A MODEL-SELECTION HANDBOOK FOR APPLIED ECONOMIC RESEARCH

From Data to Decision

Time series, panels, causal designs, machine learning,
Bayesian and regime-switching models, and DSGE models:
from the test that selects the method to the paper that reports it

Tuhin G. M. Al Mamun
Ph.D. in Economics

Manuscript, 2026

Copyright © 2026 Tuhin G. M. Al Mamun. All rights reserved.

The datasets used in this book are published datasets, listed with their sources in Appendix A. The code printed in each chapter is the code that produced the output and figures shown beside it.

From Data to Decision is a manuscript. Statements of fact, citations and numerical results should be checked against the original sources before the text is cited or submitted for publication.

Preface

An empirical paper in economics goes wrong in a small number of ways, and most of them occur before the first coefficient is read. The variables are non-stationary and no one checked. The panel is short and the model is dynamic. The treatment began in different years in different places, and the estimator assumes it did not. The prior was never written down, so it cannot be defended. None of these is a deep technical failure. Each is a failure to establish, before choosing a method, what the data and the question allow.

This book is written for the researcher who wants to avoid those failures: the doctoral student with a first dataset, the analyst in a central bank or ministry who must justify a model to a committee, and the applied economist moving into a field whose conventions are not yet familiar. It assumes a first graduate course in econometrics and a working knowledge of Python. It does not assume that the reader has read the original papers, although each part ends with a guide to them.

The idea. Every method in the book is introduced in the same way. A question is posed. The properties of the data, established by tests the reader can run, select the method. The method is then estimated on a real dataset, tested, repaired where the tests fail, checked for robustness, and written up. The chapters therefore read as short empirical papers with the reasoning shown, and the decision rules in Chapter 1 tie them together.

What is in the book. Part I sets out the decision framework and the research workflow. Part II covers eleven time-series methods, from OLS in levels to structural VARs, local projections, threshold and smooth-transition models, structural breaks and GMM. Part III covers eleven panel estimators, from pooled OLS to common-correlated-effects ARDL models and panel thresholds. Part IV covers seven causal designs. Part V covers prediction with penalised regression and tree ensembles, and causal estimation with double machine learning and causal forests. Parts VI and VII cover Bayesian VARs and their extensions and Markov-switching models. Part VIII covers nine DSGE models solved and estimated in Dynare.

How the chapters are built. Each chapter is a sequence of numbered steps. Code appears in a tinted block, and its output in a grey block below it. After the output, a box headed *How to read it* says what the numbers mean. Where the paper needs a paragraph, a box headed *How to write it* gives one whose numbers come from the output. Each chapter closes with the questions a referee will ask.

Software and data. The empirical chapters use Python with `statsmodels`, `linearmodels`, `scikit-learn`, `XGBoost`, `DoubleML` and `EconML`; the Bayesian and Markov-switching chapters use the OES engine, and the DSGE chapters use Dynare. Appendix B lists the software, and Appendix A the data of every chapter.

A note on judgement. The decision paths in this book, the choice of tests, and the design of the inference are based on my own learning, knowledge and implementation. I have tried to make every rule explicit, so that the reader can see where a dataset is routed and why, and disagree with the route if the evidence warrants. Thresholds are conventional rules of thumb, and the chapters report how sensitive their conclusions are to them. Errors that remain are mine.

Tuhin G. M. Al Mamun
Daejeon, South Korea

Contents

Preface iii

I Foundations 1

1 A Decision Framework for Applied Research 2

2 From Estimates to a Paper: Diagnostics, Robustness and Reporting 7

II Time-Series Methods 10

Overview of Part II 11

3 OLS Regression 12

4 VAR in First Differences 20

5 Vector Error-Correction Model 28

6 ARDL Bounds-Testing Model 36

7 Nonlinear ARDL (NARDL) 43

8 Structural VAR (Cholesky, Long-Run and Proxy Identification) 51

9 Local Projections (Impulse Responses) 58

10 Threshold Regression and SETAR: Regimes Defined by a Variable 65

11 Smooth-Transition Regression (LSTAR and ESTAR) 72

12 Structural Breaks at Unknown Dates (Bai–Perron) 79

13 2SLS and GMM 85

III Panel-Data Methods 95

Overview of Part III 96

14 Pooled OLS 97

15 Fixed Effects 102

16 Random Effects 108

17 Difference GMM for Dynamic Panels 113

18 Fixed Effects with Driscoll–Kraay Standard Errors **118**

19 Mean Group and CCE Mean Group **123**

20 Panel ARDL: Pooled Mean Group **128**

21 Cross-Sectionally Augmented ARDL **133**

22 Panel Model in First Differences **138**

23 Panel Local Projections **143**

24 Panel Threshold Regression (Hansen) **148**

IV Causal-Inference Designs 153

Overview of Part IV **154**

25 Instrumental Variables **155**

26 Difference-in-Differences **159**

27 Staggered Difference-in-Differences **164**

28 Generalized Synthetic Control **169**

29 Synthetic Control **174**

30 Interrupted Time Series **179**

31 Regression Discontinuity **184**

V Machine Learning for Prediction and Causal Estimation 189

Overview of Part V **190**

32 LASSO **191**

33 Ridge Regression **197**

34 Elastic Net **203**

35 Random Forest **209**

36 Gradient Boosting (XGBoost) **215**

37 Double Machine Learning: Partially Linear Model **221**

38 Double Machine Learning: Interactive Model **226**

39 Double Machine Learning with an Instrument (DML-IV) **231**

40 Causal Forest **235**

VI Bayesian Methods for Macroeconomic Systems 241

Overview of Part VI **242**

41 Bayesian Linear Regression with a g-Prior **243**

42 Bayesian VAR with Minnesota and Normal-Wishart Priors **249**

43 Mean-Adjusted (Steady-State) Bayesian VAR **257**

44 Factor-Augmented VAR (FAVAR) **264**

45 Mixed-Frequency BVAR: Nowcasting Quarterly GDP with Monthly Data **272**

46 Panel VAR: Six Estimators for Heterogeneous Countries **279**

47 Stochastic-Volatility VAR (SV-VAR) **288**

48 Time-Varying Parameter VAR (TVP-VAR) **295**

49 Bayesian Threshold VAR **302**

VII Markov-Switching Models 310

Overview of Part VII **311**

50 Markov-Switching Autoregression (Hamilton) **312**

51 Markov-Switching Regression **318**

52 Markov-Switching VAR **324**

53 Markov-Switching Panel VAR **332**

VIII DSGE Models with Dynare 340

Overview of Part VIII **341**

54 Real Business Cycle Model **342**

55 Three-Equation New Keynesian Model **348**

56 Small Open-Economy New Keynesian Model **353**

57 Optimal Monetary Policy **358**

58 Identification Before Estimation **362**

59 Bayesian Estimation of a New Keynesian Model **366**

60 Medium-Scale New Keynesian Model **374**

61 The Zero Lower Bound with OccBin **378**

62 Higher-Order Solutions and Welfare **382**

A Data of the Chapters **387**

B Software and Reproducibility **390**

References **391**

PART I

Foundations

CHAPTER 1

A Decision Framework for Applied Research

Every empirical project begins with two facts that the researcher cannot choose: a question and a dataset. The method is the only part that is a choice, and it is a choice that the data constrain far more than most textbooks admit. A regression in levels on trending series is not a bold modelling decision; it is a spurious regression (Granger and Newbold, 1974). A fixed-effects estimator applied to a panel with a short time dimension and a lagged dependent variable is not a conservative default; it is biased (Nickell, 1981). The first task of an applied researcher is therefore to find out which methods the data and the question allow, and to be able to say why.

This book is organised around that task. Each of its seven parts addresses one family of methods, and each family begins from the same idea: *the properties of the data, established by tests the reader can run, point to a small number of defensible estimators.* The chapters that follow the decision rules do not simply describe the estimators. They take one real dataset from the stated question, through the tests that select the method, to estimation, diagnostics, robustness checks and the text of the paper. Every number and figure in those chapters comes from code printed in the chapter.

Three principles that run through the book

1. The data choose among methods; preference does not. Unit-root, cointegration, specification, pre-trend and weak-instrument tests are not decoration. They decide which chapter applies.

2. Tests organise evidence; they do not replace argument. A test has low power in a short sample and can be distorted by structural breaks. The result is a starting point. The paper must still justify the choice.

3. Say what the estimate can and cannot support. An association is not an effect. Identification rests on an argument about the source of variation, and no test establishes it.

1.1 How to use this book

There are three ways in.

You have data and no model. Start with the part that matches the structure of your data: a time series (Part II), a panel (Part III), a policy or treatment with a comparison (Part IV), many predictors (Part V), or a macroeconomic system to be estimated with prior information (Parts VI–VIII). The opening page of each part contains its decision rules. Apply them to your data, find the chapter, and follow its steps.

You know the method. Go directly to the chapter. Each chapter is self-contained: it states when it applies, defines its data, and reaches a finished results section. Cross-references point to the chapters that handle the alternatives.

You are writing the paper. Every chapter closes with text for the paper, a list of what a referee will ask, and a final checklist. The shaded boxes labelled *How to read it* explain what each output means and what to conclude from it. The boxes labelled *How to write it* give sample wording in which every number is taken from the output above it.

1.2 The layout of a chapter

All method chapters have the same structure, so that a reader who has worked through one can move quickly through another.

1. The chapter opens with a short panel saying when the method applies, what dataset it uses and what software it needs.
2. The sections are numbered steps. Early steps state the question and the model, load the data and confirm the starting point (the number of observations and the unit-root or panel properties). Middle steps estimate the model and test it. Later steps repair what failed, run robustness checks, build the results table and write the paper.
3. Code is shown in a tinted block and its output in a grey block beneath it. The output is the real output of the code, produced with the data files described in Appendix A.
4. Figures are regenerated from the same code and carry a caption that names the panels.

A note on reading the output

Statistical software prints more than a paper should report. The *How to read it* boxes pick out the few numbers on which a conclusion depends, state the conclusion in words, and say what would have changed it. Learning to read the output in this way, and not only to produce it, is the main purpose of the exercise.

1.3 Decision rules by family

This section sets out the rules that route a dataset to a chapter. They are the same rules that the chapters use when they say that a dataset "is sent to" an estimator. Where a threshold appears, it is a common rule of thumb; the chapters report how sensitive the conclusion is to it.

1.3.1 Time series

Time-series methods are selected by the order of integration of the variables. A series is treated as stationary, I(0), if the augmented Dickey–Fuller (ADF) test (Dickey and Fuller, 1979) rejects a unit root in levels at the 5% level, and as I(1) if it rejects only after first differencing. The KPSS test (Kwiatkowski et al., 1992) is reported as a cross-check, and a disagreement between the two is a warning, not a verdict.

When every variable is stationary, the model can be estimated in levels (Chapter 3). When the variables are a mixture of I(0) and I(1), the ARDL bounds test (Pesaran et al., 2001) applies (Chapter 6), and its nonlinear extension (Chapter 7) when increases and decreases of a regressor may have different effects. When every variable is I(1), the Johansen trace test (Johansen, 1991) decides between a vector error-correction model (Chapter 5) and a VAR in first differences (Chapter 4). The remaining chapters are chosen by the question: structural shocks (Chapter 8), horizon-by-horizon responses (Chapter 9), regimes defined by a threshold (Chapter 10) or changing smoothly (Chapter 11), breaks at unknown dates (Chapter 12), and endogenous regressors (Chapter 13).

1.3.2 Panel data

For panels the route depends on both the number of units N and the number of periods T. Panels with T below 15 are treated as short, or micro, panels. Three tests then decide the static estimator: an F test of whether unit effects matter, the Breusch–Pagan LM test (Breusch and Pagan, 1980) of whether random effects improve on pooled OLS, and the cluster-robust Mundlak test (Mundlak, 1978), reported with the conventional Hausman test (Hausman, 1978), of whether the unit effects are correlated with the regressors. A dynamic model goes to GMM (Arellano and Bond, 1991) only if N is at least 30, and otherwise to bias-corrected fixed effects.

Panels with T of at least 15 and at least 10 units are treated as macro panels. The Pesaran CD test (Pesaran, 2015) checks cross-sectional dependence, and the Pesaran–Yamagata test (Pesaran and Yamagata, 2008) checks slope heterogeneity. Each variable is tested for a unit root with the CIPS test (Pesaran, 2007) when it is cross-sectionally dependent and with the Fisher-type ADF test otherwise. When every variable is I(1), cointegration is tested with a Fisher–Engle–Granger test if slopes are heterogeneous and with the Kao test (Kao, 1999) otherwise. With T of at least 15 but fewer than 10 units, each unit is modelled as a time series. Part III assigns the estimators to these cells.

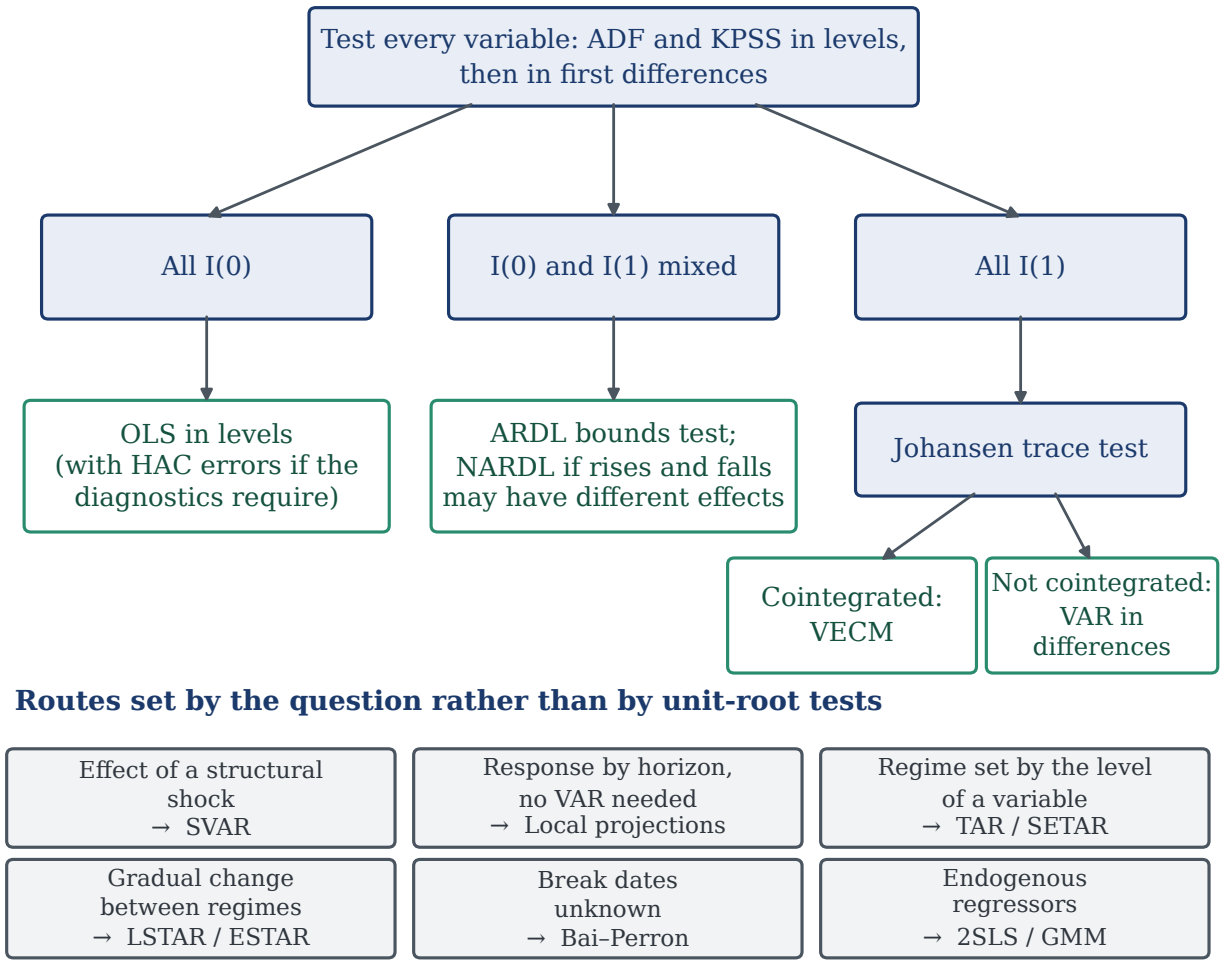


Figure 1.1. Decision map for a set of time series. The upper part is determined by unit-root and cointegration tests; the lower row lists methods chosen by the question.

1.3.3 Causal inference

The design is determined by the source of variation in the treatment, which the researcher declares, and the data then decide among designs in that family. With a running variable and a cutoff, the design is a regression discontinuity, with the Imbens–Kalyanaraman bandwidth (Imbens and Kalyanaraman, 2012) and checks of manipulation (McCrary, 2008), covariate continuity, placebo cutoffs and bandwidth sensitivity; if treatment is not fully determined by the cutoff the design is fuzzy. With an instrument, the design is instrumental variables: OLS and 2SLS are compared, the first-stage F statistic is reported (below 10 signals a weak instrument), together with the Anderson–Rubin test, which is robust to weak instruments, and an endogeneity test.

With a treatment that begins at a point in time, the number of treated units decides. With three or fewer, the design is a synthetic control (Abadie et al., 2010) if at least five untreated units are available and an interrupted time series otherwise. With more treated units, an event study tests for parallel pre-trends. Designs that fail the pre-trend test go to generalized synthetic control (Xu, 2017). The others go to difference-in-differences with one adoption date or to a heterogeneity-robust staggered estimator (Callaway and Sant'Anna, 2021) when adoption dates differ.

These rules organise the evidence. They do not establish identification, which rests on the argument that the source of variation is as good as random once the design's conditions hold.

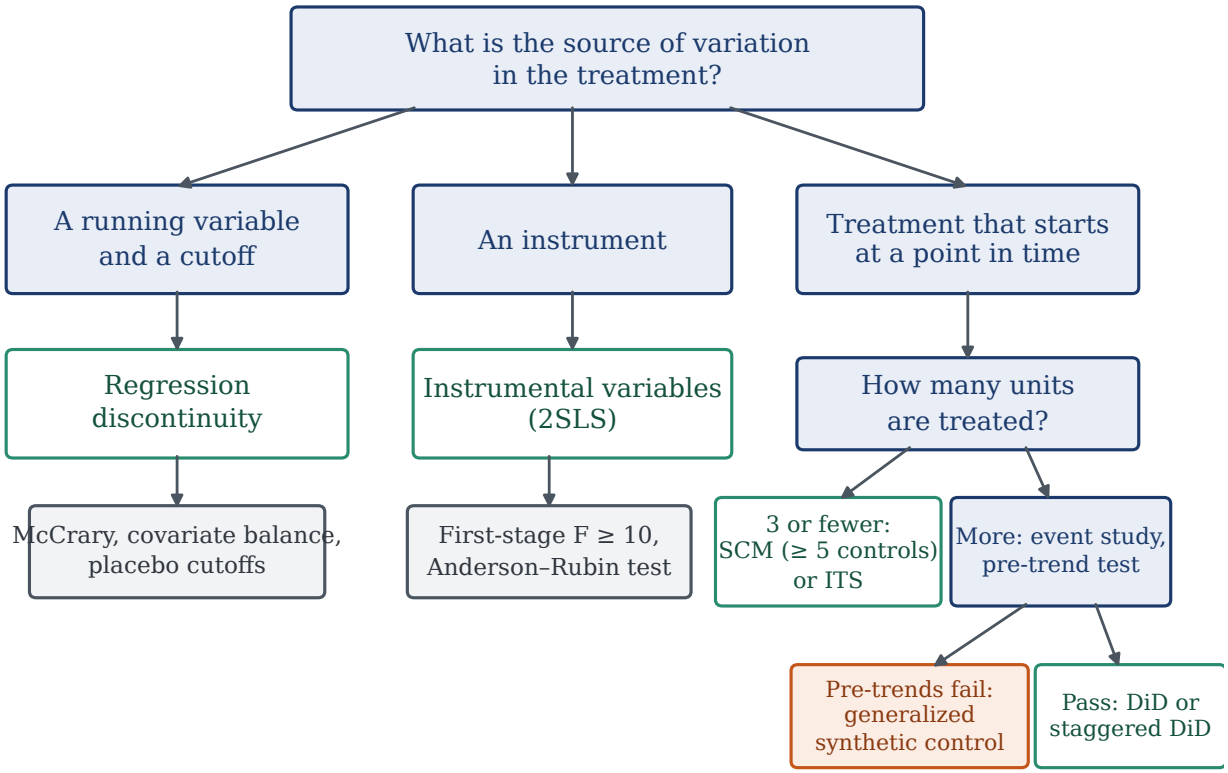


Figure 1.2. Decision map for causal designs. The source of variation, which the researcher declares, fixes the family; the number of treated units and the pre-trend test select within the policy-timing family.

1.3.4 Machine learning

The rules for machine learning begin by separating prediction from causal estimation. They are numbered because later chapters refer to them by number.

Prediction.

A1 A date column with at least 50 periods marks a time series. Each series is tested with ADF and KPSS, I(1) series are differenced, and validation uses rolling-origin splits that never shuffle time.

A2 With fewer than 100 observations only regularised linear models are used.

A3 Otherwise a random forest is compared with LASSO by five-fold cross-validated RMSE. If the forest is at least 5% better, the data are treated as nonlinear.

A4 Linear case. If any two predictors are correlated above 0.8, use the elastic net. Otherwise use LASSO if it keeps fewer than half of the predictors at the one-standard-error penalty, and ridge if it keeps more.

A5 Nonlinear case. Use a random forest below 1,000 observations and gradient boosting above.

Causal effect.

B1 If confounders are unobserved and there is no instrument, machine learning cannot help. A research design from Part IV is needed.

B2 With fewer than 10 controls that enter linearly (the forest does not beat LASSO by 5% for the outcome or for the treatment), OLS is enough, and the double machine learning estimate is a robustness check.

B3 An instrument leads to DML-IV, and the first-stage F should be at least 10. A binary treatment leads to the interactive model, after checking that the propensity scores lie in $[0.05, 0.95]$. A continuous treatment leads to the partially linear model.

B4 Double machine learning is estimated with LASSO and random-forest learners and five-fold cross-fitting. If the two differ by more than one standard error, the forest is reported.

B5 With at least 2,000 observations and a question about who is affected most, use a causal forest.

1.3.5 Bayesian and Markov-switching models

Panels go to the panel VAR; a regression equation goes to Bayesian regression; a series observed at mixed frequencies goes to the mixed-frequency BVAR; and 15 or more indicators go to the factor-augmented VAR. For a set of macro series, the stationarity of the series is checked first and, if they are not I(1) in levels, three kinds of nonlinearity are tested on a VAR(1): changing volatility (variance ratio, Levene and ARCH tests), unstable coefficients (CUSUM and sup-F) and a threshold (a bootstrap sup-LR test). The strongest rejection decides between the stochastic-volatility, time-varying-parameter and threshold VAR; if none rejects, a constant-parameter Bayesian VAR is adequate. A two-regime Markov-switching fit with a likelihood-ratio statistic above about 14 (Garcia, 1998) sends the researcher to Part VII, where the structure of the data decides the model: one series, a regression, several series or a panel. If an observed variable above or below a threshold defines the regime, the threshold VAR is the better tool. These tests are screening devices on a simple model and do not replace the diagnostics inside each chapter.

1.3.6 DSGE models

The route follows from what the model is for. To *explain a mechanism*, the model needs only the frictions that the mechanism requires: no nominal rigidities for the real business cycle model, sticky prices for the New Keynesian model, and an exchange rate and imported goods for an open economy. To *estimate*, identification comes first, and the size of the model follows from the number of observables, which must not exceed the number of shocks, since otherwise the likelihood is singular. Three observables or fewer fit a small model; four or more call for a medium-scale model with capital, habits and adjustment costs. To *design policy*, the model is solved under commitment, discretion and optimised simple rules and compared by a loss function. When nonlinearity is the point, an occasionally binding constraint calls for the OccBin method (Guerrieri and Iacoviello, 2015), and risk, precautionary behaviour or welfare call for a second- or third-order solution. On every route the steady state must solve and the Blanchard–Kahn conditions (Blanchard and Kahn, 1980) must hold before any result can be read.

1.4 What the rules cannot do

Two limits apply to every rule in this chapter. First, tests are screening devices. A unit-root test has low power in a short sample, a panel test can be distorted by structural breaks, and a pre-trend test that fails to reject is not evidence that trends are parallel. A paper that follows a rule should therefore say which rule it followed, report the test statistics, and show that the main conclusion survives the neighbouring choice.

Second, the rules do not make a model correct. They exclude methods that the data contradict. They do not replace the economic argument for why the model describes the process that generated the data, or the argument for why a coefficient has the interpretation the paper gives it. Chapter 2 turns to that argument.

Author's note on the decision paths

The decision paths, the choice of tests and the design of the inference in this book are based on the author's own learning, knowledge and implementation. Thresholds such as $T < 15$ for a short panel, a first-stage F of 10, or a 5% improvement of the forest over LASSO are conventional rules of thumb and not theorems. They are applied consistently so that the reader can see exactly where a dataset is routed and why, and can change the threshold and see whether the route changes.

CHAPTER 2

From Estimates to a Paper: Diagnostics, Robustness and Reporting

A model chosen by the rules of Chapter 1 is a hypothesis, not a result. Between the first regression and a defensible paper lie three kinds of work: testing whether the model's assumptions hold, repairing what fails, and showing that the conclusion does not depend on one specification. This chapter describes that work in general terms. The method chapters carry it out on real data, and Chapter 3 is the most complete worked example.

2.1 State the model before the result

Write down the equation, define every variable with its unit, and state the sign that theory predicts for each coefficient, and why. A table of symbols, file names, definitions and expected signs, such as the one in Section 3.1, takes ten minutes and does two things. It lets a referee check the paper against its own prior, and it prevents the explanation of a result after the fact. A coefficient with the unexpected sign is informative only if the expectation was written down first (Leamer, 1983).

2.2 Diagnose before you interpret

Each estimator rests on assumptions, and each assumption has a test. The choice of test follows from the assumption, not from the software menu. Table 2.1 lists the assumptions that recur throughout the book, the tests used for them, and the usual repair.

Two habits are worth stating plainly. First, a failed diagnostic usually has a specific remedy, and the remedy should be applied and reported rather than the test dropped. Second, the remedy that matters most depends on what the coefficient is used for. Heteroskedasticity and serial correlation leave OLS coefficients unbiased and make the conventional standard errors wrong, so the fix is to the inference. A failed functional-form test means the model itself is wrong, so the fix is to the specification.

2.3 Show that the result is robust

A result that holds in one specification only does not convince. A robustness section answers the questions a sceptical reader would ask, and the answers should be in the paper rather than in the response letter. The checks that appear in nearly every chapter of this book are the following.

- **Alternative specifications:** a different lag length, a different set of controls, a different transformation of the variables.
- **Alternative inference:** clustered, robust or HAC standard errors, and a bootstrap where the asymptotics are doubtful.
- **Sub-samples and influential observations:** the sample without the extreme period (for financial data, the 2008–09 crisis), or without one unit (the leave-one-unit-out check in the panel chapters).
- **Placebo and falsification tests:** outcomes that should not respond, dates at which nothing happened, and variables that cannot be affected by the treatment.
- **Sensitivity to the choices in this book's own rules:** the thresholds of Chapter 1, the bandwidth of a discontinuity, the number of regimes or factors.

Read down each column of a robustness table and ask whether the sign and the significance of the coefficient of interest survive. A coefficient that changes sign across reasonable specifications is a finding about fragility, and the paper should say so.

Table 2.1. Recurring assumptions, their tests and the usual repair.

Assumption	Test	If it fails
No serial correlation in the errors	Durbin–Watson, Breusch–Godfrey	Heteroskedasticity-and-autocorrelation-consistent (HAC) standard errors (Newey and West, 1987), or more dynamics in the model
Constant error variance	White (White, 1980), Breusch–Pagan, ARCH	Robust standard errors; model the variance (Chapter 47)
Correct functional form	RESET (Ramsey, 1969)	Add nonlinear terms; consider a threshold or smooth-transition model (Chapters 10, 11)
Stable coefficients	CUSUM (Brown et al., 1975), Chow and sup-F	Allow breaks (Chapter 12) or time-varying parameters (Chapter 48)
Stationary variables	ADF, KPSS	Difference, or use a cointegration framework (Chapters 5, 6)
Exogenous regressors	Durbin–Wu–Hausman, over-identification tests	Instruments (Chapters 13, 25)
Independent units in a panel	Pesaran CD	Driscoll–Kraay or common-correlated-effects estimators (Chapters 18, 21)

Table 2.2. How the steps of a method chapter feed the sections of a paper.

Section of the paper	What goes in it	Comes from
Introduction	The question, why it matters, what is found and how it differs from earlier work	The question and the main result
Data	Sources, sample, definitions, summary statistics, unit-root or panel tests	The data step and the decision rules
Method	The equation, expected signs, the estimator and the standard errors	The model step and the repair step
Results	The main table and its interpretation	The estimation and results-table steps
Robustness	Diagnostics and alternative specifications	The diagnostic and robustness steps
Conclusion	The answer, its limits and implications	All steps

2.4 Report the results in the form a referee expects

Journals expect one main table in which the specifications stand side by side, with the same variables in the same rows, the standard error in parentheses under each coefficient, stars defined in a note, and the sample size and fit statistics at the foot. The note should name the type of standard error. The table is built from stored estimation results and not copied by hand, so that every number in the text can be traced to a line of code. Figures follow the same rule: each figure is produced by a script, labelled with its units, and captioned so that it can be read without the text.

2.5 Be explicit about what the estimate means

The last discipline is a verbal one. A regression of returns on macroeconomic variables measures association. A difference-in-differences coefficient measures an effect only under parallel trends. A DSGE parameter is identified only if the data can distinguish it from the others. Each chapter ends with the questions that a referee will ask on exactly this point, and each answers them in the order in which they arise. The standard

to aim for is a paper in which every causal word (*effect*, *impact*, *leads to*) has been earned by a design, and every other result is described as what it is.

Checklist before submission

- Every variable is defined with its unit and source, and the expected sign was written down first.
- The decision rule that led to the estimator is named, with the test statistics.
- Each diagnostic that failed has been repaired or discussed.
- The main result is shown in at least three alternative specifications, with the type of standard error stated.
- Every number in the text, and every figure, is produced by code that runs from the raw data.
- Causal language appears only where a research design supports it.

PART II

Time-Series Methods

Overview

Time series are the raw material of macroeconomics and finance: a single history, observed once, in which the order of the observations matters. That one fact explains most of what is distinctive about this part. A regression of one trending series on another can look excellent and mean nothing (Granger and Newbold, 1974), so the first question is always the order of integration of each variable. The answer routes the data to a levels regression, an error-correction model, a model in differences, or a bounds test, as set out in Chapter 1.

The part then moves from *which model* to *which question*. A vector autoregression summarises the dynamics of several variables without imposing much theory (Sims, 1980); a structural VAR adds an identification argument so that a shock has an economic meaning (Kilian and Lütkepohl, 2017); local projections estimate the response horizon by horizon (Jordà, 2005). The last chapters treat departures from a constant linear relationship: abrupt regimes set by a threshold (Hansen, 2000), gradual regimes (Teräsvirta, 1994), breaks at unknown dates (Bai and Perron, 1998), and endogenous regressors that call for instruments and GMM (Hansen, 1982).

Chapters at a glance

Ch.	Chapter	Use it when
3	OLS Regression	All variables stationary in levels
4	VAR in First Differences	All variables I(1), no cointegration
5	Vector Error-Correction Model	All variables I(1), cointegrated
6	ARDL Bounds-Testing Model	A mix of I(0) and I(1) variables
7	Nonlinear ARDL (NARDL)	I(0) and I(1) variables, with asymmetric effects
8	Structural VAR (Cholesky, Long-Run and Proxy Identification)	You want the effect of a structural shock on several variables, and you can defend how the shock is identified
9	Local Projections (Impulse Responses)	You want the dynamic response of an outcome to a shock, estimated horizon by horizon, without fitting a full VAR
10	Threshold Regression and SETAR: Regimes Defined by a Variable	The relationship or the dynamics change abruptly when a variable (often the series' own past) crosses a level, and you want to find the level
11	Smooth-Transition Regression (LSTAR and ESTAR)	The coefficients depend on the state of the economy, but they change gradually, not abruptly, as a variable moves through a range
12	Structural Breaks at Unknown Dates (Bai–Perron)	A mean, a trend or a regression coefficient may have shifted, and you want to find how many times and when
13	2SLS and GMM	Endogenous regressors that need instruments

Further reading

The standard references are Hamilton (1994) for the theory of time series and Kilian and Lütkepohl (2017) for structural VARs. Stock and Watson (2001) is a short introduction to the VAR toolkit. Unit-root and stationarity tests are due to Dickey and Fuller (1979) and Kwiatkowski et al. (1992); cointegration to Engle and Granger (1987) and Johansen (1991); the bounds test to Pesaran et al. (2001); the nonlinear ARDL to Shin et al. (2014). Heteroskedasticity-and-autocorrelation-consistent inference is due to Newey and West (1987).

CHAPTER 3

OLS Regression

Applies when. All variables stationary in levels

Overview. Use this chapter when the unit-root tests show that every variable is I(0). It takes one real dataset from the model equation to the finished results table, shows what to do when diagnostic tests fail, and gives sample text for each part of the paper.

Data. U.S. stock returns and the economy, monthly, 1985–2019

Software. Python · statsmodels

3.1 The question and the model

Every empirical paper starts from a question that a regression can answer. Ours is: **do macroeconomic conditions explain monthly U.S. stock returns?**

The decision rules of Chapter 1 showed that all four variables are stationary in levels, so the model can be estimated by ordinary least squares in levels:

$$r_t = \beta_0 + \beta_1 \, \Delta ip_t + \beta_2 \, \Delta ffr_t + \beta_3 \, \Delta cs_t + \varepsilon_t$$

Symbol	Variable in the file	Definition	Expected sign
r_t	`sp500_return`	Monthly log return on the S&P 500, in percent	dependent variable
Δip_t	`ip_growth`	Monthly growth of industrial production, in percent	positive: stronger activity raises expected cash flows
Δffr_t	`d_fedfunds`	Change in the federal funds rate, percentage points	negative: a higher discount rate lowers prices
Δcs_t	`d_credit_spread`	Change in the Baa–10-year Treasury spread, percentage points	negative: wider spreads signal higher risk

Before you estimate anything

Write down the equation, define every variable with its unit, and state the sign you expect and why. Referees look for this table. It also protects you from explaining a result only after you have seen it.

How to write it: model section

To examine whether macroeconomic conditions explain stock returns, we estimate $r_t = \beta_0 + \beta_1 \Delta ip_t + \beta_2 \Delta ffr_t + \beta_3 \Delta cs_t + \varepsilon_t$*, where* r_t *is the monthly log return on the S&P 500,* Δip_t *is the growth rate of industrial production,* Δffr_t *is the change in the federal funds rate and* Δcs_t *is the change in the Baa–Treasury credit spread. We expect* $\beta_1 > 0$*, since stronger real activity raises expected cash flows, and* $\beta_2 < 0$ *and* $\beta_3 < 0$*, since higher discount rates and higher credit risk lower equity valuations.*

3.2 Load the data and confirm the starting point

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

```
import statsmodels.formula.api as smf

df = pd.read_csv("returns.csv", parse_dates=["date"]).set_index("date")
print(df.shape)
print(df.describe().T[["mean", "std", "min", "max"]].round(3))
```

```
(420, 4)
                 mean    std     min     max
sp500_return    0.705  3.537 -22.804  11.352
ip_growth       0.149  0.608  -4.485   2.039
d_fedfunds     -0.016  0.192  -0.960   0.870
d_credit_spread 0.000  0.169  -0.990   1.450
```

```
from statsmodels.tsa.stattools import adfuller

for col in df:
    stat, p = adfuller(df[col], autolag="AIC")[:2]
    print(f"{col:16s} ADF = {stat:6.2f}   p-value = {p:.3f}")
```

```
sp500_return     ADF = -15.73   p-value = 0.000
ip_growth        ADF =  -5.42   p-value = 0.000
d_fedfunds       ADF =  -4.19   p-value = 0.001
d_credit_spread  ADF =  -5.20   p-value = 0.000
```

How to read it

There are 420 monthly observations. The ADF p-values are all below 0.05, so a unit root is rejected for every variable: all four are I(0). This is the condition for using OLS in levels. If even one variable were I(1), you would be in a different guide.

Note the minimum return of –22.8%. That is October 2008. One extreme month can drive a regression, so we will check later whether the results survive without the crisis.

3.3 Estimate the baseline regression

```
model = smf.ols("sp500_return ~ ip_growth + d_fedfunds + d_credit_spread", data=df).fit()
print(model.summary().tables[1])
print(f"R-squared = {model.rsquared:.3f}   Adj. R-squared = {model.rsquared_adj:.3f}")
print(f"F-statistic = {model.fvalue:.2f}   p-value = {model.f_pvalue:.2e}   N = {int(model.nobs)}")
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.7150      0.160      4.456      0.000       0.400       1.030
ip_growth     -0.1675      0.260     -0.644      0.520      -0.678       0.343
d_fedfunds    -1.0838      0.851     -1.273      0.204      -2.757       0.589
d_credit_spread    -9.7470      0.951    -10.249      0.000     -11.616      -7.878
==============================================================================
R-squared = 0.205   Adj. R-squared = 0.199
F-statistic = 35.71   p-value = 1.49e-20   N = 420
```

How to read it

- **Coefficient.** A one-percentage-point rise in the credit spread within a month is associated with a return that is 9.7 percentage points lower. A typical monthly change in the spread is 0.17 points (its standard deviation), which corresponds to about –1.7 points of return.
- **t-statistic and p-value.** Only the credit spread is significant. Industrial production growth and the change in the federal funds rate have p-values of 0.52 and 0.20, so we cannot say their effects differ from zero.
- **R-squared.** The three variables explain about 20% of the variation in monthly returns. For stock returns this is high; most of it comes from the credit spread.
- **F-statistic.** The regressors are jointly significant.

Do not report these standard errors yet. They are valid only if the residuals pass the diagnostic tests in the next steps.

3.4 Diagnostic test 1: serial correlation

OLS standard errors assume that the errors are not correlated over time. The Breusch–Godfrey test has the null hypothesis of **no serial correlation**.

```
from statsmodels.stats.diagnostic import acorr_breusch_godfrey
from statsmodels.stats.stattools import durbin_watson

print(f"Durbin-Watson = {durbin_watson(model.resid):.2f}")
for lags in (1, 6, 12):
    lm, p = acorr_breusch_godfrey(model, nlags=lags)[:2]
    print(f"Breusch-Godfrey, {lags:2d} lags: LM = {lm:6.2f}   p-value = {p:.4f}")
```

```
Durbin-Watson = 1.61
Breusch-Godfrey,  1 lags: LM =  16.43   p-value = 0.0001
Breusch-Godfrey,  6 lags: LM =  23.27   p-value = 0.0007
Breusch-Godfrey, 12 lags: LM =  32.58   p-value = 0.0011
```

How to read it

A Durbin–Watson statistic near 2 means no first-order autocorrelation; 1.61 points to positive autocorrelation. The Breusch–Godfrey p-values are far below 0.05 at every lag length, so the null of no serial correlation is rejected.

There is a specific reason here. The S&P 500 series is a monthly *average* of daily prices, and averaging creates first-order autocorrelation in returns even when daily returns have none. Knowing your data's construction often explains a failed test.

The coefficients are still unbiased, but the standard errors are wrong. The fix comes in the step on robust standard errors.

3.5 Diagnostic test 2: heteroskedasticity

OLS standard errors also assume a constant error variance. The White and Breusch–Pagan tests have the null of **homoskedasticity**; the ARCH test checks whether volatility clusters over time.

```
from statsmodels.stats.diagnostic import het_white, het_breuschpagan, het_arch

exog = model.model.exog
print(f"White:          LM = {het_white(model.resid, exog)[0]:6.2f}   p-value = {het_white(model.resid, exog)
  ↪ [1]:.4f}")
print(f"Breusch-Pagan:  LM = {het_breuschpagan(model.resid, exog)[0]:6.2f}   p-value = {het_breuschpagan(model.
  ↪ resid, exog)[1]:.4f}")
print(f"ARCH (6 lags):  LM = {het_arch(model.resid, nlags=6)[0]:6.2f}   p-value = {het_arch(model.resid, nlags
  ↪ =6)[1]:.4f}")
```

```
White:          LM =  64.77   p-value = 0.0000
Breusch-Pagan:  LM =  19.65   p-value = 0.0002
ARCH (6 lags):  LM =  29.77   p-value = 0.0000
```

How to read it

All three tests reject at the 1% level. The error variance is not constant, and it clusters in time, which is the usual finding for financial returns: calm periods and turbulent periods alternate. As with serial correlation, the coefficients are unaffected but the conventional standard errors cannot be trusted.

3.6 Diagnostic test 3: normality of the residuals

```
from statsmodels.stats.stattools import jarque_bera

jb, p, skew, kurt = jarque_bera(model.resid)
print(f"Jarque-Bera = {jb:.2f}   p-value = {p:.4f}   skewness = {skew:.2f}   kurtosis = {kurt:.2f}")

import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 2, figsize=(8, 3))
model.resid.plot(ax=ax[0], lw=.8, title="Residuals over time")
ax[1].hist(model.resid, bins=40); ax[1].set_title("Distribution of residuals")
plt.tight_layout()
```

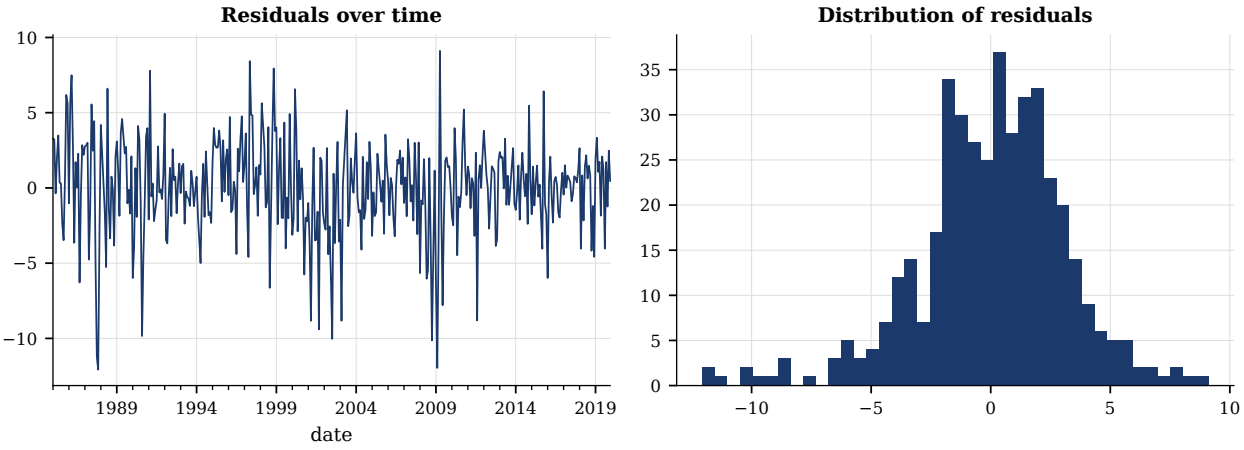


Figure 3.1. Diagnostic test 3: normality of the residuals. Panels: Residuals over time; Distribution of residuals.

```
Jarque-Bera = 72.35   p-value = 0.0000   skewness = -0.62   kurtosis = 4.62
```

How to read it

The Jarque–Bera test rejects normality: the residuals are skewed to the left (–0.62) and have fat tails (kurtosis 4.6 against 3 for a normal distribution). The plots show why: a few very large negative months, above all in 1987 and 2008.

With 420 observations this is not a serious problem. The t- and F-tests are still valid in large samples by the central limit theorem. State this in the paper rather than hiding the test. Normality matters much more in small samples.

3.7 Diagnostic test 4: functional form, multicollinearity and stability

```
from statsmodels.stats.diagnostic import linear_reset, breaks_cusumolsresid
from statsmodels.stats.outliers_influence import variance_inflation_factor

reset = linear_reset(model, power=2, use_f=True)
print(f"Ramsey RESET: F = {reset.fvalue:.2f}   p-value = {reset.pvalue:.4f}")

names = model.model.exog_names
for i in range(1, len(names)):
    print(f"VIF {names[i]:16s} = {variance_inflation_factor(model.model.exog, i):.2f}")

stat, p = breaks_cusumolsresid(model.resid, ddof=len(names))[:2]
print(f"CUSUM test of parameter stability: statistic = {stat:.2f}   p-value = {p:.4f}")
```

```
Ramsey RESET: F = 28.66   p-value = 0.0000
VIF ip_growth        = 1.04
VIF d_fedfunds       = 1.12
VIF d_credit_spread  = 1.08
CUSUM test of parameter stability: statistic = 1.51   p-value = 0.0205
```

```
rec = sm.RecursiveLS(model.model.endog, model.model.exog).fit()
fig = plt.figure(figsize=(8, 3))
rec.plot_cusum(fig=fig)
fig.axes[0].set_title("CUSUM of recursive residuals with 5% bounds")
```

How to read it

- **RESET** has the null that the linear functional form is correct. It is rejected ($p < 0.001$), so something nonlinear is missing from the model. This one affects the coefficients themselves, not just the standard errors, so it must be dealt with.
- **VIF** values are all close to 1. Values above 10 would signal harmful multicollinearity; here there is none.

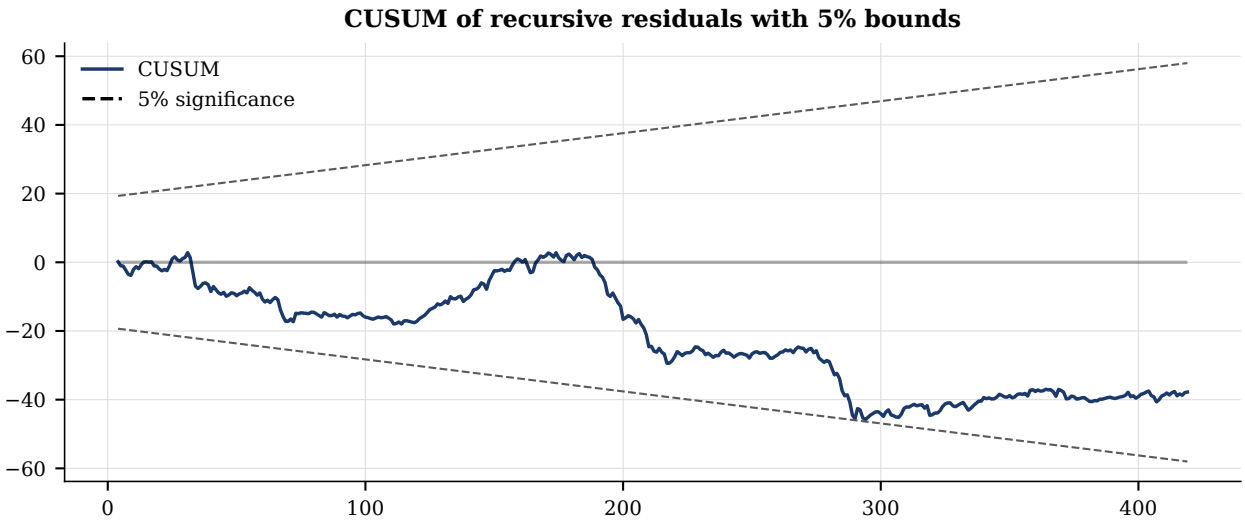


Figure 3.2. Diagnostic test 4: functional form, multicollinearity and stability: CUSUM of recursive residuals with 5% bounds.

- **CUSUM** tests whether the coefficients are stable over the sample. The test statistic rejects stability at 5% ($p = 0.02$), and in the plot the CUSUM line drifts down and touches its lower bound near observation 290, which is early 2009. Both point to a change in the relationship around the financial crisis, so a subsample check is needed later.

How to write it: diagnostics paragraph

Diagnostic tests on the baseline specification indicate serial correlation (Breusch–Godfrey LM(6) = 23.27, $p < 0.01$) and heteroskedasticity (White LM = 64.77, $p < 0.01$) in the residuals. Inference is therefore based on Newey–West heteroskedasticity- and autocorrelation-consistent standard errors throughout. The Jarque–Bera test rejects normality, reflecting a small number of extreme negative returns; given the sample of 420 observations, inference relies on asymptotic results. Variance inflation factors below 1.2 rule out multicollinearity. The RESET test rejects the linear specification ($F = 28.66$, $p < 0.01$), which motivates the nonlinear specification considered below.

3.8 Fix what failed: robust standard errors and a better specification

Two different problems need two different fixes.

Serial correlation and heteroskedasticity leave the coefficients unchanged but make the standard errors wrong. The standard remedy is Newey–West (HAC) standard errors. A common rule for the lag length is $4(T/100)^{2/9}$, which gives 5 for 420 observations.

```
lags = int(4 * (len(df) / 100) ** (2 / 9))
hac = model.get_robustcov_results(cov_type="HAC", maxlags=lags)
print(f"Newey-West lags: {lags}")
print(hac.summary().tables[1])
```

```
Newey-West lags: 5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.7150      0.180      3.974      0.000       0.361       1.069
ip_growth     -0.1675      0.299     -0.561      0.575      -0.755       0.420
d_fedfunds    -1.0838      0.942     -1.150      0.251      -2.936       0.768
d_credit_spread    -9.7470      1.792     -5.438      0.000     -13.270      -6.224
==============================================================================
```

A rejected RESET test means the model itself should change. A natural candidate is that large moves in the credit spread matter more than proportionally, so we add the squared change.

```
model2 = smf.ols("sp500_return ~ ip_growth + d_fedfunds + d_credit_spread + I(d_credit_spread**2)", data=df).
  ↪ fit()
```

```
hac2 = model2.get_robustcov_results(cov_type="HAC", maxlags=lags)
print(hac2.summary().tables[1])
print(f"R-squared = {model2.rsquared:.3f}")
print(f"RESET p-value = {linear_reset(model2, power=2, use_f=True).pvalue:.3f}")
```

```
==============================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept                     0.9036      0.179      5.044      0.000       0.551       1.256
ip_growth                    -0.1470      0.318     -0.462      0.644      -0.773       0.479
d_fedfunds                   -2.0100      0.910     -2.208      0.028      -3.799      -0.221
d_credit_spread              -8.7109      1.199     -7.267      0.000     -11.067      -6.355
I(d_credit_spread ** 2)      -7.2858      0.956     -7.621      0.000      -9.165      -5.407
==============================================================================================
R-squared = 0.260
RESET p-value = 0.100
```

How to read it

- With HAC standard errors the credit-spread coefficient is unchanged at –9.75, but its standard error almost doubles, from 0.95 to about 1.8. It is still strongly significant. This is typical: ignoring serial correlation and heteroskedasticity makes results look more precise than they are.
- In the extended model the squared term is negative and significant: large widenings of the spread hurt returns more than proportionally. RESET no longer rejects at 5% ($p = 0.10$), and R-squared rises from 0.20 to 0.26.
- The federal funds rate now has the expected negative sign and is significant at 5%. A better specification can change your conclusions about other variables, which is why the RESET result could not be ignored.

3.9 Robustness checks

A result that holds in one specification only will not convince a referee. Check the choices a reader would question: the extreme crisis months, the dynamics, and the sample.

```
spec = "sp500_return ~ ip_growth + d_fedfunds + d_credit_spread + I(d_credit_spread**2)"
fit = lambda formula, data: smf.ols(formula, data=data).fit(cov_type="HAC", cov_kwds={"maxlags": lags})

no_crisis = df[(df.index < "2008-09-01") | (df.index > "2009-06-01")]
checks = {
    "Preferred model":          fit(spec, df),
    "Excluding 2008-09 crisis": fit(spec, no_crisis),
    "With lagged return":       fit(spec + " + sp500_return.shift(1)", df),
    "1985-2001":                fit(spec, df[:"2001-12"]),
    "2002-2019":                fit(spec, df["2002-01":]),
}
print(f"{'':26s}{'credit spread':>14s}{'p-value':>9s}{'fed funds':>11s}{'p-value':>9s}{'N':>6s}")
for name, m in checks.items():
    print(f"{name:26s}{m.params['d_credit_spread']:14.2f}{m.pvalues['d_credit_spread']:9.3f}"
          f"{m.params['d_fedfunds']:11.2f}{m.pvalues['d_fedfunds']:9.3f}{int(m.nobs):6d}")
```

```
                         credit spread  p-value  fed funds  p-value     N
Preferred model                  -8.71    0.000      -2.01    0.027   420
Excluding 2008-09 crisis         -9.06    0.000      -2.12    0.018   410
With lagged return               -7.84    0.000      -2.09    0.014   419
1985-2001                        -4.18    0.084      -1.63    0.099   204
2002-2019                       -10.71    0.000      -2.35    0.035   216
```

How to read it

Read down each column and ask whether the sign, size and significance survive.

- **Credit spread.** Negative in every row, and significant at 1% everywhere except 1985–2001, where it is smaller (–4.18) and significant only at 10%. The relationship is clearly stronger after 2002. That matches the CUSUM warning from Section 3.7, and it belongs in the paper.
- **Federal funds rate.** Between –1.6 and –2.4 in every row; significant at 5% except in the early subsample.

- **The crisis.** Dropping September 2008 to June 2009 barely changes either coefficient, so the main result is not driven by the crisis months.

Report honestly what holds and what weakens. A robustness table in which one result weakens is more credible than one in which nothing ever changes.

3.10 Build the results table for the paper

Journals expect one table with the specifications side by side, standard errors in parentheses and significance stars.

```
from statsmodels.iolib.summary2 import summary_col

table = summary_col(
    [fit("sp500_return ~ ip_growth + d_fedfunds + d_credit_spread", df), checks["Preferred model"],
     checks["Excluding 2008-09 crisis"], checks["With lagged return"]],
    model_names=["(1) Baseline", "(2) Preferred", "(3) No crisis", "(4) Dynamic"],
    stars=True, float_format="%.3f",
    info_dict={"N": lambda m: f"{int(m.nobs)}"},
)
print(table)
```

```
===========================================================================
                       (1) Baseline (2) Preferred (3) No crisis (4) Dynamic
---------------------------------------------------------------------------
Intercept              0.715***     0.904***      1.114***      0.828***
                       (0.180)      (0.179)       (0.192)       (0.165)
ip_growth              -0.167       -0.147        -0.451        -0.229
                       (0.299)      (0.318)       (0.344)       (0.316)
d_fedfunds             -1.084       -2.010**      -2.121**      -2.085**
                       (0.942)      (0.910)       (0.898)       (0.849)
d_credit_spread        -9.747***    -8.711***     -9.062***     -7.843***
                       (1.792)      (1.199)       (1.667)       (1.354)
I(d_credit_spread ** 2)             -7.286***     -14.837**     -7.297***
                                    (0.956)       (7.204)       (0.874)
sp500_return.shift(1)                                           0.113*
                                                                (0.060)
R-squared              0.205        0.260         0.164         0.270
R-squared Adj.         0.199        0.253         0.156         0.262
N                      420          420           410           419
===========================================================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

How to read and format it

In column 3 the squared term has a much larger standard error (7.2 against 1.0). Without the crisis months there are few large spread movements left, so the nonlinearity is estimated imprecisely. It is identified mainly from turbulent periods, and the paper should say so.

Give the table a title and a note that a reader can understand without the text: the dependent variable, the sample, the type of standard errors and the meaning of the stars. For example: *"Dependent variable: monthly S&P 500 log return (%). Sample: 1985:01–2019:12. Newey–West standard errors with 5 lags in parentheses. *** p<0.01, ** p<0.05, * p<0.1."* To export the table to Word or LaTeX use `table.as_latex()` or `table.tables[0].to_excel("results.xlsx")`.

3.11 Write the paper

The analysis is finished. The paper now has a standard structure, and each step above feeds one part of it.

Section	What goes in it	From which step
1. Introduction	The question, why it matters, what you find, how it differs from earlier work	Sections 3.1 and 3.10
2. Literature	What is known, and the gap your paper fills	your reading
3. Data	Sources, sample, definitions, summary statistics, unit-root tests	Section 3.2 and the decision rules of Chapter 1
4. Method	The equation, expected signs, estimator, standard errors	Sections 3.1 and 3.8
5. Results	The main table and its interpretation	Sections 3.3, 3.8 and 3.10
6. Robustness	Diagnostic tests and alternative specifications	Sections 3.4 to 3.9
7. Conclusion	The answer to the question, limits, implications	all

How to write it: results paragraph

Table 2 reports the estimates. Changes in credit conditions are the dominant macroeconomic correlate of stock returns: in the baseline specification, a one-percentage-point widening of the Baa–Treasury spread is associated with a 9.7-percentage-point lower monthly return (column 1), and the relationship is nonlinear, with larger widenings having more than proportional effects (column 2). Once this nonlinearity is accounted for, increases in the federal funds rate are also associated with lower returns. Industrial production growth has no significant contemporaneous relationship with returns in any specification. The model explains 26% of the monthly variation in returns.

How to write it: abstract

This paper examines whether macroeconomic conditions explain monthly U.S. stock returns over 1985–2019. Using OLS with heteroskedasticity- and autocorrelation-consistent standard errors, we find that changes in the corporate credit spread are the main macroeconomic correlate of returns, with a nonlinear effect that strengthens for large spread movements. Changes in the federal funds rate are negatively related to returns once this nonlinearity is modelled, while industrial production growth is not. The results are robust to excluding the 2008–09 financial crisis and to allowing for return dynamics, and the relationship with credit spreads is markedly stronger after 2002.

What a referee will ask

- **Is this causal?** No. Returns and credit spreads are determined at the same time, and both respond to news. The regression measures association. Say so, and do not write "the effect of" without an identification strategy. If you need causal claims, see Chapter 13.
- **Why these variables?** Justify each from theory or earlier papers, not from what turned out significant.
- **Are the standard errors right?** State the type and the lag length.
- **Does one episode drive the result?** Show the table without the crisis.
- **Is the relationship stable?** Report subsamples, and mention the CUSUM evidence.

Final checklist before you submit

1. Every variable is defined, with its unit and source.
2. Unit-root tests are reported, and they justify the choice of OLS in levels.
3. All diagnostic tests are reported, including the ones that failed, with what you did about them.
4. Every number in the text matches the tables.
5. The code runs from the raw data file to the final table without manual steps.

CHAPTER 4

VAR in First Differences

Applies when. All variables I(1), no cointegration
Overview. Use this chapter when every variable has a unit root and the Johansen test finds no long-run relationship. It covers differencing, lag selection, stability and residual tests, Granger causality, impulse responses and variance decompositions, with sample text for the paper.
Data. U.S. stock prices and industrial production, monthly, 1985–2019
Software. Python · statsmodels

4.1 The question and the model

The question is: **do stock prices and real activity help to predict each other?** Finance theory says stock prices are forward-looking and should lead output; a feedback from output to prices is also possible.

A vector autoregression (VAR) treats every variable as endogenous. Each variable is regressed on its own lags and the lags of the others, so you do not have to decide in advance which one is "the" dependent variable.

The decision rules of Chapter 1 found that the log S&P 500 and log industrial production are both I(1) and are **not cointegrated**. With no long-run relationship to preserve, the correct model is a VAR in first differences:

$$\Delta y_t = c + A_1 \Delta y_{t-1} + \cdots + A_p \Delta y_{t-p} + u_t, \qquad y_t = \begin{pmatrix} \log SP500_t \\ \log IP_t \end{pmatrix}$$

The first difference of a log, multiplied by 100, is a percentage growth rate, so the VAR is in monthly stock returns and monthly industrial production growth.

Why not estimate the VAR in levels?

With unit roots and no cointegration, a regression in levels is a spurious regression: the usual t- and F-tests are not valid. If the series *were* cointegrated, differencing would throw away the long-run relationship, and you would need Chapter 5 instead. This is why the cointegration test comes before the model.

How to write it: model section

Augmented Dickey–Fuller tests indicate that the log S&P 500 index and log industrial production are integrated of order one, and the Johansen trace test finds no cointegrating relationship between them (trace statistic 4.74 against a 5% critical value of 15.49). We therefore model the joint dynamics of the two series with a vector autoregression in first differences, $\Delta y_t = c + \sum_{i=1}^{p} A_i \Delta y_{t-i} + u_t$, where Δy_t contains the monthly stock return and the monthly growth rate of industrial production, both in percent.

4.2 Load the data and take first differences

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller

levels = pd.read_csv("prices.csv", parse_dates=["date"]).set_index("date")
```

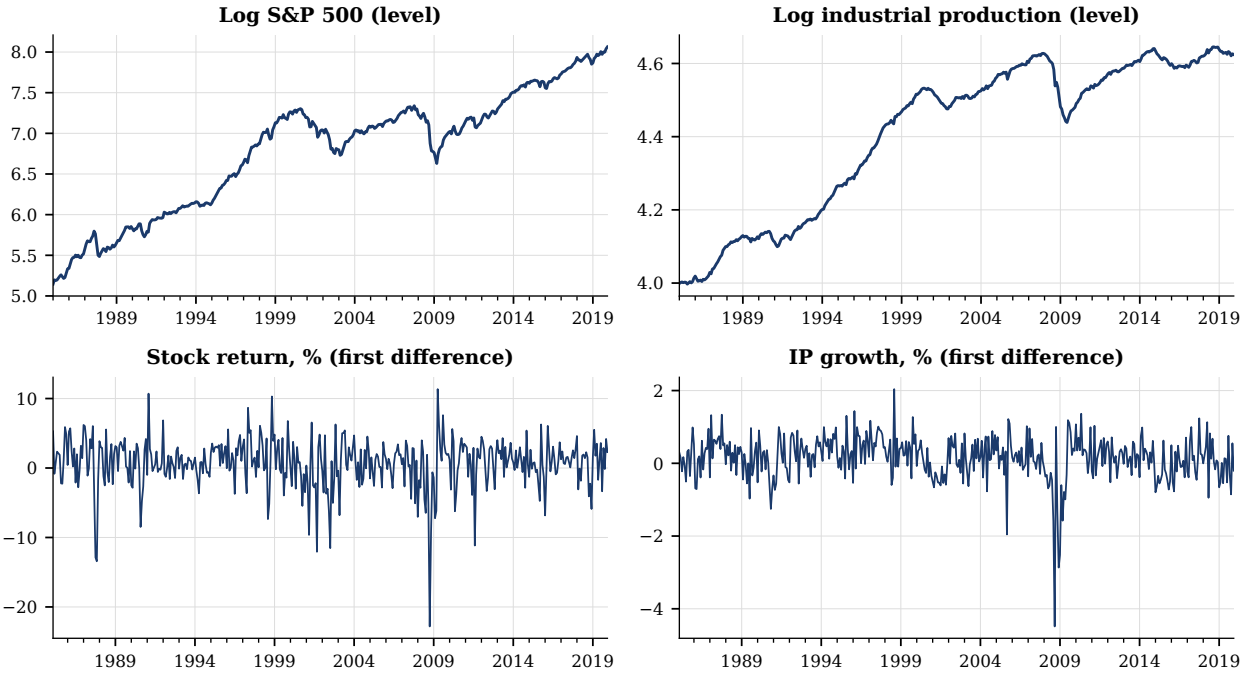


Figure 4.1. Load the data and take first differences. Panels: Log S&P 500 (level); Log industrial production (level); Stock return, % (first difference); IP growth, % (first difference).

```
levels.index.freq = "MS"
d = (100 * levels.diff().dropna()).rename(columns={"log_sp500": "ret", "log_indpro": "ipg"})

for name, data in (("levels", levels), ("first differences", d)):
    for col in data:
        print(f"{name:18s} {col:11s} ADF p-value = {adfuller(data[col], autolag='AIC')[1]:.3f}")
print(d.describe().T[["mean", "std", "min", "max"]].round(3))
```

```
levels             log_sp500   ADF p-value = 0.654
levels             log_indpro  ADF p-value = 0.303
first differences  ret         ADF p-value = 0.000
first differences  ipg         ADF p-value = 0.000
      mean    std     min     max
ret  0.697  3.537 -22.804  11.353
ipg  0.149  0.609  -4.485   2.039
```

```
fig, ax = plt.subplots(2, 2, figsize=(8, 4.4))
levels["log_sp500"].plot(ax=ax[0, 0], title="Log S&P 500 (level)"); levels["log_indpro"].plot(ax=ax[0, 1],
  ↪ title="Log industrial production (level)")
d["ret"].plot(ax=ax[1, 0], lw=.8, title="Stock return, % (first difference)"); d["ipg"].plot(ax=ax[1, 1], lw
  ↪ =.8, title="IP growth, % (first difference)")
for a in ax.ravel(): a.set_xlabel("")
plt.tight_layout()
```

How to read it

In levels the ADF p-values are far above 0.05: both series have a unit root. After differencing they are 0.000: the growth rates are stationary. The plots show the same thing. The levels trend and wander; the differences move around a constant mean.

One observation is lost by differencing, leaving 419 months.

4.3 Choose the lag order

The lag order p is chosen with information criteria. Each one trades off fit against the number of parameters; the lowest value wins.

```
selection = VAR(d).select_order(maxlags=12)
```

```
print(selection.summary())
print("Chosen lag order:", selection.selected_orders)
```

```
 VAR Order Selection (* highlights the minimums)
=================================================
       AIC         BIC         FPE         HQIC
-------------------------------------------------
0        1.567       1.586       4.791       1.575
1        1.425       1.484       4.159       1.449
2        1.294       1.392       3.647       1.333
3        1.218      1.356*       3.380      1.272*
4       1.210*       1.387      3.353*       1.280
5        1.227       1.443       3.409       1.312
6        1.223       1.479       3.397       1.324
7        1.232       1.527       3.428       1.349
8        1.249       1.584       3.486       1.381
9        1.264       1.638       3.539       1.412
10       1.280       1.693       3.596       1.443
11       1.293       1.746       3.643       1.472
12       1.294       1.787       3.649       1.489
-------------------------------------------------
Chosen lag order: {'aic': np.int64(4), 'bic': np.int64(3), 'hqic': np.int64(3), 'fpe': np.int64(4)}
```

How to read it

The star marks the minimum of each column. AIC and FPE choose 4 lags; BIC and HQIC choose 3. The criteria often disagree, because BIC penalises extra parameters more heavily.

A sound rule: start from the AIC choice, check that the residuals are free of serial correlation (Section 4.5), and use the BIC choice as a robustness check. Too few lags leave autocorrelation in the residuals, which invalidates the tests. A few too many lags only costs some precision.

4.4 Estimate the VAR

```
p = selection.aic
var = VAR(d).fit(p)
print(var.summary())
```

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Thu, 12, Nov, 2026
Time:                     08:37:37
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                    1.36962
Nobs:                     415.000    HQIC:                   1.26399
Log likelihood:          -1407.66    FPE:                    3.30327
AIC:                      1.19490    Det(Omega_mle):         3.16453
--------------------------------------------------------------------
Results for equation ret
==============================================================================
            coefficient       std. error           t-stat            prob
------------------------------------------------------------------------------
const          0.425327         0.180532            2.356           0.018
L1.ret         0.265050         0.049597            5.344           0.000
L1.ipg         0.793838         0.314812            2.522           0.012
L2.ret        -0.059726         0.051590           -1.158           0.247
L2.ipg         1.003538         0.302136            3.321           0.001
L3.ret        -0.022126         0.052038           -0.425           0.671
L3.ipg        -0.115577         0.297573           -0.388           0.698
L4.ret        -0.047959         0.052253           -0.918           0.359
L4.ipg        -0.558341         0.303367           -1.840           0.066
==============================================================================

Results for equation ipg
==============================================================================
```

```
            coefficient       std. error           t-stat            prob
----------------------------------------------------------------------------
const          0.023815         0.028297            0.842           0.400
L1.ret         0.009193         0.007774            1.183           0.237
L1.ipg        -0.005987         0.049345           -0.121           0.903
L2.ret         0.029732         0.008087            3.677           0.000
L2.ipg         0.151728         0.047358            3.204           0.001
L3.ret         0.037858         0.008157            4.641           0.000
L3.ipg         0.220567         0.046643            4.729           0.000
L4.ret        -0.001764         0.008190           -0.215           0.829
L4.ipg         0.132062         0.047551            2.777           0.005
============================================================================

Correlation matrix of residuals
            ret       ipg
ret    1.000000 -0.081456
ipg   -0.081456  1.000000
```

How to read it

There is one equation per variable. In the `ipg` equation, the second and third lags of the stock return are positive and highly significant: a rise in stock prices is followed by faster industrial production growth two to three months later. In the `ret` equation, the first two lags of industrial production growth are also significant.

Individual VAR coefficients are hard to interpret because the lags interact. Do not build your results section on them. Papers report three summaries instead: Granger causality tests, impulse responses and variance decompositions, which follow below. Most papers put the coefficient table in an appendix.

The correlation of the residuals (–0.08) is small. That will matter in Section 4.7.

4.5 Check stability and the residuals

A VAR is only usable if it is stable and its residuals are free of serial correlation.

```
print("Stable:", var.is_stable())
print("Moduli of the companion-matrix eigenvalues:", np.sort(1 / np.abs(var.roots))[::-1].round(3))

for h in (12, 24):
    test = var.test_whiteness(nlags=h)
    print(f"Portmanteau test, {h} lags: statistic = {test.test_statistic:.2f}   p-value = {test.pvalue:.3f}")

normality = var.test_normality()
print(f"Jarque-Bera (multivariate): statistic = {normality.test_statistic:.1f}   p-value = {normality.pvalue
  ↪ :.3f}")

from statsmodels.stats.diagnostic import het_arch
for col in var.resid:
    print(f"ARCH test, {col} equation: p-value = {het_arch(var.resid[col], nlags=6)[1]:.3f}")
```

```
Stable: True
Moduli of the companion-matrix eigenvalues: [0.833 0.593 0.593 0.54  0.535 0.535 0.401 0.401]
Portmanteau test, 12 lags: statistic = 28.27   p-value = 0.656
Portmanteau test, 24 lags: statistic = 66.94   p-value = 0.851
Jarque-Bera (multivariate): statistic = 1543.7   p-value = 0.000
ARCH test, ret equation: p-value = 0.004
ARCH test, ipg equation: p-value = 0.009
```

How to read it

- **Stability.** All eigenvalues of the companion matrix have modulus below 1 (the largest is 0.83), so the VAR is stable: shocks die out, and impulse responses are meaningful. A modulus at or above 1 would mean the system is not stationary, and you should check the differencing and the lag order.
- **Serial correlation.** The portmanteau test has the null of no residual autocorrelation. The p-values of 0.66 and 0.85 do not reject it, so four lags are enough. If this test rejected, you would add lags.

- **Normality.** Rejected, as is usual with financial data and extreme months such as October 2008. Estimates stay consistent; with over 400 observations the asymptotic tests remain usable. Report the result.
- **ARCH.** Both equations show volatility clustering. This does not bias the coefficients but affects standard errors, which is a reason to treat borderline p-values with caution.

How to write it: specification and diagnostics

The lag length is set to four months, as selected by the Akaike information criterion; the Schwarz criterion selects three, and results with three lags are reported as a robustness check. The estimated VAR is stable, with all eigenvalues of the companion matrix inside the unit circle (largest modulus 0.83). A portmanteau test does not reject the null of no residual autocorrelation up to 12 lags ($p = 0.66$). Multivariate normality of the residuals is rejected, reflecting a small number of extreme observations, and the residuals show ARCH effects, as is common for monthly financial data.

4.6 Granger causality

Variable x Granger-causes y if past values of x improve the forecast of y beyond what the past of y alone provides. The null hypothesis is **no Granger causality**: all coefficients on the lags of x in the equation for y are zero.

```
tests = [("ipg", "ret", "Stock returns  ->  IP growth"), ("ret", "ipg", "IP growth  ->  stock returns")]
for caused, causing, label in tests:
    t = var.test_causality(caused, causing, kind="f")
    print(f"{label:30s} F = {t.test_statistic:6.2f}   p-value = {t.pvalue:.4f}")
```

```
Stock returns  ->  IP growth   F =  13.06   p-value = 0.0000
IP growth  ->  stock returns   F =   4.58   p-value = 0.0012
```

How to read it

Both nulls are rejected at the 1% level, so predictability runs in both directions. The evidence is much stronger from stock returns to industrial production (the larger F-statistic), which fits the view that stock prices look ahead.

"Granger causality" is about **prediction, not cause**. Stock prices may rise because investors expect output to grow; that does not mean higher prices cause the growth. Use the phrase "Granger-causes" or "helps to predict", never "causes".

4.7 Impulse responses

An impulse response shows how each variable reacts over time to a one-time shock in one of them. Because the residuals of the two equations are correlated, the shocks must be separated first. The standard method is the Cholesky decomposition, which requires an **ordering**: the first variable can affect the second within the same month, but not the reverse.

We order stock returns first. The justification is practical: the S&P 500 series is an average over the month, while industrial production for a month is published only in the middle of the next month, so stock prices cannot react to it within the month.

```
irf = var.irf(24)
fig = irf.plot(orth=True, figsize=(8, 5.2), signif=0.05)
fig.suptitle("Orthogonalised impulse responses with 95% bands", fontsize=10, fontweight="bold")
plt.tight_layout()
cum = irf.orth_cum_effects
print("Cumulative response to a one-standard-deviation shock (percent)")
print(f"{'months':>7s}{'ret -> ret':>12s}{'ret -> ipg':>12s}{'ipg -> ret':>12s}{'ipg -> ipg':>12s}")
for h in (1, 3, 6, 12, 24):
    print(f"{h:7d}{cum[h, 0, 0]:12.2f}{cum[h, 1, 0]:12.2f}{cum[h, 0, 1]:12.2f}{cum[h, 1, 1]:12.2f}")
print("Size of a one-s.d. shock: ret", irf.orth_irfs[0, 0, 0].round(2), " ipg", irf.orth_irfs[0, 1, 1].round(2)
  ↪ )
```

```
Cumulative response to a one-standard-deviation shock (percent)
 months  ret -> ret  ret -> ipg  ipg -> ret  ipg -> ipg
      1        4.23       -0.01        0.42        0.52
```

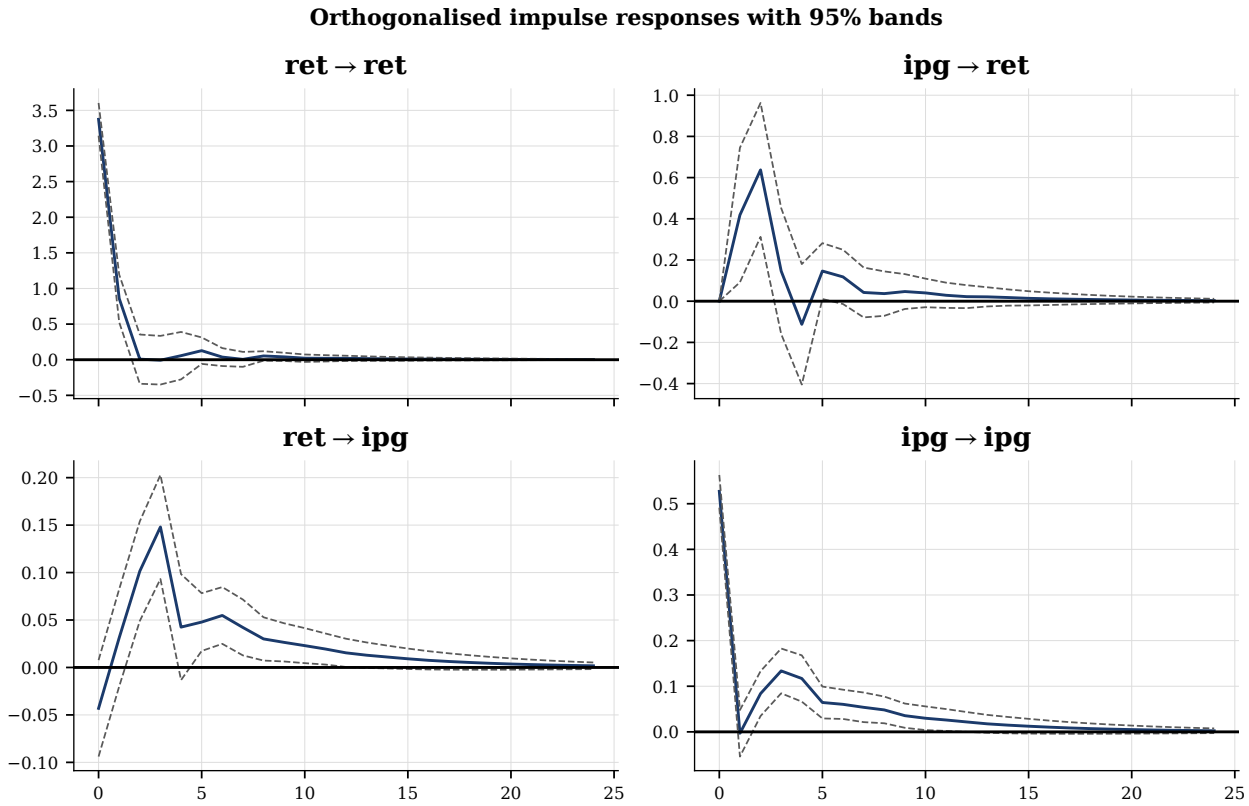


Figure 4.2. Impulse responses: Orthogonalised impulse responses with 95% bands.

```
     3        4.24        0.24        1.20        0.74
     6        4.46        0.38        1.35        0.98
    12        4.61        0.54        1.57        1.20
    24        4.69        0.61        1.68        1.29
Size of a one-s.d. shock: ret 3.37  ipg 0.53
```

How to read it

Each panel shows the response of one variable (named second in the panel title) to a shock in another. The dashed lines are 95% confidence bands: a response is statistically different from zero only where the band excludes zero.

- A stock-return shock of one standard deviation (3.4%) is followed by significantly higher industrial production growth two to three months later. The **cumulative** effect, which is the effect on the *level* of industrial production because the VAR is in growth rates, builds to about 0.6% after two years.
- An industrial-production shock (0.5%) raises stock returns over the next two months, with a cumulative effect of about 1.7%.
- All responses return to zero, as they must in a stable VAR.

Because the model is in first differences, always say whether you are describing the response of the growth rate or the cumulative response of the level. Readers usually care about the level.

4.8 Variance decomposition

The forecast error variance decomposition (FEVD) answers a different question: of the unpredictable movements in each variable, what share is due to each shock?

```
fevd = var.fevd(24)
print("Share of forecast error variance (%)")
print(f"{'horizon':>8s}{'ret: own':>10s}{'ret: ipg':>10s}{'ipg: ret':>10s}{'ipg: own':>10s}")
for h in (1, 6, 12, 24):
    s = 100 * fevd.decomp[:, h - 1, :]
    print(f"{h:8d}{s[0, 0]:10.1f}{s[0, 1]:10.1f}{s[1, 0]:10.1f}{s[1, 1]:10.1f}")
```

```
Share of forecast error variance (%)
 horizon  ret: own  ret: ipg  ipg: ret  ipg: own
       1     100.0       0.0       0.7      99.3
       6      95.0       5.0      10.9      89.1
      12      94.9       5.1      12.3      87.7
      24      94.8       5.2      12.4      87.6
```

How to read it

At a two-year horizon, stock-return shocks account for about 12% of the forecast error variance of industrial production growth, up from under 1% at one month. Industrial production shocks account for about 5% of the variance of stock returns. Each variable is mostly driven by its own shocks, which is normal; what matters is that the cross shares are not zero and that the share of stock shocks in output grows with the horizon.

Statistically significant Granger causality can go with a small variance share. Report both: one says whether the link exists, the other how much it matters.

4.9 Robustness checks

The three choices a referee will question are the ordering, the lag length and the sample.

```python
def summary(data, lags, order):
    res = VAR(data[order]).fit(lags)
    f = res.irf(24)
    i_ret, i_ip = order.index("ret"), order.index("ipg")
    share = 100 * res.fevd(24).decomp[i_ip, 23, i_ret]
    return (res.test_causality("ipg", "ret", kind="f").pvalue, res.test_causality("ret", "ipg", kind="f").
 ↪ pvalue,
            f.orth_cum_effects[24, i_ip, i_ret], share)

cases = {
    "Baseline (4 lags, ret first)": (d, 4, ["ret", "ipg"]),
    "Reverse ordering (ipg first)": (d, 4, ["ipg", "ret"]),
    "3 lags (BIC)":                 (d, 3, ["ret", "ipg"]),
    "8 lags":                       (d, 8, ["ret", "ipg"]),
    "Sample 1985-2007":             (d[:"2007-12"], 4, ["ret", "ipg"]),
    "Sample 2010-2019":             (d["2010-01":], 4, ["ret", "ipg"]),
}
print(f"{'':30s}{'GC p: ret->ipg':>15s}{'GC p: ipg->ret':>15s}{'cum. IP resp.':>14s}{'FEVD share %':>13s}")
for name, (data, lags, order) in cases.items():
    a, b, c, s = summary(data, lags, order)
    print(f"{name:30s}{a:15.4f}{b:15.4f}{c:14.2f}{s:13.1f}")
```

```
                              GC p: ret->ipg GC p: ipg->ret cum. IP resp. FEVD share %
Baseline (4 lags, ret first)          0.0000         0.0012          0.61         12.4
Reverse ordering (ipg first)          0.0000         0.0012          0.71         13.9
3 lags (BIC)                          0.0000         0.0022          0.60         13.6
8 lags                                0.0000         0.0337          0.69         14.7
Sample 1985-2007                      0.0019         0.6330          0.26          6.7
Sample 2010-2019                      0.2128         0.7586          0.09          5.9
```

How to read it

This table changes the conclusions, which is exactly what a robustness table is for.

- **Ordering and lag length do not matter.** Reversing the ordering, or using 3 or 8 lags, leaves the tests and the size of the responses almost unchanged. The ordering is harmless here because the residual correlation is only –0.08.
- **The sample matters a great deal.** Before 2008, stock returns still Granger-cause industrial production ($p = 0.002$), but the cumulative response is less than half as large (0.26% against 0.61%). The reverse channel, from industrial production to stock returns, disappears completely ($p = 0.63$).
- **After 2010 neither test rejects.** With only 120 months the tests have low power, so this is weak evidence of no relationship, but it is certainly not evidence for one.

The honest summary: stock returns lead industrial production, with an effect that the 2008–09 crisis roughly doubles; the feedback from production to returns is a crisis-period result only. A paper that reported only the full-sample numbers would overstate both findings. Always run the subsample check before you write the results section.

4.10 Write the paper

Section	What goes in it	From which step
Data	Sources, sample, transformations, unit-root and cointegration tests	Section 4.2 and the decision rules of Chapter 1
Method	The VAR equation, why first differences, lag selection, identification (ordering)	Sections 4.1, 4.3 and 4.7
Results	Granger causality table, impulse-response figure, variance decomposition table	Sections 4.6 to 4.8
Robustness	Diagnostics, alternative ordering, lag length, subsamples	Sections 4.5 and 4.9
Appendix	Full coefficient estimates	Section 4.4

How to write it: results

Table 3 reports Granger causality tests. In the full sample, the null that stock returns do not Granger-cause industrial production growth is rejected at the 1% level ($F = 13.06$, $p < 0.01$), as is the reverse null ($F = 4.58$, $p < 0.01$). Figure 2 shows the orthogonalised impulse responses. A one-standard-deviation shock to stock returns (3.4%) is followed by a statistically significant rise in industrial production growth two to three months later, and raises the level of industrial production by 0.6% after two years. Stock-return shocks account for 12% of the forecast error variance of industrial production growth at the 24-month horizon.

These full-sample results are influenced by the 2008–09 financial crisis. In the 1985–2007 subsample, stock returns continue to Granger-cause industrial production growth ($p < 0.01$), but the cumulative response falls to 0.3%, and there is no evidence of predictability running from industrial production to stock returns ($p = 0.63$). We conclude that stock prices are a leading indicator of real activity, while the apparent feedback from activity to stock prices reflects the crisis period.

What a referee will ask

- **Why this ordering?** Give an economic or institutional reason, and show the reverse ordering.
- **Why differences and not levels?** Cite your unit-root and cointegration tests.
- **Is the lag length arbitrary?** Report the criteria and a different lag length.
- **Is this causal?** Granger causality is predictive. Structural claims need identifying restrictions that you must defend.
- **Is it stable over time?** Show subsamples, especially around 2008.
- **Only two variables?** Omitted variables, such as interest rates, can create or hide Granger causality. Say so, or add them.

CHAPTER 5

Vector Error-Correction Model

Applies when. All variables I(1), cointegrated
Overview. Use this chapter when every variable has a unit root and the Johansen test finds a long-run relationship. It covers the lag order and deterministic terms, the cointegration rank, the long-run vector, adjustment speeds and weak exogeneity, short-run causality, diagnostics and robustness, with sample text for the paper.
Data. U.S. 10-year and 1-year Treasury yields, quarterly, 1960–2019
Software. Python · statsmodels

5.1 The question and the model

The question is: **are short and long interest rates tied together in the long run, and which one adjusts when they drift apart?** The expectations hypothesis of the term structure says the long rate is an average of expected future short rates plus a premium, so the two should not move apart permanently.

The decision rules of Chapter 1 found that both yields are I(1) and **cointegrated**. A vector error-correction model (VECM) is a VAR in first differences with one extra term, the deviation from the long-run relationship in the previous period:

$$\Delta y_t = \alpha\,(\beta' y_{t-1} + \rho) + \Gamma_1 \Delta y_{t-1} + \cdots + \Gamma_k \Delta y_{t-k} + u_t$$

Part	Name	What it tells you
$\beta' y_{t-1} + \rho$	error-correction term	how far the system was from its long-run relationship last period
β	cointegrating vector	the **long-run** relationship between the variables
α	adjustment (loading) coefficients	how fast each variable moves to close the gap
Γ_i	short-run coefficients	the **short-run** dynamics

Why a VECM and not a VAR in differences?

A VAR in differences uses only changes, so it ignores the information that the two rates are currently far apart or close together. When the variables are cointegrated, that gap predicts future changes, and leaving it out is a misspecification. The VECM is the model that keeps both the long-run relationship and the short-run dynamics.

How to write it: model section

Both yields are integrated of order one, and the Johansen procedure indicates one cointegrating relationship between them. We therefore estimate the vector error-correction model $\Delta y_t = \alpha(\beta' y_{t-1} + \rho) + \sum_{i=1}^{k} \Gamma_i \Delta y_{t-i} + u_t$, *where* y_t *contains the 10-year and 1-year Treasury yields,* β *is the cointegrating vector describing their long-run relationship, and* α *contains the adjustment coefficients that measure how quickly each yield responds to deviations from that relationship.*

5.2 Load the data and look at the long-run relationship

```
import pandas as pd
import numpy as np
```

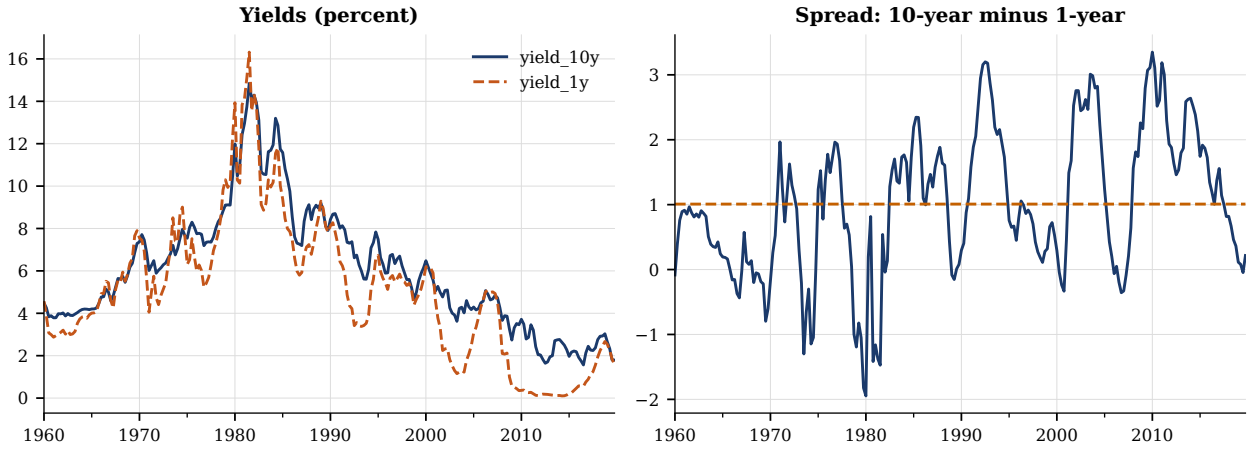


Figure 5.1. Load the data and look at the long-run relationship. Panels: Yields (percent); Spread: 10-year minus 1-year.

```
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.vector_ar.vecm import VECM, select_order, select_coint_rank

df = pd.read_csv("rates.csv").set_index("date")
df.index = pd.PeriodIndex(df.index, freq="Q").to_timestamp()
df.index.freq = "QS"

for col in df:
    print(f"{col:10s} level: ADF p = {adfuller(df[col], autolag='AIC')[1]:.3f}    "
          f"first difference: ADF p = {adfuller(df[col].diff().dropna(), autolag='AIC')[1]:.3f}")
spread = df["yield_10y"] - df["yield_1y"]
print(f"spread     level: ADF p = {adfuller(spread, autolag='AIC')[1]:.3f}    mean = {spread.mean():.2f}")
```

```
yield_10y  level: ADF p = 0.812    first difference: ADF p = 0.000
yield_1y   level: ADF p = 0.531    first difference: ADF p = 0.000
spread     level: ADF p = 0.002    mean = 1.01
```

```
fig, ax = plt.subplots(1, 2, figsize=(8, 3))
df.plot(ax=ax[0], title="Yields (percent)"); ax[0].set_xlabel("")
spread.plot(ax=ax[1], title="Spread: 10-year minus 1-year"); ax[1].axhline(spread.mean(), ls="--", color="#
  ↪ c46000"); ax[1].set_xlabel("")
plt.tight_layout()
```

How to read it

Each yield has a unit root in levels (p-values of 0.81 and 0.53) and is stationary in first differences: both are I(1). But the **difference between them is stationary** (p below 0.01). That is cointegration in its simplest form: two series that wander, but wander together.

The plot makes this visible. Both yields rose until the early 1980s and fell for decades afterwards; the spread kept returning to its mean of about 1 percentage point. Always plot your candidate long-run relationship before testing it formally.

5.3 Choose the lag order and the deterministic terms

Two specification choices come before the cointegration test.

Lag order. `k_ar_diff` is the number of lagged differences in the VECM. It equals the lag order of the VAR in levels minus one.

Deterministic terms. This choice matters more than students expect, because the critical values of the Johansen test depend on it.

Option in statsmodels	Meaning	Use when
`"n"`	no constant	almost never
`"ci"`	constant inside the cointegrating relation	the series have no trend; the long-run relation has a non-zero mean. **Interest rates, spreads, ratios.**
`"co"`	constant outside the relation	the series have linear trends (log GDP, log prices)
`"cili"`	trend inside the relation	the long-run relation itself trends

Interest rates have no deterministic trend, and their spread has a positive mean, so `"ci"` is the right choice here.

```
order = select_order(df, maxlags=8, deterministic="ci")
print(order.summary())
print("Chosen k_ar_diff:", order.selected_orders)
```

```
 VECM Order Selection (* highlights the minimums)
================================================
      AIC         BIC         FPE         HQIC
------------------------------------------------
0      -3.190      -3.070     0.04119      -3.141
1      -3.270     -3.091*     0.03801      -3.198
2      -3.282      -3.044     0.03755      -3.186
3      -3.351      -3.053     0.03504      -3.231
4      -3.340      -2.982     0.03545      -3.196
5      -3.431      -3.014     0.03235     -3.263*
6      -3.428      -2.951     0.03247      -3.236
7     -3.464*      -2.928    0.03132*      -3.248
8      -3.456      -2.860     0.03159      -3.215
------------------------------------------------
Chosen k_ar_diff: {'aic': np.int64(7), 'bic': np.int64(1), 'hqic': np.int64(5), 'fpe': np.int64(7)}
```

How to read it

AIC chooses 7 lagged differences, HQIC 5 and BIC only 1. As in any VAR, the choice is settled by the residuals: Section 5.8 shows that with 1 lag the residuals are strongly autocorrelated, so the BIC choice is too short. We use 7 and check 5 as robustness.

5.4 Test the cointegration rank

The rank r is the number of independent long-run relationships. With two variables it can be 0 (no cointegration: use a VAR in differences), 1 (one long-run relation: use a VECM), or 2 (both series are actually stationary: use a VAR in levels).

```
k = int(order.aic)
for method in ("trace", "maxeig"):
    rank = select_coint_rank(df, det_order=0, k_ar_diff=k, method=method, signif=0.05)
    print(f"{method} test  ->  rank = {rank.rank}")
    for r, (stat, crit) in enumerate(zip(rank.test_stats, rank.crit_vals)):
        print(f"    H0: rank <= {r}   statistic = {stat:6.2f}   5% critical value = {crit:6.2f}   {'reject' if
  ↪ stat > crit else 'do not reject'}")
```

```
trace test  ->  rank = 1
    H0: rank <= 0   statistic =  29.39   5% critical value =  15.49   reject
    H0: rank <= 1   statistic =   1.07   5% critical value =   3.84   do not reject
maxeig test  ->  rank = 1
    H0: rank <= 0   statistic =  28.33   5% critical value =  14.26   reject
    H0: rank <= 1   statistic =   1.07   5% critical value =   3.84   do not reject
```

How to read it

Read each test from the top. "Rank ≤ 0" is rejected (29.39 is above 15.49), so there is at least one cointegrating relation. "Rank ≤ 1" is not rejected (1.07 is below 3.84), so there is exactly one. The trace test and the

maximum-eigenvalue test agree, which makes the conclusion solid. When they disagree, most authors follow the trace test and say so.

5.5 Estimate the VECM

```
vecm = VECM(df, k_ar_diff=k, coint_rank=1, deterministic="ci").fit()

beta = vecm.beta[:, 0]; const = vecm.det_coef_coint[0, 0]
print("Cointegrating vector (normalised on yield_10y)")
print(f"   yield_10y: {beta[0]:.3f}")
print(f"   yield_1y : {beta[1]:.3f}   (s.e. {vecm.stderr_beta[1, 0]:.3f})")
print(f"   constant : {const:.3f}   (s.e. {vecm.stderr_det_coef_coint[0, 0]:.3f})")
print(f"\nLong-run relation:  yield_10y = {-const:.2f} + {-beta[1]:.3f} * yield_1y")

print("\nAdjustment coefficients (alpha)")
for i, col in enumerate(df.columns):
    print(f"   {col:10s} alpha = {vecm.alpha[i, 0]:7.3f}   s.e. = {vecm.stderr_alpha[i, 0]:.3f}   "
          f"t = {vecm.tvalues_alpha[i, 0]:6.2f}   p-value = {vecm.pvalues_alpha[i, 0]:.3f}")
```

```
Cointegrating vector (normalised on yield_10y)
   yield_10y: 1.000
   yield_1y : -0.936   (s.e. 0.041)
   constant : -1.310   (s.e. 0.245)

Long-run relation:  yield_10y = 1.31 + 0.936 * yield_1y

Adjustment coefficients (alpha)
   yield_10y  alpha =  -0.111   s.e. = 0.040   t =  -2.78   p-value = 0.005
   yield_1y   alpha =   0.027   s.e. = 0.060   t =   0.46   p-value = 0.647
```

How to read the long-run relation

The software reports the vector as $\beta' y + \rho = 0$. Moving everything except the first variable to the right-hand side **changes the signs**: a coefficient of –0.936 on the 1-year yield means the long-run relation is

$$\textbf{yield_10y} = \mathbf{1.31} + \mathbf{0.936} \times \textbf{yield_1y}$$

In the long run, a 1-point rise in the 1-year yield goes with a 0.94-point rise in the 10-year yield, and the constant of 1.31 is the average term premium. Forgetting the sign change is the most common mistake in student papers that use a VECM.

5.6 Test restrictions on the long-run relation

Theory often predicts a specific value. The expectations hypothesis implies that the long-run coefficient is exactly 1, which would make the simple spread stationary.

```
b, se = -vecm.beta[1, 0], vecm.stderr_beta[1, 0]
t = (b - 1) / se
from scipy import stats
print(f"H0: long-run coefficient = 1    estimate = {b:.3f}   t = {t:.2f}   p-value = {2 * (1 - stats.norm.cdf(
  ↪ abs(t))):.3f}")
```

```
H0: long-run coefficient = 1    estimate = 0.936   t = -1.56   p-value = 0.119
```

How to read it

The estimate of 0.936 is not significantly different from 1 ($p = 0.12$). The data are consistent with a one-for-one long-run relationship, which agrees with the stationary spread found in Section 5.2. Testing the value that theory predicts turns a statistical result into an economic one.

5.7 Adjustment speeds and weak exogeneity

The adjustment coefficients show **which variable does the correcting**.

```
a = vecm.alpha[0, 0]
print(f"10-year yield closes {abs(a) * 100:.1f}% of the gap each quarter")
```

```
print(f"Half-life of a deviation: {np.log(0.5) / np.log(1 + a):.1f} quarters")
print(f"Weak exogeneity of yield_1y (H0: its alpha = 0): p-value = {vecm.pvalues_alpha[1, 0]:.3f}")
```

```
10-year yield closes 11.1% of the gap each quarter
Half-life of a deviation: 5.9 quarters
Weak exogeneity of yield_1y (H0: its alpha = 0): p-value = 0.647
```

How to read it

- **Sign.** For the variable the vector is normalised on, the adjustment coefficient should be **negative**: when the 10-year yield is above its long-run value, it falls. Here it is –0.111 and significant, so the error-correction mechanism works. A positive or insignificant coefficient on every variable would mean there is no adjustment, which contradicts cointegration.
- **Speed.** The 10-year yield closes about 11% of any gap per quarter, so half of a deviation is gone after roughly 6 quarters.
- **Weak exogeneity.** The coefficient for the 1-year yield is small and insignificant (p = 0.65). The short rate does not respond to the gap: it is *weakly exogenous*. In economic terms, the short rate, which follows monetary policy, leads, and the long rate adjusts to it.

How to write it: long-run results

The estimated cointegrating relationship is $y_t^{10} = 1.31 + 0.936\, y_t^{1}$, with a standard error of 0.041 on the slope. The hypothesis of a unit long-run coefficient cannot be rejected in the system estimate ($p = 0.12$), consistent with the expectations hypothesis and a stationary term spread, although dynamic OLS yields a smaller coefficient of 0.84. The adjustment coefficient in the 10-year yield equation is –0.111 ($t = -2.78$), implying that about 11% of a deviation from the long-run relationship is corrected each quarter, with a half-life of roughly six quarters. The adjustment coefficient in the 1-year yield equation is statistically insignificant ($p = 0.65$), so the short rate is weakly exogenous: the long rate adjusts to the short rate rather than the reverse.

5.8 Short-run causality and diagnostic tests

```
for caused, causing in (("yield_10y", "yield_1y"), ("yield_1y", "yield_10y")):
    g = vecm.test_granger_causality(caused=caused, causing=causing)
    print(f"Granger causality {causing} -> {caused}:  F = {g.test_statistic:.2f}   p-value = {g.pvalue:.3f}")

white = vecm.test_whiteness(nlags=12)
print(f"\nPortmanteau test (12 lags): statistic = {white.test_statistic:.2f}   p-value = {white.pvalue:.3f}")
print(f"Normality (Jarque-Bera):    p-value = {vecm.test_normality().pvalue:.3f}")

short = VECM(df, k_ar_diff=1, coint_rank=1, deterministic="ci").fit()
print(f"\nWith only 1 lag (the BIC choice): portmanteau p-value = {short.test_whiteness(nlags=12).pvalue:.4f}")
```

```
Granger causality yield_1y -> yield_10y:  F = 2.62   p-value = 0.008
Granger causality yield_10y -> yield_1y:  F = 1.66   p-value = 0.106

Portmanteau test (12 lags): statistic = 14.87   p-value = 0.671
Normality (Jarque-Bera):    p-value = 0.000

With only 1 lag (the BIC choice): portmanteau p-value = 0.0001
```

How to read it

- **Causality.** In a VECM there are two channels. *Long-run* causality runs through the error-correction term (Section 5.7). *Short-run* Granger causality runs through the lagged differences, tested here. The 1-year yield Granger-causes the 10-year yield (p below 0.01); the reverse is not significant at 5%. Both channels point the same way: from the short rate to the long rate.
- **Serial correlation.** The portmanteau test does not reject (p = 0.67): 7 lags are enough. With the single lag that BIC preferred, the same test rejects strongly, which is why we did not use it.
- **Normality.** Rejected, because of the extreme interest-rate movements of 1979–82. The Johansen procedure is robust to non-normality, in particular to excess kurtosis, so this does not overturn the results. Report it.

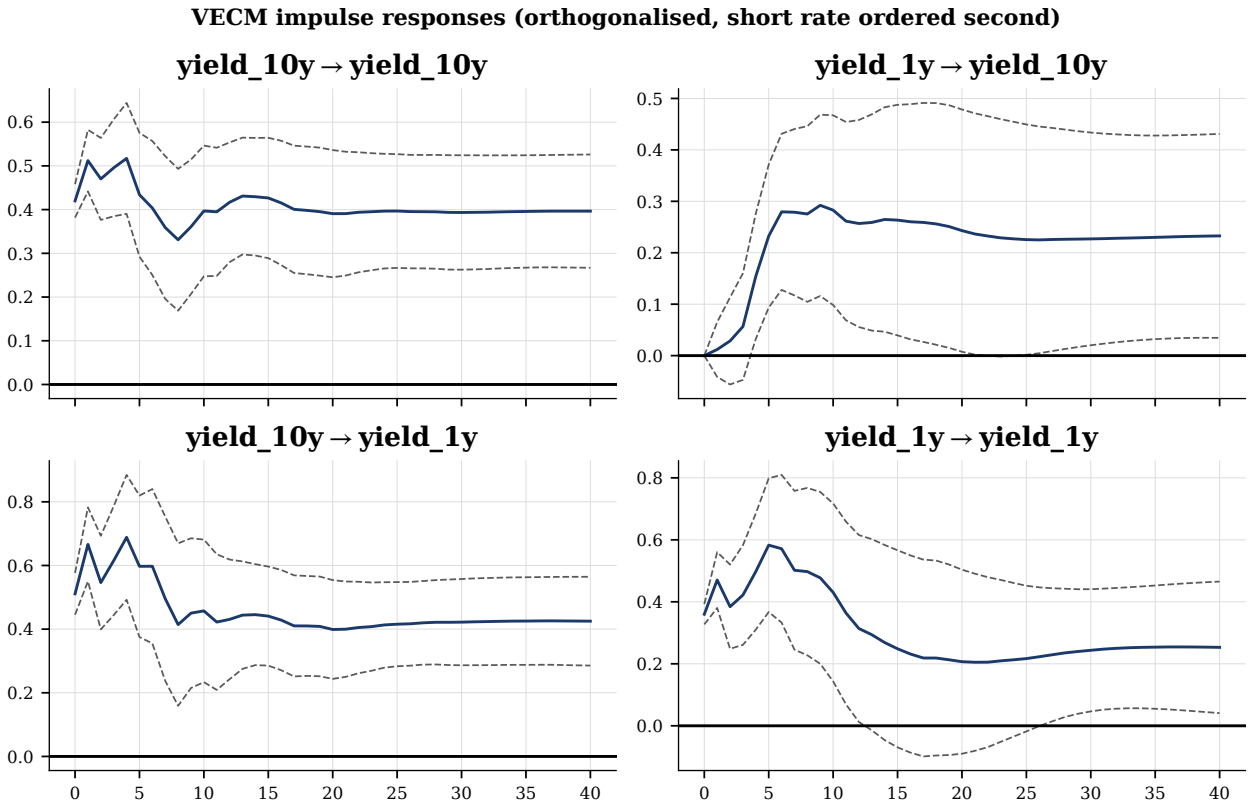


Figure 5.2. Impulse responses: VECM impulse responses (orthogonalised, short rate ordered second).

5.9 Impulse responses

```
irf = vecm.irf(40)
fig = irf.plot(orth=True, figsize=(8, 5.2))
fig.suptitle("VECM impulse responses (orthogonalised, short rate ordered second)", fontsize=10, fontweight="
  ↪ bold")
plt.tight_layout()
```

> **How to read it**
>
> VECM impulse responses differ from those of a stationary VAR in one important way: **they do not return to zero**. The variables are I(1), so a shock moves their levels permanently. What cointegration guarantees is that the two responses settle at levels that respect the long-run relationship, so the gap between them closes even though each yield ends up at a new level.

5.10 Robustness checks

```
def long_run(data, lags):
    m = VECM(data, k_ar_diff=lags, coint_rank=1, deterministic="ci").fit()
    return -m.beta[1, 0], m.stderr_beta[1, 0], m.alpha[0, 0], m.pvalues_alpha[0, 0], m.pvalues_alpha[1, 0]

cases = {"Baseline (7 lags)": (df, 7), "5 lags (HQIC)": (df, 5), "1960-1989": (df[:"1989-12"], 7), "1990-2019":
  ↪  (df["1990-01":], 7)}
print(f"{'':20s}{'long-run coef.':>15s}{'s.e.':>8s}{'alpha 10y':>11s}{'p-value':>9s}{'p alpha 1y':>12s}")
for name, (data, lags) in cases.items():
    b_, se_, a_, pa, pa1 = long_run(data, lags)
    print(f"{name:20s}{b_:15.3f}{se_:8.3f}{a_:11.3f}{pa:9.3f}{pa1:12.3f}")

# A single-equation alternative: dynamic OLS (Stock and Watson) with two leads and lags
X = pd.DataFrame({"const": 1.0, "yield_1y": df["yield_1y"]})
for j in range(-2, 3):
```

```
    X[f"d_lead_lag_{j}"] = df["yield_1y"].diff().shift(-j)
dols = sm.OLS(df["yield_10y"], X, missing="drop").fit(cov_type="HAC", cov_kwds={"maxlags": 4})
print(f"\nDynamic OLS:        long-run coef. = {dols.params['yield_1y']:.3f}   s.e. = {dols.bse['yield_1y']:.3
  ↪ f}   constant = {dols.params['const']:.2f}")
```

```
                    long-run coef.    s.e.  alpha 10y  p-value  p alpha 1y
Baseline (7 lags)            0.936   0.041     -0.111    0.005       0.647
5 lags (HQIC)                0.947   0.042     -0.065    0.062       0.064
1960-1989                    1.018   0.039     -0.204    0.016       0.925
1990-2019                    0.871   0.078     -0.067    0.175       0.111

Dynamic OLS:        long-run coef. = 0.835   s.e. = 0.030   constant = 1.86
```

How to read it

- **The long-run coefficient is stable in the Johansen estimates.** It lies between 0.87 and 1.02 across lag lengths and subsamples, always within about two standard errors of 1.
- **The adjustment is less stable.** With 5 lags the 10-year adjustment coefficient is smaller and only significant at 10%, and weak exogeneity of the 1-year yield becomes borderline ($p = 0.06$). In 1990–2019 the adjustment coefficient is not significant at all ($p = 0.18$). Error correction was fast and clear before 1990 and is hard to detect afterwards, a period that includes seven years of short rates held near zero.
- **Dynamic OLS disagrees on the size.** It gives a long-run coefficient of 0.835 with a small standard error, which is significantly below 1. Single-equation and system estimators can differ in finite samples, and when they do the paper must say so.

What can be claimed: the two yields are cointegrated, the long-run coefficient is close to one, and the long rate does the adjusting. What cannot be claimed without qualification: that the coefficient is exactly one, or that the adjustment speed is constant over time.

5.11 Write the paper

Section	What goes in it	From which step
Data	Sources, sample, plots, unit-root tests	Section 5.2
Method	The VECM equation, lag order, deterministic terms and why	Sections 5.1 and 5.3
Cointegration	Trace and maximum-eigenvalue tests in one table	Section 5.4
Long-run results	The normalised vector, tests of theory-based restrictions	Sections 5.5 and 5.6
Short-run results	Adjustment coefficients, weak exogeneity, Granger causality, impulse responses	Sections 5.7 to 5.9
Robustness	Diagnostics, lag length, subsamples, a single-equation estimator	Sections 5.8 and 5.10

How to write it: cointegration test paragraph

Table 2 reports the Johansen cointegration tests, with seven lagged differences and a constant restricted to the cointegrating relation. The trace statistic rejects the null of no cointegration (29.39 against a 5% critical value of 15.49) but not the null of at most one cointegrating vector (1.07 against 3.84). The maximum-eigenvalue test gives the same conclusion. We therefore proceed with a cointegration rank of one.

What a referee will ask

- **Why these deterministic terms?** Justify the choice from the behaviour of the data; do not leave the software default unexplained.
- **Is the rank robust to the lag length?** Show the test for at least two lag lengths.
- **Did you read the signs correctly?** Write the long-run relation as an equation in the text.
- **Is the adjustment coefficient negative and significant?** If not, the error-correction story does not hold.
- **Is the long-run relation stable?** Report subsamples; long samples of interest rates span very different monetary regimes.

- **Does a different estimator agree?** Report DOLS or FMOLS next to the Johansen estimate.

CHAPTER 6

ARDL Bounds-Testing Model

Applies when. A mix of I(0) and I(1) variables
Overview. Use this chapter when your variables have different orders of integration and none is I(2). It covers lag selection, the bounds test for a long-run relationship, long-run coefficients, the error-correction form, diagnostic and stability tests and robustness, with sample text for the paper.
Data. U.S. 10-year Treasury yield, inflation, GDP growth and unemployment, quarterly, 1960–2019
Software. Python · statsmodels

6.1 The question and the model

The question is: **what determines the 10-year Treasury yield in the long run?** The Fisher relation says nominal yields should rise with inflation; the business cycle, measured by GDP growth and unemployment, may also matter.

The decision rules of Chapter 1 found a **mix**: the yield and inflation are I(1), while GDP growth and unemployment are I(0). This rules out the Johansen approach, which needs every variable to be I(1), and it rules out OLS in levels. The autoregressive distributed lag (ARDL) model of Pesaran, Shin and Smith (2001) is built for exactly this case. Its bounds test for a long-run relationship is valid whether the regressors are I(0), I(1) or a mixture.

The ARDL(p, q_1, q_2, q_3) model includes lags of the dependent variable and of each regressor:

$$y_t = c + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \sum_{j=1}^{3} \sum_{i=0}^{q_j} \theta_{j,i} \, x_{j,t-i} + \varepsilon_t$$

It can be rewritten in error-correction form, which separates the long run from the short run:

$$\Delta y_t = c + \rho \, y_{t-1} + \sum_{j} \delta_j x_{j,t-1} + (\text{lagged differences}) + \varepsilon_t, \qquad \text{long-run coefficient of } x_j = -\frac{\delta_j}{\rho}$$

The one condition you must check

ARDL allows I(0) and I(1) variables, but **no variable may be I(2)**. With an I(2) variable the critical values of the bounds test are not valid. The decision rules of Chapter 1 check this for you; in a paper, report unit-root tests on first differences to prove it.

How to write it: model section

Unit-root tests indicate that the 10-year yield and inflation are I(1) while GDP growth and the unemployment rate are I(0), and that no variable is I(2). Since the variables are integrated of different orders, we use the ARDL bounds-testing approach of Pesaran, Shin and Smith (2001), which remains valid for any mixture of I(0) and I(1) regressors. The model is estimated in its unrestricted error-correction form, $\Delta y_t = c + \rho y_{t-1} + \sum_j \delta_j x_{j,t-1} + \sum_i \gamma_i \Delta y_{t-i} + \sum_j \sum_i \lambda_{j,i} \Delta x_{j,t-i} + \varepsilon_t$*, where the long-run effect of* x_j *on the yield is* $-\delta_j/\rho$*.*

6.2 Load the data and confirm the orders of integration

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.ardl import ARDL, UECM, ardl_select_order

df = pd.read_csv("yield.csv").set_index("date")
df.index = pd.PeriodIndex(df.index, freq="Q").to_timestamp()
df.index.freq = "QS"
y = df["yield_10y"]
X = df[["inflation", "gdp_growth", "unemployment"]]

print(f"{'variable':14s}{'ADF p, level':>14s}{'ADF p, 1st diff.':>18s}{'order':>8s}")
for col in df:
    p0 = adfuller(df[col], autolag="AIC")[1]
    p1 = adfuller(df[col].diff().dropna(), autolag="AIC")[1]
    print(f"{col:14s}{p0:14.3f}{p1:18.3f}{'I(0)' if p0 < 0.05 else 'I(1)' if p1 < 0.05 else 'I(2)?':>8s}")
```

```
variable        ADF p, level  ADF p, 1st diff.   order
yield_10y              0.812             0.000    I(1)
inflation              0.190             0.000    I(1)
gdp_growth             0.000             0.000    I(0)
unemployment           0.045             0.000    I(0)
```

How to read it

Two variables are I(1) and two are I(0). No first difference has a unit root, so nothing is I(2) and ARDL can be used.

The unemployment rate has a p-value of 0.045, right at the border. With Johansen you would have to decide whether it is I(0) or I(1), and the decision would change the model. With ARDL the answer does not matter. That robustness to borderline cases is the main practical reason the method is so widely used.

6.3 Choose the lag orders

ARDL lets every variable have its own lag length. The search tries all combinations up to a maximum and keeps the one with the lowest information criterion.

```
for ic in ("aic", "bic"):
    sel = ardl_select_order(y, maxlag=4, exog=X, maxorder=4, ic=ic, trend="c")
    print(f"{ic.upper()}: lags of the yield = {sel.model.ar_lags},  lags of regressors = {sel.model.dl_lags}")
```

```
AIC: lags of the yield = [1, 2, 3],  lags of regressors = {'inflation': [0, 1, 2], 'unemployment': [0, 1, 2, 3]}
BIC: lags of the yield = [1, 2],  lags of regressors = {'inflation': [0], 'unemployment': [0, 1, 2]}
```

How to read it

AIC picks 3 lags of the yield, lags 0–2 of inflation and 0–3 of unemployment. BIC picks a smaller model, as usual. Use 4 as the maximum for quarterly data and 12 for monthly data unless your sample is short.

Notice that **GDP growth does not appear at all**: the search found that the model fits better without it. This is itself a result. Whether to keep such a variable is a judgement: if it is part of your research question, keep it with one lag and report that it is insignificant; if it was only a control, you may drop it. Here it is part of the question, so we keep it.

```
order = {"inflation": 2, "gdp_growth": 1, "unemployment": 3}
ardl = ARDL(y, lags=3, exog=X, order=order, trend="c").fit()
print(ardl.summary().tables[1])
```

```
==============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0163      0.123      0.133      0.894      -0.225       0.258
yield_10y.L1   1.2317      0.065     18.933      0.000       1.103       1.360
yield_10y.L2  -0.4211      0.099     -4.268      0.000      -0.615      -0.227
yield_10y.L3   0.1448      0.063      2.299      0.022       0.021       0.269
```

```
inflation.L0        0.1075      0.027      3.924      0.000       0.054       0.162
inflation.L1       -0.0971      0.033     -2.948      0.004      -0.162      -0.032
inflation.L2        0.0780      0.028      2.769      0.006       0.022       0.133
gdp_growth.L0       0.0131      0.011      1.214      0.226      -0.008       0.034
gdp_growth.L1       0.0058      0.011      0.536      0.593      -0.015       0.027
unemployment.L0    -0.5661      0.139     -4.076      0.000      -0.840      -0.292
unemployment.L1     1.1536      0.235      4.916      0.000       0.691       1.616
unemployment.L2    -0.8898      0.244     -3.640      0.000      -1.372      -0.408
unemployment.L3     0.2862      0.123      2.325      0.021       0.044       0.529
==============================================================================
```

How to read it

This is the model in levels, ARDL(3, 2, 1, 3). The coefficients on individual lags alternate in sign and have no direct interpretation. For example, unemployment has coefficients of −0.58, +1.13, −0.85 and +0.28. What matters is their **sum** relative to the persistence of the yield, which the next steps compute. Never interpret single lag coefficients from this table in a paper.

6.4 The bounds test: is there a long-run relationship?

The bounds test is an F-test of the null hypothesis that the level terms in the error-correction form are jointly zero, which means **no long-run relationship**. It has two critical values instead of one.

F-statistic is…	Conclusion
above the **upper** bound	Reject the null: a long-run relationship exists
below the **lower** bound	Do not reject: no long-run relationship
between the bounds	Inconclusive

```
uecm = UECM(y, lags=3, exog=X, order=order, trend="c").fit()
bounds = uecm.bounds_test(case=3)
print(f"Bounds F-statistic = {bounds.statistic:.2f}\n")
print("Critical values       lower I(0)   upper I(1)")
for level, row in bounds.critical_values.iterrows():
    print(f"   {100 - level:4.1f}% level  {row['lower']:10.2f}{row['upper']:13.2f}")
```

```
Bounds F-statistic = 8.18

Critical values       lower I(0)   upper I(1)
   10.0% level        2.46         3.52
    5.0% level        2.88         4.01
    1.0% level        3.78         5.05
    0.1% level        4.99         6.40
```

How to read it

The F-statistic of 8.18 is above the upper bound at every significance level, including 6.40 at the 0.1% level. The null of no long-run relationship is rejected: the yield, inflation, growth and unemployment are tied together in the long run.

`case=3` means an unrestricted constant and no trend, the usual choice when the dependent variable has no deterministic trend. Always state the case in the paper, because the critical values depend on it. If the statistic had fallen below the lower bound, you would stop here and estimate the model in first differences only.

6.5 Long-run coefficients

```
long_run = pd.DataFrame({"coefficient": -uecm.ci_params, "std. error": uecm.ci_bse, "p-value": uecm.ci_pvalues
  ↪ }).drop("yield_10y")
print(long_run.round(3))
```

```
             coefficient  std. error  p-value
const              0.366       2.751    0.894
inflation          1.982       0.445    0.000
gdp_growth         0.423       0.340    0.213
```

```
unemployment       -0.360       0.464    0.438
```

How to read it

statsmodels reports the long-run relation as a vector with the dependent variable normalised to 1, so the signs must be reversed to write it as an equation. The code above does this. The result is

yield = 0.37 + 1.98 × inflation + 0.42 × GDP growth − 0.36 × unemployment

- **Inflation** is the only significant long-run determinant. A permanent 1-point rise in inflation raises the 10-year yield by about 2 points in the long run. The Fisher relation predicts at least 1; values above 1 are common because investors also require compensation for taxes and inflation risk.
- **GDP growth** and **unemployment** have the expected signs but are not significant: the business cycle does not shift the yield permanently.

6.6 The error-correction form: short-run dynamics and speed of adjustment

```
print(uecm.summary().tables[1])
rho = uecm.params["yield_10y.L1"]
print(f"\nSpeed of adjustment = {rho:.4f}  ->  {abs(rho) * 100:.1f}% of a gap closes each quarter")
print(f"Half-life = {np.log(0.5) / np.log(1 + rho):.1f} quarters")
```

```
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                0.0163      0.123      0.133      0.894      -0.225       0.258
yield_10y.L1        -0.0446      0.013     -3.543      0.000      -0.069      -0.020
inflation.L1         0.0884      0.016      5.449      0.000       0.056       0.120
gdp_growth.L1        0.0189      0.015      1.232      0.219      -0.011       0.049
unemployment.L1     -0.0160      0.019     -0.857      0.392      -0.053       0.021
D.yield_10y.L1       0.2763      0.064      4.333      0.000       0.151       0.402
D.yield_10y.L2      -0.1448      0.063     -2.299      0.022      -0.269      -0.021
D.inflation.L0       0.1075      0.027      3.924      0.000       0.054       0.162
D.inflation.L1      -0.0780      0.028     -2.769      0.006      -0.133      -0.022
D.gdp_growth.L0      0.0131      0.011      1.214      0.226      -0.008       0.034
D.unemployment.L0   -0.5661      0.139     -4.076      0.000      -0.840      -0.292
D.unemployment.L1    0.6036      0.153      3.935      0.000       0.301       0.906
D.unemployment.L2   -0.2862      0.123     -2.325      0.021      -0.529      -0.044
==============================================================================

Speed of adjustment = -0.0446  ->  4.5% of a gap closes each quarter
Half-life = 15.2 quarters
```

How to read it

- **The error-correction coefficient** is the one on `yield_10y.L1`: −0.045. It must be **negative and significant** for the long-run relationship to pull the yield back. It is, but it is small: only 4.5% of a deviation is corrected per quarter, so half of a gap remains after about 15 quarters. Long-term yields adjust slowly.
- **Short-run coefficients** are the ones starting with `D.`. A rise in inflation raises the yield immediately (`D.inflation.L0` = 0.11), and a rise in unemployment lowers it sharply within the quarter (`D.unemployment.L0` = −0.57), consistent with yields falling in recessions. So unemployment matters in the short run even though it has no long-run effect. That distinction is what the error-correction form is for.

The t-statistic on the error-correction coefficient does not follow the standard t distribution. Use the bounds test for the existence of the relationship, and report the coefficient for its sign and size.

How to write it: long-run and short-run results

The bounds F-statistic of 8.18 exceeds the upper critical bound at the 1% level (5.05), so the null hypothesis of no long-run relationship is rejected. The estimated long-run coefficients (Table 3) show that inflation is the dominant long-run determinant of the 10-year yield: a one-percentage-point increase in inflation raises the yield by 1.98 percentage points (s.e. 0.45). The long-run effects of GDP growth and unemployment are statistically

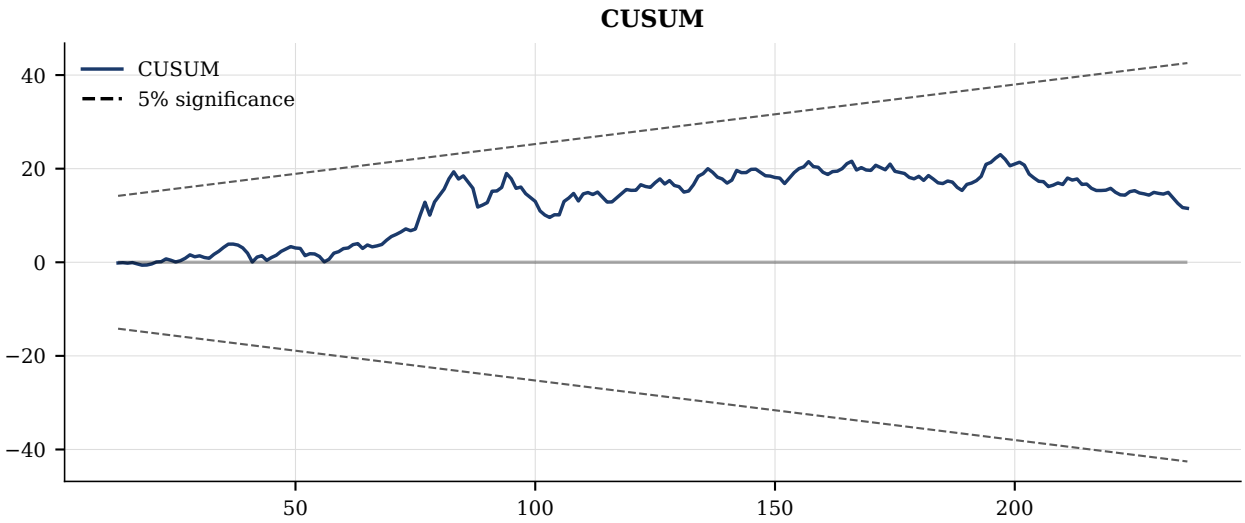


Figure 6.1. Diagnostic and stability tests: CUSUM.

insignificant. The error-correction coefficient is −0.045 and significant at the 1% level, which implies that 4.5% of any deviation from the long-run relationship is corrected each quarter, a half-life of about 15 quarters. In the short run, a one-point rise in the unemployment rate lowers the yield by 0.57 points within the quarter. The long-run relationship is identified from the full sample; in the 1990–2019 subsample, when inflation was low and stable, the bounds test does not reject the null of no long-run relationship.

6.7 Diagnostic and stability tests

The bounds test assumes serially uncorrelated errors, so that test matters most. The error-correction model is a linear regression, so it can be refitted by OLS to use the standard tests.

```
from statsmodels.stats.diagnostic import acorr_breusch_godfrey, het_white, het_arch, breaks_cusumolsresid
from statsmodels.stats.stattools import jarque_bera

ols = sm.OLS(uecm.model._y, uecm.model._x).fit()        # same regression, estimated by OLS
print(f"Breusch-Godfrey (4 lags):  LM = {acorr_breusch_godfrey(ols, nlags=4)[0]:6.2f}   p-value = {
  ↪ acorr_breusch_godfrey(ols, nlags=4)[1]:.3f}")
print(f"White heteroskedasticity:  LM = {het_white(ols.resid, ols.model.exog)[0]:6.2f}   p-value = {het_white(
  ↪ ols.resid, ols.model.exog)[1]:.3f}")
print(f"ARCH (4 lags):             LM = {het_arch(ols.resid, nlags=4)[0]:6.2f}   p-value = {het_arch(ols.resid,
  ↪  nlags=4)[1]:.3f}")
print(f"Jarque-Bera:               JB = {jarque_bera(ols.resid)[0]:6.2f}   p-value = {jarque_bera(ols.resid)
  ↪ [1]:.3f}")
print(f"CUSUM stability test:    stat = {breaks_cusumolsresid(ols.resid, ddof=len(ols.params))[0]:6.2f}   p-
  ↪ value = {breaks_cusumolsresid(ols.resid, ddof=len(ols.params))[1]:.3f}")
```

```
Breusch-Godfrey (4 lags):  LM =   5.94   p-value = 0.204
White heteroskedasticity:  LM = 161.04   p-value = 0.000
ARCH (4 lags):             LM =  11.00   p-value = 0.027
Jarque-Bera:               JB =  36.31   p-value = 0.000
CUSUM stability test:    stat =   0.78   p-value = 0.573
```

```
rec = sm.RecursiveLS(uecm.model._y, uecm.model._x).fit()
fig = plt.figure(figsize=(8, 3)); rec.plot_cusum(fig=fig); fig.axes[0].set_title("CUSUM")
fig2 = plt.figure(figsize=(8, 3)); rec.plot_cusum_squares(fig=fig2); fig2.axes[0].set_title("CUSUM of squares")
  ↪
```

How to read it

- **Serial correlation.** Breusch–Godfrey does not reject (p = 0.20). The condition the bounds test needs is met. If it had rejected, the fix is to add lags.

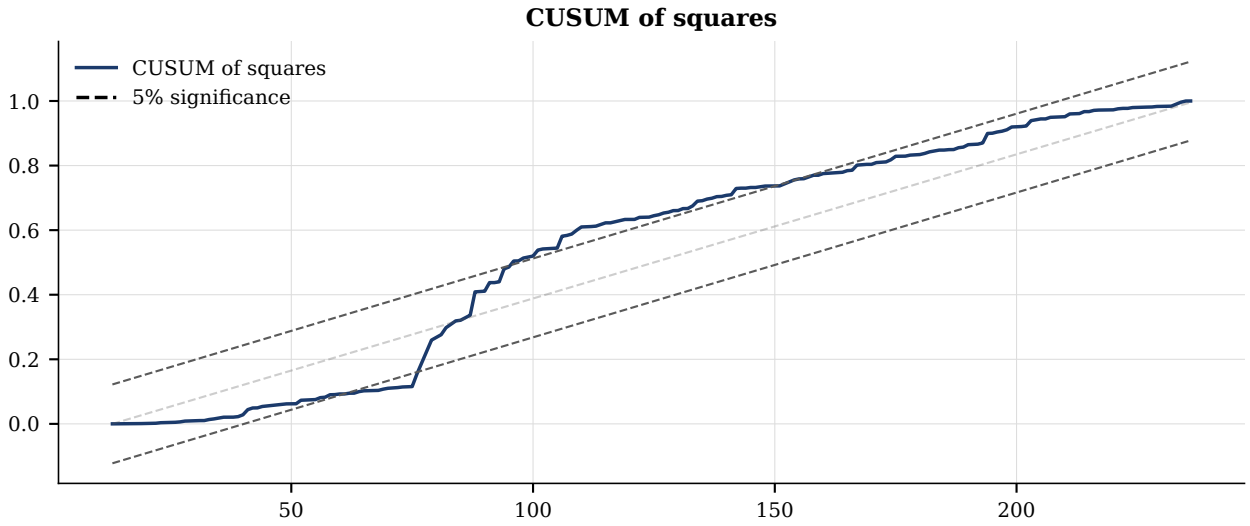


Figure 6.2. Diagnostic and stability tests: CUSUM of squares.

- **Heteroskedasticity.** The White and ARCH tests reject. The coefficients are unaffected, but standard errors should be robust; the robustness table below re-estimates with HAC standard errors.
- **Normality.** Rejected, because of the large yield movements around 1980. With 237 observations, inference rests on large-sample results.
- **CUSUM.** The line stays well inside its 5% bounds and the test does not reject ($p = 0.57$): the coefficients are stable.
- **CUSUM of squares.** This one leaves the bounds, falling below them around observations 60–78 and rising above them around 100–150. The break is at about observation 80, which is 1980: the Volcker disinflation, when the volatility of yields jumped. CUSUM of squares detects instability in the **variance**, and that agrees with the heteroskedasticity tests.

So the mean relationship is stable but the error variance is not. Report both plots, name the episode responsible, and use robust standard errors. Do not show only the plot that passes.

6.8 Robustness checks

```
def run(y_, X_, lags, order_, **fit):
    m = UECM(y_, lags=lags, exog=X_, order=order_, trend="c").fit(**fit)
    b = m.bounds_test(case=3)
    return b.statistic, b.critical_values.loc[95.0, "upper"], -m.ci_params["inflation"], m.ci_pvalues["
  ↪ inflation"], m.params["yield_10y.L1"]

cases = {
    "Baseline ARDL(3,2,1,3)":   (y, X, 3, order, {}),
    "BIC lags ARDL(2,1,1,2)":   (y, X, 2, {"inflation": 1, "gdp_growth": 1, "unemployment": 2}, {}),
    "HAC standard errors":      (y, X, 3, order, {"cov_type": "HAC", "cov_kwds": {"maxlags": 4}}),
    "1960-2007 (pre-crisis)":   (y[:"2007-12"], X[:"2007-12"], 3, order, {}),
    "1960-1989":                (y[:"1989-12"], X[:"1989-12"], 3, order, {}),
    "1990-2019":                (y["1990-01":], X["1990-01":], 3, order, {}),
}
print(f"{'':26s}{'bounds F':>9s}{'5% upper':>10s}{'LR inflation':>14s}{'p-value':>9s}{'adjustment':>12s}")
for name, (y_, X_, lags, order_, fit) in cases.items():
    f, cv, lr, p, rho_ = run(y_, X_, lags, order_, **fit)
    print(f"{name:26s}{f:9.2f}{cv:10.2f}{lr:14.2f}{p:9.3f}{rho_:12.3f}")
```

```
                          bounds F  5% upper  LR inflation  p-value  adjustment
Baseline ARDL(3,2,1,3)        8.18      4.01          1.98    0.000      -0.045
BIC lags ARDL(2,1,1,2)        6.13      4.01          1.97    0.000      -0.037
HAC standard errors           8.18      4.01          1.98    0.001      -0.045
1960-2007 (pre-crisis)        8.61      4.01          2.17    0.007      -0.047
1960-1989                     6.67      4.01          2.12    0.024      -0.053
```

1990-2019	2.25	4.01	1.59	0.047	-0.055

How to read it

- **Lag choice and standard errors do not matter.** The BIC specification and HAC standard errors leave the long-run inflation coefficient at about 2 and strongly significant.
- **The pre-crisis sample agrees.** Ending in 2007 gives a slightly larger coefficient (2.17) and a higher bounds F-statistic.
- **The relationship is not found after 1990.** In 1990–2019 the bounds F-statistic is 2.25, which is below even the *lower* bound, so the null of no long-run relationship cannot be rejected in that period. Inflation was low and stable after 1990, so there is little variation left to identify its long-run effect on yields.

The long-run link between inflation and the yield is identified mainly from the high-inflation decades before 1990. That is a limitation of the result and it belongs in the paper, in the robustness section and in the conclusion.

6.9 Write the paper

Section	What goes in it	From which step
Data	Sources, sample, unit-root tests on levels **and first differences**	Section 6.2
Method	Why ARDL, the equation, lag selection, the case for deterministic terms	Sections 6.1 and 6.3
Cointegration	Bounds F-statistic with lower and upper critical values	Section 6.4
Long-run results	Long-run coefficients as a table and as an equation	Section 6.5
Short-run results	Error-correction coefficient, half-life, short-run coefficients	Section 6.6
Robustness	Diagnostics, CUSUM plots, lag choice, subsamples	Sections 6.7 and 6.8

What a referee will ask

- **Did you rule out I(2)?** Show unit-root tests on first differences.
- **Which case for the deterministic terms, and which critical values?** State both. For samples under about 80 observations, use the small-sample critical values of Narayan (2005).
- **Is the error-correction coefficient negative and significant?** If it is positive or below –1, something is wrong with the specification.
- **Are the residuals serially uncorrelated?** The bounds test is not valid otherwise.
- **Is the long-run relationship stable?** Show CUSUM plots and subsamples.
- **Are the regressors exogenous?** ARDL assumes the dependent variable does not drive the regressors in the long run. If the yield could affect inflation, say so, or use a system method.

CHAPTER 7

Nonlinear ARDL (NARDL)

Applies when. I(0) and I(1) variables, with asymmetric effects
Overview. Use this chapter when the ARDL conditions hold and you expect increases and decreases in a regressor to act differently. It covers partial sums, the bounds test, Wald tests of long-run and short-run symmetry, dynamic multipliers, diagnostics and robustness, with sample text for the paper.
Data. U.S. Baa corporate bond yield and federal funds rate, monthly, 1985–2019
Software. Python · statsmodels

7.1 The question and the model

The question is: **do corporate borrowing costs respond differently to increases and decreases in the policy rate?** A standard ARDL model forces a rate rise and a rate cut of the same size to have equal and opposite effects. There are reasons to doubt that: credit risk premia, expectations about the business cycle and the behaviour of lenders can all differ between tightening and easing.

The nonlinear ARDL model of Shin, Yu and Greenwood-Nimmo (2014) relaxes this restriction with one simple device. The regressor x_t is split into two **partial sums**, one that accumulates only its increases and one that accumulates only its decreases:

$$x_t^+ = \sum_{s=1}^{t} \max(\Delta x_s, 0), \qquad x_t^- = \sum_{s=1}^{t} \min(\Delta x_s, 0), \qquad x_t = x_0 + x_t^+ + x_t^-$$

The two partial sums then enter an ordinary ARDL model as separate regressors:

$$\Delta y_t = c + \rho\, y_{t-1} + \delta^+ x_{t-1}^+ + \delta^- x_{t-1}^- + \sum_i \gamma_i \Delta y_{t-i} + \sum_i \left(\lambda_i^+ \Delta x_{t-i}^+ + \lambda_i^- \Delta x_{t-i}^-\right) + \varepsilon_t$$

The long-run effects are $L^+ = -\delta^+/\rho$ for increases and $L^- = -\delta^-/\rho$ for decreases. If $L^+ = L^-$ and the short-run coefficients are equal too, the model collapses to the linear ARDL.

When to use NARDL, and when not to

Use it when theory or earlier evidence gives a reason to expect asymmetry: fuel prices that rise faster than they fall, exchange-rate depreciations versus appreciations, monetary tightening versus easing. Do not use it only because it is fashionable. The symmetry tests in Sections 7.6 and 7.7 decide: if symmetry is not rejected, report the linear ARDL instead.

How to write it: model section

To allow increases and decreases in the policy rate to have different effects on corporate borrowing costs, we use the nonlinear ARDL model of Shin, Yu and Greenwood-Nimmo (2014). The federal funds rate is decomposed into partial sums of its positive and negative changes, $x_t^+ = \sum_{s\le t} \max(\Delta x_s, 0)$ and $x_t^- = \sum_{s\le t} \min(\Delta x_s, 0)$, which enter the unrestricted error-correction model as separate regressors. The long-run effects of rate increases and rate decreases are $L^+ = -\delta^+/\rho$ and $L^- = -\delta^-/\rho$. Long-run symmetry ($L^+ = L^-$) and short-run symmetry ($\sum_i \lambda_i^+ = \sum_i \lambda_i^-$) are tested with Wald tests.

7.2 Load the data and check the orders of integration

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.ardl import ARDL, UECM, ardl_select_order

full = pd.read_csv("passthrough.csv", parse_dates=["date"]).set_index("date")
full.index.freq = "MS"
df = full["1985":].copy()

for col in ("baa_yield", "fedfunds"):
    p0 = adfuller(df[col], autolag="AIC")[1]
    p1 = adfuller(df[col].diff().dropna(), autolag="AIC")[1]
    print(f"{col:10s} ADF p, level = {p0:.3f}   ADF p, first difference = {p1:.3f}")
print(len(df), "monthly observations")
```

```
baa_yield  ADF p, level = 0.102   ADF p, first difference = 0.000
fedfunds   ADF p, level = 0.132   ADF p, first difference = 0.001
420 monthly observations
```

How to read it

Both series are I(1), and neither is I(2). NARDL has the same requirement as ARDL: any mix of I(0) and I(1) is allowed, but no I(2) variable. The dependent variable should be I(1).

7.3 Build the partial sums

```
change = df["fedfunds"].diff().fillna(0)
df["ffr_pos"] = change.clip(lower=0).cumsum()     # accumulates rate increases only
df["ffr_neg"] = change.clip(upper=0).cumsum()     # accumulates rate decreases only

print(f"Months with a rise: {(change > 0).sum()}    months with a fall: {(change < 0).sum()}")
print(f"Total of all rises: {df['ffr_pos'].iloc[-1]:.2f} points    total of all falls: {df['ffr_neg'].iloc
  ↪ [-1]:.2f} points")
check = df["fedfunds"].iloc[0] + df["ffr_pos"] + df["ffr_neg"] - df["fedfunds"]
print(f"Largest gap between the rate and its rebuilt value: {check.abs().max():.10f}")

fig, ax = plt.subplots(1, 2, figsize=(8, 3))
df[["baa_yield", "fedfunds"]].plot(ax=ax[0], title="Baa yield and federal funds rate (%)"); ax[0].set_xlabel("
  ↪ ")
df[["ffr_pos", "ffr_neg"]].plot(ax=ax[1], title="Partial sums of the federal funds rate"); ax[1].set_xlabel("")
  ↪
plt.tight_layout()
```

```
Months with a rise: 198    months with a fall: 169
Total of all rises: 21.52 points    total of all falls: -28.32 points
Largest gap between the rate and its rebuilt value: 0.0000000000
```

How to read it

`ffr_pos` only ever rises and `ffr_neg` only ever falls. Added to the starting value, they rebuild the original series exactly, which the check confirms. Always run this check: a mistake in the partial sums invalidates everything that follows.

The asymmetry tests need enough observations of both kinds. Here there are 198 months with a rise and 169 with a fall, which is well balanced. If your regressor almost always moves in one direction, NARDL cannot identify the other side.

Because `ffr_neg` is negative and falling, read its coefficient with care: a **positive** coefficient on `ffr_neg` means that a rate **cut** (a fall in `ffr_neg`) **lowers** the yield.

7.4 Choose the lags and estimate the model

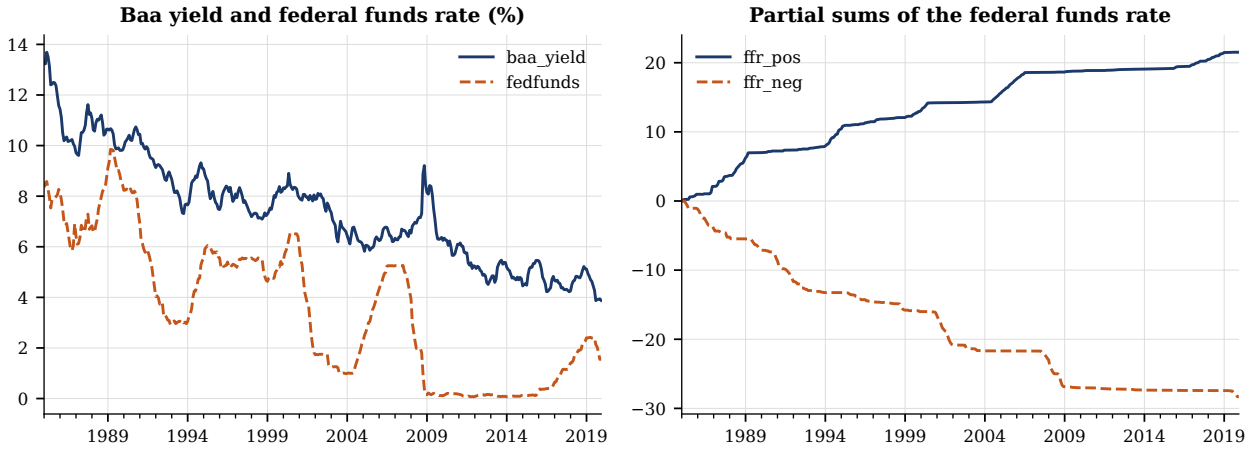


Figure 7.1. Build the partial sums. Panels: Baa yield and federal funds rate (%); Partial sums of the federal funds rate.

```
y = df["baa_yield"]
X = df[["ffr_pos", "ffr_neg"]]
for ic in ("aic", "bic"):
    sel = ardl_select_order(y, maxlag=6, exog=X, maxorder=6, ic=ic, trend="c")
    print(f"{ic.upper()}: lags of the yield = {sel.model.ar_lags},  lags of regressors = {sel.model.dl_lags}")
```

```
AIC: lags of the yield = [1, 2, 3, 4],  lags of regressors = {'ffr_pos': [0, 1], 'ffr_neg': [0, 1, 2, 3, 4, 5, 6]}
BIC: lags of the yield = [1, 2, 3],  lags of regressors = {'ffr_neg': [0]}
```

How to read it

The unrestricted search gives the two partial sums very different lag lengths, and BIC drops rate increases altogether. For the symmetry tests it is cleaner to give both partial sums the **same** lag length, so that the comparison is like for like. We use 3 lags of the yield and 3 of each partial sum, close to the AIC choice for the yield, and vary this in the robustness step.

```
order = {"ffr_pos": 3, "ffr_neg": 3}
nardl = UECM(y, lags=3, exog=X, order=order, trend="c").fit()
print(nardl.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.5311      0.152      3.491      0.001       0.232       0.830
baa_yield.L1  -0.0529      0.013     -4.110      0.000      -0.078      -0.028
ffr_pos.L1     0.0037      0.008      0.487      0.626      -0.011       0.019
ffr_neg.L1     0.0125      0.006      2.093      0.037       0.001       0.024
D.baa_yield.L1  0.3233      0.048      6.743      0.000       0.229       0.417
D.baa_yield.L2 -0.1394      0.049     -2.871      0.004      -0.235      -0.044
D.ffr_pos.L0   0.2976      0.108      2.761      0.006       0.086       0.510
D.ffr_pos.L1   0.0188      0.111      0.169      0.866      -0.200       0.238
D.ffr_pos.L2  -0.0751      0.108     -0.697      0.486      -0.287       0.137
D.ffr_neg.L0  -0.0512      0.083     -0.615      0.539      -0.215       0.112
D.ffr_neg.L1  -0.1350      0.092     -1.474      0.141      -0.315       0.045
D.ffr_neg.L2   0.0518      0.084      0.618      0.537      -0.113       0.216
==============================================================================
```

7.5 The bounds test

```
bounds = nardl.bounds_test(case=3)
print(f"Bounds F-statistic = {bounds.statistic:.2f}")
print("Critical values       lower I(0)   upper I(1)")
for level, row in bounds.critical_values.iterrows():
    print(f"   {100 - level:4.1f}% level  {row['lower']:10.2f}{row['upper']:13.2f}")
rho = nardl.params["baa_yield.L1"]
```

```
print(f"\nError-correction coefficient = {rho:.4f}   half-life = {np.log(0.5) / np.log(1 + rho):.1f} months")
```

```
Bounds F-statistic = 6.88
Critical values       lower I(0)   upper I(1)
   10.0% level        2.73         3.75
    5.0% level        3.23         4.32
    1.0% level        4.31         5.54
    0.1% level        5.76         7.14

Error-correction coefficient = -0.0529   half-life = 12.8 months
```

How to read it

The F-statistic of 6.88 is above the upper bound at the 1% level (5.54), so a long-run relationship exists between the Baa yield and the two partial sums. The bounds test is read exactly as in Chapter 6; the number of regressors counted for the critical values is two, the two partial sums.

The error-correction coefficient is negative and significant: 5.3% of a deviation is corrected each month, a half-life of about 13 months.

7.6 Long-run coefficients and the long-run symmetry test

```
L = pd.DataFrame({"long-run effect": -nardl.ci_params, "std. error": nardl.ci_bse, "p-value": nardl.ci_pvalues
  ↪ }).loc[["ffr_pos", "ffr_neg"]]
print(L.round(3))

wald_lr = nardl.wald_test("ffr_pos.L1 = ffr_neg.L1", scalar=True)
print(f"\nLong-run symmetry (H0: L+ = L-):  F = {wald_lr.statistic:.2f}   p-value = {wald_lr.pvalue:.3f}")
```

```
         long-run effect  std. error  p-value
ffr_pos            0.070       0.147    0.633
ffr_neg            0.237       0.105    0.024

Long-run symmetry (H0: L+ = L-):  F = 4.63   p-value = 0.032
```

How to read it

- **Rate increases:** the long-run effect is 0.07 and is not significantly different from zero. A permanent 1-point rise in the federal funds rate does not raise the Baa yield in the long run.
- **Rate decreases:** the long-run effect is 0.24 and significant. A 1-point cut lowers the Baa yield by about 0.24 points in the long run.
- **The symmetry test** has the null that the two long-run effects are equal. It is rejected at 5% (p = 0.03): the long-run response is asymmetric.

The test restricts the two level coefficients to be equal, which is the same as $L^{+} = L^{-}$ because both are divided by the same ρ.

7.7 The short-run symmetry test

```
pos = " + ".join(f"D.ffr_pos.L{i}" for i in range(3))
neg = " + ".join(f"D.ffr_neg.L{i}" for i in range(3))
wald_sr = nardl.wald_test(f"{pos} = {neg}", scalar=True)
print(f"Sum of short-run coefficients, increases: {sum(nardl.params[f'D.ffr_pos.L{i}'] for i in range(3)):.3f}
  ↪ ")
print(f"Sum of short-run coefficients, decreases: {sum(nardl.params[f'D.ffr_neg.L{i}'] for i in range(3)):.3f}
  ↪ ")
print(f"Short-run symmetry (H0: sums are equal):  F = {wald_sr.statistic:.2f}   p-value = {wald_sr.pvalue:.3f}
  ↪ ")
print(f"Impact effect of an increase: {nardl.params['D.ffr_pos.L0']:.3f} (p = {nardl.pvalues['D.ffr_pos.L0
  ↪ ']:.3f})")
print(f"Impact effect of a decrease:  {nardl.params['D.ffr_neg.L0']:.3f} (p = {nardl.pvalues['D.ffr_neg.L0
  ↪ ']:.3f})")
```

```
Sum of short-run coefficients, increases: 0.241
Sum of short-run coefficients, decreases: -0.135
Short-run symmetry (H0: sums are equal):  F = 3.38   p-value = 0.067
```

```
Impact effect of an increase: 0.298 (p = 0.006)
Impact effect of a decrease:  -0.051 (p = 0.539)
```

How to read it

In the month of a rate **rise**, the Baa yield goes up by 0.30 points per point, and this is significant. In the month of a rate **cut** there is no significant response. The test of equal short-run sums has a p-value of 0.07: asymmetry at the 10% level, not at 5%.

Put the two tests together and a pattern appears. Rate rises pass through **quickly but not permanently**; rate cuts pass through **slowly but permanently**. Four outcomes are possible in any NARDL study, and you should say which one you have: symmetry in both, asymmetry only in the long run, only in the short run, or in both.

7.8 Dynamic multipliers

The dynamic multiplier traces the path from the short run to the long run: the cumulative response of the yield, month by month, to a permanent 1-point rise and to a permanent 1-point cut.

