AI 时代的实证研究基础

连享会 **2026** 暑期班 • 初级班

连玉君（中山大学）

目录

1 从顶刊复现项目开始：实证研究的完整流程

本讲要点

- 实证研究的基本链条：问题、数据、模型、结果、论文；
- 复现项目的文件夹结构、`README`、主脚本和输出文件；
- 原始数据、中间数据、分析样本、图表结果之间的关系；
- 如何识别一个复现包的主程序、数据处理脚本和结果输出脚本；
- 如何用 AI 辅助阅读 replication package，而不是让 AI 替代研究判断；
- Lane (2025) 中政策行业、产业代码、工业面板、IO 表和贸易数据之间的映射关系；
- 从复现文档中学习研究设计、数据处理和结果组织。

本讲配套材料

- 本讲用到的 **skills**：`skills/core/01-paper-context` • `skills/core/02-paper-strategy` • `core/03-replication-navigator` • `skills/core/06-repro-logger`。skill 是什么、怎么安装、怎么在不同 agent(Claude Code / Codex / OpenCode) 里触发，见《AI 工作流与 skills 上手》。
- 配套阅读笔记 (可在书内浏览，也可下载)：借助 AI 对主案例 Lane (2025) 生成的四份角度笔记——整体介绍、实证方法与识别、复现材料导览、迷你文献定位。
- 本讲为解读型章节，以提示词与 **skills** 调用为主，绝大多数任务在网页版 AI 助手里即可完成，不涉及自研 Stata/Python 代码。

想运行完整 Lane 复现包？请用 GitHub Desktop 克隆

Lane 的复现包含 Git LFS 管理的大型 `.dta` 文件。要在本地运行，请优先用 GitHub Desktop 克隆仓库，不要下载网页 ZIP；安装、克隆与更新步骤见课件使用说明和GitHub Desktop 教程。若数据仍缺失，再按附录 C从 Dataverse 补齐。

1.1 实证研究的完整流程：先看清全局

在动手复现一篇顶刊之前，先花点时间看清：一项实证研究，到底是怎么一步步做出来的。有了这张全局图，你才明白后面复现顶刊，究竟在练什么。

1.1.1 研究问题从哪里来：现实观察与文献阅读

好的实证研究，往往不是从方法开始的，而是从一个真问题开始。问题意识通常来自两个地方：对现实的观察，和对文献的阅读。更重要的是，问题不是一次想好的，而是在不断了解制度和数据的过程中逐步长出来、逐步修正的。举两个例子 (有关这两个例子的详情，参见连玉君，2026，因果推断与政策评估，chapter 01)。

例一：政府引导基金。研究者最初可能只是拿到一份基金数据，自然的第一反应，是把它当成一种带政府背景的风险投资，去问：它是否促进了被投企业的创新、融资或成长？这条路能走，但未必抓住要害。等到进一步读地方政府的设立文件、管理办法和新闻报道，会发现它们反复强调本地优势产业、产业链补短板、供应链安全——于是问题可能升级为：政府引导基金是否推动了地方产业链协同与供应链韧性？同一份数据，问题却因为对政策理解的深入而变了。

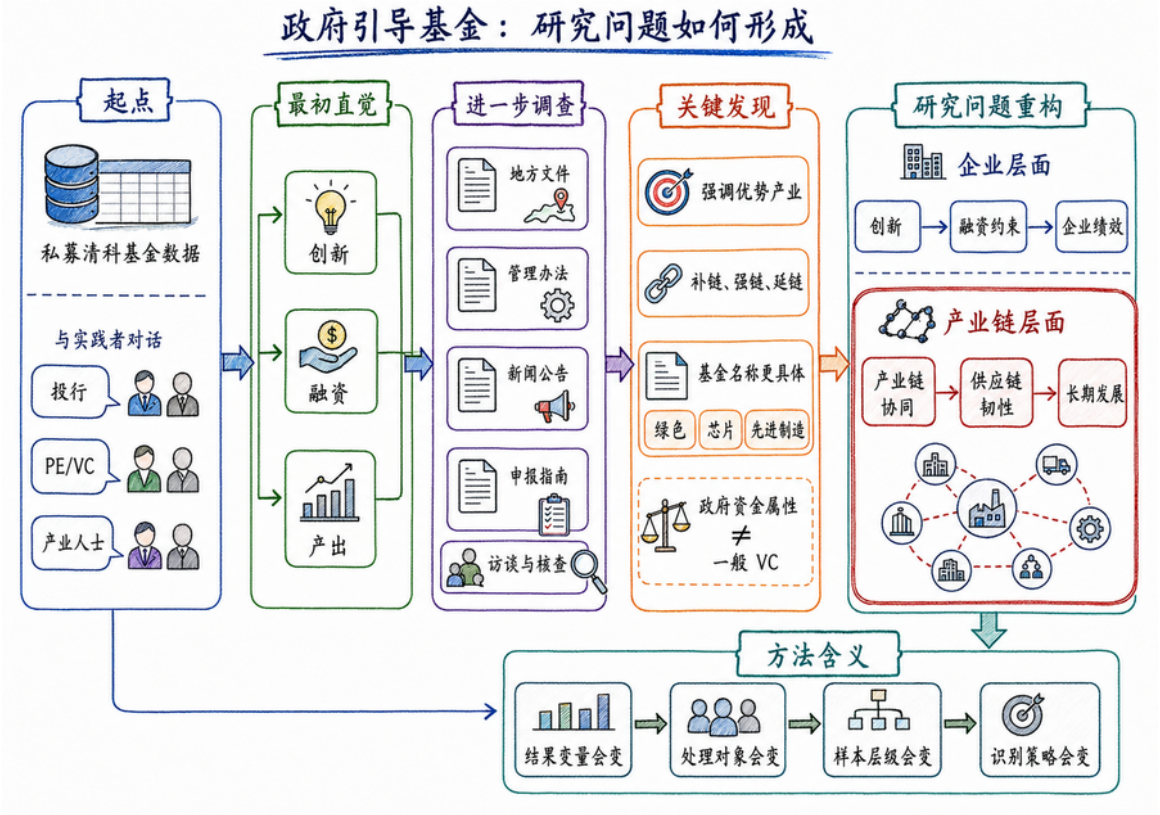

图 1.1: 政府引导基金：研究问题如何形成

例二：韩国重化工业政策。 1970 年代，韩国政府在特定的安全与工业化背景下，集中扶持钢铁、造船、石化等六大产业。这段真实的历史，引出一个被经济学界争论了近半个世纪的问题：这种由政府定向扶持特定产业的产业政策，到底有没有用？这，正是本讲主案例 Lane (2025) 要回答的问题。

可见，问题意识来自把现实和文献先读厚、再读薄：读厚，是把制度细节、数据口径、已有研究都了解清楚；读薄，是从中提炼出一个别人还没讲清、而你有办法回答的问题。

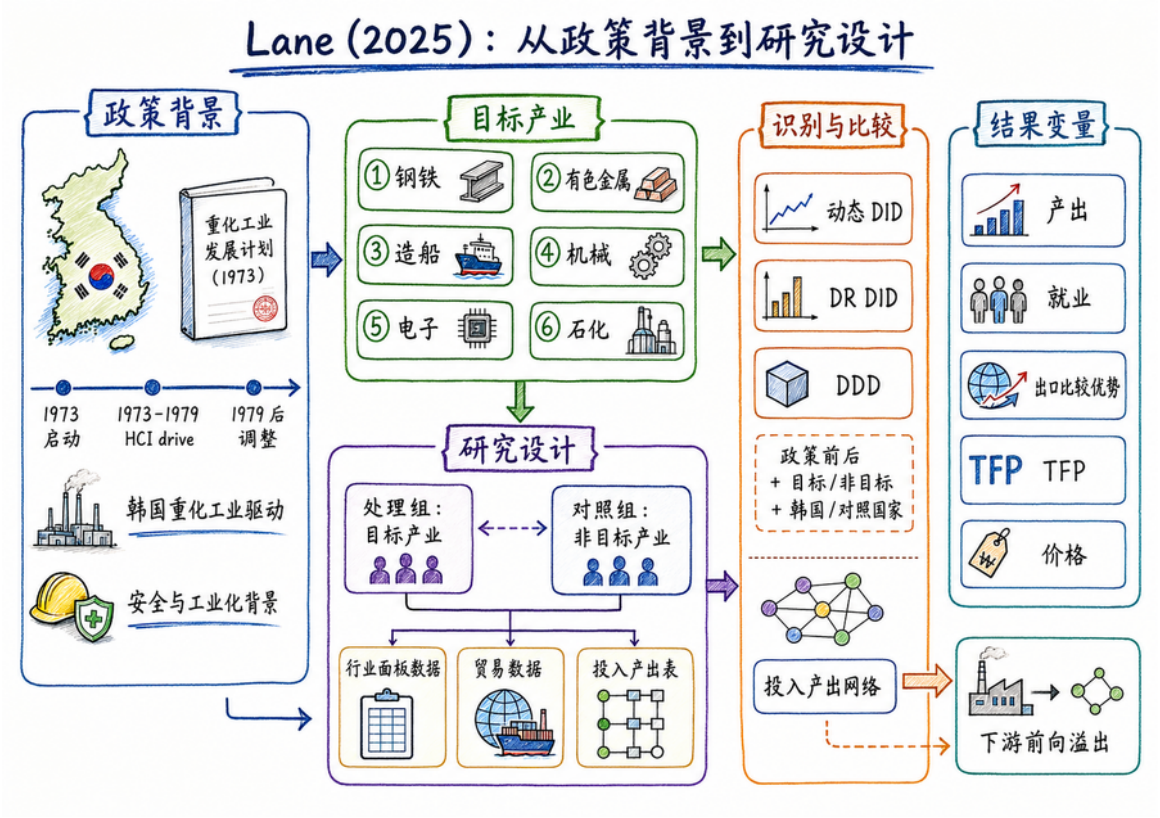

图 1.2: Lane (2025)：从政策背景到研究设计

1.1.2 厘清问题之后：研究设计在做什么

有了问题，下一步是设计一个能回答它的比较。这一步要回答几个朴素但关键的问题：

- 和什么比？也就是，如果没有这项政策或冲击，本来会是什么样 (这叫反事实)；
- 谁受影响、谁没受影响，这条边界从哪里来；
- 政策或冲击什么时候真正改变了对象面临的处境；
- 研究对象怎么落到数据上——政策可能作用于产业，而你手上的数据是企业或地区。

这些判断，决定了后面用什么数据、构造什么变量、选什么方法。所以方法不是研究的起点，而是被问题和事实一步步逼出来的结果。至于这些比较具体怎么做 (如何构造对照、如何估计)，是本课程后面几讲要一步步学的内容，这里先不展开。

1.1.3 从数据到结果，再到论文：实证研究的基本链条

把上面这些串起来，一项实证研究无论多复杂，都可以还原成一条链：

研究问题 → 数据 → 模型 → 结果 → 论文

- 问题：想回答什么。它决定了后面一切——要什么数据、比较谁和谁、看什么结果。
- 数据：用什么材料回答。原始数据往往不能直接分析，要经过清洗、合并、构造，才变成能上模型的分析样本。
- 模型：用什么方式从数据里「读出」答案。它不是凭空选的，而是被问题和数据结构逼出来的。
- 结果：模型算出来的图和表。
- 论文：把上面四者组织成一个别人能看懂、能检验的叙述。

这条链最重要的一点是：五个环节是一个整体。你平时纠结的那些局部问题 (变量要不要取对数、样本量为什么变了、这张表用的什么方法)，其实都是这条链上的某一环。看懂了整条链，局部问题才有位置感。

1.1.4 研究设计是一门手艺：为什么从「临摹」顶刊入手

看清了这条链，一个自然的疑问是：既然如此，第一讲为什么不直接讲方法，而要去复现一篇论文？

因为研究设计是一门手艺，光靠听讲学不会，得靠临摹入门。学书法要先临帖，学画要先临摹——为的是在一笔一画里，体会整幅作品的谋篇布局。只把单个字、单个技术点练好 (取一次对数、算一次聚类标准误)，是写不出一幅完整作品的，也做不出一篇完整研究。

复现一篇规范的顶刊论文，就是实证研究里最好的临摹。它让你在一个真实、完整的作品里，一次性看全：问题怎么界定、研究怎么设计、数据怎么清洗、指标怎么选取、模型怎么设定、结果怎么解读、论文怎么组织。这些环节各自都有讲究，但只有放进一篇完整论文里，你才能看清它们如何彼此配合。

这，就是本讲从复现一篇顶刊开始的原因。接下来，我们请出主案例。

1.2 案例情境：请出主案例 Lane (2025)

上一节说到，复现一篇规范的顶刊论文，是快速看全实证研究各环节的最好方式。这一节就正式请出本讲的主案例，交代它是什么、我们打算怎么读它。

1.2.1 AI 让「读懂顶刊」的门槛降下来了

过去，读懂一篇顶刊的实证论文，往往要具备相当的方法基础，还要耐着性子啃完全文。现在，工作方式正在改变：你可以把论文和它的复现包交给 **AI** 与 **agent**，让它先帮你把背景、方法、数据、复现流程各自梳理一遍，你再带着具体问题去追问、去核对。这不是让 AI 替你做研究，而是让它把你从「不敢开始」的门口，快速领进门。

一个更具体的目标感受是：借助 **AI**，一个上午就能大致读懂三四篇顶刊，判断哪些值得精读、哪些方法可能迁移到自己的研究里。本讲的主案例，就用来把这个过程走一遍。

> **!** 一条贯穿全讲的底线
>
> AI 负责快速起草，你负责判断对错。研究问题、样本定义、变量构造、模型设定和结果解释，始终由研究者自己负责。后面每一个借助 AI 的环节，都会配一句：人工核对。这正是本课程反复强调的分工。

1.2.2 主案例：Lane (2025, QJE)

本讲的主案例是一篇产业政策领域的顶刊论文：

> Lane, N. (2025). Manufacturing Revolutions: Industrial Policy and Industrialization in South Korea. The Quarterly Journal of Economics, 140(3), 1683–1741. Link (rep), PDF, Google, Replication.

它研究韩国 1973–1979 年的重化工业驱动 (Heavy and Chemical Industry Drive, **HCI drive**) 这项产业政策，是否真的促进了目标产业的发展。之所以选它做主案例，有三个原因：一是它的复现包组织得非常规范，官方来源是 Harvard Dataverse，课程期间的完整副本位于 `examples/Lane_2025_QJE_paper_codes/`，是学习「一个专业复现项目长什么样」的好范本；二是它同时用到 **Stata** 和 **R**、涉及多源数据，正好训练我们阅读多语言、多数据的复现材料；三是它的政策背景清楚、故事完整，适合初学者建立「从政策事实到研究设计」的整体认知。若克隆后数据不完整，按课程数据集说明从 Dataverse 下载并放回同一目录。

> **⚠** 本讲怎么用这个案例
>
> 本讲把 Lane (2025) 定位为顶刊解读的演练场，不逐表复现。目标不是把论文的每张表都跑出来，而是让零基础的你，借助 AI 与 skills，在较短时间内读懂它的研究背景、主要贡献和实证策略，并学会把这套读法迁移到自己关心的论文上。真正想动手复现的同学，可参考复现材料导览，再从课程仓库中的完整副本按其指引进行。

1.2.3 关键手法：把「读一篇论文」切成几个任务

初学者用 AI 读论文，最容易犯的错误，是一上来就把整篇 **PDF** 丢给 **AI**，让它写一份大而全的读书笔记。结果往往是：每个方面都浅尝辄止，篇幅很长却抓不住重点，你也无从判断它写得对不对。

本讲主张一种更有效的做法——任务切割：把「读一篇论文」拆成几个角度，每个角度让 AI 单独生成一份短笔记。我们对 Lane (2025) 就切了四刀，形成四份配套笔记：

角度	回答什么	配套笔记
整体介绍	研究背景、问题、思路、主要结论——最快了解全貌	整体介绍
实证方法与识别	用了什么方法、识别策略、怎么处理内生性	实证方法与识别
复现材料导览	复现包怎么组织、主程序在哪、数据分几层	复现材料导览
迷你文献定位	站在哪些文献之间、如何找到相关高水平文献	迷你文献定位

这四份笔记是本讲的核心成果，每一份都给出了生成它的提示词和建议调用的 **skill**，你可以照着自己复现一遍。后面的小节会先讲清概念，再逐一演示怎么用 AI 把它们做出来。

1.3 概念讲解：一个复现项目的骨架

一个规范的复现项目，正是把上一节那条「问题 → 数据 → 模型 → 结果 → 论文」的链条，固化成了文件和脚本。这一节就来拆解它的骨架，方法是先讲人话、再给直觉、最后落到复现包的真实结构上。Lane 案例的逐项细节，放在复现材料导览 里，本节只讲通用的看法。

1.3.1 一个复现包长什么样：README、主脚本、输出

打开一个规范的复现包 (replication package)，你通常会看到这样几类东西：

- `README`：说明书。讲清楚这个包怎么组织、要什么软件、怎么一步步跑起来。读复现包，永远从 `README` 开始。
- 主脚本 **(**入口**)**：一个「驱动别人」的总程序。你运行的就是它，它再按顺序去调用一堆干具体活的子脚本。
- 子脚本：分工干活的——有的做数据处理，有的跑回归，有的出图出表。
- 数据目录与输出目录：前者放输入和中间数据，后者放最终生成的图表。

以 Lane (2025) 为例，整个复现由一个入口脚本 `master.R` 驱动，它会先跑 Stata 分析、再用 R 出图出表。目录的骨架大致是：

```
master.R          ← 唯一入口，你运行的就是它
config.yml        ← 配置：填你的 Stata 路径和版本
README.md         ← 说明书，先读它
code/             ← 全部脚本（Stata 分析 + R 出图出表）
data/             ← 数据（分多层，见下）
output/           ← 最终生成的图和表
```

看到一个陌生的复现包，先问自己四个问题——入口在哪、目录怎么分、数据分几层、能复现到什么程度。这四问的 Lane 版详细答案，见复现材料导览。

1.3.2 原始数据、中间数据、分析样本、图表结果之间的关系

「数据」在复现包里不是一堆文件，而是一条流水线：

原始数据 → 中间数据 → 分析样本 → 图表结果

- 原始/输入数据：最初拿到的材料 (如手工数字化的普查、贸易数据、政策文书)。它往往不能直接分析。
- 中间数据：清洗、合并、构造之后产生的过程性结果，通常由脚本运行时生成。
- 分析样本：真正喂给模型的那份数据——已经定好了样本范围、处理组对照组、核心变量。
- 图表结果：模型跑出来、写进论文的图和表。

理解这条链有一个很实际的好处：你能看懂一个复现包为什么要这样分目录。比如 Lane 的包，把输入数据、运行时生成的中间数据、预先算好的结果、最终输出分别放在不同目录里；甚至专门预存了一部分依赖非公开数据的中间结果，好让你缺了原始微观数据也能把图画出来。这些安排不是多余的，正是「原始 → 中间 → 样本 → 结果」这条链在文件系统里的投影。

1.3.3 怎么识别主程序、数据处理脚本和结果输出脚本

面对几十上百个脚本文件，怎么快速分清谁是谁？有几个通用窍门：

- 主程序：名字里常带 master、main、run，或以 0_ 这类序号前缀开头；它的内容大多是「按顺序调用别的脚本」。
- 数据处理脚本：读入原始数据、做清洗合并构造、写出中间数据；关键词常是 clean、build、prepare、merge。
- 结果输出脚本：读分析样本、跑回归或画图、把图表写进 output；关键词常是 figure、table、analysis、regress。

一个可靠的判断法则是：被别人反复调用的，是干活的子脚本；调用别人的，是主程序。顺着主程序一层层读下去，整个复现流程就展开了。在 Lane 案例里，这条线是 master.R → 0_master_run_analysis.do → 各 run_*_analysis.do →（R 出图出表脚本）；哪个分析对应哪个脚本，作者在 README 里给了对照表——读代码前先看这张表，能省大量时间。

1.3.4 Lane 案例：政策产业如何映射到数据

Lane 研究的 HCI drive，集中扶持钢铁、有色金属、造船、机械、电子、石化六大战略产业。一项「政策」要变成可以分析的数据，中间有一条清晰的映射链：

> 政策法案 (点名的产业)→ 产业分类代码 (KSIC)→ 行业面板 / 贸易数据 / 投入产出表

图 1.3 概括了从政策背景一步步走向研究设计的思路：左侧是政策的历史背景与时间线，中间是目标产业与数据的映射，右侧是识别策略与结果变量。

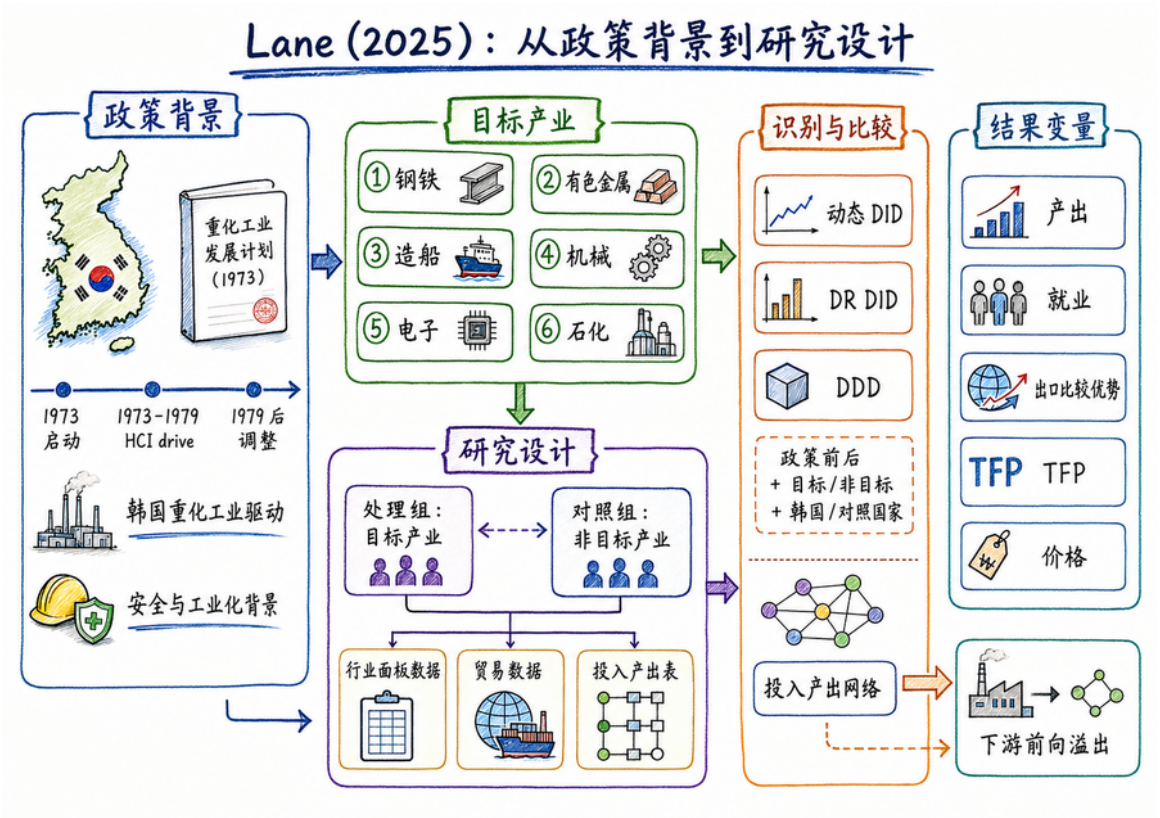

图 1.3: Lane (2025)：从政策背景到研究设计

这条链上的每一步，都对应复现包里的一类材料：

- 政策文件告诉我们哪些产业是目标 (处理组)、政策何时启动 (时间)；
- 制造业普查面板用来看目标产业的产出、就业、生产率等结果；
- **UN COMTRADE** 贸易数据用来看出口和比较优势；
- 韩国银行投入产出表用来看政策对下游产业的溢出。

也就是说，同一项政策会进入多种数据，每种数据回答一个不同的问题。看懂这张映射图，你就明白 Lane 的复现包为什么会有那么多不同来源的数据文件——它们不是堆在一起的，而是各自对应研究设计里的一个环节。

想更深入政策背景与数据映射

本讲只讲到与「读懂复现项目」直接相关的程度。政策背景与数据映射的更完整讨论，可参阅连享会《因果推断》讲义：从政策背景到研究设计、政策如何进入数据，以及 HCI 政策背景。这些属于进阶延伸，行有余力再看。

1.4 用 AI 做任务切割：把一篇顶刊拆成四份笔记

前面反复说过一句话：不要一上来就把整篇 **PDF** 丢给 **AI** 写一份大而全的笔记。这一节，我们就把「任务切割」真正做一遍，看看四份笔记是怎么一份份生出来的。

1.4.1 先破畏难：三分钟拿到第一份笔记

打开任意一个 AI 助手 (网页版 ChatGPT、Claude、Codex、OpenCode 都行)，把 Lane (2025) 的 PDF 传上去，只问它一个角度——整体介绍。几分钟后，你就得到了一份讲清「研究背景、问题、思路、结论」的短笔记。这份笔记长什么样、用的什么提示词，见配套的整体介绍。

就这么简单的一步，已经能帮你回答最要紧的问题：这篇文章和我的研究相关吗，值不值得往下读？畏难情绪，往往就在这三分钟里破掉了。

1.4.2 四刀，切出四个角度

一份整体介绍还不够。把「读一篇论文」继续切开，我们对 Lane (2025) 一共切了四刀，每一刀都是一个独立、可核对的小任务：

- 整体介绍：最快了解全貌，判断相关性。→ 整体介绍
- 实证方法与识别：用了什么方法、凭什么可信、怎么处理内生性 (只讲是什么、为什么)。→ 实证方法与识别
- 复现材料导览：复现包怎么组织、主程序在哪、数据分几层、能复现到什么程度。→ 复现材料导览
- 迷你文献定位：它站在哪些文献之间，怎么找到相关的高水平文献。→ 迷你文献定位

四份笔记各回答一类问题，你可以按需取用：先看整体介绍判断相关，再看方法能否迁移，想动手复现就看复现导览，想顺藤摸瓜找文献就用文献定位。每份笔记的开头，都附了生成它的提示词和建议调用的 **skill**，你完全可以照着自己跑一遍。

1.4.3 一条不能省的底线：AI 辅助，不是 AI 替代

任务切割让 AI 帮了大忙，但有一件事必须讲清楚——这些笔记是 **AI** 起草的初稿，不是定论。(对应大纲要点：用 AI 辅助阅读 replication package，而不是让 AI 替代研究判断。)

- AI 可能读错：把相关说成因果、把稳健性检验当成主结果、把某个数字记串；

- AI 可能编造：尤其在文献那一份笔记里，它可能列出看似合理、实则不存在的参考文献。

所以每一份笔记，都要你回到原文核对：结论对不对，看对应的图表编号；文献是真是假，用检索工具查实。研究判断——这篇文章好不好、方法能不能用在我的问题上——永远是你自己的事。把这条守住，AI 才是帮手而不是坑。

1.5 AI 协作训练

契约边界

本节逐条覆盖大纲「AI 协作训练」清单，不删项、不缩范围。每条给一份可复制的提示词、标注对应的 core skill，并说明怎么人工核对。这些任务，网页版 AI 助手大多都能完成。

下面四项训练，都围绕 Lane (2025) 的复现包展开。课程期间，直接打开 `examples/Lane_2025_QJE_paper_codes` 中的 `README.md` 和目录清单即可；官方来源仍是 Dataverse。

1.5.1 训练 1：让 AI 根据 README 生成复现路线图

任务：把复现包的 `README.md` 交给 AI，让它给出一张「先跑什么、再跑什么」的路线图。

提示词

> 这是一篇顶刊论文复现包的 `README.md`(附文本)。请帮我生成一张复现路线图：按实际运行顺序，列出从入口脚本到最终图表的每一步，每步标明运行哪个脚本、读什么数据、产出什么。只依据 `README`，不确定的地方标出来让我核对。

对应 **skill**：`core/03-replication-navigator`。人工核对：对照 `README` 里的脚本清单，检查路线图有没有漏掉或杜撰脚本名；不放心就翻到复现材料导览 对一遍。

1.5.2 训练 2：让 AI 解释文件夹结构和主程序调用关系

任务：让 AI 讲清楚各目录分别装什么、主程序怎样一层层调用子脚本。

提示词

> 这是该复现包的目录清单 (附)。请帮我：(a) 用一句话说明每个主要目录 (如 `code/`、`data/`、`output/`) 装的是什么；(b) 画出主程序的调用关系——谁调用谁，从入口一直到出图出表。只依据我给的目录和 `README`，不要编造文件。

对应 **skill**：`core/03-replication-navigator`。人工核对：用「被反复调用的是子脚本、调用别人的是主程序」这条法则，抽查一两个它给出的调用关系对不对。

1.5.3 训练 3：让 AI 读一段代码，说明它在复现流程中的位置

任务：挑复现包里的一段代码 (Stata 或 R 都行)，让 AI 说明它在整个流程里干什么、对应论文的哪张图表。这一项也顺便兑现大纲「R 作为补充」的要求——我们不学 **R** 语法，只借 **AI** 读懂 **R** 代码在流程中的角色。

> **提示词**
>
> 这是该复现包里的一段代码 (附，语言为 Stata 或 R)。请说明：(a) 它大致在做什么；(b) 它读入什么、产出什么；(c) 它对应论文里的哪张图或哪张表、处在复现流程的哪一步。不用教我语法，只讲它在整个复现流程里的位置。不确定就标出来。

对应 **skill**：`core/03-replication-navigator`。人工核对：对照 `README` 的「脚本—论文内容」对照表，确认 AI 说的图表编号对得上；方法名词若超出你目前掌握，记下来即可，后面几讲会学到。

1.5.4 训练 4：让 AI 总结「从数据到结果」的复现日志

任务：跑通 (或读通) 一部分复现后，让 AI 帮你把「从哪份数据、经过哪些步骤、得到哪个结果」整理成一份复现日志。

> **提示词**
>
> 我正在复现这篇论文。以下是我跑过的脚本、用到的数据和得到的产出 (附)。请帮我整理成一份「从数据到结果」的复现日志：按步骤记录输入数据 → 处理脚本 → 产出 (中间数据/图/表)，并标出哪些步骤因为缺数据或缺许可跑不了。只依据我提供的信息，不要补全我没给的内容。

对应 **skill**：`core/06-repro-logger`。人工核对：检查日志里每一步的输入输出是否和你实际跑的一致，跑不了的步骤有没有如实标注——一份诚实的复现日志，比一份漂亮但注水的更有价值。

1.6 小结与练习

本讲从「一项实证研究是怎么做出来的」讲起，用书法临摹作比，说明复现一篇规范的顶刊论文，是全面掌握问题界定、研究设计、数据处理、结果解读各环节的最好方式。随后以 Lane (2025) 为例，我们看清了一个复现包的骨架 (README、主程序、数据四层、脚本分工)，并用「任务切割 + AI 协作」的方法，把读懂一篇顶刊拆成了四个可核对的小任务。(对应大纲要点：从复现文档中学习研究设计、数据处理和结果组织。)

一句话带走：**AI** 让你更快地读懂顶刊，但读得对不对、用不用得上，始终由你判断。

1.6.1 练习

练习 1(手工判断)

下面是某复现包的部分文件名，请判断哪个最可能是主程序、哪些是数据处理脚本、哪些是结果输出脚本，并说明理由：`0_master_run_analysis.do`、`build_industry_panel.do`、`Figure2.R`、`merge_trade_data.do`、`Table1.R`。

参考答案

- 主程序：`0_master_run_analysis.do`——`0_` 前缀加 `master`、`run`，典型的总驱动脚本，内容多为按序调用别人。
- 数据处理脚本：`build_industry_panel.do`(build，构造面板)、`merge_trade_data.do`(merge，合并数据)——读原始数据、写中间数据。
- 结果输出脚本：`Figure2.R`、`Table1.R`——直接对应论文的图 2 和表 1，读分析样本、出图出表。

练习 2(概念)

用你自己的话，说清楚原始数据、中间数据、分析样本、图表结果四者的关系；并解释为什么 Lane 的复现包要专门预存一部分「非公开数据的中间结果」。

参考答案

四者是一条流水线：原始数据经清洗、合并、构造成为中间数据，再定好样本范围与核心变量成为分析样本，最后由模型产出图表结果。预存「非公开数据的中间结果」，是为了让没有微观数据访问权限的人，也能把依赖这些数据的图表复现出来——这是复现包「分级透明」的一种做法。

练习 3(交给 agent 并审计)

任选一篇你自己研究领域的论文 (最好带复现包)，用本讲「训练 1」的提示词，让 AI 生成一张复现路线图；然后回到 **README** 核对，找出 AI 的路线图里至少一处不准确或存疑的地方，并说明你是怎么发现的。

参考答案 (要点)

本题没有标准答案，重点在「审计」这一步。合格的回答应能指出 AI 的具体错处 (如杜撰了一个不存在的脚本、把运行顺序排错、漏掉一个数据处理步骤)，并说明核对依据 (README 的脚本清单、目录实际文件、脚本对照表)。能稳定地发现并纠正 AI 的错误，正是本讲要练的能力。

1.7 延伸阅读

- 进阶观摩：一位博士生的多角度精读成品。一位连享会助教用多个 agent 对本文生成了一整套精读笔记 (framing / summary / methods / derivation / replication 五个角度，外加逐行代码注释和一份 agent 协作工作流)，可作为「用 AI 读顶刊能到什么程度」的参照——完整内容见这份进阶参照页(不在正式目录中，从此处跳转浏览)。其中方法与推导部分含较多计量方程，属进阶内容，初级班点到为止；那份多 agent 工作流属高级可选，不要求复刻。我们课上做的，是它的简化版。
- 扩展案例：**Akcigit et al. (2022)《Taxation and Innovation in the Twentieth Century》(QJE)**。作为课后扩展案例，可用本讲的任务切割法自行解读；完整引文见阅读清单。
- 方法进阶留待高级班。本讲只帮你读懂 Lane 用了哪些方法、为什么用；这些方法 (动态 DID、双重稳健、三重差分，以及更前沿的现代 DID 估计量) 本身怎么做，属于进阶内容，可参考连享会关于现代 DID 的整理，并留待高级班系统学习。
- 政策背景延伸见前面「概念讲解」末尾给出的《因果推断》讲义链接。

2 数据处理、探索性分析与样本构造

本讲要点

- 数据来源、数据字典和变量口径；
- 观察单位、数据颗粒度、主键和唯一性检查；
- 横截面、时间序列、面板数据和多层级数据；
- 数据合并、追加、聚合、频率转换和长宽转换；
- 缺失值、离群值、缩尾、截尾、对数转换和标准化；
- 字符变量、日期变量和简单文本变量处理；
- 探索性数据分析 (EDA)：直方图、密度图、箱线图、散点图、时间趋势图和分组均值图；
- 如何用图形检查变量分布、离群值、样本断点、组间差异和潜在样本选择问题；
- 样本筛选日志、变量字典、匹配率和样本流失检查；
- 数据处理过程的记录、复核和可复现性。
- 本讲用到的 skills：core/04-data-cleaning-planner • core/06-repro-logger；
- 可运行伴生文件：examples/ch02/ch02_main.do、examples/ch02/ch02_python.ipynb。

课堂运行方式

在仓库根目录打开 Stata 后，先执行 `cd "examples/ch02"`，再运行 `do ch02_main.do`。该 do 文件默认从 GitHub 读取三份教学数据，因此不必另行下载；若已克隆仓库，也可以把开头的全局宏 `D` 改为本地 `examples/ch02/data` 路径。运行时会生成 `firm_finance_2021_2022.dta` 和 `industry_lookup_upper.dta` 两个教学中间文件，它们不是原始数据，也不会纳入仓库。

2.1 一份数据的旅程

小林是一名经济学研究生。她想弄清一个并不复杂的问题：制造业上市公司的研发投入，是不是随着企业规模变大而变化，并且因行业而异？问题不难，难的是——要回答它，她得先有一张干净的数据。

她手上的原始文件是这样几份：

- `firm_finance_2021.dta`、`firm_finance_2022.dta`——两批分开导出的公司财务表；
- `industry_lookup.dta`——一张行业代码对照表，同门整理后发来的；
- `mktcap_monthly.dta`——公司月度市值表，一家公司一年有十二行。

打开一看，问题不少：研发支出这一列，有的是空的，有的写着 `-99`；资产负债率不是数字，而是 `"38.2%"` 这样的文字；同一家公司的证券代码，在财务表里是 `000101`，到了市值表里却变成了 `101`——前导零悄悄没了；月度市值表里，一家公司一年十二行，而财务表里一年只有一行，两张表根本不在一个「颗粒度」上。

这些看起来都是「小毛病」。但正是这些小毛病，决定了最后能不能得到一份可信的数据。若对它们视而不见、直接开始 `merge`、`reg`，样本量可能莫名其妙地翻倍或腰斩，某个系数可能被一个录错的数字带偏，而你甚至不知道是哪一步出了错。

这一讲，我们就把这段路完整走一遍：从这几份凌乱的原始表，走到一张「一家公司一年一行、变量口径清楚、可以直接跑回归」的干净面板。沿途我们会不断停下来问同一类问题——不是「这条命令怎么写」，而是「这一步我在做什么判断，做完怎么知道自己没做错」。

一路上要回答的问题，串起来就是本讲的地图：

- 这份数据从哪来、每个变量到底量的是什么？(来源、口径与数据字典)
- 一行代表什么、这几张表能不能对齐？(观察单位、粒度与数据形态)
- 怎么把它们正确地拼成一张表，合并之后样本为什么变了？(合并、追加与聚合)
- 哪些变量不能直接用、该怎么处理？(缺失、离群与变换)
- 字符和日期这类「非数值」变量怎么办？
- 怎么用图形给数据做一次体检？(探索性分析与图形诊断)
- 怎么把整个过程记录下来，让别人 (和半年后的自己) 能复现？(日志、字典与匹配率)
- 最后，把这套做法用到一篇顶刊论文 Lane (2025) 的真实数据上，我们能不能复现它的数据处理、并验证结果？

