

Introduction to Econometrics with R

Christoph Hanck, Martin Arnold, Alexander Gerber, and Martin Schmelzer

2024-02-13

Contents

Preface	4
1 Introduction	6
1.1 Colophon	8
1.2 A Very Short Introduction to R and <i>RStudio</i>	11
2 Probability Theory	14
2.1 Random Variables and Probability Distributions	14
2.2 Random Sampling and the Distribution of Sample Averages	33
2.3 Exercises	45
3 A Review of Statistics using R	46
3.1 Estimation of the Population Mean	46
3.2 Properties of the Sample Mean	48
3.3 Hypothesis Tests concerning the Population Mean	54
3.4 Confidence Intervals for the Population Mean	65
3.5 Comparing Means from Different Populations	67
3.6 An Application to the Gender Gap of Earnings	69
3.7 Scatterplots, Sample Covariance and Sample Correlation	71
3.8 Exercises	74
4 Linear Regression with One Regressor	75
4.1 Simple Linear Regression	75
4.2 Estimating the Coefficients of the Linear Regression Model	77
4.3 Measures of Fit	83
4.4 The Least Squares Assumptions	85
4.5 The Sampling Distribution of the OLS Estimator	90
4.6 Exercises	98
5 Hypothesis Tests and Confidence Intervals in SLR Model	99
5.1 Testing Two-Sided Hypotheses concerning the Slope Coefficient	99
5.2 Confidence Intervals for Regression Coefficients	103
5.3 Regression when X is a Binary Variable	108
5.4 Heteroskedasticity and Homoskedasticity	111
5.5 The Gauss-Markov Theorem	119
5.6 Using the t-Statistic in Regression when the Sample Size Is Small	122
5.7 Exercises	124
6 Regression Models with Multiple Regressors	125
6.1 Omitted Variable Bias	125

6.2	The Multiple Regression Model	128
6.3	Measures of Fit in Multiple Regression	129
6.4	OLS Assumptions in Multiple Regression	131
6.5	The Distribution of the OLS Estimators in Multiple Regression	139
6.6	Exercises	142
7	Hypothesis Tests and Confidence Intervals in MR Models	143
7.1	Hypothesis Tests and Confidence Intervals for a Single Coefficient	143
7.2	An Application to Test Scores and the Student-Teacher Ratio	145
7.3	Joint Hypothesis Testing using the F-Statistic	147
7.4	Confidence Sets for Multiple Coefficients	149
7.5	Model Specification for Multiple Regression	151
7.6	Analysis of the Test Score Data Set	155
7.7	Exercises	159
8	Nonlinear Regression Functions	160
8.1	A General Strategy for Modelling Nonlinear Regression Functions	160
8.2	Nonlinear Functions of a Single Independent Variable	163
8.3	Interactions between Independent Variables	174
8.4	Nonlinear Effects on Test Scores of the Student-Teacher Ratio	190
8.5	Exercises	197
9	Assessing Studies Based on Multiple Regression	198
9.1	Internal and External Validity	199
9.2	Threats to Internal Validity of Multiple Regression Analysis	199
9.3	Internal and External Validity when the Regression is used for Forecasting	212
9.4	Example: Test Scores and Class Size	213
9.5	Exercises	223
10	Regression with Panel Data	224
10.1	Panel Data	225
10.2	Panel Data with Two Time Periods: “Before and After” Comparisons	232
10.3	Fixed Effects Regression	234
10.4	Regression with Time Fixed Effects	237
10.5	The Fixed Effects Regression Assumptions and Standard Errors for Fixed Effects Regression	239
10.6	Drunk Driving Laws and Traffic Deaths	241
10.7	Exercises	245
11	Regression with a Binary Dependent Variable	246
11.1	Binary Dependent Variables and the Linear Probability Model	247
11.2	Probit and Logit Regression	251
11.3	Estimation and Inference in the Logit and Probit Models	257
11.4	Application to the Boston HMDA Data	260
11.5	Exercises	266
12	Instrumental Variables Regression	267
12.1	The IV Estimator with a Single Regressor and a Single Instrument	267
12.2	The General IV Regression Model	272
12.3	Checking Instrument Validity	275
12.4	Application to the Demand for Cigarettes	277
12.5	Where Do Valid Instruments Come From?	282
12.6	Exercises	283

13 Experiments and Quasi-Experiments	284
13.1 Potential Outcomes, Causal Effects and Idealized Experiments	285
13.2 Threats to Validity of Experiments	286
13.3 Experimental Estimates of the Effect of Class Size Reductions	287
13.4 Quasi Experiments	297
13.5 Exercises	308
14 Introduction to Time Series Regression and Forecasting	309
14.1 Using Regression Models for Forecasting	310
14.2 Time Series Data and Serial Correlation	310
14.3 Autoregressions	317
14.4 Can You Beat the Market? (Part I)	321
14.5 Additional Predictors and The ADL Model	323
14.6 Lag Length Selection using Information Criteria	332
14.7 Nonstationarity I: Trends	335
14.8 Nonstationarity II: Breaks	347
14.9 Can You Beat the Market? (Part II)	353
15 Estimation of Dynamic Causal Effects	359
15.1 The Orange Juice Data	359
15.2 Dynamic Causal Effects	363
15.3 Dynamic Multipliers and Cumulative Dynamic Multipliers	364
15.4 HAC Standard Errors	366
15.5 Estimation of Dynamic Causal Effects with Strictly Exogeneous Regressors	368
15.6 Orange Juice Prices and Cold Weather	374
16 Additional Topics in Time Series Regression	382
16.1 Vector Autoregressions	383
16.2 Orders of Integration and the DF-GLS Unit Root Test	391
16.3 Cointegration	394
16.4 Volatility Clustering and Autoregressive Conditional Heteroskedasticity	401

Preface

Chair of Econometrics, Department of Business Administration and Economics University of Duisburg-Essen Essen, Germany info@econometrics-with-r.org. Last updated on Tuesday, February 13, 2024.

Over the last few years, the statistical programming language R has become an integral part of the curricula of econometrics classes we teach at the University of Duisburg-Essen. We regularly found that a large share of the students, especially in our introductory undergraduate econometrics courses, had not been exposed to any programming language before and thus had difficulty to engage with learning R on their own. With little background in statistics and econometrics, beginners naturally have a hard time understanding the benefits of having R skills for learning and applying econometrics. These particularly include the ability to conduct, document and communicate empirical studies and having the ability to program simulation studies which is helpful, e.g., for comprehending and validating theorems which usually are not easily grasped by mere brooding over formulas. Being applied economists and econometricians, we value and wish to share with our students all of these capabilities.

Instead of confronting students with pure coding exercises and complementary classic literature like the book by Venables and Smith (2010), we figured it would be better to provide interactive learning material that blends R code with the contents of the well-received textbook *Introduction to Econometrics* by Stock and Watson (2015) which serves as a basis for the lecture. This material is gathered in the present book *Introduction to Econometrics with R*, an empirical companion to Stock and Watson (2015). It is an interactive script in the style of a reproducible research report that not only enables students to learn how results of case studies can be replicated with R but also strengthens their ability in using the newly acquired skills in other empirical applications.

Conventions Used in this Book

- *Italic* text indicates new terms, names, buttons and the like.
- **Monospaced font** is generally used in paragraphs to refer to R code. This includes commands, variables, functions, data types, databases, and file names.
- **Monospaced font on gray background** indicates R code that can be typed literally by you. It may appear in paragraphs for better distinguishability among executable and non-executable code statements but it will mostly be encountered in shape of large blocks of R code. These blocks are referred to as *code chunks*.

Acknowledgement

We thank the *Stifterverband für die Deutsche Wissenschaft e.V.* and the *Ministry of Culture and Science of North Rhine-Westphalia* for their financial support. Also, we are grateful to Alexander Blasberg for proofreading and his effort in helping with programming the exercises. A special thanks goes to Achim Zeileis (University of Innsbruck) and Christian Kleiber (University of Basel) for their advice and

constructive criticism. Another thanks goes to Rebecca Arnold from the Münster University of Applied Sciences for several suggestions regarding the website design and for providing us with her nice designs for the book cover, logos and icons. We are also indebted to all past students of our introductory econometrics courses at the University of Duisburg-Essen for their feedback.

Chapter 1

Introduction

The interest in the freely available statistical programming language and software environment **R** (R Core Team, 2023) is soaring. By the time we wrote first drafts for this project, more than 11000 add-ons (many of them providing cutting-edge methods) were made available on the Comprehensive **R** Archive Network (CRAN), an extensive network of FTP servers around the world that store identical and up-to-date versions of **R** code and its documentation. **R** dominates over other (commercial) software for statistical computing in most fields of research in applied statistics. The benefits of it being freely available, open source and having a large and constantly growing community of users that contribute to CRAN render **R** more and more appealing for empirical economists and econometricians alike.

A striking advantage of using **R** in econometrics is that it enables students to explicitly document their analysis step-by-step such that it is easy to update and to expand. This allows to re-use code for similar applications with different data. Furthermore, **R** programs are fully reproducible, which makes it straightforward for others to comprehend and validate results.

Over the recent years, **R** has thus become an integral part of the curricula of econometrics classes we teach at the University of Duisburg-Essen. In some sense, learning to code is comparable to learning a foreign language and continuous practice is essential for the learning success. Needless to say, presenting bare **R** code on slides does not encourage the students to engage with hands-on experience on their own. This is why **R** is crucial. As for accompanying literature, there are some excellent books that deal with **R** and its applications to econometrics, e.g., Kleiber and Zeileis (2008a). However, such sources may be somewhat beyond the scope of undergraduate students in economics having little understanding of econometric methods and barely any experience in programming at all. Consequently, we started to compile a collection of reproducible reports for use in class. These reports provide guidance on how to implement selected applications from the textbook *Introduction to Econometrics* (Stock and Watson, 2015) which serves as a basis for the lecture and the accompanying tutorials. This process was facilitated considerably by `knitr` (Xie, 2023b) and **R** `markdown` (Allaire et al., 2023). In conjunction, both **R** packages provide powerful functionalities for dynamic report generation which allow to seamlessly combine pure text, LaTeX, **R** code and its output in a variety of formats, including PDF and HTML. Moreover, writing and distributing reproducible reports for use in academia has been enriched tremendously by the `bookdown` package (Xie, 2023a) which has become our main tool for this project. `bookdown` builds on top of **R** `markdown` and allows to create appealing HTML pages like this one, among other things. Being inspired by *Using R for Introductory Econometrics* (Heiss, 2016)¹ and with this powerful toolkit at hand we wrote up our own empirical companion to Stock and Watson (2015). The result, which you started to look at, is *Introduction to Econometrics with R*.

¹Heiss (2016) builds on the popular *Introductory Econometrics* (Wooldridge, 2016) and demonstrates how to replicate the applications discussed therein using **R**.

Similarly to the book by Heiss (2016), this project is neither a comprehensive econometrics textbook nor is it intended to be a general introduction to R. We feel that Stock and Watson did a great job at explaining the intuition and theory of econometrics, and at any rate better than we could in yet another introductory textbook! *Introduction to Econometrics with R* is best described as an interactive script in the style of a reproducible research report which aims to provide students with a platform-independent e-learning arrangement by seamlessly intertwining theoretical core knowledge and empirical skills in undergraduate econometrics. Of course, the focus is on empirical applications with R. We leave out derivations and proofs wherever we can. Our goal is to enable students not only to learn how results of case studies can be replicated with R but we also intend to strengthen their ability in using the newly acquired skills in other empirical applications — immediately within *Introduction to Econometrics with R*.

To realize this, each chapter contains interactive R programming exercises. These exercises are used as supplements to code chunks that display how previously discussed techniques can be implemented within R. They are generated using the DataCamp light widget and are backed by an R session which is maintained on DataCamp’s servers. You may play around with the example exercise presented below.

As you can see above, the widget consists of two tabs. `script.R` mimics an `.R`-file, a file format that is commonly used for storing R code. Lines starting with a `#` are commented out, that is, they are not recognized as code. Furthermore, `script.R` works like an exercise sheet where you may write down the solution you come up with. If you hit the button *Run*, the code will be executed, submission correctness tests are run and you will be notified whether your approach is correct. If it is not correct, you will receive feedback suggesting improvements or hints. The other tab, `R Console`, is a fully functional R console that can be used for trying out solutions to exercises before submitting them. Of course you may submit (almost any) R code and use the console to play around and explore. Simply type a command and hit the *Enter* key on your keyboard.

Looking at the widget above, you will notice that there is a `>` in the right panel (in the console). This symbol is called “prompt” and indicates that the user can enter code that will be executed. To avoid confusion, we will not show this symbol in this book. Output produced by R code is commented out with `#>`.

Most commonly we display R code together with the generated output in code chunks. As an example, consider the following line of code presented in chunk below. It tells R to compute the number of packages available on CRAN. The code chunk is followed by the output produced.

```
# check the number of R packages available on CRAN
nrow(available.packages(repos = "http://cran.us.r-project.org"))
#> [1] 20396
```

Each code chunk is equipped with a button on the outer right hand side which copies the code to your clipboard. This makes it convenient to work with larger code segments in your version of R/*RStudio* or in the widgets presented throughout the book. In the widget above, you may click on `R Console` and type `nrow(available.packages(repos = "http://cran.us.r-project.org"))` (the command from the code chunk above) and execute it by hitting *Enter* on your keyboard.²

Note that some lines in the widget are out-commented which ask you to assign a numeric value to a variable and then to print the variable’s content to the console. You may enter your solution approach to `script.R` and hit the button *Run* in order to get the feedback described further above. In case you do not know how to solve this sample exercise (don’t panic, that is probably why you are reading this), a click on *Hint* will provide you with some advice. If you still can’t find a solution, a click on *Solution* will provide you with another tab, `Solution.R` which contains sample solution code. It will often be

²The R session is initialized by clicking into the widget. This might take a few seconds. Just wait for the indicator next to the button *Run* to turn green.

the case that exercises can be solved in many different ways and `Solution.R` will present what we consider as comprehensible and idiomatic.

1.1 Colophon

This book was built with:

```
#> - Session info -----
#> setting value
#> version R version 4.3.1 (2023-06-16)
#> os      macOS Ventura 13.1
#> system  aarch64, darwin20
#> ui      X11
#> language (EN)
#> collate en_US.UTF-8
#> ctype   en_US.UTF-8
#> tz      Europe/Berlin
#> date    2024-02-13
#> pandoc  3.1.1 @ /Applications/RStudio.app/Contents/Resources/app/quarto/bin/tools/ (via rmarkdown)
#>
#> - Packages -----
#> package      * version      date (UTC) lib source
#> abind         1.4-5         2016-07-21 [1] CRAN (R 4.3.0)
#> AER          1.2-10        2022-06-14 [1] CRAN (R 4.3.0)
#> askpass      1.2.0         2023-09-03 [1] CRAN (R 4.3.0)
#> backports    1.4.1         2021-12-13 [1] CRAN (R 4.3.0)
#> base64enc    0.1-3         2015-07-28 [1] CRAN (R 4.3.0)
#> bdsmatrix    1.3-6         2022-06-03 [1] CRAN (R 4.3.0)
#> bit          4.0.5         2022-11-15 [1] CRAN (R 4.3.0)
#> bit64        4.0.5         2020-08-30 [1] CRAN (R 4.3.0)
#> bitops       1.0-7         2021-04-24 [1] CRAN (R 4.3.0)
#> blob         1.2.4         2023-03-17 [1] CRAN (R 4.3.0)
#> bookdown     0.36          2023-10-16 [1] CRAN (R 4.3.1)
#> boot         1.3-28.1      2022-11-22 [1] CRAN (R 4.3.1)
#> brio         1.1.3         2021-11-30 [1] CRAN (R 4.3.0)
#> broom        1.0.5         2023-06-09 [1] CRAN (R 4.3.0)
#> bslib        0.5.1         2023-08-11 [1] CRAN (R 4.3.0)
#> cachem       1.0.8         2023-05-01 [1] CRAN (R 4.3.0)
#> callr        3.7.3         2022-11-02 [1] CRAN (R 4.3.0)
#> car          3.1-2         2023-03-30 [1] CRAN (R 4.3.0)
#> carData      3.0-5         2022-01-06 [1] CRAN (R 4.3.0)
#> cellranger   1.1.0         2016-07-27 [1] CRAN (R 4.3.0)
#> cli          3.6.2         2023-12-11 [1] CRAN (R 4.3.1)
#> clipr        0.8.0         2022-02-22 [1] CRAN (R 4.3.0)
#> collapse     2.0.3         2023-10-17 [1] CRAN (R 4.3.1)
#> colorspace   2.1-0         2023-01-23 [1] CRAN (R 4.3.0)
#> conflicted   1.2.0         2023-02-01 [1] CRAN (R 4.3.0)
#> cpp11        0.4.7         2023-12-02 [1] CRAN (R 4.3.1)
#> crayon       1.5.2         2022-09-29 [1] CRAN (R 4.3.0)
#> cubature     2.1.0         2023-06-11 [1] CRAN (R 4.3.0)
#> curl         5.1.0         2023-10-02 [1] CRAN (R 4.3.1)
#> cvar         0.5           2022-11-03 [1] CRAN (R 4.3.0)
```

```

#> data.table      1.14.8      2023-02-17 [1] CRAN (R 4.3.0)
#> DBI              1.1.3      2022-06-18 [1] CRAN (R 4.3.0)
#> dbplyr           2.4.0      2023-10-26 [1] CRAN (R 4.3.1)
#> desc             1.4.3      2023-12-10 [1] CRAN (R 4.3.1)
#> diffobj          0.3.5      2021-10-05 [1] CRAN (R 4.3.0)
#> digest           0.6.34     2024-01-11 [1] CRAN (R 4.3.1)
#> dplyr            1.1.3      2023-09-03 [1] CRAN (R 4.3.0)
#> dtplyr           1.3.1      2023-03-22 [1] CRAN (R 4.3.0)
#> dynlm            0.3-6      2019-01-06 [1] CRAN (R 4.3.0)
#> ellipsis         0.3.2      2021-04-29 [1] CRAN (R 4.3.0)
#> evaluate         0.23       2023-11-01 [1] CRAN (R 4.3.1)
#> fansi            1.0.5      2023-10-08 [1] CRAN (R 4.3.1)
#> farver           2.1.1      2022-07-06 [1] CRAN (R 4.3.0)
#> fastICA          1.2-3      2021-09-25 [1] CRAN (R 4.3.0)
#> fastmap          1.1.1      2023-02-24 [1] CRAN (R 4.3.0)
#> fBasics          4032.96    2023-11-03 [1] CRAN (R 4.3.1)
#> fGarch           4031.90    2023-10-15 [1] CRAN (R 4.3.1)
#> fontawesome      0.5.2      2023-08-19 [1] CRAN (R 4.3.0)
#> forcats          1.0.0      2023-01-29 [1] CRAN (R 4.3.0)
#> Formula          1.2-5      2023-02-24 [1] CRAN (R 4.3.0)
#> fs                1.6.3      2023-07-20 [1] CRAN (R 4.3.0)
#> gargle           1.5.2      2023-07-20 [1] CRAN (R 4.3.0)
#> gbutils          0.5        2022-05-27 [1] CRAN (R 4.3.0)
#> generics         0.1.3      2022-07-05 [1] CRAN (R 4.3.0)
#> ggplot2          3.4.4      2023-10-12 [1] CRAN (R 4.3.1)
#> glue             1.7.0      2024-01-09 [1] CRAN (R 4.3.1)
#> googledrive      2.1.1      2023-06-11 [1] CRAN (R 4.3.0)
#> googlesheets4   1.1.1      2023-06-11 [1] CRAN (R 4.3.0)
#> gss              2.2-7      2023-08-16 [1] CRAN (R 4.3.0)
#> gtable           0.3.4      2023-08-21 [1] CRAN (R 4.3.0)
#> haven            2.5.3      2023-06-30 [1] CRAN (R 4.3.0)
#> highr            0.10       2022-12-22 [1] CRAN (R 4.3.0)
#> hms              1.1.3      2023-03-21 [1] CRAN (R 4.3.0)
#> htmltools        0.5.7      2023-11-03 [1] CRAN (R 4.3.1)
#> httr             1.4.7      2023-08-15 [1] CRAN (R 4.3.0)
#> ids              1.0.1      2017-05-31 [1] CRAN (R 4.3.0)
#> isoband          0.2.7      2022-12-20 [1] CRAN (R 4.3.0)
#> itewrpkg         1.0        2023-11-08 [1] local
#> jquerylib        0.1.4      2021-04-26 [1] CRAN (R 4.3.0)
#> jsonlite         1.8.8      2023-12-04 [1] CRAN (R 4.3.1)
#> KernSmooth       2.23-21    2023-05-03 [1] CRAN (R 4.3.1)
#> knitr            1.45       2023-10-30 [1] CRAN (R 4.3.1)
#> labeling         0.4.3      2023-08-29 [1] CRAN (R 4.3.0)
#> lattice          0.21-8     2023-04-05 [1] CRAN (R 4.3.1)
#> lifecycle        1.0.4      2023-11-07 [1] CRAN (R 4.3.1)
#> lme4             1.1-35     2023-11-03 [1] CRAN (R 4.3.1)
#> lmtest           0.9-40     2022-03-21 [1] CRAN (R 4.3.0)
#> locpol           0.8.0      2022-11-29 [1] CRAN (R 4.3.0)
#> lubridate        1.9.3      2023-09-27 [1] CRAN (R 4.3.1)
#> magrittr         2.0.3      2022-03-30 [1] CRAN (R 4.3.0)
#> MASS             7.3-60     2023-05-04 [1] CRAN (R 4.3.1)
#> Matrix           1.6-1.1    2023-09-18 [1] CRAN (R 4.3.1)