```
levels = ARDL(y, lags=3, exog=X, order=order, trend="c").fit()      # the same model written in levels
phi = [levels.params[f"baa_yield.L{i}"] for i in (1, 2, 3)]

def multiplier(name, horizon=60):
    theta = [levels.params[f"{name}.L{j}"] for j in range(4)]
    m = np.zeros(horizon + 1)
    for h in range(horizon + 1):
        m[h] = sum(theta[: h + 1]) + sum(phi[i] * m[h - i - 1] for i in range(3) if h - i - 1 >= 0)
    return m

rise = multiplier("ffr_pos")           # response to a 1-point rise
cut = -multiplier("ffr_neg")           # response to a 1-point cut (ffr_neg falls by 1)

print(f"{'months':>7s}{'1-point rise':>14s}{'1-point cut':>13s}{'asymmetry':>11s}")
for h in (0, 1, 3, 6, 12, 24, 60):
    print(f"{h:7d}{rise[h]:14.3f}{cut[h]:13.3f}{rise[h] + cut[h]:11.3f}")

fig, ax = plt.subplots(figsize=(8, 3.4))
ax.plot(rise, label="1-point rise"); ax.plot(cut, label="1-point cut"); ax.plot(rise + cut, ls="--", label="
  ↪ asymmetry (sum)")
ax.axhline(0, color="k", lw=.8); ax.set_xlabel("months after the change"); ax.set_title("Dynamic multipliers:
  ↪ response of the Baa yield"); ax.legend()
```

```
 months  1-point rise  1-point cut  asymmetry
      0         0.298        0.051      0.349
      1         0.401        0.188      0.588
      3         0.241        0.099      0.339
      6         0.214        0.034      0.248
     12         0.166       -0.056      0.110
     24         0.113       -0.156     -0.043
     60         0.074       -0.230     -0.156
```

How to read it

- **After a rise** the yield jumps by 0.30 points at once and by 0.40 after one month. The effect then fades, to 0.17 after a year and 0.07 after five years.
- **After a cut** the yield does not fall at first; the point estimates for the first months are slightly positive and not significant. The decline builds slowly: –0.06 after a year, –0.16 after two years and –0.23 after five.
- **The asymmetry line** is the sum of the two. It would be zero at every horizon if the responses were mirror images. It starts positive, because rises have the larger short-run effect, crosses zero after about two years, and ends negative, because cuts have the larger long-run effect.

Both lines converge to the long-run effects from Section 7.6 (0.07 and –0.24), which is a useful check that the multipliers were computed correctly. The multiplier figure is usually the main figure of a NARDL paper.

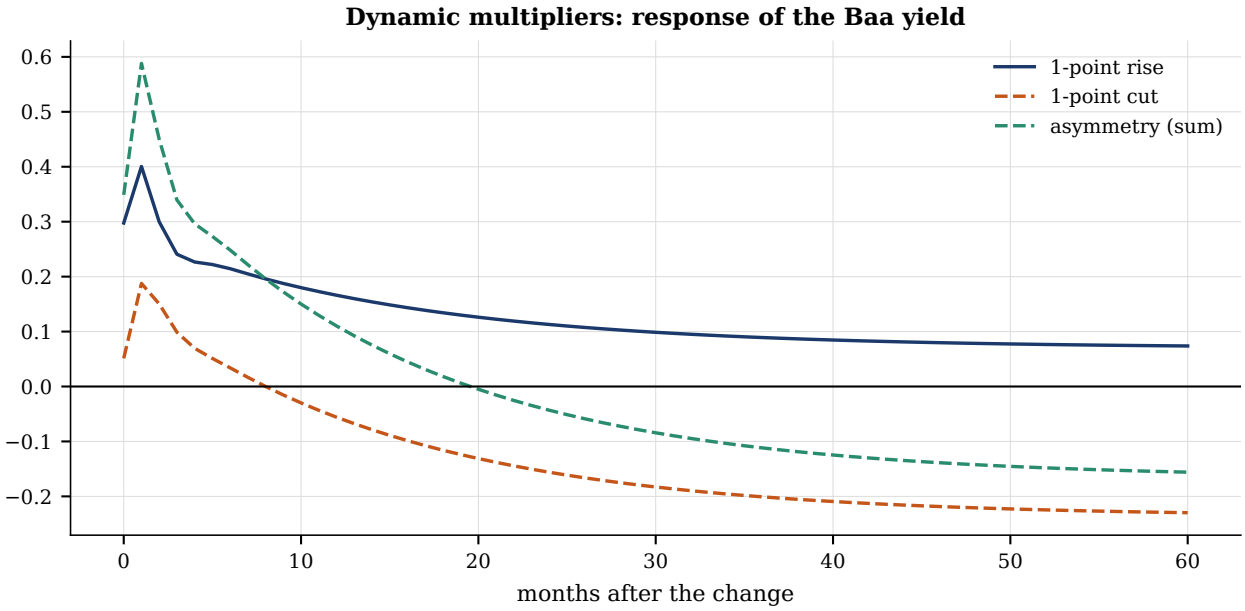


Figure 7.2. Dynamic multipliers: Dynamic multipliers: response of the Baa yield.

7.9 Diagnostic tests

```
from statsmodels.stats.diagnostic import acorr_breusch_godfrey, het_white, het_arch, breaks_cusumolsresid
from statsmodels.stats.stattools import jarque_bera

ols = sm.OLS(nardl.model._y, nardl.model._x).fit()
print(f"Breusch-Godfrey (12 lags): p-value = {acorr_breusch_godfrey(ols, nlags=12)[1]:.3f}")
print(f"White heteroskedasticity:  p-value = {het_white(ols.resid, ols.model.exog)[1]:.3f}")
print(f"ARCH (6 lags):             p-value = {het_arch(ols.resid, nlags=6)[1]:.3f}")
print(f"Jarque-Bera normality:     p-value = {jarque_bera(ols.resid)[1]:.3f}")
print(f"CUSUM stability:           p-value = {breaks_cusumolsresid(ols.resid, ddof=len(ols.params))[1]:.3f}")

robust = UECM(y, lags=3, exog=X, order=order, trend="c").fit(cov_type="HAC", cov_kwds={"maxlags": 6})
print(f"\nWith HAC standard errors: long-run symmetry p-value = {robust.wald_test('ffr_pos.L1 = ffr_neg.L1',
  ↪ scalar=True).pvalue:.3f}"
      f"   short-run symmetry p-value = {robust.wald_test(f'{pos} = {neg}', scalar=True).pvalue:.3f}")
```

```
Breusch-Godfrey (12 lags): p-value = 0.633
White heteroskedasticity:  p-value = 0.000
ARCH (6 lags):             p-value = 0.693
Jarque-Bera normality:     p-value = 0.000
CUSUM stability:           p-value = 0.514

With HAC standard errors: long-run symmetry p-value = 0.084   short-run symmetry p-value = 0.085
```

How to read it

- **Serial correlation.** Not rejected ($p = 0.63$), so the bounds test is valid.
- **ARCH and CUSUM.** Neither rejects.
- **Heteroskedasticity and normality.** Both rejected, so the Wald tests should be based on robust standard errors.
- **The symmetry tests with HAC standard errors** give p-values of 0.08 for both the long run and the short run, compared with 0.03 and 0.07 before. Because the White test rejected, these robust p-values are the ones to rely on.

That changes the wording of the conclusion: the evidence for asymmetry is significant at the 10% level, not the 5% level. Report the robust version. Choosing whichever standard errors give the smaller p-value is not acceptable practice, and referees check.

7.10 Robustness checks

```
def run(sample, p, q):
    d_ = full[sample[0]:sample[1]].copy()
    ch = d_["fedfunds"].diff().fillna(0)
    d_["ffr_pos"], d_["ffr_neg"] = ch.clip(lower=0).cumsum(), ch.clip(upper=0).cumsum()
    m = UECM(d_["baa_yield"], lags=p, exog=d_[["ffr_pos", "ffr_neg"]], order={"ffr_pos": q, "ffr_neg": q},
  ↪ trend="c").fit()
    b = m.bounds_test(case=3)
    sr = m.wald_test(" + ".join(f"D.ffr_pos.L{i}" for i in range(q)) + " = " + " + ".join(f"D.ffr_neg.L{i}"
  ↪ for i in range(q)), scalar=True).pvalue
    return (b.statistic, b.critical_values.loc[95.0, "upper"], -m.ci_params["ffr_pos"], -m.ci_params["ffr_neg"
  ↪ ],
            m.wald_test("ffr_pos.L1 = ffr_neg.L1", scalar=True).pvalue, sr)

cases = {"Baseline 1985-2019, (3,3,3)": (("1985", "2019"), 3, 3), "Lags (2,2,2)": (("1985", "2019"), 2, 2), "
  ↪ Lags (4,4,4)": (("1985", "2019"), 4, 4),
         "1985-2007": (("1985", "2007"), 3, 3), "1990-2019": (("1990", "2019"), 3, 3), "1970-2019": (("1970",
  ↪ "2019"), 3, 3)}
print(f"{'':28s}{'bounds F':>9s}{'5% upper':>9s}{'L+ (rise)':>10s}{'L- (cut)':>9s}{'LR sym. p':>10s}{'SR sym.
  ↪ p':>10s}")
for name, (sample, p, q) in cases.items():
    f, cv, lp, ln, plr, psr = run(sample, p, q)
    print(f"{name:28s}{f:9.2f}{cv:9.2f}{lp:10.2f}{ln:9.2f}{plr:10.3f}{psr:10.3f}")

linear = UECM(y, lags=3, exog=df[["fedfunds"]], order={"fedfunds": 3}, trend="c").fit()
print(f"\nLinear ARDL for comparison: long-run effect = {-linear.ci_params['fedfunds']:.2f}   bounds F = {
  ↪ linear.bounds_test(case=3).statistic:.2f}   AIC = {linear.aic:.1f}   (NARDL AIC = {nardl.aic:.1f})")
```

```
                           bounds F 5% upper L+ (rise) L- (cut) LR sym. p SR sym. p
Baseline 1985-2019, (3,3,3)     6.88     4.32      0.07     0.24     0.032     0.067
Lags (2,2,2)                    7.16     4.32      0.03     0.22     0.010     0.045
Lags (4,4,4)                    7.60     4.32      0.09     0.26     0.018     0.109
1985-2007                       7.29     4.32      0.09     0.26     0.009     0.213
1990-2019                       3.84     4.32     -0.00     0.20     0.039     0.094
1970-2019                      10.58     4.32      0.81     0.82     0.761     0.887

Linear ARDL for comparison: long-run effect = 0.56   bounds F = 6.57   AIC = -191.1   (NARDL AIC = -196.6)
```

How to read it

- **Lag length.** With 2, 3 or 4 lags the long-run effects stay at about 0.05 for rises and 0.24 for cuts, and long-run symmetry is rejected each time.
- **1985–2007.** The long-run asymmetry is unchanged, so it is not a product of the zero-rate years after 2008. Short-run asymmetry is not significant in this sample.
- **1990–2019.** The bounds F-statistic of 3.84 lies between the bounds (3.23 and 4.32): the test is inconclusive, so the long-run results for this subsample cannot be interpreted.
- **1970–2019.** With the high-inflation years included, the long-run effects are 0.81 and 0.82 and symmetry is clearly not rejected ($p = 0.76$). Over the longer sample the response is symmetric and much larger.
- **Linear ARDL.** The symmetric model gives a single long-run effect of 0.56. The NARDL has the lower AIC, so the data from 1985 prefer the asymmetric model.

The asymmetry is a feature of the period since 1985, not a constant of the economy. One reason to be careful: both yields and the policy rate trended downwards over 1985–2019, and the falling partial sum may partly capture that common decline. A paper should present the asymmetry as a finding for this period and discuss this possibility.

7.11 Write the paper

Section	What goes in it	From which step
Data	Sources, sample, unit-root tests, plot of the series and the partial sums	Sections 7.2 and 7.3
Method	Why asymmetry is expected, the partial-sum decomposition, the NARDL equation	Section 7.1
Cointegration	Bounds F-statistic with critical values	Section 7.5
Long-run results	L^+ and L^- with standard errors, long-run symmetry test	Section 7.6
Short-run results	Short-run coefficients, short-run symmetry test, dynamic multiplier figure	Sections 7.7 and 7.8
Robustness	Diagnostics, lag lengths, samples, comparison with the linear ARDL	Sections 7.9 and 7.10

How to write it: results

The bounds F-statistic of 6.88 exceeds the 1% upper critical value (5.54), indicating a long-run relationship between the Baa yield and the positive and negative partial sums of the federal funds rate. The estimated long-run effects differ markedly: a one-point reduction in the policy rate lowers the Baa yield by 0.24 points in the long run (s.e. 0.11), whereas the long-run effect of a one-point increase is 0.07 and statistically insignificant. A Wald test rejects long-run symmetry at the 10% level with heteroskedasticity- and autocorrelation-robust standard errors ($p = 0.08$). In the short run the pattern is reversed: a rate increase raises the yield by 0.30 points within the month, while a rate cut has no significant immediate effect. The dynamic multipliers in Figure 3 show that the response to increases peaks after one month and then decays, while the response to cuts builds gradually over several years. The asymmetry is specific to the period since 1985; over 1970–2019 the long-run responses to increases and cuts are both about 0.8 and symmetry is not rejected.

What a referee will ask

- **Why should the effect be asymmetric?** Give the economic reason before the statistics.
- **Is symmetry actually rejected?** Report both Wald tests. If neither rejects, the linear ARDL is the right model.
- **Are there enough rises and enough falls?** Report the counts.
- **Do the partial sums just pick up a trend?** Partial sums trend by construction, one up and one down. If the dependent variable has a long trend of its own, the two can become confused. Check subsamples and, where sensible, include a trend.
- **Where are the confidence bands on the multipliers?** Journals increasingly expect bootstrap bands around the asymmetry line.
- **Are the results stable across samples?** Asymmetry found in one period often does not hold in another.

CHAPTER 8

Structural VAR (Cholesky, Long-Run and Proxy Identification)

Applies when. You want the effect of a structural shock on several variables, and you can defend how the shock is identified

Overview. A VAR describes how variables move together, but its residuals are mixtures of structural shocks. A structural VAR (SVAR) adds the assumptions that separate them. This chapter covers the three identification schemes most used in applied work, a recursive ordering, long-run restrictions (Blanchard and Quah 1989) and an external instrument (proxy-SVAR), and shows how to choose the lags, check stability, build bootstrap bands, test how much the answer depends on the identifying assumption, and report it.

Data. U.S. quarterly data, 1960–2019: real GDP growth and the unemployment rate; U.S. monthly data, 1990–2019: policy rate, industrial production, prices, credit spread and a high-frequency monetary shock

Software. Python · OES engine · statsmodels

8.1 The question and the model

What does a structural shock do to the system? The reduced-form VAR is

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + u_t, \qquad \mathrm{E}[u_t u_t'] = \Sigma,$$

and the residuals are linked to the structural shocks ε_t (uncorrelated, unit variance) by $u_t = B\varepsilon_t$. The data give $\Sigma = BB'$, which does not pin down B: any rotation of B fits equally well, so identification needs $n(n-1)/2$ extra restrictions. The impulse response of variable i to shock j at horizon h is then $\Psi_h B$, where Ψ_h are the moving-average coefficients of the VAR.

Three schemes are common. **Recursive (Cholesky):** B is lower triangular, so a variable ordered earlier does not respond to later shocks within the period. **Long-run restrictions:** the cumulative effect of some shocks on some variables is zero, for example demand shocks have no permanent effect on output (Blanchard and Quah 1989). **External instrument (proxy-SVAR):** a series z_t correlated with one structural shock and uncorrelated with the others identifies that shock's column of B (Stock and Watson 2012, Mertens and Ravn 2013).

Three checks before you trust a structural VAR

1. **Can you defend the identifying assumption in words?** The restrictions are not tested by the data (a recursive or long-run scheme is exactly identified). Write down the economic argument, and test how much the answer depends on it by changing the ordering or the scheme.
2. **Is the VAR well specified?** Stationary or differenced variables, enough lags to leave the residuals white, and a stable system. Long-run identification also needs a stationary VAR.
3. **Is the instrument valid (proxy-SVAR)?** Relevance is testable (the first-stage F statistic) and exogeneity is not, so the argument for exogeneity must come from how the instrument was built.

How to write it: the model

We estimate a structural VAR for real GDP growth and the unemployment rate and identify supply and demand shocks with the long-run restriction of Blanchard and Quah (1989): only the supply shock has a permanent effect on the level of output. The VAR has four lags, a compromise between the Akaike criterion (eight lags) and

the Schwarz and Hannan-Quinn criteria (two lags); two, six and eight lags are robustness checks. Confidence bands are 95% percentile intervals from a wild residual bootstrap with 500 replications, which is robust to heteroskedasticity. We check the result against a recursive identification with both orderings.

8.2 Load the data and check the variables

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller, kpss

data = open("yield.csv").read()
df = pd.read_csv("yield.csv")[["date", "gdp_growth", "unemployment"]].dropna()
print(df.date.iloc[0], "to", df.date.iloc[-1], "|", len(df), "quarters")
print(df[["gdp_growth", "unemployment"]].describe().round(2).loc[["mean", "std", "min", "max"]])
for c in ["gdp_growth", "unemployment"]:
    print(f"{c:12s} ADF p = {adfuller(df[c], autolag='AIC')[1]:.3f}   KPSS p = {kpss(df[c], nlags='auto')
  ↪ [1]:.3f}")
```

```
1960Q1 to 2019Q4 | 240 quarters
     gdp_growth  unemployment
mean        3.01          5.97
std         3.25          1.61
min        -8.85          3.40
max        15.17         10.67
gdp_growth   ADF p = 0.000   KPSS p = 0.059
unemployment ADF p = 0.045   KPSS p = 0.100
```

How to read it

The sample has 240 quarters (1960Q1-2019Q4). Real GDP growth (annualised) averages 3.01% with a standard deviation of 3.25; the unemployment rate averages 5.97% with a standard deviation of 1.61. Both series are treated as stationary: the ADF test rejects a unit root for growth ($p < 0.001$) and for unemployment ($p = 0.045$, borderline), and the KPSS test does not reject stationarity for either ($p = 0.059$ and $p = 0.10$). Unemployment is very persistent, which is why the largest VAR root is close to one. Long-run identification needs a stationary VAR, so this check matters.

8.3 Choose the lags and check stability

```
r = oes.run("svar", data, dict(vars=["gdp_growth", "unemployment"], time="date", identification="longrun",
  ↪ horizon=20, boot=50))
oes.show(r, "Lag-order")
oes.show(r, "Stability")
```

```
Lag-order selection
         AIC     BIC    HQIC      FPE
Lags
0     3.2909  3.3206  3.3029  26.8665
1    -0.7553 -0.6661 -0.7193   0.4699
2    -1.0870 -0.9385 -1.0271   0.3372
3    -1.1113 -0.9033 -1.0274   0.3291
4    -1.1175 -0.8501 -1.0097   0.3271
5    -1.0946 -0.7677 -0.9628   0.3347
6    -1.0862 -0.6999 -0.9304   0.3376
7    -1.1055 -0.6598 -0.9257   0.3312
8    -1.1196 -0.6145 -0.9159   0.3266
Note: Used: 8 lag(s) (chosen by AIC).

Stability
                                Statistic p-value                                                  Detail
Test
Stability: largest root modulus     0.9305    None  stable if below 1 (a level VAR with unit roots is near 1)
```

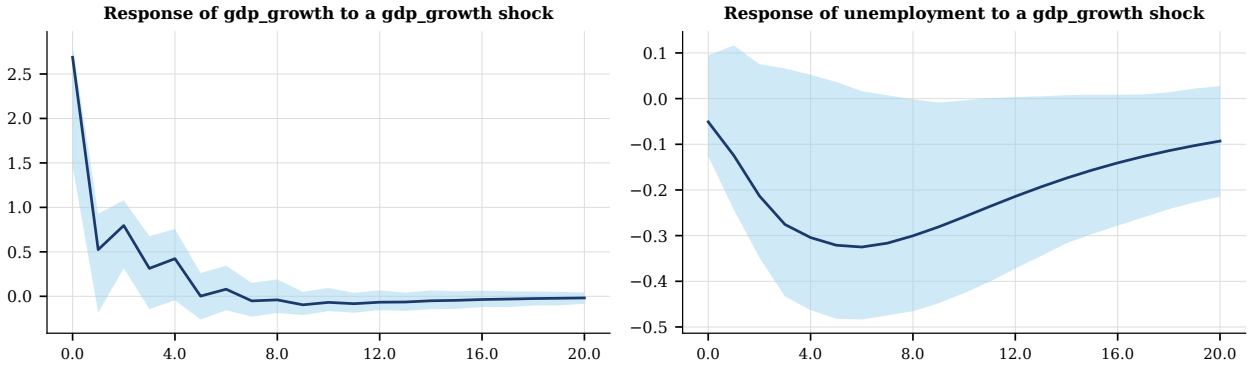


Figure 8.1. Identify with long-run restrictions (Blanchard–Quah). Panels: Response of gdp_growth to a gdp_growth shock; Response of unemployment to a gdp_growth shock.

How to read it

The Akaike criterion keeps falling slowly up to eight lags (-1.120), but most of the gain comes by two lags (-1.087 against -1.118 at four). The Schwarz and Hannan-Quinn criteria both choose two lags. We use four lags as a compromise and test two, six and eight below. With eight lags the largest root of the VAR is 0.93, so the system is stable and the long-run restriction is well defined.

8.4 Identify with long-run restrictions (Blanchard–Quah)

```
spec = dict(vars=["gdp_growth", "unemployment"], time="date", identification="longrun", lags=4, horizon=20,
  ↪ boot=500)
supply = oes.run("svar", data, dict(spec, shock="gdp_growth"))
demand = oes.run("svar", data, dict(spec, shock="unemployment"))
oes.show(supply, "Long-run impact")
oes.show(supply, "Share of forecast-error variance")
oes.show(demand, "Share of forecast-error variance")
oes.plot(supply, "Response of", ncols=2, size=(4.2, 2.6))
```

```
Long-run impact matrix
             shock gdp_growth  shock unemployment
Variable
gdp_growth              3.9656              0.0000
unemployment           -5.2331              6.1225
Note: Zeros above the diagonal are the identifying restrictions: shock j has no long-run effect on variables ordered
↪ before it.

Share of forecast-error variance explained by the gdp_growth shock (%)
             h = 1  h = 4  h = 8  h = 20
Variable
gdp_growth    87.38  81.85  81.02   80.44
unemployment   4.91  20.62  31.13   38.43
Note: Computed from the point estimates.

Share of forecast-error variance explained by the unemployment shock (%)
             h = 1  h = 4  h = 8  h = 20
Variable
gdp_growth    12.62  18.15  18.98   19.56
unemployment  95.09  79.38  68.87   61.57
Note: Computed from the point estimates.
```

How to read it

The long-run impact matrix shows the identifying restriction: the entry for the effect of the unemployment (”demand”) shock on GDP growth is exactly zero, so only the supply shock moves the level of output permanently. The supply shock raises the cumulated response of annualised growth by 3.97, which is about 1% of the level of output. The supply shock explains 87% of the forecast-error variance of GDP growth at one quarter and 80%

at five years, and 5% of that of unemployment at one quarter, rising to 38% at five years. The demand shock explains 95% of the variance of unemployment at one quarter and 62% at five years, and 13% to 20% of that of GDP growth.

```
rows = lambda r, v: [x for x in r["tables"][0]["rows"] if x[0] == v]
for name, r in [("supply shock", supply), ("demand shock", demand)]:
    print(name)
    for v in ["gdp_growth", "unemployment"]:
        print("  ", v, [(x[1], x[2], x[3], x[4]) for x in rows(r, v) if x[1] in (0, 4, 8, 20)])
```

```
supply shock
   gdp_growth [(0, 2.687724, 1.462895, 2.821603), (4, 0.423165, -0.044399, 0.756218), (8, -0.040145, -0.187534,
↪ 0.189196), (20, -0.018257, -0.084112, 0.043662)]
   unemployment [(0, -0.050894, -0.126485, 0.094414), (4, -0.304227, -0.463024, 0.052183), (8, -0.300401, -0.46549,
↪ -0.001463), (20, -0.09296, -0.214784, 0.027334)]
demand shock
   gdp_growth [(0, -1.021452, -2.368358, 0.029466), (4, 0.101285, -0.256146, 0.369106), (8, 0.165233, 0.023071,
↪ 0.413832), (20, 0.013091, -0.042316, 0.110958)]
   unemployment [(0, 0.224019, 0.181319, 0.22641), (4, 0.4472, 0.268933, 0.59008), (8, 0.319513, 0.143509, 0.489021),
↪  (20, 0.082927, -0.054683, 0.167637)]
```

How to read it

A supply shock raises GDP growth by 2.69 points on impact (band 1.46 to 2.82), 0.42 after one year, and the effect dies out after two years, which leaves the level of output permanently higher. Unemployment falls by 0.30 points after four to eight quarters (band at four quarters -0.46 to 0.05, at eight quarters -0.47 to 0.00), a borderline-significant decline that fades by year five. The demand-side shock raises unemployment by 0.22 points on impact and 0.45 after one year, and lowers GDP growth by 1.02 points on impact (band -2.37 to 0.03), after which growth is above its baseline (0.17 at two years, band 0.02 to 0.41) as unemployment returns to normal. The unemployment response is back to zero within five years (band -0.05 to 0.17), as the identification requires for the demand shock.

8.5 Compare with a recursive identification

```
chol = {}
for order in (["gdp_growth", "unemployment"], ["unemployment", "gdp_growth"]):
    for shock in order:
        r = oes.run("svar", data, dict(vars=order, time="date", identification="cholesky", lags=4, horizon=20,
  ↪  boot=100, shock=shock))
        chol[(" -> ".join(order), shock)] = {v: [x[2] for x in rows(r, v) if x[1] in (0, 4, 8, 20)] for v in
  ↪ order}
for k, v in chol.items():
    print(k, {a: [round(z, 2) for z in b] for a, b in v.items()})
```

```
('gdp_growth -> unemployment', 'gdp_growth') {'gdp_growth': [2.88, 0.36, -0.1, -0.02], 'unemployment': [-0.13, -0.44,
↪  -0.39, -0.12]}
('gdp_growth -> unemployment', 'unemployment') {'gdp_growth': [0.0, 0.25, 0.14, 0.01], 'unemployment': [0.19, 0.31,
↪ 0.19, 0.04]}
('unemployment -> gdp_growth', 'unemployment') {'unemployment': [0.23, 0.5, 0.38, 0.1], 'gdp_growth': [-1.59, 0.01,
↪ 0.17, 0.02]}
('unemployment -> gdp_growth', 'gdp_growth') {'unemployment': [0.0, -0.2, -0.22, -0.07], 'gdp_growth': [2.39, 0.44,
↪ -0.0, -0.01]}
```

How to read it

The recursive schemes give the same shapes but different impact effects, because each ordering assumes away one of the two contemporaneous responses. With growth ordered first, a growth shock lowers unemployment by 0.44 after one year and 0.39 after two; an unemployment shock has no effect on growth on impact by assumption. With unemployment ordered first, the unemployment shock raises unemployment by 0.50 after one year and lowers growth by 1.59 on impact. The long-run scheme (-1.02 on impact) lies between the two. The persistence of the unemployment response (a peak of 0.31 to 0.50 after about a year, mostly gone after five years) is similar

in all three, so the conclusion about dynamics does not depend on the identification, while the impact effect of the unemployment shock on growth does.

8.6 Proxy-SVAR: check the instrument before the responses

```
mp = open("monetary_proxy.csv").read()
for ind in ["gs1", "ffr"]:
    spec_p = dict(vars=[ind, "log_ip", "log_cpi", "credit_spread"], shock=ind, identification="proxy",
  ↪ instrument=["mp_shock"], time="date",
                  lags=12, horizon=36, boot=300)
    rp = oes.run("svar", mp, spec_p)
    F = [s["value"] for s in rp["summary"] if s["label"] == "First-stage F"][0]
    print(f"policy indicator {ind}: first-stage robust F = {F:.1f}")
    oes.show(rp, "Instrument")
```

```
policy indicator gs1: first-stage robust F = 4.5
Instrument and stability
                               Statistic p-value                                                  Detail
Test
Stability: largest root modulus     0.9978    None  stable if below 1 (a level VAR with unit roots is near 1)
First stage: robust F               4.4480    None   instrument relevance; below 10 signals a weak instrument
First stage: R-squared              0.0257    None             347 observations with a non-missing instrument

policy indicator ffr: first-stage robust F = 17.4
Instrument and stability
                               Statistic p-value                                                  Detail
Test
Stability: largest root modulus     0.9973    None  stable if below 1 (a level VAR with unit roots is near 1)
First stage: robust F              17.4410    None   instrument relevance; below 10 signals a weak instrument
First stage: R-squared              0.0486    None             347 observations with a non-missing instrument
```

How to read it

The same instrument gives very different first stages depending on the policy indicator it is paired with. For the one-year Treasury yield the robust first-stage F is 4.4 (R-squared 0.026): the instrument is weak and the engine's warning applies, so the responses would be unreliable. For the federal funds rate the F is 17.4 (R-squared 0.049), above the rule-of-thumb value of 10. Choose the policy indicator and the sample for reasons stated in advance and report the F statistic for the choice you use, not for the best of several; exogeneity of the instrument cannot be tested.

```
rp = oes.run("svar", mp, dict(spec_p, vars=["ffr", "log_ip", "log_cpi", "credit_spread"], shock="ffr"))
oes.plot(rp, "Response of", ncols=2, size=(4.2, 2.6))
for v in ["ffr", "log_ip", "log_cpi", "credit_spread"]:
    print(v, [(x[1], x[2], x[3], x[4]) for x in rows(rp, v) if x[1] in (0, 12, 24, 36)])
```

```
ffr [(0, 1.0, 1.0, 1.0), (12, 2.310243, 1.121192, 2.944452), (24, 0.920184, -0.588714, 1.851972), (36, -0.305213,
↪ -1.293137, 0.546553)]
log_ip [(0, 0.036598, -1.04351, 1.089464), (12, 1.318546, -1.918993, 4.290218), (24, 2.306006, -1.923357, 5.453535),
↪ (36, -1.011071, -4.419888, 1.645684)]
log_cpi [(0, -0.107708, -0.561073, 0.202405), (12, 0.162221, -0.707619, 1.131144), (24, 0.204431, -0.957407,
↪ 1.028917), (36, -0.261379, -1.41324, 0.759688)]
credit_spread [(0, -0.198793, -0.413386, 0.101771), (12, -0.711947, -1.272514, -0.16568), (24, 0.032432, -0.312412,
↪ 0.630101), (36, 0.544604, 0.066449, 0.877561)]
```

How to read it

The shock is scaled to raise the federal funds rate by one percentage point on impact. The rate peaks at 2.31 points after twelve months (band 1.12 to 2.94) and returns to the baseline by month 36. Industrial production rises insignificantly (1.32 after twelve months, band -1.92 to 4.29), and prices are close to zero throughout (-0.11 on impact, 0.20 after 24 months, bands including zero). The credit spread falls by 0.71 after twelve months (band -1.27 to -0.17). The estimates are too imprecise to say anything about the real effect of monetary policy; they show the workflow, and they show how wide proxy-SVAR bands can be with a short sample and a weak-to-moderate instrument.

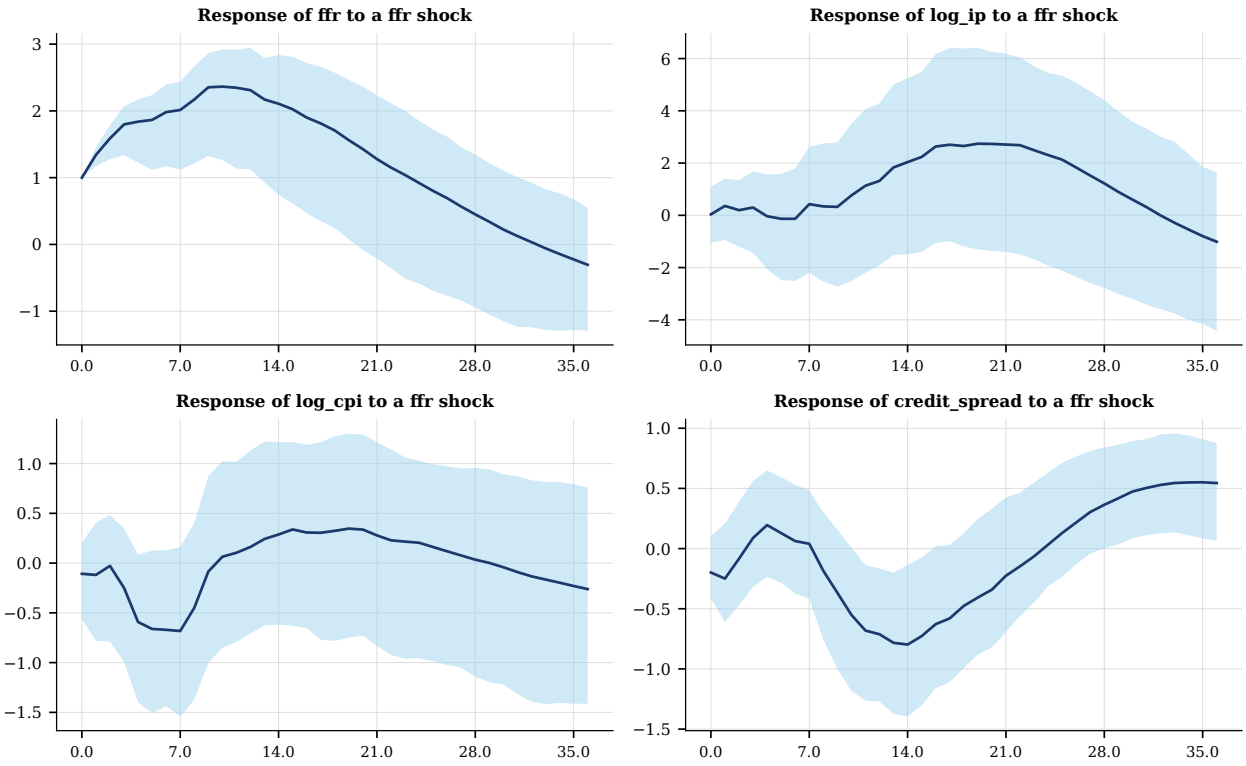


Figure 8.2. Proxy-SVAR: check the instrument before the responses. Panels: Response of ffr to a ffr shock; Response of log_ip to a ffr shock; Response of log_cpi to a ffr shock; Response of credit_spread to a ffr shock.

8.7 Robustness: lags, transformation and bootstrap

```
base = dict(vars=["gdp_growth", "unemployment"], time="date", identification="longrun", shock="gdp_growth",
  ↪ horizon=20)
variants = {"4 lags": dict(lags=4), "2 lags": dict(lags=2), "6 lags": dict(lags=6), "8 lags": dict(lags=8),
            "90% band": dict(lags=4, level="90"), "different seed": dict(lags=4, seed=777)}
out = {}
for name, v in variants.items():
    r = oes.run("svar", data, dict(base, boot=200, **v))
    g = lambda var, h: [x for x in rows(r, var) if x[1] == h][0]
    out[name] = [g("gdp_growth", 0)[2], g("unemployment", 4)[2], g("unemployment", 4)[3], g("unemployment", 4)
  ↪ [4]]
print(pd.DataFrame(out, index=["GDP growth impact", "unemployment h=4", "lower", "upper"]).T.round(3))
```

```
                GDP growth impact  unemployment h=4  lower  upper
4 lags                      2.688            -0.304 -0.447 -0.002
2 lags                      2.660            -0.248 -0.421  0.092
6 lags                      2.617            -0.295 -0.483  0.123
8 lags                      2.691            -0.351 -0.506 -0.036
90% band                    2.688            -0.304 -0.438 -0.036
different seed              2.688            -0.304 -0.484 -0.001
```

How to read it

The impact response of GDP growth is almost unchanged (2.62 to 2.69) with two, four, six or eight lags, and the response of unemployment after one year is between -0.25 and -0.35. The upper end of the 95% band is just below zero with four and eight lags and just above it with two and six, so the borderline significance of that response depends on the lag length; the 90% band excludes zero (-0.44 to -0.04). Changing the bootstrap seed moves the band limits in the second decimal place only.

8.8 Write the results

How to write it: results

A supply shock identified with the long-run restriction of Blanchard and Quah (1989) raises GDP growth by 2.69 percentage points (annualised) on impact and 0.42 points after four quarters, with a permanent effect on the level of output of about 1%. Unemployment falls by about 0.3 percentage points after one to two years, a response that is borderline significant at the 5% level (95% band -0.46 to 0.05 after four quarters, -0.47 to 0.00 after eight) and fades within five years. The supply shock explains 80% of the forecast-error variance of GDP growth and 38% of that of unemployment at a five-year horizon. A demand-side shock raises unemployment by 0.45 points after one year with no permanent effect on output. The shape of the responses is similar under recursive identifications with either ordering, which differ mainly in whether unemployment shocks move growth within the quarter (0 or -1.59 against -1.02 here). The results are stable across two to eight lags (impact on growth 2.62 to 2.69).

Report	What	Where
Model	Variables, lags, identification, bootstrap	Section 8.1
Data	Stationarity checks	Section 8.2
Specification	Lag selection, stability	Section 8.3
Estimates	Impulse responses, variance shares, long-run matrix	Section 8.4
Sensitivity	Recursive orderings against the long-run scheme	Section 8.5
Instrument	First-stage strength, responses	Section 8.6
Robustness	Lags, band level, seed	Section 8.7

References. Blanchard, O. and Quah, D. (1989), "The Dynamic Effects of Aggregate Demand and Supply Disturbances," *American Economic Review* 79. Gertler, M. and Karadi, P. (2015), "Monetary Policy Surprises, Credit Costs, and Economic Activity," *American Economic Journal: Macroeconomics* 7. Jarociński, M. and Karadi, P. (2020), "Deconstructing Monetary Policy Surprises," *American Economic Journal: Macroeconomics* 12. Mertens, K. and Ravn, M. (2013), "The Dynamic Effects of Personal and Corporate Income Tax Changes in the United States," *American Economic Review* 103. Stock, J. and Watson, M. (2012), "Disentangling the Channels of the 2007–09 Recession," *Brookings Papers on Economic Activity*. Sims, C. (1980), "Macroeconomics and Reality," *Econometrica* 48.

CHAPTER 9

Local Projections (Impulse Responses)

Applies when. You want the dynamic response of an outcome to a shock, estimated horizon by horizon, without fitting a full VAR

Overview. Use this chapter when the question is "what happens to y, h periods after a shock?" A local projection (Jordà 2005) answers it with one regression per horizon, so it needs no model of the whole system, it is robust to misspecification of the dynamics, and it extends easily to instruments, nonlinearity and panels. The chapter covers the identification of the shock, the unit-root check, the choice of controls, lags and standard errors, a comparison with a VAR, robustness and the write-up.

Data. U.S. monthly data, 1985–2019: industrial-production growth, S&P 500 return, change in the federal funds rate, change in the credit spread

Software. Python · OES engine · statsmodels

9.1 The question and the model

What is the path of y after a shock s? A local projection estimates the impulse response at each horizon $h = 0, 1, \dots, H$ with a separate regression:

$$y_{t+h} = \alpha_h + \beta_h\, s_t + \gamma_h' x_t + u_{t+h}, \qquad h = 0, 1, \dots, H,$$

where x_t holds the controls dated t or earlier (lags of y, of the shock and of other variables, and any variables ordered before the shock). The coefficient β_h is the response of y_{t+h} to the shock, and the sequence $\beta_0, \dots, \beta_H$ is the impulse response function. A VAR gets the same object by iterating one estimated system forward; a local projection estimates each horizon directly. That makes it less efficient but more robust: a misspecified lag length or a nonlinear dynamic does not propagate from short to long horizons.

Three checks before you trust a local projection

1. **Is the shock identified?** The coefficient is a causal response only if s_t is unrelated to the other things that move y. Either the shock is exogenous by construction (a narrative or high-frequency measure, an instrument), or you control for what the shock responds to within the period (the recursive, or Cholesky, assumption).
2. **Is the transformation right?** Local projections work in levels, in differences and in growth rates, but the outcome must match the question. A response of a *level* needs the level (or a cumulative sum of growth rates) on the left-hand side.
3. **Are the standard errors right?** The horizon-h errors are a moving average of order h, so ordinary standard errors are wrong. Use Newey-West errors (with bandwidth $h+1$) or lag augmentation with robust errors.

How to write it: the model

We estimate the dynamic effect of a credit-spread shock on industrial production with local projections (Jordà 2005). For each horizon $h = 0, \dots, 24$ months we regress the cumulative growth of industrial production from t to $t+h$ on the change in the credit spread at t, on the contemporaneous stock return, and on six lags of every variable. Identification is recursive: the stock market reacts to news about credit conditions within the month, whereas industrial production and the policy rate respond with a lag. Standard errors are Newey-West with bandwidth $h+1$, and the shock is scaled to one standard deviation.

9.2 Load the data and check the transformation

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller, kpss

data = open("returns.csv").read()
df = pd.read_csv("returns.csv")
print(df.describe().round(2).loc[["count", "mean", "std", "min", "max"]])
for c in ["sp500_return", "ip_growth", "d_fedfunds", "d_credit_spread"]:
    print(f"{c:16s} ADF p = {adfuller(df[c], autolag='AIC')[1]:.3f}   KPSS p = {kpss(df[c], nlags='auto')
  ↪ [1]:.3f}")
```

```
       sp500_return  ip_growth  d_fedfunds  d_credit_spread
count        420.00     420.00      420.00           420.00
mean           0.70       0.15       -0.02             0.00
std            3.54       0.61        0.19             0.17
min          -22.80      -4.48       -0.96            -0.99
max           11.35       2.04        0.87             1.45
sp500_return     ADF p = 0.000   KPSS p = 0.100
ip_growth        ADF p = 0.000   KPSS p = 0.100
d_fedfunds       ADF p = 0.001   KPSS p = 0.100
d_credit_spread  ADF p = 0.000   KPSS p = 0.100
```

How to read it

All four series are stationary: the ADF tests reject a unit root ($p \leq 0.001$) and the KPSS tests do not reject stationarity ($p = 0.10$ is the top of the table of available p-values). Stock returns, industrial-production growth and the changes in the fed funds rate and the credit spread are already in the form a local projection needs, so no further transformation is required. The sample has 420 months (1985–2019), and the credit-spread change has a standard deviation of 0.17 percentage points, so a "one-standard-deviation shock" is a 17 basis-point widening. If your shock or outcome were a trending level, use a change or a growth rate; local projections in levels are also valid (see lag augmentation in Section 9.6), but the interpretation changes.

9.3 Identify the shock and choose the controls

The choice of controls is the identification. Compare the response with and without the stock return as a contemporaneous control.

```
spec = dict(y="ip_growth", shock="d_credit_spread", contemp=["sp500_return"], controls=["d_fedfunds"], outcome
  ↪ ="cumsum",
            scale="sd", time="date", lags=6, horizon=24)
no_ctrl = oes.run("lp", data, dict(spec, contemp=[], controls=["sp500_return", "d_fedfunds"]))
with_ctrl = oes.run("lp", data, spec)
col = lambda r: [row[1] for row in r["tables"][0]["rows"]]
cmp = pd.DataFrame({"stock return only in lags": col(no_ctrl), "stock return contemporaneous": col(with_ctrl)},
  ↪  index=pd.Index(range(25), name="horizon"))
print(cmp.loc[[0, 1, 3, 6, 12, 18, 24]].round(3))
```

```
         stock return only in lags  stock return contemporaneous
horizon
0                           -0.027                        -0.060
1                           -0.107                        -0.130
3                           -0.387                        -0.320
6                           -0.652                        -0.506
12                          -0.884                        -0.638
18                          -0.980                        -0.731
24                          -0.818                        -0.554
```

How to read it

The controls change the answer. If the stock return enters only through its lags, a one-standard-deviation spread shock lowers industrial production by 0.88% after 12 months; if the contemporaneous stock return is held fixed, the fall is 0.64%. The two estimates differ because stock prices and credit spreads react to the same

news within the month. Holding the stock return fixed treats that news as already priced in by the stock market and attributes only the part of the spread movement that is not contemporaneously reflected in stock returns to the credit shock. Neither choice is "correct" without an argument. State the ordering you assume and report the other as a robustness check (Section 9.6). The baseline in the rest of the chapter holds the stock return fixed, which is the more conservative of the two.

9.4 Estimate the impulse response

```
r = oes.run("lp", data, spec)
oes.summary(r)
oes.show(r, "Impulse response")
oes.plot(r, "Response of ip_growth", ncols=1, size=(6.5, 3))
```

```
Observations (h = 0): 414
Horizons: 24
Lags: 6
Shock std. dev.: 0.1688
Peak response: -0.7352
Peak horizon: 19
Standard errors: Newey-West
Estimator: OLS
Impulse response by horizon
         Response  Std. err.  p-value  95% CI low  95% CI high  Obs.
Horizon
0         -0.0596     0.0407   0.1437     -0.1394       0.0203   414
1         -0.1300     0.0506   0.0102     -0.2293      -0.0308   413
2         -0.2079     0.0601   0.0005     -0.3257      -0.0901   412
3         -0.3201     0.0756   0.0000     -0.4682      -0.1719   411
4         -0.3402     0.0837   0.0000     -0.5043      -0.1761   410
5         -0.4300     0.0930   0.0000     -0.6123      -0.2478   409
6         -0.5061     0.0970   0.0000     -0.6961      -0.3160   408
7         -0.5377     0.1154   0.0000     -0.7639      -0.3115   407
8         -0.5803     0.1258   0.0000     -0.8269      -0.3337   406
9         -0.6044     0.1445   0.0000     -0.8876      -0.3212   405
10        -0.6286     0.1769   0.0004     -0.9752      -0.2819   404
11        -0.6120     0.1857   0.0010     -0.9760      -0.2480   403
12        -0.6381     0.2068   0.0020     -1.0434      -0.2327   402
13        -0.7178     0.2510   0.0042     -1.2097      -0.2259   401
14        -0.6707     0.2729   0.0140     -1.2057      -0.1358   400
15        -0.6677     0.2944   0.0233     -1.2447      -0.0908   399
16        -0.7190     0.3218   0.0255     -1.3496      -0.0883   398
17        -0.6896     0.3387   0.0418     -1.3535      -0.0257   397
18        -0.7307     0.3590   0.0418     -1.4344      -0.0270   396
19        -0.7352     0.3603   0.0413     -1.4413      -0.0291   395
20        -0.6390     0.3564   0.0730     -1.3375       0.0595   394
21        -0.6548     0.3765   0.0820     -1.3927       0.0832   393
22        -0.6073     0.3797   0.1097     -1.3515       0.1368   392
23        -0.6115     0.3835   0.1109     -1.3632       0.1402   391
24        -0.5541     0.3954   0.1611     -1.3291       0.2209   390
Note: Response of ip_growth to one standard deviation of the shock. Outcome: ip_growth(t) + ... + ip_growth(t+h).
↪ Each row is a separate regression.
```

How to read it

A one-standard-deviation widening of the credit spread lowers the *level* of industrial production by 0.06% in the month of the shock (not significant, p = 0.14), by 0.32% after three months, and by 0.64% after twelve months (95% band –1.04 to –0.23). The response keeps deepening to a maximum of –0.74% at 19 months and is significant at 5% from month 1 to month 19. After 20 months the band includes zero. Each row is a separate regression with 390 to 414 observations; the bands widen with the horizon (standard error 0.04 at h = 0, 0.21 at h = 12 and 0.40 at h = 24), which is the main cost of the method. In the output, "Peak response" is the largest absolute response and "Peak horizon" is the horizon at which it occurs.

9.5 Compare with a VAR

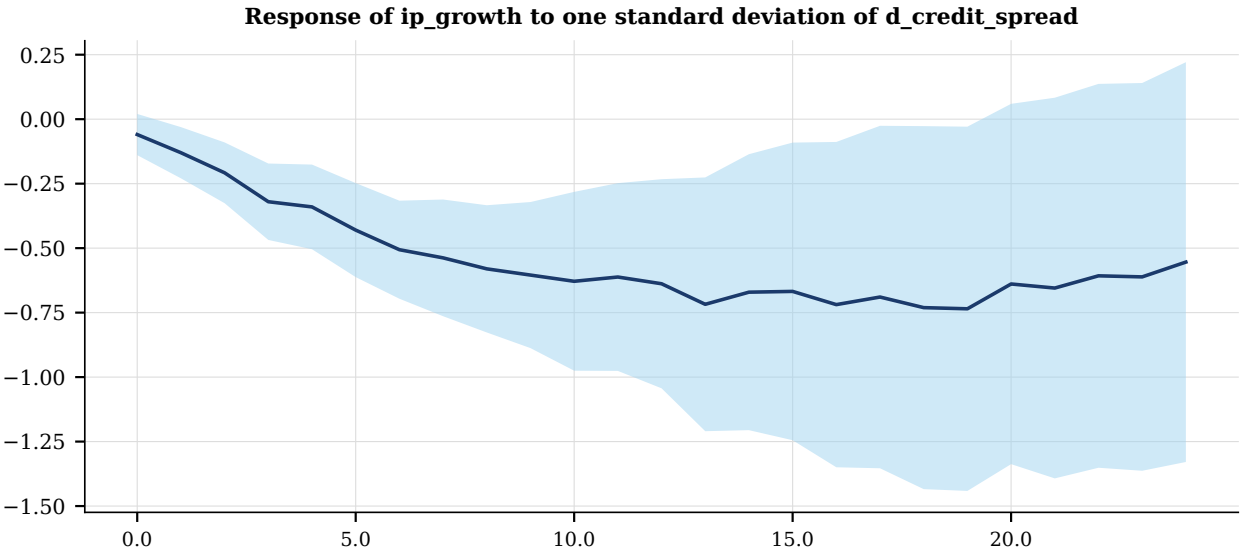


Figure 9.1. Estimate the impulse response: Response of ip_growth to one standard deviation of d_credit_spread.

A VAR with the same variables and the same ordering gives the same object by a different route. Agreement is reassuring; a large gap means that the VAR dynamics are misspecified or that the local projection is imprecise.

```
import matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR

order = ["sp500_return", "d_credit_spread", "d_fedfunds", "ip_growth"]
res = VAR(df[order]).fit(6)
chol = np.linalg.cholesky(res.sigma_u.values)
var_irf = res.irf(24).orth_cum_effects[:, order.index("ip_growth"), order.index("d_credit_spread")] / chol[1,
  ↪ 1] * df["d_credit_spread"].std()
rows = r["tables"][0]["rows"]
h = [x[0] for x in rows]; b = np.array([x[1] for x in rows]); lo = np.array([x[4] for x in rows]); hi = np.
  ↪ array([x[5] for x in rows])
fig, ax = plt.subplots(figsize=(6.5, 3))
ax.fill_between(h, lo, hi, color="#9fd3f0", alpha=.5, lw=0, label="LP 95% band")
ax.plot(h, b, lw=1.6, label="local projection"); ax.plot(h, var_irf, "--", lw=1.6, label="VAR(6)")
ax.axhline(0, color="k", lw=.6); ax.set_xlabel("months after the shock"); ax.set_title("Cumulative response of
  ↪  industrial production to a 1 sd credit-spread shock"); ax.legend()
print(pd.DataFrame({"LP": b, "VAR": var_irf}, index=pd.Index(h, name="horizon")).loc[[0, 3, 6, 12, 18, 24]].
  ↪ round(3))
```

```
            LP    VAR
horizon
0       -0.060 -0.060
3       -0.320 -0.319
6       -0.506 -0.475
12      -0.638 -0.626
18      -0.731 -0.668
24      -0.554 -0.689
```

How to read it

The VAR and the local projection agree closely for the first year: –0.32 against –0.32 at three months, –0.64 against –0.63 at twelve months. They separate at long horizons. At 18 months the figures are –0.73 and –0.67, and at 24 months the local projection shows a recovery (–0.55) while the VAR continues to deepen (–0.69). That is the usual pattern: a VAR imposes a smooth, geometrically decaying response because it extends the short-run dynamics to every horizon, whereas a local projection lets each horizon speak for itself. The difference at 24 months (0.13) is small relative to the local-projection standard error (0.40), so the two are not statistically

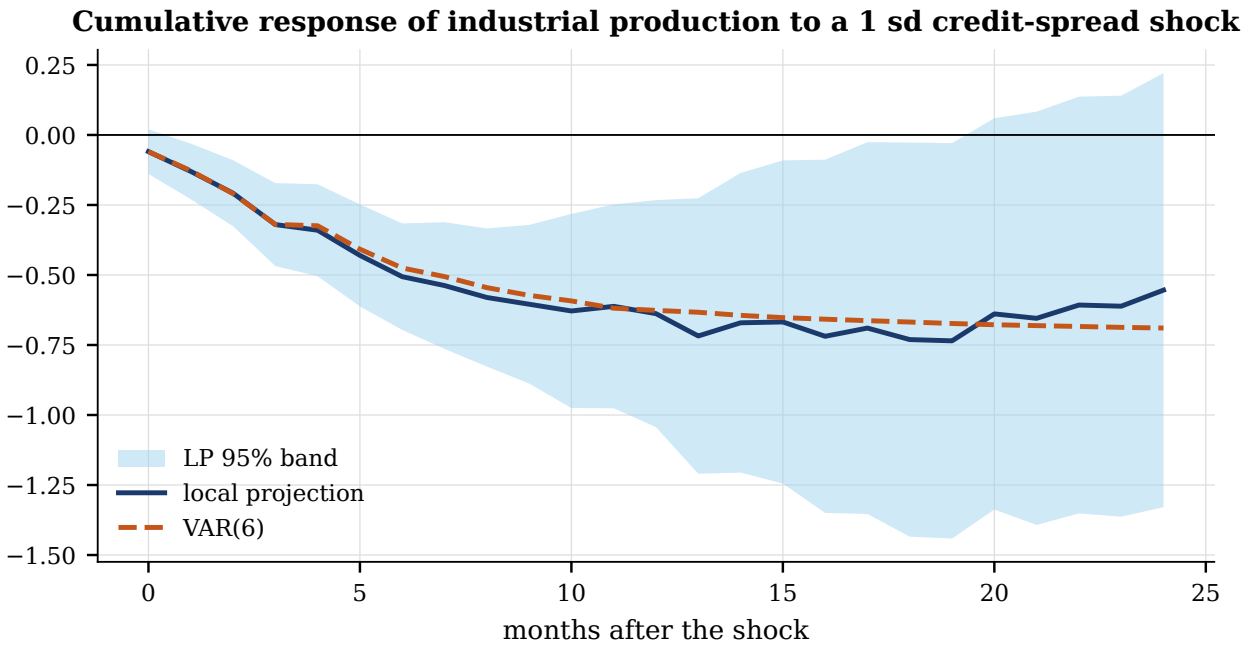


Figure 9.2. Compare with a VAR: Cumulative response of industrial production to a 1 sd credit-spread shock.

distinguishable. Report the local projection as the main result and the VAR as a cross-check, or the other way round, but state which.

9.6 Robustness: lags, lag augmentation, standard errors and the shock size

```
variants = {"baseline (6 lags, Newey-West)": {},
            "3 lags": dict(lags=3), "12 lags": dict(lags=12),
            "lag-augmented, robust errors": dict(lag_augment="yes"),
            "Newey-West bandwidth 24": dict(bandwidth=24),
            "stock return not a control": dict(contemp=[], controls=["sp500_return", "d_fedfunds"])}
out = {}
for name, v in variants.items():
    rows = oes.run("lp", data, dict(spec, **v))["tables"][0]["rows"]
    pk = min(rows, key=lambda x: x[1])
    out[name] = [rows[12][1], rows[12][2], pk[1], pk[0]]
print(pd.DataFrame(out, index=["response h=12", "std. err. h=12", "largest fall", "at horizon"]).T.round(3))
```

```
                              response h=12  std. err. h=12  largest fall  at horizon
baseline (6 lags, Newey-West)        -0.638          0.207        -0.735        19.0
3 lags                               -0.541          0.205        -0.622        19.0
12 lags                              -0.781          0.252        -0.946        19.0
lag-augmented, robust errors         -0.624          0.282        -0.728        19.0
Newey-West bandwidth 24              -0.638          0.194        -0.735        19.0
stock return not a control           -0.884          0.211        -0.980        18.0
```

How to read it

The sign and the timing are robust, the size is not. With three lags the response at twelve months is –0.54, with six lags –0.64 and with twelve lags –0.78; the lag-augmented specification with heteroskedasticity-robust errors gives –0.62 with a larger standard error (0.28 against 0.21). Widening the Newey-West bandwidth to 24 leaves the estimate unchanged and lowers the standard error slightly (0.19), so the significance does not depend on the bandwidth rule. The largest fall is at 19 months in every specification that holds the stock return fixed (18 months without it). The identification choice matters most: leaving the stock return out of

the contemporaneous controls gives –0.88 at twelve months. In the paper, say that the response at one year is between 0.5% and 0.9% depending on lags and identification, and that all specifications agree on the shape.

9.7 A level or a growth rate? The outcome option

Industrial-production growth is a monthly growth rate. With the *level* outcome the response at horizon h is the effect on growth *in* month $t + h$; with the *cumulative-sum* outcome it is the effect on the *level* of production, which is usually the question.

```
r_g = oes.run("lp", data, dict(spec, outcome="level"))
r_c = oes.run("lp", data, spec)
f = lambda r: pd.DataFrame(r["tables"][0]["rows"], columns=r["tables"][0]["columns"]).set_index("Horizon")
print(pd.DataFrame({"growth in month t+h": f(r_g)["Response"], "level: cumulative growth to t+h": f(r_c)["
  ↪ Response"],
                    "std. err. (level)": f(r_c)["Std. err."]}).round(3).loc[[0, 1, 2, 3, 6, 12, 18, 24]])
```

```
         growth in month t+h  level: cumulative growth to t+h  std. err. (level)
Horizon
0                     -0.060                          -0.060             0.041
1                     -0.071                          -0.130             0.051
2                     -0.078                          -0.208             0.060
3                     -0.112                          -0.320             0.076
6                     -0.072                          -0.506             0.097
12                    -0.025                          -0.638             0.207
18                    -0.038                          -0.731             0.359
24                     0.057                          -0.554             0.395
```

How to read it

The monthly growth effects are small (–0.06 to –0.11 percentage points per month for the first six months) and fade by the end of the first year (–0.03 at twelve months, +0.06 at 24). Read on their own, the late-horizon figures suggest that the shock has no lasting effect. The cumulative-sum outcome shows what they add up to: a level of industrial production that is 0.64% lower after twelve months and still 0.55% lower after 24. Choose the outcome to match the question, and say in the text whether the response is of a growth rate or of a level.

When the shock is not observed: LP-IV

If you have an external instrument for the shock (a high-frequency surprise, a narrative series), add it with the `instrument` option. The shock is then estimated by two-stage least squares at every horizon (LP-IV; Stock and Watson 2018), and the first-stage F statistic is reported. Check that the F statistic is well above 10 before you read the response.

```
r_iv = oes.run("lp", data, dict(spec, instrument="spread_instrument"))
oes.show(r_iv, "First stage")
```

9.8 Write the results

How to write it: results

A one-standard-deviation (17 basis-point) widening of the credit spread lowers the level of industrial production by 0.32% after three months (standard error 0.08) and by 0.64% after twelve months (standard error 0.21). The response reaches its maximum of –0.74% after 19 months, and is statistically significant at the 5% level from month 1 to month 19. The local-projection response is close to that of a VAR(6) with the same ordering for the first year; at 24 months the local projection shows a partial recovery (–0.55) while the VAR does not (–0.69). The size of the response at twelve months ranges from –0.54 to –0.88 across lag lengths and identifying assumptions; its sign and timing are the same in all of them.

Report	What	Where
Model	Shock, outcome, controls, lags, horizon, standard errors	Section 9.1
Data	Transformation and unit-root checks	Section 9.2
Identification	Contemporaneous controls and their effect on the response	Section 9.3
Estimates	Impulse response table and figure with 95% bands	Section 9.4
Cross-check	VAR with the same ordering	Section 9.5
Robustness	Lags, lag augmentation, bandwidth, contemporaneous control	Section 9.6
Outcome	Level versus cumulative response	Section 9.7

References. Jordà (2005), *American Economic Review*; Plagborg-Møller and Wolf (2021), *Econometrica*; Montiel Olea and Plagborg-Møller (2021), *Econometrica*; Stock and Watson (2018), *Economic Journal*; Ramey (2016), *Handbook of Macroeconomics*; Li, Plagborg-Møller and Wolf (2024), *Journal of Econometrics*.

CHAPTER 10

Threshold Regression and SETAR: Regimes Defined by a Variable

Applies when. The relationship or the dynamics change abruptly when a variable (often the series' own past) crosses a level, and you want to find the level

Overview. Use this chapter when behaviour differs above and below a level that you do not know: inflation that is persistent only when it is high, a policy rule that reacts only past a limit, growth that slows after a downturn. A threshold regression (Tong's TAR, and SETAR when the threshold variable is the series' own lag) estimates the level by least squares, tests the null of no threshold with a bootstrap, gives a confidence interval for the level, and estimates each regime separately. The chapter covers the model, the test, the regimes, robustness to lags, delay and trimming, and the text for the paper.

Data. U.S. quarterly data, 1960–2019: GDP-deflator inflation

Software. Python · OES engine · statsmodels

10.1 The question and the model

Does the process switch between regimes when a variable crosses a level, and where is that level? In a threshold regression the coefficients depend on whether a threshold variable s_t lies below or above a threshold γ:

$$y_t = x_t'\beta_1 \, \mathbf{1}(s_t \le \gamma) + x_t'\beta_2 \, \mathbf{1}(s_t > \gamma) + e_t.$$

When $s_t = y_{t-d}$ and x_t holds lags of y_t, the model is the self-exciting threshold autoregression (SETAR); d is the delay. The regime is determined by the data, not by the calendar, which is the difference from a structural break (see Chapter 12): the same regime can occur in 1965 and in 1985. The switch is abrupt. If you expect a gradual move between regimes, use Chapter 11.

The threshold γ is estimated as the value that minimises the sum of squared residuals, searched over observed values of s_t that leave at least a fraction ε of the sample in each regime. Because γ does not exist under the null of linearity, the usual F test is not valid. Hansen (1996) shows how to bootstrap its distribution, and the confidence interval for γ is the set of values whose likelihood-ratio statistic stays below 7.35 (Hansen 2000).

Four checks before you trust a threshold

1. **Does the bootstrap reject linearity?** Without rejection the threshold estimate is not reliable, whatever it looks like. Report the p-value.
2. **Is the threshold stable?** Re-estimate with other lags, delays and trimming. A threshold that moves everywhere is not identified.
3. **Do the regimes contain enough observations?** A regime with a handful of observations gives imprecise coefficients and can be driven by a few episodes.
4. **Is the switch really abrupt?** If the residuals look better under a gradual change, use the smooth-transition model.

How to write it: the model

We test for threshold nonlinearity in U.S. inflation with a self-exciting threshold autoregression (Tong 1978; Hansen 1996, 1997). Quarterly inflation, the GDP-deflator rate, follows an AR(2) whose intercept and coefficients may differ between two regimes defined by whether inflation d quarters earlier was above a threshold γ. The delay d is chosen from one to four quarters and the threshold is searched over values that leave at least 15% of the 236 observations in each regime. We test linearity with the sup-F statistic and a fixed-regressor bootstrap with 999 draws, which accounts for the search over γ and over d, and we report a confidence interval for γ from the likelihood-ratio statistic.

10.2 Load the data and look at the series

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller, acf

data = open("yield.csv").read()
df = pd.read_csv("yield.csv")
print(df.date.iloc[0], "to", df.date.iloc[-1], "|", len(df), "quarters")
print(df.inflation.describe().round(2).loc[["mean", "std", "min", "25%", "50%", "75%", "max"]])
print(f"ADF p = {adfuller(df.inflation, autolag='AIC')[1]:.3f}   first autocorrelation = {acf(df.inflation,
  ↪ nlags=1)[1]:.2f}")
```

```
1960Q1 to 2019Q4 | 240 quarters
mean     3.19
std      2.31
min     -0.53
25%      1.73
50%      2.42
75%      3.99
max     12.14
Name: inflation, dtype: float64
ADF p = 0.190   first autocorrelation = 0.90
```

How to read it

The sample runs from 1960Q1 to 2019Q4 (240 quarters; 236 remain after allowing for two lags and a delay of up to four quarters). Inflation averages 3.19% with a standard deviation of 2.31, ranges from –0.53% to 12.14% and is skewed: the median is 2.42%, well below the mean, because of the 1970s. The series is highly persistent (first autocorrelation 0.90, ADF p = 0.19). That matters for a threshold model: a persistent series that spends a long spell at a high level can look like a threshold process, so the bootstrap test and the robustness checks below carry the weight.

10.3 Test for a threshold

```
spec = dict(y="inflation", time="date", ar_lags=2, search_delay="yes", max_delay=4,
            trim="0.15", max_thresholds="2", boot=999, seed=12345)
r = oes.run("tar", data, spec)
oes.summary(r)
oes.show(r, "Tests for a threshold")
```

```
Observations: 236
Regressors per regime: 3
Threshold variable: inflation(-3)
Delay: 3
Trimming: 0.15
Sup-F against linearity: 25.916
Bootstrap p (linearity): 0.081
Thresholds found: 1
Threshold 1: 4.232
Tests for a threshold
                                        Statistic  Bootstrap p-value  5% critical value
↪                Detail
Test
```

```
Linearity against one threshold (sup-F)     25.916              0.081             28.556
↪    999 bootstrap draws
One threshold against two (sup-F)           10.036              0.130             12.322  first threshold re-
↪ estimated in each of 999 draws
Note: The threshold is not identified under the null of linearity, so the F statistic does not follow an F
↪ distribution. The p-value comes from Hansen's (1996) fixed-regressor bootstrap. The delay is searched over 1 to 4
↪ in the data and in every bootstrap draw.
```

How to read it

The sup-F statistic is 25.9. The bootstrap p-value is 0.081 with 999 draws, with the delay searched from one to four quarters in the data and in every draw, and the 5% bootstrap critical value is 28.6. Linearity is therefore rejected at 10% but not at 5%: the evidence for a threshold is suggestive, not conclusive. The search chooses a delay of three quarters. The test of one threshold against two does not reject (statistic 10.0, p = 0.13), so the model has one threshold and two regimes. Note that the bootstrap already accounts for the search over γ and d; testing at a delay picked after looking at the data would give a p-value that is too small.

10.4 The threshold, its interval and the regimes

```
oes.show(r, "Estimated thresholds")
oes.show(r, "Coefficients by regime")
oes.show(r, "The regimes")
oes.show(r, "Fit against the linear model")
```

```
Estimated thresholds
           Estimate 95% confidence interval  Observations at or below  Observations above
Threshold
1             4.232         4.012 to 5.5124                       180                  56
Note: The interval collects every candidate threshold whose likelihood-ratio statistic is below 7.35 (Hansen 2000).
↪ It is often wide and can be asymmetric.

Coefficients by regime (2 regimes)
                     Regime 1          Regime 2
Coefficient
const           0.4147 (0.124)   1.5498 (0.4617)
inflation(-1)  0.5135 (0.0755)   0.8824 (0.1195)
inflation(-2)  0.2991 (0.0618)  -0.1185 (0.1396)
Note: Newey-West (HAC) standard errors in parentheses, from one regression with the regime-specific columns.

The regimes
                          Rule  Observations  Share  Sum of AR coefficients  Implied mean
Regime
1       inflation(-3) <= 4.232           180  0.763                  0.8126        2.2125
2        inflation(-3) > 4.232            56  0.237                  0.7639        6.5644
Note: A sum of AR coefficients near or above 1 in a regime means the series is persistent or explosive there; the
↪ implied mean is c / (1 - sum) when the regime is stationary.

Fit against the linear model
                            SSR  Residual std. dev.     AIC
Model
Linear                 228.2781              0.9898  -1.851
Threshold (2 regimes)  205.1609              0.9445 -19.049
Note: AIC counts the thresholds as parameters. A lower value favours the threshold model.
```

How to read it

The threshold is 4.23% inflation, three quarters earlier, with a 95% interval from 4.01% to 5.51%. The lower regime holds 180 quarters and the upper regime 56 (24%). Below the threshold the AR coefficients are 0.51 (standard error 0.076) and 0.30 (0.062), they sum to 0.81, and the implied mean is 2.21%. Above it they are 0.88 (0.12) and –0.12 (0.14), sum 0.76, intercept 1.55 (0.46), and the implied mean is 6.56%. Persistence is similar in the two regimes; what differs is the level the series gravitates to, which is the intercept. The threshold model lowers the residual standard deviation from 0.990 to 0.945 and the sum of squared residuals from 228.3 to 205.2. AIC falls from –1.9 to –19.0 even after counting the extra coefficients and the threshold as parameters.

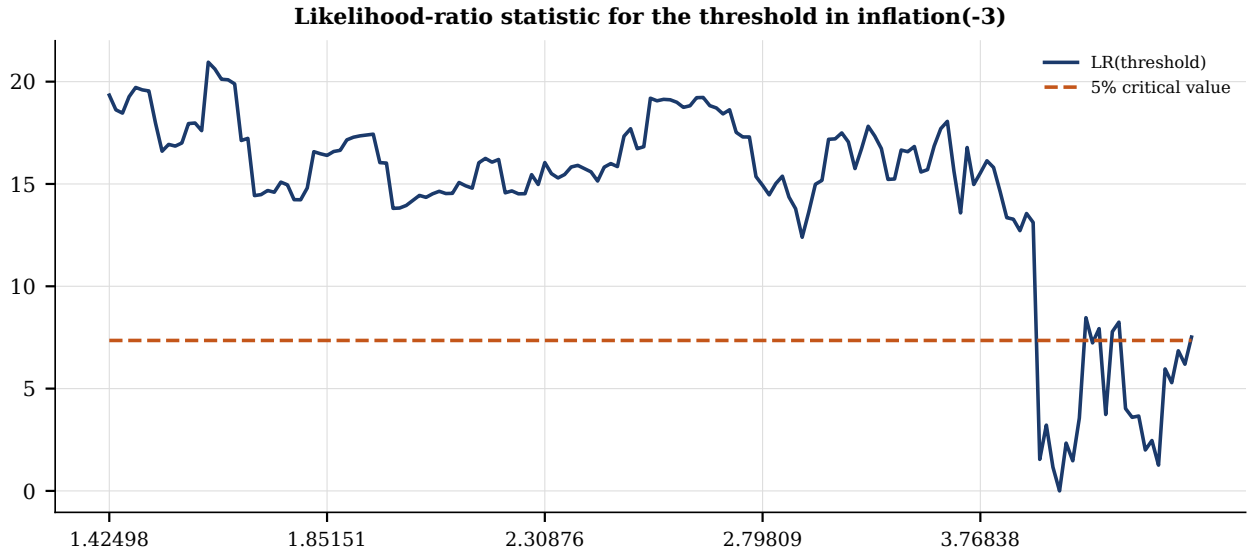


Figure 10.1. Look at the likelihood and at the regimes over time: Likelihood-ratio statistic for the threshold in inflation(-3).

10.5 Look at the likelihood and at the regimes over time

```
oes.plot(r, "Likelihood-ratio statistic for the threshold", ncols=1, size=(6.5, 3))
oes.plot(r, "inflation(-3) and the estimated", ncols=1, size=(6.5, 3))

s = {x["label"]: x["value"] for x in r["summary"]}
high = df.inflation.shift(s["Delay"]) > s["Threshold 1"]
decade = (df.date.str[:4].astype(int) // 10) * 10
print("Quarters in the high-inflation regime, by decade")
print(high.groupby(decade).sum().astype(int).to_string())
```

```
Quarters in the high-inflation regime, by decade
date
1960     4
1970    34
1980    16
1990     3
2000     0
2010     0
```

How to read it

The likelihood-ratio statistic is far above the 5% critical value of 7.35 (between about 12 and 21) for almost every candidate threshold, and it falls below it only for values between 4.0% and 5.5%. That is the confidence interval: one well-defined region, with a sharp edge, as an abrupt switch implies. The second chart shows where the regimes sit. The upper regime contains 4 quarters in the 1960s, 34 in the 1970s, 16 in the 1980s, 3 in the 1990s and none since 2000. Fifty of the 56 high-regime quarters therefore belong to the Great Inflation. This is the main caution: the threshold may be picking up that episode, which Chapter 12 dates with mean shifts in 1973Q2 and 1982Q4 and a high-inflation mean of 7.4% between them. The two readings agree on what happened. They differ on why: a threshold model says high inflation sustains itself, while a break model says the regime was a period.

10.6 Does the answer depend on the lags, the delay and the trimming?

```
out = {}
variants = {"baseline (AR(2), delay 1 to 4)": {}, "AR(1)": dict(ar_lags=1), "AR(3)": dict(ar_lags=3), "AR(4)":
  ↪ dict(ar_lags=4),
            "delay fixed at 1": dict(search_delay="no", delay=1), "delay fixed at 3": dict(search_delay="no",
  ↪ delay=3),
```

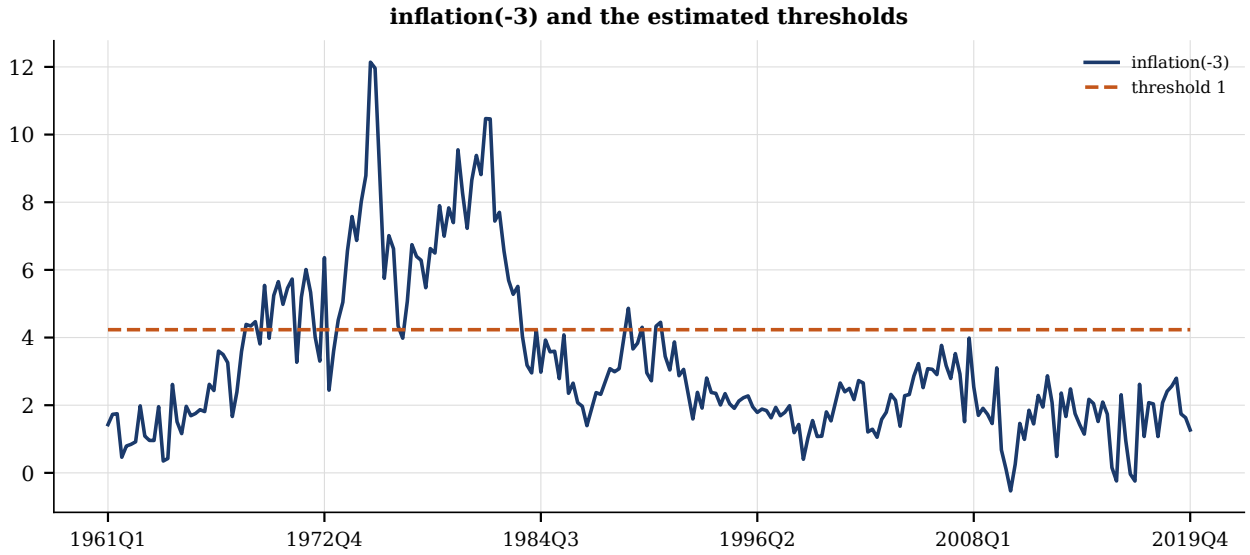


Figure 10.2. Look at the likelihood and at the regimes over time: inflation(-3) and the estimated thresholds.

```
            "trim 10%": dict(trim="0.10"), "trim 20%": dict(trim="0.20"), "trim 25%": dict(trim="0.25")}
for name, v in variants.items():
    rr = oes.run("tar", data, dict(spec, max_thresholds="1", boot=499, **v))
    s = {x["label"]: x["value"] for x in rr["summary"]}
    out[name] = (s["Delay"], s["Threshold 1"], s["Sup-F against linearity"], s["Bootstrap p (linearity)"])
print(pd.DataFrame(out, index=["delay", "threshold", "sup-F", "bootstrap p"]).T.round(3))
```

```
                             delay  threshold   sup-F  bootstrap p
baseline (AR(2), delay 1 to 4)   3.0      4.232  25.916        0.080
AR(1)                            2.0      3.978  23.755        0.028
AR(3)                            4.0      5.336  22.216        0.162
AR(4)                            2.0      4.451  24.476        0.188
delay fixed at 1                 1.0      1.395  12.256        0.320
delay fixed at 3                 3.0      4.232  26.029        0.020
trim 10%                         4.0      5.750  29.676        0.076
trim 20%                         3.0      4.232  25.916        0.046
trim 25%                         3.0      4.012  24.213        0.044
```

How to read it

Six of the nine specifications put the threshold between 3.98% and 4.45%: the baseline (4.23%), AR(1) (3.98%), AR(4) (4.45%), a delay fixed at three (4.23%) and 20% and 25% trimming (4.23% and 4.01%). The evidence for it, however, depends on the lag order. The bootstrap p-value is 0.03 with one lag, 0.08 with two, 0.16 with three and 0.19 with four, because more lags absorb part of the nonlinearity. A delay fixed at one gives no threshold ($p = 0.32$, threshold 1.40%), so the delay matters. The p-values for a fixed delay (0.020 for a delay of three) are smaller than the baseline because they ignore the search over delays, so report the baseline. With 10% trimming the search picks a delay of four and a threshold of 5.75% ($p = 0.076$). Taken together: a threshold near 4% is the most stable finding, and it is moderately supported.

10.7 Is a second threshold needed, and what if the threshold variable is another series?

```
two = oes.run("tar", data, dict(spec, trim="0.10"))
oes.show(two, "Tests for a threshold")
ext = oes.run("tar", data, dict(y="inflation", x=["unemployment"], threshold="unemployment", time="date",
                                ar_lags=1, delay=1, boot=499, max_thresholds="1"))
oes.show(ext, "Tests for a threshold")
oes.show(ext, "Estimated thresholds")
oes.show(ext, "Coefficients by regime")
```

```
Tests for a threshold
```

```
                                      Statistic  Bootstrap p-value  5% critical value
↪                Detail
Test
Linearity against one threshold (sup-F)     29.676              0.081             32.551
↪    999 bootstrap draws
One threshold against two (sup-F)            4.533              0.791             16.449  first threshold re-
↪ estimated in each of 999 draws
Note: The threshold is not identified under the null of linearity, so the F statistic does not follow an F
↪ distribution. The p-value comes from Hansen's (1996) fixed-regressor bootstrap. The delay is searched over 1 to 4
↪ in the data and in every bootstrap draw.

Tests for a threshold
                                      Statistic  Bootstrap p-value  5% critical value               Detail
Test
Linearity against one threshold (sup-F)      5.794              0.858             22.289  499 bootstrap draws
Note: The threshold is not identified under the null of linearity, so the F statistic does not follow an F
↪ distribution. The p-value comes from Hansen's (1996) fixed-regressor bootstrap.

Estimated thresholds
           Estimate 95% confidence interval  Observations at or below  Observations above
Threshold
1               5.2           4.3333 to 7.6                        81                 158
Note: The interval collects every candidate threshold whose likelihood-ratio statistic is below 7.35 (Hansen 2000).
↪ It is often wide and can be asymmetric.

Coefficients by regime (2 regimes)
                       Regime 1          Regime 2
Coefficient
const          -0.8846 (0.9536)   0.7541 (0.2979)
inflation(-1)   1.0072 (0.0887)   0.8782 (0.0405)
unemployment     0.226 (0.1854)  -0.0595 (0.0409)
Note: Newey-West (HAC) standard errors in parentheses, from one regression with the regime-specific columns.
```

How to read it

A second threshold is not supported. With 10% trimming, so that smaller regimes are allowed, the sup-F test of one threshold against two has p = 0.79. The second experiment replaces the threshold variable with the unemployment rate (lagged one quarter), in a regression of inflation on its lag and unemployment. Linearity is not rejected (sup-F 5.8, p = 0.86). The estimated threshold of 5.2% (interval 4.3% to 7.6%) is then not meaningful, and the coefficients by regime should not be interpreted. This is the output you should expect when there is no threshold: a threshold estimate always exists, and only the bootstrap test says whether it is real.

10.8 Write the results

How to write it: results

We find moderate evidence of threshold nonlinearity in U.S. inflation. The bootstrap sup-F test of linearity gives a statistic of 25.9 and a p-value of 0.081 (999 draws, with the delay searched over one to four quarters), so linearity is rejected at 10% but not at 5%. The estimated threshold is 4.23% inflation, three quarters earlier (95% interval 4.01% to 5.51%). When inflation was at or below this level (180 quarters), the AR(2) coefficients sum to 0.81 and the implied mean is 2.2%; above it (56 quarters) they sum to 0.76 and the implied mean is 6.6%. Fifty of the 56 high-regime quarters lie in the 1970s and 1980s, so the threshold may reflect the Great Inflation, which the Bai–Perron procedure dates with mean shifts in 1973Q2 and 1982Q4. The threshold near 4% is stable across lag orders, delays and trimming, but its statistical support weakens with three or four lags (p = 0.16 and 0.19) and we treat it as suggestive. We find no evidence of a second threshold (p = 0.79), nor of a threshold in the unemployment rate (p = 0.86).

Report	What	Where
Model	Regressors, threshold variable, delay, trimming	Section 10.1
Data	Persistence of the series	Section 10.2
Test	Sup-F statistic, bootstrap p-value, number of draws	Section 10.3
Threshold	Estimate, confidence interval, regimes and their coefficients	Section 10.4
Diagnostics	Likelihood-ratio plot, time in each regime	Section 10.5
Robustness	Lags, delay and trimming	Section 10.6
Extensions	Second threshold, another threshold variable	Section 10.7

References. Tong, H. (1978), "On a Threshold Model," in *Pattern Recognition and Signal Processing*. Hansen, B. E. (1996), "Inference When a Nuisance Parameter Is Not Identified Under the Null Hypothesis," *Econometrica* 64. Hansen, B. E. (1997), "Inference in TAR Models," *Studies in Nonlinear Dynamics and Econometrics* 2. Hansen, B. E. (2000), "Sample Splitting and Threshold Estimation," *Econometrica* 68. Bai, J. (1997), "Estimating Multiple Breaks One at a Time," *Econometric Theory* 13.

CHAPTER 11

Smooth-Transition Regression (LSTAR and ESTAR)

Applies when. The coefficients depend on the state of the economy, but they change gradually, not abruptly, as a variable moves through a range

Overview. Use this chapter when behaviour depends on a state variable and you expect a gradual move between regimes: a central bank that reacts to inflation more strongly in a boom, a consumer who responds differently to income in good and bad times, an exchange rate that reverts only when far from parity. A smooth-transition regression (Teräsvirta's LSTAR and ESTAR) lets the coefficients move with a transition function of that variable, estimates the location and speed of the move, tests linearity, and chooses between the two forms. The chapter covers the model, the linearity tests, the choice of the transition variable and delay, estimation, the regimes, robustness, and the text for the paper.

Data. U.S. quarterly data, 1960–2019: the federal funds rate, inflation and the output gap

Software. Python · OES engine · statsmodels · SciPy

11.1 The question and the model

Do the coefficients depend on the level of a state variable, and how fast do they move? In a smooth-transition regression

$$y_t = x_t'\phi + (x_t'\theta)\, G(s_t;\gamma,c) + e_t,$$

the transition function G rises from 0 to 1 as the transition variable s_t moves through a range. In the logistic form (LSTAR), $G = 1/\{1+\exp[-\gamma(s_t-c)/\sigma_s]\}$: the regimes at low and high s_t differ, and c is where they are halfway between. In the exponential form (ESTAR), $G = 1-\exp[-\gamma(s_t-c)^2/\sigma_s^2]$: the two outer regimes are alike and differ from the middle. The slope parameter γ (scaled by the standard deviation of s_t) sets the speed. As $\gamma \to \infty$ the logistic form becomes the threshold model of Chapter 10.

A linear model is nested in two ways, $\theta = 0$ or $\gamma = 0$, so the usual tests do not apply. Teräsvirta (1994) replaces G by a third-order Taylor approximation, which turns the linearity test into an F test, and the pattern of rejections chooses between LSTAR and ESTAR. Given γ and c, the model is linear in the other coefficients, so estimation concentrates the sum of squares on (γ, c), searches a grid and polishes with a simplex search.

Four checks before you trust a smooth-transition model

1. **Is linearity rejected for the transition variable you use?** If not, the nonlinear estimates are not determined. Try the candidate variables and delays and report the search.
2. **Is γ estimated away from its limits?** A very large γ means an abrupt switch, and its standard error is unreliable because the likelihood is almost flat. A very small γ makes the model a polynomial in s_t, with no meaningful location.
3. **Does the location c lie inside the data and away from the edge of its range?** It is searched between the 10th and 90th percentiles of s_t, so that both regimes have observations.
4. **Are the residuals clean?** Remaining serial correlation means the dynamics are too short. Add lags before interpreting the regimes.

How to write it: the model

We estimate a nonlinear monetary policy rule with a smooth-transition regression (Teräsvirta 1994). The federal funds rate depends on its own lag, inflation and the output gap, and all coefficients, including the intercept, move with a logistic function of the output gap lagged three quarters. The transition function has a speed γ and a location c. We test linearity with Teräsvirta's third-order Taylor approximation for every candidate transition variable and delay, choose the logistic or the exponential form by the sequence of tests, estimate γ and c by nonlinear least squares on 237 quarterly observations from 1960 to 2019, and report Newey–West standard errors.

11.2 Load the data and look at the series

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller

data = open("taylor.csv").read()
df = pd.read_csv("taylor.csv")
print(df.date.iloc[0], "to", df.date.iloc[-1], "|", len(df), "quarters")
print(df[["fedfunds", "inflation", "output_gap"]].describe().round(2).loc[["mean", "std", "min", "max"]])
print(df[["fedfunds", "inflation", "output_gap"]].corr().round(2))
print(f"ADF p, federal funds rate = {adfuller(df.fedfunds, autolag='AIC')[1]:.3f}")
```

```
1960Q1 to 2019Q4 | 240 quarters
      fedfunds  inflation  output_gap
mean      4.99       3.20       -0.02
std       3.65       2.19        1.43
min       0.07       0.13       -4.80
max      17.78      10.46        3.72
            fedfunds  inflation  output_gap
fedfunds        1.00       0.76        0.20
inflation       0.76       1.00        0.03
output_gap      0.20       0.03        1.00
ADF p, federal funds rate = 0.330
```

How to read it

The sample runs from 1960Q1 to 2019Q4 (240 quarters). The federal funds rate averages 4.99% with a standard deviation of 3.65 and ranges from 0.07% to 17.78%. Inflation averages 3.20% (standard deviation 2.19). The output gap averages about zero (standard deviation 1.43) and runs from –4.80 to 3.72 points, so large positive gaps are rare. The funds rate is correlated with inflation (0.76) and weakly with the output gap (0.20), and it is persistent (ADF p = 0.33), so a lag of the funds rate is included to capture interest-rate smoothing.

11.3 Test linearity and choose the transition variable and the delay

```
spec = dict(y="fedfunds", x=["inflation", "output_gap"], time="date", ar_lags=1, form="auto", trim="0.10")
rows = []
for sv in ["output_gap", "inflation"]:
    for d in [1, 2, 3, 4]:
        rr = oes.run("str", data, dict(spec, transition=sv, delay=d))
        lin = [t for t in rr["tables"] if t["title"].startswith("Linearity")][0]["rows"][0]
        s = {x["label"]: x["value"] for x in rr["summary"]}
        rows.append((sv, d, lin[1], lin[2], s["Transition function"], s["gamma"], s["c"], s["AIC gain over
  ↪ linear"]))
print(pd.DataFrame(rows, columns=["transition", "delay", "F", "p", "form", "gamma", "c", "AIC gain"]).
  ↪ to_string(index=False))
```

```
transition  delay     F      p     form   gamma      c  AIC gain
output_gap      1 2.645 0.0025 logistic   1.388 0.6613     11.25
output_gap      2 3.398 0.0001 logistic   3.025 1.4302     16.95
output_gap      3 4.728 0.0000 logistic  10.022 1.4656     57.43
output_gap      4 2.854 0.0011 logistic 108.469 1.6477     17.57
 inflation      1 6.123 0.0000 logistic   0.100 6.5359     15.74
```

```
inflation      2 3.001 0.0006 logistic   1.877 6.5469      5.64
inflation      3 2.706 0.0020 logistic   8.291 6.5580      7.28
inflation      4 2.102 0.0178 logistic 499.998 6.4049      5.62
```

How to read it

Linearity is rejected at 5% for all eight combinations of transition variable and delay, and at 1% for seven. Two things guide the choice. First, the quality of the fit: with the output gap lagged three quarters the AIC gain over the linear model is 57.4, against 11 to 18 for the other output-gap delays and 6 to 16 for inflation. Second, whether the transition is identified. With inflation as the transition variable the slope γ sits at its lower limit of 0.1 for a delay of one quarter (so the model is close to a polynomial in inflation, with a location of 6.5% that carries no meaning) and at its upper limit of 500 for a delay of four. With the output gap at a delay of three quarters, γ is 10.0 and the location is 1.47 points, inside the range of the data. We use the output gap lagged three quarters. The search is exploratory, and the p-values of the linearity tests do not account for it; the size of the AIC gain is the more informative number.

11.4 Estimate the model

```
spec = dict(y="fedfunds", x=["inflation", "output_gap"], transition="output_gap", time="date", ar_lags=1,
  ↪ delay=3, form="auto", trim="0.10")
r = oes.run("str", data, spec)
oes.summary(r)
oes.show(r, "Linearity tests")
oes.show(r, "Transition function")
```

```
Observations: 237
Transition variable: output_gap(-3)
Transition function: logistic
Linearity F (all terms): 4.728
Linearity p: 0.0
gamma: 10.022
c: 1.4656
c at the edge of its range: no
R-squared: 0.9661
AIC gain over linear: 57.43
Linearity tests (Terasvirta)
                                                F  p-value  Restrictions Detail
Test
H0: linear (all nonlinear terms zero)           4.728   0.0000            12
H04: third-order terms zero                     3.049   0.0179             4
H03: second-order terms zero, given third zero  3.348   0.0110             4
H02: first-order terms zero, given higher zero  7.105   0.0000             4
Note: Terasvirta's (1994) sequence: if the second-order test (H03) has the smallest p-value among the three, the
↪ exponential form is chosen; otherwise the logistic form. Here the rule chooses logistic.

Transition function (logistic)
                                                        Estimate  Std. error     t
Parameter
gamma (smoothness, per s.d. of the transition variable)   10.0224      7.5129  1.33
c (location, in units of output_gap(-3))                   1.4656      0.1562  9.38
Note: Standard errors are robust (Newey-West) and rely on a Gauss-Newton approximation. A large gamma makes the
↪ transition abrupt and its standard error unreliable, because the likelihood is almost flat in gamma; the location
↪ c is then the better-identified quantity.
```

How to read it

The linearity test rejects strongly: F = 4.73 for the 12 restrictions, p < 0.0001. In the sequence that chooses the form, the first-order test (H02) has the smallest p-value (below 0.0001, against 0.011 for H03 and 0.018 for H04), so the rule chooses the logistic form. The speed γ is 10.0 (standard error 7.5, so it is not precisely estimated, as expected when the transition is quick) and the location is $c = 1.47$ output-gap points (standard error 0.16, t = 9.4). A γ of 10 means that the transition function rises from about 0.12 to 0.88 as the lagged output gap moves through about half a point around 1.47, which is roughly one standard deviation above the mean gap. So the switch is quick but not a step.

11.5 The coefficients in the two regimes

```
oes.show(r, "Coefficients")
imp = [t for t in r["tables"] if t["title"].startswith("Coefficients in the two")][0]["rows"]
val = {row[0]: (float(row[1].split()[0]), float(row[2].split()[0])) for row in imp}
for k, name in enumerate(["low output gap (G = 0)", "high output gap (G = 1)"]):
    print(f"Long-run response of the funds rate to inflation, {name}: {val['inflation'][k] / (1 - val['
  ↪ fedfunds(-1)'][k]):.2f}")
```

```
Coefficients
             Linear part (G = 0)  p-value Nonlinear part (G = 1 adds)  p-value
Variable
const            0.1756 (0.1032)   0.0903            -0.3787 (0.5893)   0.5211
fedfunds(-1)     0.9629 (0.0206)   0.0000            -0.5822 (0.2659)   0.0296
inflation        0.0254 (0.0412)   0.5382              0.9023 (0.375)   0.0169
output_gap       0.2678 (0.0309)   0.0000             0.0348 (0.1444)   0.8100
Note: The coefficient at transition value G is the linear part plus G times the nonlinear part. Robust standard
↪ errors in parentheses.

Coefficients in the two extreme regimes
             G = 0 (low output_gap(-3)) G = 1 (high output_gap(-3))
Variable
const                   0.1756 (0.1032)            -0.2031 (0.5617)
fedfunds(-1)            0.9629 (0.0206)             0.3808 (0.2606)
inflation               0.0254 (0.0412)             0.9277 (0.3759)
output_gap              0.2678 (0.0309)              0.3026 (0.137)
Note: G = 0 and G = 1 are the limits of the transition function; most observations lie between them.

Long-run response of the funds rate to inflation, low output gap (G = 0): 0.68
Long-run response of the funds rate to inflation, high output gap (G = 1): 1.50
```

How to read it

The linear part is the regime when the lagged output gap is low (G = 0) and the nonlinear part is what high gaps add. When the gap was low, the funds rate adjusts slowly (coefficient on its lag 0.963, standard error 0.021), does not respond to inflation (0.025, 0.041) and responds to the output gap with 0.268 (0.031). When the gap was high (G = 1), the lag coefficient falls to 0.381 (0.261), the response to inflation rises to 0.928 (0.376) and the response to the gap is 0.303 (0.137). The changes in the lag and inflation coefficients are significant at 5% (p = 0.030 and 0.017). The changes in the intercept and the output-gap coefficient are not (p = 0.52 and 0.81). The long-run response of the funds rate to inflation, the inflation coefficient divided by one minus the lag coefficient, is 0.68 in the low regime and 1.50 in the high regime. The second is above 1, the Taylor principle, so the rule is stabilising only after a boom. The first should be read with care, because it divides a small coefficient by a number near zero. The high regime rests on few observations, so its standard errors are wide.

11.6 Fit, residuals and the transition over time

```
oes.show(r, "Fit against the linear model")
oes.show(r, "Residual diagnostics")
oes.plot(r, "Transition function against", ncols=1, size=(6.5, 3))
oes.plot(r, "Transition function over time", ncols=1, size=(6.5, 3))
```

```
Fit against the linear model
                                   SSR  Residual std. dev.      AIC     BIC
Model
Linear                         144.2935              0.7869 -109.602  -95.73
Smooth transition (logistic)  107.6517              0.6886 -167.031 -132.35
Note: AIC and BIC count gamma and c as parameters. Lower is better.

Residual diagnostics
                                       Statistic  p-value                     Detail
Test
Ljung-Box serial correlation (4 lags)    15.7161   0.0034  H0: no serial correlation
```

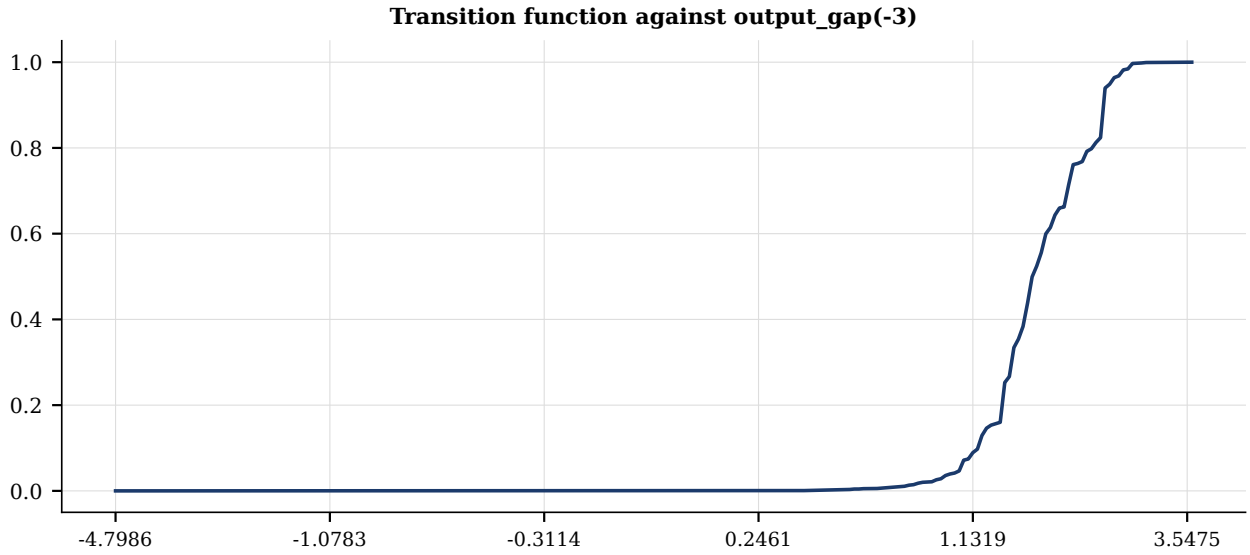


Figure 11.1. Fit, residuals and the transition over time: Transition function against output_gap(-3).

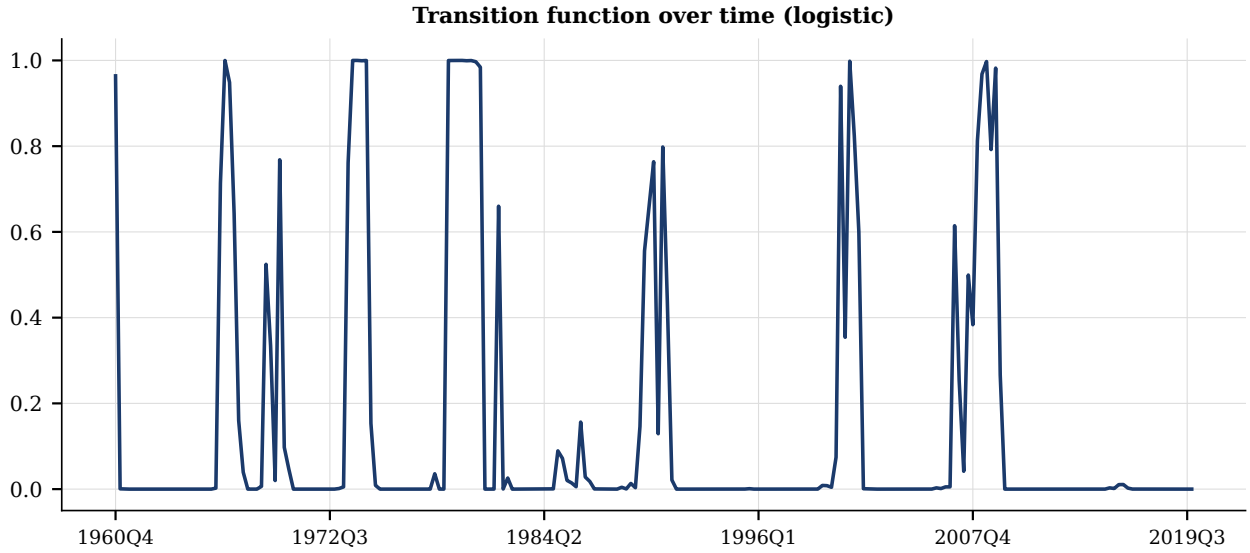


Figure 11.2. Fit, residuals and the transition over time: Transition function over time (logistic).

How to read it

The smooth-transition model lowers the sum of squared residuals from 144.3 to 107.7 and the residual standard deviation from 0.787 to 0.689. AIC falls from –109.6 to –167.0 and BIC from –95.7 to –132.4, so both criteria prefer it even after counting γ and c. The Ljung–Box test rejects no serial correlation (statistic 15.7, p = 0.003), so the dynamics are not fully captured with one lag; the AR(2) variant below gives the same γ and location, so the main result does not depend on it, but the standard errors should be treated with care. The first chart shows the transition function: it is zero for output gaps below about 0.9, rises between about 1.1 and 2.5 and is one above that. The second chart shows it over time. It is at or near zero most of the time and rises only in booms: the mid-1960s, the early 1970s, around 1979–1980, around 1990, in 2000 and in 2006–2007. It is zero throughout the 2010s.

11.7 Does the answer depend on the form, the lags, the delay and the range?

```
out = {}
```

```
variants = {"baseline (LSTAR, AR(1), delay 3)": {}, "ESTAR": dict(form="exponential"), "AR(2)": dict(ar_lags=2)
  ↪ ,
            "delay 2": dict(delay=2), "delay 4": dict(delay=4), "range for c: 5th to 95th": dict(trim="0.05"),
            "range for c: 15th to 85th": dict(trim="0.15"), "range for c: 20th to 80th": dict(trim="0.20")}
for name, v in variants.items():
    rr = oes.run("str", data, dict(spec, **v))
    s = {x["label"]: x["value"] for x in rr["summary"]}
    out[name] = (s["Transition function"], s["gamma"], s["c"], s["c at the edge of its range"], s["R-squared"],
  ↪  s["AIC gain over linear"])
print(pd.DataFrame(out, index=["form", "gamma", "c", "c at edge", "R-squared", "AIC gain"]).T.to_string())
```

```
                                      form    gamma       c c at edge R-squared AIC gain
baseline (LSTAR, AR(1), delay 3)     logistic   10.022  1.4656        no    0.9661    57.43
ESTAR                            exponential   10.424  1.6457       yes    0.9623    32.37
AR(2)                                logistic    10.48     1.5        no    0.9665    57.07
delay 2                              logistic    3.025  1.4302        no    0.9597    16.95
delay 4                              logistic  108.469  1.6477       yes    0.9598    17.57
range for c: 5th to 95th             logistic   10.022  1.4656        no    0.9661    57.43
range for c: 15th to 85th            logistic   10.225  1.4511       yes    0.9661    57.41
range for c: 20th to 80th            logistic   15.546   1.143       yes    0.9646    47.37
```

How to read it

The baseline results hold across the variants that are well behaved. With two lags of the funds rate the speed is 10.5, the location is 1.50 and the AIC gain is 57.1. Widening the range for c to the 5th to 95th percentiles changes nothing, because the baseline location (1.47) is inside the range. The variants that change the answer are flagged in the table. A narrower range for c (15th to 85th and 20th to 80th) puts the location on the edge of the range (1.45 and 1.14), which shows the range is cutting off the best location. The exponential form, imposed by hand, also ends on the edge of its range (1.65) and its AIC gain is 32.4, well below the logistic 57.4. A delay of four puts c on the edge with γ above 100, and a delay of two gives a slower transition ($\gamma = 3.0$) but an AIC gain of only 17.0. The delay of three quarters, the logistic form and a location of about 1.5 points are what the data support.

11.8 Smooth or abrupt? Compare with the threshold regression

```
th = oes.run("tar", data, dict(y="fedfunds", x=["inflation", "output_gap"], threshold="output_gap", time="date
  ↪ ",
                               ar_lags=1, delay=3, boot=499, max_thresholds="1"))
oes.show(th, "Tests for a threshold")
oes.show(th, "Estimated thresholds")
oes.show(th, "Fit against the linear model")
```

```
Tests for a threshold
                                      Statistic  Bootstrap p-value  5% critical value              Detail
Test
Linearity against one threshold (sup-F)     72.941              0.018             59.042  499 bootstrap draws
Note: The threshold is not identified under the null of linearity, so the F statistic does not follow an F
↪ distribution. The p-value comes from Hansen's (1996) fixed-regressor bootstrap.

Estimated thresholds
           Estimate 95% confidence interval  Observations at or below  Observations above
Threshold
1            1.3793        1.2202 to 1.4299                       199                  38
Note: The interval collects every candidate threshold whose likelihood-ratio statistic is below 7.35 (Hansen 2000).
↪ It is often wide and can be asymmetric.

Fit against the linear model
                           SSR  Residual std. dev.      AIC
Model
Linear                144.2935             0.7869 -109.602
Threshold (2 regimes)  109.4359             0.6913 -165.135
Note: AIC counts the thresholds as parameters. A lower value favours the threshold model.
```

How to read it

The threshold regression with the same specification and the same delay rejects linearity (sup-F = 72.9, bootstrap p = 0.018 with 499 draws, 5% critical value 59.0). It places the threshold at 1.38 (95% interval 1.22 to 1.43), with 38 of 237 observations above it. Its fit is almost the same as the smooth model: a sum of squared residuals of 109.4 against 107.7, and an AIC of –165.1 against –167.0. The data therefore cannot separate an abrupt switch from a quick smooth one, which agrees with the wide standard error of γ. Both place the switch at an output gap of about 1.4 to 1.5 points. Choose the threshold model if you want a simple statement (a rule that changes at a level), and the smooth-transition model if you want to show how the response changes through the range.

11.9 Write the results

How to write it: results

We find that the response of U.S. monetary policy to inflation depends on the state of the cycle. The federal funds rate is modelled with a logistic smooth-transition regression on its own lag, inflation and the output gap, with all coefficients moving with the output gap lagged three quarters. Teräsvirta's linearity test rejects strongly ($F = 4.73$, $p < 0.0001$) and the sequence of tests selects the logistic form. The transition is centred at an output gap of 1.47 points (standard error 0.16) with a speed of 10.0 (standard error 7.5). When the lagged gap is low, the rate adjusts slowly (coefficient on its lag 0.96) and does not respond to inflation (0.03). When the gap is high, adjustment is faster (0.38) and the coefficient on inflation is 0.93 (standard error 0.38), a long-run response of 1.5, above the Taylor-principle threshold of 1. The model lowers AIC by 57 points and BIC by 37 points relative to the linear rule. Residual serial correlation remains (Ljung–Box $p = 0.003$), and the high regime contains few observations, so these estimates are imprecise. The results are the same with two lags and for a wider range of the location, and a threshold regression with the same specification gives a similar fit and a threshold of 1.38.

Report	What	Where
Model	Regressors, transition variable, delay, function, range for c	Section 11.1
Data	Scale and persistence	Section 11.2
Search	Linearity tests for each candidate variable and delay	Section 11.3
Estimates	Linearity tests, γ and c with standard errors	Section 11.4
Regimes	Coefficients at G = 0 and G = 1, long-run response	Section 11.5
Fit	AIC, BIC, residual test, the transition function	Section 11.6
Robustness	Form, lags, delay, range	Section 11.7
Comparison	Threshold regression	Section 11.8

References. Teräsvirta, T. (1994), "Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models," *Journal of the American Statistical Association* 89. Granger, C. W. J. and Teräsvirta, T. (1993), *Modelling Nonlinear Economic Relationships*, Oxford University Press. van Dijk, D., Teräsvirta, T. and Franses, P. H. (2002), "Smooth Transition Autoregressive Models: A Survey of Recent Developments," *Econometric Reviews* 21. Hansen, B. E. (2000), "Sample Splitting and Threshold Estimation," *Econometrica* 68.

CHAPTER 12

Structural Breaks at Unknown Dates (Bai–Perron)

Applies when. A mean, a trend or a regression coefficient may have shifted, and you want to find how many times and when

Overview. Use this chapter when a series or a relationship may have changed at dates you do not know: inflation after the 1970s, the coefficient of a policy rule after a regime change, a growth slowdown. The Bai–Perron (1998, 2003) procedure finds the best partition into regimes, tests how many breaks there are, gives confidence intervals for the dates and estimates each regime separately. The chapter covers the setup, the tests, the choice of the number of breaks, the break-date intervals, robustness to serial correlation and the text for the paper.

Data. U.S. quarterly data, 1960–2019: GDP-deflator inflation and real GDP growth

Software. Python · OES engine · statsmodels

12.1 The question and the model

How many times did the parameters shift, and when? With m breaks at dates $T_1 < \cdots < T_m$ the model is

$$y_t = x_t'\beta_j + u_t, \qquad t = T_{j-1}+1, \dots, T_j, \qquad j = 1, \dots, m+1,$$

where x_t always contains a constant, so the simplest case is a model of shifting means. For each m the break dates are the least-squares minimisers over all partitions in which every regime has at least a fraction ε of the sample (the trimming). A dynamic-programming algorithm finds them exactly. The number of breaks is chosen by the $\sup F(\ell)$ tests of no break against ℓ breaks, the UDmax test against an unknown number, the sequential tests of ℓ against $\ell+1$ breaks, or an information criterion.

Three checks before you trust break dates

1. **Are the errors close to uncorrelated?** The tests assume serially uncorrelated errors. For a persistent series, shifts in the mean can be mistaken for persistence, and the reverse. Re-estimate with a lagged dependent variable among the regressors.
2. **Does the number of breaks depend on the rule?** BIC, LWZ and the sequential test can disagree. Report all of them, and show that the dates you discuss are stable.
3. **Are the break dates precise?** A small shift gives a wide confidence interval for the date. Say so, and do not present a date as exact when its interval spans years.

How to write it: the model

We test for structural change in the mean of U.S. inflation with the procedure of Bai and Perron (1998, 2003). The model allows the mean to shift at up to five unknown dates, with each regime containing at least 15% of the 240 quarterly observations (36 quarters). Break dates are the global least-squares minimisers, and the number of breaks is chosen by the BIC and checked against the sequential sup$F(\ell+1\,|\,\ell)$ test at the 5% level. Critical values are asymptotic and were simulated for this engine. Confidence intervals for the dates follow Bai (1997).

12.2 Load the data and look at the series

```
import numpy as np
```

```
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.tsa.stattools import acf

data = open("yield.csv").read()
df = pd.read_csv("yield.csv")
print(df.date.iloc[0], "to", df.date.iloc[-1], "|", len(df), "quarters")
print(df[["inflation", "gdp_growth"]].describe().round(2).loc[["mean", "std", "min", "max"]])
for c in ["inflation", "gdp_growth"]:
    print(f"{c:11s} ADF p = {adfuller(df[c], autolag='AIC')[1]:.3f}   KPSS p = {kpss(df[c], nlags='auto')
  ↪ [1]:.3f}   first autocorrelation = {acf(df[c], nlags=1)[1]:.2f}")
```

```
1960Q1 to 2019Q4 | 240 quarters
      inflation  gdp_growth
mean       3.19        3.01
std        2.31        3.25
min       -0.53       -8.85
max       12.14       15.17
inflation   ADF p = 0.190   KPSS p = 0.010   first autocorrelation = 0.90
gdp_growth  ADF p = 0.000   KPSS p = 0.059   first autocorrelation = 0.30
```

How to read it

The sample runs from 1960Q1 to 2019Q4 (240 quarters). Inflation averages 3.19% with a standard deviation of 2.31; GDP growth averages 3.01% with a standard deviation of 3.25. Inflation is highly persistent (first autocorrelation 0.90, ADF p = 0.19) and the KPSS test rejects stationarity (p = 0.01), the pattern a series with shifting means produces. GDP growth is stationary with a first autocorrelation of 0.30. Persistence matters because the break tests assume uncorrelated errors: for inflation, a shift in the mean and a high autoregressive coefficient are hard to tell apart, which the AR(1) check below makes visible.

12.3 Test for breaks and choose their number

```
spec = dict(y="inflation", time="date", trim="0.15", max_breaks=5, select="bic")
r = oes.run("bp", data, spec)
oes.summary(r)
oes.show(r, "Tests of no break")
oes.show(r, "Sequential tests")
oes.show(r, "Information criteria")
```

```
Observations: 240
Regressors changing (incl. constant): 1
Trimming: 0.15
Minimum regime length: 36
Maximum breaks: 5
Breaks selected: 3
BIC chooses: 3
Sequential chooses: 3
Tests of no break against l breaks
                                          Statistic  5% critical value Decision
Test
supF(1) : 0 versus 1                        106.887             8.646   Reject
supF(2) : 0 versus 2                        226.791             6.981   Reject
supF(3) : 0 versus 3                        179.362             5.870   Reject
supF(4) : 0 versus 4                        139.463             4.849   Reject
supF(5) : 0 versus 5                        110.171             3.863   Reject
UDmax: 0 versus an unknown number (1 to 5)  226.791             8.941   Reject
Note: Critical values are simulated for this engine (asymptotic, all coefficients change, trimming 0.15).

Sequential tests: l breaks against l + 1
                         Statistic  5% critical value       Decision
Test
supF(1 | 0) : 0 versus 1    106.887             8.646         Reject
supF(2 | 1) : 1 versus 2    174.818            10.344         Reject
supF(3 | 2) : 2 versus 3     29.658            11.284         Reject
supF(4 | 3) : 3 versus 4      6.720            11.889  Do not reject
```

```
supF(5 | 4) : 4 versus 5      0.000            12.781  Do not reject
Note: Start at l = 0 and stop at the first non-rejection; that l is the sequential estimate.

Information criteria by number of breaks
              SSR     BIC     LWZ
Breaks
0       1273.8397  1.6920  1.7177
1        879.0530  1.3667  1.4439
2        437.1670  0.7139  0.8425
3        388.3622  0.6412  0.8214
4        377.5647  0.6586  0.8906
5        379.7878  0.7102  0.9939
Note: BIC chooses 3, LWZ chooses 3, the sequential test chooses 3. Shown below: 3 break(s), chosen by BIC.
```

How to read it

Every test of no break against ℓ breaks rejects at 5%: the supF(1) statistic is 106.9 against a critical value of 8.6, and the UDmax statistic is 226.8 against 8.9. The sequential tests reject zero, one and two breaks against one more (statistics 106.9, 174.8 and 29.7 against critical values of 8.6, 10.3 and 11.3) and stop at the fourth: the statistic for three against four breaks is 6.7, below the 5% critical value of 11.9. BIC and LWZ also choose three breaks, and the sum of squared residuals falls from 1,274 with no break to 437 with two and 388 with three, then barely at all. All four rules therefore agree on three breaks.

12.4 Break dates, regimes and the fitted path

```
oes.show(r, "Estimated break dates")
oes.show(r, "Regimes of the model")
oes.plot(r, "inflation and the fitted", ncols=1, size=(6.5, 3))
```

```
Estimated break dates
      First observation of the new regime 95% confidence interval  90% half-width (obs.)  95% half-width (obs.)
Break
1                                  1973Q2        1971Q4 to 1974Q4                   3.5                    5.0
2                                  1982Q4        1981Q4 to 1983Q4                   2.2                    3.2
3                                  1991Q4        1988Q4 to 1994Q4                   7.7                   11.1
Note: Intervals follow Bai (1997) and Bai and Perron (2003): the limiting distribution of the break date under a
↪ shrinking shift, with a Newey-West long-run variance so that serially correlated errors widen the interval. They
↪ are wide when the shift is small relative to the noise.

Regimes of the model with 3 break(s)
          From      To  Observations            const
Regime
1       1960Q1  1973Q1            53  2.9532 (0.4227)
2       1973Q2  1982Q3            38   7.442 (0.4762)
3       1982Q4  1991Q3            36   3.2118 (0.206)
4       1991Q4  2019Q4           113  1.8748 (0.1256)
Note: Coefficients with Newey-West standard errors in parentheses, estimated separately in each regime.
```

How to read it

The three breaks fall in 1973Q2, 1982Q4 and 1991Q4. The mean of inflation is 2.95% before 1973 (standard error 0.42), 7.44% from 1973Q2 to 1982Q3 (0.48), 3.21% from 1982Q4 to 1991Q3 (0.21) and 1.87% from 1991Q4 onwards (0.13). These are the great inflation, the Volcker disinflation and the lower-inflation period that followed. The 95% intervals for the first two dates are tight (1971Q4 to 1974Q4 and 1981Q4 to 1983Q4), because the shifts are large; the third date is much less precise (1988Q4 to 1994Q4), because the drop from 3.21% to 1.87% is small relative to the noise.

12.5 Does the answer depend on the selection rule and on the trimming?

```
out = {}
for name, v in {"BIC": dict(select="bic"), "LWZ": dict(select="lwz"), "sequential": dict(select="sequential"),
  ↪  "UDmax then sequential": dict(select="udmax"),
```

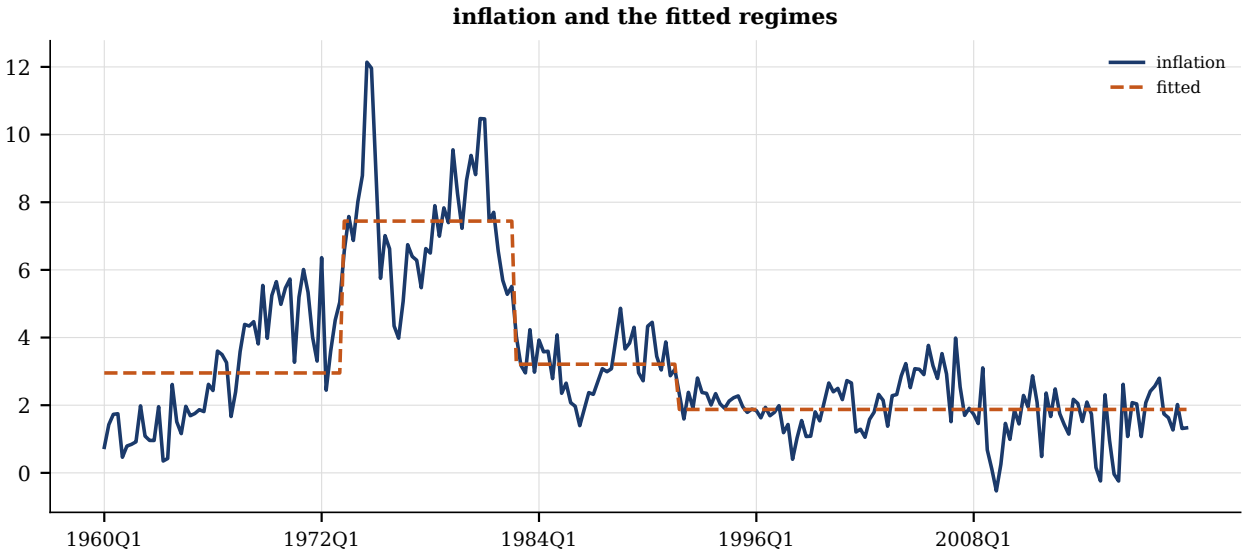


Figure 12.1. Break dates, regimes and the fitted path: inflation and the fitted regimes.

```
            "trim 10%": dict(trim="0.10", max_breaks=8), "trim 20%": dict(trim="0.20", max_breaks=3), "
  ↪ trim 25%": dict(trim="0.25", max_breaks=2)}.items():
    rr = oes.run("bp", data, dict(spec, **v))
    s = {x["label"]: x["value"] for x in rr["summary"]}
    dates = [x[1] for x in [t for t in rr["tables"] if t["title"] == "Estimated break dates"][0]["rows"]] if s
  ↪ ["Breaks selected"] else []
    out[name] = (s["Breaks selected"], ", ".join(dates) or "none")
print(pd.DataFrame(out, index=["breaks", "first quarter of the new regime"]).T)
```

```
                     breaks first quarter of the new regime
BIC                       3          1973Q2, 1982Q4, 1991Q4
LWZ                       3          1973Q2, 1982Q4, 1991Q4
sequential                3          1973Q2, 1982Q4, 1991Q4
UDmax then sequential     3          1973Q2, 1982Q4, 1991Q4
trim 10%                  4  1967Q3, 1973Q3, 1982Q2, 1991Q4
trim 20%                  3          1972Q1, 1984Q1, 1996Q1
trim 25%                  2                  1975Q1, 1990Q3
```

How to read it

The three breaks (1973Q2, 1982Q4, 1991Q4) are the same under BIC, LWZ, the sequential test and UDmax followed by the sequential test. The trimming fraction matters more. With 10% trimming the procedure adds an earlier break (1967Q3) and finds four; with 20% trimming it finds three breaks at 1972Q1, 1984Q1 and 1996Q1, because a break must leave at least 48 quarters in each regime, which rules out the 1973 and 1982 dates; with 25% it finds two. A larger trimming fraction restricts where the breaks can fall, so it moves the dates, and 15% is the usual choice because it balances precision and flexibility.

12.6 Serial correlation: add a lag of the dependent variable

```
d2 = df.copy()
d2["inflation_l1"] = d2.inflation.shift(1)
d2 = d2.dropna(subset=["inflation_l1"]).to_csv(index=False)
ar = oes.run("bp", d2, dict(spec, x=["inflation_l1"]))
oes.show(ar, "Sequential tests")
oes.show(ar, "Estimated break dates")
oes.show(ar, "Regimes of the model")
```

```
Sequential tests: l breaks against l + 1
                        Statistic  5% critical value       Decision
Test
supF(1 | 0) : 0 versus 1     21.587            11.709         Reject
supF(2 | 1) : 1 versus 2     12.530            13.414  Do not reject
```

```
supF(3 | 2) : 2 versus 3     14.484             14.725  Do not reject
supF(4 | 3) : 3 versus 4      3.097             15.289  Do not reject
supF(5 | 4) : 4 versus 5      0.000             15.733  Do not reject
Note: Start at l = 0 and stop at the first non-rejection; that l is the sequential estimate.

Estimated break dates
      First observation of the new regime 95% confidence interval  90% half-width (obs.)  95% half-width (obs.)
Break
1                                2006Q4        2005Q2 to 2008Q2                   4.0                   5.8
Note: Intervals follow Bai (1997) and Bai and Perron (2003): the limiting distribution of the break date under a
↪ shrinking shift, with a Newey-West long-run variance so that serially correlated errors widen the interval. They
↪ are wide when the shift is small relative to the noise.

Regimes of the model with 1 break(s)
          From      To  Observations            const     inflation_l1
Regime
1       1960Q2  2006Q3           186  0.3505 (0.1171)  0.9071 (0.0402)
2       2006Q4  2019Q4            53  1.1628 (0.2654)  0.2716 (0.1172)
Note: Coefficients with Newey-West standard errors in parentheses, estimated separately in each regime.
```

How to read it

Adding a lagged dependent variable changes the story: one break is found, in 2006Q4 (95% interval 2005Q2 to 2008Q2), and the sequential tests stop after the first. Before 2006Q4 inflation is very persistent (autoregressive coefficient 0.91, standard error 0.04); after 2006Q4 persistence drops to 0.27 and the intercept rises. The earlier breaks in the mean are absorbed because a high autoregressive coefficient already explains the long swings. The two models answer different questions: the mean model asks when the average level shifted, the AR(1) model asks when the dynamics changed. Report both, and be explicit about which one your argument needs.

12.7 A second series: real GDP growth

```
g = oes.run("bp", data, dict(spec, y="gdp_growth"))
oes.show(g, "Sequential tests")
oes.show(g, "Estimated break dates")
oes.show(g, "Regimes of the model")
```

```
Sequential tests: l breaks against l + 1
                         Statistic  5% critical value       Decision
Test
supF(1 | 0) : 0 versus 1     11.417              8.646         Reject
supF(2 | 1) : 1 versus 2      6.222             10.344  Do not reject
supF(3 | 2) : 2 versus 3      4.375             11.284  Do not reject
supF(4 | 3) : 3 versus 4      1.596             11.889  Do not reject
supF(5 | 4) : 4 versus 5      0.000             12.781  Do not reject
Note: Start at l = 0 and stop at the first non-rejection; that l is the sequential estimate.

Estimated break dates
      First observation of the new regime 95% confidence interval  90% half-width (obs.)  95% half-width (obs.)
Break
1                                1969Q2        1960Q1 to 1982Q4                  37.2                  53.9
Note: Intervals follow Bai (1997) and Bai and Perron (2003): the limiting distribution of the break date under a
↪ shrinking shift, with a Newey-West long-run variance so that serially correlated errors widen the interval. They
↪ are wide when the shift is small relative to the noise.

Regimes of the model with 1 break(s)
          From      To  Observations            const
Regime
1       1960Q1  1969Q1            37   4.6398 (0.623)
2       1969Q2  2019Q4           203  2.7177 (0.3029)
Note: Coefficients with Newey-West standard errors in parentheses, estimated separately in each regime.
```

How to read it

For real GDP growth the tests find one break, in 1969Q2: the mean falls from 4.64% (standard error 0.62) to 2.72% (0.30). The sequential test does not reject one break against two (6.2 against a critical value of 10.3). The

95% interval for the date is very wide (1960Q1 to 1982Q4) because the shift is small relative to the volatility of growth, so the break is detected but not dated. Do not present 1969Q2 as an exact date.

12.8 Write the results

How to write it: results

The mean of U.S. inflation shifted three times between 1960 and 2019. The supF, UDmax and sequential tests all reject the null of no break at 5% (supF(1) = 106.9, critical value 8.6), and the BIC, LWZ and sequential procedures all select three breaks, dated 1973Q2 (95% interval 1971Q4 to 1974Q4), 1982Q4 (1981Q4 to 1983Q4) and 1991Q4 (1988Q4 to 1994Q4). The mean inflation rate is 2.95% before 1973, 7.44% from 1973Q2 to 1982Q3, 3.21% from 1982Q4 to 1991Q3 and 1.87% from 1991Q4. The number of breaks is robust to the selection rule but the dates depend on the trimming fraction: with 20% trimming the breaks move to 1972Q1, 1984Q1 and 1996Q1. When a lagged dependent variable is added the tests find a single break in 2006Q4 in which persistence falls from 0.91 to 0.27, which shows that the mean shifts and the persistence of inflation cannot be separated by these tests alone. The critical values are asymptotic and simulated for the pure structural change model.

Report	What	Where
Model	Regressors, trimming, maximum breaks, selection rule	Section 12.1
Data	Persistence and stationarity	Section 12.2
Tests	supF, UDmax, sequential tests, information criteria	Section 12.3
Dates	Break dates with confidence intervals, regime means	Section 12.4
Robustness	Selection rule and trimming	Section 12.5
Serial correlation	Re-estimate with a lagged dependent variable	Section 12.6
Second series	The same procedure on GDP growth	Section 12.7

References. Bai, J. and Perron, P. (1998), "Estimating and Testing Linear Models with Multiple Structural Changes," *Econometrica* 66. Bai, J. and Perron, P. (2003), "Computation and Analysis of Multiple Structural Change Models," *Journal of Applied Econometrics* 18. Bai, J. (1997), "Estimating Multiple Breaks One at a Time," *Econometric Theory* 13. Andrews, D. (1993), "Tests for Parameter Instability and Structural Change with Unknown Change Point," *Econometrica* 61. Liu, J., Wu, S. and Zidek, J. (1997), "On Segmented Multivariate Regression," *Statistica Sinica* 7. Stock, J. and Watson, M. (2003), "Has the Business Cycle Changed and Why?" *NBER Macroeconomics Annual* 17.

CHAPTER 13

2SLS and GMM

Applies when. Endogenous regressors that need instruments

Overview. Use this chapter when a regressor is correlated with the error term, so that OLS is biased. It covers choosing instruments, two-stage least squares, instrument strength, overidentification and endogeneity tests, efficient GMM and robustness for a time-series model, then difference GMM for a dynamic panel with its specification tests, with sample text for the paper.

Data. U.S. policy rule, quarterly, 1979–2007; UK firm employment panel, 1976–1984

Software. Python · statsmodels

13.1 The question and why OLS fails

The question is: **how strongly does the Federal Reserve raise its interest rate when inflation rises?** The standard description of policy is a Taylor rule with interest-rate smoothing:

$$i_t = c + \beta\,\pi_t + \gamma\,x_t + \rho\,i_{t-1} + \varepsilon_t$$

where i_t is the federal funds rate, π_t is inflation over the past four quarters and x_t is the output gap. The long-run response to inflation is $\beta/(1-\rho)$. The **Taylor principle** says it must exceed 1, so that the real interest rate rises when inflation rises.

OLS cannot estimate this equation consistently, for two reasons.

- **Simultaneity.** The interest rate affects inflation and output within the same quarter, so the regressors respond to the dependent variable.
- **Policy shocks.** The error term ε_t is the unexpected part of policy. Because a policy surprise moves inflation and output, the regressors are correlated with the error.

A regressor that is correlated with the error term is **endogenous**. OLS is then biased, and more data does not fix it. The solution is instrumental variables.

What an instrument must do

An instrument z is a variable that satisfies two conditions.

1. **Relevance:** z is correlated with the endogenous regressor. This can be tested (Section 13.5).
2. **Exogeneity:** z is uncorrelated with the error term. This cannot be tested directly; it must be argued. With more instruments than endogenous regressors, it can be partly tested (Section 13.6).

Here the instruments are **lagged values** of inflation, the output gap and the interest rate. They are relevant because these series are persistent. They are plausibly exogenous because a policy surprise in quarter t cannot have influenced what happened in earlier quarters. This is the approach of Clarida, Galí and Gertler (2000).

How to write it: identification

Estimating the policy rule by OLS would be inconsistent because inflation and the output gap respond contemporaneously to the interest rate and are therefore correlated with the policy shock ε_t. Following Clarida, Galí and Gertler (2000), we treat current inflation and the output gap as endogenous and instrument them with their own first four lags and the second lag of the federal funds rate. These variables are predetermined with respect to the period-t policy shock and, given the persistence of inflation and output, are strongly correlated with the endogenous regressors.

13.2 Load the data and build the instruments

This chapter uses the `linearmodels` package for instrumental-variable estimation. Install it once with `pip install linearmodels`.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels.iv import IV2SLS, IVGMM

data = pd.read_csv("taylor.csv").set_index("date")
for lag in range(1, 5):
    for col in ("fedfunds", "inflation", "output_gap"):
        data[f"{col}_l{lag}"] = data[col].shift(lag)
data["const"] = 1.0

df = data.loc["1979Q3":"2007Q4"].copy()        # Volcker and Greenspan years, up to the financial crisis
print(len(df), "quarterly observations")
print(df[["fedfunds", "inflation", "output_gap"]].describe().T[["mean", "std", "min", "max"]].round(2))

endog = ["inflation", "output_gap"]                                    # endogenous regressors
exog = ["const", "fedfunds_l1"]                                        # included exogenous regressors
instruments = [f"{c}_l{k}" for c in endog for k in (1, 2, 3, 4)] + ["fedfunds_l2"]
print(f"\n{len(endog)} endogenous regressors, {len(instruments)} instruments  ->  {len(instruments) - len(
  ↪ endog)} overidentifying restrictions")
```

```
114 quarterly observations
            mean   std   min    max
fedfunds    6.48  3.68  1.00  17.78
inflation   3.20  1.99  1.01   9.78
output_gap  0.03  1.38 -4.80   2.63

2 endogenous regressors, 9 instruments  ->  7 overidentifying restrictions
```

> **How to read it**
>
> There are 114 quarters. The model has 2 endogenous regressors and 9 instruments. It is **overidentified**: there are 7 more instruments than needed. Overidentification is useful, because the extra instruments make it possible to test instrument validity in Section 13.6. With exactly as many instruments as endogenous regressors the model is *just identified* and that test is not available.
>
> The lagged interest rate is treated as exogenous because it was set before the current shock. That requires the errors to be serially uncorrelated, which is one reason the second lag is also added as an instrument.

13.3 The OLS benchmark

Always estimate OLS first. It is the benchmark that shows how much the instruments change the answer.

```
ols = sm.OLS(df["fedfunds"], df[exog + endog]).fit(cov_type="HAC", cov_kwds={"maxlags": 4})
print(ols.summary().tables[1])
rho = ols.params["fedfunds_l1"]
print(f"\nLong-run response to inflation: {ols.params['inflation'] / (1 - rho):.2f}    to the output gap: {ols.
  ↪ params['output_gap'] / (1 - rho):.2f}")
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1112      0.164      0.680      0.497      -0.209       0.432
fedfunds_l1    0.8161      0.063     12.965      0.000       0.693       0.939
inflation      0.3222      0.118      2.728      0.006       0.091       0.554
output_gap     0.2565      0.056      4.549      0.000       0.146       0.367
==============================================================================

Long-run response to inflation: 1.75    to the output gap: 1.39
```

13.4 Two-stage least squares

2SLS works in two steps. **First stage:** regress each endogenous regressor on all the instruments and exogenous variables, and keep the fitted values. **Second stage:** regress the dependent variable on those fitted values. The fitted values contain only the variation that the instruments explain, which is unrelated to the error.

Never run the two stages by hand: the standard errors from a manual second stage are wrong. Use a routine that does both.

```
iv = IV2SLS(df["fedfunds"], df[exog], df[endog], df[instruments]).fit(cov_type="kernel", kernel="bartlett",
  ↪ bandwidth=4)
print(iv.summary.tables[1])
rho = iv.params["fedfunds_l1"]
print(f"\nLong-run response to inflation: {iv.params['inflation'] / (1 - rho):.2f}    to the output gap: {iv.
  ↪ params['output_gap'] / (1 - rho):.2f}")
```

```
                            Parameter Estimates
==============================================================================
             Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const           0.1255     0.1685     0.7449     0.4563     -0.2047      0.4557
fedfunds_l1     0.8343     0.0576     14.488     0.0000      0.7214      0.9471
inflation       0.2815     0.1086     2.5926     0.0095      0.0687      0.4942
output_gap      0.1627     0.0579     2.8110     0.0049      0.0492      0.2761
==============================================================================

Long-run response to inflation: 1.70    to the output gap: 0.98
```

How to read it

Compare with OLS. The coefficient on inflation is similar (0.28 against 0.32), but the coefficient on the output gap falls from 0.26 to 0.16. OLS overstated the response to the output gap, as expected if a strong economy and a tight policy stance move together within the quarter for reasons the rule does not capture.

The smoothing coefficient of 0.83 means the Fed moves the rate gradually: only about 17% of the desired adjustment happens in a quarter. This is why the long-run responses are much larger than the impact coefficients.

`cov_type="kernel"` requests standard errors that are robust to heteroskedasticity and serial correlation, the IV equivalent of Newey–West.

13.5 Test 1: are the instruments strong?

Weak instruments, only loosely related to the endogenous regressors, are worse than no instruments: 2SLS becomes biased towards OLS and its standard errors are unreliable.

```
first = iv.first_stage.diagnostics
print(first[["rsquared", "partial.rsquared", "f.stat", "f.pval"]].round(3))

for var in endog:                                   # the classical first-stage F statistic, from the first-
  ↪ stage regression itself
    full_ = sm.OLS(df[var], df[exog + instruments]).fit()
    restricted = sm.OLS(df[var], df[exog]).fit()
    f_stat = ((restricted.ssr - full_.ssr) / len(instruments)) / (full_.ssr / full_.df_resid)
    print(f"First-stage F for {var:11s} = {f_stat:7.1f}")
```

```
            rsquared  partial.rsquared    f.stat  f.pval
inflation      0.987            0.963  4338.302     0.0
output_gap     0.834            0.834   593.425     0.0
First-stage F for inflation   =   301.3
First-stage F for output_gap  =    57.6
```

How to read it

- **Partial R-squared** is the share of each endogenous regressor explained by the instruments after removing the exogenous variables. Here it is 0.96 for inflation and 0.83 for the output gap.
- **First-stage F** tests whether the instruments are jointly zero in the first stage. The rule of thumb of Staiger and Stock (1997) is that an F below 10 signals weak instruments. Both statistics are far above 10.

These instruments are very strong, which is no surprise: four-quarter inflation and the output gap are highly persistent, so their own lags predict them well. In cross-section work you will rarely see values like these; an F of 15 to 30 is typical of a good instrument there. With several endogenous regressors, also report a joint statistic such as Cragg–Donald or Kleibergen–Paap.

13.6 Test 2: are the instruments valid? The overidentification test

```
print(iv.sargan)
```

```
Sargan's test of overidentification
H0: The model is not overidentified.
Statistic: 7.8373
P-value: 0.3472
Distributed: chi2(7)
```

How to read it

The Sargan test (called the Hansen J test in GMM) has the null hypothesis that **all instruments are valid**, meaning uncorrelated with the error. A **large** p-value is what you want. Here p = 0.35, so the null is not rejected: the data give no evidence against the instruments.

Three cautions. A rejection tells you that at least one instrument is invalid but not which one. A non-rejection is not proof of validity: the test assumes that at least enough instruments are valid to identify the model, and it has low power when all instruments share the same flaw. And with a very large number of instruments the test almost never rejects. Keep the instrument set small and justify each instrument in words.

13.7 Test 3: is the regressor really endogenous?

```
print(iv.wu_hausman())
print()
print(iv.durbin())
```

```
Wu-Hausman test of exogeneity
H0: All endogenous variables are exogenous
Statistic: 9.4228
P-value: 0.0002
Distributed: F(2,108)

Durbin test of exogeneity
H0: All endogenous variables are exogenous
Statistic: 16.9371
P-value: 0.0002
Distributed: chi2(2)
```

How to read it

The Wu–Hausman and Durbin tests have the null that the regressors are **exogenous**, in which case OLS and 2SLS estimate the same thing and OLS, being more precise, should be preferred. Both reject strongly (p below 0.001). Inflation and the output gap are endogenous, and the IV estimates are needed.

If this test does not reject, report OLS as the main result and 2SLS as a robustness check. Instrumenting a regressor that is not endogenous only makes the estimates noisier.

13.8 Efficient GMM

The generalised method of moments (GMM) uses the same instruments but weights them optimally. With heteroskedastic or serially correlated errors, 2SLS is no longer the most efficient estimator; two-step GMM is.

```
gmm = IVGMM(df["fedfunds"], df[exog], df[endog], df[instruments], weight_type="kernel", kernel="bartlett",
  ↪ bandwidth=4).fit(iter_limit=2)
print(gmm.summary.tables[1])
print(gmm.j_stat)
```

```
b, V = gmm.params, gmm.cov
rho = b["fedfunds_l1"]
print(f"\n{'':20s}{'long-run response':>18s}{'std. error':>12s}")
for var in endog:
    lr = b[var] / (1 - rho)
    grad = pd.Series(0.0, index=b.index); grad[var] = 1 / (1 - rho); grad["fedfunds_l1"] = b[var] / (1 - rho)
  ↪ ** 2
    se = float(np.sqrt(grad @ V @ grad))
    print(f"{var:20s}{lr:18.2f}{se:12.2f}")
    if var == "inflation":
        from scipy import stats
        lr_pi, se_pi = lr, se
print(f"\nTaylor principle, H0: long-run inflation response <= 1:  t = {(lr_pi - 1) / se_pi:.2f}   one-sided p-
  ↪ value = {1 - stats.norm.cdf((lr_pi - 1) / se_pi):.3f}")
```

```
                           Parameter Estimates
==============================================================================
             Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const           0.0378     0.1685     0.2245     0.8224     -0.2924      0.3680
fedfunds_l1     0.8732     0.0438     19.948     0.0000      0.7874      0.9590
inflation       0.2265     0.0790     2.8674     0.0041      0.0717      0.3813
output_gap      0.1905     0.0905     2.1047     0.0353      0.0131      0.3679
==============================================================================
H0: Expected moment conditions are equal to 0
Statistic: 7.9606
P-value: 0.3361
Distributed: chi2(7)

                    long-run response  std. error
inflation                        1.79        0.49
output_gap                       1.50        0.86

Taylor principle, H0: long-run inflation response <= 1:  t = 1.60   one-sided p-value = 0.054
```

How to read it

- **Coefficients.** GMM and 2SLS give similar estimates, as they should when the instruments are valid; large differences between them would be a warning. The GMM standard error on inflation is smaller (0.079 against 0.109), which is the efficiency gain.
- **Hansen J test.** p = 0.34, so the overidentifying restrictions are not rejected.
- **Long-run responses.** The long-run response to inflation is 1.79: in the long run the Fed raised the funds rate by about 1.8 points for each point of inflation. Its standard error, 0.49, comes from the delta method, because the long-run response is a nonlinear function of two coefficients.
- **The Taylor principle.** The estimate is above 1, but the one-sided test of "at most 1" has a p-value of 0.054. The point estimate supports the Taylor principle; the statistical evidence is borderline at the 5% level.

Report the test even when it is borderline. Writing that the Taylor principle "is satisfied" on the strength of a point estimate alone would overstate the result.

13.9 Robustness checks

```
def estimate(sample, lags_used, label):
    d_ = data.loc[sample[0]:sample[1]]
    inst = [f"{c}_l{k}" for c in endog for k in lags_used] + ["fedfunds_l2"]
    m = IVGMM(d_["fedfunds"], d_[exog], d_[endog], d_[inst], weight_type="kernel", kernel="bartlett",
  ↪ bandwidth=4).fit(iter_limit=2)
    r = m.params["fedfunds_l1"]
    print(f"{label:34s}{m.params['inflation'] / (1 - r):9.2f}{m.params['output_gap'] / (1 - r):9.2f}{r:9.2f}{m.
  ↪ j_stat.pval:10.3f}{int(m.nobs):6d}")

print(f"{'':34s}{'LR infl.':>9s}{'LR gap':>9s}{'rho':>9s}{'J p-val':>10s}{'N':>6s}")
estimate(("1979Q3", "2007Q4"), (1, 2, 3, 4), "Baseline: 1979-2007, 4 lags")
estimate(("1979Q3", "2007Q4"), (1, 2), "Fewer instruments: 2 lags")
```

```
estimate(("1979Q3", "2007Q4"), (2, 3, 4), "Instruments from lag 2 only")
estimate(("1983Q1", "2007Q4"), (1, 2, 3, 4), "After the disinflation: 1983-2007")
estimate(("1987Q3", "2007Q4"), (1, 2, 3, 4), "Greenspan: 1987-2007")
estimate(("1961Q1", "1979Q2"), (1, 2, 3, 4), "Before Volcker: 1961-1979")
```

```
                                LR infl.   LR gap      rho   J p-val     N
Baseline: 1979-2007, 4 lags         1.79     1.50     0.87     0.336   114
Fewer instruments: 2 lags           1.57     0.98     0.82     0.660   114
Instruments from lag 2 only         1.69     1.55     0.87     0.223   114
After the disinflation: 1983-2007   0.99     1.90     0.92     0.130   100
Greenspan: 1987-2007                0.35     2.51     0.91     0.214    82
Before Volcker: 1961-1979           0.52     1.69     0.78     0.646    74
```

How to read it

- **Instrument set.** With two lags, or with instruments dated from lag 2 only, the long-run inflation response stays between 1.6 and 1.8 and the J test never rejects. The result does not depend on the particular instruments.
- **Before Volcker.** For 1961–1979 the long-run response is 0.52, well below 1. Policy did not satisfy the Taylor principle in the 1970s. This contrast with the later period is the central finding of Clarida, Galí and Gertler (2000), and the data reproduce it.
- **Within the later period the result is fragile.** Dropping the 1979–82 disinflation lowers the response to 0.99, and for the Greenspan years alone it is 0.35. The strong response in the baseline comes largely from the Volcker disinflation.

Why would the estimate collapse after 1983? Inflation was low and stable, so there is little variation to identify the response from, and the smoothing coefficient above 0.9 makes the long-run ratio very sensitive. The measurement of the variables matters too: studies that use real-time data and forecasts, rather than the revised data and HP-filtered gap used here, find stronger responses in this period. A careful paper reports this sensitivity and does not claim more than the 1979–2007 result supports.

13.10 Part 2. Dynamic panels: why OLS and fixed effects both fail

The second half of this chapter applies the same idea to **panel data with a lagged dependent variable**, the setting of growth regressions across countries and of investment, leverage or employment equations across firms:

$$y_{it} = \alpha\, y_{i,t-1} + \beta' x_{it} + \eta_i + \lambda_t + \varepsilon_{it}$$

Here η_i is a fixed effect for each unit and λ_t a year effect. The lagged dependent variable is endogenous by construction, and the two usual estimators are biased in opposite directions.

Estimator	Problem	Direction of bias in α
Pooled OLS	$y_{i,t-1}$ contains η_i, which is in the error	**upwards**
Fixed effects	demeaning makes $y_{i,t-1}$ correlated with the demeaned error when T is small (Nickell bias)	**downwards**
Difference GMM (Arellano–Bond)	first-difference to remove η_i, then instrument $\Delta y_{i,t-1}$ with levels dated $t-2$ and earlier	consistent if the instruments are valid and strong

The example is the one the method was developed on: employment in 140 UK manufacturing firms, 1976–1984, from Arellano and Bond (1991). The question is **how persistent is firm employment, and how does it respond to wages and capital?**

```
from diff_gmm import diff_gmm          # helper file: time-series-guide/examples/diff_gmm.py
import statsmodels.formula.api as smf

raw = pd.read_csv("employment.csv")
```

```
panel = pd.DataFrame({"firm": raw["firm"], "year": raw["year"], "n": np.log(raw["emp"]), "w": np.log(raw["wage
  ↪ "]),
                      "k": np.log(raw["capital"]), "ys": np.log(raw["output"])})
by_firm = panel.groupby("firm")
for var, lag in (("n", 1), ("n", 2), ("w", 1), ("k", 1), ("k", 2), ("ys", 1), ("ys", 2)):
    panel[f"{var}_l{lag}"] = by_firm[var].shift(lag)

print(f"{panel['firm'].nunique()} firms, {panel['year'].min()}-{panel['year'].max()}, {len(panel)}
  ↪ observations")
print("Years per firm:", by_firm["year"].count().value_counts().sort_index().to_dict())
```

```
140 firms, 1976-1984, 1031 observations
Years per firm: {7: 103, 8: 23, 9: 14}
```

How to read it

This is the typical shape for dynamic panel GMM: **many units (140) and few periods (7 to 9)**. The method relies on the number of units being large. With a long panel of few units, such as 20 countries over 40 years, fixed effects is the better choice and the Nickell bias is small.

The variables are in logs: employment `n`, the real wage `w`, capital `k` and industry output `ys`. The model has two lags of employment, current and lagged wages, and current and two lags of capital and output, as in the original paper. Download the helper file and the data into your working folder.

13.11 Panel step 1: pooled OLS and fixed effects give the bounds

```
controls = ["w", "w_l1", "k", "k_l1", "k_l2", "ys", "ys_l1", "ys_l2"]
formula = "n ~ n_l1 + n_l2 + " + " + ".join(controls) + " + C(year)"
sample = panel.dropna()
cluster = {"cov_type": "cluster", "cov_kwds": {"groups": sample["firm"]}}

pooled = smf.ols(formula, sample).fit(**cluster)
fixed = smf.ols(formula + " + C(firm)", sample).fit(**cluster)
print(f"{'':16s}{'L1.n':>8s}{'L2.n':>8s}{'w':>8s}{'k':>8s}")
for name, m in (("Pooled OLS", pooled), ("Fixed effects", fixed)):
    print(f"{name:16s}{m.params['n_l1']:8.3f}{m.params['n_l2']:8.3f}{m.params['w']:8.3f}{m.params['k']:8.3f}")
```

```
                  L1.n    L2.n       w       k
Pooled OLS       1.045  -0.077  -0.524   0.343
Fixed effects    0.733  -0.139  -0.560   0.388
```

How to read it

The coefficient on lagged employment is 1.05 by pooled OLS and 0.73 by fixed effects. Because the first is biased upwards and the second downwards, **a credible estimate should lie between them**. This bounds check, suggested by Bond (2002), costs two regressions and is the first thing to run in any dynamic panel study.

13.12 Panel step 2: difference GMM

```
gmm_panel = diff_gmm(panel, unit="firm", time="year", y="n", lags_y=2, x=controls, gmm_lags=(2, None))
print(gmm_panel["table"].loc[["L1.n", "L2.n"] + controls].round(4))
print(f"\nObservations: {gmm_panel['nobs']}   firms: {gmm_panel['units']}   instruments: {gmm_panel['
  ↪ instruments']}")

t = gmm_panel["table"]["coef"]
persistence = 1 - t["L1.n"] - t["L2.n"]
print(f"Long-run wage elasticity:    {(t['w'] + t['w_l1']) / persistence:.2f}")
print(f"Long-run capital elasticity: {(t['k'] + t['k_l1'] + t['k_l2']) / persistence:.2f}")
```

```
         coef  std err       z  p-value
L1.n   0.6862   0.1446  4.7459   0.0000
L2.n  -0.0854   0.0560 -1.5238   0.1276
w     -0.6078   0.1782 -3.4108   0.0006
w_l1   0.3926   0.1680  2.3371   0.0194
k      0.3568   0.0590  6.0462   0.0000
```

```
k_l1  -0.0580   0.0732 -0.7926   0.4280
k_l2  -0.0199   0.0327 -0.6098   0.5420
ys     0.6085   0.1725  3.5269   0.0004
ys_l1 -0.7112   0.2317 -3.0691   0.0021
ys_l2  0.1058   0.1412  0.7493   0.4537

Observations: 611   firms: 140   instruments: 41
Long-run wage elasticity:    -0.54
Long-run capital elasticity: 0.70
```

How to read it

These are one-step estimates with standard errors robust to heteroskedasticity and to correlation within each firm. They reproduce the published results for this dataset exactly (a coefficient of 0.686 on lagged employment), which is how the helper function was checked.

- **Wages.** A 1% rise in the real wage lowers employment by 0.61% in the same year. Allowing for the dynamics, the long-run elasticity is about –0.54.
- **Capital.** A 1% rise in the capital stock raises employment by 0.36% at once, and by about 0.70% in the long run.
- **Persistence.** The coefficient on lagged employment is 0.69. Compare it with the bounds from the previous step: it lies **below** the fixed-effects estimate of 0.73, not between the two. Keep this in mind for the next two steps.

`gmm_lags=(2, None)` uses all available lags from $t-2$ back as instruments, which gives 41 instruments for 140 firms.

13.13 Panel step 3: the three specification tests

```
z1, p1 = gmm_panel["ar1"]; z2, p2 = gmm_panel["ar2"]; j, dof, pj = gmm_panel["hansen"]
print(f"Arellano-Bond AR(1) test:  z = {z1:6.2f}   p-value = {p1:.3f}")
print(f"Arellano-Bond AR(2) test:  z = {z2:6.2f}   p-value = {p2:.3f}")
print(f"Hansen J test:             chi2({dof}) = {j:.2f}   p-value = {pj:.3f}")
print(f"Instruments: {gmm_panel['instruments']}    firms: {gmm_panel['units']}")
```

```
Arellano-Bond AR(1) test:  z =  -3.60   p-value = 0.000
Arellano-Bond AR(2) test:  z =  -0.52   p-value = 0.606
Hansen J test:             chi2(25) = 31.38   p-value = 0.177
Instruments: 41    firms: 140
```

How to read it

A dynamic panel GMM table is not complete without these.

- **AR(1)** is significant, which is expected: first-differencing creates first-order correlation by construction. It tells you nothing about validity.
- **AR(2) is the test that matters.** Its null is no second-order serial correlation in the differenced residuals. If it is rejected, the errors in levels are serially correlated and levels dated $t-2$ are not valid instruments. Here p = 0.61, so the instruments pass. If it had rejected, you would start the instruments at $t-3$.
- **Hansen J** has the null that the instruments are valid as a group. Here p = 0.18, which does not reject. Be suspicious of p-values very close to 1: that is a sign of too many instruments, not of very good ones.
- **Instrument count.** 41 instruments for 140 firms. The count must stay well below the number of units, because too many instruments overfit the endogenous variable and weaken the Hansen test.

13.14 Panel step 4: robustness to the instrument set

```
specs = {"All lags (baseline)": {}, "Lags 2 to 4": {"gmm_lags": (2, 4)}, "Lags 2 to 3": {"gmm_lags": (2, 3)},
         "Lags 3 and deeper": {"gmm_lags": (3, None)}, "Collapsed instruments": {"collapse": True}}
print(f"{'':24s}{'L1.n':>7s}{'(s.e.)':>9s}{'w':>8s}{'k':>8s}{'instr.':>8s}{'AR(2) p':>9s}{'Hansen p':>10s}")
for name, options in specs.items():
    r = diff_gmm(panel, unit="firm", time="year", y="n", lags_y=2, x=controls, **options)
    c, se = r["table"]["coef"], r["table"]["std err"]
```

```
    print(f"{name:24s}{c['L1.n']:7.3f}{'(' + format(se['L1.n'], '.3f') + ')':>9s}{c['w']:8.3f}{c['k']:8.3f}{r
  ↪ ['instruments']:8d}{r['ar2'][1]:9.3f}{r['hansen'][2]:10.3f}")
print(f"\nBounds for L1.n:  fixed effects {fixed.params['n_l1']:.3f}   pooled OLS {pooled.params['n_l1']:.3f}")
  ↪
```

```
                          L1.n   (s.e.)       w       k  instr.  AR(2) p  Hansen p
All lags (baseline)      0.686  (0.145)  -0.608   0.357      41    0.606     0.177
Lags 2 to 4              0.475  (0.253)  -0.610   0.364      31    0.665     0.181
Lags 2 to 3              0.392  (0.265)  -0.600   0.370      26    0.688     0.078
Lags 3 and deeper        0.986  (0.191)  -0.683   0.317      35    0.068     0.655
Collapsed instruments    1.358  (0.365)  -0.710   0.311      21    0.405     0.289

Bounds for L1.n:  fixed effects 0.733   pooled OLS 1.045
```

How to read it

The table separates what this dataset can tell you from what it cannot.

- **The wage and capital effects are robust.** Across all five instrument sets the wage coefficient stays between –0.60 and –0.71 and the capital coefficient between 0.31 and 0.37. Neither the AR(2) test nor the Hansen test rejects at the 5% level in any row, although two rows come close (p = 0.07 and 0.08).
- **The persistence coefficient is not.** It ranges from 0.39 to 1.36 depending on the instruments, with large standard errors, and in four of the five rows it falls outside the bounds of 0.73 and 1.05.

This is a known weakness of difference GMM, not a mistake in the code. When a series is highly persistent, its past **levels** are weak instruments for its current **changes**, and the estimate of α becomes imprecise and biased towards the fixed-effects value. Blundell and Bond (1998) proposed **system GMM** for exactly this case: it adds the equation in levels, instrumented with lagged differences, and is now the standard estimator for persistent series such as output, capital or employment. In Stata it is `xtabond2` without the `noleveleq` option; in R, `plm::pgmm` with `transformation = "ld"`.

So the paper can report the wage and capital elasticities with confidence, and should say plainly that the degree of persistence is not well identified by difference GMM in this sample.

13.15 Write the paper

Section	What goes in it	From which step
Identification	Why the regressor is endogenous; what the instruments are; why each is relevant and exogenous	Section 13.1
Data	Sources, sample, construction of variables	Section 13.2
Results	OLS and IV estimates side by side	Sections 13.3, 13.4 and 13.8
Instrument diagnostics	First-stage F and partial R^2, overidentification test, endogeneity test, all in the main table	Sections 13.5 to 13.7
Robustness	Alternative instrument sets and samples	Section 13.9
Dynamic panel	Pooled OLS and fixed-effects bounds, GMM estimates, AR(1), AR(2), Hansen test, number of instruments and of units	Sections 13.10 to 13.14

How to write it: results

Table 2 reports OLS, 2SLS and two-step GMM estimates of the policy rule for 1979:Q3–2007:Q4. The instruments are strong, with first-stage F-statistics of 301 for inflation and 58 for the output gap, and the Hansen J test does not reject the overidentifying restrictions ($p = 0.34$). A Wu–Hausman test rejects the exogeneity of inflation and the output gap ($p < 0.01$), confirming the need for instrumental-variable estimation. The GMM estimates imply a smoothing coefficient of 0.87 and long-run responses of 1.79 to inflation (s.e. 0.49) and 1.50 to the output gap (s.e. 0.86). The point estimate of the inflation response exceeds one, consistent with the Taylor principle, although the hypothesis that it is at most one is rejected only at the 10% level ($p = 0.054$). For 1961–1979 the corresponding estimate is 0.52. The post-1979 estimate is sensitive to the sample: it falls to about one when the 1979–82 disinflation is excluded.

How to write it: dynamic panel results

Table 4 reports one-step difference GMM estimates of the employment equation for 140 firms over 1976–1984, with standard errors robust to heteroskedasticity and within-firm correlation. Lagged employment is instrumented with its levels dated $t-2$ and earlier, giving 41 instruments. The Arellano–Bond test does not reject the absence of second-order serial correlation in the differenced residuals ($z = -0.52$, $p = 0.61$), and the Hansen test does not reject the overidentifying restrictions ($\chi^2(25) = 31.4$, $p = 0.18$). The short-run wage elasticity of employment is −0.61 (s.e. 0.18) and the short-run capital elasticity is 0.36 (s.e. 0.06); both are stable across alternative instrument sets. The coefficient on lagged employment is 0.69, slightly below the fixed-effects estimate of 0.73, and it is sensitive to the choice of instruments. This pattern is characteristic of weak instruments in difference GMM for persistent series, so the degree of employment persistence should be interpreted with caution.

What a referee will ask

- **Why are your instruments exogenous?** This is the central question, and no test answers it. Give an economic argument for each instrument.
- **Are they strong?** Report the first-stage F for every endogenous regressor.
- **Do the results depend on the instrument set?** Show at least one alternative set.
- **How many instruments?** Report the count. In panels it must be below the number of units.
- **Does the estimate lie between fixed effects and pooled OLS?** If not, as in the panel example here, suspect weak instruments and consider system GMM.
- **Did AR(2) pass?** If it is rejected, lag-2 instruments are invalid.
- **Is IV needed at all?** Report the endogeneity test and the OLS estimates.
- **How was the output gap measured?** The gap here uses a two-sided HP filter, which uses future data; a real-time measure would be a useful check, and the paper should say so.

PART III

Panel-Data Methods

Overview

A panel follows many units over time, and its two dimensions pull in different directions. When T is short and N is large, the central problem is unobserved heterogeneity across units: whether it is correlated with the regressors decides between fixed and random effects (Mundlak, 1978; Hausman, 1978), and whether the model is dynamic decides whether GMM is needed (Arellano and Bond, 1991; Blundell and Bond, 1998). When T is long, the issues of time series return: unit roots, cointegration, and, because units in a macro panel respond to common shocks, cross-sectional dependence (Pesaran, 2006).