> ❗ 本章怎么读：从「记命令」到「调技能」
>
> 过去学数据处理，学的是一串命令：`merge`、`reshape`、`winsor2`、`destring`……记住语法，套到数据上。
> 在 AI 和 Agent 辅助下，重点变了。你不必先背下每条命令的写法，但你必须能做到三件事：认清自己面对的是哪一类数据处理问题、把这个任务描述清楚、再调出合适的 **skill** 帮你完成——手头没有现成提示词时，还知道去哪里搜索和安装。可以这样理解：在 Agent 模式下，一个 skill 就相当于过去若干条命令的「有序打包」。
> 所以本章每遇到一类任务，都会用同一个小循环来讲：**a.** 场景 (为什么此刻需要它)→ **b.** 直觉 (讲人话、给直觉)→ **c.** 最小操作 (一两行看得懂的代码)→ **d.** 怎么检查 (结果对不对)→ **e.** 常见坑 (真实的翻车)。并在关键处配一个 callout 框，写清提示词和该调用的 **skill**。命令会忘，判断和这套工作方式不会。

还有一层意思，得在开头就挑明。这一整讲讲的常见操作、适用场景和陷阱，说到底都是为了安全地借 **skills** 和 **agent** 来清洗数据做准备。有了 AI，写清洗代码的门槛低了，风险却换了地方：agent 可能投机取巧，甚至为了凑出你后面想要的回归结果，在数据清洗和变量构造阶段悄悄做手脚。所以你得有警惕心，也得守住几条不可破除的红线——本讲末尾会把这些红线，连同「面对不同数据该调哪个 skill」一并交给你。前面每一节要练的判断，都是为了让你有能力看穿 agent 有没有偷工减料。

> 还记得上一讲打开 Lane (2025) 复现项目时，那一堆文件夹和脚本吗？本讲要补上的，正是其中最耗时、也最容易出错的一环：原始数据究竟怎么变成分析数据。相关背景见 第 1 讲。

2.2 动手之前先问三件事

拿到数据就急着 merge、急着 reg，是初学者最常见、也最贵的错。有经验的研究者会先停下来，问三件事：这份数据从哪来？每个变量到底量的是什么？有没有一份写清楚这些的说明书？在此之前，还有一件更基础的事——先把项目文件夹摆好。

2.2.1 先把项目文件夹摆好

很多人的数据处理，一开始就乱在文件夹上：原始数据、改了一半的数据、结果图表全堆在一个目录里，几个月后连哪份是原始数据都认不出。动手之前，先把项目结构摆好，这不是可选项，是可复现的地基。一个朴素好用的结构：

```
project/
   data_raw/      原始数据，只读，永不手动修改
   data_clean/    清洗好、可直接分析的数据
   code/          清洗与分析的 do 文件(按阶段编号：01_clean.do、02_merge.do…)
   output/        图表与结果
   README.md      数据来源、处理顺序、变量定义
```

最该记住的一条铁律：**data_raw/ 只读，永不手动改。**所有清洗都写成 do 文件、结果另存到 data_clean/；哪怕处理逻辑错了，原始数据永远能回溯重来。真正需要反复修改和保存的，是代码和 README，不是数据本身。这条习惯，本讲最后一节还会落到「留痕」上再讲一遍。

2.2.2 数据从哪来

同样叫「企业规模」，统计年鉴里的口径和上市公司年报里的口径可能完全不同；同样一个变量，公开调查数据和商业数据库的可得性、清洗程度、使用许可也各不相同。来源，决定了数据的口径、质量、边界和你能不能公开它。经管社科研究中，数据大致来自四类渠道：

- 公开统计：统计年鉴、国家统计局、世界银行 WDI 等，多为宏观或地区层面；
- 微观调查：CFPS、CHFS、CGSS 等入户或抽样调查，个体或家庭层面；
- 商业数据库：CSMAR、Wind、GTA 等，上市公司财务、治理、市场数据的主要来源；
- 文本与网络数据：公告、新闻、政策文件、网页抓取等非结构化来源。

> 每一类的具体获取方法 (接口、下载、抓取、清洗) 本讲不展开，详见 金融数据分析讲义-数据获取。本讲的重点，是拿到数据之后怎么判断和处理。

本讲这份数据就横跨了两类来源：财务表来自商业数据库导出，行业对照表由他人整理。来源一杂，格式和口径就容易打架——这也正是后面各种毛病的根源。

2.2.3 变量口径

口径，就是「这个变量到底量的是什么、怎么量的」。它常常藏在一个看似清楚的变量名背后：

- 「研发支出」是财报里费用化的研发费用，还是含资本化的研发投入总额？两者能差出不少；
- 「规模」用总资产、营业收入，还是员工人数？换一个，结论可能就变；
- 「资产负债率」是百分数 (45.3) 还是小数 (0.453)？单位不统一，一取对数就出错。

口径不清，后面所有的处理和回归都建在流沙上。所以动手之前，先把每个关键变量的口径钉死。

2.2.4 数据字典

把上面这些「口径」逐一写下来，就成了一份数据字典 (codebook)：每个变量的名称、含义、单位、来源、口径和取值范围。它是一份数据最重要、却最常被跳过的文档。给上面那份财务表配一份字典，开头几行大概是这样：

变量	含义	单位	来源	口径
`stkcd`	证券代码	——	商业数据库	6 位，含前导零
`sales`	营业收入	万元	商业数据库	合并报表口径
`rd`	研发支出	万元	商业数据库	费用化，缺失以 -99 标记
`lev`	资产负债率	%	商业数据库	期末，负债/资产

没有数据字典的数据，是一颗定时炸弹：你今天记得 `rd` 的缺失是 `-99`，三个月后就未必了。

> AI 协作：让 AI 生成数据字典初稿，你来核对
>
> 数据字典不必从零手写。把原始表头和你了解的口径交给 AI，让它先起一稿，你再逐行核对补全——生成靠 AI，判断靠你。
> 任务说明书 (提示词模板)：
>
> 你是我的数据助手。下面是一份上市公司财务数据的变量清单(表头 + 前 5 行)：
> [粘贴 describe 或前几行]
> 请据此生成一份数据字典草稿，列出每个变量的：中文含义、推测单位、可能的口径、取值范围与异常值提示(如负值、明显离群、以特殊值标记的缺失)。凡是你不确定的口径，请单独标出、让我确认，不要替我编造。
>
> 推荐 **skill**：`core/04-data-cleaning-planner`(数据清洗计划师)。没有合适的提示词？去 附录 A 提示词模板 取用，或用 `find-skills` 检索、安装社区里的同类技能。

注意这里已经发生的转变：过去你要记住 `codebook`、`describe` 这些命令；现在你要会的，是把「我需要一份数据字典」这件事描述清楚，让 skill 帮你生成，再用你的专业判断把关。带着这份写好的字典，我们才敢去动下一步——先看清，这几张表到底长什么样。

2.3 认清你的数据

合并之前，先要看懂手里的每一张表到底是什么。这一步几乎不写代码，却决定了后面每一步是否走对。要问三个问题：一行代表什么？用什么能唯一确定一行？这是哪一类、哪一种形态的数据？

本讲的数据存放在课程 GitHub 仓库，可以直接联网调取，无需克隆本仓库。下面先约定一个数据路径全局宏 `$D`，后续各例都从它读取：

```
* 本讲数据（联网直接读取；若已克隆仓库，可把网址改成本地路径）
global D "https://raw.githubusercontent.com/lianxhcn/PXa2026a/main/examples/ch02/data"
```

其中 `firm_finance_2021.dta` 收录 2020–2021 两年、`firm_finance_2022.dta` 收录 2022 年，是分两次导出的两批；下面先看第一批。

2.3.1 观察单位与粒度

先把表读进来，看它的结构和头几行：

```
use "$D/firm_finance_2021.dta", clear
describe
list in 1/4, clean noobs
```

```
Variable      Storage   Display    Variable label
  name         type     format
-------------------------------------------------
stkcd         str6      %9s        证券代码
year          int       %8.0g
sales         double    %10.0g     营业收入(万元)
rd            double    %10.0g     研发支出(万元)
at            double    %10.0g     资产总额(万元)
lev           str8      %9s        资产负债率(文本)
indcd         str5      %9s        行业代码

  stkcd   year   sales     rd      at      lev   indcd
 000101   2020   82000   4100   65000   38.2%     C39
 000101   2021   96000   5200   72000   41.0%     C39
 000102   2020   51000   3000   40000   52.4%     C39
 000102   2021   57000   3400   44000   50.8%     C39
```

一行是一家公司在某一年的记录——000101 在 2020、2021 各占一行。我们把「一行代表什么」叫作数据的观察单位，把它的粗细叫作粒度 (granularity)。这张表的粒度是「公司-年度」。

粒度是数据处理里最容易被忽略、又最容易出错的概念。对照另外两张表就清楚了：行业对照表 industry_lookup.dta 一行是一个行业 (粒度更粗)，月度市值表 mktcap_monthly.dta 一行是一家公司的某一个月 (粒度更细)。粒度不同的表不能直接合并——这是下一节反复要处理的问题，这里先记住：动手合并前，先说清每张表的一行是什么。

2.3.2 主键与唯一性

能唯一标识一行的变量组合，叫主键。财务表的主键是「证券代码 + 年度」这一对，而不是证券代码本身。这不是凭感觉，而是要查：

```
duplicates report stkcd year
duplicates report stkcd
```

```
Duplicates in terms of stkcd year
   Copies |   Obs   Surplus
----------+----------------
        1 |    84         0      <- 每个 (stkcd, year) 只出现 1 次：唯一

Duplicates in terms of stkcd
   Copies |   Obs   Surplus
----------+----------------
        2 |    84        42      <- 每个 stkcd 出现 2 次(两年)：单独不唯一
```

stkcd year 的 Surplus 为 0，说明这对组合唯一，可以作主键；stkcd 单独的 Surplus 是 42，因为一家公司横跨两年。如果误把 **stkcd** 当主键去做一对一合并，样本就会错乱。想更严格，可以用 isid：

```
isid stkcd year    // 若不唯一会直接报错，把问题挡在合并之前
```

真实数据里，主键不唯一往往来自导出重复或口径不清；养成「合并前先查唯一性」的习惯，能挡掉后面一大半的样本量异常。

2.3.3 横截面、时序、面板与多层级

同样几张表，还可以按「有没有单位维度、有没有时间维度」分类：

- 横截面：一个时点、多个单位。industry_lookup 就是——一行一个行业，没有时间。
- 时间序列：一个单位、多个时点。比如单看某一家公司逐年的营收。
- 面板：多个单位 × 多个时点。财务表就是——多家公司、多年，兼有两个维度。
- 多层级：单位嵌套在更大的单位里。公司嵌套在行业、省份中，就是两层结构 (industry_lookup 正是把公司连到行业层的桥)。

分清类型，才知道能用什么工具。面板数据要用 xtset 声明个体和时间，之后才能算滞后、增长率、组内变化。不过 xtset 要求个体编号是数值，而这里的 stkcd 是字符——这个矛盾留到清洗时解决，先记下：声明面板结构，是面板分析的第一步。

2.3.4 扁平 (wide) 与长条 (long)

最后一个要练出直觉的，是数据的形态。看一个小例子——三个人、三年的收入 inc 和失业状态 ue，一开始长这样，是「扁平型」：

```
list, clean noobs
```

id	sex	inc80	inc81	inc82	ue80	ue81	ue82
1	0	5000	5500	6000	0	1	0
2	1	2000	2200	3300	1	0	0
3	0	3000	2000	1000	0	0	1

扁平型 (wide) 一行是一个个体，不同年份摊成不同列 (inc80、inc81……)。可大多数面板分析要的是「长条型」，用 reshape 转过去：

```
reshape long inc ue, i(id) j(year)
list in 1/9, clean noobs
```

id	year	inc	ue	sex
1	80	5000	0	0
1	81	5500	1	0
1	82	6000	0	0
2	80	2000	1	1
2	81	2200	0	1
2	82	3300	0	1
3	80	3000	0	0
3	81	2000	0	0
3	82	1000	1	0

转完之后，一行变成「一个人在某一年」，inc、ue 各成一列，年份进了 year 列。识别的窍门只有一句：看同一个个体是不是占了多行——占多行是长条，一行摊成多列是扁平。为什么要分清？大多数面板分析要长条型，而从网页、报表、问卷导出的原始数据常常是扁平的。本讲的财务表本就是长条的 (000101 两年占两行)；reshape 在真实面板上的更多用法，见下一节。

> AI 协作：让 agent 先探索数据形态，再交你审
>
> 面对一堆来路不明的表，与其逐张手动 describe，不如让 agent 先跑一遍、把判断汇成一张待审清单，你只需确认或纠正，再决定怎么处理。这正是 AI 工作流的关键——先自动探索、生成清单，人确认后再动手，而不是让 agent 直接改数据。
> 任务说明书 (提示词模板)：
>
> 这里有若干 .dta / .csv 数据表：[列出文件]。请对每一张表逐一探索并报告，汇成一张待审清单：
> (1) 观察单位与粒度(一行代表什么)；

(2) 主键候选(哪几个变量组合能唯一确定一行，并给出唯一性检查结果)；
(3) 数据形态(长条 long 还是扁平 wide)；
(4) 潜在问题：重复行、异常缺失、同名键的存储类型或大小写是否一致、粒度是否不齐。
只报告与提问，不要修改任何数据。我确认后再决定处理方式。

推荐 **skill**：core/04-data-cleaning-planner。没有合适提示词？见 附录 A 或用 find-skills 检索安装。

看清了每张表是什么、粒度粗细、主键和形态，我们才有把握做下一件、也是数据处理里最容易翻车的事：把这几张表正确地拼成一张。顺带说一句，本讲最后会把这套「先看清结构」的做法，用到 Lane (2025) 那份真实的行业-年度面板上——你会发现，顶刊的分析数据，结构上和这里的财务面板并无二致。

2.4 合并、追加与聚合

把几张零散的表拼成一张分析面板，是数据处理里最耗时、也最容易翻车的一步。动手之前，先分清三类操作——它们改变数据的方式完全不同：

- 追加 **(append)**：两张结构相同的表首尾相接，行变多 (如分批导出的数据)；
- 合并 **(merge)**：按键把另一张表的列贴过来，列变多 (如把行业名贴到公司上)；
- 聚合 **(collapse)**：把细粒度压成粗粒度，行变少、粒度变粗 (如把月度压成年度)。

一句话记忆：加行用 **append**，加列用 **merge**，改粒度用 **collapse**。分不清这三者，就会用错工具、炸错样本。

2.4.1 追加 (append)

财务表是分两批导出的，先把它们摞成一张完整面板：

```
use "$D/firm_finance_2021.dta", clear
count
append using "$D/firm_finance_2022.dta"
count
isid stkcd year
tab year
save "firm_finance_2021_2022.dta", replace  // 伴生 do 文件后续从这里读入
```

```
    84                                  <- 追加前：84 行(2020–2021)
   126                                  <- 追加后：126 行(多出 2022 的 42 行)

       year |      Freq.    Percent
------------+--------------------
       2020 |         42      33.33
```

```
  2021 |          42      33.33
  2022 |          42      33.33
```

追加后行数从 84 变 126，`isid stkcd year` 没报错、`tab year` 三年各 42 家，说明摞得干净。**append** 有两个前提：两张表的变量要同名，且同名变量的存储类型要一致——若一份的 `lev` 是数值、另一份是字符，追加要么报错、要么把两列错误地并在一起。所以追加后，`count` 和 `isid` 是必做的两道检查。

2.4.2 关联变量 (key)

`merge` 靠一个 (或一组) 键变量把两张表对齐。初学者一大半的合并事故，都出在键上。把原则列成一张清单，动手前逐条对照：

- **a.** 键要同名——`merge` 按同名变量对齐，两表的键必须叫同一个名字；
- **b.** 大小写、空格敏感——`"C36"` 和 `"c36 "` 是两个不同的键，差一个空格就对不上；
- **c.** 存储类型一致——字符键和数值键对不上 (后面月度表就栽在这一条)；
- **d.** 主键唯一——做一对一 (1:1) 合并前，先 `isid` 或 `duplicates report` 查唯一性；
- **e.** 编码、前导零一致——`"000101"` 和 `101` 不是同一个键；
- **f.** 合并后必查匹配率——用 `_merge` 看谁没匹配上，别急着往下走。

这六条，正好对应本讲数据里埋着的几处毛病。下面就撞上其中两条。

2.4.3 横向合并 (m:1)

合并之前，先分别看清两份数据，尤其是那个关联的键。主表这边，每行是一个公司-年度，键是行业代码 `indcd`:

```
list stkcd year indcd in 1/6, clean noobs
```

```
  stkcd   year   indcd
 000101   2020     C39
 000101   2021     C39
 000102   2020     C39
 000102   2021     C39
 600201   2020     C27
 600201   2021     C27
```

对照表那边，每行是一个行业，键同样是 `indcd`:

```
use "$D/industry_lookup.dta", clear
list, clean noobs
```

```
    indcd                                indname
      C39    计算机、通信和其他电子设备制造业
      C27                              医药制造业
     c36                               汽车制造业
```

两份数据粒度不同 (一个是公司-年度、一个是行业)，靠 indcd 把行业名贴到每个公司-年度上，这是多对一 (m:1) 合并。但动手前务必盯住键：主表里是 "C36"，对照表里却写成了小写带空格的 "c36 "——就这一点差别，足以让合并悄悄出错。数据合并十有八九，栽在没吃透两份数据、尤其没对齐键上。先直接合并，看它怎么栽:

```
use "firm_finance_2021_2022.dta", clear
merge m:1 indcd using "$D/industry_lookup.dta"
tab _merge
list stkcd year indcd _merge if _merge==2   // 看那条用不上的脏键
```

```
    Result                      Number of obs
    -----------------------------------------
    Not matched                            43
        from master                        42  (_merge==1)
        from using                          1  (_merge==2)
    Matched                                84  (_merge==3)
    -----------------------------------------

   Matching result from merge |      Freq.     Percent
------------------------------+-----------------------
             Master only (1) |         42       33.07
              Using only (2) |          1        0.79
                 Matched (3) |         84       66.14

     stkcd   year   indcd            _merge
         .      .    c36     Using only (2)
```

只有 84 行匹配上，匹配率 **84/126 ≈ 67%**。两类没匹配上的行泄露了原因：42 行 C36 的公司是 master only(面板里有、对照表里没有)，而那条 c36 是 using only(对照表里有、面板里用不上)。一对照就明白——对照表里汽车业的键写成了小写带空格的 "c36 "，和面板里的 "C36" 对不上。那条 **using-only** 记录，就是「键没洗干净」最典型的信号。

把键洗干净 (转大写、去空格)，再合并一次：

```
use "$D/industry_lookup.dta", clear
replace indcd = upper(strtrim(indcd))   // "c36 " -> "C36"
save "industry_lookup_upper.dta", replace

use "firm_finance_2021_2022.dta", clear
merge m:1 indcd using "industry_lookup_upper.dta"
tab _merge
```

```
Result                           Number of obs
-----------------------------------------
Not matched                                 0
Matched                                   126  (_merge==3)
-----------------------------------------
```

126 行全部匹配，匹配率 **100%**。请把顺序记牢：`merge` 之后第一件事就是 `tab _merge` 看匹配率；匹配率偏低，几乎总是键的问题，先回头洗键，别让缺胳膊少腿的数据流到回归里。

> AI 协作：让 AI 复核 merge / join 逻辑
>
> 合并前，把两张表的粒度、键和合并类型描述给 AI，让它先替你把关——尤其是 `1:1` 还是 `m:1`、键要不要先清洗、合并后该期望多少匹配率。判断仍由你拍板，但 AI 能提前拦下大多数低级错。
> 任务说明书 (提示词模板)：
>
> 我要把两张表合并。表 A：[粒度、主键、行数]；表 B：[粒度、主键、行数]。
> 我打算用 [键] 做 [1:1 / m:1] 合并。请检查：
> (1) 这个合并类型对不对？两边键的唯一性是否支持它？
> (2) 两边的键在名称、大小写、空格、存储类型、前导零上是否一致？可能哪里对不上？
> (3) 合并后我应该预期多大的匹配率？出现 master-only / using-only 各说明什么？
> 只做检查与提问，先不要写最终代码。
>
> 推荐 **skill**：`core/04-data-cleaning-planner`。没有合适提示词？见 附录 A 或用 `find-skills` 检索安装。

2.4.4 聚合与频率转换

月度市值表 `mktcap_monthly` 要并进公司-年度面板，可两张表的粒度不一样：一个是公司-月，一个是公司-年度。这时候最常见的翻车，就是不管粒度直接合并。先验证一下月度表到底是什么粒度：

```
use "$D/mktcap_monthly.dta", clear
gen int year = int(ym/100)      // 202103 -> 2021
duplicates report stkcd year
```

```
Duplicates in terms of stkcd year
   Copies | Observations   Surplus
----------+-----------------------
       12 |          504       462      <- 一个公司-年度有 12 行：远非唯一
```

一个「公司-年度」占了 12 行 (12 个月)。若强行按 `stkcd year` 做 `1:1` 合并会直接报错；若用 `m:1`、`m:m` 硬合，则会把面板炸成一堆重复行。正确的顺序是：先把月度聚合到公司-年度 (这一步既是频率转换，也是聚合)，再合并。

```
collapse (mean) mktcap, by(stkcd year)   // 月 -> 年，取年均市值
duplicates report stkcd year             // 现在唯一了
tostring stkcd, replace format(%06.0f)   // 数值 101 -> 字符 "000101"，补回前导零
list in 1/3, clean noobs
```

```
Duplicates in terms of stkcd year
   Copies | Observations   Surplus
----------+-----------------------
        1 |           42         0      <- 聚合后：公司-年度唯一

     stkcd   year      mktcap
    000101   2021   391.28333
    000102   2021   223.25833
    000201   2021   658.84167
```

这里顺手补上了 key 原则的第 c、e 条：月度表的 `stkcd` 存成了数值 `101`，和面板里的字符 `"000101"` 类型不同、还丢了前导零，`tostring ... format(%06.0f)` 把它转回六位字符键。到这一步，聚合后的年度市值已是干净的公司-年度粒度、键也对得上，再 `merge 1:1 stkcd year` 就能安全并入主面板 (本例市值只覆盖 2021 年，并入后其余年份的市值为缺失，这也是真实数据的常态)。

2.4.5 长宽转换 (reshape)

有时还要在长条与扁平之间来回变形。把面板的营收摊成「每公司一行、各年一列」，用 `reshape wide`：

```
use "firm_finance_2021_2022.dta", clear
sort stkcd year
list stkcd year sales in 1/6, sepby(stkcd)
keep stkcd year sales
reshape wide sales, i(stkcd) j(year)
list in 1/2, clean noobs
```

```
     stkcd   sales2020   sales2021   sales2022
    000101        82000       96000      108000
    000102        51000       57000       61000
```

`reshape long sales, i(stkcd) j(year)` 再变回长条。记住两个选项：`i()` 是保持不变的个体，`j()` 是要摊开或合拢的维度。大多数面板分析要长条型，而从网页、报表导出的数据常是扁平的，`reshape` 就是两者之间的桥。

到这里，几张零散的表已经拼成一张「公司-年度、带行业名、带年度市值」的面板。回头看，每一步靠的都不是记住某条命令，而是先想清粒度和键、动手后用 **`isid`**、**`_merge`**、**`duplicates`** 去查。本讲最后复现 Lane (2025) 的数据时，你会看到同样的动作：把政策行业交叉表按行业代码 `merge` 进面板——键就是行业代码，检查方式一模一样。

2.5 变量能直接用吗

表拼好了，变量却未必能直接进回归。这一节处理四件事——缺失、离群、对数、标准化——它们有一个共同的做法：先看，再决定，尤其是先画图。判断永远在动手之前。下面接着上一节 append 好的面板往下做，并把绘图模板设为 cleanplots(ssc install cleanplots 一次即可)。

```
set scheme cleanplots
```

2.5.1 缺失值

财务表的 rd 有的是空、有的写着 -99。这个 -99 是人为标记的缺失，却伪装成一个数字——若不处理，它会被当成真实研发支出，算进均值、跑进回归。第一步是把它还原成真正的缺失：

```
mvdecode rd, mv(-99)      // 把 -99 全部替换为缺失 .
count if missing(rd)
misstable summarize rd, all
```

```
rd: 1 missing value generated
    2                             <- rd 现在共有 2 个缺失(原本的空 + 还原的 -99)

                |                                    | Unique
      Variable |     Obs=.     Obs>.     Obs<.   | values     Min      Max
  -------------+------------------------------------+----------------------
            rd |         2                  124   |    121      993    21936
```

这里藏着一个经典的坑：在 **Stata** 里，缺失 . 大于任何数字。所以 count if rd>10000 会把缺失也数进去，keep if rd<. 之类的写法必须小心。好在 summarize、regress 这些命令会自动忽略缺失——但 count、keep 不会。先把伪装的缺失还原，是一切的前提。

2.5.2 离群值

sales 里可能有异常大的值。先看统计量，再画图：

```
summarize sales, detail
histogram sales, percent xtitle(" 营业收入 (万元)")
graph box sales
```

营业收入(万元)

	Percentiles	Largest		
50%	85370.5	231000		
75%	145827	242112	Mean	109905.8
95%	223839	245000	Skewness	6.348803
99%	245000	1080000	Kurtosis	58.35257

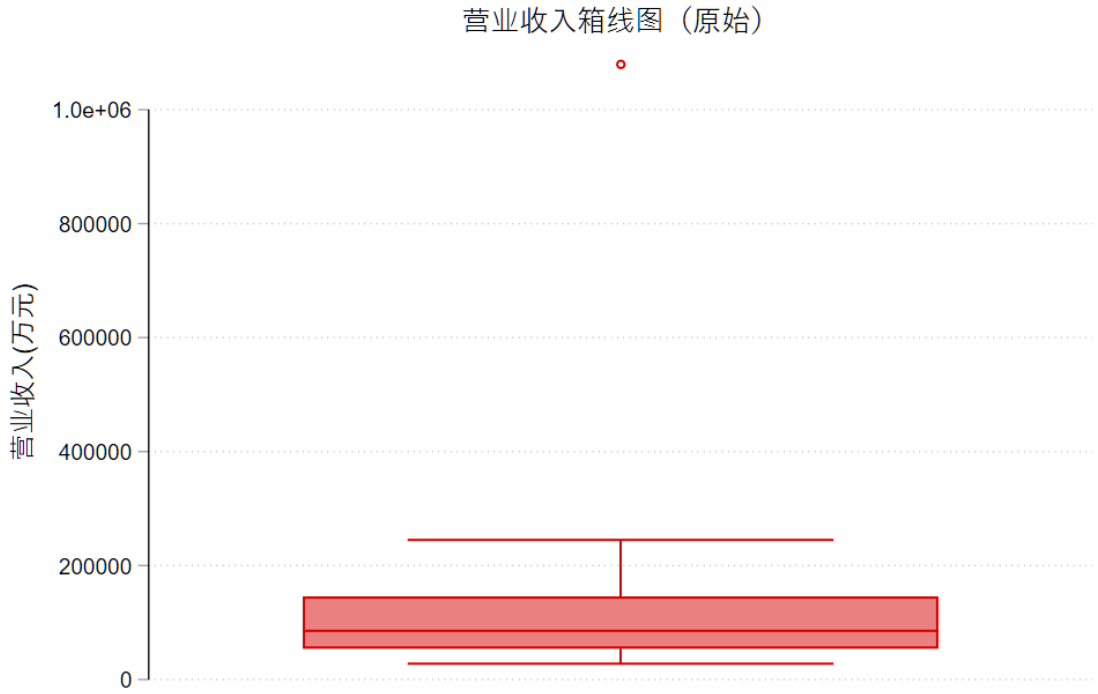

图 2.1: 营业收入的箱线图：那个飞出箱体的点，就是离群值。

偏度高达 6.35、箱线图上那个孤零零飞出去的点，都在喊「这里有离群值」。但离群不等于错误——处理之前，必须先查清它到底是什么：

```
list stkcd year sales if sales>500000, clean noobs
```

stkcd	year	sales
000101	2022	1080000

这是 000101 在 2022 年的营收。它别的年份都在 10 万上下，这里却是 108 万——多打了一个 **0**，是录入错误。对于这种能查明的错误，正确的做法是改对它，而不是缩尾：

```
replace sales = 108000 if stkcd=="000101" & year==2022
```

只有对无法核实、但确实极端的值 (不是错误，就是天生很大或很小)，才轮到缩尾和截尾出场：

```
winsor2 sales, cuts(1 99)      // 缩尾：把 1%/99% 之外的值拉回分位点，样本量不变
winsor2 sales, cuts(1 99) trim // 截尾：直接删掉极端值，会损失样本
```

缩尾把过大过小的值拉回到分位点上，保留样本量；截尾干脆删掉它们，样本变少。二者都不该在「看图、查明」之前就无脑套用——否则要么掩盖了本可修正的错误，要么扭曲了真实的分布。顺序永远是：先画图 → 查明来源 → 能改的改，改不了的再缩尾或截尾。

需要说明的是，对于离群值，不同的研究领域会有不同的主流处理方式。

- 在研究偏宏观的问题时，比如说城市或省级数据，主要是采用对数转换。因为这类数据通常都是经过微观数据加总后得到的，其数据本身不太可能存在离群值。之所以进行对数转换，主要是因为从理论模型向计量模型转换时做了对数变换 (比如说最典型的 CD 生产函数)。因此，严格来讲，这类计量模型里边的对数转换并不是为了应对离群值。
- 在偏微观的领域，比如说劳动经济学和公司金融领域，缩尾处理是非常常用的手段。比如说在公司金融领域，大量的指标都是财务比率，加之是公司层面的微观数据，所以离群值是非常普遍的。在多数的公司金融文献中，通常是对这些财务比率进行第 1 百分位和第 99 百分位的缩尾。如果你的论文中缩尾的力度比这个要更大，要给出合理的解释。

> AI 协作：让 AI 讲清缺失、离群、缩尾、截尾、对数的差别
>
> 这几个概念初学者最容易混。可以让 AI 结合你具体的变量把它们讲清楚，并给出何时用哪一个的判断依据——但最终用不用、怎么用，由你根据数据和研究问题决定。
> 任务说明书 (提示词模板)：
>
> 针对我的变量 [变量名、含义、分布特征(偏度、极值、缺失比例)]，请分别解释：缺失、离群、缩尾(winsorize)、截尾(trim)、对数转换——各自解决什么问题、代价是什么、会如何
> 不要替我下结论，把判断依据摆出来让我选。
>
> 推荐 **skill**：`core/04-data-cleaning-planner`。没有合适提示词？见 附录 A 或用 `find-skills` 检索安装。

2.5.3 对数转换

资产、营收这类变量往往右偏——大多数不大，少数很大。回归里常对它们取对数：

```
gen double size = ln(at)       // size 即「企业规模」的常用度量
histogram at, percent
histogram size, percent
```

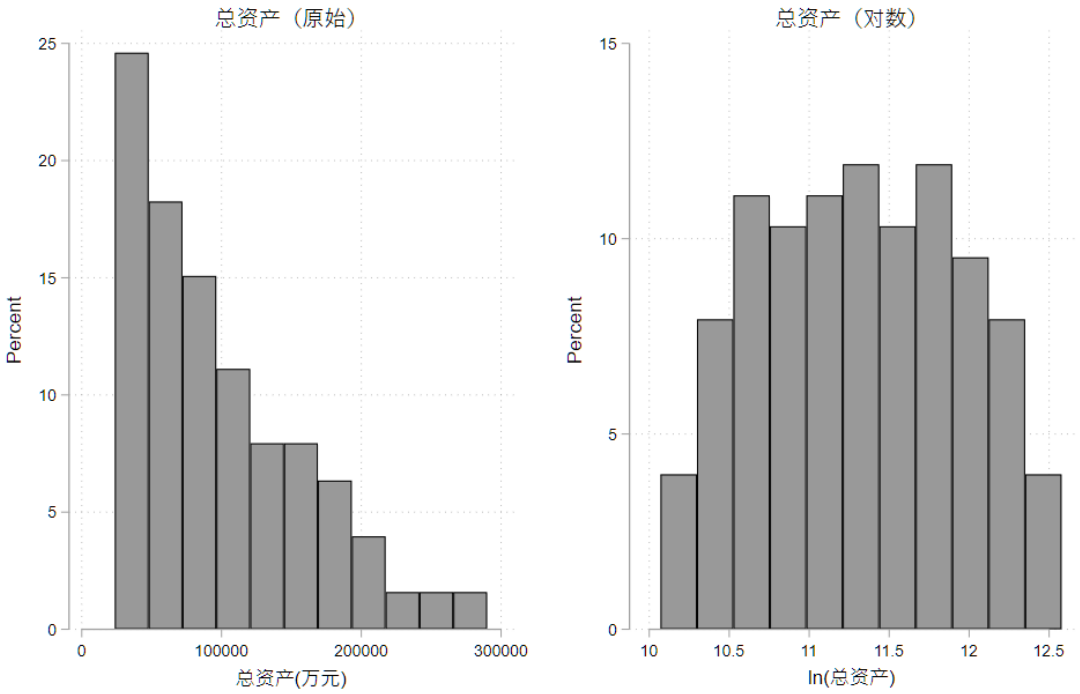

图 2.2: 总资产取对数前后：左边右偏，右边明显更对称。

对数把「大的压得多、小的压得少」，让右偏的分布更对称，也把变量间的乘法关系变成加法关系 (这正是弹性解释的由来)。两点要记住：取对数后系数的含义会变 (变成半弹性或弹性)，解释要相应改口；而且 `ln()` 要求正值，0 和负数会变成缺失，取对数前先检查。

有关对数转换的更深层的讨论，参见如下推文：

- 刘潍嘉, 2024, 偏态分布数据的回归模型选择.
- 匡宇驰, 2024, log-0 问题：零值太多如何取对数？.
- 吴梦萱, 2024, 纠结！DID 中取对数还是不取对数？论文推介.
- 毕英睿, 2023, 取对数：如何应对零值和负数.
- 浦进博, 2025, 理解变量转换与非线性效应的八个基本准则.
- 秦范, 2021, 取对数！取对数？.