```

```

#> MatrixModels      0.5-2      2023-07-10 [1] CRAN (R 4.3.0)
#> maxLik             1.5-2      2021-07-26 [1] CRAN (R 4.3.0)
#> memoise            2.0.1      2021-11-26 [1] CRAN (R 4.3.0)
#> mgcv                1.8-42     2023-03-02 [1] CRAN (R 4.3.1)
#> mime                0.12       2021-09-28 [1] CRAN (R 4.3.0)
#> minqa              1.2.6       2023-09-11 [1] CRAN (R 4.3.0)
#> miscTools          0.6-28     2023-05-03 [1] CRAN (R 4.3.0)
#> modelr             0.1.11     2023-03-22 [1] CRAN (R 4.3.0)
#> munsell            0.5.0       2018-06-12 [1] CRAN (R 4.3.0)
#> mvtnorm            1.2-3       2023-08-25 [1] CRAN (R 4.3.0)
#> nlme                3.1-162    2023-01-31 [1] CRAN (R 4.3.1)
#> nloptr             2.0.3       2022-05-26 [1] CRAN (R 4.3.0)
#> nnet                7.3-19     2023-05-03 [1] CRAN (R 4.3.1)
#> np                  0.60-17   2023-03-13 [1] CRAN (R 4.3.0)
#> numDeriv           2016.8-1.1  2019-06-06 [1] CRAN (R 4.3.0)
#> openssl            2.1.1       2023-09-25 [1] CRAN (R 4.3.1)
#> orcutt              2.3         2018-09-27 [1] CRAN (R 4.3.0)
#> pbkrtest           0.5.2       2023-01-19 [1] CRAN (R 4.3.0)
#> pillar             1.9.0       2023-03-22 [1] CRAN (R 4.3.0)
#> pkgbuild           1.4.3       2023-12-10 [1] CRAN (R 4.3.1)
#> pkgconfig          2.0.3       2019-09-22 [1] CRAN (R 4.3.0)
#> pkgload            1.3.4       2024-01-16 [1] CRAN (R 4.3.1)
#> plm                 2.6-3       2023-04-09 [1] CRAN (R 4.3.0)
#> praise             1.0.0       2015-08-11 [1] CRAN (R 4.3.0)
#> prettyunits        1.2.0       2023-09-24 [1] CRAN (R 4.3.1)
#> processx           3.8.3       2023-12-10 [1] CRAN (R 4.3.1)
#> progress           1.2.2       2019-05-16 [1] CRAN (R 4.3.0)
#> ps                  1.7.6       2024-01-18 [1] CRAN (R 4.3.1)
#> purrr              1.0.2       2023-08-10 [1] CRAN (R 4.3.0)
#> quadprog           1.5-8       2019-11-20 [1] CRAN (R 4.3.0)
#> quantmod           0.4.25     2023-08-22 [1] CRAN (R 4.3.0)
#> quantreg           5.97        2023-08-19 [1] CRAN (R 4.3.0)
#> R6                  2.5.1       2021-08-19 [1] CRAN (R 4.3.0)
#> ragg                1.2.6       2023-10-10 [1] CRAN (R 4.3.1)
#> rappdirs           0.3.3       2021-01-31 [1] CRAN (R 4.3.0)
#> rbibutils          2.2.16     2023-10-25 [1] CRAN (R 4.3.1)
#> RColorBrewer       1.1-3       2022-04-03 [1] CRAN (R 4.3.0)
#> Rcpp                1.0.11     2023-07-06 [1] CRAN (R 4.3.0)
#> RcppEigen          0.3.3.9.4  2023-11-02 [1] CRAN (R 4.3.1)
#> RCurl              1.98-1.13  2023-11-02 [1] CRAN (R 4.3.1)
#> rdd                 0.57        2016-03-14 [1] CRAN (R 4.3.0)
#> rddtools           1.6.0       2022-01-10 [1] CRAN (R 4.3.0)
#> Rdpack             2.5         2023-08-21 [1] CRAN (R 4.3.0)
#> rdrobust           2.1.1       2022-12-05 [1] CRAN (R 4.3.0)
#> readr              2.1.4       2023-02-10 [1] CRAN (R 4.3.0)
#> readxl             1.4.3       2023-07-06 [1] CRAN (R 4.3.0)
#> rematch            2.0.0       2023-08-30 [1] CRAN (R 4.3.0)
#> rematch2           2.1.2       2020-05-01 [1] CRAN (R 4.3.0)
#> reprex             2.0.2       2022-08-17 [1] CRAN (R 4.3.0)
#> rlang              1.1.3       2024-01-10 [1] CRAN (R 4.3.1)
#> rmarkdown          2.25        2023-09-18 [1] CRAN (R 4.3.1)
#> rprojroot          2.0.4       2023-11-05 [1] CRAN (R 4.3.1)

```

```
#> rstudioapi      0.15.0      2023-07-07 [1] CRAN (R 4.3.0)
#> rvest            1.0.3      2022-08-19 [1] CRAN (R 4.3.0)
#> sandwich        3.0-2      2022-06-15 [1] CRAN (R 4.3.0)
#> sass            0.4.7      2023-07-15 [1] CRAN (R 4.3.0)
#> scales          1.2.1      2022-08-20 [1] CRAN (R 4.3.0)
#> selectr         0.4-2      2019-11-20 [1] CRAN (R 4.3.0)
#> SparseM         1.81       2021-02-18 [1] CRAN (R 4.3.0)
#> spatial         7.3-16     2023-01-23 [1] CRAN (R 4.3.1)
#> stabledist      0.7-1      2016-09-12 [1] CRAN (R 4.3.0)
#> stargazer       5.2.3      2022-03-04 [1] CRAN (R 4.3.1)
#> stringi         1.8.3      2023-12-11 [1] CRAN (R 4.3.1)
#> stringr         1.5.1      2023-11-14 [1] CRAN (R 4.3.1)
#> strucchange     1.5-3      2022-06-15 [1] CRAN (R 4.3.0)
#> survival        3.5-5      2023-03-12 [1] CRAN (R 4.3.1)
#> sys             3.4.2      2023-05-23 [1] CRAN (R 4.3.0)
#> systemfonts     1.0.5      2023-10-09 [1] CRAN (R 4.3.1)
#> testthat        3.2.0      2023-10-06 [1] CRAN (R 4.3.1)
#> textshaping     0.3.7      2023-10-09 [1] CRAN (R 4.3.1)
#> tibble          3.2.1      2023-03-20 [1] CRAN (R 4.3.0)
#> tidyr           1.3.0      2023-01-24 [1] CRAN (R 4.3.0)
#> tidyselect      1.2.0      2022-10-10 [1] CRAN (R 4.3.0)
#> tidyverse       2.0.0      2023-02-22 [1] CRAN (R 4.3.0)
#> timechange      0.2.0      2023-01-11 [1] CRAN (R 4.3.0)
#> timeDate        4022.108   2023-01-07 [1] CRAN (R 4.3.0)
#> timeSeries      4031.107   2023-08-26 [1] CRAN (R 4.3.0)
#> tinytex         0.48       2023-10-13 [1] CRAN (R 4.3.1)
#> TTR             0.24.3     2021-12-12 [1] CRAN (R 4.3.0)
#> tzdb            0.4.0      2023-05-12 [1] CRAN (R 4.3.0)
#> urca            1.3-3      2022-08-29 [1] CRAN (R 4.3.0)
#> utf8            1.2.4      2023-10-22 [1] CRAN (R 4.3.1)
#> uuid            1.1-1      2023-08-17 [1] CRAN (R 4.3.0)
#> vars            1.5-9      2023-03-22 [1] CRAN (R 4.3.0)
#> vctrs           0.6.5      2023-12-01 [1] CRAN (R 4.3.1)
#> viridisLite     0.4.2      2023-05-02 [1] CRAN (R 4.3.0)
#> vroom           1.6.4      2023-10-02 [1] CRAN (R 4.3.1)
#> waldo           0.5.2      2023-11-02 [1] CRAN (R 4.3.1)
#> withr           3.0.0      2024-01-16 [1] CRAN (R 4.3.1)
#> xfun            0.41       2023-11-01 [1] CRAN (R 4.3.1)
#> xml2            1.3.6      2023-12-04 [1] CRAN (R 4.3.1)
#> xts             0.13.1     2023-04-16 [1] CRAN (R 4.3.0)
#> yaml            2.3.8      2023-12-11 [1] CRAN (R 4.3.1)
#> zoo             1.8-12     2023-04-13 [1] CRAN (R 4.3.0)
#>
#> [1] /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library
#>
#> -----
```

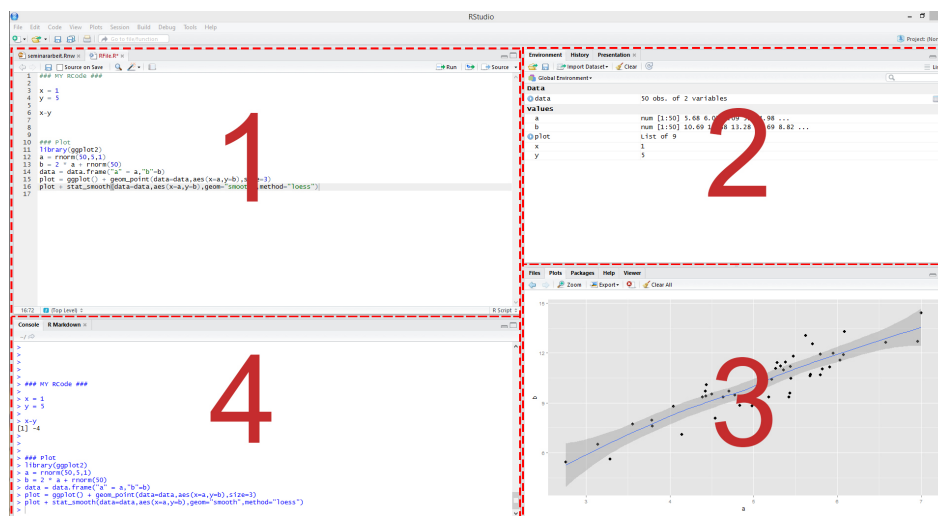


Figure 1.1: RStudio: the four panes

1.2 A Very Short Introduction to R and *RStudio*

R Basics

As mentioned before, this book is not intended to be an introduction to R but a guide on how to use its capabilities for applications commonly encountered in undergraduate econometrics. Those having basic knowledge in R programming may feel comfortable starting with Chapter 2. This section, however, is meant for those who have not worked with R or *RStudio* before. If you at least know how to create objects and call functions, you can skip it. If you would like to refresh your skills or get a feeling for how to work with *RStudio*, keep reading.

First of all, start *RStudio* and open a new R script by selecting *File, New File, R Script*. In the editor pane, type

```
1 + 1
```

and click on the button labeled *Run* in the top right corner of the editor. By doing so, your line of code is sent to the console and the result of this operation should be displayed right underneath it. As you can see, R works just like a calculator. You can do all arithmetic calculations by using the corresponding operator (+, -, *, / or ^). If you are not sure what the last operator does, try it out and check the results.

Vectors

In R, you can work with variables or more generally, objects. To define an object, you use the assignment operator <-, for example, to create a variable named *x* with the value 10, you can type *x <- 10* and then click the *Run* button. The new variable should appear in the environment pane on the top right. However, the console won't display any results because this line of code doesn't produce any visible output. If you want to see the value of *x*, you can simply type *x* in the console and press *Enter*, and R will display the corresponding value in the console.

x is a scalar, a vector of length 1. You can easily create longer vectors by using the function *c()* (*c* is for “concatenate” or “combine”). To create a vector *y* containing the numbers 1 to 5 and print it, do the following.

```
y <- c(1, 2, 3, 4, 5)
y
#> [1] 1 2 3 4 5
```

You can also create a vector of letters or words. For now just remember that characters have to be surrounded by quotes, else they will be parsed as object names.

```
hello <- c("Hello", "World")
```

Here we have created a vector of length 2 containing the words `Hello` and `World`.

Do not forget to save your script! To do so, select *File* and then *Save*.

Functions

You have seen that the function `c()` can be used to combine objects. In general, all function calls look the same: a function name is always followed by round parentheses. Often the parentheses include arguments.

Here are two simple examples.

```
# generate the vector `z`
z <- seq(from = 1, to = 5, by = 1)

# compute the mean of the entries in `z`
mean(z)
#> [1] 3
```

In the first line we use a function called `seq()` to create the exact same vector as we did in the previous section, calling it `z`. The function takes on the arguments `from`, `to` and `by` which should be self-explanatory. The function `mean()` computes the arithmetic mean of its argument `x`. Since we pass the vector `z` as the argument `x`, the result is 3!

If you are not sure which arguments a function expects, you may consult the function's documentation. Let's say we are not sure how the arguments required for `seq()` work. We then type `?seq` in the console. By hitting return, the documentation page for that function pops up in the lower right pane of *RStudio*. In there, the section *Arguments* holds the information we seek. On the bottom of almost every help page you find examples on how to use the corresponding functions. This is very helpful for beginners and we recommend to have a look.

Of course, all of the commands presented above also work in interactive widgets throughout the book. You may try them below.

This interactive application is only available in the HTML version.

Chapter 2

Probability Theory

This chapter reviews some basic concepts of probability theory and demonstrates how they can be applied in R.

Most of the statistical functionalities in base R are collected in the `stats` package. It provides simple functions which compute descriptive measures and facilitate computations involving a variety of probability distributions. It also contains more sophisticated routines that, enable the user to estimate a large number of models based on the same data or help to conduct extensive simulation studies. `stats` is part of the base distribution of R, meaning that it is installed by default, so there is no need to run `install.packages("stats")` or `library("stats")`. Simply execute `library(help = "stats")` in the console to view the documentation and a complete list of all functions gathered in `stats`. For most packages a documentation that can be viewed within *RStudio* is available. Documentation can be invoked using the `?` operator, for example, upon execution of `?stats` the documentation of the `stats` package is shown in the help tab of the bottom-right pane.

Subsequently, our focus is on (some of) the probability distributions handled by R and how to use the relevant functions to solve simple problems. Afterwards, we will review some core concepts of probability theory. Among other things, you will learn how to draw random numbers, how to compute densities, probabilities, quantiles and alike. As we shall see, it is very convenient to do these computations in R.

2.1 Random Variables and Probability Distributions

Let us briefly review some basic concepts of probability theory.

- The mutually exclusive results of a random process are called the *outcomes*. ‘Mutually exclusive’ means that only one of the possible outcomes can be observed.
- We refer to the *probability* of an outcome as the proportion that the outcome occurs in the long run, that is, if the experiment is repeated many times.
- The set of all possible outcomes of a random variable is called the *sample space*.
- An *event* is a subset of the sample space and consists of one or more outcomes.

These ideas are unified in the concept of a *random variable* which is a numerical summary of random outcomes. Random variables can be *discrete* or *continuous*.

- Discrete random variables have discrete outcomes, e.g., 0 and 1.
- A continuous random variable may take on a continuum of possible values.

Table 2.1: PDF and CDF of a Dice Roll

Outcome	1	2	3	4	5	6
Probability	1/6	1/6	1/6	1/6	1/6	1/6
Cumulative Probability	1/6	2/6	3/6	4/6	5/6	1

Probability Distributions of Discrete Random Variables

A typical example for a discrete random variable D is the result of a dice roll: in terms of a random experiment this is nothing but randomly selecting a sample of size 1 from a set of numbers which are mutually exclusive outcomes. Here, the sample space is $\{1, 2, 3, 4, 5, 6\}$ and we can think of many different events, e.g., ‘the observed outcome lies between 2 and 5’.

A basic function to draw random samples from a specified set of elements is the function `sample()`, see `?sample`. We can use it to simulate the random outcome of a dice roll. Let’s roll the dice!

```
sample(1:6, size=1)
#> [1] 5
```

The probability distribution of a discrete random variable is the list of all possible values of the variable and their probabilities that sum to 1. The cumulative probability distribution function gives the probability that the random variable is less than or equal to a particular value.