The decision rules of Chapter 1 therefore start from the size of N and T, and the chapters of this part follow the same order. The first four cover the short-panel estimators. The next five handle long panels, with homogeneous slopes (Driscoll–Kraay standard errors, Driscoll and Kraay, 1998) or heterogeneous slopes (mean group and pooled mean group, Pesaran and Smith, 1995; Pesaran et al., 1999), and common shocks (CS-ARDL, Chudik and Pesaran, 2015). The last two chapters estimate dynamic responses in a panel and a regression with a threshold (Hansen, 1999).

Chapters at a glance

Ch.	Chapter	Use it when
14	Pooled OLS	Short panel, no significant unit effects
15	Fixed Effects	Unit effects correlated with the regressors
16	Random Effects	Unit effects present, uncorrelated with the regressors
17	Difference GMM for Dynamic Panels	Dynamic panel, large N, short T
18	Fixed Effects with Driscoll–Kraay Standard Errors	Long panel, stationary variables, homogeneous slopes
19	Mean Group and CCE Mean Group	Long panel, stationary variables, heterogeneous slopes
20	Panel ARDL: Pooled Mean Group	Long-run relationship, cross-sectionally independent units
21	Cross-Sectionally Augmented ARDL	Long-run relationship with common shocks
22	Panel Model in First Differences	All variables I(1), not cointegrated
23	Panel Local Projections	You want the dynamic response of an outcome to a shock in a panel of units, horizon by horizon, with unit fixed effects
24	Panel Threshold Regression (Hansen)	The effect of a regressor may differ above and below a threshold in another variable, and you want to estimate where the threshold is

Further reading

Wooldridge (2010) is the standard graduate reference for static and dynamic panels. For macro panels with large T, the starting points are Pesaran (2006) on common correlated effects, Pesaran (2007) on unit roots under cross-sectional dependence, and Pesaran and Yamagata (2008) on testing slope homogeneity. Cointegration tests for panels are due to Kao (1999).

CHAPTER 14

Pooled OLS

Applies when. Short panel, no significant unit effects

Overview. Use this chapter when the panel is short and neither the F test nor the Breusch–Pagan LM test finds unit-specific effects. Pooling is then defensible, but the observations on the same unit are still not independent, so the chapter shows how to get the inference right and how to show a referee that pooling is safe.

Data. 140 UK manufacturing firms, annual growth rates, 1977–1984 (Arellano–Bond data)

Software. Python · linearmodels · statsmodels

14.1 The question and the model

Pooled OLS is rarely the right model for variables in levels: firms, workers and countries almost always differ permanently. It often becomes the right model once the variables are **growth rates**, because differencing removes the permanent differences in levels. The question here: **how does a firm's employment growth respond to the growth of its wages, capital and industry demand?**

$$\Delta \ln n_{it} = \beta_1 \Delta \ln w_{it} + \beta_2 \Delta \ln k_{it} + \beta_3 \Delta \ln ys_{it} + \lambda_t + \varepsilon_{it}$$

Symbol	Variable	Definition	Expected sign
$\Delta \ln n_{it}$	`g_emp`	Employment growth, %	dependent variable
$\Delta \ln w_{it}$	`g_wage`	Real wage growth, %	negative: labour demand slopes down
$\Delta \ln k_{it}$	`g_capital`	Capital stock growth, %	positive: capital and labour are complements
$\Delta \ln ys_{it}$	`g_output`	Industry output growth, %	positive: demand for the firm's goods
λ_t	—	Year effects: the 1980–81 recession	common to all firms

Before you estimate anything

Pooling assumes that all units share the same intercept and the same slopes. With growth rates the first assumption is plausible: a firm with permanently high employment does not need to have permanently high employment *growth*. But it is still an assumption, and the F and LM tests below check it. If they reject, use Chapter 15 instead.

How to write it: model section

We relate employment growth to the growth of the real wage, the capital stock and industry output, $\Delta \ln n_{it} = \beta_1 \Delta \ln w_{it} + \beta_2 \Delta \ln k_{it} + \beta_3 \Delta \ln ys_{it} + \lambda_t + \varepsilon_{it}$, where λ_t are year effects. Specifying the model in growth rates removes permanent firm-specific differences in employment levels.

14.2 Load the data and build the growth rates

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy import stats
from linearmodels.panel import PanelOLS, PooledOLS, RandomEffects
```

```
raw = pd.read_csv("employment.csv").sort_values(["firm", "year"])
for v in ["log_emp", "log_wage", "log_capital", "log_output"]:
    raw["g_" + v[4:]] = 100 * raw.groupby("firm")[v].diff()      # growth rate in percent
df = raw.dropna().set_index(["firm", "year"])

xs = ["g_wage", "g_capital", "g_output"]
T_i = df.groupby(level=0).size()
print(f"{len(T_i)} firms, {len(df)} firm-years, years per firm from {T_i.min()} to {T_i.max()} (unbalanced)")
print(df[["g_emp"] + xs].describe().T[["mean", "std", "min", "max"]].round(2))
```

```
140 firms, 891 firm-years, years per firm from 6 to 8 (unbalanced)
           mean    std     min    max
g_emp     -4.38  13.77  -99.75  80.64
g_wage     0.55   8.79  -67.53  92.45
g_capital -3.59  16.16 -108.17  88.42
g_output  -2.16   5.16  -22.49   8.67
```

How to read it

The panel is short and unbalanced: 140 firms with 6 to 8 growth observations each. Taking differences loses the first year of every firm. Employment fell on average over the period, the years of the early-1980s recession in British manufacturing.

The decision rules of Chapter 1, run on this file, finds no significant firm effects and recommends pooled OLS. The next steps show why.

14.3 Test for firm effects: F and Breusch–Pagan LM

```
years = pd.get_dummies(df.index.get_level_values(1), prefix="y", drop_first=True, dtype=float).set_index(df.
  ↪ index)
y = df["g_emp"]
X = sm.add_constant(df[xs].join(years))

fe = PanelOLS(y, X, entity_effects=True).fit()
f = fe.f_pooled
print(f"F test that all firm effects are zero:  F({f.df}, {f.df_denom}) = {f.stat:.2f}   p-value = {f.pval:.3f}
  ↪ ")

pooled = PooledOLS(y, X).fit(cov_type="clustered", cluster_entity=True)
e = pooled.resids
n, Ti = len(e), e.groupby(level=0).size()
lm = n ** 2 / (2 * (Ti * (Ti - 1)).sum()) * ((e.groupby(level=0).sum() ** 2).sum() / (e ** 2).sum() - 1) ** 2
print(f"Breusch-Pagan LM test:                  chi2(1) = {lm:.2f}   p-value = {stats.chi2.sf(lm, 1):.3f}")
```

```
F test that all firm effects are zero:  F(139, 741) = 1.10   p-value = 0.223
Breusch-Pagan LM test:                  chi2(1) = 0.14   p-value = 0.708
```

How to read it

- **F test.** Null: all 140 firms have the same intercept. It is not rejected, so firm-specific intercepts do not improve the fit.
- **Breusch–Pagan LM test.** Null: the variance of a random firm effect is zero. It is not rejected either.

With neither test rejecting, pooling the firms is defensible. Fixed effects would estimate 139 extra intercepts for no gain in fit, and lose precision.

14.4 Estimate the pooled model with clustered standard errors

```
print(pooled.summary.tables[1])
naive = PooledOLS(y, X).fit()
print("\nStandard errors:      clustered   conventional")
for v in xs:
    print(f"{v:18s}{pooled.std_errors[v]:12.4f}{naive.std_errors[v]:14.4f}")
```

```
                          Parameter Estimates
```

```
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const         -3.8700     2.1063    -1.8373     0.0665     -8.0041      0.2640
g_wage        -0.4317     0.1624    -2.6577     0.0080     -0.7505     -0.1129
g_capital      0.3964     0.0554     7.1609     0.0000      0.2878      0.5051
g_output       0.4925     0.1558     3.1610     0.0016      0.1867      0.7983
y_1978         1.6232     1.9220     0.8446     0.3986     -2.1489      5.3954
y_1979         3.3305     2.2949     1.4513     0.1471     -1.1735      7.8345
y_1980         4.7149     2.7211     1.7327     0.0835     -0.6258      10.056
y_1981         1.6101     3.5839     0.4493     0.6533     -5.4238      8.6440
y_1982         1.3137     3.3475     0.3925     0.6948     -5.2563      7.8838
y_1983         1.6624     2.7068     0.6141     0.5393     -3.6502      6.9749
y_1984         2.4376     2.9155     0.8361     0.4033     -3.2846      8.1598
==============================================================================

Standard errors:      clustered   conventional
g_wage                   0.1624         0.0457
g_capital                0.0554         0.0247
g_output                 0.1558         0.1045
```

How to read it

- **Wages.** A 1 percentage point faster growth of the real wage lowers employment growth by about 0.4 points in the same year.
- **Capital.** Faster capital growth is associated with faster employment growth: firms that invest also hire.
- **Industry output.** Faster growth of demand in the firm's industry raises employment growth: about half a point for each point of output growth.

Clustered standard errors are not optional. Even without firm effects, the errors of the same firm can be correlated across years and their variance can differ across firms. The second table compares them: the clustered standard error of the wage coefficient is more than three times the conventional one. With conventional standard errors you would overstate the precision of every estimate. Report the clustered ones.

14.5 Diagnostic tests: serial correlation and heteroskedasticity

```
r = pooled.resids.rename("e").to_frame()
r["e_lag"] = r.groupby(level=0)["e"].shift(1)
r = r.dropna()
ar = PooledOLS(r["e"], sm.add_constant(r[["e_lag"]])).fit(cov_type="clustered", cluster_entity=True)
print(f"AR(1) coefficient of the residuals: {ar.params['e_lag']:.3f}  (s.e. {ar.std_errors['e_lag']:.3f}, p-
  ↪ value {ar.pvalues['e_lag']:.3f})")

from statsmodels.stats.diagnostic import het_breuschpagan
bp = het_breuschpagan(pooled.resids, X)
print(f"Breusch-Pagan heteroskedasticity test: LM = {bp[0]:.2f}   p-value = {bp[1]:.4f}")
```

```
AR(1) coefficient of the residuals: 0.087  (s.e. 0.049, p-value 0.075)
Breusch-Pagan heteroskedasticity test: LM = 33.14   p-value = 0.0003
```

How to read it

- **Serial correlation.** Regressing each residual on the same firm's residual a year earlier gives a small positive coefficient (0.09), significant only at the 10% level. The serial correlation that remains is weak.
- **Heteroskedasticity.** The Breusch–Pagan test rejects clearly: the error variance differs across observations, as is usual with firm data (small firms have more volatile growth).

Both problems leave the pooled OLS coefficients unbiased; they only affect the standard errors. **Clustering by firm corrects for both**, which is why it was used from the start.

14.6 Robustness checks

```
fe_c = PanelOLS(y, X, entity_effects=True).fit(cov_type="clustered", cluster_entity=True)
re_c = RandomEffects(y, X).fit(cov_type="clustered", cluster_entity=True)
```

```
no_years = PooledOLS(y, sm.add_constant(df[xs])).fit(cov_type="clustered", cluster_entity=True)
trimmed = df[(df["g_emp"].abs() < df["g_emp"].abs().quantile(0.99))]
tr = PooledOLS(trimmed["g_emp"], sm.add_constant(trimmed[xs].join(years))).fit(cov_type="clustered",
  ↪ cluster_entity=True)

print(f"{'':28s}" + "".join(f"{v:>16s}" for v in xs))
for name, m in (("Pooled OLS (baseline)", pooled), ("Fixed effects", fe_c), ("Random effects", re_c),
                ("Without year effects", no_years), ("Drop top 1% |g_emp|", tr)):
    print(f"{name:28s}" + "".join(f"{m.params[v]:8.3f} ({m.std_errors[v]:.3f})" for v in xs))
```

```
                                g_wage       g_capital        g_output
Pooled OLS (baseline)         -0.432 (0.162)   0.396 (0.055)   0.493 (0.156)
Fixed effects                 -0.485 (0.168)   0.328 (0.063)   0.595 (0.188)
Random effects                -0.432 (0.162)   0.396 (0.055)   0.493 (0.156)
Without year effects          -0.416 (0.136)   0.408 (0.049)   0.409 (0.112)
Drop top 1% |g_emp|           -0.209 (0.081)   0.297 (0.035)   0.429 (0.113)
```

How to read it

- **Fixed effects** gives coefficients close to pooled OLS. This is what the F test predicted, and it is the strongest evidence a referee will accept that pooling is harmless.
- **Random effects** is identical to pooled OLS. The estimated variance of the firm effect is zero, so the random-effects transformation subtracts nothing: this is the LM test result seen from the other side.
- **Without year effects** the output coefficient changes the most: industry output and the recession move together, so without year effects output growth picks up the common cycle. Keep the year effects.
- **Dropping the most extreme 1% of employment growth** (mergers, plant closures) leaves the capital and output effects similar, but **halves the wage effect**, to –0.21. It remains significant, but its size depends on a few firm-years with very large swings. Report this check: a reader needs to know that the magnitude, though not the sign, is fragile.

14.7 Build the results table for the paper

```
def star(p):
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) Pooled OLS": pooled, "(2) FE": fe_c, "(3) RE": re_c}
print(f"{'':14s}" + "".join(f"{c:>16s}" for c in cols))
for v in xs:
    print(f"{v:14s}" + "".join(f"{m.params[v]:13.4f}{star(m.pvalues[v]):3s}" for m in cols.values()))
    print(f"{'':14s}" + "".join(f"{'(' + format(m.std_errors[v], '.4f') + ')':>13s}   " for m in cols.values())
  ↪ )
print(f"{'Observations':14s}" + "".join(f"{m.nobs:>13d}   " for m in cols.values()))
print("Year effects in all columns. Standard errors clustered by firm. *** p<0.01, ** p<0.05, * p<0.10.")
print(f"F test for firm effects: F({f.df}, {f.df_denom}) = {f.stat:.2f} (p = {f.pval:.3f}); Breusch-Pagan LM =
  ↪  {lm:.2f} (p = {stats.chi2.sf(lm, 1):.3f})")
```

```
               (1) Pooled OLS          (2) FE          (3) RE
g_wage              -0.4317***      -0.4850***      -0.4317***
                   (0.1624)        (0.1676)        (0.1624)
g_capital            0.3964***       0.3284***       0.3964***
                   (0.0554)        (0.0631)        (0.0554)
g_output             0.4925***       0.5951***       0.4925***
                   (0.1558)        (0.1876)        (0.1558)
Observations            891             891             891
Year effects in all columns. Standard errors clustered by firm. *** p<0.01, ** p<0.05, * p<0.10.
F test for firm effects: F(139, 741) = 1.10 (p = 0.223); Breusch-Pagan LM = 0.14 (p = 0.708)
```

How to read it

Column (1) is the preferred estimate; columns (2) and (3) show that it does not depend on the treatment of firm effects. The F and LM statistics in the notes are the justification for pooling.

14.8 Write the paper

How to write it: specification

Because the model is specified in growth rates, permanent differences in employment levels across firms are differenced out. Consistent with this, an F test does not reject the hypothesis that firm-specific intercepts are jointly zero, and a Breusch–Pagan LM test does not reject the absence of random firm effects. We therefore estimate the model by pooled OLS with year effects, and cluster standard errors by firm to allow for heteroskedasticity and serial correlation within firms.

How to write it: results

Real wage growth has a negative and significant effect on employment growth: a one percentage point faster increase in the real wage reduces employment growth by about 0.4 percentage points (Table 1, column 1). Capital growth and industry output growth are positively associated with employment growth. Fixed- and random-effects estimates are very close to the pooled estimates (columns 2 and 3), confirming that unobserved firm heterogeneity does not drive the results. The magnitude of the wage effect is, however, sensitive to a small number of extreme observations: excluding the top 1 percent of absolute employment changes reduces it to about –0.2, still significant.

Common mistakes

- Pooling variables in levels without testing for unit effects. In levels the F test almost always rejects.
- Reporting conventional standard errors. Pooled observations on the same unit are not independent; cluster by unit.
- Omitting year effects. Common shocks then load onto whichever regressor moves with the cycle.
- Using pooled OLS for a dynamic model with a lagged dependent variable. Pooled OLS is then biased upwards; see Chapter 17.

CHAPTER 15

Fixed Effects

Applies when. Unit effects correlated with the regressors

Overview. Use this chapter when the panel is short, the units differ systematically, and the Hausman or Mundlak test shows that those differences are correlated with the regressors. It takes one real panel from the model equation to the finished results table, with the specification tests, diagnostics, robustness checks and sample text for each part of the paper.

Data. 545 U.S. men, annual wages, 1980–1987 (Wooldridge wagepan)

Software. Python · linearmodels · statsmodels

15.1 The question and the model

The question is a classic of labour economics: **do union members earn more than comparable non-members?** A cross-section cannot answer it, because workers who join unions may differ in ability, motivation or occupation in ways the data do not record. A panel that follows the same workers over time can: it compares each worker with himself, in years with and without union membership.

$$\ln w_{it} = \beta_1 \, union_{it} + \beta_2 \, married_{it} + \beta_3 \, exper_{it}^2 + \alpha_i + \lambda_t + \varepsilon_{it}$$

Symbol	Variable in the file	Definition	Expected sign
$\ln w_{it}$	`lwage`	Log hourly wage	dependent variable
$union_{it}$	`union`	1 if wage set by a union contract	positive: the union wage premium
$married_{it}$	`married`	1 if married	positive: the marriage premium
$exper_{it}^2$	`expersq`	Years of experience, squared	negative: wage growth slows with experience
α_i	—	Worker fixed effect: ability, motivation, occupation	unobserved
λ_t	—	Year effect: inflation, the business cycle	common to all workers

Experience itself is not in the model. Every worker gains exactly one year of experience per year, so with worker and year effects the level of experience is perfectly collinear and cannot be estimated. Its square still varies within workers.

Before you estimate anything

Write down what the fixed effect α_i stands for in your application and why it may be correlated with the regressor of interest. Here: more able workers may be less likely to join a union and also earn more. If that is true, pooled OLS mixes the union effect with the ability effect, and only the within-worker comparison isolates it. That argument is the reason for using fixed effects, and the paper must make it.

How to write it: model section

We estimate $\ln w_{it} = \beta_1 union_{it} + \beta_2 married_{it} + \beta_3 exper_{it}^2 + \alpha_i + \lambda_t + \varepsilon_{it}$, *where* α_i *is a worker fixed effect and* λ_t *a year effect. The worker effect absorbs all time-invariant characteristics, including unobserved ability, which may be correlated with union status. The union premium* β_1 *is therefore identified from workers who change union status during the sample.*

15.2 Load the panel and confirm the starting point

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels.panel import PanelOLS, PooledOLS, RandomEffects, FirstDifferenceOLS, compare

raw = pd.read_csv("wagepan_full.csv")
df = raw.set_index(["nr", "year"])
print(f"{df.index.get_level_values(0).nunique()} workers x {df.index.get_level_values(1).nunique()} years = {
  ↪ len(df)} rows")

xs = ["union", "married", "expersq"]
for v in ["lwage"] + xs + ["educ"]:
    x = df[v]
    within = x - x.groupby(level=0).transform("mean")
    print(f"{v:8s} overall sd = {x.std():6.3f}   between sd = {x.groupby(level=0).mean().std():6.3f}   within
  ↪ sd = {within.std():6.3f}")

switchers = df["union"].groupby(level=0).agg(lambda s: s.nunique() > 1)
print(f"Workers who change union status at least once: {switchers.sum()} of {len(switchers)}")
```

```
545 workers x 8 years = 4360 rows
lwage    overall sd =  0.533   between sd =  0.391   within sd =  0.362
union    overall sd =  0.430   between sd =  0.329   within sd =  0.276
married  overall sd =  0.496   between sd =  0.377   within sd =  0.324
expersq  overall sd = 40.782   between sd = 26.351   within sd = 31.143
educ     overall sd =  1.746   between sd =  1.748   within sd =  0.000
Workers who change union status at least once: 246 of 545
```

How to read it

The panel is **balanced and short**: 545 workers, each observed in all 8 years. With so few periods, unit-root tests are not the issue; the question is how to treat the worker effects. The decision rules of Chapter 1, applied to this file, give recommended fixed effects (the F, LM and Mundlak tests are repeated below).

The within standard deviations matter most. Union status has a within standard deviation of about 0.28: plenty of workers move in and out of union jobs, and the line below shows how many. **Those switchers identify the fixed-effects estimate.** Education has a within standard deviation of zero: it never changes for these men, so fixed effects cannot estimate its return.

15.3 The benchmark: pooled OLS

```
years = pd.get_dummies(df.index.get_level_values(1), prefix="y", drop_first=True, dtype=float).set_index(df.
  ↪ index)
y = df["lwage"]
X = sm.add_constant(df[xs].join(years))

pooled = PooledOLS(y, X).fit(cov_type="clustered", cluster_entity=True)
print(pooled.params[xs].round(4).to_string())
print("Clustered standard errors:")
print(pooled.std_errors[xs].round(4).to_string())
```

```
union      0.1768
married    0.1521
expersq   -0.0021
Clustered standard errors:
union      0.0290
married    0.0273
expersq    0.0005
```

How to read it

Pooled OLS treats the 4,360 rows as if they were 4,360 different people. It puts the union premium at about 18%. This estimate uses both the comparison across workers (union members against non-members) and the comparison over time, so if union members differ in unobserved ways, the premium is biased.

Even in pooled OLS, **the standard errors must be clustered by worker**: the same man appears eight times, and his errors are correlated across years. Unclustered standard errors would be far too small.

15.4 Estimate the fixed-effects model

```
fe = PanelOLS(y, sm.add_constant(df[xs]), entity_effects=True, time_effects=True).fit(cov_type="clustered",
  ↪ cluster_entity=True)
print(fe.summary.tables[1])
print(f"Observations: {fe.nobs}   workers: {fe.entity_info['total']:.0f}")
```

```
                           Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const          1.8706     0.0463     40.426     0.0000      1.7799      1.9613
union          0.0800     0.0243     3.2929     0.0010      0.0324      0.1276
married        0.0467     0.0224     2.0805     0.0375      0.0027      0.0907
expersq       -0.0052     0.0009    -5.9911     0.0000     -0.0069     -0.0035
==============================================================================
Observations: 4360   workers: 545
```

How to read it

- **Union.** Comparing the same worker in union and non-union years, union membership raises the wage by about 8%. That is less than half the pooled OLS premium: much of the cross-sectional premium reflects who joins unions, not what unions pay.
- **Married.** The marriage premium falls from about 15% in pooled OLS to about 5%, significant only at the 5% level. Much of the cross-sectional marriage premium is a selection effect.
- **Experience squared.** Negative and highly significant: wage growth slows with experience.
- **Standard errors.** Clustered by worker, which allows for any pattern of heteroskedasticity and serial correlation within each worker. This is the standard choice for a short panel with many units.

The constant reported by `PanelOLS` is the average of the fixed effects and has no economic meaning.

15.5 Test 1: do the workers differ? The F test

```
f = fe.f_pooled
print(f"F test that all worker effects are zero: F({f.df}, {f.df_denom}) = {f.stat:.2f}   p-value = {f.pval:.4
  ↪ f}")
```

```
F test that all worker effects are zero: F(551, 3805) = 10.07   p-value = 0.0000
```

How to read it

The null hypothesis is that every worker has the same intercept, in which case pooled OLS would be enough. The F statistic is large and the p-value is essentially zero: **the worker effects are jointly significant**, so ignoring them is not an option. The remaining question is whether to treat them as fixed or random.

15.6 Test 2: fixed or random? Hausman and Mundlak

```
re = RandomEffects(y, X).fit(cov_type="clustered", cluster_entity=True)
re_plain = RandomEffects(y, X).fit()
fe_plain = PanelOLS(y, sm.add_constant(df[xs].join(years)), entity_effects=True).fit()

# conventional Hausman test (non-robust variances, as in the textbook formula)
d = fe_plain.params[xs] - re_plain.params[xs]
V = fe_plain.cov.loc[xs, xs] - re_plain.cov.loc[xs, xs]
H = float(d @ np.linalg.inv(V) @ d)
from scipy import stats
```

```
print(f"Hausman test:  chi2({len(xs)}) = {H:.2f}   p-value = {stats.chi2.sf(H, len(xs)):.4f}")

# Mundlak test: add each worker's average of the regressors, test them jointly with clustered errors
means = df[xs].groupby(level=0).transform("mean").add_suffix("_mean")
mundlak = RandomEffects(y, X.join(means)).fit(cov_type="clustered", cluster_entity=True)
w = mundlak.wald_test(formula=" = ".join(means.columns) + " = 0")
print(f"Mundlak test:  chi2({w.df}) = {w.stat:.2f}   p-value = {w.pval:.4f}")
print(f"Union coefficient in the Mundlak model: {mundlak.params['union']:.4f}  (FE: {fe.params['union']:.4f})")
```

```
Hausman test:  chi2(3) = 37.01   p-value = 0.0000
Mundlak test:  chi2(3) = 33.38   p-value = 0.0000
Union coefficient in the Mundlak model: 0.0800  (FE: 0.0800)
```

How to read it

Random effects is more efficient than fixed effects, but only consistent if the worker effects are **uncorrelated** with union status, marriage and experience.

- **Hausman test.** It compares the FE and RE coefficients. Under the null both are consistent and differ only by sampling error. Here the gap is large and the test rejects strongly.
- **Mundlak test.** It adds each worker's average union status, marriage rate and squared experience to the RE model. If the worker effect is correlated with the regressors, those averages pick it up. The test uses clustered standard errors, so unlike the conventional Hausman test it remains valid under heteroskedasticity and serial correlation. It also rejects.
- **A useful check.** In the Mundlak model, the coefficient on union is identical to the fixed-effects estimate. This is an exact algebraic result, and it is why the Mundlak model is also called correlated random effects.

Conclusion: random effects is rejected, and fixed effects is the estimator to report.

15.7 Diagnostic tests: serial correlation and cross-sectional dependence

```
# Wooldridge (2002) test for serial correlation: regress the first-differenced residuals on their own lag.
# Under no serial correlation in the levels errors, the coefficient is -0.5.
fd = FirstDifferenceOLS(y, df[xs]).fit(cov_type="clustered", cluster_entity=True)
e = fd.resids.rename("e").to_frame()
e["e_lag"] = e.groupby(level=0)["e"].shift(1)
e = e.dropna()
aux = PooledOLS(e["e"], e[["e_lag"]]).fit(cov_type="clustered", cluster_entity=True)
F = ((aux.params["e_lag"] + 0.5) / aux.std_errors["e_lag"]) ** 2
print(f"Wooldridge test: coefficient on lagged residual = {aux.params['e_lag']:.3f}   F(1, {fe.entity_info['total'] - 1:.0f}) = {F:.2f}   p-value = {stats.f.sf(F, 1, fe.entity_info['total'] - 1):.4f}")

# residual correlation between workers, averaged over all pairs (Pesaran's CD is not designed for T = 8, so describe only)
r = fe.resids.unstack(0)
c = r.corr().to_numpy()[np.triu_indices(r.shape[1], 1)]
print(f"Average correlation of residuals between workers: {np.nanmean(c):.3f}  (mean absolute: {np.nanmean(np.abs(c)):.3f})")
```

```
Wooldridge test: coefficient on lagged residual = -0.387   F(1, 544) = 25.62   p-value = 0.0000
Average correlation of residuals between workers: 0.004  (mean absolute: 0.367)
```

How to read it

- **Serial correlation.** The Wooldridge test rejects: the errors of the same worker are correlated over time, as they almost always are in wage data. This does not bias the FE coefficients, but it invalidates conventional standard errors. **Clustering by worker already corrects for it**, which is why the clustered standard errors are the ones to report.
- **Cross-sectional dependence.** With $T = 8$, the residual correlation between any two workers rests on eight observations and is very noisy, so a mean absolute correlation of this size is what pure noise would produce. The average signed correlation is close to zero: once year effects are included, there is no common shock left. Year effects are the standard way to deal with common shocks in short micro panels.

15.8 Robustness checks

```
models = {"Pooled OLS": pooled, "Random effects": re, "Fixed effects": fe}

fd_years = FirstDifferenceOLS(y, df[xs].join(years)).fit(cov_type="clustered", cluster_entity=True)
models["First differences"] = fd_years

cre_x = sm.add_constant(df[xs + ["educ", "black", "hisp"]].join(years).join(means))
cre = RandomEffects(y, cre_x).fit(cov_type="clustered", cluster_entity=True)
models["CRE (Mundlak)"] = cre

print(f"{'':20s}{'union':>16s}{'married':>16s}")
for name, m in models.items():
    print(f"{name:20s}" + "".join(f"{m.params[v]:8.3f} ({m.std_errors[v]:.3f})" for v in ["union", "married"]))
print(f"\nReturn to a year of education in the CRE model: {cre.params['educ']:.3f} ({cre.std_errors['educ']:.3f})")
```

```
                              union         married
Pooled OLS             0.177 (0.029)   0.152 (0.027)
Random effects         0.104 (0.021)   0.078 (0.019)
Fixed effects          0.080 (0.024)   0.047 (0.022)
First differences      0.041 (0.022)   0.038 (0.024)
CRE (Mundlak)          0.080 (0.023)   0.047 (0.021)

Return to a year of education in the CRE model: 0.094 (0.011)
```

How to read it

- **First differences** is the other estimator that removes the worker effect. It compares adjacent years only, so it is more sensitive to measurement error in union status: a misreported switch in one year looks like two genuine switches. The FD premium is smaller, which is the usual sign of measurement error in a binary regressor. Report it, and explain the difference.
- **Correlated random effects** adds the worker averages of the time-varying regressors. Its union and marriage coefficients equal the fixed-effects ones, but it can also estimate the effect of time-invariant variables such as education and race. This is the answer when a referee asks for the return to education in a fixed-effects paper.
- **The ranking pooled > RE > FE** for the union premium is exactly what positive selection on unobserved ability predicts.

15.9 Build the results table for the paper

```
def star(p):
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) Pooled OLS": pooled, "(2) RE": re, "(3) FE": fe, "(4) FD": fd_years}
print(f"{'':16s}" + "".join(f"{c:>16s}" for c in cols))
for v in xs:
    print(f"{v:16s}" + "".join(f"{m.params[v]:13.4f}{star(m.pvalues[v]):3s}" for m in cols.values()))
    print(f"{'':16s}" + "".join(f"{'(' + format(m.std_errors[v], '.4f') + ')':>13s}   " for m in cols.values()))
print(f"{'Worker effects':16s}{'no':>13s}   {'random':>13s}   {'fixed':>13s}   {'differenced':>13s}")
print(f"{'Year effects':16s}" + f"{'yes':>13s}   " * 4)
print(f"{'Observations':16s}" + "".join(f"{m.nobs:>13d}   " for m in cols.values()))
print("Standard errors clustered by worker in parentheses. *** p<0.01, ** p<0.05, * p<0.10.")
```

```
                (1) Pooled OLS          (2) RE          (3) FE          (4) FD
union                 0.1768***       0.1040***       0.0800***       0.0411*
                     (0.0290)        (0.0210)        (0.0243)        (0.0219)
married               0.1521***       0.0780***       0.0467**        0.0381
                     (0.0273)        (0.0191)        (0.0224)        (0.0242)
expersq              -0.0021***      -0.0031***      -0.0052***      -0.0058***
                     (0.0005)        (0.0005)        (0.0009)        (0.0009)
Worker effects             no          random           fixed     differenced
Year effects              yes             yes             yes             yes
Observations            4165            4165            4165            4165
Standard errors clustered by worker in parentheses. *** p<0.01, ** p<0.05, * p<0.10.
```

```
Year effects              yes             yes             yes             yes
Observations             4360            4360            4360            3815
Standard errors clustered by worker in parentheses. *** p<0.01, ** p<0.05, * p<0.10.
```

How to read it

A panel results table shows the main estimator next to the benchmarks, so the reader can see how much the unit effects matter. Report under the table: the number of workers and years, that standard errors are clustered by worker, that all columns include year effects (except where they are differenced away), and the F, Hausman and Mundlak statistics. Column (3) is the preferred estimate.

15.10 Write the paper

How to write it: specification tests

An F test rejects the hypothesis that the worker effects are jointly zero, so pooled OLS is inappropriate. A Hausman test rejects the consistency of random effects, and a cluster-robust Mundlak test, which remains valid under heteroskedasticity and serial correlation, leads to the same conclusion. We therefore estimate the model by fixed effects with worker and year effects, and cluster standard errors by worker; a Wooldridge test confirms serial correlation within workers.

How to write it: results

Table 1 reports the estimates. Pooled OLS puts the union wage premium at about 18 percent (column 1). Once time-invariant worker characteristics are controlled for, the premium falls to about 8 percent (column 3), and it remains significant at the 1 percent level. The decline is consistent with workers of higher unobserved ability being less likely to hold union jobs. The first-difference estimate is smaller (column 4), as expected if union status is measured with error.

Common mistakes

- Reporting unclustered standard errors. In a panel they are almost always too small.
- Including a regressor that moves one-for-one with time (age, experience) together with year effects. It is collinear and either drops out or gives a meaningless coefficient.
- Claiming the FE coefficient on a variable that never changes. It is not identified; use correlated random effects instead.
- Treating fixed effects as a cure for all endogeneity. It removes time-invariant confounders only. If union status responds to wage shocks in the same year, FE is still biased, and you need an instrument or a dynamic model.
- Using only the conventional Hausman test. Report the robust Mundlak version as well.

CHAPTER 16

Random Effects

Applies when. Unit effects present, uncorrelated with the regressors

Overview. Use this chapter when the panel is short, the units differ systematically, but the Hausman and Mundlak tests do not reject that those differences are uncorrelated with the regressors. Random effects then uses both the within and the between variation, keeps time-invariant regressors, and gives more precise estimates than fixed effects.

Data. 532 U.S. men, annual hours and wages, 1979–1988 (PSID, Ziliak 1997)

Software. Python · linearmodels · statsmodels

16.1 The question and the model

The question is a central one in labour economics and in macroeconomic models of the labour market: **how much do men change their working hours when their wage changes?** The answer, the labour-supply elasticity, governs how taxes affect work and how much hours move over the business cycle.

$$\ln h_{it} = \beta_1 \ln w_{it} + \beta_2\, kids_{it} + \beta_3\, age_{it} + \beta_4\, disab_{it} + \lambda_t + u_i + \varepsilon_{it}$$

Symbol	Variable in the file	Definition	Expected sign
$\ln h_{it}$	`lnhr`	Log annual hours worked	dependent variable
$\ln w_{it}$	`lnwg`	Log hourly wage	positive and small: the labour-supply elasticity
$kids_{it}$	`kids`	Number of children	positive: more needs to provide for
age_{it}	`age`	Age in years	negative over the life cycle
$disab_{it}$	`disab`	1 if a health limitation is reported	negative
u_i	—	Worker effect: tastes for work, occupation	treated as random

In the random-effects model the worker effect u_i is part of the error, with variance σ_u^2. The key assumption is that u_i is **uncorrelated with every regressor in every year**. The tests below check it.

Before you estimate anything

Random effects is not a weaker version of fixed effects. If its assumption holds, it is consistent and more efficient, and it can estimate the effect of variables that do not change over time. If the assumption fails, it is biased. So the choice is made by a test, and the paper must report it.

How to write it: model section

We model log annual hours as $\ln h_{it} = \beta_1 \ln w_{it} + \beta_2 kids_{it} + \beta_3 age_{it} + \beta_4 disab_{it} + \lambda_t + u_i + \varepsilon_{it}$*, where* λ_t *are year effects and* u_i *is a worker-specific effect capturing persistent differences in tastes for work. The coefficient* β_1 *is the elasticity of hours with respect to the wage.*

16.2 Load the panel and look at the variation

```
import pandas as pd
import numpy as np
```

```
import statsmodels.api as sm
from scipy import stats
from linearmodels.panel import PanelOLS, PooledOLS, RandomEffects, compare

df = pd.read_csv("laborsupply.csv").set_index(["id", "year"])
print(f"{df.index.get_level_values(0).nunique()} men x {df.index.get_level_values(1).nunique()} years = {len(
  ↪ df)} rows")

xs = ["lnwg", "kids", "age", "disab"]
for v in ["lnhr"] + xs:
    x = df[v]
    within = x - x.groupby(level=0).transform("mean")
    print(f"{v:6s} mean = {x.mean():7.3f}   between sd = {x.groupby(level=0).mean().std():6.3f}   within sd = {
  ↪ within.std():6.3f}")
```

```
532 men x 10 years = 5320 rows
lnhr   mean =   7.657   between sd =  0.179   within sd =  0.223
lnwg   mean =   2.609   between sd =  0.391   within sd =  0.169
kids   mean =   1.556   between sd =  1.032   within sd =  0.605
age    mean =  38.918   between sd =  7.945   within sd =  2.896
disab  mean =   0.061   between sd =  0.166   within sd =  0.173
```

How to read it

A balanced, short panel: 532 men, each observed for 10 years. The decision rules of Chapter 1, applied to this file, give recommended random effects; this chapter repeats them in full.

Look at the wage. Its between standard deviation (0.39) is more than twice its within standard deviation (0.17): **most of the wage variation is across men, not over time for the same man.** Fixed effects throws the between variation away. Random effects uses it, which is why it will be much more precise here, provided its assumption holds.

16.3 Estimate the random-effects model

```
years = pd.get_dummies(df.index.get_level_values(1), prefix="y", drop_first=True, dtype=float).set_index(df.
  ↪ index)
y = df["lnhr"]
X = sm.add_constant(df[xs].join(years))

re = RandomEffects(y, X).fit(cov_type="clustered", cluster_entity=True)
print(re.params[xs].round(4).to_frame("coef").join(re.std_errors[xs].round(4).rename("std err")).join(re.
  ↪ pvalues[xs].round(4).rename("p-value")))

vd = re.variance_decomposition
print(f"\nsigma_u^2 (worker effect) = {vd['Effects']:.4f}   sigma_e^2 (idiosyncratic) = {vd['Residual']:.4f}
  ↪  rho = {vd['Percent due to Effects']:.3f}")
print(f"theta (share of the worker mean removed): {re.theta.iloc[0, 0]:.3f}")
```

```
         coef  std err  p-value
lnwg   0.1197   0.0518   0.0210
kids   0.0082   0.0058   0.1542
age   -0.0018   0.0011   0.1072
disab -0.0770   0.0290   0.0080

sigma_u^2 (worker effect) = 0.0255   sigma_e^2 (idiosyncratic) = 0.0537   rho = 0.322
theta (share of the worker mean removed): 0.583
```

How to read it

- **Wage elasticity.** A 10% higher wage is associated with about 1.2% more hours, significant at the 5% level. This is small, as the literature on prime-age men consistently finds.
- **Children** raise hours slightly; **a health limitation** lowers them by about 7%.
- **Variance components.** About a third of the unexplained variation in hours is a permanent worker effect (ρ); the rest changes from year to year.

- θ. Random effects subtracts a fraction θ of each man's average from his data. With $\theta = 0$ it would be pooled OLS, with $\theta = 1$ fixed effects. Here θ is about 0.6, so RE sits between the two: it is a weighted average of the within and between estimators.

16.4 Test 1: are there worker effects at all? Breusch–Pagan LM

```
pooled = PooledOLS(y, X).fit(cov_type="clustered", cluster_entity=True)
e = pooled.resids
n, T_i = len(e), e.groupby(level=0).size()
lm = n ** 2 / (2 * (T_i * (T_i - 1)).sum()) * ((e.groupby(level=0).sum() ** 2).sum() / (e ** 2).sum() - 1) **
  ↪ 2
print(f"Breusch-Pagan LM test: chi2(1) = {lm:.1f}   p-value = {stats.chi2.sf(lm, 1):.4f}")
```

```
Breusch-Pagan LM test: chi2(1) = 2440.8   p-value = 0.0000
```

How to read it

The null hypothesis is $\sigma_u^2 = 0$: no worker effect, in which case pooled OLS would do. The statistic is very large, so **the worker effects are real**. Pooled OLS with clustered standard errors would still be consistent here, but random effects is more efficient, and the clustered pooled standard errors would hide this structure.

16.5 Test 2: is the random-effects assumption credible? Hausman and Mundlak

```
fe = PanelOLS(y, sm.add_constant(df[xs].join(years)), entity_effects=True, drop_absorbed=True).fit(cov_type="
  ↪ clustered", cluster_entity=True)
common = [v for v in xs if v in fe.params.index]
print("Coefficients compared:", common)

fe_p = PanelOLS(y, sm.add_constant(df[xs].join(years)), entity_effects=True, drop_absorbed=True).fit()
re_p = RandomEffects(y, X).fit()
d = fe_p.params[common] - re_p.params[common]
H = float(d @ np.linalg.inv(fe_p.cov.loc[common, common] - re_p.cov.loc[common, common]) @ d)
print(f"Hausman test:  chi2({len(common)}) = {H:.2f}   p-value = {stats.chi2.sf(H, len(common)):.4f}")

means = df[xs].groupby(level=0).transform("mean").add_suffix("_mean")
mundlak = RandomEffects(y, X.join(means)).fit(cov_type="clustered", cluster_entity=True)
w = mundlak.wald_test(formula=" = ".join(means.columns) + " = 0")
print(f"Mundlak test:  chi2({w.df}) = {w.stat:.2f}   p-value = {w.pval:.4f}")
print(mundlak.params[means.columns].round(4).to_string())
```

```
Coefficients compared: ['lnwg', 'kids', 'age', 'disab']
Hausman test:  chi2(4) = 18.46   p-value = 0.0010
Mundlak test:  chi2(4) = 6.52   p-value = 0.1635
lnwg_mean    -0.1020
kids_mean    -0.0006
age_mean      0.0170
disab_mean   -0.0681
```

How to read it

The two tests disagree, and the reason matters for every panel paper.

- **Conventional Hausman test.** It compares the FE and RE coefficients using the textbook variance formula, which assumes that RE is fully efficient: homoskedastic errors and no correlation over time other than the worker effect. It rejects (p = 0.001).
- **Mundlak test.** It adds each man's average wage, number of children, age and disability to the RE model. If tastes for work were correlated with these averages, their coefficients would be significant. Because it uses clustered standard errors, it stays valid when the errors are heteroskedastic or serially correlated. It does **not** reject (p = 0.16).

Hours data are strongly heteroskedastic: a few men have very low hours in some years (the minimum of log hours is 2.8, about 16 hours a year). Under heteroskedasticity the conventional variance formula is wrong, so the conventional test cannot be trusted. **The robust test is the one to base the decision on, and it supports random effects**, which, as the more efficient estimator, is the one to report.

Two cautions. Failing to reject is not proof that the assumption holds. And because the two tests disagree, report both, and report fixed effects alongside as the robustness check.

16.6 Compare with fixed effects

```
for name, m in (("Random effects", re), ("Fixed effects", fe)):
    print(f"{name:16s} lnwg = {m.params['lnwg']:.4f} ({m.std_errors['lnwg']:.4f})   kids = {m.params['kids
  ↪ ']:.4f} ({m.std_errors['kids']:.4f})   disab = {m.params['disab']:.4f} ({m.std_errors['disab']:.4f})")
print(f"\nStandard error of the wage elasticity is {fe.std_errors['lnwg'] / re.std_errors['lnwg']:.1f} times
  ↪ larger with fixed effects")
```

```
Random effects   lnwg = 0.1197 (0.0518)   kids = 0.0082 (0.0058)   disab = -0.0770 (0.0290)
Fixed effects    lnwg = 0.1660 (0.0860)   kids = 0.0076 (0.0076)   disab = -0.0710 (0.0328)

Standard error of the wage elasticity is 1.7 times larger with fixed effects
```

How to read it

The point estimates are close: the FE wage elasticity is somewhat larger, but the gap is well inside the confidence intervals. The main difference is **precision**: fixed effects uses only the small within variation in wages, so its standard error is larger. This is the efficiency gain that random effects buys when its assumption holds.

16.7 Diagnostic test: serial correlation

```
from linearmodels.panel import FirstDifferenceOLS
fd = FirstDifferenceOLS(y, df[["lnwg", "kids", "disab"]]).fit(cov_type="clustered", cluster_entity=True)
r = fd.resids.rename("e").to_frame()
r["e_lag"] = r.groupby(level=0)["e"].shift(1)
r = r.dropna()
aux = PooledOLS(r["e"], r[["e_lag"]]).fit(cov_type="clustered", cluster_entity=True)
F = ((aux.params["e_lag"] + 0.5) / aux.std_errors["e_lag"]) ** 2
N = df.index.get_level_values(0).nunique()
print(f"Wooldridge test: coefficient = {aux.params['e_lag']:.3f} (-0.5 under H0)   F(1, {N - 1}) = {F:.2f}   p-
  ↪ value = {stats.f.sf(F, 1, N - 1):.4f}")
```

```
Wooldridge test: coefficient = -0.407 (-0.5 under H0)   F(1, 531) = 2.61   p-value = 0.1066
```

How to read it

The random-effects model assumes that, apart from the permanent worker effect, the errors are uncorrelated over time. The Wooldridge test checks this: if the errors in levels are serially uncorrelated, the first-differenced errors have a correlation of exactly –0.5. Here the estimate is –0.41 and the test does not reject at 5% ($p = 0.11$), so **there is no strong evidence of serial correlation beyond the worker effect**, which supports the RE error structure. Had it rejected, the RE point estimates would remain consistent but the textbook standard errors would be wrong. **The clustered standard errors used throughout this chapter are valid either way**, which is why they are the ones to report.

16.8 Build the results table for the paper

```
def star(p):
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) Pooled OLS": pooled, "(2) RE": re, "(3) FE": fe, "(4) Mundlak": mundlak}
print(f"{'':12s}" + "".join(f"{c:>16s}" for c in cols))
for v in xs:
    print(f"{v:12s}" + "".join(f"{m.params[v]:13.4f}{star(m.pvalues[v]):3s}" if v in m.params else f"{'':>13s}
  ↪    " for m in cols.values()))
```

```
    print(f"{'':12s}" + "".join(f"{'(' + format(m.std_errors[v], '.4f') + ')':>13s}   " if v in m.params else
  ↪ "  " * 16 for m in cols.values()))
print(f"{'Observations':12s}" + "".join(f"{m.nobs:>13d}   " for m in cols.values()))
print("All columns include year effects. Standard errors clustered by worker. *** p<0.01, ** p<0.05, * p<0.10.
  ↪ ")
print(f"Breusch-Pagan LM = {lm:.1f} (p < 0.001); Hausman chi2({len(common)}) = {H:.2f} (p = {stats.chi2.sf(H,
  ↪ len(common)):.3f}); Mundlak chi2({w.df}) = {w.stat:.2f} (p = {w.pval:.3f})")
```

```
             (1) Pooled OLS          (2) RE          (3) FE     (4) Mundlak
lnwg               0.0828***       0.1197**        0.1660*        0.1660*
                  (0.0297)        (0.0518)        (0.0860)       (0.0861)
kids               0.0079          0.0082          0.0076         0.0076
                  (0.0060)        (0.0058)        (0.0076)       (0.0076)
age               -0.0013         -0.0018         -0.0180*       -0.0180*
                  (0.0010)        (0.0011)        (0.0104)       (0.0104)
disab             -0.0997***      -0.0770***      -0.0710**      -0.0710**
                  (0.0289)        (0.0290)        (0.0328)       (0.0328)
Observations         5320            5320            5320           5320
All columns include year effects. Standard errors clustered by worker. *** p<0.01, ** p<0.05, * p<0.10.
Breusch-Pagan LM = 2440.8 (p < 0.001); Hausman chi2(4) = 18.46 (p = 0.001); Mundlak chi2(4) = 6.52 (p = 0.163)
```

How to read it

Column (2) is the preferred estimate. Columns (1) and (3) show that the conclusion does not depend on how the worker effects are treated, and column (4) shows the Mundlak model, whose coefficients on the worker averages are the basis of the test. The three test statistics belong in the table notes.

16.9 Write the paper

How to write it: specification tests

A Breusch–Pagan LM test strongly rejects the absence of worker-specific effects, so pooled OLS is inefficient. The conventional Hausman test rejects random effects, but it assumes homoskedastic errors, which is implausible for hours data; its cluster-robust Mundlak version does not reject the hypothesis that the worker effects are uncorrelated with the regressors ($p = 0.16$). We therefore estimate the model by random effects, with year effects and standard errors clustered by worker, and report fixed-effects estimates as a robustness check.

How to write it: results

The estimated elasticity of hours with respect to the wage is about 0.12 and significant at the 5 percent level (Table 1, column 2). A 10 percent rise in the hourly wage is thus associated with a 1.2 percent increase in annual hours, in line with the small elasticities typically found for prime-age men. A reported health limitation reduces hours by about 7 percent. The fixed-effects estimate of the wage elasticity is somewhat larger, 0.17, but much less precise (column 3).

Common mistakes

- Choosing random effects because fixed effects "loses too many variables". That is a reason to use the Mundlak model, not to assume the RE condition holds.
- Reporting only the conventional Hausman test. With heteroskedastic or serially correlated errors it is invalid, as this example shows, and it can even be negative. Report the Mundlak test too.
- Reading "do not reject" as "random effects is proven". Show that the FE estimates tell the same story.
- Interpreting the wage elasticity causally. Wages may respond to hours (overtime pay, measurement error in hours, which also enters the wage as earnings divided by hours). The serious literature instruments the wage.

CHAPTER 17

Difference GMM for Dynamic Panels

Applies when. Dynamic panel, large N, short T

Overview. Use this chapter when the model has a lagged dependent variable and the panel has many units but few periods. Pooled OLS and fixed effects are then both biased, in opposite directions. The chapter estimates the Arellano–Bond difference GMM estimator on the data it was developed for, runs the three specification tests every referee asks for, and checks the sensitivity to the instrument set.

Data. 140 UK manufacturing firms, annual, 1976–1984 (Arellano and Bond 1991)

Software. Python · linearmodels · statsmodels

17.1 The question and why the usual estimators fail

The question: **how persistent is firm employment, and how does it respond to wages and capital?** Employment adjusts slowly because hiring and firing are costly, so this year's employment depends on last year's. That makes the model dynamic:

$$n_{it} = \alpha_1 n_{i,t-1} + \alpha_2 n_{i,t-2} + \beta_0 w_{it} + \beta_1 w_{i,t-1} + \gamma' k_{it} + \delta' ys_{it} + \lambda_t + \eta_i + \varepsilon_{it}$$

with all variables in logs: employment n, the real wage w, capital k (current and two lags) and industry output ys (current and two lags). η_i is a firm fixed effect.

Estimator	Problem with a lagged dependent variable	Bias in α_1
Pooled OLS	$n_{i,t-1}$ contains η_i, which is in the error	**upwards**
Fixed effects	demeaning makes $n_{i,t-1}$ correlated with the demeaned error (Nickell bias, of order 1/T)	**downwards**
Difference GMM	first-difference to remove η_i, then instrument $\Delta n_{i,t-1}$ with $n_{i,t-2}, n_{i,t-3}, \ldots$	consistent as N grows, if the instruments are valid

When to use this chapter, and when not

GMM for dynamic panels is designed for **many units and few periods**. Its consistency comes from N growing, not T. With 20 countries over 40 years the Nickell bias is small, the instruments multiply out of control, and fixed effects (or Part III) are the better choice. The decision rules of Chapter 1 send a panel here only when T is short (below 15), N is large (at least 30 units) and you say the model is dynamic. With fewer units it recommends bias-corrected fixed effects (LSDVC) instead.

How to write it: model section

We estimate a dynamic labour-demand equation in which log employment depends on its own two lags, the current and lagged log real wage, and current and lagged log capital and industry output, together with year effects λ_t and firm effects η_i. Because the lagged dependent variable is correlated with the firm effect, pooled OLS is biased upwards, and because T is short, the within estimator is subject to the Nickell bias. We therefore use the difference GMM estimator of Arellano and Bond (1991).

17.2 Load the panel and build the lags

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
from diff_gmm import diff_gmm          # helper file: panel-data-guide/examples/diff_gmm.py

panel = pd.read_csv("employment.csv").rename(columns={"log_emp": "n", "log_wage": "w", "log_capital": "k", "
  ↪ log_output": "ys"})
panel = panel.sort_values(["firm", "year"])
by_firm = panel.groupby("firm")
for var, lag in (("n", 1), ("n", 2), ("w", 1), ("k", 1), ("k", 2), ("ys", 1), ("ys", 2)):
    panel[f"{var}_l{lag}"] = by_firm[var].shift(lag)

print(f"{panel['firm'].nunique()} firms, {panel['year'].min()}-{panel['year'].max()}, {len(panel)}
  ↪ observations")
print("Years per firm:", by_firm["year"].count().value_counts().sort_index().to_dict())
```

```
140 firms, 1976-1984, 1031 observations
Years per firm: {7: 103, 8: 23, 9: 14}
```

How to read it

This is the shape GMM needs: **140 firms and 7 to 9 years each**. The decision rules of Chapter 1 applied to this file (fixed effects is preferred to random effects) and then asked whether the model is dynamic. Because employment adjusts gradually, the answer is yes, which leads here. Download the helper file and the data.

17.3 Section 17.1: pooled OLS and fixed effects give the bounds

```
controls = ["w", "w_l1", "k", "k_l1", "k_l2", "ys", "ys_l1", "ys_l2"]
formula = "n ~ n_l1 + n_l2 + " + " + ".join(controls) + " + C(year)"
sample = panel.dropna()
cluster = {"cov_type": "cluster", "cov_kwds": {"groups": sample["firm"]}}

pooled = smf.ols(formula, sample).fit(**cluster)
fixed = smf.ols(formula + " + C(firm)", sample).fit(**cluster)
print(f"{'':16s}{'L1.n':>8s}{'L2.n':>8s}{'w':>8s}{'k':>8s}")
for name, m in (("Pooled OLS", pooled), ("Fixed effects", fixed)):
    print(f"{name:16s}{m.params['n_l1']:8.3f}{m.params['n_l2']:8.3f}{m.params['w']:8.3f}{m.params['k']:8.3f}")
```

```
                  L1.n    L2.n       w       k
Pooled OLS       1.045  -0.077  -0.524   0.343
Fixed effects    0.733  -0.139  -0.560   0.388
```

How to read it

The coefficient on lagged employment is above 1 by pooled OLS and about 0.73 by fixed effects. Because the first is biased upwards and the second downwards, **a credible GMM estimate should lie between them, or close to the fixed-effects bound**. This check (Bond 2002) costs two regressions and is the first thing to run in any dynamic panel study.

17.4 Section 17.2: difference GMM

```
gmm = diff_gmm(panel, unit="firm", time="year", y="n", lags_y=2, x=controls, gmm_lags=(2, None))
print(gmm["table"].loc[["L1.n", "L2.n"] + controls].round(4))
print(f"\nObservations: {gmm['nobs']}   firms: {gmm['units']}   instruments: {gmm['instruments']}")

t = gmm["table"]["coef"]
persistence = 1 - t["L1.n"] - t["L2.n"]
print(f"Long-run wage elasticity:    {(t['w'] + t['w_l1']) / persistence:.2f}")
print(f"Long-run capital elasticity: {(t['k'] + t['k_l1'] + t['k_l2']) / persistence:.2f}")
```

```
         coef  std err       z  p-value
L1.n   0.6862   0.1446  4.7459   0.0000
L2.n  -0.0854   0.0560 -1.5238   0.1276
w     -0.6078   0.1782 -3.4108   0.0006
```

```
w_l1   0.3926   0.1680  2.3371   0.0194
k      0.3568   0.0590  6.0461   0.0000
k_l1  -0.0580   0.0732 -0.7926   0.4280
k_l2  -0.0199   0.0327 -0.6098   0.5420
ys     0.6085   0.1725  3.5269   0.0004
ys_l1 -0.7112   0.2317 -3.0691   0.0021
ys_l2  0.1058   0.1412  0.7493   0.4537

Observations: 611   firms: 140   instruments: 41
Long-run wage elasticity:    -0.54
Long-run capital elasticity: 0.70
```

How to read it

These are one-step estimates with standard errors robust to heteroskedasticity and to correlation within each firm. They reproduce the published Arellano–Bond results for this dataset (0.686 on lagged employment), which is how the helper function was checked.

- **Persistence.** About 0.69 on the first lag. Employment adjusts gradually.
- **Wages.** A 1% rise in the real wage lowers employment by about 0.6% in the same year; the long-run elasticity, which allows for the dynamics, is about –0.5.
- **Capital.** A 1% rise in capital raises employment by about 0.36% at once, and by about 0.7% in the long run.
- **The bounds.** The GMM persistence lies slightly **below** the fixed-effects estimate. That is a warning sign of weak instruments in the differenced equation: when employment is highly persistent, its lagged levels predict its differences poorly. Section 17.4 returns to this.

17.5 Section 17.3: the three specification tests

```
z1, p1 = gmm["ar1"]; z2, p2 = gmm["ar2"]; j, dof, pj = gmm["hansen"]
print(f"Arellano-Bond AR(1) test:  z = {z1:6.2f}   p-value = {p1:.3f}")
print(f"Arellano-Bond AR(2) test:  z = {z2:6.2f}   p-value = {p2:.3f}")
print(f"Hansen J test:             chi2({dof}) = {j:.2f}   p-value = {pj:.3f}")
print(f"Instruments: {gmm['instruments']}    firms: {gmm['units']}")
```

```
Arellano-Bond AR(1) test:  z =  -3.60   p-value = 0.000
Arellano-Bond AR(2) test:  z =  -0.52   p-value = 0.606
Hansen J test:             chi2(25) = 31.38   p-value = 0.177
Instruments: 41    firms: 140
```

How to read it

A dynamic panel GMM table is not complete without these.

- **AR(1)** in the differenced residuals is expected: differencing creates it. It says nothing about validity.
- **AR(2) is the test that matters.** If the errors in levels are serially uncorrelated, the differenced errors have no second-order correlation. Here it is not rejected, so levels dated $t-2$ are valid instruments. If it were rejected, the instruments would have to start at $t-3$.
- **Hansen J.** The null is that all instruments are valid. It is not rejected. Be suspicious of p-values very close to 1: that is a symptom of too many instruments, not of very good ones.
- **Instrument count.** The count must stay well below the number of firms. Too many instruments overfit the endogenous variable, pull GMM towards the biased OLS estimate and weaken the Hansen test.

17.6 Section 17.4: robustness to the instrument set

```
specs = {"All lags (baseline)": {}, "Lags 2 to 4": {"gmm_lags": (2, 4)}, "Lags 2 to 3": {"gmm_lags": (2, 3)},
         "Lags 3 and deeper": {"gmm_lags": (3, None)}, "Collapsed instruments": {"collapse": True}}
print(f"{'':24s}{'L1.n':>7s}{'(s.e.)':>9s}{'w':>8s}{'k':>8s}{'instr.':>8s}{'AR(2) p':>9s}{'Hansen p':>10s}")
results = {}
for name, options in specs.items():
    r = results[name] = diff_gmm(panel, unit="firm", time="year", y="n", lags_y=2, x=controls, **options)
    c, se = r["table"]["coef"], r["table"]["std err"]
```

```
    print(f"{name:24s}{c['L1.n']:7.3f}{'(' + format(se['L1.n'], '.3f') + ')':>9s}{c['w']:8.3f}{c['k']:8.3f}{r
  ↪ ['instruments']:8d}{r['ar2'][1]:9.3f}{r['hansen'][2]:10.3f}")
print(f"\nBounds for L1.n:  fixed effects {fixed.params['n_l1']:.3f}   pooled OLS {pooled.params['n_l1']:.3f}")
  ↪
```

```
                          L1.n   (s.e.)       w       k  instr.  AR(2) p  Hansen p
All lags (baseline)       0.686  (0.145)  -0.608   0.357      41    0.606     0.177
Lags 2 to 4               0.475  (0.253)  -0.610   0.364      31    0.665     0.181
Lags 2 to 3               0.392  (0.265)  -0.600   0.370      26    0.688     0.078
Lags 3 and deeper         0.986  (0.191)  -0.683   0.317      35    0.068     0.655
Collapsed instruments     1.358  (0.365)  -0.710   0.311      21    0.405     0.289

Bounds for L1.n:  fixed effects 0.733   pooled OLS 1.045
```

How to read it

Referees always ask whether the results depend on the instruments. Report a table like this one.

- **The wage and capital effects are stable** across all instrument sets: they are the robust findings.
- **The persistence coefficient is not.** It moves with the lag depth, and with collapsed instruments its standard error grows. Persistence is the parameter for which difference GMM is known to be weak when the series are highly persistent.
- **What to do.** Report the baseline, show this table, and estimate **system GMM** (Blundell and Bond 1998), which adds the equations in levels instrumented by lagged differences and is much stronger when persistence is high. In Stata: `xtabond2 n L(1/2).n w L.w k L(1/2).k ys L(1/2).ys i.year, gmm(L.n, collapse) iv(w L.w k L(1/2).k ys L(1/2).ys i.year) twostep robust`.

17.7 Build the results table for the paper

```
def star(p):
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

base, col = results["All lags (baseline)"], results["Collapsed instruments"]
names = ["L1.n", "L2.n", "w", "w_l1", "k"]
fmap = {"L1.n": "n_l1", "L2.n": "n_l2"}
print(f"{'':10s}{'(1) Pooled OLS':>16s}{'(2) FE':>16s}{'(3) Diff. GMM':>16s}{'(4) GMM collapsed':>19s}")
for v in names:
    o = fmap.get(v, v)
    cells = [(pooled.params[o], pooled.bse[o], pooled.pvalues[o]), (fixed.params[o], fixed.bse[o], fixed.
  ↪ pvalues[o]),
             tuple(base["table"].loc[v, ["coef", "std err", "p-value"]]), tuple(col["table"].loc[v, ["coef", "
  ↪ std err", "p-value"]])]
    print(f"{v:10s}" + "".join(f"{b:13.3f}{star(p):3s}" for b, s, p in cells))
    print(f"{'':10s}" + "".join(f"{'(' + format(s, '.3f') + ')':>13s}   " for b, s, p in cells))
print(f"{'AR(2) p':10s}{'':32s}{base['ar2'][1]:13.3f}   {col['ar2'][1]:13.3f}")
print(f"{'Hansen p':10s}{'':32s}{base['hansen'][2]:13.3f}   {col['hansen'][2]:13.3f}")
print(f"{'Instruments':10s}{'':31s}{base['instruments']:13d}   {col['instruments']:13d}")
print("Year effects in all columns. Standard errors robust to within-firm correlation. *** p<0.01, ** p<0.05, *
  ↪  p<0.10.")
```

```
            (1) Pooled OLS          (2) FE   (3) Diff. GMM  (4) GMM collapsed
L1.n              1.045***        0.733***        0.686***        1.358***
                 (0.052)         (0.066)         (0.145)         (0.365)
L2.n              -0.077          -0.139          -0.085          -0.144**
                 (0.049)         (0.087)         (0.056)         (0.062)
w                 -0.524***       -0.560***       -0.608***       -0.710***
                 (0.174)         (0.177)         (0.178)         (0.217)
w_l1               0.477***        0.315**         0.393**         0.846**
                 (0.172)         (0.159)         (0.168)         (0.399)
k                  0.343***        0.388***        0.357***        0.311***
                 (0.049)         (0.063)         (0.059)         (0.071)
AR(2) p                                           0.606           0.405
Hansen p                                          0.177           0.289
Instruments                                          41              21
Year effects in all columns. Standard errors robust to within-firm correlation. *** p<0.01, ** p<0.05, * p<0.10.
```

How to read it

Columns (1) and (2) are the bounds, columns (3) and (4) the GMM estimates. Always report the AR(2) and Hansen p-values and the instrument count in the table itself, not only in the text.

17.8 Write the paper

How to write it: estimation strategy

Because the model includes lagged employment and firm fixed effects, and T is short, both pooled OLS and the within estimator are inconsistent, with biases of opposite sign. We use the Arellano–Bond difference GMM estimator, instrumenting the differenced lags of employment with levels dated $t-2$ and earlier and treating wages, capital and output as strictly exogenous. We report one-step estimates with standard errors robust to heteroskedasticity and within-firm correlation.

How to write it: results

The Arellano–Bond test does not detect second-order serial correlation in the differenced residuals, and the Hansen test does not reject the validity of the instruments (Table 1, column 3). A 1 percent increase in the real wage reduces employment by about 0.6 percent on impact and by about 0.5 percent in the long run. The wage and capital elasticities are robust to the choice of instrument lags and to collapsing the instrument matrix, whereas the estimated persistence of employment is sensitive to the instrument set, consistent with weak instruments for highly persistent series.

Common mistakes

- Using GMM with few units (say, 20 countries). The asymptotics need large N, and the instrument count quickly exceeds the number of units.
- Not reporting the instrument count, or using all available lags without collapsing in longer panels.
- Treating a Hansen p-value of 0.99 as reassuring. It usually means too many instruments.
- Ignoring the bounds check. A GMM persistence estimate above pooled OLS or well below fixed effects signals a problem.
- Treating endogenous regressors as exogenous. If wages respond to employment shocks, they must be instrumented with their own lags, just like the lagged dependent variable.

CHAPTER 18

Fixed Effects with Driscoll–Kraay Standard Errors

Applies when. Long panel, stationary variables, homogeneous slopes

Overview. Use this chapter when the panel has a long time dimension, every variable is stationary, and the slope-homogeneity test does not reject. A pooled fixed-effects regression is then valid, but countries hit by the same shocks are not independent, so the chapter shows how Driscoll–Kraay standard errors fix the inference, and how to check whether the common shocks also bias the coefficients.

Data. 14 OECD economies, annual, 1967–1990 (Garrett 1998)

Software. Python · linearmodels · statsmodels

18.1 The question and the model

The question: **in the OECD economies of 1967–1990, how was output growth related to unemployment and to changes in trade openness?** Macro panels like this one, a few dozen countries over a few decades, are typical of empirical papers in international and environmental economics. Their defining feature is that the countries share the same shocks: the oil crises of 1973 and 1979 and the recessions that followed hit all of them at once.

$$g_{it} = \alpha_i + \beta_1 u_{it} + \beta_2 \Delta open_{it} + \varepsilon_{it}$$

Symbol	Variable in the file	Definition	Expected sign
g_{it}	`gdp_growth`	Real GDP growth, %	dependent variable
u_{it}	`unem`	Unemployment rate, %	negative: slack and low growth go together (Okun's law)
$\Delta open_{it}$	`d_trade`	Change in trade (exports + imports) as % of GDP	positive if openness raises growth
α_i	—	Country fixed effect	permanent differences in growth

Before you estimate anything

In a macro panel, three properties decide the estimator, and each needs a test: **cross-sectional dependence** (do the countries share shocks?), **stationarity** (do the variables have unit roots?) and **slope homogeneity** (is β the same in every country?). The next three steps run them. Fixed effects with Driscoll–Kraay errors is the answer when the variables are stationary and the slopes are homogeneous.

How to write it: model section

We estimate $g_{it} = \alpha_i + \beta_1 u_{it} + \beta_2 \Delta open_{it} + \varepsilon_{it}$ for 14 OECD countries over 1967–1990, where g_{it} is real GDP growth, u_{it} the unemployment rate and $\Delta open_{it}$ the change in trade openness, and α_i are country fixed effects.

18.2 Load the panel and test for cross-sectional dependence

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

```
from linearmodels.panel import PanelOLS
from panel_tests import cd_test, cips_test, fisher_adf_test, slope_test   # examples/panel_tests.py

raw = pd.read_csv("garrett.csv").sort_values(["country", "year"])
raw["d_trade"] = raw.groupby("country")["trade"].diff()
long = raw.dropna(subset=["d_trade"])
V = ["gdp_growth", "unem", "d_trade"]
print(f"{long['country'].nunique()} countries, {long['year'].min()}-{long['year'].max()}, {len(long)}
  ↪ observations")

print(f"\n{'':12s}{'CD':>8s}{'p-value':>10s}{'mean |rho|':>12s}")
for v in V:
    cd, p, r = cd_test(long.pivot(index="year", columns="country", values=v))
    print(f"{v:12s}{cd:8.2f}{p:10.4f}{r:12.3f}")
```

```
14 countries, 1967-1990, 336 observations

                 CD   p-value  mean |rho|
gdp_growth    21.17    0.0000       0.455
unem          33.08    0.0000       0.708
d_trade       29.44    0.0000       0.630
```

How to read it

A balanced panel of 14 countries over 24 years. The Pesaran CD test rejects cross-sectional independence for every variable, decisively. The average absolute correlation between two countries' growth rates is about 0.45: **the OECD economies move together**. Two consequences follow. Unit-root tests must allow for this (CIPS rather than Fisher-ADF), and standard errors that treat countries as independent will be too small.

18.3 Unit-root tests and slope homogeneity

```
print(f"{'':12s}{'CIPS':>8s}{'p-value':>10s}{'Fisher Z':>10s}{'p-value':>10s}")
for v in V:
    W = long.pivot(index="year", columns="country", values=v)
    c, f = cips_test(W), fisher_adf_test(W)
    print(f"{v:12s}{c['stat']:8.2f}{c['p']:10.4f}{f['Z']:10.2f}{f['Z_p']:10.4f}")

s = slope_test(long, "country", "gdp_growth", ["unem", "d_trade"])
print(f"\nPesaran-Yamagata: delta = {s['delta']:.2f} (p = {s['delta_p']:.3f})   adjusted delta = {s['delta_adj
  ↪ ']:.2f} (p = {s['delta_adj_p']:.3f})")
```

```
                CIPS   p-value  Fisher Z   p-value
gdp_growth     -2.97    0.0005     -8.38    0.0000
unem           -2.53    0.0079      0.05    0.5197
d_trade        -3.19    0.0005    -13.66    0.0000

Pesaran-Yamagata: delta = 0.91 (p = 0.182)   adjusted delta = 0.99 (p = 0.162)
```

How to read it

- **Stationarity.** Because the variables are cross-sectionally dependent, CIPS is the test to use. It rejects a unit root in levels for all three: **every variable is I(0)**. Notice that for unemployment the first-generation Fisher test does *not* reject. It ignores the common shocks, and the two tests disagree; CIPS is the valid one here.
- **Slope homogeneity.** The adjusted Δ test does not reject: there is no evidence that the effects of unemployment and openness differ across countries. **Pooling the slopes is defensible.**

These are the conditions under which the decision rules of Chapter 1 recommend fixed effects with Driscoll–Kraay standard errors.

18.4 Estimate fixed effects with three kinds of standard errors

```
x = long.set_index(["country", "year"])
xs = ["unem", "d_trade"]
```

```
y, X = x["gdp_growth"], sm.add_constant(x[xs])

fits = {"Conventional": PanelOLS(y, X, entity_effects=True).fit(),
        "Clustered by country": PanelOLS(y, X, entity_effects=True).fit(cov_type="clustered", cluster_entity=
  ↪ True),
        "Driscoll-Kraay": PanelOLS(y, X, entity_effects=True).fit(cov_type="kernel", kernel="bartlett",
  ↪ bandwidth=3)}
dk = fits["Driscoll-Kraay"]
print(f"{'':10s}{'coef':>8s}" + "".join(f"{k:>22s}" for k in fits))
for v in xs:
    print(f"{v:10s}{dk.params[v]:8.3f}" + "".join(f"{m.std_errors[v]:13.3f} (p {m.pvalues[v]:.3f})" for m in
  ↪ fits.values()))
```

```
              coef          Conventional  Clustered by country        Driscoll-Kraay
unem        -0.344        0.049 (p 0.000)        0.052 (p 0.000)        0.085 (p 0.000)
d_trade     -0.101        0.030 (p 0.001)        0.023 (p 0.000)        0.075 (p 0.182)
```

How to read it

The coefficients are the same in every column; only the standard errors differ.

- **Unemployment.** A one-point higher unemployment rate goes with growth about a third of a point lower. Significant under every choice of standard errors.
- **Trade openness.** With conventional or clustered standard errors, the negative coefficient looks significant. **With Driscoll–Kraay standard errors it is not.** Clustering by country allows for correlation over time within a country, but still treats countries as independent of each other. Driscoll–Kraay standard errors also allow for correlation **across countries in the same year**, which the CD test showed is strong. Openness moves together across countries (world trade booms and slumps), so its standard error is the one most affected.

The Driscoll–Kraay estimator needs a reasonably long T, since it averages over years rather than countries; with 24 years it is appropriate. The bandwidth (lags of the Newey–West kernel) is set to 3, roughly $T^{1/3}$.

18.5 Diagnostic tests on the residuals

```
e = dk.resids.unstack("country")
cd, p, r = cd_test(e)
print(f"CD test on the FE residuals: CD = {cd:.2f}   p-value = {p:.4f}   mean |rho| = {r:.3f}")

r1 = (e * e.shift(1)).sum().sum() / (e ** 2).sum().sum()
print(f"First-order autocorrelation of the residuals (pooled): {r1:.3f}")
```

```
CD test on the FE residuals: CD = 12.20   p-value = 0.0000   mean |rho| = 0.286
First-order autocorrelation of the residuals (pooled): 0.256
```

How to read it

- **Residual dependence.** Even after the two regressors, the residuals of different countries remain correlated: the common shocks are not explained by the model. This is exactly the case Driscoll–Kraay standard errors are built for.
- **Serial correlation.** The residuals are positively autocorrelated, as is usual for annual macro data. The Bartlett kernel in the Driscoll–Kraay estimator corrects for it.

But there is a deeper question. Driscoll–Kraay fixes the **standard errors**. If the common shocks are also correlated with the regressors, the **coefficients** themselves are biased, and no standard-error correction helps. The robustness checks test this.

18.6 Robustness: are the coefficients driven by common shocks?

```
twfe = PanelOLS(y, X, entity_effects=True, time_effects=True).fit(cov_type="kernel", kernel="bartlett",
  ↪ bandwidth=3)

# CCE pooled (Pesaran 2006): add cross-sectional averages, with country-specific coefficients
bars = x.groupby(level="year")[["gdp_growth"] + xs].transform("mean").add_suffix("_bar")
```

```
cty = pd.get_dummies(x.index.get_level_values("country"), dtype=float).set_index(x.index)
inter = pd.concat({f"{b}_{c}": bars[b] * cty[c] for b in bars for c in cty}, axis=1)
ccep = PanelOLS(y, sm.add_constant(x[xs]).join(inter), entity_effects=True, drop_absorbed=True).fit(cov_type="
  ↪ kernel", kernel="bartlett", bandwidth=3)

bw5 = PanelOLS(y, X, entity_effects=True).fit(cov_type="kernel", kernel="bartlett", bandwidth=5)
print(f"{'':30s}{'unem':>18s}{'d_trade':>18s}")
for name, m in (("FE, Driscoll-Kraay (baseline)", dk), ("FE, DK, bandwidth 5", bw5), ("Two-way FE (year
  ↪ effects), DK", twfe), ("CCE pooled, DK", ccep)):
    print(f"{name:30s}" + "".join(f"{m.params[v]:9.3f} ({m.std_errors[v]:.3f})" for v in xs))
```

```
                                       unem           d_trade
FE, Driscoll-Kraay (baseline)    -0.344 (0.085)   -0.101 (0.075)
FE, DK, bandwidth 5              -0.344 (0.087)   -0.101 (0.078)
Two-way FE (year effects), DK    -0.074 (0.044)    0.132 (0.025)
CCE pooled, DK                   -0.131 (0.064)    0.164 (0.027)
```

How to read it

This is the most important table in a macro-panel paper.

- **Bandwidth.** A longer kernel changes the standard errors only a little.
- **Year effects.** Adding a dummy for each year removes everything common to all countries in that year. The unemployment coefficient shrinks towards zero and the openness coefficient **turns positive and significant**.
- **CCE pooled** removes common factors through cross-sectional averages, allowing each country to respond to them differently. It tells the same story as the year effects.

The interpretation: in the baseline, both coefficients mostly reflected the **common OECD cycle**. In the 1970s oil shocks, growth fell everywhere while unemployment rose and the value of trade (inflated by oil prices) rose. Once the common shocks are removed, countries whose openness rose *faster than the OECD average* grew faster. **The baseline coefficient was biased by the common shocks, not just its standard error.**

So for this question the robustness check changes the conclusion: report the model with year effects or CCE as the main specification, and the plain FE-DK as the comparison. The decision rules of Chapter 1 flagged the residual cross-sectional dependence; this step is how you act on it.

18.7 Build the results table for the paper

```
def star(p):
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) FE": dk, "(2) Two-way FE": twfe, "(3) CCE pooled": ccep}
print(f"{'':12s}" + "".join(f"{c:>18s}" for c in cols))
for v in xs:
    print(f"{v:12s}" + "".join(f"{m.params[v]:15.3f}{star(m.pvalues[v]):3s}" for m in cols.values()))
    print(f"{'':12s}" + "".join(f"{'(' + format(m.std_errors[v], '.3f') + ')':>15s}   " for m in cols.values())
  ↪ )
print(f"{'Country FE':12s}" + f"{'yes':>15s}   " * 3)
print(f"{'Year FE':12s}{'no':>15s}   {'yes':>15s}   {'no':>15s}")
print(f"{'CS averages':12s}{'no':>15s}   {'no':>15s}   {'yes':>15s}")
print(f"{'Obs.':12s}" + "".join(f"{m.nobs:>15d}   " for m in cols.values()))
print("Driscoll-Kraay standard errors (Bartlett kernel, 3 lags). *** p<0.01, ** p<0.05, * p<0.10.")
```

```
                      (1) FE    (2) Two-way FE    (3) CCE pooled
unem                -0.344***         -0.074*          -0.131**
                   (0.085)           (0.044)          (0.064)
d_trade             -0.101            0.132***         0.164***
                   (0.075)           (0.025)          (0.027)
Country FE             yes               yes              yes
Year FE                 no               yes               no
CS averages             no                no              yes
Obs.                   336               336              336
Driscoll-Kraay standard errors (Bartlett kernel, 3 lags). *** p<0.01, ** p<0.05, * p<0.10.
```

How to read it

Report the pre-tests (CD, CIPS, slope homogeneity) in a separate table before this one. In this table, the progression from column (1) to columns (2) and (3) shows the reader how much of the raw correlation is due to common shocks.

18.8 Write the paper

How to write it: pre-tests and estimator

Pesaran's CD test rejects cross-sectional independence for all variables, so we test for unit roots with the CIPS test, which rejects a unit root in levels for each series. The Pesaran–Yamagata test does not reject slope homogeneity (adjusted Δ, $p = 0.16$). We therefore estimate a fixed-effects model with Driscoll–Kraay standard errors, which are robust to heteroskedasticity, serial correlation and cross-sectional dependence.

How to write it: results

In the fixed-effects model, higher unemployment is associated with lower growth, and the change in trade openness has a negative but insignificant coefficient once standard errors allow for cross-sectional dependence (Table 2, column 1). Because the residuals remain cross-sectionally dependent, we also control for common shocks with year effects and with the common correlated effects estimator. In both specifications, increases in trade openness relative to the OECD average are associated with significantly higher growth (columns 2 and 3), indicating that the baseline estimate was confounded by the common business cycle.

Common mistakes

- Clustering by country in a macro panel and calling the standard errors robust. They are not robust to common shocks.
- Using Driscoll–Kraay with a short T. The estimator averages over time periods, so it needs many of them.
- Thinking Driscoll–Kraay solves cross-sectional dependence. It corrects the inference only; if the common factor is correlated with the regressors, the coefficients are biased. Always compare with year effects or CCE.
- Using first-generation unit-root tests (LLC, IPS, Fisher) when the CD test rejects.

CHAPTER 19

Mean Group and CCE Mean Group

Applies when. Long panel, stationary variables, heterogeneous slopes

Overview. Use this chapter when the panel has a long time dimension and every variable is stationary, but the slope-homogeneity test rejects. A pooled estimator then reports a weighted average of very different effects, with the wrong standard error. Mean-group estimators run one regression per unit and average the slopes; the CCE version also removes common shocks.

Data. 125 countries, annual, 1961–1985 (Summers–Heston Penn World Table)

Software. Python · linearmodels · statsmodels

19.1 The question and the model

The question is one of the oldest in growth economics: **do countries that save more grow faster?** In the Solow model a higher saving rate raises the level of income and, during the transition, the growth rate. But there is no reason to expect the same relationship in Japan, Chile and Nigeria: financial systems, openness and the use of saving differ.

$$g_{it} = \alpha_i + \beta_i\, s_{it} + \varepsilon_{it}, \qquad \beta_i = \beta + v_i$$

Symbol	Variable	Definition	Expected sign
g_{it}	`growth`	Growth of real GDP per capita, % (computed below)	dependent variable
s_{it}	`sr`	Saving rate, % of GDP	positive
β_i	—	Country-specific effect of saving on growth	what we want to average

The model lets **every country have its own slope** β_i. The parameter of interest is the average slope β.

Pooled or mean group?

Fixed effects imposes $\beta_i = \beta$ for all countries. If that is wrong, the FE estimate is a weighted average of the β_i, with weights that depend on the variance of each country's saving rate, not on anything economically meaningful, and its standard error ignores the slope heterogeneity. The **mean group** estimator (Pesaran and Smith 1995) estimates each β_i separately and averages them. Its standard error comes from the dispersion of the β_i across countries, so it is valid whatever the heterogeneity. It needs T large enough to estimate a regression for each country.

How to write it: model section

We allow the effect of saving on growth to differ across countries, $g_{it} = \alpha_i + \beta_i s_{it} + \varepsilon_{it}$, and estimate the average effect $\beta = E(\beta_i)$ with the mean group estimator of Pesaran and Smith (1995), and with the common correlated effects mean group estimator of Pesaran (2006), which in addition controls for unobserved common factors.

19.2 Load the data and run the pre-tests

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

```
import matplotlib.pyplot as plt
from linearmodels.panel import PanelOLS
from panel_tests import cd_test, cips_test, fisher_adf_test, slope_test
from panel_estimators import mean_group, cce_mg                          # examples/panel_estimators.py

raw = pd.read_csv("growth_saving.csv").sort_values(["country", "year"])
raw["growth"] = 100 * np.log(raw["gdp"]).groupby(raw["country"]).diff()   # gdp is real GDP per capita
df = raw.dropna(subset=["growth"])[["country", "year", "growth", "sr"]]
print(f"{df['country'].nunique()} countries, {df['year'].min()}-{df['year'].max()}, {len(df)} observations")

for v in ["growth", "sr"]:
    W = df.pivot(index="year", columns="country", values=v)
    cd, p, r = cd_test(W); c = cips_test(W)
    print(f"{v:7s} CD = {cd:6.2f} (p {p:.3f}, mean |rho| {r:.2f})   CIPS = {c['stat']:.2f} (p {c['p']:.4f})")

s = slope_test(df, "country", "growth", ["sr"])
print(f"Pesaran-Yamagata: adjusted delta = {s['delta_adj']:.2f}   p-value = {s['delta_adj_p']:.4f}")
```

```
125 countries, 1961-1985, 3125 observations
growth  CD =  36.30 (p 0.000, mean |rho| 0.19)   CIPS = -3.35 (p 0.0005)
sr      CD =  27.76 (p 0.000, mean |rho| 0.36)   CIPS = -2.11 (p 0.0221)
Pesaran-Yamagata: adjusted delta = 6.64   p-value = 0.0000
```

How to read it

- **125 countries over 25 years**: long enough to estimate a regression for each country.
- **Cross-sectional dependence.** The CD test rejects for both variables. World growth has a common cycle, so CIPS is the right unit-root test.
- **Stationarity.** CIPS rejects a unit root for both growth and the saving rate: both are I(0), and a model in levels is valid.
- **Slope heterogeneity.** The Δ test rejects decisively. **The effect of saving on growth differs across countries**, so pooling the slopes is not defensible.

Stationary variables with heterogeneous slopes: this is where the decision rules of Chapter 1 recommend mean-group estimation.

19.3 Estimate one regression per country

```
mg = mean_group(df, "country", "year", "growth", ["sr"])
b = mg["units"]["sr"]
print(f"Country slopes: {len(b)} estimated")
print(b.describe()[["mean", "std", "min", "25%", "50%", "75%", "max"]].round(3).to_string())
print(f"Share of countries with a positive slope: {(b > 0).mean():.2f}")

fig, ax = plt.subplots()
ax.hist(b.clip(-1.5, 1.5), bins=30, color="#0033a0", alpha=.8)
ax.axvline(b.mean(), color="#c46000", lw=2, label=f"mean group = {b.mean():.2f}")
ax.axvline(0, color="black", lw=.8)
ax.set_xlabel("Effect of a one-point higher saving rate on growth, by country (clipped at ±1.5)")
ax.set_ylabel("Countries"); ax.legend()
```

```
Country slopes: 125 estimated
mean    0.346
std     1.108
min    -1.569
25%     0.058
50%     0.225
75%     0.482
max     6.813
Share of countries with a positive slope: 0.78
```

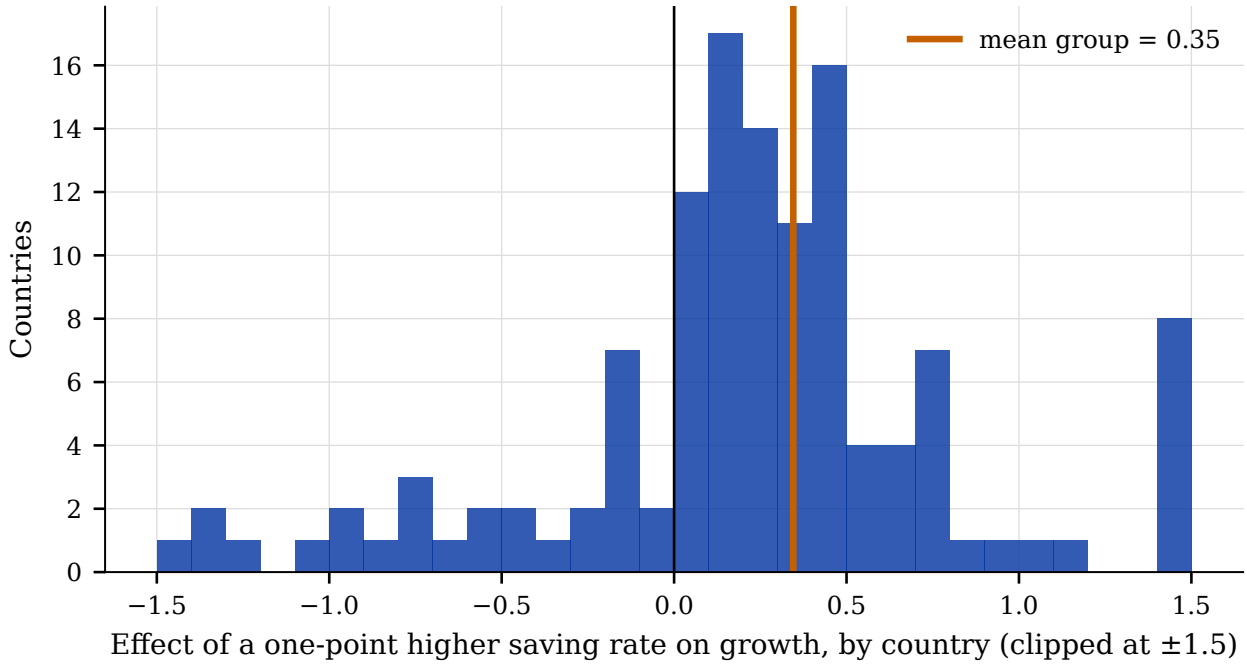


Figure 19.1. Estimate one regression per country.

How to read it

The histogram is the reason for this chapter. The country slopes are spread widely, and a substantial minority are negative. A single pooled coefficient would hide all of this. The **mean group estimate is the simple average of these slopes**, and its standard error is their standard deviation divided by $\sqrt{N}$.

The extreme slopes come from small countries with volatile data and few effective observations. Because every country gets the same weight, a few of them can move the mean; the robustness step checks this.

19.4 Mean group, CCE mean group and the pooled comparison

```
cce = cce_mg(df, "country", "year", "growth", ["sr"])
x = df.set_index(["country", "year"])
fe = PanelOLS(x["growth"], sm.add_constant(x[["sr"]]), entity_effects=True).fit(cov_type="kernel", kernel="
  ↪ bartlett", bandwidth=3)
twfe = PanelOLS(x["growth"], sm.add_constant(x[["sr"]]), entity_effects=True, time_effects=True).fit(cov_type=
  ↪ "kernel", kernel="bartlett", bandwidth=3)

rows = {"Fixed effects (DK s.e.)": (fe.params["sr"], fe.std_errors["sr"]), "Two-way FE (DK s.e.)": (twfe.
  ↪ params["sr"], twfe.std_errors["sr"]),
        "Mean group": tuple(mg["table"].loc["sr", ["coef", "std err"]]), "CCE mean group": tuple(cce["table"].
  ↪ loc["sr", ["coef", "std err"]])}
for name, (c, se) in rows.items():
    print(f"{name:26s} beta = {c:6.3f}   s.e. = {se:.3f}   t = {c / se:5.2f}")
```

```
Fixed effects (DK s.e.)    beta =  0.146   s.e. = 0.043   t =  3.37
Two-way FE (DK s.e.)       beta =  0.144   s.e. = 0.048   t =  3.01
Mean group                 beta =  0.346   s.e. = 0.099   t =  3.49
CCE mean group             beta =  0.478   s.e. = 0.145   t =  3.31
```

How to read it

- **Mean group.** On average across countries, a saving rate one percentage point higher goes with growth about 0.35 points higher, and the effect is significant.

- **CCE mean group** adds the cross-sectional averages of growth and saving to each country's regression. They proxy the unobserved common factors (the world business cycle, oil prices, world interest rates), and each country can respond to them differently. The average effect is a bit larger and remains significant.
- **Pooled fixed effects** gives a smaller slope. The difference is the weighting: FE gives more weight to countries whose saving rate varies more, and those are not the countries with the largest effects. With heterogeneous slopes the FE number has no clear interpretation, which is the point of the Δ test.

19.5 Diagnostic test: are the common shocks removed?

```
def resid_cd(estimator_bars):
    R = []
    for c, g in df.groupby("country"):
        g = g.merge(estimator_bars, on="year", how="left").dropna()
        X = np.column_stack([np.ones(len(g)), g[["sr"] + [col for col in estimator_bars.columns if col != "
 ↪ year"]]])
        b_, *_ = np.linalg.lstsq(X, g["growth"].to_numpy(), rcond=None)
        R.append(pd.Series(g["growth"].to_numpy() - X @ b_, index=g["year"], name=c))
    return cd_test(pd.concat(R, axis=1))

none = pd.DataFrame({"year": sorted(df["year"].unique())})
bars = df.groupby("year")[["growth", "sr"]].mean().add_suffix("_bar").reset_index()
for name, z in (("MG residuals", none), ("CCE-MG residuals", bars)):
    cd, p, r = resid_cd(z)
    print(f"{name:18s} CD = {cd:6.2f}   p-value = {p:.4f}   mean |rho| = {r:.3f}")
```

```
MG residuals       CD =  25.56   p-value = 0.0000   mean |rho| = 0.182
CCE-MG residuals   CD =   0.18   p-value = 0.8593   mean |rho| = 0.176
```

How to read it

The plain mean-group residuals are still cross-sectionally dependent: country regressions with only their own saving rate leave the world cycle in the error. With the cross-sectional averages included, the CD statistic falls from about 26 to about 0.2 and is no longer significant: **the common factors have been removed.** This is why CCE-MG is the preferred estimator when the CD test rejects. If the common factors were correlated with saving, plain MG would be biased. (The mean absolute correlation barely changes, because with T = 25 any two countries' residuals are correlated by chance; the CD test looks at the signed correlations, which are what common shocks create.)

19.6 Robustness: outliers and the weighting

```
trim = b[(b > b.quantile(0.05)) & (b < b.quantile(0.95))]
bc = cce["units"]["sr"]
print(f"MG, all countries:               {b.mean():.3f} (s.e. {b.std() / np.sqrt(len(b)):.3f})")
print(f"MG, trimmed 5% each tail:        {trim.mean():.3f} (s.e. {trim.std() / np.sqrt(len(trim)):.3f})")
print(f"Median country slope (MG):       {b.median():.3f}")
print(f"Median country slope (CCE-MG):   {bc.median():.3f}")
```

```
MG, all countries:               0.346 (s.e. 0.099)
MG, trimmed 5% each tail:        0.230 (s.e. 0.037)
Median country slope (MG):       0.225
Median country slope (CCE-MG):   0.307
```

How to read it

Because the mean group gives every country equal weight, report the trimmed mean and the median alongside it. Here they matter. The distribution of country slopes is skewed to the right: a few countries have very large slopes (the largest is 6.8), and they pull the mean up. **Trimming the extreme 5% at each end, or taking the median, gives an effect of about 0.23**: smaller than the baseline mean group, but still positive and, for the trimmed mean, more precisely estimated. The sign is robust; the size depends on how much weight the extreme countries get. Say so in the paper.

19.7 Build the results table for the paper

```
def star(t):
    p = 2 * (1 - __import__("scipy").stats.norm.cdf(abs(t)))
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

print(f"{'':16s}{'(1) FE':>14s}{'(2) Two-way FE':>16s}{'(3) MG':>14s}{'(4) CCE-MG':>14s}")
vals = list(rows.values())
print(f"{'Saving rate':16s}" + "".join(f"{c:11.3f}{star(c / se):3s}" for c, se in vals))
print(f"{'':16s}" + "".join(f"{'(' + format(se, '.3f') + ')':>11s}   " for c, se in vals))
print(f"{'Slopes':16s}{'common':>14s}{'common':>14s}{'by country':>14s}{'by country':>14s}")
print(f"{'Common factors':16s}{'no':>14s}{'year FE':>14s}{'no':>14s}{'CS averages':>14s}")
print(f"Countries: {mg['n']}. FE: Driscoll-Kraay standard errors. MG: Pesaran-Smith standard errors. *** p
  ↪ <0.01, ** p<0.05, * p<0.10.")
print(f"Pesaran-Yamagata adjusted delta = {s['delta_adj']:.2f} (p < 0.001).")
```

```
                      (1) FE  (2) Two-way FE        (3) MG    (4) CCE-MG
Saving rate          0.146***      0.144***      0.346***      0.478***
                    (0.043)       (0.048)       (0.099)       (0.145)
Slopes                  common        common    by country    by country
Common factors             no       year FE            no   CS averages
Countries: 125. FE: Driscoll-Kraay standard errors. MG: Pesaran-Smith standard errors. *** p<0.01, ** p<0.05, * p
↪ <0.10.
Pesaran-Yamagata adjusted delta = 6.64 (p < 0.001).
```

How to read it

Report the pooled estimates for comparison, but make column (4) the main result: it allows for both heterogeneous slopes and common shocks, the two features the pre-tests found.

19.8 Write the paper

How to write it: pre-tests and estimator

The CD test indicates strong cross-sectional dependence, and the CIPS test rejects a unit root for both growth and the saving rate. The Pesaran–Yamagata test strongly rejects slope homogeneity, so pooled estimators would average heterogeneous country effects with arbitrary weights. We therefore estimate the average effect with the mean group (MG) estimator and, to account for unobserved common factors, with the common correlated effects mean group (CCE-MG) estimator.

How to write it: results

The CCE-MG estimate implies that a saving rate one percentage point higher is associated with growth of GDP per capita about 0.5 percentage points higher on average (Table 3, column 4). The effect varies considerably across countries: it is negative in about a fifth of them, and the distribution is skewed by a few countries with very large effects. Excluding the most extreme 5 percent of country estimates at each end, or using the median country slope, yields a smaller but still positive effect of 0.2 to 0.3. Pooled fixed-effects estimates are smaller still, consistent with the arbitrary weighting of heterogeneous slopes.

Common mistakes

- Reporting a pooled FE coefficient after the slope-homogeneity test rejects, without comment.
- Using mean group estimators with short T. Each country regression needs enough observations; with T below about 20 the unit estimates are noisy and MG can be badly biased in dynamic models.
- Forgetting the common factors: MG with cross-sectionally dependent errors can be biased. Use CCE-MG and report the residual CD test.
- Reading the average slope as causal. Saving and growth are jointly determined; this is a conditional correlation unless you have a source of exogenous variation.

CHAPTER 20

Panel ARDL: Pooled Mean Group

Applies when. Long-run relationship, cross-sectionally independent units

Overview. Use this chapter when a long panel has a long-run relationship among variables that are a mix of I(0) and I(1), or I(1) and cointegrated, and the units are not cross-sectionally dependent. The pooled mean group estimator imposes the same long-run elasticities on all units while letting short-run dynamics and the speed of adjustment differ. The chapter estimates PMG, MG and dynamic fixed effects and uses the Hausman test to choose among them.

Data. 46 U.S. states, annual cigarette sales, 1963–1992 (Baltagi and Levin)

Software. Python · linearmodels · statsmodels

20.1 The question and the model

The question: **what are the long-run price and income elasticities of cigarette demand?** Smoking is addictive, so consumption adjusts slowly to price changes, and the long-run response is what matters for tax policy. The data also contain the minimum price in neighbouring states, which captures cross-border buying ("bootlegging").

The panel ARDL(1,1) model in error-correction form is

$$\Delta c_{it} = \phi_i \left(c_{i,t-1} - \theta_1 p_{i,t-1} - \theta_2 y_{i,t-1} - \theta_3 pn_{i,t-1} \right) + \gamma_{i1} \Delta p_{it} + \gamma_{i2} \Delta y_{it} + \gamma_{i3} \Delta pn_{it} + \mu_i + \varepsilon_{it}$$

Symbol	Variable	Definition	Expected sign of θ
c_{it}	`log_sales`	Log packs per capita	dependent variable
p_{it}	`log_price`	Log real price per pack	negative: the long-run price elasticity
y_{it}	`log_income`	Log real disposable income per capita	positive if cigarettes are a normal good
pn_{it}	`log_pimin`	Log real minimum price in neighbouring states	positive: cheaper neighbours draw buyers away
ϕ_i	—	Speed of adjustment to the long run	negative, between –2 and 0

PMG restricts the long-run coefficients θ to be the same in every state, and lets ϕ_i, the short-run coefficients γ_i and the intercepts differ. **MG** lets everything differ and averages. **Dynamic fixed effects (DFE)** restricts everything except the intercepts.

Why PMG suits this question

Long-run elasticities are often similar across units, because they reflect preferences and technology, whereas short-run adjustment depends on local institutions (here, state tax changes and their timing). PMG imposes the restriction where it is plausible and estimates it efficiently. Whether the restriction holds is tested by comparing PMG with MG.

How to write it: model section

We estimate an ARDL(1,1) model of cigarette demand in error-correction form, in which the long-run elasticities of per-capita sales with respect to the real price, real income and the real minimum price in neighbouring states are common across states, while the short-run coefficients and the speed of adjustment are state-specific. The model is estimated by the pooled mean group (PMG) estimator of Pesaran, Shin and Smith (1999).

20.2 Load the data and remove the common shocks

```
import pandas as pd
import numpy as np
from panel_tests import cd_test, cips_test, fisher_adf_test, slope_test
from panel_estimators import ardl_ec, hausman      # examples/panel_estimators.py

raw = pd.read_csv("cigar_raw.csv").sort_values(["state", "year"])
df = pd.DataFrame({"state": raw["state"], "year": raw["year"], "log_sales": np.log(raw["sales"]),
                   "log_price": np.log(raw["price"] / raw["cpi"]), "log_income": np.log(raw["ndi"] / raw["cpi"
  ↪ ]),
                   "log_pimin": np.log(raw["pimin"] / raw["cpi"])})
V = ["log_sales", "log_price", "log_income", "log_pimin"]
dm = df.copy()
for v in V:
    dm[v] = df[v] - df.groupby("year")[v].transform("mean")     # subtract the cross-state mean of each year

print(f"{df['state'].nunique()} states, {df['year'].min()}-{df['year'].max()}, {len(df)} observations\n")
print(f"{'':12s}{'CD raw':>9s}{'CD demeaned':>13s}{'p':>8s}")
for v in V:
    c0 = cd_test(df.pivot(index="year", columns="state", values=v))
    c1 = cd_test(dm.pivot(index="year", columns="state", values=v))
    print(f"{v:12s}{c0[0]:9.1f}{c1[0]:13.2f}{c1[1]:8.3f}")
```

```
46 states, 63-92, 1380 observations

              CD raw  CD demeaned       p
log_sales       101.5        -2.77   0.006
log_price       154.1        -3.10   0.002
log_income      166.3        -3.40   0.001
log_pimin       162.5        -3.49   0.000
```

How to read it

In the raw data every variable is strongly cross-sectionally dependent: national trends (the 1964 Surgeon General's report, federal taxes, inflation) move all states together. **Cross-sectional demeaning**, subtracting each year's average across states, removes common shocks that affect all states equally. It is the standard first step before PMG, which assumes independent units.

Demeaning removes almost all of the dependence: the CD statistics fall from above 100 to about –3. What is left is small and **negative**, partly a mechanical by-product of demeaning (subtracting the cross-state average makes deviations slightly negatively correlated). Demeaning cannot remove common shocks to which states respond differently; for those, use CS-ARDL (see that chapter, which uses the same data without demeaning). The test that matters is the CD test on the PMG residuals, below.

20.3 Unit roots and slope heterogeneity

```
print(f"{'':12s}{'CIPS level':>11s}{'p':>8s}{'Fisher level':>14s}{'p':>8s}{'CIPS diff':>11s}{'p':>8s}")
for v in V:
    W = dm.pivot(index="year", columns="state", values=v)
    a, f, b = cips_test(W), fisher_adf_test(W), cips_test(W.diff())
    print(f"{v:12s}{a['stat']:11.2f}{a['p']:8.3f}{f['Z']:14.2f}{f['Z_p']:8.3f}{b['stat']:11.2f}{b['p']:8.3f}")
s = slope_test(dm, "state", "log_sales", V[1:])
print(f"\nPesaran-Yamagata adjusted delta = {s['delta_adj']:.2f}   p-value = {s['delta_adj_p']:.4f}")
```

```
             CIPS level       p  Fisher level       p  CIPS diff       p
log_sales         -1.17   0.995         -0.48   0.315      -3.32   0.001
log_price         -2.18   0.026         -8.94   0.000      -4.07   0.001
log_income        -1.56   0.727         -2.44   0.007      -3.20   0.001
log_pimin         -2.27   0.009         -8.17   0.000      -3.79   0.001

Pesaran-Yamagata adjusted delta = 31.76   p-value = 0.0000
```

How to read it

- **Unit roots.** The variables are a mix: some do not reject a unit root in levels, others do. Every variable is stationary in first differences, and none is I(2). **This mix of I(0) and I(1) is the case the panel ARDL is designed for**: the bounds-testing logic of ARDL does not require all variables to share the same order of integration.
- **Slope heterogeneity.** The static Δ test rejects. That does not rule PMG out: PMG lets the short-run coefficients differ, and only restricts the long run. Whether the long-run restriction is acceptable is tested directly in the Hausman step.

20.4 Estimate PMG, MG and dynamic fixed effects

```
x = V[1:]
pmg = ardl_ec(dm, "state", "year", "log_sales", x, method="pmg")
mg = ardl_ec(dm, "state", "year", "log_sales", x, method="mg")
dfe = ardl_ec(dm, "state", "year", "log_sales", x, method="dfe")
print(f"PMG converged in {pmg['iterations']} iterations\n")
print(f"{'Long run':18s}" + "".join(f"{v:>20s}" for v in x))
for name, r in (("PMG", pmg), ("MG", mg), ("DFE", dfe)):
    t = r["long_run"]
    print(f"{name:18s}" + "".join(f"{t.loc[v, 'coef']:11.3f} ({t.loc[v, 'std err']:.3f})" for v in x))
print()
for name, r in (("PMG", pmg), ("MG", mg), ("DFE", dfe)):
    e = r["phi"].loc["ec"]
    print(f"{name:4s} error-correction coefficient phi = {e['coef']:.3f} ({e['std err']:.3f})   half-life = {
  ↪ np.log(.5) / np.log(1 + e['coef']):.1f} years")
```

```
PMG converged in 42 iterations

Long run                     log_price          log_income           log_pimin
PMG                    -0.766 (0.052)      0.931 (0.052)      0.418 (0.071)
MG                     -1.185 (0.471)     -0.042 (0.607)      1.106 (1.102)
DFE                    -1.397 (0.274)      0.696 (0.254)      0.270 (0.265)

PMG  error-correction coefficient phi = -0.193 (0.036)   half-life = 3.2 years
MG   error-correction coefficient phi = -0.408 (0.049)   half-life = 1.3 years
DFE  error-correction coefficient phi = -0.117 (0.020)   half-life = 5.6 years
```

How to read it

- **PMG long run.** A 10% rise in the real price lowers cigarette sales by about 7.7% in the long run. Income has a positive long-run effect, and a higher price in neighbouring states raises sales at home: when the neighbours are cheap, smokers cross the border. All three are precisely estimated, because PMG pools the long-run information from all 46 states.
- **MG** estimates each state separately. Its price elasticity is larger in absolute value and its income and neighbour-price effects are imprecise: with 29 years per state, individual long-run coefficients are noisy, and averaging noisy ratios inflates the standard errors.
- **DFE** pools everything, including the adjustment speed, which biases the long run when dynamics differ across states.
- **Speed of adjustment.** ϕ is negative and significant, which supports the long-run relationship: deviations from it are corrected. With PMG, about a fifth of the gap closes each year, a half-life of about three years. Smoking adjusts slowly, as addiction models predict.

20.5 Test 1: is the common long run acceptable? Hausman PMG vs MG

```
H, dfree, p = hausman(pmg["theta"], pmg["V"], mg["theta"], mg["V"])
print(f"Hausman test, PMG vs MG:  chi2({dfree}) = {H:.2f}   p-value = {p:.3f}")
```

```
Hausman test, PMG vs MG:  chi2(3) = 3.94   p-value = 0.268
```

How to read it

Under the null that the long-run coefficients are the same in every state, both PMG and MG are consistent and PMG is efficient. Under the alternative, only MG is consistent. The test does not reject, so **the common long-run restriction is accepted, and PMG is the preferred estimator.** This is the test referees expect in every PMG paper.

20.6 Test 2: residual cross-sectional dependence and the adjustment speeds

```
U = {}
for st, g in dm.sort_values(["state", "year"]).groupby("state"):
    ec = g["log_sales"].shift(1) - g[x].shift(1) @ pmg["theta"]
    Z = np.column_stack([np.ones(len(g)), ec, g[x].diff()])[1:]
    dy = g["log_sales"].diff().to_numpy()[1:]
    b, *_ = np.linalg.lstsq(Z, dy, rcond=None)
    U[st] = pd.Series(dy - Z @ b, index=g["year"].to_numpy()[1:])
cd_r, p_r, r_r = cd_test(pd.DataFrame(U))
print(f"CD test on PMG residuals: CD = {cd_r:.2f}   p-value = {p_r:.3f}   mean |rho| = {r_r:.3f}")
print(f"Correlation induced by demeaning alone: -1/(N-1) = {-1 / (len(U) - 1):.3f}")

phis = pmg["phi_units"]
print(f"State adjustment speeds: min {phis.min():.2f}, median {np.median(phis):.2f}, max {phis.max():.2f}; "
      f"{(phis < 0).sum()} of {len(phis)} negative, {(phis < -2).sum()} below -2")
```

```
CD test on PMG residuals: CD = -2.12   p-value = 0.034   mean |rho| = 0.163
Correlation induced by demeaning alone: -1/(N-1) = -0.022
State adjustment speeds: min -1.05, median -0.13, max 0.12; 39 of 46 negative, 0 below -2
```

How to read it

- **Residual dependence.** PMG assumes the states' errors are independent. The CD test on the residuals rejects narrowly ($p = 0.03$), but the remaining dependence is weak and negative, of the kind demeaning itself induces, not the strong positive dependence of common shocks in the raw data (CD above 100). Report this honestly, and use CS-ARDL as the robustness check: it handles any remaining common factors.
- **Adjustment speeds.** Every state should have $-2 < \phi_i < 0$ for a stable long-run relationship. None is below –2, but 7 of the 46 states have a small positive ϕ_i: for them, sales show no tendency to return to the long-run relationship within the sample. Report the range, and check that the results survive without those states.

20.7 Build the results table for the paper

```
from scipy import stats
def star(b, se):
    p = 2 * stats.norm.sf(abs(b / se))
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) PMG": pmg, "(2) MG": mg, "(3) DFE": dfe}
print(f"{'Long run':14s}" + "".join(f"{c:>16s}" for c in cols))
for v in x:
    print(f"{v:14s}" + "".join(f"{r['long_run'].loc[v, 'coef']:13.3f}{star(*r['long_run'].loc[v, ['coef', 'std
  ↪  err']]):3s}" for r in cols.values()))
    print(f"{'':14s}" + "".join(f"{'(' + format(r['long_run'].loc[v, 'std err'], '.3f') + ')':>13s}   " for r
  ↪ in cols.values()))
print(f"{'Adjustment':14s}" + "".join(f"{r['phi'].loc['ec', 'coef']:13.3f}{star(*r['phi'].loc['ec', ['coef', '
  ↪ std err']]):3s}" for r in cols.values()))
print(f"{'':14s}" + "".join(f"{'(' + format(r['phi'].loc['ec', 'std err'], '.3f') + ')':>13s}   " for r in
  ↪ cols.values()))
print(f"Hausman PMG vs MG: chi2({dfree}) = {H:.2f} (p = {p:.3f}). States: {pmg['n']}. Data cross-sectionally
  ↪ demeaned. *** p<0.01, ** p<0.05, * p<0.10.")
```

```
Long run              (1) PMG          (2) MG         (3) DFE
```

```
log_price            -0.766***       -1.185**        -1.397***
                    (0.052)         (0.471)         (0.274)
log_income            0.931***       -0.042           0.696***
                    (0.052)         (0.607)         (0.254)
log_pimin             0.418***        1.106           0.270
                    (0.071)         (1.102)         (0.265)
Adjustment           -0.193***       -0.408***       -0.117***
                    (0.036)         (0.049)         (0.020)
Hausman PMG vs MG: chi2(3) = 3.94 (p = 0.268). States: 46. Data cross-sectionally demeaned. *** p<0.01, ** p<0.05, *
↪ p<0.10.
```

How to read it

The standard PMG table: long-run coefficients and the error-correction coefficient for PMG, with MG and DFE alongside, and the Hausman statistic in the notes. Report the short-run coefficients in an appendix table.

20.8 Write the paper

How to write it: pre-tests and estimator

To remove common national shocks, all variables are expressed as deviations from their cross-state means in each year. CIPS tests indicate a mixture of I(0) and I(1) variables, none of them I(2), which motivates a panel ARDL specification. We estimate it by the pooled mean group estimator, which constrains the long-run elasticities to be equal across states while allowing state-specific short-run dynamics. A Hausman test does not reject the long-run homogeneity restriction against the mean group estimator, and the remaining cross-sectional dependence in the PMG residuals is weak (CD = −2.1).

How to write it: results

The long-run price elasticity of cigarette demand is about −0.77 (Table 5, column 1): a 10 percent increase in the real price reduces per-capita sales by 7.7 percent in the long run. A higher minimum price in neighbouring states raises home-state sales, consistent with cross-border purchases. The error-correction coefficient is negative and significant, implying that about one fifth of a deviation from long-run equilibrium is corrected each year.

Common mistakes

- Running PMG on cross-sectionally dependent data. Demean first, test the residuals, and switch to CS-ARDL if dependence remains.
- Skipping the Hausman test against MG. The common long run is a restriction and must be tested.
- Using PMG with an I(2) variable, or with short T. Each state's short-run dynamics are estimated separately and need enough years.
- Reporting only the long run. The error-correction coefficient is the evidence that a long-run relationship exists.

CHAPTER 21

Cross-Sectionally Augmented ARDL

Applies when. Long-run relationship with common shocks

Overview. Use this chapter when a long panel has a long-run relationship among I(0) and I(1) variables and the units share common shocks. Panel ARDL estimators that assume independent units are then biased. CS-ARDL (Chudik, Mohaddes, Pesaran and Raissi 2016) adds cross-sectional averages and their lags to each unit's ARDL regression, so the common factors are absorbed, and averages the unit-specific long-run effects.

Data. 46 U.S. states, annual cigarette sales, 1963–1992 (Baltagi and Levin)

Software. Python · linearmodels · statsmodels

21.1 The question and the model

The question is the same as in Chapter 20: **what are the long-run price and income elasticities of cigarette demand?** This time the data are used as they are. Cigarette consumption in every state was driven by national forces: the 1964 Surgeon General's report, advertising bans, changing social norms, federal excise taxes. These common factors affect states to different degrees, and they are correlated with prices and incomes. A model that ignores them confuses the national anti-smoking trend with the effect of price.

For each state i, CS-ARDL(1,1) estimates

$$c_{it} = a_i + \lambda_i c_{i,t-1} + \beta_{0i}' x_{it} + \beta_{1i}' x_{i,t-1} + \sum_{l=0}^{p_T} \delta_{il}' \bar{z}_{t-l} + e_{it}$$

where $x = (p, y)$ are the log real price and log real income, and $\bar{z}_t$ are the **cross-sectional averages** of c and x in year t, with p_T lags. Each state's long-run effect is $\theta_i = (\beta_{0i} + \beta_{1i})/(1 - \lambda_i)$, and the estimate reported is the mean group average of the θ_i.

The neighbours' price used in Chapter 20 is left out here. It moves almost one-for-one with the state's own real price, and once the cross-sectional averages absorb the national component, the two can no longer be separated in 29 annual observations per state: with both included, the state-level long-run estimates become erratic. With CS-ARDL, keep the model small.

Why the cross-sectional averages work

If the errors contain unobserved common factors f_t with state-specific loadings, the cross-sectional averages of the observed variables are (as N grows) a linear combination of those factors. Adding them, and their lags, to each state's regression with state-specific coefficients controls for the factors without having to name or measure them. Because every state gets its own coefficients on the averages, states can respond to the common shocks to different degrees, which year dummies cannot do.

How to write it: model section

We estimate state-specific ARDL(1,1) models of cigarette demand augmented with the contemporaneous and lagged cross-sectional averages of all variables, following Chudik et al. (2016). The averages proxy unobserved common factors, such as national anti-smoking campaigns and federal tax changes, to which states may respond heterogeneously. Long-run elasticities are computed for each state and averaged across states (mean group).

21.2 Load the data and run the pre-tests

```
import pandas as pd
import numpy as np
from panel_tests import cd_test, cips_test, slope_test
from panel_estimators import cs_ardl, ardl_ec

raw = pd.read_csv("cigar_raw.csv").sort_values(["state", "year"])
df = pd.DataFrame({"state": raw["state"], "year": raw["year"], "log_sales": np.log(raw["sales"]),
                   "log_price": np.log(raw["price"] / raw["cpi"]), "log_income": np.log(raw["ndi"] / raw["cpi"
  ↪ ]),
                   "log_pimin": np.log(raw["pimin"] / raw["cpi"])})
V = ["log_sales", "log_price", "log_income"]
x = V[1:]
print("Correlation of own and neighbours' real price:", round(df["log_price"].corr(df["log_pimin"]), 3))
print(f"{df['state'].nunique()} states x {df['year'].nunique()} years\n")
print(f"{'':12s}{'CD':>8s}{'mean |rho|':>12s}{'CIPS level':>12s}{'p':>8s}{'CIPS diff':>11s}{'p':>8s}")
for v in V:
    W = df.pivot(index="year", columns="state", values=v)
    cd, p, r = cd_test(W); a, b = cips_test(W), cips_test(W.diff())
    print(f"{v:12s}{cd:8.1f}{r:12.2f}{a['stat']:12.2f}{a['p']:8.3f}{b['stat']:11.2f}{b['p']:8.3f}")
s = slope_test(df, "state", "log_sales", x)
print(f"\nPesaran-Yamagata adjusted delta = {s['delta_adj']:.2f}   p-value = {s['delta_adj_p']:.4f}")
```

```
Correlation of own and neighbours' real price: 0.815
46 states x 30 years

                CD  mean |rho|  CIPS level       p  CIPS diff       p
log_sales    101.5        0.63       -1.66   0.591      -3.62   0.001
log_price    154.1        0.87       -2.47   0.001      -4.14   0.001
log_income   166.3        0.94       -1.66   0.584      -2.86   0.001

Pesaran-Yamagata adjusted delta = 41.64   p-value = 0.0000
```

How to read it

- **Cross-sectional dependence** is overwhelming: CD statistics above 100 and average absolute correlations between states of 0.7 to 0.9. The states are driven by national forces.
- **Unit roots (CIPS).** Sales and income are I(1); the real price rejects a unit root in levels and is I(0). A mix of I(0) and I(1), none I(2): the ARDL family applies.
- **Slopes** are heterogeneous.

Mixed integration orders and strong cross-sectional dependence: this is the combination for which the decision rules of Chapter 1 recommend CS-ARDL. It is the result the decision rules gives for this example dataset.

21.3 What goes wrong if the common factors are ignored

```
pmg_raw = ardl_ec(df, "state", "year", "log_sales", x, method="pmg")
mg_raw = ardl_ec(df, "state", "year", "log_sales", x, method="mg")
U = {}
for st, g in df.groupby("state"):
    ec = g["log_sales"].shift(1) - g[x].shift(1) @ pmg_raw["theta"]
    Z = np.column_stack([np.ones(len(g)), ec, g[x].diff()])[1:]
    dy = g["log_sales"].diff().to_numpy()[1:]
    bb, *_ = np.linalg.lstsq(Z, dy, rcond=None)
    U[st] = pd.Series(dy - Z @ bb, index=g["year"].to_numpy()[1:])
cd_pmg = cd_test(pd.DataFrame(U))
print("Long run, ignoring common factors:")
for name, r in (("PMG", pmg_raw), ("MG", mg_raw)):
    print(f"  {name:4s}" + "".join(f"{v} = {r['long_run'].loc[v, 'coef']:6.3f} ({r['long_run'].loc[v, 'std err
  ↪ ']:.3f})   " for v in x))
print(f"CD test on the PMG residuals: CD = {cd_pmg[0]:.1f}   p-value = {cd_pmg[1]:.4f}")
```

```
Long run, ignoring common factors:
  PMG log_price = -0.648 (0.026)   log_income =  0.030 (0.028)
  MG  log_price = -0.901 (0.169)   log_income = -0.424 (0.343)
```

```
CD test on the PMG residuals: CD = 41.3   p-value = 0.0000
```

How to read it

PMG and MG on the raw data assume independent states. The CD test on the PMG residuals shows that the assumption fails badly: the residuals are strongly correlated across states. With common factors in the errors that are also correlated with the regressors, these long-run estimates are biased. The clearest symptom is income: real incomes rose everywhere while smoking fell everywhere for reasons unrelated to income (health information, social norms), so the national trend cancels the income effect in PMG and turns it negative in MG. Chapter 20 deals with this by demeaning; CS-ARDL deals with it inside the model.

21.4 Estimate CS-ARDL

```
cs = cs_ardl(df, "state", "year", "log_sales", x, pt=1)
print(f"CS-ARDL(1,1) with {cs['pt']} lag(s) of the cross-sectional averages, {cs['n']} states")
print("\nLong-run coefficients (mean group):")
print(cs["long_run"].round(3))
e = cs["phi"].loc["ec"]
print(f"\nError-correction coefficient (mean of lambda_i - 1): {e['coef']:.3f} ({e['std err']:.3f})")
print(f"Half-life of a deviation: {np.log(.5) / np.log(1 + e['coef']):.1f} years")
```

```
CS-ARDL(1,1) with 1 lag(s) of the cross-sectional averages, 46 states

Long-run coefficients (mean group):
             coef  std err      z  p-value
log_price  -0.879    0.222 -3.967    0.000
log_income  0.574    0.230  2.503    0.012

Error-correction coefficient (mean of lambda_i - 1): -0.702 (0.047)
Half-life of a deviation: 0.6 years
```

How to read it

- **Price.** Once the common factors are absorbed, the long-run price elasticity is about –0.9: a 10% rise in the real price lowers sales by about 9% in the long run. The estimate is close to the PMG estimate on demeaned data (–0.77), which used a different way of dealing with the same common shocks. **Two methods agreeing is strong evidence.**
- **Income.** Positive and significant, about 0.6: cigarettes are a normal good once the national anti-smoking trend is no longer confounded with rising incomes.
- **Adjustment.** The error-correction coefficient is negative and significant: each state's consumption returns towards its long-run relationship. Adjustment is faster than in PMG on demeaned data, because the averages also absorb slow-moving national trends.
- **Standard errors** are mean-group standard errors, from the dispersion of the 46 state estimates. They are valid whatever the heterogeneity across states.

21.5 Test 1: are the common factors removed?

```
zb = df.groupby("year")[V].mean()
lagged = pd.concat({f"{c}_l{l}": zb[c].shift(l) for l in range(2) for c in V}, axis=1)
d = df.join(lagged, on="year")
R = {}
for st, g in d.groupby("state"):
    Y, X = g["log_sales"].to_numpy(), g[x].to_numpy()
    Z = np.column_stack([np.ones(len(Y) - 1), Y[:-1], X[1:], X[:-1], g[lagged.columns].to_numpy()[1:]])
    ok = np.isfinite(Z).all(1)
    bb, *_ = np.linalg.lstsq(Z[ok], Y[1:][ok], rcond=None)
    R[st] = pd.Series(Y[1:][ok] - Z[ok] @ bb, index=g["year"].to_numpy()[1:][ok])
cd, p, r = cd_test(pd.DataFrame(R))
print(f"CD test on the CS-ARDL residuals: CD = {cd:.2f}   p-value = {p:.3f}   mean |rho| = {r:.3f}")
print(f"(PMG on the raw data: CD = {cd_pmg[0]:.1f})")
```

```
CD test on the CS-ARDL residuals: CD = -2.28   p-value = 0.023   mean |rho| = 0.179
```

```
(PMG on the raw data: CD = 41.3)
```

How to read it

The CD statistic falls from the very large value of the raw PMG residuals to a small one: **the cross-sectional averages have absorbed the common factors.** Report this test; it is the evidence that the augmentation did its job. A remaining small, insignificant or weakly significant CD statistic is normal, since the averages approximate the factors only up to an error that shrinks with N.

21.6 Robustness: the number of lags of the averages

```
print(f"{'lags p_T':>9s}" + "".join(f"{v:>20s}" for v in x) + f"{'phi':>10s}")
for pt in (0, 1, 2):
    r = cs_ardl(df, "state", "year", "log_sales", x, pt=pt)
    t = r["long_run"]
    print(f"{pt:9d}" + "".join(f"{t.loc[v, 'coef']:11.3f} ({t.loc[v, 'std err']:.3f})" for v in x) + f"{r['phi
  ↪ '].loc['ec', 'coef']:10.3f}")
k = 2 + 2 * len(x)
print(f"\nRegressors per state with p_T = 2: {k + len(V) * 3}, observations per state: {df['year'].nunique() -
  ↪  1}")
```

```
 lags p_T           log_price           log_income       phi
       0     -0.733 (0.141)      0.507 (0.152)    -0.715
       1     -0.879 (0.222)      0.574 (0.230)    -0.702
       2     -0.751 (0.176)      0.608 (0.172)    -0.826

Regressors per state with p_T = 2: 15, observations per state: 29
```

How to read it

Chudik et al. recommend about $T^{1/3}$ lags of the averages, which is 3 here. But each state regression has only 29 observations, and every lag adds four regressors, so with this many variables the number of lags must stay small. Show that the results do not depend on the choice: here the long-run price elasticity stays between −0.73 and −0.88 and the income elasticity between 0.51 and 0.61. The results are robust to the lag length.

21.7 Build the results table for the paper

```
from scipy import stats
def star(b, se):
    p = 2 * stats.norm.sf(abs(b / se))
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

cols = {"(1) PMG raw": (pmg_raw["long_run"], pmg_raw["phi"]), "(2) MG raw": (mg_raw["long_run"], mg_raw["phi"])
  ↪ , "(3) CS-ARDL": (cs["long_run"], cs["phi"])}
print(f"{'Long run':12s}" + "".join(f"{c:>16s}" for c in cols))
for v in x + ["ec"]:
    get = lambda lr, ph: (ph if v == "ec" else lr).loc[v, ["coef", "std err"]]
    print(f"{v:12s}" + "".join(f"{get(*c)['coef']:13.3f}{star(*get(*c)):3s}" for c in cols.values()))
    print(f"{'':12s}" + "".join(f"{'(' + format(get(*c)['std err'], '.3f') + ')':>13s}   " for c in cols.
  ↪ values()))
print(f"{'Residual CD':12s}{cd_pmg[0]:13.1f}   {'':16s}{cd:13.2f}")
print("Columns (1)-(2) ignore common factors; column (3) includes cross-sectional averages (1 lag). *** p<0.01,
  ↪  ** p<0.05, * p<0.10.")
```

```
Long run         (1) PMG raw      (2) MG raw     (3) CS-ARDL
log_price          -0.648***       -0.901***       -0.879***
                  (0.026)         (0.169)         (0.222)
log_income          0.030          -0.424           0.574**
                  (0.028)         (0.343)         (0.230)
ec                 -0.178***       -0.339***       -0.702***
                  (0.030)         (0.034)         (0.047)
Residual CD          41.3                            -2.28
Columns (1)-(2) ignore common factors; column (3) includes cross-sectional averages (1 lag). *** p<0.01, ** p<0.05, *
↪  p<0.10.
```

How to read it

Put the estimators that ignore the common factors next to CS-ARDL, with the residual CD test in the table, so the reader sees both the bias and its correction. Column (3) is the preferred estimate. Cite the PMG-on-demeaned-data estimate as a second robustness check.

21.8 Write the paper

How to write it: pre-tests and estimator

All variables exhibit strong cross-sectional dependence (Pesaran CD tests), reflecting national anti-smoking trends and federal policy. CIPS tests indicate a mixture of I(0) and I(1) variables, and the Pesaran–Yamagata test rejects slope homogeneity. Because estimators that assume cross-sectional independence would be biased, and because states may respond differently to common shocks, we estimate the long-run elasticities with the cross-sectionally augmented ARDL (CS-ARDL) approach of Chudik et al. (2016).

How to write it: results

The mean group long-run price elasticity is about –0.9 and the income elasticity about 0.6 (Table 6, column 3). The price elasticity is close to the estimate from a pooled mean group model on cross-sectionally demeaned data, and both are robust to the number of lags of the cross-sectional averages. The error-correction coefficient is negative and significant, confirming adjustment towards a long-run relationship. The CD test on the CS-ARDL residuals indicates that the cross-sectional averages successfully absorb the common factors, in contrast with PMG and MG estimates on the raw data, whose residuals remain strongly dependent.

Common mistakes

- Adding too many lags of the averages relative to T. Each lag costs one regressor per variable in every unit regression.
- Using CS-ARDL with small N. The averages approximate the common factors only as N grows; with fewer than about 15 units they are poor proxies.
- Reporting the long run without the residual CD test, which is the evidence that the method worked.
- Forgetting that the mean group long run is an average of unit-specific ratios: a few units with λ_i close to 1 produce extreme θ_i. Check the distribution, and report the median as well.

CHAPTER 22

Panel Model in First Differences

Applies when. All variables I(1), not cointegrated
Overview. Use this chapter when every variable in a long panel has a unit root and the panel cointegration tests find no long-run relationship. A regression in levels is then spurious. The chapter shows the spurious regression, runs the cointegration tests, and estimates the short-run relationship in first differences with estimators that allow for common shocks and heterogeneous slopes.
Data. 18 OECD countries, annual, 1960–1978 (Baltagi and Griffin 1983)
Software. Python · linearmodels · statsmodels

22.1 The question and the model

The question: **how does gasoline demand respond to income and to the price of gasoline?** The dataset is the one Baltagi and Griffin used to compare panel estimators of gasoline demand across OECD countries, before and during the oil shocks of the 1970s.

$$\Delta \ln G_{it} = \beta_1 \Delta \ln Y_{it} + \beta_2 \Delta \ln P_{it} + \beta_3 \Delta \ln C_{it} + \varepsilon_{it}$$

Symbol	Variable in the file	Definition	Expected sign
$\ln G_{it}$	`lgaspcar`	Log gasoline consumption per car	dependent variable
$\ln Y_{it}$	`lincomep`	Log real income per capita	positive: income elasticity
$\ln P_{it}$	`lrpmg`	Log real price of gasoline	negative: price elasticity
$\ln C_{it}$	`lcarpcap`	Log cars per capita	negative: more cars, each driven less

In first differences, the coefficients are **short-run elasticities**: how much gasoline use per car changes in the year that income or the price changes.

Why differences, and what you give up

If the variables are I(1) and cointegrated, a long-run relationship in levels exists and an error-correction model (Chapters 20 and 21) estimates it. If they are I(1) and **not** cointegrated, there is no long-run relationship to estimate: a regression in levels would be spurious, and the valid model is in differences. You give up the long-run elasticities, which is why the cointegration tests deserve care.

How to write it: model section

We relate the growth of gasoline consumption per car to the growth of real income per capita, the real price of gasoline and the stock of cars per capita, $\Delta \ln G_{it} = \beta_1 \Delta \ln Y_{it} + \beta_2 \Delta \ln P_{it} + \beta_3 \Delta \ln C_{it} + \varepsilon_{it}$. The coefficients are short-run elasticities.

22.2 Load the panel and run the pre-tests

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels.panel import PanelOLS
from panel_tests import cd_test, cips_test, slope_test, kao_test, fisher_eg_test
from panel_estimators import mean_group, cce_mg
```

```
df = pd.read_csv("gasoline.csv").sort_values(["country", "year"])
y, xs = "lgaspcar", ["lincomep", "lrpmg", "lcarpcap"]
print(f"{df['country'].nunique()} countries, {df['year'].min()}-{df['year'].max()}, {len(df)} observations\n")

print(f"{'':10s}{'CD':>7s}{'CIPS level':>12s}{'p':>8s}{'CIPS diff':>11s}{'p':>8s}")
for v in [y] + xs:
    W = df.pivot(index="year", columns="country", values=v)
    cd = cd_test(W)[0]; c0, c1 = cips_test(W), cips_test(W.diff())
    print(f"{v:10s}{cd:7.1f}{c0['stat']:12.2f}{c0['p']:8.3f}{c1['stat']:11.2f}{c1['p']:8.3f}")

s = slope_test(df, "country", y, xs)
print(f"\nPesaran-Yamagata adjusted delta = {s['delta_adj']:.2f}   p-value = {s['delta_adj_p']:.4f}")
```

```
18 countries, 1960-1978, 342 observations

              CD  CIPS level       p  CIPS diff       p
lgaspcar    18.9       -1.38   0.796      -2.32   0.028
lincomep    52.7       -2.00   0.167      -2.68   0.002
lrpmg       22.6       -1.98   0.180      -2.91   0.001
lcarpcap    53.0       -1.70   0.459      -2.38   0.019

Pesaran-Yamagata adjusted delta = 17.05   p-value = 0.0000
```

How to read it

- **Cross-sectional dependence** is strong for every variable (the oil shocks hit every country), so CIPS is the unit-root test to use.
- **Unit roots.** CIPS does not reject a unit root in levels for any variable, and rejects it in first differences for all of them: **all four are I(1)**.
- **Slopes** are heterogeneous: the Δ test rejects. This will matter for the choice of cointegration test and of estimator.

22.3 The spurious regression in levels

```
x = df.set_index(["country", "year"])
lev = PanelOLS(x[y], sm.add_constant(x[xs]), entity_effects=True).fit(cov_type="clustered", cluster_entity=
  ↪ True)
print(lev.params[xs].round(3).to_frame("coef").join(lev.tstats[xs].round(1).rename("t")))
print(f"R-squared (within): {lev.rsquared_within:.3f}")
```

```
           coef    t
lincomep  0.662  4.3
lrpmg    -0.322 -2.6
lcarpcap -0.640 -6.6
R-squared (within): 0.840
```

How to read it

In levels, fixed effects gives large, highly significant elasticities and a high within R-squared. **This is exactly what a spurious regression looks like.** With I(1) variables, the t-statistics do not follow the t distribution, and trending series appear related whether or not they are. These numbers are only meaningful if the variables are cointegrated, which is the next test.

22.4 Test for cointegration

```
k = kao_test(df, "country", y, xs)
f = fisher_eg_test(df, "country", y, xs)
print(f"Kao ADF test (common slopes):              t* = {k['stat']:6.2f}   p-value = {k['p']:.3f}")
print(f"Fisher-Engle-Granger (country slopes):     Z  = {f['Z']:6.2f}   p-value = {f['Z_p']:.3f}")
print(f"                                           P  = {f['P']:6.2f}   p-value = {f['P_p']:.3f}   (chi2 with
  ↪ {2 * f['n']} df)")
```

```
Kao ADF test (common slopes):              t* =  -1.69   p-value = 0.045
```

```
Fisher-Engle-Granger (country slopes):     Z  =  -0.68   p-value = 0.249
                                           P  =  49.72   p-value = 0.064   (chi2 with 36 df)
```

How to read it

The null hypothesis of both tests is **no cointegration**.

- **Kao** pools the fixed-effects residuals and assumes the same slopes in every country. It rejects narrowly, at the 5% level.
- **Fisher–Engle–Granger** runs an Engle–Granger test for each country, with its own slopes, and combines the p-values. Its Z version does not reject ($p = 0.25$); its P version comes close only at the 10% level.

The tests disagree, and the pre-tests tell us which to believe: **the slopes are heterogeneous**, which violates Kao's assumption. A Kao test applied to heterogeneous panels can reject spuriously, because the pooled residuals mix different long-run relationships. The decision rules of Chapter 1 therefore uses the heterogeneous test here, and concludes that there is no cointegration.

This is a close call, and the paper must say so: report both tests, and add the Westerlund (2007) error-correction tests with bootstrap critical values, which allow for both heterogeneity and cross-sectional dependence (`xtwest` in Stata). With $T = 19$, every panel cointegration test has limited power.

22.5 Estimate the model in first differences

```python
d = df.copy()
for v in [y] + xs:
    d["d_" + v] = d.groupby("country")[v].diff()
d = d.dropna()
dy, dxs = "d_" + y, ["d_" + v for v in xs]
xd = d.set_index(["country", "year"])

fd_fe = PanelOLS(xd[dy], sm.add_constant(xd[dxs])).fit(cov_type="kernel", kernel="bartlett", bandwidth=2)
fd_tw = PanelOLS(xd[dy], sm.add_constant(xd[dxs]), time_effects=True).fit(cov_type="kernel", kernel="bartlett",
  ↪ bandwidth=2)
mg = mean_group(d, "country", "year", dy, dxs)
cce = cce_mg(d, "country", "year", dy, dxs)

res = {"Pooled, DK s.e.": (fd_fe.params[dxs], fd_fe.std_errors[dxs]), "Pooled + year FE": (fd_tw.params[dxs],
  ↪ fd_tw.std_errors[dxs]),
       "Mean group": (mg["table"]["coef"], mg["table"]["std err"]), "CCE mean group": (cce["table"]["coef"],
  ↪ cce["table"]["std err"])}
print(f"{'':20s}" + "".join(f"{v:>18s}" for v in dxs))
for name, (b, se) in res.items():
    print(f"{name:20s}" + "".join(f"{b[v]:9.3f} ({se[v]:.3f})" for v in dxs))
```

```
                        d_lincomep           d_lrpmg        d_lcarpcap
Pooled, DK s.e.         0.242 (0.091)   -0.252 (0.046)   -0.656 (0.058)
Pooled + year FE        0.200 (0.113)   -0.186 (0.037)   -0.684 (0.065)
Mean group              0.309 (0.162)   -0.212 (0.042)   -0.613 (0.076)
CCE mean group          0.290 (0.118)   -0.147 (0.052)   -0.823 (0.125)
```

How to read it

Differencing removes the country fixed effects (permanent differences in levels), so the pooled regression on differences needs no country dummies. Compare the elasticities with the spurious levels regression: **the income and price responses are much smaller in differences**, as one would expect when drivers can adjust only gradually. (The cars coefficient is similar, because the stock of cars and fuel use per car are linked almost mechanically in every year.)

- **Price.** A 10% rise in the real price of gasoline reduces consumption per car by roughly 1–3% within the year, depending on the estimator.
- **Income.** The short-run income elasticity is positive.
- **Cars per capita.** More cars means each car is driven less, so consumption per car falls.

With heterogeneous slopes and cross-sectional dependence, **CCE-MG is the preferred estimator**: it averages the country-specific elasticities and removes the common factors (the oil shocks) through the cross-sectional averages. Its standard errors are larger, because they reflect the cross-country dispersion of the elasticities.

22.6 Diagnostic test: residual cross-sectional dependence

```
fe_res = fd_fe.resids.unstack("country")
print(f"Pooled differences, residual CD:  {cd_test(fe_res)[0]:6.2f}   p-value = {cd_test(fe_res)[1]:.4f}")

bars = d.groupby("year")[[dy] + dxs].transform("mean").add_suffix("_bar")
dd = d.join(bars)
R = {}
for c, g in dd.groupby("country"):
    X = np.column_stack([np.ones(len(g)), g[dxs + list(bars.columns)]])
    b, *_ = np.linalg.lstsq(X, g[dy].to_numpy(), rcond=None)
    R[c] = pd.Series(g[dy].to_numpy() - X @ b, index=g["year"])
cd, p, r = cd_test(pd.DataFrame(R))
print(f"CCE-MG residual CD:               {cd:6.2f}   p-value = {p:.4f}")
```

```
Pooled differences, residual CD:    6.05   p-value = 0.0000
CCE-MG residual CD:                -0.68   p-value = 0.4948
```

How to read it

The pooled regression in differences leaves strong common shocks in the residuals. In the CCE-MG residuals the CD statistic is close to zero and no longer significant: the cross-sectional averages absorb the common oil-price and business-cycle shocks.

22.7 Build the results table for the paper

```
from scipy import stats
def star(b, se):
    p = 2 * stats.norm.sf(abs(b / se))
    return "***" if p < 0.01 else "**" if p < 0.05 else "*" if p < 0.10 else ""

print(f"{'':12s}" + "".join(f"{c:>18s}" for c in ["(1) Levels FE", "(2) Pooled Δ", "(3) MG Δ", "(4) CCE-MG Δ"])
  ↪ )
cols = [(lev.params[xs].values, lev.std_errors[xs].values)] + [(res[k][0].values, res[k][1].values) for k in (
  ↪ "Pooled, DK s.e.", "Mean group", "CCE mean group")]
for i, v in enumerate(xs):
    print(f"{v:12s}" + "".join(f"{b[i]:15.3f}{star(b[i], se[i]):3s}" for b, se in cols))
    print(f"{'':12s}" + "".join(f"{'(' + format(se[i], '.3f') + ')':>15s}   " for b, se in cols))
print("Column (1) is shown for comparison only: the variables are I(1) and not cointegrated, so it is spurious.
  ↪ ")
print(f"Kao t* = {k['stat']:.2f} (p = {k['p']:.3f}); Fisher-EG Z = {f['Z']:.2f} (p = {f['Z_p']:.3f}); Pesaran-
  ↪ Yamagata adj. delta = {s['delta_adj']:.2f} (p < 0.001)")
```

```
                (1) Levels FE      (2) Pooled Δ          (3) MG Δ      (4) CCE-MG Δ
lincomep             0.662***          0.242***          0.309*            0.290**
                    (0.154)           (0.091)           (0.162)           (0.118)
lrpmg                -0.322***         -0.252***         -0.212***         -0.147***
                    (0.123)           (0.046)           (0.042)           (0.052)
lcarpcap             -0.640***         -0.656***         -0.613***         -0.823***
                    (0.097)           (0.058)           (0.076)           (0.125)
Column (1) is shown for comparison only: the variables are I(1) and not cointegrated, so it is spurious.
Kao t* = -1.69 (p = 0.045); Fisher-EG Z = -0.68 (p = 0.249); Pesaran-Yamagata adj. delta = 17.05 (p < 0.001)
```

How to read it

Including the levels regression in column (1) shows the reader the spurious result the paper avoids. Column (4) is the preferred estimate.

22.8 Write the paper

How to write it: pre-tests

CIPS tests, which allow for the strong cross-sectional dependence indicated by the CD test, do not reject a unit root in levels for any variable but reject it in first differences. Because the Pesaran–Yamagata test rejects slope homogeneity, we test for cointegration with the Fisher-type combination of country-level Engle–Granger tests, which allows the long-run coefficients to differ across countries; it does not reject the null of no cointegration. The Kao test, which assumes homogeneous slopes, rejects at the 5 percent level. Given the heterogeneity of slopes, we treat the evidence for cointegration as weak and estimate the model in first differences.

How to write it: results

Table 4 reports short-run elasticities. In the common correlated effects mean group model (column 4), gasoline consumption per car rises with income and falls with the real price and with the number of cars per capita. The short-run income and price elasticities are much smaller than the coefficients of the levels regression in column 1, which, given the absence of cointegration, should not be interpreted.

Common mistakes

- Running a levels regression on I(1) variables without testing for cointegration, and interpreting the t-statistics.
- Choosing the cointegration test that gives the desired answer. Choose it from the pre-tests (homogeneous or heterogeneous slopes, dependence or not), and report the others.
- Calling short-run elasticities from a model in differences "long-run" effects.
- Ignoring cross-sectional dependence in the differenced model. Common shocks remain after differencing.

CHAPTER 23

Panel Local Projections

Applies when. You want the dynamic response of an outcome to a shock in a panel of units, horizon by horizon, with unit fixed effects

Overview. Use this chapter when you have a panel (states, countries, firms) and the question is how an outcome responds over the following periods to a shock or policy change in the unit. A panel local projection (Jordà 2005; Jordà, Schularick and Taylor 2015) runs one fixed-effects regression per horizon, so it needs no model of the whole system and handles unbalanced panels and nonlinearities easily. The chapter covers the panel structure, the effects (unit, time), clustered versus Driscoll-Kraay standard errors, robustness including a leave-one-unit-out check, and the write-up.

Data. Cigarette sales and prices, 46 U.S. states, 1963–1992

Software. Python · OES engine · statsmodels

23.1 The question and the model

How does the outcome of a unit evolve after a shock to that unit? For each horizon $h = 0, 1, \dots, H$ the panel local projection estimates

$$y_{i,t+h} = \alpha_{i,h} + \delta_{t,h} + \beta_h \, s_{i,t} + \gamma_h' x_{i,t} + u_{i,t+h},$$

where $\alpha_{i,h}$ is a unit effect, $\delta_{t,h}$ is an optional time effect, $s_{i,t}$ is the shock and $x_{i,t}$ holds the controls (lags of the outcome, of the shock and of other variables, and any variables ordered before the shock). The coefficient β_h is the impulse response at horizon h. Because every horizon is a separate fixed-effects regression, the method is robust to misspecification of the dynamics and does not require a balanced panel.

Four checks before you trust a panel local projection

1. **Is the shock plausibly exogenous to the unit's own outcome?** The shock should be a change that is not driven by anticipated changes in the outcome (a tax change, a policy reform, an external shock). A price that responds to expected demand is not a shock.
2. **Which effects?** Unit effects remove permanent differences between units. Time effects also remove everything common to all units in a period, so the response is identified from differences across units only, and the average effect of a common shock is lost.
3. **Which standard errors?** The horizon-h errors overlap across horizons and are correlated within a unit, so cluster by unit. If shocks hit all units at once, errors are also correlated across units, and Driscoll-Kraay errors (which need a long time dimension) or time effects are needed.
4. **Is T large enough?** With unit effects and lagged outcomes among the controls, a short panel has Nickell bias. Check that the response is stable to the number of lags.

How to write it: the model

We estimate the dynamic effect of a cigarette-price increase on sales with panel local projections (Jordà 2005). For each horizon $h = 0, \dots, 8$ years we regress log per-capita sales at $t + h$ on the change in the log real price at t, on the contemporaneous change in log income, on two lags of the price change, of income growth and of log sales, and on state fixed effects. The price change is scaled to one standard deviation. Standard errors are clustered by state.

