Quantitative Methods in International Business with Python

MATH60033A · HEC Montréal

Thierry Warin, PhD

6 November 2026

Table of contents

Foreword **1**
How to use it . . . 2
The slides . . . 3
The twelve chapters . . . 3

1 Introduction **5**
1.1 The question underneath every method . . . 5
1.2 Three questions that look alike . . . 5
1.3 Why international business is a hard case . . . 6
1.4 What a model is, and what it is not . . . 7
1.5 How the book is organised . . . 8
1.6 On the data, and on mess . . . 8
1.7 What this book asks of you . . . 9

2 Exploratory Data Analysis, and the First Model **11**
2.1 Looking, before deciding what to look for . . . 11
2.2 Look at it first, literally . . . 14
2.3 Centre: three answers, three questions . . . 16
2.4 Spread, and the $n - 1$ that everyone asks about . . . 17
2.5 Shape, and a rule that passes when it should not . . . 19
2.6 Transformation, and what it costs . . . 20
2.7 What is missing, and why . . . 21
2.8 The first model, as a right-angled triangle . . . 23
2.9 More than one predictor, and what "controlling for" does . . . 25
2.10 Say the slope in units . . . 27
2.11 What a summary cannot show . . . 27
2.12 A short review of the literature . . . 28
2.13 What to report . . . 28

3 Regression: Adequacy, Validity, and Robustness **31**
3.1 The line, and where it came from . . . 31
3.2 Adequacy, validity, robustness . . . 38
3.3 What the residual knows . . . 41
3.4 Goodness of fit, and what it cannot tell you . . . 43
3.5 Leverage: which observations *can* move the line . . . 45
3.6 Influence: separating the outlier from the point that matters . . . 46
3.7 Normality, and why it matters least . . . 49
3.8 When the variance is not constant . . . 49
3.9 The assumption nobody plots . . . 52
3.10 Comparing models: what AIC is, and what it is not . . . 53
3.11 When a diagnostic fails, what to actually do . . . 55

3.12 A short review of the literature . . . 56
3.13 What to report . . . 57

4 Logistic Regression: Binary, Ordinal, and Multinomial 59
4.1 When the outcome is a category . . . 59
4.2 The logistic function, and where it came from . . . 61
4.3 Estimation: no formula, and why that is fine . . . 63
4.4 Reading the coefficients . . . 64
4.5 Comparing models: the likelihood-ratio test . . . 66
4.6 Does the model discriminate, and is it calibrated? . . . 66
4.7 When the outcome is ordered . . . 68
4.8 When the categories have no order . . . 69
4.9 The failure mode worth knowing . . . 70
4.10 A short review of the literature . . . 71
4.11 What to report . . . 72

5 Regularisation: Ridge, Lasso, and the Elastic Net 73
5.1 The mechanical first attempt . . . 73
5.2 Why a biased estimator can win . . . 75
5.3 Ridge: shrink everything, targeted . . . 76
5.4 Lasso: the corner that produces a zero . . . 77
5.5 Standardise, always . . . 80
5.6 The elastic net, and correlated predictors . . . 80
5.7 Choosing λ, and what the choice costs . . . 81
5.8 Scoring the methods against a known truth . . . 82
5.9 What none of this licenses . . . 83
5.10 A short review of the literature . . . 84
5.11 What to report . . . 84

6 Regression: Advanced Considerations 87
6.1 One relationship, or thirty? . . . 87
6.2 Where the variation lives . . . 89
6.3 Fixed effects: the within transformation . . . 89
6.4 What fixed effects throws away, and what it cannot fix . . . 90
6.5 Two countries' worth of dummies, and a puzzle . . . 91
6.6 Random effects, and why the choice is testable . . . 92
6.7 Clustered standard errors, properly this time . . . 93
6.8 Does the slope differ by group? Ask it directly . . . 94
6.9 Yule's warning, which predates all of this . . . 95
6.10 Get the functional form right first . . . 95
6.11 A short review of the literature . . . 96
6.12 What to report . . . 97

7 Principal Component and Factor Analyses 99
7.1 Six columns, how many things? . . . 99

7.2 Standardise, or the biggest units win . . . 102
7.3 How many components? . . . 104
7.4 Loadings and scores are different objects . . . 105
7.5 PCA is not factor analysis . . . 106
7.6 A factor is identified only up to rotation . . . 108
7.7 What you may not call the components . . . 110
7.8 The zeros, which are not zeros . . . 110
7.9 A short review of the literature . . . 111
7.10 What to report . . . 111

8 K-Nearest Neighbours and the Bias–Variance Trade-off 113
8.1 A method with no model . . . 113
8.2 The floor nobody beats . . . 113
8.3 Where prediction error comes from . . . 114
8.4 Distance is not given, it is chosen . . . 117
8.5 The curse of dimensionality . . . 117
8.6 Choosing k, and why training error lies . . . 118
8.7 Did the flexibility earn its keep? . . . 120
8.8 A worked example: can the market's past predict today? . . . 121
8.9 What nearest neighbours cannot do . . . 125
8.10 A short review of the literature . . . 125
8.11 What to report . . . 125

9 Structural Equation Modelling 127
9.1 Measuring what cannot be observed . . . 127
9.2 Wright's diagrams, and what a path meant . . . 128
9.3 The language: five symbols and three operators . . . 129
9.4 What is actually being fitted . . . 130
9.5 Building a measurement model, step by step . . . 130
9.6 The fit indices, and what each would have to be . . . 134
9.7 Where the model misses, and what you may do about it . . . 136
9.8 Adding the structural part . . . 138
9.9 What the estimator assumes . . . 141
9.10 Good fit is not evidence that the model is correct . . . 142
9.11 What SEM does not fix . . . 143
9.12 The recipe . . . 144
9.13 A short review of the literature . . . 145
9.14 What to report . . . 145

10 Causal Inference I: Counterfactuals, Randomisation, Matching 147
10.1 The question the first nine chapters could not answer . . . 147
10.2 Potential outcomes . . . 149
10.3 Randomisation, and what it buys . . . 150
10.4 A world where the answer is known . . . 151

10.5 LaLonde's test . . . 152
10.6 The propensity score, and a match that fails its check . . . 154
10.7 A score that balances, and only then the outcome . . . 156
10.8 The control that changes the question . . . 159
10.9 What none of this fixes . . . 160
10.10 A short review of the literature . . . 161
10.11 What to report . . . 162

11 Causal Inference II: Difference-in-Differences 163
11.1 What would have happened otherwise . . . 163
11.2 The two differences . . . 165
11.3 As a regression . . . 166
11.4 When treatment arrives at different times . . . 166
11.5 The data, and what it will and will not support . . . 167
11.6 The estimates . . . 168
11.7 The estimator has a name . . . 170
11.8 Pre-trends: what this data cannot tell you . . . 172
11.9 How large a violation would it take? . . . 174
11.10 What difference-in-differences cannot fix . . . 175
11.11 A short review of the literature . . . 175
11.12 What to report . . . 176

12 Conclusion 177
12.1 One argument, twelve times . . . 177
12.2 The arithmetic never refuses . . . 177
12.3 The same data answer differently depending on what you ask . . . 177
12.4 Fit does not identify structure . . . 178
12.5 Flexibility is a purchase, not a virtue . . . 178
12.6 Selection is the problem, and it is not a technicality . . . 178
12.7 What the book checked, and what it could not . . . 179
12.8 What is not here . . . 180
12.9 What to carry . . . 180
12.10 Where the work goes now . . . 181

Appendices 183

A Setting up your tools 183
A.1 VS Codium, rather than VS Code . . . 183
A.2 Qwen 2.5 Coder, running locally . . . 183
A.3 Two ways of using it, deliberately . . . 184
A.4 git . . . 185
A.5 What they have in common . . . 185
A.6 Doing it . . . 186

A.7 Setup checklist — VS Codium, Python, and a local LLM 186
A.8 Setup guide — Git, GitHub, and how you submit work 192

B The data API **201**
B.1 Outside a clone 201
B.2 What is where 201

C Replication packages **203**
C.1 Article–data map 203
C.2 Downloading a package 207

D Further reading **209**
D.1 The two nearest neighbours of this book 209
D.2 Going deeper on the methods 209
D.3 Causal inference, where this book stops 210
D.4 The gaps the conclusion admits 210

Foreword

Ukraine supplied around seventy per cent of the world's neon. Russia controlled some forty-four per cent of its palladium. Taiwan fabricates close to two-thirds of its semiconductors. None of those three facts was secret, and none of them appeared on the balance sheet of a company that depended on all three — until a pandemic and a war made them appear at once (Warin 2022). Firms discovered the shape of their own supply chains by watching them break. The research literature had been arguing for years that concentration and fragility in global production were badly measured and badly understood; what it lacked was an event that made the point unarguable (Baldwin and Freeman 2022).

That is the difficulty this book is written against. The transformations that matter most in international business are not hidden. They are *unmeasured* by the people they will affect, and the received account of how the world economy works quietly supplies the missing numbers with assumptions. Chief among them is the assumption that global value chains, having been optimised for decades, must be efficient — an assumption that survives largely because so few people test it against data (Warin 2025c).

A second difficulty is that the unit of analysis is wrong in most public discussion. Countries do not trade; firms do. This is not a rhetorical point: heterogeneous-firm trade theory reorganised the field around it, showing that aggregate trade responds mainly through which firms enter and leave export markets rather than through all firms adjusting together (Melitz 2003; Warin 2026a). Aggregate statistics on bilateral flows are the sum of those decisions, taken under constraints — freight, insurance, policy wedges, the cost of switching a supplier — that the aggregate cannot show. The gravity equation remains the workhorse for describing those flows (Anderson and Wincoop 2003), and describing them is not the same as judging whether they are the ones that should be happening (Warin 2025a). Work at the level where the decision is actually made requires data at that level, and methods equal to it.

And the ground itself is moving. The central economic shift of this decade is not digitisation but a pivot from producing goods and knowledge toward *valuing the data generated in the course of producing them* — in agriculture, manufacturing, health, logistics and public administration alike (Warin 2026b). An economy organised around data capture and platform intermediation concentrates advantage differently than one organised around factories, and it rewards the institutions and the analysts who can read what the data say (Warin 2023).

The methods have moved with it. Econometrics has absorbed a toolkit built for prediction rather than for inference, and the two demand different care (Varian 2014); the data themselves are increasingly administrative, transactional and unstructured rather than surveyed (Einav and Levin 2014). Quantities nobody could measure at all are now constructed — economic policy uncertainty read off newspaper archives has entered mainstream macroeconomics (Baker, Bloom, and Davis 2016), and the

productive capabilities of a whole economy can be inferred from the composition of what it exports (Hidalgo and Hausmann 2009). None of this removes the need for the discipline this book teaches. It multiplies the number of places to go wrong. None of that is an argument that quantitative methods produce certainty. It is an argument that they are the only way to hold a claim about the world economy to account — to say what would have to be true for it to hold, and to notice when it stops holding. Trade theory has been rewritten repeatedly, from the mercantilists through Ricardo to the platform economy, and each rewriting followed evidence that the previous account could not absorb (Warin 2025b).

> **This book teaches the methods, on real data, from the first week.** Teaching statistics in a business school pulls in two directions: real data are messy, and pedagogical data are false. This course resolves it in favour of the mess (Warin 2024). Every technique here is applied to a panel carrying the missing values, the influential observations and the awkward panel structure of the real European sources — as teaching fixtures generated from a known structure, so that the methods behave as they would on the real series while no number here can be mistaken for a fact about Europe. Chapter 1 says what that means and what it costs.

Twelve chapters follow the arc from describing data to defending a causal claim. Each one asks a single question — *can this regression be trusted?*, *did the policy do anything?* — and each ends where a referee would begin.

There is a companion volume in R. *Data Science for International Business with R* (warin.ca/ds4ibr) covers much of the same ground — exploratory analysis, regression and panel data, logistic models, unsupervised learning, structural equation modelling — in the other language this field is written in. The two books are deliberately close in structure, so a reader fluent in one can follow the other and see what is a property of the method and what is a property of the tooling. This one is the Python volume.

How to use it

If you want to	Go to
Prepare for next week	the session's **Before the session**
Re-read a derivation you lost in class	the session's lecture sections
Know what the group work asks for	the session's **In the practice**
Find where something was covered	the **search box**, top of the sidebar
Know how you are graded	the syllabus, in the course repository
Get your machine working	Setting up your tools

> **The search box is the reason this book exists.** Slides are good for a room and bad for finding something three weeks later. Search covers every session at once, so a term you half-remember — *leverage*, *soft-thresholding*, *parallel trends* — takes you to the session that introduced it.

The slides

Each chapter links to the three decks it was built from. The slides remain the version delivered in class; the book is the same material set as continuous prose, which is the better form for reading afterwards.

The twelve chapters

	Chapter	Question
1	Foundations: Scenarios, Tools, and the Syllabus	Before we model the future, what are we claiming to know?
2	Exploratory Data Analysis, and the First Model	Before you model anything, what does the data actually look like?
3	Regression: Adequacy, Validity, and Robustness	You have fitted a regression. Can it be trusted?
4	Logistic Regression: Binary, Ordinal, and Multinomial	The outcome is a category, not a number. Now what?
5	Regularisation: Ridge, Lasso, and the Elastic Net	When is a deliberately biased estimator the better one?
6	Regression: Advanced Considerations *(asynchronous)*	Does the relationship hold across countries, and across years?
7	Principal Component and Factor Analyses	How many independent things are actually being measured?
—	**Midterm** — on paper, closed book	**Chapters 1–7**
8	K-Nearest Neighbours and the Bias–Variance Trade-off	Flexible, or just unstable?

	Chapter	Question
9	Structural Equation Modelling	Can you measure something you cannot observe?
10	Causal Inference I: Counterfactuals, Randomisation, Matching	Did the policy do anything, or were the groups different to begin with?
11	Causal Inference II: Difference-in-Differences	What would have happened otherwise?
12	Final Group Presentations	Can you make a decision-maker act on this — and say what would change your mind?

Source: github.com/warint/quantitative-methods · 6 November 2026

Chapter 1

Introduction

1.1 The question underneath every method

When can a number about the past be used to make a claim about the future? Every technique in this book is an answer to some version of that question, and every one of them arrives with conditions attached. The conditions are the subject. Anyone can run a regression; the software will not stop you, and it will return coefficients to four decimal places whatever you feed it. What distinguishes work worth acting on is the ability to say what would have to be true for the number to mean what you want it to mean — and to notice when that stops being the case.

This is harder than it sounds, because the failure is silent. A misspecified model does not raise an error. A confidence interval computed from the wrong formula looks exactly like one computed from the right formula. A coefficient driven entirely by three observations is printed in the same typeface as one supported by four hundred. The whole apparatus of diagnostics exists because the arithmetic is indifferent to whether it is being used sensibly.

George Box put the useful version of this in a sentence that has been quoted into meaninglessness: all models are wrong, but some are useful (George E. P. Box 1976). The half that gets repeated is the first. The half that matters is the second, and the word doing the work in it is *some* — the claim is not that usefulness is guaranteed but that it must be established, case by case, for a stated purpose. "Useful for what?" is the question this book keeps returning to, because a model that answers one question well is frequently worthless for the question next to it.

1.2 Three questions that look alike

Most confusion in applied quantitative work comes from running together three things that are genuinely different. They use overlapping machinery, which is why they get conflated, and they license completely different statements.

> **Description** — what is in this data? Summaries, distributions, correlations, visualisations. Description makes no claim beyond the sample. It is the least glamorous of the three and the one most often skipped, which is why so many analyses discover in week six a problem visible in a scatterplot in week one.

> **Prediction** — given what I have seen, what will the next observation look like? A predictive model is judged on out-of-sample accuracy and by nothing

else. It does not need to be interpretable, its coefficients need not correspond to anything real, and a variable can earn its place purely by being correlated with something that matters.

Causal inference — if I intervene, what changes? This asks about a world that does not exist: the same units, under a different policy. No amount of data on the world as it is answers it automatically, because the comparison required was never observed.

The distinction between the second and the third is where careers and policies go wrong. A model can predict superbly and be useless for choosing an action. The standard example is a hospital model that predicts pneumonia patients with asthma have *lower* mortality risk — true in the data, because asthmatic patients were admitted directly to intensive care. As a prediction about the observed system it is correct. As a basis for deciding whom to send home it would kill people. The correlation is real; the intervention it appears to recommend inverts the mechanism that produced it.

Leo Breiman's essay on the two cultures of statistical modelling is the classic statement of how differently these communities think — one assuming a data-generating model and interpreting its parameters, the other treating the mechanism as unknown and optimising predictive accuracy (Breiman 2001). Both cultures are represented in this book. Chapters 2 through 6 are largely the first; chapters 7 and 8 are largely the second; chapters 11 and 12 are about the third question, which neither culture answers on its own.

The credibility of empirical work in economics improved substantially once the profession took the third question seriously as a design problem rather than an estimation problem — the shift Angrist and Pischke call the credibility revolution (Angrist and Pischke 2010). Its lesson generalises well beyond economics: what makes a causal claim believable is usually a feature of *how the comparison was constructed*, not of how sophisticated the estimator was.

1.3 Why international business is a hard case

The methods in this book are general. The setting is not, and the setting makes particular trouble.

Start with the unit of analysis. Public discussion of trade is conducted in country aggregates — a deficit with one country, a surplus with another — but countries do not trade. Firms do (Warin 2026a). A bilateral flow is the sum of decisions taken inside firms under constraints that the aggregate cannot display: freight, insurance, policy wedges, the cost and risk of switching a supplier (Warin 2025a). Analysis conducted at the level where the decision was not made will attribute to nations what happened in supply chains.

Then the observations are not independent, and this is not a technicality. The data in this book are European countries observed annually. Germany in one year and Germany in the next are not two independent draws from anything; countries share shocks, policies and business cycles. Chapter 3 shows what this does to a standard error — roughly a 74% understatement in the running example — and it is the kind of error that makes a marginal finding look decisive.

The system also reorganises itself while you study it. Concentration that nobody had measured becomes visible only under stress: Ukraine supplied around seventy per cent of the world's neon, Russia some forty-four per cent of its palladium, Taiwan close to two-thirds of its semiconductors, and firms learned the shape of their own dependencies by watching them break (Warin 2022). And the deeper shift now under way is not digitisation but a pivot from producing goods toward valuing the data generated in the course of producing them, which changes where advantage accumulates and which institutions can capture it (Warin 2026b).

Finally, the received account is not neutral. The claim that global value chains are efficient — having been optimised for decades, how could they not be — functions as an assumption rather than a finding, and it survives largely because it is so rarely tested against data (Warin 2025c). Received narratives fill gaps where measurement is absent. That is precisely the space quantitative methods exist to occupy.

1.4 What a model is, and what it is not

A model is a formal statement of what you believe about how the data arose. It is not a summary of the data, and the difference matters. A summary cannot be wrong — the mean is the mean. A model can be wrong, and its being falsifiable is the whole of its value.

Every model has three parts, and confusion about which part is failing accounts for most bad diagnosis:

1. **A structural claim** — the shape of the relationship. That income depends on productivity linearly, say. Get this wrong and your estimates are biased: the coefficient is answering a question you did not ask.
2. **A stochastic claim** — how the unexplained part behaves. Constant variance, independence, sometimes normality. Get this wrong and your estimates are generally *fine* while your uncertainty is not: the right number with the wrong error bar.
3. **A claim about scope** — the population and conditions under which the first two hold. This is the part written down least often and violated most.

Three of the twelve chapters are essentially about the first claim, three about the second, and the running theme of all twelve is the third. The scope condition is where "the model fits well" quietly becomes "the model applies here", and nothing in the output flags the transition.

1.5 How the book is organised

Twelve chapters, following an arc from describing data to defending a causal claim.

Chapters 2 to 6 build a model and test it. Exploratory analysis and the first regression; then the diagnostics that decide whether a fitted regression can be trusted at all; then outcomes that are categories rather than numbers; then regularisation, which asks when a deliberately biased estimator is the better one; then what changes when the data have both a country and a time dimension.

Chapters 7 to 9 look for structure nobody labelled. Principal component and factor analysis, on how many independent things are actually being measured; nearest neighbours and the bias–variance trade-off, on whether a flexible method is genuinely flexible or merely unstable; structural equation modelling, on whether something unobservable can be measured at all.

Chapters 10 and 11 confront the question the first nine cannot answer — whether anything caused anything. Counterfactuals, randomisation and matching first; then difference-in-differences, which asks what would have happened otherwise and answers it with an assumption you must state and defend.

Chapter 12 is about making a decision-maker act on the result, and saying what would change your mind.

Each chapter takes a single question as its title question, derives the method rather than asserting it, applies it to real European economic data, and ends where a referee would begin — with what the result does not license.

1.6 On the data, and on mess

Teaching quantitative methods in a business school pulls in two directions. Real data are messy: missing values, structural breaks, definitional changes, countries that join a series halfway through. Data constructed for teaching are clean and therefore false, and they train an intuition that fails on contact with anything real (Warin 2024).

This book resolves the tension toward the mess, and does it in a specific way that you need to understand before reading a single result.

The data used throughout are **teaching fixtures**, not observations. They are generated from a known latent structure, and they carry the schema, the country and year coverage, the observation flags, the missingness patterns and the structural breaks of the real European sources they stand in for. What they do not carry is the values. Poland does not have the GDP per capita this book prints for it.

That is a deliberate trade, and it buys two things. Every method behaves as it would on the real series — the missingness is awkward in the same places, the panel structure bites in the same way, the influential observations are influential for the same reasons — while the ground truth is known, so a diagnostic can be checked against what actually generated the data. And nobody is tempted to cite a course exercise as a finding about Europe.

> **Do not cite any number in this book as a fact about Europe.** The tables in `data/spine/` are fixtures. `scripts/build_spine/build.py` replaces them with live Eurostat, Comtrade and ECB data, at which point every figure in the book re-renders against the real thing — and the numbers change.

The awkwardness is left in, because handling it *is* the skill. The influential observation is not removed before you see it. The panel structure is not flattened away. When a diagnostic reveals a problem in the running example — and in chapter 3 it does, twice — the problem is reported rather than engineered out.

> **A result that does not reproduce is not a result.** Every analysis in this book runs from a clean clone of the repository it was written in, on open data, with open tools. That is not a stylistic preference. It is the standard every serious journal now applies, and it is the only way a reader can check you rather than take your word.

1.7 What this book asks of you

Four habits, which are worth more than any individual technique.

Say the units. A coefficient without units is not a finding. "1,111" is not an answer; "1,111 euros of GDP per capita per index point of productivity" is.

Report the uncertainty. A number without an interval invites a confidence the data do not support, and the interval is frequently the more interesting half.

Say what the result does not license. Every serious empirical claim has a boundary. Stating it is not hedging; it is the part that tells a reader you know what you did.

Prefer a well-argued refusal to a confident answer. "This cannot be claimed from these data, and here is what would be needed" is a complete and often correct result. The alternative — producing a number because a number was requested — is how quantitative work loses its authority.

None of this makes the answers certain. It makes them *accountable*, which is the most any empirical method offers and considerably more than the alternatives on offer.

Chapter 2

Exploratory Data Analysis, and the First Model

2.1 Looking, before deciding what to look for

The instinct on receiving a dataset is to fit something to it. Resist it for an hour. Almost every analysis that goes wrong went wrong before the model was fitted, in a decision taken without looking — a variable whose units were not what they seemed, a distribution with two modes averaged into one, a year with no observations at all.

John Tukey made the case that this preliminary work is not preliminary (Tukey 1962). Data analysis, he argued, is a science rather than a branch of mathematics: it proceeds by examination and conjecture, it tolerates approximate answers to the right question over exact answers to the wrong one, and its most important step — deciding what question the data can support — has no formula. Confirmatory statistics can only test a hypothesis you already have. Where the hypothesis comes from is exploratory work, and pretending otherwise is how a dataset ends up answering a question nobody asked.



Florence Nightingale · 1820–1910
Made the case with a chart rather than a table, and changed policy with it.
Henry Hering, circa 1860. Public domain, via Wikimedia Commons.

From the archive · Nightingale, 1858

Florence Nightingale returned from the Crimea with an argument and a table. The argument was that most of the army's dead had been killed by preventable disease rather than by the enemy, and that sanitary reform would therefore save more soldiers than anything the surgeons could do. The table was ignored.

So she drew it. Her polar-area diagram — the "coxcomb" — put each month around a circle and the deaths in each as an area, split by cause, and the blue wedge of preventable disease overwhelmed everything else on sight. The chart was circulated to a royal commission, to ministers, to the Queen. The reforms followed.

The lesson is not that charts are prettier than tables. It is that a table asks the reader to hold twenty-four numbers in their head and perform the comparison themselves, while a chart performs the comparison for them. Human visual judgement is very good at some comparisons — position along a common scale, most of all — and poor at others, and that ordering has since been measured rather than assumed (Cleveland and McGill 1984). Nightingale had no such research. She had the intuition, and a commission that needed convincing.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, qmib

core = qmib.load("core")
by_year = core.groupby("time")["gdp_pc_eur"].mean().dropna()

fig, (ax_t, ax_c) = plt.subplots(1, 2, figsize=(9.4, 4.2),
                                 gridspec_kw={"width_ratios": [1, 1.5]})

ax_t.axis("off")
rows = [f"{int(y)}    {v:>10,.0f}" for y, v in by_year.items()]
ax_t.text(0.5, 0.5, "year        mean\n" + "\n".join(rows),
          family="monospace", fontsize=8.5, va="center", ha="center",
          transform=ax_t.transAxes, color=CL.ink)
ax_t.set_title("as a table", fontsize=10, loc="left")

ax_c.plot(by_year.index, by_year.values, marker="o", ms=4.5, lw=1.8, color=C
ax_c.set_ylabel("mean GDP per capita (EUR)")
ax_c.set_title("as position along a common scale", fontsize=10, loc="left")
ax_c.margins(x=0.04)

fig.tight_layout()
```

as a table

year	mean
2010	36,271
2011	35,499
2012	33,248
2013	34,666
2014	30,290
2015	30,735
2016	30,664
2017	29,415
2018	29,639
2019	28,608
2020	25,942
2021	25,282
2022	28,587
2023	28,232
2024	29,270

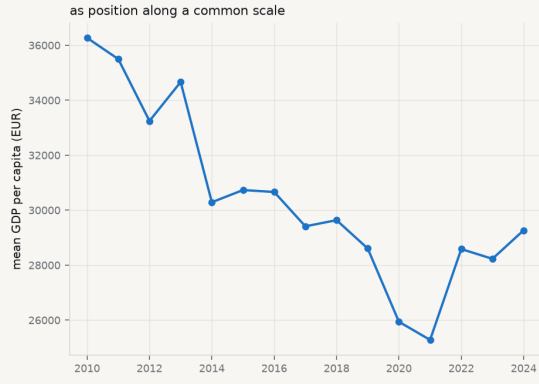


Figure 2.1. The same fourteen numbers, twice. Average GDP per capita by year in the course fixtures, as a table of figures and as position along a common scale. Nothing has been added on the right; the comparison has simply been done for you.

2.2 Look at it first, literally

Before any statistic, look at the thing. R users have `View()`; Python has no single equivalent, because a notebook renders the last expression, a script prints nothing, and `print(df)` truncates to the terminal width. `qmib.view()` gives the same answer in all three: the shape, every column with its type and how much of it is missing, and the first rows — scrollable here, printed in a terminal.

```
qmib.view(core)
```

```
core: 450 rows x 11 columns  (2.7% missing overall)
```

Table 2.1

	geo	time	gdp_pc_eur	population	employment_ths	productivity_idx	gfcf_meur	prod_growth	falling_behind	falling_behind_next	flag
0	AT	2010	38193.955	11495826.000	5265.073	103.349	95.222	—	0	0	
1	AT	2011	40644.251	12299989.000	5830.411	106.119	108.808	2.681	0	0	
2	AT	2012	33035.780	12294814.000	5746.553	108.084	87.626	—	0	0	
3	AT	2013	33595.312	11768544.000	5146.733	104.292	85.048	-3.508	0	0	
4	AT	2014	37265.749	11949079.000	5218.544	104.665	94.471	0.358	0	0	
5	AT	2015	23723.422	11555339.000	5094.443	105.127	57.017	0.441	0	1	
6	AT	2016	29535.812	13053057.000	5878.256	100.782	80.190	-4.133	1	0	
7	AT	2017	—	11379421.000	5042.292	108.135	56.972	7.296	0	1	
8	AT	2018	29754.420	12290876.000	5566.416	—	76.404	-3.278	1	0	
9	AT	2019	32816.404	12562395.000	5648.978	108.299	86.547	3.546	0	0	
10	AT	2020	28031.533	10963179.000	4655.423	104.708	64.633	-3.316	0	0	
11	AT	2021	22314.188	11769447.000	5044.122	106.423	54.926	1.638	0	0	

	dtype	missing	distinct
geo	str	0.0%	30
time	int64	0.0%	15
gdp_pc_eur	float64	2.9%	437
population	float64	3.3%	435
employment_ths	float64	1.3%	444
productivity_idx	float64	2.4%	434
gfcf_meur	float64	4.0%	432
prod_growth	float64	9.3%	399
falling_behind	Int64	0.0%	2
falling_behind_next	Int64	6.7%	2
flag	str	0.0%	5

Two things in that table are worth noticing before going further, and both are invisible in any summary statistic. Seven of the eleven columns have missing values, and `prod_growth` is missing far more often than the rest — a pattern this chapter returns to. And `flag` has five distinct values in a column that looks like it should be empty, which is the observation-status code the source uses to mark provisional and estimated figures.

2.3 Centre: three answers, three questions

The first question about a variable is where it sits. There are three standard answers and they are not interchangeable — each is the solution to a different problem.

> **Mean** — $\bar{x} = \frac{1}{n}\sum_i x_i$. The value minimising the sum of *squared* deviations, and the balance point of the distribution. It uses every observation, which is its strength and its weakness: one extreme value moves it without limit.

> **Median** — the middle order statistic, minimising the sum of *absolute* deviations. It ignores how far away the extremes are, only that they are on one side, so it is unmoved by an outlier of any magnitude. Its breakdown point is 50%: half the data must be corrupted before it fails.

> **Trimmed mean** — the mean of the data after discarding the most extreme α from each tail. A deliberate compromise: more efficient than the median when the distribution is nearly normal, more robust than the mean when it is not.

Which to report is a question about the decision the number will serve. If you are budgeting a total — total tax revenue, total emissions — the mean is the right

summary, because the total *is* the mean times the count and the extremes genuinely contribute. If you are describing a typical case — what a typical country looks like — the median is right, because the question is about the middle and not about the total.

```
from scipy import stats

y = core["gdp_pc_eur"].dropna()
mean, median = y.mean(), y.median()
trimmed = stats.trim_mean(y, 0.10)

print(f"n = {len(y)}   (of {len(core)} rows; {core['gdp_pc_eur'].isna().sum(
print(f"  mean          {mean:>10,.0f}")
print(f"  10% trimmed   {trimmed:>10,.0f}")
print(f"  median        {median:>10,.0f}")
print(f"\nmean − median = {mean − median:,.0f}, which is {(mean/median − 1):
print("A mean above the median is the signature of a right tail.")
```

```
n = 437   (of 450 rows; 13 missing)
  mean              30,373
  10% trimmed       28,723
  median            25,733

mean − median = 4,639, which is 18.0% of the median.
A mean above the median is the signature of a right tail.
```

The mean sits well above the median here, and the trimmed mean sits between them — the ordering that a right-skewed variable always produces. Reporting the mean alone would describe a country that does not exist: richer than most of the sample, because a few very rich country-years are pulling it up.

2.4 Spread, and the $n - 1$ that everyone asks about

Sample variance —

$$s^2 \;=\; \frac{1}{n-1}\sum_{i}(x_i - \bar{x})^2.$$

The divisor is $n - 1$, not n, because the deviations are taken from $\bar{x}$ rather than from the true mean μ. $\bar{x}$ is the value that *minimises* that sum of squares, so deviations from it are systematically too small, and dividing by n would return an estimate that is too low on average. One degree of freedom was spent estimating the centre; $n - 1$ remain.

That is the honest one-sentence answer, and it is worth being able to demonstrate rather than assert. The cell below does it by simulation: draw many samples from a distribution whose variance is known, and compare the average of the two estimators against the truth.

```
rng = np.random.default_rng(60033)
true_var, n, reps = 4.0, 5, 200_000
draws = rng.normal(0, np.sqrt(true_var), size=(reps, n))
dev2 = ((draws - draws.mean(axis=1, keepdims=True)) ** 2).sum(axis=1)

print(f"true variance                 {true_var:.4f}")
print(f"average of  sum/(n)    (biased)   {(dev2 / n).mean():.4f}")
print(f"average of  sum/(n-1)  (unbiased) {(dev2 / (n - 1)).mean():.4f}")
print(f"\nthe biased estimator is low by a factor of about {(n-1)/n:.2f} = (n-
```

```
true variance                 4.0000
average of  sum/(n)    (biased)   3.2017
average of  sum/(n-1)  (unbiased) 4.0021

the biased estimator is low by a factor of about 0.80 = (n-1)/n
```

With $n = 5$ the understatement is 20%, which is not a rounding error. It shrinks as n grows, which is why the choice rarely matters in a large sample and matters a great deal in a small one.
The variance is in squared units — squared euros, here, which is not a quantity anyone has an intuition for — so it is normally reported as its square root, the standard deviation. The alternative measure of spread makes no distributional assumption at all:

> **Interquartile range** — $\text{IQR} = Q_3 - Q_1$, the width of the middle half of the data. Like the median it is unaffected by the size of the extremes, and like the median it is the right choice when the distribution is skewed or heavy-tailed.

One more distinction belongs here, because the same square root is used for two different jobs and the two are confused constantly — including in published tables.

> **Standard deviation** — $s = \sqrt{s^2}$, the spread of the **observations** around their mean, in the units of the variable.

> **Standard error** — $\text{SE}(\bar{x}) = s/\sqrt{n}$, the spread of an **estimate** around the truth, across the samples you might have drawn but did not. It estimates $\sigma/\sqrt{n}$.

They share units and differ by a factor most people do not carry in their head, which is why they get swapped. What separates them is what each is the spread *of*: s describes the data, SE describes a number computed from the data.
The consequence is a test you can apply by eye. As n grows, s settles on a fixed σ — observations do not become less variable because you collected more of them — while SE shrinks as $1/\sqrt{n}$ without limit. So an error bar that stays the same width as the sample grows is a standard deviation, and one that narrows is a standard error. More data tells you more precisely where the mean is; it tells you nothing new about how spread out the observations are.

2.5 Shape, and a rule that passes when it should not

Two standardised moments describe departure from the normal shape.

> **Skewness** — the third standardised moment, $g_1 = \frac{1}{n}\sum_i \left(\frac{x_i - \bar{x}}{s}\right)^3$. Zero for any symmetric distribution; positive when the right tail is longer. The cube preserves sign, so observations far above the mean contribute positively and those far below negatively, and they do not cancel unless the distribution is symmetric.

> **Excess kurtosis** — the fourth standardised moment minus 3, so that a normal distribution scores zero. Positive means heavier tails than normal — more extreme observations than a normal would produce. It is *not* a measure of "peakedness", a description that survives in textbooks and is wrong.

The conventional threshold is that $|g_1| > 2$ or $|g_2| > 2$ indicates a serious departure. Now watch what happens when the same variable is tested by the empirical rule instead.

```
skew = stats.skew(y, bias=False)
kurt = stats.kurtosis(y, bias=False)          # already excess
sd = y.std(ddof=1)

print(f"skewness          {skew:+.3f}   (threshold ±2)")
print(f"excess kurtosis   {kurt:+.3f}   (threshold ±2)")
print(f"\nempirical rule — the fraction of observations within k standard de
for k, expected in ((1, 0.68), (2, 0.95), (3, 0.997)):
    got = ((y - y.mean()).abs() < k * sd).mean()
    print(f"  within {k} sd   {got:6.1%}   normal says {expected:.1%}")
```

```
skewness          +0.887   (threshold ±2)
excess kurtosis   +0.014   (threshold ±2)
```

```
empirical rule — the fraction of observations within k standard deviations:
  within 1 sd    68.0%   normal says 68.0%
  within 2 sd    95.7%   normal says 95.0%
  within 3 sd    99.3%   normal says 99.7%
```

Read that carefully, because it is the point of the section. The variable is visibly right-skewed — the mean exceeds the median by a wide margin, and the skewness statistic agrees. Yet the empirical rule is satisfied almost exactly: 68% within one standard deviation, close to 95% within two.

The empirical rule is a *weak* check. It constrains the bulk of the distribution, where skew has little effect, and says almost nothing about the tails, which is where the departure lives. Passing it is not evidence of normality. Its precondition is the thing being tested, which makes it useless as a test and merely reassuring as a description — a distinction worth carrying into every diagnostic in this book.

2.6 Transformation, and what it costs

The standard response to a right-skewed positive variable is to take logarithms. It works, in the sense that the skew largely disappears. It is worth watching what else changes.

```
ly = np.log(y)
fig, axes = plt.subplots(1, 2, figsize=(9.4, 3.8))

for ax, series, name in ((axes[0], y, "GDP per capita (EUR)"),
                         (axes[1], ly, "log GDP per capita")):
    ax.hist(series, bins=32, color=CL.accent, alpha=0.55, edgecolor="white", li
    ax.axvline(series.mean(), color=CL.warn, lw=1.8, label="mean")
    ax.axvline(series.median(), color=CL.warn, lw=1.5, ls="--", label="median")
    ax.set_xlabel(name)
    ax.set_title(f"skew {stats.skew(series, bias=False):+.2f}   "
                 f"excess kurtosis {stats.kurtosis(series, bias=False):+.2f}",
                 fontsize=9.5, loc="left")
axes[0].set_ylabel("count")
axes[0].legend(frameon=False, fontsize=8.5)
fig.tight_layout()
```

The skew falls almost to zero, as advertised. The excess kurtosis moves the other way, from very slightly positive to clearly negative — the transformed variable has *lighter* tails than a normal distribution, not merely lighter than before. Taking logs did not make the variable normal. It made it a different non-normal shape.

That is the general situation, and it is why "take logs to fix normality" is bad advice stated that way. The good reasons to take logs are substantive: because the effect you believe in is proportional rather than additive, because the variable is a ratio or a price, or because the relationship you are about to model is multiplicative. If

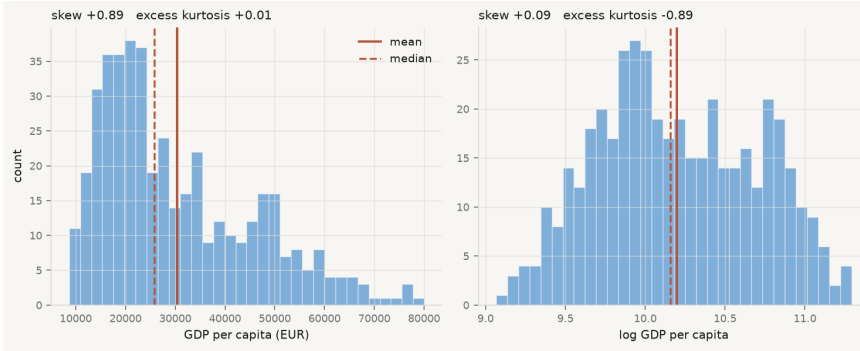


Figure 2.2. GDP per capita in the course fixtures, raw and log-transformed, with mean (solid) and median (dashed). Taking logs removes most of the skew; look at what it does to the tails.

those hold, the transformation is right whatever it does to the shape statistics. If they do not, you have changed the quantity you are modelling in order to improve a diagnostic, which is the tail wagging the dog.

2.7 What is missing, and why

A summary of the observed values says nothing about the values that are absent, and absence is rarely accidental.

> **MCAR, MAR, MNAR** — Rubin's taxonomy (DONALD B. Rubin 1976). *Missing completely at random*: the probability of being missing is unrelated to anything, observed or not — the only case where dropping incomplete rows is harmless. *Missing at random*: the probability depends on observed variables, so it can be modelled. *Missing not at random*: it depends on the unobserved value itself — the highest incomes are the ones not reported — which cannot be diagnosed from the data alone, because the evidence you would need is precisely what is missing.

```
cols = ["gdp_pc_eur", "population", "employment_ths",
        "productivity_idx", "gfcf_meur", "prod_growth"]
grid = (core.pivot_table(index="geo", columns="time", values=cols,
                         aggfunc="first", dropna=False)
        .isna())

fig, ax = plt.subplots(figsize=(9.2, 3.4))
share = pd.DataFrame({c: grid[c].mean(axis=0) for c in cols}).T
im = ax.imshow(share.values, aspect="auto", cmap="bone_r", vmin=0, vmax=1)
```

```
ax.set_yticks(range(len(cols))); ax.set_yticklabels(cols, fontsize=8.5)
ax.set_xticks(range(len(share.columns)))
ax.set_xticklabels([int(t) for t in share.columns], fontsize=8, rotation=90)
ax.set_title("share of countries missing, by variable and year", fontsize=10,
fig.colorbar(im, ax=ax, fraction=0.025, pad=0.01)
fig.tight_layout()
```

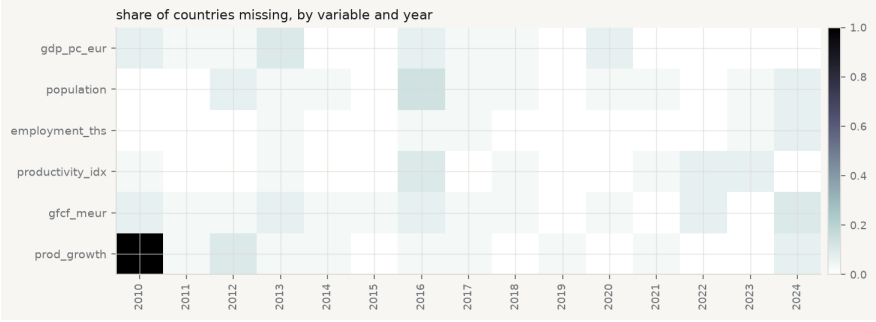


Figure 2.3. Missingness in the course fixtures. Each cell is one country-year; dark means absent. The pattern is not noise — read the first column.

```
miss = core["prod_growth"].isna()
print(f"prod_growth missing: {miss.sum()} of {len(core)} ({miss.mean():.1%})")
print(f"  in 2010 alone:     {miss[core.time == 2010].mean():.0%}")

obs = core.dropna(subset=["gdp_pc_eur"])
present = obs.loc[~obs["prod_growth"].isna(), "gdp_pc_eur"].mean()
absent = obs.loc[obs["prod_growth"].isna(), "gdp_pc_eur"].mean()
print(f"\nmean GDP per capita where prod_growth is present: {present:>9,.0f}")
print(f"                           where it is missing:   {absent:>9,.0f}")
print(f"  difference: {absent - present:+,.0f}  ({(absent/present - 1):+.1%})")
print("\nMissingness that depends on an observed variable is MAR, not MCAR.")
```

```
prod_growth missing: 42 of 450 (9.3%)
  in 2010 alone:     100%

mean GDP per capita where prod_growth is present:    30,023
                          where it is missing:      34,044
  difference: +4,021  (+13.4%)

Missingness that depends on an observed variable is MAR, not MCAR.
```

Two findings, of different kinds. The first is structural and completely explicable: `prod_growth` is missing for *every* country in 2010 because it is a year-on-year

growth rate and 2010 is the first year in the panel. There is no 2009 to difference against. That is not missing data in any interesting sense — it is a variable that does not exist for that year, and treating it as an imputation problem would be a category error.

The second is not structural. Where `prod_growth` is absent, the average GDP per capita differs materially from where it is present. Missingness is associated with an observed variable, which puts it in Rubin's *missing at random* category and rules out the convenient assumption. Dropping incomplete rows — which every regression in this book does by default — is therefore not neutral: it changes the composition of the sample toward the country-years that report. Whether that matters depends on the question, and the only wrong move is not to check.

2.8 The first model, as a right-angled triangle

Now fit something. The simplest model relates one variable to one other,

$$y_i \;=\; \beta_0 + \beta_1 x_i + \varepsilon_i,$$

and the least-squares solution in this case has a closed form worth memorising, because it says what a slope *is*:

$$\hat{\beta}_1 \;=\; \frac{\operatorname{Cov}(x, y)}{\operatorname{Var}(x)}, \qquad \hat{\beta}_0 \;=\; \bar{y} - \hat{\beta}_1 \bar{x}.$$

The slope is covariation divided by variation in the predictor: how much x and y move together, per unit of x moving at all. The intercept exists only to make the line pass through $(\bar{x}, \bar{y})$, which it always does.

The geometry is where the understanding is. Think of the n observations as a single vector in n-dimensional space. Centre it — work with $y - \bar{y}$ — and fitting a line is projecting that vector onto the direction of $x - \bar{x}$. The fitted values are the shadow; the residuals are what the shadow misses; and because a projection is orthogonal, the residual is at right angles to the fitted vector. A right angle means Pythagoras:

$$\underbrace{\lVert y - \bar{y} \rVert^2}_{\text{TSS}} \;=\; \underbrace{\lVert \hat{y} - \bar{y} \rVert^2}_{\text{ESS}} \;+\; \underbrace{\lVert y - \hat{y} \rVert^2}_{\text{RSS}}.$$

That is the analysis of variance, and it is not an algebraic coincidence — it is the hypotenuse. Two consequences follow immediately. $R^2 = \text{ESS}/\text{TSS}$ is $\cos^2$ of the angle between the observed and fitted vectors. And in a simple regression the correlation r is exactly that cosine, so $R^2 = r^2$ — a fact usually presented as a curiosity and actually a definition.

```
import statsmodels.api as sm

c = core.dropna(subset=["gdp_pc_eur", "productivity_idx"])
x, yv = c["productivity_idx"].to_numpy(), c["gdp_pc_eur"].to_numpy()
```

```
b1 = np.cov(x, yv, ddof=1)[0, 1] / np.var(x, ddof=1)
b0 = yv.mean() - b1 * x.mean()
fit = sm.OLS(yv, sm.add_constant(x)).fit()
print(f"slope by hand {b1:12,.3f}   statsmodels {fit.params[1]:12,.3f}")
print(f"intercept     {b0:12,.1f}   statsmodels {fit.params[0]:12,.1f}")

tss = ((yv - yv.mean()) ** 2).sum()
ess = ((fit.fittedvalues - yv.mean()) ** 2).sum()
rss = (fit.resid ** 2).sum()
print(f"\nTSS {tss:.6e}")
print(f"ESS {ess:.6e}")
print(f"RSS {rss:.6e}")
print(f"TSS - (ESS + RSS) = {tss - ess - rss:.3e}   (zero, to floating point)")

r = np.corrcoef(x, yv)[0, 1]
print(f"\nr = {r:.6f};  r² = {r**2:.6f};  R² = {fit.rsquared:.6f}")
print(f"angle between the centred observed and fitted vectors: "
      f"{np.degrees(np.arccos(r)):.2f}°")
```

```
slope by hand    1,111.257   statsmodels    1,111.257
intercept         -86,118.1   statsmodels    -86,118.1

TSS 1.004705e+11
ESS 6.302104e+10
RSS 3.744941e+10
TSS - (ESS + RSS) = 7.629e-06   (zero, to floating point)

r = 0.791997;  r² = 0.627259;  R² = 0.627259
angle between the centred observed and fitted vectors: 37.63°
```

```
fig, (axs, axg) = plt.subplots(1, 2, figsize=(9.4, 4.2))

axs.scatter(x, yv, s=14, color=CL.accent, alpha=0.45, edgecolor="none")
gx = np.linspace(x.min(), x.max(), 50)
axs.plot(gx, b0 + b1 * gx, color=CL.warn, lw=2)
axs.plot([x.mean()], [yv.mean()], marker="o", ms=9, mfc="none",
         mec=CL.ink, mew=1.6)
axs.annotate(r"$(\bar x, \bar y)$", (x.mean(), yv.mean()),
             textcoords="offset points", xytext=(10, -16), fontsize=9)
axs.set_xlabel("productivity index"); axs.set_ylabel("GDP per capita (EUR)")
axs.set_title("the fit", fontsize=10, loc="left")

# The triangle, drawn to the true proportions: |ESS|, |RSS|, |TSS|.
a, b_, hyp = np.sqrt(ess), np.sqrt(rss), np.sqrt(tss)
axg.plot([0, a, a, 0], [0, 0, b_, 0], color=CL.ink, lw=1.8)
```

```
axg.fill([0, a, a], [0, 0, b_], color=CL.accent, alpha=0.10)
axg.text(a/2, -b_*0.09, r"$\|\hat y-\bar y\|=\sqrt{ESS}$", ha="center", font
axg.text(a*1.02, b_/2, r"$\|y-\hat y\|=\sqrt{RSS}$", va="center", fontsize=9
axg.text(a*0.42, b_*0.62, r"$\|y-\bar y\|=\sqrt{TSS}$", rotation=np.degrees(
        ha="center", fontsize=9, color=CL.warn)
axg.annotate(rf"$\theta={np.degrees(np.arccos(r)):.1f}^\circ$", (a*0.16, b_*
axg.set_aspect("equal"); axg.axis("off")
axg.set_title(r"the same fit, in observation space", fontsize=10, loc="left"

fig.tight_layout()
```

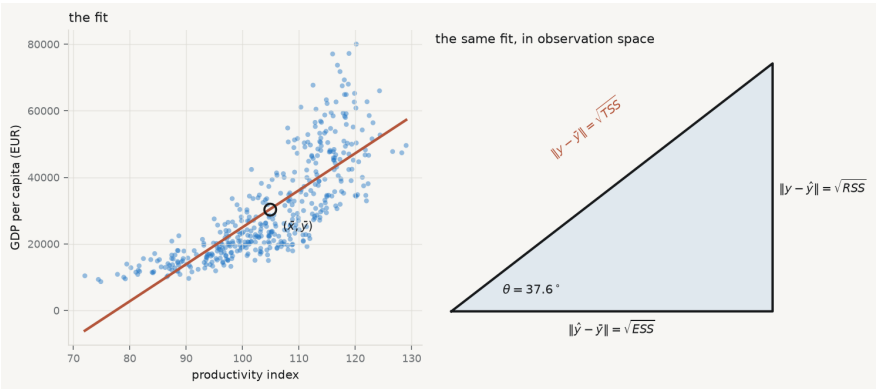


Figure 2.4. Left: the fitted line, passing through the point of means as it must. Right: the same regression as a right-angled triangle in observation space, with sides in the ratio the numbers above give. R^2 is $\cos^2\theta$.

The angle is the whole summary. A small angle means the fitted vector lies close to the observed one and R^2 is near 1; a right angle means the predictor explains nothing. Reporting $R^2 = 0.63$ is reporting an angle of about $38°$, and the geometric statement is the more honest one, because nobody is tempted to describe an angle as a percentage of anything.

2.9 More than one predictor, and what "controlling for" does

One predictor is rarely enough, and the extension is mechanical:

$$y_i \;=\; \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_{p-1} x_{p-1,i} + \varepsilon_i .$$

The interpretation is where the trouble starts. Every textbook says β_1 is the effect of x_1 "holding the other variables constant", and that phrase is worth taking apart, because nothing has been held constant. Nobody intervened. The data are what they are.

> **Partial slope** — the coefficient on x_1 in a multiple regression: the association between y and the part of x_1 that the other predictors cannot account for. It is not the effect of changing x_1; it is the association remaining after the overlap with the other predictors has been removed from *both* sides.

That is not a metaphor. It is a theorem, proved independently by Frisch and Waugh (1933) and generalised by Lovell (1963), and it can be executed in four lines. Regress y on the other predictors and keep the residuals. Regress x_1 on the other predictors and keep those residuals. Regress the first set of residuals on the second. The slope you get is *exactly* the multiple-regression coefficient on x_1 — not approximately, to floating point.