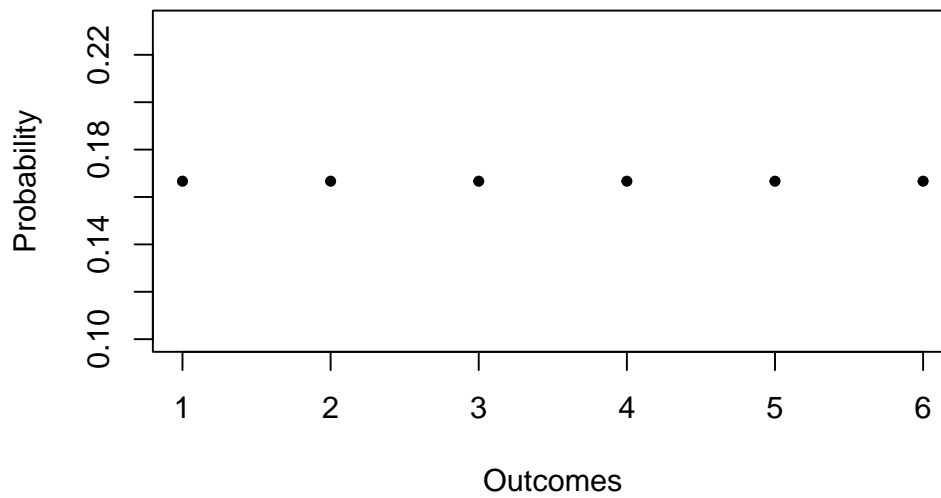
For the dice roll, the probability distribution and the cumulative probability distribution are summarized in Table 2.1.

We can easily plot both functions using R. Since the probability is equal to 1/6 for each outcome, we can set up the probability vector using the function `rep()`, which replicates a given value a specified number of times.

```
# generate the vector of probabilities
probability <- rep(1/6, 6)

# plot the probabilities
plot(probability,
     xlab = "Outcomes",
     ylab="Probability",
     main = "Probability Distribution",
     pch=20)
```

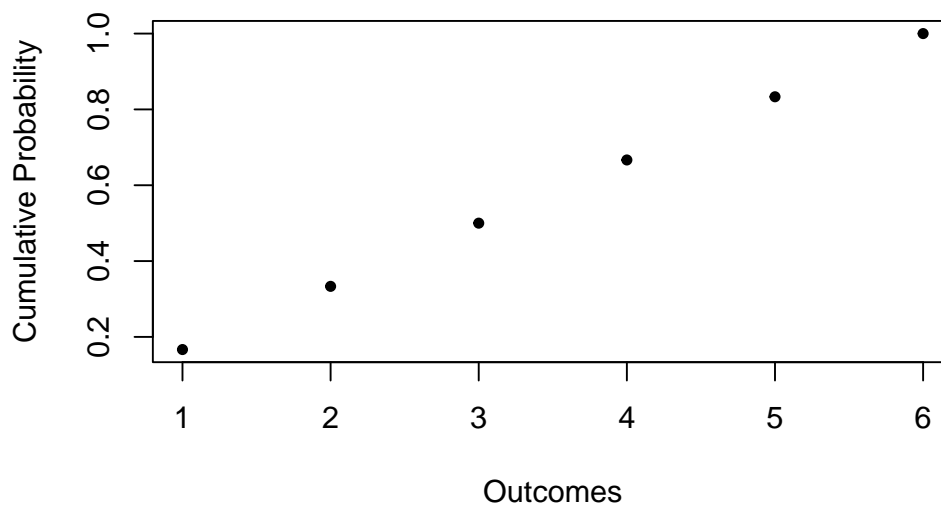

Probability Distribution



For the cumulative probability distribution we need the cumulative probabilities, i.e., we need the cumulative sums of the vector probability. These sums can be computed using `cumsum()`.

```
# generate the vector of cumulative probabilities  
cum_probability <- cumsum(probability)  
  
# plot the probabilities  
plot(cum_probability,  
      xlab = "Outcomes",  
      ylab="Cumulative Probability",  
      main = "Cumulative Probability Distribution",  
      pch=20)
```

Cumulative Probability Distribution



Bernoulli Trials

The set of elements from which `sample()` draws outcomes does not have to consist of numbers only. We might as well simulate coin tossing with outcomes H (heads) and T (tails).

```
sample(c("H", "T"), 1)
#> [1] "H"
```

The result of a single coin toss is a *Bernoulli* distributed random variable, i.e., a variable with two possible distinct outcomes.

Imagine you are about to toss a coin 10 times in a row and wonder how likely it is to end up with a 5 times heads. This is a typical example of what we call a *Bernoulli experiment* as it consists of $n = 10$ Bernoulli trials that are independent of each other and we are interested in the likelihood of observing $k = 5$ successes (H) that occur with probability $p = 0.5$ (assuming a fair coin) in each trial. Note that the order of the outcomes does not matter here.

It is a well known result that the number of successes k in a Bernoulli experiment follows a binomial distribution. We denote this as

$$k \sim B(n, p).$$

The probability of observing k successes in the experiment $B(n, p)$ is given by

$$f(k) = P(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} = \frac{n!}{k!(n-k)!} \cdot p^k \cdot (1-p)^{n-k},$$

as $\binom{n}{k}$ the binomial coefficient.

In R, we can solve problems like the one stated above by means of the function `dbinom()` which calculates $P(k|n, p)$ the probability of the binomial distribution given the parameters `x` (k), `size` (n), and `prob` (p), see `?dbinom`. Let us compute $P(k = 5|n = 10, p = 0.5)$ (we write this as $P(k = 5)$).

```
dbinom(x = 5,
       size = 10,
       prob = 0.5)
#> [1] 0.2460938
```

We conclude that $P(k = 5)$, the probability of observing Head $k = 5$ times when tossing a fair coin $n = 10$ times is about 24.6%.

Now assume we are interested in $P(4 \leq k \leq 7)$, i.e., the probability of observing 4, 5, 6 or 7 successes for $B(10, 0.5)$. This may be computed by providing a vector as the argument `x` in our call of `dbinom()` and summing up using `sum()`.

```
# compute P(4 <= k <= 7) using 'dbinom()'
sum(dbinom(x = 4:7, size = 10, prob = 0.5))
#> [1] 0.7734375
```

An alternative approach is to use `pbinom()`, the distribution function of the binomial distribution to compute

$$P(4 \leq k \leq 7) = P(k \leq 7) - P(k \leq 3).$$

```
# compute  $P(4 \leq k \leq 7)$  using 'pbinom()'
pbinom(size = 10, prob = 0.5, q = 7) - pbinom(size = 10, prob = 0.5, q = 3)
#> [1] 0.7734375
```

The probability distribution of a discrete random variable is nothing but a list of all possible outcomes that can occur and their respective probabilities. In the coin tossing example, we have 11 possible outcomes for k .

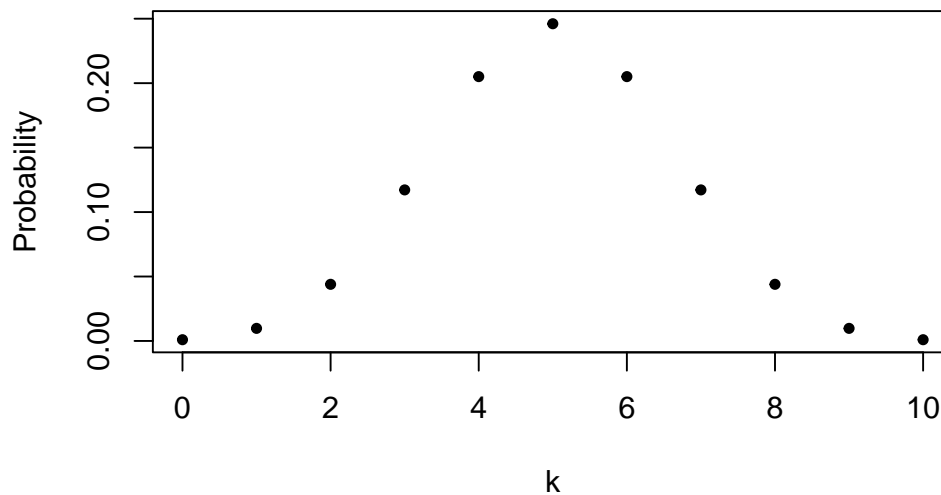
```
# set up vector of possible outcomes
k <- 0:10
k
#> [1] 0 1 2 3 4 5 6 7 8 9 10
```

To visualize the probability distribution function of k we may therefore do the following:

```
# assign the probabilities
probability <- dbinom(x = k,
                      size = 10,
                      prob = 0.5)

# plot the outcomes against their probabilities
plot(x = k,
     y = probability,
     ylab="Probability",
     main = "Probability Distribution Function",
     pch=20)
```

Probability Distribution Function

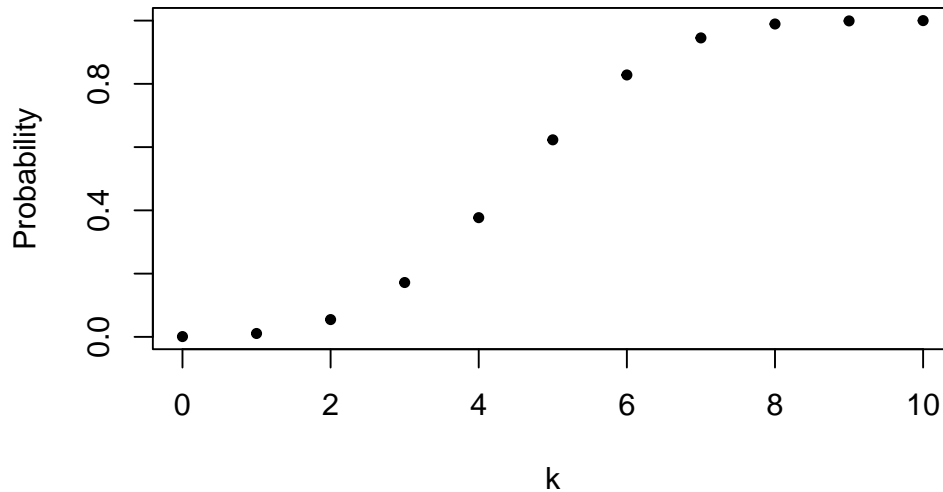


In a similar fashion we may plot the cumulative distribution function of k by executing the following code chunk:

```
# compute cumulative probabilities
prob <- pbinom(q = k,
              size = 10,
              prob = 0.5)
```

```
# plot the cumulative probabilities
plot(x = k,
     y = prob,
     ylab="Probability",
     main = "Cumulative Distribution Function",
     pch=20)
```

Cumulative Distribution Function



Expected Value, Mean and Variance

The expected value of a random variable is, loosely, the long-run average value of its outcomes when the number of repeated trials is large. For a discrete random variable, the expected value is computed as a weighted average of its possible outcomes whereby the weights are the related probabilities. This is formalized in Key Concept 2.1.

Key Concept 2.1 Expected Value and the Mean

Suppose the random variable Y takes on k possible values, y_1, \dots, y_k , where y_1 denotes the first value, y_2 denotes the second value, and so forth, and that the probability that Y takes on y_1 is p_1 , the probability that Y takes on y_2 is p_2 and so forth. The expected value of Y , $E(Y)$ is defined as

$$E(Y) = y_1p_1 + y_2p_2 + \dots + y_kp_k = \sum_{i=1}^k y_i p_i,$$

where the notation $\sum_{i=1}^k y_i p_i$ means the sum of $y_i p_i$ for i running from 1 to k : The expected value of Y is also called the mean of Y or the expectation of Y and is denoted by μ_Y .

In the dice example, the random variable, D say, takes on 6 possible values $d_1 = 1, d_2 = 2, \dots, d_6 = 6$. Assuming a fair die, each of the 6 outcomes occurs with a probability of $1/6$. It is therefore easy to calculate the exact value of $E(D)$ by hand:

$$E(D) = \frac{1}{6} \sum_{i=1}^6 d_i = 3.5.$$

$E(D)$ is simply the average of the natural numbers from 1 to 6 since all weights p_i are $1/6$. This can be easily calculated using the function `mean()` which computes the arithmetic mean of a numeric vector.

```
# compute mean of natural numbers from 1 to 6
mean(1:6)
#> [1] 3.5
```

An example of sampling with replacement is rolling a dice three times in a row.

```
# set seed for reproducibility
set.seed(1)

# rolling a dice three times in a row
sample(1:6, 3, replace = T)
#> [1] 1 4 1
```

Note that every call of `sample(1:6, 3, replace = T)` gives a different outcome since we draw with replacement at random. To allow you to reproduce the results of computations that involve random numbers, we will use `set.seed()` to set R's random number generator to a specific state. You should check that it actually works: set the seed in your R session to 1 and verify that you obtain the same three random numbers!

Sequences of random numbers generated by R are pseudo-random numbers, i.e., they are not “truly” random but approximate the properties of sequences of random numbers. Since this approximation is good enough for our purposes we refer to pseudo-random numbers as random numbers throughout this book.

In general, sequences of random numbers are generated by functions called “pseudo-random number generators” (PRNGs). The PRNG in R works by performing some operation on a deterministic value. Generally, this value is the previous number generated by the PRNG. However, the first time the PRNG is used, there is no previous value. A “seed” is the first value of a sequence of numbers — it initializes the sequence. Each seed value will correspond to a different sequence of values. In R a seed can be set using `set.seed()`.

This is convenient for us:

If we provide the same seed twice, we get the same sequence of numbers twice. Thus, setting a seed before executing R code which involves random numbers makes the outcome reproducible!

Of course, we could also consider a much bigger number of trials, let's say 10000. By doing so, it would be pointless to simply print the results to the console since, by default, R displays up to 1000 entries of large vectors and omits the remainder (give it a try). Eyeballing the numbers does not reveal much. Instead, let us calculate the sample average of the outcomes using `mean()` and see if the result comes close to the expected value $E(D) = 3.5$.

```
# set seed for reproducibility
set.seed(1)

# compute the sample mean of 10000 dice rolls
mean(sample(1:6,
           10000,
```

```
replace = T))
#> [1] 3.5138
```

We find the sample mean to be fairly close to the expected value. This result will be discussed in Chapter 2.2 in more detail.

Other frequently encountered measures are the variance and the standard deviation. Both are measures of the *dispersion* of a random variable.

Key Concept 2.2 Variance and Standard Deviation

The variance of the discrete *random variable* Y , denoted by σ_Y^2 , is

$$\sigma_Y^2 = \text{Var}(Y) = E[(Y - \mu_Y)^2] = \sum_{i=1}^k (y_i - \mu_Y)^2 p_i.$$

The standard deviation of Y is σ_Y , the square root of the variance. The units of the standard deviation are the same as the units of Y .

The variance as defined in Key Concept 2.2, being a population quantity, *is not* implemented as a function in R. Instead we have the function `var()` which computes the *sample variance*

$$s_Y^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2.$$

Remember that s_Y^2 is different from the so called *population variance* of a discrete random variable Y ,

$$\text{Var}(Y) = \frac{1}{N} \sum_{i=1}^N (y_i - \mu_Y)^2.$$

since it measures how the n observations in the sample are dispersed around the sample average \bar{y} . Instead, $\text{Var}(Y)$ measures the dispersion of the whole population (N members) around the population mean μ_Y . The difference becomes clear when we look at our dice rolling example. For D we have

$$\text{Var}(D) = \frac{1}{6} \sum_{i=1}^6 (d_i - 3.5)^2 = 2.92$$

which is obviously different from the result of s^2 as computed by `var()`.

```
var(1:6)
#> [1] 3.5
```

The sample variance as computed by `var()` is an *estimator* of the population variance. You may check this using the widget below.

Probability Distributions of Continuous Random Variables

Since a continuous random variable takes on a continuum of possible values, we cannot use the concept of a probability distribution as used for discrete random variables. Instead, the probability distribution of a continuous random variable is summarized by its *probability density function* (PDF).

The cumulative probability distribution function (CDF) for a continuous random variable is defined just as in the discrete case. Hence, the CDF of a continuous random variables states the probability that the random variable is less than or equal to a particular value.

For completeness, we present revisions of Key Concepts 2.1 and 2.2 for the continuous case.

Key Concept 2.3

Probabilities, Expected Value and Variance of a Continuous Random Variable

Let $f_Y(y)$ denote the probability density function of Y . The probability that Y falls between a and b where $a < b$ is

$$P(a \leq Y \leq b) = \int_a^b f_Y(y)dy.$$

We further have that $P(-\infty \leq Y \leq \infty) = 1$ and therefore $\int_{-\infty}^{\infty} f_Y(y)dy = 1$.

As for the discrete case, the expected value of Y is the probability weighted average of its values. Due to continuity, we use integrals instead of sums. The expected value of Y is defined as

$$E(Y) = \mu_Y = \int y f_Y(y)dy.$$

The variance is the expected value of $(Y - \mu_Y)^2$. We thus have

$$\text{Var}(Y) = \sigma_Y^2 = \int (y - \mu_Y)^2 f_Y(y)dy.$$

Let us discuss an example:

Consider the continuous random variable X with PDF

$$f_X(x) = \frac{3}{x^4}, x > 1.$$

- We can show analytically that the integral of $f_X(x)$ over the real line equals 1.

$$\int f_X(x)dx = \int_1^{\infty} \frac{3}{x^4}dx \tag{2.1}$$

$$= \lim_{t \rightarrow \infty} \int_1^t \frac{3}{x^4}dx \tag{2.2}$$

$$= \lim_{t \rightarrow \infty} -x^{-3}|_{x=1}^t \tag{2.3}$$

$$= - \left(\lim_{t \rightarrow \infty} \frac{1}{t^3} - 1 \right) \tag{2.4}$$

$$= 1 \tag{2.5}$$

- The expectation of X can be computed as follows:

$$E(X) = \int x \cdot f_X(x) dx = \int_1^\infty x \cdot \frac{3}{x^4} dx \quad (2.6)$$

$$= -\frac{3}{2} x^{-2} \Big|_{x=1}^\infty \quad (2.7)$$

$$= -\frac{3}{2} \left(\lim_{t \rightarrow \infty} \frac{1}{t^2} - 1 \right) \quad (2.8)$$

$$= \frac{3}{2} \quad (2.9)$$

- Note that the variance of X can be expressed as $\text{Var}(X) = E(X^2) - E(X)^2$. Since $E(X)$ has been computed in the previous step, we seek $E(X^2)$:

$$E(X^2) = \int x^2 \cdot f_X(x) dx = \int_1^\infty x^2 \cdot \frac{3}{x^4} dx \quad (2.10)$$

$$= -3x^{-1} \Big|_{x=1}^\infty \quad (2.11)$$

$$= -3 \left(\lim_{t \rightarrow \infty} \frac{1}{t} - 1 \right) \quad (2.12)$$

$$= 3 \quad (2.13)$$

So we have shown that the area under the curve equals one, that the expectation is $E(X) = \frac{3}{2}$ and we found the variance to be $\text{Var}(X) = \frac{3}{4}$. However, this was tedious and, as we shall see, an analytic approach is not applicable for some PDFs, e.g., if integrals have no closed form solutions.