23.2 Load the panel and check its structure

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
from statsmodels.tsa.stattools import adfuller

data = open("cigar.csv").read()
df = pd.read_csv("cigar.csv").sort_values(["state", "year"])
df["d_log_price"] = df.groupby("state").log_price.diff()
print(f"{df.state.nunique()} states, {df.year.min()}-{df.year.max()}, observations per state: {sorted(df.
  ↪ groupby('state').size().unique())}")
print(df[["log_sales", "log_price", "log_income", "d_log_price"]].describe().round(3).loc[["mean", "std", "min
  ↪ ", "max"]])
share = lambda s: np.mean([adfuller(g.dropna(), maxlag=2, autolag=None)[1] < .05 for _, g in df.groupby("state
  ↪ ")[s]])
print({c: round(share(c), 2) for c in ["log_sales", "log_price", "d_log_price"]})
```

```
46 states, 63-92, observations per state: [np.int64(30)]
      log_sales  log_price  log_income  d_log_price
mean      4.793     -0.106       4.545        0.011
std       0.225      0.152       0.210        0.060
min       3.978     -0.610       3.766       -0.117
max       5.697      0.364       5.117        0.268
{'log_sales': np.float64(0.0), 'log_price': np.float64(0.0), 'd_log_price': np.float64(0.13)}
```

How to read it

The panel is balanced: 46 states observed for 30 years (1963-92). Log sales and log price are persistent, so the shock used is the one-year change in log price (`d_log_price`, mean 0.011, s.d. 0.060). The per-state unit-root test rejects a unit root in none of the states for the levels and in only 13% for the change, but with 30 years per state and two lags the test has little power, so treat it as a warning about persistence, not proof. Because every state has all 30 years, no grid gaps need filling.

23.3 Estimate the impulse response

```
ops = [dict(op="diff", vars=["log_price", "log_income"], unit="state", time="year")]
spec = dict(y="log_sales", shock="d_log_price", contemp=["d_log_income"], unit="state", time="year",
            lags=2, horizon=8, effects="unit", cov="clustered", scale="sd", ops=ops)
r = oes.run("plp", data, spec)
oes.summary(r)
oes.show(r, "Impulse response")
oes.plot(r, "Response of log_sales", ncols=1, size=(6.5, 3))
```

```
Observations (h = 0): 1242
Units: 46
Periods: 30
Horizons: 8
Lags: 2
Peak response: -0.0369
Peak horizon: 3
Fixed effects: Unit
Impulse response by horizon
         Response  Std. err.  p-value  95% CI low  95% CI high  Obs.
Horizon
0         -0.0192     0.0015      0.0     -0.0221      -0.0163  1242
1         -0.0287     0.0014      0.0     -0.0315      -0.0260  1196
2         -0.0345     0.0018      0.0     -0.0380      -0.0311  1150
3         -0.0369     0.0018      0.0     -0.0404      -0.0333  1104
4         -0.0363     0.0020      0.0     -0.0402      -0.0323  1058
5         -0.0342     0.0021      0.0     -0.0383      -0.0301  1012
6         -0.0334     0.0025      0.0     -0.0382      -0.0285   966
7         -0.0295     0.0026      0.0     -0.0346      -0.0244   920
8         -0.0255     0.0028      0.0     -0.0310      -0.0200   874
Note: Response of log_sales to one standard deviation of the shock. Outcome: log_sales(t+h). Each row is a separate
↪ regression. Standard errors: clustered by unit.
```

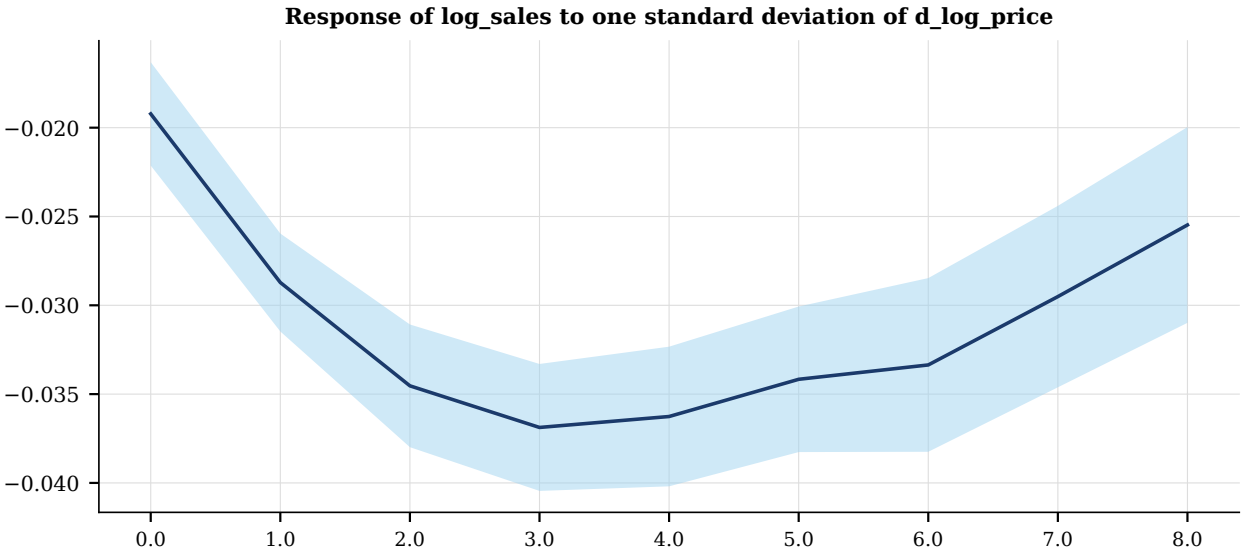


Figure 23.1. Estimate the impulse response: Response of log_sales to one standard deviation of d_log_price.

How to read it

Each row is a separate regression of log sales at t+h on the price shock, with state effects, two lags and the contemporaneous income change, scaled to a one-standard-deviation shock. Sales fall by 1.9% on impact (-0.0192, s.e. 0.0015), the response deepens to a peak of -0.0369 at horizon 3, and decays to -0.0255 by horizon 8. Every horizon is significant at 1%, and the 95% band stays well away from zero. The observation count drops from 1,242 to 874 as the horizon grows because later outcomes need more future years.

23.4 Choose the effects: pooled, unit or unit and time

```
cmp = {}
for name, eff in {"pooled": "none", "unit effects": "unit", "unit and time effects": "twoway"}.items():
    rows = oes.run("plp", data, dict(spec, effects=eff))["tables"][0]["rows"]
    cmp[name] = [rows[h][1] for h in range(9)]
print(pd.DataFrame(cmp, index=pd.Index(range(9), name="horizon")).loc[[0, 1, 2, 4, 6, 8]].round(4))
```

```
         pooled  unit effects  unit and time effects
horizon
0       -0.0191       -0.0192               -0.0227
1       -0.0290       -0.0287               -0.0228
2       -0.0348       -0.0345               -0.0238
4       -0.0382       -0.0363               -0.0180
6       -0.0386       -0.0334               -0.0136
8       -0.0332       -0.0255               -0.0080
```

How to read it

The effects choice matters for the shape. Pooled and unit-effects responses are almost identical (about -0.036 at horizon 4), so state-level differences in average sales do not drive the result. Adding year effects shrinks the response (-0.018 at horizon 4, -0.008 at horizon 8) because common national shocks, such as federal tax changes and health campaigns, move prices and sales together. Report which comparison you mean: unit effects give the within-state response to a price change; two-way effects give the response relative to other states in the same year.

23.5 Standard errors: clustered or Driscoll-Kraay

```
se = {}
for name, v in {"clustered by state": dict(cov="clustered"), "Driscoll-Kraay": dict(cov="dk")}.items():
    rows = oes.run("plp", data, dict(spec, **v))["tables"][0]["rows"]
    se[name] = [rows[h][2] for h in range(9)]
```

```
print(pd.DataFrame(se, index=pd.Index(range(9), name="horizon")).loc[[0, 1, 2, 4, 6, 8]].round(4))
```

```
         clustered by state  Driscoll-Kraay
horizon
0                    0.0015          0.0030
1                    0.0014          0.0029
2                    0.0018          0.0037
4                    0.0020          0.0081
6                    0.0025          0.0122
8                    0.0028          0.0185
```

How to read it

Driscoll-Kraay errors are two to six times larger than clustered errors and grow with the horizon (0.0081 versus 0.0020 at horizon 4, 0.0185 versus 0.0028 at horizon 8). Clustering by state allows arbitrary serial correlation within a state but assumes states are independent; Driscoll-Kraay also allows common shocks across states, which matters here because the years are shared. With only 30 periods, the Driscoll-Kraay errors are the more cautious choice; the response stays significantly negative through horizon 8 under both.

23.6 Robustness: lags, outcome, controls and leave-one-state-out

```
variants = {"baseline (2 lags)": {}, "1 lag": dict(lags=1), "4 lags": dict(lags=4), "no income control": dict(
  ↪ contemp=[]),
            "outcome: change since t-1": dict(outcome="diff"), "Driscoll-Kraay errors": dict(cov="dk")}
out = {}
for name, v in variants.items():
    rows = oes.run("plp", data, dict(spec, **v))["tables"][0]["rows"]
    out[name] = [rows[0][1], rows[4][1], rows[4][2], rows[8][1]]
print(pd.DataFrame(out, index=["h=0", "h=4", "s.e. h=4", "h=8"]).T.round(4))

loo = []
for s in df.state.unique():
    r_s = oes.run("plp", df[df.state != s], dict(spec, ops=ops))
    loo.append(r_s["tables"][0]["rows"][4][1])
print(f"Leave-one-state-out response at h=4: min {min(loo):.4f}, max {max(loo):.4f} (all 46 negative: {all(v <
  ↪ 0 for v in loo)})")
```

```
                             h=0     h=4  s.e. h=4     h=8
baseline (2 lags)         -0.0192 -0.0363    0.0020 -0.0255
1 lag                     -0.0197 -0.0391    0.0023 -0.0316
4 lags                    -0.0185 -0.0341    0.0024 -0.0082
no income control         -0.0177 -0.0332    0.0024 -0.0274
outcome: change since t-1 -0.0192 -0.0363    0.0020 -0.0255
Driscoll-Kraay errors     -0.0192 -0.0363    0.0081 -0.0255
Leave-one-state-out response at h=4: min -0.0369, max -0.0357 (all 46 negative: True)
```

How to read it

The impact and peak responses barely move: -0.018 to -0.020 at horizon 0 and -0.033 to -0.039 at horizon 4 across one lag, four lags and no income control. The far horizon is less stable (-0.008 with four lags against -0.032 with one), so quote the horizon 8 value with care. The change-since-t-1 outcome matches the baseline exactly because the regression includes a lag of the outcome, as expected. Dropping any one state leaves the horizon 4 response between -0.0369 and -0.0357, and all 46 runs are negative, so no single state drives it.

23.7 Write the results

How to write it: results

A one-standard-deviation increase in the annual change in log cigarette price (about 6 percentage points) lowers per-capita sales by 1.9% on impact. The response grows to a peak of 3.7% after three years (95% CI -4.0% to -3.3%, state-clustered) and is still 2.6% below its starting path after eight years. The estimates use local projections with state fixed effects, two lags of sales and the price change, and a contemporaneous income

control. The impact and peak responses are stable across lag length, outcome definition and leave-one-state-out checks. Driscoll-Kraay standard errors are larger but the response remains significant through horizon 8. With year effects the response is smaller (1.8% at horizon 4), so the result partly reflects national co-movement of prices and sales. With $T = 30$ and lagged outcomes among the regressors, fixed-effect estimates carry a small Nickell-type bias.

Report	What	Where
Model	Shock, outcome, controls, effects, lags, horizon, standard errors	Section 23.1
Data	Panel structure, unit-root checks	Section 23.2
Estimates	Impulse response table and figure with 95% bands	Section 23.3
Effects	Pooled, unit and two-way effects	Section 23.4
Standard errors	Clustered versus Driscoll-Kraay	Section 23.5
Robustness	Lags, outcome, controls, leave-one-unit-out	Section 23.6

References. Jordà (2005), *American Economic Review*; Jordà, Schularick and Taylor (2015), *Journal of International Economics*; Nickell (1981), *Econometrica*; Driscoll and Kraay (1998), *Review of Economics and Statistics*; Montiel Olea and Plagborg-Møller (2021), *Econometrica*; Baltagi, Griffin and Xiong (2000), *Review of Economics and Statistics*.

CHAPTER 24

Panel Threshold Regression (Hansen)

Applies when. The effect of a regressor may differ above and below a threshold in another variable, and you want to estimate where the threshold is

Overview. Use this chapter when theory suggests a regime change in a panel: deterrence that works only where enforcement is strong, debt that hurts growth only above a ratio, credit that matters only for small firms. Hansen's (1999) panel threshold regression estimates the threshold, tests whether it exists, builds a confidence interval for it and reports separate slopes in each regime. The chapter covers the setup, the bootstrap test, the interval, the second threshold, robustness and the text for the paper.

Data. North Carolina counties, 1981–1987 (Cornwell and Trumbull 1994): crime rate, probability of arrest, probability of conviction, average sentence and police per capita

Software. Python · OES engine · statsmodels

24.1 The question and the model

Does the slope change when a threshold variable crosses a value? With unit effects μ_i, a regressor x_{it}, controls z_{it} and a threshold variable q_{it}, the one-threshold model is

$$y_{it} = \mu_i + \beta_1' x_{it}\,\mathbf{1}(q_{it} \le \gamma) + \beta_2' x_{it}\,\mathbf{1}(q_{it} > \gamma) + \delta' z_{it} + e_{it}.$$

The unit effects are removed by the within transformation. For each candidate γ the model is linear in the slopes, so γ is estimated by least squares: the value that gives the smallest sum of squared residuals over a grid. Inference has two nonstandard parts. Under the null of no threshold, γ is not identified, so the usual F test fails and Hansen's bootstrap supplies the p-value. The confidence interval for γ is the set of values at which the likelihood-ratio statistic stays below a critical value, not a symmetric interval around the estimate.

Three checks before you trust a threshold model

1. **Is there a threshold at all?** Look at the bootstrap p-value of the test against a linear model first. If it does not reject, the regime estimates are descriptive.
2. **Is the threshold precisely located?** A wide likelihood-ratio interval or a flat likelihood means the data barely locate the threshold, and the regime slopes depend on where you put it.
3. **Is each regime well populated?** A regime with few observations or few units gives noisy slopes. The trimming fraction keeps every regime above a minimum share of the sample.

How to write it: the model

We estimate a panel threshold regression (Hansen 1999) in which the effect of the probability of arrest on the crime rate may differ across counties with low and high police presence. The crime rate and the probability of arrest are in logs, the model includes county fixed effects, and the probability of conviction and the average sentence enter with a common slope. The threshold variable is log police per capita. The threshold is searched over 100 grid points between the 5th and 95th percentiles of the threshold variable, the test against a linear model uses a bootstrap with 300 replications, and standard errors of the slopes are clustered by county.

24.2 Load the data and look at the threshold variable

```
import numpy as np
```

```
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("crime4.csv").read()
df = pd.read_csv("crime4.csv")
print(f"{df.county.nunique()} counties, {df.year.min()}-{df.year.max()}, observations per county: {sorted(df.
  ↪ groupby('county').size().unique())}")
print(df[["lcrmrte", "lprbarr", "lprbconv", "lavgsen", "lpolpc"]].describe().round(2).loc[["mean", "std", "min
  ↪ ", "max"]])
print(df.lpolpc.quantile([.05, .25, .5, .75, .95]).round(2).to_dict())
print("share of the variance of log police that is between counties:",
      round(df.groupby("county").lpolpc.transform("mean").var() / df.lpolpc.var(), 2))
```

```
90 counties, 81-87, observations per county: [np.int64(7)]
      lcrmrte  lprbarr  lprbconv  lavgsen  lpolpc
mean    -3.61    -1.27     -0.69     2.15   -6.49
std      0.57     0.42      0.61     0.27    0.53
min     -6.31    -2.83     -2.68     1.44   -7.69
max     -1.81     1.01      3.61     3.25   -3.34
{0.05: -7.17, 0.25: -6.73, 0.5: -6.54, 0.75: -6.32, 0.95: -5.74}
share of the variance of log police that is between counties: 0.78
```

How to read it

The panel is balanced: 90 counties observed for seven years (1981-87), 630 observations. The threshold variable, log police per capita, has a median of -6.54 and a 5th to 95th percentile range of -7.17 to -5.74. Seventy-eight percent of its variance is between counties, so the regimes are mostly defined by which counties are well policed, while the slope is identified from changes within counties over time. A regime that is almost a fixed county characteristic is worth knowing about before interpreting the result as an effect of changing police levels.

24.3 Test for a threshold and estimate it

```
spec = dict(y="lcrmrte", x=["lprbarr"], controls=["lprbconv", "lavgsen"], threshold="lpolpc", unit="county",
  ↪ time="year",
            trim=0.05, grid=100, boot=300)
r = oes.run("pthresh", data, spec)
oes.summary(r)
oes.show(r, "Tests for the number")
oes.show(r, "Estimated thresholds")
oes.plot(r, "Likelihood-ratio", ncols=1, size=(6.5, 3))
```

```
Observations: 630
Units: 90
Periods: 7
Threshold variable: lpolpc
Thresholds found: 1
Threshold estimate: -6.3825
Bootstrap p (F1): 0.027
Tests for the number of thresholds
                                                F statistic  Bootstrap p-value  10% critical value  5% critical
↪ value  1% critical value
Test
No threshold against one threshold                    31.496              0.027               21.45
↪ 24.76              33.36
One threshold against two (first threshold fixed)     12.683              0.190               16.01
↪ 18.87              26.49
Note: Bootstrap with 300 replications under the model without the threshold (Hansen 1999). The threshold is not
↪ identified under the null, so the bootstrap replaces the F distribution.

Estimated thresholds
             Estimate 95% confidence interval                              Note
Threshold
Threshold 1   -6.3825       [-6.4936, -5.756]  68.6% of observations at or below
Note: The interval is the set of thresholds at which the likelihood-ratio statistic is below 7.35 (Hansen 1999).
```

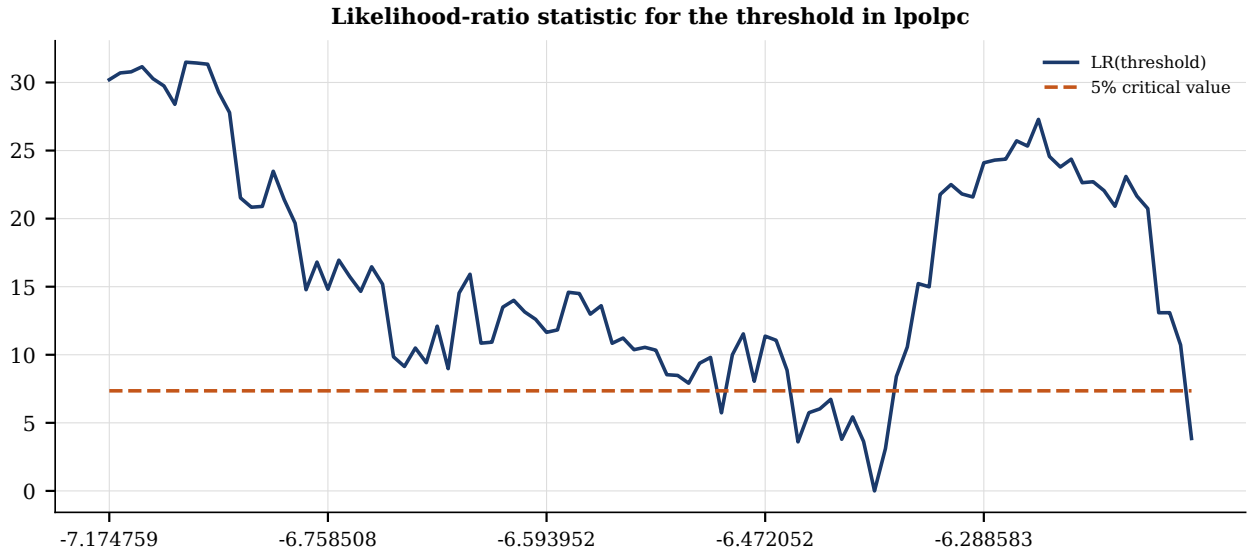


Figure 24.1. Test for a threshold and estimate it: Likelihood-ratio statistic for the threshold in lpolpc.

How to read it

The bootstrap test rejects a linear model: the F statistic is 31.5 and the bootstrap p-value is 0.027, with 5% and 1% critical values of 24.8 and 33.4. The threshold is -6.38 in log police per capita, which puts 69% of the observations (432 of 630) in the low-police regime. The 95% confidence interval runs from -6.49 to -5.76; its upper end is the upper edge of the grid (the 95th percentile of the threshold variable), so the likelihood-ratio statistic stays flat above the estimate and the data cannot rule out a higher threshold. The likelihood-ratio chart shows the same: a sharp dip at the estimate and a long flat stretch to the right.

24.4 Slopes by regime

```
oes.show(r, "Coefficients by regime")
oes.show(r, "Observations by regime")
lin = oes.run("fe", data, dict(y="lcrmrte", x=["lprbarr", "lprbconv", "lavgsen"], unit="county", time="year",
  ↪ effects="unit"))
print("Linear fixed-effects slope on lprbarr:", lin["tables"][0]["rows"][0][1:3])
```

```
Coefficients by regime (2 regimes)
                             Coef.  Std. err.       t  p-value  95% CI low  95% CI high
Variable
lprbarr | lpolpc <= -6.3825 -0.1528     0.1218 -1.2541   0.2104     -0.3921       0.0865
lprbarr | lpolpc > -6.3825  -0.2887     0.1048 -2.7535   0.0061     -0.4946      -0.0827
lprbconv                    -0.1361     0.0759 -1.7923   0.0737     -0.2852       0.0131
lavgsen                      0.0163     0.0429  0.3795   0.7045     -0.0680       0.1006
Note: Unit fixed effects removed; standard errors clustered by unit.

Observations by regime
                  Observations  Share (%)
Regime
lpolpc <= -6.3825          432       68.6
lpolpc > -6.3825           198       31.4

Linear fixed-effects slope on lprbarr: [-0.194081, 0.112254]
```

How to read it

The slope of the crime rate on the probability of arrest is -0.153 (standard error 0.122) in counties with police per capita at or below the threshold and -0.289 (0.105) above it. Only the high-police slope is significantly different from zero ($p = 0.006$); the 95% intervals of the two slopes overlap (-0.39 to 0.09 against -0.49 to -0.08). The linear fixed-effects slope is -0.194 (0.112), roughly the average of the two regimes. The pattern fits the view

that a higher arrest probability deters crime more where enforcement is stronger, but the data support it only moderately: the threshold is detected, while the difference between the two slopes is imprecisely estimated.

24.5 One threshold or two?

```
oes.show(r, "Tests for the number")
two = oes.run("pthresh", data, dict(spec, max_thresholds=2))
print([s["value"] for s in two["summary"] if s["label"] in ("Thresholds found", "Threshold estimate")])
```

```
Tests for the number of thresholds
                                                F statistic  Bootstrap p-value  10% critical value  5% critical
↪ value  1% critical value
Test
No threshold against one threshold                   31.496              0.027               21.45
↪ 24.76              33.36
One threshold against two (first threshold fixed)    12.683              0.190               16.01
↪ 18.87              26.49
Note: Bootstrap with 300 replications under the model without the threshold (Hansen 1999). The threshold is not
↪ identified under the null, so the bootstrap replaces the F distribution.

[1, -6.3825]
```

How to read it

The test of one threshold against two does not reject (F = 12.7, bootstrap p = 0.19, 5% critical value 18.9), so one threshold is enough and the engine reports the single-threshold model. The second threshold is searched with the first fixed at its estimate, which is valid because the first estimate converges faster.

24.6 Robustness: grid, trimming, bootstrap seed and controls

```
variants = {"baseline": {}, "trim 10%": dict(trim=0.10), "trim 15%": dict(trim=0.15), "grid 50": dict(grid=50),
  ↪  "grid 200": dict(grid=200),
           "other seed": dict(seed=2024), "no controls": dict(controls=[]), "add conviction slope": dict(x=["
  ↪ lprbarr", "lprbconv"], controls=["lavgsen"])}
out = {}
for name, v in variants.items():
    rr = oes.run("pthresh", data, dict(spec, **v))
    t = {s["label"]: s["value"] for s in rr["summary"]}
    c = [x for x in rr["tables"] if x["title"].startswith("Coefficients")][0]["rows"]
    arrest = [x[1] for x in c if x[0].startswith("lprbarr")]
    out[name] = [t["Threshold estimate"], t["Bootstrap p (F1)"], arrest[0], arrest[1]]
print(pd.DataFrame(out, index=["threshold", "p (no threshold)", "arrest slope, low", "arrest slope, high"]).T.
  ↪ round(3))
```

	threshold	p (no threshold)	arrest slope, low	arrest slope, high
baseline	-6.382	0.027	-0.153	-0.289
trim 10%	-6.381	0.003	-0.153	-0.293
trim 15%	-6.382	0.003	-0.152	-0.292
grid 50	-6.373	0.007	-0.150	-0.295
grid 200	-6.373	0.023	-0.150	-0.295
other seed	-6.382	0.020	-0.153	-0.289
no controls	-6.382	0.073	-0.044	-0.160
add conviction slope	-6.410	0.030	-0.164	-0.312

How to read it

The threshold estimate is stable: -6.37 to -6.41 under every variant. The test against a linear model rejects at 5% in every case except without controls (p = 0.073), where both arrest slopes are smaller in size (-0.044 and -0.160) but the high-police slope is still about three times the low-police slope. When the slope of the probability of conviction is also allowed to change with the regime, the arrest slopes become -0.164 and -0.312, so the difference is, if anything, larger. A grid of 50, 100 or 200 points, trimming of 5% to 15% and a different bootstrap seed change the threshold by at most 0.01 and the bootstrap p-value stays between 0.003 and 0.027.

```
drop = []
for y in sorted(df.year.unique()):
    rr = oes.run("pthresh", df[df.year != y].to_csv(index=False), dict(spec, boot=100))
    t = {s["label"]: s["value"] for s in rr["summary"]}
    drop.append((int(y), round(t["Threshold estimate"], 2), round(t["Bootstrap p (F1)"], 3)))
print(drop)
```

```
[(81, -6.67, 0.06), (82, -6.37, 0.0), (83, -6.38, 0.04), (84, -6.45, 0.03), (85, -5.81, 0.0), (86, -6.37, 0.03), (87,
↪  -5.72, 0.0)]
```

How to read it

Dropping one year at a time moves the threshold estimate between -6.67 and -5.72 (it is -6.37 to -6.45 in four of seven cases) and the bootstrap p-value between 0.00 and 0.06, with only 1981 pushing it above 5% (p = 0.06). The result does not depend on a single year, though with seven years a leave-one-year-out check has little power to distinguish. Report the range.

24.7 Write the results

How to write it: results

A test against a linear model rejects at the 5% level (Hansen bootstrap, $F = 31.5$, $p = 0.027$) and locates a threshold at -6.38 in log police per capita, with 69% of the county-years below it (95% confidence interval -6.49 to -5.76, bounded above by the edge of the search grid). Below the threshold the elasticity of the crime rate with respect to the probability of arrest is -0.15 (clustered standard error 0.12); above it, -0.29 (0.10). A second threshold is not supported ($p = 0.19$). The threshold is stable across grid size, trimming, the bootstrap seed and the choice of controls, and the high-police slope is larger in size in every variant, but the confidence intervals of the two slopes overlap, so we read the evidence as consistent with stronger deterrence where policing is heavier rather than as a precise estimate of the difference. Because 78% of the variation in police per capita is between counties, the regimes largely separate counties rather than periods.

Report	What	Where
Model	Regressors, threshold variable, controls, effects, grid, bootstrap	Section 24.1
Data	Distribution of the threshold variable	Section 24.2
Test and estimate	Bootstrap test, threshold, likelihood-ratio interval	Section 24.3
Slopes	Coefficients by regime, comparison with the linear model	Section 24.4
Number of thresholds	One against two	Section 24.5
Robustness	Grid, trimming, seed, controls, dropping years	Section 24.6

References. Hansen, B. (1999), "Threshold Effects in Non-Dynamic Panels: Estimation, Testing, and Inference," *Journal of Econometrics* 93. Hansen, B. (2000), "Sample Splitting and Threshold Estimation," *Econometrica* 68. Bai, J. (1997), "Estimating Multiple Breaks One at a Time," *Econometric Theory* 13. Cornwell, C. and Trumbull, W. (1994), "Estimating the Economic Model of Crime with Panel Data," *Review of Economics and Statistics* 76. Wang, Q. (2015), "Fixed-Effect Panel Threshold Model Using Stata," *Stata Journal* 15.

PART IV

Causal-Inference Designs

Overview

A causal estimate is only as good as its source of variation. This part therefore starts from the design, not the estimator. A researcher who can point to an instrument has a different problem from one with a running variable and a cutoff, and both differ from one who observes a policy introduced in a year, in some places and not in others. Chapter 1 sets out how the source of variation, and a few tests on the data, select the design.

The chapters cover instrumental variables, difference-in-differences with one adoption date and with several, synthetic controls, interrupted time series and regression discontinuity. Each uses data from a published study or a standard teaching dataset, so that the result can be compared with the literature. Each ends by listing the identifying assumption in words, because no test can establish it: parallel trends, exclusion, continuity at the cutoff, or the stability of a counterfactual.

Chapters at a glance

Ch.	Chapter	Use it when
25	Instrumental Variables	An instrument shifts the treatment
26	Difference-in-Differences	Many treated units, one adoption date
27	Staggered Difference-in-Differences	Many treated units, several adoption dates
28	Generalized Synthetic Control	Many treated units, pre-trends not parallel
29	Synthetic Control	One or a few treated units, many comparison units
30	Interrupted Time Series	One treated series, no comparison units
31	Regression Discontinuity	Treatment assigned by a cutoff

Further reading

Angrist and Pischke (2009) remains the clearest introduction to the logic of identification. On regression discontinuity, see Imbens and Lemieux (2008) and Lee (2008); on the density test, McCrary (2008). Synthetic control was introduced in Abadie and Gardeazabal (2003) and developed in Abadie et al. (2010); its generalization is Xu (2017). The recent difference-in-differences literature is summarised by Callaway and Sant'Anna (2021), Goodman-Bacon (2021) and Sun and Abraham (2021). Weak-instrument pre-tests and critical values are due to Stock and Yogo (2005), and the robust test to Anderson and Rubin (1949). Inference with grouped errors follows Bertrand et al. (2004).

CHAPTER 25

Instrumental Variables

Applies when. An instrument shifts the treatment

Overview. Use this chapter when the treatment is endogenous but a variable is available that shifts it and affects the outcome only through it. It estimates the return to schooling with Card's college-proximity instrument: OLS, the first stage and its strength, the reduced form, 2SLS, weak-instrument-robust inference, the endogeneity test, and what the estimate means.

Data. 3,010 U.S. men in 1976 (National Longitudinal Survey of Young Men, Card 1995)

Software. Python · linearmodels · statsmodels

25.1 The question and why OLS fails

What is the return to a year of schooling? A regression of log wages on years of education gives an answer, but not a causal one: people who study longer may also be more able or more motivated, and those traits raise wages on their own. Ability is not in the data, so OLS is biased (most likely upwards).

Card (1995) proposed an instrument: whether a man **grew up near a four-year college**. Living near a college lowers the cost of attending, so it raises schooling; the argument is that, given region and urban residence, it affects wages only through schooling.

$$\text{First stage: } educ_i = \pi\, nearc4_i + X_i'\gamma + v_i \qquad \text{Second stage: } \ln w_i = \beta\, \widehat{educ}_i + X_i'\delta + u_i$$

Variable	Role	Definition
`lwage`	outcome	Log hourly wage, 1976
`educ`	endogenous treatment	Years of schooling
`nearc4`	instrument	1 if a four-year college was in the county at age 14
`black`, `smsa`, `south`, `married`, `exper`	controls	Race, urban residence, region, marital status, experience

The three conditions

- **Relevance:** the instrument moves schooling. Testable: the first stage.
- **Exclusion:** proximity affects wages only through schooling. Not testable. Threat: families who live near colleges may differ in ways that also raise wages; that is why urban and regional controls matter.
- **Independence:** proximity is as good as random given the controls. Not testable. It must be argued.

A paper's IV section stands or falls on the argument for exclusion and independence, not on the regression output.

How to write it: identification

Because schooling is likely correlated with unobserved ability, we instrument years of education with an indicator for growing up near a four-year college, following Card (1995). Proximity lowers the cost of college attendance and so raises schooling; conditional on race, region, urban residence, marital status and experience, we assume it affects wages only through its effect on schooling.

25.2 Load the data and estimate OLS

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels.iv import IV2SLS

df = pd.read_csv("card.csv").dropna(subset=["lwage", "educ", "nearc4", "black", "smsa", "south", "married", "
  ↪ exper"])
controls = ["black", "smsa", "south", "married", "exper"]
W = sm.add_constant(df[controls])
print(f"{len(df):,} men; {df['nearc4'].mean():.0%} grew up near a four-year college")

ols = sm.OLS(df["lwage"], W.join(df["educ"])).fit(cov_type="HC1")
print(f"OLS return to schooling: {ols.params['educ']:.4f} (s.e. {ols.bse['educ']:.4f})")
```

```
3,003 men; 68% grew up near a four-year college
OLS return to schooling: 0.0712 (s.e. 0.0036)
```

> **How to read it**
>
> OLS says each additional year of schooling is associated with wages about 7% higher, very precisely estimated. Precision is not the problem; bias is. The IV estimate will tell us how much of this association is causal.

25.3 The first stage: is the instrument relevant?

```
fs = sm.OLS(df["educ"], W.join(df["nearc4"])).fit(cov_type="HC1")
F = fs.tvalues["nearc4"] ** 2
print(f"Effect of growing up near a college on schooling: {fs.params['nearc4']:.3f} years (s.e. {fs.bse['
  ↪ nearc4']:.3f})")
print(f"First-stage F statistic (robust): {F:.1f}")

partial = sm.OLS(df["educ"], W).fit()
print(f"Partial R-squared of the instrument: {1 - fs.ssr / partial.ssr:.4f}")
```

```
Effect of growing up near a college on schooling: 0.327 years (s.e. 0.081)
First-stage F statistic (robust): 16.5
Partial R-squared of the instrument: 0.0052
```

> **How to read it**
>
> Men who grew up near a four-year college completed about a third of a year more schooling, holding the controls fixed. The robust F statistic is about 16.5, above the Staiger–Stock rule of thumb of 10, so the instrument is not weak in the traditional sense. It is not strong either: it explains only half a percent of the variation in schooling left after the controls. Lee, McCrary, Moreira and Porter (2022) show that conventional 2SLS t-tests have correct size only with F above about 105, which is why the Anderson–Rubin test below matters.

25.4 The reduced form and 2SLS

```
rf = sm.OLS(df["lwage"], W.join(df["nearc4"])).fit(cov_type="HC1")
print(f"Reduced form: effect of proximity on log wages = {rf.params['nearc4']:.4f} (s.e. {rf.bse['nearc4']:.4f
  ↪ })")
print(f"Ratio reduced form / first stage = {rf.params['nearc4'] / fs.params['nearc4']:.4f}")

iv = IV2SLS(df["lwage"], W, df["educ"], df["nearc4"]).fit(cov_type="robust")
print(f"\n2SLS return to schooling: {iv.params['educ']:.4f} (s.e. {iv.std_errors['educ']:.4f}), 95% CI [{iv.
  ↪ conf_int().loc['educ', 'lower']:.3f}, {iv.conf_int().loc['educ', 'upper']:.3f}]")
```

```
Reduced form: effect of proximity on log wages = 0.0406 (s.e. 0.0161)
Ratio reduced form / first stage = 0.1242

2SLS return to schooling: 0.1242 (s.e. 0.0492), 95% CI [0.028, 0.221]
```

How to read it

- **Reduced form.** Men who grew up near a college earn about 4% more. Show this regression: it is the causal effect of the instrument itself, and if it is zero there is nothing to scale up.
- **2SLS.** With one instrument, the 2SLS estimate is exactly the reduced form divided by the first stage: about 0.12, a return of 12% per year of schooling, **larger** than OLS.
- **Precision.** The 2SLS standard error is more than ten times the OLS one. The confidence interval is wide.

Why would the IV estimate exceed OLS when ability bias pushes OLS up? Because the IV estimate is a **local average treatment effect** (LATE): the return for men whose schooling was changed by college proximity, typically from families for whom cost was a real barrier. Their return to schooling may be higher than average. Measurement error in schooling, which biases OLS towards zero, is another explanation.

25.5 Weak-instrument-robust inference

```
from scipy import stats
def ar_test(b0):
    r = sm.OLS(df["lwage"] - b0 * df["educ"], W.join(df["nearc4"])).fit(cov_type="HC1")
    return r.tvalues["nearc4"] ** 2

print(f"Anderson-Rubin test of no effect: F = {ar_test(0):.2f}, p-value = {stats.f.sf(ar_test(0), 1, len(df) -
  ↪ W.shape[1] - 1):.4f}")
grid = np.linspace(-0.1, 0.4, 1001)
crit = stats.f.ppf(0.95, 1, len(df) - W.shape[1] - 1)
inside = grid[[ar_test(b) <= crit for b in grid]]
print(f"95% Anderson-Rubin confidence set: [{inside.min():.3f}, {inside.max():.3f}]")
```

```
Anderson-Rubin test of no effect: F = 6.34, p-value = 0.0119
95% Anderson-Rubin confidence set: [0.032, 0.254]
```

How to read it

The Anderson–Rubin test has the correct size **however weak the instrument**. It tests each candidate value of the return b_0 by asking whether the instrument still predicts $\ln w - b_0 \cdot educ$; the confidence set collects the values that are not rejected. Here it rejects a zero return, and the 95% set runs from about 0.03 to 0.25: wider than the conventional 2SLS interval, but excluding zero. This is the interval to report when the first stage is not overwhelmingly strong.

25.6 Is schooling really endogenous?

```
df["v_hat"] = fs.resid
cf = sm.OLS(df["lwage"], W.join(df[["educ", "v_hat"]])).fit(cov_type="HC1")
print(f"Control-function (Durbin-Wu-Hausman) test: t = {cf.tvalues['v_hat']:.2f}, p-value = {cf.pvalues['v_hat
  ↪ ']:.3f}")
print(f"(OLS {ols.params['educ']:.3f} against 2SLS {iv.params['educ']:.3f})")
```

```
Control-function (Durbin-Wu-Hausman) test: t = -1.13, p-value = 0.259
(OLS 0.071 against 2SLS 0.124)
```

How to read it

The test adds the first-stage residual to the OLS regression: if schooling were exogenous, the residual would add nothing. It does not reject. That does not prove schooling is exogenous: with an imprecise 2SLS estimate the test has little power, and the gap between 0.07 and 0.12 is economically large. With one instrument, the exclusion restriction cannot be tested at all (no overidentification test); Card's full dataset adds proximity to a two-year college, which allows a Hansen J test.

25.7 Write the paper

How to write it: results

OLS estimates imply a return to schooling of about 7 percent per year. Instrumenting schooling with proximity to a four-year college, the 2SLS estimate is about 12 percent (robust standard error 0.05). The first stage is significant, with a robust F statistic of 16.5; because conventional 2SLS inference can be unreliable at this instrument strength, we also report the Anderson–Rubin 95 percent confidence set, which excludes zero. The 2SLS estimate is a local average treatment effect for men whose schooling was affected by college proximity, and its exceeding OLS is consistent with higher returns for individuals facing cost barriers to college, or with measurement error in schooling.

Common mistakes

- Treating the first-stage F above 10 as the end of the weak-instrument question. Report Anderson–Rubin intervals.
- Not reporting the first stage and the reduced form.
- Arguing exclusion with a statistical test. With one instrument there is none; even with several, the Hansen test only checks that instruments agree.
- Interpreting 2SLS as an average effect for everyone. It is local to the compliers.
- Computing the second stage by hand with OLS on fitted values: the standard errors are wrong. Use an IV routine.

CHAPTER 26

Difference-in-Differences

Applies when. Many treated units, one adoption date

Overview. Use this chapter when many units are treated at the same date, others are never treated, and the pre-treatment trends are parallel. It takes one real policy from the 2×2 comparison to the event study, the regression table and the paper, with the checks a referee will ask for.

Data. Castle Doctrine (stand-your-ground) laws, 13 states adopting in 2007 and 29 never-treated states, 2000–2010

Software. Python · linearmodels · statsmodels

26.1 The question and the design

Between 2005 and 2010 many U.S. states passed **Castle Doctrine** or "stand-your-ground" laws, which remove the duty to retreat before using lethal force in self-defence. Supporters argued they would deter crime; critics that they would escalate confrontations. **Did the laws change homicide rates?** This chapter uses the 13 states that adopted a law in 2007 and the 29 states that never did, from the data of Cheng and Hoekstra (2013). States adopting in other years are left out, so every treated state shares one adoption date. (Chapter 27 uses all of them.)

The difference-in-differences estimate compares the change in homicide in adopting states with the change in non-adopting states over the same years:

$$\ln h_{st} = \alpha_s + \lambda_t + \tau D_{st} + \varepsilon_{st}$$

Symbol	Variable in the file	Definition
$\ln h_{st}$	`l_homicide`	Log homicides per 100,000 population
D_{st}	`castle_law`	1 in a treated state from the year its law took effect
α_s, λ_t	—	State and year fixed effects
τ	—	The DiD estimate: the average effect on the treated (ATT)

The identifying assumption

τ is causal under **parallel trends**: without the laws, homicide in the adopting states would have changed by the same amount as in the non-adopting states. The levels may differ (the state effects absorb that); the *changes* must not. The assumption cannot be tested after 2007, but its implication for the years before 2007 can, and the event study does so.

How to write it: design

We compare homicide rates in the 13 states that enacted Castle Doctrine laws in 2007 with those in states that did not enact such laws between 2000 and 2010, using a difference-in-differences design with state and year fixed effects. The identifying assumption is that, absent the laws, homicide rates in adopting and non-adopting states would have followed parallel trends.

26.2 Load the data and look at the two groups

```
import pandas as pd
```

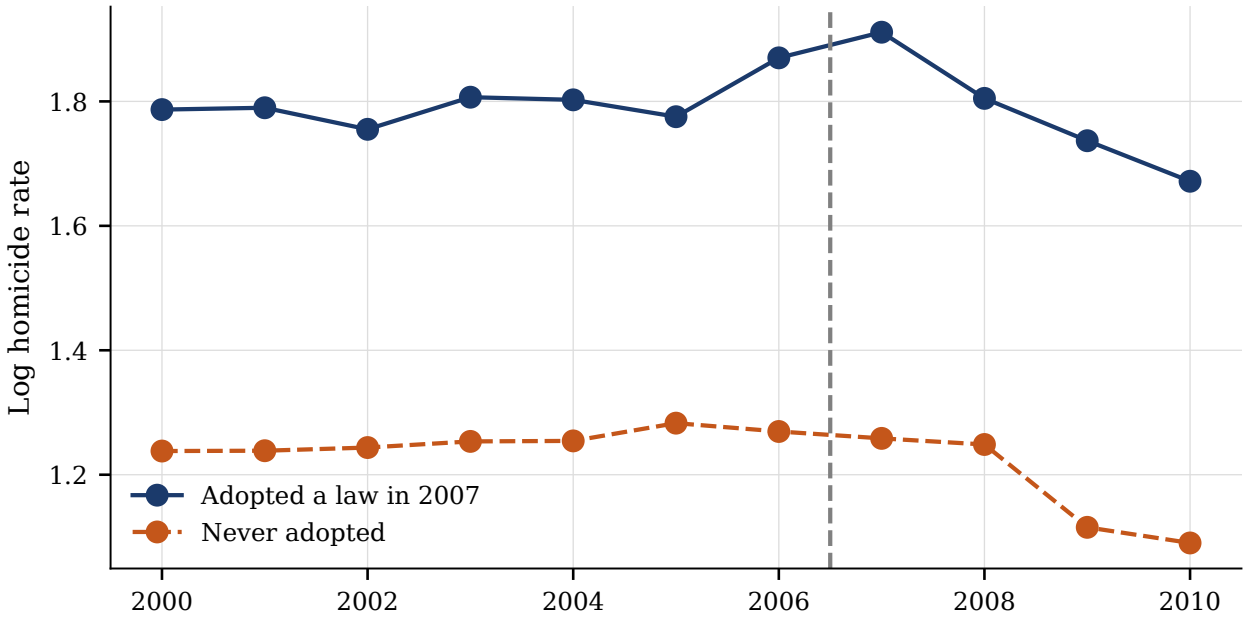


Figure 26.1. Load the data and look at the two groups.

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from linearmodels.panel import PanelOLS

df = pd.read_csv("castle_2007.csv")
treated = df.groupby("state")["castle_law"].max().rename("treated_state")
df = df.join(treated, on="state")
print(f"{df['state'].nunique()} states ({treated.sum()} treated), {df['year'].min()}-{df['year'].max()}, {len(
  ↪ df)} state-years")

means = df.groupby(["year", "treated_state"])["l_homicide"].mean().unstack()
fig, ax = plt.subplots()
ax.plot(means.index, means[1], marker="o", label="Adopted a law in 2007")
ax.plot(means.index, means[0], marker="o", label="Never adopted")
ax.axvline(2006.5, color="grey", ls="--")
ax.set_ylabel("Log homicide rate"); ax.legend()
```

```
42 states (13 treated), 2000-2010, 462 state-years
```

> **How to read it**
>
> Adopting states have higher homicide rates throughout: that difference in levels is what the state fixed effects remove. What matters is whether the two lines move **in parallel before 2007**. By eye they track each other reasonably closely, with year-to-year noise. The event study turns this impression into a test.

26.3 The 2×2 difference-in-differences

```
df["post"] = (df["year"] >= 2007).astype(int)
cell = df.groupby(["treated_state", "post"])["l_homicide"].mean().unstack()
cell.index = ["Never adopted", "Adopted in 2007"]; cell.columns = ["2000-2006", "2007-2010"]
cell["change"] = cell["2007-2010"] - cell["2000-2006"]
print(cell.round(4))
print(f"\nDifference-in-differences: {cell.loc['Adopted in 2007', 'change'] - cell.loc['Never adopted', '
  ↪ change']:.4f}")
```

```
                2000-2006  2007-2010  change
Never adopted      1.2544     1.1782 -0.0762
Adopted in 2007    1.7981     1.7811 -0.0169
```

```
Difference-in-differences: 0.0593
```

How to read it

This table is the whole idea of DiD. Each group's change from before to after removes its permanent level; the difference between the two changes removes everything common to both groups in the later years (national crime trends, the recession). What remains is attributed to the law. With one adoption date and a balanced panel, this number is exactly the two-way fixed-effects estimate in the next step.

26.4 Estimate the two-way fixed-effects model

```
panel = df.set_index(["state", "year"])
twfe = PanelOLS.from_formula("l_homicide ~ castle_law + EntityEffects + TimeEffects", panel).fit(cov_type="
  ↪ clustered", cluster_entity=True)
print(twfe.summary.tables[1])
print(f"Clusters (states): {twfe.entity_info['total']:.0f}")
print(f"Effect in percent: {100 * (np.exp(twfe.params['castle_law']) - 1):.1f}%")
```

```
                          Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
castle_law     0.0593     0.0801     0.7397     0.4599     -0.0982      0.2167
==============================================================================
Clusters (states): 42
Effect in percent: 6.1%
```

How to read it

- **Estimate.** The laws are associated with homicide rates about 6% higher, but the confidence interval is wide and includes zero: in this subsample the effect is not statistically significant.
- **Clustered standard errors.** The treatment varies at the state level and homicide is serially correlated within states, so standard errors must be clustered by state (Bertrand, Duflo and Mullainathan 2004). With 42 clusters this is fine; with fewer than about 30, use the wild cluster bootstrap.

26.5 Event study: test the pre-trends and trace the dynamics

```
df["rel"] = np.where(df["treated_state"] == 1, df["year"] - 2007, np.nan)
leads_lags = [k for k in range(-5, 4) if k != -1]          # -1 is the base year
for k in leads_lags:
    name = f"e_m{-k}" if k < 0 else f"e_p{k}"
    df[name] = (df["rel"] == k).astype(int)
df["e_m5"] = (df["rel"] <= -5).astype(int)                  # bin the distant leads
terms = [f"e_m{-k}" if k < 0 else f"e_p{k}" for k in leads_lags]
es = PanelOLS.from_formula("l_homicide ~ " + " + ".join(terms) + " + EntityEffects + TimeEffects",
                           df.set_index(["state", "year"])).fit(cov_type="clustered", cluster_entity=True)
co, se = es.params[terms], es.std_errors[terms]

fig, ax = plt.subplots()
ks = leads_lags + [-1]
ax.errorbar(leads_lags, co, yerr=1.96 * se, fmt="o", capsize=3)
ax.scatter([-1], [0], color="grey")
ax.axhline(0, color="black", lw=.8); ax.axvline(-0.5, color="grey", ls="--")
ax.set_xlabel("Years relative to adoption (2007 = 0)"); ax.set_ylabel("Effect on log homicide")

leads = [t for t in terms if t.startswith("e_m")]
w = es.wald_test(formula=", ".join(f"{t} = 0" for t in leads))
print(f"Joint test that the leads are zero: chi2({w.df}) = {w.stat:.2f}   p-value = {w.pval:.3f}")
```

```
Joint test that the leads are zero: chi2(4) = 6.71   p-value = 0.152
```

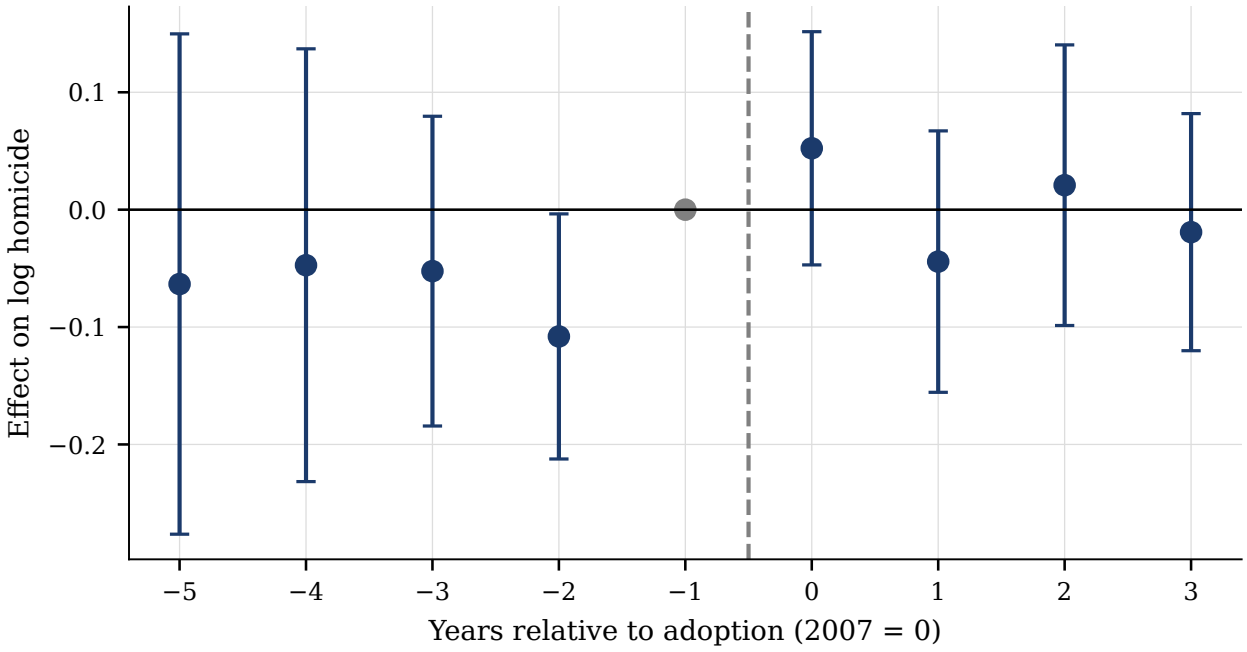


Figure 26.2. Event study: test the pre-trends and trace the dynamics.

How to read it

Each point is the difference between adopting and non-adopting states in that year, relative to 2006 (year –1).

- **Leads (before adoption).** They fluctuate around zero, and the joint test does not reject that they are all zero. There is no evidence that adopting states were already on a different path before 2007. That is the implication of parallel trends that can be checked.
- **Lags (after adoption).** The estimated effects after 2007 are small and imprecise: no clear sign of a jump in homicide among this group of states.

An insignificant pre-trend test is not proof of parallel trends: the test may simply lack power. A referee may ask for a sensitivity analysis (Rambachan and Roth 2023), which shows how large a violation of parallel trends would have to be to overturn the conclusion.

26.6 Robustness: placebo date and a different comparison

```
pre = df[df["year"] <= 2006].copy()
pre["fake"] = ((pre["treated_state"] == 1) & (pre["year"] >= 2004)).astype(int)
plac = PanelOLS.from_formula("l_homicide ~ fake + EntityEffects + TimeEffects", pre.set_index(["state", "year"
  ↪ ])).fit(cov_type="clustered", cluster_entity=True)
print(f"Placebo law in 2004 (pre-period only): {plac.params['fake']:.4f} (s.e. {plac.std_errors['fake']:.4f}),
  ↪  p-value {plac.pvalues['fake']:.3f}")

from causal_tests import event_study       # helper file on this site: examples/causal_tests.py
cs = event_study(df, unit="state", time="year", y="l_homicide", treat="castle_law")
print(f"Event-study average of post years (base 2006): {cs['att']['est']:.4f} (bootstrap s.e. {cs['att']['se
  ↪ ']:.4f})")
```

```
Placebo law in 2004 (pre-period only): 0.0059 (s.e. 0.0792), p-value 0.941
Event-study average of post years (base 2006): 0.0024 (bootstrap s.e. 0.0357)
```

How to read it

- **Placebo date.** Pretending the law took effect in 2004 and using only pre-2007 data should give no effect. It does not, which supports the comparison.

- **Different base period.** The event-study average uses only 2006 as the base year, while the TWFE estimate uses the average of all pre-treatment years. The two answers differ because individual years are noisy, but both point estimates are small relative to their uncertainty. Report both and say so.

26.7 Write the paper

How to write it: results

Table 1 reports the difference-in-differences estimates. Homicide rates in states that enacted Castle Doctrine laws in 2007 rose by about 6 percent relative to non-adopting states, but the estimate is imprecise and not statistically different from zero (standard errors clustered by state). Event-study estimates show no differential pre-trends: the coefficients for the years before adoption are jointly insignificant, and a placebo law dated to 2004 has no effect.

Common mistakes

- Using this two-way fixed-effects regression when units adopt at **different** dates. The estimate can then be badly biased; see Chapter 27.
- Not clustering at the level of treatment assignment.
- Treating insignificant leads as proof of parallel trends. Show the event-study plot, and consider a sensitivity analysis.
- Controlling for variables that the policy itself affects (bad controls).
- Ignoring anticipation: if behaviour changes when a law is announced, the base period must precede the announcement.

CHAPTER 27

Staggered Difference-in-Differences

Applies when. Many treated units, several adoption dates

Overview. Use this chapter when units adopt a policy at different dates. The conventional two-way fixed-effects regression then compares newly treated units with already-treated ones and can give a badly misleading answer. The chapter shows why, and estimates the effect with heterogeneity-robust methods: Callaway–Sant'Anna group-time effects and the imputation estimator.

Data. Castle Doctrine laws, 21 states adopting between 2006 and 2010, 50 states, 2000–2010

Software. Python · linearmodels · statsmodels

27.1 The question and why timing matters

The question is the one in Chapter 26: **did stand-your-ground laws change homicide rates?** But now all 21 adopting states are used, and they passed their laws in different years. That is the usual situation in policy evaluation: minimum-wage increases, Medicaid expansions, carbon taxes and central-bank reforms all arrive at different dates in different places.

The tempting regression is the same as before:

$$\ln h_{st} = \alpha_s + \lambda_t + \tau^{TWFE} D_{st} + \varepsilon_{st}$$

Why the conventional regression fails

Goodman-Bacon (2021) showed that τ^{TWFE} is a weighted average of every 2×2 DiD comparison in the data. Some of those comparisons use states that adopted **earlier** as the control group for states that adopt **later**. If the effect of the law grows or changes over time, the earlier adopters' outcomes are still moving because of their own treatment, and subtracting them removes part of the effect, or even flips its sign. Some weights can be negative. The fix is to estimate the effect for each adoption cohort separately, against clean controls, and then average: the **group-time ATT** of Callaway and Sant'Anna (2021).

$$ATT(g,t) = E\left[Y_t - Y_{g-1} \mid G = g\right] - E\left[Y_t - Y_{g-1} \mid \text{never treated}\right]$$

How to write it: design

States enacted Castle Doctrine laws at different dates between 2006 and 2010. Because two-way fixed-effects estimates are biased under staggered adoption when treatment effects vary over time or across cohorts (Goodman-Bacon 2021; de Chaisemartin and D'Haultfœuille 2020), we estimate cohort-specific average treatment effects on the treated, using never-treated states as controls, and aggregate them following Callaway and Sant'Anna (2021).

27.2 Load the data and look at the cohorts

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from linearmodels.panel import PanelOLS
from causal_tests import event_study        # helper file: causal-inference-guide/examples/causal_tests.py

df = pd.read_csv("castle.csv")
```

```
first = df[df["castle_law"] == 1].groupby("state")["year"].min()
df["cohort"] = df["state"].map(first)                        # adoption year, NaN if never treated
print(f"{df['state'].nunique()} states, {df['year'].min()}-{df['year'].max()}")
print("States by adoption year:", first.value_counts().sort_index().to_dict(), f"never treated: {df['cohort'].
  ↪ isna().groupby(df['state']).first().sum()}")
```

```
50 states, 2000-2010
States by adoption year: {2006: 1, 2007: 13, 2008: 4, 2009: 2, 2010: 1} never treated: 29
```

How to read it

Five adoption cohorts, with most states adopting in 2007, and 29 states that never adopt. The never-treated states are the clean comparison group. Without them, the not-yet-treated states could serve instead.

27.3 The conventional two-way fixed-effects estimate

```
panel = df.set_index(["state", "year"])
twfe = PanelOLS.from_formula("l_homicide ~ castle_law + EntityEffects + TimeEffects", panel).fit(cov_type="
  ↪ clustered", cluster_entity=True)
print(f"TWFE: {twfe.params['castle_law']:.4f} (s.e. {twfe.std_errors['castle_law']:.4f}), p-value {twfe.
  ↪ pvalues['castle_law']:.3f}")
```

```
TWFE: 0.0694 (s.e. 0.0581), p-value 0.233
```

How to read it

The TWFE estimate is about 0.07: homicide about 7% higher after adoption, though not significant with clustered standard errors. This is the number many papers would report. Under staggered adoption it is not a well-defined average of the effects, so it has to be checked against estimators that are.

27.4 Group-time effects: one clean comparison per cohort and year

```
W = df.pivot(index="state", columns="year", values="l_homicide")
never = df.loc[df["cohort"].isna(), "state"].unique()
rows = []
for g in sorted(first.unique()):
    gu = first.index[first == g]
    for t in W.columns:
        if t == g - 1:
            continue
        att = (W.loc[gu, t] - W.loc[gu, g - 1]).mean() - (W.loc[never, t] - W.loc[never, g - 1]).mean()
        rows.append(dict(cohort=g, year=t, e=t - g, att=att, n=len(gu)))
gt = pd.DataFrame(rows)
print(gt.pivot(index="cohort", columns="year", values="att").round(3).to_string())
```

```
year     2000   2001   2002   2003   2004   2005   2006   2007   2008   2009   2010
cohort
2006    0.176  0.116  0.134  0.120  0.120    NaN  0.219  0.297  0.270  0.262  0.232
2007   -0.052 -0.049 -0.089 -0.047 -0.052 -0.108    NaN  0.052 -0.044  0.021 -0.019
2008   -0.254 -0.078 -0.213 -0.109 -0.134  0.016 -0.145    NaN -0.208  0.126  0.014
2009   -0.284 -0.315 -0.069  0.042 -0.016  0.126  0.067 -0.037    NaN  0.222  0.034
2010   -0.507  0.021 -0.743 -0.134 -0.145 -0.694 -0.081 -0.463 -0.103    NaN -0.211
```

How to read it

Each cell is a separate 2×2 comparison: cohort g against the never-treated states, between year t and the year before adoption, $g-1$. Cells before adoption are placebo comparisons and should be near zero; cells after adoption are effects. Nothing in this table uses an already-treated state as a control. The cohorts with one or two states (2006, 2009, 2010) are noisy, so the cells are aggregated next.

27.5 Aggregate: the event study and the overall ATT

```
es = event_study(df, unit="state", time="year", y="l_homicide", treat="castle_law")
ev = pd.DataFrame(es["event"])
```

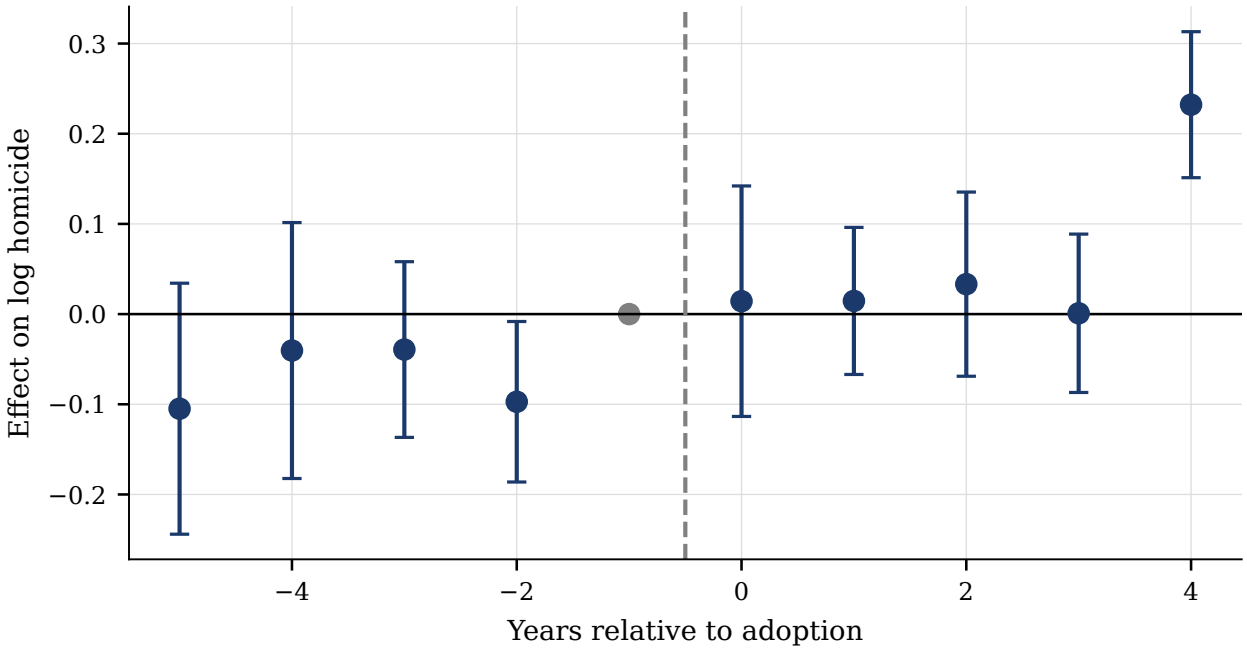


Figure 27.1. Aggregate: the event study and the overall ATT.

```
fig, ax = plt.subplots()
ax.errorbar(ev["e"], ev["est"], yerr=1.96 * ev["se"], fmt="o", capsize=3)
ax.scatter([-1], [0], color="grey"); ax.axhline(0, color="black", lw=.8); ax.axvline(-0.5, color="grey", ls="--
  ↪ ")
ax.set_xlabel("Years relative to adoption"); ax.set_ylabel("Effect on log homicide")
print(ev.round(4).to_string(index=False))
print(f"\nPre-trend test (leads -5 to -2): chi2({es['pretrend']['df']}) = {es['pretrend']['stat']:.2f}, p-
  ↪ value = {es['pretrend']['p']:.3f}")
print(f"Overall ATT (Callaway-Sant'Anna): {es['att']['est']:.4f} (bootstrap s.e. {es['att']['se']:.4f})")
print(f"TWFE for comparison:              {es['twfe']['est']:.4f} (s.e. {es['twfe']['se']:.4f})")
```

```
 e     est     se
-5 -0.1049 0.0710
-4 -0.0404 0.0724
-3 -0.0393 0.0497
-2 -0.0972 0.0454
 0  0.0143 0.0652
 1  0.0146 0.0416
 2  0.0332 0.0521
 3  0.0009 0.0448
 4  0.2322 0.0413

Pre-trend test (leads -5 to -2): chi2(4) = 8.51, p-value = 0.074
Overall ATT (Callaway-Sant'Anna): 0.0194 (bootstrap s.e. 0.0380)
TWFE for comparison:              0.0694 (s.e. 0.0558)
```

How to read it

- **Event study.** The group-time effects are averaged by years since adoption, weighting cohorts by size. Standard errors come from a cluster bootstrap over states.
- **Pre-trends.** The leads are jointly insignificant at 5% ($p \approx 0.07$). Not a clean pass: a sensitivity analysis is advisable.
- **The overall ATT** is about 0.02, a third of the TWFE estimate and far from significance.
- **Look at the leads.** They are all negative, around –0.04 to –0.10: relative to the never-treated states, homicide in adopting states was *lower* several years before adoption than in the year just before it. In other words, it was already rising. This estimator measures everything against the year before adoption, so it does not count that earlier rise as an effect of the law.

- **The large estimate four years after adoption** rests on the single 2006 adopter (Florida) and should not be over-interpreted.

27.6 Robustness: the imputation estimator

```
# state and year effects estimated on untreated observations only, then Y(0) imputed for treated observations
u = df[df["castle_law"] == 0].copy()
mu = u["l_homicide"].mean()
a = (u.groupby("state")["l_homicide"].mean() - mu)
for _ in range(50):                                    # alternate to get two-way effects on an
  ↪ unbalanced sample
    lam = (u["l_homicide"] - u["state"].map(a) - mu).groupby(u["year"]).mean()
    a = (u["l_homicide"] - u["year"].map(lam) - mu).groupby(u["state"]).mean()
tr = df[df["castle_law"] == 1].copy()
tr["y0"] = mu + tr["state"].map(a) + tr["year"].map(lam)
tr["gap"] = tr["l_homicide"] - tr["y0"]
print(f"Imputation estimator (Borusyak, Jaravel and Spiess): ATT = {tr['gap'].mean():.4f} over {len(tr)}
  ↪ treated state-years")
print(tr.assign(e=tr["year"] - tr["cohort"]).groupby("e")["gap"].mean().round(4).to_string())
```

```
Imputation estimator (Borusyak, Jaravel and Spiess): ATT = 0.0669 over 74 treated state-years
e
0.0    0.0727
1.0    0.0627
2.0    0.0825
3.0    0.0409
4.0    0.1133
```

How to read it

The imputation estimator fits state and year effects **only on untreated observations**, predicts what each treated state-year would have been without the law, and averages the gaps. Like Callaway–Sant'Anna, it never uses treated observations as controls, so it is free of the staggered-adoption bias.

It gives about 0.07, **close to TWFE**. So in this dataset the bias from comparing late with early adopters is small. The real divergence is between estimators that use **all** pre-treatment years as the baseline (TWFE, imputation: about 0.07) and the one that uses **only the year before adoption** (Callaway–Sant'Anna: about 0.02). The leads explain it: homicide in adopting states was already rising relative to controls before the laws, and an all-years baseline attributes part of that rise to the law.

The honest conclusion: the estimated effect depends on how the pre-trend is handled, and none of the estimates is statistically significant. Report all three, show the event study, and add a sensitivity analysis for violations of parallel trends (Rambachan and Roth 2023). (Standard errors for the imputation estimator are best obtained with the `did_imputation` package in Stata or a cluster bootstrap.)

27.7 Write the paper

How to write it: results

Because states adopted Castle Doctrine laws at different dates, we report heterogeneity-robust estimators alongside the two-way fixed-effects estimate. The two-way fixed-effects and imputation estimators, which use all pre-adoption years as the baseline, both indicate an increase in homicide of about 7 percent; the Callaway–Sant'Anna estimator, which uses the year before adoption as the baseline, indicates about 2 percent. None of the estimates is statistically significant. The pre-adoption event-study coefficients are negative and jointly significant at the 10 but not the 5 percent level, suggesting that homicide was already rising in adopting states; the results are therefore sensitive to how this pre-trend is treated.

Common mistakes

- Reporting only the TWFE coefficient under staggered adoption.

- Running a conventional event-study regression with relative-time dummies under staggered adoption: it suffers from the same contamination (Sun and Abraham 2021). Use cohort-specific estimates.
- Not saying which controls are used: never-treated or not-yet-treated. They rest on different assumptions; report both if possible.
- Ignoring small cohorts. A cohort of one state drives the event-time estimates where it is alone.

CHAPTER 28

Generalized Synthetic Control

Applies when. Many treated units, pre-trends not parallel

Overview. Use this chapter when the event study shows that treated and untreated units were already on different paths before treatment, so plain difference-in-differences is not credible. Generalized synthetic control (Xu 2017) lets units follow their own trends through latent common factors, estimates the counterfactual for each treated unit, and checks the fit before treatment.

Data. Castle Doctrine laws and the murder rate, 50 U.S. states, 2000–2010

Software. Python · linearmodels · statsmodels

28.1 The question and why DiD fails here

The question: **did stand-your-ground laws change the murder rate?** Chapter 26 used homicides from vital statistics; this chapter uses the murder rate from police reports (FBI Uniform Crime Reports), in logs, for the same 50 states and the same laws. A second measure of the same outcome is a standard robustness check, and here it raises a problem the first measure did not.

The difference-in-differences model assumes that, without treatment, every state's outcome would have moved by the same amount each year. Generalized synthetic control relaxes that:

$$Y_{it}(0) = \alpha_i + \xi_t + \lambda_i' f_t + \varepsilon_{it}$$

The f_t are a small number of unobserved **common factors** (for example, national trends in public-sector hiring or crime policy), and the loadings λ_i let each state respond to them differently. DiD is the special case with no factors. If treated states load more heavily on a rising factor, their outcome rises faster even without treatment, and DiD would wrongly attribute that to the law.

How the method works

1. Estimate the factors from the **never-treated** states, which are untouched by the policy.
2. For each treated state, estimate its loadings from its **pre-treatment** years only.
3. Predict its untreated outcome after adoption, $\hat{Y}_{it}(0)$, and take the gap $Y_{it} - \hat{Y}_{it}(0)$ as the effect.
4. Choose the number of factors by cross-validation: hold out each pre-treatment year in turn and see which number predicts it best.

How to write it: design

Because event-study estimates indicate that adopting and non-adopting states were on different trends before the laws took effect, we estimate the effect with the generalized synthetic control method of Xu (2017). It models untreated outcomes with state and year effects and a small number of latent factors with state-specific loadings, estimated from never-treated states, and imputes the counterfactual for each adopting state from its pre-adoption path. The number of factors is selected by cross-validation.

28.2 Load the data and test the pre-trends

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

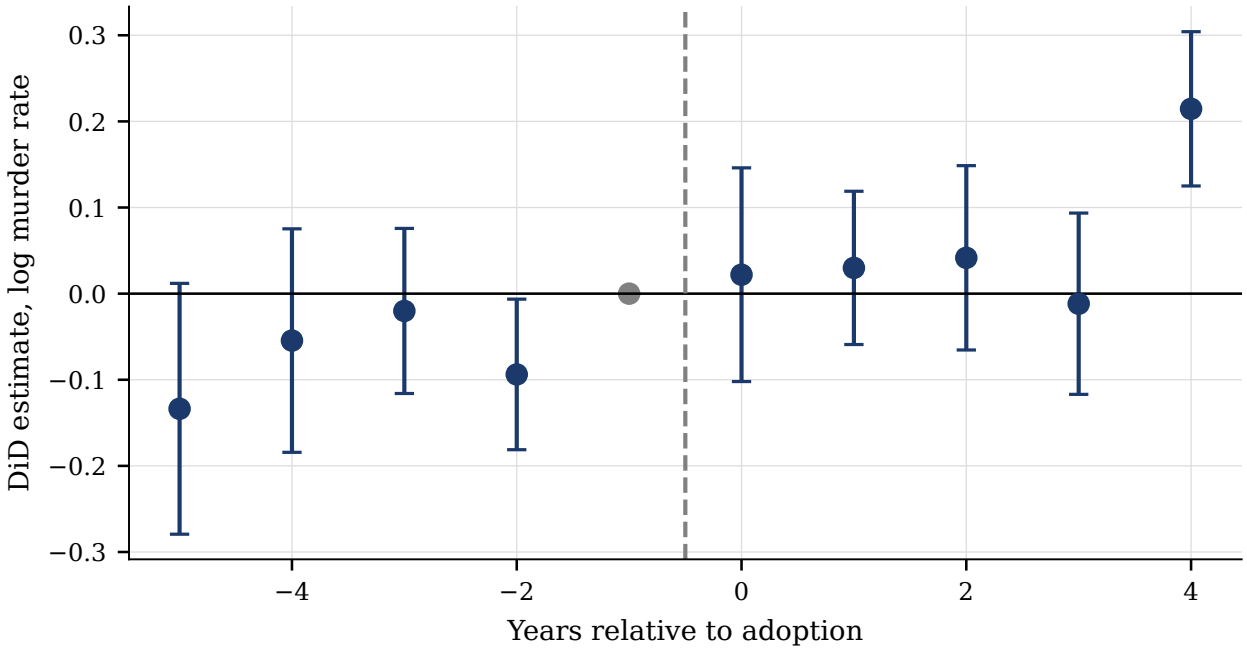


Figure 28.1. Load the data and test the pre-trends.

```
from causal_tests import event_study, gsc          # helper file: causal-inference-guide/examples/causal_tests.
  ↪ py

df = pd.read_csv("castle_murder.csv")
es = event_study(df, unit="state", time="year", y="l_murder", treat="castle_law")
ev = pd.DataFrame(es["event"])
fig, ax = plt.subplots()
ax.errorbar(ev["e"], ev["est"], yerr=1.96 * ev["se"], fmt="o", capsize=3)
ax.scatter([-1], [0], color="grey"); ax.axhline(0, color="black", lw=.8); ax.axvline(-0.5, color="grey", ls="--
  ↪ ")
ax.set_xlabel("Years relative to adoption"); ax.set_ylabel("DiD estimate, log murder rate")
print(ev.round(4).to_string(index=False))
print(f"\nPre-trend test (leads -5 to -2): chi2({es['pretrend']['df']}) = {es['pretrend']['stat']:.2f}, p-
  ↪ value = {es['pretrend']['p']:.3f}")
print(f"DiD estimates: Callaway-Sant'Anna {es['att']['est']:.4f} ({es['att']['se']:.4f}), TWFE {es['twfe']['
  ↪ est']:.4f} ({es['twfe']['se']:.4f})")
```

```
 e     est     se
-5 -0.1337 0.0743
-4 -0.0545 0.0662
-3 -0.0201 0.0489
-2 -0.0938 0.0446
 0  0.0220 0.0633
 1  0.0299 0.0454
 2  0.0416 0.0546
 3 -0.0117 0.0537
 4  0.2146 0.0457

Pre-trend test (leads -5 to -2): chi2(4) = 11.27, p-value = 0.024
DiD estimates: Callaway-Sant'Anna 0.0251 (0.0405), TWFE 0.0928 (0.0557)
```

How to read it

The leads are not flat: the joint test rejects that they are zero at the 5% level. Before adopting a law, murder rates in the adopting states were already moving differently from those in the other states. Any DiD estimate here mixes that pre-existing difference in trends with the effect of the law, which is why the decision rules of Chapter 1 send this dataset to generalized synthetic control.

28.3 Choose the number of factors

```
res = gsc(df, unit="state", time="year", y="l_murder", treat="castle_law")
print("Cross-validated mean squared prediction error by number of factors:")
for r, mspe in res["cv"].items():
    print(f"  r = {r}:  {mspe}")
print(f"Chosen: r = {res['r']}")
```

```
Cross-validated mean squared prediction error by number of factors:
  r = 0:  0.05038
  r = 1:  0.04571
  r = 2:  0.05784
  r = 3:  0.06724
  r = 4:  None
Chosen: r = 1
```

How to read it

With zero factors the model is two-way fixed effects. Adding factors lets states follow different trends; too many factors overfit the short pre-treatment period. Cross-validation picks the number that best predicts held-out pre-treatment years of the treated states. A non-zero choice confirms that unit-specific trends matter, as the event study suggested.

28.4 Estimate the effect

```
print(f"ATT = {res['att']['est']:.4f} (bootstrap s.e. {res['att']['se']:.4f}), p-value = {res['att']['p']:.3f}
  ↪ ")
g = pd.DataFrame(res["event"])
fig, ax = plt.subplots()
ax.errorbar(g["e"], g["est"], yerr=1.96 * g["se"], fmt="o", capsize=3, color="#0f7a4f")
ax.axhline(0, color="black", lw=.8); ax.axvline(-0.5, color="grey", ls="--")
ax.set_xlabel("Years relative to adoption"); ax.set_ylabel("Gap: actual minus counterfactual")
print(g.round(4).to_string(index=False))
print(f"\nRoot mean squared gap before adoption: {res['pre_rmspe']:.4f}")
```

```
ATT = 0.0211 (bootstrap s.e. 0.0820), p-value = 0.797
  e     est     se
-10 -0.4023 0.1530
 -9 -0.0233 0.1198
 -8 -0.2554 0.1196
 -7 -0.0062 0.0419
 -6  0.0008 0.0333
 -5 -0.0185 0.0375
 -4  0.0589 0.0318
 -3  0.0715 0.0377
 -2 -0.0423 0.0393
 -1  0.0431 0.0465
  0  0.0710 0.0652
  1  0.0054 0.0997
  2  0.0246 0.1039
  3 -0.0579 0.1165
  4  0.3273 0.1126

Root mean squared gap before adoption: 0.1550
```

How to read it

- **Before adoption**, the gaps between each treated state and its counterfactual should be close to zero: they measure how well the model fits. In the five years before adoption they stay within about ± 0.07, against DiD leads of up to -0.13 in the first step: the factor has absorbed most of the difference in trends. The gaps 8 to 10 years before adoption rest on the one or two states that adopted last and are noisy; they inflate the overall pre-treatment RMSE.
- **After adoption**, the gaps are the estimated effects. The average effect on the murder rate is small and not significant.

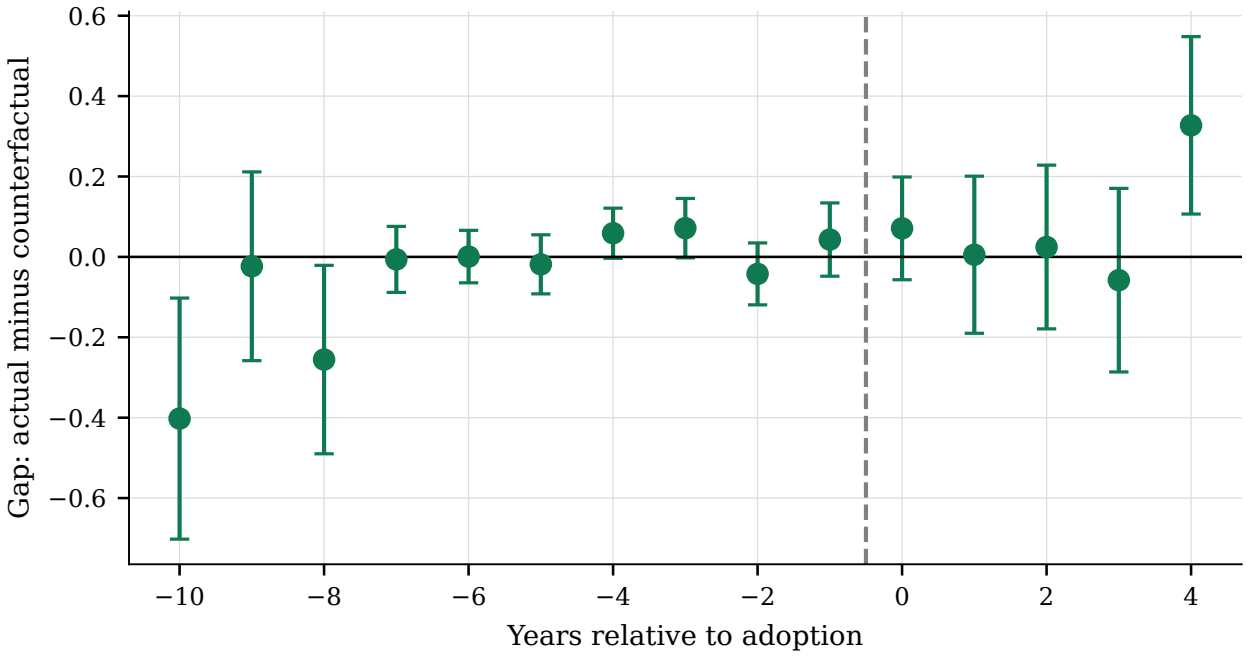


Figure 28.2. Estimate the effect.

- **Standard errors** come from bootstrapping states (both treated and control), re-estimating everything each time.

28.5 Compare the estimators

```
print(f"{'':34s}{'ATT':>9s}{'s.e.':>9s}")
print(f"{'Two-way fixed effects':34s}{es['twfe']['est']:9.4f}{es['twfe']['se']:9.4f}")
print(f"{'Callaway-Sant Anna (base g-1)':34s}{es['att']['est']:9.4f}{es['att']['se']:9.4f}")
print(f"{'Generalized synthetic control':34s}{res['att']['est']:9.4f}{res['att']['se']:9.4f}")
```

```
                                    ATT     s.e.
Two-way fixed effects            0.0928   0.0557
Callaway-Sant Anna (base g-1)    0.0251   0.0405
Generalized synthetic control    0.0211   0.0820
```

How to read it

Two-way fixed effects suggests murder rose by about 9% ($p \approx 0.10$). GSC and the Callaway–Sant'Anna estimator, which handle the pre-trend in different ways, both put the effect near 2% and far from significance. The TWFE estimate is the outlier: it averages over all pre-adoption years and so attributes the adopting states' different pre-trend to the law. Because the pre-trend test rejected parallel trends, the GSC estimate, whose counterfactual fits the pre-treatment years, is the one to lead with. Report all three: the comparison shows how much the conclusion depends on the treatment of trends.

28.6 Write the paper

How to write it: results

Event-study estimates reveal differential pre-trends in murder rates between adopting and non-adopting states (joint test of pre-adoption coefficients, $p = 0.02$), so difference-in-differences estimates are not credible for this outcome. Using the generalized synthetic control method with one latent factor selected by cross-validation, we find no significant effect of Castle Doctrine laws on the murder rate. The pre-adoption gaps between adopting states and their estimated counterfactuals are small, indicating a good fit.

Common mistakes

- Using GSC with very few pre-treatment periods. The loadings of each treated unit are estimated from its pre-treatment years; with fewer than about five, the counterfactual is unreliable.
- Using GSC with very few control units. The factors are estimated from them.
- Choosing the number of factors by looking at the post-treatment results. Fix it by cross-validation on pre-treatment data only.
- Forgetting to show the pre-treatment fit. It is the main evidence that the counterfactual is credible.
- In Stata or R, use `gsynth` (Xu) or `fect` (Liu, Wang and Xu) for the full method with covariates and parametric bootstrap.

CHAPTER 29

Synthetic Control

Applies when. One or a few treated units, many comparison units

Overview. Use this chapter when only one unit (a state, a country, a city) is treated and many untreated units are available. A DiD with one treated unit has no valid inference and an arbitrary control group. The synthetic control method builds the comparison unit from the data and judges significance by placebo tests.

Data. Texas prison expansion of 1993 and the other U.S. states, black male prisoners, 1985–2000

Software. Python · linearmodels · statsmodels

29.1 The question and the design

In 1993 Texas began a massive expansion of its prison capacity, roughly doubling the number of beds within a few years. **How much did the expansion increase the number of black men in Texas prisons?** The example follows Cunningham's *Causal Inference: The Mixtape*.

With one treated state, there is nothing to average over and no obvious control group. Which state, or average of states, shows what Texas would have looked like without the expansion? The synthetic control method (Abadie and Gardeazabal 2003; Abadie, Diamond and Hainmueller 2010) answers this with weights:

$$\hat{Y}_{\mathrm{TX},t}(0) = \sum_j w_j Y_{jt}, \qquad w_j \geq 0, \quad \sum_j w_j = 1$$

The weights are chosen so that the synthetic Texas matches the real Texas as closely as possible **before 1993**. The effect in each later year is the gap between the two.

Why the weights are restricted

Nonnegative weights that sum to one keep the synthetic unit inside the range of the actual donor states: an interpolation, not an extrapolation. The result is a transparent comparison: a reader can see which states make up synthetic Texas and judge whether they are plausible.

How to write it: design

We estimate the effect of the 1993 Texas prison expansion with the synthetic control method. The counterfactual for Texas is a weighted average of the other states, with nonnegative weights summing to one, chosen to reproduce the pre-1993 path of the outcome. Inference is based on placebo tests in which each donor state is in turn treated as if it had expanded its prisons.

29.2 Load the data and look at Texas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from causal_tests import synth_weights      # helper file: causal-inference-guide/examples/causal_tests.py

df = pd.read_csv("texas.csv")
W = df.pivot(index="year", columns="state", values="bmprison")
pre = W.index < 1993
donors = [s for s in W.columns if s != "Texas" and W[s].notna().all()]
```

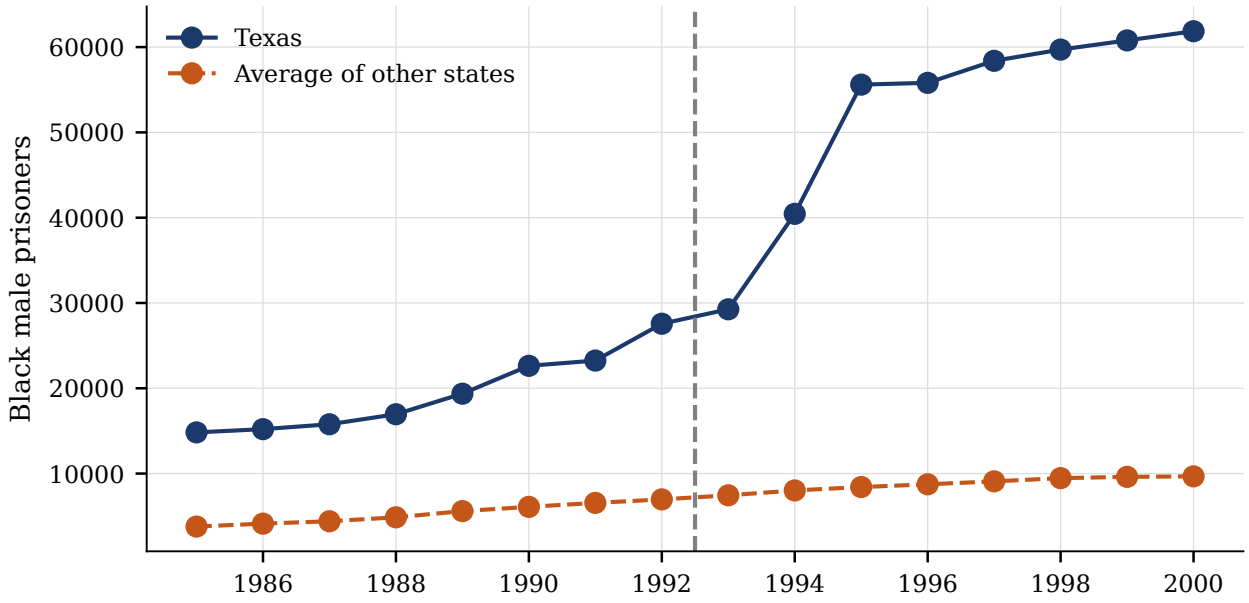


Figure 29.1. Load the data and look at Texas.

```
print(f"Texas plus {len(donors)} donor states, {W.index.min()}-{W.index.max()}; {pre.sum()} pre-treatment
  ↪ years")

fig, ax = plt.subplots()
ax.plot(W.index, W["Texas"], marker="o", label="Texas")
ax.plot(W.index, W[donors].mean(axis=1), marker="o", label="Average of other states")
ax.axvline(1992.5, color="grey", ls="--"); ax.set_ylabel("Black male prisoners"); ax.legend()
```

```
Texas plus 50 donor states, 1985-2000; 8 pre-treatment years
```

How to read it

Texas has many more black male prisoners than the average state, even before 1993, so the simple average is a poor comparison: its level and trend differ. Any DiD against the average would rest on a parallel-trends assumption that the picture makes implausible. The synthetic control builds a better comparison.

29.3 Estimate the weights

```
w = synth_weights(W.loc[pre, "Texas"], W.loc[pre, donors])
weights = pd.Series(w, index=donors).sort_values(ascending=False)
print(weights[weights > 0.001].round(3).to_string())

synth = W[donors] @ w
gap = W["Texas"] - synth
rmspe_pre = np.sqrt((gap[pre] ** 2).mean())
print(f"\nPre-1993 RMSPE: {rmspe_pre:,.0f}   (mean of Texas before 1993: {W.loc[pre, 'Texas'].mean():,.0f})")
```

```
Florida     0.373
New York    0.355
Illinois    0.272

Pre-1993 RMSPE: 863   (mean of Texas before 1993: 19,448)
```

How to read it

Synthetic Texas is a combination of a few large states. The pre-treatment root mean squared prediction error (RMSPE) is small relative to the size of the series: the weighted combination tracks Texas closely before 1993. A poor pre-treatment fit would make any post-treatment gap uninterpretable, so always report this number.

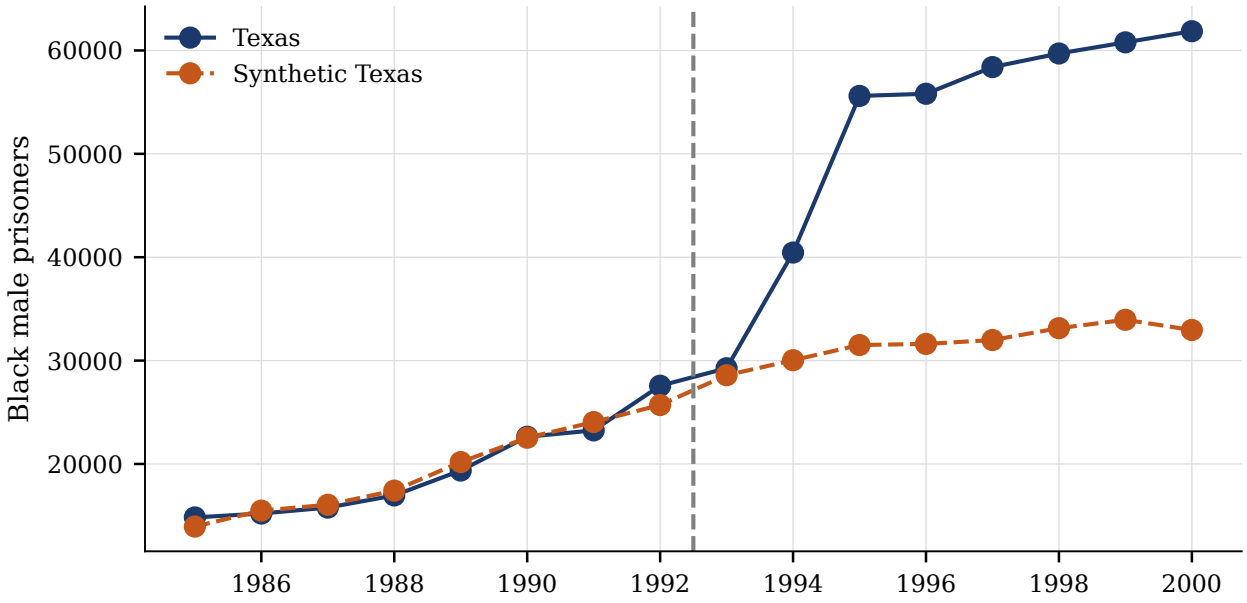


Figure 29.2. The effect: actual against synthetic Texas.

This chapter matches on the outcome path only. The full method also matches on predictors (income, unemployment, demographics) with a second set of weights on those predictors; the `synth` packages in Stata and R do this.

29.4 The effect: actual against synthetic Texas

```
fig, ax = plt.subplots()
ax.plot(W.index, W["Texas"], marker="o", label="Texas")
ax.plot(W.index, synth, ls="--", marker="o", label="Synthetic Texas")
ax.axvline(1992.5, color="grey", ls="--"); ax.set_ylabel("Black male prisoners"); ax.legend()
print(gap[~pre].round(0).astype(int).to_string())
print(f"\nAverage gap 1993-2000: {gap[~pre].mean():,.0f} prisoners")
```

```
year
1993      676
1994    10414
1995    24097
1996    24197
1997    26404
1998    26573
1999    26842
2000    28901

Average gap 1993-2000: 21,013 prisoners
```

How to read it

The two lines move together until 1993 and then separate sharply: by the late 1990s Texas held tens of thousands more black men in prison than synthetic Texas. The gap is the estimated effect in each year. It grows as the new capacity came into use.

29.5 Inference: placebo-in-space

```
def fit_gap(target, pool):
    wj = synth_weights(W.loc[pre, target], W.loc[pre, pool])
    return W[target] - W[pool] @ wj

ratios, pct, fig = {}, {}, plt.figure()
```

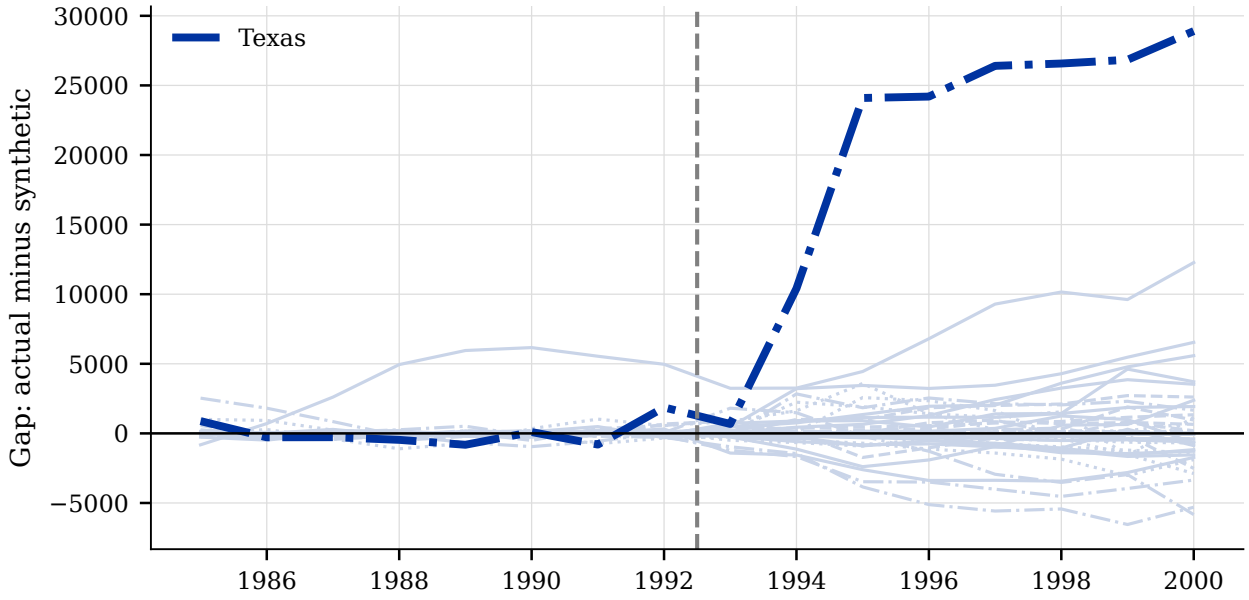


Figure 29.3. Inference: placebo-in-space.

```
ax = fig.gca()
for s in donors:
    g = fit_gap(s, [d for d in donors if d != s])
    ratios[s] = np.sqrt((g[~pre] ** 2).mean()) / np.sqrt((g[pre] ** 2).mean())
    pct[s] = g[~pre].mean() / W.loc[pre, s].mean()          # post-1993 gap relative to the state's own pre
  ↪ -1993 level
    ax.plot(W.index, g, color="#c9d4e8", lw=1)
ax.plot(W.index, gap, color="#0033a0", lw=2.5, label="Texas")
ax.axvline(1992.5, color="grey", ls="--"); ax.axhline(0, color="black", lw=.8); ax.set_ylabel("Gap: actual
  ↪ minus synthetic"); ax.legend()

ratio_tx = np.sqrt((gap[~pre] ** 2).mean()) / rmspe_pre
rank = 1 + sum(r >= ratio_tx for r in ratios.values())
print(f"Texas post/pre RMSPE ratio: {ratio_tx:.1f}")
print(f"Rank among {len(ratios) + 1} states: {rank}   permutation p-value: {rank / (len(ratios) + 1):.3f}")
print("Largest placebo ratios:", {k: round(float(v), 1) for k, v in sorted(ratios.items(), key=lambda kv: -kv
  ↪ [1])[:5]})
pct_tx = gap[~pre].mean() / W.loc[pre, "Texas"].mean()
rank2 = 1 + sum(v >= pct_tx for v in pct.values())
print(f"\nAverage post-1993 gap as a share of the pre-1993 level: Texas {pct_tx:.0%}, rank {rank2} of {len(pct)
  ↪  + 1}, p-value {rank2 / (len(pct) + 1):.3f}")
```

```
Texas post/pre RMSPE ratio: 26.7
Rank among 51 states: 9   permutation p-value: 0.176
Largest placebo ratios: {'Pennsylvania': 338.6, 'Nebraska': 50.2, 'Georgia': 46.6, 'Kentucky': 43.9, 'Mississippi':
↪ 32.2}

Average post-1993 gap as a share of the pre-1993 level: Texas 108%, rank 2 of 51, p-value 0.039
```

How to read it

Each grey line is a placebo: the same method applied to a state that did **not** expand its prisons, as if it had. If Texas's gap were unusual only by chance, many placebo gaps would look like it.

The standard comparison uses the ratio of post- to pre-treatment RMSPE, which accounts for how well each synthetic unit fitted before 1993. The **permutation p-value** is Texas's rank among all states divided by the number of states. On this statistic Texas ranks 9th of 51 ($p \approx 0.18$). But look at which states beat it: Pennsylvania's ratio is over 300 because its synthetic control fits its pre-1993 path almost perfectly, so even a

modest later deviation is enormous *relative to that fit*. With only 8 pre-treatment years, the ratio is a fragile statistic.

A second statistic is the average post-1993 gap as a share of each state's own pre-1993 level. On this measure Texas's gap, more than its entire pre-1993 level, ranks second of 51 ($p \approx 0.04$). Report both, and explain the difference: the effect is very large in size, and the conventional ratio-based p-value is weakened by placebo states with near-perfect pre-treatment fits.

A standard refinement drops placebo states whose pre-treatment fit is much worse than Texas's (Abadie et al. 2010 use two or five times its pre-RMSPE); here the problem runs the other way, which is why a second statistic helps.

29.6 Robustness: leave-one-out and an in-time placebo

```
for s in weights[weights > 0.01].index:
    pool = [d for d in donors if d != s]
    g = fit_gap("Texas", pool)
    print(f"Without {s:14s} average gap 1993-2000: {g[~pre].mean():,.0f}")

early = W.index < 1989
w_fake = synth_weights(W.loc[early, "Texas"], W.loc[early, donors])
g_fake = W["Texas"] - W[donors] @ w_fake
print(f"\nIn-time placebo (fake treatment in 1989): average gap 1989-1992 = {g_fake[(W.index >= 1989) & pre].
  ↪ mean():,.0f}")
```

```
Without Florida        average gap 1993-2000: 21,733
Without New York       average gap 1993-2000: 20,006
Without Illinois       average gap 1993-2000: 20,619

In-time placebo (fake treatment in 1989): average gap 1989-1992 = 936
```

How to read it

- **Leave-one-out.** Dropping each state with a positive weight in turn and re-estimating should not change the conclusion. If it did, the result would hinge on one donor.
- **In-time placebo.** Pretending the expansion happened in 1989 and using only earlier years for the weights, there should be no gap between 1989 and 1992. A gap would mean synthetic Texas cannot track Texas even when nothing happens.

29.7 Write the paper

How to write it: results

Synthetic Texas, a weighted combination of a few other large states, closely reproduces the number of black male prisoners in Texas before 1993. After the expansion of prison capacity, the two series diverge sharply: by the end of the decade Texas held tens of thousands more black men in prison than its synthetic counterpart. The result is robust to excluding any single donor state, and an in-time placebo shows no divergence before 1993. In placebo-in-space tests, the gap for Texas relative to its pre-1993 level is the second largest of 51 states (permutation $p = 0.04$); on the ratio of post- to pre-treatment prediction errors, Texas ranks ninth of 51, because several placebo states have near-perfect pre-treatment fits that inflate their ratios.

Common mistakes

- Including donors that were affected by the same policy, or by spillovers from the treated unit.
- Interpreting a gap when the pre-treatment fit is poor.
- Using conventional standard errors. With one treated unit, inference comes from placebo permutations.
- Choosing the pre-treatment period or the donor pool after looking at the results.
- Too short a pre-treatment period: the weights then fit noise.

CHAPTER 30

Interrupted Time Series

Applies when. One treated series, no comparison units
Overview. Use this chapter when a policy affects a single series and no untreated comparison group exists. The counterfactual then comes from the series' own past. The chapter estimates a segmented regression with seasonality and autocorrelation-robust errors, checks the residuals, and tests placebo dates.
Data. Car drivers killed or seriously injured in Great Britain, monthly, 1969–1984 (Harvey and Durbin 1986)
Software. Python · linearmodels · statsmodels

30.1 The question and the design

On 31 January 1983 wearing a front seatbelt became compulsory in Great Britain. **How many deaths and serious injuries of car drivers did the law prevent?** The law applied to the whole country at once, so there is no untreated region to compare with. This is the classic dataset of Harvey and Durbin (1986).

The interrupted time series (ITS) design uses the pre-law path of the series as the counterfactual, through a segmented regression:

$$y_t = \beta_0 + \beta_1 t + \delta\, D_t + \gamma\,(t - T_0)D_t + \sum_{m=2}^{12} \mu_m\, \text{month}_{mt} + u_t$$

Term	Meaning
β_1	Pre-law trend per month
D_t	1 from February 1983 onwards
δ	**Level change**: the immediate effect of the law
γ	**Slope change**: how the trend changed after the law
μ_m	Month effects: road deaths are seasonal

The identifying assumption

Without the law, the series would have continued along its pre-law trend and seasonal pattern. Nothing else that affects road deaths may change at the same time. This is a strong assumption, and the credibility of an ITS study rests on arguing it: listing other events (fuel prices, speed limits, recessions) and showing they did not coincide with the intervention.

How to write it: design

Because the seatbelt law applied nationwide from February 1983, no contemporaneous control group exists. We estimate its effect with an interrupted time series design: a segmented regression of monthly driver casualties on a linear trend, month effects, and a level and a slope change at the date the law took effect, with Newey–West standard errors to allow for serial correlation.

30.2 Load the data and look at the series

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

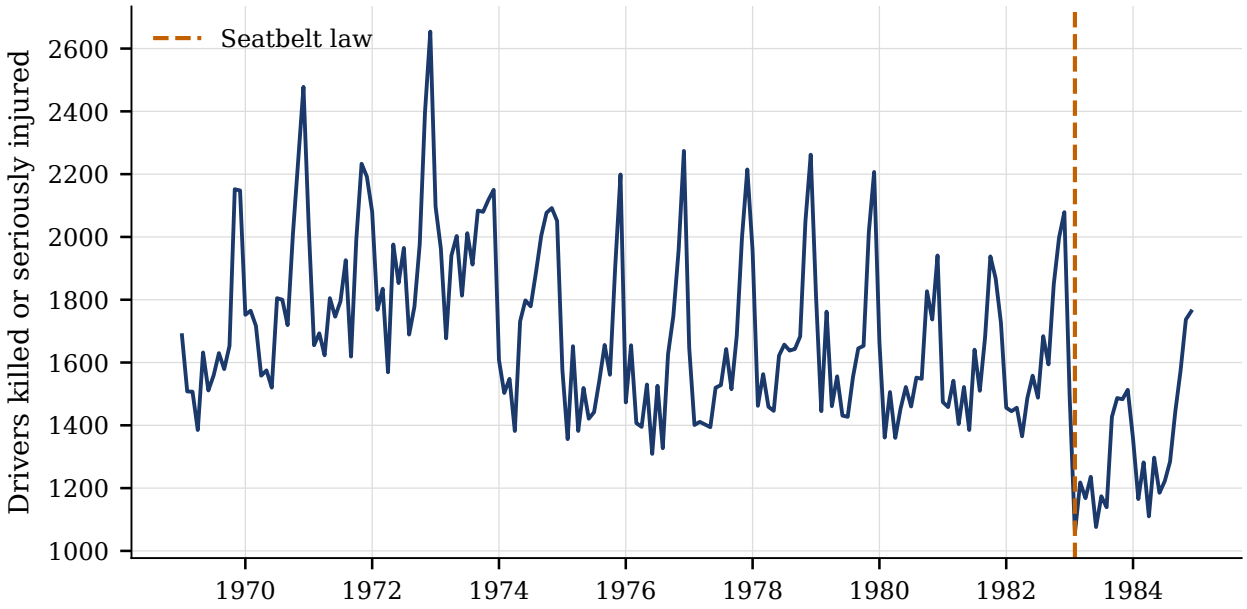


Figure 30.1. Load the data and look at the series.

```
import statsmodels.api as sm

df = pd.read_csv("seatbelt.csv")
df["month"] = (np.round((df["time"] % 1) * 12).astype(int) % 12) + 1
df["t"] = np.arange(len(df))
T0 = int(df.index[df["law"] == 1][0])
print(f"{len(df)} months from {df['time'].iloc[0]:.0f} to {df['time'].iloc[-1]:.2f}; law from observation {T0}
  ↪  (February 1983)")
print(f"Mean deaths before the law: {df.loc[:T0 - 1, 'deaths'].mean():.0f}   after: {df.loc[T0:, 'deaths'].
  ↪ mean():.0f}")

fig, ax = plt.subplots()
ax.plot(df["time"], df["deaths"], lw=1.2)
ax.axvline(df["time"].iloc[T0], color="#c46000", ls="--", label="Seatbelt law")
ax.set_ylabel("Drivers killed or seriously injured"); ax.legend()
```

```
192 months from 1969 to 1984.92; law from observation 169 (February 1983)
Mean deaths before the law: 1718   after: 1322
```

How to read it

The series has a strong seasonal pattern (more casualties in winter), a gently falling trend, and a visible drop in early 1983. There is also a fall around 1974, the time of the first oil crisis, when fuel prices jumped and a national speed limit was introduced. Keep it in mind: it is exactly the kind of event that can masquerade as, or contaminate, a policy effect.

30.3 Estimate the segmented regression

```
df["post"] = (df["t"] >= T0).astype(int)
df["t_post"] = (df["t"] - T0) * df["post"]
X = pd.get_dummies(df["month"], prefix="m", drop_first=True, dtype=float)
X = sm.add_constant(pd.concat([df[["t", "post", "t_post"]], X], axis=1))
lags = int(np.floor(4 * (len(df) / 100) ** (2 / 9))) + 1
its = sm.OLS(df["deaths"], X).fit(cov_type="HAC", cov_kwds={"maxlags": lags})
print(its.summary2().tables[1].loc[["const", "t", "post", "t_post"]].round(3))
print(f"\nLevel change as a share of the pre-law mean: {its.params['post'] / df.loc[:T0 - 1, 'deaths'].mean()
  ↪ :.1%}")
```

```
          Coef.  Std.Err.       z  P>|z|    [0.025    0.975]
```

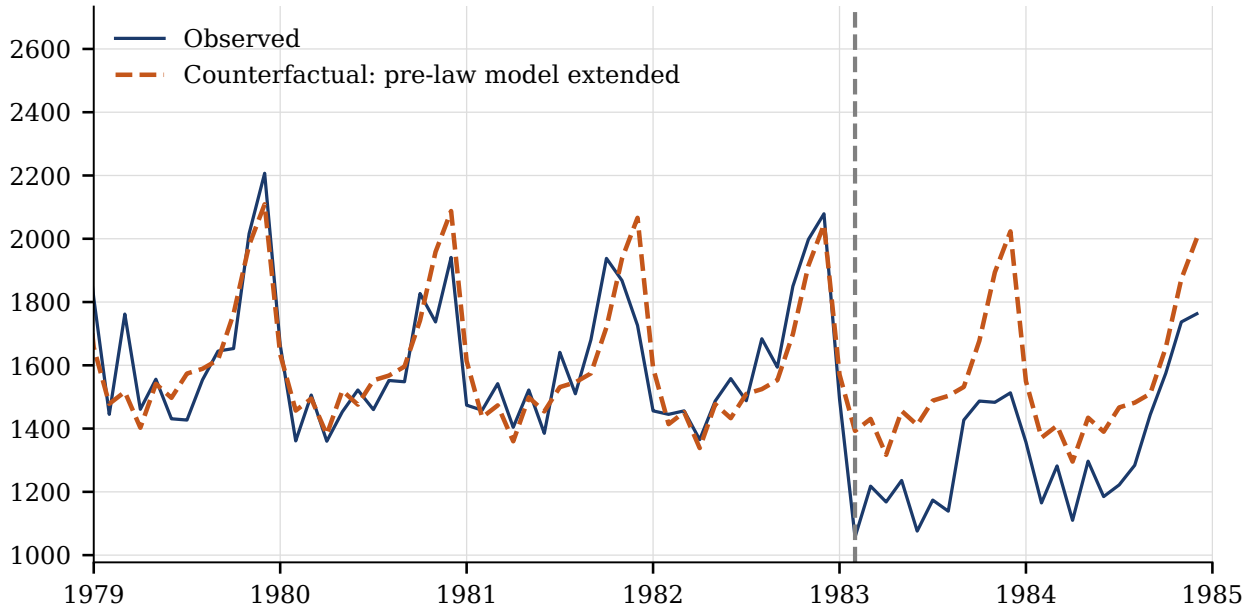


Figure 30.2. The counterfactual.

```
const   1872.062    65.584  28.544  0.000  1743.519  2000.605
t         -1.778     0.409  -4.352  0.000    -2.579    -0.978
post    -300.666    41.289  -7.282  0.000  -381.591  -219.741
t_post     6.871     2.125   3.234  0.001     2.707    11.036

Level change as a share of the pre-law mean: -17.5%
```

How to read it

- **Level change (`post`).** In the month the law took effect, driver casualties fell by about 300, roughly 17% of the pre-law monthly average. The effect is large and highly significant.
- **Slope change (`t_post`).** Positive: after the immediate drop, the series fell less steeply than before. With only 23 post-law months, this slope is estimated imprecisely, and it should not be extrapolated far.
- **Standard errors** are Newey–West with a few lags, which keeps inference valid when monthly errors are serially correlated.

30.4 The counterfactual

```
pre_fit = sm.OLS(df.loc[:T0 - 1, "deaths"], X.loc[:T0 - 1, [c for c in X.columns if c not in ("post", "t_post")
  ↪ ]]).fit()
cf = pre_fit.predict(X[[c for c in X.columns if c not in ("post", "t_post")]])
fig, ax = plt.subplots()
ax.plot(df["time"], df["deaths"], lw=1, label="Observed")
ax.plot(df["time"], cf, lw=1.4, ls="--", label="Counterfactual: pre-law model extended")
ax.axvline(df["time"].iloc[T0], color="grey", ls="--"); ax.set_xlim(1979, 1985); ax.legend()
print(f"Average monthly gap after the law: {(df['deaths'] - cf)[T0:].mean():.0f}")
print(f"Total over {len(df) - T0} months: {(df['deaths'] - cf)[T0:].sum():,.0f}")
```

```
Average monthly gap after the law: -225
Total over 23 months: -5,166
```

How to read it

The dashed line extends the pre-law trend and seasonal pattern into the post-law period: what the model says would have happened without the law. The observed series lies well below it after February 1983. Summed over the post-law months, this is several thousand fewer drivers killed or seriously injured, the number a policy reader wants.

30.5 Diagnostics: autocorrelation

```
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.stats.stattools import durbin_watson
resid = its.resid
print(f"Durbin-Watson: {durbin_watson(resid):.2f}")
print(acorr_ljungbox(resid, lags=[1, 6, 12]).round(4))

# the same model with AR(1) errors, estimated by iterated feasible GLS (Cochrane-Orcutt)
ar = sm.GLSAR(df["deaths"], X, rho=1).iterative_fit(maxiter=20)
print(f"\nWith AR(1) errors: level change = {ar.params['post']:.1f} (s.e. {ar.bse['post']:.1f}), AR
  ↪ coefficient = {ar.model.rho[0]:.2f}")
```

```
Durbin-Watson: 0.93
     lb_stat  lb_pvalue
1    54.9953        0.0
6   164.8760        0.0
12  220.0125        0.0

With AR(1) errors: level change = -277.7 (s.e. 98.5), AR coefficient = 0.53
```

How to read it

The Durbin–Watson statistic below 2 and the Ljung–Box tests show positive serial correlation in the residuals: a bad month tends to be followed by another. The Newey–West errors already allow for this. Modelling the autocorrelation directly, with AR(1) errors, is the standard robustness check. The level change remains large and significant.

30.6 Placebo dates

```
pre = df.iloc[:T0].copy()
for label, at in (("January 1976", int(np.argmin(np.abs(pre["time"] - 1976.0)))), ("May 1978", int(np.argmin(
  ↪ np.abs(pre["time"] - 1978.33))))):
    p = pre.assign(fake=(pre["t"] >= at).astype(int), t_fake=(pre["t"] - at) * (pre["t"] >= at))
    Xp = sm.add_constant(pd.concat([p[["t", "fake", "t_fake"]], pd.get_dummies(p["month"], prefix="m",
  ↪ drop_first=True, dtype=float)], axis=1))
    r = sm.OLS(p["deaths"], Xp).fit(cov_type="HAC", cov_kwds={"maxlags": lags})
    print(f"Placebo law in {label}: level change = {r.params['fake']:7.1f} (s.e. {r.bse['fake']:.1f}), p-value
  ↪  = {r.pvalues['fake']:.3f}")
```

```
Placebo law in January 1976: level change =  -153.6 (s.e. 90.0), p-value = 0.088
Placebo law in May 1978: level change =    15.1 (s.e. 57.5), p-value = 0.792
```

How to read it

A placebo date is a date when nothing happened. Estimated on pre-law data only, it should show no effect.

- **May 1978**: no significant effect, as it should be.
- **January 1976**: a large negative "effect" of about –150, significant at the 10% level. This is not a failure of the method: it picks up the 1974–75 oil-crisis fall in the data, which a single linear trend cannot capture. It shows that **this series has other breaks**, and that the trend before 1983 is not as simple as the model assumes.

Report both placebos. The case for the seatbelt effect rests on the size and sharpness of the 1983 drop, which no other event matches, and Harvey and Durbin strengthened it by modelling fuel prices and traffic volume and by comparing with rear-seat passengers, who were not covered by the law. When a comparison series exists, use it.

30.7 Write the paper

How to write it: results

The segmented regression indicates an immediate reduction of about 300 drivers killed or seriously injured per month when the seatbelt law took effect, around 17 percent of the pre-law average (Newey–West standard errors). The estimate is robust to modelling the residual autocorrelation with AR(1) errors. A placebo intervention in 1978 shows no effect; a placebo in 1976 shows a sizeable decline, significant at the 10 percent level, reflecting the fall associated with the 1974 oil crisis, which underlines that the attribution of the 1983 drop to the law rests on the absence of other changes at that date.

Common mistakes

- Ignoring seasonality: a policy that starts in a low season looks effective.
- Using conventional standard errors with autocorrelated monthly data.
- Extrapolating the pre-trend far beyond the intervention, or over-interpreting a slope change estimated from few post-periods.
- Not checking for other events at the same date. ITS has no comparison group to absorb them.
- When a comparison series exists (an unaffected group, region or outcome), use it: that turns the ITS into a comparative ITS or a DiD.

CHAPTER 31

Regression Discontinuity

Applies when. Treatment assigned by a cutoff

Overview. Use this chapter when treatment is assigned, fully or partly, by whether a running variable crosses a known cutoff: a test score, an income threshold, an age, an election margin. It estimates the jump at the cutoff with local linear regression and a data-driven bandwidth, then runs the checks that make an RD credible: manipulation, covariate balance, bandwidth sensitivity and placebo cutoffs.

Data. U.S. House elections, 1946–1995: ADA voting score of the winner by the Democratic vote share (Lee, Moretti and Butler 2004)

Software. Python · linearmodels · statsmodels

31.1 The question and the design

Do voters choose policies, or only politicians? If candidates converge to the median voter, it should not matter much for how a district is represented which party narrowly wins. If they do not, a narrow Democratic win should produce a much more liberal representative than a narrow Republican win. **Lee, Moretti and Butler (2004)** answered this with a regression discontinuity.

Whether a Democrat wins is determined by a cutoff: the Democratic vote share R_i crossing 0.5. Districts where the Democrat won 50.5% and those where she won 49.5% are, on average, alike in everything except who won. So the jump in the winner's liberal voting score at 0.5 measures the effect of electing a Democrat:

$$\tau_{RD} = \lim_{r \downarrow 0.5} E[\,Y \mid R = r\,] - \lim_{r \uparrow 0.5} E[\,Y \mid R = r\,]$$

Symbol	Variable in the file	Definition
Y	`score`	ADA score of the elected representative (0 = conservative, 100 = liberal)
R	`demvoteshare`	Democratic share of the two-party vote
D	`democrat`	1 if the Democrat won (= 1 exactly when $R \geq 0.5$: a **sharp** design)
—	`lagdemvoteshare`	Democratic vote share in the previous election (determined before the election)

The identifying assumption

The potential outcomes are **continuous** at the cutoff: nothing else jumps at 0.5. This fails if candidates can precisely control which side of 0.5 they land on (manipulation), which would make the winners of close races systematically different. It can be checked in two ways: the density of the running variable should not jump at the cutoff, and characteristics determined before the election should not jump either.

How to write it: design

We exploit the discontinuity in party control created by the 50 percent threshold in two-party House elections. Comparing districts in which the Democratic candidate barely won with districts in which she barely lost identifies the effect of electing a Democrat on the representative's voting record, under the assumption that potential outcomes are continuous in the vote share at the threshold.

31.2 Load the data and plot the discontinuity

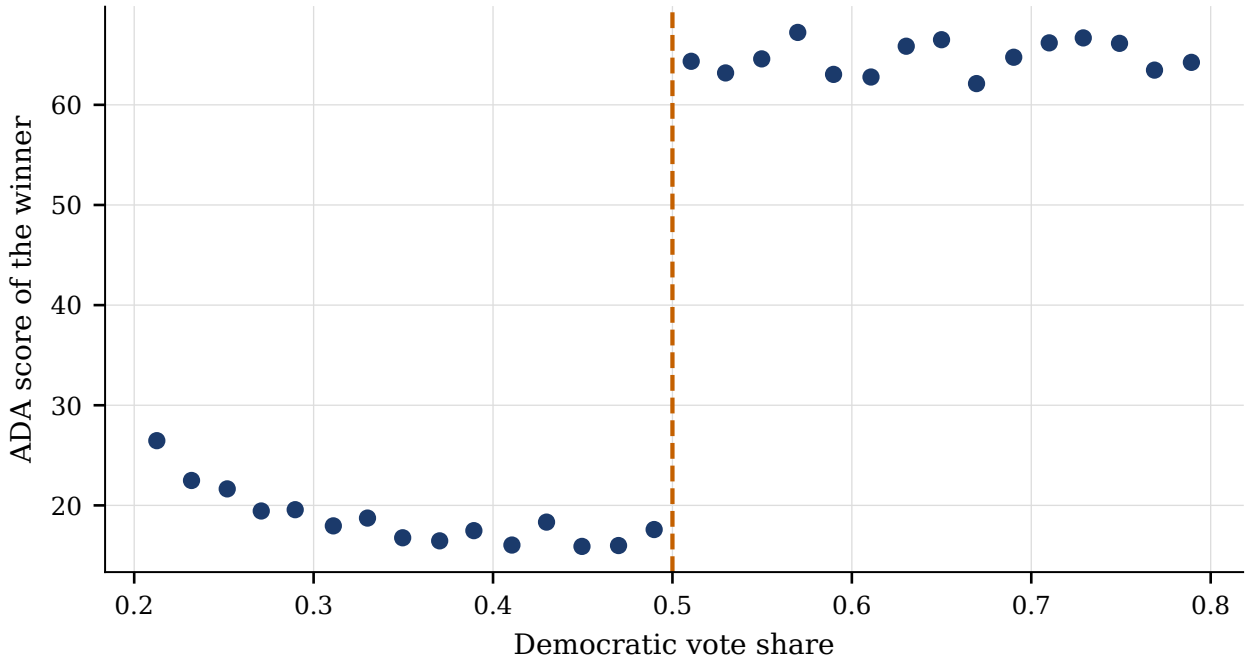


Figure 31.1. Load the data and plot the discontinuity.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from causal_tests import rd_estimate, mccrary_test       # helper file: causal-inference-guide/examples/
  ↪ causal_tests.py

df = pd.read_csv("close_elections.csv")
c = 0.5
print(f"{len(df):,} district-elections; {(df['demvoteshare'] >= c).mean():.0%} won by a Democrat")
print("Sharp design (democrat = 1 exactly when vote share >= 0.5):", bool(((df["demvoteshare"] >= c) == (df["
  ↪ democrat"] == 1)).all()))

d = df[(df["demvoteshare"] > 0.2) & (df["demvoteshare"] < 0.8)].copy()     # contested races for the plot
d["bin"] = pd.cut(d["demvoteshare"], np.arange(0.2, 0.801, 0.02))
b = d.groupby("bin", observed=True).agg(x=("demvoteshare", "mean"), y=("score", "mean"))
fig, ax = plt.subplots()
ax.scatter(b["x"], b["y"], s=18)
ax.axvline(c, color="#c46000", ls="--")
ax.set_xlabel("Democratic vote share"); ax.set_ylabel("ADA score of the winner")
```

```
13,566 district-elections; 60% won by a Democrat
Sharp design (democrat = 1 exactly when vote share >= 0.5): True
```

How to read it

Always start with this plot: averages of the outcome in narrow bins of the running variable. The jump at 0.5 is unmistakable: districts that barely elect a Democrat are represented by someone with an ADA score roughly 45–50 points higher than districts that barely elect a Republican. On either side the relationship with the vote share is fairly flat. A plot like this, with the estimate overlaid, should be Figure 1 of an RD paper.

31.3 Local linear estimate with a data-driven bandwidth

```
r = rd_estimate(df["demvoteshare"], df["score"], cutoff=c)
print(f"Bandwidth (Imbens-Kalyanaraman): h = {r['h']:.4f}")
```

```
print(f"RD estimate: {r['est']:.2f} (robust s.e. {r['se']:.2f}), using {r['n']:,} elections within the
  ↪ bandwidth")

# the same estimate written out: weighted least squares with a triangular kernel
h = r["h"]
w = np.clip(1 - np.abs(df["demvoteshare"] - c) / h, 0, None)
s = df[w > 0].assign(x=lambda z: z["demvoteshare"] - c, D=lambda z: (z["demvoteshare"] >= c).astype(int))
X = sm.add_constant(pd.DataFrame({"D": s["D"], "x": s["x"], "Dx": s["D"] * s["x"]}))
wls = sm.WLS(s["score"], X, weights=w[w > 0]).fit(cov_type="HC1")
print(f"WLS check: {wls.params['D']:.2f} (s.e. {wls.bse['D']:.2f})")
```

```
Bandwidth (Imbens-Kalyanaraman): h = 0.1084
RD estimate: 46.74 (robust s.e. 1.27), using 4,961 elections within the bandwidth
WLS check: 46.74 (s.e. 1.27)
```

How to read it

The estimate fits a separate straight line on each side of the cutoff, using only elections within a vote-share distance h of 0.5 and giving more weight to those closest to it. The jump of about 47 points is the effect of a Democratic win on the winner's ADA score: party, not the median voter, determines roll-call voting. The WLS lines reproduce it exactly, which shows there is nothing hidden in the helper function.

Local linear regression is the standard. Global high-order polynomials (fitting a fourth-degree curve to all the data) can produce misleading estimates and should not be used (Gelman and Imbens 2019).

31.4 Robustness: bandwidth and polynomial order

```
print(f"{'bandwidth':>10s}{'estimate':>10s}{'s.e.':>8s}{'obs.':>8s}")
for m in (0.5, 0.75, 1.0, 1.5, 2.0):
    rr = rd_estimate(df["demvoteshare"], df["score"], cutoff=c, h=h * m)
    print(f"{h * m:10.3f}{rr['est']:10.2f}{rr['se']:8.2f}{rr['n']:8d}")

s2 = s.assign(x2=s["x"] ** 2, Dx2=s["D"] * s["x"] ** 2)
X2 = sm.add_constant(pd.DataFrame({"D": s2["D"], "x": s2["x"], "Dx": s2["D"] * s2["x"], "x2": s2["x2"], "Dx2":
  ↪  s2["Dx2"]}))
q = sm.WLS(s2["score"], X2, weights=w[w > 0]).fit(cov_type="HC1")
print(f"\nLocal quadratic (same bandwidth): {q.params['D']:.2f} (s.e. {q.bse['D']:.2f})")
```

```
 bandwidth  estimate    s.e.    obs.
     0.054     46.24    1.82    2575
     0.081     46.35    1.47    3806
     0.108     46.74    1.27    4961
     0.163     47.16    1.05    7318
     0.217     47.57    0.92    9231

Local quadratic (same bandwidth): 45.85 (s.e. 1.90)
```

How to read it

The estimate barely moves between half and double the optimal bandwidth, and a local quadratic gives the same answer. That stability is what a referee wants to see. If the estimate changed sharply with the bandwidth, the effect would depend on functional-form assumptions away from the cutoff.

31.5 Validity 1: is there manipulation?

```
m = mccrary_test(df["demvoteshare"], cutoff=c)
print(f"McCrary test: log difference in density = {m['theta']:.3f} (s.e. {m['se']:.3f}), z = {m['z']:.2f}, p-
  ↪ value = {m['p']:.3f}")
fig, ax = plt.subplots()
ax.hist(df.loc[(df["demvoteshare"] > 0.2) & (df["demvoteshare"] < 0.8), "demvoteshare"], bins=60, color="#0033
  ↪ a0", alpha=.6)
ax.axvline(c, color="#c46000", ls="--"); ax.set_xlabel("Democratic vote share"); ax.set_ylabel("Elections")
```

```
McCrary test: log difference in density = -0.018 (s.e. 0.058), z = -0.30, p-value = 0.763
```

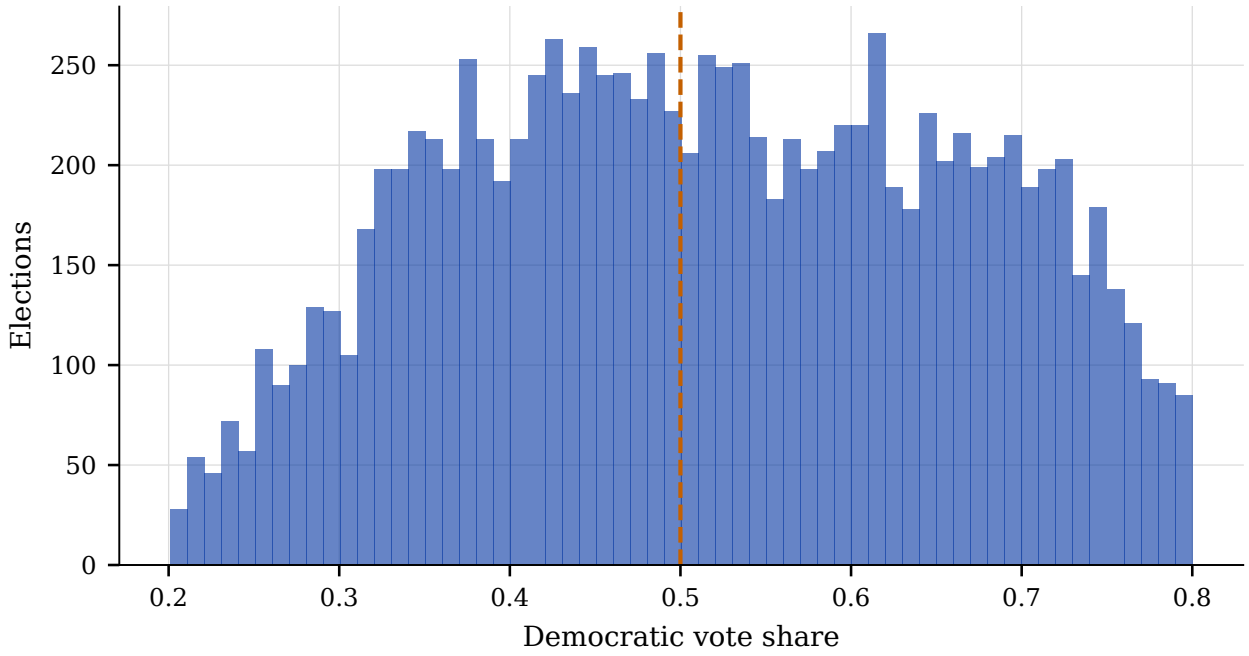


Figure 31.2. Validity 1: is there manipulation?.

How to read it

If candidates could precisely engineer narrow wins (through fraud or recounts, say), there would be too many elections just above 0.5 and too few just below. The histogram shows no such heaping, and the McCrary test finds no significant jump in the density. In settings like test scores or income eligibility, where people know the cutoff, this test often fails, and an RD with sorting is not credible.

31.6 Validity 2: covariate balance and placebo cutoffs

```
cov = rd_estimate(df["demvoteshare"], df["lagdemvoteshare"], cutoff=c, h=h)
print(f"Jump in the previous election's Democratic vote share: {cov['est']:.3f} (s.e. {cov['se']:.3f}), p = {2
  ↪  * (1 - __import__('scipy').stats.norm.cdf(abs(cov['est'] / cov['se'])))):.3f}")
narrow = rd_estimate(df["demvoteshare"], df["lagdemvoteshare"], cutoff=c, h=h / 2)
print(f"  with half the bandwidth:                          {narrow['est']:.3f} (s.e. {narrow['se']:.3f})")

Xc = X.assign(lag=s["lagdemvoteshare"])
ctl = sm.WLS(s["score"], Xc, weights=w[w > 0]).fit(cov_type="HC1")
print(f"RD estimate controlling for the previous vote share: {ctl.params['D']:.2f} (s.e. {ctl.bse['D']:.2f})\n
  ↪ ")

for pc in (0.4, 0.6):
    side = df[(df["demvoteshare"] < c) if pc < c else (df["demvoteshare"] >= c)]
    p = rd_estimate(side["demvoteshare"], side["score"], cutoff=pc, h=h)
    print(f"Placebo cutoff at {pc}: {p['est']:.2f} (s.e. {p['se']:.2f})")
```

```
Jump in the previous election's Democratic vote share: 0.028 (s.e. 0.008), p = 0.000
  with half the bandwidth:                          0.044 (s.e. 0.011)
RD estimate controlling for the previous vote share: 47.23 (s.e. 1.24)

Placebo cutoff at 0.4: -0.05 (s.e. 1.22)
Placebo cutoff at 0.6: 0.02 (s.e. 1.61)
```

How to read it

- **Covariate balance.** The previous election's vote share was determined before this election, so it should not jump at 0.5. It shows a small jump of about 3 percentage points, statistically significant, and it does not

disappear with half the bandwidth. This imbalance in close House elections is documented in the literature (Caughey and Sekhon 2011 discuss sorting in very close races). It must be reported. Re-running the main estimate with the covariate as a control leaves it essentially unchanged (47.2 against 46.7), so the imbalance does not drive the result.

- **Placebo cutoffs.** At vote shares of 0.4 and 0.6, where nothing changes, the estimated jumps are small compared with the 47-point effect at 0.5.

31.7 Write the paper

How to write it: results

Figure 1 shows a sharp discontinuity in the ADA score of the elected representative at the 50 percent vote-share threshold. Local linear estimates with the Imbens–Kalyanaraman bandwidth imply that a Democratic victory raises the representative's ADA score by about 47 points (robust standard error 1.3). The estimate is stable across bandwidths from half to twice the optimal one and under a local quadratic specification. A McCrary test shows no discontinuity in the density of the vote share at the threshold. The previous election's Democratic vote share displays a small discontinuity of about 3 percentage points, consistent with incumbency effects; controlling for it does not change the results.

Common mistakes

- Using global high-order polynomials instead of local linear regression.
- Reporting one bandwidth only.
- Skipping the density test and covariate balance: they are the evidence that the design is valid.
- Forgetting that the RD effect is **local**: it applies to units at the cutoff (close elections), not to all districts.
- Treating a fuzzy design as sharp. If crossing the cutoff only raises the probability of treatment, divide the outcome jump by the treatment jump (a local IV estimate).
- For inference, the bias-corrected robust confidence intervals of Calonico, Cattaneo and Titiunik (`rdrobust`) are now standard.

PART V

Machine Learning for Prediction and Causal Estimation

Overview

Machine learning enters economics in two roles that should not be confused. In the first, the aim is *prediction*: to forecast an outcome out of sample when there are many candidate predictors (Mullainathan and Spiess, 2017; Varian, 2014). In the second, the aim is the *causal effect* of one variable, and machine learning is used only to control flexibly for the others (Chernozhukov et al., 2018). The methods are different, the validation is different, and the conclusions that may be drawn are different.

The part follows that split. Chapters on LASSO (Tibshirani, 1996), ridge (Hoerl and Kennard, 1970), the elastic net (Zou and Hastie, 2005), random forests (Breiman, 2001) and gradient boosting (Friedman, 2001; Chen and Guestrin, 2016) treat prediction, with the validation rules that go with it. The later chapters treat double machine learning in three forms and the causal forest (Athey et al., 2019). The rules A1–A5 and B1–B5 of Chapter 1 state when each applies, including the case in which machine learning cannot help because the confounders are unobserved.

Chapters at a glance

Ch.	Chapter	Use it when
32	LASSO	Linear, few predictors matter
33	Ridge Regression	Linear, many small effects
34	Elastic Net	Linear, strongly correlated predictors
35	Random Forest	Nonlinear, fewer than 1,000 observations
36	Gradient Boosting (XGBoost)	Nonlinear, at least 1,000 observations
37	Double Machine Learning: Partially Linear Model	Continuous treatment, many or nonlinear controls
38	Double Machine Learning: Interactive Model	Binary treatment, many or nonlinear controls
39	Double Machine Learning with an Instrument (DML-IV)	Endogenous treatment, valid instrument
40	Causal Forest	Effects that differ across people, N $\geq$ 2,000

Further reading

Hastie et al. (2009) is the standard reference for the prediction methods. The double machine learning framework is developed in Chernozhukov et al. (2018), and the earlier post-double-selection idea in Belloni et al. (2014). For heterogeneous treatment effects see Wager and Athey (2018) and Athey et al. (2019). Mullainathan and Spiess (2017) and Varian (2014) explain what these methods add to the applied economist's toolkit.

CHAPTER 32

LASSO

Applies when. Linear, few predictors matter

Overview. Use this chapter when the relationship is close to linear, no two predictors are strongly correlated, and only a few of many candidate predictors are likely to matter. It takes one real dataset from the question to the finished results table: choosing the penalty, reading which predictors survive, checking that the selection is stable, and testing accuracy on data the model has not seen.

Data. 814 students, introductory economics course (Wooldridge econmath)

Software. Python · scikit-learn · statsmodels

32.1 The question and the model

The question is a practical one for any economics department: **which student characteristics predict the final score in an introductory economics course?** The file records fourteen candidate predictors for 814 students: study habits, grades, ACT scores, prior economics and calculus, attendance attitudes and family background. Some surely matter and many probably do not. Choosing among them by hand invites a specification search; LASSO makes the choice by a rule.

LASSO (least absolute shrinkage and selection operator; Tibshirani 1996) estimates a linear model but adds a penalty on the size of the coefficients:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

The absolute-value penalty sets the coefficients of weak predictors to **exactly zero**, so LASSO selects variables while it estimates. The penalty weight λ controls how many survive: $\lambda = 0$ gives OLS; a large λ drops everything.

Variable	Definition	Variable	Definition
`score`	Course score, % (outcome)	`male`	1 if male
`age`	Age in years	`calculus`	1 if took calculus
`work`	Hours worked per week	`econhs`	1 if took economics in high school
`study`	Hours studied per week	`attexc`, `attgood`	1 if attendance rated excellent / good
`colgpa`	College GPA before the course	`fathcoll`, `mothcoll`	1 if father / mother has a degree
`hsgpa`	High-school GPA	`acteng`, `actmth`	ACT English / maths score

Prediction, not causation

LASSO answers *which variables help to predict the score*. It does not say that studying more would raise a student's score: a selected variable may stand in for an omitted one, and a dropped variable may still have a causal effect that is small relative to the noise. If the question is causal, use double machine learning instead (see Chapter 37).

32.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV, Lasso, LinearRegression
from sklearn.ensemble import RandomForestRegressor

df = pd.read_csv("econmath.csv")
y = df["score"]
X = df.drop(columns="score")
print(f"N = {len(df)} students, p = {X.shape[1]} candidate predictors")

r = X.corr().abs().where(~np.eye(X.shape[1], dtype=bool))
print(f"largest |r| between predictors: {r.max().max():.2f} ({r.stack().idxmax()[0]} and {r.stack().idxmax()
  ↪ [1]})")

cv = KFold(5, shuffle=True, random_state=0)
rmse = lambda m: (-cross_val_score(m, X, y, cv=cv, scoring="neg_mean_squared_error").mean()) ** 0.5
lasso = make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0))
forest = RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=0.5, random_state=0)
r_l, r_f = rmse(lasso), rmse(forest)
print(f"cross-validated RMSE: LASSO {r_l:.2f}, random forest {r_f:.2f}, forest gain {1 - r_f / r_l:+.1%}")
```

```
N = 814 students, p = 14 candidate predictors
largest |r| between predictors: 0.77 (attexc and attgood)
cross-validated RMSE: LASSO 10.19, random forest 10.40, forest gain -2.0%
```

The decision rules of Chapter 1 are met: there are more than 100 observations, the random forest does **not** beat LASSO out of sample (it is worse), so the relationship is treated as linear, and no two predictors are correlated above 0.8. The last check, sparsity, comes in Section 32.4.

> **How to write it: data section**
>
> *The sample contains 814 students with complete records. The outcome is the course score (in per cent). Fourteen candidate predictors describe study effort, prior achievement, preparation in mathematics and economics, attitudes to attendance and family background. A random forest does not improve on a linear model in five-fold cross-validation, so we model the score as a linear function of the predictors and use LASSO to select among them.*

32.3 Standardise, then choose the penalty by cross-validation

The penalty treats every coefficient alike, so the predictors must be on the same scale: a coefficient on hours per week and one on a 0/1 dummy are otherwise not comparable. Standardise first (mean 0, standard deviation 1), inside the pipeline, so that each cross-validation fold is standardised with its own training data.

λ is chosen by five-fold cross-validation. Two choices are standard (Hastie, Tibshirani and Friedman 2009):

- $\lambda_{\min}$: the penalty with the lowest cross-validated error, the best for **prediction**;
- λ_{1se}: the largest penalty whose error is within one standard error of the minimum, the more conservative choice for **selection**, because it keeps only predictors whose contribution is clearly above noise.

```
import matplotlib.pyplot as plt

Xs = StandardScaler().fit_transform(X)
cvfit = LassoCV(cv=KFold(5, shuffle=True, random_state=0), random_state=0).fit(Xs, y)
mse, se = cvfit.mse_path_.mean(1), cvfit.mse_path_.std(1) / np.sqrt(5)
i_min = mse.argmin()
lam_min = cvfit.alphas_[i_min]
lam_1se = cvfit.alphas_[mse <= mse[i_min] + se[i_min]].max()
print(f"lambda_min = {lam_min:.3f}   lambda_1se = {lam_1se:.3f}")

fig, ax = plt.subplots()
ax.errorbar(np.log(cvfit.alphas_), mse, yerr=se, fmt="o", ms=3, color="#0033a0", ecolor="#9fb3d6", capsize=0)
for lam, lab in ((lam_min, "λ min"), (lam_1se, "λ 1se")):
```

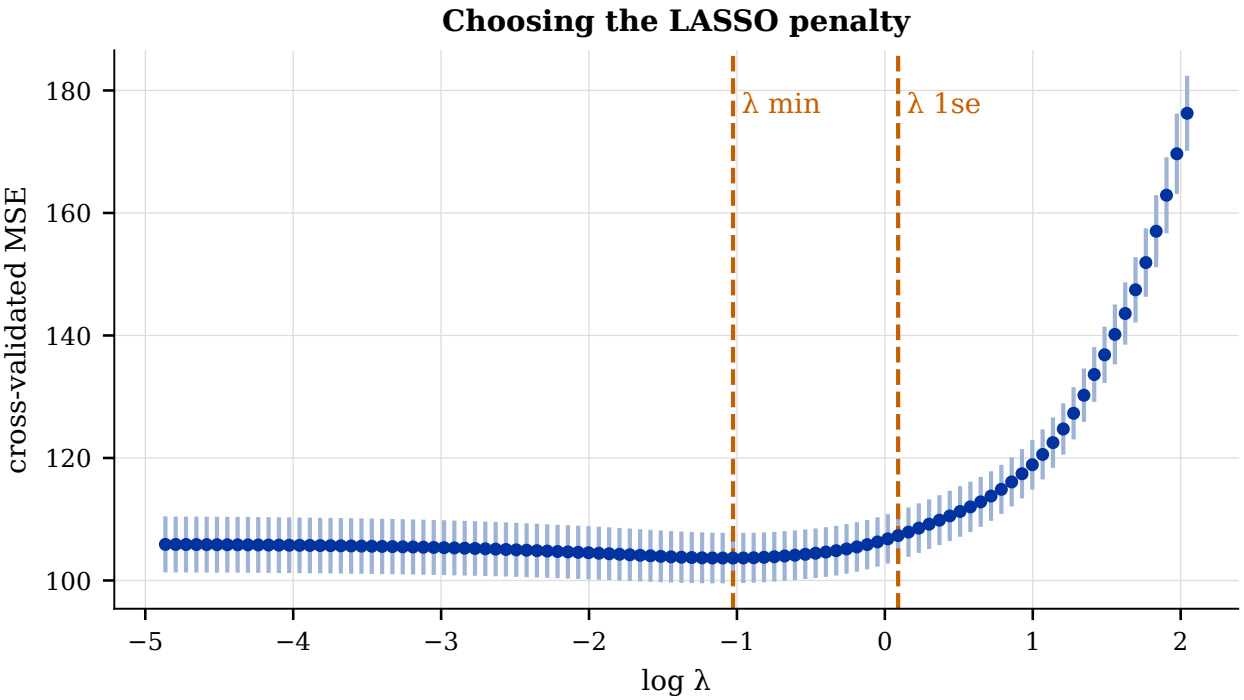


Figure 32.1. Standardise, then choose the penalty by cross-validation: Choosing the LASSO penalty.

```
    ax.axvline(np.log(lam), ls="--", color="#c46000"); ax.text(np.log(lam), mse.max(), " " + lab, color="#
  ↪ c46000")
ax.set_xlabel("log λ"); ax.set_ylabel("cross-validated MSE"); ax.set_title("Choosing the LASSO penalty")
lambda_min = 0.358   lambda_1se = 1.094
```

The error curve is flat near its minimum and rises steeply once the penalty starts to remove the strong predictors. The one-standard-error rule moves to the right along the flat part: a simpler model that predicts almost as well.

32.4 Which predictors survive?

```
def fit_at(lam):
    return Lasso(alpha=lam, max_iter=50000).fit(Xs, y).coef_

tab = pd.DataFrame({"λ min": fit_at(lam_min), "λ 1se": fit_at(lam_1se)}, index=X.columns).round(3)
tab["kept at 1se"] = np.where(tab["λ 1se"] != 0, "yes", "")
print(tab.sort_values("λ 1se", key=abs, ascending=False))
kept = list(tab.index[tab["λ 1se"] != 0])
print(f"\nkept at lambda_1se: {len(kept)} of {X.shape[1]} predictors ({len(kept) / X.shape[1]:.0%})")
```

```
          λ min  λ 1se kept at 1se
colgpa    5.996  5.667         yes
actmth    2.284  2.209         yes
calculus  1.300  0.838         yes
hsgpa     0.823  0.315         yes
work     -0.911 -0.308         yes
male      0.776  0.089         yes
study    -0.000 -0.000
econhs   -0.000 -0.000
age      -0.000 -0.000
acteng    0.208  0.000
attexc    0.000  0.000
attgood  -0.000 -0.000
fathcoll  0.000  0.000
mothcoll -0.000  0.000

kept at lambda_1se: 6 of 14 predictors (43%)
```

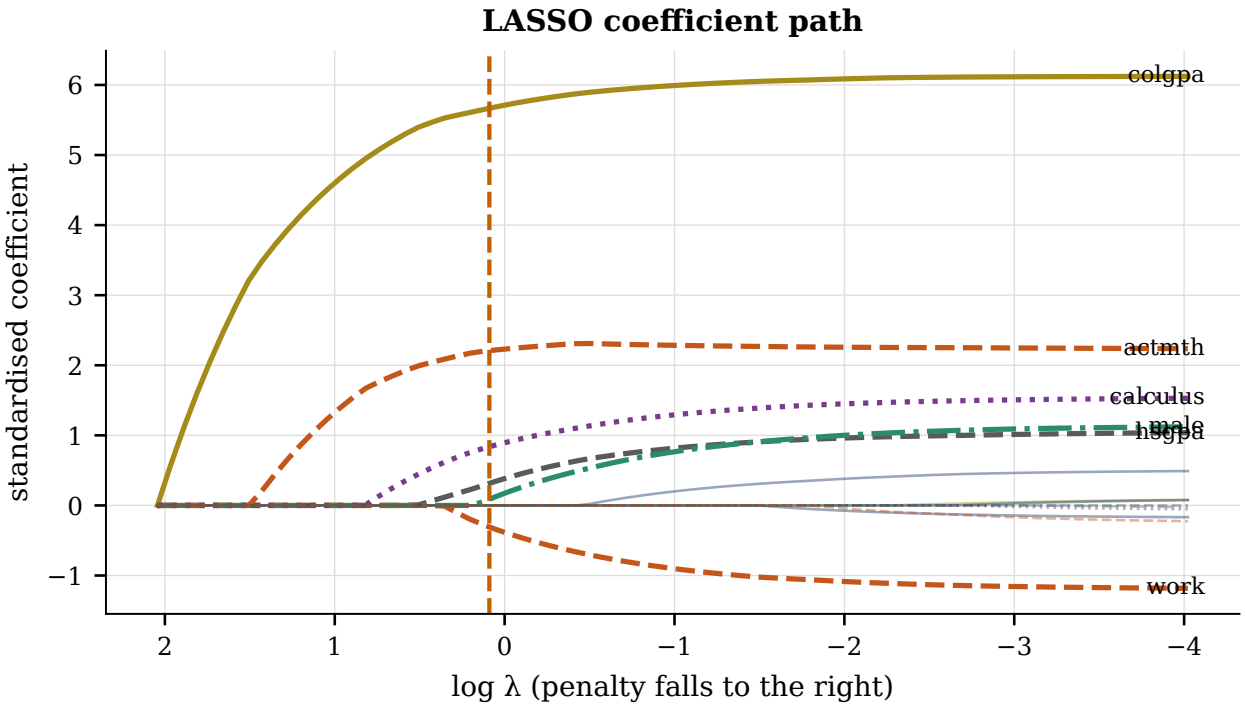


Figure 32.2. Which predictors survive?: LASSO coefficient path.

The coefficients are in units of the score per **one standard deviation** of each predictor, so their sizes can be compared directly. At λ_{1se} LASSO keeps fewer than half of the predictors: the problem is sparse, which is the condition for LASSO in the decision rules of Chapter 1 (Rule A4). College GPA and the ACT maths score carry most of the signal; high-school GPA, calculus, work hours and sex add smaller amounts.

The coefficient path shows the whole story at once: as the penalty falls (to the right), predictors enter one by one.

```
from sklearn.linear_model import lasso_path

alphas, coefs, _ = lasso_path(Xs, y, alphas=np.logspace(np.log10(lam_min / 20), np.log10(cvfit.alphas_.max()),
  ↪ 80))
fig, ax = plt.subplots()
for j, name in enumerate(X.columns):
    ax.plot(np.log(alphas), coefs[j], lw=1.6 if name in kept else 0.8, alpha=1 if name in kept else 0.45)
    if name in kept:
        ax.text(np.log(alphas[-1]) - 0.1, coefs[j, -1], name, fontsize=7, ha="right", va="center")
ax.axvline(np.log(lam_1se), ls="--", color="#c46000"); ax.invert_xaxis()
ax.set_xlabel("log λ (penalty falls to the right)"); ax.set_ylabel("standardised coefficient"); ax.set_title("
  ↪ LASSO coefficient path")
```

32.5 How stable is the selection?

A single LASSO fit can be fragile: with a slightly different sample a borderline predictor may enter or leave. A simple check is to refit LASSO on many bootstrap samples, with the same λ_{1se}, and count how often each predictor is kept (the idea behind *stability selection*; Meinshausen and Bühlmann 2010).

```
rng = np.random.default_rng(0)
B, picks = 200, np.zeros(X.shape[1])
for b in range(B):
    idx = rng.integers(0, len(y), len(y))
    picks += Lasso(alpha=lam_1se, max_iter=50000).fit(Xs[idx], y.iloc[idx]).coef_ != 0
freq = pd.Series(picks / B, index=X.columns).sort_values(ascending=False)
print((freq * 100).round(0).astype(int).astype(str).add(" %").to_string())
```

```
colgpa      100 %
actmth      100 %
```

```
calculus     97 %
hsgpa        79 %
work         77 %
male         55 %
acteng       16 %
age          12 %
fathcoll      2 %
econhs        0 %
mothcoll      0 %
attexc        0 %
study         0 %
attgood       0 %
```

Predictors kept in nearly every bootstrap sample are robust findings. Predictors kept about half the time are borderline: report them as such and do not build an argument on them.

> **Reading the frequencies**
>
> A common rule is to call a predictor stable if it is selected in at least 80 % of the resamples (Meinshausen and Bühlmann suggest thresholds between 60 % and 90 %). Below that, the evidence that it adds predictive power is weak, even if the full-sample fit keeps it.

32.6 Accuracy on data the model has not seen

Cross-validation chose λ, so the cross-validated error at $\lambda_{\min}$ is slightly optimistic. The honest test is **nested** cross-validation: an outer loop holds out each fold in turn, and the whole procedure, including the choice of λ, is run on the other four.

```
from sklearn.dummy import DummyRegressor

outer = KFold(5, shuffle=True, random_state=1)
models = {"mean only": DummyRegressor(),
          "OLS, all 14": make_pipeline(StandardScaler(), LinearRegression()),
          "LASSO λ( min)": make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0)),
          "random forest": forest}
res = {k: (-cross_val_score(m, X, y, cv=outer, scoring="neg_mean_squared_error")) ** 0.5 for k, m in models.
  ↪ items()}
out = pd.DataFrame({"RMSE": {k: v.mean() for k, v in res.items()}, "s.d. across folds": {k: v.std() for k, v
  ↪ in res.items()}}).round(2)
out["R2 (out of sample)"] = (1 - out["RMSE"] ** 2 / y.var()).round(3)
print(out)
```

```
               RMSE  s.d. across folds  R2 (out of sample)
mean only     13.29               0.56               0.002
OLS, all 14   10.26               0.59               0.405
LASSO λ( min) 10.18               0.57               0.415
random forest 10.34               0.63               0.396
```

LASSO predicts slightly better than OLS with all fourteen predictors (an RMSE of 10.18 against 10.26 points), while setting five of them to zero even at $\lambda_{\min}$; both explain about 40 % of the variance out of sample, far better than predicting the mean. The forest is worse. When OLS and LASSO predict equally well, the advantage of LASSO is the **interpretation**: it shows which few predictors carry the signal.

32.7 Post-LASSO OLS, and why its p-values are not valid

A common next step is to re-estimate OLS on the predictors LASSO selected (*post-LASSO*; Belloni and Chernozhukov 2013). This removes the shrinkage bias in the kept coefficients and gives them their natural units.

```
import statsmodels.api as sm

post = sm.OLS(y, sm.add_constant(X[kept])).fit(cov_type="HC1")
print(post.summary2().tables[1][["Coef.", "Std.Err.", "P>|z|"]].round(3))
```

```
          Coef.  Std.Err.  P>|z|
const    12.266     3.661  0.001
work     -0.129     0.040  0.001
```

```
colgpa    11.474     0.815  0.000
hsgpa      3.244     1.180  0.006
actmth     0.652     0.114  0.000
male       2.161     0.738  0.003
calculus   3.198     0.822  0.000
```

Do not report these p-values as tests

The predictors were chosen *because* they fit well in this sample, so the usual standard errors are too small and the p-values too optimistic (the post-selection inference problem). Report post-LASSO coefficients as descriptions of the selected model. If you need a valid test of one variable's effect, use the double-selection or DML methods in Part IV, which are designed for exactly that.

32.8 Write it up

```
final = pd.DataFrame({"LASSO coefficient (per s.d.)": tab.loc[kept, "λ 1se"],
                      "post-LASSO OLS (natural units)": post.params[kept].round(3),
                      "selected in bootstrap": (freq[kept] * 100).round(0).astype(int).astype(str) + " %"})
print(final)
print(f"\nNested CV RMSE: LASSO {out.loc['LASSO λ( min)', 'RMSE']}, OLS {out.loc['OLS, all 14', 'RMSE']}, mean-
  ↪ only {out.loc['mean only', 'RMSE']}")
```

```
         LASSO coefficient (per s.d.)  post-LASSO OLS (natural units) selected in bootstrap
work                           -0.308                          -0.129                  77 %
colgpa                          5.667                          11.474                 100 %
hsgpa                           0.315                           3.244                  79 %
actmth                          2.209                           0.652                 100 %
male                            0.089                           2.161                  55 %
calculus                        0.838                           3.198                  97 %

Nested CV RMSE: LASSO 10.18, OLS 10.26, mean-only 13.29
```

How to write it: results

We select predictors of the course score with LASSO, choosing the penalty by five-fold cross-validation and using the one-standard-error rule for selection. Of the fourteen candidate predictors, LASSO keeps six (Table X): college GPA and the ACT mathematics score are the strongest, followed by high-school GPA, a calculus background, hours of paid work (negative) and sex. College GPA and the ACT mathematics score are selected in every one of 200 bootstrap resamples and calculus in 97 %; high-school GPA and work hours in about 80 %, and sex in only 55 %, so we treat sex as a borderline predictor. In nested cross-validation, LASSO predicts the score with an RMSE of 10.2 points (out-of-sample R^2 of 0.42), slightly better than OLS with all fourteen predictors and far better than the sample mean (13.3 points); a random forest does not improve on it. Because the predictors were chosen on the same data, we report the post-LASSO coefficients as descriptive and do not interpret their p-values as tests.

Checklist for the paper

- State the aim (prediction or selection) and why a penalised model is used.
- Report how λ was chosen (CV folds, λ min or λ 1se) and that predictors were standardised.
- Show the selected predictors, the coefficient path, and the selection frequencies.
- Report out-of-sample accuracy against OLS and a mean-only benchmark (nested CV).
- Do not give causal readings or post-selection p-values.

CHAPTER 33

Ridge Regression

Applies when. Linear, many small effects

Overview. Use this chapter when the relationship is close to linear, no two predictors are strongly correlated, and most predictors carry some signal, so that LASSO keeps more than half of them. Ridge shrinks every coefficient towards zero without dropping any. The chapter takes one real dataset from the question to the finished results table.

Data. 663 U.S. men, 1980 wages (Blackburn and Neumark; Wooldridge wage2)

Software. Python · scikit-learn · statsmodels

33.1 The question and the model

How well can a man's wage be predicted from his ability, schooling, experience and background? The Blackburn–Neumark data, used throughout Wooldridge's textbook, record 15 candidate predictors for 663 men with complete data: two test scores (IQ and the Knowledge of the World of Work test), schooling, experience, tenure, age, marital status, race, region, urban residence, siblings, birth order and both parents' education. Unlike the exam-score example in Chapter 32, nearly all of these plausibly matter a little. That is the setting for ridge.

Ridge regression (Hoerl and Kennard 1970) adds a **squared** penalty to the least-squares objective:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

The squared penalty pulls every coefficient towards zero but, unlike the LASSO penalty, never sets one exactly to zero. It trades a little bias for a large reduction in variance, and it has a closed-form solution: $\hat{\beta} = (X'X + \lambda I)^{-1} X'y$.

Variable	Definition	Variable	Definition
`lwage`	Log monthly wage (outcome)	`married`	1 if married
`hours`	Average weekly hours	`black`	1 if black
`IQ`	IQ score	`south`, `urban`	1 if in the South / an SMSA
`KWW`	Knowledge of world of work score	`sibs`	Number of siblings
`educ`	Years of schooling	`brthord`	Birth order
`exper`, `tenure`	Years of experience / with employer	`meduc`, `feduc`	Mother's / father's schooling
`age`	Age in years		

When is ridge the right choice?

When the true model is *dense* (many predictors with small effects), LASSO's habit of setting coefficients to zero throws away real signal, and ridge predicts better. When it is *sparse* (a few strong effects), LASSO wins. The decision rules of Chapter 1 decide with the share of predictors LASSO keeps at the one-standard-error penalty: half or more means dense, so ridge.

33.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import RidgeCV, Ridge, LassoCV, Lasso, LinearRegression
from sklearn.ensemble import RandomForestRegressor

df = pd.read_csv("wage2.csv")
y, X = df["lwage"], df.drop(columns="lwage")
print(f"N = {len(df)} men, p = {X.shape[1]} predictors")
r = X.corr().abs().where(~np.eye(X.shape[1], dtype=bool))
print(f"largest |r| between predictors: {r.max().max():.2f} ({' and '.join(r.stack().idxmax())})")

cv = KFold(5, shuffle=True, random_state=0)
rmse = lambda m: (-cross_val_score(m, X, y, cv=cv, scoring="neg_mean_squared_error").mean()) ** 0.5
r_l = rmse(make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0)))
r_f = rmse(RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=0.5, random_state=0))
print(f"cross-validated RMSE: LASSO {r_l:.4f}, random forest {r_f:.4f}, forest gain {1 - r_f / r_l:+.1%}")

Xs = StandardScaler().fit_transform(X)
lc = LassoCV(cv=KFold(5, shuffle=True, random_state=0), random_state=0).fit(Xs, y)
m, se = lc.mse_path_.mean(1), lc.mse_path_.std(1) / np.sqrt(5)
lam_1se = lc.alphas_[m <= m.min() + se[m.argmin()]].max()
kept = X.columns[Lasso(alpha=lam_1se).fit(Xs, y).coef_ != 0]
print(f"LASSO at lambda_1se keeps {len(kept)} of {X.shape[1]} predictors ({len(kept) / X.shape[1]:.0%})")
```

```
N = 663 men, p = 15 predictors
largest |r| between predictors: 0.58 (sibs and brthord)
cross-validated RMSE: LASSO 0.3551, random forest 0.3594, forest gain -1.2%
LASSO at lambda_1se keeps 11 of 15 predictors (73%)
```

All three conditions for ridge hold: the forest does not beat the linear model, no correlation exceeds 0.8, and LASSO keeps well over half of the predictors even at its conservative penalty. The signal is spread across many variables.