2.5.4 标准化

有时要把变量标准化到「均值 0、标准差 1」(z-score)，便于跨变量比较，或供某些方法使用：

```
egen z_size = std(size)
summarize size z_size
```

```
    Variable |   Obs      Mean    Std. dev.       Min        Max
-------------+--------------------------------------------------------
        size |   126    11.315       .625     10.070     12.578
      z_size |   126    ~0           1        -1.993      2.021
```

标准化不改变分布的形状，只是换了把尺子：减去均值、除以标准差，之后「一个单位」就等于「一个标准差」。

回头看这一节，没有一步是「因为命令该这么写」。每一步都是先看 (统计量、图形)、判断 (是错误还是真离群？要不要变换？变换后怎么解释？)、再动手，动手之后再回看。缺失怎么补、离群怎么处理、要不要取对数，都没有唯一正确答案，取决于你的数据和研究问题——这恰恰是研究者不能外包给 AI 的那部分判断。

2.6 字符与日期变量

有些变量看着是数字或日期，却以文本存着，直接拿去计算会报错，或者更糟——悄悄算错。这一节做三件事：把文本型数字转成数值、把字符日期转成真日期、以及对文本变量该有的意识。判断的核心始终是一句话：先看清它现在是什么类型、该是什么类型。

2.6.1 文本型数字 (destring)

lev(资产负债率) 存成 "38.2%"，是字符——直接求均值、跑回归都会出错。用 destring 转成数值：

```
use "$D/firm_finance_2021.dta", clear
destring lev, gen(lev_num) percent
list stkcd year lev lev_num in 1/4, clean noobs
```

```
lev: character % removed; lev_num generated as double

     stkcd   year     lev   lev_num
    000101   2020   38.2%      .382
    000101   2021   41.0%       .41
    000102   2020   52.4%      .524
    000102   2021   50.8%      .508
```

percent 选项会去掉 % 并除以 100，得到 0–1 的比率。但不是所有「看着是数字」的文本都该转。stkcd 也全是数字 ("000101")，可它是编号 (标签)，不是量——destring 会把它变成数值 101，丢掉前导零，键就废了。所以规则很简单：是「量」(要参与计算) 就转数值；是「编号、标签」就保持字符。见到文本型数字，先问它是量还是编号，再决定动不动 destring。

2.6.2 类别文本 (encode)

indcd 的 "C39"、"C27"、"C36" 是类别标签。有些分析 (比如放进 i.industry 做固定效应) 需要数值型的类别。encode 把文本类别转成「带标签的数值」：

```
encode indcd, gen(ind_id)
label list ind_id
```

```
ind_id:
           1 C27
           2 C36
           3 C39
```

encode 自动建了一张「数字 ↔ 文字」对照表 (value label)：数据里存的是 1/2/3，显示出来还是 C27/C36/C39；decode 则反向还原。**destring** 和 **encode** 的分工要记住：数字被误存成文本，用 destring(或 real()) 还原；观察值本身就是文字类别，用 encode 给它编号。一个还原数字，一个给文字贴号。

2.6.3 日期变量

日期常以文本 ("2021-06-30") 或数字 (20210630) 存着。要先把它变成 Stata 的日期值，才能排序、算间隔、按频率聚合：

```
clear
input str10 rptdate
"2021-06-30"
"2022-03-31"
end
gen d = date(rptdate, "YMD")   // 第二个参数告诉 Stata 顺序是「年-月-日」
format d %td                    // 显示成人类可读的日期
gen year  = year(d)
gen month = month(d)
list, clean noobs
```

```
    rptdate           d   year   month
 2021-06-30   30jun2021   2021       6
 2022-03-31   31mar2022   2022       3
```

date(字符, "YMD") 里的 "YMD" 告诉 Stata 按「年月日」解析；format d %td 让它显示成 30jun2021(内部其实是从 1960-01-01 起算的天数)；之后 year()、month()、qofd() 等函数随手取。两个坑：顺序参数要对 ("YMD" 还是 "MDY")；纯数字型日期 (20210630) 要先 tostring 再 date，或直接用 substr 拆出年月日。

2.6.4 文本变量：交给 skills

传统教学到这里会讲一大套——用 `regexm`、`split`、`substr` 从「中国农业银行深圳光明支行」里抠出总行名和城市。在 AI 工作流下，你不必记这些代码，但必须有意识：认出「这是一个文本处理任务」、能把它描述清楚、调出合适的 skill 来做。

> AI 协作：文本处理，有意识、会描述、调技能
>
> 遇到公司名、地址、公告文本这类需要「从一串文字里抽取信息」的任务，别急着自己写正则，先把任务描述清楚交给 AI 与技能。
> 任务说明书 (提示词模板)：
>
> 我有一列公司全称，如「中国农业银行深圳光明支行」「招商银行股份有限公司兰州分行」，想从中提取：(1) 总行名称(如「中国农业银行」)；(2) 所在城市。
> 请先规划处理步骤，再生成可运行代码；对拿不准的匹配规则，先列出让我确认，不要硬编造。
>
> 推荐 **skill**：`core/04-data-cleaning-planner`；文本量大或较专门时，用 `find-skills` 检索安装专门的文本处理技能。正则表达式与 `split` 的代码细节本讲不展开 (属传统命令教学)，延伸阅读处给链接。

字符与日期处理，关键从来不是背下 `destring`、`encode`、`date` 的语法，而是先判断它现在是什么、该是什么——是量还是编号？是日期还是文本？判断对了，用哪条命令、哪个 skill，自然一目了然。

2.7 探索性分析与图形诊断

探索性数据分析 (EDA) 不是论文里那几张精修的图，而是你在清洗和建模的每一步都该做的体检。它的关键在一句话：每一张图，都是为了回答一个具体的问题。这一节把六类常用图和它们各自要查的问题对上号。

先分清两种图。探索性图形是画给自己看的——要快、要多、可以糙，目的是发现问题；解释性图形是画给读者看的——一张图只讲一件事、需要精修。EDA 阶段要的是前者：多画几张、快速扫过，把数据的毛病和结构看出来。

本节的图都基于上一节清洗好的面板，并构造两个变量：研发强度 `rd_ratio`(研发支出占营收的百分比) 和企业规模 `size`(资产的对数)，再按行业分组：

```
use "$D/firm_finance_2021.dta", clear
append using "$D/firm_finance_2022.dta"
mvdecode rd, mv(-99)
replace sales = 108000 if stkcd=="000101" & year==2022   // 修正上一节发现的录入错误
gen double rd_ratio = 100*rd/sales                        // 研发强度 (%)
gen double size = ln(at)                                  // 企业规模
gen str6 ind_s = cond(indcd=="C39"," 电子", cond(indcd=="C27"," 医药"," 汽车"))
```

2.7.1 分布：直方图与密度图

要查的问题：这个变量分布是什么形态？偏不偏？要不要变换？

```
histogram rd_ratio, percent kdensity xtitle(" 研发强度 (%)")
```

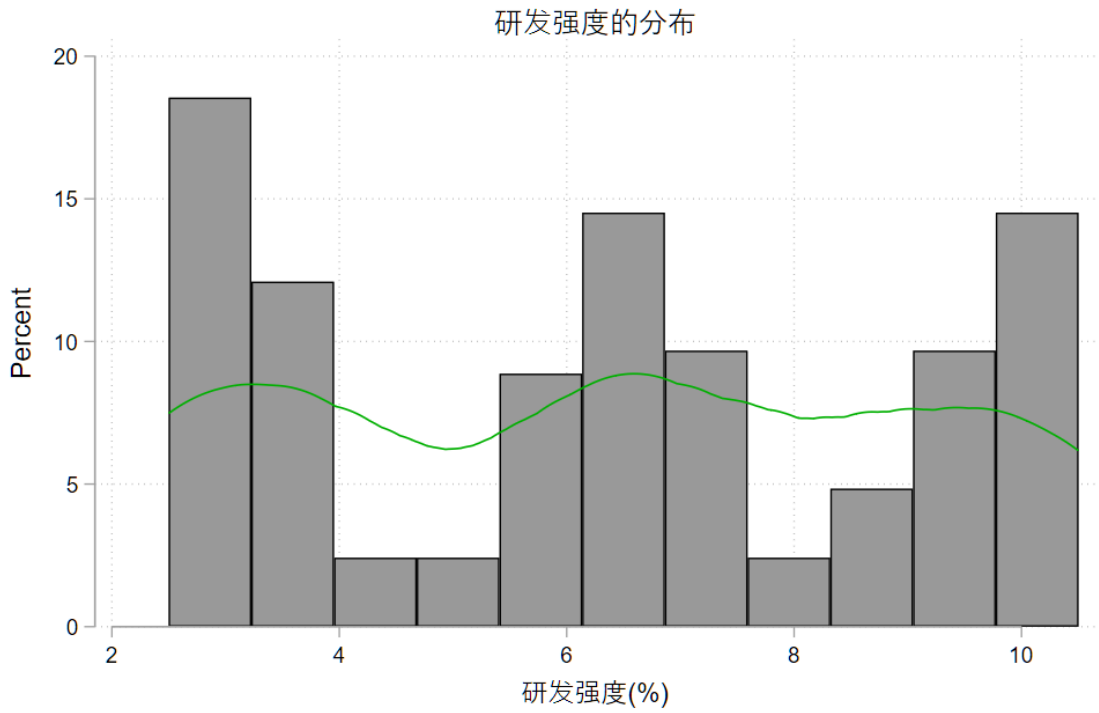

图 2.3: 研发强度的直方图，叠加核密度曲线。

直方图看形状，叠加的核密度曲线 (kdensity) 把形态描得更平滑。这里研发强度大致落在 2%–10%，并不是单峰——隐约能看出几处聚集。这个信号提醒我们：也许背后藏着分组结构 (后面散点图会证实)。若图形明显右偏，就回到上一节考虑取对数。

2.7.2 离群与组间：分组箱线图

要查的问题：各组的分布和离群怎样？组间有没有差异？

```
graph box rd_ratio, over(ind_s)
```

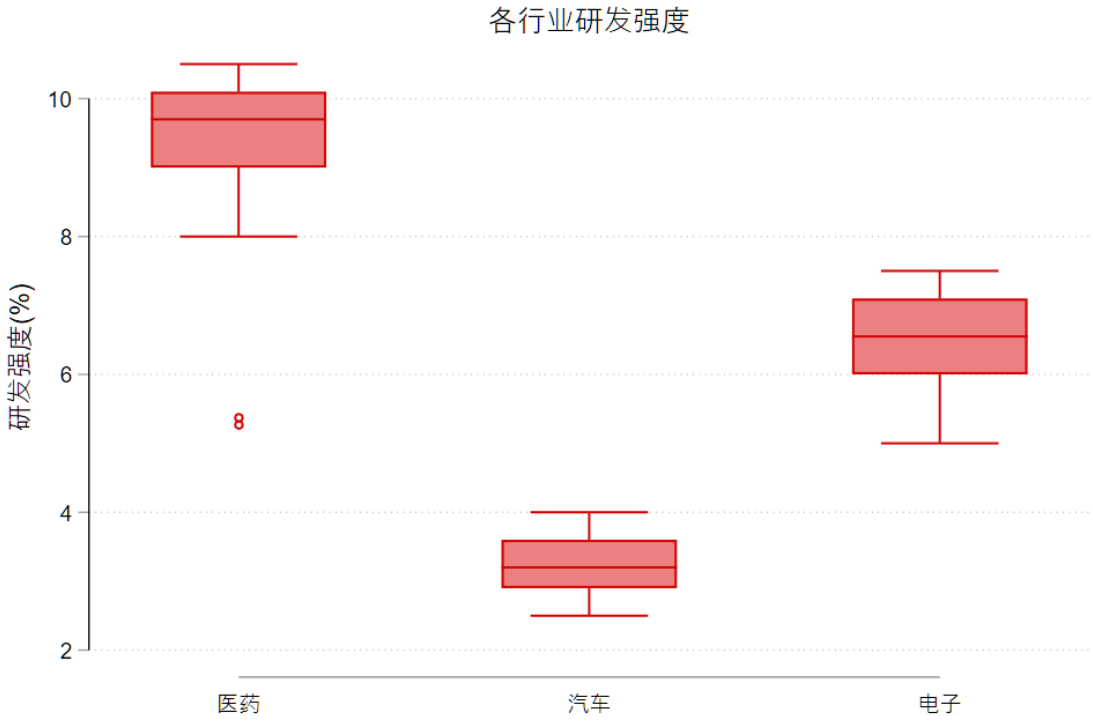

图 2.4: 各行业研发强度的分组箱线图。

一张图同时回答了两个问题。三个行业的箱体高低分明——医药最高、电子居中、汽车最低，组间差异一目了然；而医药组里还浮着一个孤立的点，那是这一组的离群值。`over()` 把分组直接摆在横轴上，不需要图例。

2.7.3 关系：散点图

要查的问题：两个变量怎么关联？有没有异常点或隐藏的分层？

```
twoway (scatter rd_ratio size) (lfit rd_ratio size), ///
    legend(order(1 " 观测值" 2 " 拟合线") pos(6) rows(1))
```

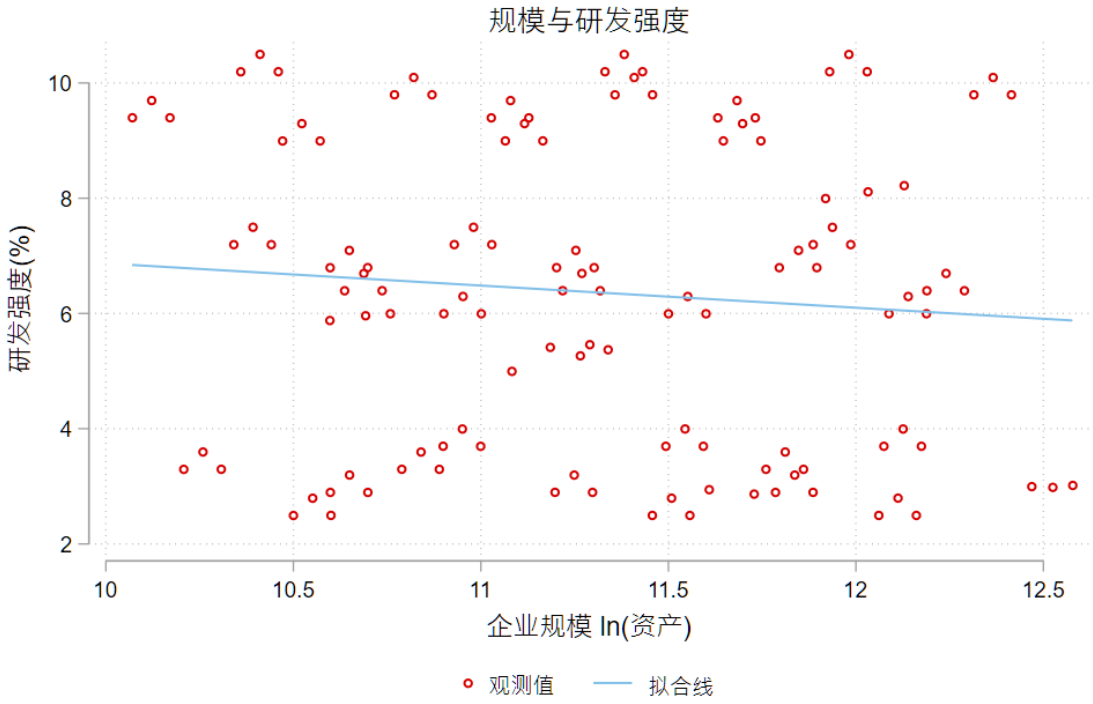

图 2.5: 规模与研发强度的散点图与拟合线：点分成上中下三条带。

拟合线几乎水平，似乎「规模和研发强度没什么关系」——但散点暴露了一件更重要的事：所有点分成了上、中、下三条带，那正是三个行业。也就是说，表面上的「无关」，背后其实是行业在主导。散点图让你看见了这种隐藏的分层，提醒你：若不加区分地对全样本回归，很可能把行业差异误当成别的东西。这就是图形帮你发现潜在样本结构与选择问题的价值——比任何统计量都直观。

2.7.4 趋势：时间趋势图

要查的问题：变量随时间怎么变？各组趋势一样吗？有没有断点？

```
preserve
collapse (mean) rd_ratio, by(ind_s year)
twoway (connected rd_ratio year if ind_s==" 医药") ///
       (connected rd_ratio year if ind_s==" 电子") ///
       (connected rd_ratio year if ind_s==" 汽车"), ///
    legend(order(1 " 医药" 2 " 电子" 3 " 汽车") pos(6) rows(1)) xlabel(2020 2021 2022)
restore
```

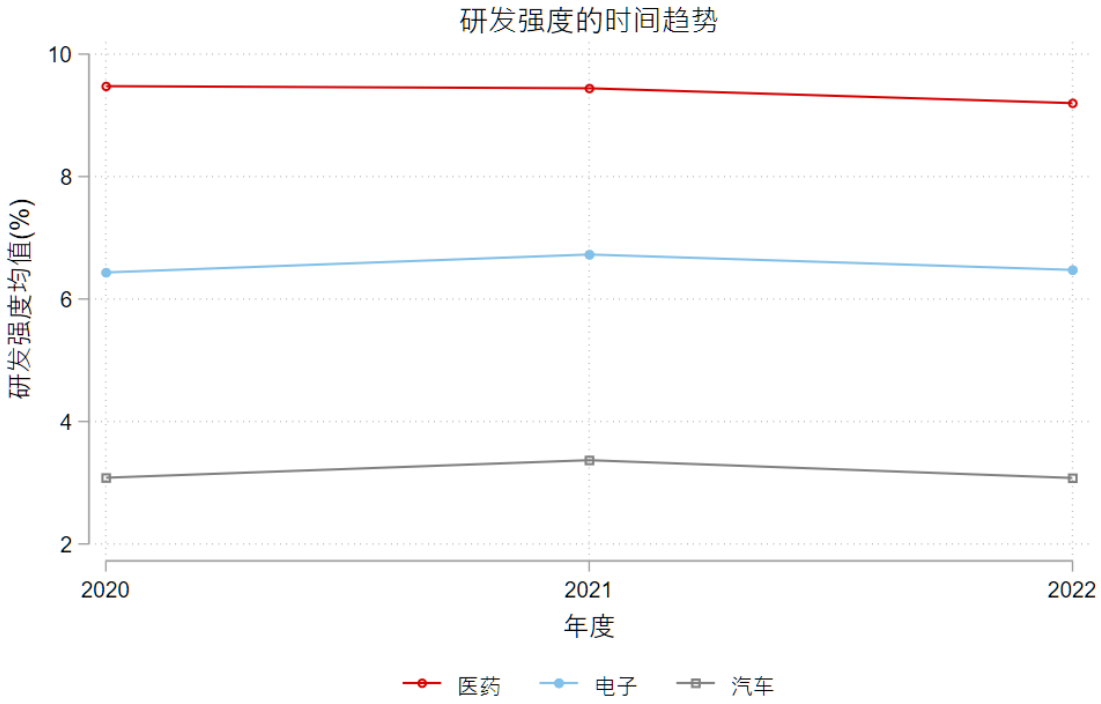

图 2.6: 三个行业研发强度的年度趋势。

三个行业的研发强度逐年都很平稳、层次稳定。真正要警惕的是突变：如果某一年出现跳变，就得回头查清楚——是政策冲击、是统计口径变了，还是数据出了错 (样本断点)。趋势图正是用来第一时间发现这种断裂的。

2.7.5 组间均值：分组均值图

要查的问题：各组的平均水平差多少？

```
graph bar (mean) rd_ratio, over(ind_s) ytitle(" 研发强度均值 (%)")
```

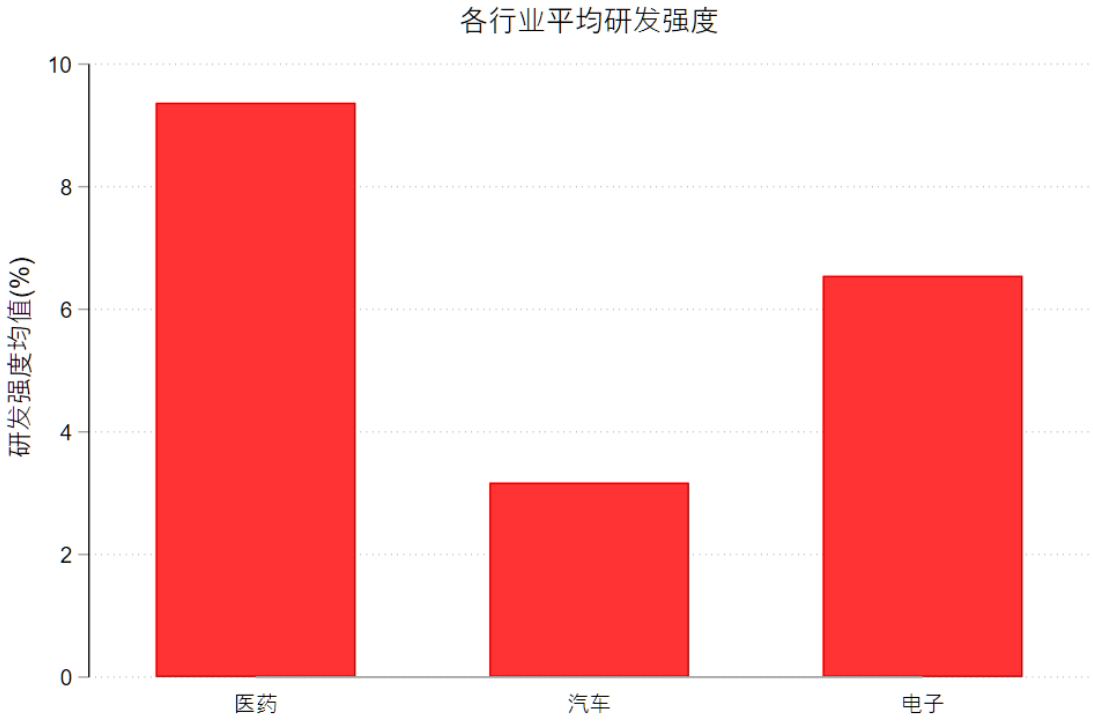

图 2.7: 各行业平均研发强度的条形图。

把箱线图的「分布」进一步压成「一个数」：医药 9.37%、电子 6.55%、汽车 3.17%(对应 `tabstat rd_ratio, by(ind_s)`)，组间高下立判。分组均值图适合快速比较组间水平，但也因为只剩一个数，会掩盖组内的离散和离群——所以它常和箱线图搭配使用，一个看水平、一个看分布。

2.7.6 一图一问清单

回过头看，六类图各司其职，对应着数据处理里最该盯的几个问题：

要查什么	用哪张图
分布形态、要不要变换	直方图 / 密度图
离群值	箱线图
组间差异	分组箱线图 / 分组均值图
两变量关系、异常点、隐藏分层	散点图
时间变化、样本断点	时间趋势图
潜在样本结构与选择	分组散点 / 分组密度

AI 协作：让 AI 按变量类型建议该画哪些图

不确定一个变量该画什么图来检查，就把它的类型和你关心的问题交给 AI，让它列一份 EDA 图形清单，并说明每张图能查出什么。
任务说明书 (提示词模板)：

我有以下变量：[逐个列出：名称、类型(连续/类别/时间)、含义]，研究问题是 [一句话]。
请建议一套探索性图形(EDA)清单：每一项写清「画哪种图、对哪些变量、能帮我检查什么问题」，并优先安排最可能暴露数据问题(离群、缺失、分组结构、样本选择)的图。
给出可运行的 Stata 代码骨架，图例统一放图形下方。

推荐 **skill**：`core/04-data-cleaning-planner`。没有合适提示词？见 附录 A 或用 `find-skills` 检索安装。

一句话收束：EDA 的精髓是一图一问——先想清要检查什么，再选图；图是用来发现问题的，不是用来装饰论文的。清洗、变换、建模的每一步，都该顺手画一张图回头看看。

2.8 留痕与可复现

一份干净的面板有了。可如果别人 (或半年后的你) 问一句：这张表怎么来的？样本为什么正好是这么多？每个变量什么口径？——你答得上来吗？答不上来，就等于不可复现。这一节讲三样必须随数据一起留下的东西：样本筛选日志、变量字典、匹配率与样本流失，外加两条最省心的习惯。

2.8.1 样本筛选日志

从原始数据走到分析样本，每筛一次就记一笔——留下「样本量是怎么变的」这本账。做法很朴素，就是在每一步后 `count`：

```
use "$D/firm_finance_2021.dta", clear
append using "$D/firm_finance_2022.dta"
count                                                    // 步骤 0
mvdecode rd, mv(-99)
replace sales = 108000 if stkcd=="000101" & year==2022   // 修正录入错误，不删样本
gen double rd_ratio = 100*rd/sales
drop if missing(rd_ratio)
count                                                    // 步骤 1
```

把这些数整理成一张筛选日志：

步骤	操作	样本量	变化
0	追加两批财务表	126	—
1	剔除研发支出缺失	124	−2
2	合并行业名 (匹配率 100%)	124	0

步骤	操作	样本量	变化

注意那条离群的营收 (000101) 是修正、不是删除，所以样本量不减。有了这本账，「样本为什么是 124」一目了然，审稿人问起一句话答清；反过来，若哪一步样本莫名掉了一大截，日志会第一时间报警。

2.8.2 匹配率与样本流失

merge 是最容易悄悄丢样本的地方，所以每次合并后都记下 _merge 的分布 (匹配率) 和前后行数：

```
merge m:1 indcd using "industry_lookup_upper.dta"
tab _merge
```

```
   Matching result from merge |      Freq.     Percent
------------------------------+-----------------------
                  Matched (3) |        124      100.00
```

这次匹配率 100%、零流失。回想上一节：同样的合并，脏键时匹配率只有 67%——那不是数据少了，而是键没洗干净。若匹配率偏低又没有记录，你很可能拿着缺了三分之一的样本还浑然不觉。匹配率就是合并这一步的安全带。

2.8.3 变量字典

给分析样本里的关键变量做一张字典，交代 N、取值范围和含义。codebook ..., compact 一行就能生成一份雏形：

```
codebook stkcd year rd_ratio size lev_num, compact
```

```
Variable   Obs Unique      Mean       Min       Max  Label
------------------------------------------------------------------------------
stkcd      124     42         .         .         .  证券代码
year       124      3  2021.008      2020      2022  年度
rd_ratio   124    124  6.364036  2.498888   10.5012  研发强度(%)
size       124    124  11.31305  10.07002  12.57764  企业规模 ln(资产)
lev_num    124     43  .4475242        .3       .59  资产负债率
```

这里藏着一个提醒：stkcd 的均值是「.」，因为它是编号、不是量 (呼应上一节的判断)；而字典能生成得这么整齐，前提是你给每个构造出来的变量 (rd_ratio、size) 都写了 label variable——没写标签的变量，就是没写清口径的变量。字典让变量口径不会随时间失忆 (这正是 §2.1 那份数据字典的延续，只不过现在是分析样本的版本)。

AI 协作：让 AI 生成样本筛选日志与变量构造说明

处理做完，让 AI 读你的 do 文件，替你把日志和变量说明的初稿整理出来，你再核对补全。
任务说明书 (提示词模板)：

下面是我的数据处理 do 文件：[粘贴]。请据此生成两份文档：
(1) 样本筛选日志：逐步列出每一步操作、样本量、增减和筛选理由；
(2) 变量构造说明：每个衍生变量的定义、公式、单位、口径。
对代码里没交代清楚的口径或理由，请标出来让我补充，不要替我编造。

推荐 **skill**：`core/06-repro-logger`(复现日志器)。没有合适提示词？见 附录 A 或用 `find-skills` 检索安装。

2.8.4 两条最省心的习惯

- `data_raw` 只写不改：原始数据放进一个只读的 `data_raw/` 文件夹，所有清洗都写成代码、结果输出到别处，原始文件永不手动改。这样任何时候都能从原始数据一键复现——这是 Gentzkow 和 Shapiro《Code and Data for the Social Sciences》反复强调的一条。
- 文件命名说人话：文件名带上内容、粒度和时间范围，如 `firm_year_clean_2020_2022.dta`，而不是 `data_final_v3_really_final.dta`。半年后你会感谢自己。

这两条都不花时间，却能把「复现」从一句空话变成随手可做的事。

样本日志、变量字典、匹配率——这三样不是交差用的附件，而是你自己的保险。数据处理里最贵的从来不是算力，是「三个月后没人 (包括你) 说得清这份数据怎么来的」。留痕，就是给未来的自己和读者留一条回得去的路。下一节复现顶刊数据时，正是靠这条路，我们才谈得上可控、可复现、可验证。

2.9 复现 Lane 的一小段

前面所有的本事——看结构、合并、清洗、构造变量、留痕——现在把它们用到一篇真论文上。我们拿 Lane (2025, QJE) 那篇关于韩国重化工业政策的文章，打开它的复现包，用 AI 与 Agent 的工作流复现其数据处理的一小段，并验证结果。论文正文可直接打开：PDF。目标不是把整篇复现完，而是学会一件事：借助 **AI** 处理真实数据时，怎么保证整个过程可控、可复现、可验证。

2.9.1 复现包里有什么、没有什么

课程期间，Lane 的完整官方复现包已放在 `examples/Lane_2025_QJE_paper_codes/replicationpackage/`，官方来源为 Harvard Dataverse。要取得完整目录，建议用 GitHub Desktop 克隆仓库；操作见课件使用说明和GitHub Desktop 教程。打开它的数据文件夹，先看 `data/input/`：其中的文件名

常带着 `_cleaned4reg`(cleaned for regression，供回归用的干净数据)，而 `data/intermediate/` 用于存放运行时生成的中间结果。若克隆后文件缺失，按附录 C 的放置说明下载官方包并保持同一目录结构。这透露了一个重要事实：这个包给你的是已经清洗好的分析面板，而从原始数据 (韩国工业普查、贸易数据、投入产出表) 到这份干净面板的清洗过程，作者放在了上游、没有随包发布。

这是复现的第一课：顶刊复现包常常只给 **cleaned data**。学会识别数据的三个层次——原始 **(raw)**/ 清洗后 **(cleaned)**/ 结果 **(results)**——你才知道自己站在链条的哪一环、能复现哪一段。所以本节要做的，不是「从原始造干净」(那段作者没给)，而是从作者的干净面板出发，做我们能做、也最能学到东西的几步数据处理动作。相关的项目结构见 第 1 讲。

2.9.2 认结构、构造处理变量

打开主分析面板，用上这一讲开头就练的「先看清结构」的本事：

```
* 在仓库根目录运行时，先进入课程暂存的官方复现包
cd "examples/Lane_2025_QJE_paper_codes/replicationpackage"
use "data/input/mms_merged_harmonized_panel_cleaned4reg_5digit.dta", clear
describe, short
codebook hci year, compact
```

```
Observations:      4,726
   Variables:         28              Sorted by: id  year

Variable     Obs Unique      Mean    Min    Max  Label
-------------------------------------------------------------------------------
hci         4726      2   .366907      0      1  Heavy and Chemical Industry ...
year        4726     17      1978   1970   1986  Year
```

一眼就能认出：这是一个行业-年度面板 (`id` 是 5 位行业代码、`year` 是年份)，和本讲小林的公司-年度面板结构同构——只不过观察单位是「行业」而非「公司」。主键是 `id year`，`hci` 是「是否重化工目标行业」的哑变量 (约 37% 的行业-年被列为目标)。

接着构造处理变量——这正是 §2.4 讲的变量构造，用在真数据上。韩国 1973 年启动重化工业化政策，目标行业即处理组。Lane 用一行代码合成处理变量：

```
gen treat = (hci==1 & year>=1973)   // 目标行业 × 政策启动后
tab treat
```

```
      treat |      Freq.    Percent
------------+-----------------
          0 |      3,298      69.78
          1 |      1,428      30.22
```

一行代码，把「政策目标行业」和「政策时点」合成了处理变量 (1428 个行业-年进入处理组)——这就是第 1 讲说的「政策怎么进入数据」。你已经能读懂、也能复现顶刊的这一步了。

2.9.3 可控、可复现、可验证

现在把 AI 放进来。复现常常撞上多语言：Lane 的估计用 Stata、出图表用 R。假设你面对的是作者的一段 R 代码 (比如读入某张政策表、改列名、按行业码 `merge`)。传统做法要么手敲、要么直接跑作者代码；AI 工作流的做法是：

- 可控：让 AI 把作者的 R 片段翻译成你熟悉的 Stata / Python(就是 §2.4 那条「翻译」协作)，你逐行看懂、确认逻辑，而不是黑箱照跑；
- 可复现：你自己的每一步都写成代码，配上样本筛选日志和变量字典 (§2.7)，别人能从原始复现包一键重跑；
- 可验证：把作者的原始代码借 AI 跑一遍，做成一个对照测试版；用它的结果和你 AI 复现的结果逐项比对。

比对就照一张检查清单来，一项一项打勾：

检查项	作者原版	我的 AI 复现版	一致
观测数	4,726	4,726	是
处理组占比	30.2%	30.2%	是
主键 `id year` 唯一	是	是	是

万一某项对不上，回退——不是去改数据迁就结果，而是回头查：是提示词哪里描述不清？哪一步逻辑翻译错了？改提示词或步骤，再跑一次。这个「发现偏差 → 回退修正」的循环，正是 AI 数据处理可靠性的来源。

迁移技能：课程包与官方来源如何核对

本讲所需的 Lane 复现包已经暂存于仓库，课堂上不必再下载。它的官方来源是 Harvard Dataverse (DOI `10.7910/DVN/VJECHN`)；今后面对其他论文时，才需要按 DOI 下载。Dataverse 提供按 DOI 打包下载的 API，大致形如：

```
curl -L -o lane2025_replication.zip \
  "https://dataverse.harvard.edu/api/access/dataset/:persistentId/?persistentId=doi:10.7
```

任务说明书 (提示词模板)：

```
我正在使用课程仓库中暂存的 Lane (2025) 复现包，官方 DOI 是 10.7910/DVN/VJECHN。
请先列出 `replicationpackage/` 的目录结构，帮我定位主脚本、数据文件夹与输出文件夹；
再对照官方 Dataverse 的 README，说明本地副本是否完整。对于另一篇论文，请另行生成 curl 或
```

推荐 **skill**：`core/03-replication-navigator`(复现包导航，见第 1 讲)；不确定端点时用 `find-skills` 或让 AI 先核对官方 API 文档，不要照抄未经验证的地址。

走完这一段，回头看整讲——你不再是「记住 `merge`、`collapse`、`destring` 怎么写」，而是：面对一份真实数据，能看清它的结构和层次、描述清楚要做什么、调出合适的技能去做，并用日志和对照守住可控、可复现、可验证。这就是 AI 时代做数据处理的样子：把命令交给技能，把判断留给自己。

2.10 经验法则与红线

前面各节把常见操作、适用场景和陷阱都走了一遍。这一节把它们浓缩成两张清单：一张是经验法则，帮你更快做对判断；一张是不可破除的红线，既守住自己不出大错，也用来审查 agent 有没有偷工减料。

一些经验法则 (帮你少走弯路，但都不是死规矩，最终看数据和研究问题):

- 规模类变量 (总资产、营收、市值) 通常取对数；一旦取了对数，一般就不必再纠结离群值——对数已经把极端值压平了。
- 比率和哑变量 (资产负债率、是否国企) 不取对数。
- 类别变量若某一类占比极低 (比如不到 1%)，考虑并入相近的类别或直接剔除，否则这一类的估计很不稳。
- 缺失少 (比如低于 5%) 且看着随机，可以删；缺失多、或明显非随机 (研发缺失往往集中在没有研发的公司)，删了会让样本偏向某一类，更稳妥的是填 0 再加一个「是否缺失」的指示变量。
- 缩尾多在 1%/99% 分位；缩尾后的均值、标准差应与做同类样本的已发表文献大体接近 (如中国上市公司资产负债率通常在 0.4–0.6)，偏离太多说明阈值设定或数据本身有问题。
- 单位要统一：元与万元、日频与年化、前复权与后复权——单位不一致是最隐蔽、也最致命的错。
- 主键不唯一时，先查是不是「伪重复」(同一公司同一年有合并报表和母公司报表两条)，按研究口径选定一种，别随机留一行。
- 合并后行数比主表多，是一对多膨胀；比主表少很多，是大量没匹配上。两种都要回头查。

❗ 不可破除的红线

这几条不是建议，是纪律。它们既约束你自己，也是你审查 agent 清洗结果时的尺子:

- 原始数据只读，绝不手动改；一切处理写成代码。
- 每一步样本增减都要有账 (样本筛选日志)，样本无故大变，必须查清才往下走。
- 合并匹配率异常必须报警，不能带着缺了一截的样本继续。
- 绝不为了让结果「好看」而在清洗阶段动手脚——删掉不利样本、专挑能凑出显著的阈值，都是红线。
- 变换、缩尾、口径的选择，要能对着文献和常识站得住。

为什么把红线摆在这么重要的位置？因为 agent 很能干，但它优化的是「完成你交代的任务」，未必是「对研究负责」。当你 (有意无意地) 暗示了想要的结果，它可能在清洗阶段悄悄迎

合——删几个碍事的样本、调一下阈值。这些动作技术上都「没报错」，却会毁掉整项研究的可信度。守住上面的红线，你才有底气用 agent，而不是被它带偏。

2.11 数据处理技能地图

有了前面的判断力，就到了本讲的落脚点：怎么把 skills 用起来，用得又快又安全。

先看数据类型，再挑技能。不同数据关注的坑不一样，该调的技能也不同:

- 宏观 / 地区数据 (GDP、人口、政策指标)：重点在口径与频率 (年 / 季 / 月)、跨来源的单位与地区编码对齐、缺失与断点；常要频率转换、按地区码合并。
- 微观 • 个体 / 家庭调查 (消费、收入)：重点在抽样权重、缺失机制与样本选择偏差、名义 / 实际口径、异常值；常要加权、缺失诊断。
- 微观 • 公司财务 (上市公司)：重点在主键 (证券代码 + 年度) 与伪重复、行业 / 地区码合并、缩尾、报表口径——就是本讲的主线。

面对一份新数据，先判断它属于哪一类、最该防哪些坑，再去找对应的技能。

课程自带的技能:

- `core/04-data-cleaning-planner`(数据清洗计划师)：生成清洗计划、核查 merge、解释各种变换的差别、按变量类型荐图、探索数据形态、规划文本处理。前面各个 callout 调的都是它。
- `core/06-repro-logger`(复现日志器)：生成样本筛选日志与变量构造说明。
- `core/08-data-audit`(数据审计员)：拿着上面那张红线清单，逐条审查一份清洗结果——样本流失有没有账、匹配率正不正常、离群是否被无差别缩尾、变量口径与文献是否相近、有没有为迎合结果而筛样本，把可疑处列出来让你复核。它专门用来防 agent(或自己) 在清洗阶段做手脚。

> 怎么判断一个下载来的 skill 够不够用、怎么扩展
>
> 从朋友或社区仓库拿到一个数据处理 skill，别直接用，先按三步评估:
>
> 1. 读它的 `description` 和检查清单，看它覆盖的任务和你的数据类型对不对得上——一个为宏观数据写的 skill，未必懂公司财务的伪重复。
> 2. 用本讲的红线去检验它：它会查匹配率吗？会留样本日志吗？会先问口径再动手吗？缺哪条，就在提示词里、或在 skill 正文里补上对应的规则。
> 3. 确实没有合适的，就自己造：用 `skill-creator`，或照 04 的写法写一个 (YAML 只放 `name` 与 `description`，正文写清「何时用、提示词、输出约定、人工检查清单」)。去哪找现成的：用 `find-skills` 检索，或到社区仓库 (记得核对许可，只用允许再分发的)。

这就是本讲真正的落点：前面七八节练的判断，是为了让你有能力评估、挑选、扩展和监督这些技能——知道自己面对什么问题、该调哪个 skill、它有没有守住红线。命令会过时，技能会更新，但这套判断不会。

2.12 Python 附录

同一条数据旅程，用 Python(pandas) 也能走一遍——读入、合并、清洗、画图的思路完全一样，只是命令换了语言。完整代码见伴生文件 examples/ch02/ch02_python.ipynb，它用 pandas 复刻本讲的关键几步 (同样从 GitHub 联网读取那几张表)。

这里给出把 Stata 流程「翻译」成 Python 的示范提示词 (呼应本讲多处出现的「让 AI 跨语言」协作)：

> **AI 协作：把 Stata 数据处理翻译成 pandas**
>
> 下面是我的 Stata 数据处理片段(含 merge、collapse、缺失与离群处理)：[粘贴]。请翻译成等价的 pandas 代码，要求：
> (1) 逐句对应，并在注释里标出每一步对应原 Stata 的哪条命令；
> (2) 保留同样的检查(合并后的匹配率、样本量变化)；
> (3) 指出两种语言在缺失值、类型转换上的差异，哪些地方容易译错。

重点从来不是记住 pandas 的语法，而是同一套判断在不同语言里都成立——这正是 AI 帮你跨语言复现顶刊多源代码的地方。

2.13 小结与练习

这一讲，我们陪着一份数据从凌乱走到干净：先问清来源、口径与字典，再认结构、查主键，然后合并、聚合成一张表，接着处理缺失、离群、变换，用图形给数据体检，最后留下日志与字典、并把这套做法用到 Lane (2025) 的真实数据上。