Luckily, R also enables us to easily find the results derived above. The tool we use for this is the function `integrate()`. First, we have to define the functions we want to calculate integrals for as R functions, i.e., the PDF $f_X(x)$ as well as the expressions $x \cdot f_X(x)$ and $x^2 \cdot f_X(x)$.

```
# define functions
f <- function(x) 3 / x^4
g <- function(x) x * f(x)
h <- function(x) x^2 * f(x)
```

Next, we use `integrate()` and set lower and upper limits of integration to 1 and ∞ using arguments `lower` and `upper`. By default, `integrate()` prints the result along with an estimate of the approximation error to the console. However, the outcome is not a numeric value one can readily do further calculation with. In order to get only a numeric value of the integral, we need to use the `$` operator in conjunction with the `value`. The `$` operator is used to extract elements by name from an object of type `list`.

```
# compute area under the density curve
area <- integrate(f,
                  lower = 1,
                  upper = Inf)$value
area
#> [1] 1

# compute E(X)
EX <- integrate(g,
                lower = 1,
                upper = Inf)$value
EX
```



```
#> [1] 1.5

# compute Var(X)
VarX <- integrate(h,
                  lower = 1,
                  upper = Inf)$value - EX^2

VarX
#> [1] 0.75
```

Although there is a wide variety of distributions, the ones most often encountered in econometrics are the normal, chi-squared, Student t and F distributions. Therefore we will discuss some core R functions that allow us to do calculations involving densities, probabilities and quantiles of these distributions.

Every probability distribution that R handles has four basic functions whose names consist of a prefix followed by a root name. As an example, take the normal distribution. The root name of all four functions associated with the normal distribution is `norm`. The four prefixes are

- `d` for “density” - probability function / probability density function
- `p` for “probability” - cumulative distribution function
- `q` for “quantile” - quantile function (inverse cumulative distribution function)
- `r` for “random” - random number generator

Thus, for the normal distribution we have the R functions `dnorm()`, `pnorm()`, `qnorm()` and `rnorm()`.

The Normal Distribution

Perhaps the most important probability distribution considered here is the normal distribution. This is not least due to the special role of the standard normal distribution and the Central Limit Theorem which is to be treated shortly. Normal distributions are symmetric and bell-shaped. A normal distribution is characterized by its mean μ and its standard deviation σ , concisely expressed by $\mathcal{N}(\mu, \sigma^2)$. The normal distribution has the PDF

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp -(x - \mu)^2 / (2\sigma^2). \quad (2.14)$$

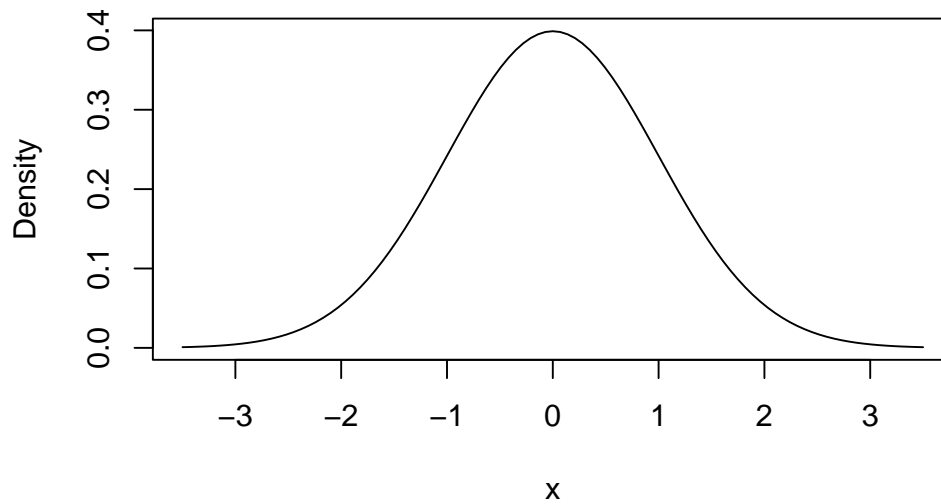
For the standard normal distribution we have $\mu = 0$ and $\sigma = 1$. Standard normal variates are often denoted by Z . Usually, the standard normal PDF is denoted by ϕ and the standard normal CDF is denoted by Φ . Hence,

$$\phi(c) = \Phi'(c) \quad , \quad \Phi(c) = P(Z \leq c) \quad , \quad Z \sim \mathcal{N}(0, 1).$$

Note that the notation $X \sim Y$ reads as “X is distributed as Y”. In R, we can conveniently obtain densities of normal distributions using the function `dnorm()`. Let us draw a plot of the standard normal density function using `curve()` together with `dnorm()`.

```
# draw a plot of the N(0,1) PDF
curve(dnorm(x),
      xlim = c(-3.5, 3.5),
      ylab = "Density",
      main = "Standard Normal Density Function")
```

Standard Normal Density Function



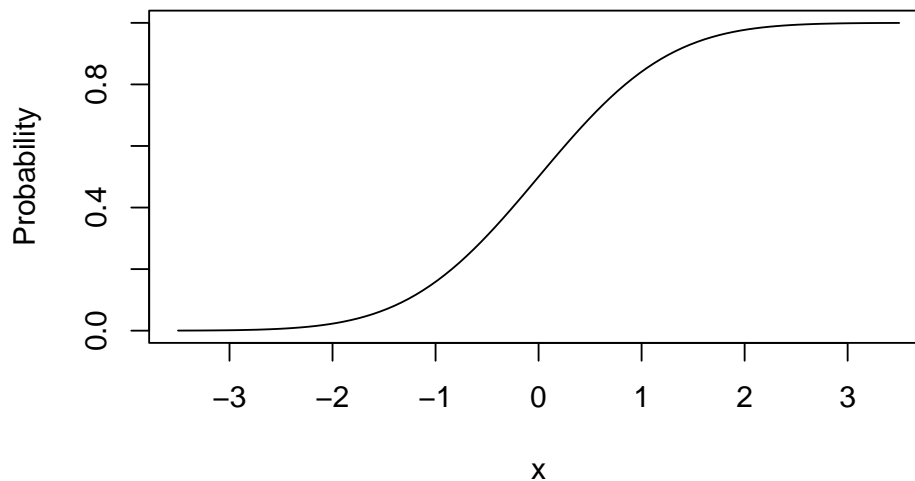
We can obtain the density at different positions by passing a vector to `dnorm()`.

```
# compute density at x=-1.96, x=0 and x=1.96
dnorm(x = c(-1.96, 0, 1.96))
#> [1] 0.05844094 0.39894228 0.05844094
```

Similar to the PDF, we can plot the standard normal CDF using `curve()`. We can use `dnorm()` for this but it is much more convenient to rely on `pnorm()`.

```
# plot the standard normal CDF
curve(pnorm(x),
      xlim = c(-3.5, 3.5),
      ylab = "Probability",
      main = "Standard Normal Cumulative Distribution Function")
```

Standard Normal Cumulative Distribution Function



We can also use R to calculate the probability of events associated with a standard normal variate.

Let us say we are interested in $P(Z \leq 1.337)$. For some continuous random variable Z on $[-\infty, \infty]$ with density $g(x)$ we would have to determine $G(x)$, the anti-derivative of $g(x)$ so that

$$P(Z \leq 1.337) = G(1.337) = \int_{-\infty}^{1.337} g(x)dx.$$

If $Z \sim \mathcal{N}(0, 1)$, we have $g(x) = \phi(x)$. There is no analytic solution to the integral above. Fortunately, R offers good approximations. The first approach makes use of the function `integrate()` which allows to solve one-dimensional integration problems using a numerical method. For this, we first define the function we want to compute the integral of as an R function `f`. In our example, `f` is the standard normal density function and hence takes a single argument `x`. Following the definition of $\phi(x)$ we define `f` as

```
# define the standard normal PDF as an R function
f <- function(x) {
  1/(sqrt(2 * pi)) * exp(-0.5 * x^2)
}
```

Let us check if this function computes standard normal densities by passing a vector.

```
# define a vector of reals
quants <- c(-1.96, 0, 1.96)

# compute densities
f(quants)
#> [1] 0.05844094 0.39894228 0.05844094

# compare to the results produced by 'dnorm()'
f(quants) == dnorm(quants)
#> [1] TRUE TRUE TRUE
```

The results produced by `f()` are indeed equivalent to those given by `dnorm()`.

Next, we call `integrate()` on `f()` and specify the arguments `lower` and `upper`, the lower and upper limits of integration.

```
# integrate f()
integrate(f,
  lower = -Inf,
  upper = 1.337)
#> 0.9093887 with absolute error < 1.7e-07
```

We find that the probability of observing $Z \leq 1.337$ is about 90.94%.

A second and much more convenient way is to use the function `pnorm()`, the standard normal cumulative distribution function.

```
# compute the probability using pnorm()
pnorm(1.337)
#> [1] 0.9093887
```

The result matches the outcome of the approach using `integrate()`.

Let us discuss some further examples:

A commonly known result is that 95% probability mass of a standard normal distribution lies in the interval $[-1.96, 1.96]$, that is, in a distance of about 2 standard deviations to the mean. We can easily

confirm this by calculating

$$P(-1.96 \leq Z \leq 1.96) = 1 - 2 \times P(Z \leq -1.96),$$

due to symmetry of the standard normal PDF. Thanks to **R**, we can abandon the table of the standard normal CDF found in many other textbooks and instead solve this fast by using `pnorm()`.

```
# compute the probability
1 - 2 * (pnorm(-1.96))
#> [1] 0.9500042
```

To make statements about the probability of observing outcomes of Y in some specific range it is more convenient when we standardize first as shown in Key Concept 2.4.

Key Concept 2.4 Computing Probabilities Involving Normal Random Variables

Suppose Y is normally distributed with mean μ and variance σ^2 :

$$Y \sim \mathcal{N}(\mu, \sigma^2)$$

Then Y is standardized by subtracting it from its mean and then dividing it by its standard deviation:

$$Z = \frac{Y - \mu}{\sigma}.$$

Let c_1 and c_2 denote two numbers whereby $c_1 < c_2$ and further $d_1 = (c_1 - \mu)/\sigma$ and $d_2 = (c_2 - \mu)/\sigma$. Then

$$\begin{aligned} P(Y \leq c_2) &= P(Z \leq d_2) = \Phi(d_2), \\ P(Y \geq c_1) &= P(Z \geq d_1) = 1 - \Phi(d_1), \\ P(c_1 \leq Y \leq c_2) &= P(d_1 \leq Z \leq d_2) = \Phi(d_2) - \Phi(d_1). \end{aligned}$$

Now consider a random variable Y with $Y \sim \mathcal{N}(5, 25)$. **R** functions that handle the normal distribution can perform the standardization. If we are interested in $P(3 \leq Y \leq 4)$ we can use `pnorm()` and adjust for a mean and/or a standard deviation that deviate from $\mu = 0$ and $\sigma = 1$ by specifying the arguments `mean` and `sd` accordingly. **Attention:** the argument `sd` requires the standard deviation, not the variance!

```
pnorm(4, mean = 5, sd = 5) - pnorm(3, mean = 5, sd = 5)
#> [1] 0.07616203
```

An extension of the normal distribution in a univariate setting is the multivariate normal distribution. The joint PDF of two random normal variables X and Y is given by

$$\begin{aligned} g_{X,Y}(x,y) &= \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho_{XY}^2}} \\ &\cdot \exp \left\{ \frac{1}{-2(1-\rho_{XY}^2)} \left[\left(\frac{x-\mu_X}{\sigma_X} \right)^2 - 2\rho_{XY} \left(\frac{x-\mu_X}{\sigma_X} \right) \left(\frac{y-\mu_Y}{\sigma_Y} \right) + \left(\frac{y-\mu_Y}{\sigma_Y} \right)^2 \right] \right\}. \end{aligned} \quad (2.15)$$

Equation (2.15) contains the bivariate normal PDF. It is somewhat hard to gain insights from this complicated expression. Instead, let us consider the special case where X and Y are uncorrelated

standard normal random variables with densities $f_X(x)$ and $f_Y(y)$ with joint normal distribution. We then have the parameters $\sigma_X = \sigma_Y = 1$, $\mu_X = \mu_Y = 0$ (due to marginal standard normality) and $\rho_{XY} = 0$ (due to independence). The joint density of X and Y then becomes

$$g_{X,Y}(x, y) = f_X(x)f_Y(y) = \frac{1}{2\pi} \cdot \exp\left\{-\frac{1}{2}[x^2 + y^2]\right\}, \quad (2.2)$$

the PDF of the bivariate standard normal distribution. The widget below provides an interactive three-dimensional plot of (2.2).

This interactive part of the book is only available in the HTML version.

By moving the cursor over the plot you can see that the density is rotationally invariant, i.e., the density at (a, b) solely depends on the distance of (a, b) to the origin: geometrically, regions of equal density are edges of concentric circles in the XY -plane, centered at $(\mu_X = 0, \mu_Y = 0)$.

The normal distribution has some remarkable characteristics. For example, for two jointly normally distributed variables X and Y , the conditional expectation function is linear: one can show that

$$E(Y|X) = E(Y) + \rho \frac{\sigma_Y}{\sigma_X}(X - E(X)).$$

The interactive widget below shows standard bivariate normally distributed sample data along with the conditional expectation function $E(Y|X)$ and the marginal densities of X and Y . All elements adjust accordingly as you vary the parameters.

This interactive part of the book is only available in the HTML version.

The Chi-Squared Distribution

The chi-squared distribution is another distribution relevant in econometrics. It is often needed when testing special types of hypotheses frequently encountered when dealing with regression models.

The sum of M squared independent standard normal distributed random variables follows a chi-squared distribution with M degrees of freedom:

$$Z_1^2 + \dots + Z_M^2 = \sum_{m=1}^M Z_m^2 \sim \chi_M^2 \quad \text{with } Z_m \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1).$$

A χ^2 distributed random variable with M degrees of freedom has expectation M , mode at $M - 2$ for $M \geq 2$ and variance $2 \cdot M$. For example, for

$$Z_1, Z_2, Z_3 \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1),$$

it holds that

$$Z_1^2 + Z_2^2 + Z_3^2 \sim \chi_3^2. \quad (2.3)$$

Using the code below, we can display the PDF and the CDF of a χ_3^2 random variable in a single plot. This is achieved by setting the argument `add = TRUE` in the second call of `curve()`. Further we adjust limits of both axes using `xlim` and `ylim` and choose different colors to make both functions better distinguishable. The plot is completed by adding a legend with help of `legend()`.

```

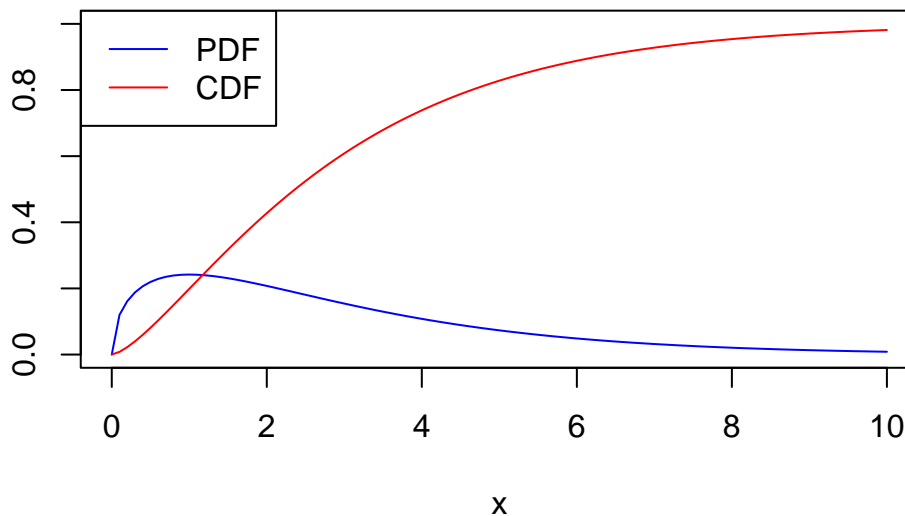
# plot the PDF
curve(dchisq(x, df = 3),
      xlim = c(0, 10),
      ylim = c(0, 1),
      col = "blue",
      ylab = "",
      main = "p.d.f. and c.d.f of Chi-Squared Distribution, M = 3")

# add the CDF to the plot
curve(pchisq(x, df = 3),
      xlim = c(0, 10),
      add = TRUE,
      col = "red")

# add a legend to the plot
legend("topleft",
      c("PDF", "CDF"),
      col = c("blue", "red"),
      lty = c(1, 1))

```

p.d.f. and c.d.f of Chi-Squared Distribution, M = 3



Since the outcomes of a χ_M^2 distributed random variable are always positive, the support of the related PDF and CDF is $\mathbb{R}_{\geq 0}$.

As expectation and variance depend (solely!) on the degrees of freedom, the distribution's shape changes drastically if we vary the number of squared standard normals that are summed up. This relation is often depicted by overlaying densities for different M , see the Wikipedia Article.

We reproduce this here by plotting the density of the χ_1^2 distribution on the interval $[0, 15]$ with `curve()`. In the next step, we loop over degrees of freedom $M = 2, \dots, 7$ and add a density curve for each M to the plot. We also adjust the line color for each iteration of the loop by setting `col = M`. At last, we add a legend that displays degrees of freedom and the associated colors.

```

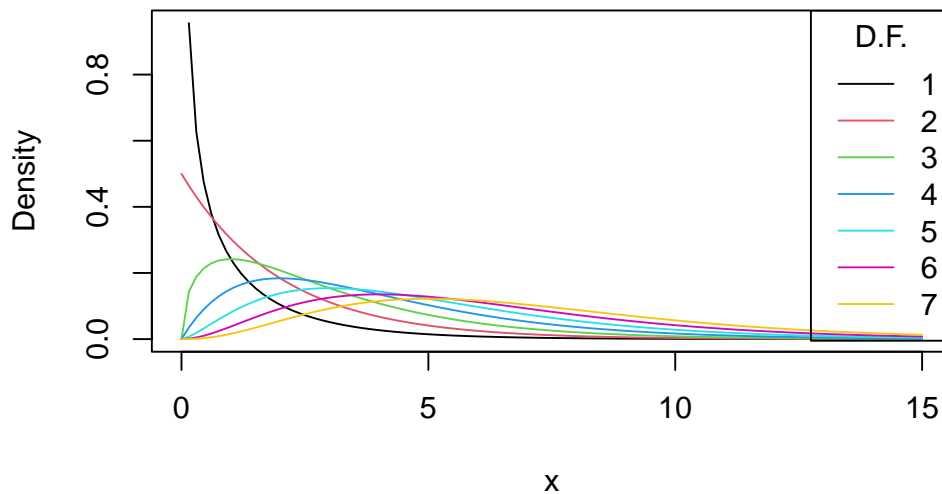
# plot the density for M=1
curve(dchisq(x, df = 1),
      xlim = c(0, 15),
      xlab = "x",
      ylab = "Density",
      main = "Chi-Square Distributed Random Variables")

# add densities for M=2,...,7 to the plot using a 'for()' loop
for (M in 2:7) {
  curve(dchisq(x, df = M),
        xlim = c(0, 15),
        add = T,
        col = M)
}

# add a legend
legend("topright",
      as.character(1:7),
      col = 1:7,
      lty = 1,
      title = "D.F.")