> **How to write it: data section**
>
> *The sample contains 663 men with complete records from the Blackburn–Neumark extract of the National Longitudinal Survey. The outcome is the log monthly wage in 1980. Fifteen predictors describe measured ability, schooling, labour-market experience, demographics and family background. A random forest does not improve on a linear model in five-fold cross-validation, and a LASSO fit retains most predictors, indicating many small effects; we therefore use ridge regression.*

33.3 Standardise, then choose the penalty by cross-validation

The penalty acts on the size of the coefficients, so the predictors must be on a common scale; standardise them inside the pipeline. Then search a grid of λ values. For ridge, leave-one-out cross-validation has a closed form and is almost free (`RidgeCV` uses it by default), but five-fold CV is shown here to match the other chapters.

```
import matplotlib.pyplot as plt

lams = np.logspace(-2, 4, 60)
cvr = [(-cross_val_score(make_pipeline(StandardScaler(), Ridge(alpha=a)), X, y, cv=cv, scoring="
  ↪ neg_mean_squared_error")) for a in lams]
mse = np.array([c.mean() for c in cvr]); sd = np.array([c.std() / np.sqrt(5) for c in cvr])
lam = lams[mse.argmin()]
print(f"best lambda = {lam:.1f}; CV RMSE = {mse.min() ** .5:.4f} (OLS, lambda = 0: {rmse(make_pipeline(
  ↪ StandardScaler(), LinearRegression())):.4f})")

fig, ax = plt.subplots()
ax.errorbar(np.log10(lams), mse, yerr=sd, fmt="o", ms=3, color="#0033a0", ecolor="#9fb3d6")
```

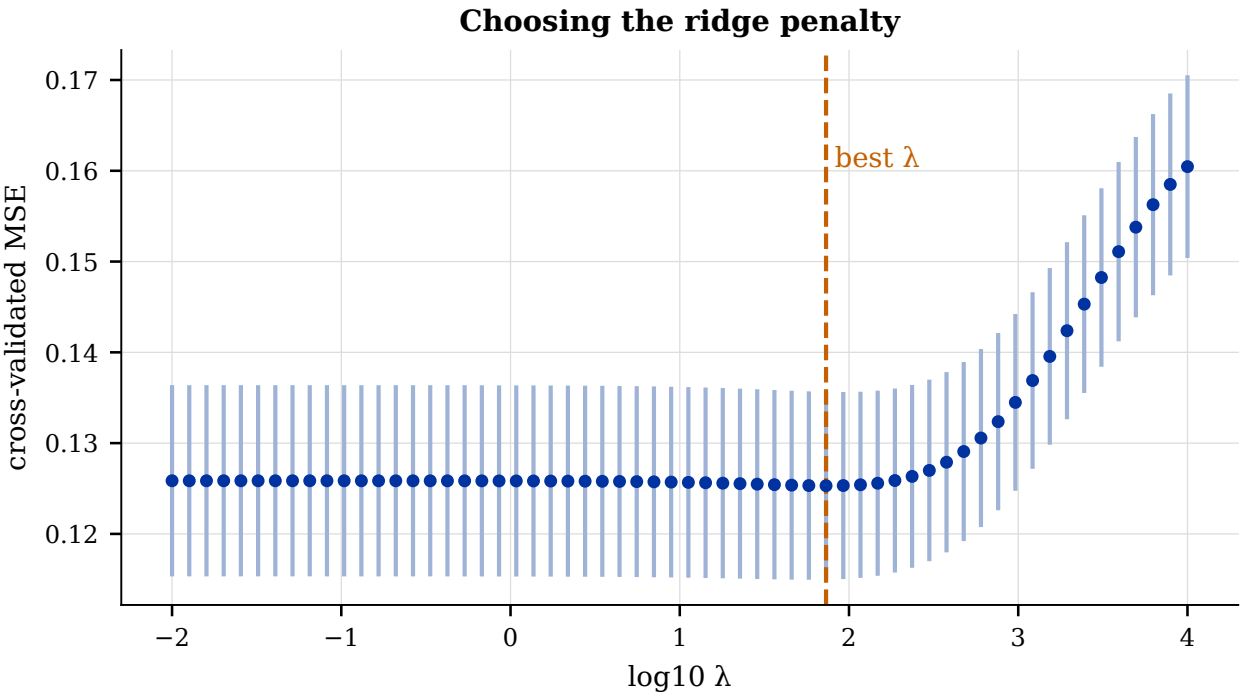


Figure 33.1. Standardise, then choose the penalty by cross-validation: Choosing the ridge penalty.

```
ax.axvline(np.log10(lam), ls="--", color="#c46000"); ax.text(np.log10(lam), mse.max(), " best λ", color="#
  ↪ c46000")
ax.set_xlabel("log10 λ"); ax.set_ylabel("cross-validated MSE"); ax.set_title("Choosing the ridge penalty")
```

```
best lambda = 73.2; CV RMSE = 0.3540 (OLS, lambda = 0: 0.3548)
```

The error curve is shallow: moderate shrinkage helps a little, and only very large penalties (which shrink everything towards zero) hurt. That is typical of a dense, well-measured problem: ridge improves on OLS at the margin rather than dramatically.

33.4 How much does ridge shrink each coefficient?

```
ols = LinearRegression().fit(Xs, y).coef_
rid = Ridge(alpha=lam).fit(Xs, y).coef_
tab = pd.DataFrame({"OLS": ols, "ridge": rid}, index=X.columns)
tab["shrinkage"] = (1 - tab["ridge"] / tab["OLS"]).map("{:.0%}".format)
print(tab.round(4).sort_values("ridge", key=abs, ascending=False))
```

```
            OLS   ridge shrinkage
urban    0.0879  0.0788       10%
educ     0.0920  0.0770       16%
married  0.0601  0.0539       10%
IQ       0.0468  0.0464        1%
hours   -0.0468 -0.0417       11%
exper    0.0457  0.0355       22%
tenure   0.0359  0.0341        5%
KWW      0.0281  0.0309      -10%
age      0.0279  0.0300       -8%
meduc    0.0272  0.0262        4%
black   -0.0288 -0.0245       15%
south   -0.0229 -0.0231       -1%
brthord -0.0274 -0.0224       18%
feduc    0.0184  0.0210      -14%
sibs     0.0215  0.0138       36%
```

Coefficients are per standard deviation of each predictor. Ridge keeps every predictor and shrinks the vector of coefficients as a whole; most individual estimates move towards zero (siblings by a third, experience by a fifth), but a few correlated ones grow slightly (the KWW score, age, father's education), because

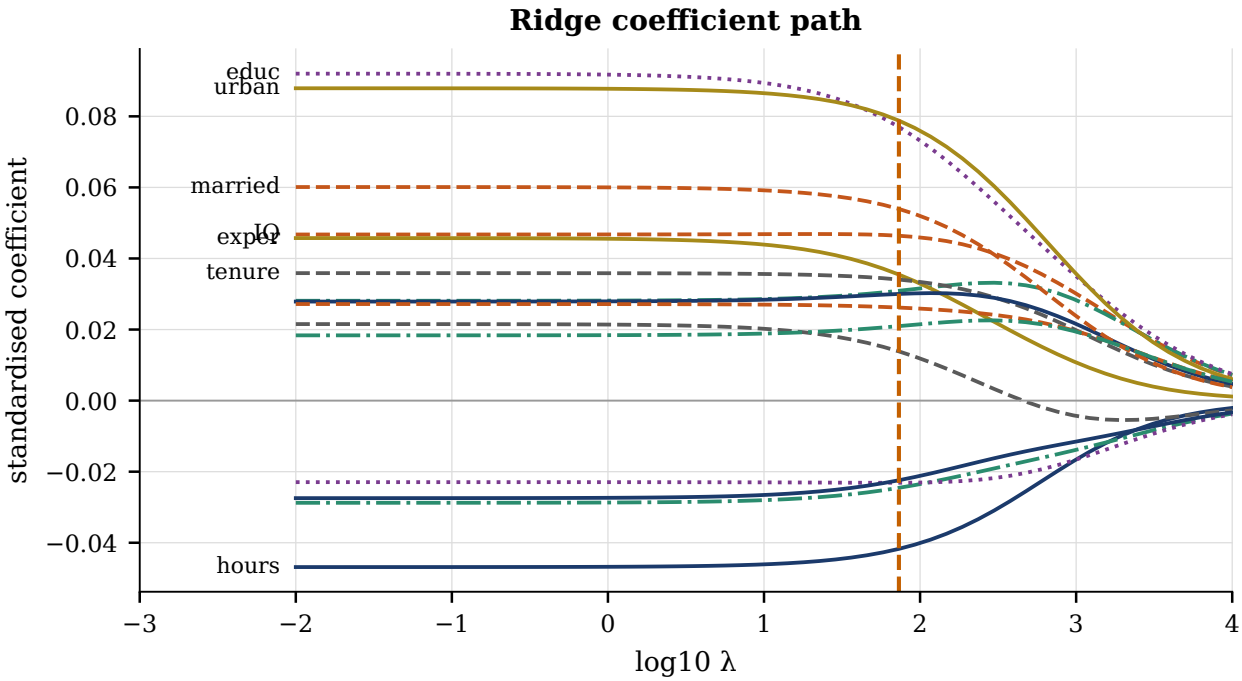


Figure 33.2. How much does ridge shrink each coefficient?: Ridge coefficient path.

ridge spreads weight across predictors that carry overlapping information. The ridge **path** shows how all coefficients move smoothly towards zero as the penalty grows, without any of them hitting zero:

```
path = np.array([Ridge(alpha=a).fit(Xs, y).coef_ for a in lams])
fig, ax = plt.subplots()
for j, name in enumerate(X.columns):
    ax.plot(np.log10(lams), path[:, j], lw=1.2)
    if abs(path[0, j]) > .03:
        ax.text(np.log10(lams[0]) - .1, path[0, j], name, fontsize=7, ha="right", va="center")
ax.axvline(np.log10(lam), ls="--", color="#c46000"); ax.axhline(0, color="#999", lw=.6)
ax.set_xlim(np.log10(lams[0]) - 1, np.log10(lams[-1])); ax.set_xlabel("log10 λ"); ax.set_ylabel("standardised
  ↪ coefficient"); ax.set_title("Ridge coefficient path")
```

33.5 Effective degrees of freedom

Ridge does not drop variables, so "number of predictors" no longer measures model complexity. The **effective degrees of freedom** do: with d_j the singular values of the standardised X,

$$\mathrm{df}(\lambda) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda},$$

which equals p at $\lambda = 0$ (OLS) and falls towards 0 as λ grows.

```
d = np.linalg.svd(Xs, compute_uv=False)
dof = lambda a: (d ** 2 / (d ** 2 + a)).sum()
print(f"effective degrees of freedom at the chosen lambda: {dof(lam):.1f} of {X.shape[1]}")
```

```
effective degrees of freedom at the chosen lambda: 13.0 of 15
```

Report this number: it tells the reader how much the penalty actually constrained the model.

33.6 Accuracy on data the model has not seen

Choosing λ on the same folds that measure the error flatters the model. Nested cross-validation repeats the whole procedure, including the choice of λ, inside each outer fold.

```
from sklearn.dummy import DummyRegressor


outer = KFold(5, shuffle=True, random_state=1)
```

```
models = {"mean only": DummyRegressor(),
          "OLS": make_pipeline(StandardScaler(), LinearRegression()),
          "ridge": make_pipeline(StandardScaler(), RidgeCV(alphas=lams)),
          "LASSO": make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0)),
          "random forest": RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=0.5,
  ↪ random_state=0)}
res = {k: (-cross_val_score(mm, X, y, cv=outer, scoring="neg_mean_squared_error")) ** 0.5 for k, mm in models.
  ↪ items()}
out = pd.DataFrame({"RMSE": {k: v.mean() for k, v in res.items()}, "s.d. across folds": {k: v.std() for k, v
  ↪ in res.items()}})
out["R2 (out of sample)"] = 1 - out["RMSE"] ** 2 / y.var()
print(out.round(4))
```

```
                RMSE  s.d. across folds  R2 (out of sample)
mean only     0.4117             0.0175              0.0027
OLS           0.3578             0.0239              0.2467
ridge         0.3567             0.0228              0.2510
LASSO         0.3587             0.0238              0.2426
random forest 0.3636             0.0180              0.2221
```

The differences between the three linear models are small relative to their variation across folds: with 663 observations and 15 predictors, OLS is not badly overfitted, and ridge's gain is modest. That is an honest and common finding. Ridge earns its place when p is large relative to n or when predictors are noisy; report the comparison either way.

33.7 Write it up

```
best = Ridge(alpha=lam).fit(Xs, y)
final = pd.DataFrame({"ridge (per s.d.)": best.coef_, "OLS (per s.d.)": ols}, index=X.columns).round(4)
print(final.sort_values("ridge (per s.d.)", key=abs, ascending=False))
print(f"\nlambda = {lam:.1f}, effective df = {dof(lam):.1f}; nested CV RMSE: ridge {out.loc['ridge', 'RMSE
  ↪ ']:.4f}, OLS {out.loc['OLS', 'RMSE']:.4f}, mean only {out.loc['mean only', 'RMSE']:.4f}")
```

```
         ridge (per s.d.)  OLS (per s.d.)
urban              0.0788          0.0879
educ               0.0770          0.0920
married            0.0539          0.0601
IQ                 0.0464          0.0468
hours             -0.0417         -0.0468
exper              0.0355          0.0457
tenure             0.0341          0.0359
KWW                0.0309          0.0281
age                0.0300          0.0279
meduc              0.0262          0.0272
black             -0.0245         -0.0288
south             -0.0231         -0.0229
brthord           -0.0224         -0.0274
feduc              0.0210          0.0184
sibs               0.0138          0.0215

lambda = 73.2, effective df = 13.0; nested CV RMSE: ridge 0.3567, OLS 0.3578, mean only 0.4117
```

How to write it: results

We predict the log wage with ridge regression on fifteen standardised predictors, choosing the penalty by five-fold cross-validation. Because LASSO retains most predictors even at its one-standard-error penalty, the signal appears to be spread across many variables, the setting in which ridge's proportional shrinkage is preferable to variable selection. The largest standardised coefficients belong to urban residence, schooling, marital status, IQ and weekly hours (negative), followed by experience and tenure (Table X); race and the South enter negatively. The chosen penalty ($\lambda = 73$) reduces the effective degrees of freedom from 15 to 13. In nested cross-validation ridge has the lowest error of the models compared (RMSE 0.357 against 0.358 for OLS and 0.359 for LASSO), although the differences are small relative to their variation across folds; all three explain about a quarter of the out-of-sample variance, and a random forest does not improve on them.

Checklist for the paper

- Explain why the problem is dense (LASSO keeps most predictors) and why ridge is used.
- State the standardisation, the λ grid and the CV scheme; report λ and the effective degrees of freedom.
- Show OLS and ridge coefficients side by side, and the ridge path.
- Report nested out-of-sample accuracy against OLS, LASSO and a mean-only benchmark.
- Ridge coefficients are shrunk on purpose: do not attach OLS-style standard errors to them.

CHAPTER 34

Elastic Net

Applies when. Linear, strongly correlated predictors

Overview. Use this chapter when the relationship is close to linear but some predictors are strongly correlated, as lags of the same series always are. LASSO picks one of a correlated group almost at random; the elastic net keeps or drops the group together. The example is a forecasting problem, so it also shows how to validate a model on time-series data without looking into the future.

Data. U.S. quarterly macro data, 1960–2009 (statsmodels macrodata)

Software. Python · scikit-learn · statsmodels

34.1 The question and the model

Can next quarter's U.S. inflation be forecast from the recent history of inflation, unemployment, interest rates, output and money? The file holds inflation and four lags of five series (20 predictors) for 198 quarters. The lags of each series are highly correlated with one another: inflation this quarter and last quarter move together. That is the setting for the elastic net (Zou and Hastie 2005), which combines the LASSO and ridge penalties:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2n} \sum_{t} \left(\pi_t - x_{t-1}'\beta\right)^2 + \lambda\left(\alpha \sum_{j} |\beta_j| + \frac{1-\alpha}{2} \sum_{j} \beta_j^2\right)$$

$\alpha = 1$ gives LASSO and $\alpha = 0$ ridge. In between, the ridge part makes correlated predictors share their weight, and the LASSO part still sets weak groups to zero. Both λ and α are chosen by cross-validation.

Predictor group	Columns	Series
Inflation	`infl_l1` … `infl_l4`	CPI inflation, annualised %
Unemployment	`unemp_l1` … `unemp_l4`	Unemployment rate, %
Interest rate	`tbilrate_l1` … `tbilrate_l4`	3-month Treasury bill rate, %
Output	`gdp_growth_l1` … `gdp_growth_l4`	Real GDP growth, annualised %
Money	`m1_growth_l1` … `m1_growth_l4`	M1 growth, annualised %

Every predictor is dated before the quarter being forecast, so the model uses only information that was available at the time.

34.2 Load the data and test for unit roots

Machine learning does not remove the need to check stationarity. A model trained on a trending series learns the trend's past range: a random forest can never predict a value it has not seen, and a linear model on I(1) series risks a spurious fit. So each series is tested first, exactly as in Part II, and I(1) series are differenced.

```
import numpy as np
import pandas as pd
from statsmodels.tsa.stattools import adfuller, kpss

df = pd.read_csv("macro.csv")
print(f"{len(df)} quarters, {df.quarter.iloc[0]} to {df.quarter.iloc[-1]}; {df.shape[1] - 2} predictors")
for v in ["infl", "unemp_l1", "tbilrate_l1", "gdp_growth_l1", "m1_growth_l1"]:
    a, k = adfuller(df[v], autolag="AIC")[1], kpss(df[v], nlags="auto")[1]
    ad = adfuller(df[v].diff().dropna(), autolag="AIC")[1]
```

```
    print(f"{v:14s} ADF p = {a:.3f}  KPSS p = {k:.3f}  ADF on difference p = {ad:.3f}  -> {'I(0)' if a < .05
  ↪ else 'I(1)'}")
```

```
198 quarters, 1960Q2 to 2009Q3; 20 predictors
infl           ADF p = 0.028  KPSS p = 0.057  ADF on difference p = 0.000  -> I(0)
unemp_l1       ADF p = 0.024  KPSS p = 0.100  ADF on difference p = 0.005  -> I(0)
tbilrate_l1    ADF p = 0.299  KPSS p = 0.055  ADF on difference p = 0.000  -> I(1)
gdp_growth_l1  ADF p = 0.000  KPSS p = 0.100  ADF on difference p = 0.000  -> I(0)
m1_growth_l1   ADF p = 0.004  KPSS p = 0.100  ADF on difference p = 0.000  -> I(0)
```

The T-bill rate has a unit root in levels (ADF does not reject, KPSS rejects stationarity) and is stationary after one difference; inflation, unemployment, GDP growth and money growth reject the unit root at 5 %. Following the decision rules of Chapter 1, the four T-bill lags enter in first differences.

```
tb = [c for c in df if c.startswith("tbilrate")]
df[tb] = df[tb].diff()
df = df.dropna().reset_index(drop=True)
y, X = df["infl"], df.drop(columns=["quarter", "infl"])
r = X.corr().abs().where(~np.eye(X.shape[1], dtype=bool))
print(f"{len(df)} quarters after differencing; largest |r| between predictors: {r.max().max():.2f} ({' and '.
  ↪ join(r.stack().idxmax())})")
```

```
197 quarters after differencing; largest |r| between predictors: 0.97 (unemp_l3 and unemp_l4)
```

Several pairs of lags are correlated above 0.9, far above the 0.8 threshold: the elastic net's case.

34.3 Validate forward in time, never backward

Ordinary k-fold cross-validation shuffles the observations, so a model would be trained on 2005 and tested on 1975, using the future to predict the past. For time series, every validation fold must come **after** the data the model was trained on. `TimeSeriesSplit` does this: each split trains on an expanding window of the past and validates on the next block.

```
from sklearn.model_selection import TimeSeriesSplit, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import ElasticNetCV, LassoCV, RidgeCV
from sklearn.ensemble import RandomForestRegressor

tscv = TimeSeriesSplit(n_splits=5)
for i, (tr, te) in enumerate(tscv.split(X)):
    print(f"split {i + 1}: train {df.quarter[tr[0]]}-{df.quarter[tr[-1]]}  validate {df.quarter[te[0]]}-{df.
  ↪ quarter[te[-1]]}")

rmse = lambda m: (-cross_val_score(m, X, y, cv=tscv, scoring="neg_mean_squared_error").mean()) ** .5
enet = make_pipeline(StandardScaler(), ElasticNetCV(l1_ratio=[.1, .3, .5, .7, .9, .99], cv=tscv, max_iter
  ↪ =50000))
cands = {"LASSO": make_pipeline(StandardScaler(), LassoCV(cv=tscv, max_iter=50000)),
         "ridge": make_pipeline(StandardScaler(), RidgeCV(alphas=np.logspace(-2, 4, 50))),
         "elastic net": enet,
         "random forest": RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=.5,
  ↪ random_state=0)}
for k, m in cands.items():
    print(f"{k:14s} rolling-origin CV RMSE = {rmse(m):.3f}")
```

```
split 1: train 1960Q3-1969Q3  validate 1969Q4-1977Q3
split 2: train 1960Q3-1977Q3  validate 1977Q4-1985Q3
split 3: train 1960Q3-1985Q3  validate 1985Q4-1993Q3
split 4: train 1960Q3-1993Q3  validate 1993Q4-2001Q3
split 5: train 1960Q3-2001Q3  validate 2001Q4-2009Q3
LASSO          rolling-origin CV RMSE = 2.853
ridge          rolling-origin CV RMSE = 2.831
elastic net    rolling-origin CV RMSE = 2.803
random forest  rolling-origin CV RMSE = 3.113
```

The elastic net has the lowest rolling-origin error, LASSO is close, and the random forest is clearly worse: a nonlinear model does not help here (Rule A3), and with lags correlated above 0.8 the decision rules chooses the elastic net (Rule A4).

34.4 Grouped selection: elastic net against LASSO

Fit both on the full sample and compare which lags they keep. Within a group of correlated lags, LASSO tends to keep one or two and drop the rest; the elastic net, the more so the smaller its mixing weight α, spreads the weight. To see the effect, the table also shows an elastic net with α fixed at 0.3.

```
Xs = StandardScaler().fit_transform(X)
en = ElasticNetCV(l1_ratio=[.1, .3, .5, .7, .9, .99], cv=tscv, max_iter=50000).fit(Xs, y)
la = LassoCV(cv=tscv, max_iter=50000).fit(Xs, y)
print(f"elastic net: alpha (mixing) = {en.l1_ratio_}, lambda = {en.alpha_:.3f}")
from sklearn.linear_model import ElasticNet
e3 = ElasticNetCV(l1_ratio=.3, cv=tscv, max_iter=50000).fit(Xs, y)
tab = pd.DataFrame({"LASSO": la.coef_, "elastic net (CV α)": en.coef_, "elastic net α( = 0.3)": e3.coef_},
  ↪ index=X.columns).round(3)
print(tab)
print(f"\nnon-zero: LASSO {int((la.coef_ != 0).sum())}, elastic net (CV) {int((en.coef_ != 0).sum())}, elastic
  ↪  net α( = 0.3) {int((e3.coef_ != 0).sum())} of {X.shape[1]}")
```

```
elastic net: alpha (mixing) = 0.9, lambda = 0.054
              LASSO  elastic net (CV α)  elastic net α( = 0.3)
infl_l1       0.819               0.816                  0.770
infl_l2       0.631               0.630                  0.619
infl_l3       0.916               0.909                  0.830
infl_l4       0.054               0.061                  0.153
unemp_l1     -0.000              -0.000                 -0.007
unemp_l2     -0.000              -0.000                 -0.000
unemp_l3     -0.000              -0.000                 -0.000
unemp_l4     -0.124              -0.122                 -0.117
tbilrate_l1   0.500               0.498                  0.467
tbilrate_l2   0.000               0.000                  0.000
tbilrate_l3   0.400               0.400                  0.402
tbilrate_l4   0.089               0.089                  0.106
gdp_growth_l1 0.073               0.073                  0.089
gdp_growth_l2 -0.145             -0.143                 -0.135
gdp_growth_l3 0.192               0.191                  0.189
gdp_growth_l4 0.000               0.000                  0.000
m1_growth_l1  0.111               0.111                  0.115
m1_growth_l2  0.109               0.110                  0.127
m1_growth_l3  0.422               0.419                  0.393
m1_growth_l4 -0.000              -0.000                  0.000

non-zero: LASSO 14, elastic net (CV) 14, elastic net α( = 0.3) 15 of 20
```

```
import matplotlib.pyplot as plt

grp = X.columns.str.replace(r"_l\d$", "", regex=True)
fig, ax = plt.subplots()
xpos = np.arange(len(X.columns))
ax.bar(xpos - .2, la.coef_, .4, label="LASSO"); ax.bar(xpos + .2, e3.coef_, .4, label="elastic net, α = 0.3")
ax.set_xticks(xpos); ax.set_xticklabels(X.columns, rotation=70, fontsize=7); ax.axhline(0, color="#999", lw=.6)
  ↪
for b in range(4, len(X.columns), 4): ax.axvline(b - .5, color="#ccc", lw=.6)
ax.set_ylabel("standardised coefficient"); ax.set_title("Lags of the same series: LASSO and elastic net"); ax.
  ↪ legend()
```

Cross-validation chooses a mixing weight of $\alpha = 0.9$, close to LASSO, so here the two give almost the same coefficients: the data prefer a mostly-LASSO penalty with a little ridge stabilisation. The $\alpha = 0.3$ column shows what a stronger ridge component does: weight is spread across correlated lags (the third lag of inflation loses weight to the fourth, which almost triples) and one more lag (`unemp_l1`) enters. Report the cross-validated α, and show that the main conclusions do not depend on it. In every version the first three lags of inflation dominate, followed by the T-bill rate and money growth.

34.5 A real forecasting test: recursive one-step-ahead forecasts

The honest test of a forecasting model is to simulate using it in real time. From 1990 onwards, each quarter the model is re-estimated on all data up to the previous quarter (penalty re-chosen each time) and forecasts

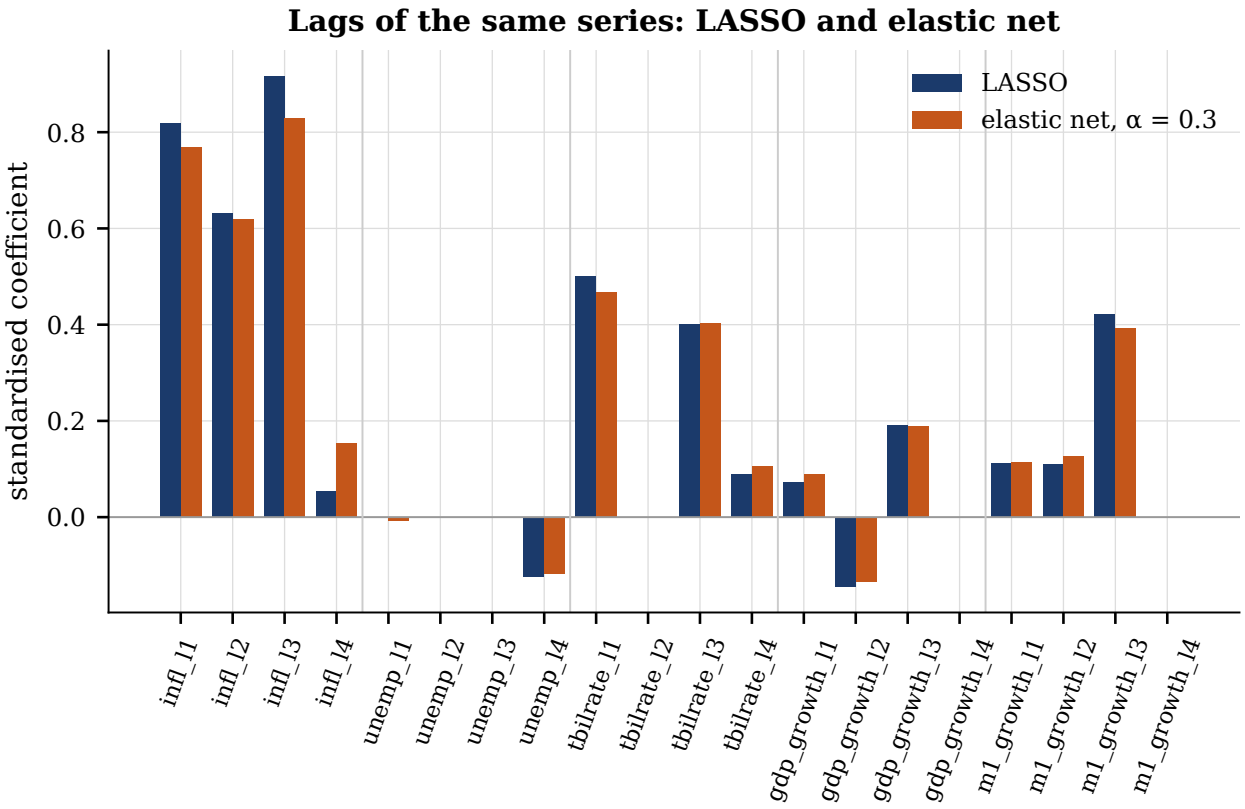


Figure 34.1. Grouped selection: elastic net against LASSO: Lags of the same series: LASSO and elastic net.

the next quarter. The benchmark every forecasting paper must beat is a simple **AR(4)** model of inflation on its own four lags, estimated the same way.

```
from sklearn.linear_model import LinearRegression

start = df.index[df.quarter == "1990Q1"][0]
fc = {"AR(4)": [], "elastic net": [], "LASSO": []}
for t in range(start, len(df)):
    Xtr, ytr, Xte = X.iloc[:t], y.iloc[:t], X.iloc[[t]]
    fc["AR(4)"].append(LinearRegression().fit(Xtr.filter(like="infl_l"), ytr).predict(Xte.filter(like="infl_l")
  ↪ )[0])
    fc["elastic net"].append(make_pipeline(StandardScaler(), ElasticNetCV(l1_ratio=[.1, .5, .9, .99], cv=
  ↪ TimeSeriesSplit(5), max_iter=50000)).fit(Xtr, ytr).predict(Xte)[0])
    fc["LASSO"].append(make_pipeline(StandardScaler(), LassoCV(cv=TimeSeriesSplit(5), max_iter=50000)).fit(Xtr,
  ↪  ytr).predict(Xte)[0])
F = pd.DataFrame(fc, index=df.quarter[start:]); F["actual"] = y.iloc[start:].to_numpy()
err = F.drop(columns="actual").sub(F["actual"], axis=0)
print(f"{len(F)} one-step-ahead forecasts, {F.index[0]} to {F.index[-1]}")
print((err ** 2).mean().pow(.5).round(3).rename("RMSE").to_frame().assign(relative_to_AR=lambda d: (d.RMSE / d.
  ↪ RMSE["AR(4)"]).round(3)))
```

```
79 one-step-ahead forecasts, 1990Q1 to 2009Q3
             RMSE  relative_to_AR
AR(4)       2.518           1.000
elastic net 2.672           1.061
LASSO       2.666           1.059
```

A ratio below 1 means the model beats the AR(4) benchmark. Here it does **not**: both penalised models forecast about 6 % worse than the AR(4) over 1990–2009. Whether a difference is statistically significant is tested with the **Diebold–Mariano** test on the loss differences (with a Newey–West variance, because forecast errors are serially correlated):

```
import statsmodels.api as sm

for m in ["elastic net", "LASSO"]:
```

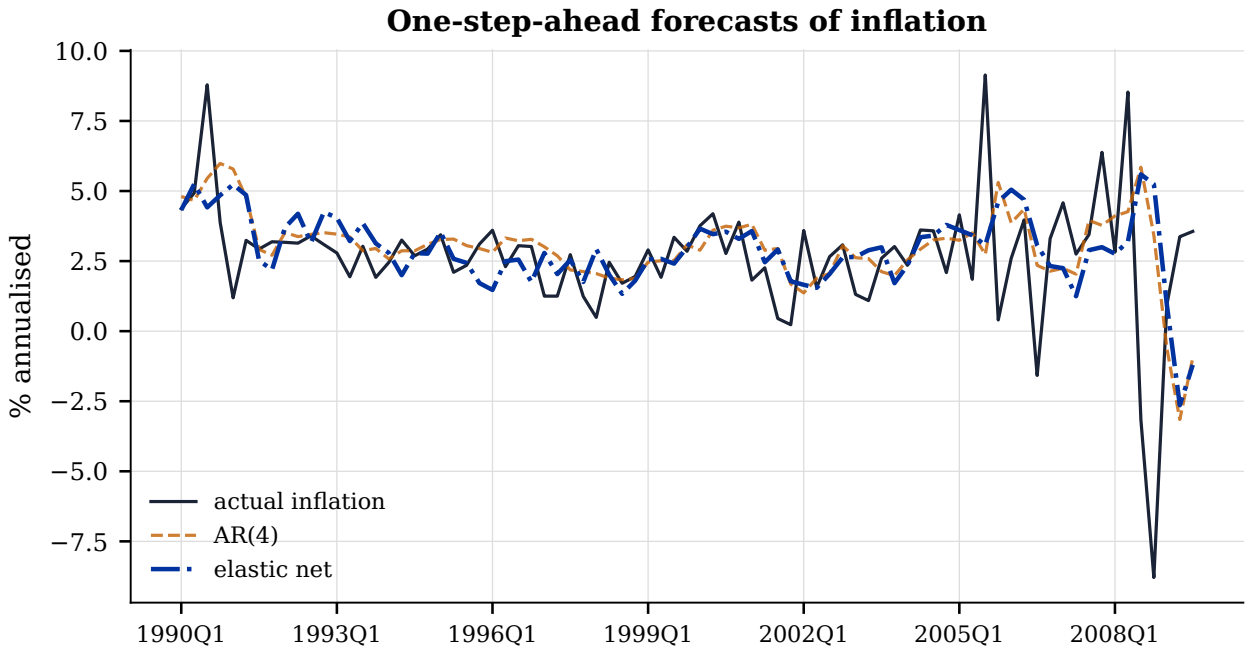


Figure 34.2. A real forecasting test: recursive one-step-ahead forecasts: One-step-ahead forecasts of inflation.

```
    dl = err[m] ** 2 - err["AR(4)"] ** 2
    dm = sm.OLS(dl.to_numpy(), np.ones(len(dl))).fit(cov_type="HAC", cov_kwds={"maxlags": 4})
    print(f"{m:12s} vs AR(4): mean loss difference {dl.mean():+.3f}, DM t = {dm.tvalues[0]:.2f}, p = {dm.
↪ pvalues[0]:.3f}")

fig, ax = plt.subplots()
xx = np.arange(len(F))
ax.plot(xx, F["actual"], color="#1b2437", lw=1, label="actual inflation")
ax.plot(xx, F["AR(4)"], color="#c46000", lw=1, alpha=.8, label="AR(4)")
ax.plot(xx, F["elastic net"], color="#0033a0", lw=1.4, label="elastic net")
ax.set_xticks(xx[::12]); ax.set_xticklabels(F.index[::12], fontsize=7)
ax.set_ylabel("% annualised"); ax.set_title("One-step-ahead forecasts of inflation"); ax.legend(fontsize=7)
```

```
elastic net  vs AR(4): mean loss difference +0.796, DM t = 1.36, p = 0.175
LASSO        vs AR(4): mean loss difference +0.765, DM t = 1.28, p = 0.200
```

34.6 Write it up

The difference is not significant ($p \approx 0.2$): the twenty extra predictors neither help nor clearly hurt. This is a well-known result, not a failure of the method: since the mid-1980s, inflation has been hard to forecast with anything better than its own recent history (Atkeson and Ohanian 2001; Stock and Watson 2007). A paper must report it honestly. The elastic net is still the right tool for the question *which* lags carry information; the forecasting test says that, out of sample, the information beyond inflation's own lags is small.

How to write it: method and results

We forecast quarterly CPI inflation with an elastic net on four lags of inflation, unemployment, the change in the Treasury-bill rate (the level has a unit root), real GDP growth and M1 growth. Because lags of the same series are highly correlated ($|r|$ up to 0.97), we prefer the elastic net to LASSO, which selects arbitrarily within correlated groups. Penalty and mixing weight are chosen by rolling-origin cross-validation, so that no model is ever validated on data earlier than its training sample. Cross-validation selects a mixing weight of 0.9, close to LASSO; the first three lags of inflation, the change in the T-bill rate and lagged money growth carry most of the weight. In recursive one-step-ahead forecasts over 1990Q1–2009Q3, re-estimating the model each quarter, the elastic net does not improve on an AR(4) benchmark: its RMSE is 2.67 percentage points against 2.52, and a Diebold–Mariano test with HAC variance does not reject equal accuracy ($p = 0.18$). Consistent with Atkeson

and Ohanian (2001), the additional macroeconomic predictors contain little information about future inflation beyond its own history.

Checklist for the paper

- Test every series for a unit root and difference I(1) series before any learning.
- Date every predictor before the forecast target; never let future information in.
- Use rolling-origin (expanding-window) validation, not shuffled k-fold.
- Report the chosen mixing weight α and penalty λ, and which lags survive.
- Compare recursive out-of-sample forecasts with an AR benchmark; report the RMSE ratio and a Diebold–Mariano test.

CHAPTER 35

Random Forest

Applies when. Nonlinear, fewer than 1,000 observations

Overview. Use this chapter when a random forest predicts clearly better than LASSO out of sample, so the outcome depends on the predictors nonlinearly or through interactions, and the sample is too small to tune a boosted model safely. It takes one real dataset from the question to the finished results table: growing and tuning the forest, testing it honestly, and opening the black box with permutation importance and partial dependence.

Data. 353 Major League Baseball players, 1993 salaries (Wooldridge mlb1)

Software. Python · scikit-learn

35.1 The question and the model

How well can a baseball player's salary be predicted from his career record? The file holds the 1993 salaries of 353 Major League Baseball players and fifteen career statistics: years in the league, games, at-bats, runs, hits, home runs, runs batted in, batting average, walks, strikeouts, stolen bases, fielding percentage, all-star seasons and race. Salaries are not linear in performance: rookies are paid little whatever they do, salaries jump once players can bargain, and stars are paid far more than a linear model would suggest.

A **random forest** (Breiman 2001) averages many regression trees. Each tree is grown on a bootstrap sample of the players, and at every split it may choose only from a random subset of the predictors. A single deep tree fits the sample closely but is very unstable; averaging hundreds of decorrelated trees keeps the flexibility and removes most of the instability:

$$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x), \qquad T_b \text{ grown on bootstrap sample } b \text{ with random feature subsets.}$$

What a forest can and cannot do

A forest captures thresholds, curvature and interactions without being told about them. It cannot extrapolate: a forest never predicts a value outside the range of the outcomes it was trained on. And it gives no coefficients; Sections 35.6 and 35.7 show how to describe what it has learned.

35.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV, LinearRegression
from sklearn.ensemble import RandomForestRegressor

df = pd.read_csv("mlb1.csv")
y, X = df["lsalary"], df.drop(columns="lsalary").astype(float)
print(f"N = {len(df)} players, p = {X.shape[1]} predictors; log salary from {y.min():.2f} to {y.max():.2f}")

cv = KFold(5, shuffle=True, random_state=0)
rmse = lambda m, XX=X: (-cross_val_score(m, XX, y, cv=cv, scoring="neg_mean_squared_error").mean()) ** .5
```

```
r_l = rmse(make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0)))
r_f = rmse(RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=.5, random_state=0))
print(f"cross-validated RMSE: LASSO {r_l:.3f}, random forest {r_f:.3f}, forest gain {1 - r_f / r_l:.1%}")
```

```
N = 353 players, p = 15 predictors; log salary from 11.60 to 15.66
cross-validated RMSE: LASSO 0.888, random forest 0.635, forest gain 28.5%
```

The forest's error is far below LASSO's, much more than the 5 % threshold (Rule A3): salaries are strongly nonlinear in the career record. With only 353 players, below 1,000, the decision rules of Chapter 1 choose a random forest over gradient boosting (Rule A5).

> **How to write it: data section**
>
> *The sample contains 353 Major League Baseball players with 1993 salaries and career statistics. The outcome is the log salary. In five-fold cross-validation a random forest predicts log salaries with an error about 28 % lower than LASSO, indicating strong nonlinearities and interactions between career statistics and pay; given the modest sample, we use a random forest rather than a boosted model, which has more tuning parameters to overfit.*

35.3 Grow the forest: how many trees?

More trees never overfit: they only reduce the variance of the average. The question is how many are needed before the error stops falling. The **out-of-bag (OOB)** error answers it for free: each player is left out of about a third of the bootstrap samples, and the trees that did not see him predict his salary.

```
import matplotlib.pyplot as plt

sizes = [25, 50, 100, 200, 300, 500, 800]
oob = []
for n in sizes:
    f = RandomForestRegressor(n_estimators=n, min_samples_leaf=5, max_features=.5, oob_score=True,
  ↪ random_state=0).fit(X, y)
    oob.append(np.sqrt(np.mean((y - f.oob_prediction_) ** 2)))
print(pd.Series(np.round(oob, 4), index=sizes, name="OOB RMSE").to_string())
fig, ax = plt.subplots()
ax.plot(sizes, oob, "o-"); ax.set_xscale("log"); ax.set_xlabel("number of trees"); ax.set_ylabel("out-of-bag
  ↪ RMSE"); ax.set_title("More trees: the error levels off")
```

```
25     0.6524
50     0.6426
100    0.6393
200    0.6382
300    0.6369
500    0.6370
800    0.6371
```

The error is essentially flat beyond a few hundred trees. Use 500: cheap, and safely on the flat part.

35.4 Tune the two parameters that matter

Two parameters control how flexible each tree is:

- **`min_samples_leaf`**: the smallest number of players in a final leaf. Larger leaves make smoother, less variable predictions.
- **`max_features`**: the share of predictors each split may consider. Smaller values decorrelate the trees more.

```
from sklearn.model_selection import GridSearchCV

grid = GridSearchCV(RandomForestRegressor(n_estimators=500, random_state=0),
                    {"min_samples_leaf": [1, 3, 5, 10, 20], "max_features": [.2, .33, .5, .75, 1.0]},
                    cv=cv, scoring="neg_mean_squared_error").fit(X, y)
res = pd.DataFrame(grid.cv_results_["params"]).assign(RMSE=np.sqrt(-grid.cv_results_["mean_test_score"]))
print(res.pivot(index="min_samples_leaf", columns="max_features", values="RMSE").round(3))
print("best:", grid.best_params_, f"RMSE {np.sqrt(-grid.best_score_):.3f}")
```

```
max_features       0.20   0.33   0.50   0.75   1.00
min_samples_leaf
1                 0.639  0.641  0.643  0.647  0.648
```

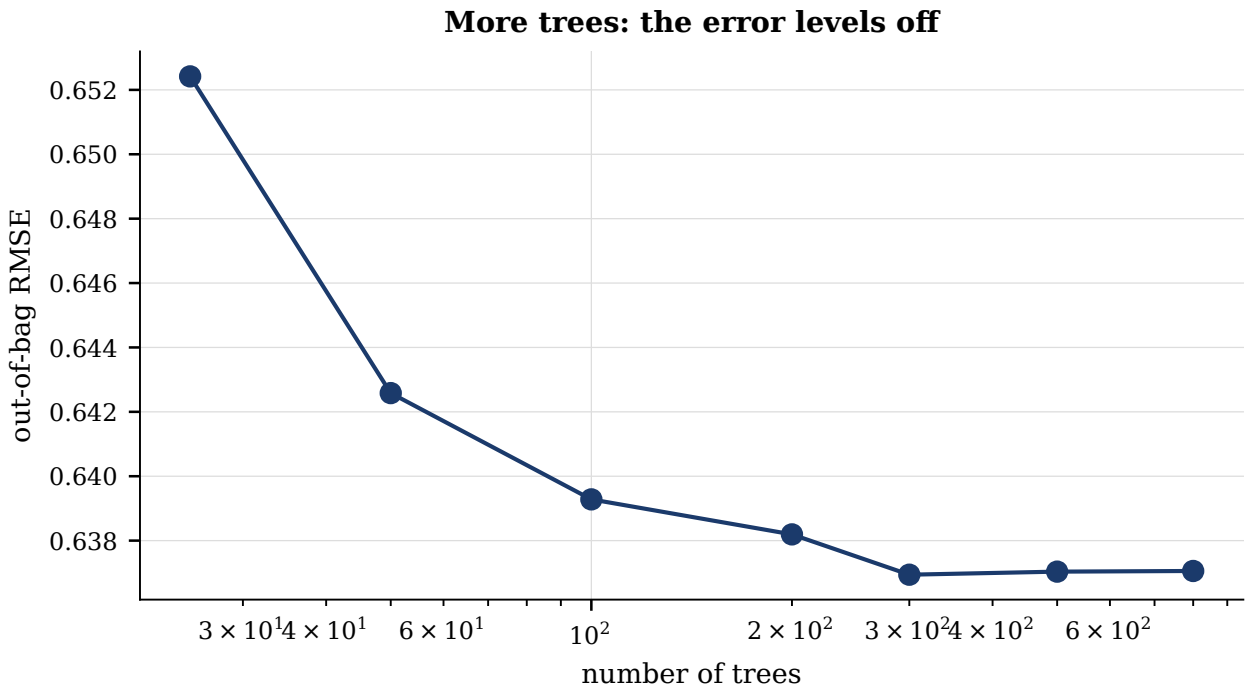


Figure 35.1. Grow the forest: how many trees?: More trees: the error levels off.

```
3                0.638  0.636  0.640  0.644  0.648
5                0.634  0.635  0.637  0.638  0.642
10               0.636  0.637  0.636  0.636  0.639
20               0.647  0.645  0.642  0.640  0.640
best: {'max_features': 0.2, 'min_samples_leaf': 5} RMSE 0.634
```

The surface is flat near the optimum: forests are forgiving, and any setting in the low-error region gives similar predictions. Report the chosen values and the grid.

35.5 Accuracy on data the model has not seen

The tuning above used the same folds that measured the error, which flatters the winner. Nested cross-validation repeats the tuning inside each outer fold.

```
from sklearn.dummy import DummyRegressor

outer = KFold(5, shuffle=True, random_state=1)
inner = GridSearchCV(RandomForestRegressor(n_estimators=300, random_state=0),
                     {"min_samples_leaf": [1, 5, 10], "max_features": [.33, .5]}, cv=5, scoring="
  ↪ neg_mean_squared_error")
models = {"mean only": DummyRegressor(), "OLS": make_pipeline(StandardScaler(), LinearRegression()),
          "LASSO": make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0)), "random forest (tuned)":
  ↪ inner}
out = {}
for k, m in models.items():
    s = (-cross_val_score(m, X, y, cv=outer, scoring="neg_mean_squared_error")) ** .5
    out[k] = {"RMSE": s.mean(), "s.d. across folds": s.std(), "R2 (out of sample)": 1 - s.mean() ** 2 / y.var()
  ↪ }
out = pd.DataFrame(out).T.round(3)
print(out)
```

```
                      RMSE  s.d. across folds  R2 (out of sample)
mean only            1.183              0.078              -0.001
OLS                  0.919              0.091               0.397
LASSO                0.895              0.042               0.427
random forest (tuned)  0.637              0.026               0.710
```

The forest explains a much larger share of the out-of-sample variance than OLS or LASSO. That gap is what justifies its loss of interpretability.

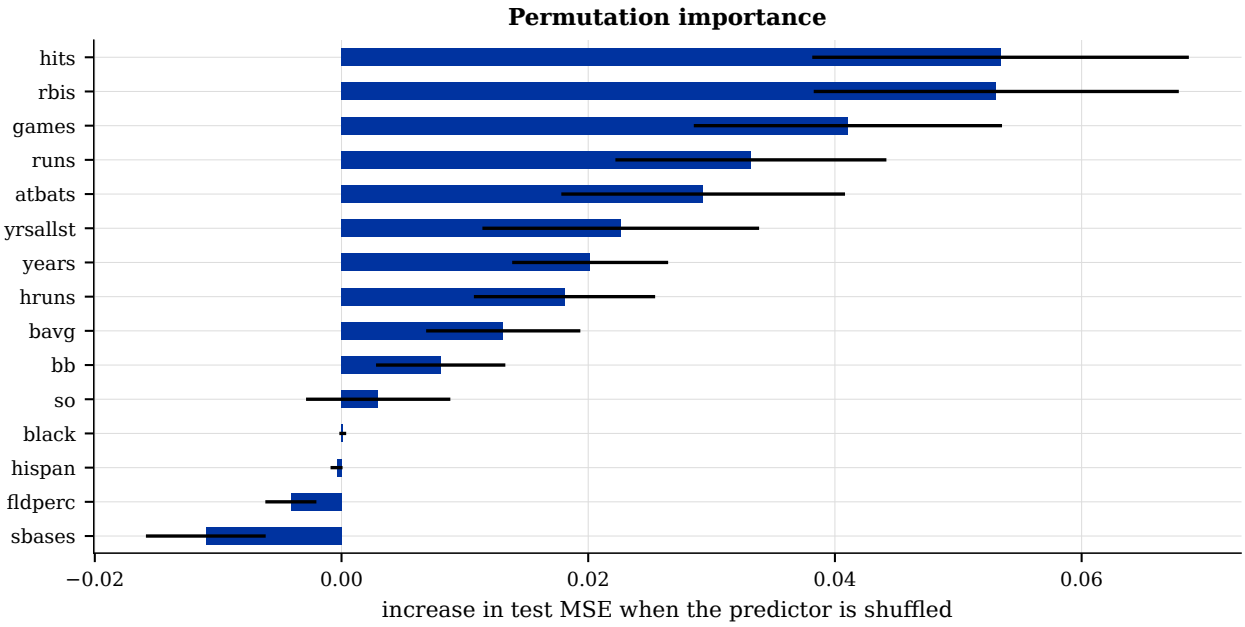


Figure 35.2. Which predictors matter? Permutation importance: Permutation importance.

35.6 Which predictors matter? Permutation importance

The forest has no coefficients. **Permutation importance** measures how much the held-out error rises when one predictor's values are shuffled, breaking its link with salary while leaving everything else intact. Compute it on a test set the forest did not see; the impurity-based importance that scikit-learn reports by default is computed on the training data and favours predictors with many distinct values.

```
from sklearn.model_selection import train_test_split
from sklearn.inspection import permutation_importance

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=.3, random_state=0)
rf = RandomForestRegressor(n_estimators=500, random_state=0, **grid.best_params_).fit(Xtr, ytr)
print(f"test RMSE {np.sqrt(np.mean((yte - rf.predict(Xte)) ** 2)):.3f}")
pi = permutation_importance(rf, Xte, yte, n_repeats=30, random_state=0, scoring="neg_mean_squared_error")
imp = pd.DataFrame({"increase in MSE": pi.importances_mean, "s.d.": pi.importances_std}, index=X.columns).
  ↪ sort_values("increase in MSE", ascending=False)
print(imp.round(4))
fig, ax = plt.subplots(figsize=(8, 3.6))
imp["increase in MSE"][::-1].plot.barh(xerr=imp["s.d."][::-1], ax=ax, color="#0033a0")
ax.set_xlabel("increase in test MSE when the predictor is shuffled"); ax.set_title("Permutation importance")
```

```
test RMSE 0.610
          increase in MSE    s.d.
hits               0.0534  0.0153
rbis               0.0531  0.0148
games              0.0411  0.0125
runs               0.0332  0.0110
atbats             0.0293  0.0115
yrsallst           0.0226  0.0112
years              0.0202  0.0063
hruns              0.0181  0.0073
bavg               0.0131  0.0063
bb                 0.0080  0.0052
so                 0.0030  0.0058
black              0.0001  0.0003
hispan            -0.0004  0.0005
fldperc           -0.0041  0.0021
sbases            -0.0110  0.0049
```

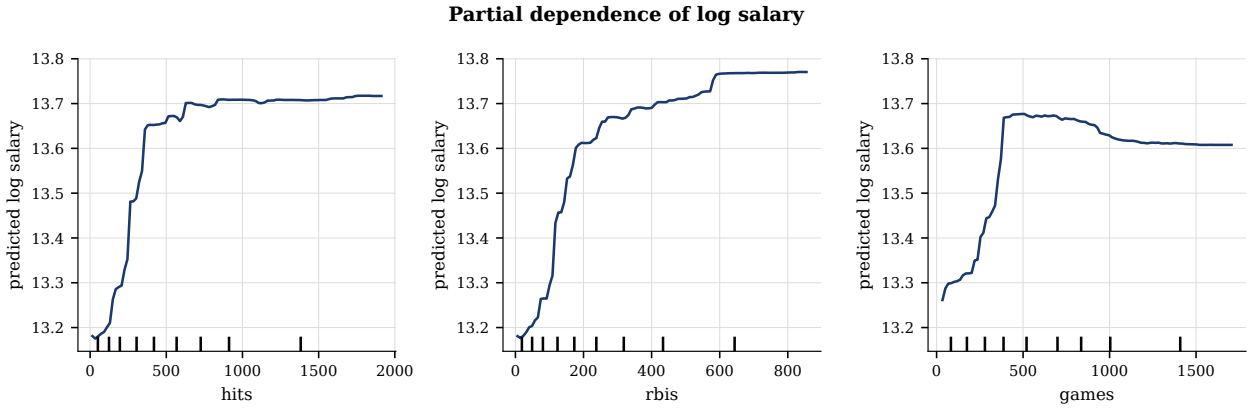


Figure 35.3. How do the key predictors affect salary? Partial dependence: Partial dependence of log salary.

> **Correlated predictors share their importance**
>
> At-bats, hits, runs and runs batted in are almost the same variable (correlations above 0.95). When one is shuffled, the forest still learns from the others, so each looks less important than the group is. Read the importances by group, or shuffle a correlated group together, before concluding that a statistic does not matter.

35.7 How do the key predictors affect salary? Partial dependence

A **partial-dependence plot** shows the average prediction as one predictor is varied over its range, with the others held at their observed values. It is the forest's answer to "what is the coefficient?", but it can bend.

```
from sklearn.inspection import PartialDependenceDisplay

rf_all = RandomForestRegressor(n_estimators=500, random_state=0, **grid.best_params_).fit(X, y)
top = list(imp.index[:3])
fig, axes = plt.subplots(1, 3, figsize=(9, 3))
PartialDependenceDisplay.from_estimator(rf_all, X, top, ax=axes, kind="average")
for a in axes: a.set_ylabel("predicted log salary")
fig.suptitle("Partial dependence of log salary", fontsize=10, fontweight="bold")
fig.tight_layout()
```

The curves are far from straight lines: salary rises steeply over the first years of a career and then flattens, which is the nonlinearity a linear model misses and the reason the forest predicts so much better.

35.8 Write it up

```
print(out.loc[["mean only", "OLS", "LASSO", "random forest (tuned)"]])
print("\ntuned parameters:", grid.best_params_, "| trees: 500")
print("top predictors:", ", ".join(imp.index[:5]))
```

```
                       RMSE  s.d. across folds  R2 (out of sample)
mean only             1.183              0.078              -0.001
OLS                   0.919              0.091               0.397
LASSO                 0.895              0.042               0.427
random forest (tuned)  0.637              0.026               0.710

tuned parameters: {'max_features': 0.2, 'min_samples_leaf': 5} | trees: 500
top predictors: hits, rbis, games, runs, atbats
```

> **How to write it: results**
>
> *We predict the log salary with a random forest of 500 trees, tuning the minimum leaf size and the share of predictors considered at each split by five-fold cross-validation. In nested cross-validation the forest's out-of-sample error is far below that of OLS and LASSO, and it explains a substantially larger share of the variance (Table X). Permutation importance on a held-out sample shows that years in the league and cumulative career output (at-bats, runs batted in, hits) dominate the predictions. Partial-dependence plots show that predicted*

salaries rise steeply over the first years of a career and then level off, a nonlinearity that linear models cannot capture.

Checklist for the paper

- Show why a nonlinear model is needed (forest vs LASSO, cross-validated).
- Report the number of trees, the tuned parameters and the tuning grid.
- Report nested out-of-sample accuracy against OLS, LASSO and the mean.
- Use permutation importance on held-out data, and read correlated predictors as a group.
- Show partial-dependence plots for the main predictors; do not call them causal effects.

CHAPTER 36

Gradient Boosting (XGBoost)

Applies when. Nonlinear, at least 1,000 observations

Overview. Use this chapter when a random forest predicts clearly better than LASSO, so the relationship is nonlinear, and there are enough observations (1,000 or more) to tune a boosted model without overfitting. It takes one real dataset through early stopping, tuning, a fair comparison with the forest, and interpretation with SHAP values and partial dependence.

Data. 6,822 injured workers in Kentucky and Michigan (Meyer, Viscusi and Durbin; Wooldridge injury)

Software. Python · XGBoost · scikit-learn

36.1 The question and the model

How long will an injured worker stay out of work? Workers' compensation insurers and labour economists both want to know. The file covers 6,822 workers injured in Kentucky and Michigan around benefit increases studied by Meyer, Viscusi and Durbin (1995). The outcome is the log number of weeks on temporary disability benefits; the eighteen predictors describe the worker (age, sex, marital status, previous earnings), the injury (medical costs, hospitalisation, body part, occupational disease), the industry, the state, and whether the injury happened after the benefit increase. Durations depend on these in strongly nonlinear ways: medical costs matter a great deal up to a point, and the type of injury changes how much.

Gradient boosting (Friedman 2001) builds trees one after another. Each new tree is fitted to the errors that the current model still makes and is added with a small weight η, the learning rate:

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \eta\, T_m(x), \qquad T_m \text{ fitted to the residuals of } \hat{f}_{m-1}.$$

A random forest averages many independent deep trees; boosting adds many small dependent ones, each correcting the last. **XGBoost** (Chen and Guestrin 2016) adds penalties on tree size and leaf weights, and is the standard implementation in applied work.

Variable	Definition	Variable	Definition
`ldurat`	Log weeks on benefits (outcome)	`ltotmed`	Log total medical costs
`afchnge`	1 if injured after the benefit increase	`hosp`	1 if hospitalised
`highearn`	1 if earnings above the benefit cap	`head` … `lowextr`	Body part injured (dummies)
`lprewage`	Log weekly wage before the injury	`occdis`	1 if occupational disease
`age`, `male`, `married`	Worker characteristics	`manuf`, `construc`	Industry dummies
`ky`	1 if Kentucky (0 = Michigan)		

36.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV, LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

```
df = pd.read_csv("injury.csv")
y, X = df["ldurat"], df.drop(columns="ldurat").astype(float)
print(f"N = {len(df)} workers, p = {X.shape[1]} predictors; median duration {np.exp(y.median()):.1f} weeks")

cv = KFold(5, shuffle=True, random_state=0)
rmse = lambda m: (-cross_val_score(m, X, y, cv=cv, scoring="neg_mean_squared_error").mean()) ** .5
r_l = rmse(make_pipeline(StandardScaler(), LassoCV(cv=5)))
r_f = rmse(RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=.5, random_state=0, n_jobs
  ↪ =-1))
print(f"cross-validated RMSE: LASSO {r_l:.4f}, random forest {r_f:.4f}, forest gain {1 - r_f / r_l:.1%}")
```

```
N = 6822 workers, p = 18 predictors; median duration 4.0 weeks
cross-validated RMSE: LASSO 1.0832, random forest 0.9828, forest gain 9.3%
```

The forest beats LASSO by more than the 5 % threshold (Rule A3), so the relationship is nonlinear, and with almost 7,000 workers there is enough data to tune a boosted model (Rule A5).

Validation must respect the structure of the data

Here every row is a different worker, so ordinary shuffled folds are fine. If the same unit appears several times (a firm over years, a route in several years), keep all of its rows in the same fold (`GroupKFold`); otherwise the model memorises units and looks better than it is. With time series, validate forward in time only (see Chapter 34). The nonlinearity test is only as honest as its folds: on panel data, shuffled folds can make a forest look far better than it is.

How to write it: data section

The sample contains 6,822 workers injured in Kentucky and Michigan (Meyer, Viscusi and Durbin 1995). The outcome is the log number of weeks on temporary disability benefits; eighteen predictors describe the worker, the injury, the industry, the state and the timing of the injury relative to a benefit increase. In five-fold cross-validation a random forest predicts log durations with an error about 8 % lower than LASSO, so we model durations with gradient-boosted trees.

36.3 The parameters that matter, and early stopping

Boosting has more tuning parameters than a forest, and unlike a forest it **can** overfit if it runs too long. The ones that matter most:

- **learning rate** η: smaller is better but needs more trees; 0.03 to 0.1 is usual;
- **number of trees**: chosen by **early stopping**, i.e. stop adding trees when the error on a validation set has not improved for a while;
- **`max_depth`**: the depth of each tree, which sets how many variables can interact;
- **`min_child_weight`**, **`subsample`**, **`colsample_bytree`**: regularisation, as in a forest.

```
import xgboost as xgb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

Xtr, Xva, ytr, yva = train_test_split(X, y, test_size=.25, random_state=0)
m = xgb.XGBRegressor(n_estimators=3000, learning_rate=.03, max_depth=3, subsample=.8, colsample_bytree=.8,
                     min_child_weight=5, early_stopping_rounds=100, eval_metric="rmse", random_state=0, n_jobs
  ↪ =-1)
m.fit(Xtr, ytr, eval_set=[(Xtr, ytr), (Xva, yva)], verbose=False)
h = m.evals_result(); n_best = m.best_iteration + 1
print(f"early stopping: {n_best} trees; validation RMSE {h['validation_1']['rmse'][m.best_iteration]:.4f}")
fig, ax = plt.subplots()
ax.plot(h["validation_0"]["rmse"], label="training sample"); ax.plot(h["validation_1"]["rmse"], label="
  ↪ validation sample")
ax.axvline(m.best_iteration, ls="--", color="#c46000"); ax.set_ylim(.8, 1.3)
ax.set_xlabel("number of trees"); ax.set_ylabel("RMSE"); ax.set_title("Early stopping"); ax.legend()
```

```
early stopping: 230 trees; validation RMSE 1.0021
```

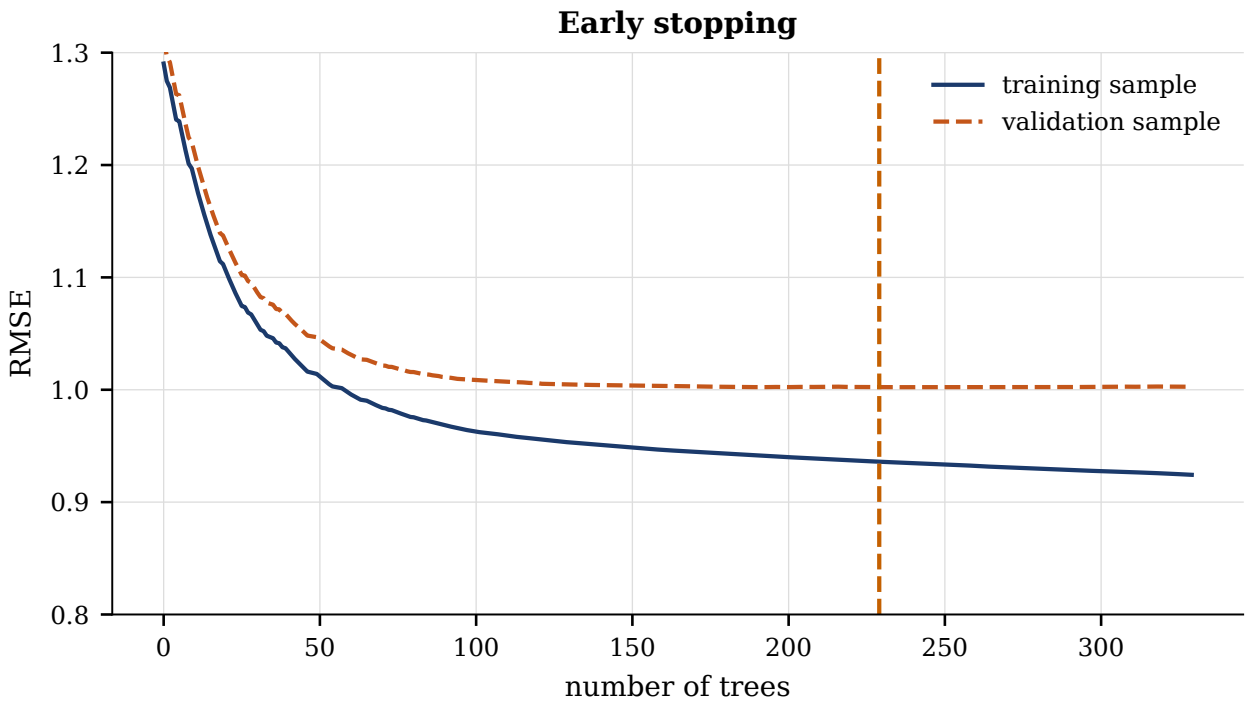


Figure 36.1. The parameters that matter, and early stopping: Early stopping.

The training error keeps falling as trees are added; the validation error falls, flattens and then creeps up as the model starts to fit noise. Early stopping picks the number of trees at the bottom of the validation curve.

36.4 Tune depth and regularisation

```
from sklearn.model_selection import GridSearchCV

base = xgb.XGBRegressor(n_estimators=n_best, learning_rate=.03, subsample=.8, colsample_bytree=.8,
  ↪ random_state=0, n_jobs=-1)
grid = GridSearchCV(base, {"max_depth": [2, 3, 4, 6], "min_child_weight": [1, 5, 20]}, cv=cv,
                    scoring="neg_mean_squared_error").fit(X, y)
res = pd.DataFrame(grid.cv_results_["params"]).assign(RMSE=np.sqrt(-grid.cv_results_["mean_test_score"]))
print(res.pivot(index="max_depth", columns="min_child_weight", values="RMSE").round(4))
best = grid.best_params_; print("best:", best)
```

```
min_child_weight      1       5       20
max_depth
2                 0.9831  0.9830  0.9832
3                 0.9759  0.9758  0.9761
4                 0.9768  0.9762  0.9761
6                 0.9830  0.9816  0.9795
best: {'max_depth': 3, 'min_child_weight': 5}
```

Trees of depth 3 do best: durations depend on interactions between a few variables at a time, not on deep combinations, and depth 6 overfits. The minimum leaf weight hardly matters.

36.5 A fair comparison with the alternatives

Nested cross-validation: the boosted model is tuned inside each outer training sample, then scored on the held-out fold.

```
from sklearn.dummy import DummyRegressor

outer = KFold(5, shuffle=True, random_state=1)
inner = GridSearchCV(xgb.XGBRegressor(n_estimators=n_best, learning_rate=.03, subsample=.8, colsample_bytree
  ↪ =.8, random_state=0, n_jobs=-1),
```

```
                    {"max_depth": [2, 3, 4], "min_child_weight": [5, 20]}, cv=4, scoring="
  ↪ neg_mean_squared_error")
models = {"mean only": DummyRegressor(), "OLS": make_pipeline(StandardScaler(), LinearRegression()),
          "LASSO": make_pipeline(StandardScaler(), LassoCV(cv=5)),
          "random forest": RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
          "XGBoost (tuned)": inner}
out = {}
for k, mm in models.items():
    s = (-cross_val_score(mm, X, y, cv=outer, scoring="neg_mean_squared_error")) ** .5
    out[k] = {"RMSE": s.mean(), "s.d. across folds": s.std(), "R2 (out of sample)": 1 - s.mean() ** 2 / y.var()
  ↪ }
out = pd.DataFrame(out).T.round(4)
print(out)
```

```
                   RMSE  s.d. across folds  R2 (out of sample)
mean only        1.3016             0.0373              0.0009
OLS              1.0830             0.0322              0.3083
LASSO            1.0828             0.0325              0.3086
random forest    0.9833             0.0200              0.4298
XGBoost (tuned)  0.9780             0.0197              0.4359
```

36.6 Open the black box: SHAP values

SHAP values (Lundberg and Lee 2017) split each prediction into additive contributions from each predictor, measured from the average prediction. XGBoost computes them exactly for trees (`pred_contribs=True`).

```
fin = xgb.XGBRegressor(n_estimators=n_best, learning_rate=.03, subsample=.8, colsample_bytree=.8, random_state
  ↪ =0, **best).fit(X, y)
shap = fin.get_booster().predict(xgb.DMatrix(X), pred_contribs=True)[:, :-1]     # last column = bias
imp = pd.Series(np.abs(shap).mean(0), index=X.columns).sort_values(ascending=False)
print("mean |SHAP| (log weeks):"); print(imp.round(4).to_string())

fig, axes = plt.subplots(1, 2, figsize=(9, 3.2))
imp[:10][::-1].plot.barh(ax=axes[0], color="#0033a0"); axes[0].set_xlabel("mean |SHAP value|"); axes[0].
  ↪ set_title("Importance (top 10)")
j = list(X.columns).index("ltotmed")
axes[1].scatter(X["ltotmed"], shap[:, j], s=3, c=X["hosp"], cmap="coolwarm", alpha=.5)
axes[1].set_xlabel("log medical costs"); axes[1].set_ylabel("SHAP value"); axes[1].set_title("Medical costs (
  ↪ colour: hospitalised)")
fig.tight_layout()
```

```
mean |SHAP| (log weeks):
ltotmed     0.6413
age         0.0677
head        0.0382
lprewage    0.0379
manuf       0.0274
hosp        0.0270
construc    0.0209
male        0.0208
lowextr     0.0094
ky          0.0084
lowback     0.0082
afchnge     0.0071
married     0.0050
trunk       0.0041
upextr      0.0023
occdis      0.0023
neck        0.0013
highearn    0.0011
```

36.7 Partial dependence

```
from sklearn.inspection import PartialDependenceDisplay
```

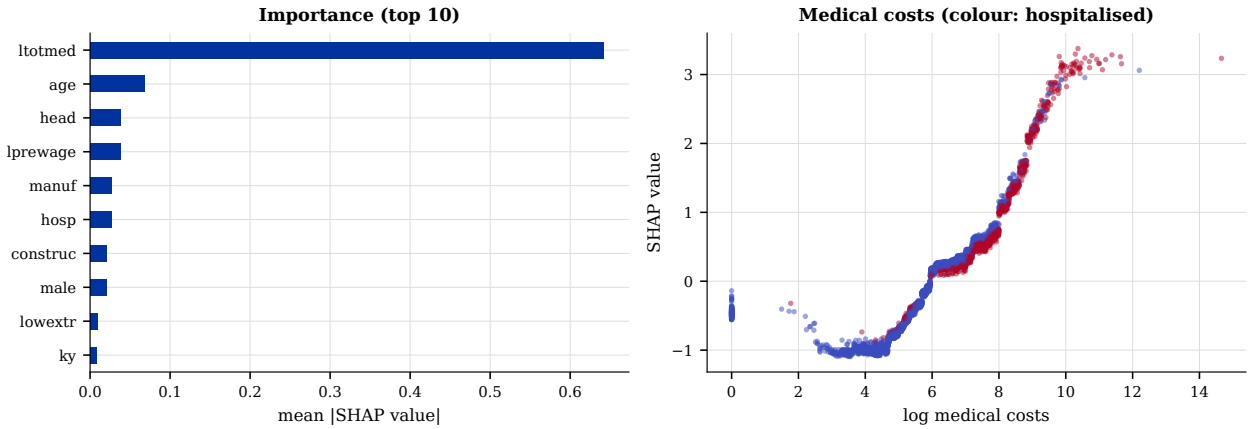


Figure 36.2. Open the black box: SHAP values. Panels: Importance (top 10); Medical costs (colour: hospitalised).

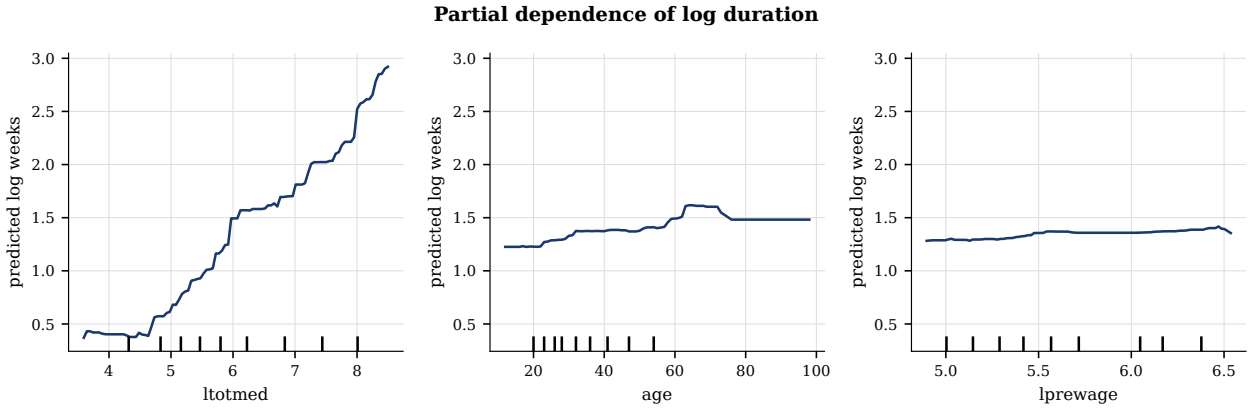


Figure 36.3. Partial dependence: Partial dependence of log duration.

```
top3 = [c for c in imp.index if X[c].nunique() > 2][:3]
fig, axes = plt.subplots(1, 3, figsize=(9, 3))
PartialDependenceDisplay.from_estimator(fin, X, top3, ax=axes)
for a in axes: a.set_ylabel("predicted log weeks")
fig.suptitle("Partial dependence of log duration", fontsize=10, fontweight="bold"); fig.tight_layout()
```

Medical costs do almost all the work: predicted durations are flat for the cheapest injuries (log costs below about 4.5, roughly $90) and then rise steeply. Age and the previous wage shift predictions far less. These are predictive patterns, not causal effects. The benefit increase (`afchnge`), for example, is the treatment in the original study; its causal effect needs a difference-in-differences design (see Part IV), not a prediction model.

36.8 Write it up

```
print(out)
print("\nfinal model:", {**best, "learning_rate": .03, "n_estimators": n_best, "subsample": .8, "
  ↪ colsample_bytree": .8})
print("importance (mean |SHAP|):", ", ".join(f"{k} {v:.3f}" for k, v in imp[:6].items()))
```

```
                   RMSE  s.d. across folds  R2 (out of sample)
mean only        1.3016             0.0373              0.0009
OLS              1.0830             0.0322              0.3083
LASSO            1.0828             0.0325              0.3086
random forest    0.9833             0.0200              0.4298
XGBoost (tuned)  0.9780             0.0197              0.4359

final model: {'max_depth': 3, 'min_child_weight': 5, 'learning_rate': 0.03, 'n_estimators': 230, 'subsample': 0.8, '
↪ colsample_bytree': 0.8}
importance (mean |SHAP|): ltotmed 0.641, age 0.068, head 0.038, lprewage 0.038, manuf 0.027, hosp 0.027
```

How to write it: results

We predict log benefit durations with gradient-boosted trees (XGBoost) with a learning rate of 0.03, choosing the number of trees by early stopping and the tree depth and minimum leaf weight by five-fold cross-validation. In nested cross-validation the boosted model has the lowest out-of-sample error of the models compared: it explains 44 % of the out-of-sample variance, against 43 % for a random forest and 31 % for OLS and LASSO (Table X). SHAP values show that medical costs dominate the predictions (a mean absolute contribution of 0.64 log weeks, ten times that of the next predictor, age), followed by head injuries, the previous wage, industry and hospitalisation. The partial-dependence plot for medical costs is flat for the cheapest injuries and then rises steeply and almost linearly in log costs, a threshold that a linear model misses; age and the previous wage matter much less. We interpret these patterns as predictive, not causal.

Checklist for the paper

- Build validation folds that respect the data's structure (groups, time).
- Report the learning rate, early stopping and the tuned depth and regularisation.
- Compare with a random forest and with linear models in nested cross-validation.
- Interpret with SHAP values and partial dependence, and say that the patterns are predictive.

CHAPTER 37

Double Machine Learning: Partially Linear Model

Applies when. Continuous treatment, many or nonlinear controls

Overview. Use this chapter when you want the effect of one continuous variable on an outcome, the effect is identified by adjusting for observed controls, and there are many controls or they may enter nonlinearly. Double machine learning (DML) learns the role of the controls with machine learning and still gives a valid estimate and standard error for the one coefficient you care about.

Data. 90 countries, growth 1965–1975, 60 controls (Barro–Lee; Belloni, Chernozhukov and Hansen)

Software. Python · DoubleML · scikit-learn · statsmodels

37.1 The question and the model

Do poorer countries grow faster? The neoclassical growth model predicts *conditional* convergence: holding constant the determinants of a country's steady state, a lower initial income should mean faster growth. The test is the sign of θ in

$$\text{growth}_i = \theta \log \text{GDP}_{i,1965} + g(X_i) + \varepsilon_i,$$

where X_i are the steady-state determinants. The Barro–Lee data used by Belloni, Chernozhukov and Hansen (2011, 2014) have 90 countries and **60** candidate controls: schooling by level and sex, population growth, fertility, investment, government spending, openness, political instability, the black-market premium and more. With 60 controls and 90 countries, OLS is barely estimable, and choosing a handful of controls by hand is exactly the specification search that convergence studies have been criticised for.

This is the **partially linear model**: the treatment D enters linearly with one coefficient θ, and the controls enter through an unknown function g that machine learning can estimate. DML (Chernozhukov et al. 2018) estimates θ in three steps:

1. predict the outcome from the controls, $\hat{\ell}(X) \approx E[Y \mid X]$, and the treatment from the controls, $\hat{m}(X) \approx E[D \mid X]$, each by machine learning;
2. form the residuals $\tilde{Y} = Y - \hat{\ell}(X)$ and $\tilde{D} = D - \hat{m}(X)$;
3. regress $\tilde{Y}$ on $\tilde{D}$: the slope is $\hat{\theta}$.

Two ingredients make the standard error valid even though the predictions are regularised: the residual-on-residual form is **Neyman-orthogonal** (small errors in $\hat{\ell}$ and $\hat{m}$ have only second-order effects on $\hat{\theta}$), and every prediction is made by **cross-fitting**, on observations the learner was not trained on.

> **The identifying assumption is the same as for OLS**
>
> DML assumes that, given X, the remaining variation in initial GDP is unrelated to unobserved determinants of growth. Machine learning makes the adjustment for X flexible; it does not make the assumption true. Argue it as you would for OLS, and if it fails, use an instrument (DML-IV) or a research design.

37.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

```
df = pd.read_csv("growth.csv")
y, d = "growth", "gdpsh465"
X = [c for c in df.columns if c not in (y, d)]
print(f"N = {len(df)} countries, {len(X)} controls; treatment {d} = log GDP per capita, 1965")

small = sm.OLS(df[y], sm.add_constant(df[[d]])).fit(cov_type="HC1")
full = sm.OLS(df[y], sm.add_constant(df[[d] + X])).fit(cov_type="HC1")
print(f"OLS without controls:   theta = {small.params[d]:+.4f} (s.e. {small.bse[d]:.4f})")
print(f"OLS with all 60 controls: theta = {full.params[d]:+.4f} (s.e. {full.bse[d]:.4f}); residual df = {int(
  ↪ full.df_resid)}")
```

```
N = 90 countries, 60 controls; treatment gdpsh465 = log GDP per capita, 1965
OLS without controls:   theta = +0.0013 (s.e. 0.0053)
OLS with all 60 controls: theta = -0.0094 (s.e. 0.0324); residual df = 28
```

Without controls, richer countries did *not* grow more slowly: unconditional convergence fails. With all 60 controls, OLS uses up most of the sample (29 residual degrees of freedom) and the estimate is too imprecise to say anything. That is the case for DML (Rule B2: 60 controls, far more than 10).

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV
from sklearn.ensemble import RandomForestRegressor

cv = KFold(5, shuffle=True, random_state=0)
rmse = lambda m, t: (-cross_val_score(m, df[X], df[t], cv=cv, scoring="neg_mean_squared_error").mean()) ** .5
lasso = make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0, max_iter=50000))
forest = RandomForestRegressor(n_estimators=300, min_samples_leaf=5, max_features=.5, random_state=0)
for t in (y, d):
    a, b = rmse(lasso, t), rmse(forest, t)
    print(f"predicting {t:9s} from the controls: LASSO {a:.4f}, forest {b:.4f}, forest gain {1 - b / a:+.1%}")
```

```
predicting growth    from the controls: LASSO 0.0513, forest 0.0483, forest gain +5.9%
predicting gdpsh465  from the controls: LASSO 0.3660, forest 0.3778, forest gain -3.2%
```

The forest predicts growth slightly better than LASSO from the controls, and initial GDP slightly worse. Neither clearly dominates, so the chapter estimates DML with both learners and checks that they agree (Rule B4).

37.3 Why not just use LASSO to pick the controls?

The tempting shortcut is to run LASSO of growth on initial GDP and the 60 controls, keep what survives, and report OLS on the survivors. This **single selection** is biased: LASSO drops controls that predict growth only weakly, even if they are strongly related to initial GDP, and those are exactly the confounders that matter. Belloni, Chernozhukov and Hansen (2014) show that one must also keep controls that predict the *treatment*. Their **double selection** is the simplest valid version of the DML idea:

LASSO's penalty matters here. Cross-validation chooses a penalty for **prediction** and, with 90 observations, keeps most of the 60 controls. For selecting controls, Belloni, Chernozhukov and Hansen use a theory-based **plug-in** penalty, $\lambda = 2c\,\hat{\sigma}\sqrt{n}\,\Phi^{-1}(1-\gamma/(2p))$ with $c = 1.1$ and $\gamma = 0.1/\log n$, iterated on the residual variance. It keeps only controls whose signal is clearly above noise.

```
from sklearn.linear_model import Lasso
from scipy.stats import norm

Z = StandardScaler().fit_transform(df[X])
n, p = Z.shape
def lasso_plugin(target, c=1.1):
    t = df[target].to_numpy()
    sigma = t.std()
    for _ in range(5):                                  # iterate the noise level on the LASSO residuals
        lam = 2 * c * sigma * np.sqrt(n) * norm.ppf(1 - .1 / np.log(n) / (2 * p))
        fit = Lasso(alpha=lam / (2 * n), max_iter=50000).fit(Z, t)
        sigma = (t - fit.predict(Z)).std()
    return {X[j] for j, b in enumerate(fit.coef_) if b != 0}
```

```
keep_y, keep_d = lasso_plugin(y), lasso_plugin(d)
single = sm.OLS(df[y], sm.add_constant(df[[d] + sorted(keep_y)])).fit(cov_type="HC1")
double = sm.OLS(df[y], sm.add_constant(df[[d] + sorted(keep_y | keep_d)])).fit(cov_type="HC1")
print(f"controls selected for growth: {len(keep_y)}; for initial GDP: {len(keep_d)} {sorted(keep_d)}")
print(f"single selection: theta = {single.params[d]:+.4f} (s.e. {single.bse[d]:.4f})")
print(f"double selection: theta = {double.params[d]:+.4f} (s.e. {double.bse[d]:.4f})")
```

```
controls selected for growth: 0; for initial GDP: 7 ['freetar', 'hm65', 'humanf65', 'lifee065', 'pop6565', 'sf65', '
↪ teapri65']
single selection: theta = +0.0013 (s.e. 0.0053)
double selection: theta = -0.0419 (s.e. 0.0167)
```

The result is a textbook illustration of the problem. LASSO of growth on the controls keeps **none** of them: no single control predicts growth strongly enough. Single selection therefore reduces to OLS without controls and finds no convergence. But several controls strongly predict initial GDP, and those are the confounders. Double selection keeps them and finds a clearly negative θ, close to the -0.050 (s.e. 0.016) that Belloni, Chernozhukov and Hansen (2014) report with the same data.

37.4 Double machine learning with DoubleML

The `DoubleML` package (Bach et al. 2022) implements the estimator with cross-fitting and repeated sample splits. The learners are passed in as scikit-learn models: `ml_l` predicts the outcome and `ml_m` the treatment.

```
import doubleml as dml

data = dml.DoubleMLData(df, y_col=y, d_cols=d, x_cols=X)
learners = {"LASSO": make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0, max_iter=50000)),
            "random forest": RandomForestRegressor(n_estimators=500, min_samples_leaf=5, max_features=.5,
  ↪ random_state=0)}
fits = {}
for name, ml in learners.items():
    np.random.seed(0)
    fits[name] = dml.DoubleMLPLR(data, ml_l=ml, ml_m=ml, n_folds=5, n_rep=10).fit()
    s = fits[name].summary
    print(f"DML-PLR, {name:13s}: theta = {fits[name].coef[0]:+.4f}  s.e. = {fits[name].se[0]:.4f}  "
          f"95% CI [{s['2.5 %'].iloc[0]:+.4f}, {s['97.5 %'].iloc[0]:+.4f}]  p = {s['P>|t|'].iloc[0]:.3f}")
```

```
DML-PLR, LASSO        : theta = -0.0408  s.e. = 0.0150  95% CI [-0.0722, -0.0113]  p = 0.007
DML-PLR, random forest: theta = -0.0391  s.e. = 0.0142  95% CI [-0.0679, -0.0112]  p = 0.006
```

With 90 observations, one random split into folds can matter, so the estimator is repeated over 10 different splits (`n_rep=10`) and the median is reported, with a standard error that includes the variation across splits.

37.5 Compare the estimates

```
rows = {"OLS, no controls": (small.params[d], small.bse[d]),
        "OLS, all 60 controls": (full.params[d], full.bse[d]),
        "LASSO single selection (biased)": (single.params[d], single.bse[d]),
        "LASSO double selection": (double.params[d], double.bse[d])}
rows.update({f"DML-PLR, {k}": (f.coef[0], f.se[0]) for k, f in fits.items()})
tab = pd.DataFrame(rows, index=["theta", "s.e."]).T
tab["t"] = tab["theta"] / tab["s.e."]
print(tab.round(4))
gap = abs(fits["LASSO"].coef[0] - fits["random forest"].coef[0]) / max(f.se[0] for f in fits.values())
print(f"\nLASSO and forest learners differ by {gap:.2f} standard errors")
```

```
                                theta    s.e.       t
OLS, no controls               0.0013  0.0053  0.2475
OLS, all 60 controls           -0.0094  0.0324 -0.2893
LASSO single selection (biased)  0.0013  0.0053  0.2475
LASSO double selection         -0.0419  0.0167 -2.5039
DML-PLR, LASSO                 -0.0408  0.0150 -2.7141
DML-PLR, random forest         -0.0391  0.0142 -2.7437

LASSO and forest learners differ by 0.12 standard errors
```

```
import matplotlib.pyplot as plt
```

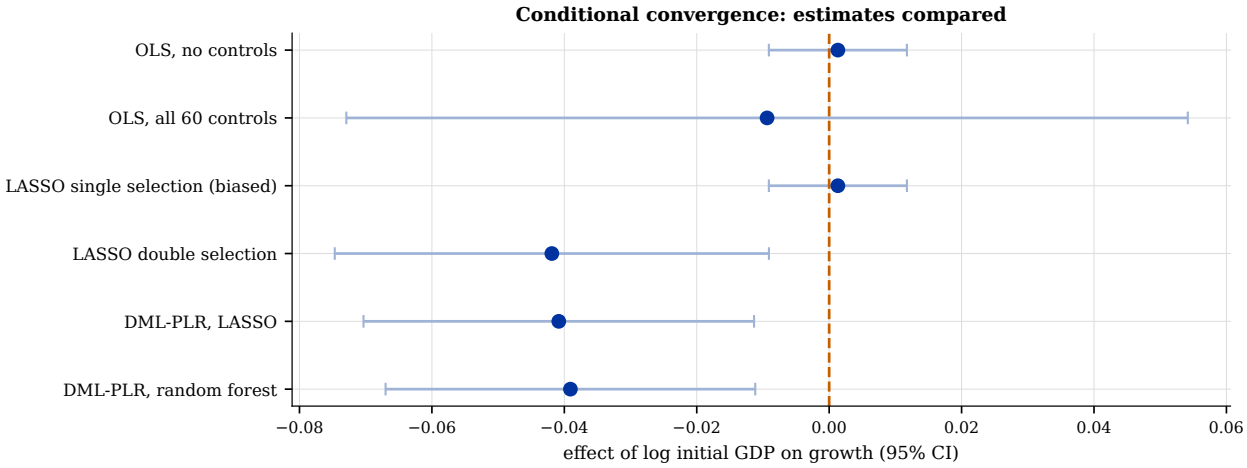


Figure 37.1. Compare the estimates: Conditional convergence: estimates compared.

```
fig, ax = plt.subplots(figsize=(8, 3.2))
ypos = np.arange(len(tab))[::-1]
ax.errorbar(tab["theta"], ypos, xerr=1.96 * tab["s.e."], fmt="o", color="#0033a0", ecolor="#9fb3d6", capsize=3)
ax.axvline(0, color="#c46000", ls="--"); ax.set_yticks(ypos); ax.set_yticklabels(tab.index, fontsize=8)
ax.set_xlabel("effect of log initial GDP on growth (95% CI)"); ax.set_title("Conditional convergence: estimates compared")
```

37.6 How much does the sample split matter?

```
thetas = pd.DataFrame({k: f.all_coef[0] for k, f in fits.items()})
print(thetas.describe().loc[["mean", "std", "min", "50%", "max"]].round(4))
fig, ax = plt.subplots()
for k in thetas: ax.hist(thetas[k], bins=12, alpha=.6, label=k)
ax.axvline(0, color="#999"); ax.set_xlabel("θ in each of the 10 sample splits"); ax.set_title("Variation across sample splits"); ax.legend()
```

```
      LASSO  random forest
mean -0.0411        -0.0394
std   0.0063         0.0024
min  -0.0529        -0.0455
50%  -0.0408        -0.0391
max  -0.0308        -0.0370
```

The spread across splits is the price of a small sample. Report the median over many splits, never the result of one split chosen after the fact.

37.7 Write it up

How to write it: method and results

We estimate the effect of log initial GDP per capita on growth in the partially linear model $\text{growth}_i = \theta \log GDP_{i,1965} + g(X_i) + \varepsilon_i$, *where* X_i *contains 60 country characteristics, using double machine learning (Chernozhukov et al. 2018). The nuisance functions are estimated with LASSO and, alternatively, with a random forest, using five-fold cross-fitting repeated over 10 random splits. Unconditionally there is no convergence* ($\hat{\theta} = 0.001$), *and OLS with all 60 controls is too imprecise to be informative (s.e. 0.032). Double machine learning gives* $\hat{\theta} \approx -0.04$ *with both learners (s.e. about 0.015,* $p < 0.01$), *and the two learners agree within a fraction of a standard error. The estimate is close to post-double-selection LASSO with a plug-in penalty (*-0.042*), and to the* -0.050 *reported by Belloni, Chernozhukov and Hansen (2014): conditional on their steady-state determinants, poorer countries grew faster (Table X).*

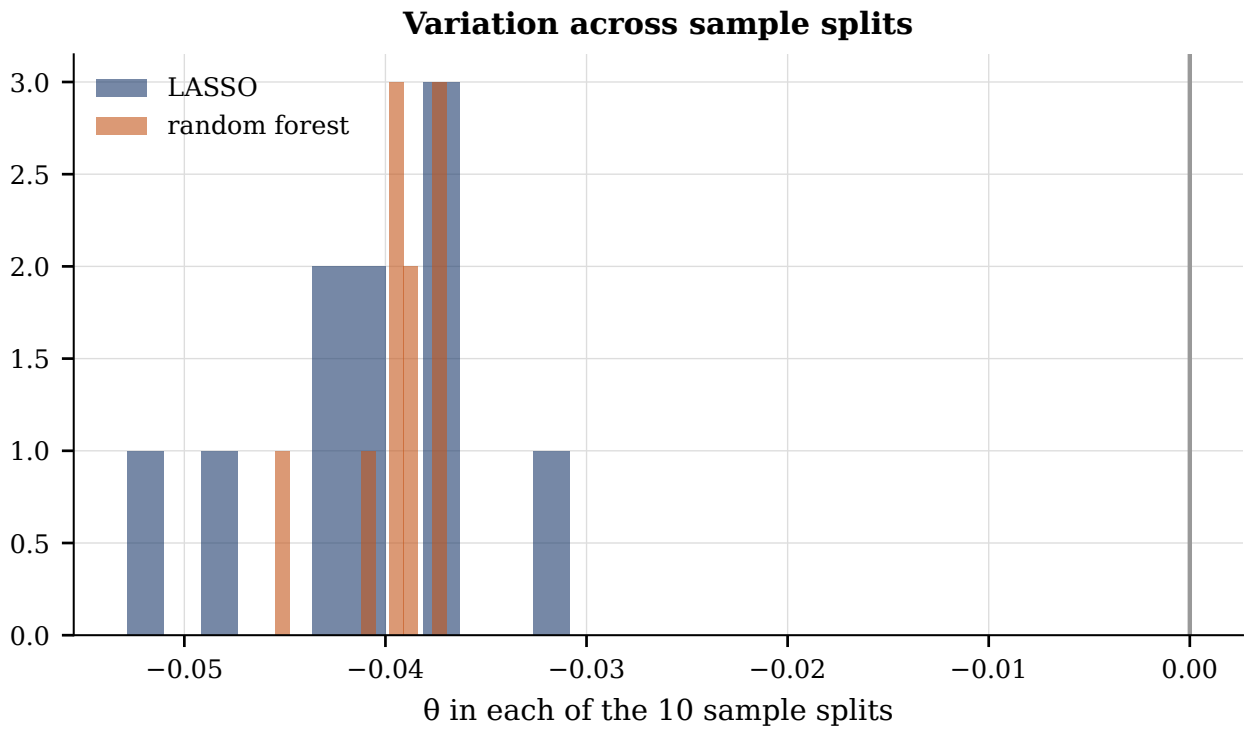


Figure 37.2. How much does the sample split matter?: Variation across sample splits.

Checklist for the paper

- State the partially linear model and argue that the controls capture the confounders.
- Explain why OLS is not enough (many controls, or nonlinear controls).
- Report the learners, the number of folds and of repeated splits.
- Show DML with at least two learners, plus OLS and double selection as benchmarks.
- Report the variation across sample splits.

CHAPTER 38

Double Machine Learning: Interactive Model

Applies when. Binary treatment, many or nonlinear controls

Overview. Use this chapter when the treatment is binary (a programme, a policy, eligibility), the effect is identified by adjusting for observed controls, and the controls are many or enter nonlinearly. The interactive model estimates the average treatment effect with a doubly robust score, which stays consistent if either the outcome model or the propensity score is right.

Data. 9,915 U.S. households, 1991 SIPP (Poterba, Venti and Wise; Chernozhukov et al. 2018)

Software. Python · DoubleML · scikit-learn · statsmodels

38.1 The question and the model

Does eligibility for a 401(k) pension plan raise households' net financial assets? In the early 1990s, eligibility was determined by the employer, not chosen by the worker. Poterba, Venti and Wise (1994, 1995) argued that, once income and other characteristics are held fixed, eligibility is as good as randomly assigned: people did not choose their job because of its pension plan. This is the classic application of DML (Chernozhukov et al. 2018).

The treatment D (eligible or not) is binary, and its effect may differ across households. The **interactive regression model** (IRM) allows that:

$$Y = g_0(D, X) + U, \qquad D = m_0(X) + V, \qquad m_0(X) = P(D = 1 \mid X),$$

and targets the **average treatment effect** $\theta = E[g_0(1, X) - g_0(0, X)]$. DML estimates it with the doubly robust (AIPW) score, averaged over cross-fitted predictions:

$$\hat{\theta} = \frac{1}{n} \sum_i \Big[\hat{g}(1, X_i) - \hat{g}(0, X_i) + \frac{D_i \, (Y_i - \hat{g}(1, X_i))}{\hat{m}(X_i)} - \frac{(1 - D_i)(Y_i - \hat{g}(0, X_i))}{1 - \hat{m}(X_i)} \Big].$$

The first two terms compare predicted outcomes with and without treatment; the last two correct the prediction errors, weighted by the inverse propensity score.

Variable	Definition
`net_tfa`	Net total financial assets, dollars (outcome)
`e401`	1 if eligible for a 401(k) (treatment)
`age`, `age_sq`	Age and its square
`inc`, `inc_sq`	Income and its square
`educ`, `educ_sq`	Years of schooling and its square
`fsize`, `marr`, `twoearn`	Family size; married; two earners
`db`, `pira`, `hown`	Defined-benefit pension; IRA; home owner

38.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

```
df = pd.read_csv("k401_irm.csv")
y, d = "net_tfa", "e401"
X = [c for c in df.columns if c not in (y, d)]
print(f"N = {len(df)} households, {len(X)} controls; {df[d].mean():.1%} eligible")
print(df.groupby(d)[[y, "inc", "age", "educ"]].mean().round(1))

raw = df.loc[df[d] == 1, y].mean() - df.loc[df[d] == 0, y].mean()
ols = sm.OLS(df[y], sm.add_constant(df[[d] + X])).fit(cov_type="HC1")
print(f"\nraw difference in means: {raw:,.0f}")
print(f"OLS with controls:       {ols.params[d]:,.0f} (s.e. {ols.bse[d]:,.0f})")
```

```
N = 9915 households, 12 controls; 37.1% eligible
      net_tfa      inc   age  educ
e401
0.0   10788.0  31493.6  40.8  12.9
1.0   30347.4  46861.7  41.5  13.8

raw difference in means: 19,559
OLS with controls:       9,396 (s.e. 1,314)
```

Eligible households hold far more financial assets, but they also earn more: the raw difference mixes the effect of eligibility with the effect of income. Adjusting for the controls cuts the difference roughly in half. With 12 controls (Rule B2: at least 10), DML is the method; with a binary treatment, the interactive model (Rule B3).

38.3 Propensity scores and overlap

The AIPW score divides by $\hat{m}(X)$ and $1 - \hat{m}(X)$. If some households are almost certain to be eligible, or almost certain not to be, those weights explode and the estimate becomes unstable. Before estimating anything, check **overlap**: the propensity scores of treated and untreated households should cover the same range, away from 0 and 1.

```
from sklearn.model_selection import cross_val_predict
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

ps = cross_val_predict(RandomForestClassifier(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                       df[X], df[d], cv=5, method="predict_proba")[:, 1]
out = (ps < .05) | (ps > .95)
print(f"propensity scores from {ps.min():.3f} to {ps.max():.3f}; {out.mean():.1%} outside [0.05, 0.95]")
print(pd.crosstab(pd.cut(ps, [0, .05, .1, .3, .5, .7, .9, .95, 1]), df[d].map({0: 'not eligible', 1: 'eligible
  ↪ '})))
fig, ax = plt.subplots()
ax.hist(ps[df[d] == 0], bins=40, alpha=.6, label="not eligible", color="#c46000")
ax.hist(ps[df[d] == 1], bins=40, alpha=.6, label="eligible", color="#0033a0")
for v in (.05, .95): ax.axvline(v, ls="--", color="#8a2a17")
ax.set_xlabel("estimated propensity score"); ax.set_title("Overlap"); ax.legend()
```

```
propensity scores from 0.000 to 0.913; 4.8% outside [0.05, 0.95]
e401         eligible  not eligible
row_0
(0.0, 0.05]        27           451
(0.05, 0.1]        62           570
(0.1, 0.3]        602          2167
(0.3, 0.5]       1246          1848
(0.5, 0.7]       1305          1009
(0.7, 0.9]        438           187
(0.9, 0.95]         2             1
```

Overlap is reasonable in the middle of the distribution, but 4.8 % of households, almost all of them not eligible (451 of the 478 below 0.05), have propensity scores outside [0.05, 0.95]: households whose characteristics almost never come with a 401(k) offer. DoubleML **trims** extreme scores at a threshold; the chapter reports the estimate at several thresholds in Section 38.5.

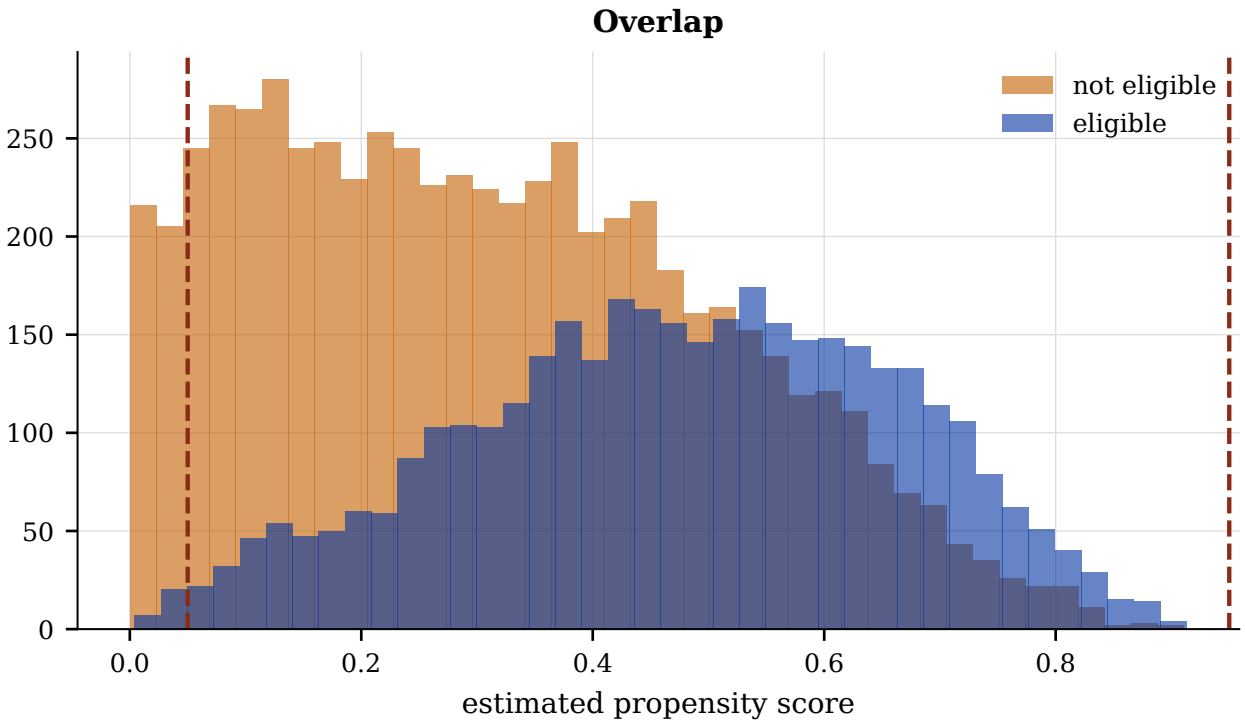


Figure 38.1. Propensity scores and overlap: Overlap.

38.4 Double machine learning with DoubleML

The interactive model needs three predictions: the outcome for treated households $\hat{g}(1, X)$, for untreated households $\hat{g}(0, X)$ (both from `ml_g`), and the propensity score $\hat{m}(X)$ (from the classifier `ml_m`). Each is cross-fitted.

```
import doubleml as dml
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV, LogisticRegressionCV
from sklearn.ensemble import RandomForestRegressor

data = dml.DoubleMLData(df, y_col=y, d_cols=d, x_cols=X)
learners = {"LASSO": (make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0, max_iter=20000)),
                      make_pipeline(StandardScaler(), LogisticRegressionCV(cv=5, penalty="l1", solver="
  ↪ liblinear", Cs=10))),
            "random forest": (RandomForestRegressor(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                              RandomForestClassifier(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1))}
fits = {}
for name, (g, m) in learners.items():
    np.random.seed(0)
    fits[name] = dml.DoubleMLIRM(data, ml_g=g, ml_m=m, n_folds=5, n_rep=3, trimming_threshold=.01).fit()
    s = fits[name].summary
    print(f"ATE, {name:13s}: {fits[name].coef[0]:9,.0f}  s.e. {fits[name].se[0]:7,.0f}  95% CI [{s['2.5 %'].
  ↪ iloc[0]:,.0f}, {s['97.5 %'].iloc[0]:,.0f}]")
```

```
ATE, LASSO        :     8,316  s.e.   1,161  95% CI [5,989, 10,592]
ATE, random forest:     7,896  s.e.   1,222  95% CI [5,502, 10,291]
```

38.5 Compare the estimates

```
rows = {"raw difference": (raw, np.nan), "OLS with controls": (ols.params[d], ols.bse[d])}
rows.update({f"DML interactive, {k}": (f.coef[0], f.se[0]) for k, f in fits.items()})
```

```
tab = pd.DataFrame(rows, index=["effect", "s.e."]).T.round(0)
print(tab)
gap = abs(fits["LASSO"].coef[0] - fits["random forest"].coef[0]) / max(f.se[0] for f in fits.values())
print(f"\nLASSO and forest learners differ by {gap:.2f} standard errors")
```

```
                              effect    s.e.
raw difference                19559.0     NaN
OLS with controls              9396.0  1314.0
DML interactive, LASSO         8316.0  1161.0
DML interactive, random forest 7896.0  1222.0

LASSO and forest learners differ by 0.34 standard errors
```

Both learners give an average effect of eligibility of about 8,000 dollars (8,316 with LASSO, 7,896 with random forests), close to the published DML estimates for these data, below the OLS estimate of 9,396 and less than half the raw difference. The two learners differ by a third of a standard error (Rule B4), so the result does not hinge on the choice of machine-learning method.

38.6 Robustness: trimming and the effect on the treated

Two checks belong in the paper. First, how much do the extreme propensity scores matter? Second, the **average effect on the treated** (ATTE), the effect for households that actually were eligible, may differ from the ATE.

```
g, m = learners["random forest"]
rob = {}
for thr in (.01, .05, .1):
    np.random.seed(0)
    f = dml.DoubleMLIRM(data, ml_g=g, ml_m=m, n_folds=5, n_rep=1, trimming_threshold=thr).fit()
    rob[f"ATE, trimming at {thr}"] = (f.coef[0], f.se[0])
np.random.seed(0)
f = dml.DoubleMLIRM(data, ml_g=g, ml_m=m, n_folds=5, n_rep=1, score="ATTE").fit()
rob["ATTE (eligible households)"] = (f.coef[0], f.se[0])
print(pd.DataFrame(rob, index=["effect", "s.e."]).T.round(0))
```

```
                           effect    s.e.
ATE, trimming at 0.01      7632.0  1227.0
ATE, trimming at 0.05      7709.0  1218.0
ATE, trimming at 0.1       7727.0  1211.0
ATTE (eligible households) 10860.0 1709.0
```

38.7 Write it up

How to write it: method and results

We estimate the average effect of 401(k) eligibility on net financial assets with the interactive regression model of Chernozhukov et al. (2018), assuming, following Poterba, Venti and Wise, that eligibility is as good as random conditional on income, age, education, family structure and other pension and asset holdings (twelve controls including squares). The outcome regressions and the propensity score are estimated with LASSO and, alternatively, random forests, with five-fold cross-fitting repeated three times and propensity scores trimmed at 0.01. Eligibility raises net financial assets by about 8,000 dollars on average (8,316 with LASSO and 7,896 with random forests; standard errors about 1,200), less than the OLS estimate of 9,396 and far less than the raw difference of 19,559 (Table X). The estimate barely moves when propensity scores are trimmed at 0.05 or 0.10 (7,709 and 7,727). The effect on eligible households is larger, about 10,900 dollars (ATTE), consistent with larger gains for the higher-income households who are more often eligible.

Checklist for the paper

- Argue why treatment is as good as random given the controls.
- Show the propensity-score distribution and the share outside [0.05, 0.95].
- Report the learners, folds, repetitions and the trimming threshold.
- Show DML with two learners next to OLS and the raw difference.

- Report the ATTE and the estimate at other trimming thresholds.

CHAPTER 39

Double Machine Learning with an Instrument (DML-IV)

Applies when. Endogenous treatment, valid instrument

Overview. Use this chapter when the treatment is chosen by the people you study, so it is endogenous even given the controls, but you have an instrument that shifts the treatment and, given the controls, affects the outcome only through it. DML-IV partials the controls out of the outcome, the treatment and the instrument with machine learning, then applies IV to what is left.

Data. 9,915 U.S. households, 1991 SIPP (Chernozhukov et al. 2018)

Software. Python · DoubleML · linearmodels · scikit-learn

39.1 The question and the model

What is the effect of actually participating in a 401(k) plan on a household's financial assets? Participation is a choice: households that save a lot anyway are more likely to join, so comparing participants with non-participants, even with controls, overstates the effect. **Eligibility** is a natural instrument. It is set by the employer, so (following Poterba, Venti and Wise) it is as good as random given income and the other controls; it strongly shifts participation, because only eligible workers can participate; and it plausibly affects assets only through participation.

The **partially linear IV model** is

$$Y = \theta D + g_0(X) + \varepsilon, \qquad Z = m_0(X) + V, \qquad E[\varepsilon \mid Z, X] = 0,$$

with D participation, Z eligibility and X the controls. DML-IV predicts Y, D and Z from X by cross-fitted machine learning, and then runs IV on the residuals: $\hat{\theta} = \sum \tilde{Z}_i \tilde{Y}_i / \sum \tilde{Z}_i \tilde{D}_i$.

> **Whose effect is this?**
>
> With a binary instrument and heterogeneous effects, IV estimates the effect for **compliers**, the households who participate because they are eligible (a local average treatment effect; Imbens and Angrist 1994). Here non-eligible households cannot participate at all (one-sided non-compliance), so the compliers are all participants and θ is the average effect of participation on the participants.

39.2 Load the data and confirm the starting point

```
import numpy as np
import pandas as pd
import statsmodels.api as sm

df = pd.read_csv("k401_iv.csv")
y, d, z = "net_tfa", "p401", "e401"
X = [c for c in df.columns if c not in (y, d, z)]
print(f"N = {len(df)} households, {len(X)} controls")
print(pd.crosstab(df[z].map({0: "not eligible", 1: "eligible"}), df[d].map({0: "not participating", 1: "
  ↪ participating"})))
print(f"participation rate among the eligible: {df.loc[df[z] == 1, d].mean():.1%}")

ols = sm.OLS(df[y], sm.add_constant(df[[d] + X])).fit(cov_type="HC1")
```

```
print(f"\nOLS of assets on participation, with controls: {ols.params[d]:,.0f} (s.e. {ols.bse[d]:,.0f})")
```

```
N = 9915 households, 12 controls
p401          not participating  participating
e401
eligible                   1088           2594
not eligible               6233              0
participation rate among the eligible: 70.5%

OLS of assets on participation, with controls: 14,159 (s.e. 1,496)
```

Nobody who is ineligible participates, and 70.5 % of the eligible do. OLS gives a large effect of participation, but it compares savers who chose to participate with households that did not.

39.3 The first stage

An instrument must move the treatment. The first-stage regression of participation on eligibility and the controls tells how strongly; the decision rules of Chapter 1 require an F statistic of at least 10 (Rule B3).

```
fs = sm.OLS(df[d], sm.add_constant(df[[z] + X])).fit(cov_type="HC1")
F = (fs.params[z] / fs.bse[z]) ** 2
print(f"first stage: coefficient on eligibility {fs.params[z]:.3f} (s.e. {fs.bse[z]:.4f}); F = {F:,.0f}")
```

```
first stage: coefficient on eligibility 0.695 (s.e. 0.0080); F = 7,619
```

The instrument is extremely strong: eligibility raises the probability of participating by 70 percentage points ($F \approx 7{,}600$). Weak-instrument problems do not arise here. When the F statistic is close to or below 10, report Anderson–Rubin confidence intervals, which stay valid however weak the instrument is.

39.4 A linear benchmark: 2SLS

```
from linearmodels.iv import IV2SLS

iv = IV2SLS(df[y], sm.add_constant(df[X]), df[d], df[z]).fit(cov_type="robust")
print(f"2SLS with linear controls: {iv.params[d]:,.0f} (s.e. {iv.std_errors[d]:,.0f})")
```

```
2SLS with linear controls: 13,526 (s.e. 1,885)
```

2SLS is the standard estimator, but it controls for X linearly. With income entering wealth and participation in a strongly nonlinear way, a linear adjustment may leave confounding behind. DML-IV relaxes that.

39.5 DML-IV with DoubleML

Three nuisance functions are cross-fitted: $\ell(X) = E[Y \mid X]$ (`ml_l`), $m(X) = E[Z \mid X]$ (`ml_m`) and $r(X) = E[D \mid X]$ (`ml_r`).

```
import doubleml as dml
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV
from sklearn.ensemble import RandomForestRegressor

data = dml.DoubleMLData(df, y_col=y, d_cols=d, x_cols=X, z_cols=z)
learners = {"LASSO": make_pipeline(StandardScaler(), LassoCV(cv=5, random_state=0, max_iter=20000)),
            "random forest": RandomForestRegressor(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1)}
fits = {}
for name, ml in learners.items():
    np.random.seed(0)
    fits[name] = dml.DoubleMLPLIV(data, ml_l=ml, ml_m=ml, ml_r=ml, n_folds=5, n_rep=3).fit()
    s = fits[name].summary
    print(f"DML-IV, {name:13s}: {fits[name].coef[0]:9,.0f}  s.e. {fits[name].se[0]:7,.0f}  95% CI [{s['2.5 %'].
  ↪ iloc[0]:,.0f}, {s['97.5 %'].iloc[0]:,.0f}]")
```

```
DML-IV, LASSO        :    13,641  s.e.   1,893  95% CI [9,930, 17,352]
DML-IV, random forest:    12,880  s.e.   1,910  95% CI [9,137, 16,622]
```

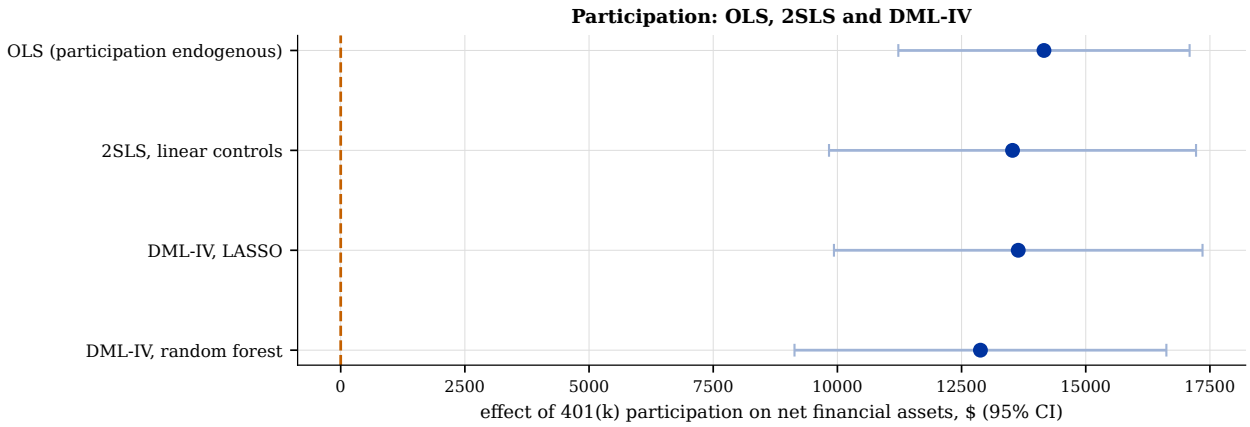


Figure 39.1. Compare the estimates: Participation: OLS, 2SLS and DML-IV.

39.6 Compare the estimates

```
rows = {"OLS (participation endogenous)": (ols.params[d], ols.bse[d]), "2SLS, linear controls": (iv.params[d],
  ↪  iv.std_errors[d])}
rows.update({f"DML-IV, {k}": (f.coef[0], f.se[0]) for k, f in fits.items()})
tab = pd.DataFrame(rows, index=["effect", "s.e."]).T.round(0)
print(tab)
gap = abs(fits["LASSO"].coef[0] - fits["random forest"].coef[0]) / max(f.se[0] for f in fits.values())
print(f"\nLASSO and forest learners differ by {gap:.2f} standard errors")

import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8, 2.8))
yy = np.arange(len(tab))[::-1]
ax.errorbar(tab["effect"], yy, xerr=1.96 * tab["s.e."], fmt="o", color="#0033a0", ecolor="#9fb3d6", capsize=3)
ax.set_yticks(yy); ax.set_yticklabels(tab.index, fontsize=8); ax.axvline(0, color="#c46000", ls="--")
ax.set_xlabel("effect of 401(k) participation on net financial assets, $ (95% CI)"); ax.set_title("
  ↪ Participation: OLS, 2SLS and DML-IV")
```

```
                              effect    s.e.
OLS (participation endogenous)  14159.0  1496.0
2SLS, linear controls           13526.0  1885.0
DML-IV, LASSO                   13641.0  1893.0
DML-IV, random forest           12880.0  1910.0

LASSO and forest learners differ by 0.40 standard errors
```

39.7 Reading the result: the intention-to-treat and the Wald ratio

With one binary instrument, the IV estimate is the effect of eligibility on assets (the **intention-to-treat**, ITT) divided by the effect of eligibility on participation (the first stage). Seeing the two pieces makes the result transparent:

```
itt = sm.OLS(df[y], sm.add_constant(df[[z] + X])).fit(cov_type="HC1").params[z]
print(f"ITT (eligibility on assets, linear controls): {itt:,.0f}")
print(f"first stage: {fs.params[z]:.3f}")
print(f"Wald ratio ITT / first stage: {itt / fs.params[z]:,.0f}  (= 2SLS with the same controls)")
```

```
ITT (eligibility on assets, linear controls): 9,396
first stage: 0.695
Wald ratio ITT / first stage: 13,526  (= 2SLS with the same controls)
```

39.8 Write it up

How to write it: method and results

Participation in a 401(k) is chosen by households and is therefore endogenous. We instrument it with eligibility, which is determined by employers and, following Poterba, Venti and Wise, is as good as random conditional on income, age, education and family characteristics. The first stage is very strong (eligibility raises participation by about 70 percentage points; F in the thousands). We estimate the partially linear IV model by double machine learning (Chernozhukov et al. 2018), with LASSO and random-forest learners, five-fold cross-fitting and three repetitions. Participation raises net financial assets by about 13,000 dollars among participants (13,641 with LASSO and 12,880 with random forests; standard errors about 1,900), close to linear 2SLS (13,526) and somewhat below OLS (14,159), which is biased upwards by the self-selection of savers into participation (Table X). The two learners differ by 0.4 standard errors.

Checklist for the paper

- Argue relevance, exclusion and as-good-as-random assignment of the instrument.
- Report the first stage and its F statistic; use Anderson–Rubin intervals if it is weak.
- State whose effect IV identifies (compliers; here, participants).
- Show DML-IV with two learners next to OLS and 2SLS.
- Report folds, repetitions and the learners for all three nuisance functions.

CHAPTER 40

Causal Forest

Applies when. Effects that differ across people, N $\geq$ 2,000
Overview. Use this chapter when the average effect is not enough and you need to know who gains most, the effect is identified by the controls (or by randomisation), and there are at least a few thousand observations. A causal forest estimates how the treatment effect varies with the controls, with honest splitting and valid confidence intervals.
Data. 9,915 U.S. households, 1991 SIPP (Chernozhukov et al. 2018)
Software. Python · EconML · DoubleML · scikit-learn

40.1 The question and the model

The DML Chapter 38 found that 401(k) eligibility raises net financial assets by about 8,000 dollars **on average**. For policy that is only half the answer. Does eligibility mainly help households that would have saved anyway, high earners with spare income, or does it also build assets for lower-income households? The question is about the **conditional average treatment effect** (CATE):

$$\tau(x) = E\big[\,Y(1) - Y(0) \mid X = x\,\big].$$

A **causal forest** (Wager and Athey 2018; Athey, Tibshirani and Wager 2019) estimates $\tau(x)$ with trees that split the sample where the *treatment effect* differs most, not where the outcome differs. Two features make its confidence intervals valid:

- **honesty**: each tree uses one half of its sample to choose the splits and the other half to estimate the effect in each leaf;
- **orthogonalisation**: as in DML, the outcome and the treatment are first residualised on the controls with machine learning, so that confounding by X does not leak into the heterogeneity estimates.

The implementation used here is `CausalForestDML` in Microsoft's EconML package.

> **Heterogeneity needs data and discipline**
>
> Effects by subgroup are noisier than the average, so a causal forest needs thousands of observations (the decision rules of Chapter 1 require 2,000). And a search for heterogeneity can always find something: decide in advance which dimensions matter for the question (here income, age, education), and report the variation honestly, including when it is small.

40.2 Load the data and recover the average effect

```
import numpy as np
import pandas as pd

df = pd.read_csv("k401_irm.csv")
y, d = "net_tfa", "e401"
X = [c for c in df.columns if c not in (y, d)]
base = ["age", "inc", "educ", "fsize", "marr", "twoearn", "db", "pira", "hown"]   # without the squares: trees
  ↪ need no polynomials
print(f"N = {len(df)} households; {df[d].mean():.1%} eligible; effect modifiers: {', '.join(base)}")
```

```
N = 9915 households; 37.1% eligible; effect modifiers: age, inc, educ, fsize, marr, twoearn, db, pira, hown
```

```
from econml.dml import CausalForestDML
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
```

```
cf = CausalForestDML(model_y=RandomForestRegressor(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                     model_t=RandomForestClassifier(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                     discrete_treatment=True, cv=5, n_estimators=2000, min_samples_leaf=20, honest=True,
  ↪ random_state=0)
cf.fit(df[y], df[d], X=df[base])
ate, ate_se = cf.ate_[0], cf.ate_stderr_[0]          # doubly robust average of the forest's effects
lo, hi = ate - 1.96 * ate_se, ate + 1.96 * ate_se
print(f"average effect from the forest: {ate:,.0f}  (s.e. {ate_se:,.0f}; 95% CI {lo:,.0f} to {hi:,.0f})")
```

```
average effect from the forest: 7,883  (s.e. 1,184; 95% CI 5,562 to 10,204)
```

The forest's average agrees with the DML estimate from Chapter 38. That is the first check: if the average of the conditional effects did not match the doubly robust ATE, something would be wrong with the forest.

40.3 How much do the effects vary?

```
import matplotlib.pyplot as plt

tau = cf.effect(df[base])
print(pd.Series(tau).describe(percentiles=[.1, .25, .5, .75, .9]).round(0).to_string())
fig, ax = plt.subplots()
ax.hist(tau, bins=60, color="#0033a0", alpha=.8)
ax.axvline(ate, color="#c46000", ls="--", label="average effect"); ax.axvline(0, color="#999")
ax.set_xlabel("estimated effect of eligibility on net financial assets, $"); ax.set_title("Distribution of
  ↪ conditional effects"); ax.legend()
```

```
count     9915.0
mean      8072.0
std       6501.0
min     -11160.0
10%       1550.0
25%       3539.0
50%       6251.0
75%      11425.0
90%      17529.0
max      46155.0
```

The estimated effects vary a great deal around the average. Part of that spread is estimation noise: individual CATEs are imprecise. The questions that can be answered reliably are about **groups** and about **which variables** drive the variation.

40.4 Which variables drive the heterogeneity?

The forest records how often, and how early, each variable is used to split on the treatment effect.

```
imp = pd.Series(cf.feature_importances_, index=base).sort_values(ascending=False)
print(imp.round(3).to_string())
fig, ax = plt.subplots(figsize=(8, 3))
imp[::-1].plot.barh(ax=ax, color="#0033a0"); ax.set_xlabel("share of splits (weighted)"); ax.set_title("What
  ↪ the effect depends on")
```

```
inc        0.694
age        0.150
educ       0.064
fsize      0.032
twoearn    0.031
pira       0.017
db         0.008
marr       0.005
hown       0.001
```

40.5 Effects by income group

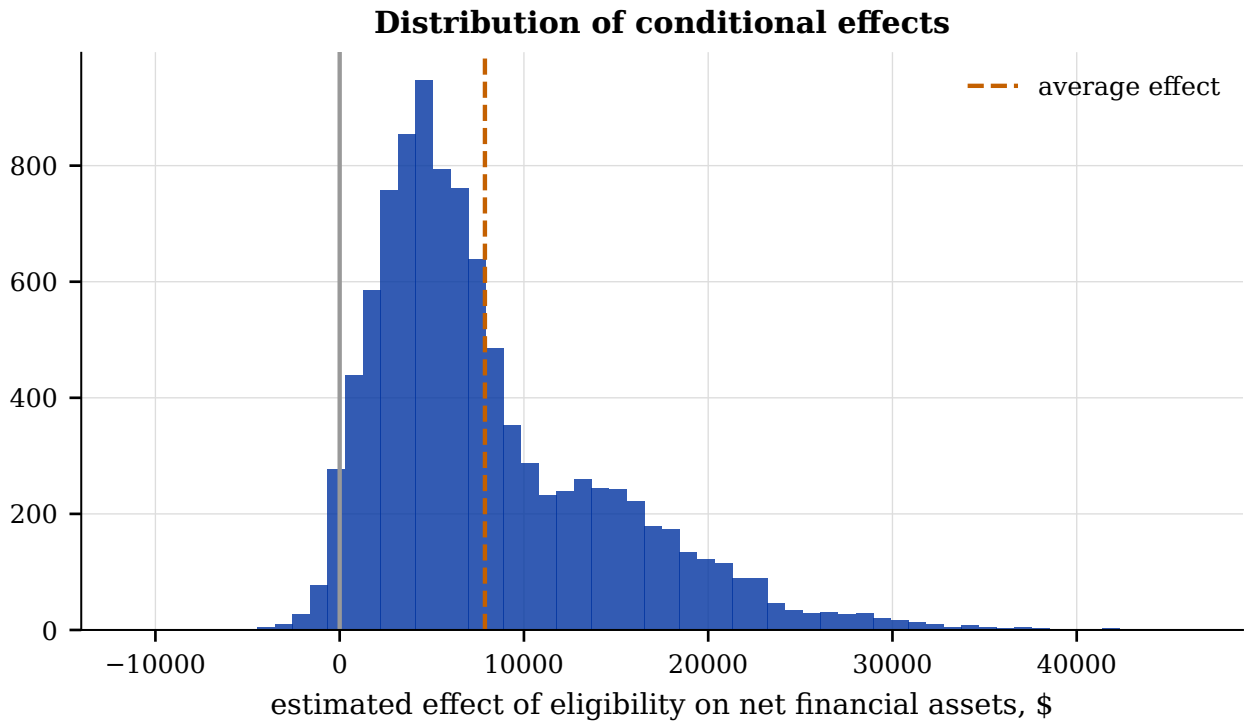


Figure 40.1. How much do the effects vary?: Distribution of conditional effects.

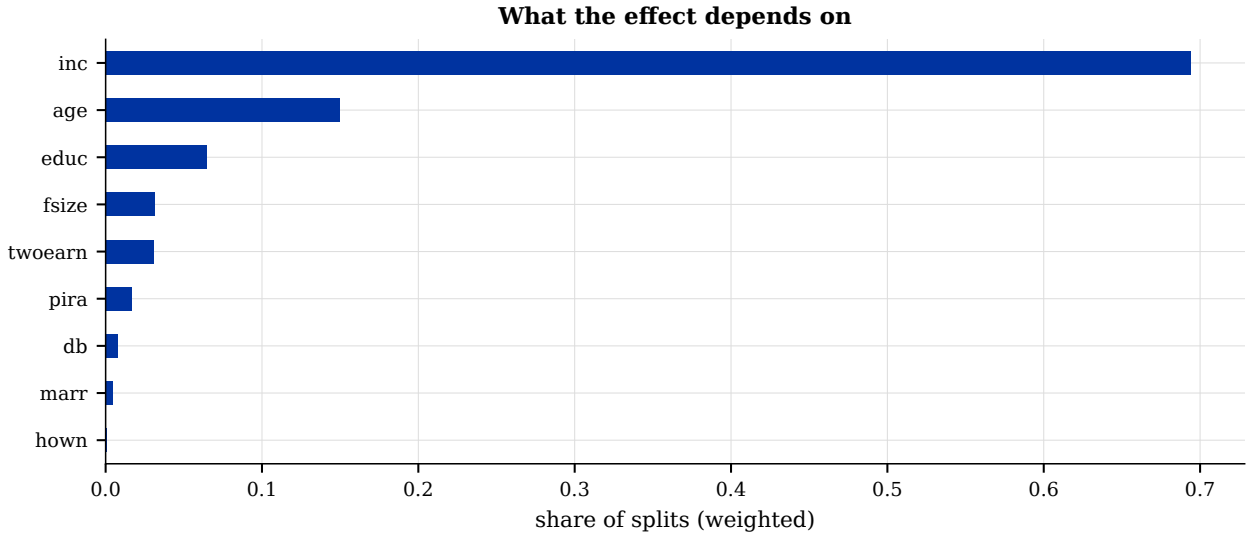


Figure 40.2. Which variables drive the heterogeneity?: What the effect depends on.

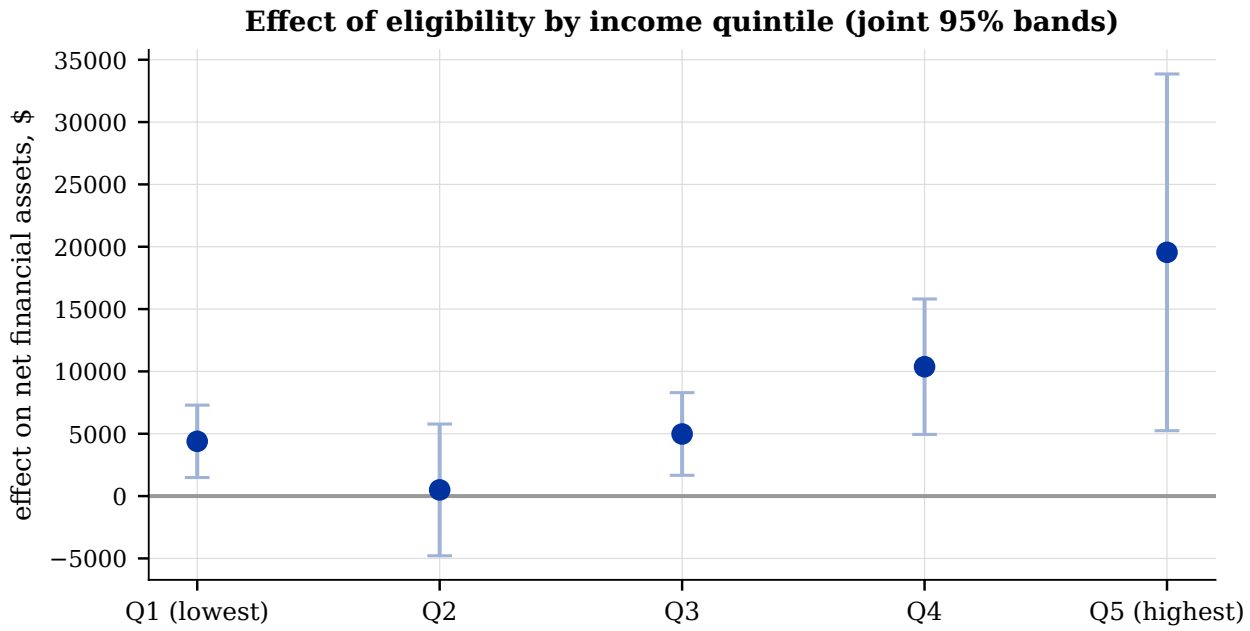


Figure 40.3. Effects by income group: Effect of eligibility by income quintile (joint 95% bands).

Income is the obvious policy dimension. The **group average treatment effect** (GATE) for each income quintile needs a valid confidence interval. Averaging the forest's individual intervals does not give one: it treats the spread of individual effects as if it were uncertainty about the average. The clean way is to average the doubly robust (AIPW) scores within each group, which is what DoubleML's `gate()` does (Semenova and Chernozhukov 2021), with joint confidence bands across the five groups.

```
import doubleml as dml

q = pd.qcut(df["inc"], 5, labels=["Q1 (lowest)", "Q2", "Q3", "Q4", "Q5 (highest)"])
groups = pd.get_dummies(q).astype(bool)
irm = dml.DoubleMLIRM(dml.DoubleMLData(df, y_col=y, d_cols=d, x_cols=X),
                      ml_g=RandomForestRegressor(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                      ml_m=RandomForestClassifier(n_estimators=300, min_samples_leaf=10, max_features=.5,
  ↪ random_state=0, n_jobs=-1),
                      n_folds=5, n_rep=1, trimming_threshold=.01)
np.random.seed(0); irm.fit()
g = irm.gate(groups=groups)
ci = g.confint(level=.95, joint=True, n_rep_boot=2000)
rng = df.groupby(q, observed=True)["inc"].agg(["min", "max"]).apply(lambda r: f"{r['min']:,.0f}–{r['max']:,.0f}
  ↪ ", axis=1)
gate = pd.DataFrame({"income range": rng.to_numpy(), "effect": ci["effect"].to_numpy(),
                     "95% CI low": ci["2.5 %"].to_numpy(), "95% CI high": ci["97.5 %"].to_numpy()}, index=q.
  ↪ cat.categories)
print(gate.round(0))
fig, ax = plt.subplots()
xx = np.arange(5); e = gate["effect"]
ax.errorbar(xx, e, yerr=[e - gate["95% CI low"], gate["95% CI high"] - e], fmt="o", color="#0033a0", ecolor="
  ↪ #9fb3d6", capsize=4)
ax.axhline(0, color="#999"); ax.set_xticks(xx); ax.set_xticklabels(gate.index, fontsize=8)
ax.set_ylabel("effect on net financial assets, $"); ax.set_title("Effect of eligibility by income quintile (
  ↪ joint 95% bands)")
```

```
                income range   effect  95% CI low  95% CI high
Q1 (lowest)    −−2,65217,196   4867.0      1707.0       8028.0
Q2             −17,21426,520   1172.0     -3195.0       5538.0
Q3             −26,52337,275   4494.0      1079.0       7910.0
Q4             −37,29653,841  10284.0      4650.0      15917.0
Q5 (highest)  −53,844242,124  19941.0      7438.0      32444.0
```

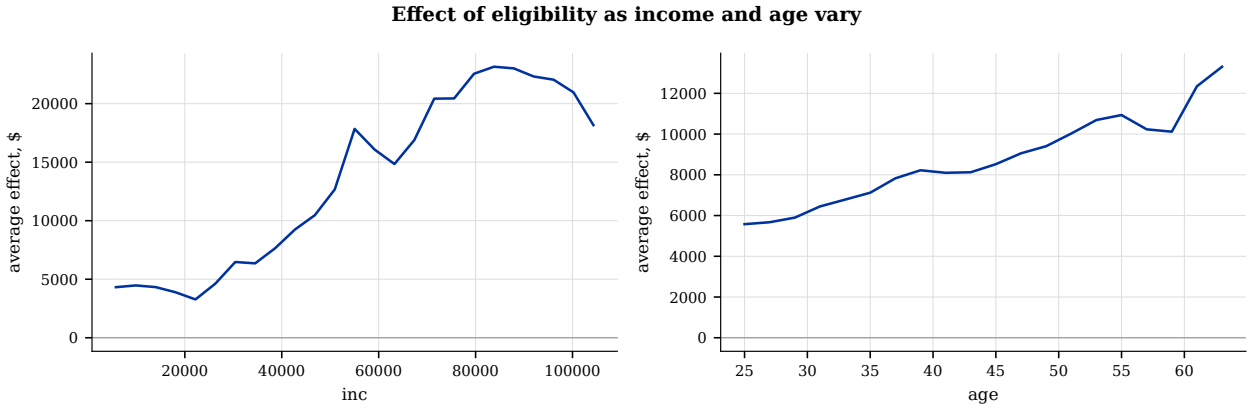


Figure 40.4. How does the effect change with income and age?: Effect of eligibility as income and age vary.

The effect is modest and roughly flat across the bottom three income quintiles and then climbs steeply: the top fifth of earners gain about four times as much as the bottom fifth. The bands are joint, so they can be read together: the top-quintile band lies entirely above the second quintile's, but the bands of neighbouring quintiles overlap, so the evidence is for a rising pattern, not for sharp differences between adjacent groups.

40.6 How does the effect change with income and age?

Hold every other variable at its observed value and vary one: the forest's predicted effect as a function of income (and, separately, age).

```
fig, axes = plt.subplots(1, 2, figsize=(9, 3))
for ax, v, grid in ((axes[0], "inc", np.linspace(df.inc.quantile(.02), df.inc.quantile(.98), 25)),
                    (axes[1], "age", np.arange(25, 65, 2))):
    pts = []
    for val in grid:
        Xg = df[base].copy(); Xg[v] = val
        pts.append(cf.effect(Xg).mean())
    ax.plot(grid, pts, color="#0033a0"); ax.axhline(0, color="#999", lw=.6)
    ax.set_xlabel(v); ax.set_ylabel("average effect, $")
fig.suptitle("Effect of eligibility as income and age vary", fontsize=10, fontweight="bold"); fig.tight_layout
  ↪ ()
```

40.7 Write it up

```
print(f"ATE (forest): {ate:,.0f}, 95% CI {lo:,.0f} to {hi:,.0f}")
print(gate.round(0).to_string())
print("main drivers:", ", ".join(imp.index[:3]))
```

```
ATE (forest): 7,883, 95% CI 5,562 to 10,204
               income range   effect  95% CI low  95% CI high
Q1 (lowest)    −-2,65217,196   4867.0      1707.0       8028.0
Q2             −17,21426,520   1172.0     -3195.0       5538.0
Q3             −26,52337,275   4494.0      1079.0       7910.0
Q4             −37,29653,841  10284.0      4650.0      15917.0
Q5 (highest)  −53,844242,124  19941.0      7438.0      32444.0
main drivers: inc, age, educ
```

How to write it: heterogeneity

To study who gains from 401(k) eligibility, we estimate conditional average treatment effects with a causal forest (Wager and Athey 2018), using the orthogonalised, honest implementation of EconML with 2,000 trees and random-forest first stages for the outcome and the propensity score. The forest's average effect matches the doubly robust DML estimate. Income accounts for most of the splits on the treatment effect, followed by age. Group average effects by income quintile, estimated from doubly robust scores with joint confidence bands (Semenova and Chernozhukov 2021), are small in the bottom three quintiles (between about 1,000 and 5,000

dollars) and then rise sharply, to about 10,300 dollars in the fourth quintile and 19,900 dollars in the top quintile (Table X): eligibility mainly raises the financial assets of higher-income households, although the effect is positive and significant even in the lowest quintile. Individual effects are imprecise; we therefore report group averages with confidence intervals and do not interpret individual estimates.

Checklist for the paper

- Start from the average effect (DML) and show that the forest reproduces it.
- Say in advance which dimensions of heterogeneity are of interest.
- Report the forest settings: number of trees, honesty, leaf size, first-stage learners.
- Report group effects with confidence intervals, not individual effects.
- Show which variables drive the heterogeneity and how the effect varies along them.

PART VI

Bayesian Methods for Macroeconomic Systems

Overview

A VAR has many coefficients and a short sample. Bayesian methods address that mismatch directly: a prior shrinks the coefficients toward a sensible benchmark, and the data are allowed to move them away from it. The Minnesota prior (Doan et al., 1984; Litterman, 1986) and its conjugate versions (Kadiyala and Karlsson, 1997; Bańbura et al., 2010) are the workhorses, and the first chapters of this part develop them, together with a mean-adjusted variant (Villani, 2009) that lets the researcher state beliefs about long-run means.

The remaining chapters extend the VAR to features of real data that a constant-coefficient model misses: many indicators (Bernanke et al., 2005), mixed frequencies, panels of countries (Love and Zicchino, 2006), changing volatility (Primiceri, 2005), time-varying coefficients (Cogley and Sargent, 2005) and thresholds. A short introduction to the whole programme is in Koop and Korobilis (2010); Sims and Zha (2006) is a reminder of how easily volatility and coefficient changes are confused. The estimation code is run through the OES engine described in Appendix B.

Chapters at a glance

Ch.	Chapter	Use it when
41	Bayesian Linear Regression with a g-Prior	A cross-section regression where you want posterior probabilities, prior information or model comparison
42	Bayesian VAR with Minnesota and Normal-Wishart Priors	A VAR with many coefficients and a short sample
43	Mean-Adjusted (Steady-State) Bayesian VAR	Medium- and long-horizon forecasts, or any VAR where you know something about the long-run means
44	Factor-Augmented VAR (FAVAR)	A VAR that should use information from many macro series without adding them all as variables
45	Mixed-Frequency BVAR: Nowcasting Quarterly GDP with Monthly Data	Quarterly GDP and monthly indicators that arrive earlier, for nowcasting or for a model at the monthly frequency
46	Panel VAR: Six Estimators for Heterogeneous Countries	A VAR for many countries, regions or firms with short time series
47	Stochastic-Volatility VAR (SV-VAR)	Shocks that were large in some periods and small in others, as in the Great Moderation
48	Time-Varying Parameter VAR (TVP-VAR)	You suspect the transmission of shocks has changed over the sample
49	Bayesian Threshold VAR	Dynamics that differ when an observed variable, such as inflation, is above or below a threshold

Further reading

Koop and Korobilis (2010) and Del Negro and Schorfheide (2011) survey Bayesian macroeconometrics. The g-prior for regression is due to Zellner (1986), and its use in model comparison is examined in Fernández et al. (2001). On the practical choice of prior tightness, see Bańbura et al. (2010). Schorfheide and Song (2015) show how a mixed-frequency model is used for real-time forecasting.

CHAPTER 41

Bayesian Linear Regression with a g-Prior

Applies when. A cross-section regression where you want posterior probabilities, prior information or model comparison

Overview. Use this chapter when you want a regression that reports the probability that an effect is positive, uses information from earlier studies, or compares specifications by how well they explain the data. With a reasonable sample the posterior means are close to OLS; the value is in the interpretation, the prior sensitivity table and the marginal likelihood. The chapter covers the prior, estimation, model comparison, diagnostics, robustness and the write-up.

Data. Wooldridge WAGE2, 663 U.S. men with complete data: log wage, education, experience, tenure, IQ and demographics

Software. Python · OES engine · statsmodels

41.1 The question and the model

How much does an extra year of education raise wages, once experience, tenure, ability and demographics are held fixed? This is the Mincer wage regression. The model is

$$\ln w_i = \alpha + x_i'\beta + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma^2),$$

with x_i containing education, experience, tenure, IQ and dummies for marriage, race, region and city. OLS gives one number per coefficient and a standard error. A Bayesian regression gives a whole **posterior distribution** for each coefficient: you can read off the probability that the return to schooling exceeds 5%, combine the data with the results of earlier studies, and compare specifications without a p-value threshold.

The OES engine uses a **conjugate** Normal-inverse-gamma prior. Everything is exact: no simulation, no convergence to check, and the log marginal likelihood is available in closed form. Two priors are offered for the slopes:

Prior	What it says	When to use it
Zellner's g-prior	$\beta \sim N(\beta_0, g\,\sigma^2(X'X)^{-1})$: the prior is as informative as $1/g$ of an observation's worth of the sample, scaled to the regressors	Default. Unit-free, so it needs no judgement about the scale of each variable. $g = n$ is the "unit-information prior"
Normal	$\beta_j \sim N(\beta_{0j}, s^2)$, the same standard deviation for every slope	When the variables are in comparable units, or when you have a prior from earlier work

The constant has a flat prior and σ^2 an inverse-gamma(0.01, 0.01) prior in both cases. A large g (or large s) lets the data speak; a small one pulls the slopes towards the prior mean β_0.

Frequentist or Bayesian?

With 663 observations and a weak prior, the two give almost the same numbers, and that is the point: a Bayesian regression is a way of *reporting* the answer (probabilities, intervals that mean what people think they mean) and of *using* extra information, not a different estimate. Use it when the sample is small, when you have genuine prior information, or when model comparison is part of the question.

How to write it: the model

We estimate the log-wage equation by Bayesian linear regression with a Zellner g-prior on the slopes ($g = n$, the unit-information prior), a flat prior on the constant and an inverse-gamma(0.01, 0.01) prior on the error variance (Zellner 1986; Liang et al. 2008). With this conjugate prior the marginal posterior of each coefficient is a Student-t distribution, available in closed form, and we report its mean, standard deviation, 95% credible interval and the posterior probability that the coefficient is positive.

41.2 Load the data

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("wage2.csv").read()
df = pd.read_csv("wage2.csv")
x = ["educ", "exper", "tenure", "IQ", "married", "black", "south", "urban"]
print(df[["lwage"] + x].describe().round(2).loc[["count", "mean", "std", "min", "max"]])
```

```
        lwage    educ   exper  tenure      IQ  married   black   south   urban
count  663.00  663.00  663.00  663.00  663.00    663.0  663.00  663.00  663.00
mean     6.81   13.68   11.40    7.22  102.48      0.9    0.08    0.32    0.72
std      0.41    2.23    4.26    5.06   14.69      0.3    0.27    0.47    0.45
min      4.74    9.00    1.00    0.00   54.00      0.0    0.00    0.00    0.00
max      8.03   18.00   22.00   22.00  145.00      1.0    1.00    1.00    1.00
```

How to read it

663 men, with log wages averaging 6.81 (about $900 a month) and a standard deviation of 0.41. Education averages 13.7 years. Nine in ten are married, 8% are Black, 32% live in the South and 72% in a city. Dummy variables like these have a standard deviation below 0.5, while IQ has 14.7: the **g-prior** is useful here because it scales the prior for each coefficient by the spread of its regressor, so you do not have to choose a different prior standard deviation for IQ and for a dummy.

41.3 Estimate

```
spec = dict(y="lwage", x=x)
r = oes.run("bayes_reg", data, spec)
oes.summary(r)
oes.show(r, "Posterior")
oes.show(r, "OLS")
```

```
Observations: 663
Prior: g-prior
Method: exact (conjugate)
Posterior mean of sigma^2: 0.1256
Log marginal likelihood: -299.42
Posterior of the coefficients (Zellner g-prior on the slopes, g = 663, flat constant)
             Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const      5.1272     0.1464  4.8402  5.1272  5.4143   1.000
educ       0.0550     0.0081  0.0392  0.0550  0.0709   1.000
exper      0.0159     0.0038  0.0084  0.0159  0.0234   1.000
tenure     0.0087     0.0029  0.0031  0.0087  0.0143   0.999
IQ         0.0038     0.0012  0.0015  0.0038  0.0061   0.999
married    0.2035     0.0463  0.1127  0.2035  0.2943   1.000
black     -0.1222     0.0540 -0.2280 -0.1222 -0.0164   0.012
south     -0.0632     0.0305 -0.1230 -0.0632 -0.0034   0.019
urban      0.2002     0.0312  0.1391  0.2002  0.2614   1.000
Note: Exact Student-t marginal posteriors with 663.02 degrees of freedom (Normal-inverse-gamma conjugate prior).

OLS for comparison
            Coef.  Std. err.
Parameter
```

```
const      5.1247     0.1472
educ       0.0551     0.0081
exper      0.0159     0.0038
tenure     0.0087     0.0029
IQ         0.0038     0.0012
married    0.2038     0.0466
black     -0.1224     0.0543
south     -0.0633     0.0307
urban      0.2005     0.0314
```

How to read it

Each row is a posterior distribution. Take education: the posterior mean is 0.055 with a standard deviation of 0.008, so a year of schooling raises wages by about 5.5%, and there is a 95% probability that the effect lies between 3.9% and 7.1%. That is a direct statement about the parameter; a frequentist confidence interval cannot be read that way. The last column, $P(>0)$, is the posterior probability that the effect is positive: 1.000 for education, experience, marriage and city; 0.999 for tenure and IQ; and 0.012 and 0.019 for Black and South, whose effects are negative with high probability.

Compare with OLS: coefficients agree to the third decimal place. The g-prior with $g = n$ carries the weight of a single observation, so with 663 observations it moves nothing. This is the expected result with a weak prior and a decent sample; Section 41.6 shows what happens with a stronger prior.

41.4 Compare specifications by marginal likelihood

The **log marginal likelihood** is the probability of the data under the model with the coefficients integrated out. It rewards fit and automatically penalises a model for having parameters it does not need. The difference between two models is the log **Bayes factor**.

```
base = oes.run("bayes_reg", data, spec)
lml = {"full model": base["summary"][4]["value"]}
for drop in ["IQ", "tenure", "black", "south", "urban"]:
    rr = oes.run("bayes_reg", data, dict(spec, x=[v for v in x if v != drop]))
    lml[f"without {drop}"] = rr["summary"][4]["value"]
out = pd.Series(lml, name="log marginal likelihood").to_frame()
out["log Bayes factor (full vs. model)"] = out.iloc[0, 0] - out["log marginal likelihood"]
print(out.round(2))
```

```
               log marginal likelihood  log Bayes factor (full vs. model)
full model                     -299.42                               0.00
without IQ                     -301.46                               2.04
without tenure                 -300.72                               1.30
without black                  -298.73                              -0.69
without south                  -298.32                              -1.10
without urban                  -316.23                              16.81
```

How to read it

A log Bayes factor above about 1 is ”some evidence” for the full model, above 2.3 is ”strong” and above 4.6 ”decisive” (Jeffreys’ scale). Dropping `urban` costs almost 17 log points: the city dummy is decisive. Dropping `IQ` costs 2.0 and `tenure` 1.3: moderate evidence for both. Dropping `south` or `black` *raises* the marginal likelihood by 1.1 and 0.7: the data, after the penalty for extra parameters, slightly prefer the model without them. This does not mean the coefficients are zero (both have $P(>0)$ below 0.02); it means they are not worth their cost in predictive terms.

Report the table if the specification is a topic in the paper, and describe the scale you use. Do not use the Bayes factor as a mechanical rule; use it with the posterior intervals and with economic reasoning.

41.5 Use prior information

If earlier studies find a return to education of about 7%, you can say so. `prior_mean` sets the prior mean of each slope: one number for all, or a list in the order of the regressors.

```
r7 = oes.run("bayes_reg", data, dict(spec, prior_mean="0.07, 0, 0, 0, 0, 0, 0, 0"))
```

```
print("log marginal likelihood:", r7["summary"][4]["value"], "versus", base["summary"][4]["value"], "with a
  ↪ zero prior mean")
oes.show(r7, "Posterior", rows=3)
```

```
log marginal likelihood: -299.352 versus -299.42 with a zero prior mean
Posterior of the coefficients (Zellner g-prior on the slopes, g = 663, flat constant)
            Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const     5.1258     0.1464  4.8388  5.1258  5.4128     1.0
educ      0.0551     0.0081  0.0393  0.0551  0.0710     1.0
exper     0.0159     0.0038  0.0084  0.0159  0.0234     1.0
Note: Exact Student-t marginal posteriors with 663.02 degrees of freedom (Normal-inverse-gamma conjugate prior).
```

How to read it

The posterior mean of the education coefficient barely moves (0.0551 against 0.0550), because this prior carries the weight of one observation. To let earlier work matter, use a **small g** or a Normal prior with a small standard deviation, and report the sensitivity table below. In a small sample the prior would matter much more, and writing it down becomes part of the paper's identification of the result.

41.6 Robustness: how much does the prior matter?