```
cols = ["gdp_pc_eur", "productivity_idx", "employment_ths", "population"]
cc = core.dropna(subset=cols)
others = sm.add_constant(cc[["employment_ths", "population"]])

simple = sm.OLS(cc["gdp_pc_eur"], sm.add_constant(cc[["productivity_idx"]])).fi
multi = sm.OLS(cc["gdp_pc_eur"],
               sm.add_constant(cc[["productivity_idx", "employment_ths", "popul

# Frisch-Waugh-Lovell: partial both sides, then regress residual on residual.
res_y = sm.OLS(cc["gdp_pc_eur"], others).fit().resid
res_x = sm.OLS(cc["productivity_idx"], others).fit().resid
fwl = sm.OLS(res_y, res_x).fit()

print(f"complete rows: {len(cc)}")
print(f"  simple regression, slope on productivity   {simple.params.iloc[1]:>12
print(f"  multiple regression, same slope            {multi.params.iloc[1]:>12,
print(f"  residual-on-residual (Frisch-Waugh-Lovell) {fwl.params.iloc[0]:>12,.2
print(f"\n  |multiple - FWL| = {abs(multi.params.iloc[1] - fwl.params.iloc[0])
print(f"  the slope falls by {(1 - multi.params.iloc[1]/simple.params.iloc[1])
      f"once employment and population are partialled out")
```

```
complete rows: 413
  simple regression, slope on productivity       1,112.44
  multiple regression, same slope                  818.55
  residual-on-residual (Frisch-Waugh-Lovell)       818.55

  |multiple - FWL| = 4.84e-09
 the slope falls by 26% once employment and population are partialled out
```

Two things to take from that output.
The first is the identity. "Controlling for employment and population" *means* removing from productivity everything employment and population can predict about

it, removing the same from GDP per capita, and relating what is left. The multiple regression does it in one step; the residual regression does it in three; the answer is the same number.

The second is the size of the change. The slope falls by a quarter. Neither number is wrong — they answer different questions. The simple slope asks how GDP per capita differs between country-years that differ in productivity, whatever else differs alongside. The partial slope asks how it differs between country-years that differ in productivity *and are otherwise similar in employment and population*. Which one you want depends on the question, and a result reported without saying what was in the regression is uninterpretable.

There is a trap here that chapters 10 and 11 will return to at length. Adding a control is not automatically an improvement. If a variable sits on the causal path between x and y — productivity raises output, which raises employment — then controlling for it removes part of the very association you were trying to measure. If a variable is a common *consequence* of x and y, controlling for it can manufacture an association that does not exist. "Add more controls" is not a strategy, and the decision about what belongs in the regression is a claim about how the world works, not a statistical one.

2.10 Say the slope in units

A slope is a rate, and a rate without units is not a finding. The number above is roughly 1,111 — of what, per what?

The units are euros of GDP per capita per one point of the productivity index. So: *a country-year one index point higher in productivity is associated with about 1,111 euros more GDP per capita.* Every part of that sentence is doing work. "Associated with" rather than "causes", because nothing here identifies a causal effect and chapters 10 and 11 explain what it would take. "A country-year" rather than "a country", because the unit of observation is the pair. And the units make the magnitude checkable: against a mean of about 30,000 euros, one index point is a few per cent, which is large enough to matter and small enough to be plausible.

Report a coefficient without its units and no reader can perform that check — which is usually the point at which an implausible result would have been caught.

2.11 What a summary cannot show

Everything in this chapter reduces a distribution to a few numbers, and every such reduction discards information that may be the only interesting thing present. Anscombe's four datasets, which chapter 3 opens with, share mean, variance, correlation and fitted line while differing completely. The demonstration has since been automated: it is possible to construct datasets with identical summary statistics to several decimal places and arbitrary shapes, including a dinosaur (Matejka and Fitzmaurice 2017).

Two further cautions belong here, because both are committed at the exploratory

stage and paid for later.

The first is judging a difference by whether error bars overlap. Two confidence intervals can overlap substantially while the difference between the estimates is comfortably significant, because the standard error of a difference is not the sum of the standard errors (Schenker and Gentleman 2001). If the question is about a difference, compute an interval for the difference.

The second is the one this whole chapter is exposed to. Exploratory analysis means looking at the data before choosing a specification, and the inference that follows is then conditional on everything you saw. The honest responses are to say so, to treat the resulting p-values as descriptive, or to report the whole space of defensible specifications rather than the one you settled on (Simonsohn, Simmons, and Nelson 2020). What is not honest is to explore thoroughly and then report as though the model had been specified in advance.

2.12 A short review of the literature

The case for exploratory work as a discipline rather than a preliminary is Tukey (1962), which argued that data analysis is a science with its own standards and that the choice of question is the part no theorem supplies. Anscombe (1973) supplied the demonstration that summary statistics are insufficient; Matejka and Fitzmaurice (2017) automated its construction, producing arbitrarily different datasets with identical summaries.

How a chart is read has been studied rather than assumed since Cleveland and McGill (1984), whose ranking of graphical perception tasks — position along a common scale first, angle and area far below — remains the basis for choosing an encoding. Healy and Moody (2014) surveys what has and has not been taken up in practice.

Missing data has a formal framework in DONALD B. Rubin (1976), whose three categories decide whether dropping incomplete rows is harmless, correctable or hopeless. It is routinely cited and routinely ignored, since complete-case analysis is the default of every statistical package.

On the shape statistics, G. E. P. Box (1953) established early that tests for non-normality are more sensitive to the assumption than the procedures they are meant to protect — "to make preliminary tests on variances is rather like putting to sea in a rowing boat to find out whether conditions are sufficiently calm for an ocean liner to leave port". The warning generalises to most preliminary testing, including the empirical rule this chapter shows passing on skewed data.

The exposure created by exploring before specifying is treated by Simonsohn, Simmons, and Nelson (2020) as a problem of reporting the whole specification space, and by Gelman and Loken (2014) as a problem of researcher degrees of freedom that requires no dishonesty to produce. Schenker and Gentleman (2001) addresses the narrower and very common error of reading significance off overlapping intervals.

2.13 What to report

A profile of a variable that would survive a referee answers five things, and takes a paragraph and one chart.

1. **How many observations, and how many missing.** With the pattern, not just the count — and a sentence on whether the missingness looks structural, related to something observed, or unexplained.
2. **Centre, with the choice defended.** Mean or median is a decision about the question, not a convention. Say which question.
3. **Spread, in the units of the variable.** Standard deviation or IQR, consistent with the choice of centre.
4. **Shape, tested against a threshold.** Skewness and excess kurtosis with the values, not the adjectives — and if you invoke the empirical rule, say that it is a weak check.
5. **The slope in units, if you fitted one**, with "associated with" rather than a causal verb, and with R^2 reported as what it is: the square of a cosine.

The chapter's own running example fails several of its own tests, which is the normal condition of real data and the reason chapter 3 exists. The variable is skewed, so the mean overstates the typical case; the missingness is not completely at random, so complete-case analysis shifts the sample; and the empirical rule passes anyway, which should increase nobody's confidence.

Chapter 3

Regression: Adequacy, Validity, and Robustness

3.1 The line, and where it came from

Adrien-Marie Legendre published the method of least squares in 1805, in an appendix to a book about the orbits of comets. Four years later Carl Friedrich Gauss published it too, in *Theoria Motus*, and added that he had been using it since 1795. The priority quarrel that followed was bitter and is still not entirely settled. What matters here is what Gauss added that Legendre did not have: not the recipe, but the *justification*. Legendre offered least squares as a sensible way to reconcile discordant observations. Gauss asked a different question — under what conditions is this the best you can do? — and that question is the subject of this chapter.



Carl Friedrich Gauss · 1777–1855
Least squares, and the theorem that says when it is the best you can do.
Christian Albrecht Jensen, 1840. Public domain, via Wikimedia Commons.



Adrien-Marie Legendre · **1752–1833**
Published least squares first, in 1805. This caricature is the only likeness of him known to exist.
Julien-Léopold Boilly, undated. Public domain, via Wikimedia Commons.

Set the problem up. You have n observations, a response y, and a matrix X holding a column of ones and $p - 1$ predictors. You want coefficients β such that $X\beta$ is close to y. Least squares defines "close" as the sum of squared vertical distances, and chooses

$$\hat{\beta} \;=\; \arg\min_{\beta} \, \|y - X\beta\|^2 .$$

Squared, rather than absolute, distance is worth pausing on, because the choice is not obvious and it has consequences you will meet later in the chapter. Squaring makes the objective differentiable everywhere, which is what delivers the closed-form solution below; it also makes the fitted values a *linear* function of y, which is what makes the whole diagnostic apparatus possible. The price is that squaring gives a doubled error four times the weight of a single one, so least squares is structurally sensitive to observations far from the line. Every technique in this chapter for finding influential points is, in a sense, paying off the debt incurred by that square.

Differentiating and setting to zero gives the *normal equations* $X^\top X\beta = X^\top y$, whose solution — whenever $X^\top X$ can be inverted — is

$$\hat{\beta} \;=\; (X^\top X)^{-1} X^\top y .$$

Every equation in this chapter is something you can execute, and executing it is the fastest way to find out whether you have understood what it claims. The normal equations are the first: solve them by hand and the answer must be, to floating-point precision, whatever `statsmodels` returns.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, statsmodels.api as sm, qmib

d = (qmib.load("core")
     .dropna(subset=["gdp_pc_eur", "productivity_idx"])
     .reset_index(drop=True))
X = sm.add_constant(d["productivity_idx"]).to_numpy()
y = d["gdp_pc_eur"].to_numpy()

# beta_hat = (X'X)^-1 X'y — written as solve(), never as inv(), because
# inverting a matrix to multiply it by a vector is slower and less stable.
beta_hand = np.linalg.solve(X.T @ X, X.T @ y)
beta_sm = sm.OLS(y, X).fit().params

print("by hand      ", np.round(beta_hand, 6))
print("statsmodels  ", np.round(beta_sm, 6))
print("agree to     ", f"{np.abs(beta_hand - beta_sm).max():.2e}")
```

```
by hand       [-86118.084706   1111.257354]
```

```
statsmodels   [-86118.084706   1111.257354]
agree to      2.26e-09
```

That is an algebraic fact. It requires no assumption about the world, no probability, no error term. Feed it any numbers and it returns coefficients. This is exactly why diagnostics are necessary: **the formula never refuses.** It has no way to tell you that the relationship is curved, that one country is dragging the whole line, or that the standard errors it reports are fiction. It returns the same confident four decimal places either way.

The statistical content arrives only when you attach a model,

$$y \;=\; X\beta + \varepsilon,$$

and make claims about ε. The Gauss–Markov theorem says that if the model is correctly specified, if $\mathrm{E}[\varepsilon \mid X] = 0$, and if the errors have constant variance and are mutually uncorrelated, then $\widehat{\beta}$ is the Best Linear Unbiased Estimator: no other estimator that is both linear in y and unbiased has smaller variance. Note what is *not* on that list. Normality is not required for Gauss–Markov. It is required for something else — the exactness of the t and F distributions in small samples — and confusing the two is one of the most common errors in applied work.

The word "regression" itself arrived later and by accident. Francis Galton, studying the heights of parents and children, found that tall parents had children who were tall but *less* tall, and called this "regression towards mediocrity" (Galton 1886). He was naming a substantive biological phenomenon. The name stuck to the statistical method he had used to find it, which is why a technique for fitting conditional means is called by a word that means moving backwards.

From the archive · Galton, 1886

Galton's claim was about heredity, but the phenomenon is general and it is not a fact about biology: it follows from the arithmetic of an imperfect correlation. Whenever two measurements of the same thing are less than perfectly correlated, the extremes of the first are, on average, less extreme in the second — not because anything pulled them back, but because part of what made them extreme was noise, and noise does not repeat.

It is worth seeing this happen in the data this book uses. These are fixtures rather than observations, which for once does not weaken the demonstration: regression towards the mean is a consequence of imperfect correlation, not of economics, so it turns up in any series where this year predicts next year imperfectly — generated or observed. Take each country's productivity in a given year, expressed as a deviation from that year's cross-country average, and ask what that deviation looks like the following year.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
```

```
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import qmib, numpy as np, pandas as pd

d = (qmib.load("core").dropna(subset=["productivity_idx"])
     .sort_values(["geo", "time"]))
d["next"] = d.groupby("geo")["productivity_idx"].shift(-1)
pairs = d.dropna(subset=["next"]).copy()

# Deviation from the cross-country mean, this year and the next.
pairs["dev"] = pairs["productivity_idx"] - pairs.groupby("time")["productivity_
pairs["dev_next"] = pairs["next"] - pairs.groupby("time")["next"].transform("me

slope, intercept = np.polyfit(pairs["dev"], pairs["dev_next"], 1)

fig, ax = plt.subplots(figsize=(6.0, 4.6))
lim = np.array([pairs["dev"].min() * 1.05, pairs["dev"].max() * 1.05])
ax.plot(lim, lim, ls=":", lw=1.3, color=CL.muted, label="no regression to the m
ax.plot(lim, intercept + slope * lim, lw=2, color=CL.warn,
        label=f"fitted, slope = {slope:.2f}")
ax.scatter(pairs["dev"], pairs["dev_next"], s=15, color=CL.accent,
           alpha=0.5, edgecolor="none", zorder=3)
ax.axhline(0, color=CL.line, lw=1, zorder=1); ax.axvline(0, color=CL.line, lw=1
ax.set_xlabel("deviation from the mean, year $t$")
ax.set_ylabel("deviation from the mean, year $t+1$")
ax.legend(frameon=False, fontsize=8.5, loc="upper left")
fig.tight_layout()
```

The dotted line is what perfect persistence would look like: a country as far above average next year as it was this year. The fitted line is flatter. The most productive decile keeps about six-sevenths of its lead from one year to the next, and the least productive recovers by a comparable amount — with nothing whatever having been done to either.

The warning Galton's contemporaries missed is the one to carry forward. If you select the extremes and then measure them again, they will improve, and the improvement will look like the effect of whatever you did in between. Chapter 10 returns to this as a threat to causal inference, where it has a name: regression to the mean is the reason a single-group before-and-after comparison is not evidence.

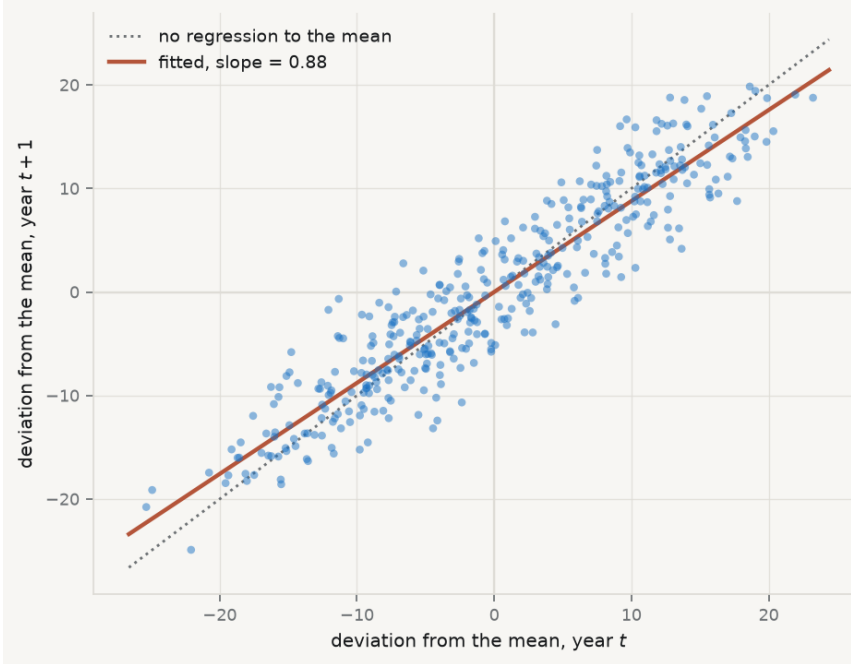


Figure 3.1. Regression towards the mean in the course fixtures, 2010–2024. Each point is one country-year: its deviation from the cross-country average that year, against its deviation the next. A slope below 1 is Galton's phenomenon. The values are generated, not observed — but the phenomenon is a property of imperfect correlation, so it appears here for the same reason it appears in real data.

3.2 Adequacy, validity, robustness

These three words get used loosely and they are not synonyms. The distinction organises everything that follows.

> **Adequacy** — whether the model describes the data you actually have. An adequate model leaves residuals with no remaining structure: nothing in what is left over could have been used to predict y better. Adequacy is assessed against the sample, and it is mostly a question you answer with plots.

> **Validity** — whether the inferences you draw from the model are warranted: whether the standard errors, confidence intervals and p-values mean what they claim. A model can be adequate and its inference invalid, most commonly because the errors are heteroscedastic or dependent. Validity is a question about the *sampling distribution*, not about fit.

> **Robustness** — whether your conclusions survive reasonable changes to the specification, the sample or the estimator. A robust finding is one that does not depend on a single influential observation, one functional form, or one way of computing standard errors. Robustness is a question about *stability*.

Keeping them apart tells you what a given failure actually costs, and this is the part most often got wrong. The four classical assumptions do not all protect the same thing:

If this fails	$\hat{\beta}$ is	Standard errors are	So the damage is to
Correct specification	**biased**	meaningless	the estimate itself
Constant variance	unbiased	**wrong**	validity only
Independent errors	unbiased	**wrong**, usually far too small	validity only
Normal errors	unbiased	fine in large samples	almost nothing, if n is large

Read the table twice. Only the first row damages the coefficient. Rows two and three leave your estimate untouched and destroy your uncertainty — you have the right number with the wrong error bar, which matters enormously if the question is whether an effect can be distinguished from zero. Row four, the assumption students worry about most, is the one that matters least.

A model with $R^2 = 0.95$ can fail all of it. This is not hypothetical: Anscombe's

four datasets share, to two decimal places, the same means, variances, correlation, regression line and R^2, and they are utterly different (Anscombe 1973).

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()
import numpy as np

x1 = [10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5]
quartet = {
    "I — the line is right":        (x1, [8.04, 6.95, 7.58, 8.81, 8.33, 9.96
    "II — the relation is curved":  (x1, [9.14, 8.14, 8.74, 8.77, 9.26, 8.10
    "III — one outlier tilts it":   (x1, [7.46, 6.77, 12.74, 7.11, 7.81, 8.8
    "IV — one point defines it":    ([8]*10 + [19], [6.58, 5.76, 7.71, 8.84,
}

fig, axes = plt.subplots(2, 2, figsize=(8.6, 6.4), sharex=True, sharey=True)
for ax, (name, (x, y)) in zip(axes.ravel(), quartet.items()):
    x, y = np.asarray(x, float), np.asarray(y, float)
    b = np.polyfit(x, y, 1)
    gx = np.linspace(2, 20, 50)
    ax.plot(gx, np.polyval(b, gx), color=CL.warn, lw=1.6, zorder=2)
    ax.scatter(x, y, s=36, color=CL.accent, zorder=3, edgecolor="white", lin
    r2 = np.corrcoef(x, y)[0, 1] ** 2
    ax.set_title(name, fontsize=10, loc="left", color=CL.ink)
    ax.text(0.97, 0.06, f"$R^2={r2:.2f}$", transform=ax.transAxes,
            ha="right", fontsize=9, color=CL.muted)
    ax.set_xlim(2, 20); ax.set_ylim(2, 14)
fig.tight_layout()
```

Every one of those four panels reports $R^2 = 0.67$ and the same fitted line. Only in the first is that line a description of the data. In the second the relationship is deterministic and curved — the model is not wrong about the strength of the association so much as wrong about its *shape*, and no amount of extra data will fix it, because more observations of a parabola still make a parabola. In the third a single aberrant point has tilted a line that would otherwise pass almost exactly through the rest; delete it and both the slope and the residual variance change sharply. In the fourth every x but one is identical, so the slope is determined entirely by one observation: delete it and the slope is not merely different, it is undefined, because $X^\top X$ becomes singular.

Anscombe's point was not that summary statistics are useless. It was that they are *insufficient*, and that the missing information is available for the cost of a plot.

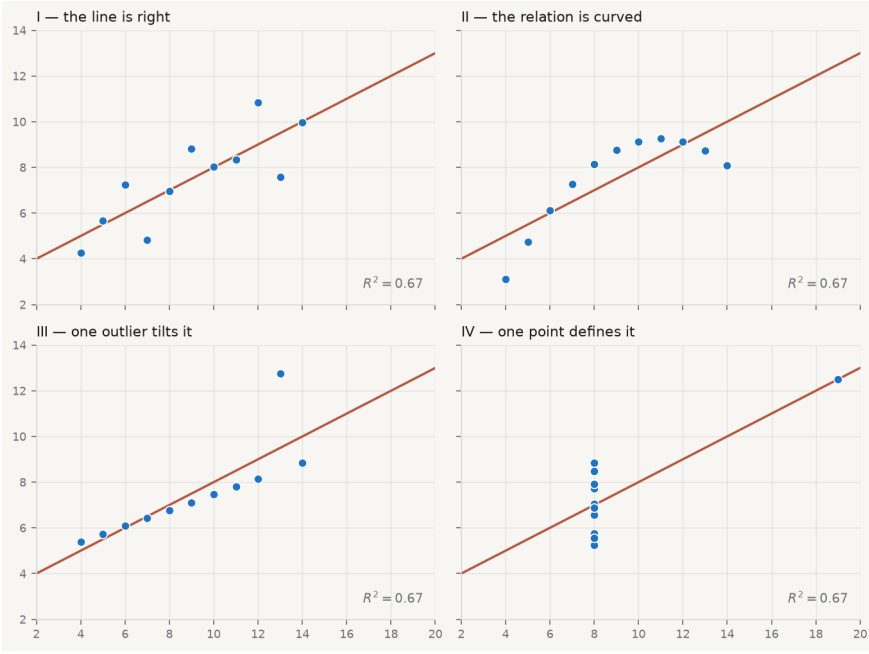


Figure 3.2. Anscombe's quartet. Identical means, variances, correlations, fitted lines and R^2 — and only the first is a dataset the line describes. Data from Anscombe (1973), Table 1.

3.3 What the residual knows

Write $\hat{y} = X\hat{\beta}$. Substituting the estimator,

$$\hat{y} \;=\; X(X^\top X)^{-1}X^\top y \;=\; Hy,$$

where $H = X(X^\top X)^{-1}X^\top$ is the *hat matrix* — so named because it puts the hat on y. Geometrically it is the orthogonal projection onto the column space of X: it takes the n-dimensional vector of observations and casts its shadow onto the p-dimensional subspace the model can reach. That picture is worth holding, because it makes the next two properties obvious rather than algebraic. H is symmetric ($H = H^\top$) and idempotent ($HH = H$) — projecting a shadow again does nothing, since it is already flat.

The residual vector is what the projection could not reach,

$$e \;=\; y - \hat{y} \;=\; (I - H)\,y,$$

and it is orthogonal to every column of X by construction. That orthogonality is the reason residual plots are informative *and* the reason they can mislead: e is guaranteed to be uncorrelated with each predictor whether or not the model is right, so a flat residual plot against x is not evidence of anything. What is *not* guaranteed is the absence of non-linear structure, which is exactly what you are looking for.

Now the consequence that governs every diagnostic below. If $\mathrm{Var}(\varepsilon) \;=\; \sigma^2 I$, then

$$\mathrm{Var}(e) \;=\; \sigma^2(I - H), \qquad \text{so} \qquad \mathrm{Var}(e_i) \;=\; \sigma^2\,(1 - h_{ii}).$$

The residuals do not have constant variance even when the errors do. An observation with large h_{ii} has a residual squeezed towards zero as a matter of arithmetic, not because the model fits it well. The most dangerous points therefore look like the best-behaved ones. This is why raw residuals are the wrong thing to plot, and why everything below is built on the *standardised* residual

$$r_i \;=\; \frac{e_i}{s\sqrt{1 - h_{ii}}}, \qquad s^2 = \frac{e^\top e}{n - p},$$

which divides out the arithmetic squeeze and restores comparability. Plotting r_i against $\hat{y}_i$ is the single most informative diagnostic there is (Anscombe and Tukey 1963). Under an adequate model the cloud is structureless: flat, centred on zero, of even width. Curvature means the functional form is wrong. A widening fan means the variance is not constant. Both are visible instantly and neither is visible in R^2.

Why plot against $\hat{y}$ rather than against x? Because with several predictors there is no single x to use, and $\hat{y}$ is the one-dimensional summary the model actually commits to. It is also the axis along which heteroscedasticity usually manifests, since the spread of an economic quantity so often scales with its level.

```
import pandas as pd, statsmodels.api as sm
from statsmodels.nonparametric.smoothers_lowess import lowess
import qmib

d = qmib.load("core").dropna(subset=["gdp_pc_eur", "productivity_idx"]).reset_
X = sm.add_constant(d["productivity_idx"])
model = sm.OLS(d["gdp_pc_eur"], X).fit()
infl = model.get_influence()
std_resid = infl.resid_studentized_internal
fitted = model.fittedvalues.values

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9.2, 3.9))

ax1.axhline(0, color=CL.muted, lw=1, zorder=1)
ax1.scatter(fitted, std_resid, s=16, color=CL.accent, alpha=0.5, edgecolor="nor
sm_line = lowess(std_resid, fitted, frac=0.5, return_sorted=True)
ax1.plot(sm_line[:, 0], sm_line[:, 1], color=CL.warn, lw=1.9, zorder=4)
ax1.set_xlabel("fitted value"); ax1.set_ylabel("standardised residual")
ax1.set_title("Residuals vs fitted", fontsize=10, loc="left")

root = np.sqrt(np.abs(std_resid))
ax2.scatter(fitted, root, s=16, color=CL.accent, alpha=0.5, edgecolor="none", z
sm2 = lowess(root, fitted, frac=0.5, return_sorted=True)
ax2.plot(sm2[:, 0], sm2[:, 1], color=CL.warn, lw=1.9, zorder=4)
ax2.set_xlabel("fitted value"); ax2.set_ylabel(r"$\sqrt{|\,r_i\,|}$")
ax2.set_title("Scale–location", fontsize=10, loc="left")

fig.tight_layout()
```

Read Figure 3.3 the way a referee would, which means saying what you looked for before saying what you saw. In the left panel you are looking for curvature in the smoother and for a cloud of even vertical width; the smoother drifts rather than sitting flat, which says a straight line is missing some of the shape of the relationship between productivity and income. In the right panel you are looking for a trend, and there is one: the spread of the residuals grows with the fitted value: the wealthier the country-year, the wider the spread of what the model gets wrong. Whether that is true of Europe is not something these fixtures can tell you — but it is true of this dataset, and it is what the standard errors below have to contend with.

Neither finding is fatal. Both change what you are allowed to say — the first about the functional form, the second about the standard errors — and the rest of this chapter is about what to do with each.

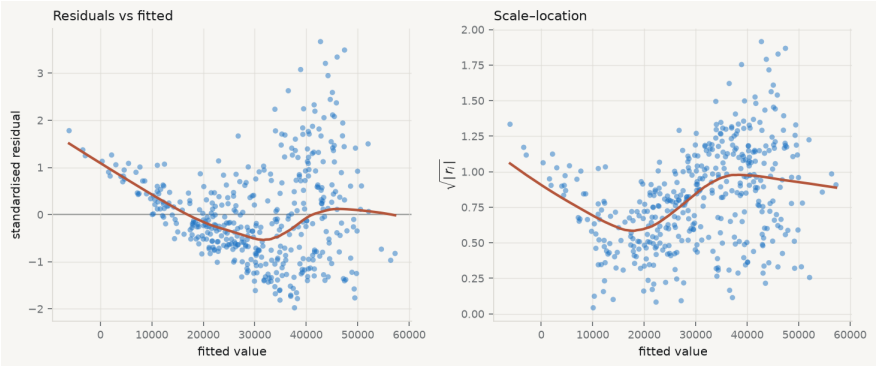


Figure 3.3. Diagnostics for GDP per capita on the productivity index — the regression Session 02 fitted. Left: standardised residuals against fitted values, with a LOWESS smoother (Cleveland 1979). Right: the scale–location plot, which removes the sign so that changing spread is easier to see.

3.4 Goodness of fit, and what it cannot tell you

Before the diagnostics, the summary statistics they are meant to supplement. The total variation in y splits into the part the model accounts for and the part it does not,

$$\underbrace{\sum_i (y_i - \bar{y})^2}_{\text{TSS}} \;=\; \underbrace{\sum_i (\hat{y}_i - \bar{y})^2}_{\text{ESS}} \;+\; \underbrace{\sum_i e_i^2}_{\text{RSS}}, \qquad R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}.$$

That decomposition holds *only* because the residuals are orthogonal to the fitted values, which is a property of least squares with an intercept and not a law of nature. Fit without an intercept and R^2 can be negative.

Its defect is the one Anscombe exploited: R^2 cannot fall when a predictor is added, however irrelevant, because the extra column can always be given a coefficient of zero. Two corrections are standard.

Adjusted R^2 — R^2 with a penalty for the parameters spent:

$$R^2_{\text{adj}} \;=\; 1 - \frac{\text{RSS}/(n-p)}{\text{TSS}/(n-1)}.$$

It can decrease when a useless variable is added, which is the point. It is still not a model-selection criterion — see AIC below.

Residual standard error — the typical size of a residual, in the units of y:

$$\text{RSE} \;=\; s \;=\; \sqrt{\frac{\text{RSS}}{n-p}}.$$

Unlike R^2 it is not unitless, which makes it the more honest number to report: "typically wrong by 6,000 euros" means something, where "$R^2 = 0.63$" means something only relative to the variance of this particular sample.

```
model = sm.OLS(d["gdp_pc_eur"], sm.add_constant(d["productivity_idx"])).fit()
n, p = int(model.nobs), int(model.df_model) + 1
resid = model.resid.to_numpy()

rss = (resid ** 2).sum()
tss = ((d["gdp_pc_eur"] - d["gdp_pc_eur"].mean()) ** 2).sum()

r2_hand = 1 - rss / tss
adj_hand = 1 - (rss / (n - p)) / (tss / (n - 1))
rse_hand = np.sqrt(rss / (n - p))

print(f"{'':14}{'by hand':>14}{'statsmodels':>14}")
print(f"{'R²':14}{r2_hand:14.6f}{model.rsquared:14.6f}")
print(f"{'adjusted R²':14}{adj_hand:14.6f}{model.rsquared_adj:14.6f}")
print(f"{'RSE':14}{rse_hand:14.4f}{np.sqrt(model.mse_resid):14.4f}")
print(f"\nTypical miss: {rse_hand:,.0f} euros of GDP per capita, "
      f"against a mean of {d['gdp_pc_eur'].mean():,.0f}.")
```

```
                    by hand   statsmodels
R²                 0.627259      0.627259
adjusted R²        0.626384      0.626384
RSE              9376.0020     9376.0020

Typical miss: 9,376 euros of GDP per capita, against a mean of 30,403.
```

Note what the last line does that $R^2 = 0.63$ does not: it puts the error in euros, next to the quantity being predicted, where a reader can judge whether it is tolerable for the decision at hand.

3.5 Leverage: which observations *can* move the line

> **Leverage** — the diagonal element h_{ii} of the hat matrix, equal to $\partial \hat{y}_i / \partial y_i$: how much the fitted value at observation i responds to a change in the observed value at i. Leverage is a property of X alone. It does not involve y at all, so an observation can have high leverage and fit perfectly.

That derivative reading is the one to carry. If $h_{ii} = 0.9$, then moving that observation's y up by one unit drags its own fitted value up by 0.9 — the line follows it almost exactly, which is another way of saying the line is not being constrained by anything else in that region.

Because H projects onto a p-dimensional space, $\sum_i h_{ii} = \operatorname{tr}(H) = p$, so leverage is a fixed budget of p shared among n observations. The average is exactly p/n, and each h_{ii} lies in $[0, 1]$. The conventional flag is $h_{ii} > 2p/n$ — twice the average — which is a rule of thumb and nothing more (Hoaglin and Welsch 1978). In simple regression it has a transparent form,

$$h_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_j (x_j - \bar{x})^2},$$

which says exactly what leverage means: a floor of $1/n$ that everyone gets, plus a term that grows with squared distance from the centre of the predictor space, measured in units of the predictors' own spread. Two consequences follow directly. Leverage is minimised at $x_i = \bar{x}$ and grows quadratically away from it. And it is *relative* — the same observation is high-leverage in a narrow sample and unremarkable in a wide one, so leverage is a statement about the design, not about the observation.

```
infl = model.get_influence()

# h_ii is the diagonal of X(X'X)^-1 X'. Forming the full n x n hat matrix to
# read its diagonal would be wasteful; einsum takes the diagonal directly.
Xd = sm.add_constant(d["productivity_idx"]).to_numpy()
XtX_inv = np.linalg.inv(Xd.T @ Xd)
h_hand = np.einsum("ij,jk,ik->i", Xd, XtX_inv, Xd)

s = np.sqrt(rss / (n - p))
r_hand = resid / (s * np.sqrt(1 - h_hand))

print(f"leverage    max |hand - statsmodels| = "
      f"{np.abs(h_hand - infl.hat_matrix_diag).max():.2e}")
print(f"std. resid  max |hand - statsmodels| = "
      f"{np.abs(r_hand - infl.resid_studentized_internal).max():.2e}")
print(f"\nsum of h_ii = {h_hand.sum():.4f}, and p = {p}  ")
```

```
      f"(they are equal by construction)")
print(f"average leverage p/n = {p/n:.5f};  "
      f"{(h_hand > 2*p/n).sum()} observations exceed the 2p/n convention")
```

```
leverage    max |hand - statsmodels| = 5.10e-16
std. resid  max |hand - statsmodels| = 4.44e-16

sum of h_ii = 2.0000, and p = 2  (they are equal by construction)
average leverage p/n = 0.00467;  27 observations exceed the 2p/n convention
```

The check on the second-to-last line is worth pausing on. $\sum_i h_{ii} = p$ is not an empirical finding about this dataset; it is the trace of a projection onto a p-dimensional space, and it must hold exactly for every regression you ever fit. If your own implementation does not reproduce it, the implementation is wrong.
The fourth Anscombe panel is the extreme case: one point at $x = 19$ among ten at $x = 8$ has leverage close to 1, and the line has no choice but to pass through it. Leverage is *potential*. It says an observation is positioned to move the line. Whether it actually does depends on whether its y is surprising, and that requires combining leverage with the residual.

3.6 Influence: separating the outlier from the point that matters

Cook's distance — how much the entire vector of fitted values moves when observation i is deleted, scaled to be comparable across observations and models:

$$D_i \;=\; \frac{(\hat{y} - \hat{y}_{(i)})^\top (\hat{y} - \hat{y}_{(i)})}{p\, s^2} \;=\; \frac{r_i^2}{p} \cdot \frac{h_{ii}}{1 - h_{ii}}.$$

The second form is the one to internalise (Cook 1977). Cook's distance is a *product* of two quantities: how badly the point is fitted, r_i^2, and how much power it has to pull, $h_{ii}/(1 - h_{ii})$. Because it is a product, either factor being near zero makes the whole thing near zero — which is why the two things students conflate are genuinely different:

- **High residual, low leverage** — an outlier in the middle of the predictor range. It inflates s and therefore widens every standard error in the model, but it barely moves $\hat{\beta}$, because it has no lever to pull on.
- **Low residual, high leverage** — a point far out in X that the line happens to pass through. It is *supporting* your slope rather than distorting it, and no misfit diagnostic will flag it. But your slope now rests on it, which is a robustness problem precisely because nothing looks wrong.
- **High on both** — the case that changes conclusions, and what Cook's distance is built to find.

```
# D_i = (r_i^2 / p) * (h_ii / (1 - h_ii)) - the product form from the defini
cook_hand = (r_hand ** 2 / p) * (h_hand / (1 - h_hand))
print(f"Cook's D    max |hand - statsmodels| = "
      f"{np.abs(cook_hand - infl.cooks_distance[0]).max():.2e}")

worst = int(np.argmax(cook_hand))
print(f"\nlargest D_i = {cook_hand[worst]:.4f}  "
      f"({d.loc[worst, 'geo']} {int(d.loc[worst, 'time'])})")
print(f"  its residual  r = {r_hand[worst]:+.2f}")
print(f"  its leverage  h = {h_hand[worst]:.4f}  "
      f"({h_hand[worst] / (p/n):.1f}x the average)")
print(f"  conventional flags: D > 1 → {(cook_hand > 1).sum()}, "
      f"D > 4/n → {(cook_hand > 4/n).sum()}")
```

```
Cook's D    max |hand - statsmodels| = 8.74e-16

largest D_i = 0.0425  (DK 2010)
  its residual  r = +3.49
  its leverage  h = 0.0069  (1.5x the average)
  conventional flags: D > 1 → 0, D > 4/n → 25
```

Note how the leverage factor behaves: $h/(1-h)$ is 0.11 at $h = 0.1$, but 9 at $h = 0.9$. Influence does not rise gently with leverage, it explodes as h_{ii} approaches 1. Common cutoffs are $D_i > 1$ or $D_i > 4/n$; both are conventions, and Belsley, Kuh, and Welsch (1980) is properly sceptical about treating any of them as a test. The useful discipline is not the threshold. It is that you *look*, you *name* the observations that stand out, and you report what happens to your conclusion with and without them. Deleting an influential point silently is misconduct. Reporting that Luxembourg is influential, and that your slope falls by a third without it, is a finding — and often a more interesting one than the coefficient.

```
lev = infl.hat_matrix_diag
cooks = infl.cooks_distance[0]
n, p = int(model.nobs), int(model.df_model) + 1
thresh = 4 / n

fig, (axL, axR) = plt.subplots(1, 2, figsize=(9.2, 4.0))

def influence_panel(ax, lev, resid, cooks, title, hi=None):
    ax.axhline(0, color=CL.muted, lw=1)
    ax.scatter(lev, resid, s=18 + 900 * np.clip(cooks, 0, None), color=CL.ac
               alpha=0.45, edgecolor="white", linewidth=0.6, zorder=3)
    gl = np.linspace(max(lev.min(), 1e-4), lev.max() * 1.05, 200)
    for sign in (1, -1):
        ax.plot(gl, sign * np.sqrt(thresh * p * (1 - gl) / gl),
```

```
                ls="--", lw=1.1, color=CL.warn, zorder=2)
    if hi is not None:
        ax.scatter([lev[hi]], [resid[hi]], s=95, facecolor="none",
                   edgecolor=CL.warn, linewidth=1.8, zorder=5)
    ax.set_xlabel("leverage $h_{ii}$"); ax.set_ylabel("standardised residual")
    ax.set_title(title, fontsize=10, loc="left"); ax.set_xlim(left=0)

influence_panel(axL, lev, std_resid, cooks, f"As fitted — max $D_i$ = {cooks.ma

d2 = d.copy()
j = int(d2["productivity_idx"].idxmax())
d2.loc[j, "productivity_idx"] = d2["productivity_idx"].max() * 2.1
d2.loc[j, "gdp_pc_eur"] = d2["gdp_pc_eur"].min()
m2 = sm.OLS(d2["gdp_pc_eur"], sm.add_constant(d2["productivity_idx"])).fit()
i2 = m2.get_influence()
influence_panel(axR, i2.hat_matrix_diag, i2.resid_studentized_internal,
                i2.cooks_distance[0],
                f"One point moved — max $D_i$ = {i2.cooks_distance[0].max():.2

fig.tight_layout()
```

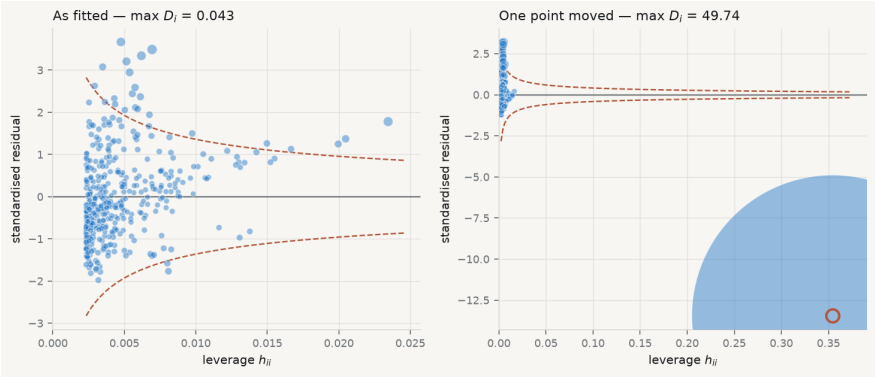


Figure 3.4. Left: the influence plot for the fitted regression — standardised residual against leverage, point area proportional to Cook's distance, dashed contours at $D = 4/n$. Right: the same data with one observation moved to a high-leverage, poorly-fitted position. The right panel is a constructed illustration, not a feature of the data.

The left panel of Figure 3.4 is what a clean diagnostic looks like: maximum Cook's distance around 0.04, every point inside the contours, no observation carrying the result. That is a reportable finding in its own right and worth stating explicitly rather than leaving as an absence — "no observation had Cook's distance above

0.05" is a sentence a referee can check. The right panel moves one observation to high leverage and poor fit; its Cook's distance jumps by two orders of magnitude and the slope moves with it. Nothing about R^2 warns you which panel you are in.

3.7 Normality, and why it matters least

Of the four assumptions this is the one students check first and the one that matters least. It is not needed for unbiasedness, not needed for Gauss–Markov, and not needed for the standard errors to be right. It is needed for the t and F statistics to follow exactly the distributions their tables assume — and only in small samples, because the central limit theorem takes over as n grows. $\hat{\beta}$ is a weighted sum of the y_i, and sums tend to normality whatever they are sums of.

The right check is a normal quantile–quantile plot: sort the standardised residuals, plot them against the quantiles a normal sample of that size would be expected to produce, and look for departure from the diagonal. What you are looking for is not perfection but the *shape* of the departure. Points curving above the line at the right and below at the left mean heavy tails — more extreme observations than a normal would give — which is a robustness concern, since least squares weights those heavily. A systematic S-shape means skew.

Formal tests exist (Shapiro and Wilk 1965), and their weakness is instructive: with $n = 428$ a Shapiro–Wilk test will reject normality on a departure far too small to affect any inference, while with $n = 20$ it will fail to reject a departure large enough to matter. This is the general pathology of diagnostic testing — the null is never exactly true, so the test measures sample size as much as misspecification — and it is the reason this chapter keeps returning to plots.

```
from scipy import stats

fig, ax = plt.subplots(figsize=(5.4, 4.2))
osm, osr = stats.probplot(std_resid, dist="norm", fit=False)
ax.plot([-3.4, 3.4], [-3.4, 3.4], color=CL.warn, lw=1.5, zorder=2)
ax.scatter(osm, osr, s=15, color=CL.accent, alpha=0.6, edgecolor="none", zor
ax.set_xlabel("theoretical normal quantile")
ax.set_ylabel("standardised residual")
ax.set_title("Normal Q–Q", fontsize=10, loc="left")
fig.tight_layout()
```

3.8 When the variance is not constant

Heteroscedasticity is the failure most often misdiagnosed, so be precise about what it costs. Under heteroscedasticity $\hat{\beta}$ remains **unbiased** and **consistent**. The coefficients are not wrong. What breaks is $\mathrm{Var}(\hat{\beta})$: the usual $s^2(X^\top X)^{-1}$ is no longer the right formula, so the standard errors, t statistics, confidence intervals and p-

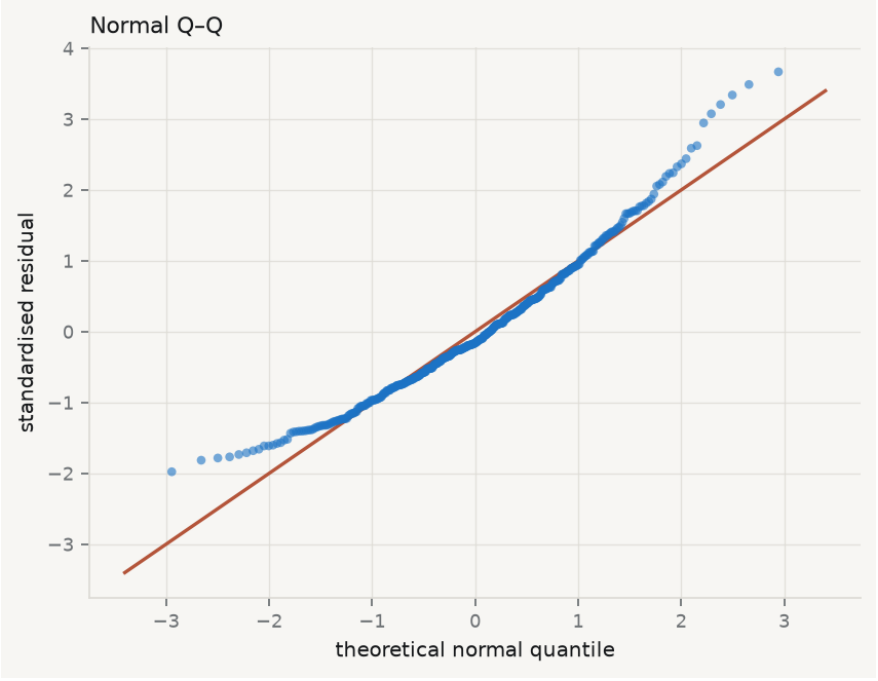


Figure 3.5. Normal quantile–quantile plot of the standardised residuals. Departure at both ends, in opposite directions, is the signature of heavy tails rather than skew.

values are all computed from an expression that does not apply. You have the right answer with the wrong uncertainty attached — which, if the question is whether an effect is distinguishable from zero, is the part that matters.

The formal test regresses the squared residuals on the predictors and asks whether they explain anything (Breusch and Pagan 1979). Worth running, but the scale–location plot usually tells you first, and the test carries the same sample-size pathology as every other.

The modern response is not to model the variance but to stop relying on the assumption. White's heteroscedasticity-consistent estimator replaces the misapplied middle of the sandwich with the squared residuals themselves (White 1980):

$$\widehat{\mathrm{Var}}(\hat{\beta}) = (X^\top X)^{-1}\Big(\textstyle\sum_i e_i^2\, x_i x_i^\top\Big)(X^\top X)^{-1}.$$

```
# The meat of the sandwich: sum_i e_i^2 x_i x_i', which is X' diag(e^2) X.
bread = np.linalg.inv(Xd.T @ Xd)
meat = Xd.T @ (Xd * (resid ** 2)[:, None])
V_hc0 = bread @ meat @ bread
se_hc0_hand = np.sqrt(np.diag(V_hc0))

fits = {
    "classical": model,
    "HC0 (White)": model.get_robustcov_results(cov_type="HC0"),
    "HC3 (use this)": model.get_robustcov_results(cov_type="HC3"),
}
print(f"{'':18}{'slope SE':>12}{'t':>9}")
for name, f in fits.items():
    # params is a Series for the pandas fit and an array for the robust ones
    # so index positionally through numpy rather than by label.
    b, se = np.asarray(f.params)[1], np.asarray(f.bse)[1]
    print(f"{name:18}{se:12.3f}{b/se:9.2f}")
se_hc0_sm = np.asarray(fits["HC0 (White)"].bse)[1]
print(f"\nHC0 by hand      {se_hc0_hand[1]:12.3f}   "
      f"(matches statsmodels to {abs(se_hc0_hand[1] - se_hc0_sm):.1e})")
```

```
                    slope SE        t
classical             41.504    26.77
HC0 (White)           42.854    25.93
HC3 (use this)        43.239    25.70

HC0 by hand            42.854   (matches statsmodels to 4.6e-12)
```

Look at what that does. The classical formula assumes every observation contributes the same σ^2 to the middle; the sandwich lets each contribute its own e_i^2. It is consistent whatever the pattern of the variance, without your having to know or

model that pattern — which is why it is called *robust* rather than *efficient*. Its small-sample behaviour is poor, because e_i^2 underestimates the variance at high-leverage points, and the corrections HC1, HC2 and HC3 exist to compensate (MacKinnon and White 1985). The practical recommendation is settled: **use HC3 by default**, certainly whenever $n < 250$ (Long and Ervin 2000). In `statsmodels` that is `fit(cov_type="HC3")` — one argument, no good reason to omit it.

Non-constant variance is not the only specification failure worth testing. The RESET procedure adds powers of the fitted values back into the regression and asks whether they matter (Ramsey 1969); if $\hat{y}^2$ has explanatory power, some non-linearity remains unmodelled.

3.9 The assumption nobody plots

Independence has no standard diagnostic plot, which is most of why it goes unexamined — and in this dataset it is the assumption that actually fails.

The `core` table is not 428 independent observations. It is 30 countries observed over 15 years. Germany in 2014 and Germany in 2015 are not two independent draws from anything: whatever makes German GDP per capita high in one year makes it high in the next. If the errors within a country are positively correlated, the sample carries far less information than its row count suggests, and the standard errors — which are computed from that row count — come out far too small.

```
import scipy.stats as st

resid = pd.Series(model.resid.values, index=d.index)
by_country = d.assign(r=resid).sort_values(["geo", "time"]).groupby("geo")["r"]
rho = by_country.apply(lambda s: s.autocorr(1)).dropna()

fits = {
    "Classical OLS": sm.OLS(d["gdp_pc_eur"], X).fit(),
    "HC3 (heteroscedasticity)": sm.OLS(d["gdp_pc_eur"], X).fit(cov_type="HC3"),
    "Clustered by country": sm.OLS(d["gdp_pc_eur"], X).fit(
        cov_type="cluster", cov_kwds={"groups": d["geo"]}),
}
rows = []
for name, f in fits.items():
    b, se = f.params.iloc[1], f.bse.iloc[1]
    rows.append(f"| {name} | {b:,.1f} | {se:,.1f} | {b/se:.1f} | ±{1.96*se:,.0

print(f"Median within-country lag-1 residual autocorrelation: "
      f"**{rho.median():.2f}** across {len(rho)} countries.\n")
print("| Standard errors | Coefficient | SE | $t$ | 95% CI half-width |")
print("|---|---|---|---|---|")
print("\n".join(rows))
```

Median within-country lag-1 residual autocorrelation: **0.76** across 30 countries.

Table 3.2

Standard errors	Coefficient	SE	t	95% CI half-width
Classical OLS	1,111.3	41.5	26.8	±81
HC3 (heteroscedasticity)	1,111.3	43.2	25.7	±85
Clustered by country	1,111.3	72.3	15.4	±142

The numbers make the point better than any argument. The coefficient is identical in all three rows — as promised, dependence does not bias it. The classical standard error and the HC3 standard error are nearly the same, because HC3 fixes heteroscedasticity and this is not heteroscedasticity. Clustering by country raises the standard error by about three quarters, and the confidence interval widens accordingly. Had the finding been marginal, the classical version would have declared significance that the clustered version withdraws.

This is Moulton's warning (Moulton 1990), and it is the single most consequential correction in applied panel work. Bertrand, Duflo, and Mullainathan (2004) showed the same mechanism producing spurious significance in difference-in-differences studies at alarming rates; Colin Cameron and Miller (2015) is the practitioner's reference for what to cluster on and when the cluster count is too small to trust. The rule of thumb is to cluster at the level at which treatment is assigned or shocks arrive — here, the country. For pure time-series dependence, the classical test is Durbin–Watson (Durbin and Watson 1950), which detects lag-1 autocorrelation in ordered residuals.

You will meet clustering properly in Session 11. The point here is that a diagnostic chapter that stopped at residual plots would have missed the largest inferential problem in its own running example, because the largest problem does not show up in a residual plot at all. **Some assumptions are checked by looking at the data. Independence is checked by knowing how the data were collected.**

3.10 Comparing models: what AIC is, and what it is not

Having diagnosed a problem you will usually fit an alternative — a quadratic term, a log transform, an extra control — and need a basis for preferring one. R^2 cannot supply it, because adding any variable, however irrelevant, cannot decrease R^2.