```

Chi-Square Distributed Random Variables



Increasing the degrees of freedom shifts the distribution to the right (the mode becomes larger) and increases the dispersion (the distribution's variance grows).

The Student t Distribution

Let Z be a standard normal variate, W a χ_M^2 random variable and further assume that Z and W are independent. Then it holds that

$$\frac{Z}{\sqrt{W/M}} =: X \sim t_M$$

and X follows a *Student t distribution* (or simply t distribution) with M degrees of freedom.

Similar to the χ_M^2 distribution, the shape of a t_M distribution depends on M . t distributions are symmetric, bell-shaped and look similar to a normal distribution, especially when M is large. This is not a coincidence: for a sufficiently large M , the t_M distribution can be approximated by the standard normal distribution. This approximation works reasonably well for $M \geq 30$. As we will illustrate later by means of a small simulation study, the t_∞ distribution *is* the standard normal distribution.

A t_M distributed random variable X has an expectation if $M > 1$ and it has a variance if $M > 2$.

$$E(X) = 0, \quad M > 1 \quad (2.16)$$

$$\text{Var}(X) = \frac{M}{M-2}, \quad M > 2 \quad (2.17)$$

Let us plot some t distributions with different M and compare them to the standard normal distribution.

```
# plot the standard normal density
curve(dnorm(x),
      xlim = c(-4, 4),
      xlab = "x",
      lty = 2,
      ylab = "Density",
      main = "Densities of t Distributions")

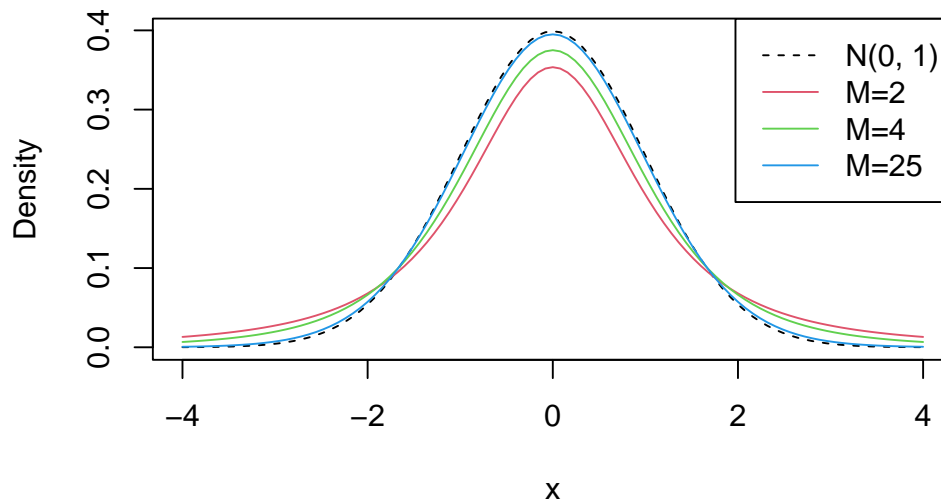
# plot the t density for M=2
curve(dt(x, df = 2),
      xlim = c(-4, 4),
      col = 2,
      add = T)

# plot the t density for M=4
curve(dt(x, df = 4),
      xlim = c(-4, 4),
      col = 3,
      add = T)

# plot the t density for M=25
curve(dt(x, df = 25),
      xlim = c(-4, 4),
      col = 4,
      add = T)

# add a legend
legend("topright",
      c("N(0, 1)", "M=2", "M=4", "M=25"),
      col = 1:4,
      lty = c(2, 1, 1, 1))
```


Densities of t Distributions



The plot illustrates what has been said in the previous paragraph: as the degrees of freedom increase, the shape of the t distribution comes closer to that of a standard normal bell curve. Already for $M = 25$, we find little difference to the standard normal density. If M is small, we find the distribution to have heavier tails than a standard normal, i.e., it has a “fatter” bell shape.

The F Distribution

Another ratio of random variables important to econometricians is the ratio of two independent χ^2 distributed random variables that are divided by their degrees of freedom M and n . The quantity

$$\frac{W/M}{V/n} \sim F_{M,n} \text{ with } W \sim \chi_M^2, \quad V \sim \chi_n^2,$$

follows an F distribution with numerator degrees of freedom M and denominator degrees of freedom n , denoted $F_{M,n}$. The distribution was first derived by George Snedecor but was named in honor of Sir Ronald Fisher.

By definition, the support of both PDF and CDF of an $F_{M,n}$ distributed random variable is $\mathbb{R}_{\geq 0}$.

Say we have an F distributed random variable Y with numerator degrees of freedom 3 and denominator degrees of freedom 14 and are interested in $P(Y \geq 2)$. This can be computed with the help of the function `pf()`. By setting the argument `lower.tail = FALSE` we ensure that R computes $1 - P(Y \leq 2)$, i.e., the probability mass in the tail right of 2.

```
pf(2, df1 = 3, df2 = 14, lower.tail = F)
#> [1] 0.1603538
```

We can visualize this probability by drawing a line plot of the related density and adding a color shading with `polygon()`.

```
# define coordinate vectors for vertices of the polygon
x <- c(2, seq(2, 10, 0.01), 10)
y <- c(0, df(seq(2, 10, 0.01), 3, 14), 0)

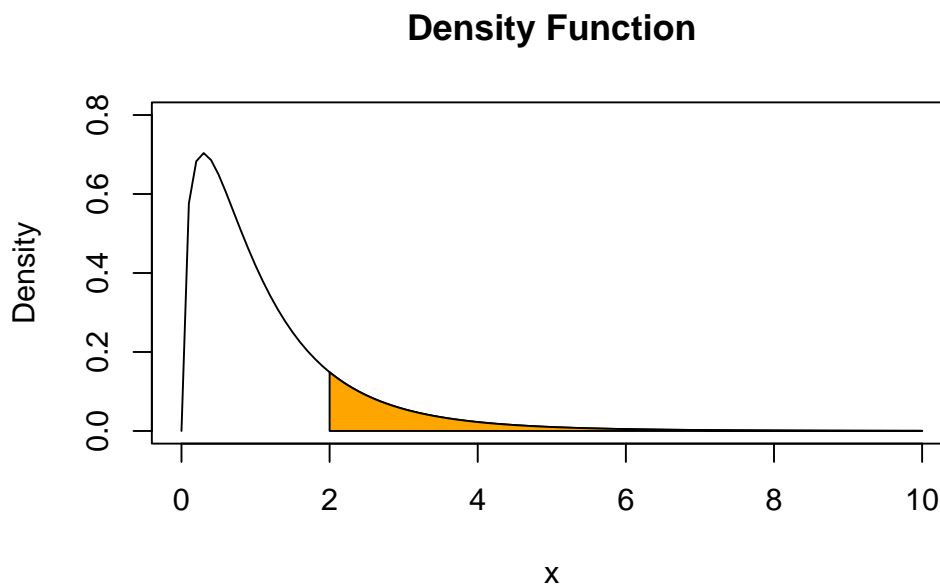
# draw density of F_{3, 14}
curve(df(x, 3, 14),
```

```

ylim = c(0, 0.8),
xlim = c(0, 10),
ylab = "Density",
main = "Density Function")

# draw the polygon
polygon(x, y, col = "orange")

```



The F distribution is related to many other distributions. An important special case encountered in econometrics arises if the denominator degrees of freedom are large such that the $F_{M,n}$ distribution can be approximated by the $F_{M,\infty}$ distribution which turns out to be simply the distribution of a χ_M^2 random variable divided by its degrees of freedom M ,

$$W/M \sim F_{M,\infty} \quad , \quad W \sim \chi_M^2.$$

2.2 Random Sampling and the Distribution of Sample Averages

To clarify the basic idea of random sampling, let us jump back to the dice rolling example:

Suppose we are rolling the dice n times. This means we are interested in the outcomes of random Y_i , $i = 1, \dots, n$ which are characterized by the same distribution. Since these outcomes are selected randomly, they are *random variables* themselves and their realizations will differ each time we draw a sample, i.e., each time we roll the dice n times. Furthermore, each observation is randomly drawn from the same population, that is, the numbers from 1 to 6, and their individual distribution is the same. Hence Y_1, \dots, Y_n are identically distributed.

Moreover, we know that the value of any of the Y_i does not provide any information on the remainder of the outcomes. In our example, rolling a six as the first observation in our sample does not alter the distributions of Y_2, \dots, Y_n : all numbers are equally likely to occur. This means that all Y_i are also independently distributed. Thus Y_1, \dots, Y_n are independently and identically distributed (*i.i.d.*). The dice example uses this most simple sampling scheme. That is why it is called *simple random sampling*. This concept is summarized in Key Concept 2.5.

Key Concept 2.5 Simple Random Sampling and i.i.d. Random Variables

In simple random sampling, n objects are drawn at random from a population. Each object is equally likely to end up in the sample. We denote the value of the random variable Y for the i^{th} randomly drawn object as Y_i . Since all objects are equally likely to be drawn and the distribution of Y_i is the same for all i , the Y_1, \dots, Y_n are independently and identically distributed (i.i.d.). This means the distribution of Y_i is the same for all $i = 1, \dots, n$ and Y_1 is distributed independently of Y_2, \dots, Y_n and Y_2 is distributed independently of Y_1, Y_3, \dots, Y_n and so forth.

What happens if we consider functions of the sample data? Consider the example of rolling a dice two times in a row once again. A sample now consists of two independent random draws from the set $\{1, 2, 3, 4, 5, 6\}$. It is apparent that any function of these two random variables, e.g. their sum, is also random. Convince yourself by executing the code below several times.

```
sum(sample(1:6, 2, replace = T))
#> [1] 7
```

Clearly, this sum, let us call it S , is a random variable as it depends on randomly drawn summands. For this example, we can completely enumerate all outcomes and hence write down the theoretical probability distribution of our function of the sample data S :

We face $6^2 = 36$ possible pairs. Those pairs are

```
(1, 1)(1, 2)(1, 3)(1, 4)(1, 5)(1, 6)
(2, 1)(2, 2)(2, 3)(2, 4)(2, 5)(2, 6)
(3, 1)(3, 2)(3, 3)(3, 4)(3, 5)(3, 6)
(4, 1)(4, 2)(4, 3)(4, 4)(4, 5)(4, 6)
(5, 1)(5, 2)(5, 3)(5, 4)(5, 5)(5, 6)
(6, 1)(6, 2)(6, 3)(6, 4)(6, 5)(6, 6)
```

Thus, possible outcomes for S are

```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}.
```

Enumeration of outcomes yields

$$P(S) = \begin{cases} 1/36, & S = 2 \\ 2/36, & S = 3 \\ 3/36, & S = 4 \\ 4/36, & S = 5 \\ 5/36, & S = 6 \\ 6/36, & S = 7 \\ 5/36, & S = 8 \\ 4/36, & S = 9 \\ 3/36, & S = 10 \\ 2/36, & S = 11 \\ 1/36, & S = 12 \end{cases} \quad (2.18)$$

We can also compute $E(S)$ and $\text{Var}(S)$ as stated in Key Concept 2.1 and Key Concept 2.2.

```
# Vector of outcomes
S <- 2:12

# Vector of probabilities
PS <- c(1:6, 5:1) / 36

# Expectation of S
ES <- sum(S * PS)
ES
#> [1] 7

# Variance of S
VarS <- sum((S - c(ES))^2 * PS)
VarS
#> [1] 5.833333
```

So the distribution of S is known. It is also evident that its distribution differs considerably from the marginal distribution, i.e., the distribution of a single dice roll's outcome, D . Let us visualize this using bar plots.

```
# divide the plotting area into one row with two columns
par(mfrow = c(1, 2), cex.main=1)

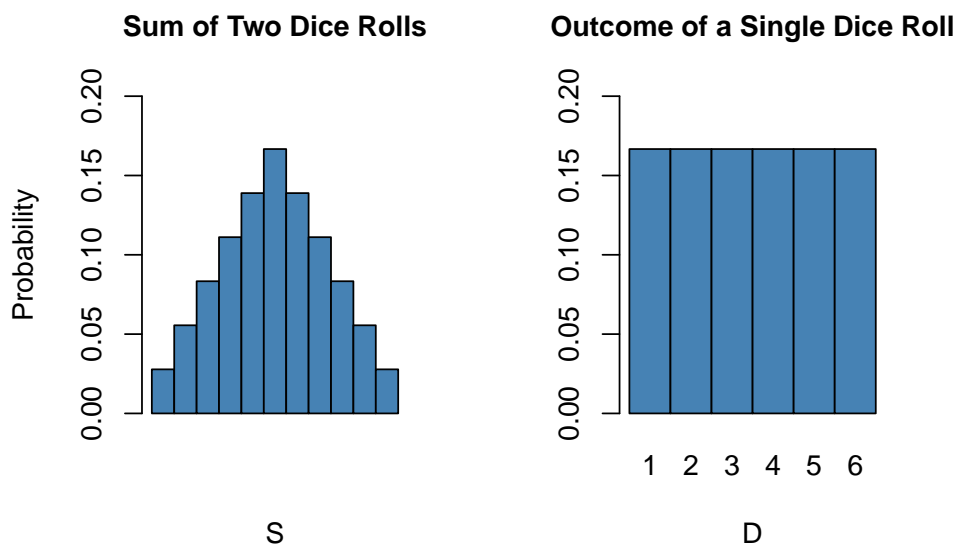
# plot the distribution of S
barplot(PS,
        ylim = c(0, 0.2),
        xlab = "S",
        ylab = "Probability",
        col = "steelblue",
        space = 0,
        main = "Sum of Two Dice Rolls")

# plot the distribution of D
probability <- rep(1/6, 6)
names(probability) <- 1:6
```

```

barplot(probability,
        ylim = c(0, 0.2),
        xlab = "D",
        col = "steelblue",
        space = 0,
        main = "Outcome of a Single Dice Roll")

```



Many econometric procedures deal with averages of sampled data. It is typically assumed that observations are drawn randomly from a larger, unknown population. As demonstrated for the sample function S , computing an average of a random sample has the effect that the average is a random variable itself. This random variable in turn has a probability distribution, called the sampling distribution. Knowledge about the sampling distribution of the average is therefore crucial for understanding the performance of econometric procedures.

The *sample average* of a sample of n observations Y_1, \dots, Y_n is

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i = \frac{1}{n} (Y_1 + Y_2 + \dots + Y_n).$$

\bar{Y} is also called the sample mean.

Mean and Variance of the Sample Mean

Suppose that Y_1, \dots, Y_n are i.i.d. and denote μ_Y and σ_Y^2 as the mean and the variance of the Y_i . Then we have that

$$E(\bar{Y}) = E\left(\frac{1}{n} \sum_{i=1}^n Y_i\right) = \frac{1}{n} E\left(\sum_{i=1}^n Y_i\right) = \frac{1}{n} \sum_{i=1}^n E(Y_i) = \frac{1}{n} \cdot n \cdot \mu_Y = \mu_Y$$

and

$$\begin{aligned}
\text{Var}(\bar{Y}) &= \text{Var}\left(\frac{1}{n} \sum_{i=1}^n Y_i\right) \\
&= \frac{1}{n^2} \sum_{i=1}^n \text{Var}(Y_i) + \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \text{cov}(Y_i, Y_j) \\
&= \frac{\sigma_Y^2}{n} \\
&= \sigma_{\bar{Y}}^2.
\end{aligned}$$

The second summand vanishes since $\text{cov}(Y_i, Y_j) = 0$ for $i \neq j$ due to independence. Consequently, the standard deviation of the sample mean is given by

$$\sigma_{\bar{Y}} = \frac{\sigma_Y}{\sqrt{n}}.$$

It is worthwhile to mention that these results hold irrespective of the underlying distribution of the Y_i .

The Sampling Distribution of \bar{Y} when Y Is Normally Distributed

If Y_1, \dots, Y_n are i.i.d. draws from a normal distribution with mean μ_Y and variance σ_Y^2 , the following holds for their sample average \bar{Y} :

$$\bar{Y} \sim \mathcal{N}(\mu_Y, \sigma_Y^2/n) \quad (2.4)$$

For example, if a sample Y_i with $i = 1, \dots, 10$ is drawn from a standard normal distribution with mean $\mu_Y = 0$ and variance $\sigma_Y^2 = 1$ we have

$$\bar{Y} \sim \mathcal{N}(0, 0.1).$$

We can use R's random number generation facilities to verify this result. The basic idea is to simulate outcomes of the true distribution of \bar{Y} by repeatedly drawing random samples of 10 observations from the $\mathcal{N}(0, 1)$ distribution and computing their respective averages. If we do this for a large number of repetitions, the simulated data set of averages should quite accurately reflect the theoretical distribution of \bar{Y} if the theoretical result holds.

The approach sketched above is an example of what is commonly known as *Monte Carlo Simulation* or *Monte Carlo Experiment*. To perform this simulation in R, we proceed as follows:

1. Choose a sample size **n** and the number of samples to be drawn, **reps**.
2. Use the function `replicate()` in conjunction with `rnorm()` to draw **n** observations from the standard normal distribution **rep** times.

Note: the outcome of `replicate()` is a matrix with dimensions **n** \times **rep**. It contains the drawn samples as *columns*.

3. Compute sample means using `colMeans()`. This function computes the mean of each column, i.e., of each sample and returns a vector.

```
# set sample size and number of samples
n <- 10
reps <- 10000

# perform random sampling
samples <- replicate(reps, rnorm(n)) # 10 x 10000 sample matrix

# compute sample means
sample.avg<math>s <- colMeans(samples)</math>
```

We then end up with a vector of sample averages. You can check the vector property of `sample.avgs`:

```
# check that 'sample.avg<math>s</math>' is a vector
is.vector(sample.avg<math>s</math>)
#> [1] TRUE

# print the first few entries to the console
head(sample.avg<math>s</math>)
#> [1] -0.1045919 0.2264301 0.5308715 -0.2243476 0.2186909 0.2564663
```

A straightforward approach to examine the distribution of univariate numerical data is to plot it as a histogram and compare it to some known or assumed distribution. By default, `hist()` will give us a frequency histogram, i.e., a bar chart where observations are grouped into ranges, also called bins. The ordinate reports the number of observations falling into each of the bins. Instead, we want it to report density estimates for comparison purposes. This is achieved by setting the argument `freq = FALSE`. The number of bins is adjusted by the argument `breaks`.