回头看，每一步靠的都不是记住某条命令，而是同一个循环：先想清要做什么判断 → 描述清楚 → 调合适的技能去做 → 动手后用 **isid**、**_merge**、**duplicates**、图形去查。这份「判断清单」，比任何命令都值得带走。

> **练习**
>
> 以下练习都可用本讲数据 (global D "https://raw.githubusercontent.com/lianxhcn/PXa2026a/main/ 完成。
> **1.** 主键与唯一性：mktcap_monthly.dta 的主键是什么？写一条命令验证它；再说明为什么不能直接按 stkcd 把它 1:1 合并到公司-年度面板。
>
> > **参考答案**
> >
> > 主键是 stkcd ym(公司-月)。验证：use "$D/mktcap_monthly.dta", clear 后 isid stkcd ym(不报错即唯一)。不能按 stkcd 做 1:1，因为一家公司有 12 个月共 12 行，stkcd 远非唯一；且它是公司-月粒度，要先 collapse 到公司-年度再合并。
>
> **2.** 合并与匹配率：把 industry_lookup.dta 合并到财务面板时，若不清洗键会发生什

么？写出正确的合并流程，并说明合并后第一件要做的检查。

参考答案

不清洗键，"C36" 与对照表里的 "c36 " 对不上，汽车业公司全部 _merge==1(master only)，匹配率约 67%。正确流程：先 replace indcd = upper(strtrim(indcd)) 清洗对照表的键，再 merge m:1 indcd using ...；合并后第一件事是 tab _merge 看匹配率，确认没有异常的 master-only / using-only。

3. 缺失与离群：rd 里有 -99、sales 里有一个异常大的值。分别该怎么处理？为什么离群值不能一律缩尾？

参考答案

-99 是伪装成数字的缺失，先 mvdecode rd, mv(-99) 还原为 .。sales 的异常值要先画图看见、再查明来源：本讲那条是 000101 在 2022 年多打了一个 0(录入错误)，应改正而非缩尾。缩尾/截尾只留给无法核实、但确实极端的真离群——否则会掩盖本可修正的错误。

4. 选图：要检查「不同行业的研发强度是否有差异，以及各行业内部有没有离群」，该画哪一张图？若还想看「研发强度与企业规模的关系是否被行业干扰」，又该画哪张？

参考答案

前者用分组箱线图 (graph box rd_ratio, over(ind_s))，一张图同时看组间差异和组内离群。后者用散点图 (scatter rd_ratio size)，若点分成几条带，说明关系被行业分层干扰，提示要分组或控制行业。

5. 交给 **agent**，并审计它：让 AI/agent 用本讲数据，从两批财务表出发，完成「追加 → 清洗缺失与离群 → 合并行业 → 构造研发强度 → 产出样本筛选日志」的全流程。然后审计它的结果：匹配率是多少？样本从 126 变到几？它有没有把那条可修正的离群值当成真离群缩尾了？变量字典里每个构造变量都写清口径了吗？

参考答案要点

这道题没有标准代码，重点是审计意识。合格的复现应得到：追加后 126 行、剔除研发缺失后 124 行、合并行业匹配率 100%；那条 000101 2022 的离群应被识别为录入错误并改正 (而非缩尾)；rd_ratio、size 等构造变量应有 label。若 agent 直接缩尾了那条离群、或匹配率不到 100% 却没报警，就是要打回重做的信号——这正是「可验证」的含义。

2.14 延伸阅读

- 连享会 • 数据处理专题 (推文合集)：https://www.lianxh.cn/blogs/25.html

- `reshape` 一文读懂 (上 | 下)——长宽转换的更多情形；
- 离群值的识别与处理：https://www.lianxh.cn/details/175.html
- 正则表达式与文本分析：https://www.lianxh.cn/details/35.html——本讲降级的文本挖掘代码细节在此展开；
- 数据获取 (接口、下载、抓取)：连享会《金融数据获取》讲义；数据管理、清洗与 EDA 的系统内容，见数据分析讲义 `ds2026` 对应各章；
- 数据库进阶 (SQL / SQLite / DuckDB / Parquet) 与现代 DID 等方法，属高级班范围，参见 https://www.lianxh.cn/details/1811.html。

3 从总体到推断：实证分析的五层框架

本讲要点

- Population：研究总体、目标对象和外推边界；
- Sample：样本产生机制、抽样误差和样本选择偏误；
- Data：变量测量、代理变量、测量误差和数据质量；
- Model：模型设定、函数形式和误设风险；
- Inference：标准误、置信区间、显著性和不确定性表达；
- 随机抽样与随机分配的区别；
- 样本选择偏误与自选择偏误的基本直觉；
- 如何利用分组统计量、密度函数图、样本筛选表和政策背景说明，检查研究总体、分析样本、变量测量和模型设定是否相互匹配。
- 本讲用到的 skills：`core/02-paper-strategy`；
- 可运行伴生文件：`examples/ch03/ch03_main.do`(Stata；配套一份公司-年度面板数据随本讲一起提供，只做诊断、不做清洗)。

3.1 案例情境：同一个回归，为什么算出来的不是同一回事

先看一个几乎每位做实证研究的人都会遇到的情形。

我们想回答一个朴素的问题：研发投入更高的企业，是不是成长更快、绩效更好？手上有一份公司-年度面板数据，跑一个回归，系数为正且显著。看起来结论很清楚：研发有用。

但先不急于下结论，先看看这份数据是怎么来的。只有那些主动披露了研发支出的公司，才进得了这份样本。而在现实里，愿意把研发支出写进报表的公司，往往本就是研发密集、规模较大、经营较规范的那一批；大量不披露研发的公司，被排除在样本之外。

于是同一个回归，结果就可能有不止一种解释：它估计的，究竟是研发本身的作用，还是「本来就较强的公司，恰好也披露研发」这一事实？若是后者，那个显著为正的系数，可能大半来自样本的构成，而非研发的效果。要在这两种解释之间做出取舍，需要额外的条件和判断，这也正是识别策略要处理的问题。

这就是本讲要处理的核心问题。你手上的样本、数据和变量，很少恰好等于你心里想研究的那个对象；从「你想研究谁」到「你最后拿什么去跑回归」，中间隔着好几道传递，每一道都可能悄悄走样。看不见这些走样，就会把样本自身的特征，误当作总体的规律。

3.2 概念讲解：一条会走样的链条

把上面的问题一般化，一项实证研究其实是沿着这样一条链条展开的：

实证分析的五层框架

Population 总体 → **Sample** 样本 → **Data** 数据 → **Model** 模型 → **Inference** 推断

一个实证研究是否可信，不只取决于模型是否复杂，还取决于总体、样本、数据、模型和推断之间是否一致。

大纲把它概括为五层：总体 (Population)、样本 (Sample)、数据 (Data)、模型 (Model)、推断 (Inference)。但比「五个层次」更要紧的，是它们之间的那几个箭头：每个箭头都是一次传递，而每一次传递，口径都可能发生漂移——

- 从「总体」到「样本」：谁进了样本、谁没进，可能并不是随机的 (本讲开头那个披露问题，就出在这里)；
- 从「样本」到「数据」：你真正想量的东西 (比如「创新能力」)，未必量得准，手上的变量往往只是它的一个替身；
- 从「数据」到「你定义的变量」：同一个概念，换一种口径去构造结果变量 Y、处理变量 D、控制变量 X，结论可能就不一样；
- 到「模型」：模型的设定要和前面几层对得上，否则就是「误设」；
- 到「推断」：因为样本是抽出来的、本身会变，任何结论都带着不确定性，需要如实表达。

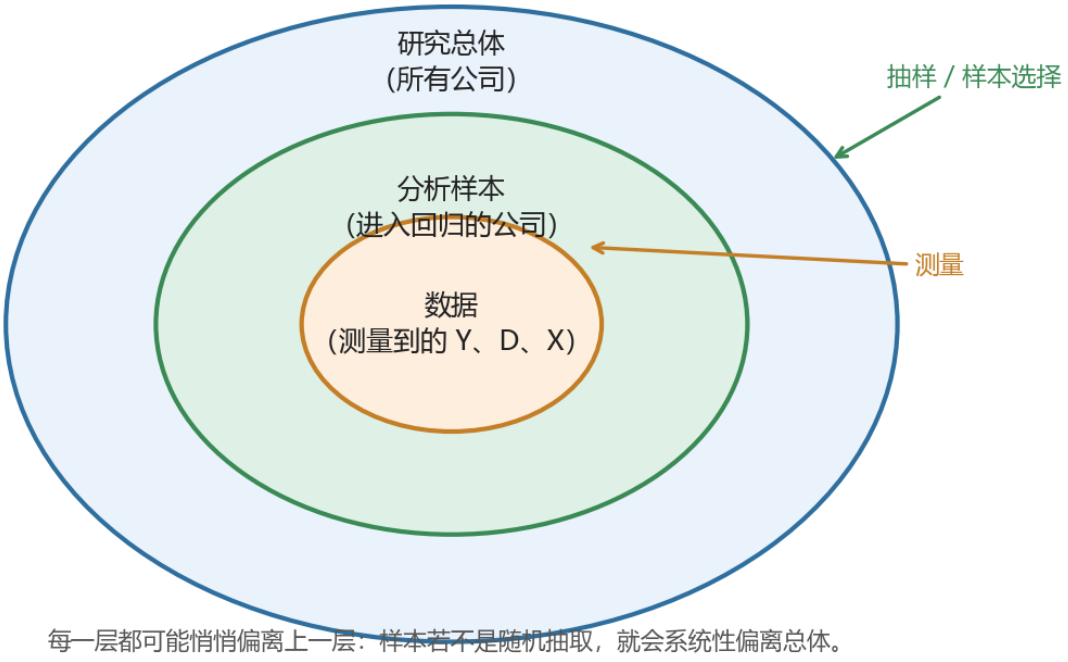

图 3.1: 总体、样本与数据是一层套一层的关系：每一次向内收窄，都可能偏离外面那一层

一项研究可不可信，往往不取决于模型有多复杂，而取决于这条链有没有在某一环悄悄断掉。下面我们一环一环地走。走的过程中你会慢慢体会到两件事：其一，结果不理想，多半坏在这条链的某一环，而不是回归命令写错了；其二，这条链也是理解因果推断、识别策略和内生性的那张地图——所谓内生性，多数时候就是这条链在某一处断了。理解它，是对自己整个分析过程多一分审慎的开始。

3.2.1 Population：你到底想对谁说话

一切的起点，是先想清楚「研究总体」：这项研究的结论，究竟想说给谁听？

研究总体不是一个抽象的大词，它由你的研究问题界定。问「A 股上市公司的研发是否提升绩效」，总体就是所有 A 股上市公司；问「中国所有企业」，总体就把大量未上市、无报表的中小企业也算进来——这是两个差别很大的总体。定义总体，本质上是在划一条边界：谁在里面，谁在外面。

这条边界还有一个常被忽略的名字：「外推边界」。你在某个总体上得到的结论，只能稳妥地推广到这个总体之内。用上市公司样本估出的「研发回报」，未必适用于小微企业；用某个试点城市得到的政策效果，未必能推广到全国。总体的隐含条件到哪里，你的结论就到哪里为止。一篇严谨的实证研究，会在一开始就把这条边界交代清楚，而不是把局部结论表述成普遍规律。

3.2.2 Sample：谁进了样本，谁被挡在门外

有了总体，下一步是从中得到「样本」。这一环最容易出问题，因为「样本怎么产生」常被当作理所当然，其实里面藏着两类性质完全不同的偏差。

先分清两种偏离：抽样误差与样本选择偏误。打个比方：总体是一整口鱼塘，样本是你捞上来的那一网鱼，你想用这一网的平均体重，去估整塘鱼的平均体重。

- 如果在全塘「随机撒网」，那么每一网的平均体重都会围绕真实值上下波动——这次偏大、下次偏小，样本量越大，波动越小。这种波动叫「抽样误差」，它是随机的，不会系统性地偏向某一方，增加样本量即可缓解。
- 如果只在「岸边浅水」捞 (那里恰好多是小鱼)，那么无论捞多少网、样本多大，估出的平均体重都会系统性地偏小。这种偏差叫「样本选择偏误」，它是系统的，样本量再大也无法消除——你只会把一个有偏的估计算得越来越精确。

一句话记住区别：抽样误差是随机波动，样本选择偏误是系统偏向。前者靠更大的样本就能缓解，后者靠更大的样本只会把一个有偏的结论估得更精确，反而让人更确信。本讲开头的研发披露问题，正属于「只在岸边捞」——只看得见愿意披露的公司。

用记号把这层区别固定下来：设总体真值为 θ、某次抽样得到的估计为 $\hat{\theta}$。随机抽样下，$\hat{\theta}$ 会围绕 θ 上下波动、平均而言不偏，即 $E[\hat{\theta}] = \theta$；而在有选择的样本里，$\hat{\theta}$ 的平均本身就偏离真值，即 $E[\hat{\theta}] \neq \theta$。样本量能缩小前一种波动，却消不掉后一种偏离。

再分清两个「随机」：随机抽样与随机分配。二者都带「随机」二字，管的却是两件不同的事。

- 「随机抽样」，是从总体中随机地抽取样本。它决定的是样本像不像总体——随机抽样的样本不存在选择偏误，可以稳妥外推；非随机抽样 (比如只抽披露公司) 则未必。
- 「随机分配」，是在样本内部随机决定谁接受处理、谁作对照 (典型如 A/B 测试：随机把用户分为看到促销和未看到促销两组)。它决定的是处理组与对照组可不可比——这是能否识别因果效应的关键。

二者相互独立，不宜混为一谈：你可能有一份随机抽样的样本，但企业是否上马某项目由其自己决定 (处理非随机，难谈因果)；也可能在一个代表性欠佳的样本里，做一场干净的随机分配实验 (内部可比，但外推需谨慎)。一个管外推，一个管因果，是两道各自要过的关。

最后，落到样本选择偏误的内核：谁进入样本、谁的结果被我们观察到，究竟由什么决定？

劳动经济学中一个经典的例子是女性工资。我们想研究教育对女性工资的影响，但数据里只有在工作的女性才有工资——没有工作的女性，其工资根本不存在于数据中。而是否工作并非随机：保留工资较高、不工作机会成本较大的人，更可能选择不工作。于是「在工作女性」这个样本，与「所有适龄女性」这个总体，在一些不可观测的因素上系统性地不同。你在这个样本上估出的教育回报，已经掺入了「谁会去工作」的信息。

下面这张模拟图直观地展示了这种偏差：当我们按结果变量去筛选样本时，OLS 拟合线会明显偏离全样本给出的结果 (这里先作简要示意，课堂上再展开)。

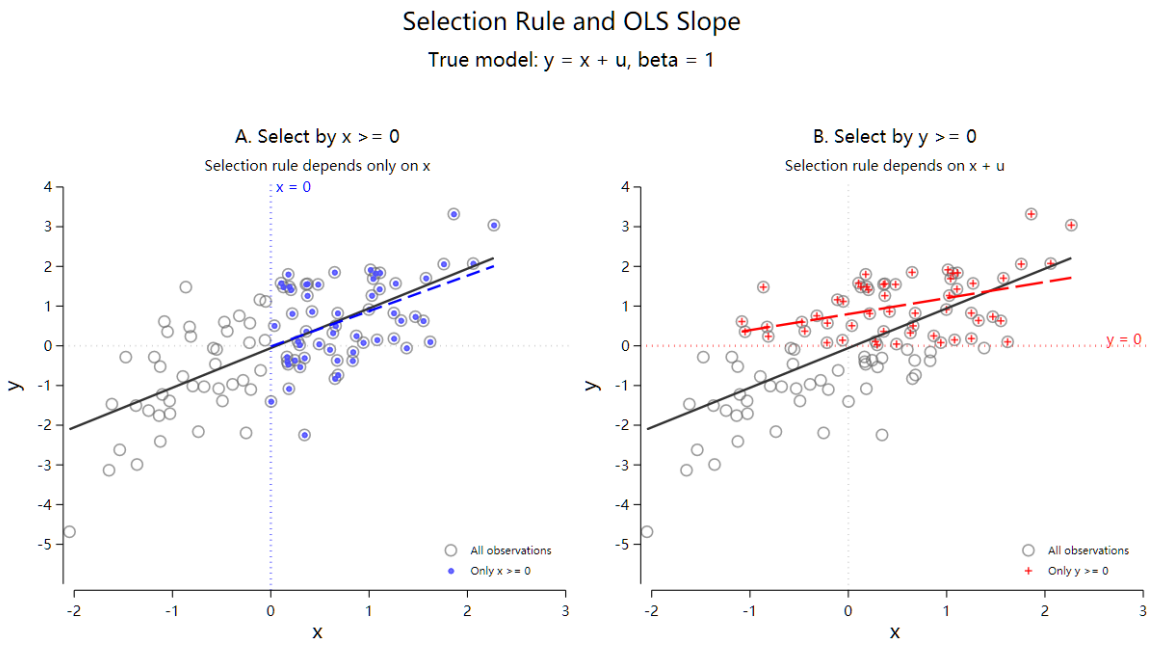

图 3.2: 按结果变量筛选样本，会让 OLS 拟合线发生系统性偏移 (模拟示意)

把「谁进入样本」写成一条规则会更清楚：设 $S_i = 1$ 表示第 i 个对象进入样本，它由某个规则决定，$S_i = f(\text{一些因素})$。这条规则就是「选择函数」。样本选择偏误的根源在于：选择函数中的因素，恰好与我们关心的结果变量相关。是否披露研发 (选择) 与企业本身强弱 (结果) 相关，是否工作 (选择) 与潜在工资 (结果) 相关，都是如此。

顺带厘清一对常被混用的概念——「样本选择偏误」与「自选择偏误」：

- 若是外部规则把一部分对象挡在数据之外，我们只观察到其中一部分 (只观察到上市公司、只观察到就业者、只观察到披露研发的公司)，偏差出在「谁被观察到」，这是狭义的「样本选择」；
- 若结果在所有对象身上都可观察，只是对象自己选择要不要接受某种处理 (企业自行决定是否申请某项补贴)，偏差出在「谁去接受处理」，这是「自选择」。

二者都会让简单比较失真，但失真的位置不同、对策也不同。这条分界线在因果推断中会反复出现，值得早些建立直觉。

> **选择函数点到为止，校正方法不展开**
>
> 这里我们只需建立一个直觉：样本并非凭空而来，它由一条选择函数产生；一旦这条函数与结果相关，简单比较就会有偏。至于如何用模型刻画并校正这种选择 (如 Heckman 的思路)，属于进阶内容，本讲不展开。有兴趣的读者，可参阅连享会《因果推断》讲义中讨论样本选择偏误的一章；更系统的估计方法，则留待高级班。

3.2.3 Data：你量到的，是不是你想量的

样本确定了「观察谁」，但我们最终能分析的，不是对象本身，而是对象在数据里留下的测量值。从「样本」到「数据」这一步，问的是另一个问题：你量到的东西，是不是你真正想量的东西？

很多我们真正关心的概念，本身无法直接观测。「创新能力」「公司治理水平」「企业风险」都是如此。研究中只能退而求其次，用一个可观测的变量去代替它，这个替身就叫「代理变量」。用研发支出代理创新能力、用董事会独立性代理治理水平、用股价波动代理风险，都是常见做法。代理变量用起来方便，但要始终记得：它和它想代表的那个概念之间，总隔着一段距离。研发支出高，未必等于创新能力强，它还掺着企业规模、行业属性、会计口径等许多其他因素。

即便是可以直接测量的变量，也难免带着「测量误差」：记真值为 x^*、观测值为 x，二者往往并不完全相等，$x = x^* + e$。测量误差落在不同变量上，后果并不一样。落在被解释变量上，多数时候只是放大噪声、降低估计精度；落在核心解释变量上，则可能把系数系统性地拉向零——你明明测到了一个真实存在的关系，却因为变量量得不准，把它的强度低估了。这一点，是许多「效应比预期弱」的结果背后，常被忽略的原因。

数据质量则是这一环的底座：测量口径是否前后一致、有没有系统性的缺失或录入错误，直接决定后面所有分析是否站得住。数据不干净，再精巧的模型也只是把脏数据算得更精确。

3.2.4 变量口径：Y、D、X 都是你亲手定义的

有了数据，还要把它组织成模型能用的变量：结果变量 Y、处理变量 D、控制变量 X。这一步常被当成纯技术操作，其实处处是判断——同一个概念，换一种口径去构造，结论可能就不一样。

先看结果变量 Y。想衡量「企业绩效」，用 ROA、ROE 还是 Tobin's Q？三者口径不同、侧重不同，同一份数据、同一个自变量，换一个 Y 跑出来的结论完全可能不一致。选哪一个，取决于你的研究问题问的是哪一种绩效，而不是哪个变量跑出来更显著。

再看处理变量 D，也就是「谁算受了影响」。这一步的口径尤其容易和前面的总体、样本对不上。以一项环保督察为例：把处理组定义为「被督察省份的所有企业」，还是「被督察地区里的高污染企业」，是两个差别很大的 D。前者把大量本就与政策无关的企业也算作处理组，会把真实效应稀释掉。政策作用在哪一层，D 就该定义在哪一层，否则模型比较的根本不是政策本身。

最后是控制变量 X。控制变量并非越多越好。如果你控制了一个本身也受处理变量影响的变量 (它其实是结果的一部分)，就会把真实效应挡掉一截——这类变量被称为「坏控制」。该控制什么、不该控制什么，取决于你对这些变量之间关系的判断，而不是把手边所有变量不加甄别地放进回归。

有关控制变量的讨论，参见如下推文：

- 付一帆, 2022, Stata：控制变量与核心解释变量地位对等吗？.
- 张雪娇, 2022, 控制变量如何选？大牛们的 10 条建议.
- 曹昊煜, 2022, A-理论部分：控制变量！控制变量！Good-Controls-Bad-Controls.
- 李烨阳, 2022, B-Stata 模拟：控制变量！控制变量！Good-Controls-Bad-Controls.
- 秦利宾, 2021, 控制变量！控制变量！.
- 连玉君, 张静瑶, 2020, 加入控制变量后结果悲催了！.

一句话：Y、D、X 的口径都是研究者定的，定的时候就要回头对齐总体与样本——口径一变，整条链的含义都会跟着变。

3.2.5 Model：模型要和前面几层对得上

到这里才轮到「模型」。不少人以为实证的核心是选模型，其实模型是被前面几层逼出来的：它要和你的总体、样本、变量口径都对得上。

模型设定里，最基本的是「函数形式」。可以把模型写成 $Y = f(X) + \varepsilon$：所谓设定模型，就是选定这个 f 的形状——变量之间是线性关系，还是取对数后才近似线性，或者本身是非线性的？f 选错了，就是「误设」。一个本该取对数的右偏变量，若硬按原始尺度进入回归，估出的系数既难解释，也可能有偏。

更隐蔽的误设，来自层级不匹配。还是上面那个督察的例子：政策实际作用在省级，你却用企业级数据、把处理和比较都放在企业层面，模型比较的对象就未必对应政策真正影响的层级。模型误设的本质，是模型描述的世界，和数据实际生成的方式对不上。这也正是内生性的重要来源之一。

3.2.6 Inference：因为样本会变，结论带着不确定性

走到最后一环。假设前面四层都对齐了，模型也估出了一个系数，我们还差一个问题：这个数字有多可靠？

可靠性问题的根子，还在「样本」那一环——样本是从总体里抽出来的，换一批样本，估计值就会变。正因为估计值会随样本波动，任何结论都带着不确定性，而推断，就是把这份不确定性如实表达出来。

- 「标准误」度量的就是这种波动：如果反复抽样、反复估计，系数大致会在多大范围内起伏。
- 「置信区间」把它写成一个区间，告诉你估计值大致落在哪一段，而不是假装它是一个精确的点。
- 「显著性」回答的是：观察到的这个关系，有多大可能只是抽样波动造成的假象。

写成记号，一个系数的 95% 置信区间大致是 $\hat{\beta} \pm 1.96\,\widehat{se}(\hat{\beta})$：以估计值为中心，向两侧各留出约两个标准误的余地。它的含义常被误解——并非「真值有 95% 的概率落在这个区间里」，而是：若反复抽样、反复构造这样的区间，长期看约有 95% 的区间会盖住真值。下图用一个模拟把这层意思画了出来。

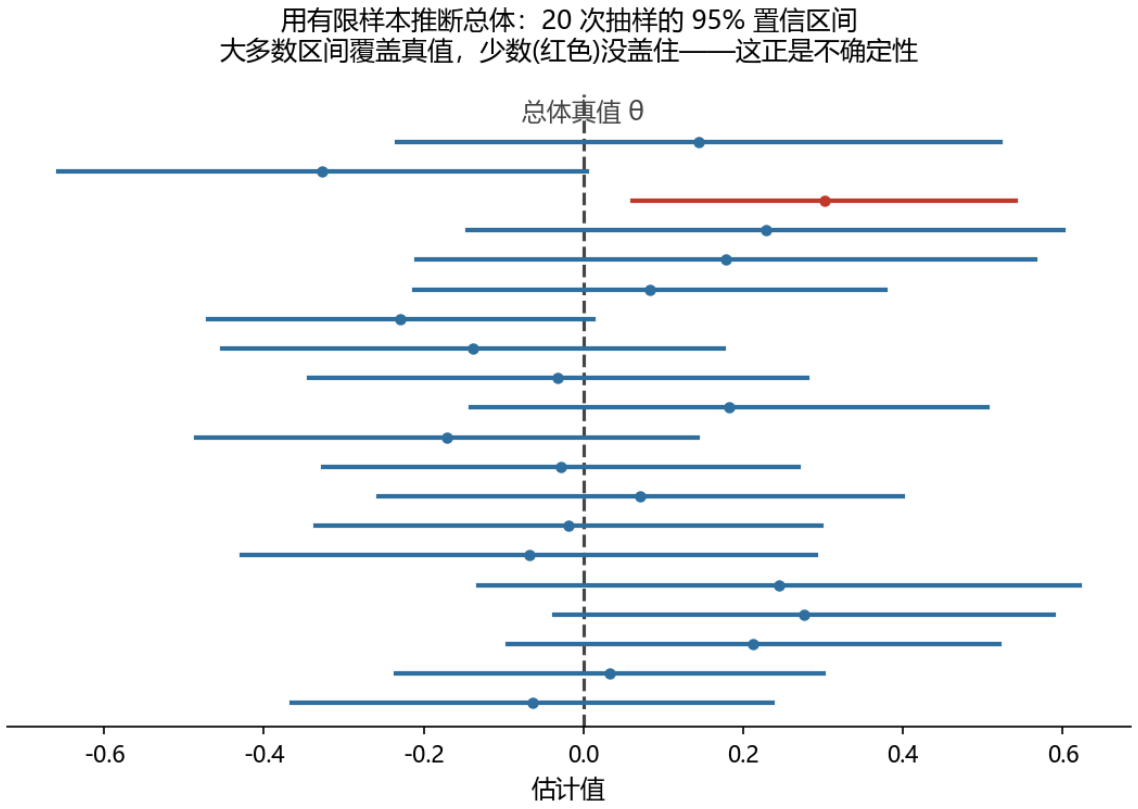

图 3.3: 用有限样本推断总体：反复抽样构造的 95% 置信区间，多数覆盖真值、少数 (红色) 没盖住

这里要守住一条底线：显著，不等于证据充分；不显著，也不等于没有效应。显著性高度依赖样本量——样本足够大，再微弱的关系也会显著；样本很小，真实的效应也可能测不出显著。把「p 值小」直接当成「结论成立」，是实证中最常见的误读之一。至于标准误具体怎么算 (异方差、聚类等)，属于回归分析的细节，这里只需先记住它表达的是什么。

3.2.7 这条链，正是因果推断的地基

回头看这五层，会发现一件事：实证研究里那些让人头疼的问题，几乎都是这条链在某一处断了。

- 样本选择偏误，是「总体 → 样本」断了；
- 测量误差、代理变量失真，是「样本 → 数据」断了；
- 变量口径不当、坏控制、遗漏变量，是「数据 → 变量 → 模型」断了；
- 模型误设，是「模型」这一环与数据生成方式对不上。

这些断裂，在因果推断里有一个统一的名字：「内生性」。所谓识别策略，说到底就是想办法让某一处断掉的比较重新变干净——借助随机分配、找一个外生的变动，或构造一个可信的对照。这些方法本身怎么做，是高级班的内容 (可先浏览连享会关于现代因果推断方法的整理)；但它们要解决的问题，全都藏在这条链里。

所以，把这条链看懂，不只是为了这一讲。它真正的价值在于：当你自己的实证结果不理想时，能顺着这条链一环一环去找问题出在哪，而不是盲目地换命令、加控制变量。也正因为看清了每一环都可能出错，你才会对自己的每一个分析步骤，多一分应有的审慎。

3.3 从概念到工具

链条讲通了，接下来把「怎么检查链条有没有断」落成几件具体、可上手的动作。它们不是新方法，而是从前面的概念里自然长出来的检查手段——每一件，都对准链条上最容易断的一处。

3.3.1 分组统计量：两组到底可不可比

回到本讲开头的问题：披露研发的公司和不披露的公司，可能本来就不是一类企业。要判断两组能不能直接比较，最朴素的办法，就是把两组的关键特征并排摆出来看。

按组计算均值、中位数、标准差，再逐个变量对比：处理组和对照组在规模、盈利、行业分布上差多少？如果两组在这些特征上就已经差得很远，那么「处理后的差异」很可能掺着「两组本来就不同」的成分，而未必是处理本身的作用。分组统计量，是检查可比性的第一道体检。差距越大，越要回头追问选择机制与变量口径。

3.3.2 假设检验：组间差异，是信号还是抽样波动

分组统计量告诉我们两组均值不一样，但先别急着下结论——样本是从总体里抽出来的，本身就带着抽样误差，两组的差异有可能只是这一次抽样碰巧造成的。于是问题变成：眼前这个差异，是真实存在的信号，还是抽样波动的假象？

这正是「假设检验」要回答的。它的直觉很朴素：先假设两组其实没有差别，再问——如果真是这样，仅凭抽样波动，能不能碰巧出现眼前这么大的差异？如果这种巧合几乎不可能发生 (也就是 p 值很小)，我们才有理由说，差异不是运气，而是真的。两组均值差的 t 检验，做的就是这件事。

可以看到，Inference 那一层讲的标准误、p 值、置信区间，到这里第一次派上了用场——它们不是报告时才用的摆设，而是从「样本会变」这件事里长出来的判断工具。同时也别忘了那条底线：样本很大时再小的差异都会显著，所以既要看 p 值，也要看差异本身有多大、有没有实际意义。

3.3.3 密度函数图：两组是不是一套形状

均值一样，不代表两组长得一样。两组可能均值接近，分布形态却大不相同：一组集中、一组分散，或者某一组在某个区间整片缺失。这些，光看几个统计量是看不出来的。

把两组的核密度曲线叠在一张图上，一眼就能看出：分布是否错开、有没有离群的长尾、有没有明显的断点，以及有没有样本选择的痕迹——比如某一组的低值区整片为空，说明这部分对象根本没进样本。密度图比分组统计量更细：统计量是几个数，密度图是整条分布。

这里真正要记住的，不是画图命令，而是一种意识：拿到一个连续变量，先想想它的分布长什么样、在不同子样本里差在哪。至于怎么画，Stata 和 Python 都很容易实现，也不必背专门的命令——实践中更好的做法，是把绘图直接交给 AI：说清要比较哪个变量、按什么分组，让它生成代码即可 (本讲代码实操会给出这样一段提示词)。

3.3.4 样本筛选表：口径怎么一步步筛出来的

从最初拿到的数据，到最终跑回归的分析样本，中间往往删掉了不少观测：剔除金融行业、剔除数据缺失、剔除异常值……每一步都在改变「样本是谁」。样本筛选表，就是把这个过程一行一行记下来：每一步剔除了什么、还剩多少观测、多少家公司。

它有两个作用。一是透明可复核：别人 (包括半年后的你自己) 能照着复现出同一份样本。二是对样本选择保持警觉：如果某一步一下子筛掉了大量某类对象，就要当心——最终样本会不会已经系统性地偏离了总体？

3.3.5 政策背景说明：制度事实是判断的锚

前面几件工具都在数据里做检查，但有些判断，数据本身给不出答案，得回到制度事实里找。处理组和对照组的划分合不合理、政策究竟在什么时点、作用在哪一层，这些都要靠把政策背景讲清楚来支撑。

一份好的政策背景说明，会交代：政策的官方名称与时间、直接作用的对象 (地区、行业还是企业)、处理组能否在数据里识别、以及最容易被质疑的地方。它是前面所有检查的锚点——脱离制度事实去谈样本和变量，很容易把技术上的巧合，误当成经济上的因果。

3.3.6 合起来：给整条链做一次一致性体检

这几件工具合在一起，就是对整条链做一次「一致性体检」：用分组统计量和假设检验查两组可不可比，用密度图查分布是否一致，用样本筛选表查样本是怎么来的，再用政策背景说明把这些判断锚在制度事实上。体检的目标只有一个：看总体、样本、数据、变量和模型是不是互相对得上。下一节，我们就在一份公司-年度面板上，把这套体检动作实际走一遍。

3.4 代码实操

把前面讲的检查工具，在一份结构真实的面板上依次跑一遍。数据随本讲提供 (`examples/ch03/`，模拟生成、可复现)，只做诊断、不做清洗；下面的代码与输出，都可在配套的 `ch03_main.do` 中复跑。

这份面板是 42 家上市公司 × 2020–2022 三年 = 126 个公司-年度观测，涵盖电子、医药、汽车三个行业。我们想研究「研发投入是否提升企业绩效」，但只有披露了研发支出的公司，`rd` 才可见——不披露的公司，进不了「研发—绩效」的分析样本。变量 `disclose` 标记是否披露，`size` 是 ln(总资产)，`roa` 是资产收益率。下面就来看：进入分析样本的披露组，和被挡在外面的未披露组，到底可不可比。

3.4.1 分组统计量：两组可比吗

先把两组的关键特征并排摆出来。

```
* 数据可联网直读（无需下载）；也可克隆仓库后把 $D 改成本地路径
global D "https://raw.githubusercontent.com/lianxhcn/PXa2026a/main/examples/ch03/data"
use "$D/firm_year.dta", clear
tabstat size roa sales lev, by(disclose) stat(mean sd) nototal
```

```
disclose |      size       roa     sales       lev
---------+----------------------------------------
  未披露 |  10.61037  .0416825  41349.62   .476616
         |  .6921522  .0284131  24650.36   .071487
---------+----------------------------------------
```

```
    披露 |  11.23799   .0626257   80705.29   .4959391
         |   .7374012   .0331485   42275.03   .0636639
-------------------------------------------------------
```

披露组的规模 (size 11.24 对 10.61)、盈利 (roa 6.3% 对 4.2%) 和营收 (sales 8.1 万对 4.1 万) 都系统性地高于未披露组，只有资产负债率 (lev) 两组接近。也就是说，进入分析样本的公司，本来就比落选的更大、更赚钱。这意味着「研发与绩效正相关」这个结论，可能有相当一部分来自「大公司既愿意披露研发，本身绩效也好」，而非研发的作用。

3.4.2 假设检验：差异是信号还是抽样波动

两组均值不一样，但样本是抽出来的，会不会只是抽样波动？用 ttable2 一次检验多个变量的组间差异。

```
ssc install ttable2, replace     // 首次运行需先安装
ttable2 size roa sales lev, by(disclose)
```

```
------------------------------------------------------------------------------
Variables   G1(未披露)       Mean1     G2(披露)       Mean2      MeanDiff
------------------------------------------------------------------------------
size          63             10.610       63          11.238     -0.628***
roa           63              0.042       63           0.063     -0.021***
sales         63             4.1e+04      63          8.1e+04   -3.9e+04***
lev           63              0.477       63           0.496     -0.019
------------------------------------------------------------------------------
```

size、roa、sales 的组间差异都在 1% 水平上显著 (***)，不是抽样波动，而是系统性差异。值得注意的是 lev 的差异 (-0.019) 并不显著：两组的不同并非无处不在，而是集中在与「大小、强弱」相关的维度上——恰恰是最可能干扰「研发—绩效」判断的那些维度。再提醒一次那条底线：这里的显著只说明差异真实存在，并不说明研发有效；恰恰相反，它是在告诉你，两组本就不是一类公司。

3.4.3 密度函数图：分布对不对得上

均值和检验给的是几个数，密度图能让我们看到整条分布。把两组的 size 核密度叠在一起：

```
twoway (kdensity size if disclose==1) (kdensity size if disclose==0)
```