Akaike's criterion comes at it from prediction rather than fit. It estimates the expected Kullback–Leibler divergence between the fitted model and the process that generated the data, and reduces to

$$\mathrm{AIC} \;=\; -2\log \hat{L} + 2k$$

for a model with k estimated parameters and maximised likelihood $\hat{L}$ (Akaike 1974). Be careful with k: the variance σ^2 was estimated from the data too, so a

strict count includes it, while `statsmodels` counts only the regression coefficients for OLS. The two differ by a constant — 2 in AIC — which is invisible in any comparison and alarming the first time you find your own calculation disagreeing with the library. The cell below shows both. The first term rewards fit, the second charges two units per parameter, and lower is better. The Bayesian criterion replaces the penalty with $k \log n$, harsher for any $n > 7$, and aims at a different target — the true model, if one is among the candidates, rather than the best predictor (Schwarz 1978). They disagree by construction, and which you use should follow from which question you are asking.

```
# For Gaussian OLS the maximised log-likelihood has a closed form.
loglik = -0.5 * n * (np.log(2 * np.pi) + np.log(rss / n) + 1)
print(f"log-likelihood: hand {loglik:.4f}  statsmodels {model.llf:.4f}")

# How many parameters were estimated? Two coefficients — and sigma^2, which was
# also estimated from the data. Counting it is the stricter reading; statsmodel
# does not, for OLS. The two conventions differ by a constant.
for k, label in ((p, "k = p     (statsmodels)"), (p + 1, "k = p + 1 (counting σ
    print(f"{label:26} AIC {-2*loglik + 2*k:10.3f}   BIC {-2*loglik + k*np.log(
print(f"{'statsmodels reports':26} AIC {model.aic:10.3f}   BIC {model.bic:10.3f
print(f"\nThe gap is 2 in AIC and log(n) = {np.log(n):.2f} in BIC — one paramet
print("worth. It is the same for every model fitted to these rows, so it cancel
print("in any difference, which is the only way AIC is ever read.")

# A comparison must use the same rows, so build the candidates on one frame.
cand = d.assign(prod2=d["productivity_idx"] ** 2)
models = {
    "linear":        ["productivity_idx"],
    "with quadratic": ["productivity_idx", "prod2"],
}
fitted = {name: sm.OLS(cand["gdp_pc_eur"], sm.add_constant(cand[cols])).fit()
          for name, cols in models.items()}
best = min(f.aic for f in fitted.values())
print(f"\n{'':18}{'AIC':>12}{'Δ':>8}")
for name, f in fitted.items():
    print(f"{name:18}{f.aic:12.1f}{f.aic - best:8.1f}")
print("\nΔ under about 2 is weak evidence of any difference at all.")
```

```
log-likelihood: hand -4520.7523  statsmodels -4520.7523
k = p     (statsmodels)    AIC   9045.505   BIC   9053.623
k = p + 1 (counting σ²)    AIC   9047.505   BIC   9059.682
statsmodels reports        AIC   9045.505   BIC   9053.623

The gap is 2 in AIC and log(n) = 6.06 in BIC — one parameter's
worth. It is the same for every model fitted to these rows, so it cancels
```

```
in any difference, which is the only way AIC is ever read.

                          AIC       Δ
linear                 9045.5    41.5
with quadratic         9004.0     0.0

Δ under about 2 is weak evidence of any difference at all.
```

Three cautions, in order of how often they are violated.
First, **AIC is comparative and unitless**. A single AIC value means nothing. Only differences between models fitted to *the same observations* are interpretable — so dropping rows through missing data silently invalidates the comparison, a mistake that is easy to make and invisible afterwards. By convention $\Delta_i = \mathrm{AIC}_i - \mathrm{AIC}_{\min}$ below about 2 is weak evidence of any difference (Burnham and Anderson 2004).
Second, **AIC does not check adequacy**. It ranks the candidates you supply. If every model you supply is misspecified it will rank them and hand you a winner, in the same confident tone it uses when one of them is right. It is a comparison, not a diagnostic, and no substitute for the plots.
Third — the one that ends careers — **selecting a model and then reporting its p-values as though the specification had been chosen in advance is invalid.** Freedman demonstrated the scale of the problem with pure noise: regress a random y on 50 random predictors, keep the significant ones, refit, and the second regression looks impressive (Freedman 1983). Nothing was there. The same mechanism operates whenever the data guide the specification and the resulting p-values are reported as if they had not — the "garden of forking paths", which requires no dishonesty at all, only the ordinary practice of looking at your data before deciding what to fit (Gelman and Loken 2014). If you select on AIC, say so, and treat the resulting inference as exploratory.

> **Stepwise selection is not here.** The deck this chapter grew from covered it alongside AIC, and it does not belong in a chapter on diagnostics: it is not a check on a fitted model but a procedure for choosing one. It opens chapter 5 instead, where it does real work — the first, almost mechanical attempt to automate variable selection, whose instability under resampling is precisely the argument for shrinking coefficients continuously rather than including or excluding them.

3.11 When a diagnostic fails, what to actually do

Diagnosis without a remedy is just anxiety. The response depends entirely on which assumption failed, which is why the table at the start of this chapter matters.
Curvature in the residuals — the functional form is wrong, and this is the only failure that biases the coefficient, so it is the only one you must fix rather than accommodate. Add a quadratic term, take logs of a variable whose effect is plau-

sibly proportional rather than additive, or fit a spline. Logs are the usual answer in economics because so many relationships are multiplicative; G. E. P. Box and Cox (1964) gives the systematic treatment. Report that you transformed, and why, before you report the result.

Non-constant variance — do not transform to fix it, since that changes the quantity you are modelling in order to repair its error term. Use HC3 standard errors and say so.

Dependence — cluster at the level at which the dependence arises, or model it explicitly with fixed effects or a panel estimator.

Heavy tails or an influential point — first identify the observation and ask what it is. An influential point is frequently the most informative observation in the dataset rather than an error: Luxembourg's productivity figures are strange because of cross-border commuting, which is a fact about the economy, not a data problem. Report the result with and without it. Delete only for a reason you can state in the text, and state it there.

Everything at once — reconsider whether one linear model over pooled heterogeneous units is the right object. Often it is not, and the honest answer is a different specification rather than a patched one.

3.12 A short review of the literature

The diagnostic tradition begins with Anscombe and Tukey (1963), who argued that residuals should be examined systematically rather than summarised, and set out the plots still in use. Anscombe (1973) made the case unforgettable with four datasets constructed to share every conventional summary while differing completely — a demonstration reproduced in Figure 3.2 and in nearly every regression text since. The smoother that makes such plots readable is Cleveland (1979).

The geometry was formalised in the 1970s. Hoaglin and Welsch (1978) gave the hat matrix its expository treatment and the $2p/n$ convention. Cook (1977) introduced the distance that separates outlying from influential observations, and Belsley, Kuh, and Welsch (1980) assembled the apparatus into a book-length treatment, with a scepticism about mechanical cutoffs that has aged well.

Robust inference developed in parallel. White (1980) provided a covariance estimator consistent under arbitrary heteroscedasticity, complementing the earlier Breusch and Pagan (1979) test. MacKinnon and White (1985) showed White's estimator behaves badly in small samples and proposed corrections; Long and Ervin (2000) evaluated them and recommended HC3, now the applied default. Specification error more broadly is treated by Ramsey (1969), and transformation by G. E. P. Box and Cox (1964).

Dependence has a separate and more consequential literature. Durbin and Watson (1950) gave the classical serial-correlation test; Moulton (1990) showed how badly standard errors mislead when observations cluster within groups; Bertrand, Duflo, and Mullainathan (2004) demonstrated the same failure producing spurious significance in difference-in-differences at alarming rates; and Colin Cameron and Miller (2015) is the practical reference for cluster-robust inference and its limits.

Model comparison follows Akaike (1974) and Schwarz (1978), whose criteria differ in penalty and in target — prediction against identification — a distinction Burnham and Anderson (2004) makes carefully and much applied work does not. The cost of ignoring it was quantified early by Freedman (1983) and framed for a modern audience by Gelman and Loken (2014): inference conditional on a specification chosen from the data is not the inference the p-value describes.

The through-line, from 1963 to now, is a single claim — that a fitted model is a hypothesis about the data, not a summary of it, and that the residuals are where that hypothesis is tested.

3.13 What to report

A diagnostic section that earns its place answers five questions in order, and takes about a page.

1. **Is the functional form right?** Residuals against fitted, with a smoother. Say what structure you looked for and whether you found it.
2. **Is the variance constant?** Scale–location. If not, report HC3 standard errors and say that you did — do not quietly switch and leave the reader to notice the numbers moved.
3. **Are the observations independent?** This one you answer from how the data were collected, not from a plot. If they cluster, cluster the standard errors and report both.
4. **Does any observation carry the result?** Leverage against standardised residual, sized by Cook's distance. Name what stands out, identify it substantively — Luxembourg, Ireland, 2020 — and give the coefficient with and without it.
5. **Would a different specification do better?** Compare on AIC over the same rows, report Δ, and state plainly whether the specification was chosen before or after seeing the data.

The failure mode this chapter exists to prevent is reporting R^2 and stopping. The four datasets in Figure 3.2 all have $R^2 = 0.67$. Three of them should never be described by a straight line, and no amount of staring at 0.67 will tell you which three.

Chapter 4

Logistic Regression: Binary, Ordinal, and Multinomial

4.1 When the outcome is a category

Everything so far has predicted a number. Most decisions are not numbers. Does this borrower default. Does this firm export. Does this country adopt the regulation. The outcome is a category, and the machinery of chapters 2 and 3 does not transfer intact.

The obvious move is to code the outcome as 0 and 1 and fit least squares to it anyway. That is the *linear probability model*, and it is not absurd — the fitted values estimate $\mathrm{E}[y \mid x]$, which for a binary variable is exactly $\Pr(y = 1 \mid x)$, and the coefficients read directly as changes in probability, which is what people want. It is used in serious applied work.

It also breaks in three specific ways, and being precise about which matters.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, statsmodels.api as sm, qmib

loans = qmib.load("loans")
y = loans["default"]
X = sm.add_constant(loans[["fico", "int_rate", "log_annual_inc", "dti"]])

lpm = sm.OLS(y, X).fit()
p_hat = lpm.fittedvalues

print(f"n = {len(loans):,}   defaults = {y.sum():,}   base rate = {y.mean():
print(f"\nfitted probabilities from the linear model:")
print(f"  minimum {p_hat.min():+.4f}")
print(f"  maximum {p_hat.max():+.4f}")
outside = ((p_hat < 0) | (p_hat > 1)).sum()
print(f"  outside [0, 1]: {outside} of {len(p_hat):,} ({outside/len(p_hat):.
```

```
n = 9,578   defaults = 1,533   base rate = 16.01%

fitted probabilities from the linear model:
  minimum -0.0283
```

```
maximum +0.3475
outside [0, 1]: 7 of 9,578 (0.07%)
```

First, it can return probabilities that are not probabilities. Seven fitted values here are negative. That is a small number, and it is worth resisting the textbook temptation to treat it as devastating: at a base rate of 16% with predictors of modest strength, most fitted values land comfortably inside the interval. The failure becomes severe when the base rate is near 0 or 1, or when a predictor takes values outside the range it was fitted on — that is, exactly when you extrapolate.

Second, the error variance cannot be constant. If y_i is Bernoulli with probability p_i, then $\text{Var}(y_i) = p_i(1 - p_i)$ by construction. The variance is a *function of the mean*, largest at $p = 0.5$ and vanishing at either end. Heteroscedasticity is not a defect of the data here; it is a property of binary outcomes, and no amount of clean data removes it. The coefficients remain unbiased — chapter 3's table applies — but the classical standard errors are wrong, always, and robust standard errors are mandatory rather than optional.

Third, and most importantly, the functional form is wrong at the ends. A straight line says a fixed change in a predictor moves the probability by a fixed amount, whether you start at 0.02 or at 0.50. That is not how probabilities behave. Moving from near-certain-not to slightly-possible is a large change in the underlying propensity; moving from 0.50 to 0.55 is a small one. The line has no way to express that.

4.2 The logistic function, and where it came from



Pierre François Verhulst · 1804–1849
Named the logistic curve in 1845, modelling how populations stop growing.
Flameng, Léopold, 1850. Public domain, via Wikimedia Commons.

From the archive · Verhulst, 1845
Malthus had argued that population grows geometrically while subsistence grows arithmetically, so that catastrophe is arithmetic. Pierre François Verhulst, a Bel-

gian mathematician, thought the premise wrong: unchecked exponential growth describes no actual population, because growth slows as a population approaches what its territory can support.
He wrote down the correction. If growth is proportional both to the current population and to the *remaining room*, the result is a curve that rises exponentially, inflects, and flattens toward a ceiling. He named it the *logistic* curve in 1845.
It arrived in statistics by a different road. Berkson proposed it in the 1940s as the natural transformation for dose-response data — the *logit* — arguing against the then-dominant probit on grounds of tractability and interpretation (Berkson 1944). Cox gave the regression treatment its modern form (Cox 1958). The function Verhulst wrote to describe populations pressing against a limit is now the standard way to model any bounded probability, which is the same mathematical situation: something that cannot exceed a ceiling and slows as it approaches one.

```
z = np.linspace(-6, 6, 400)
sigma = 1 / (1 + np.exp(-z))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9.4, 3.9))

K = 100
ax1.plot(z, K * sigma, lw=2, color=CL.accent)
ax1.axhline(K, ls="--", lw=1.2, color=CL.muted)
ax1.text(-5.6, K * 1.03, "carrying capacity $K$", fontsize=8.5, color=CL.muted
ax1.set_xlabel("time"); ax1.set_ylabel("population")
ax1.set_title("Verhulst, 1845", fontsize=10, loc="left")

ax2.plot(z, sigma, lw=2, color=CL.accent)
for lev in (0, 1):
    ax2.axhline(lev, ls="--", lw=1.2, color=CL.muted)
ax2.plot([0], [0.5], marker="o", ms=7, mfc="none", mec=CL.warn, mew=1.8)
ax2.annotate("steepest at $p=0.5$", (0, 0.5), textcoords="offset points",
             xytext=(12, -6), fontsize=8.5, color=CL.warn)
ax2.set_xlabel(r"linear index $x^\top\beta$"); ax2.set_ylabel(r"$\Pr(y=1)$")
ax2.set_title("the same curve, as a probability", fontsize=10, loc="left")

fig.tight_layout()
```

The model replaces the straight line with that curve. Write the linear index as $\eta_i = x_i^\top \beta$ and set

$$\Pr(y_i = 1 \mid x_i) \;=\; \sigma(\eta_i) \;=\; \frac{1}{1 + e^{-\eta_i}} \;=\; \frac{e^{\eta_i}}{1 + e^{\eta_i}}.$$

The function is bounded strictly between 0 and 1, so the first failure is gone by construction. It is symmetric about $\eta = 0$, where $p = 0.5$ and the slope is steepest,

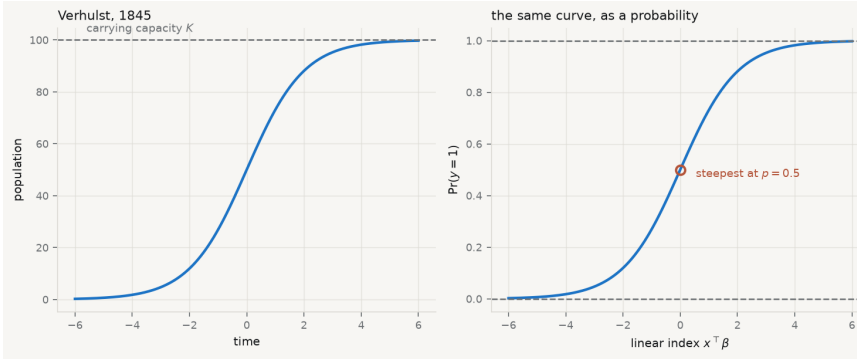


Figure 4.1. The logistic function. Left: as Verhulst wrote it, population against time approaching a carrying capacity. Right: as statistics uses it, a probability against a linear index — the same curve, relabelled.

and it flattens in both tails, so the third failure is gone too: the same change in η moves the probability a great deal in the middle and hardly at all at the extremes. Inverting it gives the form that explains the model's vocabulary:

$$\log \underbrace{\frac{p_i}{1-p_i}}_{\text{odds}} = x_i^\top \beta.$$

The model is linear — in the log-odds. That is the whole trick, and it is why coefficients are not probabilities.

4.3 Estimation: no formula, and why that is fine

Least squares had a closed form. This does not. The log-likelihood is

$$\ell(\beta) = \sum_i \Big[y_i \log \sigma(\eta_i) + (1-y_i) \log\big(1-\sigma(\eta_i)\big) \Big],$$

and setting its derivative to zero gives $X^\top(y - \sigma(X\beta)) = 0$, which has no algebraic solution because σ is nonlinear in β. The estimate is found numerically, by Newton–Raphson — equivalently, by iteratively reweighted least squares, which is the same computation dressed as a sequence of weighted regressions.

This is less alarming than it sounds. The log-likelihood for logistic regression is globally concave, so there is one maximum and any sensible optimiser finds it. The exception matters and is covered below.

```
logit = sm.Logit(y, X).fit(disp=0)
eta = X.to_numpy() @ logit.params.to_numpy()
p = 1 / (1 + np.exp(-eta))
```

```
# The likelihood, written straight from the definition.
ll_hand = (y * np.log(p) + (1 - y) * np.log(1 - p)).sum()
print(f"log-likelihood by hand  {ll_hand:>12.6f}")
print(f"statsmodels             {logit.llf:>12.6f}")

# At the optimum the score must vanish: X'(y - p) = 0.
score = X.to_numpy().T @ (y.to_numpy() - p)
print(f"\nscore X'(y - p) at the optimum, largest element: {np.abs(score).max(
print("  (zero to numerical tolerance — that is what 'converged' means)")

print(f"\nnull log-likelihood     {logit.llnull:>12.4f}")
print(f"McFadden pseudo-R²      {1 - logit.llf/logit.llnull:>12.5f}")
```

```
log-likelihood by hand  -4068.055780
statsmodels             -4068.055780

score X'(y - p) at the optimum, largest element: 1.04e-10
  (zero to numerical tolerance — that is what 'converged' means)

null log-likelihood       -4212.0203
McFadden pseudo-R²           0.03418
```

The pseudo-R^2 is 0.034, and that number deserves a warning. McFadden's statistic is *not* a proportion of variance explained and does not behave like R^2; values that would be alarming in a linear model are ordinary here. Comparing it across models fitted to different samples, or treating 0.034 as "3.4% of something", is a misreading.

4.4 Reading the coefficients

A logistic coefficient is a change in log-odds per unit of the predictor. Nobody thinks in log-odds. There are three standard translations, in increasing order of usefulness and decreasing order of popularity.

> **Odds ratio** — e^{β_j}, the multiplicative change in the odds per one-unit increase in x_j, holding the rest of the index fixed. Constant across the range of x, which is its convenience and its trap: constant on the odds scale means *not* constant on the probability scale.

> **Marginal effect** — $\partial p/\partial x_j = \beta_j\, p(1-p)$, the change in probability per unit of x_j. It depends on where you are: the same coefficient moves the probability most near $p = 0.5$ and least in the tails. The *average marginal effect* averages it over the observed sample, which is usually the number a reader wants.

> **Predicted probability** — $\sigma(x^\top \hat{\beta})$ evaluated at specified values. The most honest presentation for a non-technical audience, because it makes the nonlinearity visible instead of hiding it in a ratio.

```
b_fico = logit.params["fico"]
print(f"coefficient on fico          {b_fico:+.6f}  (log-odds per point)")
print(f"odds ratio, per point        {np.exp(b_fico):.5f}")
print(f"odds ratio, per 10 points    {np.exp(10 * b_fico):.4f}")

ame = (b_fico * p * (1 - p)).mean()
print(f"\naverage marginal effect     {ame:+.8f} per point")
print(f"  = {ame * 10 * 100:+.3f} percentage points per 10 FICO points")
print(f"  statsmodels margeff:       {logit.get_margeff().margeff[0]:+.8f}")

# The same coefficient, at three different places on the curve.
print("\nthe marginal effect is not one number:")
for label, pv in (("a safe borrower  (p=0.05)", 0.05),
                  ("a typical one    (p=0.16)", y.mean()),
                  ("a risky one      (p=0.50)", 0.50)):
    print(f"  {label}: {b_fico * pv * (1-pv) * 10 * 100:+.3f} pp per 10 poin
```

```
coefficient on fico          -0.005655  (log-odds per point)
odds ratio, per point        0.99436
odds ratio, per 10 points    0.9450

average marginal effect      -0.00073738 per point
  = -0.737 percentage points per 10 FICO points
  statsmodels margeff:       -0.00073738

the marginal effect is not one number:
  a safe borrower  (p=0.05): -0.269 pp per 10 points
  a typical one    (p=0.16): -0.760 pp per 10 points
  a risky one      (p=0.50): -1.414 pp per 10 points
```

The last block is the argument against reporting odds ratios alone. One coefficient, one odds ratio — and a change of 10 FICO points moves the probability three times as much for a borderline borrower as for a safe one. An odds ratio conceals that; a marginal effect or a plot of predicted probabilities shows it.

There is a subtler problem with odds ratios that is widely committed. Logistic coefficients are identified only up to the scale of the unobserved error, and that scale changes when predictors are added — so coefficients from nested logistic models are **not comparable** the way linear coefficients are, and an apparent change on adding a control may be pure rescaling (Mood 2009). Compare average marginal

effects or predicted probabilities across specifications, not raw coefficients or odds ratios.

4.5 Comparing models: the likelihood-ratio test

With no sums of squares there is no F test, but the likelihood supplies the analogue. For nested models, twice the difference in log-likelihood is asymptotically χ^2 with degrees of freedom equal to the number of restrictions:

$$\text{LR} \;=\; 2(\ell_{\text{full}} - \ell_{\text{restricted}}) \;\xrightarrow{d}\; \chi^2_q.$$

```
from scipy import stats as st

full = logit
small = sm.Logit(y, sm.add_constant(loans[["fico", "int_rate", "log_annual_inc"

lr = 2 * (full.llf - small.llf)
pval = st.chi2.sf(lr, df=1)
print(f"full model      log-likelihood {full.llf:12.4f}")
print(f"without dti     log-likelihood {small.llf:12.4f}")
print(f"\nLR statistic {lr:.4f} on 1 df,  p = {pval:.4f}")
print("dti adds nothing detectable here." if pval > 0.05 else "dti matters.")
```

```
full model      log-likelihood   -4068.0558
without dti     log-likelihood   -4068.0850

LR statistic 0.0584 on 1 df,  p = 0.8090
dti adds nothing detectable here.
```

Note the discipline this enforces. Both models must be fitted to *the same rows* — a predictor with extra missing values silently changes the sample and the comparison becomes meaningless — and the test only applies to nested models. For non-nested candidates, AIC from chapter 3 is the tool.

4.6 Does the model discriminate, and is it calibrated?

Two different questions, routinely conflated.

> **Discrimination** — can the model rank? The area under the ROC curve is exactly the probability that a randomly chosen positive case receives a higher predicted probability than a randomly chosen negative one (Hanley and McNeil 1982). 0.5 is coin-flipping; 1.0 is perfect separation.

> **Calibration** — are the probabilities *true*? Among cases assigned 0.30, do about 30% turn out positive? The Brier score, the mean squared difference between predicted probability and outcome, penalises both errors at once (Brier 1950).

A model can discriminate well and be badly calibrated, or the reverse. Which matters depends entirely on the use: a ranking for a limited budget of inspections needs discrimination; a price that must reflect risk needs calibration.

```
from scipy.stats import rankdata
from sklearn.metrics import roc_auc_score, brier_score_loss

# AUC = P(score of a random positive > score of a random negative). That is
# Mann–Whitney U statistic, so it can be computed from ranks alone.
r = rankdata(p)
n1, n0 = int(y.sum()), int(len(y) - y.sum())
auc_hand = (r[y == 1].sum() - n1 * (n1 + 1) / 2) / (n1 * n0)

print(f"AUC by hand (Mann–Whitney)  {auc_hand:.6f}")
print(f"sklearn roc_auc_score       {roc_auc_score(y, p):.6f}")

brier = brier_score_loss(y, p)
base = np.mean((y - y.mean()) ** 2)
print(f"\nBrier score            {brier:.6f}")
print(f"predicting the base rate {base:.6f}")
print(f"skill over base rate:  {1 - brier/base:.2%}")
```

```
AUC by hand (Mann–Whitney)  0.629502
sklearn roc_auc_score       0.629502

Brier score            0.130664
predicting the base rate 0.134437
skill over base rate:  2.81%
```

Both numbers are sobering and should be reported that way. An AUC of 0.63 is weak discrimination — better than chance, far from useful for an individual decision. The Brier score improves on simply predicting the base rate for everyone by under 3%. The model is real, in the sense that its coefficients are estimated precisely and the likelihood-ratio tests are significant, and it is close to useless as a device for deciding about a particular borrower. Those two facts are compatible, and a large n is what makes them compatible: with 9,578 observations, a very weak signal is statistically unmistakable.

4.7 When the outcome is ordered

Some categorical outcomes have an order — unlikely, somewhat likely, very likely. Multinomial logit would ignore the ordering; separate binary models would waste it. The ordinal model uses it, by fitting one slope and several thresholds:

$$\Pr(y_i \le j) \;=\; \sigma(\tau_j - x_i^\top \beta), \qquad \tau_1 < \tau_2 < \cdots < \tau_{J-1}.$$

Each threshold τ_j marks a cut on the latent scale; the *same* β shifts every cut together. That is the proportional-odds assumption, and it is a real restriction: the effect of a predictor on moving from "unlikely" to "somewhat likely" is assumed identical to its effect on moving from "somewhat likely" to "very likely".

```
from statsmodels.miscmodels.ordinal_model import OrderedModel

grad = qmib.load("ologit")
grad["apply_c"] = pd.Categorical(
    grad["apply"], categories=["unlikely", "somewhat likely", "very likely"], c

om = OrderedModel(grad["apply_c"], grad[["pared", "public", "gpa"]],
                  distr="logit").fit(method="bfgs", disp=0)

print(f"n = {len(grad)}   outcome: {list(grad['apply_c'].cat.categories)}")
print(f"\n{'predictor':12}{'coef':>10}{'odds ratio':>13}")
for name in ["pared", "public", "gpa"]:
    b = om.params[name]
    print(f"{name:12}{b:10.4f}{np.exp(b):13.4f}")
print(f"\nthresholds (latent cutpoints): "
      f"{', '.join(f'{v:.3f}' for v in om.params[-2:].values)}")
print("\nOne slope per predictor, two cutpoints — that is the proportional-odds
print("restriction. Test it before relying on it (Brant 1990); if it fails, a"
print("multinomial model that ignores the ordering is the honest fallback.")
```

```
n = 400   outcome: ['unlikely', 'somewhat likely', 'very likely']

predictor         coef   odds ratio
pared           1.0476       2.8509
public         -0.0586       0.9431
gpa             0.6158       1.8512

thresholds (latent cutpoints): 2.204, 0.740

One slope per predictor, two cutpoints — that is the proportional-odds
restriction. Test it before relying on it (Brant 1990); if it fails, a
multinomial model that ignores the ordering is the honest fallback.
```

Having a parent with a graduate degree multiplies the odds of being in a higher application category by about 2.85. The assumption that this multiplier is the same at both cutpoints should be tested rather than assumed (Brant 1990); when it fails, either a partial-proportional-odds model or the unordered multinomial below is preferable to a restriction the data reject.

4.8 When the categories have no order

With J unordered categories, one is chosen as the baseline and $J-1$ binary comparisons are estimated jointly:

$$\log \frac{\Pr(y_i = j)}{\Pr(y_i = \text{baseline})} = x_i^\top \beta_j, \qquad j \neq \text{baseline}.$$

Every coefficient is therefore relative to the baseline, and changing the baseline changes all of them — without changing a single fitted probability. A multinomial result reported without naming the baseline cannot be read.

```
hs = qmib.load("hsbdemo")
hs["prog_c"] = pd.Categorical(hs["prog"], categories=["academic", "general",
ses_code = pd.Categorical(hs["ses"], categories=["low", "middle", "high"]).c

mn = sm.MNLogit(hs["prog_c"].cat.codes,
                sm.add_constant(hs[["write"]].assign(ses=ses_code))).fit(dis

print(f"n = {len(hs)}   baseline = academic")
print(f"\n{'':10}{'general':>12}{'vocation':>12}")
for i, name in enumerate(["const", "write", "ses"]):
    print(f"{name:10}{mn.params.iloc[i,0]:12.4f}{mn.params.iloc[i,1]:12.4f}"
print(f"\nodds ratio for a 1-point writing score:")
print(f"  general  vs academic  {np.exp(mn.params.iloc[1,0]):.4f}")
print(f"  vocation vs academic  {np.exp(mn.params.iloc[1,1]):.4f}")
print(f"\nMcFadden pseudo-R² {1 - mn.llf/mn.llnull:.4f}")
```

```
n = 200   baseline = academic

               general    vocation
const           2.9426      5.5388
write          -0.0582     -0.1122
ses            -0.6367     -0.4512

odds ratio for a 1-point writing score:
  general  vs academic  0.9435
  vocation vs academic  0.8939

McFadden pseudo-R² 0.1072
```

The multinomial model carries an assumption that the ordinal one does not: *independence of irrelevant alternatives*. The ratio of probabilities for any two categories must not depend on what other categories exist. When alternatives are close substitutes the assumption fails, and the classic illustration is a transport choice between car, red bus and blue bus — introducing a bus of a different colour should not change the car's share, and in a multinomial logit it does.

4.9 The failure mode worth knowing

Logistic regression has one dramatic failure, and it looks like success.

> **Separation** — a predictor, or a combination of them, perfectly predicts the outcome in the sample. The likelihood then has no maximum: pushing the coefficient toward infinity keeps improving the fit, so the optimiser diverges. Estimates come back enormous with enormous standard errors, or the software reports non-convergence (Albert and Anderson 1984).

```
rng = np.random.default_rng(60033)
n = 60
x = rng.normal(size=n)
yy = (x > 0).astype(int)          # perfectly separated by construction

print(f"n = {n};  the outcome is exactly 1 when x > 0, with no exceptions.")

try:
    sep = sm.Logit(yy, sm.add_constant(x)).fit(disp=0, maxiter=200)
    print(f"\ncoefficient   {np.asarray(sep.params)[1]:>14,.2f}")
    print(f"std. error    {np.asarray(sep.bse)[1]:>14,.2f}")
    print(f"converged     {sep.mle_retvals['converged']}")
except Exception as exc:
    print(f"\nestimation failed: {type(exc).__name__}: {exc}")

print("\nThe fit does not merely become imprecise — it ceases to exist. Pushing
print("the coefficient toward infinity keeps improving the likelihood, so there
print("is no maximum to find; the curvature the optimiser needs goes to zero an
print("the matrix it must invert becomes singular. Whether your software return
print("an enormous coefficient, a warning, or an exception is an accident of it
print("implementation, not a difference in the underlying problem.")

# The penalised estimator has a finite maximum even under separation.
from statsmodels.discrete.discrete_model import Logit
pen = Logit(yy, sm.add_constant(x)).fit_regularized(alpha=1.0, disp=0)
print(f"\nwith an L2 penalty, the coefficient is finite: "
```

```
      f"{np.asarray(pen.params)[1]:.4f}")

n = 60;  the outcome is exactly 1 when x > 0, with no exceptions.

estimation failed: LinAlgError: Singular matrix

The fit does not merely become imprecise — it ceases to exist. Pushing
the coefficient toward infinity keeps improving the likelihood, so there
is no maximum to find; the curvature the optimiser needs goes to zero and
the matrix it must invert becomes singular. Whether your software returns
an enormous coefficient, a warning, or an exception is an accident of its
implementation, not a difference in the underlying problem.

with an L2 penalty, the coefficient is finite: 5.7571
```

The wrong response is to report the number. The right one is to recognise the diagnosis — check for a predictor that separates the outcome, especially a rare category or a small subgroup — and then either drop the offending variable, merge categories, or use a penalised estimator. Firth's method adds a penalty that guarantees finite estimates and reduces small-sample bias at the same time (Firth 1993).

4.10 A short review of the literature

The curve predates the statistics by a century: Berkson (1944) proposed the logit for bio-assay against the entrenched probit, and Cox (1958) gave binary regression its modern treatment. The two transformations remain nearly indistinguishable in fit and differ mainly in the interpretability of their coefficients.

Assessment splits along the discrimination–calibration line. Hanley and McNeil (1982) gave the AUC its standard exposition and its rank interpretation — the property that lets it be computed from a Mann-Whitney statistic, as above. Brier (1950), published in meteorology and largely ignored by econometrics for decades, is the canonical proper scoring rule for probability forecasts, and remains the right answer when the probabilities themselves matter rather than their ordering.

The comparability problem is the most under-appreciated result in applied use. Mood (2009) showed that logistic coefficients are confounded with the scale of the unobserved error, so adding controls rescales them and cross-model comparison of coefficients or odds ratios is not valid — a practice that remains routine. Average marginal effects are the standard remedy.

For extensions, Brant (1990) supplies the test of the proportional-odds restriction that ordinal models assume and rarely check. On the failure of estimation, Albert and Anderson (1984) characterised exactly when maximum likelihood estimates fail to exist under separation, and Firth (1993) supplied the penalty that is now the standard fix.

4.11 What to report

1. **The base rate.** Every performance number is meaningless without it; 95% accuracy on a 5% outcome is achieved by predicting "no" for everyone.
2. **Coefficients as odds ratios, with marginal effects beside them.** The odds ratio is constant and the probability change is not, and the reader needs both to know what a coefficient means where they intend to use it.
3. **A nested comparison, as a likelihood-ratio test**, fitted to identical rows, with the restriction stated.
4. **Discrimination and calibration, separately.** AUC and Brier, each against its baseline, with a sentence on which one the intended decision needs.
5. **The assumption the extension rests on.** Proportional odds for ordinal; independence of irrelevant alternatives for multinomial. Say that you tested it, or say that you did not.
6. **What the model does not license.** These are associations in a sample of borrowers who were granted loans — a population already selected by the lender's own approval rule, which is not a random sample of applicants and certainly not of people. Chapter 10 gives that problem its name.

Chapter 5

Regularisation: Ridge, Lasso, and the Elastic Net

5.1 The mechanical first attempt

Suppose you have more candidate predictors than you can defend and fewer observations than you would like. The obvious procedure is to let the data choose: start from nothing, add whichever variable improves the fit most, stop when nothing improves it further. That is *forward stepwise selection*, and variants of it — backward elimination, bidirectional stepwise — were the standard answer for decades and remain the default in a great deal of applied software.

It is worth watching it fail, carefully, because everything in this chapter is a response to the way it fails.

The setting is the wide slice of the course fixtures: thirty-seven complete country-years and twenty-nine candidate predictors. The fixture was built deliberately so that selection has something to get wrong. Fifteen of the columns are named `aux_00` to `aux_14`; ten of them are pure noise, drawn from a standard normal and unrelated to anything, while every third one carries a weak signal — a quarter of a standard deviation of a real latent factor. We know which is which, because the generator wrote them. That is a luxury no real dataset provides and the whole reason for working with fixtures here.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, statsmodels.api as sm, qmib
from collections import Counter

wide = qmib.load("angle_c_country").dropna()
num = wide.select_dtypes("number").drop(columns=["time"], errors="ignore")
y = num["ai_use_any"]
X = num.drop(columns=["ai_use_any"])

# Ground truth, from scripts/build_spine/make_fixtures.py.
signal_aux = {f"aux_{k:02d}" for k in range(15) if k % 3 == 0}
noise_aux = {f"aux_{k:02d}" for k in range(15)} - signal_aux

print(f"n = {len(num)}   candidate predictors p = {X.shape[1]}")
print(f"   of which pure noise: {len(noise_aux)}   weak signal: {len(signal_
```

```
def forward_aic(X, y):
    """Add the variable that lowers AIC most, until none does."""
    chosen, remaining = [], list(X.columns)
    best = sm.OLS(y, np.ones((len(y), 1))).fit().aic
    while remaining:
        scored = sorted((sm.OLS(y, sm.add_constant(X[chosen + [c]])).fit().aic,
                         for c in remaining)
        if not scored or scored[0][0] >= best - 1e-9:
            break
        best, pick = scored[0]
        chosen.append(pick)
        remaining.remove(pick)
    return chosen


selected = forward_aic(X, y)
false_positives = [c for c in selected if c in noise_aux]
print(f"\nstepwise selected {len(selected)} variables")
print(f"   pure noise among them: {len(false_positives)} — {false_positives}")
```

```
n = 37   candidate predictors p = 29
   of which pure noise: 10   weak signal: 5

stepwise selected 14 variables
  pure noise among them: 4 — ['aux_01', 'aux_04', 'aux_10', 'aux_14']
```

Fourteen variables survive, and four of them are columns of pure noise. That alone might be dismissed as bad luck. The damning result is what happens when the same procedure is run on resamples of the same data.

```
rng = np.random.default_rng(60033)
B, counts = 200, Counter()
for _ in range(B):
    idx = rng.integers(0, len(num), len(num))
    counts.update(forward_aic(X.iloc[idx], y.iloc[idx]))

print(f"forward stepwise repeated on {B} bootstrap resamples\n")
print(f"{'variable':22}{'selected':>10}   kind")
for name, k in counts.most_common(8):
    kind = "PURE NOISE" if name in noise_aux else (
        "weak signal" if name in signal_aux else "real predictor")
    print(f"{name:22}{k/B:>9.0%}   {kind}")
print(f"\nvariables selected at least once: {len(counts)} of {X.shape[1]}")
```

```
forward stepwise repeated on 200 bootstrap resamples

variable                 selected   kind
rd_pct_gdp                   98%   real predictor
ai_use_ge3                   94%   real predictor
has_ai_strategy              90%   real predictor
aux_04                       88%   PURE NOISE
aux_12                       86%   weak signal
barrier_legal                86%   real predictor
aux_10                       86%   PURE NOISE
aux_06                       84%   weak signal

variables selected at least once: 29 of 29
```

Every one of the twenty-nine candidates is selected in some resample. A column of pure noise is chosen in the high eighties per cent of resamples — as reliably as the genuine predictors. "The data selected these variables" is therefore not a finding. Perturb the data slightly and the data select different ones.

The diagnosis is that stepwise makes a sequence of *discrete* decisions, each conditional on the last. A variable is either in or out; a marginal difference in fit at step three changes the entire subsequent path. That discreteness is what makes the procedure unstable, and it points at the fix: stop making in-or-out decisions and shrink coefficients continuously instead.

5.2 Why a biased estimator can win

Gauss–Markov said least squares is best among *unbiased* linear estimators. Chapter 3 treated that as reassurance. Read it again as a restriction: the theorem says nothing about estimators that accept bias, and it leaves open whether one of those might have smaller error overall.

Decompose the expected squared error of an estimate $\hat{\theta}$ of a quantity θ:

$$\mathrm{E}\big[(\hat{\theta}-\theta)^2\big] \;=\; \underbrace{\big(\mathrm{E}[\hat{\theta}]-\theta\big)^2}_{\text{bias}^2} \;+\; \underbrace{\mathrm{Var}(\hat{\theta})}_{\text{variance}}\,.$$

Least squares sets the first term to zero exactly. If the second term is large — and with twenty-nine correlated predictors and thirty-seven observations it is enormous — then accepting a little bias to remove a lot of variance is a straightforwardly better trade. That is the entire argument for this chapter, and it is a decision-theoretic argument rather than a statistical convention.

```
full = sm.OLS(y, sm.add_constant(X)).fit()
print(f"OLS on all {X.shape[1]} predictors, n = {len(num)}")
print(f"  R²                 {full.rsquared:.4f}")
print(f"  adjusted R²        {full.rsquared_adj:.4f}")
```

```
print(f"  residual d.f.      {int(full.df_resid)}")
print(f"  largest |coef|     {np.abs(full.params[1:]).max():,.2f}")
print(f"  mean std. error    {full.bse[1:].mean():,.2f}")
print("\nAn R² of 0.99 with seven residual degrees of freedom is not a good fi
print("It is a model with almost as many parameters as observations.")
```

```
OLS on all 29 predictors, n = 37
  R²                 0.9935
  adjusted R²        0.9667
  residual d.f.      7
  largest |coef|     5.78
  mean std. error    0.42

An R² of 0.99 with seven residual degrees of freedom is not a good fit.
It is a model with almost as many parameters as observations.
```

5.3 Ridge: shrink everything, targeted

Hoerl and Kennard proposed adding a penalty on the squared size of the coefficients (Hoerl and Kennard 1970):

$$\hat{\beta}^{\text{ridge}} \;=\; \arg\min_{\beta} \; \|y - X\beta\|^2 + \lambda \|\beta\|_2^2, \qquad \lambda \geq 0.$$

Unlike the lasso below, this has a closed form, and the form is the argument:

$$\hat{\beta}^{\text{ridge}} \;=\; (X^\top X + \lambda I)^{-1} X^\top y.$$

Compare it to $(X^\top X)^{-1} X^\top y$. The only change is λI added to the diagonal — which is why the method is called *ridge*, and why it works when least squares cannot: $X^\top X$ may be singular or nearly so, but $X^\top X + \lambda I$ is invertible for any $\lambda > 0$. Ridge is defined even when $p > n$, where least squares has no unique solution at all.

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge

# Standardisation is not optional — see below.
Z = StandardScaler().fit_transform(X)
yc = (y - y.mean()).to_numpy()

lam = 18.67
beta_hand = np.linalg.solve(Z.T @ Z + lam * np.eye(Z.shape[1]), Z.T @ yc)
beta_sk = Ridge(alpha=lam, fit_intercept=False).fit(Z, yc).coef_

print(f"largest |difference| between the formula and sklearn: "
```

```
      f"{np.abs(beta_hand - beta_sk).max():.2e}")

# Why the shrinkage is targeted: it acts through the singular values of X.
sv = np.linalg.svd(Z, compute_uv=False)
print(f"\n{'singular value':>16}{'shrinkage factor':>20}")
for s in [sv[0], sv[len(sv)//2], sv[-1]]:
    print(f"{s:16.3f}{s**2/(s**2 + lam):20.4f}")
print("\nDirections in which the data vary a lot are barely touched;")
print("directions in which they barely vary are shrunk almost to nothing.")
```

```
largest |difference| between the formula and sklearn: 1.44e-15

  singular value    shrinkage factor
          20.890              0.9590
           3.456              0.3901
           0.311              0.0052

Directions in which the data vary a lot are barely touched;
directions in which they barely vary are shrunk almost to nothing.
```

That last block is the point of ridge and the answer to "why not just multiply every coefficient by 0.9". In the singular value decomposition of X, the ridge estimate shrinks the component along the j-th direction by the factor $d_j^2/(d_j^2+\lambda)$. Where the data are informative — large d_j — the factor is near 1 and almost nothing happens. Where the data barely vary, and the least-squares coefficient is therefore an unstable ratio of noise to a very small number, the factor is near 0 and the coefficient is crushed. The shrinkage goes exactly where the variance problem is.
What ridge does *not* do is select. Every coefficient becomes smaller; none becomes zero. With twenty-nine candidates you still have twenty-nine.

5.4 Lasso: the corner that produces a zero

Tibshirani's change is a single character: penalise the sum of absolute values rather than of squares (Tibshirani 1996).

$$\hat{\beta}^{\text{lasso}} \;=\; \arg\min_{\beta} \, \|y - X\beta\|^2 + \lambda \|\beta\|_1 .$$

There is no closed form in general, but there is in the case that explains everything. Suppose the predictors are orthonormal, so $X^\top X = I$ and the least-squares solution is $\hat{\beta}_j^{\text{OLS}} = z_j$. The objective then separates into one problem per coefficient, and each has the solution

$$\hat{\beta}_j^{\text{lasso}} \;=\; \operatorname{sign}(z_j)\big(|z_j| - \lambda/2\big)_+$$

— the *soft-thresholding* operator. Read what it does. Every coefficient is pulled toward zero by the same fixed amount $\lambda/2$, and any coefficient whose least-squares

value was smaller than that amount is pulled *past* zero and held there. The zeros are not a numerical accident. They are what happens when a constant subtraction meets a small number.
Ridge, in the same orthonormal case, gives $z_j/(1+\lambda)$ — a *proportional* shrinkage, which multiplies small coefficients by a factor and therefore never reaches zero.

```
from sklearn.linear_model import Lasso

def soft_threshold(z, t):
    return np.sign(z) * np.maximum(np.abs(z) - t, 0.0)

# Build a genuinely orthonormal design so the closed form applies.
rng2 = np.random.default_rng(7)
n_o, p_o = 60, 8
Q, _ = np.linalg.qr(rng2.normal(size=(n_o, p_o)))
beta_true = np.array([3.0, -2.0, 1.2, 0.4, 0.1, 0.0, 0.0, 0.0])
y_o = Q @ beta_true + rng2.normal(0, 0.05, n_o)

z = Q.T @ y_o                       # the OLS solution, since Q'Q = I
lam_o = 0.5
hand = soft_threshold(z, lam_o / 2)
sk = Lasso(alpha=lam_o / (2 * n_o), fit_intercept=False, max_iter=100000).fit(

print(f"{'OLS':>10}{'soft-threshold':>17}{'sklearn lasso':>16}")
for a, b_, c_ in zip(z, hand, sk):
    print(f"{a:10.4f}{b_:17.4f}{c_:16.4f}")
print(f"\nlargest |difference|: {np.abs(hand - sk).max():.2e}")
print(f"coefficients set exactly to zero: {(hand == 0).sum()} of {p_o}")
```

```
       OLS   soft-threshold   sklearn lasso
    3.0202           2.7702          2.7702
   -2.0371          -1.7871         -1.7871
    1.2302           0.9802          0.9802
    0.5094           0.2594          0.2594
    0.1769           0.0000          0.0000
    0.0586           0.0000          0.0000
   -0.0507          -0.0000         -0.0000
    0.1002           0.0000          0.0000

largest |difference|: 1.11e-15
coefficients set exactly to zero: 4 of 8
```

The geometry says the same thing in a picture. Both estimators can be written as minimising the residual sum of squares subject to a *budget* on the coefficients: $\|\beta\|_2^2 \le t$ for ridge, $\|\beta\|_1 \le t$ for the lasso. The first constraint region is a disc; the

second is a diamond with its corners on the axes. The solution is where the elliptical contours of the residual sum of squares first touch the region — and a diamond is touched at a corner far more often than a disc is touched at a pole, because a corner is where the boundary is not smooth. A corner on the axis means a coefficient of exactly zero.

```
from matplotlib.patches import Circle, Polygon, Ellipse

b_ols = np.array([1.15, 0.85])
fig, axes = plt.subplots(1, 2, figsize=(9.0, 4.4))

for ax, kind in zip(axes, ("lasso", "ridge")):
    if kind == "lasso":
        t = 0.75
        region = Polygon([[t, 0], [0, t], [-t, 0], [0, -t]], closed=True,
                         fc=CL.accent, alpha=0.16, ec=CL.accent, lw=1.6)
        touch = np.array([t, 0.0])
    else:
        t = 0.80
        region = Circle((0, 0), t, fc=CL.accent, alpha=0.16, ec=CL.accent, l
        d = b_ols / np.linalg.norm(b_ols)
        touch = d * t
    ax.add_patch(region)

    for k in (0.35, 0.75, 1.2):
        r = np.linalg.norm(b_ols - touch)
        ax.add_patch(Ellipse(b_ols, 2*(r + k*0.55), 2*(r + k*0.3), angle=25,
                             fill=False, ec=CL.warn, lw=1.0, alpha=0.65))
    ax.add_patch(Ellipse(b_ols, 2*np.linalg.norm(b_ols - touch),
                         2*np.linalg.norm(b_ols - touch)*0.62, angle=25,
                         fill=False, ec=CL.warn, lw=1.8))

    ax.plot(*b_ols, marker="o", ms=6, color=CL.ink)
    ax.annotate(r"$\hat\beta^{OLS}$", b_ols, textcoords="offset points",
                xytext=(8, 4), fontsize=9)
    ax.plot(*touch, marker="o", ms=8, mfc=CL.warn, mec="white", mew=1.2, zor
    ax.axhline(0, color=CL.line, lw=1); ax.axvline(0, color=CL.line, lw=1)
    ax.set_xlim(-1.3, 1.9); ax.set_ylim(-1.3, 1.6); ax.set_aspect("equal")
    ax.set_xlabel(r"$\beta_1$"); ax.set_ylabel(r"$\beta_2$")
    ax.set_title(f"{kind}: "
                 + (r"$\|\beta\|_1 \leq t$" if kind == "lasso" else r"$\|\be
                 fontsize=10, loc="left")

fig.tight_layout()
```

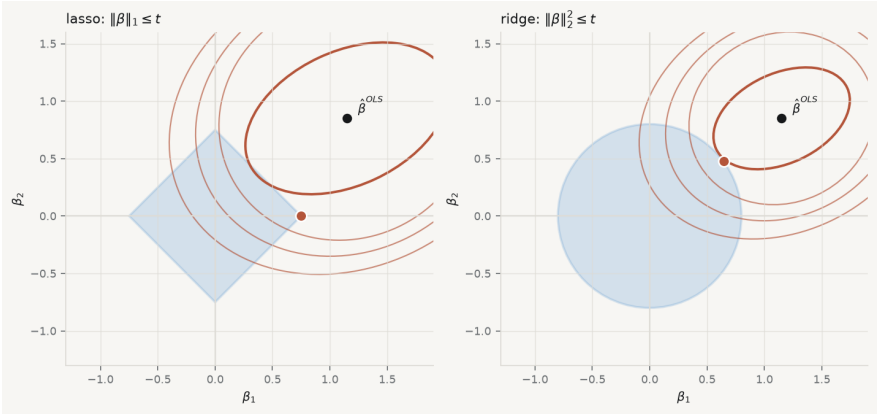


Figure 5.1. Why the lasso produces zeros and ridge does not. Contours of the residual sum of squares, centred on the least-squares solution, meeting the constraint region. The diamond is touched at its corner, where $\beta_1 = 0$; the disc is touched at a point with both coordinates nonzero.

5.5 Standardise, always

The penalties are on the *size* of the coefficients, and a coefficient's size depends on the units of its predictor. Measure a variable in euros rather than thousands of euros and its coefficient shrinks by a factor of a thousand, so the penalty barely notices it. Nothing about the data changed; the answer did.

Standardising every predictor to mean zero and unit variance before fitting is therefore not hygiene, it is part of the estimator's definition. The intercept is left unpenalised — there is no reason to shrink the overall level toward zero — which is why it is conventionally handled by centring y rather than by including a column of ones in the penalty.

5.6 The elastic net, and correlated predictors

The lasso has a specific weakness. Among a group of strongly correlated predictors it tends to pick one, essentially arbitrarily, and zero the rest. If the group is a set of imperfect measurements of the same underlying thing, the choice is close to random, and it will change on a resample — the stepwise instability returning in a new form.

Zou and Hastie's elastic net penalises both norms at once (Zou and Hastie 2005):

$$\hat{\beta}^{\mathrm{EN}} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\Big(\alpha\|\beta\|_1 + \tfrac{1-\alpha}{2}\|\beta\|_2^2\Big).$$

The ℓ_1 part still produces zeros; the ℓ_2 part produces the *grouping effect*, under which strongly correlated predictors receive similar coefficients and tend to enter or leave together. $\alpha = 1$ is the lasso, $\alpha = 0$ is ridge, and the mixing weight is

chosen the same way λ is.

5.7 Choosing λ, and what the choice costs

The penalty is not estimated from the fit — a larger λ always fits the training data worse, so anything that rewards fit alone would choose zero. It is chosen by out-of-sample performance, which is what cross-validation estimates (M. Stone 1974): split the sample into K folds, fit on $K-1$, measure error on the held-out fold, rotate, average.

