

STAT243 Lecture 9.3 Implementation of Simulation Studies

1 Computational Efficiency

1.1 并行计算与仿真研究 (Parallel Processing for Simulations)

- 仿真研究的特点

仿真 (simulation) 通常是高度可并行的任务 (embarrassingly parallel) :

- 每个重复试验 (replicate) 可以独立运行在不同的 CPU 核上
- 运行结束后再将结果汇总
- 理论上, 运行速度应与处理器数量近似线性增长

- C/C++ 编译优化

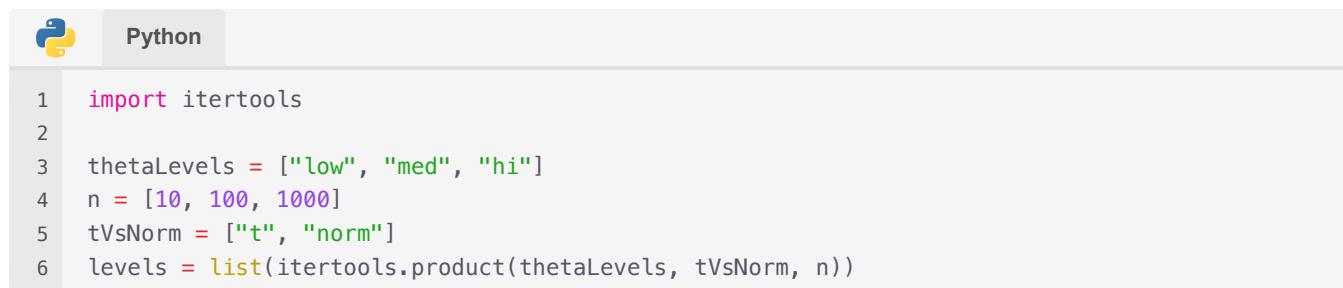
若仿真中包含大量循环或计算密集型部分, 可考虑:

- 使用 C/C++ 编写核心函数
- 在 Python 中通过绑定 (linking) 调用已编译代码
- 这种方式能显著提升性能
- 虽然本课程不涉及, 但这是常见的高效策略

1.2 Python 中的笛卡尔积工具 (Using `itertools.product`)

在仿真设计中, 我们常需要遍历多个参数的所有组合。

Python 提供 `itertools.product` 生成参数笛卡尔积:



```
Python
1 import itertools
2
3 thetaLevels = ["low", "med", "hi"]
4 n = [10, 100, 1000]
5 tVsNorm = ["t", "norm"]
6 levels = list(itertools.product(thetaLevels, tVsNorm, n))
```

结果:



```
Python
1 [("low", "t", 10), ("low", "t", 100), ("low", "t", 1000),
2  ("low", "norm", 10), ..., ("hi", "norm", 1000)]
```

应用场景:

用于生成实验条件、超参数组合或多场景仿真设定。

2 Analysis and Reporting

2.1 报告结果的方式 (Reporting Simulation Results)

- 表格与图形的选择

- 仿真结果可用表格展示（如均值、方差、覆盖率等）
 - 若有多个情境（scenarios），图形可能更直观
 - 但并非所有情况图形都优于表格，应具体分析
- **多维结果展示**
- 在多输入变量情境下：
- R 中可使用 `ggplot2::facet_wrap()` 绘制 trellis plots
 - Python 中可用 `pandas.plot`, `seaborn`, `plotly` 等工具实现类似功能
-

2.2 随机性与可复现性 (Reproducibility in Simulations)

- **固定随机种子 (set the seed)**

在实验开始前设置随机数种子，以确保结果可复现



Python

```
1 import numpy as np
2 np.random.seed(123)
```

- 在保存中间结果时，也应保存当前随机数状态
- 对于 MCMC 等逐步更新算法，这样可从中断处继续运行

- **代码与数据的可公开性 (Code and Data Transparency)**

为促进可复现性，应将：

- 仿真代码
- （若可行）仿真数据
上传至：
 - GitHub
 - 个人网站
 - 期刊附录 (Supplementary Materials)

他人应能完全再现结果，包括随机数生成与种子设置。

2.3 学术期刊对可复现性的要求 (Reproducibility Standards)

美国统计学会 (ASA) 对其期刊的计算型论文提出了如下要求：

“ASA 强烈建议作者提交与论文相关的数据集、代码、程序或附录。这些材料有助于推动可复现研究的专业标准。

- 使用任何数据集时，应完整记录其来源并在网上附录中提供。
- 出于安全或保密原因，可向编辑申请豁免。
- 若研究依赖特定代码实现，应尽可能公开该代码。

对于基于计算结果的论文，应提供足够信息以便读者评估结果质量，包括：

- 结果的估计精度
- 伪随机数生成器的描述
- 所用数值算法
- 编程语言与主要软件组件。”