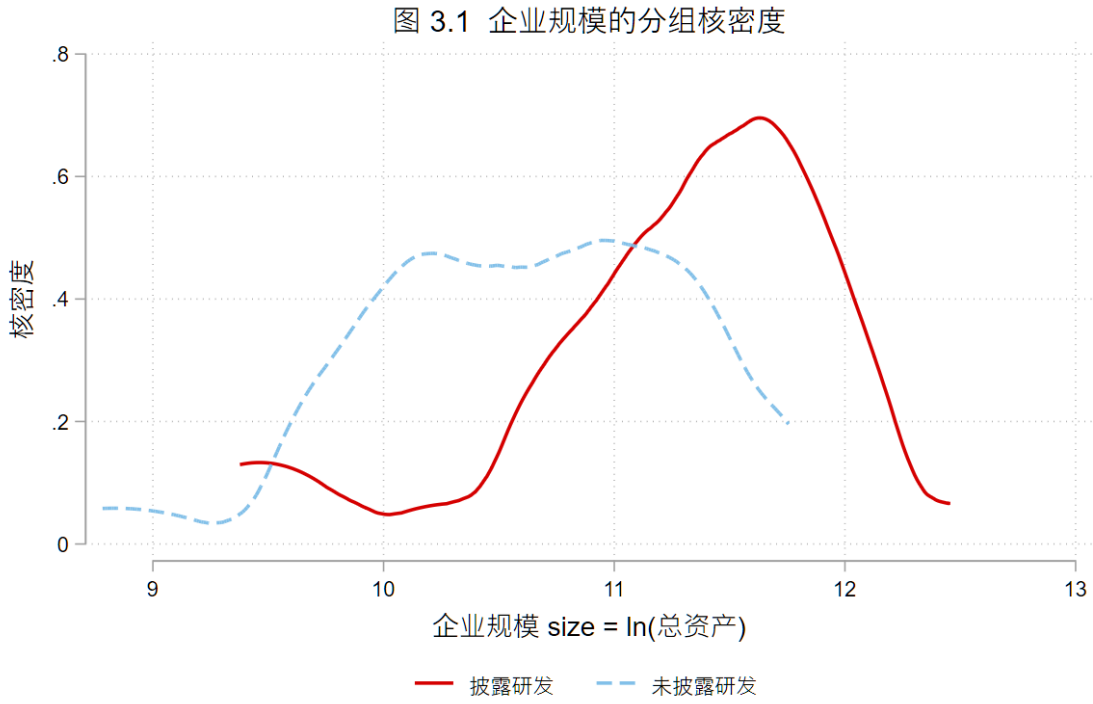

图 3.4: 图 3.1 企业规模在披露组与未披露组的分组核密度

披露组 (实线) 整体明显右移、集中在规模较大处；未披露组 (虚线) 偏向规模较小处；而在低规模区，披露组几乎为空。这正是样本选择的痕迹：规模小的公司很少进入分析样本。均值只告诉你两组中心不同，密度图进一步告诉你，它们连形状和覆盖范围都不一样。

> **提示词：把画图交给 AI**
>
> 我有一份公司-年度面板 (变量：企业规模 `size`、是否披露研发 `disclose`)。请生成 Stata 代码，画一张「按 `disclose` 分组的 `size` 核密度叠加图」，两组用不同线型，并加中文图例与坐标轴标题。若你更熟悉 Python，也可以用 seaborn 或 matplotlib 给出等价实现。

这里真正要练的，不是记住 `kdensity` 的写法，而是养成「按子样本比较分布」的意识；具体代码，说清需求后交给 AI 即可。

3.4.4 样本筛选表：分析样本是怎么来的

最后，把「样本怎么筛出来」记成一张表。

```
count                       // 初始：所有公司-年度
count if !missing(rd)       // 分析样本：披露研发、rd 可得
```

```
步骤                                  剔除观测      剩余观测    公司
-------------------------------------------------------------
初始（所有公司-年度）                    —             126       42
剔除未披露研发（rd 不可得）               63            63       21
-------------------------------------------------------------
```

126 个公司-年度里，有 63 个 (21 家公司) 因未披露研发而落选，最终分析样本只剩 63 个 (21 家)。而落选的这一半，正是前面看到的规模较小的那批。于是分析样本系统性地偏向又大又强的公司，它已经不是「所有公司」的代表了。

3.4.5 鱼塘抽样模拟：抽样误差与选择偏误的区别

最后用一个小模拟，把「抽样误差」和「选择偏误」的区别看得更透。把全体 126 家的 `size` 当作一整口鱼塘 (真实平均 = 10.924)，比较两种捞法，各重复 500 次 (完整代码见 `ch03_main.do`)。

```
* (a) 随机撒网：每次从全体随机抽 30 个，算平均规模
* (b) 只在披露组捞：每次只从披露组抽 30 个，算平均规模
* 各重复 500 次，看 500 个平均值的分布
```

```
随机撒网 (500 次):        均值的均值 = 10.927     标准差 = .118
只在披露组捞 (500 次):    均值的均值 = 11.238     标准差 = .100
```

随机撒网 500 次，平均值紧紧围绕真实值 10.924——这是抽样误差，随机、无偏，样本量越大波动越小。只在披露组捞，平均值稳定在 11.238，系统性地高出一大截——这是选择偏误。特别要注意：它的标准差 (0.100) 甚至更小。换句话说，你会把一个偏高的数估得又稳又「精确」，却始终偏离真相；样本量再大，也救不了。

把四步连起来，指向同一个结论：这份「研发—绩效」研究的分析样本，系统性地偏向又大又强的公司。若不加甄别地在这个样本上估研发的作用，很可能把「本来就强」误当成「研发有效」。这，正是本讲开头那个问题的答案——同一个回归，因为样本是谁不同，含义就不同。

3.5 AI 协作

> **契约边界**
>
> 本节逐条覆盖大纲「AI 协作训练」四项，不删项、不缩范围。每项给一份可复制的提示词、标注对应的 core skill，并说明怎么人工核对。这些任务，网页版 AI 助手大多都能完成。

这四项训练，本质上都是让 AI 陪你把前面那条链再走一遍：「顺着拆」——把一篇论文拆成五层；「反着查」——看五层之间有没有对不上。它们共用同一个 skill：core/02-paper-strategy。

3.5.1 训练 1：让 AI 把一篇论文拆成五层

任务：把论文交给 AI，让它按总体、样本、数据、模型、推断五层拆解，产出一张五层拆解卡。

> **提示词**
>
> 这是一篇实证论文 (附 PDF)。请面向计量基础较弱的读者，把它按「五层框架」拆成一张中文卡片，每层一小段：
>
> - 总体 (Population)：研究对象是谁？结论的外推边界在哪？
> - 样本 (Sample)：样本怎么产生的？有没有可能的样本选择？
> - 数据 (Data)：关键变量怎么测量的？有没有用代理变量？
> - 模型 (Model)：用了什么模型设定和函数形式？
> - 推断 (Inference)：怎么表达不确定性 (标准误、置信区间、显著性)？
>
> 只依据论文本身，不要编造；每条结论尽量标出对应的图表或页码；不确定的地方标出来，让我核对。

对应 **skill**：`core/02-paper-strategy`。人工核对：对照原文摘要与关键图表，抽查每层结论；特别留意 AI 会不会把「样本选择」凭空安到一篇其实没有这个问题的论文上。

3.5.2 训练 2：把 AI 设成 coauthor，逐层追问一致性

任务：让 AI 扮演一位严格的合作者或论文导师，从五层逐一追问这项研究站不站得住。

> **提示词**
>
> 请你扮演一位严格但善意的合作者，针对这篇研究 (附)，从下面五个层次逐一追问，每层给出你的疑问和判断依据：
>
> - 研究对象是否清楚？结论想推广到的总体是什么？
> - 样本是否可能存在选择偏误？谁被排除在样本之外？
> - 关键变量是否测量准确？有没有测量误差或代理变量失真？
> - 模型设定是否匹配研究问题？有没有明显的误设或坏控制？
> - 推断是否支撑论文的结论？显著性有没有被过度解读？
>
> 每条追问都要给出依据，并指出作者可以怎么回应或补强。不确定的地方如实说明。

对应 **skill**：`core/02-paper-strategy`。人工核对：AI 的追问是灵感来源，不是定论；哪些疑问真的成立、哪些是它想多了，要你回原文判断。

3.5.3 训练 3：生成一份「样本与研究设计一致性检查清单」

任务：把分组统计量、密度图、样本筛选表和政策背景材料交给 AI，让它据此产出一份可勾选的一致性检查清单。

提示词

> 我在检查一篇研究 (或我自己的研究) 里，样本和研究设计是否互相匹配。以下是我手头的材料 (可附其中若干)：分组统计量表、分组密度图、样本筛选表、政策背景说明。请据此帮我生成一份可勾选的「样本与研究设计一致性检查清单」，覆盖：总体与样本是否一致、处理组与对照组是否可比、关键变量口径是否清楚、样本筛选是否可能造成选择偏误、模型层级是否对应政策作用层级。每一条写成一个可回答「是 / 否 / 存疑」的检查项，并说明该怎么用我给的材料去判断。

对应 **skill**：`core/02-paper-strategy`。人工核对：清单是脚手架，每一项的「是 / 否」得由你拿着材料亲自判定，AI 不替你下结论。

3.5.4 训练 4：让 AI 找出四者之间的明显不一致

任务：把研究问题、样本口径、变量构造、模型设定四份说明交给 AI，让它交叉比对、标出冲突。

提示词

> 下面是一篇研究的四份说明 (附)：(1) 研究问题；(2) 样本口径与筛选规则；(3) 核心变量的构造方式；(4) 模型设定。请你交叉比对这四者，找出明显不一致或对不上的地方，例如：研究问题针对的总体和实际样本口径不符、处理变量的层级和模型层级不一致、控制变量里混进了坏控制等。逐条列出你发现的冲突、它可能带来的问题，以及建议怎么核实。只依据我给的材料，不要脑补我没写的内容。

对应 **skill**：`core/02-paper-strategy`。人工核对：AI 找出的「不一致」有真有假；把它当成一份待核清单，逐条回到材料和数据去确认。

3.6 小结与练习

本讲把实证研究还原成一条会走样的链条：从你想研究的总体，到实际拿到的样本，到测量出的数据，再到你亲手定义的 Y、D、X，最后落到模型与推断。每一次传递，口径都可能漂移；样本选择偏误、测量误差、变量口径不当、模型误设，都是这条链在某一处断了。随后，我们把「怎么查链条有没有断」落成了五件工具：分组统计量、假设检验、密度函数图、样本筛选表和政策背景说明，合起来就是一次一致性体检。

一句话带走：这条链，既是诊断自己结果为什么不好的地图，也是进一步理解因果推断、识别策略和内生性的地基。

3.6.1 练习

练习 1(概念 • 判断链条断在哪)

一位研究者想研究「中国所有制造业企业的融资约束」，但数据只覆盖 A 股上市的制造业公司。请指出这项研究最可能在链条的哪一环出问题、属于哪类偏误，并说明它会如何影响结论的外推。

参考答案

问题出在「总体 → 样本」这一环。目标总体是所有制造业企业，实际样本只有上市公司——能上市本身就是一道强筛选 (规模较大、经营较规范、能过审)，属于样本选择。于是结论只能稳妥地外推到上市公司这个子总体；若当作「所有制造业企业」的规律，就越过了外推边界。而且上市公司的融资约束通常更低，把它的结论套到全体制造业上，会系统性地低估整体的融资约束程度。

练习 2(概念 • 变量口径)

两位研究者用同一份数据研究「研发对企业绩效的影响」，一位用 ROA 作结果变量，另一位用 Tobin's Q，结论方向相反。请解释这是链条哪一环的问题，以及该如何判断谁更合适。

参考答案

问题出在「数据 → 变量」这一环，也就是变量口径。ROA 衡量的是当期会计盈利能力，Tobin's Q 更接近市场对企业未来价值的预期；研发短期会压低利润、却可能抬高长期价值预期，所以两个 Y 给出相反方向并不奇怪。谁更合适，取决于研究问题问的是哪一种「绩效」——短期盈利，还是长期价值，而不是哪个跑出来更显著。

练习 3(交给 agent 并审计)

任选一篇你自己研究领域的实证论文，用本讲「训练 1」的提示词让 AI 把它拆成五层；然后回到原文，找出 AI 的拆解里至少一处不准确或含糊的地方，并说明你是怎么发现的。

参考答案 (要点)

本题没有标准答案，重点在「审计」这一步。合格的回答应能指出 AI 的具体错处 (例如把稳健性检验当成主模型、误判样本口径、给一篇本无样本选择问题的论文安上选择偏误、图表编号对不上)，并说明核对依据 (原文的数据与样本说明、方法部分、图表编号)。能稳定地发现并纠正 AI 的偏差，正是本讲要练的能力。

3.7 延伸阅读

- 样本选择偏误的深入：连享会《因果推断》讲义中讨论样本选择偏误的一章，以及配套的 Heckman 选择模型扩展阅读。选择模型的估计方法本身属进阶，留待高级班。
- 中国政策场景下的处理组定义与识别风险：连享会整理的中国政策冲击案例库 (政策事实、处理组与对照组的划分、识别风险)，可作为「迁移到中国场景」的延伸材料。
- 总体、样本与测量的入门讲法：Wooldridge《计量经济学导论》第 1 章 (实证分析的步骤、数据结构、因果与反事实) 与第 9 章 (代理变量、测量误差、非随机样本与函数形式误设)，统一见阅读清单。
- 方法进阶 (现代因果推断与识别策略)：留待高级班，可先浏览连享会关于现代因果推断方法的整理。

4 线性回归的建模语言：CEF、OLS 与统计推断

本讲要点

- 条件期望函数 (CEF) 与线性投影；
- OLS 系数的含义和局限；
- 回归系数不是简单的「影响」，而是在特定条件下形成的比较；
- 模型设定、函数形式和变量尺度；
- 对数模型、比例变量、标准化变量和系数解释；
- 交乘项与异质性：调节效应、边际效应与中心化 (承接大纲 A5，见「代码实操 • 交叉项」)；
- 断点回归 (RDD) 初步：用图形识别断点处的跳跃 (承接大纲 A5，见「代码实操 • 断点回归初步」)；
- 统计显著性、经济显著性和结果解释边界；
- 异方差稳健标准误与聚类稳健标准误；
- 如何阅读回归表中的系数、标准误、置信区间、显著性和样本量；
- 如何从回归表写出不过度解释的结果说明。
- 本讲用到的 skills：`core/05-regression-interpreter`；
- 可运行伴生文件：`examples/ch04/ch04_main.do`(Stata；配套 `mroz.dta` 等数据随本讲一起提供)。

4.1 案例情境：一个系数，三种读法

先从一个几乎人人都跑过的回归说起。

手上是一份劳动经济学的经典数据 (`mroz`，421 位在业女性)，我们想看看教育和工资的关系，把时薪 `wage` 对受教育年限 `educ` 回归：

```
webuse mroz, clear
* 只保留在职女性（wage>0），剔除个别极端高薪，并把人数过少的教育年限并组（详见下一节）
drop if wage==0
drop if wage>17
replace educ = 9  if educ<=9
replace educ = 14 if educ==15
reg wage educ
```

```
------------------------------------------------------------------------------
        wage | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
        educ |      0.522      0.048    10.76   0.000        0.426       0.617
       _cons |     -2.744      0.624    -4.40   0.000       -3.971      -1.517
------------------------------------------------------------------------------
  N = 421     F(1, 419) = 115.85     R-squared = 0.217     Root MSE = 2.11
```

> 数据说明：`mroz` 是 Stata 自带的网络数据，用 `webuse mroz` 即可直接载入，无需下载。本讲用到的另外两份数据 (`xtcs`、`B1_production`) 随本仓库提供，用一个全局宏指到线上路径即可读取 (见「代码实操」开头)。

结果很干净：`educ` 的系数是 0.522，高度显著。写成方程就是

$$\widehat{wage} = -2.74 + 0.52 \times educ$$

它大致意味着「受教育年限每多一年，时薪高约 0.52 美元」。

问题来了：这个 0.52，到底该怎么读？至少有三种读法——

- 读法一 (因果)：「让一个人多读一年书，她的时薪会上涨约 0.52 美元。」
- 读法二 (比较)：「在这份样本里，受教育多一年的那一批人，时薪平均高约 0.52 美元。」
- 读法三 (预测)：「知道一个人多读了一年书，我就把对她时薪的猜测调高约 0.52 美元。」

三种读法，数值一样，含义天差地别。读法一是因果断言，需要额外的识别假设才站得住；读法二只是描述性比较，是回归系数本本分分能给的东西；读法三则把回归当成一台预测机器。回归命令算出的那个数，本身并不告诉你该用哪一种读法。把数值当成因果，是实证研究里最常见、也最隐蔽的过度解释。

这就是本讲要做的事：把「跑回归」变成「理解模型」。跑一个回归只要一行命令，但要说清楚那个系数是什么、能读到哪一步、不能读到哪一步，得先弄明白回归到底在估计什么。想清楚这件事，你写结果时才不会把「比较」误写成「影响」。

4.2 概念讲解：回归到底在估计什么

4.2.1 条件期望函数：按 X 分组求平均

抛开公式，先看回归在直觉上做了一件什么事。

还是教育和工资。为了把「分组平均」看清楚，我们先对数据做一点基础清理 (这也是上一段代码里那几行预处理在做的事)：只留在职女性、剔掉个别极端高薪、把人数太少的教育年限并到相邻组，剩下 421 位、8 个教育档位。然后按受教育年限分组，算出每一组的平均时薪：

```
webuse mroz, clear
drop if wage==0
drop if wage>17
replace educ = 9  if educ<=9
replace educ = 14 if educ==15
bysort educ: egen wage_m = mean(wage)   // 每个 educ 组的平均 wage
```

受教育年限 educ	9	10	11	12	13	14	16	17
平均时薪 wage_m	2.5	3.2	2.6	3.4	3.9	4.1	4.7	7.4

这张表在说一件朴素的事：给定受教育年限，这一组人的平均时薪是多少。读过 12 年书的人，平均时薪 3.4；读过 16 年的，4.7。把每一组的平均值画成一个点，就得到下面这张图里的深蓝菱形——这些「在每个 **X** 上、**Y** 的平均值」的点，就是条件期望函数 **(Conditional Expectation Function，CEF)**：

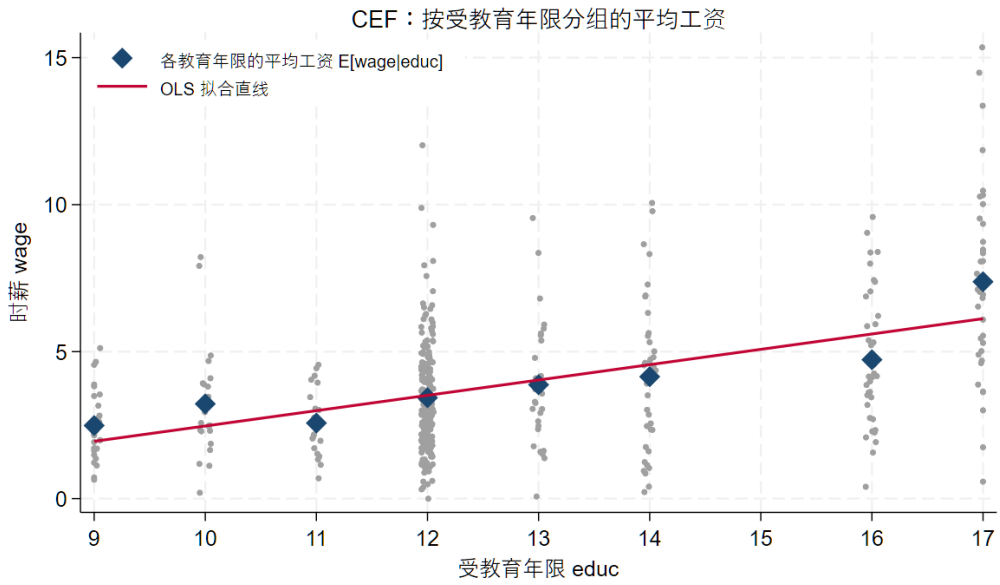

图 4.1: 按受教育年限分组的平均时薪：灰点是个体，深蓝菱形是各组均值 (CEF)，红线是 OLS 拟合直线

CEF 用记号写出来特别简洁：

$$\mathbb{E}[Y \mid X = x] = m(x)$$

意思是「在 $X = x$ 这一组里，Y 的平均值」，它是 x 的一个函数 $m(x)$。放到我们的例子里，

$$\mathbb{E}[wage \mid educ = 12] = m(12) = 3.4$$

用 Stata 求某一组的 CEF，其实就是求一个条件均值——下面两条命令给出的常数项是一回事：

```
sum wage if educ==12   // 直接求 educ=12 这组的平均 wage
reg wage if educ==12   // 只含常数项的回归，截距 = 该组均值
```

CEF 是回归真正的估计目标。我们关心教育和工资的关系，本质上就是想知道：从一个受教育水平挪到另一个，平均工资怎么变。这正是 CEF 描述的东西。

把观测值和 CEF 的关系写清楚，就得到建模的出发点。任何一个 Y_i，都可以拆成「它所在组的平均值」加「一个偏离」：

$$Y = m(X) + e, \qquad \mathbb{E}[e \mid X] = 0, \qquad \mathbb{E}[e^2 \mid X] = \sigma^2(X)$$

- $\mathbb{E}[e \mid X] = 0$ 是 CEF 的定义带来的：既然 $m(X)$ 是每组的平均，组内的偏离自然平均为零；
- $\mathbb{E}[e^2 \mid X] = \sigma^2(X)$ 说的是每组内部的离散程度，它可以随 X 变化——这一点后面讲异方差时会回来用到。

4.2.2 回归估计的就是 CEF

上面这条 CEF(8 个深蓝点) 和我们一开始跑的那条 OLS 直线 (红线) 到底什么关系？一个小实验能把它讲透。

我们分别跑三个回归：(1) 对全部 421 个个体回归；(2) 只对 8 个分组平均值回归；(3) 对 8 个分组平均值回归、但按每组人数加权 (`aweight`)：

```
reg wage   educ                          // (1) 个体回归
reg wage_m educ                          // (2) 只用 8 个组均值
reg wage_m educ [aweight = Nj]           // (3) 组均值，按组容量 Nj 加权
```

```
------------------------------------------------------------
                  (1)             (2)             (3)
          individual       groupmean    groupmean_aw
------------------------------------------------------------
educ            0.522***        0.497**         0.522**
              (0.048)         (0.102)         (0.103)
_cons          -2.744***       -2.361          -2.744
              (0.624)         (1.334)         (1.331)
------------------------------------------------------------
N                 421               8               8
------------------------------------------------------------
```

看第 (1) 列和第 (3) 列：斜率都是 **0.522**，截距都是 **−2.744**，一模一样。这不是巧合——对个体做 **OLS**，等价于对各组的条件均值 **(CEF** 点**)** 做加权回归，权重就是每组的人数。换句话说，OLS 并没有去拟合那一大片散点，它真正在拟合的，是那 8 个 CEF 点。这就是「回归估计的是 CEF」这句话最直接的证据。

(第 (2) 列不加权，把人数悬殊的组一视同仁，斜率略有偏离——这也说明权重不是可有可无的。)

4.2.3 线性投影：给 CEF 选一条直线

CEF 未必是直的。看 图 4.1：从 12 年到 13 年只涨了一点，从 16 年到 17 年却猛跳。如果每一段都单独估计，既费数据又难解释。于是我们退一步，用一条直线去逼近这条 **CEF**：

$$\mathbb{E}[Y \mid X] \approx \alpha + \beta X$$

这条「最能贴合 CEF 的直线」有个名字，叫线性投影。它不是假装 CEF 真的是直线，而是承认「我用一条直线来近似它」。怎么算出这条直线？OLS 的办法是让所有点到直线的竖直距离平方和最小，即最小化残差平方和 (Residual Sum of Squares, RSS)：

$$\min_{\alpha,\,\beta} \ \mathrm{RSS} = \sum_{i=1}^{N} \left(y_i - \alpha - \beta x_i\right)^2$$

求解出来，斜率有一个很直观的形式——它就是 x 与 y 的样本协方差除以 x 的样本方差：

$$\hat{\beta} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \qquad \hat{\alpha} = \bar{y} - \hat{\beta}\,\bar{x}$$

拿到 $\hat{\alpha}, \hat{\beta}$，每个观测的拟合值与残差分别是

$$\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i, \qquad \hat{e}_i = y_i - \hat{y}_i$$

上面那个 0.52，就是用一条直线概括「教育每多一年、平均时薪大约高多少」的结果。公式服务于直觉：只要记住「回归 = 用直线概括 CEF」，就抓住了这一讲的地基。

4.2.4 换一个 f(•)，就是换一个模型

「用直线逼近 CEF」只是最简单的一种选择。CEF 本身是弯的，我们完全可以用别的函数形式 $f(\cdot)$ 去逼近它。事实上，本讲后面要讲的各种模型，都是在给同一条 CEF 选不同的 $f(\cdot)$：

$$
\begin{aligned}
&(1)\quad wage_i = \alpha + \beta\, educ_i + e_i && \text{(线性)}\\
&(2)\quad wage_i = \alpha + \beta_1 educ_i + \beta_2\, educ_i^2 + e_i && \text{(平方项：允许弯曲)}\\
&(3)\quad \ln(wage_i) = \alpha + \beta\, educ_i + e_i && \text{(对数：改变尺度)}\\
&(4)\quad wage_i = \alpha + \beta\, educ_i + \gamma D_i + \theta\,(D_i \times educ_i) + e_i && \text{(虚拟变量 + 交叉项)}
\end{aligned}
$$

其中 (4) 里 $D_i = \mathbf{1}(educ_i > 16)$ 是一个 0/1 变量。这四个模型形式不同，但分析的都是同一样东西——条件期望 **CEF**，只是对 $m(\cdot)$ 的形状做了不同假设。

把这四种设定的拟合线画在同一张散点图上，差别一目了然 (图 4.2)：线性是一条直线，平方项让它可以弯曲，对数 (换算回工资尺度后) 在高教育端更平缓，分段设定则在某个门槛处折一下。同一批数据，选不同的 $f(\cdot)$，得到的拟合值就不一样——这正是「模型设定」四个字最直观的样子。

```
* 四种设定各自的拟合值，叠在一张图上
reg wage educ
predict yhat_lin                          // (1) 线性
reg wage educ c.educ#c.educ
predict yhat_quad                         // (2) 平方项
reg lwage educ
predict lyhat
gen yhat_log = exp(lyhat)                 // (3) 对数：换算回 wage 尺度
gen Dhi = (educ>13)
reg wage c.educ##i.Dhi
predict yhat_dum                          // (4) 分段（虚拟变量）
twoway (scatter wage educ, mcolor(gs11) msize(vsmall)) ///
       (line yhat_lin educ) (line yhat_quad educ) ///
       (line yhat_log educ) (line yhat_dum educ), sort
```

想清楚这一点，本讲后面「函数形式」「对数模型」「交叉项」「平方项」几节就不再是零散的技巧，而是同一个问题的不同侧面：你打算用什么形状的 $f(\cdot)$，去描述 **X** 和 **Y** 的条件均值关系。

> 扩展：条件期望之外，还有条件分位数、条件概率
>
> 把「条件期望 $\mathbb{E}[Y \mid X]$」换成「条件中位数」或其它分位数，就得到分位数回归；换成「条件概率 $P(Y=1 \mid X)$」，就得到 Logit / Probit 这类模型。同一套「给定 **X**、看 **Y** 的某个特征」的思路，衍生出一大家子模型。本讲只走条件期望这一支，其余留待各自的专题。

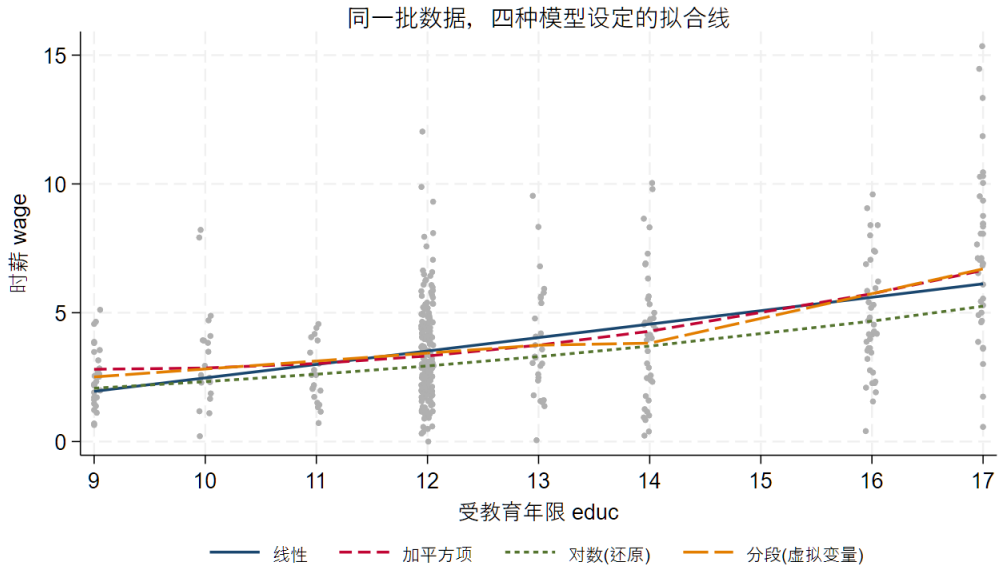

图 4.2: 同一批数据，四种模型设定得到的拟合线各不相同

4.2.5 OLS 是估计方法，不是模型

这里要澄清一个常见的糊涂账：**OLS** 不是一个模型，而是一种估计方法。

顺序是这样的。我们先有一个假设——X 和 Y 的条件期望呈线性关系：

$$\mathbb{E}[Y \mid X] = \alpha + \beta X \tag{1}$$

注意，(1) 式是假设，没法直接拿数据去估。结合前面 $Y = m(X) + e$，把 (1) 代进去，就得到常见的回归模型的设定：

$$Y_i = \alpha + \beta X_i + e_i \tag{2}$$

到这里还只是模型。怎么把其中的 α、β 估出来，有很多办法：

- **OLS**：假设 x 与 e 不相关，最小化残差平方和 $\sum e_i^2$；
- **MLE**：假设 e 服从正态分布，即 $e_i \sim N(0, \sigma^2)$，最大化似然函数 $\prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e_i^2}{2\sigma^2}\right)$；
- **GMM**：构造矩条件 $\mathbb{E}[x\, e] = 0$，进而得到样本矩条件 $\frac{1}{n}\sum x_i e_i = 0$，最小化加权矩条件的平方和或加权平方和。

它们是三种不同的估计方法，估的却是同一个模型 (2)。所以「OLS」和「线性回归模型」是两件事：一个是路径，一个是目的地。想清楚这一点，后面遇到「同一个模型换一种估计方法」(比如面板里的固定效应、工具变量) 时，就不会打结。

4.2.6 系数的含义与局限

回到那个 β。它到底是什么？

β 是线性投影的斜率，含义是：X 每增加一单位，Y 的平均值大约变化 β。注意落点在「平均值」——它说的不是某一个具体个体，而是「X 相差一单位的两群人之间，Y 的平均差异」。

但这个解释有前提。OLS 要把 β 估得没有系统偏差，靠的是一个关键假设：

$$\mathbb{E}[e \mid X] = 0$$

即那些没进模型的因素 e，与 X 不相关。一旦 X 和 e 相关 (比如「能力」既影响教育、又影响工资，却没进模型)，估出的 $\widehat{\beta}$ 就不再是我们想要的那条 CEF 斜率了。这正是「内生性」的起点，也是后面几讲要反复处理的核心问题；本讲先把系数在「干净情形」下的含义讲透。

4.2.7 系数是「比较」，不是「影响」

有了上面的铺垫，就能回答案例情境里那个问题了：0.52 该怎么读？

严格说，它是一个比较：在这份样本里，受教育多一年的那群人，平均时薪高约 0.52。它比较的是两群不同的人 (多读一年的 vs 少读一年的)，而不是「把同一个人的教育年限拨高一年」会发生什么。

这两者的区别是实证解释的命门。「多读一年书的人工资更高」是数据里的事实；「多读一年书会让工资更高」是一个因果断言。后者要成立，得保证「受教育多和少的两群人，除了教育之外其它方面都可比」——而现实中他们往往在能力、家庭、机会上系统性地不同。回归系数天然给你的是比较，要把比较升级成因果，需要额外的识别假设，那是回归命令本身给不了的。

一句话记住：系数刻画的是「特定条件下形成的比较」，不是「影响」。写结果时，把「导致 / 提升 / 影响」这类因果动词，换成「相关 / 更高 / 在控制……之后」这类比较措辞，往往才是诚实的表述。这条护栏，本讲后面还会专门演示怎么落到笔头，以及怎么让 AI 帮你把关。

4.2.8 多元回归：系数是「控制其他变量之后」的比较

前面都是一个自变量。真实研究里，我们会往回归里加控制变量。加了之后，系数的含义会再精细一层——有时还会面目全非。

换一份 Stata 自带数据 `nlsw88`(1988 年美国女性)，看一个更有戏剧性的例子。我们想知道「在本单位任职越久 (`tenure`)，工资是不是越高」，先只放 `tenure`，再逐步加入总工龄 `ttl_exp` 和受教育程度 `grade`：

```
sysuse nlsw88, clear
eststo clear
eststo n1: reg wage tenure
eststo n2: reg wage tenure ttl_exp
eststo n3: reg wage tenure ttl_exp grade
esttab n1 n2 n3, se star(* 0.10 ** 0.05 *** 0.01) r2 ///
       mtitle(" 只放 tenure" "+ 总工龄" "+ 受教育")
```

```
------------------------------------------------------------
              只放 tenure       + 总工龄        + 受教育
------------------------------------------------------------
tenure           0.186***        0.041           0.038
                (0.022)        (0.026)         (0.025)
ttl_exp                          0.301***        0.233***
                                (0.031)         (0.030)
grade                                            0.649***
                                                (0.046)
_cons            6.681***        3.772***       -3.864***
------------------------------------------------------------
N                 2231            2231            2229
------------------------------------------------------------
```

只看第一列，任职年限每多一年、工资高约 0.19，高度显著。但一旦控制住总工龄，tenure 的系数骤降到 0.04、而且不再显著！原因不难理解：在本单位待得久的人，往往总的工作经验也长 (两者相关系数约 0.58)，第一列里 tenure 的系数其实「捞」走了大半本属于经验的功劳。控制总工龄后，tenure 的系数才回到它真正的含义——「在总工龄相同的人之间，多在本单位待一年对应的工资差异」，这个「纯任职」的效应其实很小。

这就是多元回归里每个系数的本质：「其它变量给定不变时」的比较，即所谓偏效应：

$$\beta_j = \frac{\partial \mathbb{E}[Y \mid X]}{\partial X_j}$$

系数从 0.19 掉到 0.04 不是算错，而是问题变了——第一列问「任职久的人工资高不高」，后面几列问「经验一样、只是任职更久，工资差多少」。「设定一变，系数就变」是回归的常态：你每改一次模型、多控制一个变量，问的都是一个略有不同的问题，系数自然跟着变。这条线索，在本讲后面讲交叉项时会看得最极端 (系数甚至会变号)。而「加入控制变量后系数为什么变、变了多少」这个更一般的问题，背后有一套简洁的几何解释 (FWL 定理)，留到下一讲专门展开。

4.3 代码实操

概念讲清楚了，这一节动手。代码以 Stata 为主，静态展示、输出已冻结；你把 `examples/ch04/ch04_main.do` 下载下来就能一段段复跑。

本讲用到几份数据，读取方式分两类：mroz、nlsw88、auto 是 Stata 自带的，用 webuse / sysuse 直接载入；xtcs、B1_production 随本仓库提供，先用一个全局宏指到线上路径，再 use 即可，无需下载到本地：

```
global D "https://raw.githubusercontent.com/lianxhcn/PXa2026a/main/examples/ch04/data"
* 之后：  use "$D/xtcs.dta", clear    // 联网直接读；也可把 $D 改成本地路径
```

4.3.1 跑第一个回归：从命令到拟合值

回归只要一行 reg y x。跑完之后，Stata 在内存里留下了拟合值和残差，用 predict 取出来(沿用前面清理后的 mroz 样本)：

```
webuse mroz, clear
drop if wage==0 | wage>17
replace educ = 9 if educ<=9
replace educ = 14 if educ==15
reg wage educ
predict wage_hat            // 拟合值 y-hat = -2.74 + 0.52*educ
predict e_hat, residual     // 残差 e-hat = y - y-hat
list wage wage_hat e_hat in 1/3, clean
```

拟合值 $\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i$ 是模型对第 i 个人的「预测」，残差 $\hat{e}_i = y_i - \hat{y}_i$ 是预测与实际的差。OLS 的目标 (最小化 $\sum \hat{e}_i^2$) 保证了这些残差平方和最小，也保证了 $\sum \hat{e}_i = 0$、且残差与自变量不相关。图 4.1 里那条红线就是拟合线，每个灰点到红线的竖直距离就是它的残差。

4.3.2 读懂一张回归表

做研究时，我们很少只跑一个模型。把几个模型并排放进一张表，是论文的标准做法。用 esttab(estout 包) 可以一键生成：

```
eststo clear
eststo m1: reg wage educ
eststo m2: reg wage educ exper
esttab m1 m2, se star(* 0.10 ** 0.05 *** 0.01) r2 mtitle(" 模型 1" " 模型 2")
```

```
--------------------------------------------
                    模型 1           模型 2
--------------------------------------------
educ              0.522***         0.524***
                 (0.048)          (0.048)
exper                              0.051***
                                  (0.013)
_cons             -2.744***        -3.448***
```

```
                (0.624)         (0.637)
-------------------------------------------
N                   421             421
R-sq              0.217           0.247
-------------------------------------------
Standard errors in parentheses
* p<0.10 ** p<0.05 *** p<0.01
```

一张回归表要会读五样东西：

- 系数 (0.524)：偏效应，前面讲过——「其它变量不变时，X 多一单位、Y 平均差多少」；
- 标准误 (括号里的 0.048)：这个系数估计的不确定性有多大，越小越精确 (下一小节细讲)；
- 星号 (***)：显著性，星越多，系数越不可能只是偶然 (*** 表示 $p < 0.01$)；
- 样本量 **N**(421)：这张表是用多少个观测算出来的——换了模型若 N 变了，往往是某个变量有缺失，结论的可比性就打了折扣；
- R^2(0.247)：模型解释了 Y 变动的百分之多少。R^2 不是越高越好，也不能用来判断因果，本讲不展开。

系数和标准误还能拼出置信区间：$\hat{\beta} \pm 1.96 \times se(\hat{\beta})$，给出「这个系数大致落在哪个区间」。这五样东西连起来，才是一张回归表完整的信息。

4.3.3 换个单位，系数会变吗：变量的尺度

初学者常有的困惑：把工资从「美元」换成「美分」，或把教育从「年」换成「月」，系数会不会变？会，但变的只是数字，不是结论。

```
gen wage_cents  = wage*100      // 工资改用美分
gen educ_months = educ*12       // 教育改用月
reg wage       educ             // 原始
reg wage_cents educ             // 因变量 ×100
reg wage       educ_months      // 自变量 ×12
```