```
from sklearn.linear_model import LassoCV, lasso_path

Zs = StandardScaler().fit_transform(X)
ys = (y - y.mean()).to_numpy() / y.std(ddof=0)

alphas, coefs, _ = lasso_path(Zs, ys, n_alphas=120)
cv = LassoCV(cv=5, random_state=0, max_iter=100000).fit(Zs, ys)

fig, (axp, axc) = plt.subplots(1, 2, figsize=(9.6, 4.2))

for j in range(coefs.shape[0]):
    is_noise = X.columns[j] in noise_aux
    axp.plot(np.log10(alphas), coefs[j],
             lw=1.6 if not is_noise else 0.9,
             color=CL.muted if is_noise else CL.accent,
             alpha=0.55 if is_noise else 0.9, zorder=1 if is_noise else 2)
axp.axvline(np.log10(cv.alpha_), color=CL.warn, lw=1.6, ls="--")
axp.set_xlabel(r"$\log_{10}\lambda$"); axp.set_ylabel("coefficient")
axp.set_title("the lasso path (grey = pure noise)", fontsize=10, loc="left")

mse = cv.mse_path_.mean(axis=1)
se = cv.mse_path_.std(axis=1) / np.sqrt(cv.mse_path_.shape[1])
axc.plot(np.log10(cv.alphas_), mse, lw=1.8, color=CL.accent)
axc.fill_between(np.log10(cv.alphas_), mse - se, mse + se, color=CL.accent,
axc.axvline(np.log10(cv.alpha_), color=CL.warn, lw=1.6, ls="--", label="mini
best = mse.argmin()
thresh = mse[best] + se[best]
one_se = cv.alphas_[np.where(mse <= thresh)[0]].max()
axc.axvline(np.log10(one_se), color=CL.ink, lw=1.4, ls=":", label="one stand
axc.set_xlabel(r"$\log_{10}\lambda$"); axc.set_ylabel("cross-validated MSE")
axc.set_title("choosing the penalty", fontsize=10, loc="left")
axc.legend(frameon=False, fontsize=8.5)

fig.tight_layout()
```

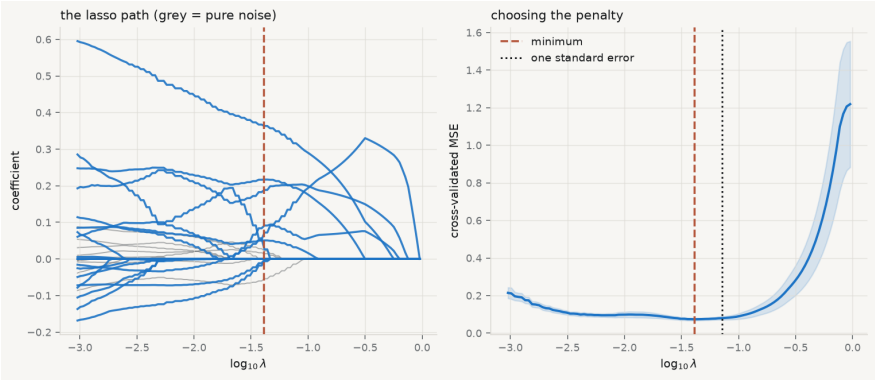


Figure 5.2. Left: the lasso path — every coefficient against the penalty, with the cross-validated choice marked. Coefficients arrive at zero and stay there. Right: the cross-validation curve, with the minimum and the one-standard-error choice.

The dotted line is the *one-standard-error rule*: rather than the λ with the lowest estimated error, take the largest λ whose error is within one standard error of the minimum. The cross-validation curve is itself an estimate with noise in it, and its minimum is therefore optimistically located. The rule buys a simpler model for an error difference the data cannot resolve.

5.8 Scoring the methods against a known truth

Because the fixture's generator is known, this is one of the rare occasions when selection methods can be marked rather than admired.

```
from sklearn.linear_model import RidgeCV, ElasticNetCV

fits = {
    "ridge":       RidgeCV(alphas=np.logspace(-3, 3, 60)),
    "lasso":       LassoCV(cv=5, random_state=0, max_iter=100000),
    "elastic net": ElasticNetCV(l1_ratio=[.1, .5, .7, .9, .95, .99],
                                cv=5, random_state=0, max_iter=100000),
}
print(f"{'method':14}{'kept':>6}{'noise kept':>12}   penalty")
for name, mdl in fits.items():
    mdl.fit(Zs, ys)
    kept = pd.Series(mdl.coef_, index=X.columns)
    kept = kept[kept.abs() > 1e-8]
    junk = [c for c in kept.index if c in noise_aux]
    extra = f"λ={mdl.alpha_:.4g}"
    if hasattr(mdl, "l1_ratio_"):
```

```
        extra += f", α={mdl.l1_ratio_}"
    print(f"{name:14}{len(kept):>6}{len(junk):>12}   {extra}")

print(f"\nfor reference, stepwise kept {len(selected)} with "
      f"{len(false_positives)} pure-noise variables")
print(f"and there are {len(noise_aux)} pure-noise columns in total")
```

```
method          kept  noise kept   penalty
ridge             29          10   λ=18.67
lasso             12           3   λ=0.04122
elastic net       12           3   λ=0.04164, α=0.99

for reference, stepwise kept 14 with 4 pure-noise variables
and there are 10 pure-noise columns in total
```

Three things in that table are worth stating plainly, including the one that is not flattering.

Ridge keeps everything, as promised — all ten noise columns included, merely with small coefficients. If the deliverable is a shortlist, ridge does not produce one.

The lasso cuts the model from twenty-nine variables to twelve and reduces the pure-noise variables from ten to three. That is a large improvement and it is not a solution. Three noise columns survive cross-validated selection, and a reader told "the lasso selected these twelve" would have no way to know.

The elastic net chose a mixing weight of 0.99 — almost pure lasso — and returned the same model, despite predictor correlations as high as 0.96. That is worth reporting because it contradicts the tidy expectation. Cross-validation optimises predictive error, and the grouping effect buys stability rather than prediction; when the criterion is prediction alone, it will often decline to pay for it.

5.9 What none of this licenses

The chapter's most important sentence is a prohibition.

> **A p-value from a model chosen by the lasso is not a p-value.** Fit the selected variables by ordinary least squares and report the resulting standard errors, and every one of them is wrong — computed as though the specification had been fixed in advance, when in fact it was chosen using the same data.

This is chapter 3's Freedman problem with a modern face, and it has been treated formally. Valid inference after selection requires either conditioning on the selection event or widening the intervals to cover every model that might have been selected (Berk et al. 2013). Tests designed specifically for the lasso path exist (Lockhart et al. 2014) and are not what a naive refit computes.

The practical responses, in order of how often they are available:

Split the sample. Select on one half, estimate and test on the other. The inference is then honest, at the cost of half the data, and it is the only remedy that needs no theory.
Report stability rather than significance. Refit on many resamples and report the proportion of times each variable is selected (Meinshausen and Bühlmann 2010). That is a statement the data support, and — as the bootstrap at the top of this chapter showed — it is often far less impressive than the single-fit result.
Treat selection as exploratory and say so. A shortlist for further work is a legitimate output. A shortlist presented as a tested finding is not.

5.10 A short review of the literature

Ridge regression begins with Hoerl and Kennard (1970), who proposed biased estimation explicitly as a response to instability under near-collinearity — the title says "biased estimation" without apology. Frank and Friedman (1993) set ridge, subset selection and their relatives in a common framework as members of a family indexed by the power of the penalty, which is where the lasso's ℓ_1 sits between ridge's ℓ_2 and subset selection's ℓ_0.
Tibshirani (1996) introduced the lasso and observed the property that made it dominant: continuous shrinkage and automatic selection in the same estimator. Efron et al. (2004) supplied least angle regression, which computes the entire coefficient path at the cost of a single least-squares fit and turned the lasso from attractive into practical. Zou and Hastie (2005) diagnosed the lasso's behaviour among correlated predictors and proposed the elastic net, with the grouping effect as the explicit remedy.
Choosing the penalty rests on M. Stone (1974), whose formulation of cross-validation as a general criterion for model choice long predates its use here.
The inferential warning has the sharpest literature. Berk et al. (2013) established what valid post-selection inference requires and how much wider the intervals must be; Lockhart et al. (2014) developed a test for the lasso path specifically, making clear how different it is from a naive refit. Meinshausen and Bühlmann (2010) offers stability selection as the practical alternative — report how often a variable is chosen across resamples, rather than a significance level that assumes it was never chosen at all.

5.11 What to report

1. **The path, not just the endpoint.** A plot of coefficients against λ shows how contingent the chosen model is. If several variables leave within a narrow band of λ, say so.
2. **How λ was chosen**, with the cross-validation curve, and whether the minimum or the one-standard-error rule was used.
3. **That the predictors were standardised.** The estimator is not defined without it.
4. **What survived, and how reliably.** Selection frequencies across resamples are worth more than the single selected set.

5. **The prohibition, explicitly.** State that the reported coefficients are post-selection and that conventional standard errors and p-values do not apply to them. If you split the sample, say which half did what.
6. **Why this penalty.** Lasso for a shortlist, ridge for prediction under collinearity with no selection wanted, elastic net when correlated groups must move together — and if cross-validation chose an α near 1 despite correlated predictors, report that rather than the story you expected.

Chapter 6

Regression: Advanced Considerations

6.1 One relationship, or thirty?

Chapters 2 and 3 fitted a single line to thirty countries observed over fifteen years, and reported one slope. That slope was an answer to a question nobody had asked precisely: *whose* relationship is it? The relationship between productivity and income across countries at a moment in time, or the relationship within a country as it changes over time, are different questions with different answers, and pooling them produces a number that is neither.

Start by asking whether there is one relationship at all.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, statsmodels.api as sm, qmib

core = (qmib.load("core").dropna(subset=["gdp_pc_eur", "productivity_idx"])
        .reset_index(drop=True))
y, x = "gdp_pc_eur", "productivity_idx"

pooled = sm.OLS(core[y], sm.add_constant(core[[x]])).fit()

fig, ax = plt.subplots(figsize=(7.2, 4.8))
slopes = {}
for geo, g in core.groupby("geo"):
    if len(g) < 4:
        continue
    b = np.polyfit(g[x], g[y], 1)
    slopes[geo] = b[0]
    gx = np.linspace(g[x].min(), g[x].max(), 20)
    ax.plot(gx, np.polyval(b, gx), lw=1.0, color=CL.muted, alpha=0.55, zorde

gx = np.linspace(core[x].min(), core[x].max(), 50)
ax.plot(gx, pooled.params.iloc[0] + pooled.params.iloc[1] * gx,
        lw=2.6, color=CL.warn, zorder=3, label="pooled")
ax.scatter(core[x], core[y], s=9, color=CL.accent, alpha=0.30,
           edgecolor="none", zorder=2)
```

```
ax.set_xlabel("productivity index"); ax.set_ylabel("GDP per capita (EUR)")
ax.legend(frameon=False, fontsize=9)
fig.tight_layout()

s = pd.Series(slopes)
print(f"pooled slope        {pooled.params.iloc[1]:>10,.0f}")
print(f"country slopes: min {s.min():>10,.0f}   median {s.median():>10,.0f}   m
print(f"negative in {(s < 0).sum()} of {len(s)} countries")
```

```
pooled slope             1,111
country slopes: min       -975   median        219   max      3,247
negative in 8 of 30 countries
```

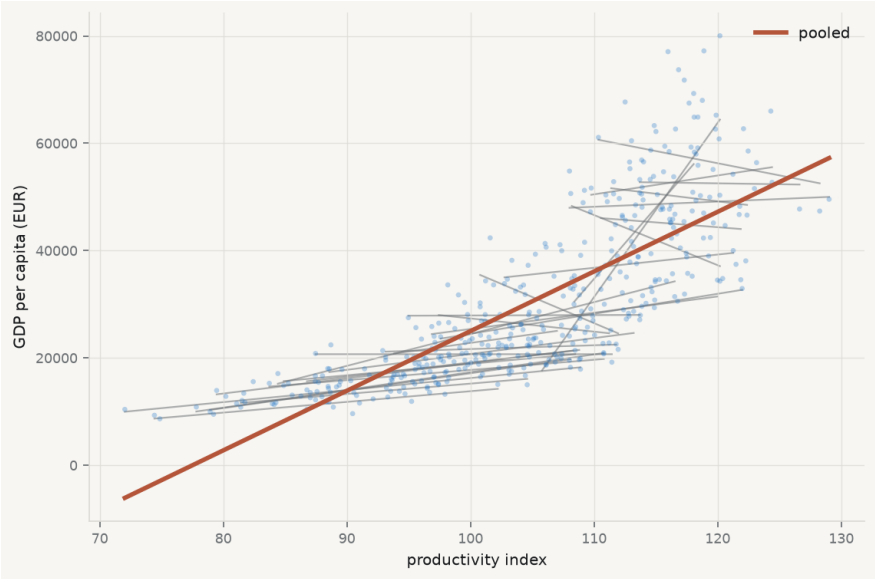


Figure 6.1. One line per country, fitted separately, against the single pooled line in red. The pooled slope is not a summary of the country slopes — it is mostly a comparison between countries.

The picture is the chapter. A single steep line rises through a cloud of much flatter, and in eight cases *downward-sloping*, country lines. The pooled slope is not an average of the country slopes; it is dominated by the fact that countries with high productivity are countries with high income — a comparison *between* countries, not a description of what happens *within* one.

Whether that is the quantity you want depends entirely on the claim you intend to make. If the claim is "countries with higher productivity are richer", the pooled estimate is the right object. If the claim is "raising productivity raises income", it

is close to useless, because the second claim is about movement within a country and the first estimate contains almost none of it.

6.2 Where the variation lives

The reason is arithmetic, and it can be inspected before any model is fitted. Total variation in a panel variable splits into variation *between* units and variation *within* them, and a regression uses whichever is left after the controls have absorbed the rest.

```
core["log_gdp"] = np.log(core[y])

print(f"{'variable':20}{'total var':>12}{'within':>10}{'between':>10}")
for c in [x, "log_gdp"]:
    tot = core[c].var(ddof=1)
    within = (core[c] - core.groupby("geo")[c].transform("mean")).var(ddof=1
    print(f"{c:20}{tot:12.4f}{within/tot:9.1%}{1 - within/tot:10.1%}")

print("\nmean annual drift:")
for c in [x, "log_gdp"]:
    print(f"  {c:20}{np.polyfit(core['time'], core[c], 1)[0]:+10.4f} per yea
```

```
variable              total var    within   between
productivity_idx        119.5167    29.6%     70.4%
log_gdp                   0.2484    13.9%     86.1%

mean annual drift:
  productivity_idx       +0.9671 per year
  log_gdp                -0.0076 per year
```

Two facts from that output govern everything below. The great majority of the variation in both variables is *between* countries — 70% for productivity, 86% for log income — so a pooled regression is overwhelmingly a cross-sectional comparison wearing panel clothing. And productivity drifts upward by about one index point a year while log income does not drift at all, which means the within-country variation in the predictor is substantially a common trend that the outcome does not share.

6.3 Fixed effects: the within transformation

The standard response is to give each country its own intercept:

$$y_{it} \;=\; \alpha_i + \beta x_{it} + \varepsilon_{it}.$$

The α_i absorb everything about a country that does not change over the sample — institutions, geography, legal tradition, whatever is durable — whether or not you can measure it or even name it. That is the appeal, and it is a large one: an

unmeasured confounder that is constant within a country cannot bias $\hat{\beta}$, because it has been differenced away.
There are three equivalent ways to compute it, and their equivalence is Frisch–Waugh–Lovell from chapter 2 in a new costume. Include a dummy for each country. Or subtract each country's mean from each variable and regress the demeaned outcome on the demeaned predictor. Or partial the dummies out of both sides and regress residual on residual. All three are the same estimator.

```
# (a) demeaning — the "within" transformation
dm = core.copy()
for c in [y, x]:
    dm[c + "_w"] = dm[c] - dm.groupby("geo")[c].transform("mean")
within = sm.OLS(dm[y + "_w"], dm[[x + "_w"]]).fit()

# (b) a dummy for every country
geo_d = pd.get_dummies(core["geo"], drop_first=True, dtype=float)
dummies = sm.OLS(core[y], sm.add_constant(pd.concat([core[[x]], geo_d], axis=1

# (c) a purpose-built panel estimator
from linearmodels.panel import PanelOLS
panel = core.set_index(["geo", "time"])
fe = PanelOLS(panel[y], sm.add_constant(panel[[x]]), entity_effects=True).fit(

print(f"pooled                      {pooled.params.iloc[1]:>12,.2f}")
print(f"within, by demeaning        {within.params.iloc[0]:>12,.2f}")
print(f"within, by country dummies  {dummies.params[x]:>12,.2f}")
print(f"within, by PanelOLS         {fe.params[x]:>12,.2f}")
print(f"\nthe pooled slope is {pooled.params.iloc[1]/fe.params[x]:.1f}x the wi
```

```
pooled                          1,111.26
within, by demeaning              274.59
within, by country dummies        274.59
within, by PanelOLS               274.59

the pooled slope is 4.0x the within slope
```

The three agree exactly, and the pooled slope is four times the within slope. That gap is the quantitative version of the picture above: three quarters of the pooled estimate was between-country comparison, and it disappears the moment each country is allowed its own level.

6.4 What fixed effects throws away, and what it cannot fix

Fixed effects are not free, and the costs are systematically undersold.
Anything constant within a unit becomes inestimable. A country's legal origin, its language, its distance from the equator — all absorbed, none estimable. If the

effect you care about is a time-invariant characteristic, fixed effects delete your research question rather than answering it.

The estimator uses only within variation, so precision falls. Here that is 70% of the variation in the predictor discarded. If the within variation is small or badly measured, fixed effects trade bias for a great deal of variance — and measurement error is *amplified* by demeaning, because differencing removes signal while leaving noise.

Time-varying confounders survive untouched. The reassurance that "fixed effects control for unobserved heterogeneity" holds only for heterogeneity that does not move. Anything that changes within a country over the sample period — a reform, a business cycle, a shock — passes straight through.

6.5 Two countries' worth of dummies, and a puzzle

Time shocks are the obvious example of a confounder that fixed effects do not remove. A recession that hits every country at once creates common movement in both variables that has nothing to do with the relationship of interest. The remedy is a second set of dummies, one per year:

$$y_{it} \;=\; \alpha_i + \gamma_t + \beta x_{it} + \varepsilon_{it}.$$

The identifying variation is now what remains after both country means and year means have been removed — the purely idiosyncratic part of each observation.

```
two = PanelOLS(panel[y], sm.add_constant(panel[[x]]),
               entity_effects=True, time_effects=True).fit()

print(f"{'specification':32}{'slope':>12}")
print(f"{'pooled':32}{pooled.params.iloc[1]:>12,.2f}")
print(f"{'country effects':32}{fe.params[x]:>12,.2f}")
print(f"{'country + year effects':32}{two.params[x]:>12,.2f}")
```

```
specification                           slope
pooled                              1,111.26
country effects                       274.59
country + year effects              1,149.28
```

That is not a typographical error, and it is not a bug — a design matrix built by hand with 45 columns of full rank returns the same number. Adding year effects moves the slope from 275 back up past the pooled estimate, to about 1,149. Three defensible specifications give answers spanning a factor of four, and a reader shown any one of them alone would draw a different conclusion.

The explanation is available here in a way it almost never is in real work, because these are fixtures and the generator is in the repository. Productivity was constructed with a deterministic trend of $+1.15$ index points per year; income was not given one. So the within-country variation in the predictor is dominated by a

common upward drift that the outcome does not share — variation in x with no matching variation in y, which attenuates the slope toward zero. That is the 275. Year effects remove precisely that common drift, along with the COVID shock the generator applied to both. What is left is idiosyncratic country-year movement, in which the same latent factor drives both variables strongly — and the slope comes back up.

The general lesson survives the specific example. **Each specification identifies β from a different slice of the variation, and if the slices carry different signal-to-noise ratios the estimates will differ — not because one is wrong, but because they are answering different questions.** Reporting a specification without saying which variation identifies it is reporting half a result. This is also why the modern literature is careful about two-way fixed effects when effects are heterogeneous: the estimator is a weighted average of many comparisons, and the weights are not always the ones you would choose (Chaisemartin and D'Haultfœuille 2020).

6.6 Random effects, and why the choice is testable

The alternative treats the unit effects as draws from a distribution rather than as parameters:

$$y_{it} \;=\; \mu + \beta x_{it} + u_i + \varepsilon_{it}, \qquad u_i \sim (0, \sigma_u^2).$$

This is more efficient when it is true, because it uses between as well as within variation, and it keeps time-invariant regressors estimable. It buys that with a strong assumption: u_i must be uncorrelated with the regressors. If countries with high unobserved capacity also have high productivity — which is exactly what the fixture's four country types encode — the assumption fails and random effects are biased.

The comparison is testable, because fixed effects are consistent whether or not the assumption holds while random effects are consistent only if it does. A systematic difference between the two therefore indicts the assumption (Hausman 1978). Mundlak's reformulation is the more illuminating one: include each unit's *mean* of the regressors alongside the regressors themselves, and the coefficient on the mean tests exactly the correlation at issue — turning the specification test into a coefficient you can look at (Mundlak 1978).

```
from linearmodels.panel import RandomEffects

re_fit = RandomEffects(panel[y], sm.add_constant(panel[[x]])).fit()
print(f"random effects slope   {re_fit.params[x]:>12,.2f}")
print(f"fixed effects slope    {fe.params[x]:>12,.2f}")

# Mundlak: add the country mean of x to a pooled regression.
aug = core.copy()
aug["x_mean"] = aug.groupby("geo")[x].transform("mean")
mund = sm.OLS(aug[y], sm.add_constant(aug[[x, "x_mean"]])).fit(
```

```
    cov_type="cluster", cov_kwds={"groups": aug["geo"]})

print(f"\nMundlak regression (clustered by country):")
print(f"  within coefficient on {x:18} {mund.params[x]:>12,.2f}")
print(f"  coefficient on the country mean          {mund.params['x_mean']:>1
      f"   p = {mund.pvalues['x_mean']:.4f}")
print("\nA country mean that matters is a unit effect correlated with the")
print("regressor — which is the random-effects assumption failing.")
```

```
random effects slope         817.47
fixed effects slope          274.59

Mundlak regression (clustered by country):
  within coefficient on productivity_idx         274.59
  coefficient on the country mean              1,187.89   p = 0.0000

A country mean that matters is a unit effect correlated with the
regressor — which is the random-effects assumption failing.
```

6.7 Clustered standard errors, properly this time

Chapter 3 showed that clustering by country raised the standard error on the pooled slope by about three quarters, and deferred the explanation. Here it is.

The classical formula assumes each of the 428 rows contributes independent information. It does not: residuals within a country are strongly correlated, so thirty countries observed fifteen times carry far less information than 450 independent observations would. The cluster-robust estimator replaces the independence assumption with independence *between* clusters, allowing arbitrary correlation within:

$$\widehat{\mathrm{Var}}(\hat{\beta}) = (X^\top X)^{-1}\Big(\sum_{g=1}^{G} X_g^\top e_g e_g^\top X_g\Big)(X^\top X)^{-1}.$$

The sandwich is the same shape as chapter 3's White estimator; only the middle changes, from a sum over observations to a sum over groups.

```
print(f"{'standard errors':34}{'SE':>10}{'t':>8}{'95% CI half-width':>20}")
variants = [
    ("classical", dict()),
    ("HC3 (heteroscedasticity only)", dict(cov_type="HC3")),
    ("clustered by country (30)", dict(cov_type="cluster",
                                       cov_kwds={"groups": core["geo"]})),
    ("clustered by year (15)", dict(cov_type="cluster",
                                    cov_kwds={"groups": core["time"]})),
    # statsmodels needs numeric group codes for two-way clustering.
    ("two-way clustered", dict(cov_type="cluster",
```

```
                                cov_kwds={"groups": np.column_stack([
                                    pd.factorize(core["geo"])[0],
                                    pd.factorize(core["time"])[0]])}),
]
for label, kw in variants:
    f = sm.OLS(core[y], sm.add_constant(core[[x]])).fit(**kw)
    b, se = f.params.iloc[1], f.bse.iloc[1]
    print(f"{label:34}{se:10.2f}{b/se:8.2f}{1.96*se:20,.0f}")
```

```
standard errors                        SE      t   95% CI half-width
classical                           41.50  26.77                 81
HC3 (heteroscedasticity only)       43.24  25.70                 85
clustered by country (30)           72.30  15.37                142
clustered by year (15)              74.28  14.96                146
two-way clustered                   94.34  11.78                185
```

Three points of discipline. **Cluster at the level at which shocks arrive**, not at the level that gives the answer you prefer — and if you are unsure, the question is about the sampling process rather than the data (Abadie et al. 2022). **The number of clusters is what matters, not the number of observations**: with thirty countries the asymptotics are thin, and with fewer than about forty clusters the cluster-robust estimator understates uncertainty and a bootstrap is the better tool. And **clustering is not a fix for a misspecified model** — it widens the interval around whatever the specification estimates, correct or not.

6.8 Does the slope differ by group? Ask it directly

Heterogeneity can be modelled rather than absorbed. An interaction lets the slope itself vary:

$$y_{it} \;=\; \alpha_i + \beta x_{it} + \delta\,(x_{it} \times g_i) + \varepsilon_{it},$$

where g_i marks a group. The coefficient δ *is* the difference in slopes, so its standard error is the test — which is the right way to compare two groups, rather than fitting separately and eyeballing whether intervals overlap, an error chapter 2 warned about.

```
med = core.groupby("geo")[y].mean().median()
core["rich"] = (core.groupby("geo")[y].transform("mean") > med).astype(float)
core["x_rich"] = core[x] * core["rich"]

inter = sm.OLS(core[y], sm.add_constant(core[[x, "rich", "x_rich"]])).fit(
    cov_type="cluster", cov_kwds={"groups": core["geo"]})

print(f"slope in lower-income countries      {inter.params[x]:>12,.2f}")
print(f"difference in slope (interaction)    {inter.params['x_rich']:>12,.2f}"
```

```
      f"   p = {inter.pvalues['x_rich']:.4f}")
print(f"slope in higher-income countries     "
      f"{inter.params[x] + inter.params['x_rich']:>12,.2f}")
print("\nThe interaction coefficient is the difference, so its p-value tests
print("the difference — which is the comparison you actually wanted.")
```

```
slope in lower-income countries            450.95
difference in slope (interaction)          789.93   p = 0.0000
slope in higher-income countries         1,240.88

The interaction coefficient is the difference, so its p-value tests
the difference — which is the comparison you actually wanted.
```

6.9 Yule's warning, which predates all of this

From the archive · Yule, 1903

Long before panel data existed, George Udny Yule showed that an association present in every subgroup of a table can vanish or reverse when the subgroups are combined, purely as an artefact of how the groups differ in size and composition (Yule 1903). Simpson revisited it half a century later, and the phenomenon carries his name in most textbooks and Yule's in the careful ones (Simpson 1951).

The eight countries with negative slopes in the opening figure are the same phenomenon in continuous form. A relationship that is negative within most units and strongly positive across them is not a contradiction and not a paradox — it is two different quantities, and the aggregate one is a weighted comparison between units that happens to run the other way.

The lesson is not that the aggregate is wrong. It is that **the direction of an association is a property of the level of aggregation, not of the world**, and that reporting one level without naming it is how the same data support opposite policy conclusions.

6.10 Get the functional form right first

One more result belongs here, because it is a chapter 3 loose end and a warning against reaching for panel machinery too early.

Chapter 3 found curvature in the residuals of the level-on-level regression and reported it as a specification failure. It was. The generator writes income as *exponential* in the latent factor and productivity as *linear* in it, so the true relationship between the two is log-linear, and no amount of fixed effects or clustering repairs a model fitted on the wrong scale.

```
from statsmodels.nonparametric.smoothers_lowess import lowess
```

```
print(f"{'specification':28}{'R²':>8}{'residual curvature':>22}")
for label, yv in [("level of income", core[y]), ("log of income", core["log_gdp
    m = sm.OLS(yv, sm.add_constant(core[[x]])).fit()
    r = m.resid / m.resid.std()
    sm_line = lowess(r, m.fittedvalues, frac=0.5, return_sorted=True)[:, 1]
    print(f"{label:28}{m.rsquared:8.4f}{sm_line.max() - sm_line.min():22.3f}")
print("\n(curvature = range of the smoother through the standardised residuals
print(" smaller is flatter is better specified)")

log_fe = PanelOLS(panel.assign(log_gdp=np.log(panel[y]))["log_gdp"],
                  sm.add_constant(panel[[x]]), entity_effects=True,
                  time_effects=True).fit()
print(f"\ntwo-way FE on log income: {log_fe.params[x]:+.5f} per index point")
print(f"  = {(np.exp(log_fe.params[x]) - 1) * 100:+.3f}% per index point")
```

```
specification                     R²    residual curvature
level of income               0.6273                 2.037
log of income                 0.7410                 1.404

(curvature = range of the smoother through the standardised residuals;
 smaller is flatter is better specified)

two-way FE on log income: +0.03441 per index point
  = +3.501% per index point
```

On the log scale the coefficient also becomes interpretable in the way economists want: a semi-elasticity, readable as a percentage change per unit of the predictor. That is not a cosmetic gain. “Zero point six per cent per index point” is a sentence a reader can check against their own knowledge; “1,149 euros per index point” is one they cannot, because it depends on the level.

6.11 A short review of the literature

The pooling problem is old. Yule (1903) demonstrated that associations reverse under aggregation, and Simpson (1951) gave the phenomenon its modern textbook treatment and, unfairly, its name.

The fixed-versus-random choice was settled as a testable proposition by Hausman (1978), whose test compares the two estimators on the logic that only one of them is consistent under the null. Mundlak (1978) reframed it more usefully: the random-effects estimator is a fixed-effects estimator with a restriction, and adding the unit means of the regressors makes the restriction a coefficient you can test directly. Mundlak’s formulation is easier to teach and harder to misapply, and it is the one used above.

Two literatures complicate the modern picture. On standard errors, Abadie et al. (2022) argues that clustering is a question about the sampling and assignment process rather

than a default to be applied, and that clustering when it is not needed costs precision for nothing. On the estimator itself, Chaisemartin and D'Haultfœuille (2020) shows that two-way fixed effects with heterogeneous effects is a weighted average of comparisons whose weights can be negative, which is a substantive problem rather than a technicality and returns in chapter 11.
Nickell (1981) is the classic result on what goes wrong when a lagged outcome is added to a fixed-effects model: the demeaning correlates the lag with the error, biasing the estimate by an amount that shrinks only as the number of periods grows. With fifteen years, that bias is not negligible.

6.12 What to report

1. **Which variation identifies the estimate.** Between, within, or within-and-within. State it, because the estimates differ by factors here and a reader cannot infer it from the coefficient.
2. **More than one specification, side by side.** Pooled, one-way, two-way. If they disagree, that disagreement is a finding and hiding it is a choice.
3. **What the fixed effects absorbed**, and therefore what you can no longer estimate. If your question concerns a time-invariant characteristic, say that fixed effects cannot answer it.
4. **The clustering level, with the reason.** How many clusters, why that level, and — below about forty — what you did about it.
5. **Heterogeneity tested, not assumed.** An interaction coefficient with its standard error, not two subgroup regressions compared by eye.
6. **The functional form, checked before the panel machinery.** Fixed effects on the wrong scale is a precise answer to the wrong question.
7. **What none of it identifies.** Fixed effects remove time-invariant confounders and nothing else. Time-varying confounding, reverse causation and selection all survive, and chapters 10 and 11 are about what it takes to address them.

Chapter 7

Principal Component and Factor Analyses

7.1 Six columns, how many things?

A dataset with six numeric columns does not necessarily measure six things. Budget, revenue, popularity, runtime, average rating and vote count are six columns describing a film; they are plainly not six independent facts about it, because expensive films earn more, popular films are voted on more, and almost everything correlates with almost everything. The question this chapter answers is how many distinct quantities are actually varying, and what they are.

Two methods answer it, they are routinely confused, and the confusion is not harmless. *Principal component analysis* is a rotation of the data: it finds new axes, ordered by how much variance they capture, with no model and no error term. *Factor analysis* posits unobserved causes and asks what would have to be true of them to produce the correlations you see. The first is a description; the second is a hypothesis.



Karl Pearson · 1857–1936
Principal components, 1901 — lines and planes of closest fit.
Elliott & Fry, undated. Public domain, via Wikimedia Commons.

From the archive · Pearson, 1901

Karl Pearson's paper is called *On lines and planes of closest fit to systems of points in space*, and the title contains the whole idea (Pearson 1901). He asked what line best fits a cloud of points when there is no distinguished variable to predict — when x and y are on the same footing and the question is the shape of the cloud rather

than the conditional mean of one coordinate given the other.
The answer differs from regression in a way worth seeing rather than being told. Least squares minimises *vertical* distances, because it treats y as the thing to be explained and x as given. Pearson's line minimises *perpendicular* distances, because neither coordinate is privileged. They are different lines, and the difference grows as the correlation weakens.
Hotelling gave the method its modern algebra and its name three decades later, deriving the components as the eigenvectors of the covariance matrix and establishing that they successively capture maximal variance (Hotelling 1933).

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, qmib

rng = np.random.default_rng(60033)
n = 90
xs = rng.normal(0, 1, n)
ys = 0.72 * xs + rng.normal(0, 0.72, n)
xs, ys = xs - xs.mean(), ys - ys.mean()

b_reg = np.polyfit(xs, ys, 1)[0]
M = np.column_stack([xs, ys])
_, _, Vt = np.linalg.svd(M, full_matrices=False)
b_pc = Vt[0, 1] / Vt[0, 0]

fig, ax = plt.subplots(figsize=(6.2, 4.8))
gx = np.linspace(xs.min() * 1.1, xs.max() * 1.1, 40)

# the residuals each method minimises
for xi, yi in zip(xs, ys):
    ax.plot([xi, xi], [yi, b_reg * xi], color=CL.muted, lw=0.6, alpha=0.55,
d = np.array([Vt[0, 0], Vt[0, 1]])
for xi, yi in zip(xs, ys):
    proj = (np.array([xi, yi]) @ d) * d
    ax.plot([xi, proj[0]], [yi, proj[1]], color=CL.warn, lw=0.6, alpha=0.5,

ax.plot(gx, b_reg * gx, color=CL.muted, lw=2.2, label=f"regression  (slope {
ax.plot(gx, b_pc * gx, color=CL.warn, lw=2.2, label=f"first component  (slop
ax.scatter(xs, ys, s=18, color=CL.accent, alpha=0.6, edgecolor="none", zorde
ax.set_xlabel("x"); ax.set_ylabel("y")
ax.legend(frameon=False, fontsize=8.5, loc="upper left")
```

```
ax.set_aspect("equal")
fig.tight_layout()
```

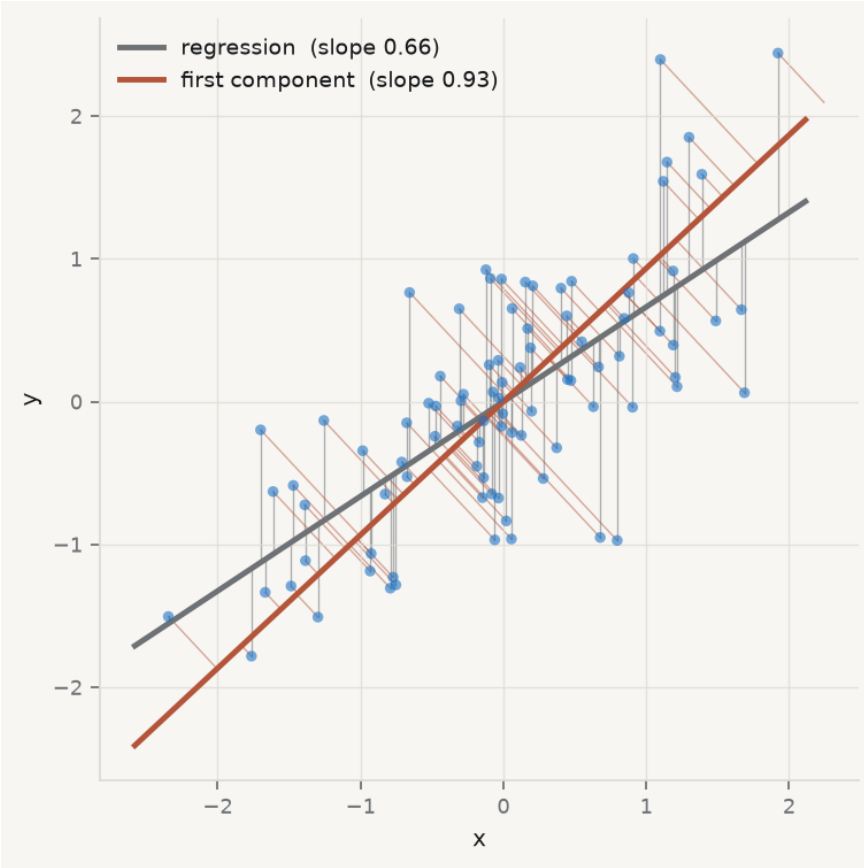


Figure 7.1. Regression and the first principal component are not the same line. Least squares minimises the vertical distances (grey); Pearson's line minimises the perpendicular ones (red). The regression line is always the flatter of the two.

7.2 Standardise, or the biggest units win

PCA maximises variance, and variance has units. A variable measured in hundreds of millions will have a variance many orders of magnitude larger than one measured from zero to ten, and the first component will therefore be, almost entirely, that variable. Nothing has been discovered; the units have been reported back.

```
movies = qmib.load("movies").select_dtypes("number").dropna()

# A data-quality problem first — see the section at the end of this chapter.
usable = movies[(movies["budget"] > 0) & (movies["revenue"] > 0)
                & (movies["runtime"] > 0)].copy()
X = usable.to_numpy(float)
cols = list(usable.columns)
print(f"rows used: {len(usable):,} of {len(movies):,}")

def eigen(A):
    ev, V = np.linalg.eigh(A)
    return ev[::-1], V[:, ::-1]

Xc = X - X.mean(0)
ev_raw, V_raw = eigen(np.cov(Xc, rowvar=False))
print(f"\nRAW UNITS: first component captures {ev_raw[0]/ev_raw.sum():.2%} o
print("  its loadings:")
for c, l in sorted(zip(cols, V_raw[:, 0]), key=lambda t: -abs(t[1])):
    print(f"    {c:14}{l:+8.4f}")

Z = (X - X.mean(0)) / X.std(0, ddof=1)
ev, V = eigen(np.corrcoef(Z, rowvar=False))
print(f"\nSTANDARDISED: first component captures {ev[0]/ev.sum():.2%}")
print(f"  eigenvalues: {np.round(ev, 3)}")
```

```
rows used: 5,369 of 45,203

RAW UNITS: first component captures 97.50% of variance
  its loadings:
    revenue        -0.9840
    budget         -0.1783
    vote_count     -0.0000
    popularity     -0.0000
    runtime        -0.0000
    vote_average   -0.0000

STANDARDISED: first component captures 47.32%
  eigenvalues: [2.839 1.203 0.831 0.632 0.306 0.189]
```

Ninety-seven and a half per cent of the variance on one component, loading almost entirely on revenue, and it means nothing at all. Revenue is measured in dollars and runtime in minutes; the analysis has ranked the columns by the size of their units. Standardising each variable to unit variance removes the arbitrariness, and it changes the answer completely: the first component now captures under half the variance, and there is a second worth looking at.

Standardisation is therefore part of the method rather than a preliminary — the same conclusion chapter 5 reached about the penalties, for the same reason. The practical consequence is that PCA is almost always performed on the *correlation* matrix, which is the covariance matrix of standardised data.

7.3 How many components?

> **Eigenvalue** — the variance of the data along a component. On standardised data the eigenvalues sum to the number of variables, so an eigenvalue of 1 corresponds to the variance of one original variable — a component that explains no more than a single column would have on its own.

That gives the oldest rule: keep components with eigenvalue above 1. It is a convention, not a test, and it is known to over-retain. Two better instruments exist, and they disagree usefully.

Cattell's *scree test* looks for the elbow: plot the eigenvalues in order and retain those above the point where the curve flattens into rubble (Cattell 1966). It is subjective by construction, which is a fair criticism and not a fatal one — the judgement is at least visible.

Horn's *parallel analysis* makes it empirical (Horn 1965). Generate data of the same shape with no correlation at all, compute its eigenvalues, and retain only components whose eigenvalue exceeds what pure noise of that size produces. It answers the right question — is this component larger than chance — and it is the method to use when you can.

```
n_obs, n_var = Z.shape
rng2 = np.random.default_rng(7)
sim = np.array([np.linalg.eigvalsh(np.corrcoef(
    rng2.normal(size=(n_obs, n_var)), rowvar=False))[::-1] for _ in range(200)])
noise_95 = np.percentile(sim, 95, axis=0)

fig, ax = plt.subplots(figsize=(6.6, 4.2))
k = np.arange(1, n_var + 1)
ax.plot(k, ev, marker="o", ms=6, lw=2, color=CL.accent, label="observed", zorde
ax.plot(k, noise_95, marker="s", ms=4, lw=1.5, ls="--", color=CL.warn,
        label="95th percentile of uncorrelated data", zorder=2)
ax.axhline(1.0, color=CL.muted, lw=1.2, ls=":", label="Kaiser: eigenvalue = 1")
ax.set_xlabel("component"); ax.set_ylabel("eigenvalue")
ax.set_xticks(k)
ax.legend(frameon=False, fontsize=8.5)
fig.tight_layout()

print(f"Kaiser (>1):            retain {(ev > 1).sum()}")
print(f"parallel analysis:      retain {(ev > noise_95).sum()}")
```

```
print(f"cumulative variance of the first two: {(ev[:2].sum()/ev.sum()):.1%}"

Kaiser (>1):            retain 2
parallel analysis:      retain 2
cumulative variance of the first two: 67.4%
```

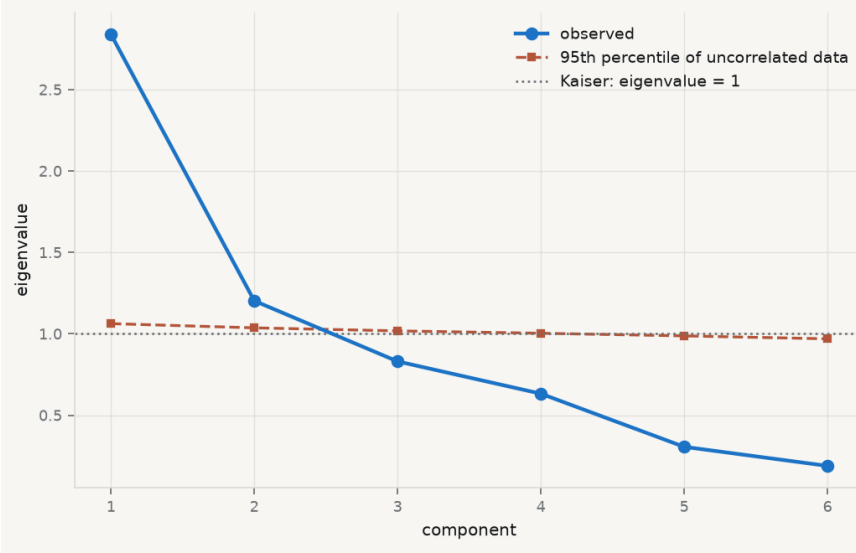


Figure 7.2. Choosing the number of components three ways. The Kaiser line at 1 retains two; the elbow suggests two; parallel analysis against uncorrelated data of the same shape retains the components lying above the dashed noise curve.

7.4 Loadings and scores are different objects

The single most common confusion in applied use, and it has consequences for what you are allowed to plot.

> **Loading** — an element of the eigenvector: how much an original *variable* contributes to a component. There are as many loadings per component as there are variables, and they describe what the component *means*.

> **Score** — the value of a component for an *observation*, computed as Zv_j. There are as many scores per component as there are rows, and they are the new coordinates you would use in a subsequent regression.

Loadings interpret; scores are data. A biplot shows both at once and is therefore the standard display, but the two are on different scales and the relative lengths of the loading arrows are meaningful only against each other.

```
scores = Z @ V[:, :2]

fig, ax = plt.subplots(figsize=(7.0, 5.4))
ax.scatter(scores[:, 0], scores[:, 1], s=6, color=CL.accent, alpha=0.18,
           edgecolor="none", zorder=1)
scale = np.abs(scores).max() * 0.72
for j, name in enumerate(cols):
    ax.arrow(0, 0, V[j, 0] * scale, V[j, 1] * scale, color=CL.warn,
             width=0.004, head_width=0.10, alpha=0.9, zorder=3)
    ax.text(V[j, 0] * scale * 1.12, V[j, 1] * scale * 1.12, name,
            fontsize=8.5, color=CL.ink, ha="center", va="center", zorder=4)
ax.axhline(0, color=CL.line, lw=1); ax.axvline(0, color=CL.line, lw=1)
ax.set_xlabel(f"component 1 ({ev[0]/ev.sum():.1%} of variance)")
ax.set_ylabel(f"component 2 ({ev[1]/ev.sum():.1%})")
fig.tight_layout()

print("loadings on the first two components:")
print(f"{'variable':14}{'PC1':>9}{'PC2':>9}")
for j, name in enumerate(cols):
    print(f"{name:14}{V[j,0]:+9.3f}{V[j,1]:+9.3f}")
```

```
loadings on the first two components:
variable            PC1      PC2
budget           -0.462   +0.298
popularity       -0.370   +0.078
revenue          -0.533   +0.182
runtime          -0.213   -0.605
vote_average     -0.207   -0.710
vote_count       -0.526   +0.023
```

7.5 PCA is not factor analysis

The two are distinguished by what they do with the variance that a variable does not share with the others.

PCA decomposes *all* the variance. Every component is a weighted sum of the observed variables, so the components are defined whatever the correlations are — even if there are none, PCA returns six components for six variables and simply reports eigenvalues near 1. It is a rotation, and rotations are always available.

Factor analysis splits the variance in two. Each observed variable is modelled as a combination of a few common factors plus something specific to itself:

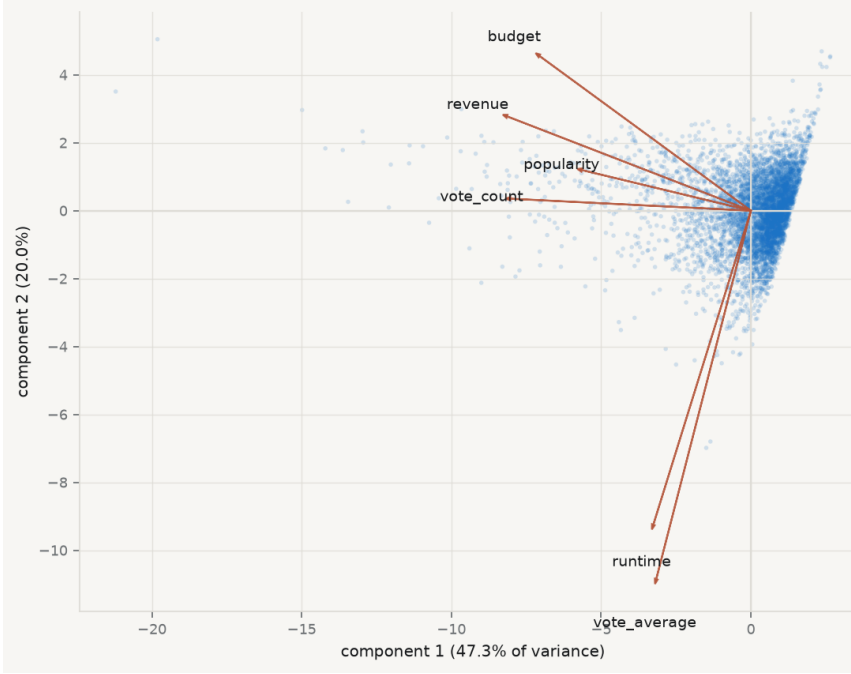


Figure 7.3. The first two components. Points are scores — one per film. Arrows are loadings — one per variable. The first component orders films by overall scale and reach; the second separates long, well-rated films from expensive, widely-voted ones.

$$x \;=\; \Lambda f + u, \qquad \operatorname{Var}(x) \;=\; \underbrace{\Lambda\Lambda^{\top}}_{\text{common}} + \underbrace{\Psi}_{\text{unique}},$$

with Ψ diagonal. The model asserts that the *correlations between* variables are produced entirely by the common factors, and that whatever is left is particular to each variable. That is a testable claim: a set of correlations may be incompatible with any small number of factors.

> **Communality** — the share of a variable's variance explained by the common factors, $\sum_j \lambda_{ij}^2$. Its complement is the *uniqueness*. A variable with low communality does not belong to the factor structure and is telling you so.

The practical difference: use PCA when the goal is to compress many correlated columns into a few for prediction or display, and factor analysis when you believe in a latent quantity and want to estimate it. Spearman's original argument for a general factor of intelligence is the founding example of the second (Spearman 1904), and the tradition of using PCA where the question was really factor-analytic is long and mostly unhelpful (Fabrigar et al. 1999).

7.6 A factor is identified only up to rotation

This is the deepest fact in the chapter and the one most often glossed.
If Λ reproduces the correlations, so does ΛR for *any* orthogonal matrix R, because

$$(\Lambda R)(\Lambda R)^{\top} \;=\; \Lambda R R^{\top}\Lambda^{\top} \;=\; \Lambda\Lambda^{\top}.$$

The model fits exactly as well. Every communality is unchanged. Nothing observable distinguishes them — and yet the loadings, which is to say the *interpretation*, can be completely different. There is no statistical basis for preferring one rotation; the choice is made on grounds of interpretability, and rotation criteria such as varimax formalise a preference for loadings that are large or near zero rather than middling (Kaiser 1958).

```
from sklearn.decomposition import FactorAnalysis
from sklearn.preprocessing import StandardScaler

# The car-attribute battery: fourteen rated features, ninety respondents.
cars = qmib.load("efa").select_dtypes("number").dropna()
Zc = StandardScaler().fit_transform(cars)

fa = FactorAnalysis(n_components=3, random_state=0).fit(Zc)
L = fa.components_.T
```

```
def varimax(L, tol=1e-6, iters=200):
    """Kaiser's criterion, by the standard SVD iteration."""
    p, k = L.shape
    R, d = np.eye(k), 0.0
    for _ in range(iters):
        d_old, Lam = d, L @ R
        B = L.T @ (Lam**3 - Lam @ np.diag(np.diag(Lam.T @ Lam)) / p)
        u, s, vt = np.linalg.svd(B)
        R, d = u @ vt, s.sum()
        if d_old and d / d_old < 1 + tol:
            break
    return L @ R, R


Lr, R = varimax(L)

print(f"n = {len(cars)}, variables = {cars.shape[1]}, factors = 3\n")
print(f"{'':22}{'unrotated':>14}{'varimax':>14}")
print(f"{'||Lambda Lambda^T||':22}{np.linalg.norm(L @ L.T):14.8f}"
      f"{np.linalg.norm(Lr @ Lr.T):14.8f}")
print(f"{'sum of communalities':22}{(L**2).sum():14.8f}{(Lr**2).sum():14.8f}")
print(f"\nR is orthogonal: ||R'R - I|| = {np.linalg.norm(R.T @ R - np.eye(3)):.2e}")
print("\nIdentical fit. Now the interpretation:")
tab = pd.DataFrame(Lr, index=cars.columns)
for c in tab.columns:
    print(f"  factor {c+1}: {', '.join(tab[c].abs().nlargest(3).index)}")
```

```
n = 90, variables = 14, factors = 3

                          unrotated       varimax
||Lambda Lambda^T||       2.79837923    2.79837923
sum of communalities      4.56171180    4.56171180

R is orthogonal: ||R'R - I|| = 5.38e-16

Identical fit. Now the interpretation:
  factor 1: space_comfort, fuel_type, after_sales_service
  factor 2: resale_value, maintenance, fuel_efficiency
  factor 3: testimonials, color, fuel_efficiency
```

Two solutions, the same fit to eight decimal places, and different stories. If a paper reports rotated loadings without naming the rotation, it has not reported its result.

7.7 What you may not call the components

A component is a weighted sum of variables that happens to capture variance. It is not a thing, it has no units, and its sign is arbitrary — multiply an eigenvector by -1 and it is still an eigenvector, so "high on component 1" can be reversed by a convention in the software.

The temptation is to name it anyway. The first component above loads positively on budget, revenue, votes and popularity, so it is tempting to call it *commercial success* and then to treat that name as a measurement. Three things are wrong with doing so.

The name is an interpretation, not a finding. It is your reading of a loading pattern, and a different reader may propose a different name that fits the same numbers.

Naming invites reification. Once "commercial success" has a column, it gets used in regressions, and its coefficient gets discussed as though the quantity existed independently of the six columns it was built from.

The component is sample-specific. Add films, drop a variable, or restrict to one decade, and the loadings move. A named factor should be stable across reasonable perturbations, and that is checkable — refit on resamples and look.

The disciplined form is to say what the component *is*: "the first principal component, which loads positively on all six variables and most strongly on revenue and vote count, and accounts for 47% of the standardised variance". That sentence is defensible. "A film's commercial success" is a claim about the world.

7.8 The zeros, which are not zeros

One data-quality note, because it changes every number in this chapter and it is invisible in a correlation matrix.

