度$)$ 对。以下实用函数可让用户将这些边缘分布作为连续分布进行处理。

A.1 密度评估

函数 `dmarginal(x, marginal)` 使用单调分段三次厄米特插值（PCHIP）在一个或多个点处评估后验密度：

$$d(x) = p(\theta = x \mid \mathbf{y}). \tag{17}$$

对于支撑集之外的点，该函数返回零。

A.2 累积分布

函数`pmarginal(x, marginal)` 计算累积概率：

$$F(x) = P(\theta \le x \mid \mathbf{y}) = \int_{-\infty}^{x} p(\theta \mid \mathbf{y})\, d\theta. \tag{18}$$

这对于后验假设评估很有用。例如，对于回归系数 β，量 $1 - F(0)$ 给出了 β 为正的后验概率。

A.3 分位数函数

函数`qmarginal(p, marginal)` 返回逆累积分布函数：

$$Q(p) = F^{-1}(p) = \inf\{x : F(x) \ge p\}. \tag{19}$$

等尾可信区间可表示为 $[Q(\alpha/2) \;\; Q(1 - \alpha/2)]$。

A.4 变换

函数 `tmarginal(g, marginal)` 通过单调函数 g对边缘分布进行变换，并通过变量变换公式返回 $Y = g(\theta)$ 的边缘分布：

$$p_Y(y) = p_\theta(g^{-1}(y)) \cdot \left| \frac{d}{dy} g^{-1}(y) \right|. \tag{20}$$

一种常见用途是将对数尺度参数转换为自然尺度（例如，通过 $g(\theta) = \exp(-\theta/2)$ 将对数精度转换为标准差），或者通过 $g(\theta) = \exp(\theta)$ 根据对数优势计算优势比。

A.5 函数的期望值

函数`emarginal(g, marginal)` 计算：

$$\mathbb{E}[g(\theta) \mid \mathbf{y}] = \int g(\theta) \cdot p(\theta \mid \mathbf{y})\, d\theta, \tag{21}$$

使用离散边缘分布进行数值求积。

A.6 最高后验密度区间

函数 `hpdmarginal(level, marginal)` 返回包含概率质量 $1-\alpha$的最短区间：

$$\mathrm{HPD}_{1-\alpha} = \{x : p(x \mid \mathbf{y}) \geq k_\alpha\}, \tag{22}$$

其中， k_α的选取满足 $P(\theta \in \mathrm{HPD}_{1-\alpha}) = 1-\alpha$。对于偏斜的后验分布，在相同覆盖水平下，最高后验密度区间比等尾区间更短。

A.7 汇总统计量

函数 `zmarginal(marginal)` 返回一个字典，其中包含后验均值、标准差、中位数以及 2.5%/97.5% 分位数，这些数值是根据离散边缘分布表示计算得出的。

A.8 随机抽样与众数

函数 `rmarginal(n, marginal)` 使用逆变换抽样生成 n 个随机样本： $X = F^{-1}(U)$，其中 $U \sim$ 服从均匀分布$(0,1)$。函数 `mmarginal(marginal)` 返回后验众数（最大后验估计）：

$$\hat{\theta}_{\mathrm{MAP}} = \arg\max_{\theta}\ p(\theta \mid \mathbf{y}). \tag{23}$$

A.9 用法示例

以下展示了拟合模型后典型的工作流程：

```
import pyinla import numpy as np # 获取固定效应的边际分布 marg = result.marginals_fixed["x"]# 汇总统计量 summary = pyinla.zmarginal(marg) # 系数为正的后验概率 prob_positive = 1.0 -pyinla.pmarginal(0.0, marg) # 95% 最高后验密度区间
```

```
hpd95 =pyinla.hpdmarginal(0.95,
marg) # 将对数优势转换为优势比 marg_or
 = pyinla.tmarginal(np.exp, marg) # 派生
量的期望值 e_or =
pyinla.emarginal(np.exp, marg)
```