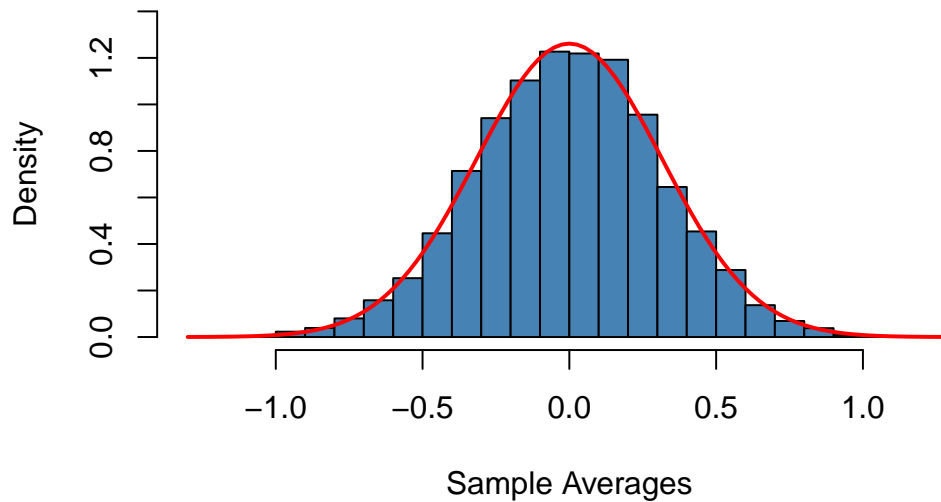
Using `curve()`, we overlay the histogram with a red line, the theoretical density of a $\mathcal{N}(0, 0.1)$ random variable. Remember to use the argument `add = TRUE` to add the curve to the current plot. Otherwise R will open a new graphic device and discard the previous plot!¹

```
# Plot the density histogram
hist(sample.avg<math>s</math>,
      ylim = c(0, 1.4),
      col = "steelblue" ,
      freq = F,
      breaks = 20,
      main="Histogram of Sample Averages",
      xlab="Sample Averages")

# overlay the theoretical distribution of sample averages on top of the histogram
curve(dnorm(x, sd = 1/sqrt(n)),
      col = "red",
      lwd = "2",
      add = T)
```

¹Hint: T and F are alternatives for TRUE and FALSE.

Histogram of Sample Averages



The sampling distribution of \bar{Y} is indeed very close to that of a $\mathcal{N}(0, 0.1)$ distribution so the Monte Carlo simulation supports the theoretical claim.

Let us discuss another example where using simple random sampling in a simulation setup helps to verify a well known result. As discussed before, the Chi-squared distribution with M degrees of freedom arises as the distribution of the sum of M independent squared standard normal distributed random variables.

To visualize the claim stated in equation (2.3), we proceed similarly as in the example before:

1. Choose the degrees of freedom, DF , and the number of samples to be drawn **reps**.
2. Draw **reps** random samples of size DF from the standard normal distribution using `replicate()`.
3. For each sample, square the outcomes and sum them up column-wise. Store the results.

Again, we produce a density estimate for the distribution underlying our simulated data using a density histogram and overlay it with a line graph of the theoretical density function of the χ_3^2 distribution.

```
# number of repetitions
reps <- 10000

# set degrees of freedom of a chi-Square Distribution
DF <- 3

# sample 10000 column vectors à 3 N(0,1) R.V.S
Z <- replicate(reps, rnorm(DF))

# column sums of squares
X <- colSums(Z^2)

# histogram of column sums of squares
hist(X,
     freq = F,
     col = "steelblue",
     breaks = 40,
     ylab = "Density",
```

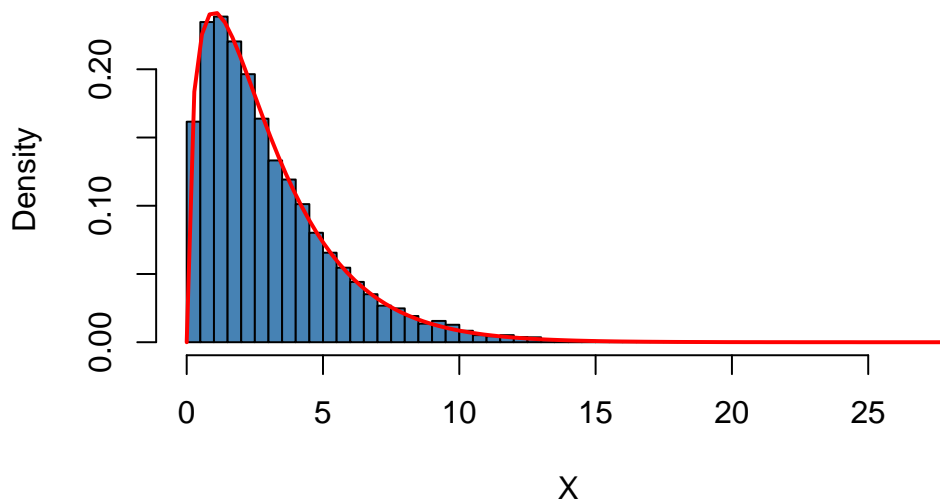


```

main = "")

# add theoretical density
curve(dchisq(x, df = DF),
      type = 'l',
      lwd = 2,
      col = "red",
      add = T)

```



Large Sample Approximations to Sampling Distributions

Sampling distributions as considered in the last section play an important role in the development of econometric methods. There are two main approaches in characterizing sampling distributions: an “exact” approach and an “approximate” approach.

The exact approach aims to find a general formula for the sampling distribution that holds for any sample size n . We call this the *exact distribution* or *finite-sample distribution*. In the previous examples of dice rolling and normal variates, we have dealt with functions of random variables whose sample distributions are *exactly known* in the sense that we can write them down as analytic expressions. However, this is not always possible. For \bar{Y} , result (2.4) tells us that normality of Y_i implies normality of \bar{Y} (we demonstrated this for the special case of $Y_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$ with $n = 10$ using a simulation study that involves simple random sampling). Unfortunately, the *exact* distribution of \bar{Y} is generally unknown and often hard to derive (or even untraceable) if we drop the assumption that the Y_i have a normal distribution.

Therefore, as it can be guessed from its name, the “approximate” approach aims to find an approximation to the sampling distribution where it is required that the sample size n is large. A distribution that is used as a large-sample approximation to the sampling distribution is also called the *asymptotic distribution*. This is due to the fact that the asymptotic distribution *is* the sampling distribution for $n \rightarrow \infty$, i.e., the approximation becomes exact if the sample size goes to infinity. However, the difference between the sampling distribution and the asymptotic distribution is negligible for moderate or even small sample sizes so that approximations using the asymptotic distribution are useful.

In this section we will discuss two well known results that are used to approximate sampling distributions and thus constitute key tools in econometric theory: the *law of large numbers* and the *central limit theorem*. The law of large numbers states that in large samples, \bar{Y} is close to μ_Y with high probability.

The central limit theorem says that the sampling distribution of the standardized sample average, that is, $(\bar{Y} - \mu_Y)/\sigma_{\bar{Y}}$ is asymptotically normally distributed. It is particularly interesting that both the results do not depend on the distribution of Y . In other words, being unable to describe the complicated sampling distribution of \bar{Y} if Y is not normal, approximations of the latter using the central limit theorem simplify the development and applicability of econometric procedures enormously. This is a key component underlying the theory of statistical inference for regression models. Both the results are summarized in Key Concept 2.6 and Key Concept 2.7.

Key Concept 2.6

Convergence in Probability, Consistency and the Law of Large Numbers

The sample average \bar{Y} converges in probability to μ_Y : \bar{Y} is *consistent* for μ_Y if the probability that \bar{Y} is in the range $(\mu_Y - \epsilon)$ to $(\mu_Y + \epsilon)$ becomes arbitrary close to 1 as n increases for any constant $\epsilon > 0$. We write this as

$$P(\mu_Y - \epsilon \leq \bar{Y} \leq \mu_Y + \epsilon) \rightarrow 1, \epsilon > 0 \text{ as } n \rightarrow \infty.$$

Consider the independently and identically distributed random variables $Y_i, i = 1, \dots, n$ with expectation $E(Y_i) = \mu_Y$ and variance $\text{Var}(Y_i) = \sigma_Y^2$. Under the condition that $\sigma_Y^2 < \infty$, that is, large outliers are unlikely, the law of large numbers states

$$\bar{Y} \xrightarrow{p} \mu_Y.$$

The following application simulates a large number of coin tosses (you may set the number of trials using the slider) with a fair coin and computes the fraction of heads observed for each additional toss. The result is a random path that, as stated by the law of large numbers, shows a tendency to approach the value of 0.5 as n grows.

This interactive application is only available in the HTML version.

The core statement of the law of large numbers is that, under quite general conditions, the probability of obtaining a sample average \bar{Y} that is close to μ_Y is high if we have a large sample size.

Consider the example of repeatedly tossing a coin where Y_i is the result of the i^{th} coin toss. Y_i is a Bernoulli distributed random variable with p the probability of observing head

$$P(Y_i) = \begin{cases} p, & Y_i = 1 \\ 1 - p, & Y_i = 0 \end{cases},$$

where $p = 0.5$ as we assume a fair coin. It is straightforward to show that

$$\mu_Y = p = 0.5.$$

Let R_n denote the proportion of heads in the first n tosses,

$$R_n = \frac{1}{n} \sum_{i=1}^n Y_i. \quad (2.5)$$

According to the law of large numbers, the observed proportion of heads converges in probability to $\mu_Y = 0.5$, the probability of tossing head in a *single* coin toss,

$$R_n \xrightarrow{p} \mu_Y = 0.5 \text{ as } n \rightarrow \infty.$$

This result is illustrated by the interactive application in Key Concept 2.6. We now show how to replicate this using R.

The procedure is as follows:

1. Sample N observations from the Bernoulli distribution, e.g., using `sample()`.
2. Calculate the proportion of heads R_n as in (2.5). A way to achieve this is to call `cumsum()` on the vector of observations Y to obtain its cumulative sum and then divide by the respective number of observations.

We continue by plotting the path and also add a dashed line for the benchmark probability $p = 0.5$.

```
# set seed
set.seed(1)

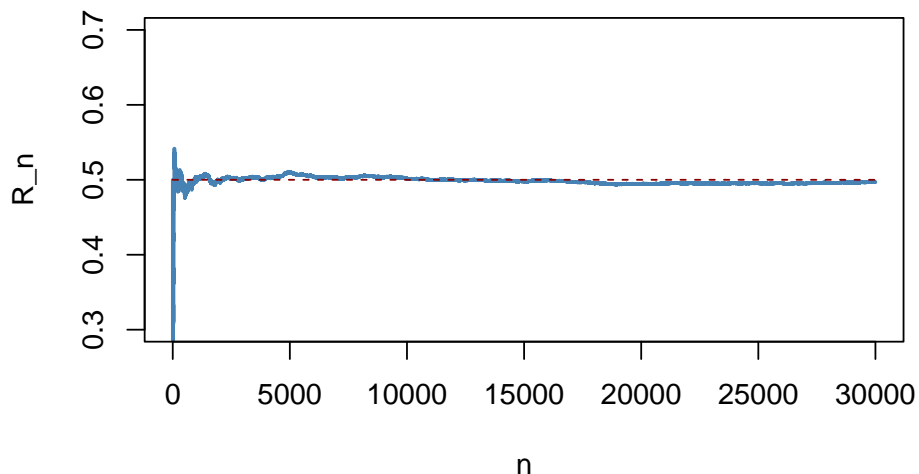
# set number of coin tosses and simulate
N <- 30000
Y <- sample(0:1, N, replace = T)

# Calculate R_n for 1:N
S <- cumsum(Y)
R <- S/(1:N)

# Plot the path.
plot(R,
     ylim = c(0.3, 0.7),
     type = "l",
     col = "steelblue",
     lwd = 2,
     xlab = "n",
     ylab = "R_n",
     main = "Converging Share of Heads in Repeated Coin Tossing")

# Add a dashed line for R_n = 0.5
lines(c(0, N),
      c(0.5, 0.5),
      col = "darkred",
      lty = 2,
      lwd = 1)
```

Converging Share of Heads in Repeated Coin Tossing



There are several things to be said about this plot.

- The blue graph shows the observed proportion of heads when tossing a coin n times.
- Since Y_i are random variables, R_n is a random variate, too. The path depicted is only one of many possible realizations of R_n as it is determined by the 30000 observations sampled from the Bernoulli distribution.
- If the number of coin tosses n is small, the proportion of heads may be anything but close to its theoretical value, $\mu_Y = 0.5$. However, as more and more observation are included in the sample, we find that the path stabilizes in the neighborhood of 0.5. The average of multiple trials shows a clear tendency to converge to its expected value as the sample size increases, just as claimed by the law of large numbers.

Key Concept 2.7 The Central Limit Theorem

Suppose that Y_1, \dots, Y_n are independently and identically distributed random variables with expectation $E(Y_i) = \mu_Y$ and variance $\text{Var}(Y_i) = \sigma_Y^2$ where $0 < \sigma_Y^2 < \infty$. The Central Limit Theorem (CLT) states that, if the sample size n goes to infinity, the distribution of the standardized sample average

$$\frac{\bar{Y} - \mu_Y}{\sigma_{\bar{Y}}} = \frac{\bar{Y} - \mu_Y}{\sigma_Y/\sqrt{n}},$$

becomes arbitrarily well approximated by the standard normal distribution.

The application below demonstrates the CLT for the sample average of normally distributed random variables with mean 5 and variance 25^2 . You may check the following properties:

- The distribution of the sample average is normal.
- As the sample size increases, the distribution of \bar{Y} tightens around the true mean of 5.
- The distribution of the standardized sample average is close to the standard normal distribution for large n .

This interactive application is only available in the HTML version.

According to the CLT, the distribution of the sample mean \bar{Y} of the Bernoulli distributed random variables $Y_i, i = 1, \dots, n$, is well approximated by the normal distribution with parameters $\mu_Y = p = 0.5$ and $\sigma_Y^2 = p(1-p)/n = 0.25/n$ for large n . Consequently, for the standardized sample mean we conclude that

$$\frac{\bar{Y} - 0.5}{0.5/\sqrt{n}} \quad (2.6)$$

should be well approximated by the standard normal distribution $\mathcal{N}(0, 1)$. We employ another simulation study to demonstrate this graphically. The idea is as follows.

Draw a large number of random samples, 10000 say, of size n from the Bernoulli distribution and compute the sample averages. Standardize the averages as shown in (2.6). Next, visualize the distribution of the generated standardized sample averages by means of a histogram and compare to the standard normal distribution. Repeat this for different sample sizes n to see how increasing the sample size n impacts the simulated distribution of the averages.

In R, realize this as follows:

1. We start by defining that the next four subsequently generated figures shall be drawn in a 2×2 array such that they can easily be compared. This is done by calling `par(mfrow = c(2, 2))` before generating the figures.
2. We define the number of repetitions `reps` as 10000 and create a vector of sample sizes named `sample.sizes`. We consider samples of sizes 5, 20, 75, and 100.
3. Next, we combine two `for()` loops to simulate the data and plot the distributions. The inner loop generates 10000 random samples, each consisting of `n` observations that are drawn from the Bernoulli distribution, and computes the standardized averages. The outer loop executes the inner loop for the different sample sizes `n` and produces a plot for each iteration.

```
# subdivide the plot panel into a 2-by-2 array
par(mfrow = c(2, 2))

# set the number of repetitions and the sample sizes
reps <- 10000
sample.sizes <- c(5, 20, 75, 100)

# set seed for reproducibility
set.seed(123)

# outer loop (loop over the sample sizes)
for (n in sample.sizes) {

  samplemean <- rep(0, reps) #initialize the vector of sample means
  stdsamplemean <- rep(0, reps) #initialize the vector of standardized sample means

  # inner loop (loop over repetitions)
  for (i in 1:reps) {
    x <- rbinom(n, 1, 0.5)
    samplemean[i] <- mean(x)
    stdsamplemean[i] <- sqrt(n)*(mean(x) - 0.5)/0.5
  }

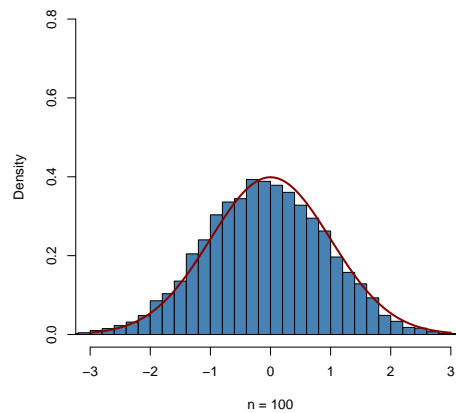
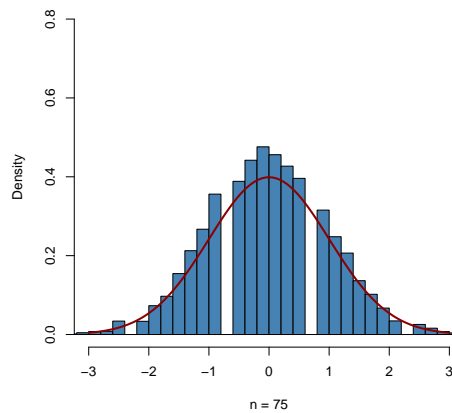
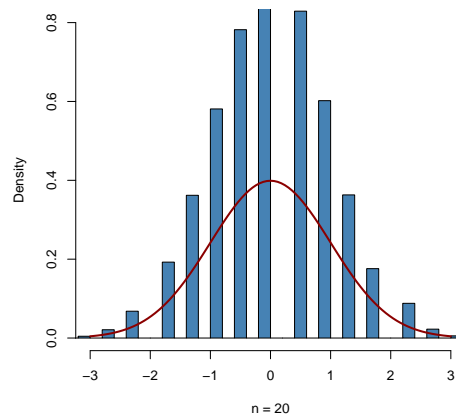
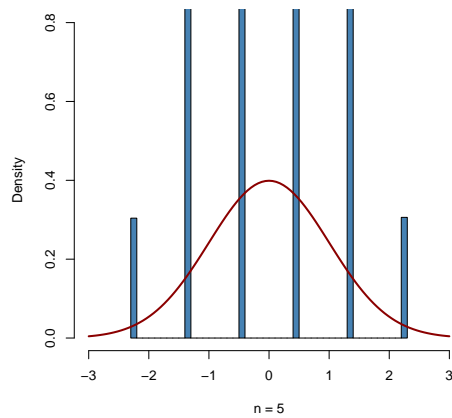
  # plot histogram and overlay the N(0,1) density in every iteration
  hist(stdsamplemean,
```

```

col = "steelblue",
freq = FALSE,
breaks = 40,
xlim = c(-3, 3),
ylim = c(0, 0.8),
xlab = paste("n =", n),
main = "")

curve(dnorm(x),
      lwd = 2,
      col = "darkred",
      add = TRUE)
}

```



We see that the simulated sampling distribution of the standardized average tends to deviate strongly from the standard normal distribution if the sample size is small, e.g., for $n = 5$ and $n = 10$. However as n grows, the histograms approach the standard normal distribution. The approximation works quite well, see $n = 100$.

2.3 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 3

A Review of Statistics using R

This section reviews important statistical concepts:

- Estimation of unknown population parameters
- Hypothesis testing
- Confidence intervals

These methods are heavily used in econometrics. We will discuss them in the simple context of inference about an unknown population mean and discuss several applications in R. These R applications rely on the following packages which are not part of the base version of R:

- `readxl` - allows to import data from *Excel* to R.
- `dplyr` - provides a flexible grammar for data manipulation.
- `MASS` - a collection of functions for applied statistics.