```
print(f"{'variable':14}{'zeros':>10}{'share':>9}")
for c in movies.columns:
    z = int((movies[c] == 0).sum())
    print(f"{c:14}{z:>10,}{z/len(movies):>9.1%}")
print(f"\nrows with budget, revenue and runtime all positive: "
      f"{len(usable):,} of {len(movies):,} ({len(usable)/len(movies):.1%})")
```

```
variable           zeros    share
budget            36,323    80.4%
popularity            60     0.1%
revenue           37,801    83.6%
runtime            1,558     3.4%
vote_average       2,899     6.4%
vote_count         2,800     6.2%

rows with budget, revenue and runtime all positive: 5,369 of 45,203 (11.9%)
```

Four out of five films in this dataset have a budget of exactly zero, and five out of six have revenue of exactly zero. No film costs nothing. These are missing values recorded as the number 0 — which arithmetic cannot distinguish from a measurement, so every mean, every correlation and every eigenvalue computed on the full table is contaminated.

This is chapter 2's missingness section arriving with consequences. The correlation matrix on the full data is not a weaker version of the truth; it is a description of a mixture of films and non-observations. Dropping to the 5,369 complete rows is the minimum defensible response, and it is not neutral either: films with recorded budgets are larger, better documented and more recent, so the analysis now describes that subset. Say so.

7.9 A short review of the literature

Pearson (1901) posed the problem geometrically — the best-fitting line and plane to a cloud of points, minimising perpendicular rather than vertical distance — and Hotelling (1933) supplied the eigen-decomposition and the name. The two papers are worth reading in sequence for how differently the same method looks from a geometric and an algebraic starting point.

Factor analysis has a separate origin in Spearman (1904), whose argument for a general factor from correlations among school subjects established the model of observed variables as combinations of unobserved ones. The distinction from PCA was blurred almost immediately, and Fabrigar et al. (1999) documents how routinely PCA is reported where the research question was factor-analytic, along with the consequences.

On how many to retain, Cattell (1966) introduced the scree test, whose subjectivity is its standard criticism and its visible honesty; Horn (1965) replaced the judgement with a comparison against uncorrelated data of the same dimensions, which is the method used above and still the best available.

Kaiser (1958) gave rotation its dominant criterion. The underlying indeterminacy — that loadings are identified only up to an orthogonal transformation — is not a defect of any method but a property of the model, and it means that the interpretive step is a choice the analyst makes and must disclose.

7.10 What to report

1. **That the inputs were standardised**, and on what matrix the decomposition was performed. Without it the result is a ranking of measurement units.
2. **The full eigenvalue spectrum**, not only the retained ones, and the cumulative variance.
3. **How many components you kept and by what rule** — with parallel analysis if you can, and the scree plot either way so the reader can disagree.
4. **Loadings, with the rotation named.** Unrotated, varimax, oblimin: they fit identically and mean different things.
5. **Communalities**, if the model is factor analysis. A variable the factors do not explain should be discussed, not silently retained.

6. **The description rather than the name.** What the component loads on, and how much variance it carries. If you must give it a label, mark it as a label.
7. **What the components are not.** They are not measurements, they have arbitrary sign, they are specific to this sample and these variables, and a regression using them as predictors inherits all of that.

Chapter 8

K-Nearest Neighbours and the Bias–Variance Trade-off

8.1 A method with no model

Every method so far has assumed a form. Least squares assumes a line, logistic regression assumes a curve, and the penalties in chapter 5 assume the line is worth shrinking toward. Each assumption is a claim that can be wrong, and chapter 3 was about detecting when it is.

Nearest neighbours makes no such claim. To predict for a new observation, find the k observations in the training data closest to it, and average their outcomes:

$$\hat{f}(x_0) \;=\; \frac{1}{k} \sum_{i \in N_k(x_0)} y_i,$$

where $N_k(x_0)$ is the set of the k nearest training points. For a classification problem, take the majority vote instead. There is no fitting stage, no coefficient, no optimisation — the training data *is* the model.

This looks like getting something for nothing, and the chapter is about what it actually costs. The method is not assumption-free; it has simply moved its assumption. Averaging the neighbours presumes that the function is smooth enough that points close in x have similar y, and it presumes you know what "close" means. Both presumptions can fail badly, and neither shows up as a diagnostic plot.

No portrait accompanies this chapter. The method is due to Evelyn Fix and Joseph Hodges, in a 1951 technical report for the US Air Force School of Aviation Medicine that went unpublished for nearly forty years (Fix and Hodges 1989), and no freely licensed likeness of either author appears to exist. The convention this book follows is a real portrait or none.

8.2 The floor nobody beats

Before asking how well a classifier does, ask how well anything could.

> **Bayes classifier** — the rule that assigns each x to the most probable class given x: predict class j where $\Pr(Y = j \mid X = x)$ is largest. Its error rate, the *Bayes rate*, is the lowest achievable by any classifier whatsoever.

The proof is a one-liner: at each x, any rule either picks the most probable class or does not, and picking it minimises the chance of being wrong there. Integrating over x gives the result.

The Bayes classifier is not a method, because computing it requires $\Pr(Y \mid X)$ — which is the thing you are trying to estimate. Its value is conceptual and it is considerable. **A non-zero Bayes rate means that error is not evidence of a bad**

model. If the outcome is genuinely uncertain given the predictors, no amount of flexibility, data or cleverness removes the remaining error, and a practitioner who keeps adding complexity to chase it is chasing noise.
The regression analogue is the irreducible error σ^2: the variance of y around $f(x)$, which no estimator of f can touch. It appears explicitly in the decomposition below.

8.3 Where prediction error comes from

Take a target $y = f(x) + \varepsilon$ with $\mathrm{E}[\varepsilon] = 0$ and $\mathrm{Var}(\varepsilon) = \sigma^2$. Let $\hat{f}$ be an estimate built from a random training sample. The expected squared error at a point x_0, averaged over training samples and over the noise in a new observation, splits into three:

$$\mathrm{E}\big[(y_0 - \hat{f}(x_0))^2\big] \;=\; \underbrace{\sigma^2}_{\text{irreducible}} \;+\; \underbrace{\big(\mathrm{E}[\hat{f}(x_0)] - f(x_0)\big)^2}_{\text{bias}^2} \;+\; \underbrace{\mathrm{Var}\big(\hat{f}(x_0)\big)}_{\text{variance}}.$$

Chapter 5 used the last two terms to argue for a biased estimator. Here all three matter, and the first is the one that makes the others meaningful: because $\sigma^2 > 0$, the total error has a floor, and the *only* quantity a modeller controls is how the remaining budget is split between bias and variance.
The words are worth being precise about. **Bias** is how far the method's *average* prediction is from the truth — an error it makes systematically, in every sample. **Variance** is how much the prediction moves when the training sample changes — an error it makes differently each time. A rigid method has high bias and low variance; a flexible one has the reverse.
For nearest neighbours, k is the dial. Small k averages few points, follows the data closely, and changes a lot when the data change: low bias, high variance. Large k averages many points, smooths across regions where the truth differs, and barely moves between samples: high bias, low variance.
The decomposition is usually presented as algebra and left there. It is measurable, if you are willing to simulate a world whose truth you know.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, qmib
from sklearn.neighbors import KNeighborsRegressor

rng = np.random.default_rng(60033)
```

```
def truth_fn(x):
    return np.sin(2 * np.pi * x) + 0.5 * x

SIGMA, N, REPS = 0.35, 120, 400
x_test = np.linspace(0.05, 0.95, 60).reshape(-1, 1)
f_true = truth_fn(x_test.ravel())

ks = [1, 2, 3, 5, 8, 12, 20, 30, 45, 60, 80]
rows = []
for k in ks:
    preds = np.empty((REPS, len(x_test)))
    mses = np.empty(REPS)
    for r in range(REPS):
        xs = rng.uniform(0, 1, N).reshape(-1, 1)
        ys = truth_fn(xs.ravel()) + rng.normal(0, SIGMA, N)
        preds[r] = KNeighborsRegressor(n_neighbors=k).fit(xs, ys).predict(x_
        mses[r] = np.mean((f_true + rng.normal(0, SIGMA, len(x_test)) - pred
    rows.append(dict(k=k,
                     bias2=np.mean((preds.mean(0) - f_true) ** 2),
                     var=np.mean(preds.var(0)),
                     mse=mses.mean()))

bv = pd.DataFrame(rows)
bv["total"] = bv.bias2 + bv["var"] + SIGMA ** 2

fig, ax = plt.subplots(figsize=(7.2, 4.6))
ax.plot(bv.k, bv.bias2, marker="o", ms=4, lw=1.8, color=CL.warn, label="bias
ax.plot(bv.k, bv["var"], marker="s", ms=4, lw=1.8, color=CL.accent, label="v
ax.axhline(SIGMA ** 2, ls=":", lw=1.4, color=CL.muted, label=r"irreducible $
ax.plot(bv.k, bv.total, marker="^", ms=4, lw=2.2, color=CL.ink, label="sum")
ax.plot(bv.k, bv.mse, ls="--", lw=1.4, color=CL.good, label="measured test M
best = bv.loc[bv.total.idxmin(), "k"]
ax.axvline(best, color=CL.muted, lw=1, alpha=0.6)
ax.set_xlabel("$k$"); ax.set_ylabel("expected squared error")
ax.set_xscale("log"); ax.set_xticks(ks); ax.set_xticklabels(ks)
ax.legend(frameon=False, fontsize=8.5)
fig.tight_layout()

print(f"{'k':>4}{'bias²':>10}{'variance':>11}{'σ²':>9}{'sum':>10}{'test MSE'
for _, r in bv.iterrows():
    print(f"{int(r.k):>4}{r.bias2:10.5f}{r['var']:11.5f}{SIGMA**2:9.5f}"
          f"{r.total:10.5f}{r.mse:11.5f}")
print(f"\nminimum total error at k = {int(best)}")
```

```
   k     bias²   variance       σ²       sum   test MSE
```

```
  1   0.00023    0.12467  0.12250   0.24741    0.24868
  2   0.00014    0.06256  0.12250   0.18520    0.18548
  3   0.00009    0.04060  0.12250   0.16318    0.16148
  5   0.00006    0.02551  0.12250   0.14807    0.14673
  8   0.00011    0.01625  0.12250   0.13886    0.14049
 12   0.00032    0.01164  0.12250   0.13446    0.13424
 20   0.00307    0.00889  0.12250   0.13446    0.13501
 30   0.01376    0.00769  0.12250   0.14395    0.14445
 45   0.03776    0.00673  0.12250   0.16699    0.16619
 60   0.06789    0.00666  0.12250   0.19705    0.19732
 80   0.16629    0.00783  0.12250   0.29661    0.29528

minimum total error at k = 12
```

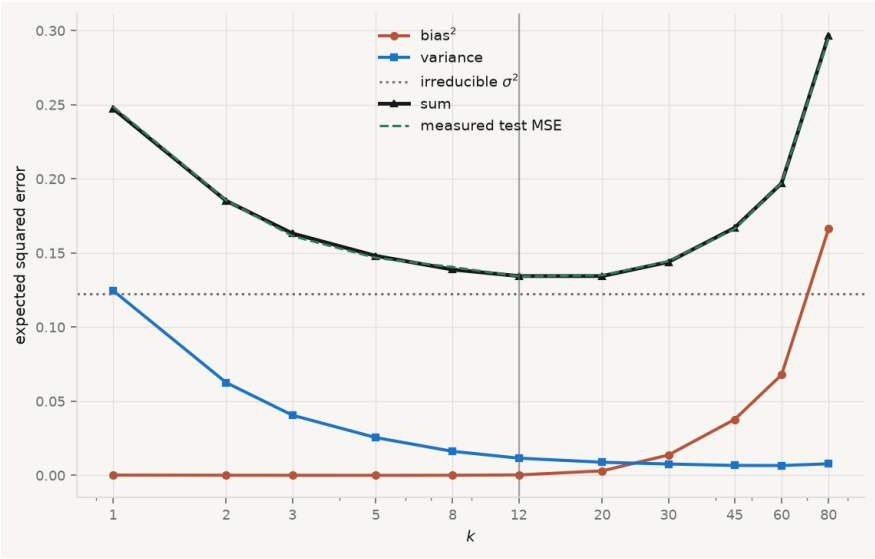


Figure 8.1. The decomposition, measured. Four hundred training samples are drawn from a known function; for each k the squared bias and the variance of the prediction are computed across samples. Their sum plus the irreducible error tracks the test error to within 0.002 everywhere.

Read the table rather than the shape. Variance falls by a factor of sixteen between $k = 1$ and $k = 80$; squared bias rises by a factor of several hundred. The sum is U-shaped with a minimum in between, and the identity holds — the sum of the three estimated components matches the independently measured test error at every k. That is the decomposition being *verified*, not illustrated.

Note also what happens at the extremes. At $k = 1$ the method has essentially no bias and all the error is variance plus noise. At $k = 80$, two thirds of the sample

is averaged into every prediction and the bias dominates. Neither extreme is a modelling failure; both are the same dial at different settings.

> **Effective degrees of freedom** — for k-nearest neighbours, roughly n/k. At $k = 1$ the method has as many effective parameters as observations, which is why it interpolates the training data exactly; at $k = n$ it has one, and predicts the global mean everywhere. It is the same quantity chapter 5's penalty controlled continuously.

8.4 Distance is not given, it is chosen

Nearest neighbours needs a definition of "near". The default is Euclidean distance,

$$d(x_0, x_i) \;=\; \sqrt{\textstyle\sum_{j=1}^{p} (x_{0j} - x_{ij})^2},$$

and that formula adds squared differences measured in different units. A difference of one euro and a difference of one index point are treated as identical quantities. In practice the variable with the largest numerical range determines the neighbours entirely, and every other predictor is decoration.

This is the third time this book has reached the same conclusion — PCA in chapter 7, the penalties in chapter 5, and now here — and the reason is always the same: any method that compares magnitudes across variables must first make them comparable. **Standardise before computing a distance.** The alternative is a method whose answer depends on whether you recorded revenue in dollars or millions of dollars.

8.5 The curse of dimensionality

The deeper problem is that "near" stops existing as the number of predictors grows, and it does so much faster than intuition suggests.

```
rng2 = np.random.default_rng(7)
print(f"{'dimensions':>11}{'nearest':>10}{'farthest':>10}{'contrast':>11}")
for p in [1, 2, 5, 10, 25, 50, 100, 500]:
    X = rng2.uniform(0, 1, (500, p))
    q = rng2.uniform(0, 1, (1, p))
    dist = np.sqrt(((X - q) ** 2).sum(axis=1))
    print(f"{p:>11}{dist.min():10.3f}{dist.max():10.3f}"
          f"{(dist.max() - dist.min())/dist.min():11.3f}")
```

```
 dimensions   nearest  farthest   contrast
          1     0.000     0.579   1495.857
          2     0.036     1.129     30.482
          5     0.201     1.335      5.650
```

```
  10     0.665     1.842     1.769
  25     1.358     2.779     1.046
  50     2.319     3.559     0.535
 100     3.236     4.601     0.422
 500     8.126     9.851     0.212
```

The last column is what matters: the gap between the nearest and the farthest point, relative to the nearest. In one dimension the farthest point is around a thousand times further away than the nearest. By fifty dimensions it is half again as far; by five hundred, a fifth. All the points are approximately the same distance from the query, so "the ten nearest" is close to a random selection of ten (Beyer et al. 1999).

There is a second, geometric way to see the same thing. To capture a fixed fraction r of the data in a hypercube of p dimensions, the neighbourhood must span $r^{1/p}$ of the range of *each* variable. Capturing 1% of the data needs 10% of the range in two dimensions, 63% in ten, and 95% in a hundred — a "neighbourhood" covering almost the entire space, which is not local in any useful sense.

The consequence for practice is blunt. Nearest neighbours works well with few predictors and abundant data, and degrades quickly as predictors are added. Adding a variable to a linear model costs one parameter; adding one to a nearest neighbour method costs a dimension of the space in which distance must remain meaningful.

8.6 Choosing k, and why training error lies

The training error of 1-nearest-neighbour is exactly zero: every training point is its own nearest neighbour, so it predicts itself perfectly. That number is not a measure of anything, and any procedure that selects k by fitting error will select $k = 1$ every time.

Cross-validation is the answer, as it was in chapter 5. Split, fit on the rest, measure on the held-out part, rotate.

```
from sklearn.model_selection import cross_val_score, KFold

xs = rng.uniform(0, 1, 200).reshape(-1, 1)
ys = truth_fn(xs.ravel()) + rng.normal(0, SIGMA, 200)
grid = [1, 2, 3, 5, 8, 12, 20, 30, 45, 60, 90, 130]

train_err, cv_err = [], []
for k in grid:
    m = KNeighborsRegressor(n_neighbors=k).fit(xs, ys)
    train_err.append(np.mean((ys - m.predict(xs)) ** 2))
    cv_err.append(-cross_val_score(KNeighborsRegressor(n_neighbors=k), xs, ys,
                                   cv=KFold(10, shuffle=True, random_state=0),
                                   scoring="neg_mean_squared_error").mean())
```

```
fig, ax = plt.subplots(figsize=(7.0, 4.3))
ax.plot(grid, train_err, marker="o", ms=4, lw=1.8, color=CL.muted, label="tr
ax.plot(grid, cv_err, marker="s", ms=4, lw=2.0, color=CL.accent, label="10-f
ax.axhline(SIGMA ** 2, ls=":", lw=1.4, color=CL.warn, label=r"irreducible $\
ax.axvline(grid[int(np.argmin(cv_err))], color=CL.ink, lw=1, ls="--")
ax.set_xlabel("$k$"); ax.set_ylabel("mean squared error")
ax.set_xscale("log"); ax.set_xticks(grid); ax.set_xticklabels(grid)
ax.legend(frameon=False, fontsize=8.5)
fig.tight_layout()

print(f"training error at k=1: {train_err[0]:.6f}  (exactly zero — every poi
print(f"CV chooses k = {grid[int(np.argmin(cv_err))]}, CV error {min(cv_err)
print(f"irreducible floor σ² = {SIGMA**2:.4f}")
```

```
training error at k=1: 0.000000  (exactly zero — every point is its own neighbour
CV chooses k = 20, CV error 0.1484
irreducible floor σ² = 0.1225
```

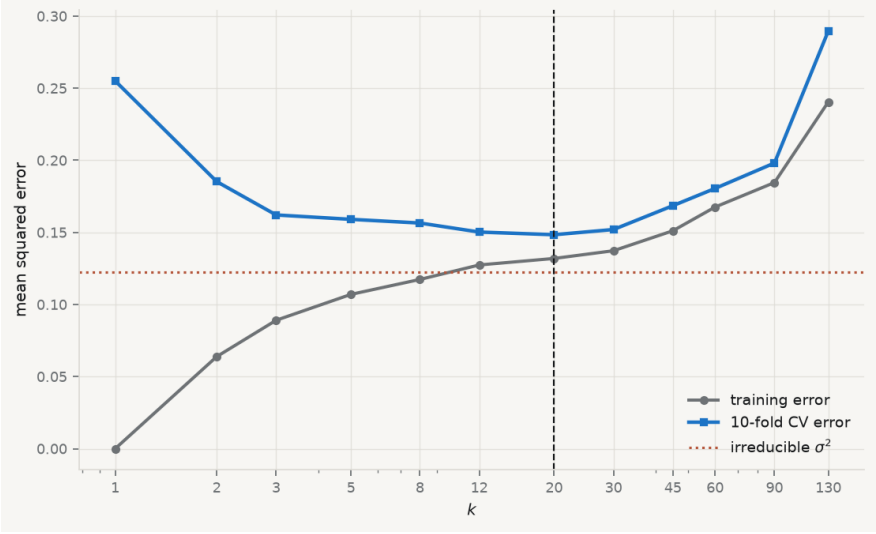


Figure 8.2. Training error against cross-validated error for the same data. Training error rises monotonically with k and is zero at $k = 1$; only the cross-validated curve has a minimum, and it is not at $k = 1$.

Two things are visible. The training curve is monotone and useless for selection. And the cross-validated minimum sits close to, but above, the irreducible floor — which is what a well-tuned method looks like. A cross-validated error *below* σ^2 would indicate a leak between the folds rather than a triumph.

8.7 Did the flexibility earn its keep?

Now the real question, on real data. Nearest neighbours is flexible; flexibility costs variance; and the only justification for paying is that the truth is non-linear in a way a linear model misses. Whether that holds is an empirical question, and it is answered by comparison rather than by assertion.

```
from sklearn.model_selection import StratifiedKFold
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

core = qmib.load("core").dropna(
    subset=["falling_behind_next", "gdp_pc_eur", "productivity_idx",
            "employment_ths", "gfcf_meur"])
Xc = core[["gdp_pc_eur", "productivity_idx", "employment_ths", "gfcf_meur"]]
yc = core["falling_behind_next"]
cv = StratifiedKFold(5, shuffle=True, random_state=0)

print(f"n = {len(core)}   base rate = {yc.mean():.3f}   "
      f"majority-class accuracy = {1 - yc.mean():.4f}\n")
print(f"{'model':30}{'accuracy':>10}{'ROC AUC':>10}")
models = [
    ("kNN, k=5, standardised",  make_pipeline(StandardScaler(), KNeighborsClass
    ("kNN, k=25, standardised", make_pipeline(StandardScaler(), KNeighborsClass
    ("logistic regression",     make_pipeline(StandardScaler(), LogisticRegress
]
for name, mdl in models:
    acc = cross_val_score(mdl, Xc, yc, cv=cv, scoring="accuracy").mean()
    auc = cross_val_score(mdl, Xc, yc, cv=cv, scoring="roc_auc").mean()
    print(f"{name:30}{acc:10.4f}{auc:10.4f}")
```

```
n = 385   base rate = 0.101   majority-class accuracy = 0.8987

model                          accuracy   ROC AUC
kNN, k=5, standardised           0.8909    0.5147
kNN, k=25, standardised          0.8987    0.5609
logistic regression              0.8987    0.6754
```

The result is unflattering to the method this chapter is about, and it is the most useful output in it.

On accuracy, nothing beats predicting the majority class. All three models score about 0.899, which is exactly what you get by declaring that no country falls behind, ever. Chapter 4 warned about this: with a base rate of 10%, accuracy is a measure of the base rate rather than of the model.

On AUC, the models separate completely. Nearest neighbours at $k = 5$ scores 0.51 — a coin flip, no information whatsoever. At $k = 25$ it recovers to 0.56. Logistic regression, the rigid method with the strong assumption, reaches 0.68. So the flexibility did not earn its keep. There is signal in these predictors, a linear model finds it, and the flexible method spends its budget on variance and comes back with almost nothing. Had the comparison been made on accuracy alone, all three would have looked identical and the conclusion would have been that nothing works — which is false, and would have been the wrong lesson.

8.8 A worked example: can the market's past predict today?

The lecture runs one complete prediction exercise, the way it would be done by someone whose job depended on it, and the chapter repeats it here because every step carries a rule. The question is whether the S&P 500's return two days ago says anything about whether the index rises or falls today. The data are `Smarket` from *An Introduction to Statistical Learning*: 1,250 trading days, 2001 to 2005, with the five previous days' returns (`Lag1` to `Lag5`) and the day's `Direction`.

The work is split into two files, because in practice the two jobs happen at different times. The first is the laboratory: it chooses K, judges the model once, and saves it. The second is the trading desk: it loads the saved model and predicts one new date, with no data to download and no training. Both are published with the lecture as `session08_1_train.py` and `session08_2_predict.py`.

Two decisions come before any model. One late day, 28 December 2005, is put aside to play "tomorrow", and the two days after it are dropped, because on the morning of the 28th they are the future. The rest is split **by time**: the model learns from 2002–2004 and is judged on 2005. A random split would put days of 2005 into the training data, and the model would learn from the future to "predict" the past.

Cross-validation has to respect time as well. The ordinary k-fold scheme of chapter 5 would fit on 2004 to predict 2002. For a time series each round fits on everything up to a date and validates on the block that follows, which is what scikit-learn's `TimeSeriesSplit` does.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

URL = "https://raw.githubusercontent.com/intro-stat-learning/ISLP/main/ISLP/
smarket = pd.read_csv(URL)
smarket = smarket[smarket["Year"] >= 2002]
predictors = ["Lag2"]
```

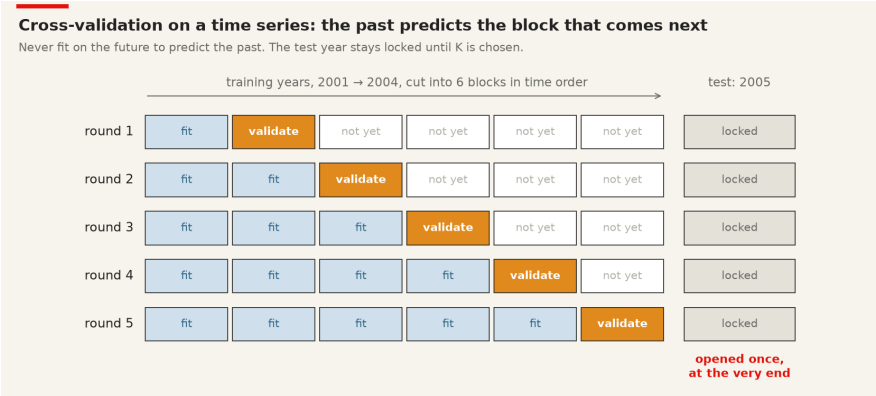


Figure 8.3. Cross-validation on a time series: each round fits on the past and validates on the next block, never the reverse.

```
new_day, history = smarket.iloc[[-3]], smarket.iloc[:-3]     # 28 Dec 2005, and
train, test = history[history["Year"] < 2005], history[history["Year"] == 2005]
X_train, y_train = train[predictors], train["Direction"]
X_test, y_test = test[predictors], test["Direction"]

# The scaler sits inside the pipeline, so each round standardises on its own fi
k_values = list(range(1, 126, 2))          # odd, so a two-class vote cannot ti
search = GridSearchCV(make_pipeline(StandardScaler(), KNeighborsClassifier()),
                      {"kneighborsclassifier__n_neighbors": k_values},
                      cv=TimeSeriesSplit(n_splits=5))
search.fit(X_train, y_train)

best_k = search.best_params_["kneighborsclassifier__n_neighbors"]
majority = y_train.value_counts(normalize=True).max()
print(f"best K = {best_k}, cross-validated accuracy = {search.best_score_:.3f}"
print(f"always predicting the commonest direction = {majority:.3f}")

plt.figure(figsize=(7, 3.2))
plt.plot(k_values, search.cv_results_["mean_test_score"], "o-", markersize=3,
         label="KNN, cross-validated accuracy")
plt.axhline(majority, color="red", linestyle="--", label="always the commonest
plt.axvline(best_k, color="grey", linestyle=":", label=f"chosen K = {best_k}")
plt.xlabel("K"); plt.ylabel("accuracy on the hidden blocks")
plt.legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()
```

```
best K = 123, cross-validated accuracy = 0.511
```

```
always predicting the commonest direction = 0.516
```

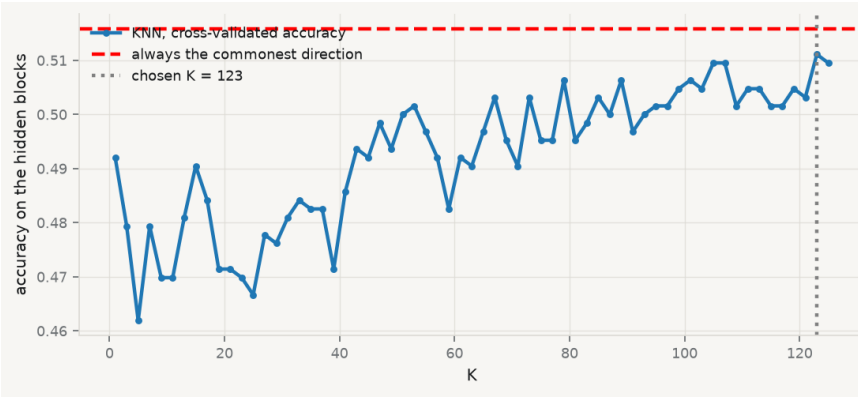


Figure 8.4. Cross-validated accuracy of KNN on the 2002–2004 training days, for every odd K from 1 to 125, against always predicting the commonest direction. No K clears the line.

The search ran the cross-validation loop for each of 63 values of K and kept $K = 123$, with an accuracy of 0.511 on the hidden blocks. "Best" means best of the values tried, not good: it is just below the 0.516 of always predicting the commonest direction. Cross-validation is already saying that `Lag2` carries little information about today.

The test year is opened once. K is now fixed, and only now is 2005 consulted. If K were changed after seeing this score, 2005 would have become training data and the score would stop being a test.

```
from sklearn.metrics import confusion_matrix

y_pred = search.predict(X_test)
table = pd.DataFrame(confusion_matrix(y_test, y_pred, labels=["Down", "Up"])
                     index=["actual Down", "actual Up"],
                     columns=["predicted Down", "predicted Up"])
print(table, "\n")
print(f"KNN accuracy on 2005:                      {(y_pred == y_test).mean(
print(f"always predicting the commonest direction: {y_test.value_counts(norm
```

```
             predicted Down  predicted Up
actual Down              35            74
actual Up                31           109

KNN accuracy on 2005:                      0.578
always predicting the commonest direction: 0.562
```

The diagonal holds the right calls, 35 down days and 109 up days, so accuracy is $(35 + 109)/249 = 0.578$, a little better than "always up" at 0.562. Read by row, the model catches 78% of the up days but only 32% of the down days: it says "up" on 183 of 249 days, and much of its accuracy is the market rising more often than not.

Training and predicting are different jobs. Before it is used, the model is refitted on every day it is allowed to know, with the chosen K, and saved to a file. The second script loads that file and predicts the new day from one number, the return two trading days earlier.

```
import joblib

final_model = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors=
final_model.fit(history[predictors], history["Direction"])
joblib.dump(final_model, "knn_smarket.joblib")           # file 1 ends here

model = joblib.load("knn_smarket.joblib")                # file 2 starts here
new_date = pd.DataFrame({"Lag2": [0.043]})               # 23 December 2005: +0
print("prediction for 28 December 2005:", model.predict(new_date)[0])
for label, prob in zip(model.classes_, model.predict_proba(new_date)[0]):
    print(f"   Pr({label}) = {prob:.3f}")

prediction for 28 December 2005: Up
   Pr(Down) = 0.423
   Pr(Up) = 0.577
```

The probabilities are the votes of the K nearest past days. The index did rise on 28 December 2005, so the prediction was right — and one date proves nothing either way. The test year is how the model is judged; the second file is how it is used.

> **How this example was chosen.** The example was built to work, and the lecture says so. The natural first attempt — both lags, all five years, the very last day as the new date — scored 50.6% on 2005, below the 56.2% of "always up", and got its date wrong. So 54 slices of the data were tried (predictors, training years, test year) and one that wins was kept. On 249 test days, accuracy moves by about three points from chance alone ($\sqrt{0.25/249} \approx 0.032$); among 54 tries, a winner that beats the baseline by 1.6 points is easy to find with no signal at all. And every look at 2005 made 2005 training data: the test was opened 54 times, not once. This is the garden of forking paths (Gelman and Loken 2014), and it is how most strategies that "beat the market" are found. The steps above are right. The evidence is not.

8.9 What nearest neighbours cannot do

It cannot explain. There are no coefficients, so there is nothing to interpret, no marginal effect, no test of a hypothesis about a mechanism. It is a prediction machine, and chapter 1's distinction applies with full force.

It cannot extrapolate. A query outside the range of the training data is answered by whichever training points happen to be least far away, which will be the boundary points, no matter how far outside you go. The prediction is confidently wrong and nothing signals it.

It is slow where it matters. Fitting is free and predicting is expensive: the entire training set must be searched for every query. That is the opposite of the profile you usually want in production.

It is exposed to irrelevant variables. Adding a predictor that carries no information does not merely fail to help — it actively corrupts the distances, diluting the informative dimensions. Chapter 5's penalties can shrink a useless variable to zero; a distance cannot.

8.10 A short review of the literature

> The method originates in an unpublished 1951 technical report by Fix and Hodges (1989), whose belated reprinting in 1989 is the citable version. Cover and Hart (1967) proved the result that made it respectable: as the sample grows, the error rate of the 1-nearest-neighbour rule is bounded above by twice the Bayes rate — so the simplest possible classifier, using one point, is asymptotically within a factor of two of the best any method can do. C. J. Stone (1977) established the conditions under which nearest-neighbour regression is consistent, completing the theoretical case.
>
> The trade-off itself is stated in its modern form by Geman, Bienenstock, and Doursat (1992), who framed it as a dilemma rather than a choice: any method flexible enough to fit an arbitrary function is flexible enough to fit the noise, and the resolution has to come from outside the data, as a restriction, a penalty, or a tuning parameter chosen by resampling.
>
> Beyer et al. (1999) is the reference for the failure in high dimensions, showing that under broad conditions the distance to the nearest neighbour approaches the distance to the farthest as dimension grows — so the query on which the whole method depends becomes meaningless, not merely imprecise.

8.11 What to report

1. **The baseline first.** Majority-class accuracy, or the mean for regression. Every subsequent number is read against it.
2. **A metric the base rate does not dominate.** AUC, balanced accuracy or a proper scoring rule. Accuracy on a 10% outcome tells you the outcome is 10%.
3. **How k was chosen**, with the cross-validation curve, and the training curve alongside it if you want to make the point that it is uninformative. With data

in time order, say that the folds followed time.

4. **That the test set was opened once**, and how many specifications were tried before the one reported. A score found after many looks is not a test score.
5. **That the predictors were standardised**, and how many there are. With more than a handful, say what you did about the dimensionality.
6. **The comparison against a rigid model.** Flexibility is a purchase, and the report should say what it bought. If a linear model does as well or better, that is the finding.
7. **What the method cannot support.** No interpretation, no extrapolation, and no claim about mechanism — only about prediction, within the range of the training data.

Chapter 9

Structural Equation Modelling

9.1 Measuring what cannot be observed

Some of the most important quantities in international business are not measured, because they cannot be. *Absorptive capacity*, *institutional quality*, *brand strength*, *digital maturity* — each is meant to be a real property of a firm or a country, each is invoked in explanations, and none of them appears in any dataset. What appears instead is a set of indicators, each a noisy and partial reflection of the thing.
Chapter 7 met this and handled it descriptively: factor analysis found a small number of common factors behind a battery of correlated items. This chapter takes the same idea and makes it a model you can *state and test* — a set of explicit claims about which indicators measure which construct, and about how the constructs relate to one another. And it builds that model the way you should build your own: on paper first, then in one line of text per construct, then in four lines of Python, and only then with numbers.

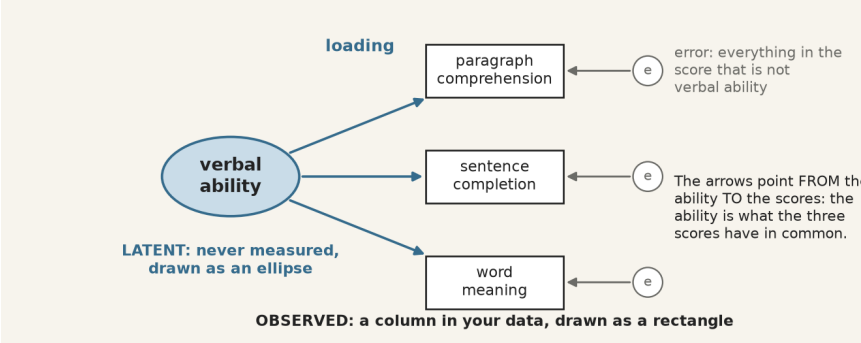


Figure 9.1. One construct, three indicators. The construct is never measured; it is what the three scores have in common, and the error circles hold everything else in each score.

The apparatus divides into two parts, and keeping them apart is the first thing to learn.

> **Measurement model** — the part that links unobserved constructs to observed indicators. It answers: *does this battery of questions measure one thing, and how well does each item measure it?* Its coefficients are loadings.

Structural model — the part that links constructs to each other. It answers: *does this construct predict that one?* Its coefficients are path coefficients, and they are regression coefficients between quantities that were never observed.

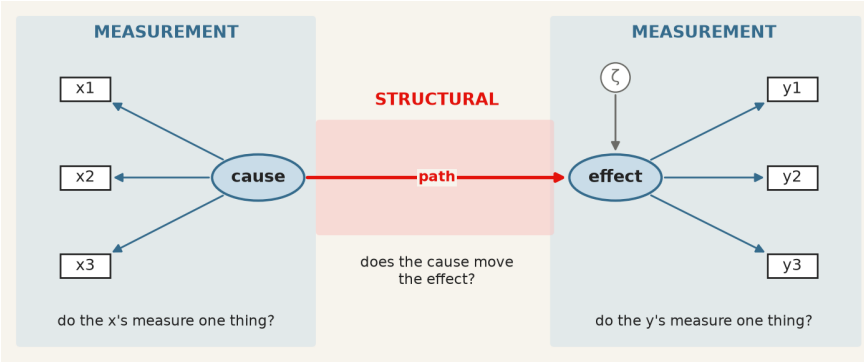


Figure 9.2. The two halves of a structural equation model. Blue, the measurement models: does each set of indicators measure one thing? Red, the structural model: does one construct move the other? ζ stands for what the path leaves unexplained.

A model can have both, and much of what makes SEM valuable is estimating them jointly: the structural relationships are corrected for the unreliability of the measurement, which a two-step approach of "build a scale, then regress it" does not do.

9.2 Wright's diagrams, and what a path meant

From the archive · Wright, 1918–1934

Sewall Wright was a geneticist working on the inheritance of coat colour in guinea pigs, and he needed to express something regression could not: a system in which variables cause each other in a specified pattern, some directly and some through intermediaries. He drew it — variables as boxes, causal claims as arrows — and worked out the algebra that lets the correlations be decomposed along the paths (Wright 1934).

Two features of that invention survive intact into every SEM package in use today. The diagram *is* the model: each arrow is a parameter to be estimated, and an arrow you do not draw is a claim that the effect is zero. And the implied correlation between any two variables is the sum of the contributions along every path connecting them, which is what makes the model testable — it predicts a whole covariance matrix, not a single coefficient.

Wright was explicit that the method could not discover the diagram. It required the

causal structure to be supplied from outside, on subject-matter grounds, and it then estimated the strengths and checked the implications. Ninety years of software has not changed that, and the section on equivalent models below is what happens when it is forgotten.

9.3 The language: five symbols and three operators

Wright's convention is still universal. Observed variables are rectangles, unobserved ones ellipses, a one-headed arrow is a directed effect, a two-headed arrow an unanalysed covariance, and a small circle the error left over.

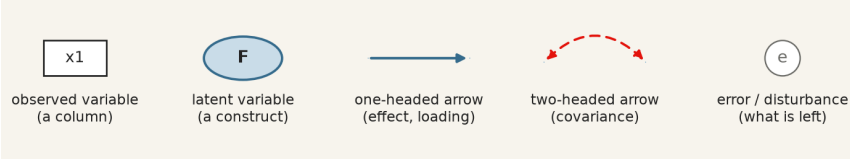


Figure 9.3. The five symbols of a path diagram. Every SEM figure in every paper is made of these.

Each drawing has an exact translation into text. The notation, standard since `lavaan` made it so (Rosseel 2012) and used by `semopy` in Python, is close to readable English:

You draw	You write	operator means
ellipse → rectangles	`visual =~ x1 + x2 + x3`	`=~` *is measured by*
ellipse → ellipse	`dem65 ~ dem60`	`~` *is regressed on*
two-headed arrow	`y1 ~~ y5`	`~~` *covaries with*
error circles	nothing	added by the software, one per indicator

So `visual =~ x1 + x2 + x3` asserts that a construct called *visual* exists, that these three observed scores are reflections of it, and — by what is absent — that nothing else in the model measures it. The corresponding equation for indicator j of construct ξ is

$$x_j = \lambda_j \xi + \delta_j,$$

with λ_j the loading and δ_j everything specific to that indicator: its own content, and its measurement error. This is chapter 7's factor model written one indicator at a time, and the interpretive move is the same — the construct is what the indicators have in common, and nothing else.

Identification. A latent variable has no units, so a scale must be imposed before anything can be estimated. Two conventions exist: fix one loading to 1, which gives the construct the units of that indicator, or fix the construct's variance to 1, which makes it standardised. Software picks the first by default, which is why one loading in every output has no standard error — it was not estimated. Reading it as "the strongest indicator" is a common and complete misunderstanding.

9.4 What is actually being fitted

The model does not fit the data. It fits the *covariance matrix* of the data.
Every parameter set θ implies a covariance matrix $\Sigma(\theta)$ among the observed variables, and estimation chooses θ to make that as close as possible to the observed S. The simplest case shows why a diagram is a prediction. Standardise everything; two indicators of the same construct share nothing but the construct, so the model says their correlation must be

$$\operatorname{corr}(x_i, x_j) \;=\; \lambda_i \lambda_j .$$

Every arrow you draw, and every one you leave out, implies a number of this kind for every pair of indicators. The discrepancy between those numbers and the observed ones is what all the fit statistics measure, and the χ^2 test is a formal test of

$$H_0 : \; \Sigma = \Sigma(\theta),$$

that is, of *exact* fit. Its degrees of freedom are the number of distinct elements in the covariance matrix minus the number of free parameters: nine indicators have $9 \times 10/2 = 45$ variances and covariances, the three-factor model below estimates 21 parameters, and the 24 left over are what make it testable. A model that says more about the covariances than it has parameters to absorb can be wrong, and therefore can be checked.

9.5 Building a measurement model, step by step

The data are the classic of the field: Holzinger and Swineford gave nine ability tests to 301 seventh- and eighth-graders in two Chicago schools in 1939. They ship with `semopy`, so every cell below runs as written, with nothing to download.

Column	Test	Meant to measure
`x1` · `x2` · `x3`	visual perception · cubes · lozenges	**visual** ability
`x4` · `x5` · `x6`	paragraph comprehension · sentence completion · word meaning	**textual** ability

Column	Test	Meant to measure
`x7` · `x8` · `x9`	speeded addition · speeded dot counting · straight vs curved capitals	**speed**

Step 1 and 2 · Load the data, and look before you model. The correlation matrix is what the model will try to reproduce, so it is the first thing to see.

```
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import semopy
from semopy.examples import holzinger39

data = holzinger39.get_data()
tests = ["x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9"]
corr = data[tests].corr()

plt.figure(figsize=(6, 5))
plt.imshow(corr, cmap="Blues", vmin=0, vmax=1)
plt.colorbar()
plt.xticks(range(9), tests)
plt.yticks(range(9), tests)
for i in range(9):
    for j in range(9):
        plt.text(j, i, round(corr.iloc[i, j], 2), ha="center", va="center",
plt.show()
```

Three dark squares sit on the diagonal. The textual tests correlate at about 0.7 with each other and 0.1–0.4 with everything else; the visual and speed tests form weaker blocks. If you cannot see your constructs in this picture, no fit index will find them for you.

Step 3 · Draw the model on paper. Each test measures one ability, and the three abilities may correlate. Every missing arrow is a claim: no arrow from *visual* to `x9` says the capitals test has nothing to do with visual ability.

Step 4 and 5 · Write it as text, and fit it. One line per ellipse. `semopy` adds the rest: one loading per construct fixed to 1, an error term for each indicator, and covariances among the constructs.

```
description = """
visual  =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed   =~ x7 + x8 + x9
```

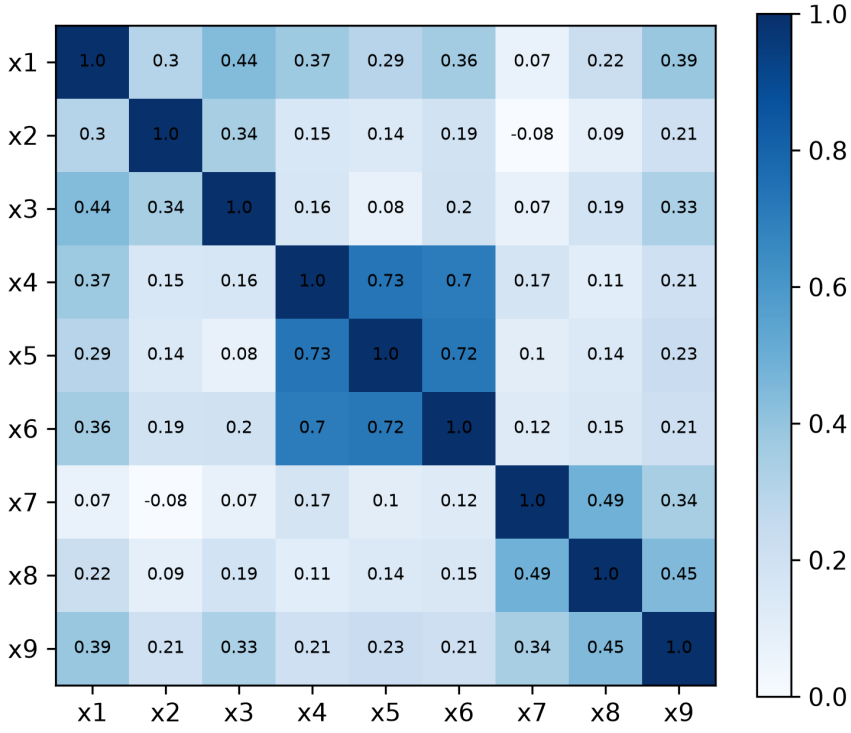


Figure 9.4. The correlations between the nine tests. Three blocks on the diagonal: the groups are visible before any model is fitted.

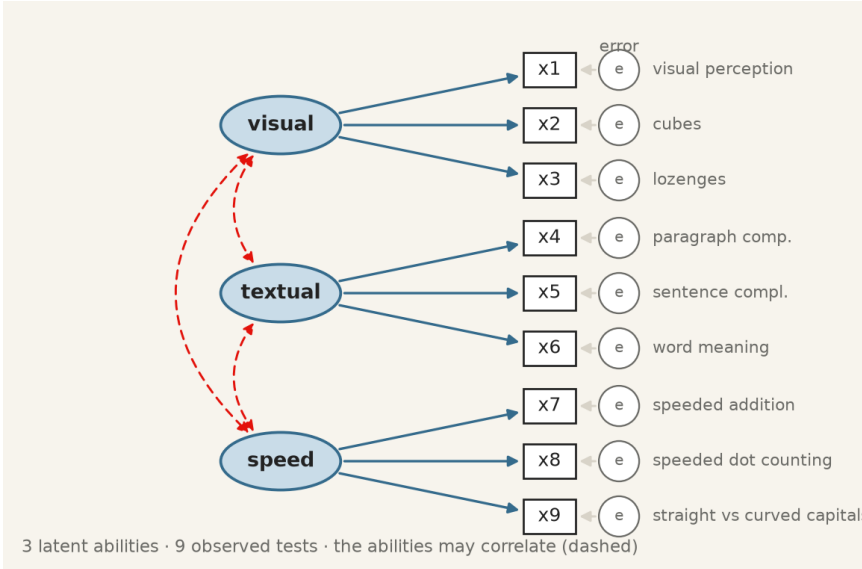


Figure 9.5. The three-factor model, before any number is estimated.

```
"""
model = semopy.Model(description)
model.fit(data)

estimates = model.inspect(std_est=True)
loadings = estimates[estimates["op"] == "~"]
print(loadings[["lval", "op", "rval", "Estimate", "Est. Std", "p-value"]].ro
```

	lval	op	rval	Estimate	Est. Std	p-value
0	x1	~	visual	1.000	0.771	-
1	x2	~	visual	0.554	0.424	0.0
2	x3	~	visual	0.731	0.582	0.0
3	x4	~	textual	1.000	0.852	-
4	x5	~	textual	1.113	0.855	0.0
5	x6	~	textual	0.926	0.838	0.0
6	x7	~	speed	1.000	0.569	-
7	x8	~	speed	1.180	0.723	0.0
8	x9	~	speed	1.083	0.665	0.0

`semopy` writes a loading as `x2 ~ visual` — "x2 depends on visual", which is what `=~` means read from the other side. `Estimate` is in the indicator's own units, and the rows equal to 1 with no p-value are the ones fixed in step 4. `Est. Std` is the column to read: a standardised loading is the correlation between the indicator and its construct, and its square is the share of the indicator the construct explains.

Step 6 · Put the numbers back on the drawing.

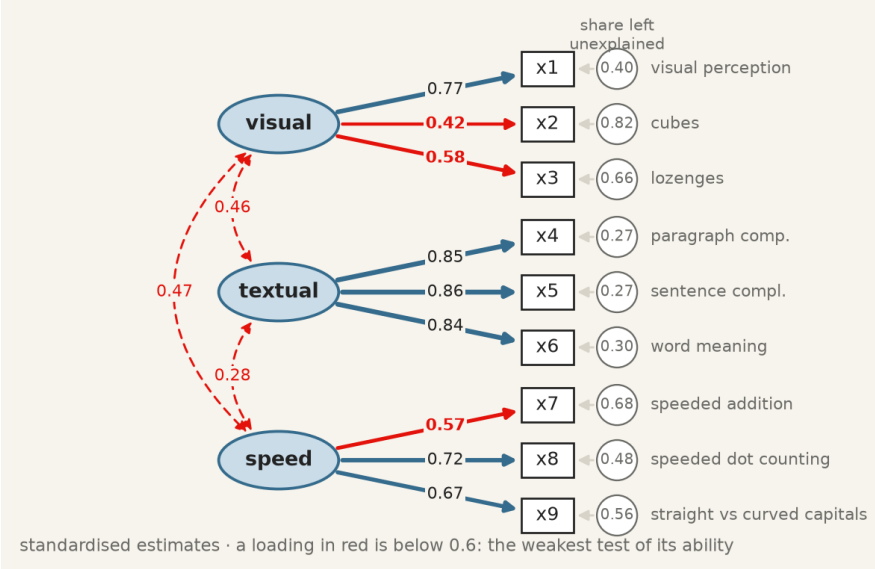


Figure 9.6. The fitted three-factor model, standardised. Red loadings are below 0.6; the circles hold the share of each test the model leaves unexplained, which is one minus the squared loading.

Textual is measured well, each test sharing about 72% of its variance with the construct. Cubes, `x2`, is the weak indicator: $0.42^2 = 0.18$, so 82% of a child's cubes score is something other than visual ability. Below about 0.4 an item has little to do with the thing it is supposed to measure, and keeping it degrades the construct rather than enriching it.

The rule from the previous section can be checked by hand. `x4` and `x5` both measure *textual*, so the model says their correlation is $0.85 \times 0.86 = 0.73$ — and the correlation in the first figure is 0.73. The three abilities correlate at 0.28 to 0.47: related, but distinct.

9.6 The fit indices, and what each would have to be

The χ^2 test is the only one with a null hypothesis, and it is almost never used alone, for a reason worth being honest about: with a large sample it rejects models that are approximately right, and with a small one it fails to reject models that are badly wrong. It is a test of exact fit, and no model is exactly right.

The alternatives are *approximate* fit indices, and three are conventionally reported together.

CFI — comparative fit index (Bentler 1990). How far the model has moved from a baseline in which all observed variables are uncorrelated, toward perfect fit. Scaled to $[0, 1]$; **above about 0.95** is the modern convention.

TLI — Tucker–Lewis index, the same comparison with a penalty for parameters, so adding a path that does not help can *lower* it. Not bounded above by 1 in practice. **Above about 0.95.**

RMSEA — root mean square error of approximation (Browne and Cudeck 1992). The misfit per degree of freedom, so it rewards parsimony directly. **Below about 0.06** is good, above 0.10 poor.

Those thresholds come from a simulation study (Hu and Bentler 1999) that has been cited tens of thousands of times and whose authors were careful to say they were not proposing universal cut-offs. They are conventions, and the honest use is to report the numbers and the convention together, and let the reader apply their own judgement.