```
oes.show(r, "Prior sensitivity")
```

```
Prior sensitivity: posterior means of the slopes
                          educ   exper  tenure      IQ  married   black   south   urban
Prior
g-prior, g = 1          0.0276  0.0080  0.0044  0.0019   0.1019 -0.0612 -0.0316  0.1003
g-prior, g = 10         0.0501  0.0145  0.0079  0.0035   0.1853 -0.1112 -0.0575  0.1823
g-prior, g = 663        0.0550  0.0159  0.0087  0.0038   0.2035 -0.1222 -0.0632  0.2002
g-prior, g = 6630       0.0551  0.0159  0.0087  0.0038   0.2038 -0.1223 -0.0633  0.2005
normal prior, s.d. 0.1  0.0528  0.0163  0.0092  0.0048   0.0748 -0.0366 -0.0458  0.1116
normal prior, s.d. 1    0.0551  0.0159  0.0087  0.0039   0.2004 -0.1196 -0.0632  0.1988
normal prior, s.d. 10   0.0551  0.0159  0.0087  0.0038   0.2038 -0.1223 -0.0633  0.2005
Note: Conclusions that hold across the rows are robust to the prior. Small g or a small prior s.d. pulls the slopes
↪ towards the prior mean.
```

How to read it

Each row re-estimates the slopes under a different prior. For $g \geq 663$ and for the Normal priors with standard deviation 1 or 10, the slopes are the same as OLS: the data dominate. With $g = 10$ the coefficients shrink by about 9% (education 0.050), and with $g = 1$ by half (education 0.028; marriage 0.10 against 0.20). The tight Normal prior (standard deviation 0.1) pulls every slope towards zero, with the largest effect on marriage (0.075) and race (-0.037), which have the largest coefficients.

What to say: the signs and ranking of the effects do not depend on the prior unless it is much more informative than the data. If a result flips only for a prior that acts like a handful of observations, it is not robust. The table is cheap evidence to show a referee.

For a non-conjugate prior (a Normal prior with the *same* standard deviation for each slope and independent of σ^2), the posterior is obtained by a Gibbs sampler, which needs a convergence check:

```
rg = oes.run("bayes_reg", data, dict(spec, method="gibbs", prior="normal", prior_sd=1, draws=3000, burn=500))
oes.show(rg, "MCMC")
```

```
MCMC convergence diagnostics
           Effective sample size  Geweke z  p-value
Parameter
const                     3000.0    -0.115   0.9087
educ                      3000.0     0.120   0.9046
exper                     3000.0     0.086   0.9314
tenure                    3000.0     0.790   0.4296
IQ                        3000.0    -0.329   0.7425
married                   3000.0     0.972   0.3309
black                     3000.0     2.003   0.0451
south                     3000.0    -0.581   0.5609
urban                     3000.0    -0.774   0.4388
Note: Geweke compares the first 10% and last 50% of the chain; a large |z| suggests the chain has not converged.
```

How to read it

The effective sample size equals the number of draws (3000) for every coefficient, because a Gibbs sampler for a linear model draws from two exact conditional distributions and the chain mixes immediately. Geweke's test compares the first 10% and the last 50% of the chain: a statistic with |z| above 2 would mean the chain had not settled. It is borderline for `black` (z = 2.0): with nine parameters one such value is expected by chance, and increasing the burn-in confirms nothing changes. Report effective sample sizes for any sampler you use.

41.7 Diagnostics: residuals

```
import statsmodels.api as sm
oes.show(r, "Residual diagnostics")
ols = sm.OLS(df["lwage"], sm.add_constant(df[x])).fit(cov_type="HC3")
cmp = pd.DataFrame({"posterior sd": oes.frame(next(t for t in r["tables"] if t["title"].startswith("Posterior")
  ↪ ))["Std. dev."],
                    "OLS HC3 s.e.": ols.bse})
cmp["ratio"] = cmp["OLS HC3 s.e."] / cmp["posterior sd"]
print(cmp.round(4))
```

```
Residual diagnostics (at the posterior mean)
                                Statistic  p-value
↪                Detail
Test
Portmanteau (Hosking), 12 lags      11.7807   0.4634           df = 12; H0: no residual autocorrelation (approximate
↪ after estimating lags)
Ljung-Box, residual                 11.8163   0.4605
↪ H0: no autocorrelation
Jarque-Bera, residual              135.2706   0.0000
↪   H0: normal residuals
ARCH LM, residual                    3.1750   0.5290                                                    H0: no
↪ conditional heteroskedasticity
Breusch-Pagan heteroskedasticity    23.4996   0.0028  H0: constant variance; if rejected, the normal likelihood
↪ understates the uncertainty

          posterior sd  OLS HC3 s.e.   ratio
const           0.1464        0.1425  0.9734
educ            0.0081        0.0085  1.0494
exper           0.0038        0.0039  1.0285
tenure          0.0029        0.0030  1.0447
IQ              0.0012        0.0011  0.9337
married         0.0463        0.0486  1.0487
black           0.0540        0.0487  0.9020
south           0.0305        0.0323  1.0599
urban           0.0312        0.0311  0.9982
```

How to read it

Residual autocorrelation is rejected by neither the portmanteau nor the Ljung-Box test, as expected in a cross-section. Residuals are not normal (Jarque-Bera rejects) and the variance is not constant (Breusch-Pagan, p = 0.003). The regression coefficients stay consistent, but the posterior standard deviations assume a constant variance and are too small if the variance is unevenly spread: the table compares the posterior standard deviations with heteroskedasticity-robust (HC3) OLS standard errors. The ratios run from 0.90 (Black) to 1.06 (South), so the robust errors are within about 10% of the posterior ones and the conclusions stand, with one qualification: with robust errors the South coefficient is borderline (t about 1.96). Report the robust errors next to the posterior as a check; the point estimates are the same.

41.8 Write the results

How to write it: results

Table 1 reports posterior summaries. A year of education raises hourly wages by 5.5% (posterior standard deviation 0.8 percentage points; 95% credible interval 3.9% to 7.1%), with a posterior probability of one that the effect is positive. Experience, tenure, IQ, marriage and living in a city have positive effects (probability above 0.999), while the effects of being Black or living in the South are negative with probabilities of 0.988 and 0.981. The posterior means are almost identical to OLS because the g-prior carries the weight of one observation. The log Bayes factors indicate decisive support for the city dummy (16.8), and moderate support for IQ (2.0) and tenure (1.3); the data slightly prefer a model without the South and Black dummies, though their credible intervals exclude zero. The signs and ranking of the effects are unchanged for all priors except those that carry the weight of fewer than ten observations (Table 2). Residual diagnostics show heteroskedasticity, so we also compare with heteroskedasticity-robust OLS standard errors, which are within 10% of the posterior standard deviations; the South coefficient is then only borderline significant.

Report	What	Where
Prior	Family, g or s, β_0, the σ^2 prior	Section 41.1
Results	Posterior mean, standard deviation, 95% interval, $P(>0)$	Section 41.3
Comparison	Log marginal likelihood or Bayes factors, with the scale used	Section 41.4
Robustness	Posterior means under different g and s	Section 41.6
Diagnostics	Residual tests; effective sample size and Geweke for Gibbs	Sections 41.6 and 41.7

References. Zellner (1986), *Bayesian Inference and Decision Techniques*; Liang, Paulo, Molina, Clyde and Berger (2008), *Journal of the American Statistical Association*; Kass and Raftery (1995), *Journal of the American Statistical Association*; Mincer (1974), *Schooling, Experience, and Earnings*; Gelman et al. (2013), *Bayesian Data Analysis*, 3rd ed.

CHAPTER 42

Bayesian VAR with Minnesota and Normal-Wishart Priors

Applies when. A VAR with many coefficients and a short sample

Overview. Use this chapter when a VAR has more coefficients than the data can pin down, which is almost always the case with four lags of three or more quarterly variables. A Bayesian VAR shrinks the coefficients towards a simple benchmark, gives sharper impulse responses and better forecasts, and reports its uncertainty directly. The chapter covers the choice of prior, lag length and tightness, estimation, diagnostics, robustness and the write-up.

Data. U.S. quarterly GDP growth, inflation and the 3-month T-bill rate, 1959–2009

Software. Python · OES engine · statsmodels

42.1 The question and the model

What happens to output and inflation after an unexpected rise in the short-term interest rate? This is the classic question of the monetary VAR. The model is a VAR(p) in three variables, quarterly GDP growth, CPI inflation and the 3-month Treasury bill rate:

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + u_t, \qquad u_t \sim N(0, \Sigma).$$

With $p = 4$ lags each equation has $1 + 3 \times 4 = 13$ coefficients, 39 in all, plus 6 in Σ. Fifty years of quarterly data give about 200 observations: enough to estimate the coefficients, but not to estimate them *precisely*. Unrestricted OLS fits the noise, impulse responses come out jagged, and forecasts are poor.

A **Bayesian VAR** adds a prior that pulls every coefficient towards a sensible benchmark. The **Minnesota prior** (Litterman 1986) says that each variable behaves like its own random walk or white noise, that recent lags matter more than distant ones, and that other variables' lags matter less than a variable's own. The data then move the coefficients away from this benchmark only as far as they have evidence for.

Term	Meaning	Default here
δ_i	prior mean of variable i's own first lag	1 for an I(1) series in levels, 0 for a stationary one
λ_1	overall tightness: how strongly all coefficients are pulled to the prior	0.2, then chosen by the data
λ_2	extra tightness on the lags of *other* variables	0.5
λ_3	how fast the prior tightens with the lag length ($1/l^{\lambda_3}$)	1
λ_4	tightness on the constant	100 (essentially flat)

Four priors, one idea

The OES engine offers the four priors of the ECB's BEAR toolbox. They share the Minnesota means and scales and differ in how they treat Σ:

- **Minnesota (Litterman):** Σ is fixed at the residual variances of univariate AR models. Exact, fast, equation by equation. Good for forecasting.
- **Normal-Wishart:** Σ is estimated with an inverse-Wishart prior, and the posterior and the *marginal likelihood* are analytic, so lag length and tightness can be chosen by the data. It forces $\lambda_2 = 1$.

- **Independent Normal-Wishart:** frees λ_2 and allows block exogeneity; estimated by Gibbs sampling.
- **Normal-diffuse:** Minnesota prior on the coefficients, no information on Σ; Gibbs sampling.

Start with Normal-Wishart to choose the lags and λ_1, then report the prior you prefer and show the others as robustness.

How to write it: the model

We estimate a Bayesian VAR in real GDP growth, CPI inflation and the 3-month Treasury bill rate, $y_t = c + \sum_{l=1}^{p} A_l y_{t-l} + u_t$, $u_t \sim N(0, \Sigma)$, on quarterly U.S. data for 1959Q3–2009Q3. Following Litterman (1986) and Kadiyala and Karlsson (1997), we use a Normal-Wishart prior with Minnesota moments: the prior mean of each variable's own first lag is one for the interest rate, which is persistent, and zero for growth and inflation; all other coefficients have prior mean zero, with prior standard deviations that shrink with the lag length.

42.2 Load the data and test for unit roots

The prior mean of the own first lag depends on whether a series is persistent. Test first.

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
print(df[["date", "gdp_growth", "inflation", "tbilrate"]].describe().round(2).loc[["count", "mean", "std", "
  ↪ min", "max"]])

spec = dict(vars=["gdp_growth", "inflation", "tbilrate"], time="date", lags=4, prior="nw", lambda1=0.2,
            delta="0, 0, 1", sens_shock="tbilrate", sens_response="gdp_growth")
r = oes.run("bvar", data, spec)
oes.show(r, "Unit-root")
```

```
       gdp_growth  inflation  tbilrate
count      202.00     202.00    202.00
mean         3.10       3.98      5.32
std          3.52       3.25      2.80
min         -8.28      -8.79      0.12
max         15.43      14.62     15.33
Unit-root tests
            ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
gdp_growth                  0.0000               0.0000                      0.0         0.1000       I(0)
inflation                   0.0271               0.0822                      0.0         0.0808       I(0)
tbilrate                    0.2805               0.4462                      0.0         0.0538       I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

GDP growth and inflation reject a unit root in levels (ADF p-values below 0.05) and are treated as stationary: prior mean 0 on their own first lag. The T-bill rate does not reject; it is stationary only after differencing, so it gets a random-walk prior (prior mean 1). Notice that nothing is differenced: a Bayesian VAR is estimated in levels, and the prior handles the persistence. Sims, Stock and Watson (1990) show that inference on impulse responses remains valid in levels even with unit roots or cointegration.

42.3 Choose the lag length and the tightness

The Normal-Wishart prior gives the **marginal likelihood**, the probability of the data under the model with the coefficients integrated out. It rewards fit and penalises complexity automatically, so it can choose both the number of lags and λ_1 (Giannone, Lenza and Primiceri 2015).

```
oes.show(r, "Lag order")
```

```
oes.show(r, "Overall tightness")
```

```
Lag order: Bayesian marginal likelihood and BIC
      Log marginal likelihood (higher is better)  BIC of the OLS VAR (lower is better)
Lags
1                                     -1226.032                                  3.8637
2                                     -1215.847                                  3.8769
3                                     -1210.105                                  3.9040
4                                     -1209.707                                  4.0682
5                                     -1208.721                                  4.2123
6                                     -1207.614                                  4.3131
7                                     -1207.258                                  4.5501
8                                     -1207.261                                  4.7121
Note: Common sample; Normal-Wishart prior with lambda1 = 0.2. Highest marginal likelihood: 7 lag(s).

Overall tightness lambda1 by marginal likelihood
         Log marginal likelihood
lambda1
0.02                  -1292.188
0.05                  -1264.523
0.10                  -1246.079
0.20                  -1235.260
0.50                  -1239.139
1.00                  -1255.290
5.00                  -1309.492
Note: Normal-Wishart prior, 4 lag(s). The data prefer lambda1 = 0.2 (Giannone, Lenza and Primiceri 2015).
```

How to read it

- **Lags.** The marginal likelihood rises quickly up to three or four lags and then flattens: extra lags add little once the prior has shrunk them. BIC, which has no prior to shrink with, prefers fewer lags. With quarterly data four lags is the convention, and the marginal likelihood supports it; more lags would be harmless under the prior.
- **Tightness.** The marginal likelihood peaks at $\lambda_1 = 0.2$: looser priors overfit, tighter ones ignore the data. This is the value used below. Always report how λ_1 was chosen; a value picked to deliver a result is a red flag for referees.

42.4 Estimate the BVAR

```
oes.summary(r)
oes.show(r, "Equation: tbilrate")
```

```
Observations: 198
Prior: Normal-Wishart
Lags: 4
lambda1: 0.2
Share of stable draws: 0.982
Distance from OLS (relative): 0.556
Log marginal likelihood: -1235.26
Equation: tbilrate
                 Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const         -0.0267     0.1641 -0.3434 -0.0275  0.2864   0.426
L1.gdp_growth  0.0338     0.0179 -0.0013  0.0332  0.0678   0.973
L1.inflation   0.0003     0.0255 -0.0473  0.0002  0.0487   0.502
L1.tbilrate    0.9315     0.0656  0.8064  0.9316  1.0633   1.000
L2.gdp_growth  0.0221     0.0152 -0.0082  0.0217  0.0521   0.929
L2.inflation   0.0315     0.0215 -0.0095  0.0316  0.0722   0.927
L2.tbilrate   -0.0768     0.0654 -0.2007 -0.0795  0.0506   0.144
L3.gdp_growth  0.0056     0.0125 -0.0203  0.0060  0.0291   0.657
L3.inflation   0.0329     0.0183 -0.0020  0.0323  0.0676   0.964
L3.tbilrate    0.0979     0.0499  0.0029  0.0970  0.1969   0.979
L4.gdp_growth  0.0017     0.0101 -0.0190  0.0020  0.0204   0.571
L4.inflation   0.0026     0.0160 -0.0279  0.0028  0.0335   0.567
L4.tbilrate   -0.0377     0.0377 -0.1118 -0.0371  0.0349   0.155
```

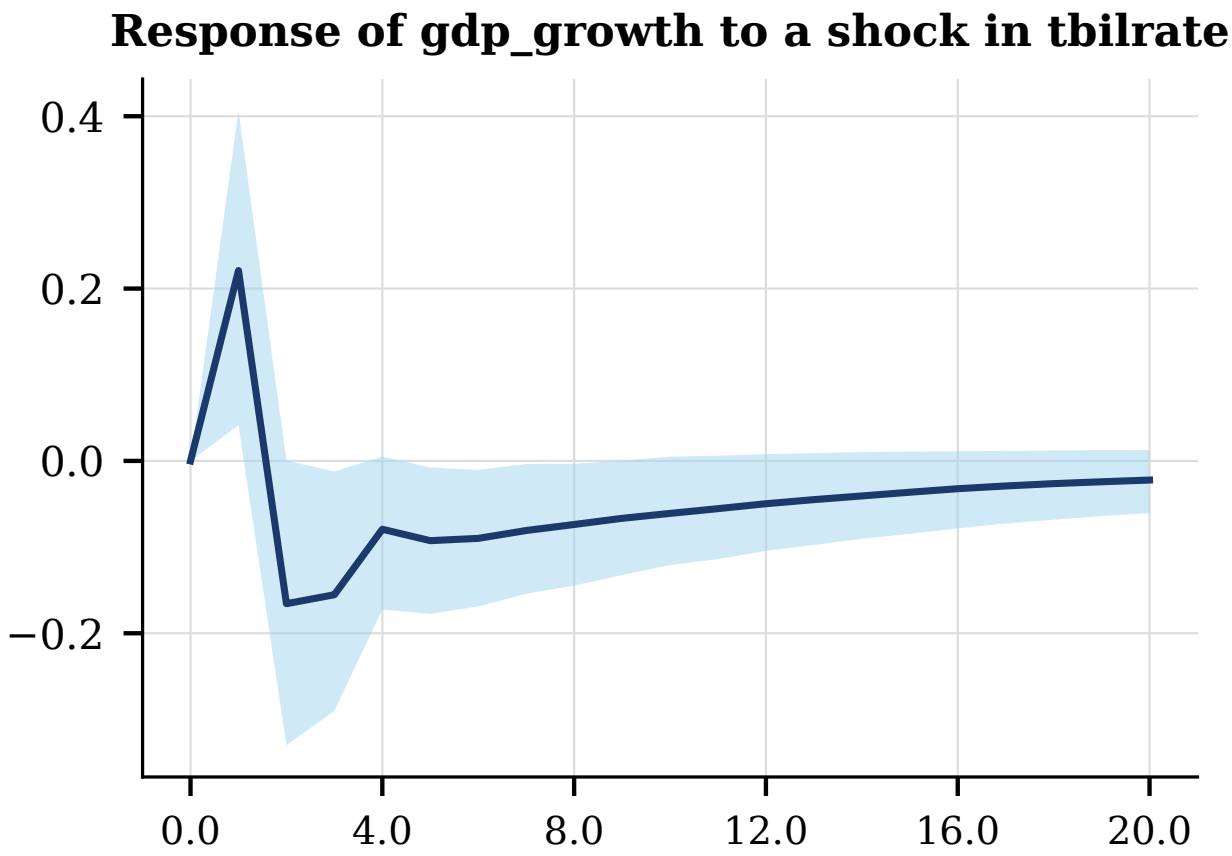


Figure 42.1. Impulse responses: a monetary policy shock: Response of gdp_growth to a shock in tbilrate.

How to read it

Each row is a posterior: its mean, standard deviation, 95% credible interval and the posterior probability that the coefficient is positive. The interest-rate equation is a monetary policy rule in reduced form: the T-bill rate is highly persistent (own first lag about 0.93) and rises after strong GDP growth and higher inflation, with posterior probabilities above 0.9 for several of these lags. "Distance from OLS" in the summary shows how much the prior moved the coefficients: about half, which is typical for a VAR of this size.

A credible interval is read literally: given the model and the prior, there is a 95% probability that the coefficient lies in it. Bayesian results do not need p-values, but P(> 0) close to 0 or 1 plays the same role.

42.5 Impulse responses: a monetary policy shock

The policy shock is identified recursively (Cholesky), with the interest rate ordered **last**: the Fed reacts within the quarter to output and inflation, while output and prices respond to the interest rate only with a lag. This is the standard ordering of Christiano, Eichenbaum and Evans (1999).

```
fig = oes.plot(r, "Response of gdp_growth to a shock in tbilrate")
fig = oes.plot(r, "Response of inflation to a shock in tbilrate")
```

How to read it

After a one-standard-deviation rise in the T-bill rate, GDP growth falls, and the 68% credible band stays below zero for about two years. Inflation, however, **rises** before it falls back: the well-known *price puzzle* (Sims 1992). It usually means the VAR omits information the Fed uses to forecast inflation, such as commodity prices; Chapter 44 shows how adding many indicators addresses it. Report the puzzle honestly rather than hiding it; it is a known limitation of small monetary VARs.

Bayesian VAR papers report 68% bands (16th to 84th percentiles) by convention, as here; add 90% bands if referees ask.

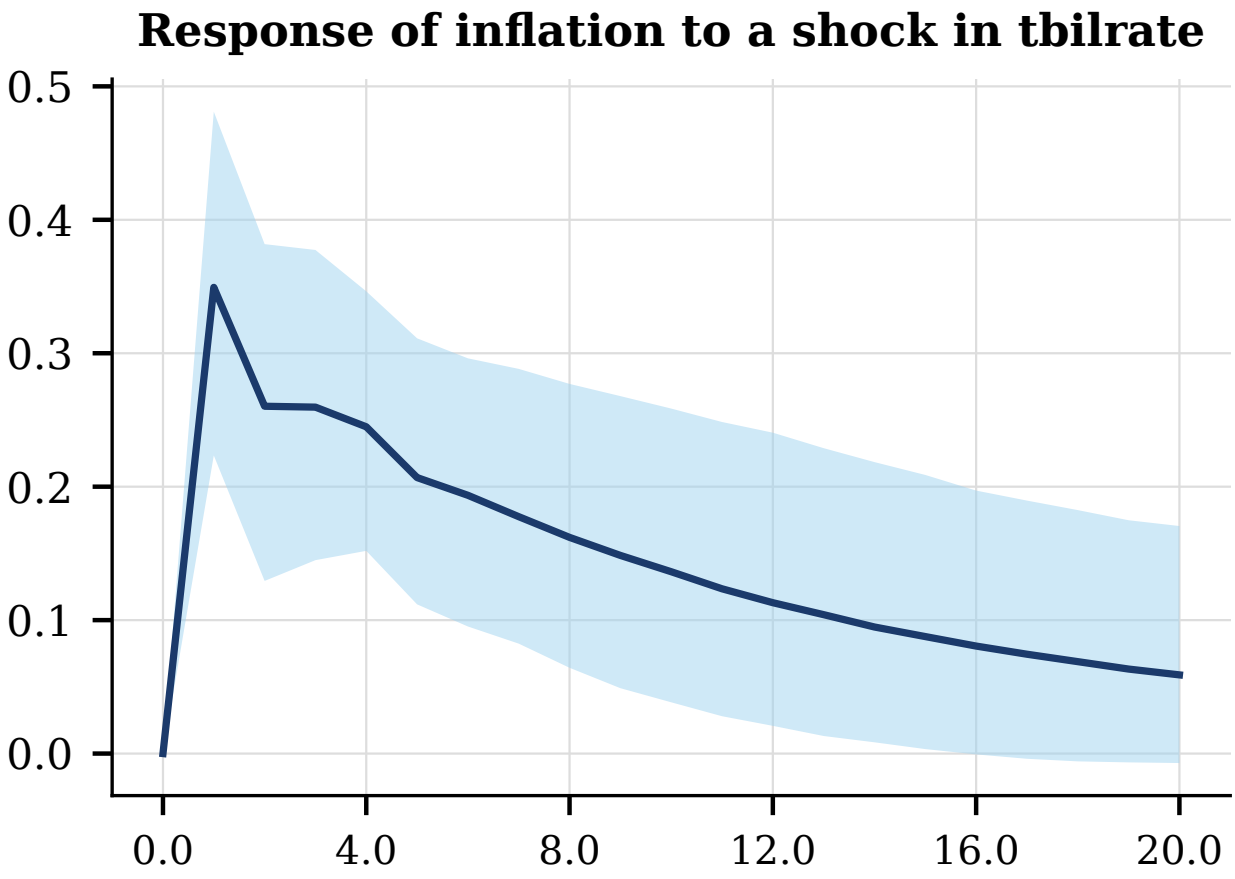


Figure 42.2. Impulse responses: a monetary policy shock: Response of inflation to a shock in tbilrate.

42.6 How important is the shock? Variance decomposition

```
oes.show(r, "Forecast-error variance")
```

```
Forecast-error variance decomposition (posterior median, %)
            Horizon  shock: gdp_growth  shock: inflation  shock: tbilrate
Variable
gdp_growth        1              100.0               0.0              0.0
gdp_growth        4               96.3               2.1              1.4
gdp_growth        8               93.3               4.7              1.7
gdp_growth       12               92.2               5.4              2.0
gdp_growth       20               91.6               5.9              2.2
inflation         1                1.5              98.5              0.0
inflation         4                3.9              91.8              3.5
inflation         8                5.5              88.4              5.0
inflation        12                6.3              86.7              5.7
inflation        20                7.1              85.3              6.4
tbilrate          1                9.2              11.7             78.6
tbilrate          4               20.4              14.7             64.1
tbilrate          8               24.4              19.3             54.2
tbilrate         12               25.4              21.8             50.2
tbilrate         20               25.9              23.7             47.8
Note: Share of each variable's h-step forecast-error variance due to each Cholesky shock.
```

How to read it

Monetary policy shocks explain only a few percent of the forecast-error variance of GDP growth and inflation, even after five years. Most movements in output and inflation are due to other shocks; the interest rate itself is driven increasingly by output and inflation, as the policy rule implies. This is the standard result: policy matters for *how* the economy responds, but surprises in policy are a small source of fluctuations.

42.7 Diagnostics: residuals, stability and convergence

```
oes.show(r, "Residual diagnostics")
oes.show(r, "Stability")
```

```
Residual diagnostics (at the posterior mean)
                               Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   160.7980   0.0007  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth             10.4062   0.5804                                                          H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           32.6349   0.0000                                                            H0:
↪ normal residuals
ARCH LM, gdp_growth                3.9381   0.4144                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation              14.6870   0.2590                                                          H0: no
↪ autocorrelation
Jarque-Bera, inflation           124.0184   0.0000                                                            H0:
↪ normal residuals
ARCH LM, inflation                28.1739   0.0000                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               34.9229   0.0005                                                          H0: no
↪ autocorrelation
Jarque-Bera, tbilrate           1909.6458   0.0000                                                            H0:
↪ normal residuals
ARCH LM, tbilrate                 50.7086   0.0000                                          H0: no conditional
↪ heteroskedasticity

Stability
                                                                        Value
Item
Share of posterior draws with all companion roots inside the unit circle  0.9820
Largest root modulus at the posterior mean                                0.9326
Note: Unit roots in levels data make many draws non-stationary; that is expected with a random-walk prior and is not
↪ a failure.
```

How to read it

- **Stability.** Almost every posterior draw has all roots inside the unit circle, and the largest root at the posterior mean is below one: the VAR is stationary and impulse responses die out.
- **Autocorrelation.** Ljung-Box does not reject for GDP growth and inflation but does for the T-bill rate, and the joint portmanteau test rejects. A fifth or sixth lag removes most of it; check that the responses do not change (they do not, in Section 42.7).
- **Normality and ARCH.** Strongly rejected for inflation and the interest rate. The 1970s oil shocks and the Volcker disinflation produce large residuals, and their variance changes over time. The coefficient estimates remain valid, but bands that assume constant variance are too narrow in volatile periods. Chapter 47 models this directly; mention it as a limitation.

With the Normal-Wishart prior the posterior draws are independent, so there is nothing to converge. The Gibbs samplers (Independent Normal-Wishart, Normal-diffuse) need a check:

```
r_inw = oes.run("bvar", data, dict(spec, prior="inw", draws=2000, burn=1000))
oes.show(r_inw, "MCMC")
```

```
MCMC convergence diagnostics (coefficients)
                         Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
gdp_growth: const                       2000.0                       1.0    -1.131   0.2579
inflation: const                        2000.0                       1.0     0.070   0.9445
gdp_growth: L1.gdp_growth               2000.0                       1.0     0.758   0.4487
tbilrate: L1.gdp_growth                 2000.0                       1.0     1.109   0.2675
inflation: L1.inflation                 2000.0                       1.0    -0.340   0.7341
inflation: L1.tbilrate                  2000.0                       1.0     0.298   0.7660
tbilrate: L1.tbilrate                   2000.0                       1.0     0.268   0.7884
gdp_growth: L2.gdp_growth               2000.0                       1.0     1.722   0.0851
```

```
gdp_growth: L2.inflation                2000.0                      1.0     0.403   0.6872
inflation: L2.inflation                 2000.0                      1.0     0.230   0.8183
inflation: L3.inflation                 2000.0                      1.0     2.399   0.0164
tbilrate: L3.tbilrate                   2000.0                      1.0    -0.619   0.5357
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 1.0.
```

How to read it

Effective sample sizes are close to the number of draws and the Geweke statistics are small: the chain mixes well and has settled. If |z| were above 2 for several parameters, increase the burn-in.

42.8 Robustness: prior, tightness and forecasts

A result that depends on the prior is not a result. Three checks:

```
oes.show(r, "Prior sensitivity")
for prior in ["minnesota", "nw", "inw", "nd"]:
    rr = oes.run("bvar", data, dict(spec, prior=prior))
    irf = next(c for c in rr["charts"] if c["title"] == "Response of gdp_growth to a shock in tbilrate")["
  ↪ series"][0]["y"]
    print(f"{prior:10s} response of GDP growth after 4, 8, 12 quarters: " + ", ".join(f"{irf[h]:+.3f}" for h
  ↪ in (4, 8, 12)))
```

```
Prior sensitivity: response of gdp_growth to a tbilrate shock
               h = 0   h = 4   h = 8  h = 12
lambda1
0.05             0.0 -0.1185 -0.1106 -0.0933
0.1              0.0 -0.1085 -0.1081 -0.0884
0.2              0.0 -0.0818 -0.0731 -0.0508
0.5              0.0 -0.0205 -0.0365 -0.0042
1.0              0.0  0.0277 -0.0335  0.0046
OLS (no prior)   0.0  0.0599 -0.0363  0.0059
Note: Posterior-mean responses under looser and tighter priors. Conclusions that hold across the rows are robust to
↪ the prior.

minnesota  response of GDP growth after 4, 8, 12 quarters: -0.113, -0.104, -0.082
nw         response of GDP growth after 4, 8, 12 quarters: -0.079, -0.074, -0.050
inw        response of GDP growth after 4, 8, 12 quarters: -0.080, -0.079, -0.063
nd         response of GDP growth after 4, 8, 12 quarters: -0.079, -0.079, -0.063
```

How to read it

- **Tightness.** With tight and moderate priors ($\lambda_1 \leq 0.2$) the output response is negative at every horizon. With very loose priors and with OLS, the response at four quarters turns *positive*: without shrinkage the VAR fits noise, and the sign is not robust. This is exactly why a Bayesian VAR is used, and the table is the evidence a referee needs.
- **Prior family.** All four priors give a fall in GDP growth of similar size. Report the Normal-Wishart results and these as robustness.

```
oes.show(r, "Out-of-sample")
```

```
Out-of-sample forecast RMSE (recursive, last 20% of the sample, 38 forecasts)
             Horizon  RMSE gdp_growth  RMSE inflation  RMSE tbilrate
Model
BVAR               1           2.8534          3.0924         0.5883
BVAR               4           3.4712          3.3026         1.8417
OLS VAR            1           2.8274          3.2156         0.5840
OLS VAR            4           3.4003          3.4269         1.7889
Random walk        1           2.9150          3.5139         0.5583
Random walk        4           3.2805          4.1690         1.7917
Note: The BVAR uses the Normal-Wishart posterior mean with the same lambda1; lower RMSE is better. Columns follow the
↪  variable order.
```

How to read it

Recursive one- and four-quarter-ahead forecasts over the last 20% of the sample. The BVAR beats the OLS VAR for inflation at both horizons and is close for the other variables; a random walk is hard to beat for the interest rate at short horizons, as usual. In a forecasting paper, test the differences formally with the Diebold-Mariano test and report several evaluation windows.

42.9 Block exogeneity

When some variables cannot be affected by the others, such as world variables in a model of a small open economy, the Independent Normal-Wishart and Normal-diffuse priors can impose it: the coefficients of the other variables in the block's equations get (almost) zero prior variance.

```
rb = oes.run("bvar", data, dict(spec, prior="inw", block=["gdp_growth"], draws=1000, burn=500))
oes.show(rb, "Equation: gdp_growth", rows=7)
```

```
Equation: gdp_growth
                     Mean   Std. dev.    2.5%      Median   97.5%  P(> 0)
Parameter
const          1.9906e+00  3.6494e-01  1.2866  1.9833e+00  2.6974   1.000
L1.gdp_growth  2.4514e-01  6.7131e-02  0.1095  2.4688e-01  0.3767   1.000
L1.inflation   2.0000e-06  1.4100e-04 -0.0003  5.0000e-06  0.0003   0.510
L1.tbilrate    4.0000e-06  3.9400e-04 -0.0008 -6.0000e-06  0.0007   0.496
L2.gdp_growth  1.1276e-01  5.8372e-02 -0.0044  1.1364e-01  0.2322   0.973
L2.inflation   1.0000e-06  7.1000e-05 -0.0001  1.0000e-06  0.0001   0.511
L2.tbilrate    3.0000e-06  1.9400e-04 -0.0004  1.4000e-05  0.0004   0.523
```

Here GDP growth is (for illustration) made block-exogenous: the lags of inflation and the interest rate in its equation are pushed to zero. In a real application use it for foreign variables, oil prices or anything determined outside the economy you model.

42.10 Write the results

How to write it: results

The marginal likelihood selects four lags and an overall tightness of $\lambda_1 = 0.2$. Figure 1 shows the responses to a one-standard-deviation monetary policy shock, identified recursively with the interest rate ordered last. GDP growth falls by about 0.1 percentage point within a year, and the 68% credible band excludes zero for two years. Inflation initially rises, the price puzzle familiar from small monetary VARs (Sims 1992), before returning to its baseline. Policy shocks account for less than 3% of the forecast-error variance of output growth at all horizons. The output response is negative under all four priors (Minnesota, Normal-Wishart, Independent Normal-Wishart and Normal-diffuse) and for any $\lambda_1 \leq 0.2$; with much looser priors, and with OLS, the response becomes erratic, which illustrates the value of shrinkage in a sample of this length.

Report	What	Where
Data	Sources, sample, transformations, unit-root tests	Section 42.2
Prior	Family, δ_i, λ_1–λ_4, how λ_1 and p were chosen	Sections 42.1 and 42.3
Identification	Ordering and its economic justification	Section 42.5
Results	Impulse responses with 68% (and 90%) bands, variance decomposition	Sections 42.5 and 42.6
Diagnostics	Stability, residual tests, MCMC convergence for Gibbs samplers	Section 42.7
Robustness	Other priors, other tightness, more lags, forecast comparison	Section 42.8

References. Litterman (1986), *Journal of Business and Economic Statistics*; Kadiyala and Karlsson (1997), *Journal of Applied Econometrics*; Sims, Stock and Watson (1990), *Econometrica*; Christiano, Eichenbaum and Evans (1999), *Handbook of Macroeconomics*; Bańbura, Giannone and Reichlin (2010), *Journal of Applied Econometrics*; Giannone, Lenza and Primiceri (2015), *Review of Economics and Statistics*; Dieppe, Legrand and van Roye (2016), *The BEAR toolbox*, ECB Working Paper 1934.

CHAPTER 43

Mean-Adjusted (Steady-State) Bayesian VAR

Applies when. Medium- and long-horizon forecasts, or any VAR where you know something about the long-run means

Overview. Use this chapter when you know something about where the variables go in the long run, such as an inflation target or trend growth, and you want a VAR whose forecasts respect it. A steady-state BVAR puts the prior directly on the long-run means, so you can say what you believe in the units you think in. The chapter covers choosing the prior on the means, estimation, convergence, forecasts, robustness and the write-up.

Data. U.S. quarterly GDP growth, inflation and the 3-month T-bill rate, 1959–2009

Software. Python · OES engine · statsmodels

43.1 The question and the model

Where will growth, inflation and the interest rate be in five years, given what we know about their long-run levels? In an ordinary VAR the long-run mean of the variables is a nonlinear function of all the coefficients, $\mu = (I - A_1 - \cdots - A_p)^{-1}c$, so a prior on the constant says nothing sensible about it. Forecasts converge to a mean the researcher never chose. The **mean-adjusted VAR** of Villani (2009) moves the constant out of the way:

$$y_t - \psi = A_1(y_{t-1} - \psi) + \cdots + A_p(y_{t-p} - \psi) + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \Sigma).$$

Here ψ is the **steady state**, the unconditional mean of y_t, and it gets its own prior: $\psi \sim N(\psi_0, \Omega)$. Forecasts converge to ψ, so the prior disciplines the long horizon directly. The lag coefficients get a Minnesota prior with mean zero (the variables are deviations from the steady state, so they are stationary by construction). Estimation is by Gibbs sampling.

Where does ψ_0 come from? From outside the sample: an inflation target, a long-run growth estimate from a growth model, an interest rate equal to the target plus the neutral real rate. State this in the paper; it is the main modelling choice.

What the model assumes

The VAR is **stationary** around ψ: the steady state exists only if the variables do not have a unit root. Growth rates and inflation are fine. Interest rates and unemployment are borderline in postwar U.S. data, as the unit-root table in Section 43.2 shows. If a series is better described by a random walk, use the ordinary BVAR with a random-walk prior instead, or transform it (differences, deviations from a trend).

How to write it: the model

We estimate a mean-adjusted Bayesian VAR (Villani 2009) in GDP growth, inflation and the three-month Treasury bill rate, $y_t - \psi = \sum_{l=1}^{p} A_l(y_{t-l} - \psi) + \varepsilon_t$. The prior on the steady state ψ is normal with means of 3% (growth), 3.5% (inflation) and 5% (interest rate) and standard deviations of 0.5, 1 and 1 percentage point. The lag coefficients have a Minnesota prior with mean zero and overall tightness $\lambda_1 = 0.1$. The posterior is simulated by Gibbs sampling with 1,000 draws after a burn-in of 500.

43.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
vars_ = ["gdp_growth", "inflation", "tbilrate"]
print(df[vars_].describe().round(2).loc[["mean", "std", "min", "max"]])
spec = dict(vars=vars_, time="date", lags=2, ss_mean="3, 3.5, 5", ss_sd="0.5, 1, 1", draws=1000, burn=500)
r = oes.run("mabvar", data, spec)
oes.show(r, "Unit-root")
```

```
      gdp_growth  inflation  tbilrate
mean        3.10       3.98      5.32
std         3.52       3.25      2.80
min        -8.28      -8.79      0.12
max        15.43      14.62     15.33
Unit-root tests
           ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
gdp_growth                 0.0000               0.0000                      0.0         0.1000       I(0)
inflation                  0.0271               0.0822                      0.0         0.0808       I(0)
tbilrate                   0.2805               0.4462                      0.0         0.0538       I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

Growth rejects a unit root clearly ($p < 0.001$) and inflation at 5% ($p = 0.03$). The T-bill rate does not reject ($p = 0.28$, KPSS $p = 0.05$): over 1959 to 2009 it has a long upward and downward swing, so a stationary model around a fixed mean is an approximation. The largest root of the system will be close to one (Section 43.7). This is the usual compromise for a steady-state VAR; the prior on ψ is what keeps the long-run forecast anchored, and Section 43.8 shows how much the answer depends on it.

43.3 Choose the prior for the steady state

The prior mean and standard deviation of ψ are the model's judgement. The sample means (3.1, 4.0, 5.3) are one data-based guide but a poor prior, since the data would then be used twice. A better guide:

Variable	Prior mean	Prior s.d.	Reason
GDP growth	3	0.5	Trend growth of about 3% a year
Inflation	3.5	1	Between the 1960s low and the 1970s peak, wide enough to leave room for the data
T-bill rate	5	1	A real rate of about 1.5% plus inflation of 3.5%

A **tight** prior (small standard deviation) fixes the steady state at your belief and makes the forecast an implication of it; a **loose** prior lets the data choose and approximately reproduces the sample mean. Section 43.8 compares them.

43.4 Estimate

```
oes.summary(r)
oes.show(r, "Steady states")
oes.show(r, "MCMC convergence diagnostics (steady")
```

```
Observations: 200
Lags: 2
Steady state of gdp_growth: 3.1088
Steady state of inflation: 3.8274
Steady state of tbilrate: 5.0109
```

```
Steady states (posterior)
              Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
gdp_growth  3.1088     0.2720  2.5799  3.1114  3.6565     1.0
inflation   3.8274     0.4394  2.9774  3.8313  4.6842     1.0
tbilrate    5.0109     0.6124  3.7194  5.0283  6.1495     1.0
Note: Prior: gdp_growth ~ N(3, 0.5^2); inflation ~ N(3.5, 1^2); tbilrate ~ N(5, 1^2)

MCMC convergence diagnostics (steady states)
                         Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
steady state: gdp_growth                1000.0                       1.0    -1.517   0.1292
steady state: inflation                 1000.0                       1.0     0.028   0.9779
steady state: tbilrate                  1000.0                       1.0    -0.594   0.5525
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 1.0.
```

How to read it

Each row shows the posterior of one steady state. Growth: 3.11% a year (95% interval 2.58 to 3.66); inflation: 3.83% (2.98 to 4.68); T-bill rate: 5.01% (3.72 to 6.15). Compare with the prior in the note: the posterior standard deviations (0.27, 0.44, 0.61) are about half the prior ones (0.5, 1, 1), so the data are informative about the long-run means, and they move the inflation mean from the prior 3.5 up to 3.83. Where the posterior is close to the prior and the standard deviation has not fallen, the data have said nothing and the prior is your result; say so.

Convergence: the effective sample size equals the number of draws (1000) and Geweke's |z| is below 2 for all three steady states, so the chain has settled and the posterior means are reliable.

43.5 Equations and variance decomposition

```
oes.show(r, "Equation: tbilrate")
oes.show(r, "Forecast-error variance")
```

```
Equation: tbilrate
                Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const          0.3045     0.1301  0.0654  0.3017  0.5787   0.995
L1.gdp_growth  0.0124     0.0104 -0.0083  0.0125  0.0325   0.894
L1.inflation   0.0169     0.0150 -0.0129  0.0175  0.0452   0.866
L1.tbilrate    0.7692     0.0408  0.6883  0.7708  0.8448   1.000
L2.gdp_growth  0.0056     0.0060 -0.0060  0.0057  0.0178   0.813
L2.inflation   0.0078     0.0088 -0.0096  0.0081  0.0253   0.817
L2.tbilrate    0.1403     0.0376  0.0710  0.1407  0.2125   1.000

Forecast-error variance decomposition (posterior median, %)
           Horizon  shock: gdp_growth  shock: inflation  shock: tbilrate
Variable
gdp_growth       1              100.0               0.0              0.0
gdp_growth       4               98.9               0.7              0.3
gdp_growth       8               98.1               1.1              0.7
gdp_growth      12               97.7               1.2              1.0
gdp_growth      20               97.4               1.3              1.2
inflation        1                1.1              98.9              0.0
inflation        4                1.5              96.9              1.5
inflation        8                1.8              94.2              3.6
inflation       12                2.0              92.7              4.9
inflation       20                2.3              91.3              5.9
tbilrate         1                8.9              14.4             76.3
tbilrate         4               11.6              17.7             70.2
tbilrate         8               12.4              19.1             67.9
tbilrate        12               12.6              19.6             67.1
tbilrate        20               12.8              19.9             66.7
Note: Share of each variable's h-step forecast-error variance due to each Cholesky shock.
```

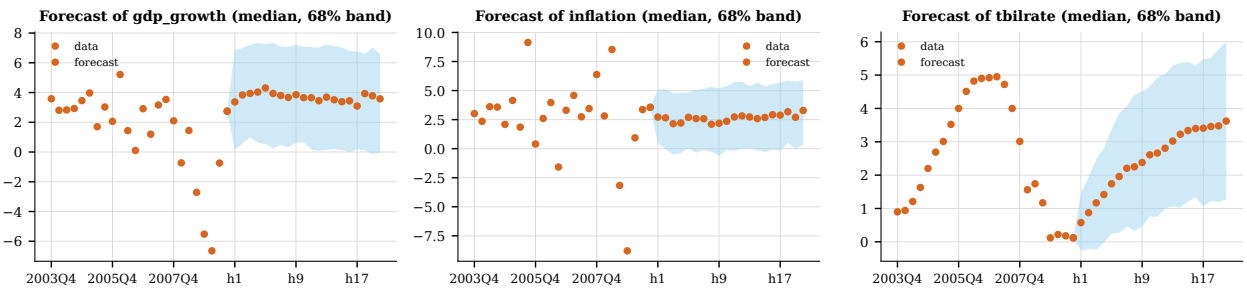


Figure 43.1. Forecasts: what the steady state does. Panels: Forecast of gdp_growth (median, 68% band); Forecast of inflation (median, 68% band); Forecast of tbilrate (median, 68% band).

How to read it

The interest-rate equation is a policy rule in deviations from the steady state: the rate is persistent (own lags 0.77 and 0.14, sum 0.91) and responds positively but weakly to growth and inflation (posterior probabilities of a positive sign 0.81 to 0.89; none of the intervals excludes zero). There is no constant to interpret: the constant of the ordinary VAR is replaced by ψ and the lag coefficients.

The variance decomposition rows show the Cholesky shocks (growth, inflation, T-bill rate). The interest-rate shock explains only 1% of the forecast-error variance of GDP growth and 6% of inflation after five years: policy surprises are a small source of fluctuations, as in Chapter 42.

43.6 Forecasts: what the steady state does

```
oes.plot(r, "Forecast of")
oes.show(r, "Out-of-sample")
```

```
Out-of-sample forecast RMSE (recursive, last 20% of the sample, 38 forecasts)
             Horizon  RMSE gdp_growth  RMSE inflation  RMSE tbilrate
Model
BVAR               1           2.9293          3.0069         0.7048
BVAR               4           3.5116          3.1452         2.0655
OLS VAR            1           2.8233          3.2524         0.5964
OLS VAR            4           3.5315          3.3830         1.8563
Random walk        1           2.9150          3.5139         0.5583
Random walk        4           3.2805          4.1690         1.7917
Note: The BVAR uses the Normal-Wishart posterior mean with the same lambda1; lower RMSE is better. Columns follow the
↪  variable order.
```

How to read it

The charts show the median forecast and a 68% band 20 quarters ahead from 2009Q3, when the T-bill rate was close to zero. The forecast of the T-bill rate rises from 0.6% to 3.6% after five years: it is converging towards the steady state of 5.0%, and the speed is set by the persistence of the system (Section 43.7). Inflation and growth, already close to their steady states, hardly move on the long horizon. This is the purpose of the model: the long-run forecast is an explicit function of the prior and the data.

The out-of-sample table is for the ordinary Normal-Wishart BVAR with the same tightness, **not** for the steady-state model, and it is a useful benchmark in a paper rather than a test of the steady-state prior. For inflation the BVAR has the lowest error at both horizons (3.01 and 3.15, against 3.25 and 3.38 for the OLS VAR and 3.51 and 4.17 for a random walk). For the interest rate a random walk is best, and for GDP growth the OLS VAR is best at one quarter and a random walk at four. In a forecasting paper, evaluate the steady-state model itself against these benchmarks, with Diebold-Mariano tests.

43.7 Diagnostics

```
oes.show(r, "Residual diagnostics")
oes.show(r, "Stability")
oes.show(r, "MCMC convergence diagnostics (coeff", rows=6)
```

```
Residual diagnostics (at the posterior mean)
```

```
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   219.7585   0.0000  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth             14.2531   0.2848                                                        H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           17.0965   0.0002                                                          H0:
↪ normal residuals
ARCH LM, gdp_growth                7.2157   0.1249                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation              48.7477   0.0000                                                        H0: no
↪ autocorrelation
Jarque-Bera, inflation            71.7802   0.0000                                                          H0:
↪ normal residuals
ARCH LM, inflation                23.0384   0.0001                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               44.1931   0.0000                                                        H0: no
↪ autocorrelation
Jarque-Bera, tbilrate           1133.4775   0.0000                                                          H0:
↪ normal residuals
ARCH LM, tbilrate                 65.3232   0.0000                                          H0: no conditional
↪ heteroskedasticity

Stability
                                                                    Value
Item
Share of posterior draws with all companion roots inside the unit circle  1.000
Largest root modulus at the posterior mean                                0.928
Note: Unit roots in levels data make many draws non-stationary; that is expected with a random-walk prior and is not
↪ a failure.

MCMC convergence diagnostics (coefficients)
                         Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
gdp_growth: const                       1000.0                     1.000    -2.371   0.0177
inflation: const                         796.0                     0.796    -0.770   0.4412
tbilrate: const                          843.0                     0.843    -0.747   0.4551
gdp_growth: L1.gdp_growth               1000.0                     1.000     1.182   0.2370
tbilrate: L1.gdp_growth                 1000.0                     1.000    -0.796   0.4261
inflation: L1.inflation                  878.0                     0.878    -0.217   0.8285
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.796.
```

How to read it

Residuals are not white noise: Ljung-Box rejects for inflation and the interest rate ($p < 0.001$) but not for GDP growth ($p = 0.28$); normality is rejected for all three, extremely for the interest rate (Jarque-Bera 1133, from the 1979 to 1982 period), and ARCH is present in inflation and the interest rate. These are the usual features of postwar U.S. data; Chapter 47 models the changing volatility. Note them as limitations and check that the main results do not change with four lags (Section 43.8).

Stability: every posterior draw is stationary, and the largest root at the posterior mean is 0.93, which is high: deviations from the steady state take about nine quarters to halve ($\ln 0.5 / \ln 0.928$). The steady state is therefore a slow attractor, and a five-year forecast does not reach it. The coefficient chains mix well (efficiency of 0.80 or above); one Geweke statistic is above 1.96 in absolute value (the constant of the growth equation, -2.4), which is expected by chance among many parameters; raise the burn-in to confirm that nothing changes.

43.8 Robustness: the prior on the steady state

```python
def last(res, var, H=20):
    c = next(c for c in res["charts"] if c["title"].startswith(f"Forecast of {var}"))
    v = [x for x in c["series"][-1]["y"] if x is not None][-H:]
    return v[0], v[-1]
```

```
no_ss = {k: v for k, v in spec.items() if not k.startswith("ss_")}          # blank = sample means and s.d. as
  ↪ the prior
priors = {"informative (guide)": dict(ss_mean="3, 3.5, 5", ss_sd="0.5, 1, 1"),
          "sample mean and s.d.": dict(),
          "diffuse": dict(ss_mean="3, 3.5, 5", ss_sd="10, 10, 10"),
          "low-rate beliefs": dict(ss_mean="3, 2, 3.5", ss_sd="0.5, 0.25, 0.5")}
rows = []
for name, pr in priors.items():
    rr = oes.run("mabvar", data, dict(no_ss, **pr))
    ss = oes.frame(next(t for t in rr["tables"] if t["title"].startswith("Steady states")))
    rows.append([name] + [round(ss["Mean"][v], 2) for v in vars_] + [round(last(rr, "inflation")[1], 2), round
  ↪ (last(rr, "tbilrate")[1], 2)])
print(pd.DataFrame(rows, columns=["prior", "psi growth", "psi inflation", "psi T-bill", "inflation, 5 yrs", "T-
  ↪ bill, 5 yrs"]).set_index("prior"))

rb = oes.run("bvar", data, dict(vars=vars_, time="date", lags=2, prior="nw", lambda1=0.2, delta="0, 0, 1",
  ↪ fhorizon=20))
print("ordinary BVAR, 5 years ahead: inflation %.2f, T-bill %.2f" % (last(rb, "inflation")[1], last(rb, "
  ↪ tbilrate")[1]))
for p in (1, 2, 4):
    s = oes.run("mabvar", data, dict(spec, lags=p))["summary"]
    print(f"{p} lag(s): steady states", [round(x["value"], 2) for x in s[2:]])
```

```
                    psi growth  psi inflation  psi T-bill  inflation, 5 yrs  T-bill, 5 yrs
prior
informative (guide)        3.11           3.83        5.01              3.29           3.62
sample mean and s.d.       3.13           3.91        5.11              3.26           3.72
diffuse                    3.14           3.88        5.06              3.22           3.67
low-rate beliefs           3.31           2.27        3.22              1.48           2.00
ordinary BVAR, 5 years ahead: inflation 2.92, T-bill 3.14
1 lag(s): steady states [3.07, 3.82, 5.06]
2 lag(s): steady states [3.11, 3.83, 5.01]
4 lag(s): steady states [3.1, 3.83, 5.0]
```

How to read it

The table re-estimates the model with four priors on the steady state and reports the posterior steady states and the five-year forecasts.

- **Data-based or loose priors** (sample means and standard deviations, the chapter's prior, a nearly flat prior with standard deviation 10): the steady states are the same within 0.1 point (growth 3.1, inflation 3.8 to 3.9, T-bill 5.0 to 5.1) and so are the five-year forecasts (inflation 3.2 to 3.3, T-bill 3.6 to 3.7). The data decide.
- **A tight prior with different beliefs** (inflation 2 ± 0.25, T-bill 3.5 ± 0.5): the posterior inflation mean is 2.27 and the T-bill mean 3.22, although the sample averages are 4.0 and 5.3, and the five-year forecasts fall to 1.5% and 2.0%. A tight prior wins over 200 observations. That is its point: it is an assumption, and the table shows exactly how much the answer depends on it.
- **Ordinary BVAR**, with no prior on the long-run mean: 2.9% and 3.1%, between the two. Its long-run mean is whatever the lags imply, which is why it is a poor tool for a long-horizon question.
- **Lags** (1, 2, 4): the steady states change by less than 0.1.

The rule for a paper: report the posterior steady states under a data-based prior and under your preferred prior, and discuss any difference as a statement about beliefs, not as an empirical finding.

43.9 Write the results

How to write it: results

The posterior mean of the steady state is 3.1% for GDP growth (95% credible interval 2.6 to 3.7), 3.8% for inflation (3.0 to 4.7) and 5.0% for the Treasury bill rate (3.7 to 6.1); the posterior standard deviations are about half the prior ones, so the data are informative about the long-run means. Starting from the low interest rate of

2009Q3, the median forecast of the bill rate rises to 3.6% after five years, converging slowly towards its steady state because the largest root of the system is 0.93. These results hardly change when the sample-mean prior or a nearly flat prior replaces our prior, or when the lag length varies between one and four. They depend strongly on the prior when it is tight: a prior of 2 ± 0.25% for inflation and 3.5 ± 0.5% for the interest rate lowers the five-year forecasts to 1.5% and 2.0%. Residual tests show serial correlation, non-normality and conditional heteroskedasticity in inflation and the interest rate, so the credible bands should be read as approximate.

Report	What	Where
Prior on ψ	Means, standard deviations and where they come from	Sections 43.1 and 43.3
Data	Sample, stationarity of the series	Section 43.2
Steady states	Posterior mean, interval, and prior versus posterior	Section 43.4
Forecasts	Fan charts, out-of-sample accuracy against the OLS VAR and a random walk	Section 43.6
Diagnostics	Convergence, residual tests, stability	Section 43.7
Robustness	Other priors on ψ, lags, comparison with the ordinary BVAR	Section 43.8

References. Villani (2009), *Journal of Applied Econometrics*; Österholm (2008), *Journal of Forecasting*; Jarociński and Maćkowiak (2018), *Journal of Applied Econometrics*; Litterman (1986), *Journal of Business and Economic Statistics*; Clark (2011), *Journal of Business & Economic Statistics*.

CHAPTER 44

Factor-Augmented VAR (FAVAR)

Applies when. A VAR that should use information from many macro series without adding them all as variables
Overview. Use this chapter when a small VAR omits information that the policymaker or the economy uses, and you have a wide panel of related series. A FAVAR summarises the panel in a few factors, adds them to the VAR, and traces the effect of a shock on every series in the panel. The chapter covers the factor choice, estimation, impulse responses of the whole panel, diagnostics, robustness and the write-up, and is honest about what a small information set can and cannot do.
Data. U.S. quarterly macro data, 1959–2009: inflation, unemployment and the T-bill rate, with six real-activity series as the information set
Software. Python · OES engine · statsmodels

44.1 The question and the model

Does a small monetary VAR omit information that matters? A VAR in inflation, unemployment and the interest rate has the price puzzle: inflation rises after a tightening. One explanation is that the central bank sees far more than three series, and the VAR leaves that information in the residual. A **factor-augmented VAR** (Bernanke, Boivin and Eliasz 2005) adds a few unobserved factors F_t that summarise a large panel X_t of N informational series:

$$X_t = \Lambda^F F_t + \Lambda^Y Y_t + e_t, \qquad \begin{pmatrix} F_t \\ Y_t \end{pmatrix} = c + A_1 \begin{pmatrix} F_{t-1} \\ Y_{t-1} \end{pmatrix} + \cdots + u_t,$$

where Y_t are the observed variables (here inflation, unemployment and the policy rate, ordered last). Two things follow from the first equation. The factors enter the VAR, so the policy shock is identified with more information in the system. And every series in X_t has an impulse response, $\Lambda' \times$ the VAR response, so you can report the effect of the shock on dozens of series without estimating a dozen VARs.

OES implements the **two-step** estimator: the factors are the principal components of the standardised panel, and the VAR on (F_t, Y_t) is a Bayesian VAR with the Minnesota-type priors of Chapter 42, with prior mean zero on the factors.

What this example is, and is not

The example panel has only **six** series, all real-activity growth rates from the same national accounts. A real FAVAR uses 50 to 150 series (the McCracken-Ng FRED-QD data are the standard source) with prices, interest rates, employment, financial and monetary variables. With six series the factors can say little that the three observed variables do not already say, and the chapter shows this honestly in Section 44.8. Use this chapter for the mechanics and diagnostics, and bring a large panel to your own application. The OES page accepts any number of series.

How to write it: the model

We estimate a factor-augmented VAR (Bernanke, Boivin and Eliasz 2005). The informational panel, N quarterly series standardised to mean zero and variance one, is summarised by K principal components. The factors, together with inflation, unemployment and the Treasury bill rate, enter a VAR(p) estimated with a Normal-Wishart prior with Minnesota moments (Litterman 1986), with prior mean zero for the factors and for stationary variables and one for the variables with a unit root. The policy shock is the innovation to the Treasury bill rate,

identified recursively with the rate ordered last. Responses of the panel series are obtained from their estimated loadings on the factors and the observed variables.

44.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
y = ["inflation", "unemp", "tbilrate"]                      # observed variables, policy rate last
x = ["gdp_growth", "cons_growth", "inv_growth", "gov_growth", "income_growth", "m1_growth"]   # information
  ↪ set
spec = dict(y=y, x=x, time="date", factors=2, lags=4, delta="0, 1, 1")
r = oes.run("favar", data, spec)
oes.summary(r)
oes.show(r, "Unit-root")
```

```
Observations: 198
Factors: 2
Informational series: 6
Variance explained: 0.605
Prior: Normal-Wishart
Unit-root tests
              ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
inflation                     0.0271              0.0822                   0.0000         0.0808       I(0)
unemp                         0.1107              0.3084                   0.0008         0.1000       I(1)
tbilrate                      0.2805              0.4462                   0.0000         0.0538       I(1)
gdp_growth                    0.0000              0.0000                   0.0000         0.1000       I(0)
cons_growth                   0.0000              0.0001                   0.0000         0.1000       I(0)
inv_growth                    0.0000              0.0000                   0.0000         0.1000       I(0)
gov_growth                    0.0006              0.0043                   0.0000         0.1000       I(0)
income_growth                 0.0000              0.0000                   0.0000         0.0298       I(0)
m1_growth                     0.0023              0.0122                   0.0000         0.1000       I(0)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

The principal components do not need stationary data, but they behave well when the panel is stationary, which is why informational series are transformed to growth rates or differences. All six series reject a unit root. Of the observed variables, inflation rejects (ADF p = 0.03), but unemployment and the T-bill rate do not, so they get a random-walk prior (`delta = "0, 1, 1"`). The factors are standardised and stationary, so they get zero.

44.3 Choose the number of factors

```
oes.show(r, "Number of factors")
```

```
Number of factors: Bai and Ng (2002) IC_p2
         IC_p2 (lower is better)  Cumulative variance share
Factors
1                        -0.2380                       0.418
2                        -0.3193                       0.605
3                        -0.5670                       0.773
4                        -1.0260                       0.895
5                        -3.1155                       0.990
Note: The criterion prefers 5 factor(s); the model uses 2.
```

How to read it

Two tools: the cumulative share of variance, and the **Bai and Ng (2002)** information criterion, which finds the number of factors in a large panel. It needs a large N and T; with $N = 6$ it is not informative (it keeps falling up to the maximum of five factors, which is nearly the whole panel). In a real application, $N \geq 50$ and you would choose with the criterion and a scree plot, then check that the results do not depend on the choice (Section 44.8). Here two factors explain 61% of the variance of the six series, which is what the model uses.

44.4 The factors and the VAR

```
oes.show(r, "Principal-component")
oes.show(r, "Equation: tbilrate")
```

```
Principal-component factors
        Share of variance  Cumulative
Factor
F1                 0.4176      0.4176
F2                 0.1876      0.6052

Equation: tbilrate
                Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const         0.1629     0.2757 -0.3705  0.1553  0.7032   0.709
L1.F1        -0.0337     0.0375 -0.1047 -0.0335  0.0397   0.180
L1.F2        -0.0545     0.0497 -0.1537 -0.0538  0.0403   0.135
L1.inflation  0.0073     0.0205 -0.0321  0.0075  0.0482   0.637
L1.unemp     -0.1527     0.1443 -0.4243 -0.1558  0.1541   0.141
L1.tbilrate   0.9426     0.0494  0.8482  0.9432  1.0385   1.000
L2.F1        -0.0068     0.0247 -0.0565 -0.0060  0.0410   0.389
L2.F2        -0.0138     0.0350 -0.0848 -0.0136  0.0545   0.356
L2.inflation  0.0183     0.0146 -0.0094  0.0183  0.0457   0.894
L2.unemp      0.0711     0.1439 -0.2212  0.0795  0.3531   0.687
L2.tbilrate  -0.0272     0.0414 -0.1064 -0.0274  0.0562   0.248
L3.F1        -0.0042     0.0176 -0.0379 -0.0040  0.0300   0.403
L3.F2        -0.0037     0.0263 -0.0545 -0.0033  0.0462   0.446
L3.inflation  0.0148     0.0113 -0.0076  0.0149  0.0374   0.912
L3.unemp      0.0230     0.0993 -0.1714  0.0227  0.2164   0.608
L3.tbilrate   0.0262     0.0301 -0.0333  0.0263  0.0847   0.817
L4.F1        -0.0011     0.0143 -0.0295 -0.0004  0.0250   0.493
L4.F2        -0.0059     0.0203 -0.0446 -0.0058  0.0314   0.389
L4.inflation  0.0013     0.0089 -0.0152  0.0013  0.0185   0.556
L4.unemp      0.0660     0.0703 -0.0662  0.0675  0.2112   0.827
L4.tbilrate  -0.0132     0.0235 -0.0591 -0.0131  0.0326   0.286
```

How to read it

The first factor explains 42% of the panel: the common real-activity cycle. The second adds 19%. The interest-rate equation is a policy rule: the rate is persistent and responds to inflation, unemployment and the real-activity factors; the question for a FAVAR is whether the *factors* add to the rule beyond the three observed variables.

44.5 The policy shock and the whole panel

```
oes.plot(r, "Response of inflation to a shock in tbilrate")
oes.plot(r, "Response of unemp to a shock in tbilrate")
oes.show(r, "Responses of the informational")
oes.plot(r, "Response of gdp_growth to a shock in tbilrate")
oes.plot(r, "Response of inv_growth to a shock in tbilrate")
```

```
Responses of the informational series to a shock in tbilrate (median)
               Impact   h = 4  h = 12  R-squared of loadings
Series
gdp_growth    -0.0252 -0.1535 -0.0423                  0.912
cons_growth    0.0021 -0.1146 -0.0316                  0.572
inv_growth    -0.0609 -0.5810 -0.0738                  0.631
gov_growth     0.0959  0.0602  0.0853                  0.699
income_growth  0.0464 -0.0901 -0.0246                  0.439
m1_growth     -0.0607 -0.0128  0.1727                  0.488
```

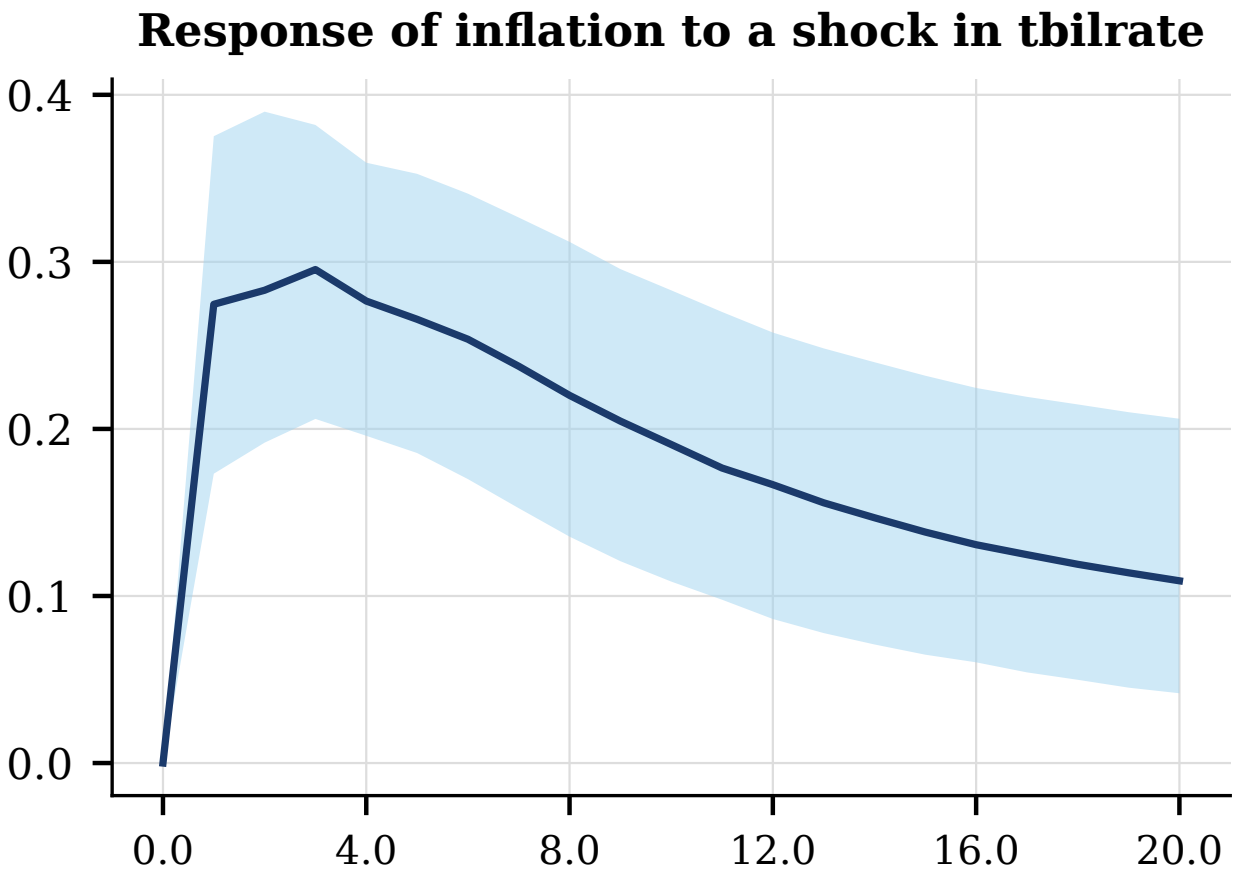


Figure 44.1. The policy shock and the whole panel: Response of inflation to a shock in tbilrate.

How to read it

Observed variables first. A one-standard-deviation rise in the T-bill rate (about 0.6 percentage point) raises unemployment gradually, by 0.10 percentage point after two years and 0.14 after three, with the 68% band above zero from the second year. This is the expected sign and size. **Inflation, however, rises** on impact, by 0.28 percentage point, with a band that excludes zero for more than a year: the price puzzle is still there. This is a result, not an error: six real-activity series carry no information about prices that the VAR does not already have. Section 44.8 shows that the response is the same with and without the factors.

The advantage of a FAVAR shows in the table and the last two charts: the same shock traced through all six panel series, with no extra VARs. Output growth falls by about 0.15 percentage point after one year (68% band from −0.23 to −0.09) and returns to zero after about three years; consumption and income growth fall by 0.12 and 0.09; investment growth falls by 0.58, four times more than output, with a wide band, as in most monetary VARs. Government spending growth hardly moves. The R^2 column tells you how well the factors and observed variables explain each series (0.91 for GDP growth, 0.44 for income growth): the response of a series with a low R^2 is mostly about the common component, so treat it with care.

44.6 Variance decomposition

```
oes.show(r, "Forecast-error variance")
```

```
Forecast-error variance decomposition (posterior median, %)
          Horizon  shock: F1  shock: F2  shock: inflation  shock: unemp  shock: tbilrate
Variable
F1              1      100.0        0.0              0.0           0.0              0.0
F1              4       95.8        0.8              1.8           0.4              0.6
F1              8       91.6        0.8              3.3           1.9              1.6
F1             12       90.2        0.9              3.5           2.8              1.9
F1             20       89.6        0.9              3.6           3.0              2.0
```

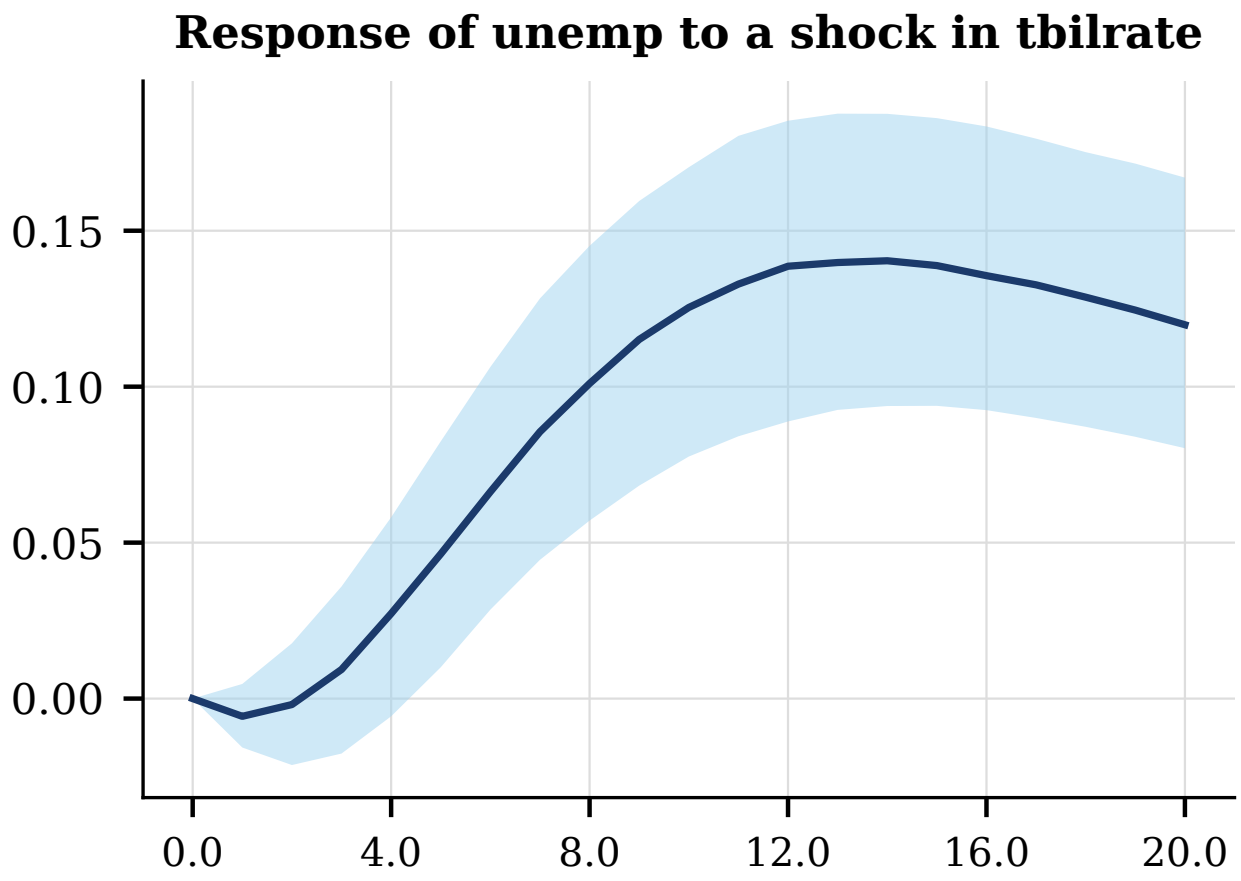


Figure 44.2. The policy shock and the whole panel: Response of unemp to a shock in tbilrate.

F2	1	0.2	99.8	0.0	0.0	0.0
F2	4	2.7	95.1	0.5	0.9	0.3
F2	8	4.5	92.2	0.7	1.7	0.5
F2	12	4.9	91.1	0.9	1.9	0.8
F2	20	5.1	89.6	1.4	1.9	1.4
inflation	1	0.8	1.1	97.7	0.0	0.0
inflation	4	2.7	1.4	91.2	0.4	3.2
inflation	8	5.1	1.5	85.3	1.1	6.1
inflation	12	6.2	1.6	82.1	1.4	7.5
inflation	20	6.8	1.5	79.8	1.5	8.9
unemp	1	36.8	0.2	1.0	61.3	0.0
unemp	4	58.0	0.4	0.6	40.1	0.1
unemp	8	60.9	0.5	1.9	33.4	1.5
unemp	12	56.0	0.6	5.4	29.7	5.8
unemp	20	46.9	0.8	11.0	24.6	14.4
tbilrate	1	13.1	4.0	10.9	4.3	66.6
tbilrate	4	19.6	2.1	13.6	5.6	57.8
tbilrate	8	21.8	1.8	16.2	5.4	53.0
tbilrate	12	21.1	1.7	17.7	5.0	52.3
tbilrate	20	19.2	1.6	19.2	4.6	52.6

Note: Share of each variable's h-step forecast-error variance due to each Cholesky shock.

How to read it

Rows are the variables, columns the shocks (Cholesky). The policy shock explains 9% of the forecast-error variance of inflation after five years, 14% of unemployment and 2% of the real-activity factor; the interest rate's own shock accounts for 53% of its variance. The first factor, the real cycle, is the main driver of unemployment (47% to 61%). The policy shock is a modest source of fluctuations, the standard result; policy matters through the rule, not through surprises.

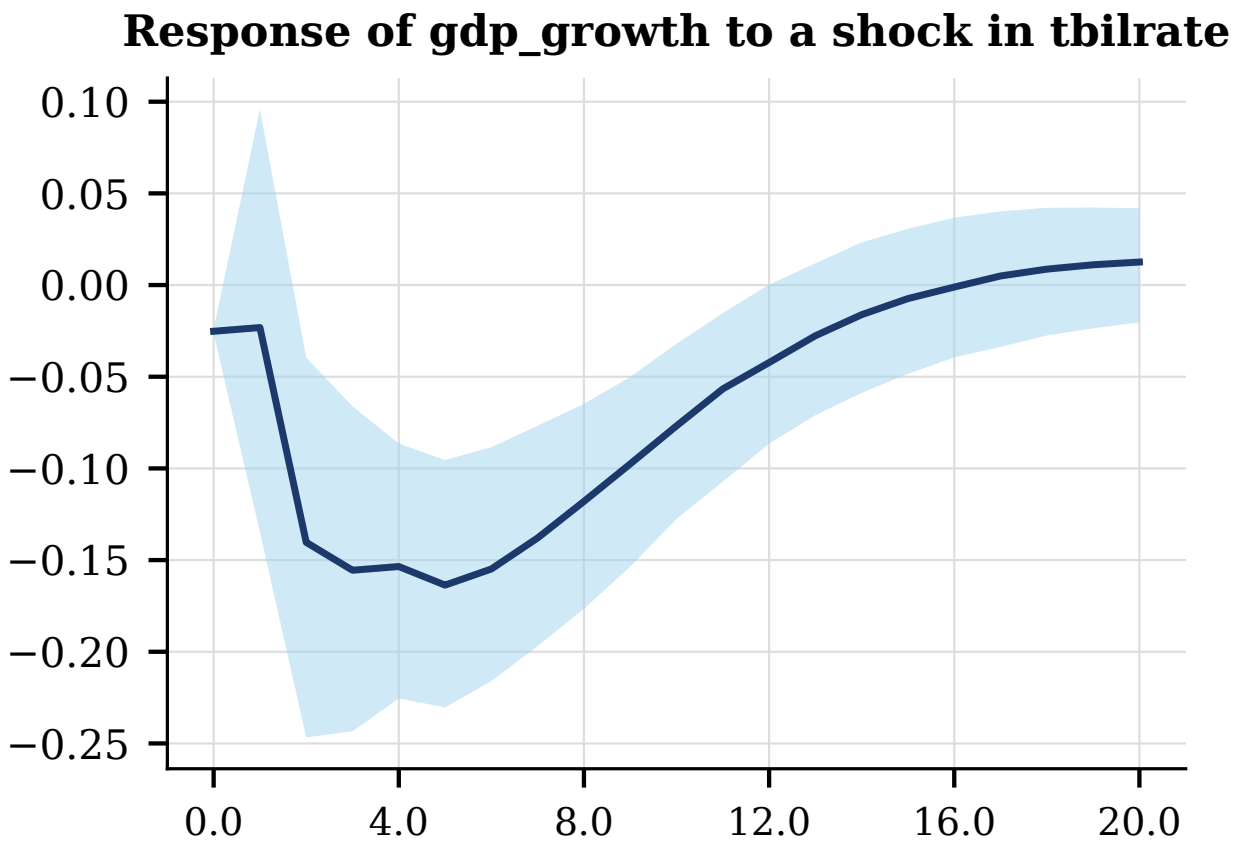


Figure 44.3. The policy shock and the whole panel: Response of gdp_growth to a shock in tbilrate.

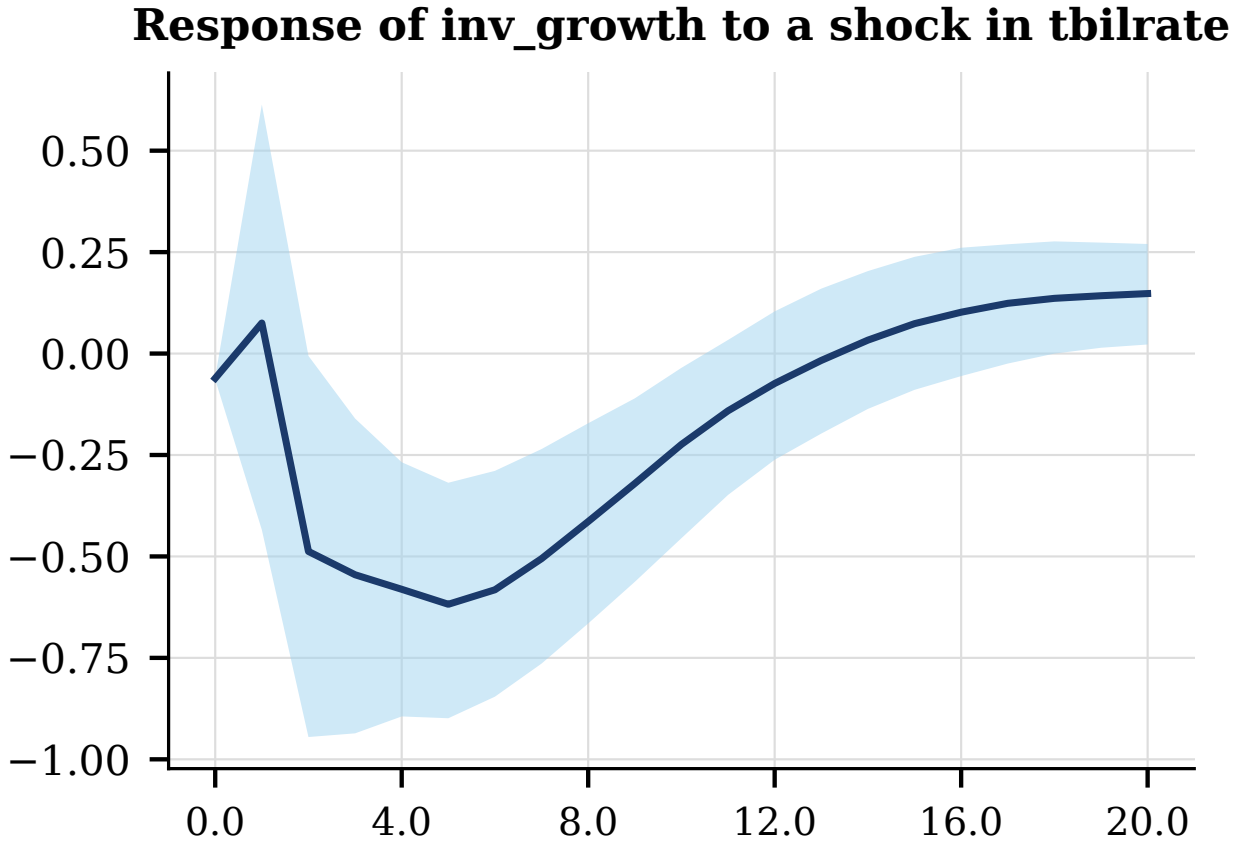


Figure 44.4. The policy shock and the whole panel: Response of inv_growth to a shock in tbilrate.

44.7 Diagnostics

```
oes.show(r, "Residual diagnostics")
```

```
Residual diagnostics of the FAVAR (at the posterior mean)
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   405.6872   0.0000  df = 300; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, F1                     18.0123   0.1153                                                          H0: no
↪ autocorrelation
Jarque-Bera, F1                   23.8743   0.0000                                                            H0:
↪ normal residuals
ARCH LM, F1                        8.7192   0.0685                                              H0: no conditional
↪ heteroskedasticity
Ljung-Box, F2                     20.9253   0.0515                                                          H0: no
↪ autocorrelation
Jarque-Bera, F2                    2.0738   0.3546                                                            H0:
↪ normal residuals
ARCH LM, F2                        1.3027   0.8609                                              H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation              36.7859   0.0002                                                          H0: no
↪ autocorrelation
Jarque-Bera, inflation            92.7501   0.0000                                                            H0:
↪ normal residuals
ARCH LM, inflation                21.5160   0.0003                                              H0: no conditional
↪ heteroskedasticity
Ljung-Box, unemp                  31.2963   0.0018                                                          H0: no
↪ autocorrelation
Jarque-Bera, unemp                96.7007   0.0000                                                            H0:
↪ normal residuals
ARCH LM, unemp                    19.5490   0.0006                                              H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               42.2316   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, tbilrate           1875.2330   0.0000                                                            H0:
↪ normal residuals
ARCH LM, tbilrate                 57.0051   0.0000                                              H0: no conditional
↪ heteroskedasticity
```

How to read it

The factors are fairly well behaved (Ljung-Box does not reject at 5% for either factor), but the observed variables have serial correlation in the residuals (Ljung-Box p below 0.002), non-normal residuals and ARCH. The interest rate is the extreme case (Jarque-Bera 1875): the largest quarterly changes in the T-bill rate, up to 5.9 percentage points, are all in 1980 to 1982, the Volcker disinflation. Adding more lags helps only a little (the portmanteau rejects with two, four and six lags, p = 0.0002 with six), and the joint test has 300 degrees of freedom with 198 observations, so it over-rejects in samples this short; rely on the individual tests. The practical answer is the stochastic-volatility VAR (see Chapter 47), which allows the shock variance to change; in the paper, state the limitation and show that the main results do not depend on the lag length (Section 44.8).

44.8 Robustness: factors, lags and the plain VAR

```
def irf(res, title, h=(1, 4, 8, 12)):
    c = next(c for c in res["charts"] if c["title"] == title)["series"][0]["y"]
    return [round(c[i], 3) for i in h]

title = "Response of inflation to a shock in tbilrate"
rows = {}
for k in (1, 2, 3, 4):
    rows[f"{k} factor(s), 4 lags"] = irf(oes.run("favar", data, dict(spec, factors=k)), title)
for p in (2, 6):
    rows[f"2 factors, {p} lags"] = irf(oes.run("favar", data, dict(spec, lags=p)), title)
rb = oes.run("bvar", data, dict(vars=y, time="date", lags=4, prior="nw", lambda1=0.1, delta="0, 1, 1"))
```

```
rows["no factors (BVAR)"] = irf(rb, title)
print(pd.DataFrame(rows, index=["h=1", "h=4", "h=8", "h=12"]).T)
```

```
                      h=1    h=4    h=8   h=12
1 factor(s), 4 lags  0.281  0.276  0.219  0.160
2 factor(s), 4 lags  0.275  0.277  0.220  0.167
3 factor(s), 4 lags  0.274  0.264  0.218  0.162
4 factor(s), 4 lags  0.257  0.262  0.212  0.160
2 factors, 2 lags    0.267  0.318  0.265  0.207
2 factors, 6 lags    0.284  0.312  0.196  0.131
no factors (BVAR)    0.279  0.286  0.236  0.178
```

How to read it

The response of inflation to a policy shock, at 1, 4, 8 and 12 quarters, under different choices. It is stable across the number of factors (0.26 to 0.28 at one quarter, 0.16 to 0.17 at three years) and lags, and the same as the BVAR without factors (0.28, 0.29, 0.24, 0.18). Shorter or longer lag lengths move the response at one year by about 0.04, never the sign. Two conclusions for a paper: (1) the FAVAR results are robust to K and p; (2) with this *small* information set the factors do not change the answer. When the factors do not matter, say so; in a real application with 100 series, this comparison is the main test that the FAVAR is worth the extra complexity.

44.9 Write the results

How to write it: results

Two factors, which explain 61% of the variance of the six informational series, are added to a VAR(4) in inflation, unemployment and the Treasury bill rate. A one-standard-deviation shock to the policy rate (about 0.6 percentage point) raises unemployment gradually, by 0.10 percentage point after two years, with a 68% credible band above zero from the second year. Output growth falls by 0.15 percentage point after one year and returns to its baseline after about three years; investment growth falls by 0.58, with a wide band. Inflation rises by 0.28 percentage point on impact and stays above zero for about three years, a price puzzle that is unchanged when the number of factors varies between one and four or the lags between two and six, and that is the same as in the VAR without factors. We conclude that the six real-activity series add no information about prices; a larger panel with price and commodity series is needed to test whether omitted information explains the puzzle. Policy shocks explain 9% of the forecast-error variance of inflation and 14% of that of unemployment after five years.

Report	What	Where
Data	Panel (source, N, transformations), observed variables, sample	Section 44.2
Factors	Number and how chosen (Bai-Ng, variance share), R^2 of the loadings	Sections 44.3 and 44.5
Prior	Family, λ_1, prior means of the own first lags	Sections 44.1 and 44.2
Identification	Ordering of the observed variables, the policy rate last	Section 44.5
Results	Responses of the observed variables and of selected panel series with 68% bands, variance decomposition	Sections 44.5 and 44.6
Diagnostics	Residual tests	Section 44.7
Robustness	Number of factors, lags, comparison with the VAR without factors	Section 44.8

References. Bernanke, Boivin and Eliasz (2005), *Quarterly Journal of Economics*; Stock and Watson (2002), *Journal of the American Statistical Association*; Bai and Ng (2002), *Econometrica*; Sims (1992), *European Economic Review*; Litterman (1986), *Journal of Business and Economic Statistics*; McCracken and Ng (2020), *FRED-QD: A Quarterly Database for Macroeconomic Research.*

CHAPTER 45

Mixed-Frequency BVAR: Nowcasting Quarterly GDP with Monthly Data

Applies when. Quarterly GDP and monthly indicators that arrive earlier, for nowcasting or for a model at the monthly frequency

Overview. Use this chapter when some variables are observed monthly and others quarterly, and you want to estimate the VAR at the monthly frequency, to infer the monthly path of GDP, or to nowcast the current quarter before it is published. The model treats monthly GDP growth as an unobserved series that must aggregate to the published quarterly number. The chapter covers the data layout, the model, convergence (this model needs long chains), a pseudo real-time nowcast test against simple benchmarks, robustness and the write-up.

Data. U.S. monthly industrial production growth and federal funds rate, and quarterly GDP growth, 1985–2019 (the example file shipped with OES)

Software. Python · OES engine · statsmodels

45.1 The question and the model

What is GDP growth this quarter, before the statistical office has published it? Industrial production and interest rates are published monthly, within weeks, but GDP only quarterly, after a month or more. A **mixed-frequency VAR** estimates the monthly dynamics of all variables jointly, including a *latent* monthly GDP growth series y_t^*, and links it to the published quarterly figure y^Q:

$$z_t = c + A_1 z_{t-1} + \cdots + A_p z_{t-p} + u_t, \qquad z_t = (\text{IP growth}_t, \text{fed funds}_t, y_t^*)', \qquad y_t^Q = \tfrac{1}{9}\left(y_t^* + 2y_{t-1}^* + 3y_{t-2}^* + 2y_{t-3}^* + y_{t-4}^*\right)$$

in the last month t of each quarter, and unobserved in the other months. The triangle weights $(1, 2, 3, 2, 1)/9$ of Mariano and Murasawa (2003) express the fact that quarterly *growth* is a weighted average of monthly growth rates; for variables in levels, the alternative is a three-month average. The VAR has a Normal-Wishart prior with Minnesota-type moments, and the posterior is simulated by a Gibbs sampler that alternates between the VAR parameters and the latent monthly path, which is drawn with the Carter-Kohn simulation smoother (Schorfheide and Song 2015).

What you get

Three outputs: (1) a **monthly estimate of GDP growth** with credible bands, which the data never contain; (2) a **nowcast** of the current quarter, updated each month as new monthly data arrive (the "Nowcasts" table gives the implied quarterly value after each month of the quarter); (3) VAR coefficients and impulse responses at the monthly frequency.

The latent monthly series is poorly identified from quarterly data alone: the bands are wide (Section 45.4). The model does not create information; it uses the monthly indicators to fill in the months between quarterly releases.

How to write it: the model

We estimate a mixed-frequency Bayesian VAR (Schorfheide and Song 2015) with industrial production growth, the federal funds rate and quarterly GDP growth, with five monthly lags. GDP growth is modelled as an unobserved monthly series whose triangle-weighted sum over five months (Mariano and Murasawa 2003) equals the published quarterly growth rate in the last month of each quarter. The VAR has a Normal-Wishart prior

with Minnesota moments (overall tightness 0.2, prior mean of the first lags zero). The posterior is simulated with a Gibbs sampler, drawing the latent months with the Carter-Kohn algorithm; we use 3,000 draws after 2,000 burn-in.

45.2 Prepare the data

The file has one row per month. Put each quarterly value in the row of the quarter's **last** month and leave the other months of the quarter blank.

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("mixed_frequency.csv").read()
df = pd.read_csv("mixed_frequency.csv")
print(df.iloc[:7].to_string(index=False))
print(df.notna().sum().rename("non-missing values").to_frame().T.to_string(index=False))
print(df.describe().round(2).loc[["mean", "std", "min", "max"]])
spec = dict(monthly=["ip_growth", "fedfunds"], quarterly=["gdp_growth"], time="date", lags=5, draws=3000, burn
  ↪ =2000)
r = oes.run("mfbvar", data, spec)           # about two minutes on a laptop
oes.show(r, "Unit-root")
```

```
   date  ip_growth  fedfunds  gdp_growth
1985-02      3.516      8.50         NaN
1985-03      1.752      8.58      3.8572
1985-04     -2.640      8.27         NaN
1985-05      2.040      7.97         NaN
1985-06     -0.636      7.53      3.5066
1985-07     -6.036      7.88         NaN
1985-08      3.876      7.90         NaN
 date  ip_growth  fedfunds  gdp_growth
  419        419       419         139
      ip_growth  fedfunds  gdp_growth
mean       1.79      3.57        2.64
std        7.30      2.78        2.26
min      -53.82      0.07       -8.85
max       24.47      9.85        7.22
Unit-root tests (observed data)
            Frequency  Observations  ADF p-value (level)  ADF p-value (difference)
Variable
ip_growth     monthly           419               0.0000                    0.0000
fedfunds      monthly           419               0.1052                    0.0007
gdp_growth  quarterly           139               0.0000                    0.0000
Note: Use growth rates (triangle aggregation) for I(1) variables such as GDP, or levels with the 3-month average.
```

How to read it

419 months from 1985-02 to 2019-12, 139 quarterly observations (GDP growth appears in March, June, September and December only). Industrial production growth is annualised monthly growth and very volatile (standard deviation 7.3, range −54 to 24). Quarterly GDP growth averages 2.6% with a standard deviation of 2.3.

Unit roots: IP growth and GDP growth reject ($p < 0.001$); the federal funds rate does not ($p = 0.11$). Use growth rates for I(1) variables such as GDP (triangle weights) and levels for stationary ones (three-month average). The rate enters in levels, which is common in monthly VARs; the prior mean of its own lag is zero by default, which pulls it towards a constant: consider `delta` close to one if the persistence of the rate matters for your question.

45.3 Estimate and the monthly path of GDP

```
oes.summary(r)
oes.plot(r, "Monthly estimate", ncols=1, size=(7, 3.2))
```

```
Months: 419
Monthly variables: 2
```

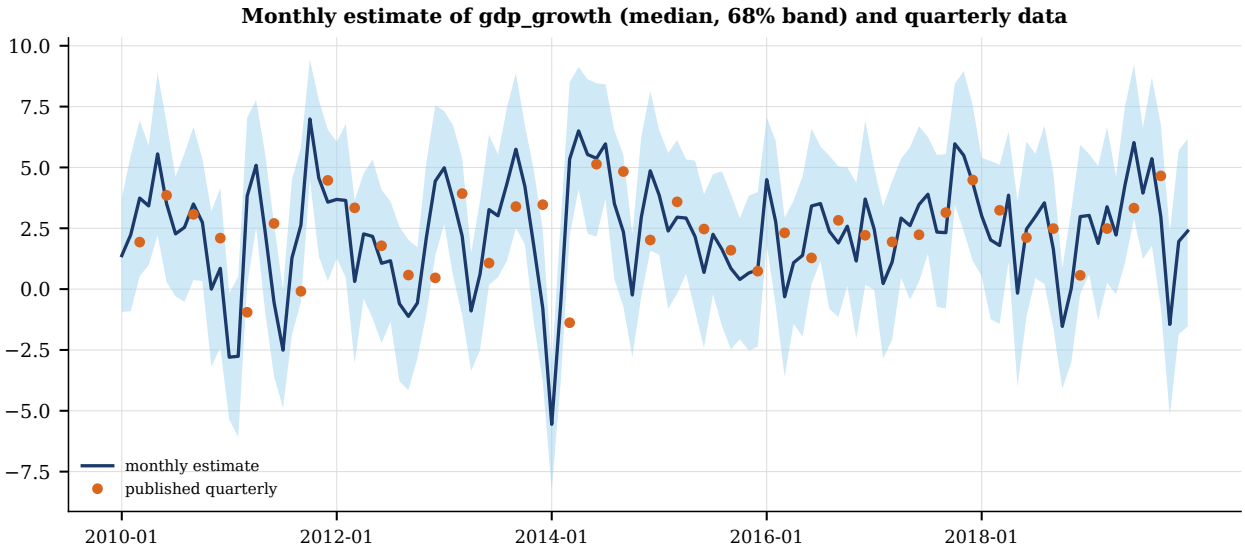


Figure 45.1. Estimate and the monthly path of GDP: Monthly estimate of gdp_growth (median, 68% band) and quarterly data.

```
Quarterly variables: 1
Aggregation: triangle (growth rates)
Lags: 5
Draws: 3000
```

How to read it

The figure shows the posterior median of monthly GDP growth (annualised) and a 68% band. The monthly path is **very uncertain**: the band is typically 6 to 8 percentage points wide, and the median swings between about −5 and 7 from month to month, because only the quarterly aggregate is observed and the model has to split it across three months using industrial production and the interest rate. Use the monthly path to see the direction and timing of movements (for example the dip in early 2014), not to read off precise monthly values. The published quarterly points are drawn as dots; the monthly path aggregates to them by construction.

45.4 The nowcast

```
oes.show(r, "Nowcasts")
```

```
Nowcasts: months after the last quarterly release
              Month  Monthly (median)     16%     84%  Implied quarterly value
Variable
gdp_growth  2019-10           -1.4509 -5.2193  2.3500                   3.8263
gdp_growth  2019-11            1.9586 -1.8558  5.7052                   2.5122
gdp_growth  2019-12            2.3872 -1.5327  6.1710                   1.4219
Note: The implied quarterly value aggregates the latent monthly path with the chosen weights; at the quarter's last
↪ month it is the nowcast of the release.
```

How to read it

After the last published quarter, the model produces a monthly estimate for each month as industrial production and the federal funds rate arrive: −1.45 for October 2019 (68% band −5.2 to 2.4), 1.96 for November and 2.39 for December. The implied value for the quarter, which aggregates the monthly path with the triangle weights, moves from 3.83 after October to 2.51 after November and 1.42 after December. The December figure is the nowcast of the release, 1.4% annualised. The monthly bands are wide, so state the nowcast with its uncertainty, and the large change between months shows how much the nowcast depends on the first monthly data points: update it as each month arrives, rather than treating the first value as final.

45.5 The monthly VAR

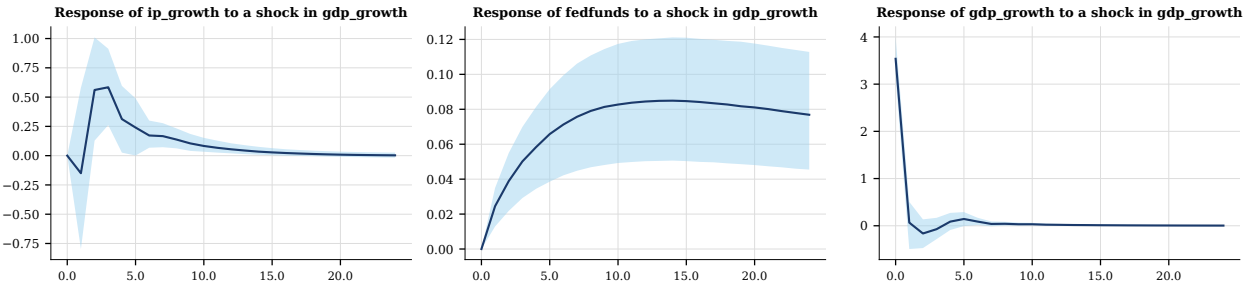


Figure 45.2. The monthly VAR. Panels: Response of ip_growth to a shock in gdp_growth; Response of fedfunds to a shock in gdp_growth; Response of gdp_growth to a shock in gdp_growth.

```
oes.show(r, "Monthly VAR equation: gdp_growth")
oes.show(r, "Monthly VAR equation: fedfunds", rows=5)
oes.plot(r, "Response of", ncols=3, size=(3.6, 2.6))
```

```
Monthly VAR equation: gdp_growth
                 Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const          2.2339     0.4283  1.4076  2.2246  3.1198   1.000
L1.ip_growth   0.1377     0.0795 -0.0148  0.1358  0.2897   0.963
L1.fedfunds    0.4261     1.3792 -2.2418  0.4004  3.2107   0.623
L1.gdp_growth  0.0182     0.1394 -0.2488  0.0184  0.3003   0.545
L2.ip_growth   0.0457     0.0476 -0.0452  0.0463  0.1401   0.827
L2.fedfunds   -0.4152     1.8682 -4.0202 -0.4252  3.2893   0.413
L2.gdp_growth -0.0632     0.0812 -0.2220 -0.0639  0.0965   0.214
L3.ip_growth   0.0134     0.0359 -0.0567  0.0131  0.0849   0.648
L3.fedfunds    0.0559     1.2982 -2.5073  0.0682  2.6085   0.525
L3.gdp_growth -0.0390     0.0568 -0.1487 -0.0400  0.0707   0.247
L4.ip_growth   0.0095     0.0273 -0.0458  0.0091  0.0636   0.636
L4.fedfunds    0.1113     0.9869 -1.8642  0.1287  2.0570   0.557
L4.gdp_growth -0.0104     0.0428 -0.0951 -0.0095  0.0727   0.410
L5.ip_growth   0.0065     0.0213 -0.0353  0.0069  0.0486   0.623
L5.fedfunds   -0.1243     0.6982 -1.4587 -0.1396  1.2735   0.422
L5.gdp_growth  0.0170     0.0358 -0.0536  0.0170  0.0858   0.690

Monthly VAR equation: fedfunds
                 Mean  Std. dev.        2.5%  Median   97.5%  P(> 0)
Parameter
const         -0.0129     0.0159 -4.3671e-02 -0.0131  0.0190   0.213
L1.ip_growth   0.0031     0.0015  8.9000e-05  0.0031  0.0059   0.979
L1.fedfunds    1.2590     0.0443  1.1711e+00  1.2597  1.3461   1.000
L1.gdp_growth  0.0068     0.0032  4.6800e-04  0.0069  0.0132   0.983
L2.ip_growth   0.0002     0.0013 -2.1910e-03  0.0002  0.0027   0.573
```

How to read it

In the monthly GDP equation, only industrial production has a clear effect: the coefficient on its first lag is 0.14 (posterior probability of being positive 0.96); the interest-rate coefficients are very imprecise (the first-lag mean of 0.43 has a standard deviation of 1.4) and the lags of GDP are close to zero. The interest-rate equation is close to a random walk: its own first lag is 1.26, and it responds positively to industrial production and GDP growth (probabilities of being positive 0.98). The impulse responses to a one-standard-deviation GDP-growth shock show industrial production rising after two to three months and the interest rate rising slowly for about 14 months, with 68% bands that exclude zero after the first months. The responses are from a model with constant volatility (see the diagnostics below).

45.6 Convergence and diagnostics

```
oes.show(r, "MCMC")
oes.show(r, "Residual diagnostics")
```

```
MCMC convergence diagnostics
                        Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
```

```
Parameter
gdp_growth: const                        288.0                    0.096    -2.062   0.0393
ip_growth: L1.ip_growth                  239.0                    0.080     0.583   0.5597
fedfunds: L1.ip_growth                   568.0                    0.189    -1.240   0.2151
gdp_growth: L1.ip_growth                 135.0                    0.045    -0.055   0.9560
ip_growth: L1.fedfunds                  3000.0                    1.000    -0.775   0.4386
fedfunds: L1.fedfunds                   3000.0                    1.000     0.550   0.5824
fedfunds: L1.gdp_growth                  399.0                    0.133     0.698   0.4852
ip_growth: L2.ip_growth                 1801.0                    0.600     2.076   0.0379
fedfunds: L2.fedfunds                   3000.0                    1.000    -0.443   0.6576
ip_growth: L3.ip_growth                 2346.0                    0.782    -2.141   0.0322
ip_growth: L4.ip_growth                 3000.0                    1.000    -1.588   0.1124
fedfunds: L5.fedfunds                   3000.0                    1.000    -0.185   0.8530
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.045.

Residual diagnostics of the monthly VAR (posterior means)
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   421.0191   0.0000  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, ip_growth              11.9878   0.4467                                                          H0: no
↪ autocorrelation
Jarque-Bera, ip_growth          1512.4668   0.0000                                                            H0:
↪ normal residuals
ARCH LM, ip_growth                27.5087   0.0000                                        H0: no conditional
↪ heteroskedasticity
Ljung-Box, fedfunds               22.7295   0.0301                                                          H0: no
↪ autocorrelation
Jarque-Bera, fedfunds            657.8576   0.0000                                                            H0:
↪ normal residuals
ARCH LM, fedfunds                 38.7290   0.0000                                        H0: no conditional
↪ heteroskedasticity
Ljung-Box, gdp_growth            123.2309   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth          571.4688   0.0000                                                            H0:
↪ normal residuals
ARCH LM, gdp_growth                5.9210   0.2051                                        H0: no conditional
↪ heteroskedasticity
```

How to read it

The convergence check is not clean. Effective sample sizes are between 135 and 3000 out of 3000 draws; the GDP-equation coefficient on the first lag of industrial production has only 135 (efficiency 4.5%), below the rule of thumb of 200, and three of the twelve reported parameters have Geweke statistics beyond 1.96 (p between 0.03 and 0.04). With twelve parameters, one such value is expected by chance, so three together with the low ESS means the chains need to be longer for a paper: use at least 10,000 draws. Residual tests reject: the monthly GDP residuals are autocorrelated (Ljung-Box $p < 0.001$), which is expected in part because the latent monthly series is constructed from overlapping quarterly aggregates, and industrial production and the interest rate show ARCH effects ($p < 0.001$) and non-normal residuals (the industrial production growth series has a minimum of -53.8). A constant-volatility VAR is a rough description of 1985–2019, which includes the 2008–09 crisis; Chapter 47 shows how to allow for changing volatility.

45.7 Does it nowcast better than a simple rule? A pseudo real-time test

For each of the last eight quarters, we drop the quarter's GDP figure, re-estimate the model on the data available up to that quarter's last month, and compare the nowcast with the published value and with two benchmarks: the previous quarter's value, and the historical mean.

```
q_rows = df.index[df["gdp_growth"].notna()].tolist()[-8:]
out = []
```

```
for k in q_rows:
    sub = df.iloc[:k + 1].copy()
    actual = sub.loc[sub.index[-1], "gdp_growth"]
    sub.loc[sub.index[-1], "gdp_growth"] = np.nan                    # not yet published
    past = sub["gdp_growth"].dropna()
    rr = oes.run("mfbvar", sub.to_csv(index=False), dict(spec, draws=800, burn=500))
    nowcast = next(t for t in rr["tables"] if t["title"].startswith("Nowcasts"))["rows"][-1][-1]
    out.append([df.loc[k, "date"], actual, nowcast, past.iloc[-1], past.mean()])
ev = pd.DataFrame(out, columns=["quarter ending", "published", "MF-BVAR", "last quarter", "historical mean"]).
  ↪ set_index("quarter ending")
print(ev.round(2))
print("RMSE:", {c: round(float(np.sqrt(((ev[c] - ev["published"]) ** 2).mean())), 2) for c in ["MF-BVAR", "
  ↪ last quarter", "historical mean"]})
```

```
               published  MF-BVAR  last quarter  historical mean
quarter ending
2017-12             4.48     3.65          3.14             2.63
2018-03             3.24     2.98          4.48             2.64
2018-06             2.12     3.54          3.24             2.64
2018-09             2.49     2.69          2.12             2.64
2018-12             0.57     2.44          2.49             2.64
2019-03             2.49     0.62          0.57             2.62
2019-06             3.33     1.21          2.49             2.62
2019-09             4.65     1.77          3.33             2.63
RMSE: {'MF-BVAR': 1.68, 'last quarter': 1.35, 'historical mean': 1.28}
```

How to read it

For each of the last eight quarters the model is re-estimated on the data available at the end of that quarter, without the quarter's GDP figure, and its nowcast is compared with the published value and two benchmarks. **The mixed-frequency BVAR does not beat the benchmarks over these eight quarters**: its RMSE is 1.68, against 1.35 for repeating the previous quarter and 1.28 for the historical mean. Its error is smaller than that of the previous-quarter rule in five of the eight quarters, but it misses badly in the last three (errors of 1.9, 2.1 and 2.9 points in 2019, when published growth was 2.5, 3.3 and 4.65 and the nowcasts were 0.6, 1.2 and 1.8). With eight quarters, differences in RMSE of this size are not statistically meaningful, and we did not run a formal test; the honest conclusion is that this sample gives no evidence that the model improves on simple rules, so do not present a nowcast gain. The published quarterly figures are those in the example file shipped with OES.

45.8 Robustness: lags, tightness and the nowcast

```
def last_nowcast(res):
    return round(float(next(t for t in res["tables"] if t["title"].startswith("Nowcasts"))["rows"][-1][-1]), 2)
  ↪

variants = {"baseline (5 lags, lambda1 = 0.2)": {}, "3 lags": dict(lags=3), "8 lags": dict(lags=8),
            "lambda1 = 0.1": dict(lambda1=0.1), "lambda1 = 0.5": dict(lambda1=0.5)}
print(pd.Series({k: last_nowcast(oes.run("mfbvar", data, dict(spec, draws=800, burn=500, **v))) for k, v in
  ↪ variants.items()},
                name="nowcast of the last quarter (2019Q4)").to_frame())
```

```
                                  nowcast of the last quarter (2019Q4)
baseline (5 lags, lambda1 = 0.2)                                  1.47
3 lags                                                            1.55
8 lags                                                            1.44
lambda1 = 0.1                                                     1.59
lambda1 = 0.5                                                     1.40
```

How to read it

The nowcast of the last quarter varies between 1.40 ($\lambda_1 = 0.5$) and 1.59 ($\lambda_1 = 0.1$); three lags give 1.55 and eight lags 1.44. These runs use 800 draws, so they differ slightly from the 1.42 of Section 45.4, which used 3000. The spread of 0.2 points across specifications is small next to the width of the monthly bands and next to the

evaluation errors in Section 45.7, so specification choices are not what limits this nowcast; the uncertainty in the data is.

45.9 Write the results

How to write it: results

We estimate a mixed-frequency Bayesian VAR (Schorfheide and Song 2015) with industrial production growth, the federal funds rate and quarterly GDP growth, 1985–2019, with five monthly lags and a Minnesota-type prior (overall tightness 0.2). Monthly GDP growth is a latent series that aggregates, with triangle weights over five months (Mariano and Murasawa 2003), to the published quarterly growth rate. The posterior (3,000 draws after 2,000 burn-in) places the latent monthly path within a 68% band about 6 to 8 points wide, and the nowcast for 2019Q4 evolves from 3.8 after October to 2.5 after November and 1.4% annualised after December. The nowcast is robust to the number of lags and the prior tightness (1.4 to 1.6). In a pseudo real-time exercise over the last eight quarters, the model's RMSE of 1.68 is higher than that of the previous-quarter rule (1.35) and of the historical mean (1.28), so we do not claim a forecasting gain over simple benchmarks in this sample. Effective sample sizes are low for some coefficients (as low as 135 of 3,000) and the residuals show autocorrelation and ARCH; results should be confirmed with longer chains and stochastic volatility.

Report	What	Where
Data	Frequencies, sample, which months carry the quarterly value, transformations	Section 45.2
Model	Aggregation weights, lags, prior, draws	Section 45.1
Monthly path	Latent monthly GDP growth with bands	Section 45.3
Nowcast	Nowcast after each month of the quarter, with bands	Section 45.4
Evaluation	Pseudo real-time errors against benchmarks (RMSE, number of quarters)	Section 45.7
Diagnostics	Effective sample sizes, residual tests	Section 45.6
Robustness	Lags, tightness	Section 45.8

References. Mariano and Murasawa (2003), *Journal of Applied Econometrics*; Schorfheide and Song (2015), *Journal of Business & Economic Statistics*; Eraker, Chiu, Foerster, Kim and Seoane (2015), *Journal of Business & Economic Statistics*; Giannone, Reichlin and Small (2008), *Journal of Monetary Economics*; Carter and Kohn (1994), *Biometrika*; Bańbura, Giannone, Modugno and Reichlin (2013), *Handbook of Economic Forecasting*.

CHAPTER 46

Panel VAR: Six Estimators for Heterogeneous Countries

Applies when. A VAR for many countries, regions or firms with short time series

Overview. Use this chapter when the same VAR should hold for many units but each unit has only 25 or 30 observations, too few to estimate a VAR on its own. A panel VAR shares information across units. The chapter helps you choose among six estimators (mean group, pooled, Zellner-Hong, hierarchical, static and dynamic factors), and covers the pre-tests, estimation, spillovers between countries, convergence, robustness and the write-up.

Data. Garrett (1998) OECD panel: GDP growth and unemployment in 14 countries, 1966–1990

Software. Python · OES engine · statsmodels

46.1 The question and the model

How do output growth and unemployment move together across rich countries, and are the dynamics the same everywhere? The data are 14 OECD countries observed for 25 years, 336 usable observations. Each country has the VAR

$$y_{it} = c_i + A_{i1} y_{i,t-1} + \cdots + A_{ip} y_{i,t-p} + u_{it}, \qquad y_{it} = (\text{GDP growth}_{it}, \text{unemployment}_{it})'.$$

With one lag each country has 6 coefficients and 25 observations per equation: estimable, but noisy. Pool all countries and the noise averages out, at the price of assuming that every country has the same dynamics. A panel VAR is a choice on the line between these two extremes:

Estimator	Idea	Use it when
OLS mean group (Pesaran-Smith)	Estimate each country separately and average	T is not too small, units differ, and you want the average effect with honest standard errors
Bayesian pooled	One set of coefficients for all units, Minnesota prior	Units are similar and you want the most precise estimate
Zellner-Hong random effects	Unit coefficients are drawn around a common mean, with a fixed amount of shrinkage λ_1	You want partial pooling and can set the shrinkage yourself
Hierarchical (Jarociński 2010)	Same, but the amount of shrinkage is estimated	You want the data to decide how alike the units are
Static factor (Canova-Ciccarelli)	Coefficients are sums of common, unit, variable and lag factors; all units' lags enter every equation	Countries influence each other and you want spillovers
Dynamic factor	Same, with factors that change over time	The relationships are unstable

How to choose

Start with mean group and pooled: if they give similar answers, units are similar and the choice does not matter. If they differ, use the hierarchical model to let the data decide how much to pool (its shrinkage parameter is

a measure of heterogeneity). Use the factor estimators when cross-country spillovers are the question, or the pre-tests show strong dependence between countries (Section 46.2). Report at least two estimators.

How to write it: the model

We estimate a panel VAR in GDP growth and unemployment for 14 OECD countries, 1966–1990, with one lag. To allow for heterogeneity across countries we use the mean-group estimator of Pesaran and Smith (1995), which averages country-specific VARs and computes standard errors from the dispersion of the coefficients across countries, and compare it with a Bayesian pooled estimator with a Minnesota prior, the Zellner-Hong (1989) random-effects estimator and the hierarchical prior of Jarociński (2010). Spillovers between countries are examined with the static and dynamic factor estimators of Canova and Ciccarelli (2009).

46.2 Load the data and pre-test

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("garrett.csv").read()
df = pd.read_csv("garrett.csv")
print(df.groupby("country").size().describe().round(1)[["count", "min", "max"]])
print(df[["gdp_growth", "unem"]].describe().round(2).loc[["mean", "std", "min", "max"]])

spec = dict(vars=["gdp_growth", "unem"], unit="country", time="year", lags=1, estimator="mg")
r = oes.run("pvar", data, spec)
oes.show(r, "Panel unit-root")
```

```
count    14.0
min      25.0
max      25.0
dtype: float64
      gdp_growth   unem
mean        3.25   4.99
std         2.43   3.24
min        -4.30   0.68
max        12.80  13.00
Panel unit-root and cross-sectional dependence tests
                                    Statistic  p-value                                                  Detail
Test
Fisher ADF (Maddala-Wu): gdp_growth   131.3500   0.0000              14 units; H0: all units have a unit root
Pesaran CD: gdp_growth                 21.3155   0.0000  mean |rho| = 0.449; H0: cross-sectional independence
Fisher ADF (Maddala-Wu): unem          22.4900   0.7583              14 units; H0: all units have a unit root
Pesaran CD: unem                       34.4771   0.0000  mean |rho| = 0.723; H0: cross-sectional independence
Note: Stationary series are needed for a panel VAR in levels; strong cross-sectional dependence favours the factor
↪ estimators, which model spillovers between units.
```

How to read it

Two tests, one per variable. The **Fisher ADF** test combines the unit-root tests of the 14 countries: GDP growth rejects a unit root ($p < 0.001$), but **unemployment does not** ($p = 0.76$): it is persistent, and a VAR in levels is an approximation. The **Pesaran CD** test rejects cross-sectional independence for both variables (mean absolute correlation 0.45 for growth and 0.72 for unemployment): countries share common shocks, such as the oil crises. This matters in two ways: standard errors that assume independent countries are too small, and the factor estimators of Section 46.7 are designed for this.

What to do about persistent unemployment: the VAR in levels with a persistent variable is common in the literature and consistent, but you should say so and check robustness with the variable in first differences, using a time-series transformation of your own data before loading it.

46.3 Choose the number of lags

```
oes.show(r, "Lag order")
```

```
Lag order: BIC of the pooled VAR (common sample)
      BIC (lower is better)
Lags
1                    0.8488
2                    0.7391
3                    0.7504
4                    0.8112
Note: BIC prefers 2 lag(s); the model uses 1.
```

How to read it

BIC of the pooled VAR is lowest at two lags (0.739), against 0.849 for one lag and 0.750 for three. With 25 observations per country, one lag already costs six parameters per unit, so the chapter uses one lag for the main results and two as a robustness check in Section 46.10. Report the BIC table, whichever lag length you choose.

46.4 Mean-group estimates

```
oes.summary(r)
oes.show(r, "Mean-group equation")
```

```
Estimator: OLS mean group
Lags: 1
Units: 14
Periods: 25
Mean-group equation: gdp_growth
                 Mean  Std. err.       z  Min over units  Max over units
Regressor
const          2.3818     0.4965  4.7974         -1.1527          6.2006
L1.gdp_growth  0.2483     0.0468  5.3087         -0.1395          0.5616
L1.unem       -0.0099     0.1014 -0.0977         -0.6179          0.7318

Mean-group equation: unem
                 Mean  Std. err.        z  Min over units  Max over units
Regressor
const          1.6456     0.1740   9.4561          0.6654          2.5704
L1.gdp_growth -0.1896     0.0229  -8.2705         -0.3604         -0.0440
L1.unem        0.8070     0.0216  37.4198          0.6907          0.9414
```

How to read it

Each row is the average over the 14 country VARs, with the Pesaran-Smith standard error (the standard deviation across countries divided by $\sqrt{14}$); the last two columns show the smallest and largest country coefficient. **Unemployment equation:** unemployment is persistent (0.81, every country between 0.69 and 0.94) and falls when growth is high: a one-point increase in growth lowers unemployment next year by 0.19 percentage point ($z = -8.3$), negative in every one of the 14 countries (range –0.36 to –0.04). **Growth equation:** growth has modest persistence (0.25) and is not affected by lagged unemployment (–0.01, $z = -0.1$), but the country values range from –0.62 to +0.73, so the average hides large differences between countries.

46.5 Compare the estimators

```
def key_coefficients(res):
    """L1.unem and L1.gdp_growth in the unemployment equation; L1.unem in the growth equation."""
    out = {}
    for t in res["tables"]:
        title = t["title"]
        if title.endswith(": unem") and "equation" in title.lower():
            f = oes.frame(t); out["unem <- own lag"] = f.loc["L1.unem"].iloc[0]; out["unem <- growth"] = f.loc
  ↪ ["L1.gdp_growth"].iloc[0]
        if title.endswith(": gdp_growth") and "equation" in title.lower():
            out["growth <- unem"] = oes.frame(t).loc["L1.unem"].iloc[0]
    return out

rows = {}
```

```
for est in ["mg", "pooled", "zh", "hierarchical"]:
    rows[est] = key_coefficients(oes.run("pvar", data, dict(spec, estimator=est, draws=2000, burn=1000)))
print(pd.DataFrame(rows).T.round(3))
```

```
             growth <- unem  unem <- own lag  unem <- growth
mg                   -0.010            0.807          -0.190
pooled               -0.073            0.882          -0.125
zh                   -0.025            0.887          -0.137
hierarchical         -0.088            0.840          -0.153
```

How to read it

Four estimators, three coefficients. All agree on the sign and the main picture: unemployment is persistent (0.81 to 0.89) and falls when growth rises (−0.13 to −0.19), while lagged unemployment has little or a slightly negative effect on growth. They differ in size. The mean-group effect of growth on unemployment (−0.19) is about 50% larger than the pooled one (−0.125), and pooled persistence (0.88) is higher than mean-group (0.81). This is the pattern expected when slopes differ across units and a pooled estimator imposes a common value: dynamic coefficients are biased towards more persistence (Pesaran and Smith 1995). The hierarchical estimate sits between the two (−0.15 and 0.84), which is what partial pooling should do. For growth on lagged unemployment, mean group finds nothing (−0.01, standard error 0.10), while the pooled and hierarchical estimates are small and negative (−0.07 and −0.09): this relationship is weak and not robust to the estimator, and should not be a headline result.

In a paper: report the mean-group estimate as the main result if the aim is the average effect and the units differ, show the others as robustness, and describe how they differ.

46.6 Impulse responses and variance decomposition

```
oes.plot(r, "Response of unem to a shock in gdp_growth", ncols=1, size=(6, 3))
oes.plot(r, "Response of gdp_growth to a shock in unem", ncols=1, size=(6, 3))
oes.show(r, "Forecast-error variance")
```

```
Forecast-error variance decomposition (posterior median, %)
            Horizon  shock: gdp_growth  shock: unem
Variable
gdp_growth        1              100.0          0.0
gdp_growth        4               99.6          0.4
gdp_growth        8               99.2          0.8
gdp_growth       12               98.9          1.1
unem              1               42.7         57.3
unem              4               75.5         24.5
unem              8               78.0         22.0
unem             12               78.9         21.1
Note: Share of each variable's h-step forecast-error variance due to each Cholesky shock.
```

How to read it

The response is to a one-standard-deviation growth shock (2.3 percentage points), with growth ordered first: unemployment may respond within the year, growth responds to unemployment only with a lag. The median country sees unemployment fall by 0.5 point on impact and by 0.9 point after one year (range across countries −1.1 to −0.4), 0.5 after four years and 0.3 after eight; it has not quite returned to baseline after twelve (−0.17). Growth itself fades in two years (2.3, 0.5, 0.2). The response of growth to an unemployment shock is zero on average (−0.008), with country values from −0.09 to +0.17.

The bands in this model are the 16th to 84th percentiles **across countries**, not estimation uncertainty: they measure heterogeneity. Say so under the figure.

Variance decomposition: growth shocks explain 43% of the forecast-error variance of unemployment at one year, 76% at four and 79% at twelve, while unemployment shocks explain under 1% of the variance of growth: in this panel, growth drives unemployment and not the reverse.

46.7 Spillovers between countries: the factor estimators

```
rs = oes.run("pvar", data, dict(spec, estimator="dynamic", draws=3000, burn=2000))
oes.show(rs, "MCMC")
```

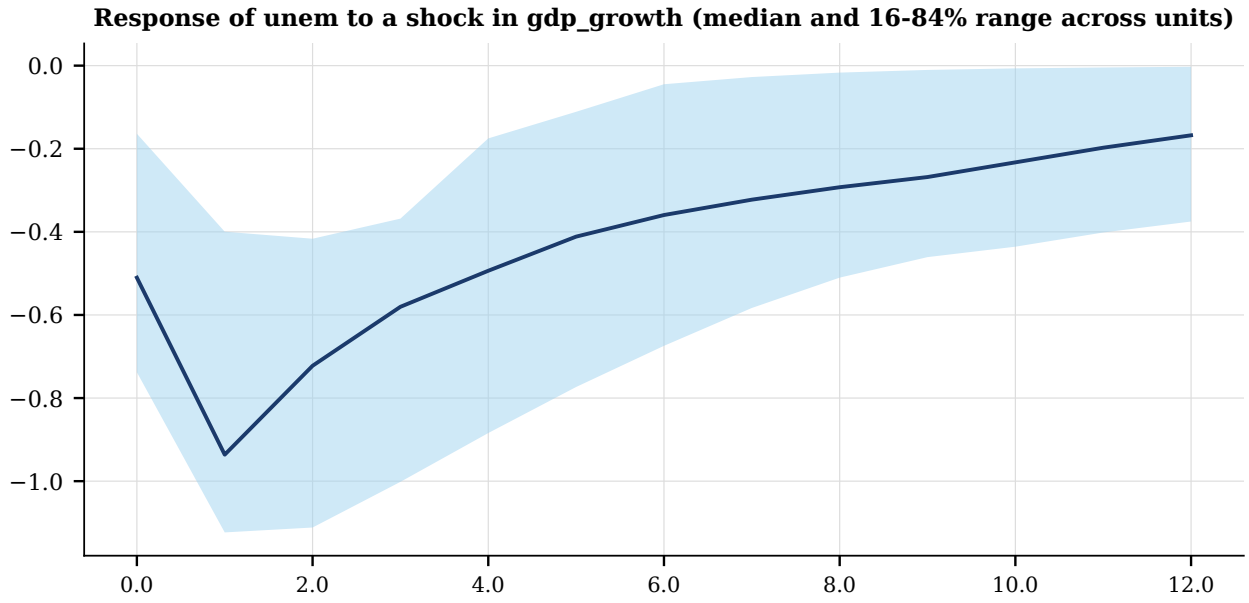


Figure 46.1. Impulse responses and variance decomposition: Response of unem to a shock in gdp_growth (median and 16-84% range across units).

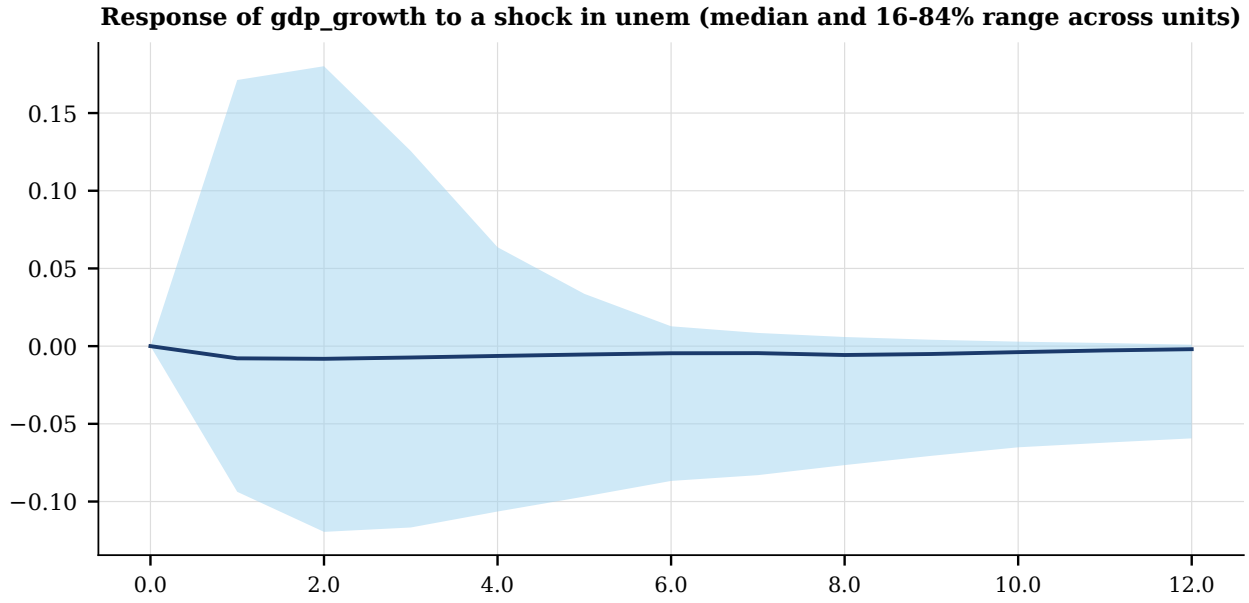


Figure 46.2. Impulse responses and variance decomposition: Response of gdp_growth to a shock in unem (median and 16-84% range across units).

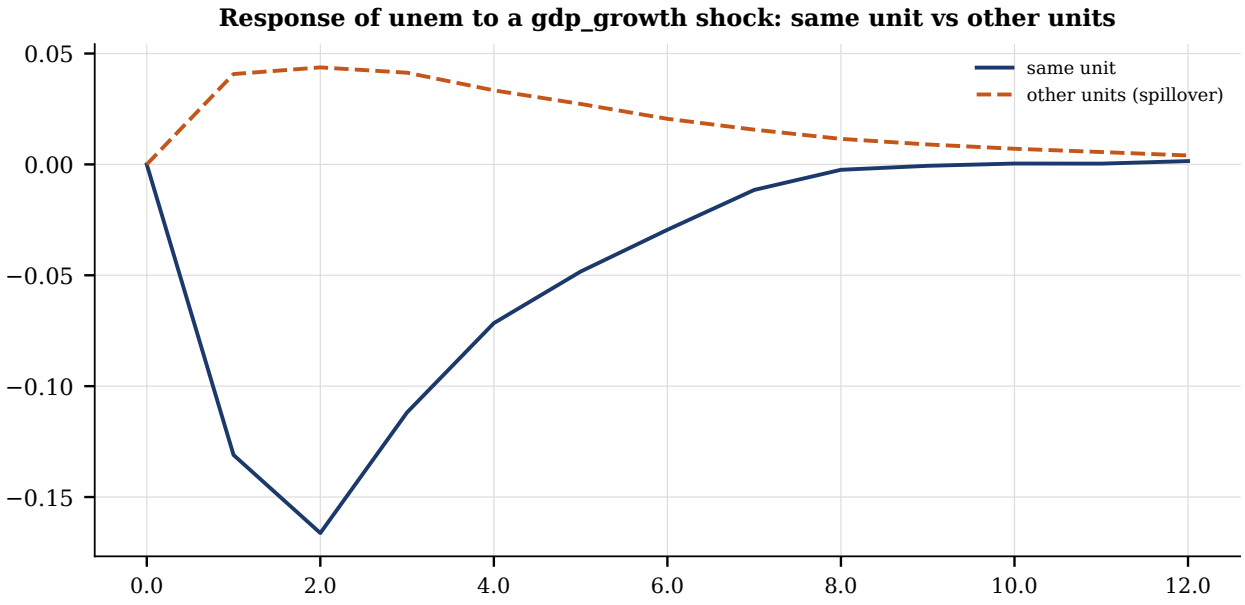


Figure 46.3. Spillovers between countries: the factor estimators: Response of unem to a gdp_growth shock: same unit vs other units.

```
oes.plot(rs, "Response of unem to a gdp_growth shock", ncols=1, size=(6, 3))
```

```
MCMC convergence diagnostics
                              Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
own lag                                       964.0                     0.321     0.158   0.8745
sigma2 United States:gdp_growth              2506.0                     0.835    -0.165   0.8692
sigma2 United Kingdom:gdp_growth             2345.0                     0.782     1.753   0.0796
sigma2 Norway:gdp_growth                     2528.0                     0.843    -1.956   0.0505
sigma2 Finland:gdp_growth                    2381.0                     0.794     1.044   0.2963
sigma2 Sweden:gdp_growth                     2517.0                     0.839     0.018   0.9853
sigma2 Belgium:gdp_growth                    1101.0                     0.367     0.157   0.8756
sigma2 Canada:gdp_growth                     2268.0                     0.756     0.447   0.6549
sigma2 Austria:gdp_growth                    2190.0                     0.730    -0.774   0.4390
sigma2 Japan:gdp_growth                      2620.0                     0.873     1.199   0.2307
sigma2 Italy:gdp_growth                      2234.0                     0.745    -1.217   0.2236
sigma2 West Germany:gdp_growth               1652.0                     0.551     0.478   0.6325
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.321.
```

How to read it

The factor estimators let every country's equation include the lags of all countries and report the effect of a growth shock in one country on **the same country** and on **the other countries** (average per country). Convergence first: with 3000 draws after 2000 burn-in the effective sample size of the own-lag factor is 964 and all Geweke statistics are below 2 in absolute value (with the default 1000 draws and 500 burn-in it was 181 and the statistic was −4.6: do not use the defaults for the dynamic model).

The responses: a shock to growth in one country lowers its own unemployment by 0.13 to 0.17 over the next two years, and moves other countries' growth by −0.04 (3% of the domestic growth response of 1.2 after a year) and their unemployment by +0.04 (about a third of the domestic fall of 0.13, with the opposite sign). Spillovers are small and their sign is not stable across estimators: the static version gives −0.01 to −0.02 for the unemployment spillover. Two cautions. First, these estimators assume uncorrelated shocks, so there is no contemporaneous effect by construction, although the CD test in Section 46.2 shows strong correlation between countries. Second, the own-country response of unemployment is much smaller than in the mean-group model because the factor prior shrinks coefficients strongly in a short panel and there is no impact effect. Use these estimators to compare the size of domestic and foreign effects, not for the size of the domestic effect.

46.8 Hierarchical estimation: how alike are the countries?

```
rh = oes.run("pvar", data, dict(spec, estimator="hierarchical", draws=3000, burn=1500))
oes.show(rh, "Overall shrinkage")
oes.show(rh, "MCMC")
oes.show(rh, "Own first-lag")
```

```
Overall shrinkage
             Mean  Std. dev.    2.5%  Median  97.5%  P(> 0)
Parameter
lambda1    0.0164     0.0072  0.0056  0.0152  0.034     1.0
Note: Small lambda1: units are similar and are pooled; large lambda1: units differ.

MCMC convergence diagnostics
                            Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
b gdp_growth: const                         232.0                     0.077    -0.111   0.9116
b gdp_growth: L1.gdp_growth                 281.0                     0.094     1.280   0.2005
b gdp_growth: L1.unem                       245.0                     0.082    -0.470   0.6387
b unem: const                               300.0                     0.100     0.477   0.6331
b unem: L1.gdp_growth                       245.0                     0.082    -1.125   0.2607
b unem: L1.unem                            1063.0                     0.354    -0.066   0.9471
lambda1                                     280.0                     0.093     1.876   0.0607
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.077.

Own first-lag coefficient by unit (posterior mean)
               gdp_growth    unem
Unit
United States       0.1272  0.8710
United Kingdom      0.0738  0.9270
Norway              0.2130  0.8034
Netherlands         0.1973  0.8549
Denmark             0.1487  0.8106
Finland             0.1694  0.8041
Sweden              0.2452  0.8274
Belgium             0.0744  0.8154
France              0.2370  0.8949
Canada              0.1285  0.8052
Austria             0.2485  0.8155
Japan               0.3377  0.7326
Italy               0.1929  0.9216
West Germany        0.1809  0.8780
```

How to read it

The shrinkage parameter λ_1 measures how different the countries are: it is 0.016 (95% interval 0.006 to 0.034), small, so the hierarchical model pools the countries strongly. The country-specific own-lag coefficients are accordingly close together: unemployment persistence from 0.73 (Japan) to 0.93 (United Kingdom), growth persistence from 0.07 (United Kingdom, Belgium) to 0.34 (Japan), a much narrower range than the unit-by-unit OLS estimates behind the mean-group table (growth: −0.14 to 0.56). The reading is that most of the apparent difference between countries in Section 46.4 is noise from 25 observations, not real heterogeneity. That supports reporting the mean-group and pooled estimates together.

Convergence: with 3000 draws after 1500 burn-in the effective sample sizes are 232 or more and every Geweke statistic is below 2 (the largest is 1.88, for λ_1). With the default 1000 draws they were 85 to 124 for most parameters, too few: use at least 3000 for this estimator.

46.9 Diagnostics

```
oes.show(r, "Residual diagnostics")
```

```
Residual diagnostics (unit VARs, pooled residuals)
                                Statistic  p-value
↪      Detail
Test
```

```
Portmanteau (Hosking), 12 lags    78.9900   0.0032  df = 48; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth             30.1740   0.0026                                                    H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           45.9052   0.0000                                                      H0:
↪ normal residuals
ARCH LM, gdp_growth                3.1358   0.5354                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, unem                   12.3740   0.4161                                                    H0: no
↪ autocorrelation
Jarque-Bera, unem                100.7840   0.0000                                                      H0:
↪ normal residuals
ARCH LM, unem                      8.7026   0.0690                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

Growth residuals have remaining autocorrelation (Ljung-Box p = 0.003) while unemployment residuals have none (p = 0.42), and the joint portmanteau test rejects (p = 0.003; with 336 observations and 48 degrees of freedom it is only approximate). Normality is rejected for both (Jarque-Bera 46 and 101): the residuals have heavy tails, as expected with recessions in the sample. There is no sign of ARCH at 5% (p = 0.54 and 0.07). Serial correlation in the growth equation suggests a second lag; Section 46.10 shows that the main result does not depend on it. Dependence between countries is not in this table: it is tested in Section 46.2 and motivates the factor estimators.

46.10 Robustness: lags, estimators and dropping one country at a time

```python
for p in (1, 2, 3):
    k = key_coefficients(oes.run("pvar", data, dict(spec, lags=p)))
    print(f"mean group, {p} lag(s): unem <- growth =", round(float(k["unem <- growth"]), 3))

est = []
for c in sorted(df["country"].unique()):
    sub = df[df["country"] != c].to_csv(index=False)
    est.append((c, key_coefficients(oes.run("pvar", sub, spec))["unem <- growth"]))
est = pd.DataFrame(est, columns=["dropped country", "unem <- growth"]).set_index("dropped country")
print(est.round(3).T)
print("range:", est.min().round(3).iloc[0], "to", est.max().round(3).iloc[0])
```

```
mean group, 1 lag(s): unem <- growth = -0.19
mean group, 2 lag(s): unem <- growth = -0.179
mean group, 3 lag(s): unem <- growth = -0.2
dropped country  Austria  Belgium  Canada  Denmark  Finland  France  Italy  Japan  Netherlands  Norway  Sweden  \
unem <- growth    -0.195   -0.191  -0.192   -0.185   -0.185   -0.19 -0.195 -0.201       -0.181  -0.189  -0.199

dropped country  United Kingdom  United States  West Germany
unem <- growth           -0.176         -0.189        -0.187
range: -0.201 to -0.176
```

How to read it

Three checks of the effect of growth on unemployment (mean group, −0.19). **Lags:** with one, two and three lags the coefficient is −0.19, −0.18 and −0.20. **Leave one country out:** re-estimating without each country in turn gives values between −0.20 (dropping Japan) and −0.18 (dropping the United Kingdom): no single country drives the result. **Estimators:** Section 46.5 shows −0.13 to −0.19. The conclusion that growth lowers unemployment next year is robust; the size depends on the estimator, by about a third.

46.11 Write the results

How to write it: results

Table 1 reports the mean-group estimates. Unemployment is persistent (0.81) and responds negatively to lagged growth: a one-percentage-point increase in growth lowers unemployment by 0.19 percentage point in the following year (Pesaran-Smith standard error 0.023), and the coefficient is negative in all 14 countries. Lagged unemployment has no effect on growth on average (−0.01, standard error 0.10) but the country estimates range widely. A one-standard-deviation growth shock lowers unemployment by 0.9 percentage point after one year (the 16th to 84th percentile across countries is −1.1 to −0.4); the effect halves after about four years. Growth shocks explain 76% of the forecast-error variance of unemployment at a four-year horizon. The pooled, Zellner-Hong and hierarchical estimators give the same signs and a smaller effect (−0.13 to −0.15), consistent with the upward bias in the persistence of pooled dynamic estimators when slopes differ. The hierarchical model estimates a small degree of heterogeneity ($\lambda_1 = 0.016$). Spillovers estimated with the Canova-Ciccarelli factor model are small: a growth shock in one country moves other countries' growth by 3% of its domestic effect and their unemployment by about a third of the domestic effect, with the opposite sign, and the sign is not stable across the static and dynamic versions. Results are unchanged with two or three lags and when any one country is dropped (−0.20 to −0.18). Unemployment fails the panel unit-root test, so we also note that the estimates relate to a persistent variable in levels.

Report	What	Where
Data	Countries, years, balance, unit-root and cross-sectional dependence tests	Section 46.2
Estimators	Which, and why; prior settings for the Bayesian ones	Section 46.1
Lags	BIC and the choice	Section 46.3
Results	Average coefficients with standard errors, heterogeneity (range across units), impulse responses	Sections 46.4 to 46.6
Spillovers	Factor-model responses, convergence statistics	Section 46.7
Diagnostics	Residual tests, effective sample sizes for samplers	Sections 46.7 to 46.9
Robustness	Other estimators, lags, leave-one-country-out	Sections 46.5 and 46.10

References. Pesaran and Smith (1995), *Journal of Econometrics*; Canova and Ciccarelli (2009), *International Economic Review*; Jarociński (2010), *Journal of Applied Econometrics*; Zellner and Hong (1989), *Journal of Econometrics*; Pesaran (2004), CESifo Working Paper 1229; Garrett (1998), *Partisan Politics in the Global Economy*; Canova and Ciccarelli (2013), ECB Working Paper 1507.

CHAPTER 47

Stochastic-Volatility VAR (SV-VAR)

Applies when. Shocks that were large in some periods and small in others, as in the Great Moderation
Overview. Use this chapter when the size of the shocks hitting the economy changed over time. A constant-variance VAR then gives bands that are too narrow in turbulent periods and too wide in calm ones, and may mistake changing volatility for changing coefficients. The SV-VAR keeps the coefficients fixed and lets each shock's variance follow a random walk. The chapter covers the model, volatility paths, the check that volatility explains the ARCH, impulse responses at different dates, robustness and the write-up.
Data. U.S. quarterly inflation, unemployment and the T-bill rate, 1959–2009
Software. Python · OES engine · statsmodels

47.1 The question and the model

Were the shocks of the 1970s larger than those of the 1990s, and does it matter for inference? The model is a VAR with constant coefficients and time-varying shock variances:

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + u_t, \qquad u_t = A_0^{-1} H_t^{1/2} \varepsilon_t, \qquad \varepsilon_t \sim N(0, I),$$

$$\ln h_{i,t} = \ln h_{i,t-1} + \eta_{i,t}, \qquad \eta_{i,t} \sim N(0, s_i),$$

where $H_t = \mathrm{diag}(h_{1t}, h_{2t}, h_{3t})$ holds the variances of the three structural (Cholesky) shocks and A_0 is a lower-triangular matrix of contemporaneous relations (Cogley and Sargent 2005). The log-variances follow random walks: volatility can rise or fall gradually, but cannot jump back. The coefficients get the Minnesota prior of Chapter 42.

The model is estimated by Gibbs sampling, drawing the log-variances with the mixture-of-normals approximation of Kim, Shephard and Chib (1998). The parameter s_i, the variance of the innovation to the log-volatility, controls how fast volatility moves.

Volatility or coefficients?

Changing volatility and changing coefficients are hard to tell apart, and a model that allows only one of them will attribute all the instability to it. Sims and Zha (2006) found that most of the apparent change in U.S. monetary policy since the 1960s was a change in the size of the shocks. Chapter 48 allows time-varying coefficients with constant variance; this chapter allows time-varying variance with constant coefficients. Compare them, and if both matter, use a TVP-SV-VAR (not in OES).

How to write it: the model

We estimate a VAR with stochastic volatility (Cogley and Sargent 2005) in inflation, unemployment and the three-month Treasury bill rate, with two lags. The VAR coefficients are constant, with a Minnesota prior with overall tightness 0.1; the structural shocks, identified recursively with the interest rate ordered last, have volatilities that follow independent random walks in logs. The posterior is simulated with a Gibbs sampler in which the log-volatilities are drawn using the mixture approximation of Kim, Shephard and Chib (1998); we use 4,000 draws after 2,000 burn-in.

47.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
vars_ = ["inflation", "unemp", "tbilrate"]
spec = dict(vars=vars_, time="date", lags=2, lambda1=0.1, delta="0, 1, 1", draws=4000, burn=2000)
r = oes.run("svvar", data, spec)
oes.show(r, "Unit-root")
```

```
Unit-root tests
          ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
inflation                  0.0271               0.0822                   0.0000         0.0808       I(0)
unemp                      0.1107               0.3084                   0.0008         0.1000       I(1)
tbilrate                   0.2805               0.4462                   0.0000         0.0538       I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

Inflation rejects a unit root ($p = 0.03$); unemployment and the T-bill rate do not ($p = 0.11$ and 0.28), so they get a random-walk prior on their own first lag (`delta = "0, 1, 1"`). The VAR is in levels, as in Chapter 42.

47.3 Estimate

```
oes.summary(r)
oes.show(r, "Equation: tbilrate")
oes.show(r, "Variance of the log")
```

```
Observations: 200
Lags: 2
Draws: 4000
Last volatility: inflation: 4.5788
Last volatility: unemp: 0.8971
Last volatility: tbilrate: 0.5236
Equation: tbilrate
                Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const          0.4896     0.1730  0.1412  0.4887  0.8285   0.998
L1.inflation   0.0139     0.0116 -0.0081  0.0140  0.0370   0.883
L1.unemp      -0.0739     0.0667 -0.2050 -0.0746  0.0563   0.136
L1.tbilrate    1.0319     0.0429  0.9454  1.0325  1.1156   1.000
L2.inflation   0.0078     0.0078 -0.0075  0.0078  0.0227   0.839
L2.unemp       0.0389     0.0651 -0.0867  0.0390  0.1646   0.733
L2.tbilrate   -0.1026     0.0398 -0.1801 -0.1035 -0.0241   0.005

Variance of the log-volatility innovations
                Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
s[inflation]   0.1993     0.0847  0.0813  0.1845  0.4185     1.0
s[unemp]       0.3789     0.1267  0.1707  0.3624  0.6727     1.0
s[tbilrate]    0.2264     0.0776  0.1070  0.2138  0.4135     1.0
```

How to read it

The coefficients are constant, so the interest-rate equation is read as in Chapter 42: the policy rate is persistent (own lags 1.03 and −0.10, sum 0.93) and responds to inflation with a positive sign (probability 0.88 for the first lag, with an interval that includes zero). The new objects are the s_i, the variances of the quarterly innovations in log volatility: 0.20 for inflation, 0.38 for unemployment and 0.23 for the interest rate, all clearly above zero (the 95% intervals start at 0.08, 0.17 and 0.11). A value of 0.2 means that the log-variance of a shock moves by about 0.45 per quarter, so its standard deviation by about 22%. Unemployment shocks have the most erratic

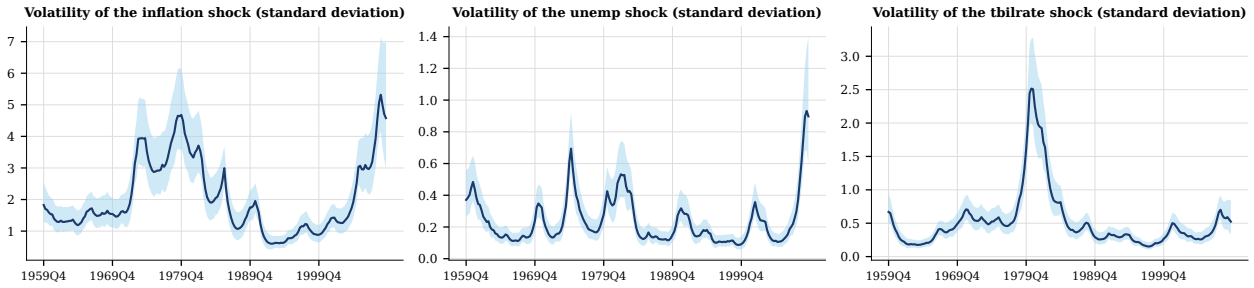


Figure 47.1. How did volatility change?. Panels: Volatility of the inflation shock (standard deviation); Volatility of the unemp shock (standard deviation); Volatility of the tbilrate shock (standard deviation).

volatility. If an s_i had been close to zero, that shock would have constant variance and the model would reduce to the ordinary BVAR for it.

47.4 How did volatility change?

```
oes.plot(r, "Volatility of", ncols=3, size=(3.6, 2.6))
oes.show(r, "Shock volatility")
sd = {c["title"].split(" of the ")[1].split(" shock")[0]: pd.Series(c["series"][0]["y"], index=c["x"]) for c
  ↪ in r["charts"] if c["title"].startswith("Volatility of")}
sd = pd.DataFrame(sd)
year = sd.index.str[:4].astype(int)
per = pd.DataFrame({"1960-1983": sd[(year <= 1983)].mean(), "1984-2006": sd[(year >= 1984) & (year <= 2006)].
  ↪ mean(), "2007-2009": sd[year >= 2007].mean()}).T
print(per.round(2))
print("ratio 1984-2006 / 1960-1983:", (per.loc["1984-2006"] / per.loc["1960-1983"]).round(2).to_dict())
```

```
Shock volatility over the sample (posterior median s.d.)
          Minimum    Date  Maximum    Date    Last
Variable
inflation   0.6005  1993Q1   5.3173  2008Q4  4.5788
unemp       0.0867  1999Q2   0.9315  2009Q2  0.8971
tbilrate    0.1514  1997Q3   2.5160  1980Q3  0.5236

          inflation  unemp  tbilrate
1960-1983       2.41   0.27      0.69
1984-2006       1.41   0.16      0.34
2007-2009       4.17   0.54      0.57
ratio 1984-2006 / 1960-1983: {'inflation': 0.58, 'unemp': 0.6, 'tbilrate': 0.5}
```

How to read it

The charts show the posterior median standard deviation of each structural shock. The table gives the minimum, the maximum and the last value. The interest-rate shock is largest in 1980Q3 (2.5 percentage points, the Volcker period) and smallest in 1997Q3 (0.15): a 17-fold range. Inflation shocks are smallest in 1993Q1 (0.6) and largest in 2008Q4 (5.3), the collapse of prices at the start of the financial crisis. Unemployment shocks range from 0.09 (1999Q2) to 0.93 (2009Q2).

The period averages show the **Great Moderation**: from 1960–1983 to 1984–2006 the volatility of inflation fell by 42%, that of unemployment by 40% and that of the interest rate by 50%, and it rose again in 2007–2009 (inflation shocks 4.2, above the 1960–1983 average of 2.4). Say how you chose the break at 1984 (it is the convention, not an estimate), and note that the table shows posterior medians without bands.

47.5 Is the volatility model needed? Compare with constant variance

```
oes.show(r, "Diagnostics of the standardised")
rc = oes.run("bvar", data, dict(vars=vars_, time="date", lags=2, prior="nw", lambda1=0.1, delta="0, 1, 1"))
oes.show(rc, "Residual diagnostics")
```

```
Diagnostics of the standardised structural shocks
```

```
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   256.3691   0.0000  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, shock inflation        59.2243   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, shock inflation       4.2770   0.1178                                                            H0:
↪ normal residuals
ARCH LM, shock inflation           2.1071   0.7161                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, shock unemp            46.4842   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, shock unemp           7.5828   0.0226                                                            H0:
↪ normal residuals
ARCH LM, shock unemp               4.2702   0.3707                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, shock tbilrate         14.3963   0.2761                                                          H0: no
↪ autocorrelation
Jarque-Bera, shock tbilrate        6.3191   0.0424                                                            H0:
↪ normal residuals
ARCH LM, shock tbilrate            6.2427   0.1817                                          H0: no conditional
↪ heteroskedasticity
Note: Shocks divided by their estimated volatility should be white noise and close to normal if the volatility model
↪ fits; compare with a constant-volatility VAR, where ARCH tests usually reject.

Residual diagnostics (at the posterior mean)
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   256.5220   0.0000  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, inflation              51.0310   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, inflation            57.4347   0.0000                                                            H0:
↪ normal residuals
ARCH LM, inflation                23.0871   0.0001                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, unemp                  65.0997   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, unemp               151.4370   0.0000                                                            H0:
↪ normal residuals
ARCH LM, unemp                    40.6066   0.0000                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               48.4647   0.0000                                                          H0: no
↪ autocorrelation
Jarque-Bera, tbilrate           1651.9622   0.0000                                                            H0:
↪ normal residuals
ARCH LM, tbilrate                 57.1218   0.0000                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

If the volatility model is right, the shocks divided by their estimated volatility should be white noise with no ARCH. They are: the ARCH test does not reject for any of the three standardised shocks (p = 0.72, 0.37 and 0.18), while in the constant-variance BVAR it rejects for all (p below 0.0001). The heavy tails also disappear: the Jarque-Bera statistic of the interest-rate residual falls from 1,652 to 6.3, and normality is not rejected for the inflation shock (p = 0.12). Volatility changes, not fat-tailed shocks, explain most of the non-normality.

What the volatility model does not fix is serial correlation: the Ljung-Box test still rejects for the standardised inflation and unemployment shocks (statistics 59 and 46), as in the constant-variance VAR. That is a problem with the conditional mean, and the remedy is more lags (Section 47.8). Report both tables: the contrast is the argument for the SV-VAR.

47.6 Impulse responses in calm and volatile periods

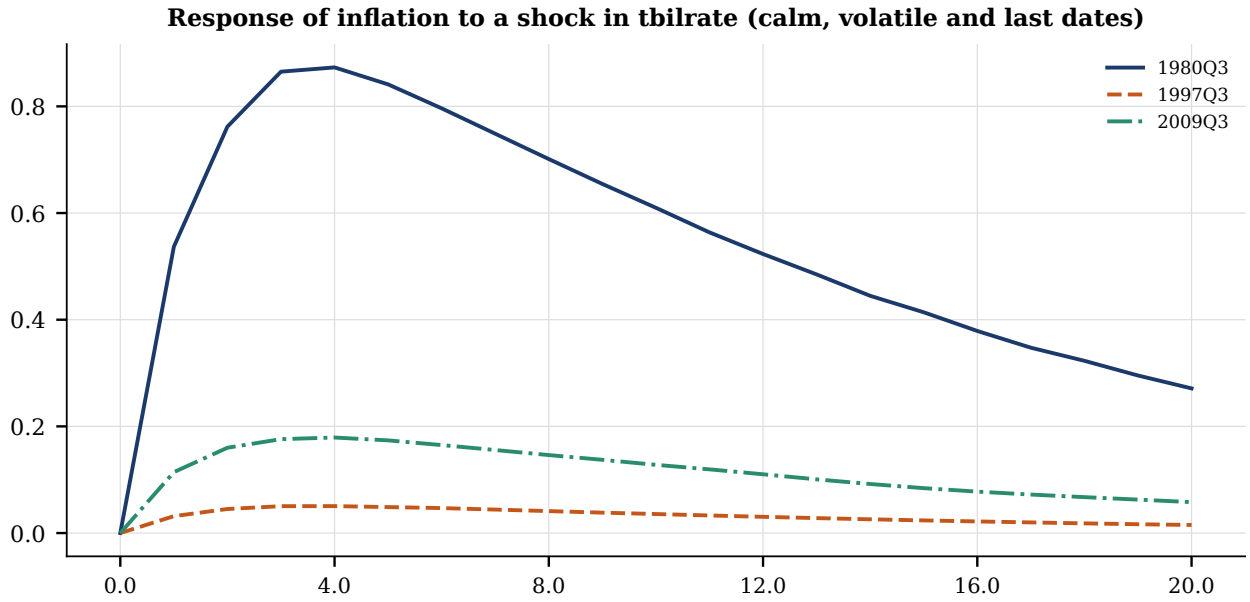


Figure 47.2. Impulse responses in calm and volatile periods: Response of inflation to a shock in tbilrate (calm, volatile and last dates).

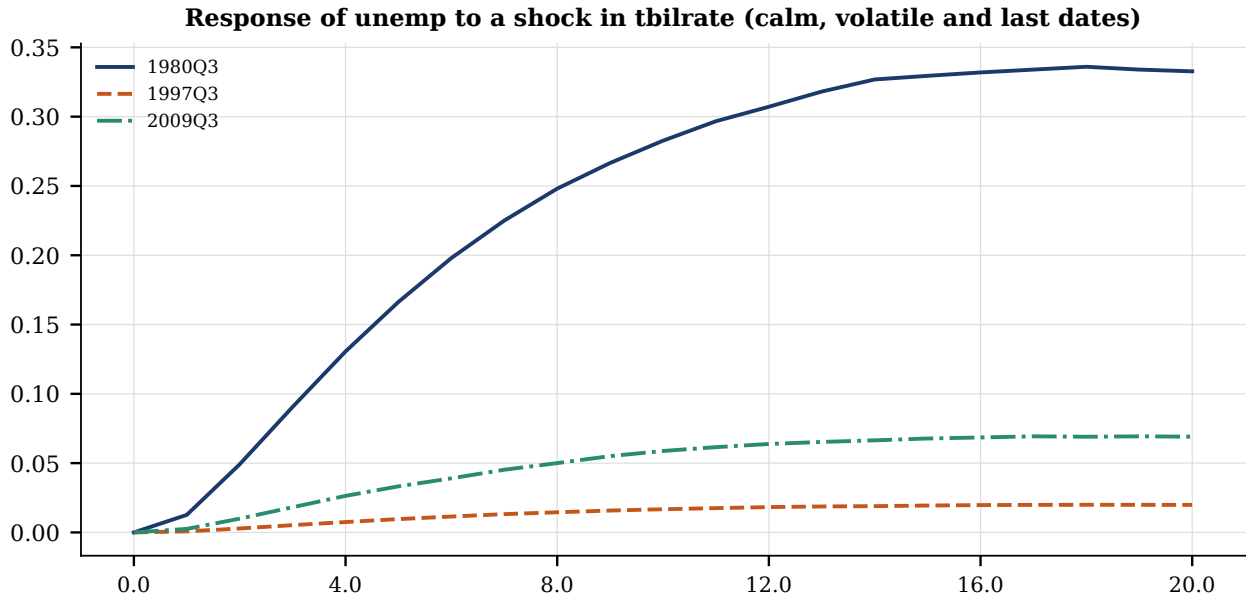


Figure 47.3. Impulse responses in calm and volatile periods: Response of unemp to a shock in tbilrate (calm, volatile and last dates).