Make sure these are installed before you go ahead and try to replicate the examples. The safest way to do so is by checking whether the following code chunk executes without any errors.

```
library(dplyr)
library(MASS)
library(readxl)
```

3.1 Estimation of the Population Mean

Key Concept 3.1 Estimators and Estimates

Estimators are functions of sample data that are drawn randomly from an unknown population. *Estimates* are numeric values computed by estimators based on the sample data. Estimators are random variables because they are functions of *random* data. Estimates are nonrandom numbers.

Think of some economic variable, for example hourly earnings of college graduates, denoted by Y . Suppose we are interested in μ_Y the mean of Y . In order to exactly calculate μ_Y we would have to interview every working graduate in the economy. We simply cannot do this due to time and cost

constraints. However, we can draw a random sample of n i.i.d. observations Y_1, \dots, Y_n and estimate μ_Y using one of the simplest estimators in the sense of Key Concept 3.1 one can think of, that is,

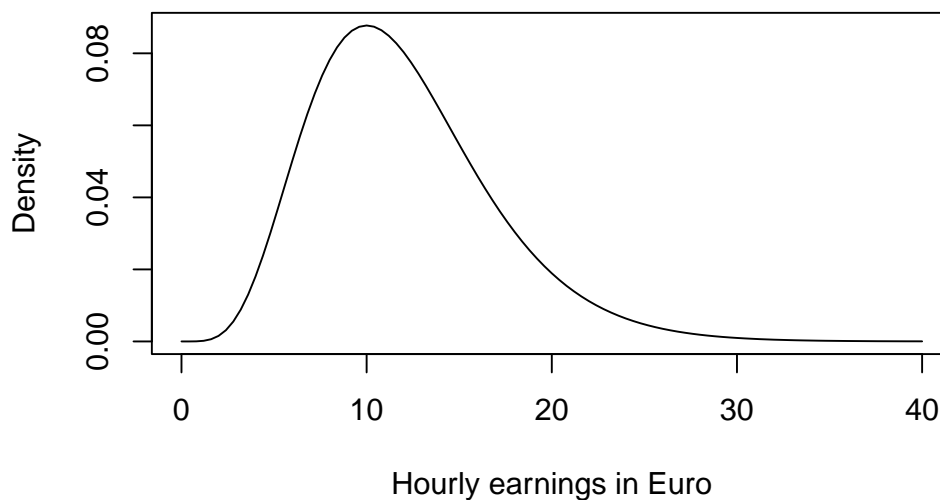
$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i,$$

the sample mean of Y . Then again, we could use an even simpler estimator for μ_Y : the very first observation in the sample, Y_1 . Is Y_1 a good estimator? For now, assume that

$$Y \sim \chi_{12}^2,$$

which is not too unreasonable as hourly income is non-negative and we expect many hourly earnings to be in a range of 5 Euro to 15 Euro. Moreover, it is common for income distributions to be skewed to the right — a property of the χ_{12}^2 distribution.

```
# plot the chi_12^2 distribution
curve(dchisq(x, df=12),
      from = 0,
      to = 40,
      ylab = "Density",
      xlab = "Hourly earnings in Euro")
```



We now draw a sample of $n = 100$ observations and take the first observation Y_1 as an estimate for μ_Y

```
# set seed for reproducibility
set.seed(1)

# sample from the chi_12^2 distribution, use only the first observation
rsamp <- rchisq(n = 100, df = 12)
rsamp[1]
#> [1] 8.257893
```

The estimate 8.26 is not too far away from $\mu_Y = 12$ but it is somewhat intuitive that we could do better: the estimator Y_1 discards a lot of information and its variance is the population variance:

$$\text{Var}(Y_1) = \text{Var}(Y) = 2 \cdot 12 = 24$$

This brings us to the following question: What is a *good* estimator of an unknown parameter in the first place? This question is tackled in Key Concepts 3.2 and 3.3.

Key Concept 3.2 Bias, Consistency and Efficiency

Desirable characteristics of an estimator include unbiasedness, consistency and efficiency.

Unbiasedness:

If the mean of the sampling distribution of some estimator $\hat{\mu}_Y$ for the population mean μ_Y equals μ_Y ,

$$E(\hat{\mu}_Y) = \mu_Y,$$

the estimator is unbiased for μ_Y . The *bias* of $\hat{\mu}_Y$ then is 0:

$$E(\hat{\mu}_Y) - \mu_Y = 0.$$

Consistency:

We want the uncertainty of the estimator μ_Y to decrease as the number of observations in the sample grows. More precisely, we want the probability that the estimate $\hat{\mu}_Y$ falls within a small interval around the true value μ_Y to get increasingly closer to 1 as n grows. We write this as

$$\hat{\mu}_Y \xrightarrow{p} \mu_Y.$$

Variance and efficiency:

We want the estimator to be efficient. Suppose we have two estimators, $\hat{\mu}_Y$ and $\tilde{\mu}_Y$ and for some given sample size n it holds that

$$E(\hat{\mu}_Y) = E(\tilde{\mu}_Y) = \mu_Y,$$

but

$$\text{Var}(\hat{\mu}_Y) < \text{Var}(\tilde{\mu}_Y).$$

We then prefer to use $\hat{\mu}_Y$ as it has a lower variance than $\tilde{\mu}_Y$, meaning that $\hat{\mu}_Y$ is more *efficient* in using the information provided by the observations in the sample.

3.2 Properties of the Sample Mean

A more precise way to express consistency of an estimator $\hat{\mu}$ for a parameter μ is

$$P(|\hat{\mu} - \mu| < \epsilon) \xrightarrow[n \rightarrow \infty]{p} 1 \quad \text{for any } \epsilon > 0.$$

This expression says that the probability of observing a deviation from the true value μ that is smaller than some arbitrary $\epsilon > 0$ converges to 1 as n grows. Consistency does not require unbiasedness.

To examine properties of the sample mean as an estimator for the corresponding population mean, consider the following R example.

We generate a population, denoted as `pop`, consisting of observations Y_i , where $i = 1, \dots, 10000$. These observations are generated from a normal distribution with mean $\mu = 10$ and variance $\sigma^2 = 1$.

To investigate the behavior of the estimator $\hat{\mu} = \bar{Y}$, we can draw random samples from this population

and calculate \bar{Y} for each of them. This is easily done by making use of the function `replicate()`. The argument `expr` is evaluated `n` times. In this case we draw samples of sizes $n = 5$ and $n = 25$, compute the sample means and repeat this exactly $N = 25000$ times.

For comparison purposes we store results for the estimator Y_1 , the first observation in a sample of size 5, separately.

```
# generate a fictitious population
pop <- rnorm(10000, 10, 1)

# sample from the population and estimate the mean
est1 <- replicate(expr = mean(sample(x = pop, size = 5)), n = 25000)

est2 <- replicate(expr = mean(sample(x = pop, size = 25)), n = 25000)

fo <- replicate(expr = sample(x = pop, size = 5)[1], n = 25000)
```

Check that `est1` and `est2` are vectors of length 25000:

```
# check if object type is vector
is.vector(est1)
#> [1] TRUE
is.vector(est2)
#> [1] TRUE

# check length
length(est1)
#> [1] 25000
length(est2)
#> [1] 25000
```

The code chunk below produces a plot of the sampling distributions of the estimators \bar{Y} and Y_1 on the basis of the 25000 samples in each case. We also plot the density function of the $\mathcal{N}(10, 1)$ distribution.

```
# plot density estimate Y_1
plot(density(fo),
     col = "green",
     lwd = 2,
     ylim = c(0, 2),
     xlab = "Estimates",
     main = "Sampling Distributions of Unbiased Estimators")

# add density estimate for the distribution of the sample mean with n=5 to the plot
lines(density(est1),
     col = "steelblue",
     lwd = 2,
     bty = "n")

# add density estimate for the distribution of the sample mean with n=25 to the plot
lines(density(est2),
     col = "red2",
     lwd = 2)

# add a vertical line at the true parameter
```

```

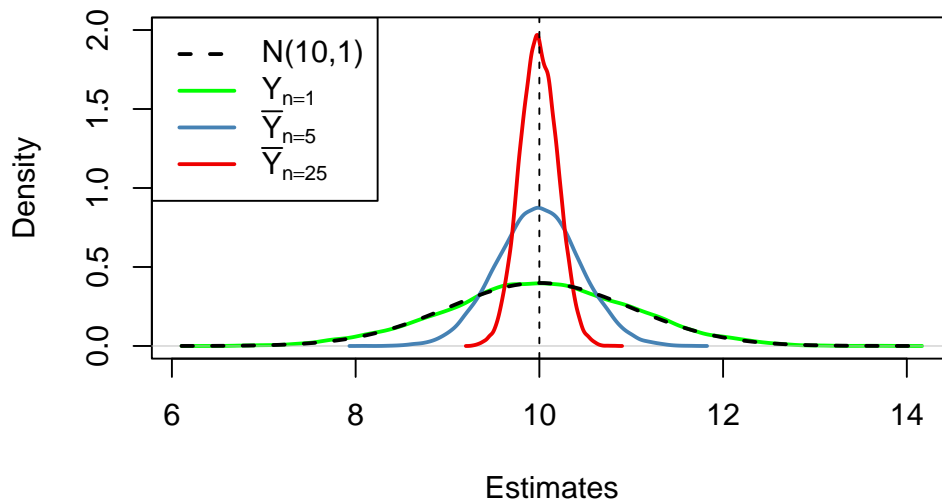
abline(v = 10, lty = 2)

# add N(10,1) density to the plot
curve(dnorm(x, mean = 10),
      lwd = 2,
      lty = 2,
      add = T)

# add a legend
legend("topleft",
      legend = c("N(10,1)",
                expression(Y[n == 1]),
                expression(bar(Y)[n == 5]),
                expression(bar(Y)[n == 25])
                ),
      lty = c(2, 1, 1, 1),
      col = c("black", "green", "steelblue", "red2"),
      lwd = 2)

```

Sampling Distributions of Unbiased Estimators



First, *all* sampling distributions (represented by the solid lines) are centered around $\mu = 10$. This is evidence for the *unbiasedness* of Y_1 , \bar{Y}_5 and \bar{Y}_{25} . Of course, the theoretical density $\mathcal{N}(10, 1)$ is also centered at 10.

Next, have a look at the spread of the sampling distributions. Several things are noteworthy:

- The sampling distribution of Y_1 (green curve) tracks the density of the $\mathcal{N}(10, 1)$ distribution (black dashed line) pretty closely. In fact, the sampling distribution of Y_1 is the $\mathcal{N}(10, 1)$ distribution. This is less surprising if you keep in mind that the Y_1 estimator does nothing but reporting an observation that is randomly selected from a population with $\mathcal{N}(10, 1)$ distribution. Hence, $Y_1 \sim \mathcal{N}(10, 1)$. Note that this result does not depend on the sample size n : the sampling distribution of Y_1 is *always* the population distribution, no matter how large the sample is, Y_1 is a good estimate of μ_Y , but we can do better.
- Both the sampling distributions of \bar{Y} show less dispersion than the sampling distribution of Y_1 .

This means that \bar{Y} has a lower variance than Y_1 . In view of Key Concepts 3.2 and 3.3, we find that \bar{Y} is a more efficient estimator than Y_1 . In fact, this holds for all $n > 1$.

- \bar{Y} shows a behavior illustrating consistency (see Key Concept 3.2). The blue and the red densities are much more concentrated around $\mu = 10$ than the green one. As the number of observations is increased from 1 to 5, the sampling distribution tightens around the true parameter. By increasing the sample size to 25, this effect becomes more apparent. This implies that the probability of obtaining estimates that are close to the true value increases with n . This is also reflected by the estimated values of the density function close to 10: the larger the sample size, the larger the value of the density.

We encourage you to go ahead and modify the code. Try out different values for the sample size and see how the sampling distribution of \bar{Y} changes!

\bar{Y} is the Least Squares Estimator of μ_Y

Assume you have some observations Y_1, \dots, Y_n on $Y \sim \mathcal{N}(10, 1)$ (which is unknown) and would like to find an estimator m that predicts the observations as good as possible. Here, we aim to find an estimator m that results in a small total squared deviation between the predicted and observed values. Mathematically, this means we want to find an m that minimizes

$$\sum_{i=1}^n (Y_i - m)^2. \quad (3.1)$$

Think of $Y_i - m$ as the mistake made when predicting Y_i by m . We could also minimize the sum of absolute deviations from m but minimizing the sum of squared deviations is mathematically more convenient (and will lead to a different result). That is why the estimator we are looking for is called the *least squares estimator*. $m = \bar{Y}$, the sample mean, is this estimator.

We can show this by generating a random sample and plotting (3.1) as a function of m .

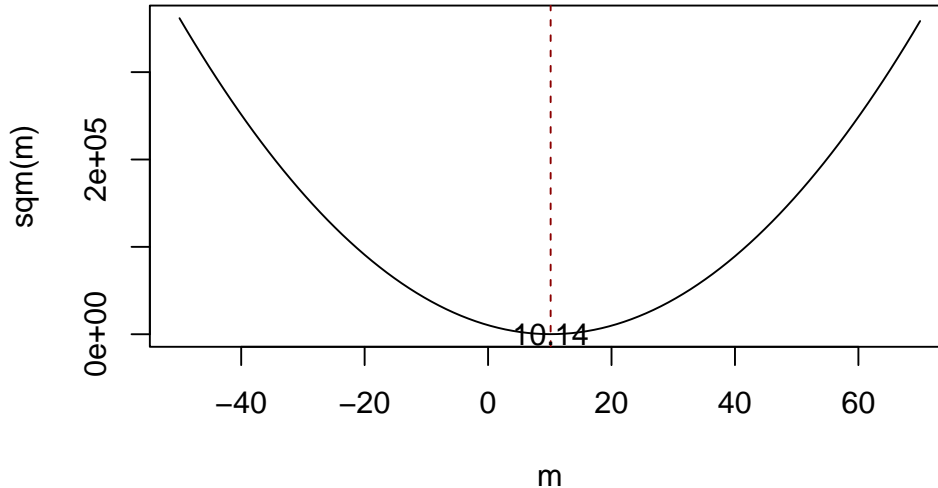
```
# define the function and vectorize it
sqm <- function(m) {
  sum((y-m)^2)
}
sqm <- Vectorize(sqm)

# draw random sample and compute the mean
y <- rnorm(100, 10, 1)
mean(y)
#> [1] 10.1364

# plot the objective function
curve(sqm(x),
      from = -50,
      to = 70,
      xlab = "m",
      ylab = "sqm(m)")

# add vertical line at mean(y)
abline(v = mean(y),
       lty = 2,
       col = "darkred")
```

```
# add annotation at mean(y)
text(x = mean(y),
     y = 0,
     labels = paste(round(mean(y), 2)))
```



Notice that (3.1) is a quadratic function so that there is only one minimum. The plot shows that this minimum lies exactly at the sample mean of the sample data.

Some R functions can only interact with functions that take a vector as an input and evaluate the function body on every entry of the vector, for example `curve()`. We call such functions vectorized functions and it is often a good idea to write vectorized functions yourself, although this is cumbersome in some cases. Having a vectorized function in R is never a drawback since, these functions work on both single values and vectors.

Let us look at the function `sqm()`, which is non-vectorized:

```
sqm <- function(m) {
  sum((y-m)^2) #body of the function
}
```

Providing, e.g., `c(1,2,3)` as the argument `m` would cause an error since then the operation `y-m` is invalid: the vectors `y` and `m` are of incompatible dimensions. This is why we cannot use `sqm()` in conjunction with `curve()`.

Here `Vectorize()` comes into play. It generates a vectorized version of a non-vectorized function.

Why is Random Sampling Important ?

So far, we assumed (sometimes implicitly) that the observed data Y_1, \dots, Y_n are the result of a sampling process that satisfies the assumption of simple random sampling. This assumption often is fulfilled when estimating a population mean using \bar{Y} . If this is not the case, estimates may be biased.

Let us fall back to `pop`, the fictitious population of 10000 observations and compute the population mean μ_{pop} :

```
# compute the population mean of pop
mean(pop)
#> [1] 9.992604
```

Next we sample 25 observations from `pop` with `sample()` and estimate μ_{pop} using \bar{Y} repeatedly. However, now we use a sampling scheme that deviates from simple random sampling: instead of ensuring that each member of the population has the same chance to end up in a sample, we assign a higher probability of being sampled to the 2500 smallest observations of the population by setting the argument `prob` to a suitable vector of probability weights:

```
# simulate outcomes for the sample mean when the i.i.d. assumption fails
est3 <- replicate(n = 25000,
                 expr = mean(sample(x = sort(pop),
                                   size = 25,
                                   prob = c(rep(4, 2500), rep(1, 7500))))))

# compute the sample mean of the outcomes
mean(est3)
#> [1] 9.443067
```

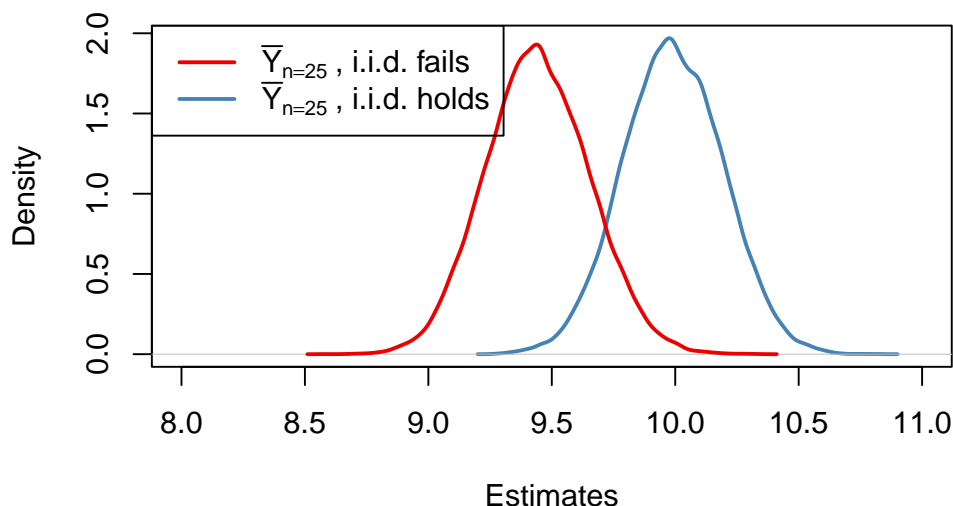
Next we plot the sampling distribution of \bar{Y} for this non-i.i.d. case and compare it to the sampling distribution when the i.i.d. assumption holds.

```
# sampling distribution of sample mean, i.i.d. holds, n=25
plot(density(est2),
     col = "steelblue",
     lwd = 2,
     xlim = c(8, 11),
     xlab = "Estimates",
     main = "When the i.i.d. Assumption Fails")

# sampling distribution of sample mean, i.i.d. fails, n=25
lines(density(est3),
      col = "red2",
      lwd = 2)

# add a legend
legend("topleft",
      legend = c(expression(bar(Y)[n == 25]~", i.i.d. fails"),
                expression(bar(Y)[n == 25]~", i.i.d. holds")
                ),
      lty = c(1, 1),
      col = c("red2", "steelblue"),
      lwd = 2)
```

When the i.i.d. Assumption Fails



Here, the failure of the i.i.d. assumption implies that, on average, we *underestimate* μ_Y using \bar{Y} : the corresponding distribution of \bar{Y} is shifted to the left. In other words, \bar{Y} is a *biased* estimator for μ_Y if the i.i.d. assumption does not hold.