```
fit = semopy.calc_stats(model)
print(fit[["chi2", "DoF", "chi2 p-value", "CFI", "TLI", "RMSEA"]].T.round(3)
```

```
                Value
chi2           85.306
DoF            24.000
chi2 p-value    0.000
CFI             0.931
TLI             0.896
RMSEA           0.092
```

Index	Convention	Three-factor model	Verdict
χ^2	$p > 0.05$	85.3 on 24 df, $p < 0.001$	rejects exact fit
CFI	> 0.95	0.931	close, not there
TLI	> 0.95	0.896	short
RMSEA	< 0.06	0.092	mediocre

A model drawn from a well-established theory of mental abilities, and it does not quite fit. That is a normal outcome, and the useful question is where it misses.

9.7 Where the model misses, and what you may do about it

The model implies a correlation for every pair of tests. Subtracting it from the observed one shows where the diagram is wrong.

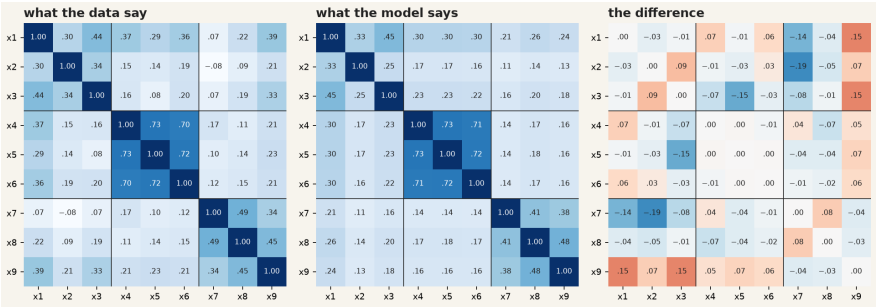


Figure 9.7. Observed correlations, the correlations the fitted model implies, and their difference. The textual block is reproduced almost exactly; the largest misses are in the last row, where x9 correlates with the visual tests more than the model allows.

x9 correlates with x1 and x3 by about 0.15 more than the model allows. Is there a reason? The test asks children to tell straight from curved capital letters, quickly: a speed task that also needs visual discrimination. An arrow from *visual* to x9 can therefore be argued from what the test is, not merely from the numbers. Comparing it with the alternatives is one loop:

```
models = {
    "one factor":     "ability =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9",
    "three factors":  "visual =~ x1 + x2 + x3 \n textual =~ x4 + x5 + x6 \n spe
    "three + x9 vis": "visual =~ x1 + x2 + x3 + x9 \n textual =~ x4 + x5 + x6 \
}
for name, desc in models.items():
    m = semopy.Model(desc)
    m.fit(data)
    s = semopy.calc_stats(m)
    print(f"{name:15} chi2 = {s['chi2'].iloc[0]:6.1f}  df = {s['DoF'].iloc[0]:
          f"  CFI = {s['CFI'].iloc[0]:.3f}  TLI = {s['TLI'].iloc[0]:.3f}"
          f"  RMSEA = {s['RMSEA'].iloc[0]:.3f}")
```

```
one factor     chi2 =  312.3  df = 27  CFI = 0.677  TLI = 0.569  RMSEA = 0.188
three factors  chi2 =   85.3  df = 24  CFI = 0.931  TLI = 0.896  RMSEA = 0.092
three + x9 vis chi2 =   52.4  df = 23  CFI = 0.967  TLI = 0.948  RMSEA = 0.065
```

One general ability is rejected decisively. Three abilities are far better, and x9 on *visual* as well brings CFI to 0.967 and RMSEA to 0.065.

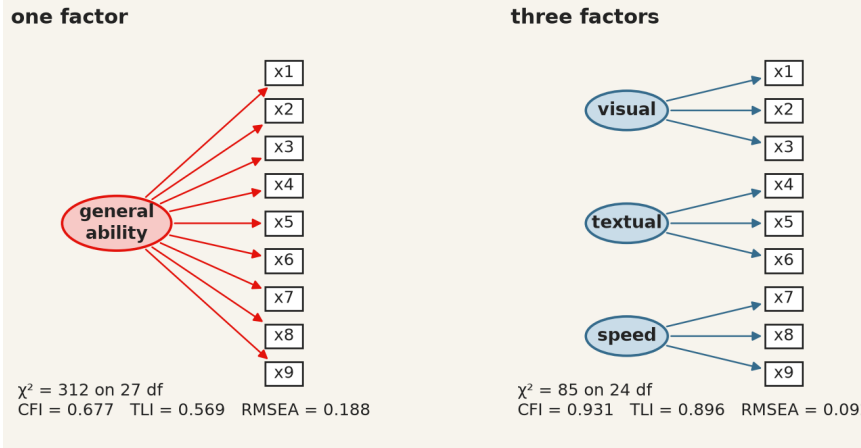


Figure 9.8. One general ability against three. The single factor is rejected on every index: the tests do not all measure one thing.

What to do about a failing model. Not, in the first instance, to modify it until it passes. Software offers modification indices — the improvement in χ^2 from freeing each fixed parameter — and following them is how a theoretical model becomes an atheoretical one that fits this sample. The third model above was chosen *after looking at the misfit*, which makes it exploratory however good its argument. The honest check is data it has not seen. The two schools give a partial one:

```
for school in ["Pasteur", "Grant-White"]:
    one_school = data[data["school"] == school]
    for name in ["three factors", "three + x9 vis"]:
        m = semopy.Model(models[name])
        m.fit(one_school)
        s = semopy.calc_stats(m)
        print(f"{school:12} n = {len(one_school)}  {name:15} CFI = {s['CFI']
              f"  RMSEA = {s['RMSEA'].iloc[0]:.3f}")
```

```
Pasteur      n = 156  three factors   CFI = 0.903  RMSEA = 0.104
Pasteur      n = 156  three + x9 vis  CFI = 0.927  RMSEA = 0.092
Grant-White  n = 145  three factors   CFI = 0.941  RMSEA = 0.089
Grant-White  n = 145  three + x9 vis  CFI = 0.989  RMSEA = 0.040
```

In Grant-White the modified model fits well (CFI 0.989, RMSEA 0.040). In Pasteur it helps but still falls short (CFI 0.927, RMSEA 0.092). The change is real in one school and only partial in the other — a finding about the tests, and the kind of result that a single pooled fit would have hidden.

9.8 Adding the structural part

A measurement model says what the constructs are. A structural model says what they do to each other. Bollen's political-democracy data (Bollen 1989), also bundled with `semopy`, are the standard example: 75 developing countries, with industrialisation in 1960 measured by three indicators and democracy measured by the same four expert ratings in 1960 and again in 1965.

Construct	Indicators
ind60 · industrialisation, 1960	`x1` GNP per capita · `x2` energy per capita · `x3` share of labour in industry
dem60 · democracy, 1960	`y1` press freedom · `y2` freedom of opposition · `y3` fair elections · `y4` effective legislature
dem65 · democracy, 1965	`y5`–`y8`: the same four questions, five years later

The theory supplies three structural claims: industrialisation raises democracy, democracy persists, and industrialisation may also act on 1965 directly. The design supplies one more. `y1` and `y5` are the same question, rated by the same experts, so whatever makes a rater generous on press freedom in 1960 plausibly does so again in 1965: their errors may covary. That is decided before fitting, from how the data were collected.

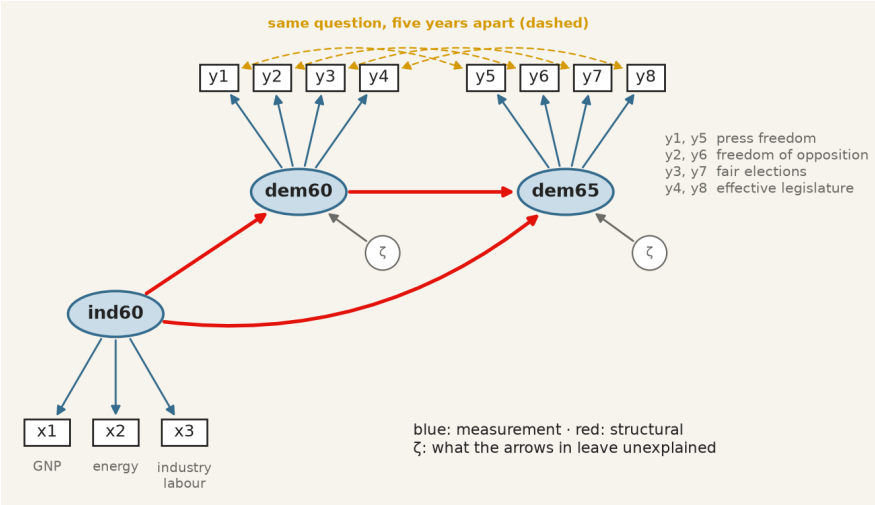


Figure 9.9. The democracy model before fitting. Blue arrows are measurement, red arrows structural, gold dashed arrows the same question asked twice.

All three operators now appear in one description:

```
from semopy.examples import political_democracy

countries = political_democracy.get_data()
description = """
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + y2 + y3 + y4
dem65 =~ y5 + y6 + y7 + y8
dem60 ~ ind60
dem65 ~ ind60 + dem60
y1 ~~ y5
y2 ~~ y6
y3 ~~ y7
y4 ~~ y8
"""
pd_model = semopy.Model(description)
pd_model.fit(countries)
est = pd_model.inspect(std_est=True)

paths = est[(est["op"] == "~") & est["lval"].isin(["dem60", "dem65"])]
print(paths[["lval", "op", "rval", "Estimate", "Est. Std", "p-value"]].round
```

```
    lval op   rval  Estimate  Est. Std   p-value
0  dem60  ~  ind60     1.435     0.448  0.000193
1  dem65  ~  ind60     0.507     0.168  0.015297
2  dem65  ~  dem60     0.816     0.869       0.0
```

A country one standard deviation more industrialised in 1960 was 0.45 standard deviations more democratic that year. Democracy is very persistent (0.87), and industrialisation adds a smaller direct push to 1965 on top of that persistence (0.17, $p = 0.015$). Industrialisation explains 20% of the variance of democracy in 1960; with 1960 democracy it explains 91% of 1965. All eleven loadings exceed 0.69.

Direct and indirect effects. Industrialisation reaches 1965 by two roads: straight there, and through 1960 democracy. Along a road, standardised paths multiply.

```
a = est.loc[(est["lval"] == "dem60") & (est["rval"] == "ind60"), "Est. Std"]
b = est.loc[(est["lval"] == "dem65") & (est["rval"] == "dem60"), "Est. Std"]
c = est.loc[(est["lval"] == "dem65") & (est["rval"] == "ind60"), "Est. Std"]
print(f"direct    c         = {c:.2f}")
print(f"indirect  a * b     = {a:.2f} * {b:.2f} = {a * b:.2f}")
print(f"total     c + a * b = {c + a * b:.2f}")
```

```
direct    c         = 0.17
indirect  a * b     = 0.45 * 0.87 = 0.39
total     c + a * b = 0.56
```

Two-thirds of the total effect (0.39 of 0.56) runs through 1960 democracy: a mediation, in which dem60 carries most of what industrialisation does for dem65. The

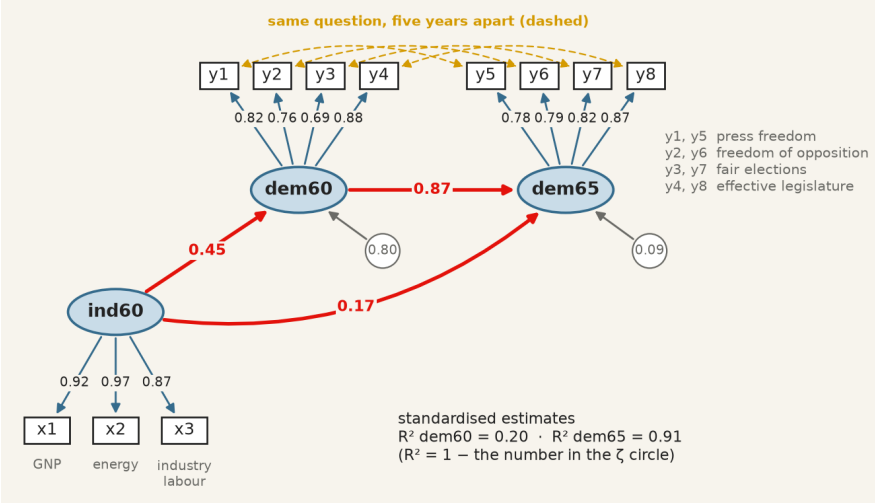


Figure 9.10. The fitted democracy model, standardised. The numbers in the ζ circles are the unexplained shares, so R^2 is one minus each.

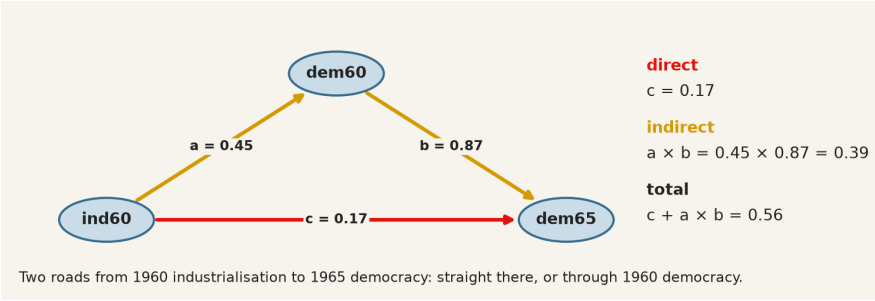


Figure 9.11. Two routes from ind60 to dem65: the direct path c, and the indirect route a × b through dem60.

arithmetic is exact within the model; whether it is a causal decomposition depends entirely on whether the arrows are, which is the subject of the next two sections.

Every extra arrow needs a reason. The four same-question covariances changed the fit, and it is worth seeing by how much:

```
base = """
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + y2 + y3 + y4
dem65 =~ y5 + y6 + y7 + y8
dem60 ~ ind60
dem65 ~ ind60 + dem60
"""
same_question = "y1 ~~ y5 \n y2 ~~ y6 \n y3 ~~ y7 \n y4 ~~ y8"
for name, desc in {"without": base, "with": base + same_question}.items():
    m = semopy.Model(desc)
    m.fit(countries)
    s = semopy.calc_stats(m)
    print(f"{name:8} chi2 = {s['chi2'].iloc[0]:5.1f}  df = {s['DoF'].iloc[0]
          f"  p = {s['chi2 p-value'].iloc[0]:.3f}  CFI = {s['CFI'].iloc[0]:.
          f"  RMSEA = {s['RMSEA'].iloc[0]:.3f}")
```

```
without  chi2 =  72.5  df = 41  p = 0.002  CFI = 0.953  RMSEA = 0.102
with     chi2 =  50.8  df = 37  p = 0.064  CFI = 0.980  RMSEA = 0.071
```

With them, χ^2 no longer rejects ($p = 0.064$), CFI reaches 0.980 and RMSEA 0.071, and the structural paths barely move. They are legitimate here because the design predicts them. The same four lines added *because they raise the CFI* would be fitting noise — and correlated residuals added until a model passes are the most common way an SEM paper overstates its evidence.

9.9 What the estimator assumes

Nothing above said how θ is chosen. The default, in `semopy` as in `lavaan`, is **maximum likelihood**, and the likelihood it maximises is the multivariate normal one. That assumption is not decoration: the χ^2 statistic and every standard error in the output are derived from it.

When the data are not multivariate normal — and survey items with floor or ceiling effects rarely are — the failure runs in one direction, which is what makes it dangerous. The χ^2 statistic is inflated, so a model that fits is rejected. The standard errors are too small, so paths look more significant than they are. Non-normality does not make you cautious. It makes you confident.

The first check is one test per indicator, with functions you already have:

```
from scipy import stats
```

```
for t in tests:
    W, p = stats.shapiro(data[t])
    print(f"{t}  W = {W:.3f}  p = {p:.4f}  skew = {data[t].skew():5.2f}"
          f"  excess kurtosis = {data[t].kurt():5.2f}")
```

```
x1  W = 0.993  p = 0.1582  skew = -0.26  excess kurtosis =  0.36
x2  W = 0.970  p = 0.0000  skew =  0.47  excess kurtosis =  0.38
x3  W = 0.952  p = 0.0000  skew =  0.39  excess kurtosis = -0.89
x4  W = 0.983  p = 0.0011  skew =  0.27  excess kurtosis =  0.12
x5  W = 0.977  p = 0.0001  skew = -0.35  excess kurtosis = -0.53
x6  W = 0.954  p = 0.0000  skew =  0.87  excess kurtosis =  0.88
x7  W = 0.991  p = 0.0560  skew =  0.25  excess kurtosis = -0.27
x8  W = 0.981  p = 0.0004  skew =  0.53  excess kurtosis =  1.24
x9  W = 0.994  p = 0.3070  skew =  0.21  excess kurtosis =  0.34
```

Six of the nine tests reject univariate normality at the 1% level, though none is far from it: the skewness stays below 0.9 and the excess kurtosis below 1.3, small enough that maximum likelihood is a defensible default here. Passing one indicator at a time is not the assumption actually being made, which is joint. Mardia's measures of multivariate skewness and kurtosis,

$$b_1 = \frac{1}{n^2}\sum_i\sum_j d_{ij}^3, \qquad b_2 = \frac{1}{n}\sum_i d_{ii}^2, \qquad d_{ij} = (x_i - \bar{x})^\top S^{-1}(x_j - \bar{x}),$$

test it directly; under normality b_2 has mean $k(k+2)$, and every indicator can pass Shapiro–Wilk while the joint distribution fails both.

The answer to non-normality is not to abandon the model but to change the estimator. **Robust maximum likelihood** — `MLR`, resting on the Satorra–Bentler scaled χ^2 — keeps the ML point estimates and rescales the test statistic and the standard errors by the non-normality actually observed. For ordinal indicators, **WLSMV** treats a five-point item as a coarsened continuous variable rather than pretending its categories are numbers. Either way, report the estimator alongside the fit.

9.10 Good fit is not evidence that the model is correct

This is the chapter's most important section and the objective most often recited without being understood.

A model that fits well has demonstrated one thing: the covariances it implies are close to the covariances observed. It has *not* demonstrated that its causal structure is right, because **other models — with different, even opposite, causal claims — can imply exactly the same covariance matrix.** They are called equivalent models, they exist for almost every model anyone fits, and no amount of data distinguishes them.

```
measurement = "ind60 =~ x1 + x2 + x3 \n dem60 =~ y1 + y2 + y3 + y4 \n"
for arrow in ["dem60 ~ ind60", "ind60 ~ dem60"]:
```

```
    m = semopy.Model(measurement + arrow)
    m.fit(countries)
    s = semopy.calc_stats(m)
    print(f"{arrow}:  chi2 = {s['chi2'].iloc[0]:.4f}  df = {s['DoF'].iloc[0]
          f"  CFI = {s['CFI'].iloc[0]:.4f}  RMSEA = {s['RMSEA'].iloc[0]:.4f}
```

```
dem60 ~ ind60:  chi2 = 23.7380  df = 13  CFI = 0.9722  RMSEA = 0.1057
ind60 ~ dem60:  chi2 = 23.7384  df = 13  CFI = 0.9722  RMSEA = 0.1057
```

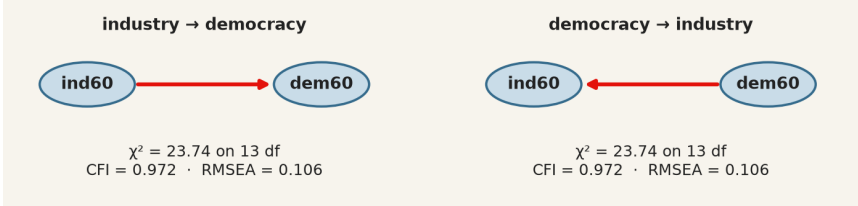


Figure 9.12. Two opposite causal stories with one fit.

Identical χ^2, identical degrees of freedom, identical CFI and RMSEA — for two models asserting opposite causal directions. No fit index can prefer one, because fit indices measure distance from the observed covariances and both reproduce them equally.

This is the same structural fact as chapter 7's rotation indeterminacy, in a more dangerous costume: there, the alternatives were obviously arbitrary and everyone knew a rotation had been chosen; here, the alternative is a different theory of how the world works, and the output looks like a result.

The consequence for practice is exact. **The direction of the arrows is an assumption you supply, and the fit statistics cannot audit it.** What can audit it is subject knowledge, temporal order — a cause cannot follow its effect, which is why the 1960-to-1965 arrow is more defensible than the 1960-to-1960 one — and design, which is what chapters 10 and 11 are about.

9.11 What SEM does not fix

It does not make cross-sectional data longitudinal. Estimating a path from A to B measured at the same moment identifies a direction only because you declared one.

It does not remove confounding. An omitted common cause of two constructs biases the path between them exactly as it biases a regression coefficient. The latent variables absorb measurement error, not confounders.

It is demanding of sample size. With a small n the χ^2 test has little power to reject a wrong model, and the fit indices are themselves unstable — so "good fit" in a small sample is the weakest possible evidence. Bollen's 75 countries are a

small sample, and the good fit of the democracy model should be read with that in mind.

Comparisons across groups require an extra assumption. Comparing a construct between, say, countries presumes it is measured the same way in each. That is *measurement invariance*, it is testable by fitting the model with loadings constrained equal across groups, and comparing means without testing it can manufacture a difference out of a translation (Cheung and Rensvold 2002). The two schools above, where one modification helped in one and not the other, are a small instance of the same problem.

9.12 The recipe

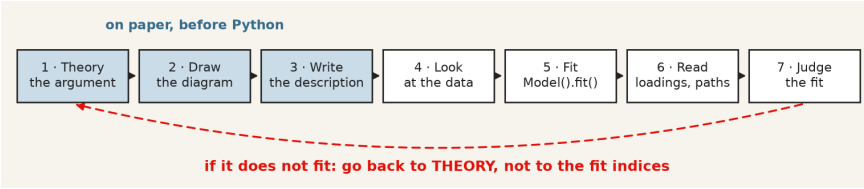


Figure 9.13. Seven steps from theory to judging the fit. When the model does not fit, the arrow goes back to the theory, not to the fit indices.

Step	What you do	In Python
1–3	argue which indicators measure which construct, draw it, write one line per ellipse	a string with =~, ~, ~~
4	look at the correlations: can you see your groups?	`data[cols].corr()`, `plt.imshow`
5	fit	`semopy.Model(description)`, `.fit(data)`
6	read loadings and paths, standardised	`.inspect(std_est=True)`
7	judge the fit, all four indices	`semopy.calc_stats(model)`

9.13 A short review of the literature

The method has two ancestors that met in the 1970s. Wright (1934) developed path analysis in genetics, with the diagram as the model and the decomposition of correlations along paths; Spearman (1904) and the factor-analytic tradition supplied latent variables measured by indicators. Jöreskog (1971) combined them into the general covariance-structure model that all current software implements, Bollen (1989) wrote the textbook that taught a generation to use it — the political-democracy model above is its running example — and Rosseel (2012) gave the modern open-source implementation whose syntax `semopy` follows in Python.

Assessment of fit is where the applied literature has moved most. Bentler (1990) introduced the comparative fit index and Browne and Cudeck (1992) the RMSEA, both as responses to the chi-square test's sensitivity to sample size. Hu and Bentler (1999) proposed the cut-offs now universally quoted — CFI and TLI above 0.95, RMSEA below 0.06 — while cautioning against exactly the mechanical application that followed.

MacCallum and Austin (2000) reviewed a decade of applied practice and found the recurring faults to be the ones this chapter warns about: models modified until they fit and then reported as though specified in advance, equivalent models never considered, and causal language attached to cross-sectional designs. Cheung and Rensvold (2002) supplies the invariance testing that any cross-group comparison requires and most omit.

9.14 What to report

1. **The model, as a diagram or as its description.** Every arrow is a claim; a reader cannot evaluate claims they cannot see.
2. **How the latent scale was set** — a fixed loading or a fixed variance — so nobody misreads the parameter without a standard error.
3. **Standardised loadings for every indicator**, not only the significant ones, and what you did about weak items. Dropping an indicator after seeing its loading is a data-dependent decision; say so.
4. **The chi-square with its degrees of freedom and p-value**, alongside CFI, TLI and RMSEA, and the convention you judged them against. Reporting only the indices that pass is the single most common failure.
5. **Whether the model was modified after fitting**, and on what grounds. A model reached through modification indices or residuals is exploratory whatever its fit, until it holds on data it has not seen.
6. **At least one equivalent model, named.** If you cannot think of one, that is a reason to look harder rather than evidence there is none.
7. **What the arrows do not license.** Direction was assumed, not discovered. With cross-sectional data the model estimates the strength of a relationship under a causal structure you supplied, and a good fit is consistent with that structure being wrong.

Chapter 10

Causal Inference I: Counterfactuals, Randomisation, Matching

10.1 The question the first nine chapters could not answer

Every method so far has described an association and then, carefully, declined to call it an effect. Chapter 2 insisted on "associated with" rather than "causes". Chapter 6 showed the same association taking four different values depending on which variation identified it. Chapter 9 produced two models with identical fit and opposite causal arrows.

The reason for all that caution is that the question — *did the policy do anything?* — is not a question about the data you have. It is a question about data you do not have and cannot get.



David Hume · 1711–1776
Asked what entitles us to say one thing causes another, and did not find a satisfying answer. Neither has anyone since.
Allan Ramsay, 1766. Public domain, via Wikimedia Commons.

From the archive · Hume, 1739

David Hume asked what we actually observe when we say one thing causes another, and answered: contiguity, succession, and constant conjunction. The billiard ball moves; another moves after it; this has always happened. What we never observe

is the *necessary connection* — the thing that would make the second event have to follow.

His second formulation is the one that founded modern causal inference, though it took two centuries to be taken up. A cause, he wrote, is an object followed by another, *and where, if the first object had not been, the second never had existed*. That subordinate clause is a counterfactual: a statement about a world that did not happen. It cannot be observed, in principle, for the same reason a road not taken has no destination.

Hume treated this as a sceptical problem about knowledge. Modern statistics treats it as a *missing data* problem, which turns out to be the more productive framing — because missing data can sometimes be recovered, under assumptions you can state, defend and occasionally test.

10.2 Potential outcomes

The framework that formalises Hume's clause is due to Neyman and, in its modern form, to Rubin (Donald B. Rubin 1974). For each unit i define two quantities:

$$Y_i(1) = \text{the outcome if unit } i \text{ is treated}, \qquad Y_i(0) = \text{the outcome if it is not.}$$

Both are properties of the unit, defined whether or not the treatment happens. The causal effect *for that unit* is the difference

$$\tau_i \;=\; Y_i(1) - Y_i(0),$$

and it is exactly what Hume said we cannot observe: whichever treatment the unit receives, the other potential outcome is missing, permanently and by construction. This is the **fundamental problem of causal inference** (Holland 1986), and it means that no individual causal effect is ever identified. What we can hope for is an *average*.

> **ATE** — average treatment effect, $\mathrm{E}[Y(1) - Y(0)]$ over the whole population. What would happen if everyone were treated versus no one.

> **ATT** — average treatment effect on the treated, $\mathrm{E}[Y(1) - Y(0) \mid T = 1]$. What the treatment did for those who actually received it. Usually the policy-relevant quantity, and usually not equal to the ATE.

Now the arithmetic that explains why naive comparisons fail. What you can compute from data is the difference in observed means. Add and subtract the same term:

firm	sales WITH the programme	sales WITHOUT the programme	effect for this firm	what you observe
1	?	5.0	?	did not
2	?	9.4	?	did not
3	9.2	?	?	took it
4	13.0	?	?	took it
5	?	11.3	?	did not
6	10.5	?	?	took it

Figure 10.1. The table you never get to see. Each firm has two potential outcomes; whichever choice it made, the other is missing, and so is its effect. The numbers come from the simulation later in this chapter, where, unusually, both columns exist.

$$\underbrace{\mathrm{E}[Y \mid T{=}1] - \mathrm{E}[Y \mid T{=}0]}_{\text{what you can compute}} = \underbrace{\mathrm{E}[Y(1) - Y(0) \mid T{=}1]}_{\text{ATT}} + \underbrace{\mathrm{E}[Y(0) \mid T{=}1] - \mathrm{E}[Y(0) \mid T{=}0]}_{\text{selection bias}}.$$

The second term is the difference between the treated and untreated groups *in the absence of treatment*. If countries that adopt a policy would have differed from those that did not even without it, the observed difference is the effect plus that gap, and nothing in the data separates them.

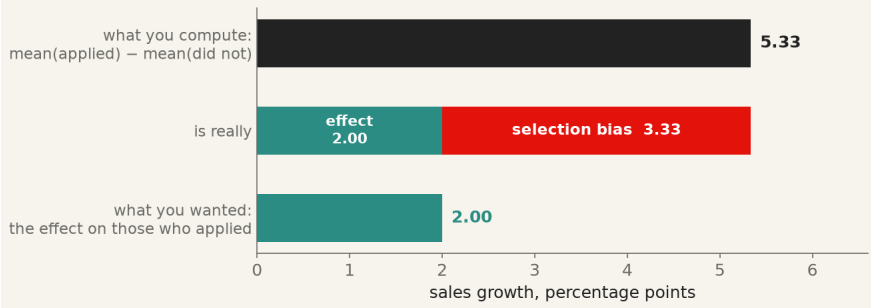


Figure 10.2. The decomposition in numbers, from the simulated firms below. The naive comparison says 5.33; the programme's effect is 2.00; the other 3.33 is how much faster the joining firms would have grown anyway.

10.3 Randomisation, and what it buys

Randomised assignment sets the selection term to zero, and it is worth being precise about why. If treatment is assigned by a coin flip, then T is independent of

everything about the unit — including $Y(0)$ and $Y(1)$ — so $\mathrm{E}[Y(0) \mid T{=}1] = \mathrm{E}[Y(0) \mid T{=}0]$ and the bias term vanishes.

Note what randomisation does *not* do. It does not make the groups identical; in any finite sample they will differ, sometimes substantially. It makes them differ *in a way unrelated to the outcome*, so the difference is noise rather than bias — and noise is what standard errors are for. It also balances variables nobody measured or thought of, which is the property no observational method can replicate.

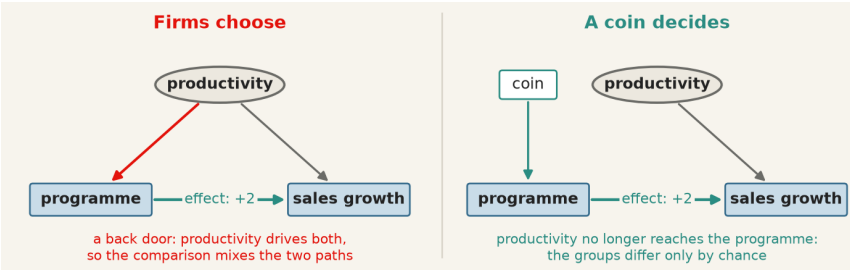


Figure 10.3. Left: productivity drives both the decision to join and growth, so the comparison runs through a back door. Right: a coin decides, productivity can no longer reach the programme, and the back door is shut.

When randomisation is unavailable, and in international business it usually is — countries are not randomly assigned national strategies — the alternative is to assume something strong enough to recover the same result.

> **Conditional independence** (unconfoundedness, ignorability) — given a set of covariates X, treatment is as good as random: $\{Y(0), Y(1)\} \perp T \mid X$. Every determinant of both treatment and outcome is in X. **This assumption is not testable**, because it concerns the missing potential outcome.

> **Overlap** (common support) — for every value of X, the probability of treatment is strictly between 0 and 1. If some countries would never adopt under any circumstances, there is no comparison to be made for them, and an estimate that includes them is extrapolating rather than comparing. Unlike conditional independence, overlap **is** checkable.

10.4 A world where the answer is known

Everything above can be watched happening in a simulation, which is the one setting where both potential outcomes exist. A thousand firms; a programme that adds exactly two points to every firm's export growth; and a productivity that nobody in the ministry observes, which raises growth with or without the programme.

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from sklearn.neighbors import NearestNeighbors

rng = np.random.default_rng(10)
n = 1000
productivity = rng.normal(0, 1, n)
sales_without = 10 + 3 * productivity + rng.normal(0, 2, n)
sales_with = sales_without + 2                          # the true effect, for e
applies = (productivity + rng.normal(0, 1, n) > 0.5).astype(int)

observed = np.where(applies == 1, sales_with, sales_without)
naive = observed[applies == 1].mean() - observed[applies == 0].mean()
bias = sales_without[applies == 1].mean() - sales_without[applies == 0].mean()
print(f"self-selection   naive {naive:.2f} = effect 2.00 + selection bias {bias

draws = []
for _ in range(1000):                                   # a thousand randomised
    coin = rng.integers(0, 2, n)
    seen = np.where(coin == 1, sales_with, sales_without)
    draws.append(seen[coin == 1].mean() - seen[coin == 0].mean())
print(f"coin flip        mean of 1,000 estimates {np.mean(draws):.2f}, spread
```

```
self-selection   naive 5.33 = effect 2.00 + selection bias 3.33
coin flip        mean of 1,000 estimates 2.00, spread 0.22
```

The two lines are the chapter's argument in miniature. Self-selection overstates the effect by a factor of more than two and a half, and the excess is exactly the selection term: the joining firms would have grown 3.33 points faster without the programme. The coin is right on average and wrong in any single draw by about a quarter of a point, which is noise, which a standard error describes. The selection term cannot be computed in real data, since it needs the column nobody sees; what *can* be computed is how different the groups were before treatment, here about one standard deviation of productivity. That comparison — balance — is the diagnostic the rest of the chapter is built on.

10.5 LaLonde's test

The simulation knows the answer because it wrote it. A real data set that knows the answer is rarer, and the most famous one is a job-training experiment.
The National Supported Work demonstration, run in the United States in 1976–77,

assigned disadvantaged workers to a training placement **by lottery**. The lottery's estimate of the effect on 1978 earnings is therefore credible. LaLonde (1986) then did something that has shaped the field since: he kept the trainees, threw away the lottery's controls, replaced them with people from national surveys, and asked whether the estimators economists applied to survey data would recover the experiment's answer. They did not, and the paper became the standard argument for experiments. Dehejia and Wahba (1999) returned to the same data with propensity-score matching and recovered it, and A. Smith and E. Todd (2005) showed how fragile that recovery is. The data are on Rajeev Dehejia's NBER page, and `pd.read_stata` reads them directly.

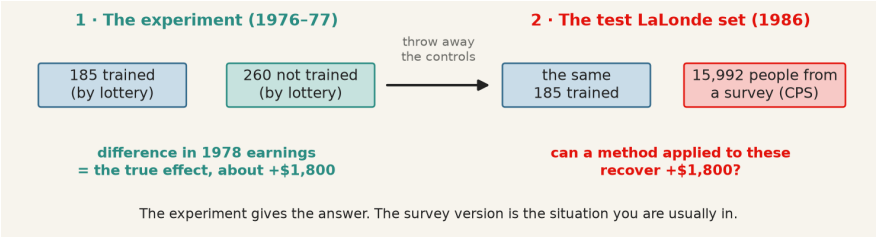


Figure 10.4. The design. The experiment gives the true effect; the survey version is the situation applied work is usually in.

```
url = "https://users.nber.org/~rdehejia/data/"
nsw = pd.read_stata(url + "nsw_dw.dta")
cps = pd.read_stata(url + "cps_controls.dta")
X = ["age", "education", "black", "hispanic", "married", "nodegree", "re74",

experiment = smf.ols("re78 ~ treat", data=nsw).fit()
print(f"experiment:  {experiment.params['treat']:+,.0f}  (SE {experiment.bse

data = pd.concat([nsw[nsw["treat"] == 1], cps], ignore_index=True)
treated, control = data[data["treat"] == 1], data[data["treat"] == 0]
print(f"naive:       {treated['re78'].mean() - control['re78'].mean():+,.0f}

regression = smf.ols("re78 ~ treat + " + " + ".join(X), data=data).fit()
print(f"regression:  {regression.params['treat']:+,.0f}   (same eight contro

sd = np.sqrt((treated[X].var() + control[X].var()) / 2)
print("\nstandardised differences before matching:")
print(((treated[X].mean() - control[X].mean()) / sd).astype(float).round(2).

experiment:  +1,794  (SE 633)
naive:       −8,498
regression:  +699   (same eight controls, linear)
```

```
standardised differences before matching:
age          -0.80
education    -0.68
black         2.43
hispanic     -0.05
married      -1.23
nodegree      0.90
re74         -1.57
re75         -1.75
```

Training raised 1978 earnings by about $1,800 in the experiment. Against the survey, the naive comparison says it *cut* them by $8,500: the wrong sign and five times too large, because the survey's average worker earned $14,000 a year before the programme and the trainees under $2,100. The standardised differences put that in common units — the gap in means divided by the pooled standard deviation — and most are between 0.7 and 2.4, against a convention that calls anything under 0.1 balanced.

Regression with all eight characteristics does much better, and is still off by more than half. The reason is worth understanding, because it is what matching fixes. A linear regression uses *every* survey respondent, including the thousands earning $30,000 who resemble no trainee, and lets a straight line carry the comparison from them to the trainees. The estimate depends on that line being right far outside the region where the trainees live.

10.6 The propensity score, and a match that fails its check

Rosenbaum and Rubin (1983) proved that if treatment is as good as random given X, it is as good as random given the single number $e(X) = P(T = 1 \mid X)$, the **propensity score**. Eight dimensions collapse to one, and matching becomes the nearest-neighbour problem of chapter 8 on one column.

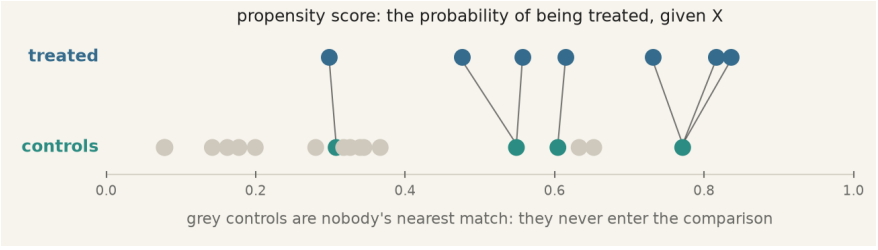


Figure 10.5. Matching on the score. Each treated unit is joined to the control with the closest score; controls unlike every treated unit are never used.

```
first = "treat ~ " + " + ".join(X)
data["pscore"] = smf.logit(first, data=data).fit(disp=0).predict(data)
treated, control = data[data["treat"] == 1], data[data["treat"] == 0]

bins = np.linspace(0, 0.5, 26)
plt.figure(figsize=(7, 3.4))
plt.hist(control["pscore"], bins=bins, alpha=0.6, label="survey (control)")
plt.hist(treated["pscore"], bins=bins, alpha=0.6, label="trained")
plt.yscale("log")
plt.xlabel("propensity score")
plt.legend()
plt.show()

print(f"score range, trained: {treated['pscore'].min():.3f} – {treated['psco
print(f"survey respondents with a score above 0.05: {(control['pscore'] > 0.
```

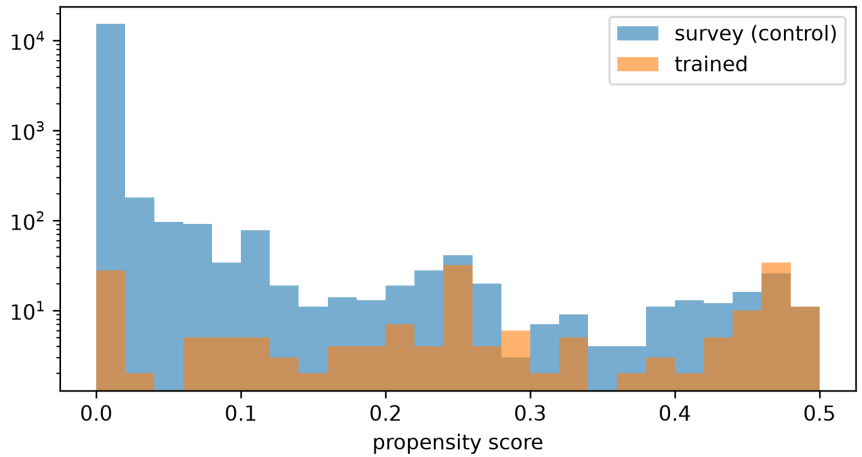


Figure 10.6. The propensity score in the two groups, on a logarithmic scale. Almost every survey respondent has a score near zero, but every bin that contains trainees also contains some survey respondents: the overlap needed for matching exists.

```
score range, trained: 0.001 – 0.487
survey respondents with a score above 0.05: 534
```

Overlap is the checkable half of the identifying assumptions, and here it holds, barely: of almost sixteen thousand survey respondents, a few hundred look anything like a trainee. That is enough, because matching needs only one good control per trainee.

```
nn = NearestNeighbors(n_neighbors=1).fit(control[["pscore"]])
distance, index = nn.kneighbors(treated[["pscore"]])
matched = control.iloc[index.ravel()]

print(f"{len(treated)} trainees, {matched.index.nunique()} distinct controls, "
      f"largest score gap {distance.max():.4f}")
print(((treated[X].mean() - matched[X].mean()) / sd).astype(float).round(2).to_
```

```
185 trainees, 124 distinct controls, largest score gap 0.0038
age          0.31
education    0.00
black        0.05
hispanic     0.07
married      0.09
nodegree     0.00
re74         0.01
re75         0.03
```

Matching with replacement lets a good control serve more than once, which is why 185 trainees use fewer distinct controls. Balance has improved enormously on earnings, race and schooling, but age is still out by 0.31: the logit has treated the probability of training as moving in one direction with age, and it does not. **This is the moment the procedure is designed for.** The response is to change the score and check again, and to do it *without looking at the outcome*, because a score chosen for the estimate it produces is a specification search with a respectable name.

10.7 A score that balances, and only then the outcome

```
better = first + " + I(age**2) + I(re74 == 0) + I(re75 == 0)"
data["pscore"] = smf.logit(better, data=data).fit(disp=0).predict(data)
treated, control = data[data["treat"] == 1], data[data["treat"] == 0]
nn = NearestNeighbors(n_neighbors=1).fit(control[["pscore"]])
distance, index = nn.kneighbors(treated[["pscore"]])
matched = control.iloc[index.ravel()]

before = (treated[X].mean() - control[X].mean()) / sd
after = (treated[X].mean() - matched[X].mean()) / sd
plt.figure(figsize=(7, 3.6))
plt.scatter(before, X, label="before")
plt.scatter(after, X, label="after")
plt.axvline(-0.1, color="grey", linestyle="--")
plt.axvline(0.1, color="grey", linestyle="--")
plt.xlabel("standardised difference")
plt.legend()
plt.show()
```

```
att = treated["re78"].mean() - matched["re78"].mean()
print(f"largest |difference| after matching: {after.abs().max():.2f}")
print(f"ATT by matching: {att:+,.0f}    experiment: {experiment.params['trea
```

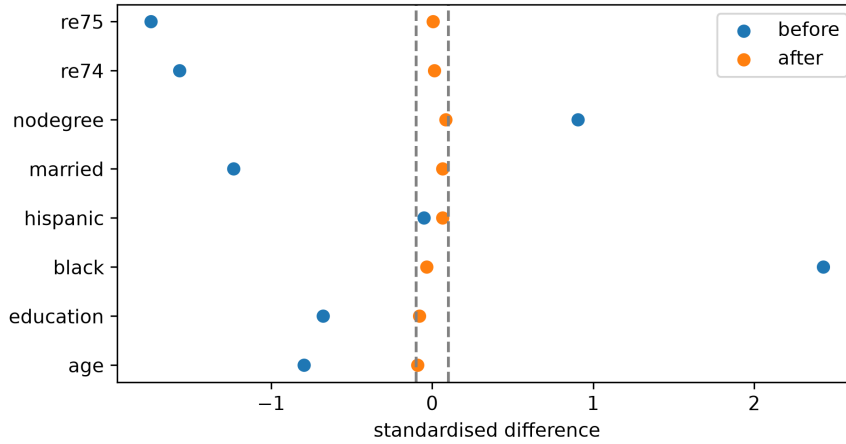


Figure 10.7. Standardised differences before matching (blue) and after matching on the improved score (orange). The dashed lines mark ±0.1. Every covariate ends inside them.

```
largest |difference| after matching: 0.09
ATT by matching: +2,001    experiment: +1,794
```

Age squared lets the probability of training rise and fall with age; the two indicators mark people who earned nothing in 1974 or 1975, which is most trainees and few survey respondents. With them, every standardised difference sits inside 0.1. The figure — a **love plot** — is the evidence a reader needs, and it is the evidence to report, rather than the logit's coefficients or fit, which are a means and not an end.

Only now is the outcome looked at. The matched ATT is about $2,000, within a third of a standard error of the experiment.

Two cautions keep this from being a triumph. The first is uncertainty. The standard error printed by a regression on the matched sample ignores that the score was estimated and that controls were reused; a bootstrap that redoes the whole procedure is the honest version.

```
estimates = []
for b in range(200):
    t = treated.sample(len(treated), replace=True, random_state=b)
    c = control.sample(len(control), replace=True, random_state=1000 + b)
```

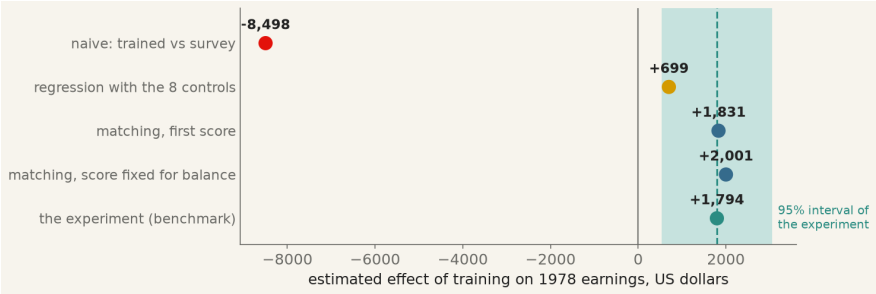


Figure 10.8. Every estimate on one axis. Both matching estimates fall inside the experiment's 95% interval; the naive comparison has the wrong sign, and the linear regression is pulled down by respondents who resemble no trainee.

```
    nn_b = NearestNeighbors(n_neighbors=1).fit(c[["pscore"]])
    _, idx = nn_b.kneighbors(t[["pscore"]])
    estimates.append(t["re78"].mean() - c.iloc[idx.ravel()]["re78"].mean())
print(f"bootstrap standard error of the ATT: {np.std(estimates):,.0f}")
print(f"95% interval: {np.percentile(estimates, 2.5):+,.0f} to {np.percentile(e
```

```
bootstrap standard error of the ATT: 900
95% interval: −42 to +3,309
```

(Strictly the score should be re-estimated inside each replication too; holding it fixed, as here, keeps the cell short and understates the spread a little.) The interval is wide — the matched estimate rests on 185 trainees and a much smaller number of distinct controls — and it comfortably contains the experimental answer. It also, worth saying plainly, reaches zero: on the survey data alone, matching cannot rule out that training did nothing, which the experiment, with its own controls, could. The second caution is that one success is not a guarantee. LaLonde's paper used a second survey, the Panel Study of Income Dynamics, and the same procedure there tells a less tidy story.

```
psid = pd.read_stata(url + "psid_controls.dta")
data2 = pd.concat([nsw[nsw["treat"] == 1], psid], ignore_index=True)
data2["pscore"] = smf.logit(better, data=data2).fit(disp=0).predict(data2)
t2, c2 = data2[data2["treat"] == 1], data2[data2["treat"] == 0]
nn2 = NearestNeighbors(n_neighbors=1).fit(c2[["pscore"]])
_, idx2 = nn2.kneighbors(t2[["pscore"]])
m2 = c2.iloc[idx2.ravel()]
sd2 = np.sqrt((t2[X].var() + c2[X].var()) / 2)

print(f"naive {t2['re78'].mean() - c2['re78'].mean():+,.0f}   "
      f"ATT {t2['re78'].mean() - m2['re78'].mean():+,.0f}   "
```

```
      f"distinct controls {m2.index.nunique()}")
print(((t2[X].mean() - m2[X].mean()) / sd2).astype(float).round(2).to_string
```

```
naive -15,205   ATT +1,665   distinct controls 56
age          0.11
education   -0.17
black        0.11
hispanic    -0.03
married      0.09
nodegree     0.21
re74         0.03
re75         0.01
```

The estimate is again close to the experiment, but balance is not achieved: four covariates stay outside 0.1, schooling and the no-diploma indicator by the most, and the whole match rests on a few dozen distinct controls. A careful analyst would report that, and would not present the closeness of the number as vindication, because in real work there is no experiment to compare with. A. Smith and E. Todd (2005) make the general point: the Dehejia–Wahba success depends on which subsample and which score, and the lesson is about the conditions under which matching works — rich pre-treatment earnings, comparison groups from the same labour market — rather than about matching as such.

10.8 The control that changes the question

Deciding what goes into the score is a claim about causal structure. The rule is to condition only on variables fixed **before** treatment, and the reason is clearest when it is broken.

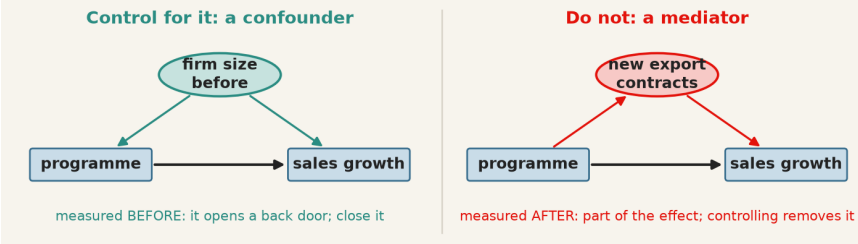


Figure 10.9. A confounder, measured before treatment, opens a back door and should be controlled for. A mediator, created by the treatment, carries part of the effect, and controlling for it removes that part.

```
rng = np.random.default_rng(10)
size = rng.normal(0, 1, n)                                    # before the pro
joins = (size + rng.normal(0, 1, n) > 0.5).astype(int)
```

```
contracts = 1.5 * joins + rng.normal(0, 1, n)              # created by the pr
growth = 10 + 2 * size + 0.5 * joins + 1.0 * contracts + rng.normal(0, 2, n)
firms = pd.DataFrame({"size": size, "joins": joins, "contracts": contracts, "gr

for formula in ["growth ~ joins", "growth ~ joins + size", "growth ~ joins + si
    print(f"{formula:36} {smf.ols(formula, data=firms).fit().params['joins']:.2
print("true total effect: 0.5 direct + 1.5 × 1.0 through contracts = 2.0")
```

```
growth ~ joins                       4.19
growth ~ joins + size                2.09
growth ~ joins + size + contracts    0.31
true total effect: 0.5 direct + 1.5 × 1.0 through contracts = 2.0
```

Without controls, selection on size inflates the estimate. Controlling for size — a confounder, fixed before the programme — recovers the total effect of 2. Adding new export contracts, which the programme itself creates, drops the estimate to the direct effect alone, because conditioning on a mediator holds fixed the channel through which most of the effect travels. The regression is not wrong; it answers a different and usually uninteresting question. The related case of a **collider**, a common consequence of treatment and outcome, is worse: conditioning on it *creates* an association that does not exist (Elwert and Winship 2014). Cinelli, Forney, and Pearl (2022) is the short guide worth keeping beside the keyboard.