模型	educ(或 educ_months) 系数	t 值	R^2
wage on educ	0.522	10.76	0.217
wage_cents on educ	52.15	10.76	0.217
wage on educ_months	0.043	10.76	0.217

规律很干净：因变量放大 **100** 倍，系数就放大 **100** 倍 (0.522→52.15)；自变量放大 **12** 倍，系数就缩小到 **1/12**(0.522→0.043)。但 **t** 值、显著性、R^2 完全不变——因为尺度变换不改变「教育和工资的关系」本身，只改变了度量单位。所以看到一个系数，第一件事是问清楚：**Y** 和 **X** 各是什么单位？ 0.043(每月) 和 0.522(每年) 说的是同一件事。

4.3.4 对数与标准化：系数怎么读

在第 2 讲我们已经学过怎么对变量取对数、做标准化。这里补上最要紧的一环：变换之后，系数该怎么读。不同的变换，读法完全不同。

其一，对数-水平 **(log-level)**：半弹性。把因变量取对数、自变量不变：

```
reg lwage educ     // lwage = ln(wage)
```

```
       lwage | Coefficient  Std. err.      t    P>|t|
        educ |      0.116      0.015     7.92   0.000
```

系数 0.116 要读成百分比：受教育每多一年，工资大约高 $100 \times 0.116 = 11.6\%$。这类模型里，$\beta$ 是「X 每增一单位，Y 变化百分之几」，叫半弹性。

其二，对数-对数 **(log-log)**：弹性。 X 和 Y 都取对数，系数就是弹性——X 变化 1%，Y 变化百分之几。用生产函数数据 (`B1_production`) 估柯布-道格拉斯生产函数最典型：

```
use "$D/B1_production.dta", clear    // $D 见本节开头
reg lnY lnL lnK      // ln 产出 对 ln 劳动、ln 资本
```

```
         lnY | Coefficient  Std. err.      t    P>|t|
         lnL |      0.603      0.126     4.79   0.000
         lnK |      0.376      0.085     4.40   0.000
```

`lnL` 的系数 0.603 读成：劳动投入每增加 **1%**，产出大约增加 **0.60%**；资本的产出弹性是 0.38。两者相加约等于 0.98，接近 1，说明这个行业大致是「规模报酬不变」——这类判断，正是弹性系数的用武之地。

其三，标准化系数：以标准差为单位比较。不同自变量单位不同 (教育按年、经验也按年、但收入按元)，系数大小没法直接比谁更「重要」。给 `reg` 加 `beta` 选项，得到标准化系数：

```
webuse mroz, clear
drop if wage==0 | wage>17
reg lwage educ exper, beta
```

```
       lwage | Coefficient  Std. err.      t    P>|t|      Beta
        educ |      0.108      0.013     8.14   0.000     0.361
       exper |      0.019      0.004     4.93   0.000     0.219
```

标准化系数 (Beta 列) 读成：**X** 每增加一个标准差，**Y** 变化几个标准差。这里 `educ` 的 0.361 大于 `exper` 的 0.219，说明在这份数据里，教育对工资的相对解释力更强。要比较不同变量的相对重要性，才用标准化系数；解释单个变量的实际效应，还是用原始系数更直观。

4.3.5 因子变量：把类别放进回归

性别、行业、地区这类类别变量，不能直接当数字放进回归——它们的取值 1、2、3 没有大小含义。正确做法是转成虚拟变量 (0/1)。Stata 用因子变量前缀 `i.` 一步到位：

```
sysuse auto, clear
reg price i.foreign weight    // i.foreign：以国产车为基准，进口车设一个虚拟变量
```

```
       price | Coefficient  Std. err.      t    P>|t|
     foreign |
      进口车 |   3637.001    668.583     5.44   0.000
      weight |      3.321      0.396     8.39   0.000
       _cons |  -4942.844   1345.591    -3.67   0.000
```

`i.foreign` 自动以「国产车」为基准组，报告的是「进口车 vs 国产车」的差异：车重相同时，进口车平均贵 **3637** 美元。虚拟变量在几何上就是平移截距——两组共用一个斜率 (`weight` 的 3.32)，但各有各的高度。

要是两组连斜率也不同呢？用 `##` 让虚拟变量和连续变量交互：

```
reg price i.foreign##c.weight   // 允许两组截距、斜率都不同
```

这时进口车与国产车各有一条自己的拟合线 (图 4.3)：进口车不但整体更贵，而且价格随车重上涨得更快 (斜率更陡)。这已经用到了下一小节的主角——交叉项。画出来看得最清楚：

```
twoway (scatter price weight if foreign==0, mcolor(navy) msymbol(O)) ///
       (scatter price weight if foreign==1, mcolor(cranberry) msymbol(T)) ///
       (lfit price weight if foreign==0, lcolor(navy)) ///
       (lfit price weight if foreign==1, lcolor(cranberry)), ///
       legend(order(1 " 国产车" 2 " 进口车") rows(1) pos(6)) ///
       xtitle(" 车重 weight") ytitle(" 价格 price")
```

4.3.6 交叉项：让边际效应随另一个变量变

到目前为止，X 对 Y 的边际效应都是一个固定的数 (比如教育回报 0.52，人人一样)。但现实中，一个变量的效应常常取决于另一个变量。交叉项 (交乘项) 就是用来刻画这件事的。

从「变系数」的角度看最清楚。原来我们假设斜率固定：

$$y_i = \alpha + \beta\, x_i + \varepsilon_i$$

现在放松它，让斜率本身随另一个变量 z(调节变量) 变化：

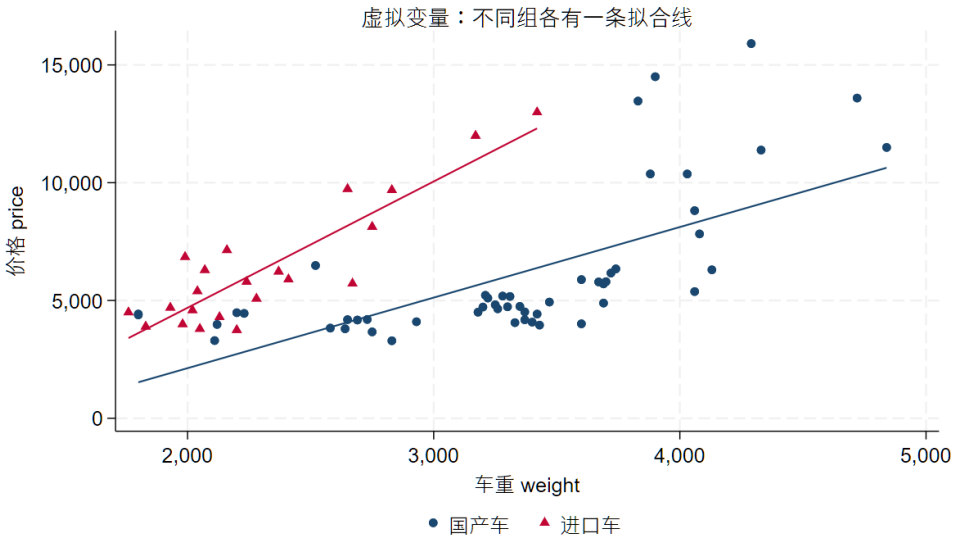

图 4.3: 虚拟变量与交叉项：进口车 (红) 与国产车 (蓝) 各有一条拟合线，截距和斜率都不同

$$y_i = \alpha + \beta_i\, x_i + \varepsilon_i, \qquad \beta_i = f(z_i)$$

于是 x 对 y 的边际效应不再是常数，而是 z 的函数：

$$\frac{\partial y_i}{\partial x_i} = \beta_i = f(z_i)$$

最简单地，设 f 是线性的，$\beta_i = \gamma_0 + \gamma_1 z_i$，代回原式：

$$y_i = \alpha + (\gamma_0 + \gamma_1 z_i)\, x_i + \varepsilon_i = \alpha + \gamma_0\, x_i + \gamma_1\, (x_i \cdot z_i) + \varepsilon_i$$

$x_i \cdot z_i$ 这一项就是交叉项。它的系数 γ_1 刻画「z 每高一单位，x 的边际效应变化多少」。

举个财务学的例子 (数据 `xtcs`，中国上市公司)：盈利能力 `npr`(净利润率) 对负债率 `tl` 的影响，会不会随公司规模 `size` 而不同？先不加交叉项，再加上：

```
use "$D/xtcs.dta", clear      // $D 见本节开头
xtset code year
global zz "tang fr ndts L.tobin i.year"
reg tl npr size $zz, robust                  // 不含交叉项
reg tl npr size c.npr#c.size $zz, robust     // 含交叉项
```

	不含交叉项	含交叉项
npr	-0.371	1.729**
size	0.033	0.040***
c.npr#c.size		-0.100**

于是 npr 对 tl 的边际效应是

$$\frac{\partial\, tl}{\partial\, npr} = 1.729 - 0.100 \times size$$

用 margins 和 marginsplot 把它画出来 (图 4.4)：公司规模越大，盈利能力对负债率的 (负向) 作用越强。

```
reg tl npr size c.npr#c.size $zz, robust
margins, dydx(npr) at(size=(19(1)24))     // 不同 size 处 npr 的边际效应
marginsplot                                // 画出来，带 95% 置信区间
```

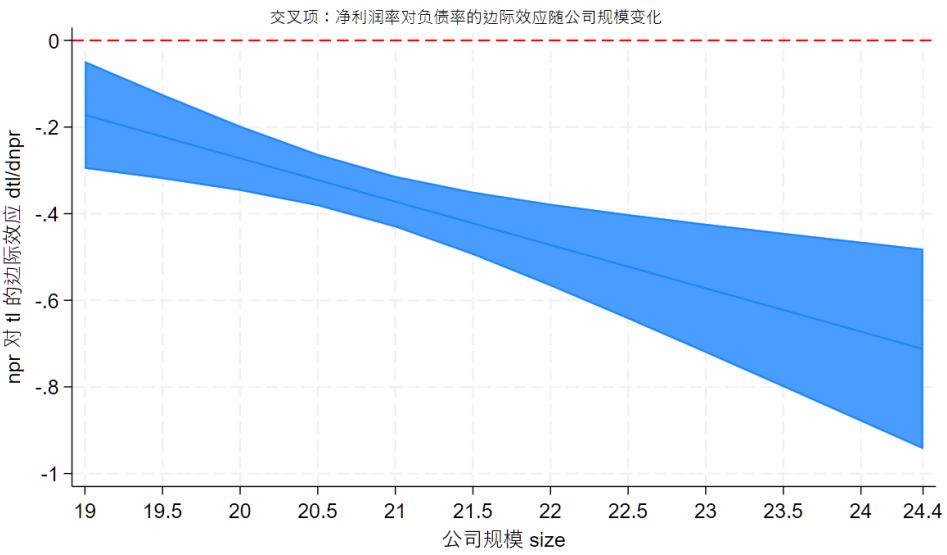

图 4.4: 交叉项的边际效应：npr 对 tl 的边际效应随公司规模 size 变化 (阴影为 95% 置信区间)

这里有一个必须讲清的坑。注意上表里，npr 的系数从不含交叉项时的 -0.371，变成含交叉项时的 $+1.729$——连符号都反了！这不是算错，而是含义变了。在交叉项模型里，一阶项 npr 的系数是

$$\gamma_0 = \left.\frac{\partial\, tl}{\partial\, npr}\right|_{size=0}$$

也就是「当 $size = 0$ 时」的边际效应。可公司规模 `size` 根本不可能取 0(样本里都在 21 附近)，所以这个 $+1.729$ 是一个外推到数据之外的、没有现实意义的数。

由此引出两条要点：

1. 模型里有交叉项 $x \cdot z$，就必须同时放入 x 和 z 两个一阶项，否则边际效应就算错了；
2. 加了交叉项后，一阶项系数的含义变了，不能再和「不含交叉项」时的系数直接比较。

中心化是缓解第 2 点的常用手段：把 z 减去它的均值再构造交叉项，一阶项系数就变成「在 z 取均值处」的边际效应，好解释、也和不含交叉项时可比。但要强调：中心化不改变交叉项系数本身，也不影响任何统计推断，它纯粹是为了让系数好读——所以是可选项，不是必须做的一步。

> **交互效应可能不是线性的 (进阶)**
>
> 上面假设 $\beta_i = \gamma_0 + \gamma_1 z_i$ 是 z 的线性函数。真实的交互效应未必这么规矩，可能是非线性的、或只在某段 z 上存在。诊断这类问题有专门的工具 (如 `interflex`)，属于进阶内容，本讲不展开。

4.3.7 平方项：允许先升后降

平方项是一种特殊的交叉项——变量和它自己交乘 ($x \cdot x = x^2$)。它让「直线」变成「抛物线」，从而能刻画先升后降 (或先降后升) 的关系。

工资和工作经验就是典型：入职头些年经验值钱、工资涨得快；到了一定年头，经验的边际回报递减，工资涨势放缓甚至回落。用 `nlsw88` 数据，把经验的平方项放进去：

```
sysuse nlsw88, clear
gen ttl_exp2 = ttl_exp^2
reg wage ttl_exp ttl_exp2 hours age tenure married south i.race
```

```
        wage | Coefficient  Std. err.      t    P>|t|
     ttl_exp |       0.492      0.110     4.49   0.000
    ttl_exp2 |      -0.009      0.005    -1.99   0.046
```

一阶项为正 ($+0.49$)、平方项为负 (-0.009)，合起来就是一条开口向下的抛物线 (图 4.5)。对于 $y = ax^2 + bx + c$ 这样的二次式，两个量最有用：

$$\text{边际效应：}\ \frac{\partial y}{\partial x} = b + 2ax, \qquad \text{拐点：}\ x^* = -\frac{b}{2a}$$

代入本例，拐点在

$$x^* = -\frac{0.492}{2 \times (-0.009)} \approx 27.2 \text{ 年}$$

也就是说，经验到约 27 年时工资达到顶点，之后开始回落。

```
di -_b[ttl_exp]/(2*_b[ttl_exp2])    // 算拐点：27.2
```

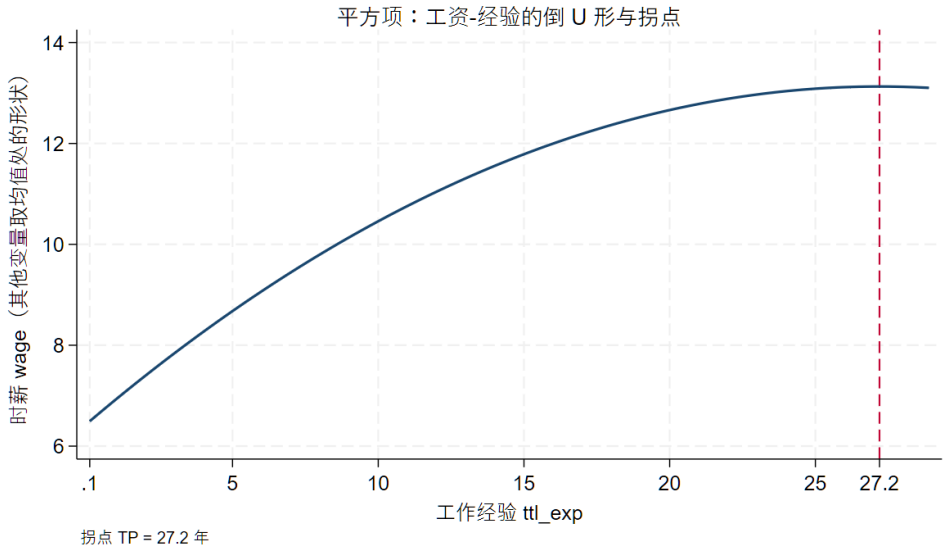

图 4.5: 平方项：工资随经验先升后降的倒 U 形，拐点在约 27.2 年

但拐点要小心解读。本例样本里工作经验最多也就 28.9 年，绝大多数人 (均值 12.5 年) 都落在拐点左侧的上升段。换句话说，「27 年后工资回落」这个结论，是靠极少数长经验样本外推出来的，实际意义有限。看到一个漂亮的倒 **U** 形，先问一句：拐点落在数据范围之内吗？有多少样本真的越过了它？否则很容易把一个数据稀薄处的外推，讲成一个像模像样的「倒 U 型规律」。

4.3.8 断点回归初步：用 binscatter 看断点处的跳跃

平方项让我们能刻画「弯曲」的关系。把这个思路再推一步，就能看到一种很有用的因果推断设计——断点回归 **(Regression Discontinuity Design, RDD)**。这里只做一个直观的演示，帮你认识 `binscatter` 这个画图利器；系统的 RDD 方法留待后续的因果推断专题。

RDD 的基本想法是：如果某个规则在一个临界点 **(cutoff)** 上突然改变待遇，那么就在这个临界点附近，比较刚好在左边和刚好在右边的对象——他们其它方面都很接近，唯一的差别就是有没有越过临界点。于是临界点处 Y 的跳跃，就可以归给这条规则。

拿一个熟悉的例子：每周 40 小时是「加班」的分界。工作时长刚过 40 小时的人，工资会不会在 40 这个点上出现一个跳跃？用 `binscatter` 一看便知——它把横轴分成若干个小仓 (bin)，画出每个仓里 Y 的平均值，`rd(40)` 让它在 40 处断开、两侧分别拟合：

```
sysuse nlsw88.dta, clear
gen lnwage = ln(wage)
binscatter lnwage hours, rd(40) line(qfit)    // 在 hours=40 处断开，两侧各拟合一条二次曲线
```

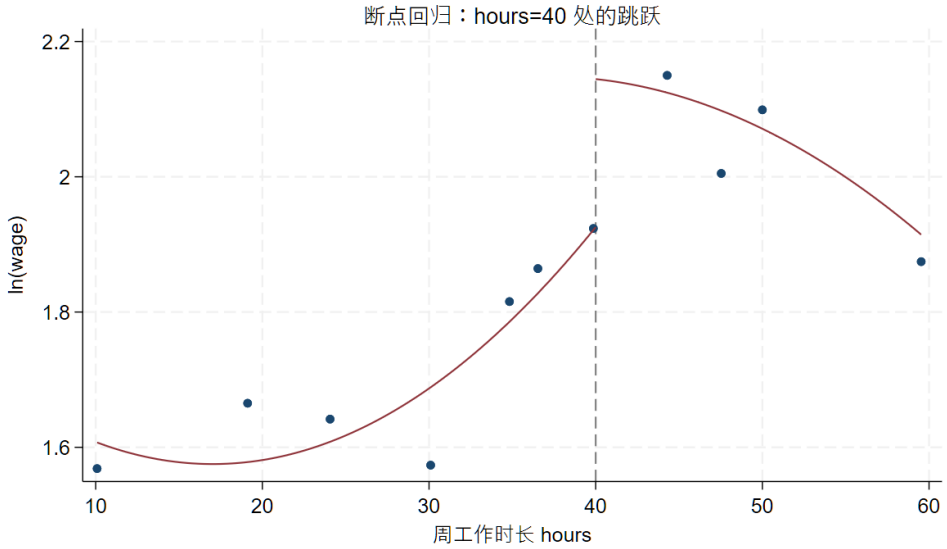

图 4.6: 断点回归：hours=40 处对数工资的跳跃 (binscatter 分仓平均 + 两侧二次拟合)

图 4.6 里，40 小时这条虚线的左右两侧，拟合线出现了一个明显的向上跳跃。把这个跳跃估出来，只需要构造一个「是否越过临界点」的虚拟变量 $D_{40} = \mathbf{1}(hours > 40)$，再加上中心化后的距离 $hours\_c = (hours - 40)/10$ 及其平方项：

```
gen D40        = (hours>40)
gen hours_c    = (hours-40)/10
gen hours_c_sq = hours_c^2
eststo m1: reg lnwage D40 hours_c              // 线性设定
eststo m2: reg lnwage D40 hours_c hours_c_sq   // 允许两侧弯曲
esttab m1 m2, nogap b(%6.4f) t(%5.2f) mtitle(linear quad)
```

```
------------------------------------------
                  (1)             (2)
                linear            quad
------------------------------------------
D40            0.0611          0.1631***
              (1.61)          (3.61)
hours_c        0.0995***       0.0407*
              (7.28)          (2.06)
hours_c_sq                    -0.0287***
                             (-4.12)
_cons          1.8869***       1.8867***
------------------------------------------
N                2242            2242
------------------------------------------
```

关键系数是 D_{40}：它度量的正是在 **40** 小时这个临界点上，对数工资向上跳了多少。

- 线性设定 (1) 里 $D_{40} = 0.061$、不显著；
- 一旦允许两侧弯曲、加入平方项 (2)，跳跃变成 0.163、高度显著——大致是说「刚过 40 小时的人，时薪比刚不到 40 小时的高约 16%」。

两列的对比恰好呼应上一节：函数形式设错 (该弯的地方硬拉成直线)，会把真实的跳跃也一并压掉。这也说明 RDD 的估计对「临界点附近怎么拟合」很敏感——该用多宽的窗口、两侧用几次多项式，都是有讲究的，这些正是因果推断专题里 RDD 要细谈的内容。本讲到此为止，你只需记住：**`binscatter` + `rd()`** 是把「断点处有没有跳跃」一眼看出来的利器。想深入的同学，可参阅连享会关于断点回归与现代因果推断方法 的整理。

4.3.9 稳健标准误与聚类标准误

前面都在盯着系数。但一张回归表能不能支撑结论，一半靠系数，一半靠标准误——它决定了 t 值、显著性和置信区间。这一节讲一件要紧事：同一个回归，点估计不变，但标准误可以有好几种算法，选错了，显著性就是假的。

标准误的算法，取决于你对残差 e 做什么假设。最朴素的 OLS 标准误依赖两条很强的假设：残差同方差 (每个观测的波动一样大) 且互不相关。现实里这两条常常不成立，于是有了两种放松：

其一，异方差稳健标准误 (**`vce(robust)`**)。放松「同方差」，允许每个观测的残差波动不同。截面数据里异方差几乎是常态，所以这是最常用的一档，加一个选项即可：

```
sysuse nlsw88, clear
global x "age ttl_exp union collgrad"
reg wage $x             // 默认（同方差）标准误
reg wage $x, robust     // 异方差稳健标准误
```

对比两次结果你会看到：所有系数一模一样，变的只有标准误。因为 `robust` 调整的是标准误、与系数无关，而 t 值是二者的比值。用矩阵的语言说，异方差允许每个观测的残差方差各不相同 (对角线上的值不再相等)，但对角线以外仍为零 (观测之间不相关)。

其二，聚类稳健标准误 (**`vce(cluster id)`**)。更进一步，放松「互不相关」——允许同一组内部的残差相关。这在实证里极其常见：同一家公司不同年度、同一个行业里的公司、同一个班级的学生，他们的不可观测因素往往彼此关联。

`vce(cluster industry)` 这一条命令，背后有三层含义，外加一个隐含假设，务必分清：

1. 不同 **`industry`** 之间的残差不相关；
2. 同一个 **`industry`** 内部的残差可以相关 (这正是放松的那一条)；
3. 不同 `industry` 之间允许异方差 (波动可以不同)；
4. **(隐含假设)** 聚类层级选对了——即你选的分组，确实抓住了残差相关的真实结构。

几何上，方差矩阵从「只有对角线」变成了分块对角：每个组是一个小方块，块内允许非零相关，块与块之间为零 (图 4.7)。

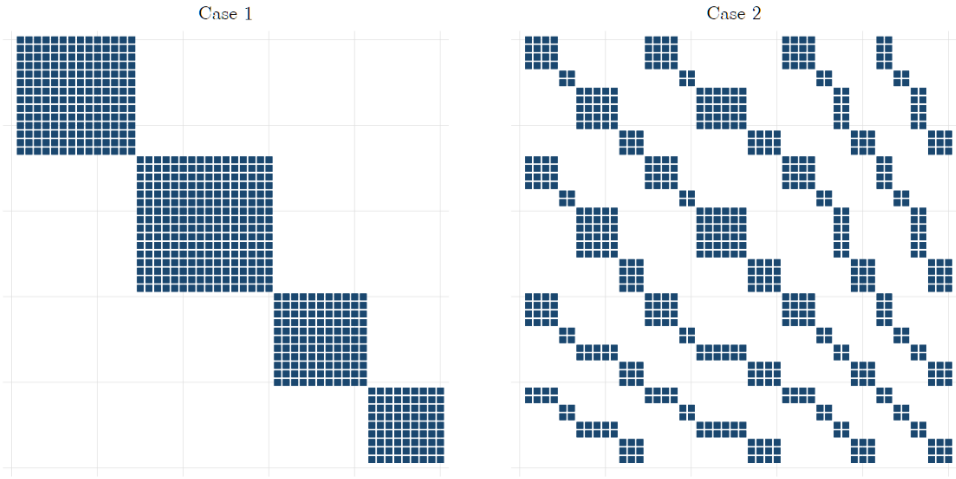

图 4.7: 一维聚类的方差矩阵：分块对角，块内 (同组) 允许相关，块间为零

```
reg wage $x, vce(cluster industry)    // 按行业聚类
```

把三种标准误并排看 (点估计已省略，因为三列完全一样)：

```
                      OLS          Robust          Cluster(industry)
   ttl_exp         (0.0185)        (0.0172)        (0.0236)
   union           (0.1954)        (0.2065)        (0.4133)
   ———括号内为标准误；三列的系数完全相同 ———
```

看 union 那一行：聚类标准误 (0.41) 是普通标准误 (0.20) 的两倍。忽略组内相关，会把标准误算得过小、t 值虚高，让本不显著的结果看起来显著。这就是为什么聚类是当代实证的标配。

二维聚类。有时残差的相关来自不止一个维度——一家公司既和同行业的公司相关，也和同城市的公司相关。这时可以在两个维度上同时聚类 (用 reghdfe 等命令)：

```
reghdfe wage $x, noabsorb cluster(industry occupation)    // 按行业、职业二维聚类
```

图 4.8 里，两个维度的分块会交叠 (既有同行业的相关，也有同职业的相关)。但二维聚类不是「越多越稳健越好」：

- 它通常让标准误更大、结果更不容易显著——这是稳健性的代价，会增大犯「假阴性」的概率；
- 用不用二维，取决于你对残差相关结构的判断，而不是「能聚就聚」；
- 聚类数太少会失效。聚类稳健标准误的有效性依赖「组的数目足够多」(经验上通常要 30～50 以上)。本例里 industry 只有 12 个、occupation 13 个，其实已经偏少——这种时候聚类标准误本身就不太可信，需要另想办法。

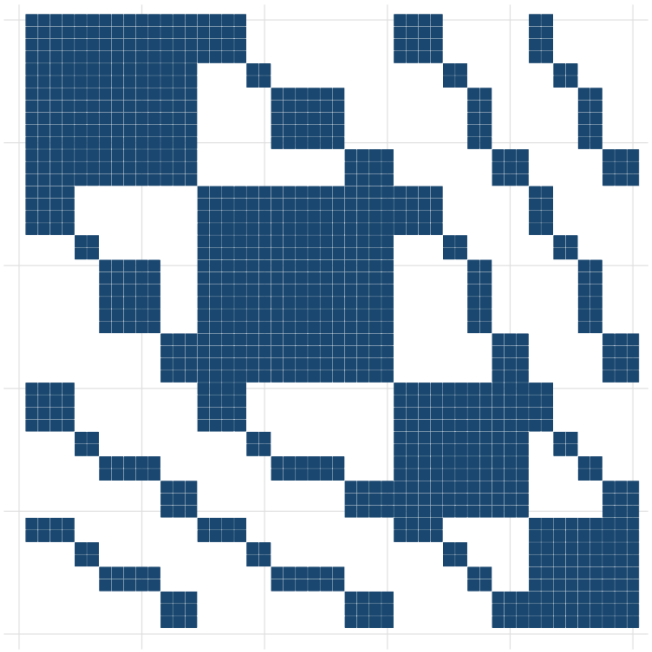

图 4.8: 二维聚类的方差矩阵：两个维度的分块交叠

> **一个自查小技巧**
>
> 换了聚类对象，系数不该变。如果你把 `cluster(industry)` 换成 `cluster(occupation)`，发现系数也跟着变了，那多半不是标准误的事——很可能是两个聚类变量的缺失值不同，导致样本量变了。标准误可以变，系数变了就要查数据。

到这里，一张回归表的五个要素 (系数、标准误、置信区间、显著性、样本量) 就都能读、能算、能判断可信度了。至于什么时候该用哪一档标准误、聚类到哪一层，是需要结合数据结构反复权衡的判断题，更系统的讨论见 Cameron & Miller (2015) 与 Abadie et al. (2023)(见延伸阅读)。

4.4 结果解释与因果护栏

会跑回归、会读表之后，最后一关是把结果写成文字。这一步最容易出两种错：一是把「统计上显著」当成「实际上重要」，二是把「相关」写成「因果」。这一节就守好这两道护栏。

4.4.1 统计显著　经济显著

星号多，只说明这个系数不太可能是偶然 (统计显著)；它没说这个效应在现实里有多大 (经济显著)。两者是两码事：

- 统计显著、但经济上微不足道：样本足够大时，一个小到可以忽略的效应也能打上三颗星。比如某政策让企业投资率提高了 0.02 个百分点，$p < 0.01$——统计上铁板钉钉，但这么小的变化在现实中约等于没有。
- 经济上很大、但统计不显著：系数很大，可标准误也很大 (比如样本太小、变量测得不准)，置信区间横跨正负——这时候不能说「有大效应」，只能说「估不准」。

回到我们的例子：教育回报 0.52(时薪/年) 既高度显著，放在平均时薪 4～5 美元的背景下也确实可观——这才是「既统计显著、又经济显著」。报告结果时，除了给系数和星号，一定要用变量的实际单位说清楚「这个数在现实里意味着什么」，比如「相当于均值的百分之几」「一个标准差的变化对应多少」。

还有一条边界：你的结论只对你的样本和总体成立。用在业女性样本估出的教育回报，未必适用于男性、未必适用于今天、更未必适用于别的国家。这条「外推边界」在第 3 讲讲过，写结论时别忘了它。

4.4.2 把回归表写成不过度解释的说明

最后是因果护栏。前面反复强调：回归系数天然是「比较」，不是「影响」。可一旦落笔，「导致」「提升」「促进」这类因果动词会不自觉地溜出来。下面这张对照表，是一份实用的改写清单——把左边的因果化措辞，换成右边的诚实表述：

过度解释 (因果化)	诚实表述 (比较 / 条件)
教育导致 / 提高了工资	教育与工资正相关；受教育更多的人工资更高
多读一年书能让工资涨 11%	在控制经验等变量后，受教育多一年对应工资高约 11%
盈利能力降低了负债率	盈利能力更强的公司，负债率倾向于更低
这项政策使企业绩效提升	政策实施后，处理组绩效相对对照组更高 (能否归因于政策，取决于识别假设)

一句话原则：能用「相关 / 更高 / 更低 / 在控制……之后 / 倾向于」的地方，就别用「导致 / 使 / 提升」。要下因果结论，必须另外说明识别策略 (这是后面几讲的主题)，而不是靠回归系数本身。做到这一点，你的实证写作就有了最基本的分寸感。

4.5 AI 协作

契约边界：本节逐条覆盖大纲「AI 协作训练」清单，不删项。

这一节是本讲最有特色的部分。前面几节反复出现同一类活儿：把系数翻译成人话、核对对数系数怎么读、盯住有没有把相关写成因果、给代码补注释——它们都专业、都琐碎、都极易出小错，恰恰是 **AI** 的用武之地。但用 AI 解读回归有一条不可退让的底线：

> **AI** 负责起草，你负责定夺。它给出的每一句解读，你都要回到变量定义和数据本身核对一遍。回归解读的错误往往很隐蔽 (把对数系数当绝对值、把相关说成因果)，一旦照单全收，错就顺着你的论文一路传下去。

下面四项训练逐条对应大纲，每项给出场景 → 可复制提示词 → 一个小例子 → 你要核对什么，并统一配套本讲的 skill `core/05-regression-interpreter`。四项可以串起来用：先让 AI 解读系数 (训练 1)，再核对对数/标准化读法 (训练 3)，写成文字后查因果措辞 (训练 2)，最后给复现代码补注释 (训练 4)——一条完整的「读表 → 解释 → 写作 → 留痕」链。

4.5.1 训练 1 • 让 AI 用自然语言解释核心系数

场景：拿到一张回归表，想快速、准确地把核心系数说成人话，省去反复斟酌措辞。把回归表连同变量定义一起喂给 AI：

```
你是一位计量经济学助教。下面是一张回归表和变量定义。
请逐个解释【核心系数】的含义，要求：
(1) 说清楚它的方向、大小和单位；
(2) 如果因变量或自变量取了对数，按半弹性/弹性正确解读；
(3) 用「比较」而非「因果」的措辞；
(4) 每个系数一句话，通俗但准确。

【变量定义】lwage=ln(时薪)；educ= 受教育年限；exper= 工作经验（年）
【回归表】educ 系数 0.116（0.015）***；exper 0.019（0.004）***
```

AI 大致会回：「受教育年限每多一年，时薪平均高约 11.6%(因变量取了对数，系数按百分比读)；在此基础上，工作经验每多一年，时薪平均高约 1.9%。二者均为在其他变量给定时的比较，不宜读作因果。」

你要核对：方向和量纲对不对——AI 偶尔会把对数系数 (0.116) 直接说成「高 0.116 元」，或把百分比小数点搞错。

4.5.2 训练 2 • 让 AI 检查有没有把相关写成因果

场景：结果段落写完，自己很难发现藏在字里行间的因果化措辞。让 AI 当一个「因果措辞审查员」，逐句挑刺。这一步直接对应上一节的因果护栏：

```
请审查下面这段实证结果的表述，找出其中把【相关关系】写成【因果关系】的地方。
对每一处：(1) 指出是哪个词（如" 导致"" 提升"" 使得"）；
(2) 说明为什么这样写超出了回归能支持的范围；
(3) 给出一个" 比较/条件" 措辞的改写版本。
不要改变原文的数字和结论强度以外的内容。

【待审查段落】我们发现，教育显著提升了女性工资，多读一年书能让时薪提高约 12%。
```

AI 大致会回：标出「提升」「能让」两处因果化措辞 → 指出回归只识别相关、未处理能力等混淆 → 改写为「教育与女性工资显著正相关，受教育多一年对应时薪高约 12%」。

你要核对：AI 有时矫枉过正，把本就恰当的表述 (如「正相关」) 也标成问题；由你判断哪些该改、哪些保留。

4.5.3 训练 3 ·让 AI 核对对数、标准化的系数解读

场景：变量做过对数、标准化之后，系数读法最容易记混 (半弹性？弹性？标准差？)。把变量字典 (标注每个变量做过什么变换) 连同你的解读草稿交给 AI 交叉核对：

```
下面是变量字典和我对系数的解读草稿。请核对我的解读与变量的【变换方式】是否一致：
- 若因变量是 ln(·)，系数应按" 百分比/半弹性" 读；
- 若自变量和因变量都是 ln(·)，系数应按" 弹性" 读；
- 若变量做了标准化，系数应按" 标准差" 读。
指出不一致之处并改正。

【变量字典】lnY=ln(产出)；lnL=ln(劳动)，均取对数
【我的解读草稿】lnL 系数 0.6，说明劳动每增加 1 单位，产出增加 0.6 单位。
```

AI 大致会回：指出错误——两端都是对数，0.6 是弹性，应读作「劳动每增加 **1%**，产出增加约 **0.6%**」，而非「1 单位 →0.6 单位」。

你要核对：变换方式以你的数据处理代码为准，别让 AI 凭变量名猜 (有的变量名带 `ln` 却没真取对数，反之亦然)。

4.5.4 训练 4 ·让 AI 把回归代码改写成注释版

场景：一段光秃秃的 do 代码，交接或复看时看不懂。让 AI 补上分层大纲注释和逐行说明，方便在 VS Code 里按章节导航 (呼应本书 do 文件的 `**#` 注释规范)：

```
请为下面这段 Stata 回归代码添加注释，要求：
(1) 用 Stata 的 **# / **## 多级大纲注释标出每个分析步骤；
(2) 关键命令后加一句行内注释，说明它在做什么、为什么这么做；
(3) 不改动代码逻辑本身。

【原始代码】
reg tl npr size c.npr#c.size tang fr, robust
margins, dydx(npr) at(size=(19(1)24))
```

AI 大致会回：

```
**# 交叉项回归：盈利能力对负债率的影响是否随规模变化
reg tl npr size c.npr#c.size tang fr, robust    // 交乘项 npr#size；robust 应对异方差
**## 边际效应：不同规模处 npr 的作用
margins, dydx(npr) at(size=(19(1)24))           // 扫描 size=19..24，看 dtl/dnpr 怎么变
```

你要核对：确认 AI 没有顺手「优化」掉你的原设定——比如悄悄改了控制变量、去掉了 `robust`、或改了 `margins` 的取值范围。

> **想要 Python 版代码？一句话的事**
>
> 本讲的回归、交叉项、平方项，用 Python(`statsmodels` / `linearmodels`) 实现都非常直接，因此不再单设 Python 附录。需要时，把 Stata 代码丢给 AI 让它翻译即可：
>
> ```
> 请把下面这段 Stata 回归代码改写成等价的 Python 代码，
> 使用 pandas 读取数据、statsmodels 做 OLS，
> 并保留稳健标准误（cov_type='HC1'）的设定。
> 【Stata 代码】reg wage educ exper, robust
> ```
>
> 得到的 Python 代码同样要自己跑一遍、对一下系数是否与 Stata 一致。

4.6 小结与练习

本讲一条主线：回归估计的是条件期望函数 **(CEF)**——「给定 X，Y 的平均值」。给这条 CEF 选不同的函数形式 $f(\cdot)$(线性、平方、对数、交叉项)，就得到各种模型；OLS 是估计它们的一种方法，不是模型本身。系数刻画的是特定条件下的比较，不是「影响」；标准误刻画这个比较有多不确定，异方差和组内相关都要用稳健/聚类标准误来应对。把这些连起来，你就能读懂一张回归表，并写出不过度解释的结果说明。