```
oes.plot(r, "Response of inflation to a shock in tbilrate", ncols=1, size=(6, 3))
oes.plot(r, "Response of unemp to a shock in tbilrate", ncols=1, size=(6, 3))
```

How to read it

The charts show the response of inflation and unemployment to an interest-rate shock at three dates: the volatile 1980Q3, the calm 1997Q3 and the last date. With constant coefficients the responses have the **same shape** at every date, and differ only in the size of a one-standard-deviation shock: after a year inflation rises by 0.87 in 1980Q3 and by 0.05 in 1997Q3, a ratio of 17, which is the ratio of the shock sizes (2.5 and 0.15). Per percentage point of policy shock, both are about 0.35. Do not read the difference as a change in monetary transmission: it is a change in the size of the shock. Report responses to a shock of fixed size (such as 25 basis points) or normalise them, as Chapter 42 does. This is the key contrast with the TVP-VAR, where the

responses change shape because the coefficients change. As in Chapter 42, inflation rises after the shock (the price puzzle).

47.7 Convergence

```
oes.show(r, "MCMC")
```

```
MCMC convergence diagnostics
                        Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
tbilrate: const                        1483.0                     0.371     1.258   0.2085
inflation: L1.inflation                2161.0                     0.540     0.132   0.8948
unemp: L1.unemp                        1426.0                     0.357    -0.723   0.4696
inflation: L1.tbilrate                 2965.0                     0.741    -0.162   0.8713
tbilrate: L1.tbilrate                  2277.0                     0.569     1.005   0.3150
inflation: L2.inflation                2446.0                     0.611     0.544   0.5868
unemp: L2.unemp                        1402.0                     0.351     0.962   0.3361
unemp: L2.tbilrate                     4000.0                     1.000     0.791   0.4291
tbilrate: L2.tbilrate                  2090.0                     0.523    -2.146   0.0318
s[inflation]                            155.0                     0.039     0.382   0.7028
s[unemp]                                125.0                     0.031     1.887   0.0591
s[tbilrate]                             187.0                     0.047    -0.468   0.6401
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.031.
```

How to read it

For the VAR coefficients the effective sample sizes are above 1,400 (efficiency of 35% to 100%) and the Geweke statistics are below 2 in absolute value except one (the second T-bill lag in the interest-rate equation, −2.1), unremarkable among many parameters. The variances s_i of the volatility innovations mix more slowly: effective sample sizes of 125 to 187 (efficiency of 3% to 5%), below the guideline of 200, though their Geweke statistics are fine (up to 1.9). For a paper, use 10,000 draws or more and check that the volatility paths do not change.

47.8 Robustness: lags, tightness and the amount of volatility drift

```
def per_ratio(res):
    s = {c["title"].split(" of the ")[1].split(" shock")[0]: pd.Series(c["series"][0]["y"], index=c["x"]) for
  ↪ c in res["charts"] if c["title"].startswith("Volatility of")}
    s = pd.DataFrame(s); y = s.index.str[:4].astype(int)
    return (s[(y >= 1984) & (y <= 2006)].mean() / s[y <= 1983].mean()).round(2)

rows = {"baseline (2 lags, lambda1 = 0.1)": per_ratio(r)}
for lg in (1, 4):
    rows[f"{lg} lag(s)"] = per_ratio(oes.run("svvar", data, dict(spec, lags=lg, draws=2000, burn=1000)))
for lam in (0.05, 0.3):
    rows[f"lambda1 = {lam}"] = per_ratio(oes.run("svvar", data, dict(spec, lambda1=lam, draws=2000, burn=1000))
  ↪ )
print(pd.DataFrame(rows).T)
```

```
                               inflation  unemp  tbilrate
baseline (2 lags, lambda1 = 0.1)      0.58   0.60      0.50
1 lag(s)                              0.56   0.58      0.51
4 lag(s)                              0.60   0.60      0.48
lambda1 = 0.05                        0.48   0.58      0.52
lambda1 = 0.3                         0.68   0.65      0.47
```

How to read it

Each row is a variant of the model, and each number the ratio of average volatility in 1984–2006 to that in 1960–1983 (below one means the Great Moderation). With one, two or four lags the ratios are 0.56 to 0.60 for inflation, 0.58 to 0.60 for unemployment and 0.48 to 0.51 for the interest rate. A tighter prior ($\lambda_1 = 0.05$) gives 0.48 for inflation, a looser one ($\lambda_1 = 0.3$) 0.68; the interest-rate ratio is stable at about 0.5. The conclusion

that volatility fell by 30% to 50% is robust; the size of the fall for inflation depends on the prior by about 20 percentage points, so report a range.

47.9 Write the results

How to write it: results

Constant-variance residuals show strong conditional heteroskedasticity in all three variables (ARCH test $p < 0.0001$) and heavy tails (Jarque-Bera 1,652 for the interest rate). Allowing the variance of each structural shock to follow a random walk removes both: the standardised shocks show no ARCH (p between 0.18 and 0.72) and are close to normal. Estimated volatility fell sharply: between 1960–1983 and 1984–2006 the average standard deviation of inflation shocks fell by 42%, that of unemployment shocks by 40% and that of interest-rate shocks by 50%, and it rose again in the financial crisis. The interest-rate shock was 17 times larger in 1980Q3, at the height of the Volcker disinflation, than in 1997Q3. The ratios hardly change with one to four lags, and the fall in volatility ranges from 30% to 53% across lag lengths and prior tightness. Because the coefficients are constant, the impulse responses at different dates differ only in the size of the shock; for time-varying propagation we refer to the TVP-VAR. Standardised inflation and unemployment shocks remain serially correlated, which is a limitation of the model's conditional mean.

Report	What	Where
Model	Constant coefficients, random-walk log-volatilities, prior, draws	Section 47.1
Volatility	Paths with bands; averages by period	Section 47.4
Need for SV	Diagnostics of standardised shocks against a constant-variance VAR	Section 47.5
Responses	At calm and volatile dates, with bands	Section 47.6
Diagnostics	Convergence	Section 47.7
Robustness	Lags, λ_1, volatility ratio	Section 47.8

References. Cogley and Sargent (2005), *Review of Economic Dynamics*; Kim, Shephard and Chib (1998), *Review of Economic Studies*; Primiceri (2005), *Review of Economic Studies*; Sims and Zha (2006), *American Economic Review*; Stock and Watson (2002), *NBER Macroeconomics Annual*; Clark (2011), *Journal of Business & Economic Statistics*.

CHAPTER 48

Time-Varying Parameter VAR (TVP-VAR)

Applies when. You suspect the transmission of shocks has changed over the sample

Overview. Use this chapter when the question is whether the relationships between variables, such as the effect of interest rates on inflation, have changed over time. A TVP-VAR lets every coefficient drift as a random walk. The chapter covers the training-sample prior, the evidence for time variation (which is often weaker than the charts suggest), responses at different dates, convergence, robustness and the write-up.

Data. U.S. quarterly inflation, unemployment and the T-bill rate, 1959–2009 (first 10 years used as a training sample)

Software. Python · OES engine · statsmodels

48.1 The question and the model

Has the effect of monetary policy on inflation changed since the 1960s? The model lets all VAR coefficients drift:

$$y_t = X_t'\beta_t + u_t, \qquad \beta_t = \beta_{t-1} + v_t, \qquad v_t \sim N(0, Q), \quad u_t \sim N(0, \Sigma),$$

where X_t holds a constant and p lags of the three variables. With three variables and two lags each date has 21 coefficients; the random walk lets them move a little each quarter. How much they move is governed by Q, whose prior is set from a **training sample**: the first 40 observations are used only to estimate a constant-parameter VAR, whose coefficients and covariance matrix centre the prior for β_0 and Q (Primiceri 2005). The scale k_Q says how fast coefficients may drift: small values give nearly constant coefficients, large values a flexible but noisy path.

Be sceptical of time variation

A TVP-VAR can always find movement in coefficients, even when the true model is constant, because the random walk is flexible and the data are noisy. Three habits keep a paper honest: (1) test whether coefficients really moved, using the posterior probability that they changed (Section 48.4); (2) show that the results do not depend on k_Q and the training sample (Section 48.8); (3) in a macroeconomic VAR, ask whether changing *volatility* (Chapter 47) rather than changing coefficients explains the instability: much of the apparent time variation in the 1970s and 1980s turned out to be a change in shock variances (Sims and Zha 2006).

How to write it: the model

We estimate a time-varying parameter VAR with random-walk coefficients and a constant error covariance matrix, in inflation, unemployment and the three-month Treasury bill rate, with two lags (Cogley and Sargent 2005; Primiceri 2005). The prior for the initial coefficients and the covariance of their innovations is calibrated on a training sample of the first 40 quarters (1959–1969), so estimation starts in 1969Q4 with 160 observations; the prior scale of the innovation variance is $k_Q = 0.01$. The posterior is simulated with a Gibbs sampler, drawing the coefficient paths with the Carter-Kohn (1994) simulation smoother, using 6,000 draws after 4,000 burn-in draws. Impulse responses at a date use the coefficients of that date, identified recursively with the interest rate ordered last.

48.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
vars_ = ["inflation", "unemp", "tbilrate"]
print(df[vars_].describe().round(2).loc[["mean", "std", "min", "max"]])
spec = dict(vars=vars_, time="date", lags=2, training=40, kQ=0.01, draws=6000, burn=4000)
r = oes.run("tvpvar", data, spec)           # about 80 seconds on a laptop; several minutes in the browser
oes.show(r, "Unit-root")
```

```
      inflation  unemp  tbilrate
mean       3.98   5.89      5.32
std        3.25   1.46      2.80
min       -8.79   3.40      0.12
max       14.62  10.70     15.33
Unit-root tests
           ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
inflation                  0.0271               0.0822                   0.0000         0.0808       I(0)
unemp                      0.1107               0.3084                   0.0008         0.1000       I(1)
tbilrate                   0.2805               0.4462                   0.0000         0.0538       I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

A TVP-VAR is estimated in levels without a prior on unit roots: the training sample sets the prior mean of the coefficients, and the random walk lets the persistence change. Inflation rejects a unit root at 5% (p = 0.03), unemployment and the T-bill rate do not (p = 0.11 and 0.28). If you use differences, say so. Because coefficients can drift above one, the model can produce explosive paths in short stretches; the summary line "Iterations with no stable path" counts the draws where this could not be avoided (zero here).

48.3 Estimate

```
oes.summary(r)
oes.show(r, "Residual covariance", rows=4)
```

```
Observations: 160
Training sample: 40
Lags: 2
kQ: 0.01
Draws: 6000
Iterations with no stable path: 0
Residual covariance (posterior)
                             Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
Sigma[inflation,inflation]  4.9311     0.6638  3.7521  4.8772  6.4184     1.0
Sigma[inflation,unemp]     -0.1967     0.0534 -0.3082 -0.1945 -0.0953     0.0
Sigma[inflation,tbilrate]   0.8075     0.2011  0.4391  0.7959  1.2307     1.0
Sigma[unemp,inflation]     -0.1967     0.0534 -0.3082 -0.1945 -0.0953     0.0
```

How to read it

160 observations are used for estimation, 1969Q4 to 2009Q3, with the first 40 quarters kept for the prior. The covariance matrix of the residuals is constant here (stochastic volatility is the subject of Chapter 47): inflation shocks have a variance of about 5, shocks to unemployment about 0.06, and interest-rate and inflation shocks are positively correlated (0.81 covariance), while unemployment shocks are negatively correlated with both.

48.4 Is there time variation?

The central table compares each coefficient at the start and the end of the estimation sample, with the posterior probability that it increased.

```
oes.show(r, "Coefficients at the start")
tab = oes.frame(next(t for t in r["tables"] if t["title"].startswith("Coefficients at")))
p = tab["P(change > 0)"].astype(float)
n_out = int(((p < 0.05) | (p > 0.95)).sum())
print(f"{n_out} of {len(p)} coefficients have P(change > 0) below 0.05 or above 0.95; chance alone would give
  ↪ about {0.1 * len(p):.1f}.")
```

```
Coefficients at the start (1969Q4) and the end (2009Q3) of the sample
             Regressor  Start (mean)  End (mean)  Change  P(change > 0)
Equation
inflation         const        1.9084      2.3602  0.4518          0.547
inflation  L1.inflation        0.1162      0.1441  0.0279          0.546
inflation      L1.unemp       -0.7334     -0.9905 -0.2571          0.435
inflation   L1.tbilrate        0.7807     -0.0962 -0.8769          0.170
inflation  L2.inflation        0.3619     -0.1872 -0.5491          0.008
inflation      L2.unemp        0.4812      0.8389  0.3577          0.581
inflation   L2.tbilrate       -0.5385      0.4205  0.9590          0.865
unemp             const        0.7631      0.8555  0.0925          0.545
unemp      L1.inflation        0.0068     -0.0193 -0.0262          0.179
unemp          L1.unemp        1.3865      1.4675  0.0810          0.654
unemp       L1.tbilrate       -0.1059     -0.1515 -0.0456          0.367
unemp      L2.inflation        0.0253     -0.0225 -0.0478          0.046
unemp          L2.unemp       -0.5096     -0.5637 -0.0541          0.397
unemp       L2.tbilrate        0.0481      0.0988  0.0507          0.698
tbilrate          const        1.0334      0.2126 -0.8209          0.252
tbilrate   L1.inflation        0.0392     -0.0191 -0.0583          0.232
tbilrate       L1.unemp       -0.4101     -0.4663 -0.0562          0.461
tbilrate    L1.tbilrate        0.9832      1.1939  0.2107          0.758
tbilrate   L2.inflation        0.0960      0.0125 -0.0834          0.139
tbilrate       L2.unemp        0.3450      0.4835  0.1386          0.596
tbilrate    L2.tbilrate       -0.2350     -0.2708 -0.0358          0.458
Note: If P(change > 0) is near 0 or 1, the coefficient has moved over the sample: evidence of time variation.

2 of 21 coefficients have P(change > 0) below 0.05 or above 0.95; chance alone would give about 2.1.
```

How to read it

Each row shows a coefficient at the start (1969Q4) and the end (2009Q3), and the posterior probability that it increased. Only **2 of 21** coefficients have a probability below 0.05 or above 0.95, which is what chance alone would give (about 2.1). So the evidence that the conditional means moved is thin. The exception is the second lag of inflation in the inflation equation: it falls from 0.36 to −0.19 (probability of an increase 0.008), and the sum of the two own lags of inflation falls from about 0.5 to about zero, in line with the finding that U.S. inflation became less persistent (Cogley and Sargent 2005); a paper would test the sum directly, from the draws.

Look at the T-bill coefficients in the inflation equation to see why single coefficients mislead: the first lag falls by 0.88 and the second rises by 0.96, with probabilities of 0.17 and 0.87, because the two lags move in opposite directions and their sum barely changes (0.24 at the start, 0.32 at the end). Always judge time variation by sums of lags and by impulse responses, not by one coefficient.

48.5 The coefficient paths

```
oes.plot(r, "Time-varying own first lag", ncols=3, size=(3.6, 2.6))
oes.plot(r, "Time-varying constant", ncols=3, size=(3.6, 2.6))
```

How to read it

The charts show the posterior median path of the T-bill rate's own first lag and of the constant of each equation. The own lag is about 0.98 in 1969, dips to 0.91 around 1979, and rises above one after 1990, reaching 1.19 by 2009. This should not be read as an explosive process: the second lag is about −0.27 at the end, so the sum of the two lags is about 0.92. The constant in the interest-rate equation falls from about 1.0 to about 0.2, as inflation and rates came down. A path of this kind is a statement about the model's best description of each

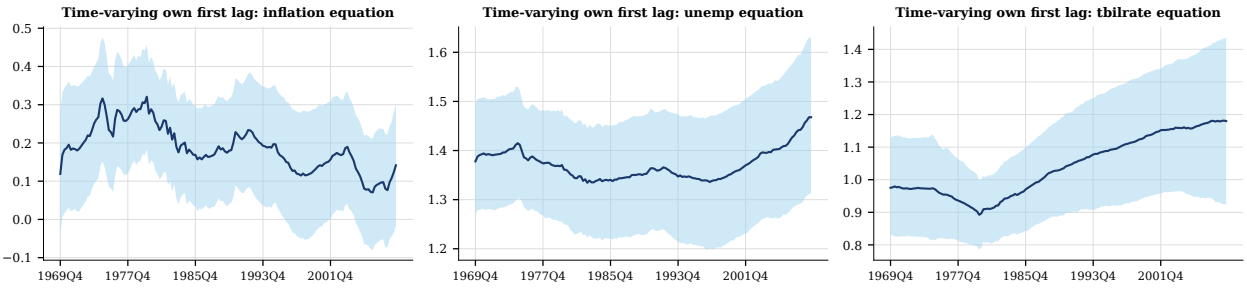


Figure 48.1. The coefficient paths. Panels: Time-varying own first lag: inflation equation; Time-varying own first lag: unemp equation; Time-varying own first lag: tbilrate equation.

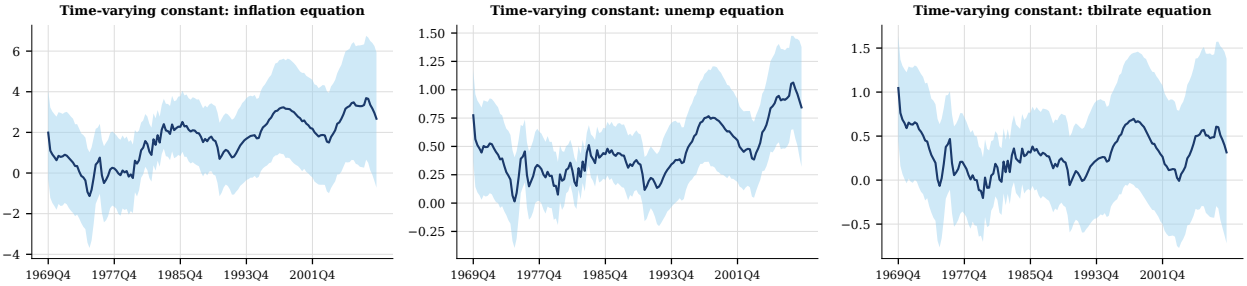


Figure 48.2. The coefficient paths. Panels: Time-varying constant: inflation equation; Time-varying constant: unemp equation; Time-varying constant: tbilrate equation.

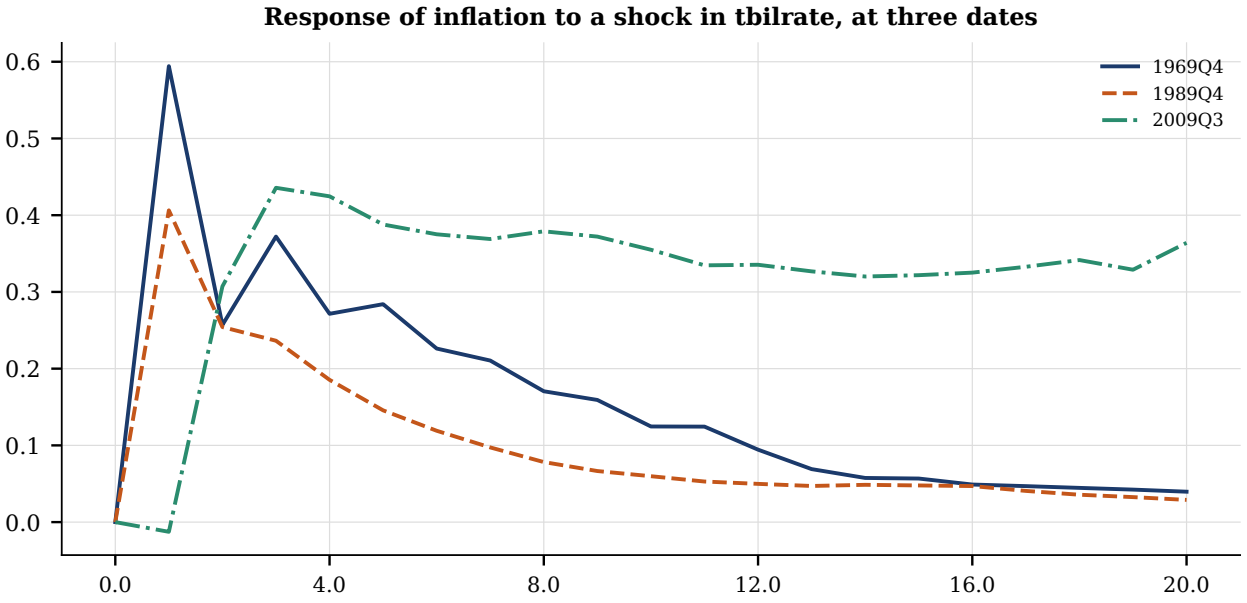


Figure 48.3. Impulse responses at three dates: Response of inflation to a shock in tbilrate, at three dates.

period, estimated with few effective observations at each date: treat its wiggles as noise, and its broad drift as the finding.

48.6 Impulse responses at three dates

```
oes.plot(r, "Response of inflation to a shock in tbilrate", ncols=1, size=(6, 3))
oes.plot(r, "Response of unemp to a shock in tbilrate", ncols=1, size=(6, 3))
```

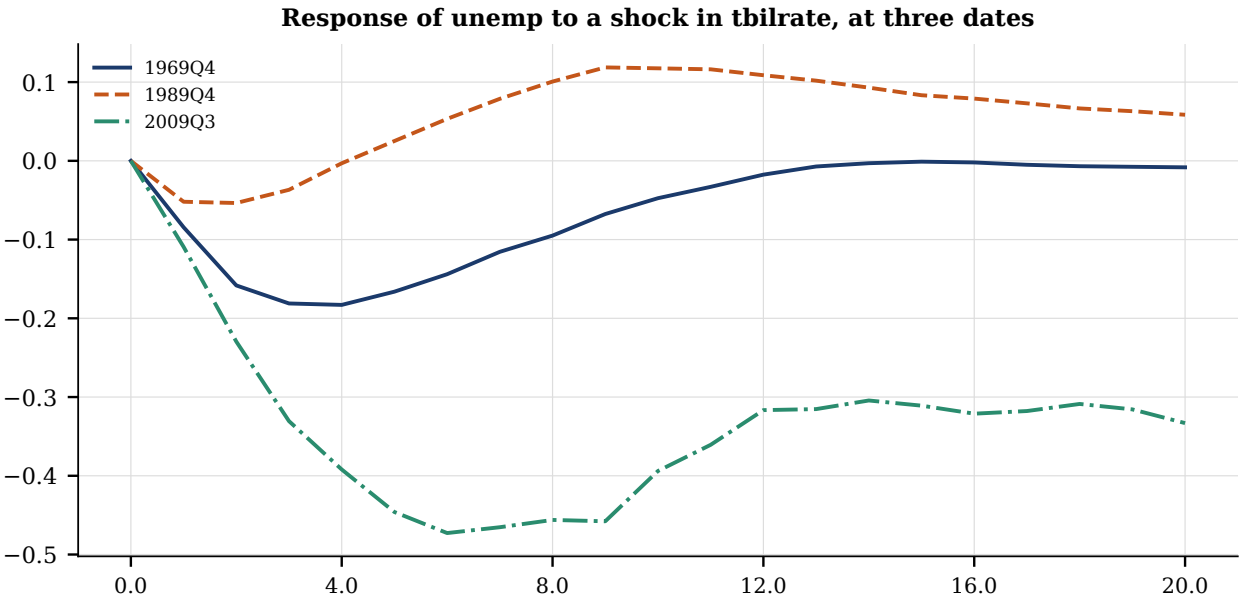


Figure 48.4. Impulse responses at three dates: Response of unemp to a shock in tbilrate, at three dates.

How to read it

The charts show the response of inflation and unemployment to a one-standard-deviation rise in the T-bill rate, using the coefficients of 1969Q4, 1989Q4 and 2009Q3. (The date-specific responses here are posterior medians; compute bands from the draws if the paper relies on the comparison.)

- **Inflation** rises after the shock at all three dates: the price puzzle holds in every period. The response is 0.27 after two quarters in 1969 and 0.24 in 1989, and in 2009 it is delayed (0.02 after one quarter, 0.31 after two) and then more persistent, at 0.34 after eight quarters and 0.22 after twenty, against 0.19 and 0.03 in 1969.
- **Unemployment** falls by about 0.2 point after a year in 1969 and hardly moves in 1989 (0 after four quarters, +0.10 after eight, the expected sign). In 2009 it falls by 0.45 after eight quarters, the wrong sign for a tightening.

The 2009 responses are the least reliable: the policy rate was close to zero, its own lag exceeds one, and a shock to it is a very different object from a shock to the funds rate in 1969. A common remedy is to end the sample before the crisis or to treat it separately; if your sample ends in the crisis, say that the final responses reflect the zero lower bound.

48.7 Convergence and diagnostics

```
oes.show(r, "MCMC")
oes.show(r, "Residual diagnostics")
```

```
MCMC convergence diagnostics
                                Effective sample size  Efficiency (ESS / draws)  Geweke z  p-value
Parameter
unemp: const (last date)                        309.0                     0.052    -1.627   0.1038
unemp: L1.unemp (last date)                     336.0                     0.056    -0.277   0.7819
tbilrate: L1.tbilrate (last date)               235.0                     0.039    -0.501   0.6162
Sigma[inflation,inflation]                     1710.0                     0.285    -0.750   0.4531
Sigma[inflation,unemp]                         2838.0                     0.473    -0.235   0.8143
Sigma[inflation,tbilrate]                      2440.0                     0.407     3.461   0.0005
Sigma[unemp,inflation]                         2838.0                     0.473    -0.235   0.8143
Sigma[unemp,unemp]                             1601.0                     0.267    -1.678   0.0933
Sigma[unemp,tbilrate]                          2237.0                     0.373     0.447   0.6551
Sigma[tbilrate,inflation]                      2440.0                     0.407     3.461   0.0005
Sigma[tbilrate,unemp]                          2237.0                     0.373     0.447   0.6551
Sigma[tbilrate,tbilrate]                       3681.0                     0.613     2.446   0.0144
```

```
Note: Geweke compares the first 10% with the last 50% of the chain: |z| > 1.96 suggests the chain has not settled;
↪ run more draws or a longer burn-in. ESS above about 200 gives reliable posterior means and bands. Lowest efficiency
↪ : 0.039.

Residual diagnostics (posterior-mean coefficient paths)
                              Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   197.8716   0.0000  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, inflation              25.2212   0.0138                                                        H0: no
↪ autocorrelation
Jarque-Bera, inflation            67.0675   0.0000                                                          H0:
↪ normal residuals
ARCH LM, inflation                10.5632   0.0319                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, unemp                  29.4345   0.0034                                                        H0: no
↪ autocorrelation
Jarque-Bera, unemp                24.8573   0.0000                                                          H0:
↪ normal residuals
ARCH LM, unemp                    17.9541   0.0013                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               45.0674   0.0000                                                        H0: no
↪ autocorrelation
Jarque-Bera, tbilrate           1523.8435   0.0000                                                          H0:
↪ normal residuals
ARCH LM, tbilrate                 45.2237   0.0000                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

Convergence of a TVP-VAR is demanding, because there are hundreds of states. With 6,000 draws after 4,000 burn-in, the effective sample sizes of the coefficients at the last date are 235 to 336 (efficiency of 4% to 6%, above 200 but low) and those of the covariance terms 1,600 to 3,700. **Two covariance parameters fail Geweke's test**: $\Sigma_{\text{inflation, T-bill}}$ ($z = 3.5$) and $\Sigma_{\text{T-bill, T-bill}}$ ($z = 2.4$): the chain for the covariance matrix is still drifting. With the default 1,000 draws the sample size for the T-bill coefficient was 26 and the statistic -4.5. In a paper, run at least 15,000 draws with a long burn-in, and report the convergence table, or the posterior means for the covariance matrix are suspect.

Residual diagnostics: serial correlation is rejected for all three variables (Ljung-Box $p = 0.014$, 0.003 and below 0.001), as is normality (Jarque-Bera 1,524 for the interest rate), and so is the absence of ARCH, in all three. The constant covariance matrix is therefore wrong, and the time variation in coefficients may be absorbing changes in volatility: Chapter 47 models these directly, and a paper that claims changing transmission should show that it survives stochastic volatility.

48.8 Robustness: the amount of drift, the training sample and the lags

```python
def path(res, title_start="Time-varying own first lag: tbilrate"):
    c = next(c for c in res["charts"] if c["title"].startswith(title_start))["series"][0]["y"]
    return [round(c[0], 2), round(c[len(c) // 2], 2), round(c[-1], 2)]

rows = {}
for kq in (0.003, 0.01, 0.03):
    rows[f"kQ = {kq}"] = path(oes.run("tvpvar", data, dict(spec, kQ=kq, draws=2000, burn=1500)))
for tr in (30, 50):
    rows[f"training = {tr}"] = path(oes.run("tvpvar", data, dict(spec, training=tr, draws=2000, burn=1500)))
rows["1 lag"] = path(oes.run("tvpvar", data, dict(spec, lags=1, draws=2000, burn=1500)))
print(pd.DataFrame(rows, index=["start", "middle", "end"]).T)
```

```
              start  middle   end
kQ = 0.003     0.98    1.04  1.20
kQ = 0.01      0.97    1.04  1.22
kQ = 0.03      0.95    1.03  1.21
training = 30  0.94    1.09  1.23
training = 50  0.88    0.99  1.09
```

```
1 lag           0.90    0.93  0.91
```

How to read it

The table follows the own first lag of the T-bill rate at the start, the middle and the end of the sample. k_Q (0.003, 0.01, 0.03): the path barely changes (end value 1.20 to 1.22), so the amount of drift permitted by the prior does not drive the result. **Training sample** of 30 or 50 quarters: the end value is 1.23 and 1.09; it is lower with the longer training sample, as the model has less time to drift. **One lag:** the own lag is flat at about 0.9 throughout. The comparison with the two-lag model shows that the rise above one is a feature of the two-lag specification (offset by the second lag), while the persistence of the rate (the sum of the lags) rises from about 0.75 to about 0.92 in the two-lag model and is roughly 0.9 throughout in the one-lag model. The honest summary is that the TVP-VAR finds modest changes in the persistence of the policy rate, and that their size depends on the specification.

48.9 Write the results

How to write it: results

We find limited evidence of time variation in the conditional means. Of the 21 coefficients, only two changed between 1969Q4 and 2009Q3 with posterior probability above 95%, no more than would be expected by chance; the main exception is the persistence of inflation, where the sum of the two own lags falls from about 0.5 to about zero. The policy-rate equation shows a decline in the constant from about 1.0 to 0.2 and a persistence that is higher at the end of the sample than at the start, with a size that depends on the lag length. Inflation rises after a monetary tightening at all three dates studied (1969Q4, 1989Q4, 2009Q3), so the price puzzle is not a feature of one regime. The responses at the end of the sample are affected by the near-zero policy rate and should be read with caution. These results do not depend on the prior scale of the coefficient innovations (k_Q between 0.003 and 0.03). Residual tests reject a constant covariance matrix, so we cannot rule out that changing volatility, not changing coefficients, explains part of the variation; the stochastic-volatility VAR in the next section addresses this. Convergence statistics (Table A) show adequate effective sample sizes for the coefficients but not for two covariance parameters, for which longer chains are needed.

Report	What	Where
Model	Random-walk coefficients, training sample, k_Q, draws	Section 48.1
Evidence	Coefficients at two dates with $P(\text{change} > 0)$; how many pass a threshold	Section 48.4
Results	Paths of key coefficients; impulse responses at selected dates	Sections 48.5 and 48.6
Diagnostics	Effective sample sizes and Geweke statistics; residual tests	Section 48.7
Robustness	k_Q, training sample, lags	Section 48.8
Alternative	Constant-parameter BVAR; SV-VAR	BVAR and Chapter 47

References. Cogley and Sargent (2005), *Review of Economic Dynamics*; Primiceri (2005), *Review of Economic Studies*; Carter and Kohn (1994), *Biometrika*; Sims and Zha (2006), *American Economic Review*; Koop and Korobilis (2010), *Foundations and Trends in Econometrics*; Del Negro and Primiceri (2015), *Review of Economic Studies*.

CHAPTER 49

Bayesian Threshold VAR

Applies when. Dynamics that differ when an observed variable, such as inflation, is above or below a threshold
Overview. Use this chapter when the economy behaves differently in two states that you can observe, such as high and low inflation, and you want to know where the threshold is and whether the two-regime model beats a single linear VAR. The Bayesian threshold VAR finds the threshold and the delay with an exact posterior, and the Bayes factor tells you whether the nonlinearity is needed. The chapter covers choosing the threshold variable, estimation, regime dates, responses by regime, robustness and the write-up.
Data. U.S. quarterly GDP growth, inflation and the T-bill rate, 1959–2009
Software. Python · OES engine · statsmodels

49.1 The question and the model

Does monetary policy work differently when inflation is high? A threshold VAR (Tsay 1998; Koop and Potter 2006) splits the sample into two regimes according to whether a *threshold variable* z_{t-d} lies above a threshold r:

$$y_t = \begin{cases} c_1 + A_{11}y_{t-1} + \cdots + A_{1p}y_{t-p} + u_t, \; u_t \sim N(0, \Sigma_1), & z_{t-d} \le r, \\ c_2 + A_{21}y_{t-1} + \cdots + A_{2p}y_{t-p} + u_t, \; u_t \sim N(0, \Sigma_2), & z_{t-d} > r. \end{cases}$$

The coefficients and the covariance matrix differ between regimes. The threshold r and the delay d are unknown. In the Bayesian version used here, each regime has a Normal-Wishart prior (Minnesota-type moments), so its marginal likelihood is available in closed form and the **posterior of** (r, d) **is exact**, computed over a grid of candidate thresholds. There is nothing to simulate for the threshold, so no convergence to check; coefficients are then drawn within each regime.

Two checks before you trust a threshold model

1. **Does the data prefer two regimes?** The log Bayes factor of the threshold model against the linear VAR answers this directly. A threshold VAR can always fit better than a linear VAR; the Bayes factor penalises the extra parameters. If it is negative, use the linear VAR.
2. **Is the threshold variable the right one?** Different candidates give very different Bayes factors (Section 49.3). Choose by the Bayes factor, and say that you did: the choice is part of the model.

Also check that each regime has enough observations: with two lags of three variables, a regime has 21 coefficients per equation, so a regime of 30 quarters is thin. The prior helps but results for the small regime are imprecise.

How to write it: the model

We estimate a Bayesian threshold VAR (Tsay 1998; Koop and Potter 2006) in GDP growth, inflation and the three-month Treasury bill rate, with two lags. The regime is determined by whether inflation, d quarters earlier, exceeds a threshold r. Each regime has its own coefficients and covariance matrix, with a Normal-Wishart prior with Minnesota moments (overall tightness 0.2). Because the marginal likelihood is available analytically in each regime, the joint posterior of the threshold (over a 40-point grid between the 15th and 85th percentiles of the threshold variable) and the delay (one or two quarters) is computed exactly. We compare the threshold model with a linear VAR using the log Bayes factor, and report results for the posterior mode of the threshold.

49.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
vars_ = ["gdp_growth", "inflation", "tbilrate"]
print(df[vars_].describe().round(2).loc[["mean", "std", "min", "max"]])
spec = dict(vars=vars_, threshold_var="inflation", time="date", lags=2, max_delay=2, trim=0.15, grid=40,
  ↪ lambda1=0.2, horizon=20, draws=1000)
r = oes.run("tvar", data, spec)
oes.show(r, "Unit-root")
```

```
      gdp_growth  inflation  tbilrate
mean        3.10       3.98      5.32
std         3.52       3.25      2.80
min        -8.28      -8.79      0.12
max        15.43      14.62     15.33
Unit-root tests
            ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
gdp_growth                  0.0000               0.0000                      0.0         0.1000       I(0)
inflation                   0.0271               0.0822                      0.0         0.0808       I(0)
tbilrate                    0.2805               0.4462                      0.0         0.0538       I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

Inflation and GDP growth reject a unit root; the T-bill rate does not ($p = 0.28$). The regimes are defined by a *stationary* variable, which is what a threshold model needs: a trending threshold variable would spend the whole sample in one regime. Quarterly annualised inflation is volatile (it ranges widely), so the high regime also catches quarters of temporary price spikes, not only the high-inflation years.

49.3 Do you need a threshold, and on which variable?

Run the model with each candidate threshold variable and compare the log Bayes factor against the linear VAR.

```
def threshold_summary(res):
    t = oes.frame(next(t for t in res["tables"] if t["title"].startswith("Threshold and regimes")))["Value"]
    return t

rows = []
for z in ["gdp_growth", "inflation", "tbilrate", "unemp", "cons_growth", "inv_growth", "m1_growth"]:
    t = threshold_summary(oes.run("tvar", data, dict(spec, threshold_var=z)))
    rows.append([z, round(t["Threshold r"], 2), int(t["Observations: low regime"]), int(t["Observations: high
  ↪ regime"]),
                 round(t["Log Bayes factor, threshold vs linear VAR"], 1)])
print(pd.DataFrame(rows, columns=["threshold variable", "threshold r", "n low", "n high", "log Bayes factor"]).
  ↪ set_index("threshold variable"))
```

```
                  threshold r  n low  n high  log Bayes factor
threshold variable
gdp_growth               1.61     58     142              -4.0
inflation                6.65    170      30              26.4
tbilrate                 6.82    152      48              16.8
unemp                    6.20    136      64              -8.4
cons_growth              1.66     48     152              -3.7
inv_growth              -6.77     47     153              -2.9
m1_growth                1.66     52     148               1.0
```

How to read it

A positive log Bayes factor favours the threshold model; above 4.6 is "decisive" on Jeffreys' scale. **Inflation** (26.4) and the **T-bill rate** (16.8) are strongly supported; GDP growth (−4.0), consumption growth (−3.7), unemployment (−8.4) and investment growth (−2.9) are not (the linear VAR is preferred), and money growth is at zero (1.0). So the economy's dynamics depend on the level of inflation or interest rates, not on the stage of the business cycle measured by growth. Inflation is the natural choice for the question, and its Bayes factor is the largest; we use it below. This is a model choice made on the data: report the table.

Caution: a Bayes factor in favour of a regime split can also reflect different *volatility* in the two regimes (the 1970s were more volatile than the 1990s). Compare with Chapter 47, which allows changing volatility without splitting the coefficients.

49.4 Estimate

```
oes.summary(r)
oes.show(r, "Posterior of the delay")
oes.show(r, "Threshold and regimes")
```

```
Observations: 200
Threshold: 6.6467
Delay: 2
High-regime share: 0.15
Lags: 2
Posterior of the delay
         Posterior probability
Delay d
1                          0.0
2                          1.0

Threshold and regimes (at the posterior mode)
                                            Value
Item
Threshold r                                6.6467
Posterior mean of r                        6.6459
Delay d                                    2.0000
Observations: low regime                 170.0000
Observations: high regime                 30.0000
Log Bayes factor, threshold vs linear VAR  26.4090
Note: Regime is high when inflation(t - d) > r. Bayes factor averages the marginal likelihood over the grid (uniform
↪ prior).
```

How to read it

The data choose a **threshold of 6.65%** annualised quarterly inflation, with a delay of two quarters (the posterior probability of a delay of one quarter is below 0.0001): the economy is in the high regime when inflation two quarters earlier exceeded 6.65%. That regime holds 30 of the 200 quarters (15%) and the low regime 170. The log Bayes factor against a single linear VAR is **26.4**, overwhelming support for two regimes (on Jeffreys' scale, above 4.6 is decisive). The posterior mean of the threshold (6.646) equals the mode (6.647): with a finer grid of 100 points, 95% of the posterior mass sits on 6.65 and 3% on 6.58. Do not over-read this precision: it arises because moving a single quarter between regimes changes the marginal likelihood a great deal when the high regime has only 30 quarters. Section 49.9 shows how the threshold moves when the minimum regime size changes.

49.5 Regime dates

```
oes.show(r, "Regime dating")
oes.plot(r, "inflation(t-2) and the threshold", ncols=1, size=(6.5, 3))
```

```
Regime dating (regime 0 = low, regime 1 = high)
            From      To  Periods  Average probability
Regime
Regime 0  1959Q4  1969Q2       39                  1.0
Regime 1  1969Q3  1969Q3        1                  1.0
Regime 0  1969Q4  1973Q2       15                  1.0
```

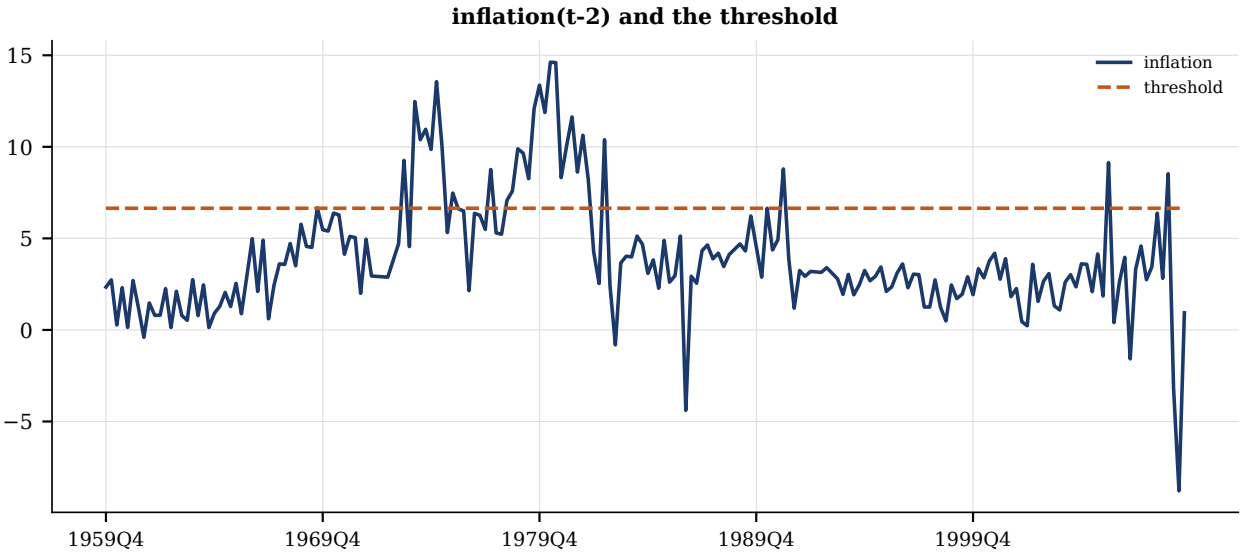


Figure 49.1. Regime dates: inflation(t-2) and the threshold.

```
Regime 1  1973Q3  1973Q3        1                1.0
Regime 0  1973Q4  1973Q4        1                1.0
Regime 1  1974Q1  1975Q2        6                1.0
Regime 0  1975Q3  1975Q3        1                1.0
Regime 1  1975Q4  1975Q4        1                1.0
Regime 0  1976Q1  1977Q2        6                1.0
Regime 1  1977Q3  1977Q3        1                1.0
Regime 0  1977Q4  1978Q1        2                1.0
Regime 1  1978Q2  1982Q1       16                1.0
Regime 0  1982Q2  1982Q3        2                1.0
Regime 1  1982Q4  1982Q4        1                1.0
Regime 0  1983Q1  1990Q4       32                1.0
Regime 1  1991Q1  1991Q1        1                1.0
Regime 0  1991Q2  2005Q4       59                1.0
Regime 1  2006Q1  2006Q1        1                1.0
Regime 0  2006Q2  2008Q3       10                1.0
Regime 1  2008Q4  2008Q4        1                1.0
Regime 0  2009Q1  2009Q3        3                1.0
```

How to read it

Because the regime is a function of an observed variable, regime membership is known once the threshold is fixed, so every probability is 1. The 30 high-inflation quarters fall into two real episodes and eight isolated quarters. The episodes are 1974Q1–1975Q2 (the first oil shock) and **1978Q2–1982Q1** (the second oil shock and the Volcker disinflation, 16 quarters). The single quarters are 1969Q3, 1973Q3, 1975Q4, 1977Q3, 1982Q4, 1991Q1, 2006Q1 and 2008Q4: spikes in the quarterly CPI, some from energy prices, not episodes of sustained inflation. In a paper, describe the high regime as largely the oil-shock years, and note that it is close to a period dummy for the 1970s, which limits what can be inferred about "high inflation" in general.

49.6 What differs between the regimes?

```
oes.show(r, "Low regime, equation: tbilrate")
oes.show(r, "High regime, equation: tbilrate")
```

```
Low regime, equation: tbilrate
                 Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const          0.1469     0.1267 -0.0972  0.1411  0.3981   0.876
L1.gdp_growth  0.0455     0.0138  0.0165  0.0463  0.0715   1.000
L1.inflation  -0.0026     0.0202 -0.0437 -0.0023  0.0379   0.454
L1.tbilrate    0.9102     0.0546  0.8042  0.9106  1.0187   1.000
L2.gdp_growth  0.0202     0.0119 -0.0039  0.0207  0.0422   0.950
```

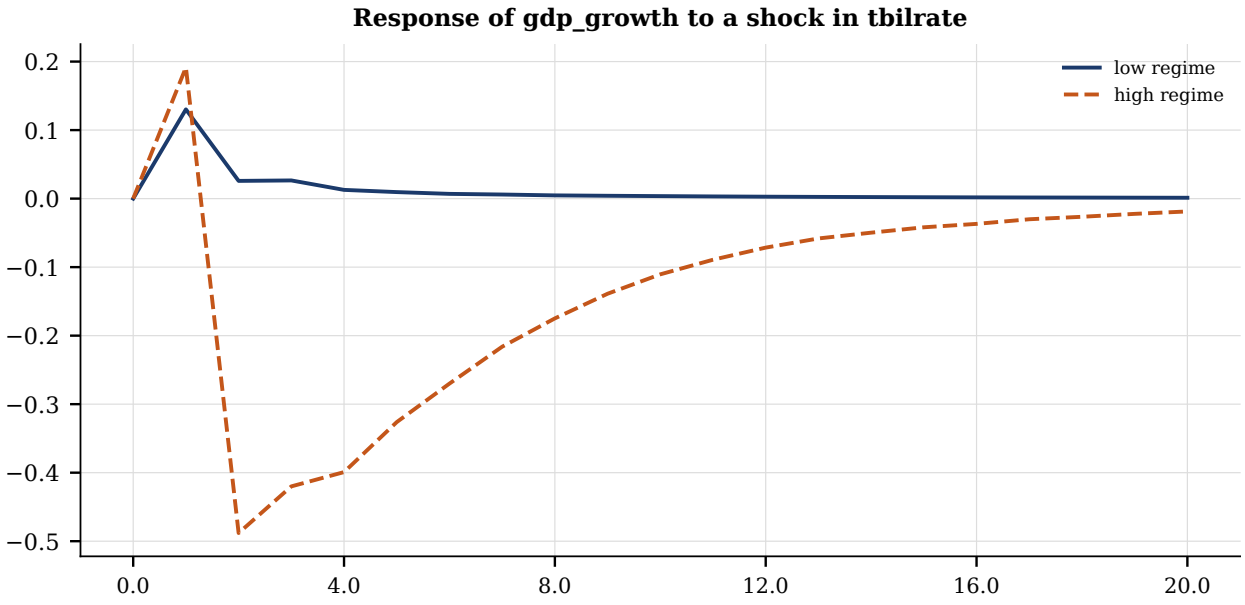


Figure 49.2. Impulse responses by regime: Response of gdp_growth to a shock in tbilrate.

```
L2.inflation   0.0259     0.0189 -0.0099  0.0257  0.0630   0.919
L2.tbilrate   -0.0109     0.0516 -0.1096 -0.0125  0.0947   0.402

High regime, equation: tbilrate
                 Mean  Std. dev.    2.5%  Median   97.5%  P(> 0)
Parameter
const          1.2060     1.1302 -1.0087  1.1844  3.5095   0.870
L1.gdp_growth  0.0155     0.0605 -0.1009  0.0148  0.1379   0.597
L1.inflation   0.0589     0.0845 -0.1094  0.0563  0.2190   0.768
L1.tbilrate    0.7055     0.1620  0.3803  0.7029  1.0162   1.000
L2.gdp_growth  0.0131     0.0441 -0.0737  0.0137  0.1019   0.623
L2.inflation  -0.0297     0.0711 -0.1591 -0.0317  0.1073   0.327
L2.tbilrate    0.1440     0.1429 -0.1238  0.1426  0.4321   0.848
```

How to read it

The policy-rate equation shows what changes. In the low regime the rate follows the economy: it responds to lagged GDP growth (0.046, 95% interval 0.016 to 0.072; probability of a positive sign 1.00) and is very persistent (own lag 0.91). In the high regime the response to growth is small and imprecise (0.015, interval −0.10 to 0.14), the response to inflation is 0.059 (−0.11 to 0.22), and the own lag is lower (0.71, interval 0.38 to 1.02). With only 30 quarters, the intervals are wide: the data cannot say that the rule changed, only that the low-regime rule is estimated precisely and the high-regime rule is not. Report this limitation rather than reading the point estimates as a regime-dependent policy rule.

49.7 Impulse responses by regime

```
oes.plot(r, "Response of gdp_growth to a shock in tbilrate", ncols=1, size=(6, 3))
oes.plot(r, "Response of inflation to a shock in tbilrate", ncols=1, size=(6, 3))
```

How to read it

Responses to a one-standard-deviation rise in the T-bill rate, within each regime (the chart shows posterior medians without credible bands). **Output:** in the low-inflation regime GDP growth does not respond (+0.01 after a year); in the high-inflation regime it falls by 0.40 percentage point after a year, 0.18 after two and 0.07 after three. **Inflation:** in the low regime it rises (0.15 after a year, 0.11 after two): the price puzzle of Chapter 42; in the high regime there is no price puzzle (−0.006 after a year and −0.06 after two). So the textbook sign of monetary transmission appears in the high regime, which is the regime of the oil-shock years,

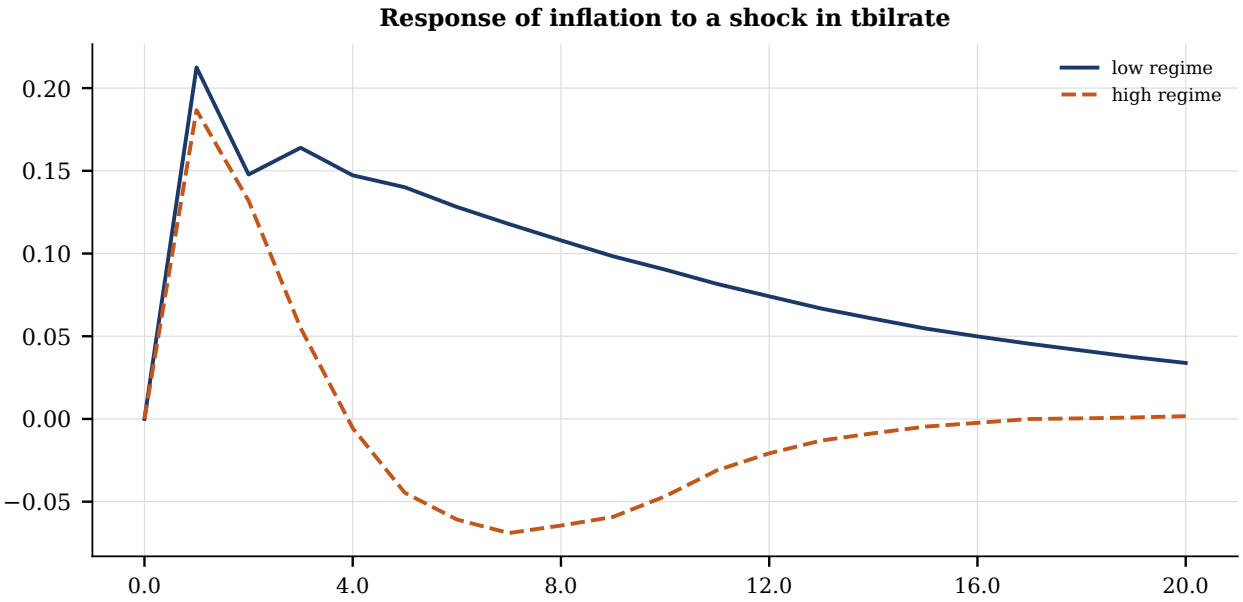


Figure 49.3. Impulse responses by regime: Response of inflation to a shock in tbilrate.

and the price puzzle in the low regime. Because the high regime has 30 observations, do not report these differences as significant without computing credible bands from the draws; the Bayes factor in Section 49.3 says that the regimes differ, not that each difference is large.

49.8 Diagnostics

```
oes.show(r, "Residual diagnostics, low")
oes.show(r, "Residual diagnostics, high")
```

```
Residual diagnostics, low regime (posterior mean)
                               Statistic  p-value
↪       Detail
Test
Portmanteau (Hosking), 12 lags   127.0303   0.1020  df = 108; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth             12.5846   0.3999                                                          H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth            3.6170   0.1639                                                            H0:
↪ normal residuals
ARCH LM, gdp_growth               10.3668   0.0347                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation              39.2313   0.0001                                                          H0: no
↪ autocorrelation
Jarque-Bera, inflation            80.3909   0.0000                                                            H0:
↪ normal residuals
ARCH LM, inflation                 4.5236   0.3398                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               22.3379   0.0339                                                          H0: no
↪ autocorrelation
Jarque-Bera, tbilrate            173.5066   0.0000                                                            H0:
↪ normal residuals
ARCH LM, tbilrate                  3.8023   0.4334                                          H0: no conditional
↪ heteroskedasticity

Residual diagnostics, high regime (posterior mean)
                              Statistic  p-value
↪     Detail
Test
Portmanteau (Hosking), 4 lags    34.1688   0.5559  df = 36; H0: no residual autocorrelation (approximate after
↪ estimating lags)
```

```
Ljung-Box, gdp_growth             1.5957   0.8096                                          H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           1.9981   0.3682                                            H0: normal
↪  residuals
ARCH LM, gdp_growth               1.0273   0.7946                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation             11.9657   0.0176                                          H0: no
↪ autocorrelation
Jarque-Bera, inflation            0.2579   0.8790                                            H0: normal
↪  residuals
ARCH LM, inflation                5.4662   0.1407                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, tbilrate               4.0302   0.4019                                          H0: no
↪ autocorrelation
Jarque-Bera, tbilrate             7.7219   0.0210                                            H0: normal
↪  residuals
ARCH LM, tbilrate                 7.0021   0.0718                          H0: no conditional
↪ heteroskedasticity
```

How to read it

In the low regime (170 quarters) the joint portmanteau test does not reject (p = 0.10), but Ljung-Box rejects for inflation (p = 0.0001) and the interest rate (p = 0.034), normality is rejected for both, and the ARCH test rejects for GDP growth (p = 0.035). In the high regime (30 quarters) Ljung-Box rejects for inflation (p = 0.018) and nothing else does, but these tests have little power with 30 observations, so the absence of rejections says little. Remaining autocorrelation in inflation suggests more lags; Section 49.9 shows that the threshold and the Bayes factor survive three and four lags. Changing volatility between the regimes is not modelled beyond the regime-specific covariance matrix; if it matters to your question, see Chapter 47.

49.9 Robustness: lags, trimming, grid and prior

```
def row(**kw):
    t = threshold_summary(oes.run("tvar", data, dict(spec, **kw)))
    return [round(t["Threshold r"], 2), int(t["Observations: high regime"]), round(t["Log Bayes factor,
  ↪ threshold vs linear VAR"], 1)]

variants = {"baseline": {}, "1 lag": dict(lags=1), "3 lags": dict(lags=3), "4 lags": dict(lags=4),
            "min regime share 10%": dict(trim=0.10), "min regime share 25%": dict(trim=0.25),
            "grid of 100": dict(grid=100), "lambda1 = 0.1": dict(lambda1=0.1), "lambda1 = 0.5": dict(lambda1
  ↪ =0.5), "delay up to 4": dict(max_delay=4)}
print(pd.DataFrame({k: row(**v) for k, v in variants.items()}, index=["threshold r", "n high", "log Bayes
  ↪ factor"]).T)
```

```
                     threshold r  n high  log Bayes factor
baseline                    6.65    30.0              26.4
1 lag                       6.65    30.0              31.8
3 lags                      6.65    30.0              22.9
4 lags                      6.65    30.0              23.9
min regime share 10%        7.05    29.0              27.7
min regime share 25%        5.00    50.0              16.7
grid of 100                 6.65    30.0              25.5
lambda1 = 0.1               6.65    30.0               5.4
lambda1 = 0.5               6.65    30.0              45.7
delay up to 4               6.65    30.0              26.3
```

How to read it

Each row changes one thing. **Lags:** with one to four lags the threshold is 6.65 and the Bayes factor is 31.8, 26.4, 22.9 and 23.9. **Minimum regime share:** allowing 10% in each regime moves the threshold to 7.05 (29 high quarters); requiring 25% moves it to 5.0 (50 high quarters), with a Bayes factor of 16.7: the threshold is sensitive to how small the high regime may be, so report the trimming. **Grid and delay:** a grid of 100 points and a maximum delay of four quarters change nothing (the delay stays at two). **Prior:** the Bayes factor ranges from 5.4 ($\lambda_1 = 0.1$) to 45.7 ($\lambda_1 = 0.5$). Marginal likelihoods depend on the prior, so the *size* of the Bayes

factor is not robust, but it is positive and large in every variant, so the conclusion that two regimes beat one is. Report a range, not a single number.

49.10 Write the results

How to write it: results

We compare a Bayesian threshold VAR with a linear VAR using the log Bayes factor, for seven candidate threshold variables. The data support inflation (log Bayes factor 26.4) and the Treasury bill rate (16.8) as threshold variables; GDP growth, consumption growth, unemployment and investment growth are not supported (log Bayes factors between −8.4 and −2.9). With inflation as the threshold variable the posterior concentrates on a delay of two quarters and a threshold of 6.65%, which defines a high-inflation regime of 30 quarters, 22 of them in the oil-shock episodes of 1974–75 and 1978–82. In the low-inflation regime, a rise in the policy rate does not reduce output and raises inflation (the price puzzle); in the high-inflation regime output falls by 0.4 percentage point after a year and inflation does not rise. These regime differences are estimated with 30 observations and are imprecise. The threshold, the delay and the sign of the Bayes factor do not change with one to four lags, the grid, or the prior tightness; the threshold moves between 5.0 and 7.05 depending on the minimum regime size (10% to 25%), and the Bayes factor ranges from 5.4 to 45.7 with the prior.

Report	What	Where
Model	Threshold variable, delay and threshold grid, prior, minimum regime share	Section 49.1
Choice	Log Bayes factors for the candidate threshold variables; linear VAR comparison	Section 49.3
Threshold	Posterior mode and mean, regime sizes, regime dates	Sections 49.4 and 49.5
Differences	Regime-specific coefficients; responses by regime	Sections 49.6 and 49.7
Diagnostics	Residual tests by regime	Section 49.8
Robustness	Lags, trimming, grid, prior, delay	Section 49.9

References. Tsay (1998), *Journal of the American Statistical Association*; Koop and Potter (2006), *Econometrics Journal*; Balke (2000), *Review of Economics and Statistics*; Hubrich and Teräsvirta (2013), *Journal of Economic Surveys*; Kass and Raftery (1995), *Journal of the American Statistical Association*; Chen and Lee (1995), *Journal of Business & Economic Statistics*.

PART VII

Markov-Switching Models

Overview

Some economic relationships change, and what changes is not observed. Recessions and expansions, calm and turbulent markets, high- and low-inflation regimes: the econometrician sees the data and not the regime. The Markov-switching model of Hamilton (1989) treats the regime as a latent variable that follows a Markov chain, estimates the probability of each regime in every period, and lets the data say how many regimes there are and how long each lasts. This distinguishes it from the threshold models of Part II, where an observed variable defines the regime.

The four chapters build up from one series to a regression, a VAR and a panel. Throughout, three questions recur: whether the data support more than one regime at all (the likelihood-ratio test has a non-standard distribution, Garcia, 1998), what switches, and whether a solution is a real regime or a collapsed one that explains a few outliers.

Chapters at a glance

Ch.	Chapter	Use it when
50	Markov-Switching Autoregression (Hamilton)	A single series whose mean shifts between expansion-like and recession-like states that you do not observe
51	Markov-Switching Regression	A regression whose intercept, slopes or error variance shift between hidden regimes, such as a Phillips curve in calm and turbulent periods
52	Markov-Switching VAR	Several macro variables whose means, volatility or dynamics shift between hidden regimes, such as calm and turbulent periods
53	Markov-Switching Panel VAR	A panel of countries or units that share the same unobserved regimes, such as a global slowdown that hits every country at once

Further reading

The foundational paper is Hamilton (1989); Hamilton (1994) gives the full treatment, and Kim and Nelson (1999) is the standard book on state-space models with regime switching. Markov-switching VARs are treated in Krolzig (1997).

CHAPTER 50

Markov-Switching Autoregression (Hamilton)

Applies when. A single series whose mean shifts between expansion-like and recession-like states that you do not observe

Overview. Use this chapter when you have one time series, such as GDP growth, that seems to move between a fast-growth state and a slow or negative-growth state, and you want the data to date the states and tell you how long they last. It is Hamilton's (1989) model. The chapter covers the unit-root check, choosing the number of regimes, the linearity test, estimation, regime dates and a comparison with NBER recession dates, diagnostics, robustness and the write-up.

Data. U.S. quarterly GNP growth, 1951Q2–1984Q4 (Hamilton 1989)

Software. Python · OES engine · statsmodels

50.1 The question and the model

Are there recessions in the data, and can the data find them without being told? In a Markov-switching autoregression (Hamilton 1989) the mean of the series depends on an unobserved state s_t:

$$y_t - \mu_{s_t} = \phi_1(y_{t-1} - \mu_{s_{t-1}}) + \cdots + \phi_p(y_{t-p} - \mu_{s_{t-p}}) + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \sigma^2),$$

and the state follows a first-order Markov chain with transition probabilities $p_{ij} = \Pr(s_t = j \mid s_{t-1} = i)$. The model is estimated by maximum likelihood with the Hamilton filter. For each period it gives a *filtered* probability (using data up to t) and a *smoothed* probability (using the whole sample) of being in each state.

Three checks before you trust a regime model

1. **Is the series stationary?** The regimes should describe shifts in the mean of a stationary series. A trending series produces "regimes" that are just the trend.
2. **Does the data need two regimes?** Compare one, two and three regimes with information criteria, and run the linearity test. A regime model fits better than a linear one by construction; the question is whether it fits *enough* better.
3. **Are the regimes sharply identified?** A good model puts each period in one regime with high probability. The regime classification measure (RCM, Ang and Bekaert 2002) summarises this.

How to write it: the model

We estimate Hamilton's (1989) Markov-switching autoregression for quarterly GNP growth, with a state-dependent mean, a common variance and four autoregressive lags. The state follows a two-state first-order Markov chain. We estimate the model by maximum likelihood using the Hamilton filter and report smoothed state probabilities (Kim 1994). To compare the number of regimes we use AIC and BIC, and we test linearity with the likelihood-ratio statistic and the critical values of Garcia (1998) and the bound of Davies (1987), because the standard asymptotic distribution does not apply when parameters are unidentified under the null.

50.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
```

```
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("hamilton_gnp.csv").read()
df = pd.read_csv("hamilton_gnp.csv")
print(df.describe().round(2).loc[["count", "mean", "std", "min", "max"]])
spec = dict(y="gnp_growth", time="date", order=4, regimes=2, reference="nber_recession")
r = oes.run("ms_ar", data, spec)
oes.show(r, "Unit-root")
```

```
       gnp_growth  nber_recession
count      135.00           135.0
mean         0.74             0.2
std          1.07             0.4
min         -2.39             0.0
max          3.11             1.0
Unit-root tests
                    ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
dependent variable                     0.0                  0.0                      0.0            0.1       I(0)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

Quarterly GNP growth is stationary: the ADF tests reject a unit root and the KPSS test does not reject stationarity. That is what a regime-in-the-mean model needs. If your series is in levels and trending, difference it (growth rates) first.

50.3 How many regimes? Is the series linear?

```
oes.show(r, "Number of regimes")
oes.show(r, "Linearity")
```

```
Number of regimes: information criteria
         Log-likelihood      AIC      BIC     HQIC
Regimes
1              -183.669  379.338  396.589  386.348
2              -181.263  380.527  406.404  391.042
3              -179.184  386.368  426.621  402.725
Note: Regimes = 1 is the linear model. Under the null of no switching some parameters are unidentified, so a standard
↪  LR test is not valid (Hansen 1992; Garcia 1998): compare models with the criteria and with the economic meaning
↪ of the regimes.

Linearity: likelihood-ratio statistic
                                                                    Value
Statistic
LR = 2 (logL switching - logL linear)                               4.812
Garcia (1998) 5% critical value, two-regime models (approximate)  about 14
Davies upper-bound p-value                                        0.9501
Note: A large LR (well above the Garcia critical value) rejects the linear model.
```

How to read it

One regime has the lowest AIC (379.3) and BIC (396.6); two regimes give 380.5 and 406.4, three give 386.4 and 426.6. The information criteria do **not** prefer a switching model for this sample. The linearity test agrees: the statistic is 4.8, far below the Garcia 5% critical value of about 14, and the Davies bound gives $p = 0.95$. So the data **do not reject** a linear AR model. This is an honest and well-known result for Hamilton's sample: with 131 quarters, the evidence for regimes is weak when judged by the likelihood alone. The regimes are still useful as a description, because they line up with the NBER dates (Section 50.5), but you should not claim that the test proves two regimes. Report this table.

50.4 Estimate

```
oes.show(r, "Parameter estimates")
oes.show(r, "Transition")
oes.show(r, "Regimes")
oes.show(r, "Regime classification")
```

```
Parameter estimates
           Estimate  Std. err.        z  p-value
Parameter
p[0->0]      0.7547     0.0965   7.8186   0.0000
p[1->0]      0.0959     0.0377   2.5417   0.0110
const[0]    -0.3588     0.2645  -1.3563   0.1750
const[1]     1.1635     0.0745  15.6144   0.0000
sigma2       0.5914     0.1026   5.7614   0.0000
ar.L1        0.0135     0.1200   0.1123   0.9106
ar.L2       -0.0575     0.1377  -0.4179   0.6760
ar.L3       -0.2470     0.1069  -2.3103   0.0209
ar.L4       -0.2129     0.1105  -1.9264   0.0540
Note: p[i->j] is the probability of moving from regime i to regime j; [r] marks a regime-specific parameter.

Transition probabilities (row: from, column: to)
          Regime 0  Regime 1
From
Regime 0    0.7547    0.2453
Regime 1    0.0959    0.9041

Regimes
          Expected duration (periods)  Share of periods (smoothed prob. > 0.5)  Mean of gnp_growth in the regime
Regime
Regime 0                        4.076                                    0.275                          -0.4157
Regime 1                       10.426                                    0.725                           1.1501

Regime classification
                                                          Value
Measure
Regime classification measure, RCM (Ang and Bekaert 2002)  21.67
Note: RCM = 0: every period is assigned to a regime with certainty; RCM = 100: no information. Below about 30 is a
↪ sharp classification.
```

How to read it

Regime 1 is the high-growth state: the mean growth is 1.15% per quarter (about 4.6% a year) and it lasts 10.4 quarters on average. Regime 0 is the low-growth state: the mean is −0.42% per quarter, it lasts 4.1 quarters on average, and the state occupies 27.5% of the sample. The transition probabilities are $p_{00} = 0.75$ and $p_{11} = 0.90$; the expected duration of a regime is $1/(1 - p_{ii})$. The low-growth mean is imprecise ($z = -1.4$, p = 0.18) while the high-growth mean is very precisely estimated ($z = 15.6$): the recession state has few observations. The estimate $p_{10} = 0.096$ is significant (p = 0.011), so a move from expansion to the low state is likely about once in ten quarters. Only the third lag of growth is significant at 5% (−0.25). The RCM is 21.7, below the usual threshold of 30, so the periods are assigned to regimes fairly sharply.

50.5 Regime dates and the NBER recession dates

```
oes.show(r, "Regime dating")
oes.show(r, "Comparison")
oes.plot(r, "Smoothed regime probabilities", ncols=1, size=(6.5, 3))
oes.plot(r, "gnp_growth and its regime-dependent fitted mean", ncols=1, size=(6.5, 3))
```

```
Regime dating (most likely regime, consecutive periods)
            From      To  Periods  Average probability
Regime
Regime 1  1952Q2  1953Q2        5                0.891
Regime 0  1953Q3  1954Q2        4                0.934
Regime 1  1954Q3  1956Q4       10                0.882
Regime 0  1957Q1  1958Q1        5                0.932
Regime 1  1958Q2  1960Q1        8                0.959
Regime 0  1960Q2  1960Q4        3                0.899
Regime 1  1961Q1  1969Q2       34                0.960
```

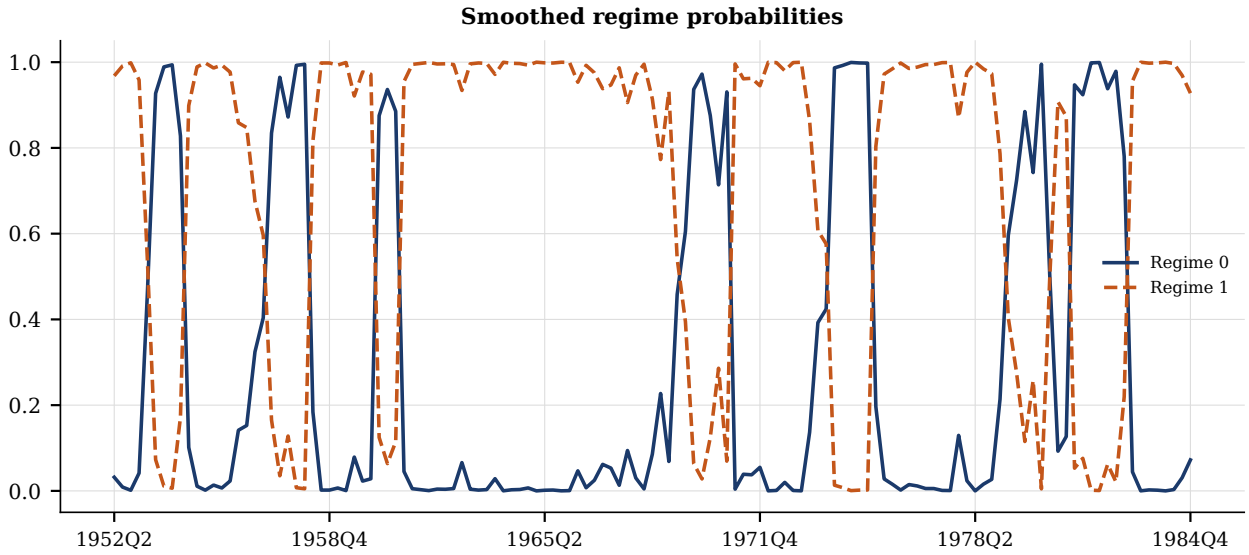


Figure 50.1. Regime dates and the NBER recession dates: Smoothed regime probabilities.

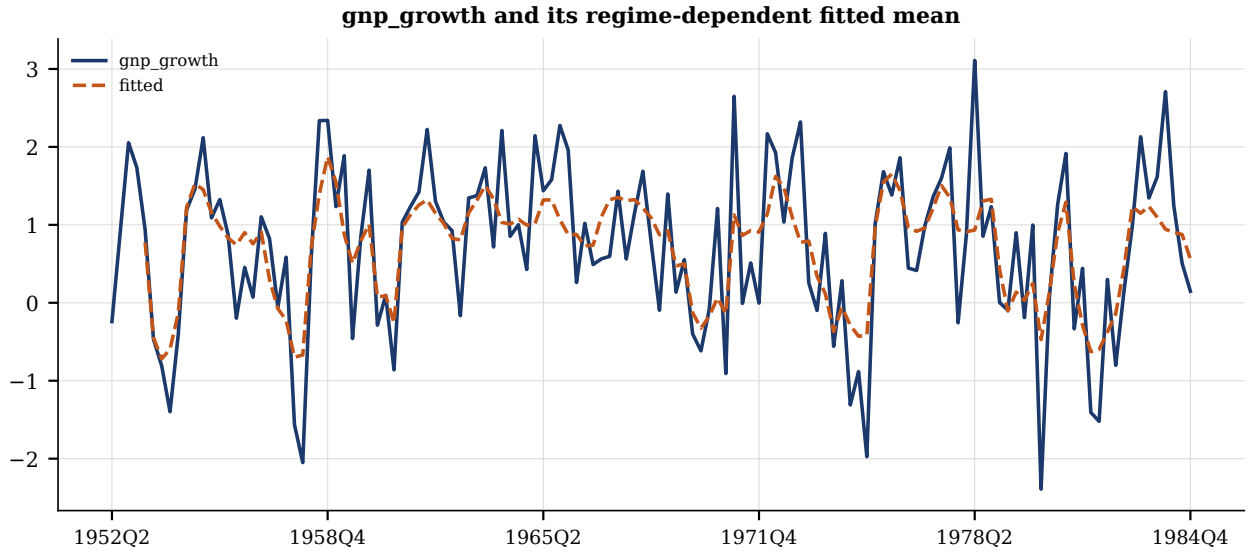


Figure 50.2. Regime dates and the NBER recession dates: gnp_growth and its regime-dependent fitted mean.

```
Regime 0  1969Q3  1970Q4        6                0.839
Regime 1  1971Q1  1973Q4       12                0.907
Regime 0  1974Q1  1975Q1        5                0.995
Regime 1  1975Q2  1979Q1       16                0.957
Regime 0  1979Q2  1980Q3        6                0.742
Regime 1  1980Q4  1981Q1        2                0.890
Regime 0  1981Q2  1982Q4        7                0.938
Regime 1  1983Q1  1984Q4        8                0.981

Comparison with nber_recession (regime 0)
                                                 Value
Measure
Concordance (share of periods classified alike)  0.916
Reference periods detected                       0.963
False signals (share of non-reference periods)   0.096
```

How to read it

The low-growth regime is dated 1953Q3–1954Q2, 1957Q1–1958Q1, 1960Q2–1960Q4, 1969Q3–1970Q4, 1974Q1–1975Q1, 1979Q2–1980Q3 and 1981Q2–1982Q4, mostly with an average probability above 0.9. These are the post-war US recessions. Against the NBER recession indicator, 91.6% of quarters are classified alike, 96.3% of NBER recession quarters are detected, and 9.6% of non-recession quarters are flagged as low growth (false signals). The model finds the recessions without being told where they are. The dated episodes also include 1969Q3–1970Q4 and 1979Q2–1980Q3, which overlap the NBER recessions of 1969–70 and 1980. When you write this up, say which dates you compare with and how you define "detected" and "false signal".

50.6 Diagnostics

```
oes.show(r, "Residual diagnostics")
```

```
Residual diagnostics (observed minus regime-weighted fit)
                               Statistic  p-value
↪      Detail
Test
Portmanteau (Hosking), 12 lags    12.3576   0.4174  df = 12; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, residual               12.5540   0.4023                                                        H0: no
↪ autocorrelation
Jarque-Bera, residual              3.3402   0.1882                                                          H0:
↪ normal residuals
ARCH LM, residual                  1.3403   0.8545                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

The residuals pass: no autocorrelation (Ljung-Box p = 0.40; Hosking portmanteau p = 0.42), no sign of non-normality (Jarque-Bera p = 0.19) and no ARCH effects (p = 0.85). The model that allows only the mean to switch is adequate for the series' dynamics. If the ARCH test rejected, you would let the variance switch too (Section 50.7).

50.7 Robustness: lags, switching variance, switching dynamics and regimes

```
def summary(res):
    s = {x["label"]: x["value"] for x in res["summary"]}
    t = [t for t in res["tables"] if t["title"].startswith("Comparison")]
    c = oes.frame(t[0])["Value"].tolist() if t else [np.nan] * 3
    return [s["Log-likelihood"], s["BIC"]] + c

variants = {"baseline": {}, "AR(2)": dict(order=2), "AR(1)": dict(order=1),
            "switching variance": dict(switching_variance="yes"),
            "switching AR coefficients": dict(switching_ar="yes"), "3 regimes": dict(regimes=3)}
tab = {k: summary(oes.run("ms_ar", data, dict(spec, **v))) for k, v in variants.items()}
print(pd.DataFrame(tab, index=["log-lik", "BIC", "concordance", "recessions detected", "false signals"]).T.
  ↪ round(3))
```

```
                           log-lik      BIC  concordance  recessions detected  false signals
baseline                  -181.263  406.404        0.916                0.963          0.096
AR(2)                     -185.668  405.568        0.910                0.556          0.000
AR(1)                     -187.081  403.550        0.903                0.519          0.000
switching variance        -180.677  410.107        0.855                0.963          0.173
switching AR coefficients -176.251  415.879        0.740                0.667          0.240
3 regimes                 -177.016  422.286        0.847                0.259          0.000
```

How to read it

The lag order matters a great deal for the dating. With four lags (Hamilton's choice) the model detects 96% of NBER recession quarters; with two lags it detects 56% and with one lag 52%, though concordance stays above 90% because most quarters are expansions. The shorter models put the low state on fewer quarters, so a claim like "the model finds the recessions" holds for the four-lag specification only. Allowing the variance to

switch keeps the detection rate (96%) but raises false signals to 17% and worsens the BIC (410.1 against 406.4). Allowing the autoregressive coefficients to switch raises the log-likelihood to −176.3 but the BIC is 415.9 and only 67% of recession quarters are detected. A third regime improves the log-likelihood to −177.0 but the BIC is 422.3 and one of the three regimes is a single quarter. The baseline is the best on the information criteria among the switching models.

50.8 Write the results

How to write it: results

The two-regime model separates a high-growth state (mean 1.15% per quarter, expected duration 10.4 quarters) from a low-growth state (mean −0.42% per quarter, expected duration 4.1 quarters). The transition probabilities are 0.75 for staying in the low state and 0.90 for staying in the high state. The smoothed probabilities date seven low-growth episodes between 1953 and 1982 that correspond to the post-war US recessions: 91.6% of quarters are classified in agreement with the NBER indicator, 96.3% of NBER recession quarters are detected and 9.6% of other quarters are flagged. The regime classification measure is 21.7, which indicates a sharp classification. However, the likelihood evidence for two regimes is weak: one regime has the lowest AIC and BIC, and the linearity statistic (4.8) is below the Garcia 5% critical value (about 14; Davies p = 0.95). The regime estimates are therefore best read as a description of the cycle that agrees with the NBER dates, not as a rejection of a linear model. The dating depends on the lag order: with two or fewer lags only about half of the NBER recession quarters are detected.

Report	What	Where
Model	Series, lags, regimes, what switches, estimation method	Section 50.1
Data	Unit-root tests	Section 50.2
Regimes	AIC, BIC, linearity statistic and its critical value	Section 50.3
Estimates	Regime means, transition probabilities, durations, RCM	Section 50.4
Dates	Regime dates, comparison with NBER dates, figures	Section 50.5
Diagnostics	Residual tests	Section 50.6
Robustness	Lags, switching variance and dynamics, number of regimes	Section 50.7

References. Hamilton (1989), *Econometrica*; Kim (1994), *Journal of Econometrics*; Garcia (1998), *International Economic Review*; Davies (1987), *Biometrika*; Hansen (1992), *Journal of Applied Econometrics*; Ang and Bekaert (2002), *Review of Financial Studies*; Hamilton (2016), *Handbook of Macroeconomics*.

CHAPTER 51

Markov-Switching Regression

Applies when. A regression whose intercept, slopes or error variance shift between hidden regimes, such as a Phillips curve in calm and turbulent periods

Overview. Use this chapter when the relationship you estimate may change between unobserved states, for example between calm and turbulent inflation, and you want the data to date the states and say which parameters differ. The chapter covers the unit-root check, the number of regimes, the linearity test, estimation with regime-specific intercepts, slopes and variances, regime dates, diagnostics, how to recognise a collapsed solution, robustness and the write-up.

Data. U.S. quarterly inflation and unemployment, 1959–2009

Software. Python · OES engine · statsmodels

51.1 The question and the model

Does the Phillips curve differ between calm and turbulent regimes? In a Markov-switching regression (Goldfeld and Quandt 1973; Hamilton 1989) the coefficients and the error variance depend on an unobserved state s_t:

$$y_t = \mu_{s_t} + x_t'\beta_{s_t} + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \sigma^2_{s_t}), \qquad \Pr(s_t = j \mid s_{t-1} = i) = p_{ij}.$$

Here y_t is inflation and x_t contains lagged inflation and lagged unemployment. You choose which parts switch: the intercept always does; the slopes and the variance can switch or be common. The model is estimated by maximum likelihood with the Hamilton filter.

Three checks before you trust a regime regression

1. **Is the dependent variable stationary?** Regimes in a trending series are the trend in disguise.
2. **Does the data need regimes at all?** Compare the one-regime regression with two and three regimes, and use the linearity test with the Garcia critical value or the Davies bound (the usual χ^2 test is not valid).
3. **Is each regime a real regime?** A regime that contains two or three observations is not a regime but an outlier, and its variance goes to zero. Check the share and duration of every regime (Section 51.5).

How to write it: the model

We estimate a two-regime Markov-switching regression (Hamilton 1989) of quarterly inflation on its own lag and the lagged unemployment rate. The intercept, the slopes and the error variance are regime-specific, and the regime follows a first-order Markov chain. We estimate it by maximum likelihood, starting from the solution of the restricted model with common slopes and from several random starts, and keep the solution with the highest likelihood. We compare one, two and three regimes with AIC and BIC, and test linearity with the likelihood-ratio statistic, the critical values of Garcia (1998) and the bound of Davies (1987), because the standard asymptotic distribution does not hold when parameters are unidentified under the null.

51.2 Load the data and test for unit roots

```
import numpy as np
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine
```

```
df = pd.read_csv("us_macro.csv")
df["inflation_lag"] = df["inflation"].shift(1)
df["inflation_lag2"] = df["inflation"].shift(2)
df["unemp_lag"] = df["unemp"].shift(1)
df = df.dropna()                      # one common sample for every model below
data = df.to_csv(index=False)
print(df[["inflation", "unemp"]].describe().round(2).loc[["count", "mean", "std", "min", "max"]])
spec = dict(y="inflation", x=["inflation_lag", "unemp_lag"], time="date", regimes=2, switching_exog="yes",
  ↪ switching_variance="yes")
r = oes.run("ms_reg", data, spec)
oes.show(r, "Unit-root")
```

```
       inflation   unemp
count     200.00  200.00
mean        4.00    5.89
std         3.26    1.47
min        -8.79    3.40
max        14.62   10.70
Unit-root tests
                    ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
dependent variable                   0.025               0.0709                     0.0         0.0772       I(0)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

Quarterly inflation rejects a unit root at 5% with a constant ($p = 0.025$), is borderline with a trend ($p = 0.07$), and the KPSS test does not reject stationarity at 5% ($p = 0.077$). We treat it as stationary, which is what a regime-in-the-mean model needs; the evidence is not overwhelming, so say so. If the series were clearly I(1) (as the federal funds rate is, with $p = 0.29$), you would model its change instead.

51.3 How many regimes? Is the relationship linear?

```
oes.show(r, "Number of regimes")
oes.show(r, "Linearity")
```

```
Number of regimes: information criteria
         Log-likelihood      AIC      BIC     HQIC
Regimes
1              -466.070  940.140  953.333  945.479
2              -426.044  872.087  905.070  885.435
3              -404.794  845.589  904.958  869.615
Note: Regimes = 1 is the linear model. Under the null of no switching some parameters are unidentified, so a standard
↪  LR test is not valid (Hansen 1992; Garcia 1998): compare models with the criteria and with the economic meaning
↪ of the regimes.

Linearity: likelihood-ratio statistic
                                                                    Value
Statistic
LR = 2 (logL switching - logL linear)                              80.052
Garcia (1998) 5% critical value, two-regime models (approximate)  about 14
Davies upper-bound p-value                                           0.0
Note: A large LR (well above the Garcia critical value) rejects the linear model.
```

How to read it

The log-likelihood rises from −466.1 with one regime to −426.0 with two, and the linearity statistic is 80.1, far above the Garcia 5% critical value of about 14 (Davies $p < 0.001$): **the data reject a single regime**. Two regimes against three is a closer call: the BIC is 905.1 for two and 905.0 for three, AIC prefers three (845.6 against 872.1). We stay with two regimes for two reasons: the BIC does not prefer three, and the three-regime solution collapses (Section 51.7). The three-regime row in this table comes from one default start, so treat it as indicative.

51.4 Estimate

```
oes.show(r, "Parameter estimates")
oes.show(r, "Transition")
oes.show(r, "Regimes")
oes.show(r, "Regime classification")
print(r["notes"])
```

```
Parameter estimates
           Estimate  Std. err.        z  p-value
Parameter
p[0->0]      0.9665     0.0184  52.6602   0.0000
p[1->0]      0.0600     0.0342   1.7579   0.0788
const[0]     1.6611     0.6397   2.5969   0.0094
const[1]     3.5224     2.1242   1.6582   0.0973
x1[0]        0.5018     0.0860   5.8316   0.0000
x1[1]        0.5608     0.1003   5.5911   0.0000
x2[0]       -0.0375     0.1046  -0.3586   0.7199
x2[1]       -0.1389     0.2974  -0.4670   0.6405
sigma2[0]    1.6250     0.2327   6.9839   0.0000
sigma2[1]   13.5182     2.4612   5.4924   0.0000
Note: p[i->j] is the probability of moving from regime i to regime j; [r] marks a regime-specific parameter.

Transition probabilities (row: from, column: to)
           Regime 0  Regime 1
From
Regime 0    0.9665    0.0335
Regime 1    0.0600    0.9400

Regimes
           Expected duration (periods)  Share of periods (smoothed prob. > 0.5)  Mean of inflation in the regime
Regime
Regime 0                       29.807                                    0.665                          2.9110
Regime 1                       16.653                                    0.335                          6.1483

Regime classification
                                                          Value
Measure
Regime classification measure, RCM (Ang and Bekaert 2002)  15.41
Note: RCM = 0: every period is assigned to a regime with certainty; RCM = 100: no information. Below about 30 is a
↪ sharp classification.

['Markov-switching regression (Hamilton 1989) with a switching constant, switching slopes and switching variance;
↪ maximum likelihood (EM, then quasi-Newton), standard errors from the Hessian.']
```

How to read it

Regime 0 is the calm regime: it holds 66.5% of the quarters, lasts 29.8 quarters on average ($p_{00} = 0.967$), and inflation averages 2.9%. Regime 1 is the turbulent regime: 33.5% of the quarters, 16.7 quarters on average ($p_{11} = 0.94$), inflation averages 6.1%. The sharpest difference is the **error variance**: 1.6 against 13.5, more than eight times larger. The intercepts differ (1.66, p = 0.009, and 3.52, p = 0.10), but the second is imprecise. The coefficient on lagged inflation is almost the same in both regimes (0.50 and 0.56, both highly significant), and the coefficient on lagged unemployment is small and insignificant in both (−0.04 and −0.14). So the regimes differ in the level and the volatility of inflation, not in how it responds to unemployment. The probability of leaving the turbulent regime (0.06) is only marginally significant (p = 0.08). The RCM of 15.4 shows a sharp classification. No collapse warning was raised.

51.5 Regime dates

```
oes.show(r, "Regime dating")
oes.plot(r, "Smoothed regime probabilities", ncols=1, size=(6.5, 3))
oes.plot(r, "inflation and its regime-dependent fitted mean", ncols=1, size=(6.5, 3))
```

```
Regime dating (most likely regime, consecutive periods)
            From      To  Periods  Average probability
Regime
```

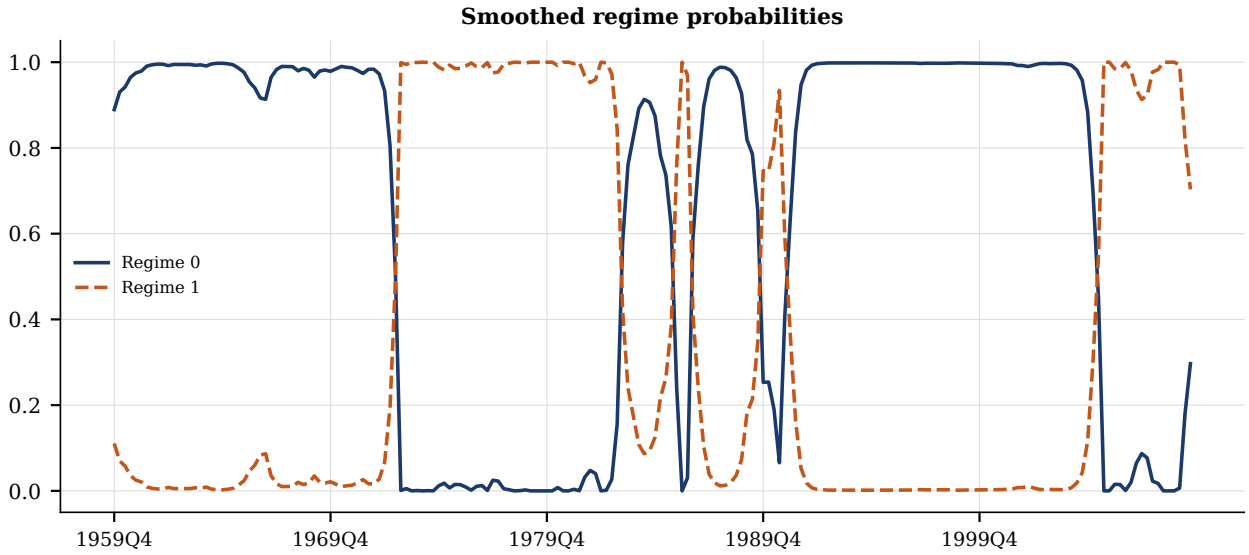


Figure 51.1. Regime dates: Smoothed regime probabilities.

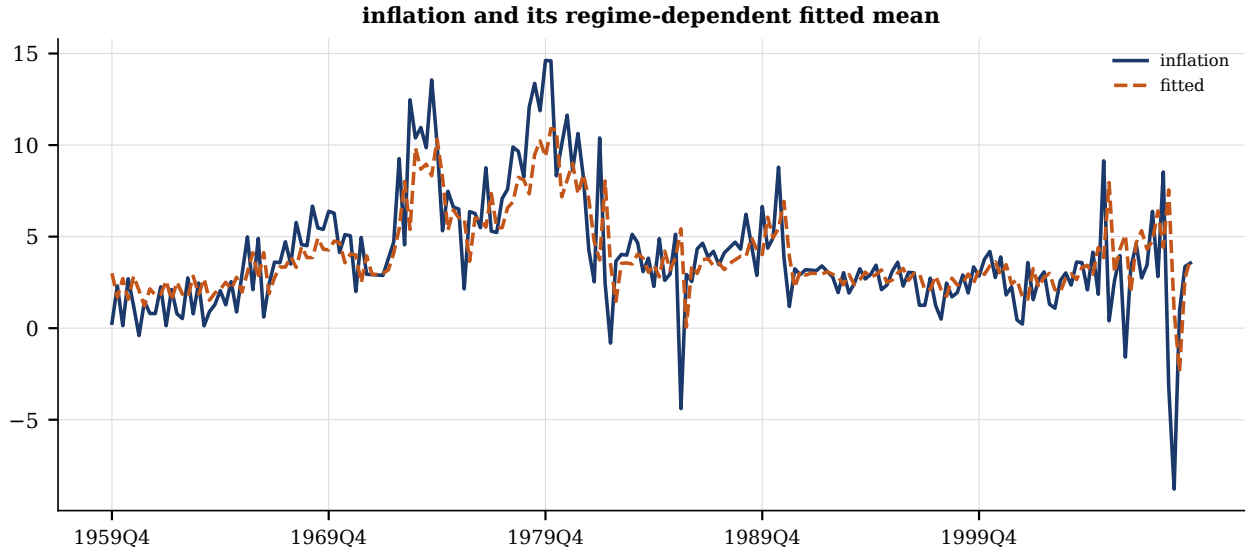


Figure 51.2. Regime dates: inflation and its regime-dependent fitted mean.

Regime 0	1959Q4	1972Q4	53	0.965
Regime 1	1973Q1	1983Q1	41	0.988
Regime 0	1983Q2	1985Q3	10	0.789
Regime 1	1985Q4	1986Q2	3	0.909
Regime 0	1986Q3	1989Q3	13	0.869
Regime 1	1989Q4	1990Q4	5	0.766
Regime 0	1991Q1	2005Q1	57	0.978
Regime 1	2005Q2	2009Q3	18	0.930

How to read it

The turbulent regime covers 1973Q1–1983Q1 (41 quarters, average probability 0.99, the oil shocks and the disinflation of 1979–82), short episodes in 1985–86 and 1989–90, and 2005Q2–2009Q3 (18 quarters, the energy price swings and the 2008–09 crisis). The calm regime covers 1959Q4–1972Q4 and 1991Q1–2005Q1. The short episodes (3 and 5 quarters) have lower probabilities (0.77–0.91); do not interpret them as separate events. Show the probability figure, not only the dates.

51.6 Diagnostics

```
oes.show(r, "Residual diagnostics")
```

```
Residual diagnostics (observed minus regime-weighted fit)
                              Statistic  p-value
↪      Detail
Test
Portmanteau (Hosking), 12 lags    35.8763   0.0003  df = 12; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, residual               36.2350   0.0003                                                        H0: no
↪ autocorrelation
Jarque-Bera, residual            193.9496   0.0000                                                          H0:
↪ normal residuals
ARCH LM, residual                 26.8059   0.0000                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

The residuals fail all three tests: autocorrelation (Ljung-Box p = 0.0003), non-normality (Jarque-Bera p < 0.001) and ARCH (p < 0.001). Quarterly inflation is noisy, has outliers (the minimum is −8.8) and has dynamics beyond one lag, which the regime-dependent variance only partly absorbs. The standard errors are therefore approximate; report that, and note that the regime conclusion (a turbulent and a calm state) does not depend on them. If dynamics are the point, use the Markov-switching autoregression with more lags or the Markov-switching VAR.

51.7 Robustness, and what a collapsed regime looks like

```
def run_variant(**kw):
    res = oes.run("ms_reg", data, dict(spec, **kw))
    s = {x["label"]: x["value"] for x in res["summary"]}
    reg = oes.frame(next(t for t in res["tables"] if t["title"] == "Regimes"))
    warn = [n for n in res["notes"] if "collapsed" in n or "almost zero" in n]
    return [s["Log-likelihood"], s["BIC"], " / ".join(f"{v:.0%}" for v in reg.iloc[:, 1]), "yes" if warn else
  ↪ "no"]

variants = {"baseline": {}, "common slopes": dict(switching_exog="no"), "common variance": dict(
  ↪ switching_variance="no"),
            "without unemployment": dict(x=["inflation_lag"]), "two lags": dict(x=["inflation_lag", "
  ↪ inflation_lag2", "unemp_lag"]),
            "three regimes": dict(regimes=3)}
tab = {k: run_variant(**v) for k, v in variants.items()}
print(pd.DataFrame(tab, index=["log-lik", "BIC", "share of periods", "collapse warning"]).T)
```

```
                     log-lik      BIC share of periods collapse warning
baseline             -426.044   905.07        66% / 34%               no
common slopes        -426.202  894.791        66% / 34%               no
common variance       -449.27  946.226        54% / 46%               no
without unemployment -426.219  894.825        66% / 34%               no
two lags               -304.5   672.58         2% / 98%              yes
three regimes         -350.77  796.911   2% / 58% / 40%              yes
```

How to read it

Slopes: letting the slopes switch gives almost the same likelihood as common slopes (−426.0 against −426.2), and the BIC is 10 points lower with common slopes (894.8 against 905.1), so the slopes do not need to switch. **Unemployment:** dropping it leaves the likelihood and BIC unchanged (−426.2, 894.8): the Phillips-curve term adds nothing. **Variance:** forcing a common variance lowers the log-likelihood by 23 (to −449.3) and raises the BIC to 946.2, and the regimes become 54% and 46%: switching variance is the essential feature. **Collapse:** with two lags, and with three regimes, the estimation converges to a regime that holds only 2% of the quarters, and the warning flags both. In the three-regime case one regime holds three quarters and has a variance of zero, and the log-likelihood rises to −350.8, higher than the non-degenerate solutions, which is exactly why a higher likelihood alone is not a criterion. **Report the number of observations in each regime for every specification.**

51.8 Write the results

How to write it: results

We estimate a two-regime Markov-switching regression of quarterly U.S. inflation on lagged inflation and lagged unemployment, 1959Q4–2009Q3. The data reject a single regime (likelihood-ratio statistic 80.1, against a Garcia 5% critical value of about 14). The calm regime (66.5% of quarters, expected duration 29.8 quarters) has average inflation of 2.9% and error variance 1.6; the turbulent regime (33.5% of quarters, 16.7 quarters) has average inflation of 6.1% and error variance 13.5. The turbulent regime is dated 1973Q1–1983Q1 and 2005Q2–2009Q3, with short episodes in 1985–86 and 1989–90. The slope on lagged inflation is 0.50 in the calm and 0.56 in the turbulent regime; the coefficient on unemployment is not significant in either (−0.04 and −0.14). A specification with common slopes has a lower BIC (894.8 against 905.1), and dropping unemployment changes nothing, so the regimes differ in level and volatility, not in the response of inflation to unemployment. Allowing the variance to switch is essential (the log-likelihood falls by 23 without it). The residuals show autocorrelation and ARCH, so standard errors are approximate. Specifications with three regimes or two lags converge to a regime of fewer than five quarters, and we do not report them.

Report	What	Where
Model	Dependent variable, regressors, what switches, estimation and starting values	Section 51.1
Data	Unit-root tests	Section 51.2
Regimes	AIC, BIC for one to three regimes; linearity statistic and its critical value	Section 51.3
Estimates	Regime-specific intercepts, slopes, variances; transition probabilities; durations	Section 51.4
Dates	Regime dates and probabilities, figures	Section 51.5
Diagnostics	Residual tests	Section 51.6
Robustness	Common slopes or variance, lags, regressors, number of regimes, collapse warnings	Section 51.7

References. Goldfeld and Quandt (1973), *Journal of Econometrics*; Hamilton (1989), *Econometrica*; Kim and Nelson (1999), *State-Space Models with Regime Switching*, MIT Press; Garcia (1998), *International Economic Review*; Davies (1987), *Biometrika*; Hansen (1992), *Journal of Applied Econometrics*; Ang and Bekaert (2002), *Review of Financial Studies*.

CHAPTER 52

Markov-Switching VAR

Applies when. Several macro variables whose means, volatility or dynamics shift between hidden regimes, such as calm and turbulent periods

Overview. Use this chapter when you want to know whether the joint behaviour of several variables changes between unobserved regimes, how long each regime lasts, and whether shocks have different effects in each. The chapter covers the unit-root check, choosing the number of regimes and what switches (MSI, MSIH, MSIAH), estimation, regime dates, regime-specific impulse responses, diagnostics, robustness and the write-up.

Data. U.S. quarterly GDP growth and inflation, 1959–2009

Software. Python · OES engine

52.1 The question and the model

Do GDP growth and inflation behave differently in calm and turbulent regimes, and do shocks have different effects? A Markov-switching VAR (Krolzig 1997) lets the parameters of a VAR depend on an unobserved state s_t that follows a Markov chain:

$$y_t = \nu_{s_t} + A_{1,s_t} y_{t-1} + \cdots + A_{p,s_t} y_{t-p} + u_t, \qquad u_t \sim N(0, \Sigma_{s_t}).$$

You choose what switches. **MSI**: only the intercepts. **MSIH**: intercepts and covariance matrix; the autoregressive coefficients are common to all regimes. **MSIAH**: everything switches. The more switches, the more parameters and the more regimes can be fitted to outliers. The model is estimated by maximum likelihood with the EM algorithm (the Hamilton filter and the Kim smoother) from several starting values; standard errors come from the numerical Hessian.

Three checks before you trust a Markov-switching VAR

1. **Are the variables stationary?** Regimes should describe shifts in means and volatility of stationary variables.
2. **How many regimes, and what switches?** A model that switches too much can create a regime out of a few outliers. Compare the information criteria and look at the *size* of every regime (Section 52.7).
3. **Is it better than a linear VAR?** The usual χ^2 test is not valid; use the information criteria and the Davies bound.

How to write it: the model

We estimate a Markov-switching VAR (Krolzig 1997) with two lags in GDP growth and inflation. The regime, which is unobserved, follows a first-order Markov chain with two states. The intercepts and the covariance matrix of the innovations are regime-specific (MSIH); the autoregressive coefficients are common to both regimes. We estimate the model by maximum likelihood with the EM algorithm from several starting values, using the Hamilton filter and the Kim smoother for the regime probabilities, and take standard errors from the numerical Hessian. We choose the number of regimes with AIC and BIC and compare with a linear VAR using the likelihood-ratio statistic and its Davies bound (Davies 1987), because the standard asymptotic distribution does not hold when the transition probabilities are unidentified under the null.

52.2 Load the data and test for unit roots

```
import numpy as np
```

```
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("us_macro.csv").read()
df = pd.read_csv("us_macro.csv")
print(df[["gdp_growth", "inflation"]].describe().round(2).loc[["count", "mean", "std", "min", "max"]])
spec = dict(vars=["gdp_growth", "inflation"], time="date", lags=2, regimes=2, switching="MSIH", horizon=20)
r = oes.run("msvar", data, spec)
oes.show(r, "Unit-root")
```

```
       gdp_growth  inflation
count      202.00     202.00
mean         3.10       3.98
std          3.52       3.25
min         -8.28      -8.79
max         15.43      14.62
Unit-root tests
            ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p Conclusion
Variable
gdp_growth                   0.000               0.0000                      0.0         0.1000       I(0)
inflation                    0.025               0.0709                      0.0         0.0772       I(0)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

GDP growth rejects a unit root at any usual level; inflation rejects it at 5% with a constant ($p = 0.025$), is borderline with a trend ($p = 0.07$), and the KPSS test does not reject stationarity at 5% ($p = 0.077$). Both are treated as stationary, so the VAR can be estimated in levels of these growth rates.

52.3 How many regimes? Better than a linear VAR?

```
oes.show(r, "Number of regimes")
oes.show(r, "Model comparison")
```

```
Number of regimes: information criteria
         Log-likelihood       AIC       BIC      HQIC
Regimes
1              -968.789  1963.578  2006.456  1980.930
2              -910.825  1861.651  1927.617  1888.346
3              -897.809  1853.618  1949.269  1892.326
Note: Regimes = 1 is the linear VAR. A standard LR test is not valid under the null of no switching; compare the
↪ criteria.

Model comparison
                                       Value
Statistic
Log-likelihood (Markov switching)   -910.825
Log-likelihood (linear VAR)         -968.789
LR statistic                         115.927
Davies upper-bound p-value (approx.)   0.000
AIC (MS)                            1861.651
AIC (linear)                        1963.578
BIC (MS)                            1927.617
BIC (linear)                        2006.456
Note: Under the null of no switching the transition probabilities are unidentified, so the chi-square p-value is only
↪  a rough guide; prefer the information criteria.
```

How to read it

The two-regime model fits much better than a linear VAR: the log-likelihood is -910.8 against -968.8 (LR statistic 115.9, Davies $p < 0.001$), and AIC (1861.7 against 1963.6) and BIC (1927.6 against 2006.5) both prefer

two regimes. Two against three regimes: AIC prefers three (1853.6), BIC prefers two (1927.6 against 1949.3). We use two, because the BIC prefers it and the third regime is hard to interpret (Section 52.8).

52.4 Estimate

```
oes.show(r, "Regime-dependent intercepts")
oes.show(r, "Implied regime means")
oes.show(r, "Transition")
oes.show(r, "Regimes")
oes.show(r, "Autoregressive")
oes.show(r, "Residual covariance, regime 0")
oes.show(r, "Residual covariance, regime 1")
oes.show(r, "Regime classification")
```

```
Regime-dependent intercepts
                     Estimate  Std. err.       z  p-value
Parameter
gdp_growth, regime 0    2.2145     0.7856  2.8190   0.0048
gdp_growth, regime 1    3.0152     0.4706  6.4072   0.0000
inflation, regime 0     1.2251     0.5620  2.1799   0.0293
inflation, regime 1     0.7281     0.3007  2.4216   0.0155
Note: Standard errors from the numerical Hessian of the log-likelihood at the EM estimates.

Implied regime means (steady state within each regime)
            Regime 0  Regime 1
Variable
gdp_growth    1.9365    3.8236
inflation     5.1582    3.2760

Transition probabilities (row: from, column: to)
                 Regime 0         Regime 1
From
Regime 0  0.9040 (0.0439)  0.0960 (0.0439)
Regime 1  0.0623 (0.0292)  0.9377 (0.0292)
Note: Standard errors in parentheses (delta method).

Regimes
          Expected duration  Ergodic probability  Share of periods (smoothed prob. > 0.5)
Regime
Regime 0             10.413               0.3935                                    0.395
Regime 1             16.052               0.6065                                    0.605

Autoregressive coefficients (common to all regimes)
                eq. gdp_growth     eq. inflation
Regressor
L1.gdp_growth   0.2048 (0.0673)   0.0817 (0.0389)
L1.inflation   -0.0310 (0.1064)   0.3825 (0.0748)
L2.gdp_growth   0.1749 (0.0682)  -0.0625 (0.0387)
L2.inflation   -0.1654 (0.1036)   0.3728 (0.0716)
Note: Standard errors in parentheses.

Residual covariance, regime 0
            gdp_growth  inflation

gdp_growth     17.7205     2.6341
inflation       2.6341    12.1852

Residual covariance, regime 1
            gdp_growth  inflation

gdp_growth      4.8780     0.0567
inflation       0.0567     1.0781

Regime classification
                                                    Value
```

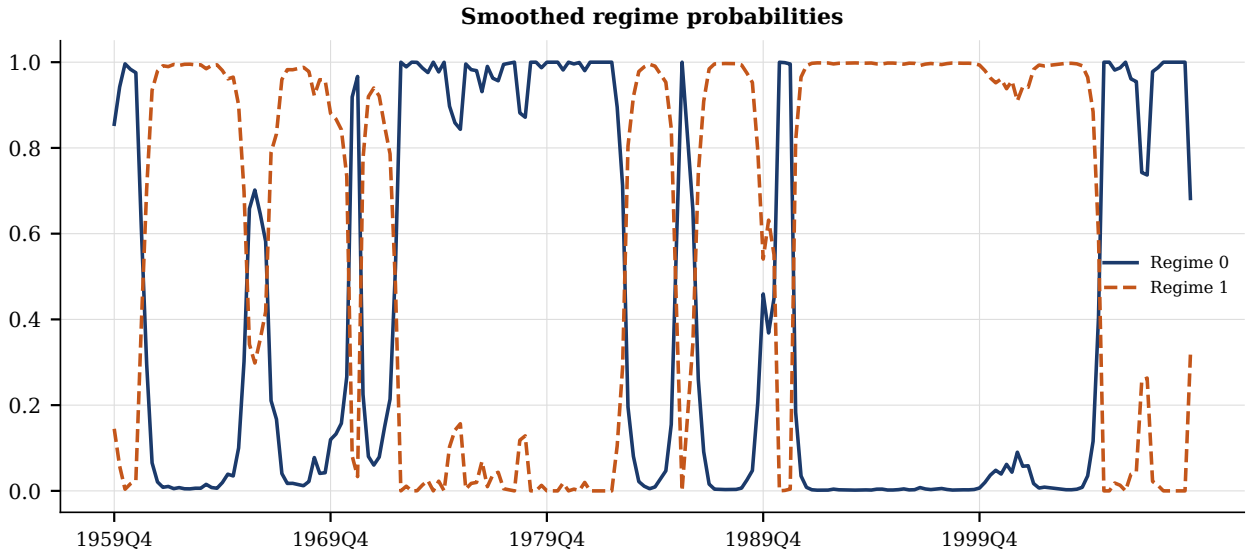


Figure 52.1. Regime dates: Smoothed regime probabilities.

```
Measure
Regime classification measure, RCM (Ang and Bekaert 2002)  19.51
Note: RCM = 0: sharp classification; 100: none. Below about 30 is sharp.
```

How to read it

Regime 0 is the turbulent regime: 39.5% of quarters, 10.4 quarters on average ($p_{00} = 0.90$, standard error 0.04); implied mean GDP growth 1.9% and inflation 5.2%. Regime 1 is the calm regime: 60.5% of quarters, 16.1 quarters on average ($p_{11} = 0.94$, standard error 0.03); mean GDP growth 3.8% and inflation 3.3%. The covariance matrices differ most: the standard deviation of the GDP-growth innovation is 4.2 in regime 0 and 2.2 in regime 1, and that of inflation is 3.5 against 1.0. The correlation of the two innovations is 0.18 in regime 0 and about zero in regime 1. All four intercepts are significant. The autoregressive coefficients are common: inflation is persistent (0.38 and 0.37 on its two lags), GDP growth depends on its own lags (0.20 and 0.17) and, weakly, inflation affects growth negatively (−0.17 at lag 2 but not significant). The RCM is 19.5, a fairly sharp classification.

52.5 Regime dates

```
oes.show(r, "Regime dating")
oes.plot(r, "Smoothed regime probabilities", ncols=1, size=(6.5, 3))
oes.plot(r, "inflation and the regime-weighted mean", ncols=1, size=(6.5, 3))
```

```
Regime dating (most likely regime, consecutive periods)
            From      To  Periods  Average probability
Regime
Regime 0  1959Q4  1961Q1        6                0.895
Regime 1  1961Q2  1965Q4       19                0.950
Regime 0  1966Q1  1966Q4        4                0.647
Regime 1  1967Q1  1970Q3       15                0.911
Regime 0  1970Q4  1971Q1        2                0.943
Regime 1  1971Q2  1972Q3        6                0.865
Regime 0  1972Q4  1983Q2       43                0.957
Regime 1  1983Q3  1985Q3        9                0.939
Regime 0  1985Q4  1986Q3        4                0.768
Regime 1  1986Q4  1990Q2       15                0.870
Regime 0  1990Q3  1991Q1        3                0.998
Regime 1  1991Q2  2005Q2       57                0.975
Regime 0  2005Q3  2009Q3       17                0.942
```

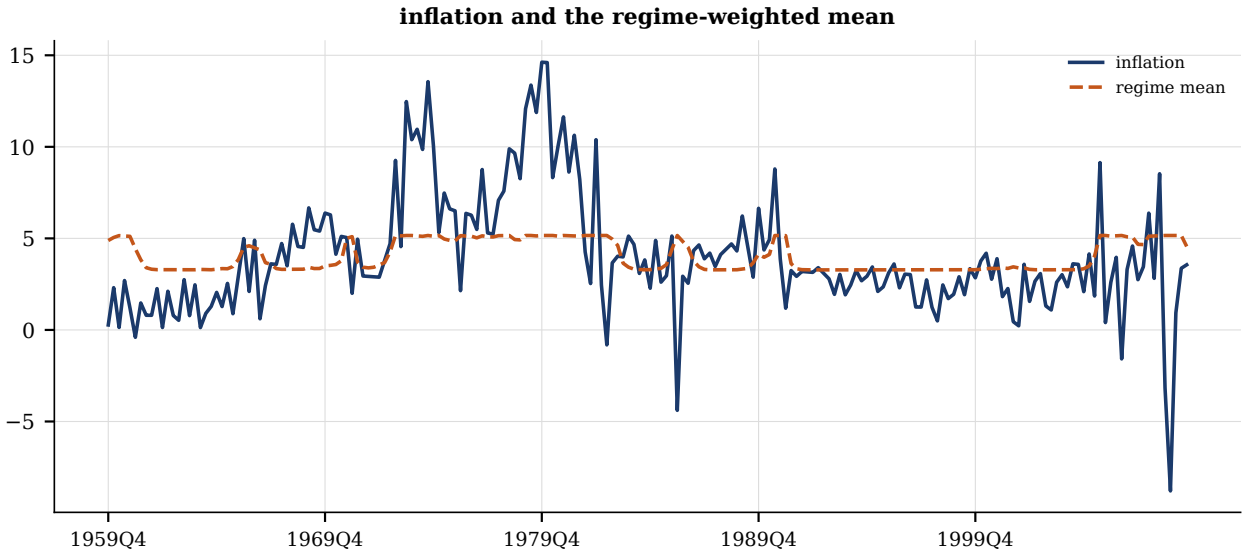


Figure 52.2. Regime dates: inflation and the regime-weighted mean.

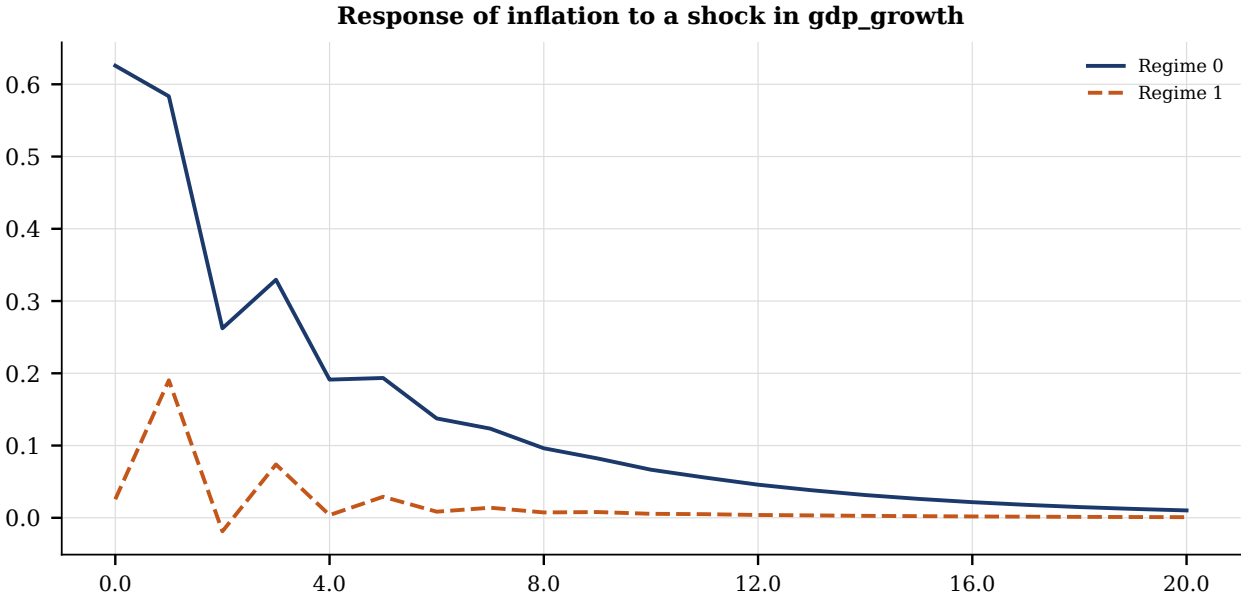


Figure 52.3. Impulse responses by regime: Response of inflation to a shock in gdp_growth.

How to read it

The turbulent regime is dated 1972Q4–1983Q2 (43 quarters, average probability 0.96) and 2005Q3–2009Q3 (17 quarters, 0.94), with short episodes in 1959–61, 1966, 1970–71, 1985–86 and 1990–91. The calm regime covers 1991Q2–2005Q2 (57 quarters) and most of the 1960s. The short episodes of two to six quarters have lower and more variable probabilities (for 1966 the average is 0.65): do not read them as separate events. Present the probability figure with the table.

52.6 Impulse responses by regime

```
oes.plot(r, "Response of inflation to a shock in gdp_growth", ncols=1, size=(6, 3))
oes.plot(r, "Response of gdp_growth to a shock in inflation", ncols=1, size=(6, 3))
```

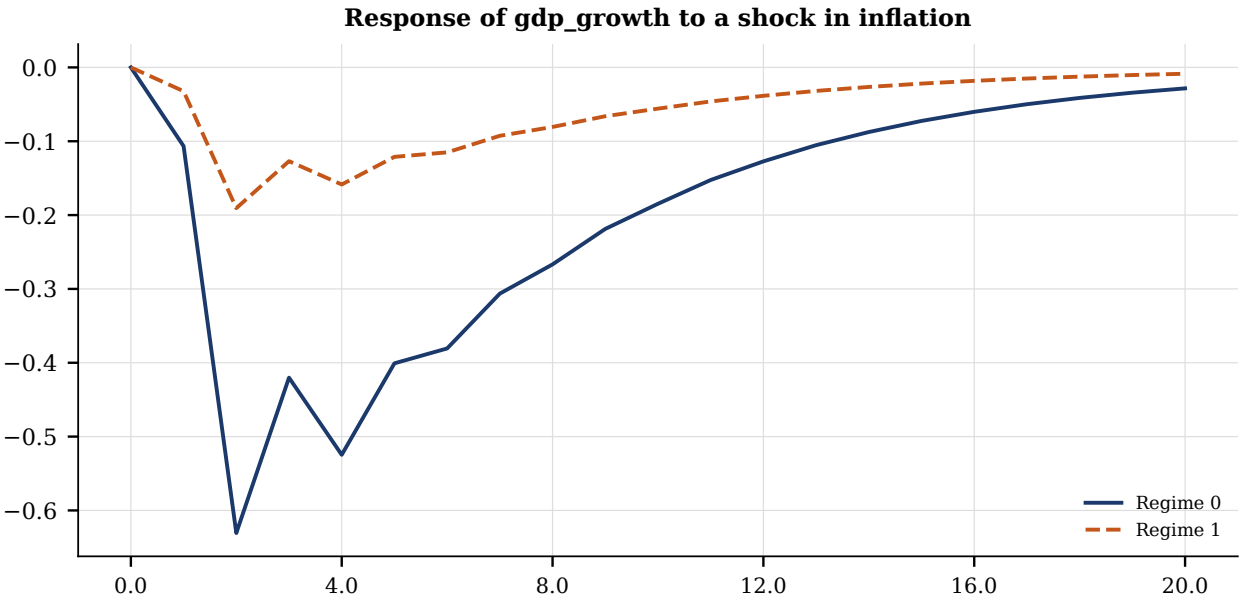


Figure 52.4. Impulse responses by regime: Response of gdp_growth to a shock in inflation.

How to read it

The shock is one standard deviation of the regime's Cholesky factor, with GDP growth ordered first (so inflation does not affect GDP growth on impact: this is an assumption, state it). In the MSIH-VAR the autoregressive coefficients are common, so the **shape** of the responses is the same in both regimes and the difference is the **size of the shocks**: a one-standard-deviation inflation shock is 3.4 percentage points in the turbulent regime and 1.0 in the calm one. A GDP-growth shock raises inflation by 0.63 on impact and 0.58 a quarter later in regime 0, against 0.03 and 0.19 in regime 1. An inflation shock lowers GDP growth by 0.52 after four quarters in regime 0 and by 0.16 in regime 1. If you want to claim that the *propagation* of shocks differs between regimes, you need the MSIAH model (Section 52.8). These responses have no confidence bands; do not claim significance for the differences.

52.7 Diagnostics

```
oes.show(r, "Residual diagnostics")
```

```
Residual diagnostics (observed minus regime-weighted fit)
                              Statistic  p-value
↪      Detail
Test
Portmanteau (Hosking), 12 lags    68.8202   0.0260  df = 48; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth              9.6657   0.6453                                                      H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           44.5542   0.0000                                                        H0:
↪ normal residuals
ARCH LM, gdp_growth                5.8986   0.2068                                          H0: no conditional
↪ heteroskedasticity
Ljung-Box, inflation              23.6173   0.0229                                                      H0: no
↪ autocorrelation
Jarque-Bera, inflation           195.0853   0.0000                                                        H0:
↪ normal residuals
ARCH LM, inflation                34.0124   0.0000                                          H0: no conditional
↪ heteroskedasticity
```

How to read it

Residuals are acceptable for GDP growth (Ljung-Box p = 0.65, ARCH p = 0.21) and not for inflation: autocorrelation (p = 0.02) and ARCH ($p < 0.001$). Both reject normality ($p < 0.001$), which reflects outliers.

The multivariate portmanteau test rejects at 5% (p = 0.026). The standard errors are therefore approximate. Residual ARCH in inflation suggests that two variance regimes do not capture all the volatility change; the SV-VAR in the Bayesian section handles this.

52.8 Robustness: what switches, lags, regimes and starting values

```
def run_variant(**kw):
    res = oes.run("msvar", data, dict(spec, **kw))
    s = {x["label"]: x["value"] for x in res["summary"]}
    ic = oes.frame(next(t for t in res["tables"] if t["title"].startswith("Number of regimes")))
    k = int(s["Regimes"])
    reg = oes.frame(next(t for t in res["tables"] if t["title"] == "Regimes"))
    return [s["Log-likelihood"], ic.loc[k, "BIC"], " / ".join(f"{v:.0%}" for v in reg["Share of periods (
  ↪ smoothed prob. > 0.5)"]),
            " / ".join(f"{v:.1f}" for v in reg["Expected duration"])]

variants = {"baseline (MSIH, 2 lags)": {}, "MSI: intercepts only": dict(switching="MSI"), "MSIAH: all switch":
  ↪  dict(switching="MSIAH"),
            "1 lag": dict(lags=1), "3 lags": dict(lags=3), "4 lags": dict(lags=4), "3 regimes": dict(regimes=3)
  ↪ , "20 starting values": dict(starts=20)}
tab = {k: run_variant(**v) for k, v in variants.items()}
print(pd.DataFrame(tab, index=["log-lik", "BIC", "share of periods", "duration (quarters)"]).T)
```

```
                        log-lik       BIC share of periods duration (quarters)
baseline (MSIH, 2 lags) -910.825  1927.617        40% / 60%         10.4 / 16.1
MSI: intercepts only    -945.627  1981.326         6% / 94%          1.4 / 21.0
MSIAH: all switch       -907.407  1963.167        39% / 61%         13.6 / 21.2
1 lag                   -937.439   1959.73        38% / 62%          9.9 / 16.1
3 lags                  -892.942  1912.923        36% / 64%          6.1 / 9.9
4 lags                  -884.359  1916.789        42% / 58%          7.6 / 10.7
3 regimes               -897.809  1949.269  27% / 28% / 44%   7.1 / 51.8 / 12.5
20 starting values      -910.825  1927.617        40% / 60%         10.4 / 16.1
```

How to read it

What switches. With intercepts only (MSI), the second regime is a 1.4-quarter episode holding 6% of the periods: a collapse onto outliers, and the BIC is 1981.3. Letting all coefficients switch (MSIAH) raises the log-likelihood by only 3.4 and the BIC to 1963.2, against 1927.6 for MSIH, so there is no evidence that the dynamics switch. **Lags.** BIC is lowest with three lags (1912.9; four lags 1916.8, two lags 1927.6, one lag 1959.7; the sample shrinks by one quarter per extra lag, so these are close but not exact comparisons). With three lags the classification agrees with the baseline in 91% of the quarters, but the regimes no longer differ in the level of inflation (implied means 4.12 and 4.08), only in growth (1.9% and 3.8%) and volatility: **the label "high-inflation regime" depends on the lag order**, so describe the regimes by volatility and growth unless the result holds across lags. **Regimes.** Three regimes hold 27%, 28% and 44% of the periods, with expected durations of 7.1, 51.8 and 12.5 quarters. **Starting values.** Twenty starting values give exactly the same log-likelihood as the default, so the solution is not a local maximum.

52.9 Write the results

How to write it: results

We estimate a Markov-switching VAR (Krolzig 1997) with two regimes and two lags in quarterly GDP growth and inflation, 1959Q4–2009Q3, with regime-specific intercepts and covariance matrices (MSIH). It fits much better than a linear VAR (BIC 1927.6 against 2006.5; likelihood-ratio statistic 115.9, Davies $p < 0.001$). The turbulent regime holds 39.5% of the quarters, lasts 10.4 quarters on average, and has mean GDP growth of 1.9%, mean inflation of 5.2% and innovation standard deviations of 4.2 and 3.5; the calm regime holds 60.5% of the quarters, lasts 16.1 quarters, and has mean growth of 3.8%, mean inflation of 3.3% and innovation standard deviations of 2.2 and 1.0. The turbulent regime is dated 1972Q4–1983Q2 and 2005Q3–2009Q3. Because the autoregressive coefficients are common, impulse responses have the same shape in both regimes and differ in size, since shocks are about three times larger in the turbulent regime; allowing the coefficients to switch does

> *not improve the BIC (1963.2). The result is robust to the starting values (20 starts give the same maximum). It is not robust to the lag order in one respect: with three lags the regimes no longer differ in mean inflation, only in growth and volatility. A model with intercepts only collapses onto a 1.4-quarter regime. Residuals for inflation show autocorrelation and ARCH, so standard errors are approximate.*

Report	What	Where
Model	Variables, lags, what switches, number of regimes, estimation and starting values	Section 52.1
Data	Unit-root tests	Section 52.2
Regimes	AIC, BIC for one to three regimes; comparison with the linear VAR	Section 52.3
Estimates	Regime intercepts and means, covariances, transition probabilities, durations	Section 52.4
Dates	Regime dates, probabilities, figures	Section 52.5
Responses	Regime-specific impulse responses (Cholesky ordering stated)	Section 52.6
Diagnostics	Residual tests	Section 52.7
Robustness	What switches, lags, regimes, starting values, regime sizes	Section 52.8

References. Krolzig (1997), *Markov-Switching Vector Autoregressions*, Springer; Hamilton (1989), *Econometrica*; Kim (1994), *Journal of Econometrics*; Sims and Zha (2006), *American Economic Review*; Davies (1987), *Biometrika*; Ang and Bekaert (2002), *Review of Financial Studies*.

CHAPTER 53

Markov-Switching Panel VAR

Applies when. A panel of countries or units that share the same unobserved regimes, such as a global slowdown that hits every country at once

Overview. Use this chapter when you have a panel of units (countries, regions, firms) observed over time and you believe that all of them move between the same hidden regimes, for example a boom and a slowdown, with common dynamics. The chapter covers the unit-root check, the number of regimes, what the shared-regime assumption means, estimation, regime dates, impulse responses, diagnostics, a leave-one-country-out check and the write-up.

Data. 14 OECD countries, annual GDP growth and unemployment, 1966–1990

Software. Python · OES engine

53.1 The question and the model

Did the OECD economies share a regime change in the 1970s? The panel Markov-switching VAR has the same form as the Markov-switching VAR for one economy, with one important restriction: **one unobserved regime path is shared by all units**, and the parameters are pooled across units,

$$y_{it} = \nu_{s_t} + A_1 y_{i,t-1} + \cdots + A_p y_{i,t-p} + u_{it}, \qquad u_{it} \sim N(0, \Sigma_{s_t}),$$

where i indexes countries, t years and s_t is the regime in year t (the same for every country). The likelihood adds up the units' contributions given the regime, so all countries inform the regime probabilities. As in the single-economy model, you choose what switches: **MSI** (intercepts), **MSIH** (intercepts and covariance matrices) or **MSIAH** (also the autoregressive coefficients). By default the data are demeaned by country, so that country-specific levels do not create regimes.

Three checks before you trust a panel regime model

1. **Is the shared-regime assumption plausible?** The model cannot find country-specific regimes. If countries had different cycles, the estimated regime is an average. Check that the dating survives dropping each country in turn (Section 53.8).
2. **Are the variables stationary?** Persistent series such as unemployment can create regimes out of trends.
3. **How many regimes, and how many time periods?** The regime path has only T points however many countries you have. Here $T = 24$, so regimes of one or two years carry little information.

How to write it: the model

We estimate a panel Markov-switching VAR with one lag in GDP growth and the unemployment rate for 14 OECD countries, 1966–1990. A single regime path, a first-order Markov chain with two states, is shared by all countries: the regime in a given year is the same for every country. Intercepts and covariance matrices are regime-specific (MSIH), and the autoregressive coefficients are pooled and common to the regimes. The variables are demeaned by country. We estimate the model by maximum likelihood with the EM algorithm from several starting values and take standard errors from the numerical Hessian. We choose the number of regimes with AIC and BIC.

53.2 Load the data and test for unit roots

```
import numpy as np
```

```
import pandas as pd
import oes_engine as oes            # pip install -e open-econometric-systems/oes-engine

data = open("garrett.csv").read()
df = pd.read_csv("garrett.csv")
print(df.groupby("country").size().describe().loc[["count", "min", "max"]])
print(df[["gdp_growth", "unem"]].describe().round(2).loc[["mean", "std", "min", "max"]])
spec = dict(vars=["gdp_growth", "unem"], unit="country", time="year", lags=1, regimes=2, switching="MSIH",
  ↪ horizon=20)
r = oes.run("mspvar", data, spec)
oes.show(r, "Unit-root")
```

```
count    14.0
min      25.0
max      25.0
dtype: float64
      gdp_growth   unem
mean        3.25   4.99
std         2.43   3.24
min        -4.30   0.68
max        12.80  13.00
Unit-root tests (cross-unit average of the series)
            ADF level, constant: p  ADF level, trend: p  ADF first difference: p  KPSS level: p
↪ Conclusion
Variable
gdp_growth                  0.0573              0.0570                  0.0758         0.0578  I(1) or I(2):
↪ borderline
unem                        0.5018              0.7695                  0.0245         0.0227
↪ I(1)
Note: ADF H0: unit root; KPSS H0: stationary (p-values are truncated at 0.01 and 0.10). For a Minnesota-type prior
↪ use a prior mean of 1 on the own first lag for I(1) variables in levels (random walk) and 0 for stationary
↪ variables. A VAR in levels with a random-walk prior remains valid when the variables are I(1) or cointegrated (
↪ Sims, Stock and Watson 1990).
```

How to read it

The panel is balanced: 14 countries with 25 years each (350 observations; 24 usable years per country with one lag). The unit-root tests, computed on the cross-country average, give a mixed picture. **Unemployment is I(1)**: the ADF test does not reject a unit root in levels ($p = 0.50$) but rejects it in first differences ($p = 0.02$), and KPSS rejects stationarity ($p = 0.02$). **GDP growth** is borderline (ADF $p = 0.06$, KPSS $p = 0.06$). So the unemployment series is persistent, and part of what the regimes capture could be its upward drift in the 1970s and 1980s. We estimate the model on demeaned levels as the default does; for a stricter treatment, use the change in unemployment and check that the dating survives.

53.3 How many regimes? Better than a linear panel VAR?

```
oes.show(r, "Number of regimes")
oes.show(r, "Model comparison")
```

```
Number of regimes: information criteria
         Log-likelihood       AIC       BIC      HQIC
Regimes
1             -1055.198  2128.395  2162.749  2142.090
2              -985.409  2002.818  2063.892  2027.164
3              -955.444  1960.887  2056.315  1998.928
Note: Regimes = 1 is the linear VAR. A standard LR test is not valid under the null of no switching; compare the
↪ criteria.

Model comparison
                                        Value
Statistic
Log-likelihood (Markov switching)     -985.409
Log-likelihood (linear VAR)          -1055.198
LR statistic                           139.577
Davies upper-bound p-value (approx.)     0.000
AIC (MS)                              2002.818
```

```
AIC (linear)                          2128.395
BIC (MS)                              2063.892
BIC (linear)                          2162.749
Note: Under the null of no switching the transition probabilities are unidentified, so the chi-square p-value is only
↪  a rough guide; prefer the information criteria.
```

How to read it

Two regimes fit much better than a linear panel VAR: the log-likelihood is −985.4 against −1055.2 (LR statistic 139.6, Davies $p < 0.001$), and AIC (2002.8 against 2128.4) and BIC (2063.9 against 2162.7) both prefer two regimes to one. **Three regimes are better still on both criteria** (AIC 1960.9, BIC 2056.3). We keep two regimes as the baseline, because the three-regime model has a short regime (expected duration 2.5 years, 21% of the years, in two separate spells) and because with 24 distinct regime dates any extra regime rests on very few years; but you must report that the criteria favour three and show what three regimes look like (Section 53.5 and Section 53.8).

53.4 Estimate

```
oes.show(r, "Regime-dependent intercepts")
oes.show(r, "Implied regime means")
oes.show(r, "Transition")
oes.show(r, "Regimes")
oes.show(r, "Autoregressive")
oes.show(r, "Residual covariance, regime 0")
oes.show(r, "Residual covariance, regime 1")
oes.show(r, "Regime classification")
```

```
Regime-dependent intercepts
                     Estimate  Std. err.       z  p-value
Parameter
gdp_growth, regime 0   -1.2385     0.1373 -9.0177   0.0000
gdp_growth, regime 1    1.9539     0.2267  8.6186   0.0000
unem, regime 0          0.3626     0.0625  5.8053   0.0000
unem, regime 1         -0.1645     0.0645 -2.5482   0.0108
Note: Standard errors from the numerical Hessian of the log-likelihood at the EM estimates.

Implied regime means (steady state within each regime)
            Regime 0  Regime 1
Variable
gdp_growth   -0.6696    1.7192
unem          2.3201   -2.0337

Transition probabilities (row: from, column: to)
                 Regime 0         Regime 1
From
Regime 0  0.8624 (0.0902)  0.1376 (0.0902)
Regime 1  0.3238 (0.1607)  0.6762 (0.1607)
Note: Standard errors in parentheses (delta method).

Regimes
          Expected duration  Ergodic probability  Share of periods (smoothed prob. > 0.5)
Regime
Regime 0              7.266               0.7018                                    0.625
Regime 1              3.088               0.2982                                    0.375

Autoregressive coefficients (common to all regimes)
                eq. gdp_growth          eq. unem
Regressor
L1.gdp_growth  0.2331 (0.0481)  -0.1355 (0.0182)
L1.unem        0.3125 (0.0481)   0.8046 (0.0205)
Note: Standard errors in parentheses.

Residual covariance, regime 0
            gdp_growth    unem
```

```
gdp_growth      2.7479 -0.7520
unem           -0.7520  0.6605

Residual covariance, regime 1
            gdp_growth    unem

gdp_growth      3.7256 -0.5040
unem           -0.5040  0.3227

Regime classification
                                                          Value
Measure
Regime classification measure, RCM (Ang and Bekaert 2002)   2.75
Note: RCM = 0: sharp classification; 100: none. Below about 30 is sharp.
```

How to read it

Regime 0 is the slow-growth regime: it holds 62.5% of the years, lasts 7.3 years on average ($p_{00} = 0.86$, standard error 0.09), and the implied mean is 0.7 percentage points of GDP growth below the country average and 2.3 points of unemployment above it. Regime 1 is the fast-growth regime: 37.5% of the years, 3.1 years on average ($p_{11} = 0.68$, standard error 0.16), with growth 1.7 points above the country average and unemployment 2.0 points below. The regimes are in deviation from each country's own mean because the data are demeaned. The pooled dynamics are: growth depends on its own lag (0.23) and on lagged unemployment (+0.31), unemployment is persistent (0.80) and falls after faster growth (−0.14). The innovation covariance matrices differ less than the means: the variance of the growth innovation is 2.7 in regime 0 and 3.7 in regime 1, and that of unemployment is 0.66 and 0.32. The correlation of the two innovations is −0.56 and −0.46. The RCM is 2.8, a very sharp classification, which is what a regime path shared by 14 countries should deliver. The standard errors on the transition probabilities are large, so treat the durations as rough.

53.5 Regime dates

```
oes.show(r, "Regime dating")
oes.plot(r, "Smoothed regime probabilities", ncols=1, size=(6.5, 3))
oes.plot(r, "gdp_growth (average over units) and the regime-weighted mean", ncols=1, size=(6.5, 3))
```

```
Regime dating (most likely regime, consecutive periods)
          From    To  Periods  Average probability
Regime
Regime 1  1967  1973        7                1.000
Regime 0  1974  1975        2                1.000
Regime 1  1976  1976        1                1.000
Regime 0  1977  1978        2                0.948
Regime 1  1979  1979        1                0.922
Regime 0  1980  1990       11                1.000
```

How to read it

The slow-growth regime covers 1974–75, 1977–78 and 1980–1990 (11 years in a row), and the fast-growth regime covers 1967–1973 (7 years), 1976 and 1979, all with average probabilities of 0.92 or above. So the model finds the break at the 1973–74 oil shock, a short rebound in 1976, a second oil shock around 1979, and a long slowdown through the 1980s. The one-year episodes (1976 and 1979) have to be reported as single years, not as regimes, because with one year the data cannot tell a regime from a shock. With three regimes (Section 53.8), 1974–75 and 1981–83 (the two deep slowdowns) form one regime, 1977–80 and 1984–90 a second, and 1967–73 with 1976 the third.

53.6 Impulse responses by regime

```
oes.plot(r, "Response of unem to a shock in gdp_growth", ncols=1, size=(6, 3))
oes.plot(r, "Response of gdp_growth to a shock in unem", ncols=1, size=(6, 3))
```

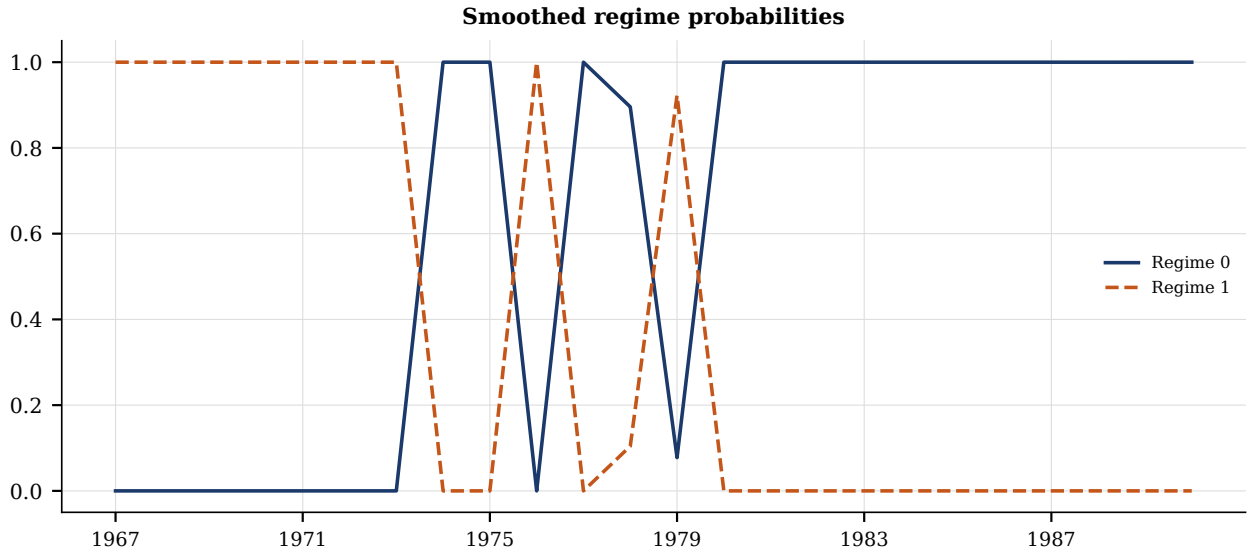


Figure 53.1. Regime dates: Smoothed regime probabilities.

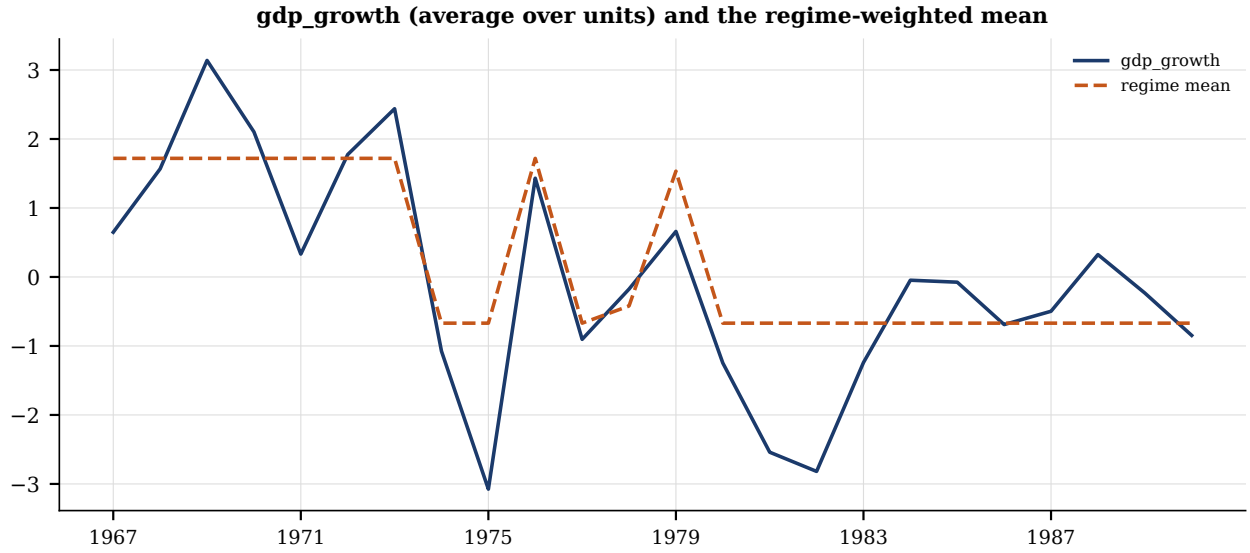


Figure 53.2. Regime dates: gdp_growth (average over units) and the regime-weighted mean.

How to read it

Impulse responses are regime-specific Cholesky responses, with GDP growth ordered first, and the shock is one standard deviation. A growth shock lowers unemployment by 0.45 on impact and by 0.59 a year later in regime 0 (0.26 and 0.47 in regime 1), and the effect disappears after about eight years. A one-standard-deviation unemployment shock raises growth by 0.21 a year later in regime 0 (0.16 in regime 1). Because the autoregressive coefficients are common, the shape is the same in both regimes and the response is larger in regime 0 only because its shocks are larger. The positive response of growth to an unemployment shock is a mean-reversion effect in demeaned data, not an economic channel: do not give it a structural interpretation. These responses have no confidence bands.

53.7 Diagnostics

```
oes.show(r, "Residual diagnostics")
Residual diagnostics (observed minus regime-weighted fit)
                              Statistic  p-value
↪      Detail
```

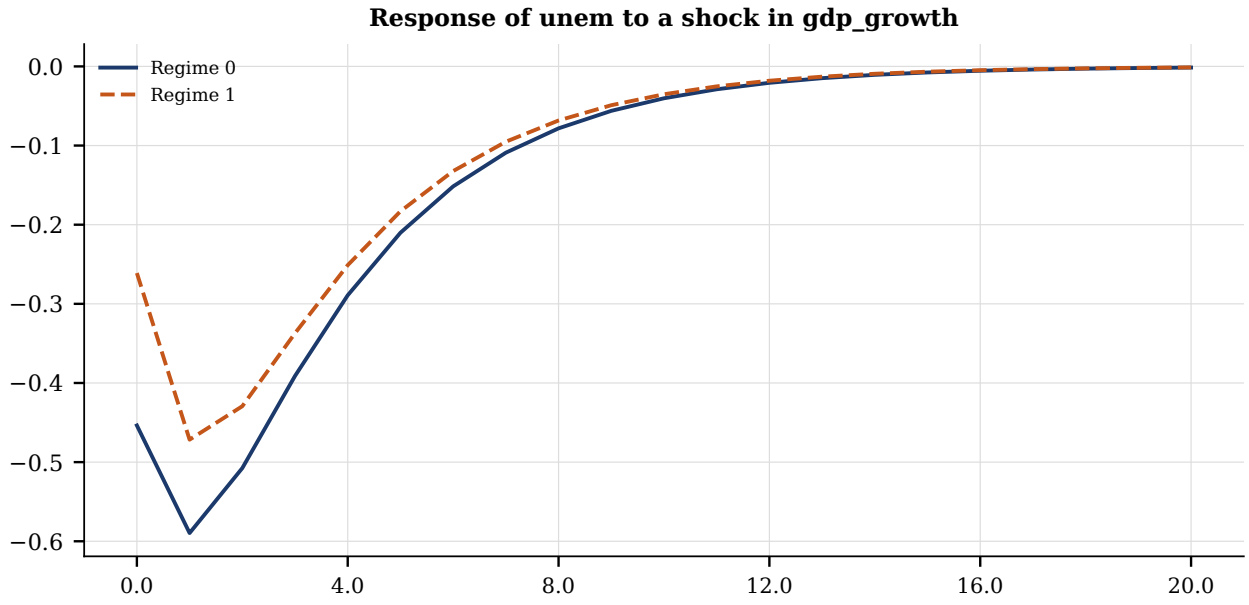


Figure 53.3. Impulse responses by regime: Response of unem to a shock in gdp_growth.

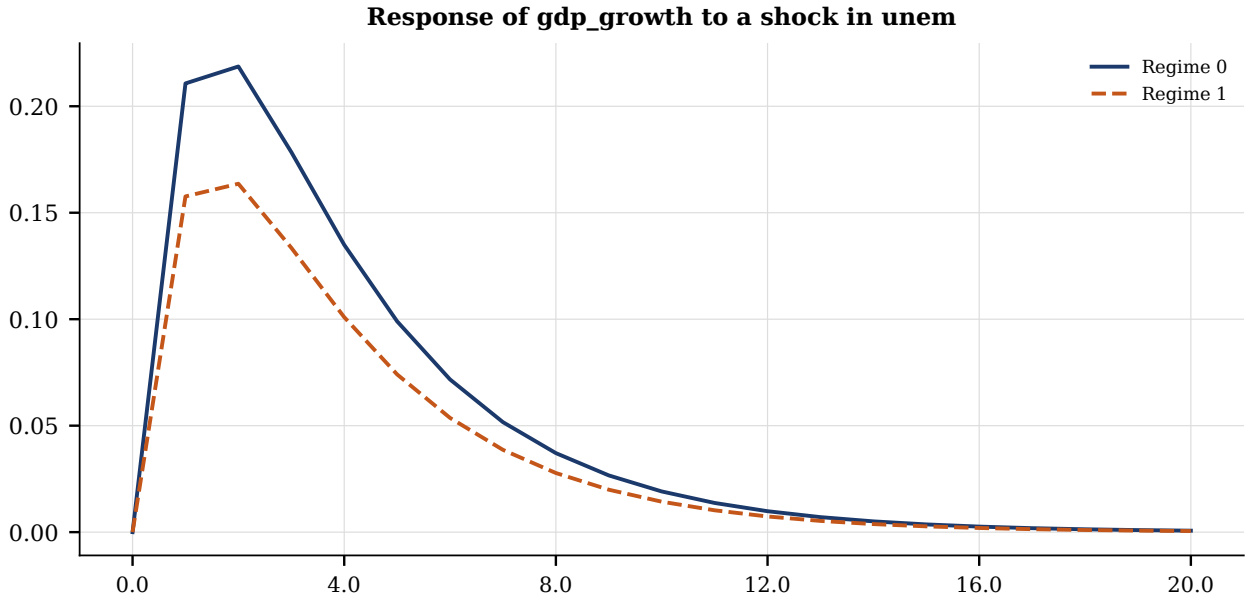


Figure 53.4. Impulse responses by regime: Response of gdp_growth to a shock in unem.

```
Test
Portmanteau (Hosking), 12 lags    79.7917   0.0027  df = 48; H0: no residual autocorrelation (approximate after
↪ estimating lags)
Ljung-Box, gdp_growth             15.6242   0.2091                                                      H0: no
↪ autocorrelation
Jarque-Bera, gdp_growth           22.3530   0.0000                                                        H0:
↪ normal residuals
ARCH LM, gdp_growth                2.5364   0.6381                                  H0: no conditional
↪ heteroskedasticity
Ljung-Box, unem                   29.3894   0.0034                                                      H0: no
↪ autocorrelation
Jarque-Bera, unem                 87.3575   0.0000                                                        H0:
↪ normal residuals
ARCH LM, unem                      5.7309   0.2202                                  H0: no conditional
↪ heteroskedasticity
```

How to read it

Growth residuals have no autocorrelation (Ljung-Box p = 0.21) and no ARCH (p = 0.64); unemployment residuals have autocorrelation (p = 0.003) and no ARCH (p = 0.22); both reject normality ($p < 0.001$). The panel portmanteau test rejects (p = 0.003). The remaining autocorrelation in unemployment suggests that one lag is not enough: with two lags the log-likelihood rises from −985.4 to −932.8 and the regime dating is unchanged (Section 53.8). The standard errors are approximate.

53.8 Robustness: specification and leave-one-country-out

```
def probs(res):
    c = res["charts"][0]
    return dict(zip(c["x"], c["series"][0]["y"]))

def run_variant(d=data, **kw):
    res = oes.run("mspvar", d, dict(spec, **kw))
    s = {x["label"]: x["value"] for x in res["summary"]}
    reg = oes.frame(next(t for t in res["tables"] if t["title"] == "Regimes"))
    return res, [s["Log-likelihood"], " / ".join(f"{v:.0%}" for v in reg["Share of periods (smoothed prob. >
  ↪ 0.5)"]),
                 " / ".join(f"{v:.1f}" for v in reg["Expected duration"])]

base_res, _ = run_variant()
pb = probs(base_res)
variants = {"baseline": {}, "MSI: intercepts only": dict(switching="MSI"), "MSIAH: all switch": dict(switching
  ↪ ="MSIAH"),
            "2 lags": dict(lags=2), "3 regimes": dict(regimes=3), "no country demeaning": dict(demean="no"), "
  ↪ 20 starting values": dict(starts=20)}
rows = {}
for k, v in variants.items():
    res, row = run_variant(**v)
    p = probs(res)
    common = [t for t in p if t in pb]
    rows[k] = row + [round(np.mean([(p[t] > .5) == (pb[t] > .5) for t in common]), 3)]
print(pd.DataFrame(rows, index=["log-lik", "share of years", "duration (years)", "agreement with baseline"]).T)
  ↪

loo = {}
for c in sorted(df["country"].unique()):
    res, row = run_variant(df[df["country"] != c].to_csv(index=False))
    p = probs(res)
    loo[c] = row[1:] + [round(np.mean([(p[t] > .5) == (pb[t] > .5) for t in p if t in pb]), 3)]
print(pd.DataFrame(loo, index=["share of years", "duration (years)", "agreement with baseline"]).T)
```

	log-lik	share of years	duration (years)	agreement with baseline
baseline	-985.409	62% / 38%	7.3 / 3.1	1.0
MSI: intercepts only	-997.44	67% / 33%	14.8 / 4.0	0.958
MSIAH: all switch	-973.348	58% / 42%	6.5 / 3.3	0.958
2 lags	-932.835	61% / 39%	6.6 / 3.0	0.957
3 regimes	-955.444	21% / 46% / 33%	2.5 / 10.0 / 4.0	0.583
no country demeaning	-1041.749	58% / 42%	7.3 / 3.3	0.958
20 starting values	-985.409	62% / 38%	7.3 / 3.1	1.0

	share of years	duration (years)	agreement with baseline
Austria	58% / 42%	6.6 / 3.3	0.958
Belgium	62% / 38%	7.0 / 3.0	1.0
Canada	62% / 38%	7.0 / 3.1	1.0
Denmark	58% / 42%	6.8 / 3.2	0.958
Finland	58% / 42%	6.6 / 3.3	0.958
France	62% / 38%	7.2 / 3.1	1.0
Italy	67% / 33%	14.6 / 4.0	0.958
Japan	62% / 38%	7.0 / 3.1	1.0
Netherlands	62% / 38%	7.5 / 3.1	1.0
Norway	67% / 33%	14.1 / 3.9	0.958
Sweden	67% / 33%	14.4 / 3.9	0.958

```
United Kingdom      62% / 38%       7.2 / 3.0                 1.0
United States       62% / 38%       7.0 / 3.0                 1.0
West Germany        67% / 33%      13.6 / 3.9               0.958
```

How to read it

Specification. Intercepts only (MSI) gives 14.8 years for the slow-growth regime and agrees with the baseline in 96% of the years (23 of 24). Letting all coefficients switch (MSIAH) raises the log-likelihood by 12 and gives the same dating up to one year (96%), with a BIC of 2063.0 against 2063.9: **no clear evidence that the dynamics switch.** Two lags and no demeaning both agree with the baseline in 96% of the years; starting values (20 against 6) make no difference. **Three regimes.** The dating agrees with the baseline in only 58% of the years, and the regime shares are 21%, 46% and 33%: the third regime is not a small refinement of the two-regime result. Report it next to the baseline. **Leave one country out.** Dropping each of the 14 countries in turn leaves the dating almost unchanged: agreement with the baseline is 96% to 100% (23 or 24 of 24 years), and the share of years in the slow-growth regime stays between 58% and 67%. The expected duration of the slow-growth regime is 6.6 to 7.5 years for ten of the fourteen runs and 13.6 to 14.6 years when Italy, Norway, Sweden or West Germany is dropped, because the 1979 interruption then disappears (the slow-growth regime runs from 1977 to 1990). So the main result is not driven by one country, but the one-year interruption of 1979 is fragile.