> **"Closer to the right answer" is not the same as "the right method."** In real work the truth is unknown, so the only evidence available is whether the procedure is justified. A procedure that is wrong for a reason you can state will be wrong by an unknown amount in the next dataset, and no diagnostic will announce it.

10.9 What none of this fixes

Matching came within $210 of the truth on the CPS. It would be a mistake to conclude that it solved the problem, and the simulation shows why. Suppose the analyst sees only firm size, a noisy sign of the productivity that actually drove selection.

```
rng = np.random.default_rng(10)
productivity = rng.normal(0, 1, n)
sales_without = 10 + 3 * productivity + rng.normal(0, 2, n)
applies = (productivity + rng.normal(0, 1, n) > 0.5).astype(int)
size = productivity + rng.normal(0, 1, n)
firms = pd.DataFrame({"productivity": productivity, "size": size, "applies": ap
                      "sales": sales_without + 2 * applies})
t, c = firms[firms["applies"] == 1], firms[firms["applies"] == 0]
for v in ["size", "productivity"]:
    nn_v = NearestNeighbors(n_neighbors=1).fit(c[[v]])
```

```
    _, idx = nn_v.kneighbors(t[[v]])
    print(f"matched on {v:13} ATT {t['sales'].mean() - c.iloc[idx.ravel()]['
```

```
matched on size          ATT 3.94   (truth 2)
matched on productivity  ATT 2.21   (truth 2)
```

Matching on the variable that drove selection recovers the effect. Matching on a proxy for it leaves more than half of the bias in place — and size, the variable matched on, would look perfectly balanced. Balance can only be checked on what was measured; conditional independence is a claim about everything else, and it is untestable. On LaLonde's data, matching worked because the survey records two years of earnings before the programme, which capture most of what decided who enrolled. Remove them and no score recovers the experiment.

The honest response is to ask how wrong the assumption could be. Sensitivity analysis asks how strong an unobserved confounder would have to be — how strongly associated with both treatment and outcome — to overturn the conclusion. If the answer is "about as strong as the covariates already controlled for", the finding is fragile. If it is "stronger than any known determinant", it is robust. Either way the reader learns something a point estimate does not tell them.

The deeper limitation is that matching and regression adjustment are the *same assumption* in different clothing. Both require that conditioning on observed covariates suffices. Matching implements it better — it compares like with like instead of extrapolating along a line — but neither creates identification. When the assumption fails, both fail, and in the same direction.

One caution about the method itself. Propensity-score matching is the most popular approach here and not the best one: matching on a scalar score discards information and can *increase* imbalance as the caliper tightens, a pathology documented at length by King and Nielsen (2019). Matching directly on the covariates, or weighting rather than matching, generally does better. The score remains a good diagnostic for overlap even where it is a poor matching device.

Chapter 11 takes the other route: rather than assuming the confounders are all observed, it uses the structure of *when* treatment arrived to difference the unobserved ones away.

10.10 A short review of the literature

> The potential-outcomes notation is Donald B. Rubin (1974), though the idea traces to Neyman's 1923 work on agricultural experiments. Holland (1986) named the fundamental problem and drew the boundary the discipline still uses — "no causation without manipulation" — which remains contested and clarifying.
>
> LaLonde's 1986 comparison of experimental and non-experimental estimates of a job-training programme is the benchmark the matching literature is still judged against; Dehejia and Wahba (1999) showed propensity-score matching recovering the experimental answer, and A. Smith and E. Todd (2005) showed how much that depends on the sample and the specification.

Rosenbaum and Rubin (1983) proved the result that made observational adjustment tractable: if treatment is unconfounded given a covariate vector, it is unconfounded given the scalar propensity score alone, so a many-dimensional matching problem collapses to one dimension. Imbens (2004) surveys the estimators that followed and Stuart (2010) is the practical review, covering matching, weighting and the diagnostics that should accompany them.

The corrective literature is as important as the constructive one. King and Nielsen (2019) argues that propensity-score matching specifically — as opposed to matching in general — often increases imbalance and should be replaced by methods that match on covariates directly. Elwert and Winship (2014) sets out endogenous selection bias, the family of errors created by conditioning on colliders, and Cinelli, Forney, and Pearl (2022) gives the compact treatment of which controls help and which do damage.

10.11 What to report

1. **The estimand, named.** ATE or ATT — they answer different questions and matching usually estimates the second.
2. **The identifying assumption, in words.** Which variables are claimed to close every path from treatment to outcome, and why that list is complete. This is an argument, not a table.
3. **Balance before and after**, for every covariate, with standardised differences — a love plot. Not the propensity model's coefficients or its fit. And say that the score was settled before the outcome was looked at.
4. **Overlap**, with the range of the score in both groups, and what you did about units outside common support.
5. **The naive estimate alongside the adjusted one.** The distance between them is the size of the selection problem, and hiding it hides the reason for the analysis.
6. **A sensitivity analysis.** How strong an unobserved confounder would need to be to overturn the result.
7. **That every control is pre-treatment**, and that you know it. If a covariate is measured after treatment, say why it is not a mediator or a collider — and if you cannot, drop it, even when it improves the number.

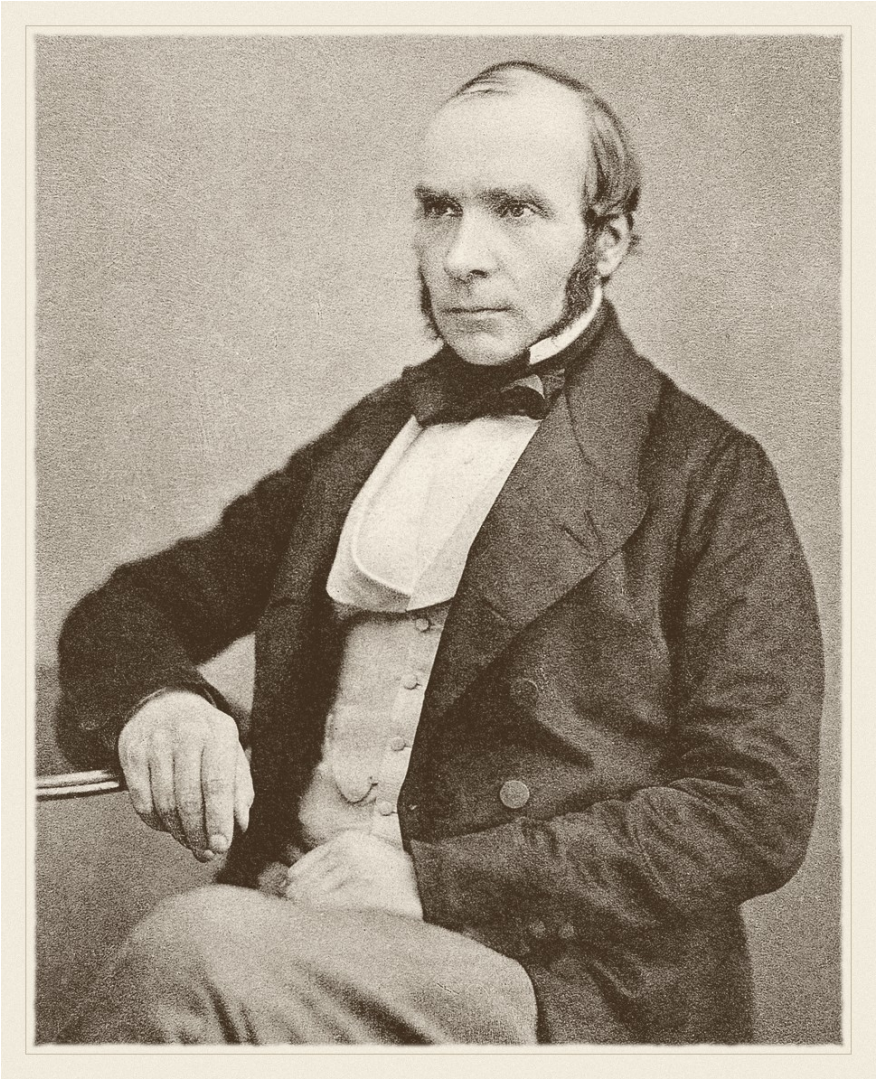
Chapter 11

Causal Inference II: Difference-in-Differences

11.1 What would have happened otherwise

Chapter 10 recovered a causal effect by assuming that the variables driving treatment were all observed, and then conditioning on them. That assumption is untestable and, in most settings, false — countries do not adopt policies for reasons that are fully captured by three indicators.

This chapter takes the opposite route. Rather than measuring the confounders, it uses *timing* to remove them: if a country's unobserved characteristics are stable over a short window, then comparing that country to itself before and after differences them away. The remaining problem — that things change over time for everyone — is handled by a second comparison, against units that did not adopt. Two differences, hence the name.



John Snow · 1813–1858
Mapped cholera deaths in 1854 and read a natural experiment off the water supply — before the statistics existed to justify it.
unknown, Autotype 1856, published in 1887. Public domain, via Wikimedia Commons.

From the archive · Snow, 1855

John Snow is remembered for the Broad Street pump, which is the wrong story. The map of cholera deaths clustered around one pump was suggestive, but a sceptic could reply that the people near the pump were poorer, more crowded, and different

in a dozen ways from those further off — the selection problem of chapter 10, in 1854.

His second study answered that objection, and it is the one that matters here. Two companies supplied water to overlapping districts of south London, their pipes running down the same streets so that neighbouring houses drew from different sources. In 1852 one of them, Lambeth, moved its intake upstream of the sewage outflows; the other, Southwark and Vauxhall, did not. Snow could therefore compare cholera mortality in households that differed in water supply but not, in any systematic way, in anything else — and compare the change over time between the two.

He called it "an experiment on the grandest scale", and was explicit that its force came from the *assignment*: no one chose their supplier, the pipes had been laid years earlier for commercial reasons unrelated to the disease. That is the argument every difference-in-differences paper is still making. Snow made it before the germ theory of disease existed — he had no mechanism, only a design.

11.2 The two differences

Let $\bar{Y}_{g,t}$ be the average outcome for group g in period t, with $g \in \{\text{treated}, \text{control}\}$ and $t \in \{\text{pre}, \text{post}\}$. The estimator is

$$\widehat{\text{DiD}} = \underbrace{\left(\bar{Y}_{\text{tr,post}} - \bar{Y}_{\text{tr,pre}}\right)}_{\text{change among the treated}} - \underbrace{\left(\bar{Y}_{\text{co,post}} - \bar{Y}_{\text{co,pre}}\right)}_{\text{change among the controls}} .$$

The first difference removes everything time-invariant about the treated group, observed or not — its wealth, its institutions, its geography. The second estimates what would have happened anyway, and subtracting it removes any shock common to both groups.

What licenses the subtraction is a single assumption, and it is worth stating in its exact form because the loose version is misleading.

> **Parallel trends** — in the absence of treatment, the average outcome of the treated group would have followed the same *trajectory* as the control group. Formally,
>
> $$\mathrm{E}[Y_{\text{tr,post}}(0) - Y_{\text{tr,pre}}(0)] = \mathrm{E}[Y_{\text{co,post}}(0) - Y_{\text{co,pre}}(0)].$$
>
> It is not an assumption that the groups are *similar* — they may differ by any amount in level — only that the gap between them would have stayed constant.

The assumption concerns $Y(0)$ for the treated group in the post period, which is precisely the counterfactual that does not exist. **Parallel trends is therefore untestable**, in the same way and for the same reason as chapter 10's conditional independence. What can be examined is whether the trends were parallel *before*

treatment, which is evidence about plausibility rather than a test of the assumption itself.

Note also that parallel trends is not invariant to how the outcome is measured. If two groups have parallel trends in levels they generally do not in logs, and vice versa, so the choice of scale is part of the identifying assumption rather than a presentational detail.

11.3 As a regression

The two-by-two table is easier to work with as a regression, and the regression form is one you have already met. With unit and period fixed effects,

$$Y_{it} \;=\; \alpha_i + \gamma_t + \delta\, D_{it} + \varepsilon_{it},$$

where D_{it} is 1 when unit i is treated in period t, the coefficient δ *is* the difference-in-differences estimate. That is chapter 6's two-way fixed effects specification with the treatment indicator in place of a continuous regressor: the unit effects absorb the first difference, the period effects the second.

The regression form buys three things — covariates, more than two periods and more than two groups, and standard errors. On the last, chapter 3's finding returns with force: DiD data are panels, residuals are serially correlated within unit, and treating the country-years as independent understates the standard error badly. Clustering by unit is the minimum, and the classic demonstration of what happens otherwise is Bertrand, Duflo and Mullainathan's, cited in chapter 3.

11.4 When treatment arrives at different times

The clean two-by-two case has one treatment date. Real policies arrive in different years for different units — which is exactly what the course fixture contains — and for two decades the profession simply put a treatment dummy into a two-way fixed effects regression and read off the coefficient.

That turns out to be wrong in a way nobody noticed until recently. With staggered timing, the TWFE estimate is a *weighted average* of all the two-by-two comparisons available in the data, and some of those comparisons use already-treated units as controls for later-treated ones. When effects vary over time — when the policy's impact grows, say — those comparisons subtract a treatment effect rather than a counterfactual trend, and the implied weights can be **negative** (Chaisemartin and D'Haultfœuille 2020; Goodman-Bacon 2021). A weighted average with negative weights can lie outside the range of the things being averaged, so the estimate can have the wrong sign while every underlying effect has the right one.

The remedy is to build the average yourself: estimate each cohort's effect against clean controls — units not yet treated, or never treated — and aggregate with weights you chose (Callaway and Sant'Anna 2021; Sun and Abraham 2021).

11.5 The data, and what it will and will not support

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
from plotstyle import setup, CL
plt = setup()

import numpy as np, pandas as pd, statsmodels.api as sm, qmib

Y, D = "ai_use_any", "has_ai_strategy"
TRUE_EFFECT = 2.4               # from scripts/build_spine/make_fixtures.py

wide = qmib.load("angle_c_country")

# Adoption year must come from the FULL treatment series, not from the rows
# the outcome happens to be observed. Two countries adopted in 2022 and have
# missing 2022 outcome; taking the first year they are *seen* treated would
# them in the 2023 cohort and silently misdate the policy.
adopted = wide[wide[D] == 1].groupby("geo")["time"].min()

panel = wide[(wide["time"] >= 2021) & wide[Y].notna()].copy()
panel["cohort"] = panel["geo"].map(adopted).fillna(9999).astype(int)

print(f"country-years with the outcome observed: {len(panel)}")
print(f"the outcome exists only from {int(panel.time.min())} — "
      f"the AI module did not exist before it\n")
print(f"{'cohort':>10}{'countries':>11}   pre-periods available")
for c in sorted(panel["cohort"].unique()):
    n = panel.loc[panel["cohort"] == c, "geo"].nunique()
    if c == 9999:
        print(f"{'never':>10}{n:>11}   —")
    else:
        pre = [int(t) for t in sorted(panel.time.unique()) if t < c]
        shown = ", ".join(str(t) for t in pre) if pre else "(none)"
        print(f"{c:>10}{n:>11}   {len(pre)}  {shown}")
```

```
country-years with the outcome observed: 112
the outcome exists only from 2021 — the AI module did not exist before it

    cohort  countries   pre-periods available
      2021          8   0  (none)
      2022          9   1  2021
      2023          1   2  2021, 2022
     never         12   —
```

Three facts from that table decide what is possible, and each is a constraint a real analyst meets constantly.

The 2021 cohort cannot be used at all. The outcome series begins in 2021 and those countries adopted in 2021, so there is no pre-treatment observation. No before, no difference, no difference-in-differences. Their data are not weak evidence — they are no evidence for this design.

The 2023 cohort is one country. An estimate from it is a single country's trajectory, and its standard error would be a fiction.

There is a trap in even getting this table right. Two countries adopted in 2022 and have no 2022 outcome recorded, so if the adoption year is taken from the rows where the outcome is observed — the obvious way to write it — they are assigned to the 2023 cohort and the policy is misdated by a year. The cohort must be read from the treatment series, which is complete, rather than from the analysis sample, which is not.

The 2022 cohort has exactly one pre-period. That is enough to compute the estimator and *not* enough to examine pre-trends, which needs at least two pre-periods to have a trend to look at.

11.6 The estimates

```
def twoby(cohort, pre, post):
    """Clean 2x2: one cohort against the never-treated, two periods.

    Returns NaN when either group is unobserved in either period — a small
    cohort can simply be absent from a year, and that is not an estimate of ze
    """
    sub = panel[panel["cohort"].isin([cohort, 9999]) & panel["time"].isin([pre
    tr = sub[sub["cohort"] == cohort].groupby("time")[Y].mean()
    co = sub[sub["cohort"] == 9999].groupby("time")[Y].mean()
    if not {pre, post} <= set(tr.index) or not {pre, post} <= set(co.index):
        return float("nan")
    return (tr[post] - tr[pre]) - (co[post] - co[pre])

# The specification two decades of papers used.
design = pd.concat([
    panel[[Y, D]].reset_index(drop=True),
    pd.get_dummies(panel["geo"], drop_first=True, dtype=float).reset_index(drop
    pd.get_dummies(panel["time"], prefix="yr", drop_first=True, dtype=float).re
], axis=1)
twfe = sm.OLS(design[Y], sm.add_constant(design.drop(columns=[Y]))).fit(
    cov_type="cluster", cov_kwds={"groups": panel["geo"].reset_index(drop=True

print(f"{'estimator':44}{'estimate':>10}{'bias':>9}")
print(f"{'the truth (from the generator)':44}{TRUE_EFFECT:>10.3f}{0.0:>9.3f}")
```

```
print(f"{'two-way fixed effects, all cohorts':44}"
      f"{twfe.params[D]:>10.3f}{twfe.params[D] - TRUE_EFFECT:>9.3f}"
      f"   (SE {twfe.bse[D]:.3f})")

print(f"\nclean 2x2 estimates, cohort against never-treated:")
clean = []
for cohort, pre in ((2022, 2021), (2023, 2022)):
    n = panel.loc[panel["cohort"] == cohort, "geo"].nunique()
    for post in range(cohort, 2025):
        v = twoby(cohort, pre, post)
        label = f"   cohort {cohort} ({n} countr{'y' if n == 1 else 'ies'}),
        if np.isnan(v):
            print(f"{label} not estimable (a group is unobserved that year)"
            continue
        clean.append((cohort, post, v, n))
        print(f"{label} {v:+7.3f}   bias {v - TRUE_EFFECT:+7.3f}")

usable = [v for c, _, v, n in clean if n > 1]
print(f"\naveraging only the cohort with more than one country: "
      f"{np.mean(usable):+.3f}   bias {np.mean(usable) - TRUE_EFFECT:+.3f}")
```

```
estimator                                        estimate     bias
the truth (from the generator)                      2.400    0.000
two-way fixed effects, all cohorts          2.811   0.411  (SE 0.834)

clean 2x2 estimates, cohort against never-treated:
   cohort 2022 (9 countries), 2021→2022:  +1.701   bias  −0.699
   cohort 2022 (9 countries), 2021→2023:  +2.702   bias  +0.302
   cohort 2022 (9 countries), 2021→2024:  +2.292   bias  −0.108
   cohort 2023 (1 country), 2022→2023:  +1.666   bias  −0.734
  cohort 2023 (1 country), 2022→2024: not estimable (a group is unobserved that

averaging only the cohort with more than one country: +2.232  bias −0.168
```

The two-way fixed effects estimate is close to the truth here, and it would be dishonest to stage a dramatic failure that the data do not produce. With four periods and effects that are roughly constant once treatment begins, the negative-weighting problem is mild. The point stands anyway: **the estimate is close for reasons you can only check by decomposing it**, and a paper reporting the TWFE coefficient alone offers no way to know whether its weights were benign.

The cohort estimates show what the decomposition looks like. The 2022 cohort — nine countries with one clean pre-period — gives +1.70, +2.70 and +2.29 in successive years, averaging +2.23 against a truth of +2.40, with the sampling noise you would expect from that many units. The 2023 cohort is a single country; its one estimable comparison happens to land at +1.67, and the second is not estimable

at all because that country is absent from the final year. Both are reported rather than quietly dropped, because a cohort of one is a fact about the design and hiding it would make the evidence look stronger than it is.

11.7 The estimator has a name

The loop in that cell is not an improvisation. Estimating each cohort against a clean comparison group and aggregating with weights you chose is precisely the procedure of Callaway and Sant'Anna (2021), and there is a Python implementation of it, `diff-diff`. It is a young package and the reference implementation is still R's `did`, so the first thing to do with it is what this section does: run it beside a calculation you already understand, and find out where the two disagree.

That is worth doing here for three reasons. It states out loud which cells it refuses to estimate. It attaches standard errors to the cohort estimates, which the hand-rolled loop had no way to produce. And it does disagree — in a way that turns out to be the hand-rolled version's fault.

```
import diff_diff as dd

# The estimator wants one column holding the adoption year, and 0 — not a
# sentinel it would read as a year — for the never-treated.
panel["g"] = panel["cohort"].replace(9999, 0)

cs = dd.CallawaySantAnna(cluster="geo").fit(
    panel, outcome=Y, unit="geo", time="time", first_treat="g")
gt = cs.to_dataframe()

print(f"{'cohort':>7}{'year':>6}{'ATT(g,t)':>10}{'SE':>7}")
for _, r in gt.iterrows():
    g, t = int(r.group), int(r.time)
    if pd.isna(r.effect):
        print(f"{g:>7}{t:>6}{'–':>10}{'–':>7}   dropped: {r.skip_reason}")
    else:
        tag = "  ← pre-treatment, not an effect" if t < g else ""
        print(f"{g:>7}{t:>6}{r.effect:>10.3f}{r.se:>7.3f}{tag}")

print(f"\n{'overall ATT':>18}{cs.overall_att:>10.3f}{cs.overall_se:>7.3f}"
      f"   bias {cs.overall_att - TRUE_EFFECT:+.3f}, {cs.n_clusters} clusters")
for _, r in cs.aggregate("group").to_dataframe().iterrows():
    n = panel.loc[panel["cohort"] == int(r.label), "geo"].nunique()
    print(f"{'cohort ' + str(r.label):>18}{r.att:>10.3f}{r.se:>7.3f}"
          f"   {n} countr{'y' if n == 1 else 'ies'}")

# The hand-rolled estimates and these ones differ. Recomputing each 2x2 over
# only the countries observed in *both* of its two years isolates why.
```

```
print("\nthe hand-rolled numbers, and what was wrong with them:")
print(f"{'g,t':>9}{'as written':>12}{'same units':>12}{'ATT(g,t)':>10}")
for g, t in ((2022, 2022), (2022, 2023), (2022, 2024), (2023, 2023)):
    sub = panel[panel["cohort"].isin([g, 9999]) & panel["time"].isin([g - 1,
    seen = sub.groupby("geo")["time"].nunique()
    bal = sub[sub["geo"].isin(seen[seen == 2].index)]
    tr = bal[bal["cohort"] == g].groupby("time")[Y].mean()
    co = bal[bal["cohort"] == 9999].groupby("time")[Y].mean()
    fixed = (tr[t] - tr[g - 1]) - (co[t] - co[g - 1])
    att = gt.loc[(gt.group == g) & (gt.time == t), "effect"].iloc[0]
    print(f"{f'{g},{t}':>9}{twoby(g, g - 1, t):>12.3f}{fixed:>12.3f}{att:>10
```

```
 cohort  year  ATT(g,t)     SE
   2021  2021         –      –   dropped: missing_period
   2021  2022         –      –   dropped: missing_period
   2021  2023         –      –   dropped: missing_period
   2021  2024         –      –   dropped: missing_period
   2022  2022     1.739  1.135
   2022  2023     2.751  0.843
   2022  2024     2.138  0.977
   2023  2022    -1.126  0.440  ← pre-treatment, not an effect
   2023  2023     1.842  0.553
   2023  2024         –      –   dropped: zero_treated_control

       overall ATT     2.196  0.782   bias -0.204, 30 clusters
       cohort 2022     2.209  0.809   9 countries
       cohort 2023     1.842  0.553   1 country

the hand-rolled numbers, and what was wrong with them:
      g,t  as written  same units  ATT(g,t)
2022,2022       1.701       1.739     1.739
2022,2023       2.702       2.751     2.751
2022,2024       2.292       2.138     2.138
2023,2023       1.666       1.842     1.842
```

The first table is this chapter's own reasoning, produced by the estimator rather than argued for in advance. The 2021 cohort is dropped as `missing_period` — no pre-treatment observation exists, which is the conclusion reached two sections ago by reading the adoption structure. ATT(2023, 2024) is dropped as `zero_treated_control`, the same absent country the hand-rolled version returned `NaN` for. Neither refusal is buried inside a coefficient.

The standard errors are new. They come from the influence function of each group-time effect, clustered by country, and they carry through to the aggregate: an overall ATT of +2.20 with a standard error of 0.78, against a truth of 2.40. The interval covers it, which is the honest summary of what thirty countries over four years can

establish about an effect of this size.

> **Read the cohort lines again.** The nine-country cohort reports a standard error of 0.81. The one-country cohort reports 0.55. One country's trajectory is not measured more precisely than nine countries' — the smaller number is what comes out when there is no between-country variation left to estimate, and the software has no way to know that a cohort of one should not be reported at all. The estimator computes whatever you ask it for. Deciding what is worth asking for is still yours.

The last table is the uncomfortable one. The hand-rolled estimates and Callaway–Sant'Anna's differ — +1.70 against +1.74, +2.29 against +2.14 — and the middle column says why. `twoby` averaged whichever countries happened to be present in each year, and this panel is not balanced: Malta and Estonia have no 2022 observation, Luxembourg no 2021, Spain no 2023, Croatia neither. The "before" mean and the "after" mean were therefore computed over different sets of countries, and part of what looked like an effect was the change in who was being averaged. Hold the units fixed and the estimator is reproduced to three decimals.

That failure is on this chapter's own list — it is the **composition change** warning further down, committed in code written to demonstrate the method, on a dataset small enough to audit by hand. Which is the argument for using the package rather than the argument against it: not that it is cleverer, but that the choices the hand-rolled version made silently are ones it is forced to make out loud.

11.8 Pre-trends: what this data cannot tell you

```
cohort = 2022
sub = panel[panel["cohort"].isin([cohort, 9999])].copy()
sub["treated"] = (sub["cohort"] == cohort).astype(int)

tr = sub[sub["treated"] == 1].groupby("time")[Y].mean()
co = sub[sub["treated"] == 0].groupby("time")[Y].mean()
years = sorted(sub["time"].unique())
eff = [(t - cohort, (tr[t] - tr[2021]) - (co[t] - co[2021])) for t in years]

fig, ax = plt.subplots(figsize=(6.8, 4.0))
xs = [e for e, _ in eff]; ys = [v for _, v in eff]
ax.axhline(0, color=CL.line, lw=1)
ax.axhline(TRUE_EFFECT, color=CL.good, lw=1.4, ls=":", label="true effect")
ax.axvline(-0.5, color=CL.muted, lw=1.2, ls="--")
ax.plot(xs, ys, marker="o", ms=7, lw=1.8, color=CL.accent, zorder=3)
ax.annotate("reference period\n(zero by construction)", (-1, 0),
            textcoords="offset points", xytext=(6, 26), fontsize=8, color=CL.mu
ax.set_xlabel("years since adoption"); ax.set_ylabel("estimated effect (pp)")
```

```
ax.set_xticks(xs)
ax.legend(frameon=False, fontsize=8.5, loc="lower right")
fig.tight_layout()

print(f"{'event time':>12}{'estimate':>11}")
for e, v in eff:
    note = "  reference — not evidence" if e == -1 else ""
    print(f"{e:>12}{v:>11.3f}{note}")
print(f"\npre-periods available for this cohort: "
      f"{sum(1 for e, _ in eff if e < 0)}")
```

```
  event time   estimate
          -1      0.000  reference — not evidence
           0      1.701
           1      2.702
           2      2.292

pre-periods available for this cohort: 1
```

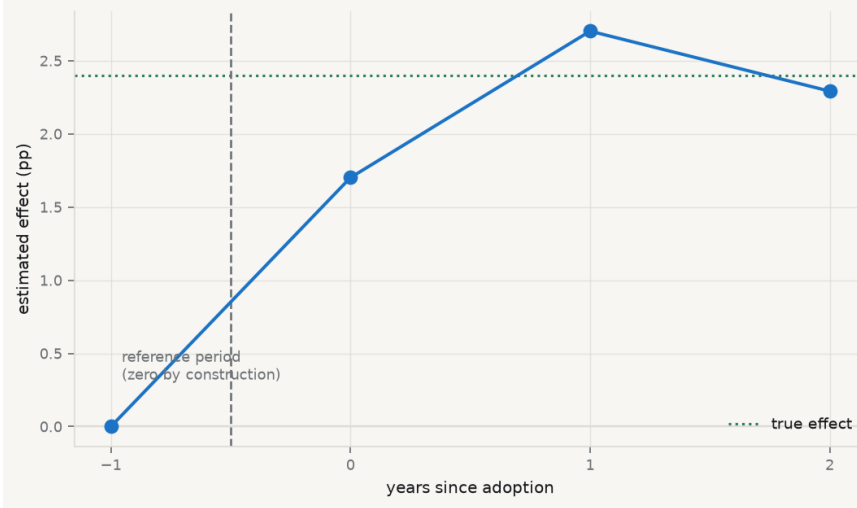


Figure 11.1. Event study for the 2022 cohort against the never-treated. Event time 0 is the year of adoption. The point at −1 is the reference period and is zero by construction, not by evidence — with a single pre-period there is no pre-trend to examine.

An event study is the standard way to make parallel trends look plausible: plot the estimated effect at each period relative to adoption, and look for a flat line before treatment. Here the pre-treatment portion consists of a single point, which is the reference period and therefore exactly zero by arithmetic.

A flat pre-trend that contains one mechanically-zero point is not evidence of anything. This is not a defect of the analysis; it is a property of the data, and it means the identifying assumption of this chapter cannot be examined at all in this dataset. The honest report says so, states the estimate as conditional on an assumption that could not be checked, and treats the finding accordingly.
It is worth noticing how much weaker that is than what chapter 10 could do. There the assumption was also untestable, but the covariates could at least be shown to be balanced. Here, there is nothing to show.

11.9 How large a violation would it take?

"The assumption cannot be checked" is where this dataset leaves you. It is not where the literature leaves you. Rambachan and Roth (2023) changes the question: rather than testing parallel trends, ask how large a violation of it would have to be before the finding stops holding, and report *that* number. The output is a family of intervals indexed by how much post-treatment departure from parallel trends you are prepared to allow — measured, in the relative-magnitudes version, as a multiple M of the largest departure actually visible before treatment.

```
es = cs.aggregate("event_study")

print("the whole of the pre-treatment evidence in this dataset:")
for _, r in es.to_dataframe().query("event_time < 0").iterrows():
    print(f"   event time {int(r.event_time):>3}: {r.att:+.3f}  "
          f"(SE {r.se:.3f}, p {r.p_value:.3f}), from {int(r.n)} cohort")

print(f"\n{'M':>5}{'identified set':>22}{'95% robust CI':>22}   rules out zero'
for M in (0.0, 0.25, 0.5, 1.0, 2.0):
    h = dd.HonestDiD(method="relative_magnitude", M=M).fit(es)
    print(f"{M:>5.2f}   [{h.lb:>7.3f},{h.ub:>7.3f}]     "
          f"[{h.ci_lb:>7.3f},{h.ci_ub:>7.3f}]   "
          f"{'yes' if h.is_significant else 'no'}")
```

```
the whole of the pre-treatment evidence in this dataset:
   event time  -1: -1.126  (SE 0.440, p 0.016), from 1 cohort

    M        identified set         95% robust CI   rules out zero
 0.00   [  2.212,  2.212]     [  0.613,  3.812]   yes
 0.25   [  1.649,  2.776]     [  0.050,  4.375]   yes
 0.50   [  1.086,  3.339]     [ -0.513,  4.938]   no
 1.00   [ -0.040,  4.465]     [ -1.640,  6.064]   no
 2.00   [ -2.293,  6.718]     [ -3.892,  8.317]   no
```

Two things to read off that, and the second matters more.
The pre-treatment evidence is a single number, from a single cohort, and that cohort is a single country. Germany adopted in 2023, so it is the only unit whose pre-period

is something other than its own reference year, and its 2021→2022 change ran 1.13 points below the never-treated countries'. Everything in the table below rests on that one comparison. The 2022 cohort, nine countries strong, contributes nothing here at all — its only pre-period *is* the reference.

The sensitivity table then says: the result survives at $M = 0.25$ and not at $M = 0.5$. In words, if post-treatment departures from parallel trends could be as much as half the size of the one pre-treatment departure visible in the data, the interval already contains zero. That is a low breakdown point, and it ought to be — it is what an estimate resting on one country's single pre-period is worth.

> **This is not a test, and nothing here has been tested.** The honest sentence is "the finding requires post-treatment violations of parallel trends no larger than about a quarter of the single pre-treatment deviation we can observe, and we have one country from which to judge that." A sensitivity analysis converts an untestable assumption into a statement about how much of it the result needs. That is real progress over asserting the assumption, and it is not the same thing as evidence for it.

11.10 What difference-in-differences cannot fix

Anything that changes at the same time as treatment. Fixed effects remove what is constant; the second difference removes what is common. A shock hitting only the treated group in the treatment year is indistinguishable from the treatment, and no amount of data separates them. Countries that adopt an AI strategy in a given year may be doing several other things that year.

Anticipation. If units change behaviour before adoption because they know it is coming, the pre-period is already contaminated and the "before" is not a clean baseline. This shows as a non-zero pre-trend, when there are enough pre-periods to see one.

Composition change. If the units in the panel change — countries entering or leaving the sample, or a survey changing who it asks — the first difference is comparing different populations rather than the same one over time.

Spillovers. The control group must be unaffected. If a national strategy in one country shifts behaviour in its neighbours, the controls are partially treated and the estimate is attenuated toward zero.

11.11 A short review of the literature

The design is older than its name: Wing, Simon, and Bello-Gomez (2018) traces it to Snow's 1855 water study and sets out the modern practice, including the pre-trend examination and the sensitivity checks a credible paper is expected to show.

The last decade has been dominated by a single discovery about staggered adoption. Goodman-Bacon (2021) decomposed the two-way fixed effects estimator into the un-

derlying two-by-two comparisons and showed that some of them use already-treated units as controls; Chaisemartin and D'Haultfœuille (2020) showed that the implied weights can be negative, so the estimand may lie outside the range of every underlying effect. Neither paper is a curiosity — between them they called into question a large body of published work.

The constructive replacements estimate cohort-specific effects against clean comparison groups and aggregate them explicitly. Callaway and Sant'Anna (2021) gives the group-time average treatment effects and the aggregation schemes; Sun and Abraham (2021) does the equivalent for event-study specifications, where the same negative weighting contaminates the coefficients on individual leads and lags.

Where the pre-trend evidence is thin — and outside a handful of long panels it usually is — Rambachan and Roth (2023) replaces the ritual pre-trend test with a sensitivity analysis. Report the range of estimates consistent with violations of parallel trends up to a stated size, and the reader can see how much of the assumption the finding actually needs rather than being told it holds.

On inference, the reference remains Bertrand, Duflo and Mullainathan, discussed in chapter 3: DiD data are serially correlated panels, and standard errors that ignore it reject true nulls at many times the nominal rate.

11.12 What to report

1. **The adoption structure.** How many units in each cohort, how many periods before and after, and which cohorts are unusable. A table of this is worth more than a paragraph asserting the design is sound.
2. **The parallel-trends argument**, as an argument. Why should the gap have stayed constant? Institutional reasoning, not a citation to the assumption.
3. **An event study**, with the reference period marked as such — and if there are too few pre-periods to see a trend, say that instead of implying the flat line is evidence.
4. **A staggered-robust estimator alongside TWFE.** If they agree, the weights were benign and you have shown it. If they disagree, the decomposition is the finding. `diff-diff`'s `CallawaySantAnna` is the Python route; the `did` package is the R one.
5. **Standard errors clustered by unit**, with the number of clusters. Below about forty, say what you did about it.
6. **The scale.** Levels or logs is part of the assumption, because parallel trends in one is not parallel trends in the other.
7. **What could have moved at the same time.** The most credible DiD papers name the confounding events they considered and explain why each is implausible — which is the modern version of Snow pointing out that nobody chose their water company.
8. **How much violation the result survives.** A breakdown value — the size of departure from parallel trends at which the finding stops holding — is worth more than a pre-trend p-value, and unlike the p-value it can be reported even when there is nothing to test.

Chapter 12

Conclusion

12.1 One argument, twelve times

The introduction asked when a number about the past can be used to make a claim about the future, and said that every method in this book is an answer to some version of that question, each with conditions attached. Eleven chapters later the conditions are the thing worth collecting, because the methods will be looked up again and the conditions will not.

The single sentence the book has been making is this: **a quantitative result is an argument, and the arithmetic is only its smallest part.** What makes a number worth acting on is the case that can be made for the conditions under which it means what it appears to mean — and that case is constructed by the analyst, not computed by the software.

Five things recurred often enough to be worth naming as the book's actual content.

12.2 The arithmetic never refuses

Chapter 3 put it first because everything else depends on it. The normal equations return coefficients for any numbers you feed them. Logistic regression converges on separated data by running its coefficients to infinity, and reports a large number rather than an error. Nearest neighbours predicts confidently outside the range of its training data. A structural equation model fits the covariance matrix implied by an entirely wrong causal diagram.

None of these failures announces itself. A misspecified model does not raise an exception; it returns four decimal places in the same typeface as a correct one. This is the reason diagnostics exist as a discipline rather than a formality, and the reason the phrase "the model says" is never a complete sentence.

12.3 The same data answer differently depending on what you ask

Chapter 6 is the clearest case. One dataset, one pair of variables, three defensible specifications, and estimates spanning a factor of four — 1,111 pooled, 275 with country effects, 1,149 with country and year effects. Each was correct for the question its identifying variation could answer, and a reader shown any one alone would draw a different conclusion.

The same pattern appeared everywhere once the book started looking. Chapter 2's slope fell 26% when two controls were added, because "controlling for" is literally a residual regression. Chapter 10's estimate ran from +6.07 to +2.68 depending on whether selection was addressed. Chapter 7's first component explained 97.5% of

the variance or 47%, depending only on whether the columns had been standardised.
The lesson is not that results are arbitrary. It is that **a coefficient is meaningless without the specification that produced it**, and reporting one without the other is reporting half a result.

12.4 Fit does not identify structure

Chapter 7 showed that a factor solution and its rotation fit identically to eight decimal places while telling different stories. Chapter 9 showed two structural models with opposite causal arrows returning identical χ^2, identical degrees of freedom, identical CFI and identical AIC. Chapter 3 showed four datasets sharing every summary statistic and differing completely.
In each case the data were silent between the alternatives, and the choice was made by the analyst on grounds outside the data. That is not a defect to be engineered away. It is the permanent condition of empirical work, and the honest response is to name the alternative you rejected and say why — which is a different activity from reporting a fit index.

12.5 Flexibility is a purchase, not a virtue

Chapter 5 introduced a deliberately biased estimator and showed it beating the unbiased one, because unbiasedness is worth nothing if the variance is large enough. Chapter 8 made the trade explicit and then measured it: squared bias, variance and irreducible error summing to the test error at every setting of k, with a minimum somewhere in the middle.
Then chapter 8 spent its own method's budget and reported the bill. Nearest neighbours on real data scored an AUC of 0.51 — a coin flip — where logistic regression, the rigid method with the strong assumption, reached 0.68. The flexibility did not earn its keep, and on accuracy alone the comparison would have shown all methods identical and concluded that nothing worked.
Complexity has to be justified against a simpler alternative that was actually fitted. Not against an imagined one.

12.6 Selection is the problem, and it is not a technicality

The last two chapters were about the gap between an association and an effect, and the size of that gap is worth remembering as a number rather than a warning. In chapter 10 the naive comparison overstated a known effect of +2.4 percentage points by more than 150%, because the countries that adopted the policy were already different. Nothing in the fit statistics of that naive comparison would have suggested a problem.
That gap is the reason for the whole apparatus of matching, balance diagnostics, parallel trends and event studies — none of which removes the need for an argument

about *why* treatment arrived where it did.

12.7 What the book checked, and what it could not

Something unusual was possible here. Because the course data are fixtures generated from a known structure, several claims could be *verified* rather than asserted, which is the form of evidence this book has been arguing for throughout.

```
import sys, warnings
sys.path.insert(0, "../slides"); sys.path.insert(0, "..")
warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, statsmodels.api as sm, qmib

core = qmib.load("core").dropna(
    subset=["gdp_pc_eur", "productivity_idx", "employment_ths", "population"
y = core["gdp_pc_eur"]

# 1. Pythagoras — chapter 2. The residual is orthogonal to the fit, so the
#    sums of squares add exactly.
fit = sm.OLS(y, sm.add_constant(core[["productivity_idx"]])).fit()
tss = ((y - y.mean()) ** 2).sum()
ess = ((fit.fittedvalues - y.mean()) ** 2).sum()
rss = (fit.resid ** 2).sum()

# 2. Frisch–Waugh–Lovell — chapter 2. "Controlling for" is residual regressi
others = sm.add_constant(core[["employment_ths", "population"]])
multi = sm.OLS(y, sm.add_constant(
    core[["productivity_idx", "employment_ths", "population"]])).fit()
fwl = sm.OLS(sm.OLS(y, others).fit().resid,
             sm.OLS(core["productivity_idx"], others).fit().resid).fit()

# 3. Leverage — chapter 3. The hat matrix diagonal sums to the number of
#    parameters, always, for every regression ever fitted.
X = sm.add_constant(core[["productivity_idx"]]).to_numpy()
h = np.einsum("ij,jk,ik->i", X, np.linalg.inv(X.T @ X), X)

print(f"{'identity':44}{'discrepancy':>14}")
print(f"{'TSS - (ESS + RSS)':44}{tss - ess - rss:14.3e}")
print(f"{'multiple regression - FWL residual slope':44}"
      f"{abs(multi.params['productivity_idx'] - fwl.params.iloc[0]):14.3e}")
print(f"{'sum of leverages - p':44}{abs(h.sum() - X.shape[1]):14.3e}")
print("\nThese are not approximations that happened to work on this dataset.
print("They are algebraic identities, and they hold for every regression.")
```

```
identity                                        discrepancy
```

```
TSS − (ESS + RSS)                              −3.052e−05
multiple regression − FWL residual slope        4.840e−09
sum of leverages − p                            2.265e−14

These are not approximations that happened to work on this dataset.
They are algebraic identities, and they hold for every regression.
```

Those hold exactly. Other things the book checked did not, and the failures were more instructive than the successes: the lasso retained three pure-noise columns out of ten after cross-validated selection; regression adjustment on proxy covariates left 0.7 percentage points of bias; a measurement model taken from an exploratory factor solution failed every fit index; and a post-treatment control moved an estimate *closer* to the truth while being methodologically wrong.

That last one is the book's most useful result. **Being closer to the right answer is not the same as using the right method**, and outside a textbook you never learn which one you had.

12.8 What is not here

A book of twelve chapters leaves out more than it contains, and knowing the shape of the gap matters when you meet a problem this book did not prepare you for.

Time series proper. The panels here are short and wide. Long series bring autocorrelation structure, unit roots, cointegration and forecasting — a substantial field that chapter 6's clustering only gestures at.

Bayesian inference. Everything here is frequentist. The Bayesian treatment answers a different question — the probability of a parameter given the data, rather than of the data given a parameter — and is often the more natural framing for the decision problems in this book's examples.

Modern machine learning at scale. Gradient boosting, random forests and neural networks are absent. Chapter 8's bias–variance framework is the right foundation for all of them, and chapter 5's regularisation is the mechanism they use, but the methods themselves and their tooling are a course of their own.

Text, networks and unstructured data. The introduction cited work measuring policy uncertainty from newspaper archives and productive capability from export composition. Constructing quantities from unstructured sources is where much of the growth in this field is, and it is not covered here.

Causal machine learning. Double machine learning, causal forests and the literature that lets flexible methods estimate treatment effects sit exactly at the junction of chapters 5, 8 and 10, and are the natural next thing to read.

12.9 What to carry

Six habits, which will outlast every technique in this book.

1. **Look at the data before modelling it.** Nearly every analysis that goes wrong

went wrong before the model was fitted.
2. **Say the units.** A coefficient without units cannot be checked against anything a reader knows, and checking against what you already know is the fastest way to catch an implausible result.
3. **Report the uncertainty, and say what kind.** Classical, robust, clustered — they differed by more than a factor of two on one coefficient in chapter 6.
4. **Name the assumption that identifies the estimate**, and say whether it is testable. Most of the important ones are not, which is exactly why they must be argued rather than assumed.
5. **Fit the simpler thing too.** Complexity should have to beat something.
6. **Say what the result does not license.** Every chapter here ends with that section because it is the part that distinguishes analysis from assertion with numbers attached.

And the one that subsumes them: **a well-argued refusal is a complete result.** "This cannot be claimed from these data, and here is what would be needed" is often the most valuable output of a serious analysis, and it is the answer that takes the most expertise to produce.

12.10 Where the work goes now

The methods in this book are ordinary. What is not ordinary — and what the introduction argued the field is short of — is the judgement to know which question a given number can answer, and the discipline to say so when it cannot. Language models have made the arithmetic and the code effectively free. They have not made that judgement free, and there is no sign that they will.

That is the argument for learning to derive a method before using it. Not because the derivation will be needed at the keyboard, but because it is what lets you notice that the answer on the screen cannot be right.

Appendix A

Setting up your tools

The toolchain for this course is a text editor without telemetry, a language model that runs on your own laptop, and a version control system. Every one of them is open source, and none of them requires an account, a subscription or a network connection to work. That is not frugality. Each choice teaches something the alternative would obscure, and the reasons are worth stating before you spend ninety minutes installing them.

A.1 VS Codium, rather than VS Code

Visual Studio Code is developed in the open, under a permissive licence. The build Microsoft distributes is not the same artefact: the binary adds telemetry and ships under a proprietary licence, and its extension marketplace is a Microsoft service with terms of its own. VS Codium is the identical source, compiled without the telemetry and the branding.

The practical reason for preferring it is that in this course you will open data you do not own. The teaching data are public, but the habit is not about these data — it is about the working file you will one day open under a non-disclosure agreement or a research ethics protocol. An editor that reports usage to a third party is a decision you should make deliberately, once, rather than by default.

The larger reason is that it makes visible something students consistently underestimate: how much of the commercial software estate is open source underneath. This is not a marginal phenomenon. The servers that host almost every service you use run Linux. Every Android phone runs a Linux kernel. Every browser you are likely to open is built on Chromium or WebKit, both open source. SQLite — public domain, maintained by three people — is embedded in essentially every phone, browser and operating system in the world. The scientific stack this course runs on, Python and NumPy and pandas and scikit-learn and statsmodels, is open source without exception, and the commercial data science platforms sold at considerable expense are, in large part, packaging around exactly those libraries.

The point for a business student is not ideological. It is that the capability you are buying when you buy a platform is frequently available directly, that the difference in price is buying convenience and support rather than capability, and that knowing which is which is a procurement skill.

A.2 Qwen 2.5 Coder, running locally

The model you install is Chinese, developed by Alibaba. It is worth being direct about that, because the reflex is to treat provenance as the security question, and

here it is not.

What matters is that the weights are open and the model runs on your machine. An open-weight model downloaded to your laptop and run offline cannot send your data anywhere; it has no network connection to send it over. A closed model behind an API, wherever the company is domiciled, receives everything you type by construction. On the dimension people actually care about — does my work leave this room — local open weights are the stronger guarantee, and the nationality of the developer does not enter into it. You can verify this claim rather than trust it: disconnect from the network and watch the model keep working.

This is also your first encounter with a distinction the course returns to. Qwen 2.5 Coder is a *vertical* model: specialised for code, and consequently far better at code, for its size, than a general assistant many times larger. A three-billion-parameter model on a laptop is not competitive with a frontier system at open-ended reasoning. At completing a pandas idiom it is entirely adequate. Choosing a small specialised model over a large general one — and knowing which tasks admit that trade — is the applied judgement that separates a sensible AI deployment from an expensive one, and it is the same judgement that chapter 5 will make about a deliberately biased estimator.

The course installs the 3-billion variant rather than the 7, and the reason is worth stating because it is the same reasoning in miniature. Most student laptops have no dedicated graphics card, so the model runs on the processor. On a processor the 7B is not a better model: it produces a comparable answer to the same question at roughly a quarter of the speed, and a tool slow enough to interrupt your thinking is a tool you stop using. The binding constraint is not quality, it is whether you will actually reach for it.

The standing rule of the course applies to it from the first day: every deliverable must document at least one instance where you identified a model output as wrong, unverifiable or misleading, and explain how you established that. You are never penalised for using the model. You are penalised for using it uncritically.

A.3 Two ways of using it, deliberately

The same model is reached through two different tools, and the split is not redundancy.

Continue puts the model in the editor. You select code and ask about it, and the answer arrives beside the file. This is the mode that matters most in a course where you will read far more code than you write — the session decks, the authors' replication packages, your group's scripts. The friction of copying code into a browser is small, and it is enough to stop most people asking.

aider puts the model in the terminal, inside your repository, where it can change files and commit what it changed. That is a different kind of act, and the course treats it as one. You see every command it proposes before it runs, and whatever it writes is committed under your name — which is the point, since the commit history is one of the records used to check that all three members of a group did the

work. Delegating the typing does not delegate the answerability.

The honest limitation, stated once: on a machine without a graphics card, a small model is a capable reader and an unreliable editor. It will explain a function correctly and then fail to produce a well-formed multi-file edit, because emitting a precise diff is a harder task than describing code. That is a fact about small models rather than about you, and the practical response is to use aider for understanding and do the editing yourself — which is, in any case, the position the course would prefer you were in.

A.4 git

git is the least glamorous item on the list and the one most likely to matter to your career.

Its immediate function is to make your work reproducible. This book takes the position that a result which does not reproduce from a clean clone is not a result — and that standard is empty without a mechanism. git is the mechanism. It records what the analysis was at the moment the number was produced, so that "we got 1,111" becomes a claim someone can check rather than one they must accept.

Its second function is provenance under disagreement. In group work, the log answers questions that memory cannot: who changed the specification, when, and what the estimate was before. That is why the individual multiplier on group work is gated on the git history rather than on assertion. It is not surveillance; it is the same reason laboratory notebooks are dated.

Its third function is that it is how the rest of the field works. The replication packages you will download are distributed as repositories. The libraries you rely on develop in public, and their issue trackers are often the only accurate documentation of a bug you have just hit. Being able to read a diff, open an issue and submit a fix is a professional literacy, and the distance between using open source and contributing to it is one commit.

You will not become fluent in git this term. You need four commands — `clone`, `add`, `commit`, `push` — and the understanding that a commit is a checkpoint you can return to. The rest can wait.

A.5 What they have in common

Open source, no account required, and everything running on hardware you control. It means the whole course works with no internet in the room, that nothing you do is contingent on a vendor's pricing decision, and that every result in this book can be reproduced by anyone, on a modest machine, without paying for the privilege.

That last property is not a convenience. It is what makes the work checkable, and checkability is the only thing that separates quantitative analysis from assertion with numbers in it.

A.6 Doing it

> Budget **60–90 minutes for the first half, before Session 1**, and **45 minutes for git, before Session 2**. Installation time is not class time.

A.7 Setup checklist — VS Codium, Python, and a local LLM

complete before Session 01 · budget 60–90 minutes

> **The walkthrough is the slide deck**, `MATH60033A-S01-Pre-Session.pptx` in this folder. Work through that. This page is the checklist you scan afterwards, plus the troubleshooting table. If the two ever disagree, the deck is right.

A.7.1 Why this stack

Choice	Reason
VS Codium rather than VS Code	Same editor, telemetry removed. Your work involves confidential and licensed data; the tooling should not exfiltrate it.
A local LLM rather than a hosted one	Your data never leave your machine. You also learn what is achievable *without* a frontier system — the realistic institutional constraint.
Plain Python + git rather than a notebook platform	Reproducibility is graded. A commit history is auditable; a notebook's execution state is not.

Tooling is a governance decision, and you should be able to defend yours. That is the first substantive lesson of the course.

A.7.2 What your machine needs

No graphics card is required. Everything runs on the processor, more slowly.

	Enough	Comfortable
RAM	8 GB	16 GB
Free disk	12 GB	20 GB

	Enough	Comfortable
CPU	anything from the last five years	Apple Silicon, or Intel i7 10th gen and later
Graphics	none	8 GB VRAM, which makes the model 5–10× faster

Speed, in tokens per second — roughly a short word each:

	tokens/s	a short answer
Dedicated GPU, or Apple Silicon	15–40	a second or two
CPU only, `qwen2.5-coder:3b`	8–20	a few seconds
CPU only, `qwen2.5-coder:7b`	3–8	half a minute or more

Use the 3b unless `ollama ps` reports `100% GPU`. On a processor the 7b is not a better model — it is the same answer, four times slower.

A.7.2.1 Per tool

	RAM	Disk	Notes
Ollama	8 GB for a 3b, 16 GB for a 7b	~2 GB per model	A model needs its file size plus about a fifth, in memory
Continue	the model's, plus nothing much	~50 MB	Does **not** manage the context window — set it yourself
aider	as Ollama	its own environment via pipx	Sets the context window per request itself

aider on a CPU-only machine: fine for reading and asking, unreliable for editing. Small models often fail to produce its edit format and report *"the LLM did not conform to the edit format"* instead of making a change. If you want it to edit anyway, `--edit-format whole` asks it to rewrite files rather than produce diffs, which a 3b manages far more often.

A.7.3 The checklist

Everything typed goes in the **VS Codium integrated terminal** (`View → Terminal`, or `Ctrl+`` `) — never the macOS Terminal app.

- ☐ **VS Codium** installed from https://vscodium.com and it opens
- ☐ **Extensions** installed: Python (`ms-python.python`), Jupyter (`ms-toolsai.jupyter`), Continue (`Continue.continue`)
- ☐ **Course materials** downloaded — *Code → Download ZIP* from https://github.com/warint/quantitative-methods, unzipped **directly on your Desktop** as `quantitative-methods`, and opened with `File → Open Folder…`
- ☐ **Python 3.12.10 specifically** — not "the latest" — from https://www.python.org/downloads/release/python-31210/; on Windows tick *"Add python.exe to PATH"*. Later 3.12 releases are security-only and ship as source code with no installer, and `aider-chat` requires Python `<3.13`, so 3.13 and 3.14 cannot run this course's toolchain
- ☐ `python3.12 --version` prints 3.12.x — on Windows `py -3.12 --version`, and `py --list` shows every Python you have. Build the environment with the version named (`py -3.12 -m venv .venv`, `python3.12 -m venv .venv`), because a bare `py` or `python3` picks the newest one installed
- ☐ `.venv` created and activated — your prompt shows `(.venv)`
- ☐ `pip install -r requirements.txt` completed without red text
- ☐ `python -c "import numpy, pandas, sklearn; print('ok')"` prints `ok`
- ☐ **Ollama** installed from https://ollama.com/download; `ollama --version` works
- ☐ A model pulled: **`qwen2.5-coder:3b`** — the default, and the one that works without a dedicated graphics card. Try `:7b` only after `ollama ps` shows **100% GPU** on the 3b
- ☐ `ollama pull nomic-embed-text` — needed in Session 09
- ☐ **The wifi test:** turn wifi off, run `ollama run qwen2.5-coder:3b "hello"`. It still answers
- ☐ **aider** installed **outside** the course environment, via pipx — `python3.12 -m pip install --user pipx`, `python3.12 -m pipx ensurepath`, new terminal, `pipx install aider-chat` (`py` for `python3` on Windows) — never `pip install aider-chat` into `.venv`. `aider --version` answers in a new terminal
- ☐ `verify_environment.py` passes **with the environment activated** — the prompt shows `(.venv)`. Check 0 reports which interpreter is running; if it fails, every package check below it fails too and means nothing. Bring the output

Git and GitHub are **not** part of Session 01. They are set up before Session 02: `setup-git-and-github.md`.

A.7.4 Connecting Continue to Ollama

Continue's config lives at `~/.continue/config.yaml`:

```yaml
name: qmib
version: 0.0.1
models:
  - name: Local coder
    provider: ollama
    model: qwen2.5-coder:3b
    roles: [chat, edit, apply]
  - name: Local embedder
    provider: ollama
    model: nomic-embed-text
    roles: [embed]
context:
  - provider: file
  - provider: code
  - provider: diff
  - provider: terminal
```

Replace the model name with whichever you pulled. Test it: open a `.py` file, select a few lines, press `Ctrl/Cmd+L`.

A.7.5 Troubleshooting

Symptom	Likely cause	Fix
`ollama: command not found`	Not on PATH yet	Open a new terminal; on Windows restart VS Codium
Dependency resolver errors while installing aider	It was installed into `.venv`, where it competes with the course's packages	Install it with pipx instead — see the checklist above
Dependency conflicts installing aider, or a build error naming a compiler	Your Python is 3.13 or newer; aider requires `<3.13`	Give aider a 3.12 of its own: `pipx install --python python3.12 aider-chat`

Symptom	Likely cause	Fix
`pipx` or `aider` *n'est pas reconnu* / *not recognized*	`pip install --user` writes scripts to a folder Windows does not have on PATH	`py -m pipx …` works regardless; `py -m pipx ensurepath` plus a new terminal fixes PATH itself
A newly installed extension or command does nothing	The window or terminal predates the install	*Developer: Reload Window*; for PATH changes, quit VS Codium and reopen
`Python introuvable` / `Python was not found` (Windows)	The Microsoft Store alias is shadowing Python	Start → **Manage app execution aliases** → turn **off** `python.exe` and `python3.exe`, then open a new terminal. Do not install from the Store.
`python3: command not found` (Windows)	There is no `python3` on Windows	Use `py`. If that is missing too, re-run the python.org installer with **Add python.exe to PATH** ticked
Continue answers nothing	Server not running	`ollama serve` in a separate terminal
Model extremely slow	`ollama ps` shows 100% CPU — no dedicated graphics card	Normal on most laptops with the 3b. A larger model will be slower, not faster
aider "forgets" the file you gave it	The context window is too small — `ollama ps` shows it	Raise it in the **Ollama app's own settings** (Settings → Context length), then quit Ollama from the tray or menu bar and restart. The app overrides `setx` and `launchctl`, so those appear to do nothing. Linux: `systemctl edit ollama.service`
`address already in use`, or *une seule utilisation de chaque adresse de socket*	Ollama is already running; the installer starts it on every platform	Do not run `ollama serve`. Set the variable, then restart Ollama itself

Symptom	Likely cause	Fix
`connection refused` from aider	Nothing listening on 11434	Start Ollama — the app on macOS and Windows, `sudo systemctl start ollama` on Linux
`ModuleNotFoundError` inside VS Codium	Wrong interpreter	`Ctrl/Cmd+Shift+P` → *Python: Select Interpreter* → `.venv`
`pip` fails	Environment not active	Check `(.venv)` is in your prompt
Windows: *l'exécution de scripts est désactivée sur ce système*	PowerShell's execution policy blocks `Activate.ps1`	`Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned` — user-scoped, no admin rights, asked once
Windows: *le répertoire n'est pas vide* when deleting `.venv`	VS Codium holds the interpreter it has selected	Quit VS Codium; delete from a plain PowerShell; `cmd /c "rmdir /s /q .venv"` as a last resort
`deactivate` is not recognised	Activation defines it, so it exists only inside an active environment	You are not in one. Skip it — the prompt shows `(.venv)` when you are
LaTeX shows as raw `$...$`	Preview extension missing	Install *Markdown Preview Enhanced*; open with `Ctrl/Cmd+K V`
Apple Silicon: slow inference	Rosetta Python	Install the arm64 build

A.7.6 Rendering the mathematics

The lecture notes use LaTeX. GitHub renders `$...$` and `$$...$$` natively. In VS Codium, install **Markdown+Math** or **Markdown Preview Enhanced** and open the preview with `Ctrl/Cmd+K V`.

A.7.7 Using the LLM well

The local model is a **sparring partner**, not an oracle.

1. **Ask it to argue against you.** Its most useful output is an objection you had not considered.

2. **Verify every factual claim.** It states plausible falsehoods with complete confidence, especially about library APIs, statistical results and citations.
3. **Record what you checked.** Every practice deliverable requires at least one documented instance where you caught the model being wrong or unverifiable. This is graded.

You are never penalised for using the model. You are penalised for using it uncritically.

Back to pre-session · Session 01 overview

A.8 Setup guide — Git, GitHub, and how you submit work

complete this before Session 02 · budget 45 minutes

> Follows on from the Session 01 setup guide. You should already have VS Codium, a Python environment, and Ollama serving a local model.

A.8.1 `XX` is a placeholder — replace it

Wherever you see **`XX`** in a command on this page, or anywhere else in the course, it means **your group's two-digit number**. Type your own; do not type the letters `XX`.

If you are	you type
Group 1	`group-01`
Group 7	`group-07`
Group 10	`group-10`

So a student in **group 7** running the branch command types:

```
git checkout group-07
```

not `git checkout group-XX`. Same for folder paths: `groups/A2026/group-07/`. If you do not know your group number yet, you get it in Session 01. Do not guess.

A.8.2 How submission works — read this first

There is **one repository for the whole course**, and you already have a copy of it as a ZIP from Session 01. You are about to replace that snapshot with a real clone.

```
github.com/warint/quantitative-methods      ← the course repository
│
├── main              ← the instructor's branch. You never push to it.
├── group-01                ← group 01 works here
├── group-02                ← group 02 works here
└── group-XX                ← your group works here
```

Your group has one branch, named `group-XX`. Everything your group produces is committed on that branch and pushed to it, into your group's own directory:

```
groups/
└── A2026/                      ← the cohort: autumn 2026
    ├── group-03/
    └── group-XX/               ← yours
        ├── session-02/         ← one folder per practice
        ├── session-03/
        └── ...
```

`groups/A2026/` ships **empty**. Your group creates its own folder the first time it submits — `mkdir -p groups/A2026/group-XX/session-02` — and committing a file inside it is what puts your group in the repository. There is no registration step, and git does not track an empty directory.

Your whole term is one directory, which is how you find in week ten what you wrote in week two. The cohort folder above it is what lets next year's students start clean rather than on top of you — and it is why the path is `A2026` and not just `groups/`.

Why one repository and not one per group. Your commit history is one of the four records used to check that all three of you did the work, and `scripts/assess.py contributions` reads it from this repository. Work pushed somewhere else is invisible to it — and invisible work counts as work you did not do.

You do not need to create a repository. You need an account, and the instructor adds you.

A.8.3 Step 1 — Install Git

Check first — you may already have it. In the VS Codium terminal:

```
git --version
```

If that prints a version number, git is installed and you are done with this step.
macOS. Git ships with Apple's Command Line Tools, so most Macs already have it. If the command above is not found, macOS will offer to install the tools itself — accept — or run:

```
xcode-select --install
```

You do **not** need Homebrew for this course.
Windows. Download from https://git-scm.com/downloads and accept every default. Restart VS Codium afterwards so the new `PATH` is picked up.
Linux. `sudo apt install git`, or your distribution's equivalent.
All commands below go in the **VS Codium integrated terminal** (`View → Terminal`, or `Ctrl+\``), never the macOS Terminal app.

A.8.4 Step 2 — Your identity

```
git --version
git config --global user.name  "Your Full Name"
git config --global user.email "your.name@hec.ca"
```

> **Use your real name and your university email.** Commits under `unknown@localhost`, or under an email not on the roster, cannot be attributed to you. The contribution report will show you as having done nothing.

Check what you set, and **report both values to the instructor in Session 02**:

```
git config --global user.name
git config --global user.email
```

A.8.5 Step 3 — A GitHub account

1. Go to https://github.com and choose **Sign up**.
2. Use an email you will still have after you graduate.
3. Choose a username you would put on a CV.
4. Verify the email GitHub sends you.
5. Enable **two-factor authentication** when prompted. GitHub requires it.

Then send your GitHub username to the instructor. You cannot push until you have been added as a collaborator, and that is done by hand from the list of usernames.

A.8.6 Step 4 — Clone the course repository

This replaces the Session 01 ZIP. Delete the unzipped folder afterwards so you do not edit the wrong copy.

```
cd ~/Desktop                          # macOS / Linux
git clone https://github.com/warint/quantitative-methods.git quantitative-me
cd quantitative-methods
```

On Windows PowerShell, use `Set-Location "$HOME\Desktop"` for the first line. Then `File → Open Folder…` and choose the `quantitative-methods` folder on your Desktop.
Rebuild your environment inside the clone:

```
python3.12 -m venv .venv              # Windows: py -3.12 -m venv .venv
source .venv/bin/activate             # Windows: .venv\Scripts\activate
pip install -r requirements.txt
```

A.8.6.1 Authenticating

On your first push GitHub rejects your account password and asks for a token:

1. github.com → avatar → **Settings**
2. **Developer settings → Personal access tokens → Tokens (classic)**
3. **Generate new token**, tick the **`repo`** scope, set an expiry
4. Copy it — **it is shown once** — and paste it when git asks for your *password*

```
git config --global credential.helper store
```

stores it so you are asked once.

A.8.7 Step 5 — Your group branch

One person per group creates it; the others just check it out.

```
# the first person, once   — group 7 shown; use your own number
git checkout -b group-07
git push -u origin group-07

# everyone else, after that
git fetch origin
git checkout group-07
```

> **Never `git push` while on `main`.** `main` is protected, so the push will be refused rather than cause damage — but check where you are before you commit:
>
> ```
> git branch --show-current
> ```

A.8.8 Step 6 — The round trip

```
# group 7 shown throughout — substitute your own number everywhere
mkdir -p groups/A2026/group-07/session-02
echo "# Group 07 — session 02" > groups/A2026/group-07/session-02/NOTES.md

git add groups/A2026/group-07/session-02/
git commit -m "Add group 07 notes for session 02"
git push
```

Refresh the repository page in your browser, switch to your group's branch, and your file is there.
Each of the three of you does this at least once, from your own machine. Not one per group.

A.8.9 If your group of three is already formed — do a trial run now

Groups are confirmed in Session 04, but if the three of you have already agreed to work together, you can rehearse the whole cycle before class. It takes ten minutes and it removes the one thing that reliably eats practice time: discovering in the room that two of you cannot push.
One person, once. Create the branch and push it:

```
git checkout -b group-07              # your number, not 07
git push -u origin group-07
```

The other two, after that. Fetch it and switch to it:

```
git fetch origin
git checkout group-07
```

Then each of you, one at a time, adds a line with your own name and pushes:

```
git pull                              # always pull before you start
echo "- Ana tested the push, 2 September" >> \
  groups/A2026/group-07/session-02/NOTES.md

git add groups/A2026/group-07/session-02/NOTES.md
git commit -m "Ana: trial push"
git push
```

Wait for each person to finish before the next starts. When all three are done:

```
git pull
git log --oneline -5
```

You should see three commits, with three different author names. That is exactly what the participation record looks like, and confirming it now is worth more than any amount of reading about git.

If someone's push is rejected, they almost certainly need to `git pull` first — someone else pushed in between. That is normal, not a fault; see Working together below.

A.8.10 Working together

A.8.10.1 Pull before you start

```
git pull
```

Most conflicts come from someone editing a stale copy for two hours.

A.8.10.2 Getting the instructor's corrections

The course material lives on `main` and gets fixed during the term. To bring those fixes into your branch:

```
git fetch origin
git merge origin/main
```

Do this at the start of each session.

A.8.10.3 Reading a conflict

```
<<<<<<< HEAD
alpha = 0.05
=======
alpha = 0.01
>>>>>>> origin/main
```

Delete the markers, keep the version you agree on, then `git add` and `git commit`. A conflict is git refusing to guess which of you was right.

A.8.10.4 Commit messages

`git commit -m "update"` tells nobody anything. Write for whoever reads it in six weeks — usually you:

```
Fix leakage: move scaler inside the CV pipeline
```

A.8.10.5 Credit your pair

```
git commit -m "Add stability bootstrap

Co-authored-by: Partner Name <partner.name@hec.ca>"
```

The contribution report counts co-authors. One person typing does not mean one person working.

A.8.10.6 aider and your history

aider commits automatically after each change it applies, and this course counts commits as evidence of who did the work.

```
git log --oneline
aider --model ollama_chat/qwen2.5-coder:3b --no-auto-commits
```

What the agent writes is attributed to **you**. Read the diff before accepting: you are answerable for every line on your branch.

A.8.11 Everyone pushes

There are **no assigned roles**. All three of you work on the practice together — and all three commit and push from your own machine, every week.

```
git add -A && git commit -m "..." && git push
```

`python scripts/assess.py contributions` prints a per-member, per-week grid built from the history. A week in which you pushed nothing is visible to everyone, including you.
The presenter for the two-minute report is **drawn at random** when your group is called, which is why all three of you must understand the whole analysis.

A.8.12 Troubleshooting

Symptom	Fix
`git: command not found`	Reopen the terminal; on Windows restart VS Codium
Password rejected on push	Use a personal access token, not your password

Symptom	Fix
`remote: Permission denied`	You have not been added as a collaborator yet — send your username
`protected branch` on push	You are on `main`. Check out your group branch
`rejected — non-fast-forward`	`git pull` first, resolve, then push
Committed under the wrong email	Fix `git config`, tell the instructor; past commits stay wrong

---

A.8.13 Before Session 02

- ☐ Git installed; `user.name` and `user.email` set to your real details
- ☐ GitHub account with two-factor authentication
- ☐ **GitHub username sent to the instructor**
- ☐ Course repository cloned; the Session 01 ZIP folder deleted
- ☐ Your `group-NN` branch checked out — your own number, not `XX`
- ☐ One commit pushed **by you personally** to your group branch, visible on github.com
- ☐ Your exact `user.name` and `user.email` written down to report

A.8.14 For the instructor

Once the usernames are in:

1. **Settings → Collaborators → Add people** — add every student with **Write** access.
2. **Settings → Branches → Add branch protection rule** for `main`: require a pull request, and tick *"Do not allow bypassing"* off for yourself. Students then cannot push to `main`.
3. Group branches need no protection — a group breaking its own branch is recoverable.
4. To review a group's work:

```
git fetch origin
git checkout group-07
python scripts/assess.py contributions
```

`contributions` reads the history of whatever is checked out, so run it after fetching all branches:

```
git fetch --all
python scripts/assess.py contributions
```

Session 02 overview · Session 01 setup guide

Appendix B

The data API

Every dataset in this course loads with one call. There is no path to get right, no download step in the practice, and no dependence on the room's wifi.

```
import qmib

df = qmib.load("core")        # the course spine
qmib.catalog()                # everything available
```

> `load()` resolves in three steps and stops at the first that works: a **parquet cache** in `data/`, then the **committed spine** in `data/spine/`, then the **published copy** on GitHub Pages. The first call downloads; every call after it reads the cache.

> **Why it matters for the practice.** Run `qmib.load()` once at home and the ninety minutes in class work offline. A room of thirty people downloading the same file at 15:35 is the single most reliable way to lose the first twenty minutes of a session.

B.1 Outside a clone

In Colab, or anywhere without the repository checked out, point the module at the published copy:

```
import qmib
qmib.REMOTE = "https://warint.github.io/quantitative-methods/data"
df = qmib.load("core")
```

B.2 What is where

- Columns, units and traps, per group: data dictionaries
- Provenance and flags: `PROVENANCE.md`
- The module itself: `qmib.py`

> Raw data files are git-ignored on purpose. Never commit one — `qmib` fetches and caches instead. The synthetic spine is the deliberate exception: small, licence-free, and committed so the practice runs with no network at all.

Appendix C

Replication packages

Sessions 02–06 and 08–11 (and 13, after the course) pair one **academic article** with the **data behind it** — a Harvard Dataverse replication package where the authors deposited one, and otherwise the public database the paper itself is built on, fetched by `qmib.load()`. Before class, read and annotate the article, download the package, and identify the table cell, coefficient, or figure named in the pre-session deck. ISLR is optional background throughout.
During the 90-minute practice, each group:

1. states the article's question, sample, and target result;
2. reproduces one published result from the package;
3. compares it with the session method or a hard benchmark;
4. breaks one assumption or changes one defensible specification; and
5. presents one slide: **article result · our replication · the catch**.

Session 01 is the exception: the pre-session installs the workstation and assigns *Europe 2031*; the practice is a conversation about evidence, scenario, prediction, and what a local language model actually does.

C.1 Article–data map

Session	Academic article	Harvard Dataverse package	Practice target
02 · Exploratory data analysis	Fraiberger et al. (2021), *Media sentiment and international asset prices* · article	10.7910/DVN/QNKHFF	Profile the sentiment index — centre, spread, shape, thresholds tested — then fit the paper's simplest return regression and state the slope in units.

Session	Academic article	Harvard Dataverse package	Practice target
03 · Regression diagnostics	Amsili, van Es & Schindelbeck (2024), *Pedotransfer Functions for Field Capacity, Permanent Wilting Point, and Available Water Capacity* · article	10.7910/DVN/U5DARP	Reproduce one predictive regression, then diagnose it: residuals versus fitted, scale–location, leverage and Cook's distance. Which conclusions survive?
04 · Logistic regression	Saganowski et al. (2019), *Analysis of group evolution prediction in complex networks* · article	10.7910/DVN/ONOHR7	Reproduce one classification result as a logistic model; interpret the odds ratios in words and test one nested comparison.
05 · Regularisation	Blonigen & Piger (2014), *Determinants of foreign direct investment* · article · free preprint	No package published — rebuilt by `scripts/build_fdi_dataset.py` from OECD, CEPII, World Bank and Freedom House; load with `qmib.load("fdi")`	The paper runs Bayesian model averaging over 56 candidate determinants of bilateral FDI and finds about sixteen survive. Refit the same 56 for 2019 with lasso, ridge and elastic net, and say which of its conclusions still hold.

Session	Academic article	Harvard Dataverse package	Practice target
06 · Panel data and interactions	Topalova & Khandelwal (2011), *Trade Liberalization and Firm Productivity* · article	10.7910/DVN/8WEXKD	Reproduce one firm-productivity regression; refit it as a panel with fixed and then random effects, and say which you would report.
07 · Four papers, one toolkit	The data of four articles of sessions 02, 04, 05 and 06	One CSV per paper on the course site (`data/session-07/`), read by the scripts in a folder of your own	Three hours in autonomy: plug each paper's data into the course's models — OLS, the logits, ridge, the lasso, the elastic net — with exactly the code seen in class; step-by-step guides in `07-…/02-practice/replica`
08 · KNN and bias–variance	Bluwstein, Buckmann, Joseph, Kapadia & Şimşek (2023), *Credit growth, the yield curve and financial crisis prediction* · article · free working paper	Jordà–Schularick–Taylor Macrohistory Database, public; load with `qmib.load("jst")`	The paper reports that flexible machine learning beats logistic regression at predicting financial crises. Fit KNN against logistic regression on the same 18-country panel, choose k by cross-validation, and say whether flexibility earned its keep on **your** feature set.

Session	Academic article	Harvard Dataverse package	Practice target
09 · Structural equation modelling	Bennani & Romelli (2024), *Exploring the informativeness and drivers of tone during committee meetings* · article	10.7910/DVN/TZENTH	The paper builds a measure of an unobservable — tone — from observed indicators. Reproduce one tone result, then specify it as a measurement model and report the fit indices.
10 · Causal inference I	Atkin, Khandelwal & Osman (2017), *Exporting and Firm Performance: Evidence from a Randomized Experiment* · article	10.7910/DVN/QOGNEV	Egyptian rug producers were randomly given export orders, so the counterfactual is known. Reproduce the experimental estimate; then throw the randomisation away, re-estimate by propensity-score matching on observables, and report how far matching lands from the truth.

Session	Academic article	Harvard Dataverse package	Practice target
11 · Causal inference II	Cavallo, Gopinath, Neiman & Tang (2021), *Tariff Passthrough at the Border and at the Store: Evidence from US Trade Policy* · article	10.7910/DVN/JV7FCH	The 2018–19 US tariffs hit some goods and not others — a difference-in-differences design with a real policy as the treatment. Reproduce one passthrough estimate, show the parallel-trends evidence, and compare conventional with design-aware uncertainty.
13 · One more thing — PCA and factor analysis *(self-study)*	Gygli, Haelg, Potrafke & Sturm (2019), *The KOF Globalisation Index — revisited* · article	No deposit — the index itself is public; load with `qmib.load("kof")`	The index asserts that globalisation has three dimensions — economic, social, political — built from six sub-dimensions. Run PCA on the sub-indices and ask how many the data actually supports. Defend the number you retain, and interpret the loadings.

C.2 Downloading a package

Use the DOI link in the pre-session deck and download once, before class. Extract the files to:

```
Desktop/quantitative-methods/NN-session-name/data/replication/
```

Keep the authors' folder structure and README. Large downloads belong in the git-ignored `data/` folder, never inside a practice script. The analysis should run locally after the download.

Session 05 is the exception: its authors published no replication package, so the course rebuilt their design from the sources the paper names. It needs no download — `qmib.load("fdi")` reads a file committed to the repository. `scripts/build_fdi_determinants.py` documents every source and every substitution.

If command-line download is useful, Dataverse also exposes a dataset endpoint:

```
curl -L "https://dataverse.harvard.edu/api/access/dataset/:persistentId/?persis
  -o data/topalova.zip
unzip data/topalova.zip -d data/topalova/
```

Cite the article and the Dataverse package separately. They are distinct research contributions with distinct DOIs.

Appendix D

Further reading

Optional. Nothing here is required, nothing here is examinable, and the course is complete without any of it. This is the list for the reader who finishes a chapter and wants the longer treatment — or who meets, in their own work, one of the problems the conclusion admits this book does not cover.

Where a book is legally free online, that is the link given. Where it is not, the publisher's page is, so you can find it in a library rather than guess at an edition.

D.1 The two nearest neighbours of this book

An Introduction to Statistical Learning — James, Witten, Hastie, Tibshirani. statlearning.com · free PDF

The closest thing to a companion volume. Covers chapters 2 through 8 of this book at a similar level with more examples and less econometrics, and its treatment of the bias–variance trade-off and of regularisation is the standard one. Editions exist for both R and Python. If you read one book on this list, read this.

Data Science for International Business with R — Warin. warin.ca/ds4ibr

The R volume of this course. Exploratory analysis, regression and panel data, logistic models, unsupervised learning and structural equation modelling, in the other language of the field. Reading the two side by side is the fastest way to see which difficulties belong to the method and which belong to the tooling.

D.2 Going deeper on the methods

The Elements of Statistical Learning — Hastie, Tibshirani, Friedman. hastie.su.domains/Elem · free PDF

The graduate version of ISLR, and the reference for chapters 5, 7 and 8. Harder, complete, and the place to look when you want the derivation rather than the result. Its chapter on model assessment is the fuller treatment of what chapter 8 demonstrates by simulation.

Categorical Data Analysis — Agresti. doi.org/10.1002/0471249688

Chapter 4 in book form: logistic regression, ordinal and multinomial models, and the exact-inference machinery for the small samples where the asymptotics this book relies on stop being safe.

Econometric Analysis of Panel Data — Baltagi. link.springer.com

Chapter 6 at length. Fixed and random effects, the Hausman and Mundlak formulations, dynamic panels and the Nickell bias, and the estimators that address it.

Python Data Science Handbook — VanderPlas. jakevdp.github.io · free online

Not a statistics book. It is the reference for the tools this course uses — NumPy,

pandas, matplotlib, scikit-learn — and the right place to go when the obstacle is the library rather than the method.

D.3 Causal inference, where this book stops

Chapters 10 and 11 are an introduction to a field with an unusually good literature, most of it free.

Causal Inference: The Mixtape — Cunningham. mixtape.scunning.com · free online
Potential outcomes, matching, difference-in-differences, regression discontinuity and instrumental variables, with code and an unusually direct voice. The natural next book after chapter 11.

The Effect — Huntington-Klein. theeffectbook.net · free online
Starts further back than the Mixtape and spends longer on what a research design *is* before reaching estimators. The chapters on causal diagrams are the clearest short treatment of the good-and-bad-controls problem chapter 10 raises.

Mostly Harmless Econometrics — Angrist and Pischke. press.princeton.edu
The book that made design-based inference the standard in applied economics. Terse, opinionated, and assumes more econometrics than this course does. **Mastering 'Metrics** by the same authors (Princeton) is the gentler version, and a better first read.

Causal Inference for Statistics, Social, and Biomedical Sciences — Imbens and Rubin. cambridge.org
The formal treatment of the potential-outcomes framework chapter 10 introduces, by the people who built it. Where to go when you need the assumptions stated precisely rather than described.

D.4 The gaps the conclusion admits

Statistical Rethinking — McElreath. xcelab.net/rm
The Bayesian course this book is not. Rebuilds inference from probability rather than from sampling distributions, and its treatment of multilevel models is the one to read after chapter 6. Lectures are free on the author's site.

Regression and Other Stories — Gelman, Hill and Vehtari. avehtari.github.io/ROS-Examples · free PDF
Sits between this book and McElreath's. Applied, Bayesian by default, and better than anything else on the list at the part this course keeps insisting on: what a model does and does not license you to say.

A. Smith, Jeffrey, and Petra E. Todd. 2005. "Does Matching Overcome LaLonde's Critique of Nonexperimental Estimators?" *Journal of Econometrics* 125 (1-2): 305–53. https://doi.org/10.1016/j.jeconom.2004.04.011.

Abadie, Alberto, Susan Athey, Guido W Imbens, and Jeffrey M Wooldridge. 2022. "When Should You Adjust Standard Errors for Clustering?" *The Quarterly Journal of Economics* 138 (1): 1–35. https://doi.org/10.1093/qje/qjac038.

Akaike, H. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6): 716–23. https://doi.org/10.1109/TAC.1974.1100705.

Albert, A., and J. A. Anderson. 1984. "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 71 (1): 1–10. https://doi.org/10.1093/biomet/71.1.1.

Anderson, James E, and Eric van Wincoop. 2003. "Gravity with Gravitas: A Solution to the Border Puzzle." *American Economic Review* 93 (1): 170–92. https://doi.org/10.1257/000282803321455214.

Angrist, Joshua D, and Jörn-Steffen Pischke. 2010. "The Credibility Revolution in Empirical Economics: How Better Research Design Is Taking the Con Out of Econometrics." *Journal of Economic Perspectives* 24 (2): 3–30. https://doi.org/10.1257/jep.24.2.3.

Anscombe, F. J. 1973. "Graphs in Statistical Analysis." *The American Statistician* 27 (1): 17–21. https://doi.org/10.1080/00031305.1973.10478966.

Anscombe, F. J., and John W. Tukey. 1963. "The Examination and Analysis of Residuals." *Technometrics* 5 (2): 141–60. https://doi.org/10.1080/00401706.1963.10490071.

Baker, Scott R., Nicholas Bloom, and Steven J. Davis. 2016. "Measuring Economic Policy Uncertainty*." *The Quarterly Journal of Economics* 131 (4): 1593–1636. https://doi.org/10.1093/qje/qjw024.

Baldwin, Richard, and Rebecca Freeman. 2022. "Risks and Global Supply Chains: What We Know and What We Need to Know." *Annual Review of Economics* 14 (1): 153–80. https://doi.org/10.1146/annurev-economics-051420-113737.

Belsley, David A., Edwin Kuh, and Roy E. Welsch. 1980. "Regression Diagnostics." Wiley. https://doi.org/10.1002/0471725153.

Bentler, P. M. 1990. "Comparative Fit Indexes in Structural Models." *Psychological Bulletin* 107 (2): 238–46. https://doi.org/10.1037/0033-2909.107.2.238.

Berk, Richard, Lawrence Brown, Andreas Buja, Kai Zhang, and Linda Zhao. 2013. "Valid Post-Selection Inference." *The Annals of Statistics* 41 (2): 802–37. https://doi.org/10.1214/12-AOS1077.

Berkson, Joseph. 1944. "Application of the Logistic Function to Bio-Assay." *Journal of the American Statistical Association* 39 (227): 357–65. https://doi.org/10.1080/01621459.1944.10500699.

Bertrand, M., E. Duflo, and S. Mullainathan. 2004. "How Much Should We Trust Differences-In-Differences Estimates?" *The Quarterly Journal of Economics* 119 (1): 249–75. https://doi.org/10.1162/003355304772839588.

Beyer, Kevin, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. "When Is 'Nearest Neighbor' Meaningful?" In *Lecture Notes in Computer Science*, 217–35. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49257-7_15.

Bollen, Kenneth A. 1989. "Structural Equations with Latent Variables." Wiley. https://doi.org/10.1002/9781118619179.

Box, G. E. P. 1953. "NON-NORMALITY AND TESTS ON VARIANCES."

Biometrika 40 (3-4): 318–35. https://doi.org/10.1093/biomet/40.3-4.318.

Box, G. E. P., and D. R. Cox. 1964. "An Analysis of Transformations." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 26 (2): 211–43. https://doi.org/10.1111/j.2517-6161.1964.tb00553.x.

Box, George E. P. 1976. "Science and Statistics." *Journal of the American Statistical Association* 71 (356): 791–99. https://doi.org/10.1080/01621459.1976.10480949.

Brant, Rollin. 1990. "Assessing Proportionality in the Proportional Odds Model for Ordinal Logistic Regression." *Biometrics* 46 (4): 1171. https://doi.org/10.2307/2532457.

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statistical Science* 16 (3): 199–231. https://doi.org/10.1214/ss/1009213726.

Breusch, T. S., and A. R. Pagan. 1979. "A Simple Test for Heteroscedasticity and Random Coefficient Variation." *Econometrica* 47 (5): 1287. https://doi.org/10.2307/1911963.

Brier, GLENN W. 1950. "VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY." *Monthly Weather Review* 78 (1): 1–3. https://doi.org/10.1175/1520-0493(1950)078%3C0001:VOFEIT%3E2.0.CO;2.

Browne, Michael W., and Robert Cudeck. 1992. "Alternative Ways of Assessing Model Fit." *Sociological Methods & Research* 21 (2): 230–58. https://doi.org/10.1177/0049124192021002005.

Burnham, Kenneth P., and David R. Anderson. 2004. "Multimodel Inference." *Sociological Methods & Research* 33 (2): 261–304. https://doi.org/10.1177/0049124104268644.

Callaway, Brantly, and Pedro H. C. Sant'Anna. 2021. "Difference-in-Differences with Multiple Time Periods." *Journal of Econometrics* 225 (2): 200–230. https://doi.org/10.1016/j.jeconom.2020.12.001.

Cattell, Raymond B. 1966. "The Scree Test For The Number Of Factors." *Multivariate Behavioral Research* 1 (2): 245–76. https://doi.org/10.1207/s15327906mbr0102_10.

Chaisemartin, Clément de, and Xavier D'Haultfœuille. 2020. "Two-Way Fixed Effects Estimators with Heterogeneous Treatment Effects." *American Economic Review* 110 (9): 2964–96. https://doi.org/10.1257/aer.20181169.

Cheung, Gordon W., and Roger B. Rensvold. 2002. "Evaluating Goodness-of-Fit Indexes for Testing Measurement Invariance." *Structural Equation Modeling: A Multidisciplinary Journal* 9 (2): 233–55. https://doi.org/10.1207/S15328007SEM0902_5.

Cinelli, Carlos, Andrew Forney, and Judea Pearl. 2022. "A Crash Course in Good and Bad Controls." *Sociological Methods & Research* 53 (3): 1071–1104. https://doi.org/10.1177/00491241221099552.

Cleveland, William S. 1979. "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of the American Statistical Association* 74 (368): 829–36. https://doi.org/10.1080/01621459.1979.10481038.

Cleveland, William S., and Robert McGill. 1984. "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods." *Journal of the American Statistical Association* 79 (387): 531–54. https://doi.org/10.1080/01621459.1984.10478080.

Colin Cameron, A., and Douglas L. Miller. 2015. "A Practitioner's Guide to Cluster-Robust Inference." *Journal of Human Resources* 50 (2): 317–72. https://doi.org/10.3368/jhr.50.2.317.

Cook, R. Dennis. 1977. "Detection of Influential Observation in Linear Regression." *Technometrics* 19 (1): 15–18. https://doi.org/10.1080/00401706.1977.10489493.

Cover, T., and P. Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13 (1): 21–27. https://doi.org/10.1109/TIT.1967.1053964.

Cox, D. R. 1958. "The Regression Analysis of Binary Sequences." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 20 (2): 215–32. https://doi.org/10.1111/j.2517-6161.1958.tb00292.x.

Dehejia, Rajeev H., and Sadek Wahba. 1999. "Causal Effects in Nonexperimental Studies: Reevaluating the Evaluation of Training Programs." *Journal of the American Statistical Association* 94 (448): 1053–62. https://doi.org/10.1080/01621459.1999.10473858.

Durbin, J., and G. S. Watson. 1950. "Testing for Serial Correlation in Least Squares Regression: i." *Biometrika* 37 (3/4): 409. https://doi.org/10.2307/2332391.

Efron, Bradley, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. "Least Angle Regression." *The Annals of Statistics* 32 (2): 407–99. https://doi.org/10.1214/009053604000000067.

Einav, Liran, and Jonathan Levin. 2014. "Economics in the Age of Big Data." *Science* 346 (6210). https://doi.org/10.1126/science.1243089.

Elwert, Felix, and Christopher Winship. 2014. "Endogenous Selection Bias: The Problem of Conditioning on a Collider Variable." *Annual Review of Sociology* 40 (1): 31–53. https://doi.org/10.1146/annurev-soc-071913-043455.

Fabrigar, Leandre R., Duane T. Wegener, Robert C. MacCallum, and Erin J. Strahan. 1999. "Evaluating the Use of Exploratory Factor Analysis in Psychological Research." *Psychological Methods* 4 (3): 272–99. https://doi.org/10.1037/1082-989X.4.3.272.

Firth, DAVID. 1993. "Bias Reduction of Maximum Likelihood Estimates." *Biometrika* 80 (1): 27–38. https://doi.org/10.1093/biomet/80.1.27.

Fix, Evelyn, and J. L. Hodges. 1989. "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties." *International Statistical Review / Revue Internationale de Statistique* 57 (3): 238. https://doi.org/10.2307/1403797.

Frank, lldiko E., and Jerome H. Friedman. 1993. "A Statistical View of Some Chemometrics Regression Tools." *Technometrics* 35 (2): 109–35. https://doi.org/10.1080/00401706.1993.10485033.

Freedman, David A. 1983. "A Note on Screening Regression Equations." *The American Statistician* 37 (2): 152–55. https://doi.org/10.1080/00031305.1983.

10482729.
Frisch, Ragnar, and Frederick V. Waugh. 1933. "Partial Time Regressions as Compared with Individual Trends." *Econometrica* 1 (4): 387. https://doi.org/10.2307/1907330.
Galton, Francis. 1886. "Regression Towards Mediocrity in Hereditary Stature." *The Journal of the Anthropological Institute of Great Britain and Ireland* 15: 246. https://doi.org/10.2307/2841583.
Gelman, Andrew, and Eric Loken. 2014. "The Statistical Crisis in Science." *American Scientist* 102 (6): 460. https://doi.org/10.1511/2014.111.460.
Geman, Stuart, Elie Bienenstock, and René Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4 (1): 1–58. https://doi.org/10.1162/neco.1992.4.1.1.
Goodman-Bacon, Andrew. 2021. "Difference-in-Differences with Variation in Treatment Timing." *Journal of Econometrics* 225 (2): 254–77. https://doi.org/10.1016/j.jeconom.2021.03.014.
Hanley, J A, and B J McNeil. 1982. "The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve." *Radiology* 143 (1): 29–36. https://doi.org/10.1148/radiology.143.1.7063747.
Hausman, J. A. 1978. "Specification Tests in Econometrics." *Econometrica* 46 (6): 1251. https://doi.org/10.2307/1913827.
Healy, Kieran, and James Moody. 2014. "Data Visualization in Sociology." *Annual Review of Sociology* 40 (1): 105–28. https://doi.org/10.1146/annurev-soc-071312-145551.
Hidalgo, César A., and Ricardo Hausmann. 2009. "The Building Blocks of Economic Complexity." *Proceedings of the National Academy of Sciences* 106 (26): 10570–75. https://doi.org/10.1073/pnas.0900943106.
Hoaglin, David C., and Roy E. Welsch. 1978. "The Hat Matrix in Regression and ANOVA." *The American Statistician* 32 (1): 17–22. https://doi.org/10.1080/00031305.1978.10479237.
Hoerl, Arthur E., and Robert W. Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55–67. https://doi.org/10.1080/00401706.1970.10488634.
Holland, Paul W. 1986. "Statistics and Causal Inference." *Journal of the American Statistical Association* 81 (396): 945–60. https://doi.org/10.1080/01621459.1986.10478354.
Horn, John L. 1965. "A Rationale and Test for the Number of Factors in Factor Analysis." *Psychometrika* 30 (2): 179–85. https://doi.org/10.1007/BF02289447.
Hotelling, H. 1933. "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology* 24 (6): 417–41. https://doi.org/10.1037/h0071325.
Hu, Li□tze, and Peter M. Bentler. 1999. "Cutoff Criteria for Fit Indexes in Covariance Structure Analysis: Conventional Criteria Versus New Alternatives." *Structural Equation Modeling: A Multidisciplinary Journal* 6 (1): 1–55. https:

//doi.org/10.1080/10705519909540118.

Imbens, Guido W. 2004. "Nonparametric Estimation of Average Treatment Effects Under Exogeneity: A Review." *Review of Economics and Statistics* 86 (1): 4–29. https://doi.org/10.1162/003465304323023651.

Jöreskog, K. G. 1971. "Statistical Analysis of Sets of Congeneric Tests." *Psychometrika* 36 (2): 109–33. https://doi.org/10.1007/BF02291393.

Kaiser, Henry F. 1958. "The Varimax Criterion for Analytic Rotation in Factor Analysis." *Psychometrika* 23 (3): 187–200. https://doi.org/10.1007/BF02289233.

King, Gary, and Richard Nielsen. 2019. "Why Propensity Scores Should Not Be Used for Matching." *Political Analysis* 27 (4): 435–54. https://doi.org/10.1017/pan.2019.11.

Lockhart, Richard, Jonathan Taylor, Ryan J. Tibshirani, and Robert Tibshirani. 2014. "A Significance Test for the Lasso." *The Annals of Statistics* 42 (2): 413–68. https://doi.org/10.1214/13-AOS1175.

Long, J. Scott, and Laurie H. Ervin. 2000. "Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model." *The American Statistician* 54 (3): 217–24. https://doi.org/10.1080/00031305.2000.10474549.

Lovell, Michael C. 1963. "Seasonal Adjustment of Economic Time Series and Multiple Regression Analysis." *Journal of the American Statistical Association* 58 (304): 993–1010. https://doi.org/10.1080/01621459.1963.10480682.

MacCallum, Robert C., and James T. Austin. 2000. "Applications of Structural Equation Modeling in Psychological Research." *Annual Review of Psychology* 51 (1): 201–26. https://doi.org/10.1146/annurev.psych.51.1.201.

MacKinnon, James G, and Halbert White. 1985. "Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties." *Journal of Econometrics* 29 (3): 305–25. https://doi.org/10.1016/0304-4076(85)90158-7.

Matejka, Justin, and George Fitzmaurice. 2017. "Same Stats, Different Graphs." In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 1290–94. ACM. https://doi.org/10.1145/3025453.3025912.

Meinshausen, Nicolai, and Peter Bühlmann. 2010. "Stability Selection." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 72 (4): 417–73. https://doi.org/10.1111/j.1467-9868.2010.00740.x.

Melitz, Marc J. 2003. "The Impact of Trade on Intra-Industry Reallocations and Aggregate Industry Productivity." *Econometrica* 71 (6): 1695–725. https://doi.org/10.1111/1468-0262.00467.

Mood, C. 2009. "Logistic Regression: Why We Cannot Do What We Think We Can Do, and What We Can Do About It." *European Sociological Review* 26 (1): 67–82. https://doi.org/10.1093/esr/jcp006.

Moulton, Brent R. 1990. "An Illustration of a Pitfall in Estimating the Effects of Aggregate Variables on Micro Units." *The Review of Economics and Statistics* 72 (2): 334. https://doi.org/10.2307/2109724.

Mundlak, Yair. 1978. "On the Pooling of Time Series and Cross Section Data."

Econometrica 46 (1): 69. https://doi.org/10.2307/1913646.
Nickell, Stephen. 1981. "Biases in Dynamic Models with Fixed Effects." *Econometrica* 49 (6): 1417. https://doi.org/10.2307/1911408.
Pearson, Karl. 1901. "LIII. <I>On Lines and Planes of Closest Fit to Systems of Points in Space." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (11): 559–72. https://doi.org/10.1080/14786440109462720.
Rambachan, Ashesh, and Jonathan Roth. 2023. "A More Credible Approach to Parallel Trends." *Review of Economic Studies* 90 (5): 2555–91. https://doi.org/10.1093/restud/rdad018.
Ramsey, J. B. 1969. "Tests for Specification Errors in Classical Linear Least-Squares Regression Analysis." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 31 (2): 350–71. https://doi.org/10.1111/j.2517-6161.1969.tb00796.x.
Rosenbaum, PAUL R., and DONALD B. Rubin. 1983. "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika* 70 (1): 41–55. https://doi.org/10.1093/biomet/70.1.41.
Rosseel, Yves. 2012. "<B>lavaan : An r Package for Structural Equation Modeling." *Journal of Statistical Software* 48 (2). https://doi.org/10.18637/jss.v048.i02.
Rubin, Donald B. 1974. "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies." *Journal of Educational Psychology* 66 (5): 688–701. https://doi.org/10.1037/h0037350.
Rubin, DONALD B. 1976. "Inference and Missing Data." *Biometrika* 63 (3): 581–92. https://doi.org/10.1093/biomet/63.3.581.
Schenker, Nathaniel, and Jane F Gentleman. 2001. "On Judging the Significance of Differences by Examining the Overlap Between Confidence Intervals." *The American Statistician* 55 (3): 182–86. https://doi.org/10.1198/000313001317097960.
Schwarz, Gideon. 1978. "Estimating the Dimension of a Model." *The Annals of Statistics* 6 (2): 461–64. https://doi.org/10.1214/aos/1176344136.
Shapiro, S. S., and M. B. Wilk. 1965. "An Analysis of Variance Test for Normality (Complete Samples)." *Biometrika* 52 (3-4): 591–611. https://doi.org/10.1093/biomet/52.3-4.591.
Simonsohn, Uri, Joseph P. Simmons, and Leif D. Nelson. 2020. "Specification Curve Analysis." *Nature Human Behaviour* 4 (11): 1208–14. https://doi.org/10.1038/s41562-020-0912-z.
Simpson, E. H. 1951. "The Interpretation of Interaction in Contingency Tables." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 13 (2): 238–41. https://doi.org/10.1111/j.2517-6161.1951.tb00088.x.
Spearman, C. 1904. ""General Intelligence," Objectively Determined and Measured." *The American Journal of Psychology* 15 (2): 201. https://doi.org/10.2307/1412107.
Stone, Charles J. 1977. "Consistent Nonparametric Regression." *The Annals of*

Statistics 5 (4): 595–620. https://doi.org/10.1214/aos/1176343886.

Stone, M. 1974. "Cross-Validatory Choice and Assessment of Statistical Predictions." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 36 (2): 111–33. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x.

Stuart, Elizabeth A. 2010. "Matching Methods for Causal Inference: A Review and a Look Forward." *Statistical Science* 25 (1): 1–21. https://doi.org/10.1214/09-STS313.

Sun, Liyang, and Sarah Abraham. 2021. "Estimating Dynamic Treatment Effects in Event Studies with Heterogeneous Treatment Effects." *Journal of Econometrics* 225 (2): 175–99. https://doi.org/10.1016/j.jeconom.2020.09.006.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58 (1): 267–88. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Tukey, John W. 1962. "The Future of Data Analysis." *The Annals of Mathematical Statistics* 33 (1): 1–67. https://doi.org/10.1214/aoms/1177704711.

Varian, Hal R. 2014. "Big Data: New Tricks for Econometrics." *Journal of Economic Perspectives* 28 (2): 3–28. https://doi.org/10.1257/jep.28.2.3.

Warin, Thierry. 2022. "Supply Chains Under Pressure: How Can Data Science Help?" CIRANO. https://doi.org/10.54932/njyx4623.

———. 2023. "Vers Une Économie de Données : Réflexions Pour Hausser La Productivité de l'économie Québécoise à l'heure de La Révolution Des Données." CIRANO. https://doi.org/10.54932/csxq4709.

———. 2024. "Access Statistics Canada's Open Economic Data for Statistics and Data Science Courses." *Technology Innovations in Statistics Education* 15 (1). https://doi.org/10.5070/t5.1868.

———. 2025a. "Gravity Models Versus Comparative Advantage: It Is Not Enough for Trade to Be Free; Trade Should Also Be Fit." CIRANO. https://doi.org/10.54932/xqay2333.

———. 2025b. "Historical and Contemporary Evolution of International Trade: From Mercantilism to the Platform Economy." CIRANO. https://doi.org/10.54932/iqen6866.

———. 2025c. "Not Efficient, Not Optimal: The Biases That Built Global Trade and the Data Tools That Could Fix It." CIRANO. https://doi.org/10.54932/zihs7852.

———. 2026a. "Countries Do Not Trade, Firms Do." World Scientific Publishing Co. https://doi.org/10.1142/q0619.

———. 2026b. "Middle Powers in the Platform Economy: Digital Sovereignty After the Data Pivot." *Data & Policy* 8. https://doi.org/10.1017/dap.2026.10085.

White, Halbert. 1980. "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48 (4): 817. https://doi.org/10.2307/1912934.

Wing, Coady, Kosali Simon, and Ricardo A. Bello-Gomez. 2018. "Designing Difference in Difference Studies: Best Practices for Public Health Policy Re-

search." *Annual Review of Public Health* 39 (1): 453–69. https://doi.org/10.1146/annurev-publhealth-040617-013507.

Wright, Sewall. 1934. "The Method of Path Coefficients." *The Annals of Mathematical Statistics* 5 (3): 161–215. https://doi.org/10.1214/aoms/1177732676.

Yule, G. UNDY. 1903. "NOTES ON THE THEORY OF ASSOCIATION OF ATTRIBUTES IN STATISTICS." *Biometrika* 2 (2): 121–34. https://doi.org/10.1093/biomet/2.2.121.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection Via the Elastic Net." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67 (2): 301–20. https://doi.org/10.1111/j.1467-9868.2005.00503.x.