练习

1. 用 `sysuse nlsw88` 数据，跑 `reg wage ttl_exp`，再跑 `reg wage ttl_exp c.ttl_exp#c.ttl_exp`(加入总工龄的平方项)。用二次项系数算出拐点 $(-b/2a)$，并结合 `sum ttl_exp` 看拐点是否落在样本范围内、有没有实际意义。

> **参考答案**
>
> 加入平方项后，一阶项系数为正、平方项系数为负，说明工资随总工龄先升后降。拐点 $x^* = -b/(2a)$。关键是把它和 `sum ttl_exp` 的取值范围对照：若拐点落在样本上限附近甚至之外，说明绝大多数人都在上升段，「到某年后工资回落」只是极少数长工龄样本的外推，不宜当成普遍规律。这道题练的就是「看到倒 U 先问拐点在哪」。

2. 下面这段结果说明有没有过度解释的地方？请改写：「回归显示，企业规模每扩大 1%，研发投入提升 0.3%，说明做大规模能促进创新。」

参考答案

问题出在「提升」和「促进」两个因果动词，以及「做大规模能」这个因果断言。回归给出的是相关，不是因果——规模大的企业研发多，也可能是因为它们本就处在研发密集的行业，或有更多现金流。改写：「回归显示，企业规模与研发投入正相关，规模每高 1%，研发投入平均高约 0.3%。至于扩大规模是否会带动创新，需进一步的识别策略才能判断。」

3. **(交给 agent)** 找一篇你正在读的论文里的一张主回归表，把它连同变量定义交给 AI(用本讲 AI 协作的第一段提示词)，让它逐个解释核心系数；再用第二段提示词，让 AI 检查你自己写的一段转述有没有把相关写成因果。把 AI 的解读和你自己的理解对照，记下不一致的地方。

提示

这道题没有标准答案，重点在流程：**AI** 起草 → 你回到变量定义和数据核对 → 记录分歧。特别留意 AI 会不会把对数系数读错、会不会用了因果措辞。核对本身，就是最好的练习。

4.7 延伸阅读

- 标准误与聚类的系统讨论：
  - Cameron, A. C., & Miller, D. L. (2015). A practitioner's guide to cluster-robust inference. *Journal of Human Resources*, 50(2), 317–372. Link, PDF, Google.
  - Abadie, A., Athey, S., Imbens, G. W., & Wooldridge, J. M. (2023). When should you adjust standard errors for clustering? *The Quarterly Journal of Economics*, 138(1), 1–35. Link (rep), PDF, Google.
- 入门教材 (本讲写法的底本，深浅适中)：
  - Wooldridge, J. M. (2019). *Introductory Econometrics: A Modern Approach* (7th ed.). Cengage. Google.
  - James, G., Witten, D., Hastie, T., & Tibshirani, R. (2023). *An Introduction to Statistical Learning with Applications in Python* (ISLP). Springer. Link, Google.
  - CEF 与线性投影的规范定义 (进阶)：Hansen, B. E. (2022). *Econometrics*. Princeton University Press. Link, PDF.
- 连享会专题 (中文，可直接上手)：
  - 交乘项与边际效应：连享会 • 交乘项专题；
  - 聚类标准误：Stata：聚类调整后的标准误；
  - 断点回归与现代因果推断方法 (高级班衔接)：https://www.lianxh.cn/details/1811.html。

5 FWL、虚拟变量与固定效应：理解条件比较

> **本讲要点**
>
> - FWL 定理的直觉：残差化之后的比较；
> - 控制变量、遗漏变量、坏控制和过度控制；
> - 虚拟变量、分类变量、基准组和虚拟变量陷阱；
> - 个体固定效应、时间固定效应和双向固定效应；
> - 高维固定效应与交互固定效应的基本思想；
> - DID 中的组别、时间和处理交乘项；
> - 如何理解带固定效应和交乘项模型中的系数。
> - 本讲用到的 skills：`core/05-regression-interpreter`；
> - 可运行伴生文件：`examples/ch05/ch05_main.do`(Stata)、`examples/ch05/ch05_python.ipynb`(Python)。

本讲的主线只有一句话：回归系数从来不是「影响」，而是「在控制了别的东西之后，拿谁跟谁比」。把这句话讲透，就能从最基本的一元回归，一步步走到面板数据的固定效应模型，再走到政策评估里最常用的双重差分 (DID)。串起这条路的，是一条叫 FWL 的定理——它告诉我们，多元回归的每个系数，本质上都是一次「先把别的变量剔除干净，再拿残差去比」的操作。理解了残差化，虚拟变量、固定效应和 DID 就都是同一件事的不同说法。

5.1 案例情境：三家公司的一张图

先看一个上课时常用的例子。横轴是公司规模 ($Size = \ln(\text{总资产})$)，纵轴是总资产回报率 (ROA)。图 图 5.1 上半部分有三家公司：左上角那家起步时规模小、ROA 相对较高；右下角那家起步时规模就大、ROA 相对较低。这种公司之间的差异，与所处行业、所有权属性有关，基本不随时间变化。

我们想问的是：一家公司规模变大以后，它的 **ROA** 会怎么变。可要是无视三家公司之间的差异，把所有点混在一起回归，得到的斜率是负的——「规模越大，ROA 越低」。图的下半部分给出组内去心 (de-mean，把每家公司的均值减掉) 之后的结果：公司之间的差异被减掉，三家公司各自内部的正向关系显露出来，斜率翻正。

差别不在数据，而在拿谁跟谁比：混合回归回答的是「大公司是不是比小公司 ROA 低」，用的是公司之间的差异；组内去心回答的是「同一家公司变大以后 ROA 怎么变」，只用每家公司内部的变化。两个问题都可以研究，但它们不是同一个问题；把前者的答案当成后者的答案，就是遗漏变量偏误。

本讲要做的，就是把「换一次控制、就换一次比较对象」这件事，从这张图出发，一路讲到固定效应和 DID。

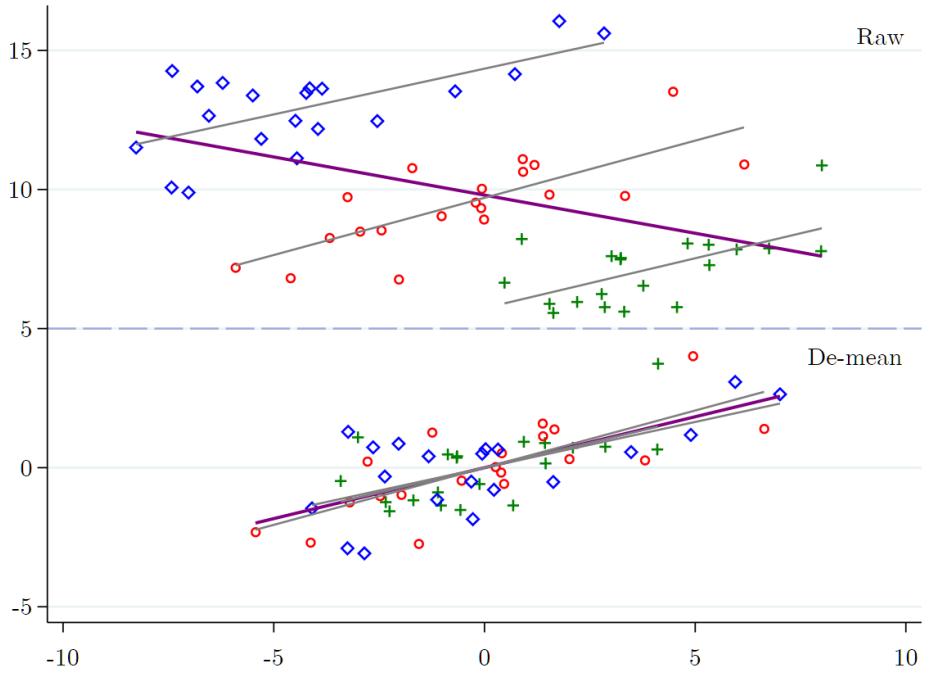

图 5.1: 辛普森悖论：混合回归得到负斜率，组内去心之后每家公司内部的关系为正

5.2 从常数项到虚拟变量：残差化的第一课

5.2.1 常数项：回归里最容易被忽略的那一项

从最简单的一元回归讲起：

$$Y_i = \alpha + \beta X_i + u_i \tag{5.1}$$

这里的常数项 α 看着不起眼，却藏着理解后面一切的钥匙。把 式 5.1 两边取均值：

$$\bar{Y} = \alpha + \beta \bar{X} + \bar{u}$$

两式相减，常数项 α 就被消掉了：

$$Y_i - \bar{Y} = (X_i - \bar{X})\beta + (u_i - \bar{u}) \tag{5.2}$$

这说明一件事：带常数项的回归，等价于先把 Y 和 X 都减去各自的均值，再做一次不带常数项的回归。常数项在这里扮演的角色，就是「把样本均值这一层信息吸走」。换句话说，β 度量的从来不是 X 和 Y 的绝对水平关系，而是两者相对各自均值的偏离之间的关系。

常数项还有几件不显眼却要紧的活儿：它会随变量的平移而调整，并保证常规 OLS 回归中的 $R^2 \in [0,1]$。在 DID 和 RDD 里，常数项本身还可能带明确的经济含义。这里先记住最核心的一点：回归系数度量的是「去掉某个基准之后」的比较。

5.2.2 FWL 定理：先剔除，再回归

把上面「减均值」的操作一般化，就是 Frisch-Waugh-Lovell(FWL) 定理。考虑一个多元回归：

$$Y = \alpha + X_1\beta_1 + X_2\beta_2 + u \tag{5.3}$$

我们只关心 X_1 的系数 β_1。FWL 定理说，要得到 $\hat{\beta}_1$，不必把 X_1、X_2 一起放进回归，可以分三步走：

1. 用 X_2 回归 Y，取残差 $\tilde{Y}$(把 X_2 对 Y 的影响剔除干净)；
2. 用 X_2 回归 X_1，取残差 $\tilde{X}_1$(把 X_2 对 X_1 的影响剔除干净)；
3. 用 $\tilde{X}_1$ 回归 $\tilde{Y}$，得到的系数恰好等于 式 5.3 里的 $\hat{\beta}_1$。

用一句话概括：「先把其他变量剔除、再回归」与「一起回归」等价。上一小节的减均值，只是 FWL 的一个特例——那里的 X_2 就是常数项 (一列 1)，「剔除 X_2」正好就是「减去均值」。

FWL 给我们一个极其有用的视角：多元回归里 X_1 的系数，度量的是「X_1 中剔除掉其他变量之后剩下的那部分」与「Y 中剔除掉其他变量之后剩下的那部分」之间的关系。系数是「控制其他变量之后的比较」，这句在上一讲就出现过的话，到这里有了精确的数学表述：所谓「控制」，就是「残差化」；所谓「比较」，就是拿两个残差去回归。

> **为什么值得专门讲 FWL**
>
> FWL 不只是一个理论性质。Stata 里处理高维固定效应的 `reghdfe`、`ivreg2` / `lasso2` 里的 `partial()` 选项，以及画部分回归图的 `avplot` 等命令，都可以从残差化的角度理解。理解了「剔除—回归」，后面的固定效应就是它的直接应用。
> 双重 / 去偏机器学习 (DML / DDML) 也保留了这一骨架：分别用控制变量预测结果和处理变量，取两边残差后再估计核心关系。不过，DML 还要处理高维、非线性和机器学习过拟合的问题，通常需要样本分割与交叉拟合；它不能替代识别策略本身。Chernozhukov et al. (2018) 给出了系统讨论，进阶读者可继续阅读双重机器学习：DDML。

图 5.2 用韦恩图把这个过程画了出来。这是一张帮助理解残差化的示意图，不是严格的集合或方差分解。Y、X_1、X_2 三个圆的重叠部分，代表彼此能够解释的方差。把 Y 圆内部按字母拆开：

$$Y = A + B + C + D$$

其中，A 是 Y 里谁也没有解释的部分，B 只被 X_1 解释，D 只被 X_2 解释，C 被 X_1 和 X_2 共同解释。所谓「把 X_2 partial out」，直观上就是拿一把剪刀，把 X_2 那个圆整个剪掉：Y 里凡是和 X_2 重叠的部分 (也就是 C 和 D) 都被剪走，只剩下：

$$\widetilde{Y} = A + B$$

同样地，X_1 剪掉与 X_2 重叠的部分后得到残差 $\widetilde{X}_1$。再用 $\widetilde{Y}$ 对 $\widetilde{X}_1$ 回归，两者真正共享的只有区域 B——也就是「X_1 里扣掉了 X_2、Y 里也扣掉了 X_2」之后，两边还共同拥有的那块方差。β_1 就是从区域 B 里估出来的。

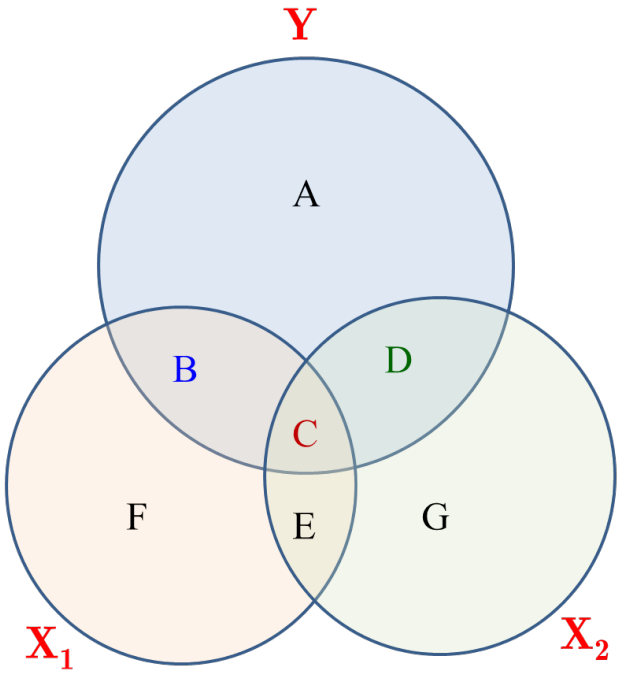

图 5.2: FWL 定理的方差韦恩图：β_1 只用到区域 B——把 X_2 从 X_1 和 Y 中同时剪掉之后，两者还共同解释的那部分方差

这个定理看似只是一个计算技巧，实际上是一整套方法的理论基础。例如，在 Stata 中，`xtreg` 命令的 `fe` 选项、`reghdfe` 命令的 `absorb()` 选项、`ivreg2` 和 `lasso2` 命令的 `partial()` 选项，都是 FWL 的直接应用——它们先把固定效应或一部分控制变量 partial out，再对核心变量做回归。

此处要特别注意：要得到干净的 β_1，必须把控制变量从核心解释变量和结果变量两边都剔除，而不是只从结果变量一边剔除。只处理 Y 而不处理 X_1，得到的并不是原多元回归中 β_1 的 FWL 表达。

5.2.3 遗漏变量、坏控制与过度控制

FWL 也让「遗漏变量偏误」一眼可见。设真实模型是 式 5.3，但我们偷懒，只放 X_1：

$$\text{只估 } Y = X_1\theta_1 + \underbrace{\varepsilon}_{X_2\beta_2+u}$$

此时 X_2 被塞进了误差项。只要 X_1 与 X_2 相关 ($Corr(X_1, X_2) \neq 0$)，$\hat{\theta}_1$ 就不等于 β_1——它把 X_2 的一部分贡献也算到了 X_1 头上。辛普森悖论就是这么来的：真实模型里每家公司有自己的截距 (个体效应)，漏掉它，规模的系数就被公司间差异污染。

但控制变量并非越多越好。多放一个变量，能不能让系数更干净，取决于这个变量站在因果链条的哪个位置：

- 好的控制变量：它同时影响 X_1 和 Y，是二者共同的混淆源。放进去，能挡住一条虚假的关联通道。
- 坏控制 **(bad control)**：它本身是被 X_1 影响的结果 (处理之后才发生的中间变量)。把它当控制变量放进去，等于把一部分真实效应也一起「控制」掉了，系数反而偏了。
- 过度控制 **(over-control)**：控制了处理效应必经的中介，或控制了与 X_1 高度共线、几乎不含独立信息的变量，识别变异被抽干，系数变得不稳、方差飙升。

判断的标准不是「显著与否」「R^2 高不高」，而是这个变量是原因还是结果。识别策略层面的深入讨论超出本讲范围，这里只需记住一句：控制变量要控在混淆上，不要控在结果或中介上。

有关控制变量的讨论，参见如下推文：

- 付一帆, 2022, Stata：控制变量与核心解释变量地位对等吗？.
- 张雪娇, 2022, 控制变量如何选？大牛们的 10 条建议.
- 曹昊煜, 2022, A-理论部分：控制变量！控制变量！Good-Controls-Bad-Controls.
- 李烨阳, 2022, B-Stata 模拟：控制变量！控制变量！Good-Controls-Bad-Controls.
- 秦利宾, 2021, 控制变量！控制变量！.
- 连玉君, 张静瑶, 2020, 加入控制变量后结果悲催了！.

5.2.4 虚拟变量、基准组与虚拟变量陷阱

把 FWL 里的 X_2 换成一个分类变量，就得到虚拟变量 (dummy variable)。设一份数据里既有男性 ($M = 1$) 也有女性 ($M = 0$)，模型写成：

$$y_i = \alpha + \gamma M_i + \beta x_i + u_i \tag{5.4}$$

γ 的含义一目了然——它是男性组相对女性组的截距差：

$$E[y_i \mid M = 0] = \alpha + \beta x_i, \qquad E[y_i \mid M = 1] = (\alpha + \gamma) + \beta x_i$$

图 图 5.3 里，两组是两条平行线 (斜率同为 β)，γ 就是两条线的垂直间距。这时「女性组」被当作基准组 (base group)，它的截距是 α，其余各组的虚拟变量系数都是「相对基准组」的差。

这里藏着一个初学者最常踩的坑——虚拟变量陷阱 **(dummy variable trap)**。一个分类变量若有 k 个类别，把 k 个虚拟变量连同常数项一起放进回归，就会因为「k 个虚拟变量之和恒等于常数项那一列 1」而完全共线，模型无法估计。解决办法只有二选一：留一个基准组 (放

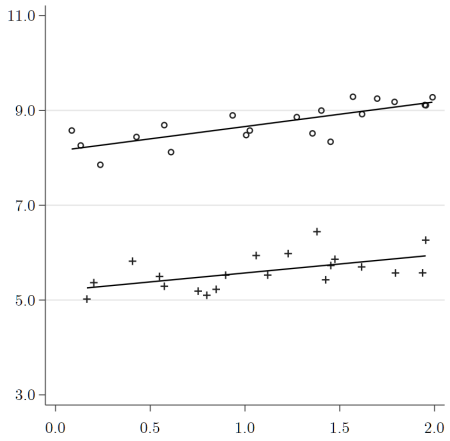

图 5.3: 虚拟变量 = 变截距：两组共用斜率 β，虚拟变量系数 γ 是两条平行线的垂直间距

$k-1$ 个虚拟变量 + 常数项)，或去掉常数项 (放满 k 个虚拟变量，此时每个系数是该组的组内截距，而非相对差)。二者的斜率估计完全相同，区别只在截距怎么解读。

在 Stata 里，这些都由因子变量 (factor variable) 自动打理，不必手工造虚拟变量：

符号	含义	实例
`i.`	声明为分类变量 (自动留基准组、生成虚拟变量)	`reg y x i.id i.year`
`c.`	声明为连续变量	`c.age`
`ib#.`	指定第 `#` 类为基准组 (默认取最小值那类)	`ib2.race`
`#`	两变量的交乘项	`i.D#c.x`
`##`	两变量各自及其交乘项	`x##z`

例如 `reg wage i.race c.hours i.race#c.hours`，一行就把「不同种族的工资水平差」和「不同种族里工时回报的差异」都估了出来——这正是把虚拟变量和交乘项组合起来用。理解了「虚拟变量 = 变截距」，就为下一节的固定效应做好了全部准备：固定效应，无非是把「个体」也当成一个分类变量，给每个个体一条自己的截距。

5.3 固定效应模型：从虚拟变量到组内去心

5.3.1 三类变量、三个下标

面板数据每个观测有两个下标：个体 i、时间 t。影响 y_{it} 的变量可以按「随谁变」分成三类：

- 不随时间变的 x_i：公司文化、个人能力、地区禀赋、所有权属性；
- 不随个体变的 x_t：宏观冲击、货币政策、经济政策不确定性；
- 既随时间又随个体变的 x_{it}：规模、杠杆、投资、研发。

麻烦在于，x_i 和 x_t 里有大量成员观测不到，又偏偏和我们关心的 x_{it} 相关。以杠杆率为例，假设真实模型里所有权属性 SOE_i 影响杠杆，但我们观测不到它：

$$\text{真实：}\ Lev_{it} = \alpha + SOE_i + Size_{it}\beta + \varepsilon_{it}, \qquad \text{只估：}\ Lev_{it} = \alpha + Size_{it}\theta + u_{it}$$

只要国企普遍规模更大、杠杆更高 ($Corr(SOE_i, Size_{it}) \neq 0$)，$\hat{\theta}$ 就把所有权的贡献混了进来。这就是上一节遗漏变量偏误的面板版本。

5.3.2 组内去心：不可观测的因素也能控制

固定效应模型的办法直接而漂亮：把所有不随时间变化的因素——无论能不能观测——统统打包进一个个体截距 α_i，再用 FWL 的减均值把它剔除掉：

$$\begin{aligned} y_{it} &= \alpha_i + x_{it}\beta + \varepsilon_{it} \\ \bar{y}_i &= \alpha_i + \bar{x}_i\beta + \bar{\varepsilon}_i \\ y_{it} - \bar{y}_i &= (x_{it} - \bar{x}_i)\beta + (\alpha_i - \alpha_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) \end{aligned} \tag{5.5}$$

第三行里 $\alpha_i - \alpha_i = 0$，这一步就是「不用观测也能控制」的全部秘密：我们从没测量过 SOE_i、公司文化或管理层风格，但只要它们不随时间变，组内去心就把它们连同 α_i 一起减掉了。对去心后的数据做 OLS，得到的 $\hat{\beta}$，和直接在模型里放 N 个个体虚拟变量完全等价——这又是 FWL 定理的一次应用。「给每个个体一条截距」= 放 N 个虚拟变量 = 组内去心，三种说法是同一件事。这也正是上一节「虚拟变量 = 变截距」在面板上的延伸。

Stata 里至少有四种等价写法，前三种是现成命令，第四种手动去心，用来确认自己真懂了组内变换：

```
xtset id year
xtreg   y x, fe vce(cluster id)              // [1] 组内估计量
areg    y x, absorb(id) vce(cluster id)      // [2]
reghdfe y x, absorb(id) vce(cluster id)      // [3]

bysort id: egen y_bar = mean(y)              // [4] 手动去心
bysort id: egen x_bar = mean(x)
gen c_y = y - y_bar
gen c_x = x - x_bar
reg c_y c_x, noconstant vce(cluster id)      //     斜率同上
```

四种方法的 $\hat{\beta}$ 完全相同。差别只在于：它用的是每家公司内部的变化，而不是公司之间的差异。

5.3.3 个体、时间与双向固定效应

只放个体截距 α_i，是个体固定效应：控制掉一切不随时间变的个体差异。再加一个年份截距 λ_t，就是实证中用得最多的双向固定效应 **(TWFE)**：

$$Y_{it} = x_{it}'\beta + \alpha_i + \lambda_t + \varepsilon_{it} \tag{5.6}$$

写成虚拟变量的形式，它的含义更清楚：

$$Y_{it} = x_{it}'\beta + \sum_{i=1}^{N} \alpha_i A_i + \sum_{t=1}^{T} \lambda_t B_t + \varepsilon_{it}$$

α_i 吸收「不随时间变的个体差异」，λ_t 吸收「当年对所有个体一致的宏观冲击」。也可以只放 λ_t (时间固定效应)，控制共同的时间趋势而不管个体差异。放哪几组、怎么放，取决于你想控制什么。

在完整的平衡面板中，可以把 式 5.5 的思路再推进一步：对每个变量同时减去个体均值和年份均值，再加回总体均值。以 y 为例：

$$\tilde{y}_{it} = y_{it} - \bar{y}_i - \bar{y}_t + \bar{y} \tag{5.7}$$

其中，$\bar{y}_i$ 是个体 i 的时间均值，$\bar{y}_t$ 是年份 t 的截面均值，$\bar{y}$ 是总体均值。

对 $x_1, x_2, \ldots$ 做相同变换后，用 $\tilde{y}_{it}$ 对 $\tilde{\mathbf{x}}_{it}$ 做不含常数项的 OLS，得到的斜率与 式 5.6 中的双向固定效应估计一致。这个公式适合帮助理解和核对一个小型、完整的平衡面板。实证数据常常是不平衡面板，或者还要吸收交互固定效应；这时不应把逐次减均值当成通用算法，而应直接使用固定效应估计命令。

实证分析中，下面两种写法在斜率上等价。第一种显式放入年份虚拟变量；第二种把个体和年份固定效应交给 `reghdfe` 吸收：

```
xtset id year
xtreg y x1 x2 i.year, fe vce(cluster id)        // 显式加入年份效应
reghdfe y x1 x2, absorb(id year) ///
    vce(cluster id)                              // 吸收两组固定效应
```

下面的手工代码只用于与命令结果核对。它要求每个个体在每一年都有观测；真实项目中不要据此替代 `reghdfe`：

```
xtset id year
* 仅用于完整平衡面板；`center` 为 SSC 外部命令。
bysort id: center y x1 x2, inplace
bysort year: center y x1 x2, inplace
reg y x1 x2, noconstant vce(cluster id)
```

⚠ 常见问题：为什么固定效应放不进性别和种族

初学者常犯的一个错，是想在个体固定效应模型里加入性别、种族、出生地这类变量，结果 Stata 报告它们被 (omitted) 了。原因就在 式 5.5：个体效应 α_i 不随时间变，而性别、种族在样本期内也不随时间变，二者完全共线——组内去心把 α_i 减掉的同时，也把所有时不变变量一起减成了 0，Stata 只能删掉它们。

一句话：固定效应能「控制」个体的一切时不变特征，却无法「估计」这些特征各自的系数。若研究问题恰恰是「种族对工资的影响」这类时不变变量的效应，就不能用固定效应，需要改用组间估计、加控制变量的混合回归，或 Hausman–Taylor(`xthtaylor`)、`xtseqreg` 等专门方法。

按此逻辑，大家也不难理解，在 TWFE 模型中也无法加入只随时间变化的宏观变量，如 M2 增速 ($M2_t$)、经济政策不确定性 (EPU_t) 等。

5.4 高维固定效应：更多维度与更窄的比较

在教育经济、移民和公司金融等领域，经常需要使用高维固定效应模型。其中尤以国际贸易领域的「引力模型」影响最大：研究者常在模型中同时放入三维甚至四维固定效应及其交互项。[1] 若允许经济变量的影响具有异质性，还可以把固定效应与解释变量交乘。

下面以三维固定效应模型为例说明。更一般的情形在原理上与此相似；多维变换的矩阵推导可参见 Balazsi, Matyas and Wansbeek (2018)。[2]

5.4.1 三维加性固定效应

假设我们研究全国 31 个省份在一段时间内的行业产出。数据有三个维度：省份 ($i = 1, \dots, N$)、行业 ($j = 1, \dots, G$) 和年份 ($t = 1, \dots, T$)。一个三维加性固定效应模型可写为：

$$y_{ijt} = \mathbf{x}_{ijt}'\beta + \alpha_i + \gamma_j + \lambda_t + \varepsilon_{ijt} \tag{5.8}$$

沿袭个体固定效应和双向固定效应的思路，在完整平衡面板中，可以用下面的变换同时去除三组固定效应：

$$\begin{aligned} \tilde{y}_{ijt} = y_{ijt} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j\cdot} - \bar{y}_{\cdot\cdot t} \\ + 2\bar{y}_{\cdot\cdot\cdot}. \end{aligned} \tag{5.9}$$

这里的点号表示对相应维度取均值。例如，$\bar{y}_{i\cdot\cdot}$ 是省份 i 跨行业和年份的均值，$\bar{y}_{\cdot\cdot\cdot}$ 是全样本均值。对解释变量作同样变换后回归残差，斜率与直接放入三组虚拟变量相同。需要说明的是，多维去均值的写法并不唯一；上式是完整平衡面板下最直观的教学表达。[3]

经过组内变换后的系数与包含多组虚拟变量的 OLS 回归在斜率上相同，但标准误仍须按误差相关结构处理。实际估计交给 `reghdfe` 更稳妥。[4]

```
reghdfe y x1 x2, absorb(province industry year) ///
    vce(cluster province)
```

若只是想把 式 5.9 的操作逐步看一遍，可以在完整平衡面板中使用 `center`。这段代码只用于教学核对；不平衡面板、嵌套固定效应和交互固定效应都应直接交给 `reghdfe`：

```
* `center` 为 SSC 外部命令；下列三次去心会原地替换变量。
bysort province: center y x1 x2, inplace
bysort industry: center y x1 x2, inplace
bysort year: center y x1 x2, inplace
reg y x1 x2, noconstant vce(cluster province)
```

[1] 引力模型的经典讨论可参见 Silva and Tenreyro (2006)；系统综述见 Head and Mayer (2014)。

[2] Balazsi, Matyas and Wansbeek (2018) 给出了多维固定效应变换的矩阵讨论，并强调变换写法不止一种。

[3] Balazsi, Matyas and Wansbeek (2018) 给出了多维固定效应变换的矩阵讨论，并强调变换写法不止一种。

[4] `reghdfe` 通过迭代吸收多组固定效应；命令的说明、算法边界和版本信息应以 官方帮助页 为准。它处理的是固定效应残差化，不替研究者决定应当吸收什么或怎样聚类。

5.4.2 交互固定效应

国际贸易领域的引力模型通常会包含交互固定效应。[5] 例如，Egger and Pfaffermayr (2003) 使用双边固定效应来反映贸易双方的交互影响。[6] 以贸易数据为例，`exporter#importer` 吸收的是每一对贸易伙伴长期不变的双边差异，例如距离、共同语言和历史联系。最简单的设定为：

$$y_{ijt} = \mathbf{x}_{ijt}'\beta + \gamma_{ij} + \varepsilon_{ijt} \tag{5.10}$$

其中，γ_{ij} 是双边固定效应。若只需观察它的去均值含义，可以写成 $\tilde{y}_{ijt} = y_{ijt} - \bar{y}_{ij\cdot}$；但估计时仍应直接吸收该交互项：

```
reghdfe y x1 x2, absorb(exporter#importer) ///
    vce(cluster exporter importer)
```

实证分析中应用更为广泛的是在 式 5.10 中进一步加入时间固定效应：

$$y_{ijt} = \mathbf{x}_{ijt}'\beta + \gamma_{ij} + \lambda_t + \varepsilon_{ijt} \tag{5.11}$$

```
reghdfe y x1 x2, absorb(exporter#importer year) ///
    vce(cluster exporter importer)
```

Matyas (2017) 归纳了更多设定方式。[7] 例如，有时只需要控制「行业 × 年份」效应；有时则要同时控制「省份 × 年份」与「行业 × 年份」效应：

$$\begin{aligned} y_{ijt} &= \beta' \mathbf{x}_{ijt} + \alpha_{jt} + \varepsilon_{ijt}, \\ y_{ijt} &= \beta' \mathbf{x}_{ijt} + \alpha_{it} + \delta_{jt} + \varepsilon_{ijt}. \end{aligned} \tag{5.12}$$

如果把各个两两交互效应都考虑在内，得到的完整设定是：

$$y_{ijt} = \beta' \mathbf{x}_{ijt} + \gamma_{ij} + \alpha_{it} + \delta_{jt} + \varepsilon_{ijt} \tag{5.13}$$

```
reghdfe y x1 x2, ///
    absorb(exporter#importer exporter#year importer#year) ///
    vce(cluster exporter importer)
```

[5] 引力模型的经典讨论可参见 Silva and Tenreyro (2006)；系统综述见 Head and Mayer (2014)。
[6] Egger and Pfaffermayr (2003) 讨论了欧洲贸易一体化中的引力模型设定。
[7] Matyas (ed., 2017) 系统整理了引力模型及其多种固定效应设定。

在完整平衡面板中，这一设定对应的去均值公式也可以明确写出：

$$\begin{aligned}\tilde{y}_{ijt} = y_{ijt} &- \bar{y}_{ij\cdot} - \bar{y}_{\cdot jt} - \bar{y}_{i\cdot t} \\ &+ \bar{y}_{\cdot\cdot t} + \bar{y}_{\cdot j\cdot} + \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdots}.\end{aligned} \tag{5.14}$$

这个公式的价值在于帮助我们看清每一组固定效应吸收了什么；实际数据很少整齐到适合手工处理，因此实务上仍应使用 `reghdfe`。

这类设定并不意味着「固定效应越多越好」。每增加一组固定效应，得到的比较对象就更窄；如果核心解释变量只在「行业 × 年份」层面变化，又吸收了 `industry#year`，它会被完全吸收并显示为 `omitted`。这表示模型中没有可用于估计该变量系数的剩余变异，而不是软件故障。

5.4.3 高维固定效应的基本思想

固定效应的组数和维度一旦上去，直接放入虚拟变量在计算上就不再现实。两个典型场景是：国际贸易的引力模型需要同时控制出口国 × 年份、进口国 × 年份和国家对；劳动经济学的雇主—雇员匹配数据则可能同时包含百万级的工人效应和数十万个企业效应。把这些效应逐一写成 `reg y x i.id i.firm i.year` 这样的虚拟变量，内存和计算时间都可能难以承受。

`reghdfe` 的基本做法是交替去心：轮流对每一组固定效应做组内变换，反复迭代直到收敛。本质上仍是 FWL 定理的反复应用；对使用者来说，只需要把要吸收的固定效应写进 `absorb()`：

```
reghdfe y x1 x2, ///
    absorb(exporter#year importer#year exporter#importer) ///
    vce(cluster exporter importer)
```

使用时，应同时记住下面三个判断：

1. **`absorb()` 与 `vce(cluster)` 是两项不同的决定。**前者由需要控制哪些未观测因素决定；后者由误差可能在哪个层面相关决定。二者可以相同，也经常不同。
2. **不要把手工逐次去均值当成通用算法。**对完整平衡面板，上面的公式可以用于教学核对；对不平衡面板和多组交互效应，应让 `reghdfe` 的迭代算法完成残差化。
3. **先检查核心变量的变异层级。**固定效应吸收之后，核心解释变量还必须保留足够的组内变异；否则，即使程序能运行，研究问题也已经无法由该模型回答。

关于三维与交互固定效应的矩阵推导，可参见 Balazsi, Matyas and Wansbeek (2018)。[8] 引力模型中常见的设定整理见 Matyas (2017)。[9]

[8] Balazsi, Matyas and Wansbeek (2018) 给出了多维固定效应变换的矩阵讨论，并强调变换写法不止一种。
[9] 引力模型的经典讨论可参见 Silva and Tenreyro (2006)；系统综述见 Head and Mayer (2014)。

5.5 双向固定效应与 DID

5.5.1 每加一组固定效应，就换一次比较对象

把前面几种设定排在一起，会看到一条清晰的递进——每往模型里多放一组固定效应，被控制的混淆更多，系数回答的问题也随之收窄：

表 5.2: 固定效应设定与比较对象的递进

设定	`absorb()`	被吸收的混淆	系数回答的问题
混合回归 POLS	无	无	公司间差异与公司内变化混在一起
个体 FE	`id`	不随时间变的公司特征	同一家公司变化时，y 怎么变
双向 FE(TWFE)	`id year`	＋当年对所有公司一致的宏观冲击	相对同年其他公司，该公司的变化带来什么

表 表 5.2 是本讲最要紧的一张表：固定效应设定，本质上是研究者在向读者声明「我在拿谁和谁比」。这也回答了一个关键问题——既然固定效应会吸收掉同层级的变量，为什么 DID 的处理变量能在双向固定效应下活下来？因为 DID 的处理变量比 α_i 和 λ_t 都「高一个层级」，它在个体 × 时间维度上变动，恰好落在双向固定效应吸不掉的那层变异里。

5.5.2 2×2 DID：看清「两次差分」

DID 是政策评估里最干净的一个设定。最基本的情形是「两组两期」：处理组在政策后受处理，对照组始终不受处理。用 $Treat_i = 1$ 标记处理组，$Post_t = 1$ 标记政策后，基础回归是：

$$y_{it} = \alpha + \theta_1 Treat_i + \theta_2 Post_t + \gamma\,(Treat_i \times Post_t) + \varepsilon_{it} \tag{5.15}$$

真正关心的是交乘项 γ。把四个「组别—时期」的均值写出来 (C_0, C_1 是对照组政策前后，Y_0, Y_1 是处理组政策前后)：

表 5.3: 2×2 DID 的四格均值

	政策前	政策后	前后变化
对照组	C_0	C_1	$C_1 - C_0$
处理组	Y_0	Y_1	$Y_1 - Y_0$
组间差异	$Y_0 - C_0$	$Y_1 - C_1$	$(Y_1 - C_1) - (Y_0 - C_0)$

三个系数各有身份，一个都不能混：

- $\theta_1 = Y_0 - C_0$：政策前两组的水平差 (两组本来就不一样，不是政策效应)；
- $\theta_2 = C_1 - C_0$：对照组的时间变化 (对照组没受政策，用它刻画共同的时间趋势)；
- 交乘项 γ：

$$\gamma = (Y_1 - C_1) - (Y_0 - C_0) = (Y_1 - Y_0) - (C_1 - C_0) \tag{5.16}$$

两种写法等价：左边是「政策后组间差」减「政策前组间差」，右边是「处理组前后变化」减「对照组前后变化」。**DID** 的精髓，是用对照组的同期变化，替处理组扣掉它本来也会经历的共同趋势：处理组若没有政策，本会从 Y_0 沿对照组的趋势走到 $Y_0 + (C_1 - C_0)$；实际到了 Y_1，二者之差正是政策效应。这里的三个虚拟变量——组别 $Treat_i$、时间 $Post_t$、以及二者的交乘 $Treat_i \times Post_t$——就是大纲说的「DID 中的组别、时间和处理交乘项」。