53.9 Write the results

How to write it: results

We estimate a panel Markov-switching VAR for GDP growth and unemployment in 14 OECD countries, 1966–1990, with a single regime path shared by all countries. The data prefer two regimes to a linear panel VAR (BIC 2063.9 against 2162.7; likelihood-ratio statistic 139.6, Davies $p < 0.001$); the information criteria favour three regimes, which we report as a robustness check. The slow-growth regime holds 62.5% of the years, with growth 0.7 points below and unemployment 2.3 points above the country means, and an expected duration of 7.3 years; the fast-growth regime holds 37.5% of the years, with growth 1.7 points above and unemployment 2.0 points below the country means, and an expected duration of 3.1 years. The slow-growth regime is dated 1974–75, 1977–78 and 1980–1990; the fast-growth regime is dated 1967–1973, 1976 and 1979. The regime dating is robust to the specification (intercepts only, all coefficients switching, two lags, no demeaning) and to dropping any one country (agreement with the baseline of 96% to 100% of the years); the one-year interruption of 1979 is fragile, and the three-regime solution is different (58% agreement). Unemployment is I(1) in these data and has residual autocorrelation, so the regimes may partly reflect its persistence, and standard errors are approximate.

Report	What	Where
Model	Variables, lags, what switches, shared regime path, demeaning, estimation	Section 53.1
Data	Unit-root tests, panel size	Section 53.2
Regimes	AIC, BIC for one to three regimes; comparison with the linear panel VAR	Section 53.3
Estimates	Regime intercepts and means, covariances, transition probabilities, durations	Section 53.4
Dates	Regime dates, probabilities, figures	Section 53.5
Responses	Regime-specific impulse responses (ordering stated)	Section 53.6
Diagnostics	Residual tests	Section 53.7
Robustness	Specification, number of regimes, leave-one-country-out	Section 53.8

References. Krolzig (1997), *Markov-Switching Vector Autoregressions*, Springer; Hamilton (1989), *Econometrica*; Hamilton and Owyang (2012), *Review of Economics and Statistics*; Garrett (1998), *Partisan Politics in the Global Economy*, Cambridge University Press; Ang and Bekaert (2002), *Review of Financial Studies*.

PART VIII

DSGE Models with Dynare

Overview

Dynamic stochastic general-equilibrium (DSGE) models put structure on the macroeconomy: households and firms optimise, markets clear, and shocks move the economy around a steady state. The questions they answer are of a different kind from those of the earlier parts. A VAR shows how variables co-move; a DSGE model says *why*, in terms of preferences, technology and frictions, and what would change under a different policy.

The part follows the three uses of the model in Chapter 1. The first four chapters *explain a mechanism*: the real business cycle model (Kydland and Prescott, 1982), the three-equation New Keynesian model (Galí, 2015), a small open economy (Galí and Monacelli, 2005), and optimal monetary policy under commitment and discretion. The next three *estimate*: identification comes first (Iskrev, 2010), then Bayesian estimation of a small model (An and Schorfheide, 2007), then a medium-scale model (Smets and Wouters, 2007). The last two treat *nonlinearity*: an occasionally binding zero lower bound and higher-order solutions for risk and welfare. All of the models are solved in Dynare (Adjemian et al., 2011), and each chapter prints the model file.

Chapters at a glance

Ch.	Chapter	Use it when
54	Real Business Cycle Model	Explain a mechanism, no nominal rigidities
55	Three-Equation New Keynesian Model	Explain a mechanism with nominal rigidities
56	Small Open-Economy New Keynesian Model	Explain a mechanism in an open economy
57	Optimal Monetary Policy	Design policy: commitment, discretion or simple rules
58	Identification Before Estimation	Estimate with data: check which parameters the data can pin down
59	Bayesian Estimation of a New Keynesian Model	Estimate with data: a small model, 3 observables
60	Medium-Scale New Keynesian Model	Estimate with data: a richer model, 5 observables
61	The Zero Lower Bound with OccBin	Nonlinearity: an occasionally binding constraint
62	Higher-Order Solutions and Welfare	Nonlinearity: risk, large shocks, welfare

Further reading

Galí (2015) is the standard textbook for the New Keynesian model and Woodford (2003) the more formal treatment of monetary policy. Herbst and Schorfheide (2016) cover Bayesian estimation of DSGE models in depth. The medium-scale model follows Smets and Wouters (2007) and Christiano et al. (2005). Solution methods are in Blanchard and Kahn (1980) and, for higher orders, Schmitt-Grohé and Uribe (2004). The OccBin method is due to Guerrieri and Iacoviello (2015).

CHAPTER 54

Real Business Cycle Model

Applies when. Explain a mechanism, no nominal rigidities

Overview. Use this chapter when the question is about real mechanisms (productivity, capital accumulation, labour supply) and money and price rigidities play no role. It builds the canonical real business cycle model in Dynare from equations to steady state, checks the solution, computes impulse responses and compares the model's business-cycle moments with U.S. data.

Data. Calibrated to the U.S. economy; moments compared with U.S. data, 1966–2007

Software. Dynare 6 · Octave / MATLAB

54.1 The question and the model

Can productivity shocks alone generate business cycles that look like the data? That was the question of Kydland and Prescott (1982), and the real business cycle (RBC) model is still the starting point of every DSGE model: the New Keynesian models in the later chapters add frictions to exactly this core.

A representative household chooses consumption C_t, hours H_t and investment to maximise

$$E_0 \sum_{t=0}^{\infty} \beta^t \left[\ln C_t - \psi \frac{H_t^{1+\varphi}}{1+\varphi} \right]$$

subject to $C_t + I_t = Y_t$, $K_{t+1} = (1-\delta)K_t + I_t$ and production $Y_t = A_t K_t^{\alpha} H_t^{1-\alpha}$, with productivity $\ln A_t = \rho \ln A_{t-1} + \varepsilon_t$. The equilibrium conditions are:

Condition	Equation
Euler equation	$\frac{1}{C_t} = \beta E_t \left[\frac{1}{C_{t+1}} \left(\alpha \frac{Y_{t+1}}{K_{t+1}} + 1 - \delta \right) \right]$
Labour supply	$\psi H_t^{\varphi} C_t = W_t = (1-\alpha) \frac{Y_t}{H_t}$
Production	$Y_t = A_t K_t^{\alpha} H_t^{1-\alpha}$
Capital accumulation	$K_{t+1} = (1-\delta)K_t + I_t$
Resource constraint	$Y_t = C_t + I_t$

Before writing any code

Write the model down completely: every agent's problem, every constraint, the first-order conditions and the market-clearing conditions. Count equations and endogenous variables: they must be equal. Most Dynare errors are mistakes in this step, not in the code.

54.2 Calibration

Parameter	Value	Source or target
β	0.99	Annual real interest rate of about 4%
α	0.33	Capital share of income
δ	0.025	Annual depreciation of about 10%
φ	1	Frisch elasticity of labour supply of 1
ψ	set in the steady state	Hours equal to 1/3 of the time endowment
ρ	0.95	Persistence of measured productivity (Solow residual)
σ_ε	0.007	Volatility of the Solow residual

How to write it: calibration

The model is calibrated at a quarterly frequency to the U.S. economy. The discount factor of 0.99 implies an annual real interest rate of about 4 percent, the capital share is 0.33, and the depreciation rate of 0.025 corresponds to 10 percent per year. The labour-disutility parameter is chosen so that households work one third of their time in the steady state. The productivity process has persistence 0.95 and innovation standard deviation 0.007, consistent with estimates for the Solow residual.

54.3 Write the model in Dynare

```
// Real business cycle model. Variables are in logs, so impulse responses are percentage deviations.
var y c i k h w r a;
varexo e;
parameters beta alpha delta phi psi rho sigma_e h_ss;

beta = 0.99;  alpha = 0.33;  delta = 0.025;  phi = 1;
rho = 0.95;   sigma_e = 0.007;  h_ss = 1/3;

model;
  exp(-c) = beta * exp(-c(+1)) * (alpha * exp(y(+1) - k) + 1 - delta);    // Euler equation (k is K_{t+1})
  psi * exp(phi*h) * exp(c) = exp(w);                                     // labour supply
  exp(w) = (1 - alpha) * exp(y - h);                                      // wage = marginal product of labour
  exp(r) = alpha * exp(y - k(-1));                                        // rental rate = marginal product of
  ↪ capital
  exp(y) = exp(a) * exp(alpha*k(-1)) * exp((1 - alpha)*h);                // production
  exp(k) = (1 - delta) * exp(k(-1)) + exp(i);                             // capital accumulation
  exp(y) = exp(c) + exp(i);                                               // resource constraint
  a = rho * a(-1) + e;                                                    // productivity
end;

steady_state_model;
  // analytical steady state: hours are fixed at h_ss and psi is backed out
  r = log(1/beta - 1 + delta);
  KY = alpha / exp(r);
  k_h = KY^(1/(1 - alpha));
  h = log(h_ss);
  k = log(k_h * h_ss);
  y = log(k_h^alpha * h_ss);
  i = log(delta * exp(k));
  c = log(exp(y) - exp(i));
  w = log((1 - alpha) * exp(y) / h_ss);
  psi = exp(w) / (h_ss^phi * exp(c));
  a = 0;
end;

shocks;
  var e; stderr sigma_e;
end;
```

```
steady;
check;
stoch_simul(order=1, irf=40, hp_filter=1600, periods=0) y c i h w;
```

```
dynare rbc
STEADY-STATE RESULTS:

y          0.00509623
c          -0.26283
i          -1.44292
k          2.24596
h          -1.09861
w          0.703231
r          -3.34953
a          0

EIGENVALUES:
         Modulus             Real        Imaginary
            0.95             0.95                0
          0.9531           0.9531                0
            1.06             1.06                0
       7.959e+15        7.959e+15                0

There are 2 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition is verified.

MODEL SUMMARY

  Number of variables:         8
  Number of stochastic shocks: 1
  Number of state variables:   2
  Number of jumpers:           2
  Number of static variables:  4

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables           e
e            0.000049

POLICY AND TRANSITION FUNCTIONS
                                    y               c               i               h               w
Constant                     0.005096       -0.262830       -1.442920       -1.098612        0.703231
k(-1)                        0.217624        0.553075       -0.874157       -0.167725        0.385349
a(-1)                        1.243572        0.367237        4.095753        0.438168        0.805405
e                            1.309024        0.386565        4.311319        0.461229        0.847794

THEORETICAL MOMENTS (HP filter, lambda = 1600)
VARIABLE         MEAN  STD. DEV.   VARIANCE
y              0.0051     0.0120     0.0001
c             -0.2628     0.0040     0.0000
i             -1.4429     0.0394     0.0016
h             -1.0986     0.0042     0.0000
w              0.7032     0.0079     0.0001

MATRIX OF CORRELATIONS (HP filter, lambda = 1600)
Variables         y       c       i       h       w
y            1.0000  0.9088  0.9902  0.9799  0.9942
c            0.9088  1.0000  0.8417  0.8074  0.9483
i            0.9902  0.8417  1.0000  0.9982  0.9695
h            0.9799  0.8074  0.9982  1.0000  0.9529
w            0.9942  0.9483  0.9695  0.9529  1.0000
```

How to read it

- **Blocks.** `var` declares the endogenous variables, `varexo` the shocks, `parameters` the parameters. The `model` block contains one equation per endogenous variable. Variables dated `(+1)` are expectations, `(-1)` lags. Capital `k` is the end-of-period stock, so production uses `k(-1)`.

- **Logs.** Writing `exp(y)` wherever Y appears makes `y` the log of output, so all results are percentage deviations from the steady state.
- **`steady_state_model`.** The steady state is computed analytically, which is faster and more reliable than a numerical solver. The parameter `psi` is set here so that hours equal 1/3: a standard way to calibrate a parameter to a target.
- **`steady` and `check`** compute the steady state and verify the Blanchard–Kahn conditions. **`stoch_simul`** solves the model by a first-order perturbation and reports moments of HP-filtered variables (`hp_filter=1600`), comparable with HP-filtered data.

54.4 Check the steady state and the solution

```
names = cellstr(M_.endo_names);
fprintf('%-6s %10s %10s\n', 'var', 'log', 'level');
for j = 1:numel(names)
  fprintf('%-6s %10.4f %10.4f\n', names{j}, oo_.steady_state(j), exp(oo_.steady_state(j)));
end
fprintf('\nK/Y (quarterly) = %.2f   I/Y = %.3f   C/Y = %.3f   psi = %.3f\n', ...
  exp(oo_.steady_state(4) - oo_.steady_state(1)), exp(oo_.steady_state(3) - oo_.steady_state(1)), ...
  exp(oo_.steady_state(2) - oo_.steady_state(1)), M_.params(strcmp(cellstr(M_.param_names), 'psi')));
ev = abs(oo_.dr.eigval);
fprintf('Eigenvalues larger than 1 in modulus: %d   forward-looking variables: %d\n', sum(ev > 1 + 1e-6), M_.
  ↪ nfwrd + M_.nboth);
```

```
var           log      level
y          0.0051     1.0051
c         -0.2628     0.7689
i         -1.4429     0.2362
k          2.2460     9.4495
h         -1.0986     0.3333
w          0.7032     2.0203
r         -3.3495     0.0351
a          0.0000     1.0000

K/Y (quarterly) = 9.40   I/Y = 0.235   C/Y = 0.765   psi = 7.883
Eigenvalues larger than 1 in modulus: 2   forward-looking variables: 2
```

How to read it

- **Steady state.** A capital–output ratio of about 9.4 (quarterly, so about 2.4 annually), an investment share of about 24% and a consumption share of about 76% are close to the long-run ratios of the U.S. economy. Check these ratios in every model before going further: a wrong steady state invalidates everything after it.
- **Blanchard–Kahn condition.** A unique stable solution exists when the number of eigenvalues outside the unit circle equals the number of forward-looking variables. Here they match, so the model is determinate. `check` stops with an error when they do not; Chapter 55 shows such a failure.

54.5 Impulse responses to a productivity shock

```
vars = {'y', 'c', 'i', 'h', 'w'};
labels = {'Output', 'Consumption', 'Investment', 'Hours', 'Real wage'};
f = figure('position', [0 0 860 330]);
for j = 1:5
  subplot(2, 3, j);
  plot(0:39, 100 * oo_.irfs.([vars{j} '_e'])); hold on; plot([0 39], [0 0], 'k-', 'linewidth', 0.5);
  title(labels{j}); xlim([0 39]); grid on;
end
par = @(n) M_.params(strcmp(cellstr(M_.param_names), n));
subplot(2, 3, 6); plot(0:39, 100 * par('sigma_e') * par('rho').^(0:39)); title('Productivity'); xlim([0 39]);
  ↪ grid on;
print(f, '-dpng', '-r110', 'irf.png');
fprintf('Impact responses (percent): output %.2f, consumption %.2f, investment %.2f, hours %.2f\n', ...
  100*oo_.irfs.y_e(1), 100*oo_.irfs.c_e(1), 100*oo_.irfs.i_e(1), 100*oo_.irfs.h_e(1));
```

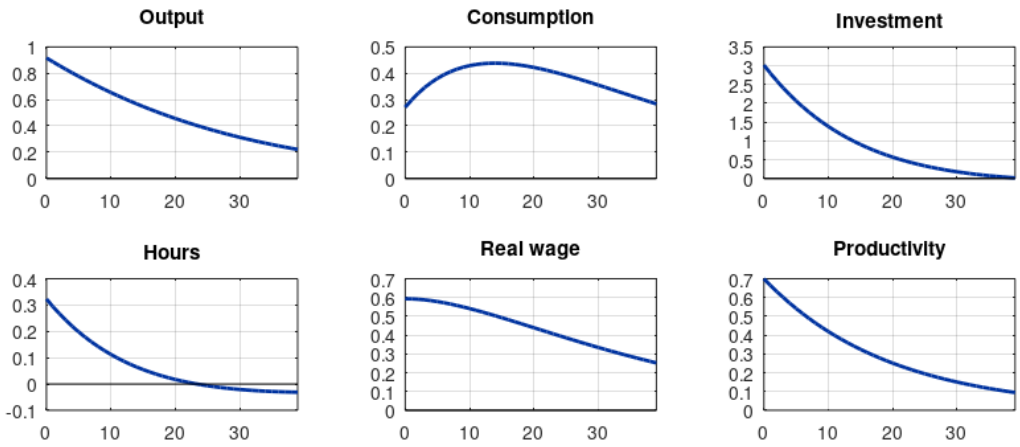


Figure 54.1. Impulse responses of the real business cycle model to a productivity shock, in percent deviations from the steady state.

```
Impact responses (percent): output 0.92, consumption 0.27, investment 3.02, hours 0.32
```

How to read it

A one-standard-deviation productivity shock (0.7%) raises output on impact by more than the shock itself, because hours and capital respond. **Investment moves far more than consumption**: households smooth consumption and channel the temporary windfall into capital. Hours rise because the wage is temporarily high (intertemporal substitution). These are the qualitative predictions to compare with the data.

54.6 Compare the model with the data

```
data = csvread('us_levels.csv', 1, 1);           % log output, consumption, investment, 1966Q1-2007Q4
[~, cyc] = sample_hp_filter(data, 1600);         % Dynare's HP filter
sd_data = 100 * std(cyc);
cm = corrcoef(cyc); corr_data = cm(1, :);
sd_model = 100 * sqrt(diag(oo_.var(1:3, 1:3)))';
corr_model = oo_.var(1, 1:3) ./ sqrt(oo_.var(1, 1) * diag(oo_.var(1:3, 1:3))');
names = {'Output', 'Consumption', 'Investment'};
fprintf('%-12s %10s %10s %12s %12s\n', '', 'sd data', 'sd model', 'corr(y) data', 'corr(y) model');
for j = 1:3
  fprintf('%-12s %10.2f %10.2f %12.2f %12.2f\n', names{j}, sd_data(j), sd_model(j), corr_data(j), corr_model(j)
  ↪ );
end
fprintf('\nRelative volatility (sd / sd of output): consumption data %.2f model %.2f; investment data %.2f
  ↪ model %.2f\n', ...
  sd_data(2)/sd_data(1), sd_model(2)/sd_model(1), sd_data(3)/sd_data(1), sd_model(3)/sd_model(1));
```

```
                sd data   sd model corr(y) data corr(y) model
Output             1.49       1.20         1.00         1.00
Consumption        1.22       0.40         0.86         0.91
Investment         6.35       3.94         0.91         0.99

Relative volatility (sd / sd of output): consumption data 0.82 model 0.34; investment data 4.25 model 3.29
```

How to read it

This is the classic RBC table. The model is judged on second moments it was **not** calibrated to match.

- **Output volatility.** The model produces about 80% of the observed volatility of output (1.20% against 1.49%) with productivity shocks alone: the original RBC finding.
- **Relative volatility.** As in the data, investment is several times more volatile than output (3.3 times in the model, 4.3 in the data) and consumption less volatile. The model gets the ranking right.

- **Comovement.** Consumption and investment are strongly procyclical in both.
- **Failures.** The model's consumption is far too smooth (a relative volatility of 0.34 against 0.82 in the data) and its correlations too close to one: with a single shock, everything moves together. These failures motivate the extra shocks and frictions of the medium-scale model.

54.7 Write the paper

How to write it: results

Figure 1 shows the responses to a one-standard-deviation productivity shock. Output rises on impact by more than productivity itself, as hours and, gradually, the capital stock increase; investment responds several times more strongly than consumption, reflecting households' desire to smooth consumption. Table 2 compares the model's business-cycle moments with HP-filtered U.S. data for 1966–2007. Productivity shocks alone account for most of the observed volatility of output and reproduce the ranking of volatilities: investment is far more volatile than output, and consumption less. The model, however, generates too little volatility in consumption and too much comovement among the aggregates.

Common mistakes

- Timing of capital: Dynare uses end-of-period stocks, so production uses `k(-1)`. Getting this wrong produces Blanchard–Kahn errors.
- Comparing model moments with unfiltered data, or filtering one and not the other.
- Calibrating a parameter to a moment and then claiming the model "explains" that moment.
- Solving for the steady state numerically when it has a closed form: a wrong initial value can give a wrong steady state without an error message.

CHAPTER 55

Three-Equation New Keynesian Model

Applies when. Explain a mechanism with nominal rigidities
Overview. Use this chapter when the question is about monetary policy, inflation or demand shocks, so sticky prices matter. It writes the three-equation New Keynesian model in Dynare, checks determinacy (and shows what an indeterminate model looks like), computes impulse responses and variance decompositions, and compares policy rules.
Data. Calibrated quarterly model (Galí 2015 calibration)
Software. Dynare 6 · Octave / MATLAB

55.1 The question and the model

How does a monetary policy shock move output and inflation, and how should the central bank respond to shocks? The three-equation New Keynesian model is the workhorse for this question. It adds monopolistic competition and Calvo price setting to the household and firm problems of the RBC model, and replaces money with an interest-rate rule. Log-linearised around a zero-inflation steady state:

$$
\begin{aligned}
x_t &= E_t x_{t+1} - \tfrac{1}{\sigma}\left(i_t - E_t \pi_{t+1} - r_t^n\right) && \text{dynamic IS curve} \\
\pi_t &= \beta E_t \pi_{t+1} + \kappa x_t + u_t && \text{New Keynesian Phillips curve} \\
i_t &= \rho_i i_{t-1} + (1-\rho_i)(\phi_\pi \pi_t + \phi_x x_t) + v_t && \text{Taylor rule}
\end{aligned}
$$

with $\kappa = \frac{(1-\theta)(1-\beta\theta)}{\theta}(\sigma + \varphi)$. Here x_t is the output gap, π_t inflation, i_t the nominal rate, r_t^n the natural rate of interest (a demand shock), u_t a cost-push shock and v_t a monetary policy shock.

Parameter	Value	Meaning
β	0.99	Discount factor
σ	1	Inverse intertemporal elasticity of substitution
φ	1	Inverse Frisch elasticity
θ	0.75	Calvo probability of not resetting prices (average price duration of 4 quarters)
ϕ_π, ϕ_x	1.5, 0.125	Taylor-rule responses to inflation and the output gap
ρ_i	0.8	Interest-rate smoothing

Why the model is written in gaps and deviations

Every variable is a deviation from a zero-inflation steady state, so the steady state is zero and no `steady_state_model` is needed. The economics lies in the coefficients: κ summarises how strongly inflation reacts to the output gap, and it falls quickly as prices become stickier.

55.2 Write the model in Dynare

```
// Three-equation New Keynesian model. Inflation and interest rates are quarterly rates in percent;
// the reports below annualise them.
var x pi i rn u v;
varexo e_rn e_u e_v;
parameters beta sigma varphi theta kappa phi_pi phi_x rho_i rho_rn rho_u;
```

```
beta = 0.99;  sigma = 1;  varphi = 1;  theta = 0.75;
kappa = (1 - theta) * (1 - beta*theta) / theta * (sigma + varphi);
phi_pi = 1.5;  phi_x = 0.125;  rho_i = 0.8;
rho_rn = 0.8;  rho_u = 0.5;

model(linear);
  x  = x(+1) - (1/sigma) * (i - pi(+1) - rn);                      // dynamic IS curve
  pi = beta * pi(+1) + kappa * x + u;                               // Phillips curve
  i  = rho_i * i(-1) + (1 - rho_i) * (phi_pi*pi + phi_x*x) + v;     // Taylor rule
  rn = rho_rn * rn(-1) + e_rn;                                      // natural-rate (demand) shock
  u  = rho_u * u(-1) + e_u;                                         // cost-push shock
  v  = e_v;                                                         // monetary policy shock
end;

shocks;
  var e_rn; stderr 0.5;
  var e_u;  stderr 0.2;
  var e_v;  stderr 0.25;
end;

steady;
check;
stoch_simul(order=1, irf=20) x pi i;
```

```
dynare nk
STEADY-STATE RESULTS:

x           0
pi          0
i           0
rn          0
u           0
v           0

EIGENVALUES:
         Modulus             Real        Imaginary
             0.5              0.5                0
          0.5345           0.5345                0
             0.8              0.8                0
           1.101            1.101                0
           1.373            1.373                0

There are 2 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition is verified.

MODEL SUMMARY

  Number of variables:         6
  Number of stochastic shocks: 3
  Number of state variables:   3
  Number of jumpers:           2
  Number of static variables:  1

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables        e_rn       e_u       e_v
e_rn         0.250000  0.000000  0.000000
e_u          0.000000  0.040000  0.000000
e_v          0.000000  0.000000  0.062500

POLICY AND TRANSITION FUNCTIONS
                                    x             pi              i
i(-1)                       -1.975490       -0.720289        0.534526
rn(-1)                       1.975490        0.720289        0.265474
u(-1)                       -0.585845        0.570120        0.156390
e_rn                         2.469363        0.900361        0.331842
e_u                         -1.171690        1.140240        0.312780
```

```
e_v                       -2.469363       -0.900361        0.668158

THEORETICAL MOMENTS
VARIABLE         MEAN  STD. DEV.   VARIANCE
x              0.0000     1.6861     2.8430
pi             0.0000     0.6418     0.4119
i              0.0000     0.5646     0.3187

VARIANCE DECOMPOSITION (in percent)
        e_rn     e_u     e_v
x      75.07    6.16   18.77
pi     68.89   13.89   17.22
i      83.78    3.96   12.26

MATRIX OF CORRELATIONS
Variables         x      pi       i
x            1.0000  0.8349  0.3253
pi           0.8349  1.0000  0.4101
i            0.3253  0.4101  1.0000
```

How to read it

- **`model(linear)`** tells Dynare the equations are already linear, so it skips the linearisation.
- **The check.** The model has two forward-looking variables (the output gap and inflation) and two eigenvalues outside the unit circle: the Blanchard–Kahn conditions hold, and the equilibrium is unique.
- **Variance decomposition.** In this calibration demand shocks dominate both the output gap (75%) and inflation (69%); cost-push shocks explain about 14% of inflation and monetary shocks under a fifth of each. With different shock variances these shares change: they are a property of the calibration, not a finding.

55.3 Impulse responses

```
shocks = {'e_v', 'e_rn', 'e_u'};
titles = {'Monetary policy shock (+25 bp)', 'Demand shock', 'Cost-push shock'};
f = figure('position', [0 0 900 290]);
for s = 1:3
  subplot(1, 3, s);
  plot(0:19, oo_.irfs.(['x_' shocks{s}])); hold on;
  plot(0:19, 4 * oo_.irfs.(['pi_' shocks{s}]), '--');
  plot(0:19, 4 * oo_.irfs.(['i_' shocks{s}]), ':');
  plot([0 19], [0 0], 'k-', 'linewidth', 0.5); title(titles{s}); grid on; xlim([0 19]);
  if s == 1, legend('Output gap', 'Inflation (annual)', 'Interest rate (annual)', 'location', 'southeast');
  ↪ end
end
print(f, '-dpng', '-r110', 'irf.png');
fprintf('Monetary shock, impact: output gap %.2f%%, annual inflation %.2f pp, annual rate %.2f pp\n', ...
        oo_.irfs.x_e_v(1), 4*oo_.irfs.pi_e_v(1), 4*oo_.irfs.i_e_v(1));
```

```
Monetary shock, impact: output gap -0.62%, annual inflation -0.90 pp, annual rate 0.67 pp
```

How to read it

- **Monetary policy shock.** A surprise increase in the policy rate lowers the output gap and inflation on impact. Because inflation and the gap fall, the Taylor rule partly offsets the shock, so the interest rate rises by less than the shock itself.
- **Demand shock.** A rise in the natural rate raises the output gap and inflation together; the central bank raises the rate. Demand shocks move output and inflation in the **same** direction.
- **Cost-push shock.** Inflation rises while the output gap falls: the central bank faces a **trade-off**, the reason inflation targeting is hard in practice.

55.4 Comparing policy rules

```
par = @(n) find(strcmp(cellstr(M_.param_names), n));
```

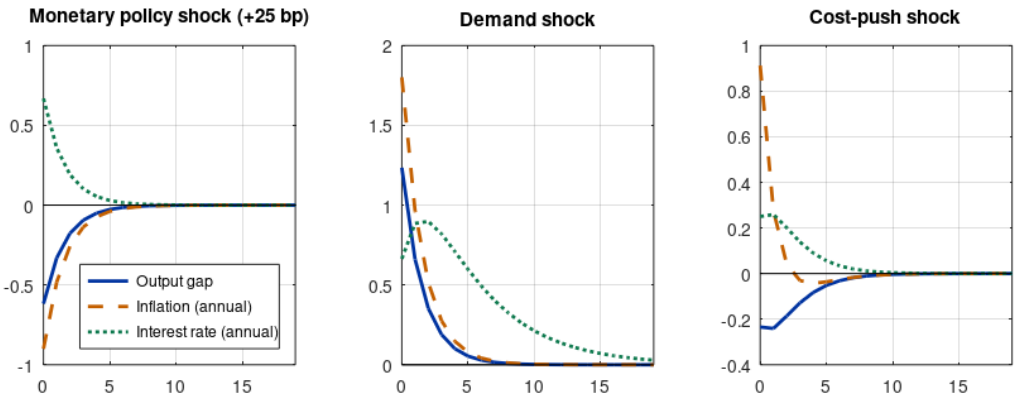


Figure 55.1. Impulse responses of the output gap, annualised inflation and the annualised interest rate to a monetary-policy shock, a demand shock and a cost-push shock.

```
rules = [1.5 0.125; 3.0 0.125; 1.5 0.5];
fprintf('%-26s %10s %10s %10s\n', 'Rule (phi_pi, phi_x)', 'sd(x)', 'sd(pi) ann', 'sd(i) ann');
options_.noprint = 1; options_.nograph = 1;
for k = 1:size(rules, 1)
  M_.params(par('phi_pi')) = rules(k, 1); M_.params(par('phi_x')) = rules(k, 2);
  [info, oo_, options_] = stoch_simul(M_, options_, oo_, {'x', 'pi', 'i'});
  sd = sqrt(diag(oo_.var));
  fprintf('%-26s %10.3f %10.3f %10.3f\n', sprintf('(%.2f, %.3f)', rules(k, :)), sd(1), 4*sd(2), 4*sd(3));
end
M_.params(par('phi_pi')) = 1.5; M_.params(par('phi_x')) = 0.125;
```

```
Rule (phi_pi, phi_x)           sd(x) sd(pi) ann  sd(i) ann
(1.50, 0.125)                  1.686      2.567      2.258
(3.00, 0.125)                  1.256      1.655      2.448
(1.50, 0.500)                  1.240      2.028      2.374
```

How to read it

Doubling the response to inflation lowers the volatility of **both** inflation and the output gap. That is because demand shocks dominate in this calibration, and for demand shocks there is no trade-off: stabilising inflation also stabilises the gap (the "divine coincidence"). A stronger response to the output gap also lowers both, by less for inflation. The policy rate becomes more volatile in both cases. A trade-off appears only through cost-push shocks; with larger cost-push shocks, the ranking of rules depends on how society weighs the two goals, which is the subject of Chapter 57.

55.5 Determinacy: the Taylor principle

```
var x pi i rn u v;
varexo e_rn e_u e_v;
parameters beta sigma varphi theta kappa phi_pi phi_x rho_i rho_rn rho_u;
beta = 0.99;  sigma = 1;  varphi = 1;  theta = 0.75;
kappa = (1 - theta) * (1 - beta*theta) / theta * (sigma + varphi);
phi_pi = 0.9;  phi_x = 0.125;  rho_i = 0.8;  rho_rn = 0.8;  rho_u = 0.5;     // passive policy: phi_pi < 1
model(linear);
  x  = x(+1) - (1/sigma) * (i - pi(+1) - rn);
  pi = beta * pi(+1) + kappa * x + u;
  i  = rho_i * i(-1) + (1 - rho_i) * (phi_pi*pi + phi_x*x) + v;
  rn = rho_rn * rn(-1) + e_rn;  u = rho_u * u(-1) + e_u;  v = e_v;
end;
steady;
check;
try
  dynare nk_passive noclearall nograph nolog
catch err
```

```
  disp('Dynare stopped with the message:'); disp(err.message);
end
```

```
STEADY-STATE RESULTS:

x           0
pi          0
i           0
rn          0
u           0
v           0

EIGENVALUES:
         Modulus             Real        Imaginary
             0.5              0.5                0
          0.5612           0.5612                0
             0.8              0.8                0
          0.9842           0.9842                0
           1.463            1.463                0

There are 1 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition ISN'T verified!
```

How to read it

Dynare reports one eigenvalue outside the unit circle for two forward-looking variables: "The rank condition ISN'T verified!". With $\phi_\pi = 0.9$, the central bank raises the nominal rate by less than one-for-one with inflation, so the **real** rate falls when inflation rises. Self-fulfilling expectations of inflation are then possible, and the model has many equilibria (it is **indeterminate**). This is the **Taylor principle**: the response to inflation must exceed one (with a small output-gap response, roughly $\phi_\pi > 1$). When you see this Blanchard–Kahn error in your own model, check the policy rule before anything else.

55.6 Write the paper

How to write it: results

Figure 1 shows the responses to monetary, demand and cost-push shocks in the calibrated model. A 25-basis-point monetary tightening lowers the output gap and annualised inflation on impact, and its effects dissipate within a few quarters. Demand shocks move output and inflation in the same direction, while cost-push shocks create a trade-off between stabilising inflation and stabilising the output gap. The model is determinate under the baseline rule; with a response to inflation below one, the equilibrium is indeterminate, as implied by the Taylor principle.

Common mistakes

- Mixing quarterly and annual rates. Write the model in quarterly rates and annualise only when reporting.
- Reporting impulse responses from an indeterminate model. Dynare refuses to solve it; do not "fix" this by changing timing until it runs.
- Treating the variance decomposition of a calibrated model as evidence. It reflects the assumed shock variances.
- Forgetting interest-rate smoothing, which makes the response of the policy rate gradual, as in the data.

CHAPTER 56

Small Open-Economy New Keynesian Model

Applies when. Explain a mechanism in an open economy

Overview. Use this chapter when exchange rates, terms of trade or imported inflation matter. It writes the Galí–Monacelli small open-economy model in Dynare, uses the macro language to switch between monetary regimes, and compares domestic-inflation targeting, CPI-inflation targeting and an exchange-rate peg by volatility and welfare.

Data. Calibrated quarterly model (Galí and Monacelli 2005, Canada calibration)

Software. Dynare 6 · Octave / MATLAB

56.1 The question and the model

Should a small open economy target domestic inflation, CPI inflation, or the exchange rate? Many emerging-market central banks face exactly this choice. Galí and Monacelli (2005) answered it in a New Keynesian model in which home and foreign goods are imperfect substitutes and the home economy is too small to affect the rest of the world.

With the parameter values of Cole and Obstfeld (unit elasticities, $\sigma = \eta = \gamma = 1$), the log-linear model reduces to:

$$\tilde{y}_t = E_t\tilde{y}_{t+1} - \tfrac{1}{\sigma_\alpha}\left(i_t - E_t\pi_{H,t+1} - \bar{r}^n_t\right) \quad \text{IS curve, in terms of domestic inflation}$$

$$\pi_{H,t} = \beta E_t\pi_{H,t+1} + \kappa_\alpha\tilde{y}_t \quad \text{domestic Phillips curve}$$

$$s_t = \sigma_\alpha\, y_t \quad \text{terms of trade (world output fixed)}$$

$$\pi_t = \pi_{H,t} + \alpha\,\Delta s_t \quad \text{CPI inflation}$$

$$\Delta e_t = \Delta s_t + \pi_{H,t} \quad \text{nominal exchange rate (world prices fixed)}$$

where $\tilde{y}_t = y_t - \bar{y}^n_t$ is the output gap, $\bar{y}^n_t = \Gamma_a a_t$ natural output driven by domestic productivity a_t, $\bar{r}^n_t = -\sigma_\alpha\Gamma_a(1-\rho_a)a_t$ the natural rate, α the degree of openness, and $\kappa_\alpha = \lambda(\sigma_\alpha + \varphi)$.

Four regimes are compared:

Regime	Rule
DIT	Strict domestic-inflation targeting: $\pi_{H,t} = 0$, the optimal policy in this model
DITR	Domestic-inflation Taylor rule: $i_t = \phi_\pi\pi_{H,t}$
CITR	CPI-inflation Taylor rule: $i_t = \phi_\pi\pi_t$
PEG	Fixed exchange rate: $e_t = 0$

Why one model file with a switch

The regimes differ in one equation. Writing four files invites inconsistencies. Dynare's macro language (`@#if`, `@#define`) keeps one file and selects the rule at run time: `dynare soe -DRULE=2`.

56.2 Write the model in Dynare

```
// Galí-Monacelli (2005) small open economy, Cole-Obstfeld case. Quarterly rates in percent.
@#ifndef RULE
```

```
  @#define RULE = 1
@#endif
var x y yn pih pi s e i a rrn;
varexo eps_a;
parameters beta alpha theta varphi phi_pi rho_a sigma_a lambda kappa_a Gamma_a;

beta = 0.99;  alpha = 0.4;  theta = 0.75;  varphi = 3;  phi_pi = 1.5;  rho_a = 0.66;
sigma_a = 1;                                          // sigma/((1-alpha)+alpha*omega) = 1 in the Cole-
  ↪ Obstfeld case
lambda  = (1 - theta) * (1 - beta*theta) / theta;
kappa_a = lambda * (sigma_a + varphi);
Gamma_a = (1 + varphi) / (sigma_a + varphi);

model(linear);
  x   = x(+1) - (1/sigma_a) * (i - pih(+1) - rrn);       // IS curve
  pih = beta * pih(+1) + kappa_a * x;                     // domestic Phillips curve
  x   = y - yn;                                           // output gap
  yn  = Gamma_a * a;                                      // natural output
  rrn = -sigma_a * Gamma_a * (1 - rho_a) * a;             // natural real rate
  s   = sigma_a * y;                                      // terms of trade
  pi  = pih + alpha * (s - s(-1));                        // CPI inflation
  e - e(-1) = s - s(-1) + pih;                            // nominal exchange rate
  a   = rho_a * a(-1) + eps_a;                            // domestic productivity
@#if RULE == 0
  pih = 0;                                                // strict domestic-inflation targeting (the optimum)
@#elseif RULE == 1
  i = phi_pi * pih;                                       // DITR
@#elseif RULE == 2
  i = phi_pi * pi;                                        // CITR
@#else
  e = 0;                                                  // PEG
@#endif
end;

shocks;
  var eps_a; stderr 0.712;                                // percent, as in Galí-Monacelli
end;

steady;
check;
stoch_simul(order=1, irf=16, nograph) y pih pi s e i x;
```

```
dynare soe
Normalization failed with cutoff, trying symbolic normalization...
Could not normalize the static model. Variable e is not in the maximum cardinality matching.

STEADY-STATE RESULTS:

x            0
y            0
yn           0
pih          0
pi           0
s            0
e            0
i            0
a            0
rrn          0

EIGENVALUES:
        Modulus             Real        Imaginary
              0                0                0
           0.66             0.66                0
              1                1                0
          1.237            1.178           0.3762
```

```
          1.237           1.178         -0.3762

There are 2 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition is verified.

MODEL SUMMARY

  Number of variables:         10
  Number of stochastic shocks: 1
  Number of state variables:   3
  Number of jumpers:           2
  Number of static variables:  5

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables       eps_a
eps_a        0.506944
```

How to read it

The default run uses the domestic-inflation rule (`RULE = 1`). The model is determinate. Note how the exchange rate enters: the level `e` appears with a lag, so it is a state variable, and under the peg it is pinned at zero.

56.3 Impulse responses under the three regimes

```
names = {'DIT', 'DITR', 'CITR', 'PEG'};
vars = {'y', 'pih', 'pi', 's', 'e', 'i'};
titles = {'Output', 'Domestic inflation', 'CPI inflation', 'Terms of trade', 'Nominal exchange rate', '
  ↪ Interest rate'};
R = struct();
for r = 1:4
  clear functions;                                % Octave caches the compiled model; clear it before re-
  ↪ running
  evalc(sprintf('dynare soe noclearall nograph nolog -DRULE=%d', r - 1));   % run quietly
  for v = 1:numel(vars), R(r).(vars{v}) = oo_.irfs.([vars{v} '_eps_a']); end
  R(r).sd = sqrt(diag(oo_.var))'; R(r).x = oo_.irfs.x_eps_a;
end
f = figure('position', [0 0 900 420]);
for v = 1:numel(vars)
  subplot(2, 3, v); hold on;
  for r = 1:4, plot(0:15, R(r).(vars{v})); end
  plot([0 15], [0 0], 'k-', 'linewidth', 0.5); title(titles{v}); grid on; xlim([0 15]);
  if v == 1, legend(names, 'location', 'northeast'); end
end
print(f, '-dpng', '-r110', 'irf.png');
```

```
Normalization failed with cutoff, trying symbolic normalization...
Could not normalize the static model. Variable e is not in the maximum cardinality matching.
Normalization failed with cutoff, trying symbolic normalization...
Could not normalize the static model. Variable e is not in the maximum cardinality matching.
Normalization failed with cutoff, trying symbolic normalization...
Could not normalize the static model. Variable e is not in the maximum cardinality matching.
```

How to read it

A domestic productivity improvement lowers home production costs and raises natural output.

- **DIT (the optimum).** Domestic inflation stays at zero and output rises one-for-one with its natural level. The interest rate falls with the natural rate and the currency depreciates: the terms of trade worsen so that foreigners buy the extra home output. The exchange rate does all the adjusting, as it would with flexible prices.
- **DITR.** The simple rule cuts the rate too little, so output rises by less than its natural level, the output gap is negative and domestic inflation falls.

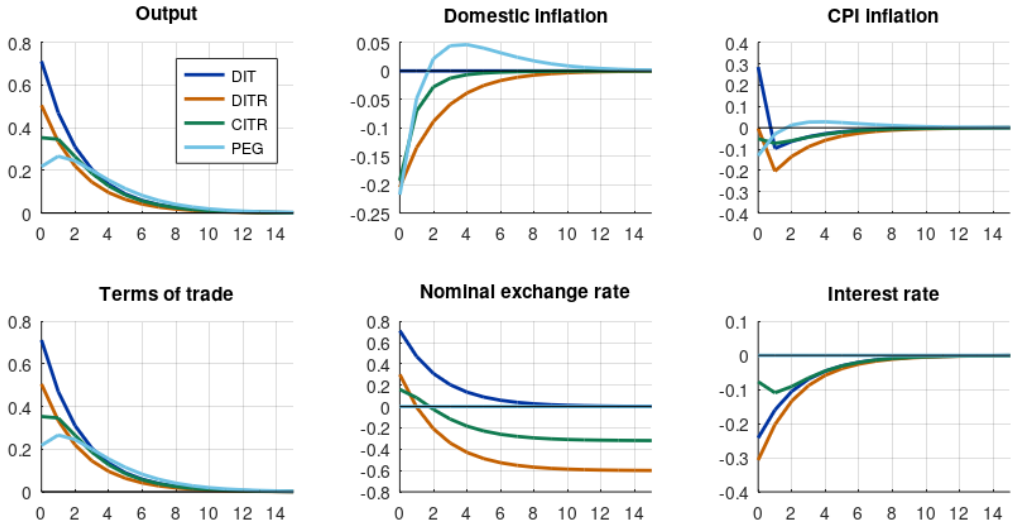


Figure 56.1. Responses of the small open economy to a shock under four monetary-policy regimes, labelled in the legend.

- **CITR.** CPI inflation includes the rise in import prices from the depreciation, so it falls less than domestic inflation and the central bank cuts the rate even less at first. Output rises less, but the gap closes faster.
- **PEG.** The nominal exchange rate cannot move, so the relative price must adjust through sticky domestic prices. Domestic inflation falls and the adjustment is slow.

56.4 Volatility and welfare

```
eps = 6;                                        % elasticity of substitution between goods
lam = (1 - 0.75) * (1 - 0.99*0.75) / 0.75;
fprintf('%-6s %8s %8s %8s %8s %8s %8s %10s\n', '', 'sd(y)', 'sd(piH)', 'sd(pi)', 'sd(s)', 'sd(e)', 'sd(i)', '
  ↪ loss');
for r = 1:4
  sd = R(r).sd;                                 % order: y pih pi s e i x
  loss = (1 - 0.4)/2 * ((eps/lam) * sd(2)^2 + (1 + 3) * sd(7)^2);
  fprintf('%-6s %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %10.4f\n', names{r}, sd(1:6), loss);
end
```

```
          sd(y)  sd(piH)   sd(pi)    sd(s)    sd(e)    sd(i)       loss
DIT       0.948    0.000    0.313    0.948      NaN    0.322     0.0000
DITR      0.673    0.272    0.271    0.673      NaN    0.408     1.6460
CITR      0.615    0.208    0.124    0.615      NaN    0.186     1.0808
PEG       0.518    0.241    0.145    0.518    0.000    0.000     1.5709
```

How to read it

The table follows Table 1 of Galí and Monacelli (2005). The welfare loss is their second-order approximation to household welfare, which penalises domestic-inflation variance (price dispersion) and output-gap variance:

$$\mathbb{L} = \frac{1-\alpha}{2}\left[\frac{\epsilon}{\lambda}\,\mathrm{var}(\pi_H) + (1+\varphi)\,\mathrm{var}(\tilde{y})\right]$$

- **The optimum.** Strict domestic-inflation targeting has zero loss: it reproduces the flexible-price allocation. Every simple rule falls short of it.
- **Among the simple rules**, the CPI rule does best in this calibration and the domestic-inflation Taylor rule worst, with the peg close to it. The ranking of simple rules is not universal: it depends on the response coefficient, the shock process and the degree of openness. Re-run the comparison under your own calibration rather than quoting a ranking.

- **The exchange rate** has a unit root under the floating regimes (its level drifts after every shock), so its unconditional standard deviation is infinite and Dynare reports `NaN`. Under the peg it is zero by construction.
- **Caveat.** These results rest on the Cole–Obstfeld assumptions, complete risk sharing and the absence of foreign-currency debt. Emerging-market models with currency mismatch or imported-input costs can change them substantially.

56.5 Write the paper

How to write it: results

Figure 1 compares the responses to a domestic productivity shock under three monetary regimes. Under domestic-inflation targeting, the nominal exchange rate depreciates and the terms of trade adjust, leaving domestic inflation and the output gap nearly unchanged. Under an exchange-rate peg, the adjustment of relative prices must occur through sticky domestic prices, generating larger fluctuations in domestic inflation and the output gap. Strict domestic-inflation targeting replicates the flexible-price allocation and is optimal. Among simple rules, a CPI-inflation Taylor rule yields a smaller welfare loss than a domestic-inflation Taylor rule or a peg in our calibration.

Common mistakes

- Writing the Phillips curve in CPI inflation. Firms set domestic prices; CPI inflation follows from the definition.
- Forgetting that the exchange-rate level is a state variable once a peg is possible.
- Ranking regimes by the volatility of the variables they target. Rank them by a welfare criterion.
- Using the Cole–Obstfeld special case and then generalising the ranking to economies with incomplete markets or dollar debt.

CHAPTER 57

Optimal Monetary Policy

Applies when. Design policy: commitment, discretion or simple rules

Overview. Use this chapter when the question is how policy should be set, not how it was set. It computes optimal policy under commitment (Ramsey) and under discretion, and an optimised simple Taylor rule, in one Dynare file, and compares them by the central bank's loss.

Data. Calibrated New Keynesian model with cost-push and demand shocks

Software. Dynare 6 · Octave / MATLAB

57.1 The question and the setup

How should a central bank respond to shocks, and how much does commitment matter? In the New Keynesian model, a second-order approximation to household welfare gives the central bank a quadratic loss in inflation and the output gap (Woodford 2003; Galí 2015):

$$\mathbb{L} = E_0 \sum_{t=0}^{\infty} \beta^t \left(\pi_t^2 + \lambda_x x_t^2\right), \qquad \lambda_x = \frac{\kappa}{\epsilon}$$

The constraints are the IS curve and the Phillips curve of Chapter 55, with a cost-push shock u_t and a natural-rate (demand) shock r_t^n. Three policies are compared:

Policy	Dynare command	Meaning
Commitment	`ramsey_model`	The central bank chooses the whole future path of policy once and sticks to it
Discretion	`discretionary_policy`	It re-optimises every period, taking future policy as given
Optimal simple rule	`osr`	The best Taylor rule $i_t = \phi_\pi \pi_t + \phi_x x_t$

Why the comparison matters

Commitment is the best the central bank can do, but it requires credibility: promises about future policy must be believed. Discretion is what happens without that credibility. A simple rule is what central banks can actually communicate. The gaps between the three losses measure the value of commitment and the cost of simplicity.

57.2 Write the model in Dynare

```
// Optimal policy in the New Keynesian model. Run with -DPOL=1 (commitment), 2 (discretion) or 3 (optimal
  ↪ simple rule).
@#ifndef POL
  @#define POL = 1
@#endif
var x pi i u rn;
varexo e_u e_rn;
parameters beta sigma theta varphi eps kappa lambda_x phi_pi phi_x rho_u rho_rn;

beta = 0.99;  sigma = 1;  theta = 0.75;  varphi = 1;  eps = 6;
kappa = (1 - theta) * (1 - beta*theta) / theta * (sigma + varphi);
lambda_x = kappa / eps;                                    // welfare weight on the output gap
```

```
phi_pi = 1.5;  phi_x = 0.125;  rho_u = 0.5;  rho_rn = 0.8;

model(linear);
  x  = x(+1) - (1/sigma) * (i - pi(+1) - rn);              // IS curve
  pi = beta * pi(+1) + kappa * x + u;                      // Phillips curve
  u  = rho_u * u(-1) + e_u;                                // cost-push shock
  rn = rho_rn * rn(-1) + e_rn;                             // natural-rate shock
@#if POL == 3
  i  = phi_pi * pi + phi_x * x;                            // simple rule, coefficients optimised below
@#endif
end;

shocks;
  var e_u;  stderr 0.2;
  var e_rn; stderr 0.5;
end;

@#if POL == 1
planner_objective pi^2 + lambda_x * x^2;
ramsey_model(planner_discount = 0.99, instruments = (i));
stoch_simul(order = 1, irf = 16, nograph) x pi i;
@#elseif POL == 2
planner_objective pi^2 + lambda_x * x^2;
discretionary_policy(instruments = (i), irf = 16, planner_discount = 0.99, nograph) x pi i;
@#else
optim_weights; pi 1; x lambda_x; end;
osr_params phi_pi phi_x;
osr_params_bounds; phi_pi, 1.01, 10; phi_x, 0, 5; end;
osr(opt_algo = 9, irf = 16, nograph) x pi i;
@#endif
```

```
dynare opt
MODEL SUMMARY

  Number of variables:         9
  Number of stochastic shocks: 2
  Number of state variables:   4
  Number of jumpers:           4
  Number of static variables:  1

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables         e_u      e_rn
e_u          0.040000  0.000000
e_rn         0.000000  0.250000
```

How to read it

- Under commitment and discretion there is **no policy rule** in the model block: the interest rate is the planner's instrument, and Dynare derives the optimality conditions itself. That is why the model has one equation fewer than variables.
- `planner_objective` states the period loss; `planner_discount` its discount factor.
- For the simple rule, the rule is in the model and `osr` searches over its coefficients to minimise the weighted sum of variances given in `optim_weights`. The bounds keep ϕ_π above 1 (determinacy) and within a range a central bank could announce.
- The output above is the commitment solution: note how the cost-push shock of the previous period still affects today's choices, a sign of the history dependence of commitment.

57.3 Solve under the three policies

```
pols = {'Commitment', 'Discretion', 'Simple rule'};
P = struct();
for p = 1:3
  clear functions;
```

```
  evalc(sprintf('dynare opt noclearall nograph nolog -DPOL=%d', p));
  lam = M_.params(strcmp(cellstr(M_.param_names), 'lambda_x'));
  sd = sqrt(diag(oo_.var));                    % moments of the listed variables: x, pi, i
  P(p).sdx = sd(1); P(p).sdpi = sd(2); P(p).sdi = sd(3);
  P(p).loss = P(p).sdpi^2 + lam * P(p).sdx^2;
  P(p).irf_x = oo_.irfs.x_e_u; P(p).irf_pi = oo_.irfs.pi_e_u; P(p).irf_i = oo_.irfs.i_e_u;
  if p == 3
    P(p).coef = M_.params(strcmp(cellstr(M_.param_names), 'phi_pi')) * [1 0] + M_.params(strcmp(cellstr(M_.
  ↪ param_names), 'phi_x')) * [0 1];
  end
end
fprintf('%-12s %8s %8s %8s %10s %12s\n', '', 'sd(x)', 'sd(pi)', 'sd(i)', 'loss', 'vs commitment');
for p = 1:3
  fprintf('%-12s %8.3f %8.3f %8.3f %10.4f %11.1f%%\n', pols{p}, P(p).sdx, P(p).sdpi, P(p).sdi, P(p).loss,
  ↪ 100*(P(p).loss/P(1).loss - 1));
end
fprintf('\nOptimised simple rule: phi_pi = %.2f, phi_x = %.2f\n', P(3).coef);
```

```
                sd(x)   sd(pi)    sd(i)       loss vs commitment
Commitment      0.843    0.102    0.858     0.0306         0.0%
Discretion      0.903    0.150    0.986     0.0459        50.0%
Simple rule     1.059    0.130    1.052     0.0489        59.8%

Optimised simple rule: phi_pi = 10.00, phi_x = 0.39
```

How to read it

- **Commitment** has the lowest loss by construction.
- **Discretion** is worse: the stabilisation bias. Without the ability to promise future restraint, the central bank must fight a cost-push shock entirely with today's output gap, while under commitment it spreads the adjustment over time by promising to keep policy tight after the shock fades.
- **The optimised simple rule** does worse than both. It hits the upper bound of $\phi_\pi = 10$: optimal policy fully offsets demand shocks (the divine coincidence), and a rule can only approximate that by responding ever more strongly to inflation. Even so, a rule that reacts only to current inflation and the gap cannot reproduce the history dependence of commitment, the promise to stay tight after a cost-push shock has passed. Adding the lagged interest rate or the price level to the rule is the standard way to close much of this gap.
- **Volatility of the interest rate** is not in the loss here. Adding a term in i_t^2 to the objective is the standard way to obtain smoother, more realistic optimal rules.

57.4 Responses to a cost-push shock

```
f = figure('position', [0 0 900 290]);
t = 0:15; vars = {'irf_x', 'irf_pi', 'irf_i'}; titles = {'Output gap', 'Inflation (quarterly)', 'Interest rate
  ↪  (quarterly)'};
for v = 1:3
  subplot(1, 3, v); hold on;
  for p = 1:3, plot(t, P(p).(vars{v})); end
  plot([0 15], [0 0], 'k-', 'linewidth', 0.5); title(titles{v}); grid on; xlim([0 15]);
  if v == 1, legend(pols, 'location', 'southeast'); end
end
print(f, '-dpng', '-r110', 'costpush.png');
fprintf('Cumulative change in the price level after 16 quarters: commitment %.3f, discretion %.3f\n', sum(P(1).
  ↪ irf_pi), sum(P(2).irf_pi));
```

```
Cumulative change in the price level after 16 quarters: commitment 0.000, discretion 0.261
```

How to read it

Under **discretion**, the output gap and inflation simply follow the shock: once the shock is gone, policy has no reason to do anything. Under **commitment**, the output gap stays negative after the shock has faded and inflation turns slightly negative: the central bank promises to undo part of the price increase. Because

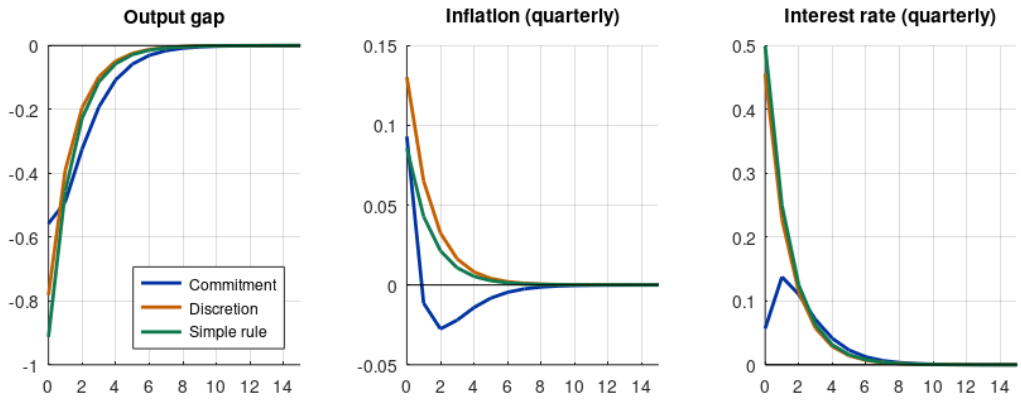


Figure 57.1. Responses to a cost-push shock under commitment, discretion and a simple rule.

firms expect this, they raise prices less today. The cumulative change in the price level is close to zero under commitment, which is why optimal commitment policy resembles **price-level targeting**.

57.5 Write the paper

How to write it: results

We compute optimal monetary policy under commitment and under discretion, and the Taylor rule that minimises the central bank's loss, in the calibrated New Keynesian model. Discretion raises the loss substantially relative to commitment, reflecting the stabilisation bias: under commitment, the central bank keeps policy tight after a cost-push shock has dissipated, which lowers inflation expectations and spreads the adjustment over time, so that the price level returns close to its initial path. A Taylor rule with optimised coefficients performs worse than discretion in this model, because a rule responding only to current inflation and the output gap cannot replicate the history dependence of optimal commitment policy.

Common mistakes

- Leaving the policy rule in the model when computing Ramsey or discretionary policy: the instrument must be free.
- Using an ad hoc loss function and calling the result "optimal". Derive the weights from welfare, or state that the loss is a mandate.
- Optimising a simple rule without bounds: coefficients run off to extreme values that no central bank would announce.
- Comparing policies by one variable's volatility rather than by the loss.

CHAPTER 58

Identification Before Estimation

Applies when. Estimate with data: check which parameters the data can pin down
Overview. Use this chapter before estimating any DSGE model. A parameter is identified only if changing it changes what the model implies for the observables. If two parameters always enter together, no amount of data can separate them, and a Bayesian posterior will simply return the prior. Dynare's identification command checks this at the prior, before a single draw is taken.
Data. The An–Schorfheide New Keynesian model of Chapter 59
Software. Dynare 6 · Octave / MATLAB

58.1 The question and why it comes first

Which parameters can the data actually tell us about? In a DSGE model the observables depend on the parameters through the solution of the model, and some parameters may have no separate effect: they enter only in a combination, or their effect is indistinguishable from another parameter's. Such parameters are **not identified**. Estimating them anyway gives posteriors that mirror the priors, and conclusions that are really assumptions.

Iskrev (2010) and Komunjer and Ng (2011) showed how to check identification from the model alone: a parameter is (locally) identified if the moments, or the spectrum, of the observables change when the parameter changes, in a way that no combination of other parameters can replicate. Dynare implements both checks in the `identification` command.

Two kinds of problems

- **Non-identification**: a parameter has no effect on the observables, or two parameters enter only as a product or ratio. The Jacobian of the moments with respect to the parameters is rank-deficient.
- **Weak identification**: the parameter is identified in principle, but its effect is small or nearly collinear with another's, so the data are only weakly informative. The Jacobian is close to rank-deficient.

58.2 Check the estimation model

```
// The An-Schorfheide model of the estimation guide, checked for identification at the prior mean.
var y pi R g z YGR INFL INT;
varexo e_R e_g e_z;
parameters tau kappa psi1 psi2 rho_R rho_g rho_z r_A;
tau = 2;  kappa = 0.2;  psi1 = 1.5;  psi2 = 0.5;  rho_R = 0.5;  rho_g = 0.8;  rho_z = 0.66;  r_A = 2;
model(linear);
  # beta = 1 / (1 + r_A/400);
  y  = y(+1) - (1/tau) * (R - pi(+1) - z(+1)) + g - g(+1);
  pi = beta * pi(+1) + kappa * (y - g);
  R  = rho_R * R(-1) + (1 - rho_R) * (psi1*pi + psi2*(y - g)) + e_R;
  g  = rho_g * g(-1) + e_g;  z = rho_z * z(-1) + e_z;
  YGR = y - y(-1) + z;  INFL = 4 * pi;  INT = 4 * R;
end;
shocks; var e_R; stderr 0.4; var e_g; stderr 1; var e_z; stderr 0.5; end;
steady;
estimated_params;
  tau, gamma_pdf, 2.00, 0.50;   kappa, gamma_pdf, 0.20, 0.10;
  psi1, gamma_pdf, 1.50, 0.25;  psi2, gamma_pdf, 0.50, 0.25;
```

```
  rho_R, beta_pdf, 0.50, 0.20;  rho_g, beta_pdf, 0.80, 0.10;  rho_z, beta_pdf, 0.66, 0.15;
  stderr e_R, inv_gamma_pdf, 0.4, 4;  stderr e_g, inv_gamma_pdf, 1.0, 4;  stderr e_z, inv_gamma_pdf, 0.5, 4;
end;
varobs YGR INFL INT;
identification(parameter_set = prior_mean, nograph);
```

```
dynare as_id
REDUCED-FORM:
    !!!WARNING!!!
    The rank of Tau (Jacobian of steady state and reduced-form solution matrices) is deficient!

    SE_e_R is collinear w.r.t. all other parameters!
    psi1 is collinear w.r.t. all other parameters!
    psi2 is collinear w.r.t. all other parameters!
    rho_R is collinear w.r.t. all other parameters!

MINIMAL SYSTEM (KOMUNJER AND NG, 2011):
    All parameters are identified in the Jacobian of steady state and minimal system (rank(Deltabar) is full with tol
↪  = robust).

SPECTRUM (QU AND TKACHENKO, 2012):
    !!!WARNING!!!
    The rank of Gbar (Jacobian of mean and spectrum) is deficient!

    psi1 is collinear w.r.t. all other parameters!
    psi2 is collinear w.r.t. all other parameters!

MOMENTS (ISKREV, 2010):
    !!!WARNING!!!
    The rank of J (Jacobian of first two moments) is deficient!…
```

How to read it

Dynare runs three tests, and they do not all agree:

- **Komunjer and Ng (2011)**, the test on the model's minimal state-space system, finds **all parameters identified**.
- **The reduced-form test (Iskrev 2010) and the spectrum test (Qu and Tkachenko 2012)** flag the **policy-rule block** (ψ_1, ψ_2, ρ_R and the standard deviation of the monetary shock) as collinear with the other parameters: their effects on the observables can be closely mimicked by combinations of other parameters.

This is not a quirk of the code. Cochrane (2011) argued that Taylor-rule coefficients are hard to identify in New Keynesian models precisely because, under determinacy, the equilibrium never shows the off-equilibrium responses those coefficients govern. In practice it means the data will be only weakly informative about the policy rule, and its posterior will lean on the prior: compare the prior and posterior of ψ_1 in Chapter 59 before interpreting it. The other parameters are cleanly identified.

58.3 A model that is not identified

```
// The same model, but the Phillips-curve slope is written in terms of two deep parameters,
// the Calvo probability theta and the inverse Frisch elasticity varphi, and both are estimated.
var y pi R g z YGR INFL INT;
varexo e_R e_g e_z;
parameters tau theta varphi psi1 psi2 rho_R rho_g rho_z r_A;
tau = 2;  theta = 0.75;  varphi = 1;  psi1 = 1.5;  psi2 = 0.5;  rho_R = 0.5;  rho_g = 0.8;  rho_z = 0.66;  r_A
  ↪  = 2;
model(linear);
  # beta = 1 / (1 + r_A/400);
  # kappa = (1 - theta) * (1 - beta*theta) / theta * (tau + varphi);
  y  = y(+1) - (1/tau) * (R - pi(+1) - z(+1)) + g - g(+1);
  pi = beta * pi(+1) + kappa * (y - g);
  R  = rho_R * R(-1) + (1 - rho_R) * (psi1*pi + psi2*(y - g)) + e_R;
  g  = rho_g * g(-1) + e_g;  z = rho_z * z(-1) + e_z;
  YGR = y - y(-1) + z;  INFL = 4 * pi;  INT = 4 * R;
end;
```

```
shocks; var e_R; stderr 0.4; var e_g; stderr 1; var e_z; stderr 0.5; end;
steady;
estimated_params;
  tau, gamma_pdf, 2.00, 0.50;   theta, beta_pdf, 0.75, 0.05;   varphi, gamma_pdf, 1.00, 0.25;
  psi1, gamma_pdf, 1.50, 0.25;  psi2, gamma_pdf, 0.50, 0.25;
  rho_R, beta_pdf, 0.50, 0.20;  rho_g, beta_pdf, 0.80, 0.10;  rho_z, beta_pdf, 0.66, 0.15;
  stderr e_R, inv_gamma_pdf, 0.4, 4;  stderr e_g, inv_gamma_pdf, 1.0, 4;  stderr e_z, inv_gamma_pdf, 0.5, 4;
end;
varobs YGR INFL INT;
identification(parameter_set = prior_mean, nograph);
clear functions;
dynare as_bad noclearall nograph nolog console
```

```
REDUCED-FORM:
    !!!WARNING!!!
    The rank of Tau (Jacobian of steady state and reduced-form solution matrices) is deficient!

    [varphi,theta] are PAIRWISE collinear!
    SE_e_R is collinear w.r.t. all other parameters!
    psi1 is collinear w.r.t. all other parameters!
    psi2 is collinear w.r.t. all other parameters!
    rho_R is collinear w.r.t. all other parameters!

MINIMAL SYSTEM (KOMUNJER AND NG, 2011):
    !!!WARNING!!!
    The rank of Deltabar (Jacobian of steady state and minimal system) is deficient!

    [varphi,theta] are PAIRWISE collinear!

SPECTRUM (QU AND TKACHENKO, 2012):
    !!!WARNING!!!
    The rank of Gbar (Jacobian of mean and spectrum) is deficient!

    [varphi,theta] are PAIRWISE collinear!
    SE_e_R is collinear w.r.t. all other parameters!
    psi1 is collinear w.r.t. all other parameters!
    psi2 is collinear w.r.t. all other parameters!…
```

How to read it

The Phillips curve depends on the Calvo parameter θ and the labour-supply parameter φ only through the slope κ. Many combinations of θ and φ give the same κ, and therefore exactly the same predictions for output, inflation and interest rates. Now **all three tests** fail, including Komunjer–Ng, and each names the same culprit: **[varphi, theta] are pairwise collinear**. Unlike the weak identification of the policy rule, this is exact non-identification.

This is a typical situation, not a contrived one. Small models usually identify the slope of the Phillips curve but not the structural parameters behind it. The remedies are:

- **estimate the reduced-form coefficient** (κ) and calibrate one of the structural parameters;
- **add data** that the other parameter affects separately (hours or wages identify φ);
- **use informative priors** from micro evidence, and say clearly that the posterior for that parameter reflects the prior.

58.4 Write the paper

How to write it: identification

Before estimation, we verify local identification of the estimated parameters at the prior mean using the rank conditions of Iskrev (2010) and Komunjer and Ng (2011), as implemented in Dynare. The Komunjer–Ng test indicates that all estimated parameters are locally identified, although the reduced-form and spectral tests point to weak identification of the policy-rule parameters, a known feature of New Keynesian models (Cochrane 2011); we therefore interpret their posteriors with reference to the priors. In an alternative parameterisation in which the

Calvo probability and the inverse Frisch elasticity are estimated separately, the two are not separately identified, because they enter the model only through the slope of the Phillips curve; we therefore estimate the slope directly.

Common mistakes

- Estimating first and checking identification only when a posterior looks like its prior.
- Taking a well-behaved posterior as evidence of identification: a tight prior produces a tight posterior even for an unidentified parameter.
- Checking identification at one parameter value only. Dynare can check it across draws from the prior (`prior_mc` option), which reveals regions where it fails.
- Estimating more structural parameters than the observables can inform, instead of estimating reduced-form coefficients.

CHAPTER 59

Bayesian Estimation of a New Keynesian Model

Applies when. Estimate with data: a small model, 3 observables

Overview. Use this chapter when the model's parameters should come from the data rather than from calibration. It estimates the An–Schorfheide (2007) New Keynesian model on U.S. data with Dynare: observables and measurement equations, priors, the posterior mode, the Metropolis–Hastings sampler and its convergence, posterior estimates, Bayesian impulse responses and a historical shock decomposition.

Data. U.S. output growth, inflation and the federal funds rate, 1966Q1–2007Q4

Software. Dynare 6 · Octave / MATLAB

59.1 The question and the model

How strongly did the Federal Reserve respond to inflation, and which shocks drove U.S. inflation? Calibration cannot answer this: it requires estimating the policy rule jointly with the rest of the economy. Bayesian estimation (Smets and Wouters 2003; An and Schorfheide 2007) combines the model's likelihood, computed with the Kalman filter, with prior information about the parameters.

The model is the New Keynesian model of An and Schorfheide (2007), with a technology growth shock z_t, a demand (government spending) shock g_t and a monetary policy shock ε_t^R:

$$
\begin{aligned}
y_t &= E_t y_{t+1} + g_t - E_t g_{t+1} - \tfrac{1}{\tau}\left(R_t - E_t \pi_{t+1} - E_t z_{t+1}\right) \\
\pi_t &= \beta E_t \pi_{t+1} + \kappa (y_t - g_t) \\
R_t &= \rho_R R_{t-1} + (1-\rho_R)\left[\psi_1 \pi_t + \psi_2 (y_t - g_t)\right] + \varepsilon_t^R
\end{aligned}
$$

and **measurement equations** that link the model to the data:

$$
\mathrm{YGR}_t = y_t - y_{t-1} + z_t, \qquad \mathrm{INFL}_t = 4\pi_t, \qquad \mathrm{INT}_t = 4R_t
$$

Data and measurement are part of the model

- **Three observables and three shocks.** A linear DSGE model with fewer shocks than observables is *stochastically singular*: it implies an exact linear relation among the observables that the data reject. Never use more observables than shocks (or add measurement errors).
- **Demeaned data.** Each observable has its sample mean removed, so the model is estimated in deviations and the constants (trend growth, the inflation target, the real rate) are not estimated. Estimating them requires that the model's steady state matches the data's means, which with these data would force a real interest rate near zero. Demeaning is a common, transparent choice; state it in the paper.
- **Units.** Output growth is in percent per quarter; inflation and the interest rate are annualised percentages, hence the factor 4.

How to write it: data

We estimate the model on three quarterly U.S. series for 1966Q1–2007Q4: the growth rate of real GDP, inflation measured by the GDP deflator (annualised), and the effective federal funds rate. The sample ends before the zero-lower-bound period. All series are demeaned prior to estimation.

59.2 Look at the data

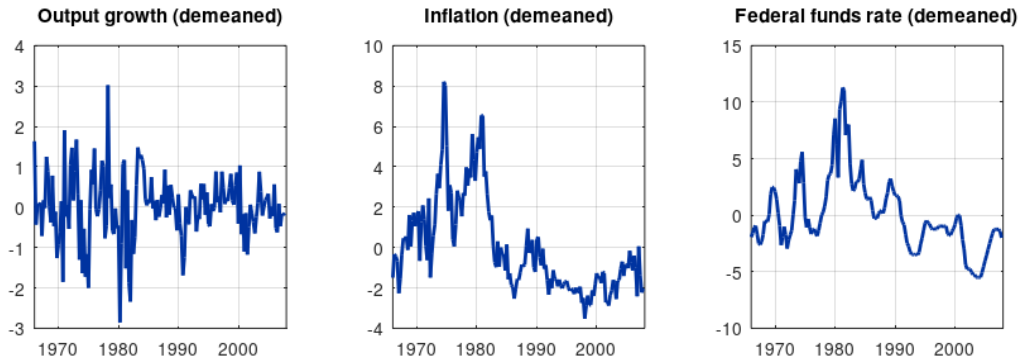


Figure 59.1. Observables used in the estimation, demeaned: output growth, inflation and the federal funds rate, 1966Q1–2007Q4.

```
d = csvread('us_small_demeaned.csv', 1, 0);
t = 1966:0.25:2007.75;
f = figure('position', [0 0 900 260]);
tt = {'Output growth (demeaned)', 'Inflation (demeaned)', 'Federal funds rate (demeaned)'};
for j = 1:3, subplot(1, 3, j); plot(t, d(:, j)); title(tt{j}); grid on; xlim([1966 2008]); end
print(f, '-dpng', '-r110', 'data.png');
fprintf('%d quarters; standard deviations: output growth %.2f, inflation %.2f, interest rate %.2f\n', size(d,
  ↪ 1), std(d));
```

```
168 quarters; standard deviations: output growth 0.81, inflation 2.36, interest rate 3.26
```

59.3 Priors and the model file

Parameter	Meaning	Prior	Mean	Std. dev.
τ	inverse intertemporal elasticity	Gamma	2.0	0.5
κ	slope of the Phillips curve	Gamma	0.2	0.1
ψ_1	response to inflation	Gamma	1.5	0.25
ψ_2	response to the output gap	Gamma	0.5	0.25
ρ_R, ρ_g, ρ_z	persistence	Beta	0.5, 0.8, 0.66	0.2, 0.1, 0.15
$\sigma_R, \sigma_g, \sigma_z$	shock standard deviations	Inverse gamma	0.4, 1.0, 0.5	4 (degrees of freedom)

The priors follow An and Schorfheide (2007): their means are standard values from earlier studies, and their spreads allow the data to move the estimates substantially. Gamma priors keep parameters positive, Beta priors keep persistence between 0 and 1.

```
// An-Schorfheide (2007) New Keynesian model, estimated on demeaned U.S. data.
var y pi R g z YGR INFL INT;
varexo e_R e_g e_z;
parameters tau kappa psi1 psi2 rho_R rho_g rho_z r_A;

tau = 2;  kappa = 0.2;  psi1 = 1.5;  psi2 = 0.5;  rho_R = 0.5;  rho_g = 0.8;  rho_z = 0.66;
r_A = 2;                                                  // annual real rate, fixes beta (not estimated)

model(linear);
  # beta = 1 / (1 + r_A/400);
  y  = y(+1) - (1/tau) * (R - pi(+1) - z(+1)) + g - g(+1);
  pi = beta * pi(+1) + kappa * (y - g);
  R  = rho_R * R(-1) + (1 - rho_R) * (psi1*pi + psi2*(y - g)) + e_R;
  g  = rho_g * g(-1) + e_g;
  z  = rho_z * z(-1) + e_z;
  YGR  = y - y(-1) + z;                                   // measurement equations
```

```
  INFL = 4 * pi;
  INT  = 4 * R;
end;

shocks;
  var e_R; stderr 0.4;  var e_g; stderr 1;  var e_z; stderr 0.5;
end;

steady;
check;

estimated_params;
  tau,   gamma_pdf, 2.00, 0.50;
  kappa, gamma_pdf, 0.20, 0.10;
  psi1,  gamma_pdf, 1.50, 0.25;
  psi2,  gamma_pdf, 0.50, 0.25;
  rho_R, beta_pdf,  0.50, 0.20;
  rho_g, beta_pdf,  0.80, 0.10;
  rho_z, beta_pdf,  0.66, 0.15;
  stderr e_R, inv_gamma_pdf, 0.4, 4;
  stderr e_g, inv_gamma_pdf, 1.0, 4;
  stderr e_z, inv_gamma_pdf, 0.5, 4;
end;

varobs YGR INFL INT;

estimation(datafile = 'us_small_demeaned.csv', mode_compute = 5, mh_replic = 10000, mh_nblocks = 2,
           mh_jscale = 0.6, mh_drop = 0.5, bayesian_irf, irf = 16, nograph) y pi R;
shock_decomposition(nograph) INFL;
```

```
dynare as
ESTIMATION RESULTS

Log data density (Modified Harmonic Mean) is -768.017715.
parameters
        prior mean   post. mean        90% HPD interval    prior       pstdev

tau          2.000       4.4320      3.4072      5.4476     gamm       0.5000
kappa        0.200       0.1549      0.0876      0.2252     gamm       0.1000
psi1         1.500       1.1645      1.0191      1.2993     gamm       0.2500
psi2         0.500       0.3580      0.0921      0.6109     gamm       0.2500
rho_R        0.500       0.7687      0.7251      0.8151     beta       0.2000
rho_g        0.800       0.9780      0.9644      0.9925     beta       0.1000
rho_z        0.660       0.9713      0.9553      0.9882     beta       0.1500

standard deviation of shocks
        prior mean   post. mean        90% HPD interval    prior       pstdev

e_R          0.400       0.2803      0.2476      0.3081     invg       4.0000
e_g          1.000       0.9685      0.8751      1.0643     invg       4.0000
e_z          0.500       0.0919      0.0728      0.1097     invg       4.0000
    gaminv at line 101 column 7
    prior_bounds at line 57 column 26
    dynare_estimation_init at line 193 column 16
    evaluate_smoother at line 66 column 76
    shock_decomposition at line 87 column 74
    driver at line 430 column 5
    dynare at line 318 column 5

    gaminv at line 101 column 7
    prior_bounds at line 57 column 26…
```

How to read it

- **estimated_params** lists each estimated parameter with its prior: distribution, mean, standard deviation.

- **varobs** names the observables; their names must match the column headers of the data file.
- **estimation** first finds the **posterior mode** (`mode_compute = 5`, Dynare's Newton-type optimiser), then runs two independent **Metropolis–Hastings** chains of 10,000 draws each, discards the first half of each chain as burn-in (`mh_drop`), and reports posterior means and 90% highest-posterior-density intervals. `mh_jscale` scales the proposal distribution; it is tuned so that about a quarter to a third of proposals are accepted.
- **Chains for a paper** should be much longer (at least 2 × 100,000); the chapter uses short chains so that every result can be reproduced in minutes.

The output reports, for each parameter: the prior mean, the posterior mean, the 90% HPD interval, the prior distribution and the prior standard deviation.

59.4 Did the sampler converge?

```
fn = M_.fname;
files = dir(fullfile(fn, 'metropolis', [fn '_mh*_blck*.mat']));
nb = options_.mh_nblck; draws = cell(1, nb);
for b = 1:nb
  x = [];
  ff = dir(fullfile(fn, 'metropolis', sprintf('%s_mh*_blck%d.mat', fn, b)));
  [~, o] = sort(cellfun(@(s) sscanf(regexprep(s, '.*_mh(\d+)_.*', '$1'), '%d'), {ff.name}));
  for k = o, S = load(fullfile(fn, 'metropolis', ff(k).name)); x = [x; S.x2]; end
  draws{b} = x(floor(end/2)+1:end, :);                    % drop the first half as burn-in
end
pn = cellstr(bayestopt_.name);
n = size(draws{1}, 1);
m = cellfun(@(x) mean(x), draws, 'UniformOutput', false); m = vertcat(m{:});
v = cellfun(@(x) var(x), draws, 'UniformOutput', false); v = vertcat(v{:});
rec = load_last_mh_history_file(fullfile(fn, 'metropolis'), fn); acc_ratio = rec.AcceptanceRatio;
W = mean(v); B = n * var(m); Rhat = sqrt(((n-1)/n * W + B/n) ./ W);
fprintf('Acceptance ratios by chain: %s\n', sprintf('%.3f  ', acc_ratio));
fprintf('%-8s %8s\n', 'param', 'R-hat');
for j = 1:numel(pn), fprintf('%-8s %8.3f\n', pn{j}, Rhat(j)); end
f = figure('position', [0 0 900 300]);
show = {'psi1', 'kappa', 'rho_R'};
for j = 1:3
  k = find(strcmp(pn, show{j}));
  subplot(1, 3, j); plot(draws{1}(:, k), 'linewidth', 0.6); hold on; plot(draws{2}(:, k), 'linewidth', 0.6);
  title(['Trace of ' show{j} ' (two chains)']); grid on;
end
print(f, '-dpng', '-r110', 'trace.png');
```

```
Acceptance ratios by chain: 0.321  0.322
param       R-hat
e_R         1.002
e_g         1.000
e_z         1.003
tau         1.001
kappa       1.004
psi1        1.000
psi2        1.000
rho_R       1.006
rho_g       1.008
rho_z       1.000
```

How to read it

- **Acceptance ratio.** Between about 0.2 and 0.4 is the target. Too high means the sampler takes tiny steps and explores slowly; too low means it rarely moves.
- **R-hat** (Gelman and Rubin 1992) compares the variance between the two chains with the variance within them. Values close to 1 (below about 1.1) indicate that the chains have converged to the same distribution. Dynare reports the related Brooks–Gelman diagnostics graphically.

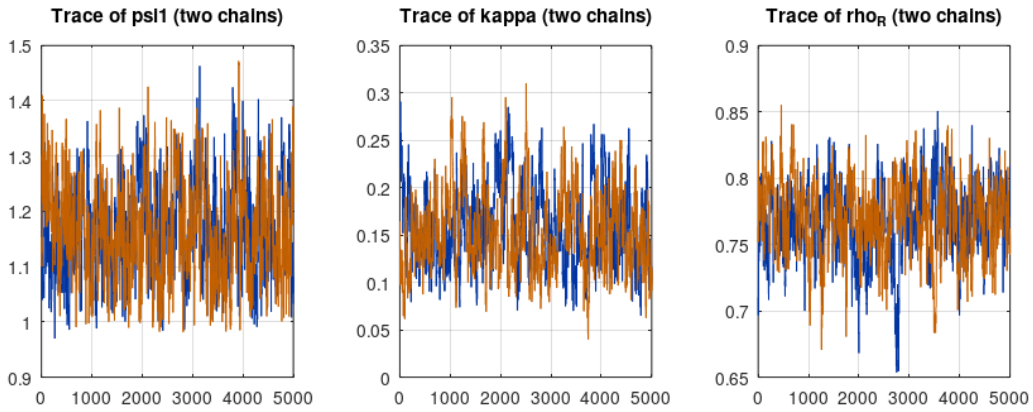


Figure 59.2. Trace plots of two Metropolis–Hastings chains for three parameters.

- **Trace plots.** The two chains should overlap and move freely around the same region, with no trends.

59.5 Posterior estimates against the priors

```
f = figure('position', [0 0 900 420]);
alld = [draws{1}; draws{2}];
for j = 1:numel(pn)
  subplot(3, 4, j);
  [ff, xi] = hist(alld(:, j), 40); bar(xi, ff / trapz(xi, ff), 1, 'facecolor', [0.75 0.82 0.93], 'edgecolor',
  ↪ 'none'); hold on;
  [xp, fp] = draw_prior_density(j, bayestopt_);
  plot(xp, fp, 'k--', 'linewidth', 1.2); title(pn{j}); grid on;
end
print(f, '-dpng', '-r110', 'posterior.png');
fprintf('%-8s %10s %10s %10s %10s\n', 'param', 'prior mean', 'post. mean', '5%', '95%');
for j = 1:numel(pn)
  q = sort(alld(:, j)); fprintf('%-8s %10.3f %10.3f %10.3f %10.3f\n', pn{j}, bayestopt_.p1(j), mean(alld(:, j))
  ↪ , q(round(0.05*end)), q(round(0.95*end)));
end
```

```
param    prior mean post. mean         5%        95%
e_R           0.400      0.280      0.252      0.314
e_g           1.000      0.968      0.876      1.068
e_z           0.500      0.092      0.075      0.113
tau           2.000      4.432      3.459      5.537
kappa         0.200      0.155      0.094      0.234
psi1          1.500      1.164      1.032      1.317
psi2          0.500      0.358      0.126      0.690
rho_R         0.500      0.769      0.720      0.811
rho_g         0.800      0.978      0.963      0.992
rho_z         0.660      0.971      0.954      0.988
```

How to read it

Bars are the posterior, dashed lines the prior. Where the two differ, the data are informative.

- **The policy rule.** The response to inflation ψ_1 is estimated just above 1, lower than its prior mean: over 1966–2007, which includes the high-inflation 1970s, the Federal Reserve on average barely satisfied the Taylor principle. Interest-rate smoothing ρ_R is high.
- **The Phillips curve slope** κ is estimated at about 0.15: inflation responds weakly to the output gap.
- **Shock persistence.** The demand and technology shocks are very persistent (ρ_g, ρ_z near 0.97), a common finding in small models: persistent shocks absorb dynamics the model's frictions cannot generate.
- **Where the posterior looks like the prior**, the data say little about that parameter. Chapter 58 shows how to check this before estimating.

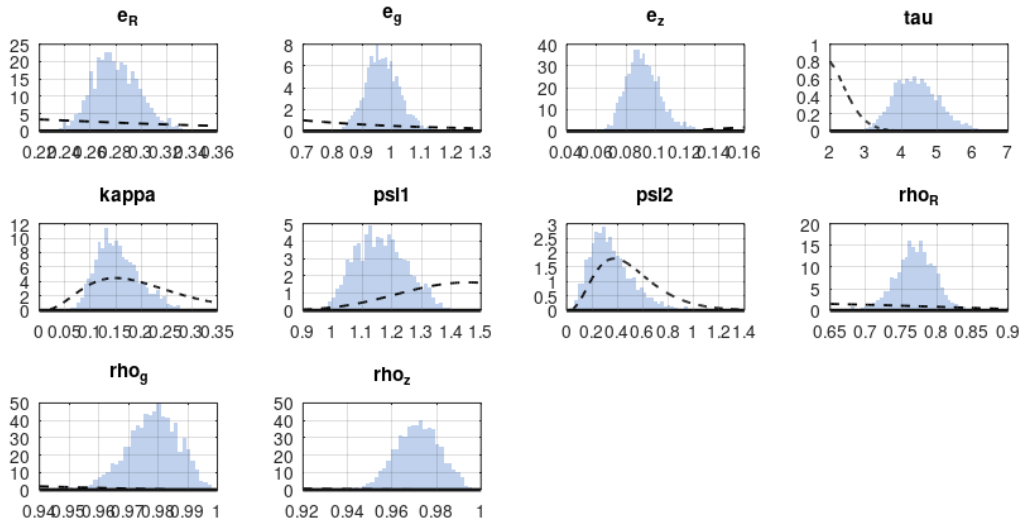


Figure 59.3. Prior (dashed line) and posterior (histogram) distributions of the estimated parameters.

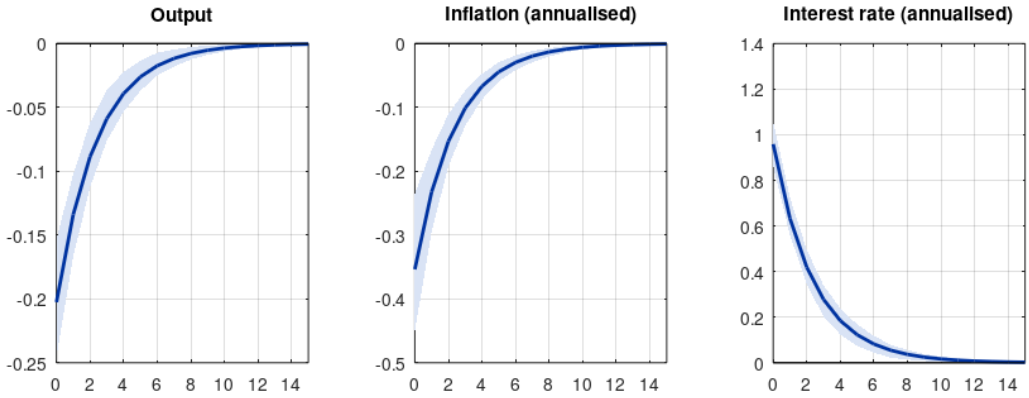


Figure 59.4. Posterior impulse responses of output, inflation and the interest rate to a monetary-policy shock. The shaded bands show posterior uncertainty.

59.6 Bayesian impulse responses

```
f = figure('position', [0 0 900 290]);
vv = {'y', 'pi', 'R'}; tt = {'Output', 'Inflation (annualised)', 'Interest rate (annualised)'}; sc = [1 4 4];
for j = 1:3
  r = oo_.PosteriorIRF.dsge;
  m = sc(j) * r.Mean.([vv{j} '_e_R']); lo = sc(j) * r.HPDinf.([vv{j} '_e_R']); hi = sc(j) * r.HPDsup.([vv{j} '
  ↪ _e_R']);
  subplot(1, 3, j); fill([0:15 15:-1:0], [lo' fliplr(hi')], [0.85 0.89 0.96], 'edgecolor', 'none'); hold on;
  plot(0:15, m); plot([0 15], [0 0], 'k-', 'linewidth', 0.5); title(tt{j}); grid on; xlim([0 15]);
end
print(f, '-dpng', '-r110', 'irf.png');
fprintf('Monetary shock, impact (posterior mean): output %.2f%%, inflation %.2f pp, interest rate %.2f pp\n',
  ↪ ...
  oo_.PosteriorIRF.dsge.Mean.y_e_R(1), 4*oo_.PosteriorIRF.dsge.Mean.pi_e_R(1), 4*oo_.PosteriorIRF.dsge.Mean.
  ↪ R_e_R(1));
```

```
Monetary shock, impact (posterior mean): output -0.20%, inflation -0.35 pp, interest rate 0.96 pp
```

How to read it

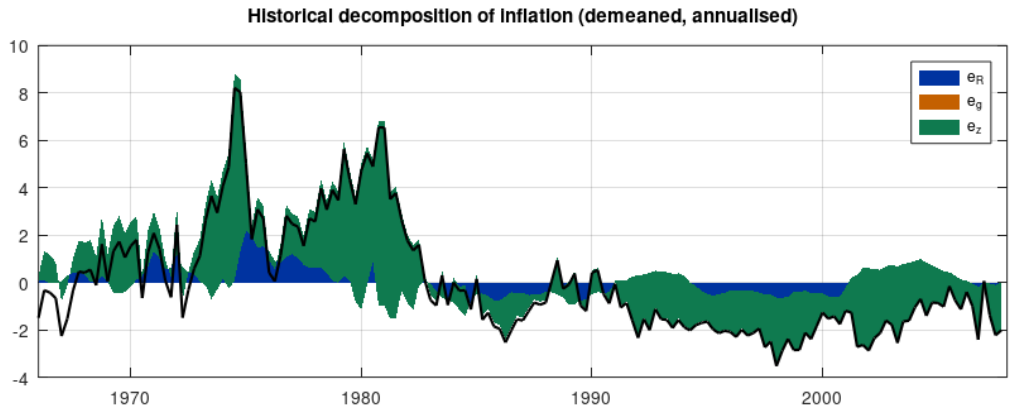


Figure 59.5. Historical decomposition of inflation (demeaned, annualised) into the contributions of the three shocks e_R, e_g and e_z. The black line is the data.

Each line is the posterior mean response to a one-standard-deviation monetary policy shock; the shaded band is the 90% HPD interval, which reflects parameter uncertainty. Unlike a VAR, the DSGE model identifies the shock through its structure: every response follows from the estimated equations.

59.7 Which shocks drove inflation?

```
sd = oo_.shock_decomposition;                                % variables x (shocks + initial values + smoothed) x
  ↪ time
iv = find(strcmp(cellstr(M_.endo_names), 'INFL'));
contrib = squeeze(sd(iv, 1:3, :))';
f = figure('position', [0 0 900 300]);
area(t, contrib, 'edgecolor', 'none'); hold on; plot(t, squeeze(sd(iv, end, :)), 'k', 'linewidth', 1.4);
legend(cellstr(M_.exo_names), 'location', 'northeast'); title('Historical decomposition of inflation (demeaned,
  ↪ annualised)'); grid on; xlim([1966 2008]);
print(f, '-dpng', '-r110', 'decomp.png');
share = sum(contrib.^2) / sum(sum(contrib.^2));
fprintf('Share of the squared contributions to inflation: %s\n', strjoin(cellfun(@(s, v) sprintf('%s %.0f%%',
  ↪ s, 100*v), cellstr(M_.exo_names)', num2cell(share), 'UniformOutput', false), ', '));
```

```
Share of the squared contributions to inflation: e_R 6%, e_g 0%, e_z 94%
```

How to read it

The black line is demeaned inflation; the coloured areas show how much of it each smoothed shock accounts for in each quarter (plus the effect of initial conditions, not shown). In this small model the persistent technology growth shock accounts for almost all of it (94% of the squared contributions), including the Great Inflation of the 1970s and the disinflation of the 1980s; monetary policy shocks account for the rest. The demand shock does not move inflation at all: in this model it shifts output and its natural level together. That is a property of a three-shock model with no cost-push shock: a richer model would split inflation between more shocks, so read the decomposition as model-dependent.

59.8 Write the paper

How to write it: results

Table 2 reports the prior and posterior distributions. The posterior mean of the response of the policy rate to inflation is about 1.2, with a 90 percent interval just above one, indicating that over 1966–2007 the Federal Reserve on average satisfied the Taylor principle only narrowly; interest rates were adjusted gradually (posterior mean of the smoothing coefficient about 0.77). The slope of the Phillips curve is estimated at about 0.15. Two Metropolis–Hastings chains were run; the Gelman–Rubin statistics are close to one for all parameters. A contractionary monetary policy shock lowers output and inflation (Figure 3).

Common mistakes

- More observables than shocks: stochastic singularity.
- Data and model in different units (annualised vs quarterly, percent vs fractions, levels vs growth rates).
- Reporting estimates from chains that have not converged, or from very short chains.
- Running the sampler without first checking the mode (`mode_check`) and identification.
- Priors so tight that the posterior equals the prior, then presenting the result as an estimate. Always plot priors against posteriors.

CHAPTER 60

Medium-Scale New Keynesian Model

Applies when. Estimate with data: a richer model, 5 observables

Overview. Use this chapter when the question involves investment, consumption dynamics or several shocks at once, so a three-equation model is too small. It estimates a medium-scale model in the spirit of Smets and Wouters (2007), with capital, habit formation, investment adjustment costs and price indexation, on five U.S. observables, and reads the results the way a referee will.

Data. U.S. output, consumption and investment growth, inflation and the federal funds rate, 1966Q1–2007Q4

Software. Dynare 6 · Octave / MATLAB

60.1 The question and the model

Which frictions and which shocks are needed to fit the joint behaviour of output, consumption, investment, inflation and interest rates? Christiano, Eichenbaum and Evans (2005) and Smets and Wouters (2007) showed that a New Keynesian model with a handful of real and nominal frictions can fit U.S. data about as well as a VAR. Their model is the basis of most policy models in central banks.

This chapter estimates a compact version: capital accumulation with **investment adjustment costs**, consumption **habits**, Calvo prices with **indexation**, a Taylor rule, and six shocks (technology, preference, investment-specific, government spending, price mark-up and monetary policy). Wages are flexible, which keeps the model small but, as the results show, pushes the price-rigidity parameter to absorb what wage rigidity would explain.

Equation	Log-linear form (deviations from the steady state)
Consumption Euler, with habits h	$c_t = \frac{h}{1+h}c_{t-1} + \frac{1}{1+h}E_t c_{t+1} - \frac{1-h}{1+h}(R_t - E_t\pi_{t+1}) + b_t$
Investment, adjustment cost φ_i	$i_t = \frac{1}{1+\beta}i_{t-1} + \frac{\beta}{1+\beta}E_t i_{t+1} + \frac{1}{(1+\beta)\varphi_i}q_t + \varepsilon_t^i$
Value of capital	$q_t = \beta(1-\delta)E_t q_{t+1} + (1-\beta(1-\delta))E_t r^k_{t+1} - (R_t - E_t\pi_{t+1})$
Phillips curve, indexation ι_p	$\pi_t = \frac{\iota_p}{1+\beta\iota_p}\pi_{t-1} + \frac{\beta}{1+\beta\iota_p}E_t\pi_{t+1} + \kappa\, mc_t + \varepsilon_t^p$
Labour supply (flexible wages)	$w_t = \sigma_l l_t + \frac{1}{1-h}(c_t - h c_{t-1})$

Five observables, six shocks

The model has more shocks than observables, so it is not stochastically singular. Each observable has a natural shock: preference shocks for consumption, investment-specific shocks for investment, government spending for output, mark-up shocks for inflation and monetary shocks for the interest rate. With fewer shocks, the model would be forced to explain everything with too few sources of variation.

60.2 Write and estimate the model

```
// Medium-scale New Keynesian model (Smets-Wouters type, flexible wages), estimated on demeaned U.S. data.
var y c i k q rk l w mc pi R a b ei g ep YGR CGR IGR INFL INT;
varexo e_a e_b e_i e_g e_p e_R;
parameters beta delta alpha cy iy gy h phi_i sigma_l iota_p xi_p psi_pi psi_y rho_R rho_a rho_b rho_i rho_g
  ↪ rho_p;

beta = 0.995;  delta = 0.025;  alpha = 0.3;  gy = 0.18;
iy = delta * alpha / (1/beta - 1 + delta);  cy = 1 - iy - gy;      // steady-state expenditure shares
```

```
h = 0.7;  phi_i = 5;  sigma_l = 2;  iota_p = 0.3;  xi_p = 0.75;
psi_pi = 1.5;  psi_y = 0.1;  rho_R = 0.8;
rho_a = 0.95;  rho_b = 0.7;  rho_i = 0.7;  rho_g = 0.9;  rho_p = 0.5;

model(linear);
  # kappa = (1 - xi_p) * (1 - beta*xi_p) / xi_p / (1 + beta*iota_p);
  c  = h/(1+h)*c(-1) + 1/(1+h)*c(+1) - (1-h)/(1+h)*(R - pi(+1)) + b;      // consumption Euler with habits
  i  = 1/(1+beta)*i(-1) + beta/(1+beta)*i(+1) + 1/((1+beta)*phi_i)*q + ei; // investment
  q  = beta*(1-delta)*q(+1) + (1 - beta*(1-delta))*rk(+1) - (R - pi(+1)); // Tobin's q
  k  = (1-delta)*k(-1) + delta*i;                                        // capital
  y  = a + alpha*k(-1) + (1-alpha)*l;                                    // production
  rk = w + l - k(-1);                                                    // cost minimisation
  mc = alpha*rk + (1-alpha)*w - a;                                       // real marginal cost
  w  = sigma_l*l + (c - h*c(-1))/(1-h);                                  // labour supply
  pi = iota_p/(1+beta*iota_p)*pi(-1) + beta/(1+beta*iota_p)*pi(+1) + kappa*mc + ep;  // Phillips curve
  y  = cy*c + iy*i + gy*g;                                               // resource constraint
  R  = rho_R*R(-1) + (1-rho_R)*(psi_pi*pi + psi_y*y) + e_R;               // Taylor rule
  a  = rho_a*a(-1) + e_a;   b = rho_b*b(-1) + e_b;   ei = rho_i*ei(-1) + e_i;
  g  = rho_g*g(-1) + e_g;   ep = rho_p*ep(-1) + e_p;
  YGR = y - y(-1);  CGR = c - c(-1);  IGR = i - i(-1);  INFL = 4*pi;  INT = 4*R;   // measurement
end;

shocks;
  var e_a; stderr 0.5;  var e_b; stderr 0.3;  var e_i; stderr 0.5;
  var e_g; stderr 1;    var e_p; stderr 0.2;  var e_R; stderr 0.2;
end;

steady;
check;

estimated_params;
  h,      beta_pdf,  0.70, 0.10;   phi_i,  gamma_pdf, 4.00, 1.50;   sigma_l, gamma_pdf, 2.00, 0.75;
  iota_p, beta_pdf,  0.50, 0.15;   xi_p,   beta_pdf,  0.66, 0.10;
  psi_pi, gamma_pdf, 1.50, 0.25;   psi_y,  gamma_pdf, 0.125, 0.05;  rho_R,  beta_pdf,  0.75, 0.10;
  rho_a,  beta_pdf,  0.50, 0.20;   rho_b,  beta_pdf,  0.50, 0.20;   rho_i,  beta_pdf,  0.50, 0.20;
  rho_g,  beta_pdf,  0.50, 0.20;   rho_p,  beta_pdf,  0.50, 0.20;
  stderr e_a, inv_gamma_pdf, 0.5, 2;  stderr e_b, inv_gamma_pdf, 0.3, 2;  stderr e_i, inv_gamma_pdf, 0.5, 2;
  stderr e_g, inv_gamma_pdf, 0.5, 2;  stderr e_p, inv_gamma_pdf, 0.2, 2;  stderr e_R, inv_gamma_pdf, 0.2, 2;
end;

varobs YGR CGR IGR INFL INT;

estimation(datafile = 'us_medium_demeaned.csv', mode_compute = 5, mh_replic = 4000, mh_nblocks = 2,
           mh_jscale = 0.25, mh_drop = 0.5, moments_varendo, nograph) y c i pi R;
```

```
dynare med
ESTIMATION RESULTS

Log data density (Modified Harmonic Mean) is -1277.873366.
parameters
          prior mean   post. mean        90% HPD interval    prior       pstdev

h              0.700       0.9142      0.8853      0.9417     beta       0.1000
phi_i          4.000       5.5200      3.3575      7.5202     gamm       1.5000
sigma_l        2.000       1.2920      0.5052      2.0709     gamm       0.7500
iota_p         0.500       0.5866      0.3342      0.8011     beta       0.1500
xi_p           0.660       0.9581      0.9462      0.9700     beta       0.1000
psi_pi         1.500       1.2709      1.0288      1.5284     gamm       0.2500
psi_y          0.125       0.2359      0.1594      0.3110     gamm       0.0500
rho_R          0.750       0.8464      0.8178      0.8789     beta       0.1000
rho_a          0.500       0.9556      0.9250      0.9885     beta       0.2000
rho_b          0.500       0.2831      0.1589      0.3896     beta       0.2000
rho_i          0.500       0.2822      0.1591      0.4121     beta       0.2000
rho_g          0.500       0.9648      0.9442      0.9892     beta       0.2000
rho_p          0.500       0.1426      0.0263      0.2901     beta       0.2000
```

```
standard deviation of shocks
          prior mean   post. mean        90% HPD interval    prior       pstdev

e_a            0.500       0.8062      0.2006      1.3690     invg       2.0000
e_b            0.300       0.2453      0.2060      0.2840     invg       2.0000
e_i            0.500       1.7419      1.4937      1.9895     invg       2.0000
e_g            0.500       2.7673      2.5208      3.0121     invg       2.0000
e_p            0.200       0.1290      0.1098      0.1453     invg       2.0000
e_R            0.200       0.2348      0.2116      0.2514     invg       2.0000
Estimation::compute_moments_varendo: I'm computing endogenous moments (this may take a while)...

Posterior mean variance decomposition (in percent)
         e_a     e_b     e_i     e_g     e_p     e_R
y      10.38    7.38   56.27    9.15    4.42   12.41…
```

How to read it

Two short chains of 4,000 draws are run so that the page can be rebuilt quickly; for a paper, run at least 2 × 100,000 and check convergence as in Chapter 59.

- **Frictions.** Habit persistence is very high (around 0.9): consumption responds sluggishly, as in the data. The investment adjustment cost is clearly positive, which delivers the hump-shaped response of investment.
- **Price rigidity.** The Calvo parameter is estimated near 0.96, implying that prices stay fixed for about 25 quarters (six years) on average, far longer than micro evidence suggests. This is the classic symptom of a missing friction: without sticky wages, marginal cost moves too much, and the only way to fit smooth inflation is to make prices extremely sticky. Smets and Wouters add wage rigidity for exactly this reason. Report the number and say what it means.
- **Policy.** The inflation response is about 1.3 with strong smoothing (about 0.85), close to the small model.
- **Persistence.** Technology and government spending shocks are very persistent; the investment and preference shocks much less so.

60.3 Which shocks drive which variables?

```
sh = cellstr(M_.exo_names)';
vn = {'y', 'c', 'i', 'pi', 'R'};
VD = oo_.PosteriorTheoreticalMoments.dsge.VarianceDecomposition.Mean;   % posterior mean over the draws
vd = zeros(5, numel(sh));
for j = 1:5, for k = 1:numel(sh), vd(j, k) = VD.(vn{j}).(sh{k}); end, end
if max(vd(:)) <= 1, vd = 100 * vd; end                  % report in percent
fprintf('%-6s', ''); fprintf('%8s', sh{:}); fprintf('\n');
for j = 1:5, fprintf('%-6s', vn{j}); fprintf('%8.1f', vd(j, :)); fprintf('\n'); end
```

```
           e_a     e_b     e_i     e_g     e_p     e_R
y         10.4     7.4    56.3     9.1     4.4    12.4
c         17.2    31.0     5.1    42.1     0.9     3.7
i          3.7     6.0    71.7    10.2     2.8     5.7
pi        60.9     0.9     2.5     4.5    30.7     0.5
R         48.3     3.5     8.8     4.7     6.6    28.0
```

How to read it

The table gives, for each variable, the percentage of its unconditional variance due to each shock, averaged over the posterior draws (computed by the `moments_varendo` option). Investment is dominated by the investment-specific shock (about 70%), and through investment so is output (more than half); consumption leans on the government-spending and preference shocks; inflation is driven mainly by technology shocks, with mark-up shocks second (about 30%). Each shock has a clear role, but the investment shock explaining most of output is a familiar warning sign. Whether that is economics or a sign that the shocks are absorbing misspecification is the central debate about medium-scale models (Chari, Kehoe and McGrattan 2009). A paper should report this table and discuss it.

60.4 Write the paper

How to write it: results

We estimate a medium-scale New Keynesian model with habit formation, investment adjustment costs and partially indexed Calvo prices on five U.S. observables. The posterior points to strong habit persistence and substantial investment adjustment costs, in line with Smets and Wouters (2007). The estimated degree of price stickiness is implausibly high relative to micro evidence, which we attribute to the absence of nominal wage rigidity in our specification. Investment-specific shocks account for most of the variance of investment and output, and technology shocks for most of the variance of inflation, with mark-up shocks second.

Common mistakes

- Adding shocks until the model fits, then interpreting each shock as structural.
- Ignoring implausible estimates (a Calvo parameter of 0.96): they are diagnostics of misspecification.
- Short chains in a model with twenty parameters: check convergence for every parameter.
- Comparing models by marginal data density without the same data and the same treatment of the means.

CHAPTER 61

The Zero Lower Bound with OccBin

Applies when. Nonlinearity: an occasionally binding constraint
Overview. Use this chapter when a constraint binds only sometimes: the zero lower bound on interest rates, a borrowing limit, a capacity or collateral constraint. A linear solution ignores the constraint and can let the interest rate fall to minus five percent. Dynare's OccBin solver (Guerrieri and Iacoviello 2015) switches between the regime where the constraint is slack and the regime where it binds, and shows how much worse a recession is when policy cannot respond.
Data. Calibrated New Keynesian model, deep demand shock
Software. Dynare 6 · Octave / MATLAB

61.1 The question and the model

How much deeper is a recession when the central bank cannot cut the interest rate below zero? Between 2008 and 2015, and again in 2020, policy rates in many economies sat at or near zero, and a central question of macroeconomics became how the zero lower bound (ZLB) changes the effects of shocks.

The model is the three-equation New Keynesian model of Chapter 55, with one change: the interest rate follows the Taylor rule only when the rule prescribes a rate above the bound.

$$i_t = \max\{\, \underline{i},\quad \phi_\pi \pi_t + \phi_x x_t \,\}$$

Variables are deviations from the steady state, so the bound is $\underline{i} = -i^*$, minus the steady-state nominal rate. With a 2% annual real rate and a 2% inflation target, $i^* = 4\%$ a year, 1% a quarter.

How OccBin works

OccBin treats the model as two linear models, one for each regime: the Taylor rule (constraint slack) and $i_t = \underline{i}$ (constraint binding). For a given path of shocks it guesses in which periods the constraint binds, solves the model backwards through time with the right regime in each period, checks whether the guess is consistent (the constraint binds exactly when the shadow rate is below the bound), and iterates until it is. The solution is nonlinear globally but piecewise linear, and expectations take into account that the economy will leave the ZLB later.

61.2 Write the model in Dynare

```
// New Keynesian model with the zero lower bound, solved with OccBin. Quarterly percent deviations.
@#ifndef ILB
  @#define ILB = -1.0
@#endif
var x pi i rn inot;
varexo e_rn;
parameters beta sigma kappa phi_pi phi_x rho_rn ilb;

beta = 0.995;  sigma = 1;  kappa = 0.1717;  phi_pi = 1.5;  phi_x = 0.125;  rho_rn = 0.8;
ilb = @{ILB};                                  // the bound: minus the steady-state nominal rate (1% a
  ↪ quarter)

model;
  x = x(+1) - (1/sigma) * (i - pi(+1) - rn);    // IS curve
  pi = beta * pi(+1) + kappa * x;               // Phillips curve
```

```
  inot = phi_pi * pi + phi_x * x;              // the rate the Taylor rule prescribes (the shadow rate)
  [name = 'policy', relax = 'zlb']
  i = inot;                                    // regime 1: the rule applies
  [name = 'policy', bind = 'zlb']
  i = ilb;                                     // regime 2: the rate is stuck at the bound
  rn = rho_rn * rn(-1) + e_rn;                 // natural-rate (demand) shock
end;

occbin_constraints;
  name 'zlb'; bind inot <= ilb; relax inot > ilb;
end;

steady;
check;

shocks(surprise);
  var e_rn; periods 1; values -1.25;           // a fall in the natural rate of 5 percentage points (
  ↪ annualised)
end;

occbin_setup;
occbin_solver(simul_periods = 40);
```

```
dynare zlb
STEADY-STATE RESULTS:

x             0
pi            0
i             0
rn            0
inot          0

EIGENVALUES:
         Modulus             Real        Imaginary
             0.8              0.8                0
           1.179            1.151            0.253
           1.179            1.151           -0.253

There are 2 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition is verified.
```

How to read it

- **Two versions of the policy equation**, tagged with the same `name` and with `relax` and `bind`, tell Dynare which equation holds in which regime.
- **`occbin_constraints`** states when the constraint binds and when it is relaxed, in terms of model variables. Using the shadow rate `inot` makes the condition transparent.
- **`shocks(surprise)`** defines the shock path for the simulation: an unexpected fall in the natural rate in period 1. **`occbin_solver`** computes the piecewise-linear solution and, for comparison, the linear solution that ignores the bound.
- The `check` output confirms that the model is determinate in the regime where the Taylor rule applies.

61.3 Compare the OccBin and the linear solution

```
P = oo_.occbin.simul.piecewise; L = oo_.occbin.simul.linear;
names = cellstr(M_.endo_names); id = @(v) find(strcmp(names, v));
T = 16; t = 1:T; ilb = M_.params(strcmp(cellstr(M_.param_names), 'ilb'));
f = figure('position', [0 0 900 300]);
vv = {'x', 'pi', 'i'}; tt = {'Output gap (%)', 'Inflation (annualised, pp)', 'Interest rate (annualised, pp)'};
  ↪  sc = [1 4 4];
for j = 1:3
  subplot(1, 3, j); plot(t, sc(j)*P(t, id(vv{j}))); hold on; plot(t, sc(j)*L(t, id(vv{j})), '--');
  if j == 3, plot(t, sc(j)*ilb*ones(size(t)), 'k:'); end
```

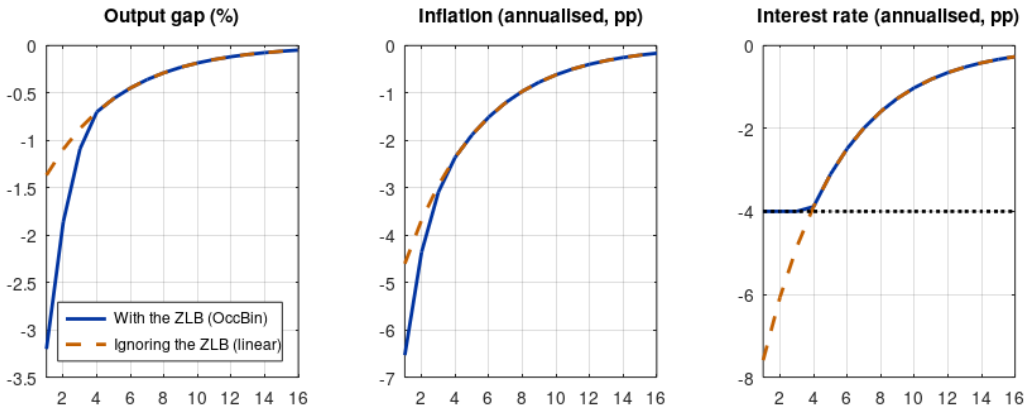


Figure 61.1. Response to a deep demand shock with the zero lower bound imposed (OccBin) and ignored (linear solution).

```
  title(tt{j}); grid on; xlim([1 T]);
  if j == 1, legend('With the ZLB (OccBin)', 'Ignoring the ZLB (linear)', 'location', 'southeast'); end
end
print(f, '-dpng', '-r110', 'zlb.png');
nb = sum(P(:, id('i')) <= ilb + 1e-8);
fprintf('Quarters at the ZLB: %d\n', nb);
fprintf('Trough of the output gap: OccBin %.2f%%, linear %.2f%%  (amplification x%.1f)\n', min(P(:, id('x'))),
  ↪  min(L(:, id('x'))), min(P(:, id('x')))/min(L(:, id('x'))));
fprintf('Trough of annual inflation: OccBin %.2f pp, linear %.2f pp\n', 4*min(P(:, id('pi'))), 4*min(L(:, id('
  ↪ pi'))));
fprintf('Lowest interest rate (annualised): OccBin %.2f pp, linear %.2f pp\n', 4*min(P(:, id('i'))), 4*min(L(:,
  ↪  id('i'))));
```

```
Quarters at the ZLB: 3
Trough of the output gap: OccBin -3.20%, linear -1.37%  (amplification x2.3)
Trough of annual inflation: OccBin -6.53 pp, linear -4.60 pp
Lowest interest rate (annualised): OccBin -4.00 pp, linear -7.59 pp
```

How to read it

- **The linear solution** lets the interest rate fall below the bound, by about 3.6 percentage points (annualised) beyond the steady-state rate of 4%, so it implies a negative nominal rate.
- **With the ZLB**, the rate stays at the bound for a few quarters. The real interest rate rises as inflation falls, the opposite of what the economy needs, so the output gap and inflation fall **much further** than in the linear solution.
- **Expectations matter.** Households and firms know the rate will be stuck for a while, which lowers expected inflation and raises the expected real rate even before the bound binds: the amplification comes partly from the future.
- This is why the ZLB is a nonlinearity a linear model cannot approximate: the size of a recession depends on whether the constraint binds.

61.4 Policy question: does a higher inflation target help?

```
fprintf('%-28s %10s %14s %16s\n', 'Inflation target (annual)', 'ZLB qtrs', 'trough x (%)', 'trough pi (pp)');
for target = [1 2 3 4]
  b = -(2 + target) / 4;                          % bound = -(real rate + target), quarterly
  clear functions; evalc(sprintf('dynare zlb noclearall nograph nolog console -DILB=%g', b));
  P2 = oo_.occbin.simul.piecewise;
  fprintf('%-28s %10d %14.2f %16.2f\n', sprintf('%d%%', target), sum(P2(:, id('i')) <= b + 1e-8), min(P2(:, id
  ↪ ('x'))), 4*min(P2(:, id('pi'))));
end
```

```
Inflation target (annual)      ZLB qtrs   trough x (%)   trough pi (pp)
1%                                    5          -4.70            -8.77
```

2%	3	-3.20	-6.53
3%	2	-2.33	-5.45
4%	2	-1.79	-4.90

How to read it

A higher inflation target raises the steady-state nominal rate and so gives the central bank more room to cut before reaching zero. The same shock then produces a shorter spell at the bound and a milder recession: with a 4% target instead of 2%, the trough of the output gap is about 45% shallower. This is the argument of Blanchard, Dell'Ariccia and Mauro (2010) for raising inflation targets. The model leaves out the costs of higher steady-state inflation (price dispersion, relative-price distortions), which a welfare analysis must add before drawing a policy conclusion.

61.5 Write the paper

How to write it: results

We solve the model with the occasionally binding zero lower bound using the OccBin algorithm of Guerrieri and Iacoviello (2015), as implemented in Dynare. Following a large adverse shock to the natural rate of interest, the policy rate reaches the bound and remains there for several quarters. Relative to a linear solution that ignores the bound, the fall in the output gap is more than twice as large and the fall in inflation substantially deeper, because the real interest rate rises as inflation falls. A higher inflation target shortens the spell at the bound and attenuates the recession.

Common mistakes

- Simulating a ZLB episode with a linear model and truncating the interest rate afterwards: expectations are then wrong.
- Writing the bound in levels when the model is in deviations (or the reverse). The bound in deviations is minus the steady-state rate.
- Forgetting that OccBin is a perfect-foresight solution conditional on the shock path: agents do not take into account the probability of future ZLB episodes. Global methods do, at a much higher computational cost.
- Estimating a model on ZLB-era data with a linear filter. Dynare's OccBin also provides a piecewise Kalman filter for estimation.

CHAPTER 62

Higher-Order Solutions and Welfare

Applies when. Nonlinearity: risk, large shocks, welfare
Overview. Use this chapter when the question involves risk, the average effect of volatility, asymmetric responses to large shocks, or welfare. First-order (linear) solutions cannot answer these: by certainty equivalence, uncertainty has no effect on average outcomes. The chapter solves the model at second and third order with pruning and computes precautionary saving and the welfare cost of business cycles.
Data. The calibrated RBC model of Chapter 54, extended with a welfare recursion
Software. Dynare 6 · Octave / MATLAB

62.1 The question and why first order is not enough

How costly are business cycles, and does uncertainty change how the economy behaves on average? In a first-order solution, the mean of every variable equals its steady state, whatever the size of the shocks: households behave as if they were certain about the future (certainty equivalence). That makes first-order solutions unable to measure:

- **precautionary behaviour**: uncertainty raising saving and the capital stock;
- **welfare**: the average utility loss from fluctuations, which depends on variances, a second-order term;
- **asymmetry**: large positive and negative shocks having responses of different size.

The model is the RBC model of Chapter 54 with one extra equation, a recursive definition of lifetime utility:

$$\mathcal{W}_t = \ln C_t - \psi \frac{H_t^{1+\varphi}}{1+\varphi} + \beta E_t \mathcal{W}_{t+1}$$

Pruning

Higher-order perturbation solutions can explode in simulations, because the squared terms feed back on themselves. Pruning (Kim, Kim, Schaumburg and Sims 2008; Andreasen, Fernández-Villaverde and Rubio-Ramírez 2018) removes the terms of higher order than the approximation, keeping simulations stable and making theoretical moments available. Always use it at order 2 and above.

62.2 Write the model in Dynare

```
// RBC model with a welfare recursion, solved at second (or third) order with pruning: dynare rbc2 -DORD=3
@#ifndef ORD
  @#define ORD = 2
@#endif
var y c i k h a W;
varexo e;
parameters beta alpha delta phi psi rho sigma_e h_ss;

beta = 0.99;  alpha = 0.33;  delta = 0.025;  phi = 1;  rho = 0.95;  sigma_e = 0.007;  h_ss = 1/3;

model;
  exp(-c) = beta * exp(-c(+1)) * (alpha * exp(y(+1) - k) + 1 - delta);
  psi * exp(phi*h) * exp(c) = (1 - alpha) * exp(y - h);
  exp(y) = exp(a) * exp(alpha*k(-1)) * exp((1 - alpha)*h);
```

```
  exp(k) = (1 - delta) * exp(k(-1)) + exp(i);
  exp(y) = exp(c) + exp(i);
  a = rho * a(-1) + e;
  W = c - psi * exp((1 + phi)*h) / (1 + phi) + beta * W(+1);       // lifetime utility (log C = c)
end;

steady_state_model;
  r = 1/beta - 1 + delta;
  k_h = (alpha / r)^(1/(1 - alpha));
  h = log(h_ss);  k = log(k_h * h_ss);  y = log(k_h^alpha * h_ss);
  i = log(delta * exp(k));  c = log(exp(y) - exp(i));
  psi = (1 - alpha) * exp(y) / h_ss / (h_ss^phi * exp(c));
  a = 0;
  W = (c - psi * h_ss^(1 + phi) / (1 + phi)) / (1 - beta);
end;

shocks;
  var e; stderr sigma_e;
end;

steady;
check;
stoch_simul(order = @{ORD}, pruning, irf = 0, periods = 0) y c k h W;
```

```
dynare rbc2
STEADY-STATE RESULTS:

y          0.00509623
c          -0.26283
i          -1.44292
k          2.24596
h          -1.09861
a          0
W          -70.0759

EIGENVALUES:
         Modulus             Real        Imaginary
            0.95             0.95                0
          0.9531           0.9531                0
            1.01             1.01                0
            1.06             1.06                0
       5.548e+17       -5.548e+17                0

There are 3 eigenvalue(s) larger than 1 in modulus for 3 forward-looking variable(s)
The rank condition is verified.

MODEL SUMMARY

  Number of variables:         7
  Number of stochastic shocks: 1
  Number of state variables:   2
  Number of jumpers:           3
  Number of static variables:  2

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables           e
e            0.000049
```

How to read it

The policy functions now contain a **constant correction** and terms in squared states. The constant (the `ghs2` term in Dynare's notation) is how risk enters: it shifts every variable away from its steady state even when no shock occurs in a given period. The theoretical moments below are computed from the pruned second-order solution.

62.3 Precautionary saving and the welfare cost of fluctuations

```
names = cellstr(M_.endo_names);
id = @(v) find(strcmp(names, v));
beta_ = M_.params(strcmp(cellstr(M_.param_names), 'beta'));
mean2 = oo_.mean; ss = oo_.dr.ys;               % pruned second-order means of the listed variables (y c k h
  ↪ W) and the steady state
lv = {'y', 'c', 'k', 'h'};
fprintf('%-4s %12s %14s %14s\n', '', 'steady state', 'mean (order 2)', 'difference (%)');
for j = 1:4
  fprintf('%-4s %12.4f %14.4f %14.3f\n', lv{j}, ss(id(lv{j})), mean2(j), 100*(mean2(j) - ss(id(lv{j}))));
end
dW = mean2(5) - ss(id('W'));
lambda = 1 - exp((1 - beta_) * dW);
fprintf('\nWelfare: steady state %.4f, mean %.4f\n', ss(id('W')), mean2(5));
fprintf('Welfare cost of fluctuations: %.4f%% of consumption every period\n', 100*lambda);
```

```
     steady state mean (order 2) difference (%)
y          0.0051         0.0052          0.009
c         -0.2628        -0.2626          0.024
k          2.2460         2.2464          0.043
h         -1.0986        -1.0987         -0.008

Welfare: steady state -70.0759, mean -70.0500
Welfare cost of fluctuations: -0.0259% of consumption every period
```

How to read it

- **Precautionary saving.** With uncertainty, the average capital stock is above its steady state: households save more as insurance against bad productivity draws. Output is higher on average as a result. A first-order solution would show zero difference for every variable.
- **Welfare cost.** The consumption-equivalent cost λ solves $E[\mathcal{W}] = \mathcal{W}_{ss} + \ln(1-\lambda)/(1-\beta)$: the share of consumption households would give up every period to live in the steady state instead of the fluctuating economy. Here it is **negative**, about –0.03%: households are slightly *better off* with fluctuations. That is not a bug. With elastic labour supply and capital, households can work and invest more when productivity is high and less when it is low, and that option is worth more than the utility lost to consumption volatility (Cho, Cooley and Kim 2015). Lucas (1987) found the cost of fluctuations to be tiny in an endowment economy; in this production economy it is tiny and of the opposite sign. Models with incomplete markets, unemployment risk or nominal rigidities typically produce positive and larger costs.

62.4 How the answer changes with the size of shocks

```
s0 = M_.params(strcmp(cellstr(M_.param_names), 'sigma_e'));
fprintf('%-22s %16s %16s\n', '', 'mean k - ss (%)', 'welfare cost (%)');
for scale = [0.5 1 2 3]
  M_.Sigma_e = (scale * s0)^2; options_.noprint = true;
  [info, oo_, options_] = stoch_simul(M_, options_, oo_, {'y', 'c', 'k', 'h', 'W'});   % theoretical pruned
  ↪ moments
  m = oo_.mean;
  fprintf('%-22s %16.4f %16.4f\n', sprintf('shock sd x %.1f', scale), 100*(m(3) - ss(id('k'))), 100*(1 - exp
  ↪ ((1 - beta_)*(m(5) - ss(id('W'))))));
end
M_.Sigma_e = s0^2;
```

```
                       mean k - ss (%) welfare cost (%)
shock sd x 0.5                  0.0108          -0.0065
shock sd x 1.0                  0.0430          -0.0259
shock sd x 2.0                  0.1720          -0.1037
shock sd x 3.0                  0.3871          -0.2334
```

How to read it

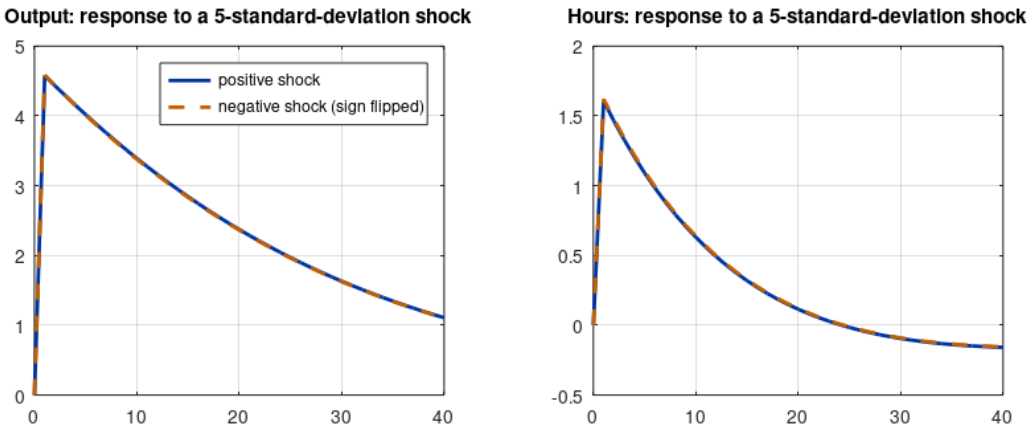


Figure 62.1. Responses of output and hours to a positive and to a negative five-standard-deviation shock, the latter sign-flipped. The two lines nearly coincide.

- **Size of shocks.** Doubling the shock standard deviation multiplies the precautionary capital and the welfare effect by four, tripling it by nine: both are proportional to the **variance** of the shocks, the signature of a second-order effect.
- **Third order.** The model file can be solved at third order with `dynare rbc2 -DORD=3`. With symmetric shocks, third-order terms leave average outcomes essentially unchanged; they matter for time-varying risk premia and models with stochastic volatility. Dynare provides theoretical moments only up to second order, so third-order results come from long pruned simulations.

62.5 Asymmetric responses to large shocks

```
clear functions; evalc('dynare rbc2 noclearall nograph nolog -DORD=2');
T = 40; big = 5 * s0;
base = simult_(M_, options_, oo_.dr.ys, oo_.dr, zeros(T, 1), 2);
up   = simult_(M_, options_, oo_.dr.ys, oo_.dr, [big; zeros(T-1, 1)], 2);
down = simult_(M_, options_, oo_.dr.ys, oo_.dr, [-big; zeros(T-1, 1)], 2);
f = figure('position', [0 0 860 300]);
vv = {'y', 'h'}; tt = {'Output', 'Hours'};
for j = 1:2
  subplot(1, 2, j);
  plot(0:T, 100*(up(id(vv{j}), :) - base(id(vv{j}), :))); hold on;
  plot(0:T, -100*(down(id(vv{j}), :) - base(id(vv{j}), :)), '--');
  title([tt{j} ': response to a 5-standard-deviation shock']); grid on; xlim([0 T]);
  if j == 1, legend('positive shock', 'negative shock (sign flipped)'); end
end
print(f, '-dpng', '-r110', 'asym.png');
fprintf('Peak output response: positive %.2f%%, negative %.2f%% (sign flipped)\n', 100*max(up(id('y'), :) -
  ↪ base(id('y'), :)), -100*min(down(id('y'), :) - base(id('y'), :)));
```

```
Peak output response: positive 4.58%, negative 4.59% (sign flipped)
```

How to read it

With a linear solution the two lines would coincide exactly. At second order the squared terms add the same correction to positive and negative shocks, so their responses can differ in size. In this smooth RBC model the difference is **virtually nil**, even for a five-standard-deviation shock: the model is close to linear around its steady state. Strong asymmetries need strong nonlinearities, such as the occasionally binding constraint of Chapter 61.

62.6 Write the paper

How to write it: results

We solve the model with a second-order perturbation and pruning, which captures the effect of uncertainty on average outcomes. Productivity risk raises the average capital stock above its deterministic steady state, reflecting precautionary saving. The welfare effect of business-cycle fluctuations, measured as the permanent percentage of consumption that would make households indifferent between the stochastic economy and its deterministic steady state, is very small and slightly favourable: with elastic labour supply, households exploit productivity fluctuations by working and investing more in good times. Both effects scale with the variance of the shocks.

Common mistakes

- Computing welfare from a first-order solution. The linear approximation of a utility function gets welfare comparisons wrong; use second order.
- Simulating at order 2 or 3 without pruning.
- Comparing welfare across policies without a common starting point: conditional and unconditional welfare can rank policies differently.
- Reporting means from a first-order solution as "effects of uncertainty": they equal the steady state by construction.

CHAPTER A

Data of the Chapters

Table A.1 lists the dataset that each method chapter uses, with its source as given in the chapter. The files are small enough to be distributed with the code, so that every table and figure can be reproduced from the raw data.

Table A.1. Dataset of each method chapter.

Ch.	Chapter	Data
3	OLS Regression	U.S. stock returns and the economy, monthly, 1985–2019
4	VAR in First Differences	U.S. stock prices and industrial production, monthly, 1985–2019
5	Vector Error-Correction Model	U.S. 10-year and 1-year Treasury yields, quarterly, 1960–2019
6	ARDL Bounds-Testing Model	U.S. 10-year Treasury yield, inflation, GDP growth and unemployment, quarterly, 1960–2019
7	Nonlinear ARDL (NARDL)	U.S. Baa corporate bond yield and federal funds rate, monthly, 1985–2019
8	Structural VAR (Cholesky, Long-Run and Proxy Identification)	U.S. quarterly data, 1960–2019: real GDP growth and the unemployment rate; U.S. monthly data, 1990–2019: policy rate, industrial production, prices, credit spread and a high-frequency monetary shock
9	Local Projections (Impulse Responses)	U.S. monthly data, 1985–2019: industrial-production growth, S&P 500 return, change in the federal funds rate, change in the credit spread
10	Threshold Regression and SETAR: Regimes Defined by a Variable	U.S. quarterly data, 1960–2019: GDP-deflator inflation
11	Smooth-Transition Regression (LSTAR and ESTAR)	U.S. quarterly data, 1960–2019: the federal funds rate, inflation and the output gap
12	Structural Breaks at Unknown Dates (Bai–Perron)	U.S. quarterly data, 1960–2019: GDP-deflator inflation and real GDP growth
13	2SLS and GMM	U.S. policy rule, quarterly, 1979–2007; UK firm employment panel, 1976–1984
14	Pooled OLS	140 UK manufacturing firms, annual growth rates, 1977–1984 (Arellano–Bond data)
15	Fixed Effects	545 U.S. men, annual wages, 1980–1987 (Wooldridge wagepan)
16	Random Effects	532 U.S. men, annual hours and wages, 1979–1988 (PSID, Ziliak 1997)
17	Difference GMM for Dynamic Panels	140 UK manufacturing firms, annual, 1976–1984 (Arellano and Bond 1991)
18	Fixed Effects with Driscoll–Kraay Standard Errors	14 OECD economies, annual, 1967–1990 (Garrett 1998)
19	Mean Group and CCE Mean Group	125 countries, annual, 1961–1985 (Summers–Heston Penn World Table)

Ch.	Chapter	Data
20	Panel ARDL: Pooled Mean Group	46 U.S. states, annual cigarette sales, 1963–1992 (Baltagi and Levin)
21	Cross-Sectionally Augmented ARDL	46 U.S. states, annual cigarette sales, 1963–1992 (Baltagi and Levin)
22	Panel Model in First Differences	18 OECD countries, annual, 1960–1978 (Baltagi and Griffin 1983)
23	Panel Local Projections	Cigarette sales and prices, 46 U.S. states, 1963–1992
24	Panel Threshold Regression (Hansen)	North Carolina counties, 1981–1987 (Cornwell and Trumbull 1994): crime rate, probability of arrest, probability of conviction, average sentence and police per capita
25	Instrumental Variables	3,010 U.S. men in 1976 (National Longitudinal Survey of Young Men, Card 1995)
26	Difference-in-Differences	Castle Doctrine (stand-your-ground) laws, 13 states adopting in 2007 and 29 never-treated states, 2000–2010
27	Staggered Difference-in-Differences	Castle Doctrine laws, 21 states adopting between 2006 and 2010, 50 states, 2000–2010
28	Generalized Synthetic Control	Castle Doctrine laws and the murder rate, 50 U.S. states, 2000–2010
29	Synthetic Control	Texas prison expansion of 1993 and the other U.S. states, black male prisoners, 1985–2000
30	Interrupted Time Series	Car drivers killed or seriously injured in Great Britain, monthly, 1969–1984 (Harvey and Durbin 1986)
31	Regression Discontinuity	U.S. House elections, 1946–1995: ADA voting score of the winner by the Democratic vote share (Lee, Moretti and Butler 2004)
32	LASSO	814 students, introductory economics course (Wooldridge econmath)
33	Ridge Regression	663 U.S. men, 1980 wages (Blackburn and Neumark; Wooldridge wage2)
34	Elastic Net	U.S. quarterly macro data, 1960–2009 (statsmodels macrodata)
35	Random Forest	353 Major League Baseball players, 1993 salaries (Wooldridge mlb1)
36	Gradient Boosting (XGBoost)	6,822 injured workers in Kentucky and Michigan (Meyer, Viscusi and Durbin; Wooldridge injury)
37	Double Machine Learning: Partially Linear Model	90 countries, growth 1965–1975, 60 controls (Barro–Lee; Belloni, Chernozhukov and Hansen)
38	Double Machine Learning: Interactive Model	9,915 U.S. households, 1991 SIPP (Poterba, Venti and Wise; Chernozhukov et al. 2018)
39	Double Machine Learning with an Instrument (DML-IV)	9,915 U.S. households, 1991 SIPP (Chernozhukov et al. 2018)
40	Causal Forest	9,915 U.S. households, 1991 SIPP (Chernozhukov et al. 2018)
41	Bayesian Linear Regression with a g-Prior	Wooldridge WAGE2, 663 U.S. men with complete data: log wage, education, experience, tenure, IQ and demographics
42	Bayesian VAR with Minnesota and Normal-Wishart Priors	U.S. quarterly GDP growth, inflation and the 3-month T-bill rate, 1959–2009
43	Mean-Adjusted (Steady-State) Bayesian VAR	U.S. quarterly GDP growth, inflation and the 3-month T-bill rate, 1959–2009

Ch.	Chapter	Data
44	Factor-Augmented VAR (FAVAR)	U.S. quarterly macro data, 1959–2009: inflation, unemployment and the T-bill rate, with six real-activity series as the information set
45	Mixed-Frequency BVAR: Nowcasting Quarterly GDP with Monthly Data	U.S. monthly industrial production growth and federal funds rate, and quarterly GDP growth, 1985–2019 (the example file shipped with OES)
46	Panel VAR: Six Estimators for Heterogeneous Countries	Garrett (1998) OECD panel: GDP growth and unemployment in 14 countries, 1966–1990
47	Stochastic-Volatility VAR (SV-VAR)	U.S. quarterly inflation, unemployment and the T-bill rate, 1959–2009
48	Time-Varying Parameter VAR (TVP-VAR)	U.S. quarterly inflation, unemployment and the T-bill rate, 1959–2009 (first 10 years used as a training sample)
49	Bayesian Threshold VAR	U.S. quarterly GDP growth, inflation and the T-bill rate, 1959–2009
50	Markov-Switching Autoregression (Hamilton)	U.S. quarterly GNP growth, 1951Q2–1984Q4 (Hamilton 1989)
51	Markov-Switching Regression	U.S. quarterly inflation and unemployment, 1959–2009
52	Markov-Switching VAR	U.S. quarterly GDP growth and inflation, 1959–2009
53	Markov-Switching Panel VAR	14 OECD countries, annual GDP growth and unemployment, 1966–1990
54	Real Business Cycle Model	Calibrated to the U.S. economy; moments compared with U.S. data, 1966–2007
55	Three-Equation New Keynesian Model	Calibrated quarterly model (Galí 2015 calibration)
56	Small Open-Economy New Keynesian Model	Calibrated quarterly model (Galí and Monacelli 2005, Canada calibration)
57	Optimal Monetary Policy	Calibrated New Keynesian model with cost-push and demand shocks
58	Identification Before Estimation	The An–Schorfheide New Keynesian model of the estimation guide
59	Bayesian Estimation of a New Keynesian Model	U.S. output growth, inflation and the federal funds rate, 1966Q1–2007Q4
60	Medium-Scale New Keynesian Model	U.S. output, consumption and investment growth, inflation and the federal funds rate, 1966Q1–2007Q4
61	The Zero Lower Bound with OccBin	Calibrated New Keynesian model, deep demand shock
62	Higher-Order Solutions and Welfare	The calibrated RBC model of the first guide, extended with a welfare recursion

CHAPTER B

Software and Reproducibility

B.1 Python

The empirical chapters use Python 3 with the following packages: `numpy`, `pandas`, `scipy` and `matplotlib` for computation and graphics; `statsmodels` for regression, time-series models and diagnostic tests; `linearmodels` for panel estimators and instrumental variables; `scikit-learn` and `XGBoost` for prediction; `DoubleML` and `EconML` for double machine learning and causal forests. Helper files for the specification tests that are not in a standard library accompany the chapters that need them.

B.2 The OES engine

The Bayesian and Markov-switching chapters call the engine of the Open Econometric Systems project (`oes_engine`). It implements the priors, samplers, regime-switching likelihoods and reporting functions of those chapters behind a single interface: a specification is passed as a dictionary, `oes.run` estimates the model, and `oes.show`, `oes.summary` and `oes.plot` report it. The chapters print every specification in full.

B.3 Dynare

The DSGE chapters use Dynare, run from MATLAB or GNU Octave. The model file is printed in each chapter, and the output reproduced is Dynare's own. The figures of that part are Dynare's graphs exported from the estimation and simulation runs.

B.4 Figures

Figures in the Python chapters were regenerated from the code in the chapters with a common style: one typeface family, a colour-blind-safe palette in which series also differ in line style, no top or right axis lines, and vector output. The decision maps of Chapter 1 were drawn with the same settings.

B.5 Random numbers

Cross-validation, bootstrap and simulation steps set a seed in the code where it matters. Estimates that depend on simulation (the Bayesian chains, bootstrap tests, forests) can differ in the last digit across software versions and platforms, and the text of each chapter describes them with the precision that is stable.

Bibliography

Alberto Abadie and Javier Gardeazabal. The Economic Costs of Conflict: A Case Study of the Basque Country. *American Economic Review*, 93(1):113–132, 2003.

Alberto Abadie, Alexis Diamond, and Jens Hainmueller. Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association*, 105(490):493–505, 2010.

Stéphane Adjemian et al. Dynare: Reference manual, version 4. Dynare Working Paper 1, CEPREMAP, 2011.

Sungbae An and Frank Schorfheide. Bayesian Analysis of DSGE Models. *Econometric Reviews*, 26(2–4): 113–172, 2007.

T. W. Anderson and Herman Rubin. Estimation of the Parameters of a Single Equation in a Complete System of Stochastic Equations. *Annals of Mathematical Statistics*, 20(1):46–63, 1949.

Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, Princeton, 2009.

Manuel Arellano and Stephen Bond. Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations. *Review of Economic Studies*, 58(2):277–297, 1991.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized Random Forests. *Annals of Statistics*, 47(2): 1148–1178, 2019.

Jushan Bai and Pierre Perron. Estimating and Testing Linear Models with Multiple Structural Changes. *Econometrica*, 66(1):47–78, 1998.

Marta Bańbura, Domenico Giannone, and Lucrezia Reichlin. Large Bayesian Vector Auto Regressions. *Journal of Applied Econometrics*, 25(1):71–92, 2010.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on Treatment Effects after Selection among High-Dimensional Controls. *Review of Economic Studies*, 81(2):608–650, 2014.

Ben S. Bernanke, Jean Boivin, and Piotr Eliasz. Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach. *Quarterly Journal of Economics*, 120(1):387–422, 2005.

Marianne Bertrand, Esther Duflo, and Sendhil Mullainathan. How Much Should We Trust Differences-in-Differences Estimates? *Quarterly Journal of Economics*, 119(1):249–275, 2004.

Olivier Jean Blanchard and Charles M. Kahn. The Solution of Linear Difference Models under Rational Expectations. *Econometrica*, 48(5):1305–1311, 1980.

Richard Blundell and Stephen Bond. Initial Conditions and Moment Restrictions in Dynamic Panel Data Models. *Journal of Econometrics*, 87(1):115–143, 1998.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

Trevor S. Breusch and Adrian R. Pagan. The Lagrange Multiplier Test and its Applications to Model Specification in Econometrics. *Review of Economic Studies*, 47(1):239–253, 1980.

R. L. Brown, J. Durbin, and J. M. Evans. Techniques for Testing the Constancy of Regression Relationships over Time. *Journal of the Royal Statistical Society: Series B*, 37(2):149–192, 1975.

Brantly Callaway and Pedro H. C. Sant'Anna. Difference-in-Differences with Multiple Time Periods. *Journal of Econometrics*, 225(2):200–230, 2021.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/Debiased Machine Learning for Treatment and Structural Parameters. *Econometrics Journal*, 21(1):C1–C68, 2018.

Lawrence J. Christiano, Martin Eichenbaum, and Charles L. Evans. Nominal Rigidities and the Dynamic Effects of a Shock to Monetary Policy. *Journal of Political Economy*, 113(1):1–45, 2005.

Alexander Chudik and M. Hashem Pesaran. Common Correlated Effects Estimation of Heterogeneous Dynamic Panel Data Models with Weakly Exogenous Regressors. *Journal of Econometrics*, 188(2):393–420, 2015.

Timothy Cogley and Thomas J. Sargent. Drifts and Volatilities: Monetary Policies and Outcomes in the Post WWII US. *Review of Economic Dynamics*, 8(2):262–302, 2005.

Marco Del Negro and Frank Schorfheide. Bayesian Macroeconometrics. In *The Oxford Handbook of Bayesian Econometrics*, pages 293–389. Oxford University Press, 2011.

David A. Dickey and Wayne A. Fuller. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.

Thomas Doan, Robert Litterman, and Christopher Sims. Forecasting and Conditional Projection Using Realistic Prior Distributions. *Econometric Reviews*, 3(1):1–100, 1984.

John C. Driscoll and Aart C. Kraay. Consistent Covariance Matrix Estimation with Spatially Dependent Panel Data. *Review of Economics and Statistics*, 80(4):549–560, 1998.

Robert F. Engle and C. W. J. Granger. Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, 55(2):251–276, 1987.

Carmen Fernández, Eduardo Ley, and Mark F. J. Steel. Benchmark Priors for Bayesian Model Averaging. *Journal of Econometrics*, 100(2):381–427, 2001.

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

Jordi Galí. *Monetary Policy, Inflation, and the Business Cycle: An Introduction to the New Keynesian Framework and Its Applications*. Princeton University Press, Princeton, 2015.

Jordi Galí and Tommaso Monacelli. Monetary Policy and Exchange Rate Volatility in a Small Open Economy. *Review of Economic Studies*, 72(3):707–734, 2005.

René Garcia. Asymptotic Null Distribution of the Likelihood Ratio Test in Markov Switching Models. *International Economic Review*, 39(3):763–788, 1998.

Andrew Goodman-Bacon. Difference-in-Differences with Variation in Treatment Timing. *Journal of Econometrics*, 225(2):254–277, 2021.

C. W. J. Granger and P. Newbold. Spurious Regressions in Econometrics. *Journal of Econometrics*, 2(2): 111–120, 1974.

Luca Guerrieri and Matteo Iacoviello. OccBin: A Toolkit for Solving Dynamic Models with Occasionally Binding Constraints Easily. *Journal of Monetary Economics*, 70(1):22–38, 2015.

James D. Hamilton. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, 57(2):357–384, 1989.

James D. Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, 1994.

Bruce E. Hansen. Threshold Effects in Non-Dynamic Panels: Estimation, Testing, and Inference. *Journal of Econometrics*, 93(2):345–368, 1999.

Bruce E. Hansen. Sample Splitting and Threshold Estimation. *Econometrica*, 68(3):575–603, 2000.

Lars Peter Hansen. Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, 50(4):1029–1054, 1982.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.

Jerry A. Hausman. Specification Tests in Econometrics. *Econometrica*, 46(6):1251–1271, 1978.

Edward P. Herbst and Frank Schorfheide. *Bayesian Estimation of DSGE Models*. Princeton University Press, Princeton, 2016.

Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

Guido Imbens and Karthik Kalyanaraman. Optimal Bandwidth Choice for the Regression Discontinuity Estimator. *Review of Economic Studies*, 79(3):933–959, 2012.

Guido W. Imbens and Thomas Lemieux. Regression Discontinuity Designs: A Guide to Practice. *Journal of Econometrics*, 142(2):615–635, 2008.

Nikolay Iskrev. Local Identification in DSGE Models. *Journal of Monetary Economics*, 57(2):189–202, 2010.

Søren Johansen. Estimation and Hypothesis Testing of Cointegration Vectors in Gaussian Vector Autoregressive Models. *Econometrica*, 59(6):1551–1580, 1991.

Òscar Jordà. Estimation and Inference of Impulse Responses by Local Projections. *American Economic Review*, 95(1):161–182, 2005.

K. Rao Kadiyala and Sune Karlsson. Numerical Methods for Estimation and Inference in Bayesian VAR-Models. *Journal of Applied Econometrics*, 12(2):99–132, 1997.

Chihwa Kao. Spurious Regression and Residual-Based Tests for Cointegration in Panel Data. *Journal of Econometrics*, 90(1):1–44, 1999.

Lutz Kilian and Helmut Lütkepohl. *Structural Vector Autoregressive Analysis*. Cambridge University Press, Cambridge, 2017.

Chang-Jin Kim and Charles R. Nelson. *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*. MIT Press, Cambridge, MA, 1999.

Gary Koop and Dimitris Korobilis. Bayesian Multivariate Time Series Methods for Empirical Macroeconomics. *Foundations and Trends in Econometrics*, 3(4):267–358, 2010.

Hans-Martin Krolzig. *Markov-Switching Vector Autoregressions: Modelling, Statistical Inference, and Application to Business Cycle Analysis*. Springer, Berlin, 1997.

Denis Kwiatkowski, Peter C. B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root. *Journal of Econometrics*, 54(1–3):159–178, 1992.

Finn E. Kydland and Edward C. Prescott. Time to Build and Aggregate Fluctuations. *Econometrica*, 50 (6):1345–1370, 1982.

Edward E. Leamer. Let's Take the Con out of Econometrics. *American Economic Review*, 73(1):31–43, 1983.

David S. Lee. Randomized Experiments from Non-Random Selection in U.S. House Elections. *Journal of Econometrics*, 142(2):675–697, 2008.

Robert B. Litterman. Forecasting with Bayesian Vector Autoregressions: Five Years of Experience. *Journal of Business & Economic Statistics*, 4(1):25–38, 1986.

Inessa Love and Lea Zicchino. Financial Development and Dynamic Investment Behavior: Evidence from Panel VAR. *Quarterly Review of Economics and Finance*, 46(2):190–210, 2006.

Justin McCrary. Manipulation of the Running Variable in the Regression Discontinuity Design: A Density Test. *Journal of Econometrics*, 142(2):698–714, 2008.

Sendhil Mullainathan and Jann Spiess. Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.

Yair Mundlak. On the Pooling of Time Series and Cross Section Data. *Econometrica*, 46(1):69–85, 1978.

Whitney K. Newey and Kenneth D. West. A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, 55(3):703–708, 1987.

Stephen Nickell. Biases in Dynamic Models with Fixed Effects. *Econometrica*, 49(6):1417–1426, 1981.

M. Hashem Pesaran. Estimation and Inference in Large Heterogeneous Panels with a Multifactor Error Structure. *Econometrica*, 74(4):967–1012, 2006.

M. Hashem Pesaran. A Simple Panel Unit Root Test in the Presence of Cross-Section Dependence. *Journal of Applied Econometrics*, 22(2):265–312, 2007.

M. Hashem Pesaran. Testing Weak Cross-Sectional Dependence in Large Panels. *Econometric Reviews*, 34 (6–10):1089–1117, 2015.

M. Hashem Pesaran and Ron Smith. Estimating Long-Run Relationships from Dynamic Heterogeneous Panels. *Journal of Econometrics*, 68(1):79–113, 1995.

M. Hashem Pesaran and Takashi Yamagata. Testing Slope Homogeneity in Large Panels. *Journal of Econometrics*, 142(1):50–93, 2008.

M. Hashem Pesaran, Yongcheol Shin, and Ron P. Smith. Pooled Mean Group Estimation of Dynamic Heterogeneous Panels. *Journal of the American Statistical Association*, 94(446):621–634, 1999.

M. Hashem Pesaran, Yongcheol Shin, and Richard J. Smith. Bounds Testing Approaches to the Analysis of Level Relationships. *Journal of Applied Econometrics*, 16(3):289–326, 2001.

Giorgio E. Primiceri. Time Varying Structural Vector Autoregressions and Monetary Policy. *Review of Economic Studies*, 72(3):821–852, 2005.

J. B. Ramsey. Tests for Specification Errors in Classical Linear Least-Squares Regression Analysis. *Journal of the Royal Statistical Society: Series B*, 31(2):350–371, 1969.

Stephanie Schmitt-Grohé and Martín Uribe. Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function. *Journal of Economic Dynamics and Control*, 28(4):755–775, 2004.

Frank Schorfheide and Dongho Song. Real-Time Forecasting with a Mixed-Frequency VAR. *Journal of Business & Economic Statistics*, 33(3):366–380, 2015.

Yongcheol Shin, Byungchul Yu, and Matthew Greenwood-Nimmo. Modelling Asymmetric Cointegration and Dynamic Multipliers in a Nonlinear ARDL Framework. In *Festschrift in Honor of Peter Schmidt: Econometric Methods and Applications*, pages 281–314. Springer, 2014.

Christopher A. Sims. Macroeconomics and Reality. *Econometrica*, 48(1):1–48, 1980.

Christopher A. Sims and Tao Zha. Were There Regime Switches in U.S. Monetary Policy? *American Economic Review*, 96(1):54–81, 2006.

Frank Smets and Rafael Wouters. Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach. *American Economic Review*, 97(3):586–606, 2007.

James H. Stock and Mark W. Watson. Vector Autoregressions. *Journal of Economic Perspectives*, 15(4): 101–115, 2001.

James H. Stock and Motohiro Yogo. Testing for Weak Instruments in Linear IV Regression. In *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, pages 80–108. Cambridge University Press, 2005.

Liyang Sun and Sarah Abraham. Estimating Dynamic Treatment Effects in Event Studies with Heterogeneous Treatment Effects. *Journal of Econometrics*, 225(2):175–199, 2021.

Timo Teräsvirta. Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models. *Journal of the American Statistical Association*, 89(425):208–218, 1994.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

Hal R. Varian. Big Data: New Tricks for Econometrics. *Journal of Economic Perspectives*, 28(2):3–28, 2014.

Mattias Villani. Steady-State Priors for Vector Autoregressions. *Journal of Applied Econometrics*, 24(4): 630–650, 2009.

Stefan Wager and Susan Athey. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Halbert White. A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 48(4):817–838, 1980.

Michael Woodford. *Interest and Prices: Foundations of a Theory of Monetary Policy*. Princeton University Press, Princeton, 2003.

Jeffrey M. Wooldridge. *Econometric Analysis of Cross Section and Panel Data*. MIT Press, Cambridge, MA, 2010.

Yiqing Xu. Generalized Synthetic Control Method: Causal Inference with Interactive Fixed Effects Models. *Political Analysis*, 25(1):57–76, 2017.

Arnold Zellner. On Assessing Prior Distributions and Bayesian Regression Analysis with g-Prior Distributions. In *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*, pages 233–243. North-Holland, 1986.

Hui Zou and Trevor Hastie. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320, 2005.