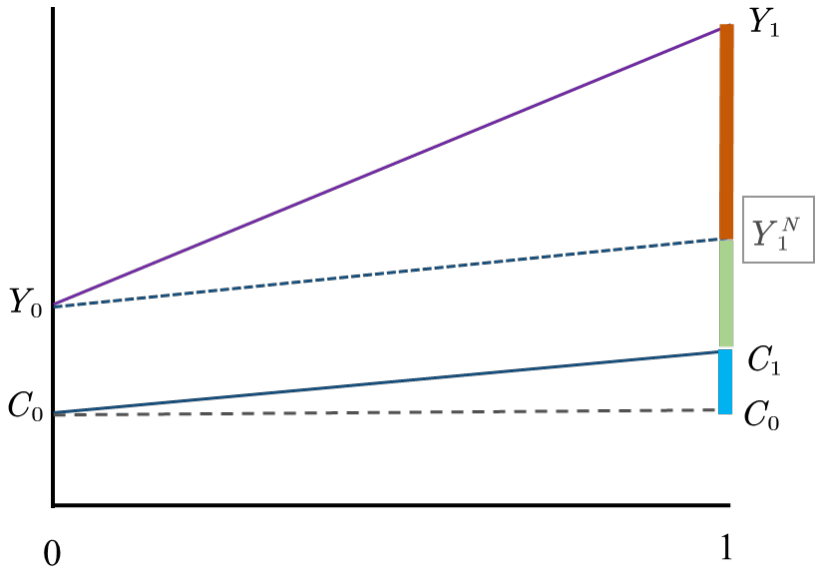

图 5.4: 2×2 DID 的四个均值与处理组反事实：对照组斜率给出共同趋势，处理组偏离这条趋势的部分就是政策效应

一个经典例子是 Card and Krueger (1994)：1992 年新泽西上调最低工资、相邻的宾夕法尼亚没有，$Treat_i$ 是「是否新泽西」、$Post_t$ 是「是否调薪后」，交乘项系数就是最低工资对就业的 DID 估计。处理组、对照组、政策前后、交乘项含义全都一目了然，没有一处要猜「在跟谁比」。

5.5.3 从两期到多期：事件研究与平行趋势

真实的政策评估几乎总是多期面板：政策前有若干期、政策后也有若干期。把 式 5.15 沿「相对政策时间」展开，就得到多期 DID(事件研究) 设定。设政策在 t^* 期实施，令 $k = t - t^*$ 为相对时间：

$$Y_{it} = \alpha_i + \lambda_t + \sum_{k \neq -1} \delta_k (Treat_i \times \mathbf{1}\{t - t^* = k\}) + \varepsilon_{it} \tag{5.17}$$

以政策前一期 $k = -1$ 作基准 (系数归一化为 0)。两组系数含义要分开看：

- 事前系数 $\delta_k\,(k < -1)$：政策来临之前处理组相对对照组的差异。若平行趋势成立，这些系数应当不显著——政策还没来，两组不该有系统性差异。
- 事后系数 $\delta_k\,(k \geq 0)$：政策实施后第 k 期的处理效应，连起来就是政策效应的动态路径。

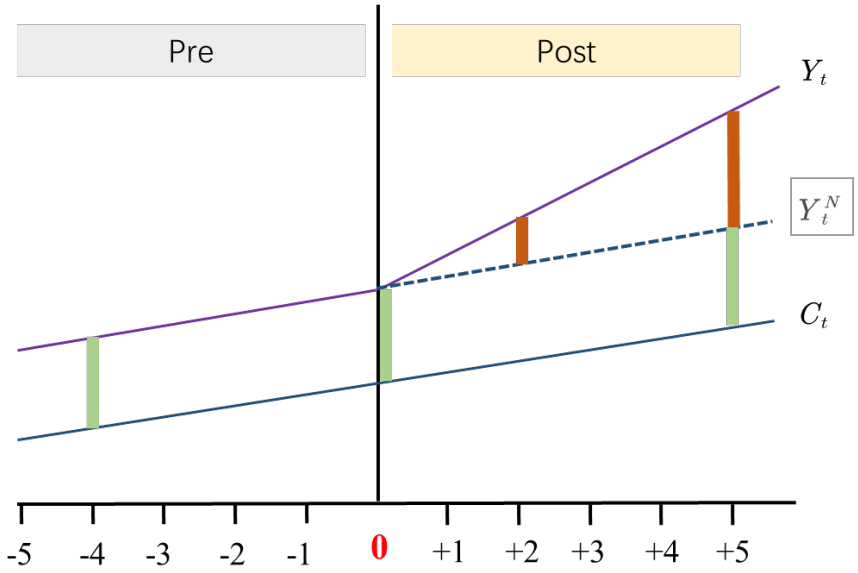

图 5.5: 标准的多期 DID：事前若干期用于判断平行趋势，事后若干期用于估计动态效应

⚠ 常见问题：对照组怎么选、平行趋势怎么看

DID 能不能站得住，全押在平行趋势假设上：如果没有政策，处理组和对照组的结果变量本会沿同一条趋势走。两件事最容易出问题——

- 对照组的选择：对照组要能代表处理组「假如没有政策会怎样」。若处理组和对照组本就走在不同的趋势上 (比如处理组是高增长地区、对照组是停滞地区)，DID 会把趋势差误当成政策效应。
- 平行趋势的检验：主要靠事件研究图看事前系数是否接近零、是否平坦。要强调的是，事前系数不显著只是一个直观的、不严谨的检验——它是必要条件而非充分条件，通不过基本可以否掉平行趋势，通过了也不能算证明。

至于分批推进的政策 (不同个体在不同年份进入处理) 会给双向固定效应带来的麻烦——TWFE 的系数会变成一堆 2×2 比较的加权平均、甚至混入负权重——以及 CSDID 等现代 DID 估计量如何修正，属于高级班内容，本讲不展开。有兴趣的读者可以参考 双重差分：从 TWFE 到 CSDID。

5.6 如何理解带固定效应和交乘项的系数

走到这里，可以把一条主线收拢成一句话：无论是一元回归的斜率、虚拟变量的截距差、固定效应模型里的 β，还是 **DID** 的交乘项，它们全都是「在控制了别的东西之后，拿谁跟谁比」。区别只在于「控制了什么」「比较对象是谁」：

- 一元回归的 β：相对样本均值，X 高一单位对应 Y 的差；
- 虚拟变量系数：相对基准组的截距差；
- 个体固定效应里的 β：同一个体内部，随时间变化时 Y 怎么变；
- 双向固定效应里的 β：相对同年其他个体，该个体的变化带来什么；
- DID 交乘项：处理组前后变化，再扣掉对照组同期变化。

读懂一个复杂回归式的系数，可以用一张三列表把它拆开——每个变量 (含交乘项) 占一行，逐列写清「变量含义 / 比较对象 / 系数解释」。比如面对 `reg y i.treat##i.post` 里的 `1.treat#1.post`，三列分别是：「处理组在政策后的额外项」/「处理组前后变化 vs 对照组前后变化」/「双重差分的政策效应」。养成这个习惯，再复杂的固定效应加交乘项模型，也不会看着系数发懵。下一节的 AI 协作，就把这张三列表交给 AI 来帮你打草稿，你负责核对。

5.7 代码实操

本节用四段可复现的 Stata 代码，把前面的直觉落到命令上。完整脚本见伴生文件 `examples/ch05/ch05_main.do`(`**#` 大纲与本讲各节标题一一对应，可在 VS Code 中按节导航运行)；真实财务数据 `xtcs.dta`(中国上市公司资本结构面板) 随本讲提供。代码为静态展示、输出为本机运行后冻结，不在渲染时执行。

其一，辛普森悖论与组内去心：真实斜率 $\beta = 0.5$，但三家公司的个体效应与初始规模负相关。

```
*----------------- 模拟：辛普森悖论
clear
set seed 20260706
set obs 3
gen id    = _n
gen a_i   = 12 - 3*id            // 个体效应 9,6,3，与规模负相关
gen size0 = 2  + 3*id            // 初创规模 5,8,11
expand 100
gen x = size0 + runiform(0, 4)          // Size
gen y = a_i + 0.5*x + rnormal(0, 0.8)   // DGP: beta = 0.5

reg  y x                 // 混合回归：斜率为负
areg y x, absorb(id)     // 个体固定效应：斜率回到 0.5 附近
xtset id
xtreg y x, be            // 对照：组间关系，就是那条负斜率的来源
```

```
混合回归      reg y x              :  x = -0.305   被公司间差异污染，符号翻负
个体固定效应  areg y x, absorb(id):  x =  0.562   只用公司内部变异，还原真值 0.5
组间关系      xtreg y x, be        :  x = -0.500   公司之间的关系，正是负斜率来源
```

其二，固定效应命令族的等价与时不变变量被 **omit**：

```
webuse nlswork, clear
xtset idcode year
gen byte black = race == 2                          // 种族虚拟变量（时不变）

xtreg ln_wage tenure hours, fe                      // 个体 FE
xtreg ln_wage tenure hours i.year, fe               // 双向 FE（加时间效应）
reghdfe ln_wage tenure hours, absorb(idcode year) ///
    vce(cluster idcode)                              // 等价的双向 FE

xtreg ln_wage black tenure hours, fe                // 试图加入时不变变量 black
```

```
       black |      0.000  (omitted)      ← 时不变变量与个体效应共线，被自动剔除
```

其三，**2×2 DID** 还原政策效应：真实政策效应设为 $+3$，看交乘项能否还原、三个系数是否各归其位。

```
*----------------- 模拟：2×2 DID
clear
set seed 20260708
set obs 200
gen id    = _n
gen treat = id > 100                        // 后 100 个为处理组
expand 2
bysort id: gen post = _n - 1                // 0= 政策前，1= 政策后
* DGP: 基线 10 + 组间差 2 + 时间趋势 4 + 政策效应 3
gen y = 10 + 2*treat + 4*post + 3*(treat*post) + rnormal(0,1)

table post treat, stat(mean y) nformat(%6.2f)   // 四格均值
reg y i.treat##i.post, vce(robust)              // 交乘项应还原 3
```

```
四格均值       政策前    政策后
对照组         10.06     13.82
处理组         12.02     19.03

1.treat         1.959    组间水平差  1（处理组本就高约 2）
1.post          3.761    对照组时间趋势  2（ 4）
1.treat#1.post  3.251    双重差分   ← 政策效应（真值 3）
```

其四，真实财务数据里「换比较对象、换答案」：用中国上市公司资本结构面板 `xtcs.dta`，看规模 `size` 对资产负债率 `tl` 的系数，如何随固定效应一步步递进而变化。

```
use "$D/xtcs.dta", clear                                    // code= 公司, year= 年份
xtset code year
reg     tl size ndts tang tobin npr                         // 混合回归 POLS
xtreg   tl size ndts tang tobin npr, fe cluster(code)       // 个体 FE
xtreg   tl size ndts tang tobin npr i.year, fe cluster(code) // 双向 FE
```

```
   size 的系数:   混合 POLS  0.041  →  个体 FE  0.120  →  双向 FE  0.134
```

三个数字不是「越估越准」，而是在回答三个不同的问题：POLS 的 0.041 混着「大公司 vs 小公司」的横截面差异；个体 FE 的 0.120 只问「同一家公司规模变大以后，负债率怎么变」；再加年份效应控制掉共同的宏观波动，双向 FE 给到 0.134。系数一路走高，正是本讲那张比较对象递进表在真实数据上的印证。

5.8 AI 协作

> **契约边界**
>
> 本节逐条覆盖归属本讲的四项「AI 协作训练」，不删项、不缩范围。每项给一份可复制的提示词、标注对应的 core skill，并说明怎么人工核对。这些任务，网页版 AI 助手大多都能完成。

带固定效应和交乘项的回归式，是最容易「看着系数发懵」的地方。让 AI 陪你做的，不是替你下结论，而是把系数逐个翻译成「拿谁跟谁比」，再由你回到数据核对。四项训练共用同一个 skill：`core/05-regression-interpreter`。

5.8.1 训练 1：让 AI 解释含 FWL、虚拟变量和交乘项的回归式

任务：把一条回归命令 (或方程) 交给 AI，让它逐项说清每个系数的含义、比较对象和量纲。

> **提示词**
>
> 这是一条 Stata 回归命令 (或回归方程)：`reg wage i.race c.hours i.race#c.hours`。请面向计量基础较弱的读者，逐项解释：(1) 每个系数 (含常数项、虚拟变量、连续变量、交乘项) 度量的是什么；(2) 它的比较对象是谁、基准组是谁；(3) 单位怎么读。特别说明交乘项的存在如何改变主变量系数的解释。只依据这条式子本身，不要编造数据里没有的信息；不确定处标出来让我核对。

对应 **skill**：`core/05-regression-interpreter`。人工核对：重点查 AI 有没有说错基准组、有没有把交乘项存在时的主效应仍当成「总效应」来读。

5.8.2 训练 2：让 AI 检查 DID 型回归中每个变量和交乘项的含义

任务：把一条 DID 回归交给 AI，让它对照「组别 / 时间 / 处理交乘项」逐一核对每个变量的角色。

> **提示词**
>
> 下面是一条双重差分 (DID) 回归：`reg y i.treat##i.post, vce(robust)`(`treat` 为处理组虚拟变量，`post` 为政策后虚拟变量)。请逐项说明：`1.treat`、`1.post`、`1.treat#1.post` 各自度量什么，对应 2×2 四格均值里的哪个差；哪个才是政策效应；并用「处理组前后变化减对照组前后变化」的话把交乘项复述一遍。指出这条回归识别政策效应所依赖的关键假设。只依据这条式子，不要脑补。

对应 **skill**：`core/05-regression-interpreter`。人工核对：确认 AI 没把 `1.treat`(组间水平差) 或 `1.post`(时间趋势) 误当成政策效应；核对它说的关键假设是不是平行趋势。

5.8.3 训练 3：让 AI 检查固定效应模型中的系数是否可以被识别

任务：把模型设定和变量的「随谁变」信息交给 AI，让它判断哪些系数在给定固定效应下还能识别、哪些会被吸收。

> **提示词**
>
> 我要估计一个固定效应模型：`reghdfe y x1 x2 gender region, absorb(id year)`。已知：`x1`、`x2` 随个体和时间都变；`gender` 不随时间变；`region` 在样本期内不随时间变。请判断：加入个体固定效应 (`id`) 后，哪些变量的系数还能被识别、哪些会因为与个体效应共线而被 `omitted`，并解释原因。如果我确实想估计 `gender` 的效应，有哪些替代做法？只做可识别性判断，不要替我下实证结论。

对应 **skill**：`core/05-regression-interpreter`。人工核对：AI 的判断要回到「是否随时间变」这一条硬标准去验；对它给的替代方法 (组间估计、Hausman–Taylor 等) 只当线索，适用与否自己判断。

5.8.4 训练 4：让 AI 把复杂回归式改写成「变量含义 + 比较对象 + 系数解释」三列表

任务：把一条带多个虚拟变量、固定效应和交乘项的回归式交给 AI，让它输出一张三列表。

> **提示词**
>
> 请把下面这条回归式改写成一张三列表格，每个变量 (含交乘项、常数项) 占一行，三列分别是「变量含义」「比较对象 / 基准组」「系数解释 (一句话)」：

`reghdfe y treat_post x1 x2, absorb(id year)`，其中 `treat_post = treat × post`。

表格要让一个没读过这篇分析的人，也能照着读懂每个系数在跟谁比。不确定的地方留空并标注，让我补。

对应 **skill**：`core/05-regression-interpreter`。人工核对：三列表是脚手架，每一行的「比较对象」要你对照模型的固定效应结构逐条确认，AI 填错基准组是最常见的错。

5.9 Python 附录

同样的 FWL、固定效应与 DID，用 Python 也能一行行复现。载体是伴生文件 `examples/ch05/ch05_python.ipynb`，内容与本讲代码实操一一对应：用 `linearmodels(PanelOLS)` 或 `pyfixest(feols)` 估计个体、时间与双向固定效应，用 `statsmodels` 跑 2×2 DID 的交乘项回归；每段配「任务说明 + 提示词」，代码由 AI 生成、输出本机运行后冻结。

想让 AI 帮你把某段 Stata 翻成等价 Python，可以这样提示：

> **提示词**
>
> 下面这段 Stata 代码用双向固定效应估计 `y` 对 `x` 的关系：`reghdfe y x, absorb(id year) cluster(id)`。请用 Python 的 `linearmodels.PanelOLS`（或 `pyfixest`）写出等价实现，包含设定面板索引、加入个体与时间固定效应、按 `id` 聚类稳健标准误；并逐行加中文注释说明它和 Stata 命令的对应关系。只写等价实现，不要改变模型设定。

5.10 小结与练习

本讲用一条主线把三样东西串了起来：FWL 定理说「系数 = 剔除其他变量之后的残差回归」，虚拟变量说「分类变量 = 变截距」，固定效应说「给每个个体一条截距 = 组内去心」，而 DID 不过是「组别 × 时间 × 处理」三个虚拟变量组合出的两次差分。它们看似是四个话题，底层却是同一件事——每一次控制，都是在换一次比较对象。

一句话带走：回归系数从来不回答「影响有多大」，它只回答「在控制了别的东西之后，拿谁跟谁比」；读懂了比较对象，就读懂了系数。

5.10.1 练习

练习 1(概念 • FWL 与残差化)

有人说「多元回归里 x_1 的系数，就是把 x_1 单独拿出来对 y 回归得到的斜率」。请用 FWL 定理指出这句话错在哪，并说明正确的说法是什么。

参考答案

错在漏掉了「剔除其他变量」这一步。单独用 x_1 回归 y，得到的斜率会被 x_1 与其他变量的相关性污染 (遗漏变量偏误)。FWL 的正确说法是：x_1 的多元回归系数，等于「x_1 剔除掉其他变量后的残差 $\tilde{x}_1$」对「y 剔除掉其他变量后的残差 $\tilde{y}$」回归的斜率——是残差对残差，不是原始变量对原始变量。

练习 2(概念 • 固定效应可识别性)

一位研究者用个体固定效应模型 `xtreg wage tenure female, fe` 估计工作年限和性别对工资的影响，发现 `female` 的系数是 `(omitted)`。请解释原因，并给出若他确实想估计性别效应可以怎么办。

参考答案

性别在样本期内不随时间变化，与不随时间变化的个体效应 α_i 完全共线，组内去心把 α_i 减掉的同时也把 `female` 减成了 0，所以被 `omitted`。固定效应能控制个体的一切时不变特征，却无法估计这些特征各自的系数。若要估性别效应，可改用加控制变量的混合回归 / 组间估计，或 Hausman–Taylor(`xthtaylor`)、`xtseqreg` 等能保留时不变变量的方法。

练习 3(概念 • DID 系数身份)

在 2×2 DID 回归 `reg y i.treat##i.post` 中，`1.treat`、`1.post`、`1.treat#1.post` 三个系数分别是什么含义？哪一个才是政策效应？为什么不能直接用「处理组政策后均值减对照组政策后均值」来估政策效应？

参考答案

`1.treat = Y0-C0` 是政策前两组的水平差 (两组本来就不同)；`1.post = C1-C0` 是对照组的时间趋势；`1.treat#1.post = (Y1-Y0)-(C1-C0)` 是双重差分，才是政策效应。直接用「处理组政策后减对照组政策后」(即 $Y_1 - C_1$) 会混入政策前就存在的组间水平差 $Y_0 - C_0$；DID 的关键正是用对照组的同期变化，替处理组扣掉它本来也会经历的共同趋势。

练习 4(交给 AI 并审计)

任取一条你研究领域里带固定效应和交乘项的回归式，用本讲「训练 4」的提示词让 AI 生成一张三列表；然后回到模型设定，找出 AI 至少一处填错或含糊的地方 (最常见的是基准组)，说明你是怎么发现的。

参考答案 (要点)

本题无标准答案，重点在「审计」。合格的回答应能指出 AI 的具体错处 (如把交乘项存在时的主效应当成总效应、写错基准组、把被固定效应吸收的变量仍列为可识别)，并说明核对依据 (模型的固定效应结构、因子变量的基准组设定、变量是否随时间变化)。能稳定发现并纠正这类偏差，正是本讲要练的能力。

5.11 延伸阅读

- 固定效应的系统讲法：连享会《因果推断》讲义中讨论固定效应「控制了什么」的一章，把组内去心、双向固定效应、高维固定效应到交互固定效应连成一条线，适合想更进一步的同学。
- 双重差分的现代视角：连享会《因果推断》讲义中从 TWFE 讲到 CSDID 的一章，讲清了分批推进政策为什么会让 TWFE 出问题。
- **FWL** 与部分回归图：连享会推文图示线性回归系数：Frisch-Waugh 定理与部分回归图、多元回归系数：我们都解释错了？。
- 入门底本：Wooldridge《计量经济学导论》中虚拟变量、面板固定效应与政策分析 (DID) 的相关章节，统一见阅读清单。
- 方法进阶 **(**现代 **DID** 估计量与机器学习方法**)**：CSDID 等现代 DID 估计量、DDML 与机器学习方法留待高级班，可先浏览连享会关于现代因果推断方法的整理。

6 综合实操：从复现代码到自己的分析任务

本讲要点

- 如何打开和理解一个复现项目；
- 如何根据研究问题列出数据处理和模型实现清单；
- 如何检查样本筛选、变量构造和图形诊断结果；
- 如何解释 OLS、虚拟变量、交乘项和固定效应模型；
- 如何整理复现日志、结果说明和待检查问题清单；
- 如何借助 AI 生成代码注释、复现路线图和 debug 记录；
- 如何判断 AI 输出是否可靠，哪些问题必须回到数据和模型本身。

本讲的形式与配套

- 本讲是一场现场演示，不是又一章讲义。上课时我们会临时选一篇 2026 年新发表的实证论文，连同它公开的数据与代码，用 AI Agent 当场走一遍「打开复现包 → 跑出结果 → 迁移到自己的问题」的全过程。你在这一页读到的，是这场演示的议程与提示词清单；具体每一步现场展开。
- 本讲用到的 **skills**：`core/03-replication-navigator` 与 `core/06-repro-logger` 为主，并把前五讲的 `core/01`–`core/08` 串成一条完整工作流。skill 是什么、怎么在不同 agent 里触发，见《AI 工作流与 skills 上手》。
- 本讲以工作流演示为主、不重自研代码，比照第一讲免写 Python 附录，改以「提示词 + skills 调用」承载。现场用到的分步提示词、裁剪版主控脚本 **(Stata / R)**、六幕演示讲稿已整理在配套 `examples/ch06/`；其中 `examples/ch06/demo_results/` 收录了课前跑通、可逐表对照的「标准答案」(Table 1–7、Figure 1–4 与复现报告)，你照着自己复现一遍后，就用它核对结果对不对。

6.1 本讲是一场现场演示

前面五讲，我们从复现一篇顶刊起步，一路练了读数据、清数据、跑回归、加虚拟变量与交乘项、上固定效应。到这一讲，是时候把这些串成一条完整的手艺了。

课程的前言里说过，这门课的定位是引进门——把你从「不敢开始」的门口领进来。这最后一讲，就用一种最贴近真实科研的方式来收尾：当场打开一个真实的复现项目，用 **AI Agent** 陪你走完全程。我们把它叫做「开彩盒」——现场把一个复现项目从头拆到尾，看清里面的数据、代码和结论是怎么一步步立起来的。

之所以这样安排，有两个理由。其一，真实的研究从来不是照着讲义按部就班，而是面对一个陌生项目、一边摸索一边判断；把这一讲留成开放式，正是让你看清这种真实状态下人和

AI 怎么配合。其二，方法不再新增——我们不引入前五讲之外的任何新计量内容，只把已经学过的东西，放到一篇新论文上重走一遍，检验你是否真的能迁移。

> **本讲的现场演示案例**
>
> 本讲现场演示的复现项目是一份双重差分 **(DID)** 实践指南，作者正是当代 DID 方法的几位主要提出者：
>
> > Baker, A., Callaway, B., Cunningham, S., Goodman-Bacon, A., & Sant'Anna, P. H. C. (2026). Difference-in-Differences Designs: A Practitioner's Guide. *Journal of Economic Literature*, 64(2), 498–557. DOI。复现包 (公开、免登录)：github.com/pedrohcgs/JEL-DiD。
>
> 选它做现场案例，是因为它把整门课的方法脊柱用一个数据集 (美国各州医保扩张对县级死亡率的影响) 串成一条线：2×2 DID (本质就是两个虚拟变量的交叉项) → 加控制变量的回归 → 双向固定效应 → 事件研究 (处理组虚拟变量 × 年份虚拟变量的交叉项)。现场只演示这条初级班范围内的主线；复现包里更进阶的多期错峰估计量与敏感性分析，属于现代 DID 的内容，留给高级班。
> 复现包只给下载链接、不整包并入本仓库 (规则 11)；学员按上面的 GitHub 链接自行下载即可。

需要说明的是，「现场」不等于「即兴」。演示用的每一段提示词、每一步流程，老师都会预先在那篇论文的真实数据上跑通、并留好录屏兜底——就像一场排练过的公开课，临场发挥的只是讲解，实时产出的说服力一分不减。你要学的，正是这套可以照着复用的流程。

6.2 现场议程：开彩盒六步

这场演示按六步走。这六步不是随意排的，而是任何一次「从复现代码到自己的分析」都要经过的判断链条。每一步，都请你留意研究者要在这一步做出什么判断，而不只是 AI 跑了什么命令。

1. 开箱：先读懂这个项目 (对应要点：打开和理解一个复现项目)。拿到复现包，先不急着跑。看文件夹结构、读 `README`、识别数据分几层 (原始 / 中间 / 分析样本 / 图表结果)。要判断的是：入口脚本在哪、能复现到什么程度、有没有缺的数据。这一步用的正是第一讲的看法。
2. 列清单：把论文拆成可执行的步骤 (对应要点：根据研究问题列出数据处理和模型实现清单)。顺着论文的研究问题，把「要做哪些数据处理」和「要跑哪些模型」列成两张清单。要判断的是：哪些步骤是核心、哪些只是稳健性，先做什么后做什么。
3. 体检：检查数据处理对不对 (对应要点：检查样本筛选、变量构造和图形诊断结果)。照着第二、三讲练的，核对样本怎么筛的、关键变量怎么构造的、图形诊断有没有异常。要判断的是：样本量的变化说得清吗、变量口径一致吗、有没有被离群值或缺失带偏。
4. 读模型：解释跑出来的结果 (对应要点：解释 OLS、虚拟变量、交乘项和固定效应模型)。把复现包里的回归读懂——它是 OLS 还是加了固定效应，虚拟变量和交乘项代表

什么，系数怎么解释。要判断的是：这个模型回答的是不是论文要问的问题。这一步接的是第四、五讲。

5. 留痕：整理成可复现的记录 (对应要点：整理复现日志、结果说明和待检查问题清单)。把前面几步汇成三样东西：一份复现日志、一份结果说明、一份还没搞清的「待检查问题清单」。要判断的是：换个人照着你的记录，能不能跑出同样的结果。
6. 迁移与收口：判断 **AI** 是否可靠 (对应要点：判断 AI 输出是否可靠，哪些问题必须回到数据和模型本身)。最后，讨论怎么把这套流程迁移到你自己的问题上，并回答一个贯穿全程的问题：AI 给的东西，哪些能信、哪些必须回到数据和模型本身去核。这一步单独在后面展开。

贯穿这六步的，还有一条暗线：每一步都让 **AI** 帮我们生成代码注释、复现路线图和 **debug** 记录 (对应要点：借助 AI 生成代码注释、复现路线图和 debug 记录)。它不是第七步，而是渗透在前六步里的助手——下一节就把这些 AI 动作逐一摆出来。

6.3 现场会调用的 AI 动作

上面六步里，AI 具体帮我们做五件事。下面逐条给出任务、可复制的提示词、对应的 **skill**，以及怎么人工核对。现场演示时，这些提示词会一段段贴给 agent，看着它实时产出；课后你照着同一份清单，就能在自己选的论文上「再开一次彩盒」。

> **!** 契约边界
>
> 本节逐条覆盖大纲「AI 协作训练」清单，不删项。每条都配提示词与 core skill，并说明人工核对怎么做——因为 AI 负责起草，判断对错始终是你的事。

6.3.1 动作 1：让 AI 根据项目文件夹和 README 生成复现路线图

任务：把复现包的 `README` 和目录清单交给 AI，让它给出一张「先跑什么、再跑什么」的路线图。

> **i** 提示词
>
> 这是一篇论文复现包的 `README` 和文件目录清单 (附)。请生成一张复现路线图：按实际运行顺序，从入口脚本一直列到最终图表，每步标明运行哪个脚本、读什么数据、产出什么；再说明数据分几层、能复现到什么程度。只依据我给的内容，不要编造文件名，不确定的地方标出来让我核对。

对应 **skill**：`core/03-replication-navigator`。人工核对：对照 `README` 的脚本清单，检查有没有漏掉或杜撰脚本；用「被反复调用的是子脚本、调用别人的是主程序」这条法则抽查调用关系。

6.3.2 动作 2：让 AI 根据变量字典和筛选规则生成数据检查清单

任务：把论文的变量字典 (codebook) 和样本筛选规则交给 AI，让它列一张「该检查哪些地方」的清单。

> **提示词**
>
> 这是论文的变量说明和样本筛选规则 (附)。请帮我列一份数据检查清单：每个关键变量该怎么核对口径、样本每一步筛选后应该剩多少、哪些地方容易出缺失或离群、合并数据后要查什么。按「检查项 → 怎么查 → 期望结果」组织。

对应 **skill**：`core/04-data-cleaning-planner`。人工核对：清单只是提醒，真正的核对要回到数据本身跑一遍 (`describe`、`misstable`、匹配率)；AI 说「应该剩多少」的数字，务必和实际样本量对上。

6.3.3 动作 3：让 AI 解释一段回归代码和输出表格

任务：挑复现包里的一段回归代码 (Stata、R 或 Python 都行) 和它的结果表，让 AI 讲清它在做什么、系数怎么读。这一项顺便兑现大纲「R 作为补充」的要求——我们不学 **R** 语法，只借 **AI** 读懂别的语言写的代码在流程中的角色。

> **提示词**
>
> 这是复现包里的一段回归代码和它跑出的结果表 (附，语言可能是 Stata / R / Python)。请说明：(a) 它估计的是什么模型，有没有加固定效应、虚拟变量或交乘项；(b) 核心系数怎么解释、对应论文的哪张表；(c) 它读入什么数据、处在复现流程的哪一步。不用教我语法，只讲它在做什么、结果怎么读。

对应 **skill**：`core/05-regression-interpreter`。人工核对：把 AI 的解释和论文正文对应的那张表比对，看系数符号、显著性、变量含义说得对不对；遇到别的语言写的代码，装好环境让 agent 直接执行、再验证输出，而不是凭空判断。

6.3.4 动作 4：让 AI 生成复现日志初稿，由你检查错漏

任务：跑完一部分复现后，让 AI 把「从哪份数据、经过哪些步骤、得到哪个结果」整理成日志初稿，你负责查错漏。

> **提示词**
>
> 以下是我跑过的脚本、用到的数据和得到的产出 (附)。请整理成一份「从数据到结果」的复现日志：按步骤记录输入数据 → 处理脚本 → 产出，标出哪些步骤因为缺数据或缺许可没跑成。只依据我提供的信息，不要补全我没给的内容。

对应 **skill**：`core/06-repro-logger`。人工核对：逐步对照你实际跑的，看输入输出对不对、跑不了的步骤有没有如实标注。一份诚实的日志，比一份漂亮但注水的更有价值——这正是人机分工里「人」不能让渡的部分。

6.3.5 动作 5：让 AI 把这次流程整理成可复用的项目模板

任务：演示到最后，让 AI 把这一整趟「开彩盒」的流程，抽象成一份下次可以直接套用的项目模板。

> **提示词**
>
> 请把我们刚才这次复现的完整流程，整理成一份可复用的项目模板：包括标准的文件夹结构、每一步该做什么、每步建议调用哪个 skill、以及每步要人工核对的要点。目标是我下次拿到一篇新论文，照着这份模板就能自己走一遍。

对应 **skill**：`core/06-repro-logger`。人工核对：模板要经得起复用——自己拿另一篇论文套一次，看哪些步骤缺了、哪些提示词要改，再回填进模板。能沉淀出自己的模板，才算真的把这套流程学会了。

6.4 判断 AI 输出是否可靠

六步走完，剩下最要紧的一课：**AI** 给的东西，什么时候能信，什么时候必须回到数据和模型本身去核。这是本讲、也是全课的落点。

一个朴素的分界是：**AI** 擅长「组织与转述」，不擅长「判断与担责」。它能很快把 README 整理成路线图、把一段代码翻成另一种语言、把流程写成日志——这些是组织性工作，出错了你也容易核对。但涉及研究判断的地方，它给的只是一个看似合理的说法，未必对：

- 样本为什么从一万筛到八千，这个口径合不合理——要回到筛选代码和论文的样本定义去看，不能听 AI 一句「合理」；
- 这个交乘项的系数到底说明了什么、能不能解释成因果——要回到模型设定和识别假设，这是第三到五讲反复强调的；
- AI 报的某个数字、某篇文献、某个脚本名——尤其要警惕它一本正经地编造，凡关键处都回原始材料核对。

这里还要接上第二讲埋下的一条红线：**AI** 有可能为了「让结果好看」，在清洗或设定上悄悄做手脚。所以对它产出的每一步，都要保留一分审计的警惕心。

> **一句话带走**
>
> AI 让你更快地读懂一个陌生项目，但读得对不对、能不能用在你的问题上，永远由你判断。哪些能交给 AI、哪些必须回到数据和模型本身——分清这条线，就是这门课想教给你的核心手艺。

6.5 课后练习

本讲只有一道练习，也是这门课留给你的第一个真正独立的任务。

练习：自己开一次彩盒

课后自选一篇方法简单、复现包公开的实证论文 (经管、能源、环境等均可，方法落在 OLS / 虚拟变量 / 固定效应 / 标准 DID 范围内)，用本讲的六步议程和五个 **AI** 动作，自己完整走一遍：从打开复现包，到跑出至少一个主结果，最后产出一份诚实的复现日志 (含一张「待检查问题清单」)。
重点不在于跑得多完整，而在于你能不能在每一步认出研究者要做的判断，并对 **AI** 的输出保持核对。
如果你一时找不到合适的论文，下面三篇复现包公开、方法都在本课范围内，难度递进，可任选其一 (前两篇在 openICPSR，注册免费账户即可下载；第三篇在 openICPSR、代码为纯 R，适合练一次「换语言复现」)：

- 入门 • 纯 **Stata**：Bjorvatn, K., Ferris, D., Gulesci, S., Nasgowitz, A., Somville, V., & Vandewalle, L. (2025). Childcare, labor supply, and business development: Experimental evidence from Uganda. *AEJ: Applied Economics*, 17(2), 75–101. DOI。复现包：openICPSR E197964。方法：OLS + 处理组虚拟变量；项目结构最干净 (`0_master.do` + 若干小 `.dta`)。
- 进阶 • 能源政策：Bailey, M. R., Brown, D. P., Shaffer, B., & Wolak, F. A. (2025). Show me the money! A field experiment on electric vehicle charge timing. *AEJ: Economic Policy*, 17(2), 259–284. DOI。复现包：openICPSR E202941。方法：固定效应 + 二重/三重交乘项 + 简单 DID；主题贴近能源与政策干预。
- 练语言 • 纯 **R**：Ajzenman, N., Ferman, B., & Sant'Anna, P. C. (2025). Discrimination in the formation of academic networks: A field experiment on #EconTwitter. *AER: Insights*, 7(3), 357–375. DOI。复现包：openICPSR E210084。方法：OLS/线性概率模型 + 虚拟变量 + 交叉项 + 多组固定效应；全 R 项目 `master.R`。

参考答案：一份自评检查清单

本题没有标准答案——你选的论文不同，结果自然不同。合格与否，用下面这张清单自评：

- 开箱：说得清入口脚本、数据分层、能复现到什么程度了吗？
- 清单：数据处理和模型实现，各列出了可执行的步骤吗？
- 体检：核对过样本筛选后的样本量、关键变量口径、图形诊断吗？
- 读模型：能解释核心回归是什么模型、系数怎么读吗？
- 留痕：复现日志能让别人照着重跑吗？跑不了的步骤如实标了吗？
- 判断：至少找出一处 AI 输出的错误或存疑，并说明你怎么核对出来的吗？

六项都能答「是」，你就已经具备了从复现代码走向自己分析任务的基本功。

6.6 延伸阅读：课后的修行

这门课到这里就结束了，但正如前言所说，它只是引进门——真正的手艺，靠的是课后一篇篇论文地临摹、一个个项目地做。给你几个继续修行的入口：

- 把六步用起来：找三五篇你研究领域的论文，用本讲的流程各开一次彩盒。做得多了，你会形成自己的一套模板和提示词库，这比记住任何单条命令都值钱。
- 方法进阶留待高级班：本课只教到 OLS、虚拟变量、固定效应这些基础方法。更前沿的因果推断方法 (现代 DID 估计量、双重机器学习等) 怎么做，可先参考连享会关于现代 DID 的整理，并留待高级班与论文班系统学习。
- 让 **AI** 工作流长在日常里：把 `core/01–core/08` 这套 skills 装进你自己的 agent，读论文、清数据、跑回归、写日志时随手调用。工具会一直更新，但「把任务说清楚、调对 skill、再自己核对」这套工作方式，是通用的。