3.3 Hypothesis Tests concerning the Population Mean

In this section we briefly review concepts in hypothesis testing and discuss how to conduct hypothesis tests in R. We focus on drawing inferences about an unknown population mean.

About Hypotheses and Hypothesis Testing

In a significance test we want to exploit the information contained in a sample as evidence in favor of against a hypothesis. Essentially, hypotheses are simple questions that can be answered by ‘yes’ or ‘no’. In a hypothesis test we typically deal with two different hypotheses:

- The *null hypothesis*, denoted by H_0 , is the hypothesis we are interested in testing.
- There must be an *alternative hypothesis*, denoted by H_1 , the hypothesis that is thought to hold if the null hypothesis is rejected.

The null hypothesis that the population mean of Y equals the value $\mu_{Y,0}$ is written as

$$H_0 : E(Y) = \mu_{Y,0}.$$

Often the alternative hypothesis is chosen such that it is the most general one,

$$H_1 : E(Y) \neq \mu_{Y,0},$$

meaning that $E(Y)$ may be anything but the value under the null hypothesis. This is called a *two-sided* alternative.

For the sake of brevity, we only consider two-sided alternatives in the subsequent sections of this chapter.

The p-Value

Assume that the null hypothesis is *true*. The p -value is the probability of drawing data and observing a corresponding test statistic that is at least as adverse to what is stated under the null hypothesis as the test statistic actually computed using the sample data.

In the context of the population mean and the sample mean, this definition can be stated mathematically in the following way:

$$p\text{-value} = P_{H_0} \left[|\bar{Y} - \mu_{Y,0}| > |\bar{Y}^{act} - \mu_{Y,0}| \right], \quad (3.2)$$

In (3.2), \bar{Y}^{act} is the sample mean for the data at hand (a value). In order to compute the p -value as in (3.2), knowledge about the sampling distribution of \bar{Y} (a random variable) when the null hypothesis is true (the *null distribution*) is required. However, in most cases the sampling distribution and thus the null distribution of \bar{Y} are unknown. Fortunately, the CLT (see Key Concept 2.7) allows for the large-sample approximation

$$\bar{Y} \approx \mathcal{N}(\mu_{Y,0}, \sigma_{\bar{Y}}^2), \quad \sigma_{\bar{Y}}^2 = \frac{\sigma_Y^2}{n},$$

assuming the null hypothesis $H_0 : E(Y) = \mu_{Y,0}$ is true. With some algebra it follows for large n that

$$\frac{\bar{Y} - \mu_{Y,0}}{\sigma_Y/\sqrt{n}} \sim \mathcal{N}(0, 1).$$

So in large samples, the p -value can be computed *without* knowledge of the exact sampling distribution of \bar{Y} using the above normal approximation.

Calculating the p-Value when the Standard Deviation is Known

For now, let us assume that $\sigma_{\bar{Y}}$ is known. Then, we can rewrite (3.2) as

$$p\text{-value} = P_{H_0} \left[\left| \frac{\bar{Y} - \mu_{Y,0}}{\sigma_{\bar{Y}}} \right| > \left| \frac{\bar{Y}^{act} - \mu_{Y,0}}{\sigma_{\bar{Y}}} \right| \right] \quad (3.3)$$

$$= 2 \cdot \Phi \left[- \left| \frac{\bar{Y}^{act} - \mu_{Y,0}}{\sigma_{\bar{Y}}} \right| \right]. \quad (3.4)$$

The p -value is this area in the tails of the $\mathcal{N}(0, 1)$ distribution that lies beyond

$$\pm \left| \frac{\bar{Y}^{act} - \mu_{Y,0}}{\sigma_{\bar{Y}}} \right|, \quad (3.5)$$

We now use R to visualize what is stated in (3.4) and (3.5). The next code chunk replicates Figure 3.1 of the book.

```
# plot the standard normal density on the interval [-4,4]
curve(dnorm(x),
      xlim = c(-4, 4),
      main = "Calculating a p-Value",
      yaxs = "i",
```



```

xlab = "z",
ylab = "",
lwd = 2,
axes = "F")

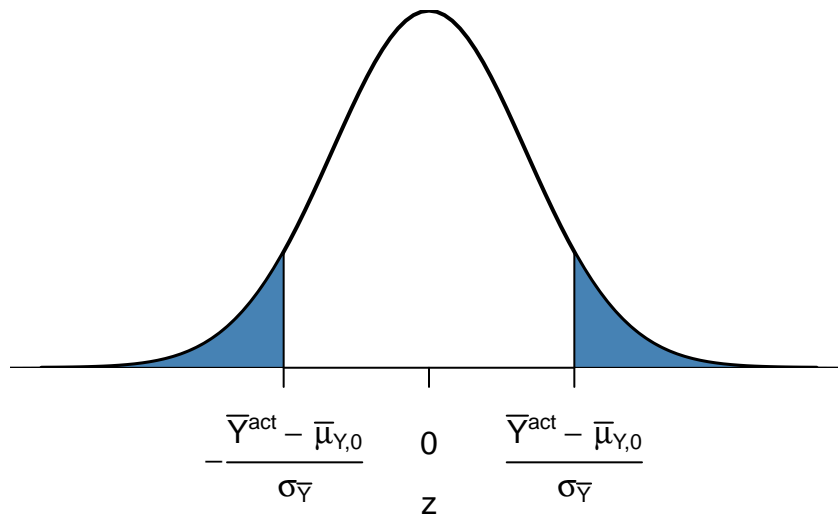
# add x-axis
axis(1,
     at = c(-1.5, 0, 1.5),
     padj = 0.75,
     labels = c(expression(-frac(bar(Y)^"act"---bar(mu)["Y,0"], sigma[bar(Y)])),
                0,
                expression(frac(bar(Y)^"act"---bar(mu)["Y,0"], sigma[bar(Y)])))

# shade p-value/2 region in left tail
polygon(x = c(-6, seq(-6, -1.5, 0.01), -1.5),
        y = c(0, dnorm(seq(-6, -1.5, 0.01)), 0),
        col = "steelblue")

# shade p-value/2 region in right tail
polygon(x = c(1.5, seq(1.5, 6, 0.01), 6),
        y = c(0, dnorm(seq(1.5, 6, 0.01)), 0),
        col = "steelblue")

```

Calculating a p-Value



Sample Variance, Sample Standard Deviation and Standard Error

If $\sigma_{\bar{Y}}$ is unknown, it must be estimated. This can be done using the sample variance

$$s_{\bar{Y}}^2 = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2. \quad (3.6)$$

Furthermore

$$s_Y = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2} \quad (3.7)$$

is a suitable estimator for the standard deviation of Y . In R, s_Y is implemented in the function `sd()`, see `?sd`.

Using R we can illustrate that s_Y is a consistent estimator for σ_Y , that is

$$s_Y \xrightarrow{p} \sigma_Y.$$

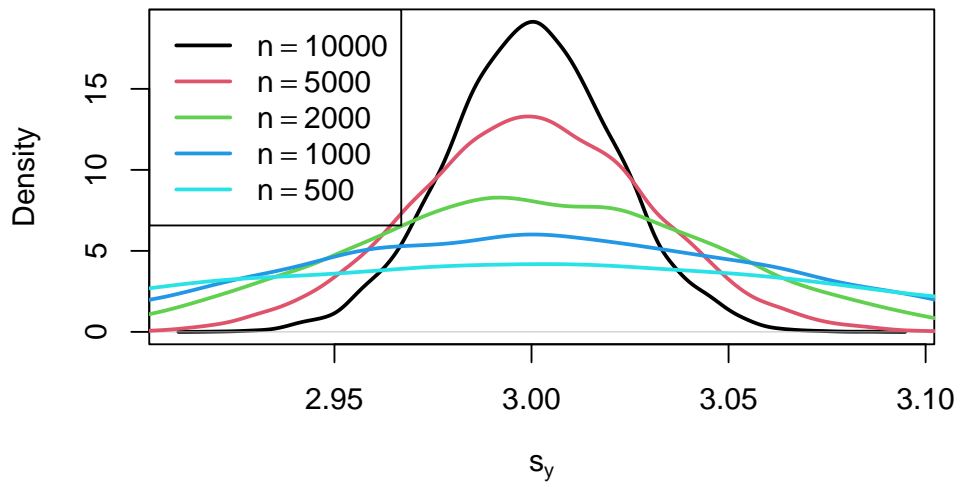
The idea here is to generate a large number of samples Y_1, \dots, Y_n where, $Y \sim \mathcal{N}(10, 9)$ say, estimate σ_Y using s_Y and investigate how the distribution of s_Y changes as n gets larger.

```
# vector of sample sizes
n <- c(10000, 5000, 2000, 1000, 500)

# sample observations, estimate using 'sd()' and plot the estimated distributions
sq_y <- replicate(n = 10000, expr = sd(rnorm(n[1], 10, 3)))
plot(density(sq_y),
     main = expression("Sampling Distributions of" ~ s[Y]),
     xlab = expression(s[y]),
     lwd = 2)

for (i in 2:length(n)) {
  sq_y <- replicate(n = 10000, expr = sd(rnorm(n[i], 10, 3)))
  lines(density(sq_y),
        col = i,
        lwd = 2)
}

# add a legend
legend("topleft",
      legend = c(expression(n == 10000),
                  expression(n == 5000),
                  expression(n == 2000),
                  expression(n == 1000),
                  expression(n == 500)),
      col = 1:5,
      lwd = 2)
```

Sampling Distributions of s_Y 

The plot shows that the distribution of s_Y tightens around the true value $\sigma_Y = 3$ as n increases.

The function that estimates the standard deviation of an estimator is called the *standard error of the estimator*. Key Concept 3.4 summarizes the terminology in the context of the sample mean.

Key Concept 3.4 The Standard Error of \bar{Y}

Take an i.i.d. sample Y_1, \dots, Y_n . The mean of Y is consistently estimated by \bar{Y} , the sample mean of the Y_i . Since \bar{Y} is a random variable, it has a sampling distribution with variance $\frac{\sigma_Y^2}{n}$. The standard error of \bar{Y} , denoted $SE(\bar{Y})$ is an estimator of the standard deviation of \bar{Y} :

$$SE(\bar{Y}) = \hat{\sigma}_{\bar{Y}} = \frac{s_Y}{\sqrt{n}}.$$

The caret ($\hat{\cdot}$) over σ indicates that $\hat{\sigma}_{\bar{Y}}$ is an estimator for $\sigma_{\bar{Y}}$.

As an example to underpin Key Concept 3.4, consider a sample of $n = 100$ i.i.d. observations of the Bernoulli distributed variable Y with success probability $p = 0.1$. Thus $E(Y) = p = 0.1$ and $\text{Var}(Y) = p(1 - p)$. $E(Y)$ can be estimated by \bar{Y} , which then has variance

$$\sigma_{\bar{Y}}^2 = p(1 - p)/n = 0.0009$$

and standard deviation

$$\sigma_{\bar{Y}} = \sqrt{p(1 - p)/n} = 0.03.$$

In this case the standard error of \bar{Y} can be estimated by

$$SE(\bar{Y}) = \sqrt{\bar{Y}(1 - \bar{Y})/n}.$$

Let us check whether \bar{Y} and $SE(\bar{Y})$ estimate the respective true values, on average.

```

# draw 10000 samples of size 100 and estimate the mean of Y and
# estimate the standard error of the sample mean

mean_estimates <- numeric(10000)
se_estimates <- numeric(10000)

for (i in 1:10000) {

  s <- sample(0:1,
             size = 100,
             prob = c(0.9, 0.1),
             replace = T)

  mean_estimates[i] <- mean(s)
  se_estimates[i] <- sqrt(mean(s) * (1 - mean(s)) / 100)

}

mean(mean_estimates)
#> [1] 0.100233
mean(se_estimates)
#> [1] 0.02958415

```

Both estimators seem to be unbiased for the true parameters. In fact, this is true for the sample mean, but not for $SE(\bar{Y})$. However, both estimators are *consistent* for the true parameters.

Calculating the p-value When the Standard Deviation is Unknown

When σ_Y is unknown, the p -value for a hypothesis test concerning μ_Y using \bar{Y} can be computed by replacing $\sigma_{\bar{Y}}$ in (3.4) by the standard error $SE(\bar{Y}) = \hat{\sigma}_{\bar{Y}}$. Then,

$$p\text{-value} = 2 \cdot \Phi \left(- \left| \frac{\bar{Y}^{act} - \mu_{Y,0}}{SE(\bar{Y})} \right| \right).$$

This is easily done in R:

```

# sample and estimate, compute standard error
samplemean_act <- mean(
  sample(0:1,
        prob = c(0.9, 0.1),
        replace = T,
        size = 100))

SE_samplemean <- sqrt(samplemean_act * (1 - samplemean_act) / 100)

# null hypothesis
mean_h0 <- 0.1

# compute the p-value
pvalue <- 2 * pnorm(- abs(samplemean_act - mean_h0) / SE_samplemean)
pvalue
#> [1] 0.2396777

```

Later in the book, we will encounter more convenient approaches to obtain t -statistics and p -values using R.

The t -statistic

In hypothesis testing, the standardized sample average

$$t = \frac{\bar{Y} - \mu_{Y,0}}{SE(\bar{Y})} \quad (3.8)$$

is called a t -statistic. This t -statistic plays an important role in testing hypotheses about μ_Y . It is a prominent example of a test statistic.

Implicitly, we already have computed a t -statistic for \bar{Y} in the previous code chunk.

```
# compute a t-statistic for the sample mean
tstatistic <- (samplemean_act - mean_h0) / SE_samplemean
tstatistic
#> [1] -1.175793
```

Using R we can illustrate that if $\mu_{Y,0}$ equal the true value, that is, if the null hypothesis is true, (3.8) is approximately $\mathcal{N}(0, 1)$ distributed when n is large.

```
# prepare empty vector for t-statistics
tstatistics <- numeric(10000)

# set sample size
n <- 300

# simulate 10000 t-statistics
for (i in 1:10000) {

  s <- sample(0:1,
             size = n,
             prob = c(0.9, 0.1),
             replace = T)

  tstatistics[i] <- (mean(s)-0.1)/sqrt(var(s)/n)

}
```

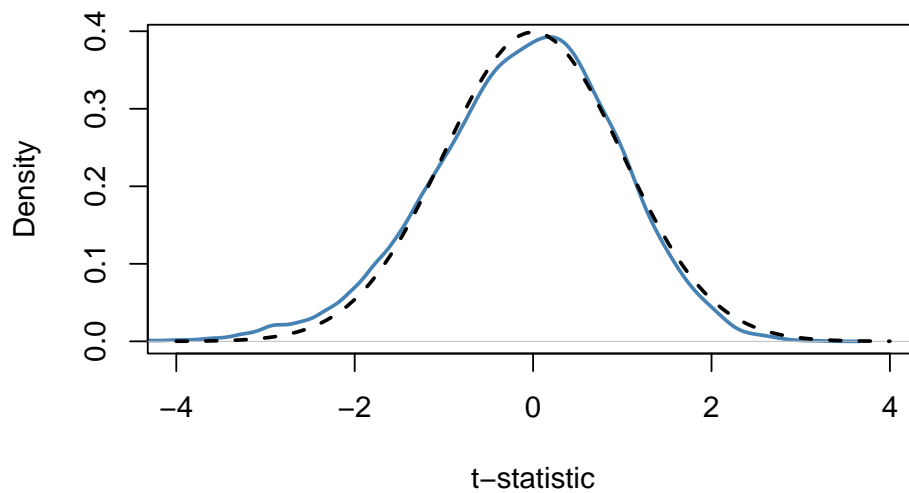
In the simulation above we estimate the variance of the Y_i using `var(s)`. This is more general than `mean(s)*(1-mean(s))` since the latter requires that the data are Bernoulli distributed and that we know this.

```
# plot density and compare to N(0,1) density
plot(density(tstatistics),
     xlab = "t-statistic",
     main = "Estimated Distribution of the t-statistic when n=300",
     lwd = 2,
     xlim = c(-4, 4),
     col = "steelblue")

# N(0,1) density (dashed)
```

```
curve(dnorm(x),  
      add = T,  
      lty = 2,  
      lwd = 2)
```

Estimated Distribution of the t-statistic when n=300



Judging from the plot, the normal approximation works reasonably well for the chosen sample size. This normal approximation has already been used in the definition of the p -value, see (3.8).

Hypothesis Testing with a Prespecified Significance Level

Key Concept 3.5 The Terminology of Hypothesis Testing

In hypothesis testing, two types of mistakes are possible:

1. The null hypothesis *is* rejected although it is true (type-I-error)
2. The null hypothesis *is not* rejected although it is false (type-II-error)

The *significance level* of the test is the probability to commit a type-I-error we are willing to accept in advance. E.g., using a prespecified significance level of 0.05, we reject the null hypothesis if and only if the *p*-value is less than 0.05. The significance level is chosen before the test is conducted.

An equivalent procedure is to reject the null hypothesis if the observed test statistic is, in absolute value terms, larger than the *critical value* of the test statistic. The critical value is determined by the significance level chosen and defines two disjoint sets of values which are called *acceptance region* and *rejection region*. The acceptance region contains all values of the test statistic for which the test does not reject while the rejection region contains all the values for which the test does reject.

The *p*-value is the probability that, in repeated sampling under the same conditions, a test statistic is observed that provides just as much evidence against the null hypothesis as the test statistic actually observed.

The actual probability that the test rejects the true null hypothesis is called the *size of the test*. In an ideal setting, the size equals the significance level.

The probability that the test correctly rejects a false null hypothesis is called *power*.

Reconsider the `pvalue` computed further above:

```
# check whether p-value < 0.05
pvalue < 0.05
#> [1] FALSE
```

The condition is not fulfilled so we do not reject the null hypothesis correctly.

When working with a *t*-statistic instead, it is equivalent to apply the following rule:

$$\text{Reject } H_0 \text{ if } |t^{act}| > 1.96.$$

We reject the null hypothesis at the significance level of 5% if the computed *t*-statistic lies beyond the critical value of 1.96 in absolute value terms. 1.96 is the 0.975-quantile of the standard normal distribution.

```
# check the critical value
qnorm(p = 0.975)
#> [1] 1.959964

# check whether the null is rejected using the t-statistic computed further above
```

```
abs(tstatistic) > 1.96
#> [1] FALSE
```

Just like using the p -value, we cannot reject the null hypothesis using the corresponding t -statistic. Key Concept 3.6 summarizes the procedure of performing a two-sided hypothesis test about the population mean $E(Y)$.

Key Concept 3.6

Testing the Hypothesis $E(Y) = \mu_{Y,0}$ Against the Alternative $E(Y) \neq \mu_{Y,0}$

1. Estimate μ_Y using \bar{Y} and compute the standard error of \bar{Y} , $SE(\bar{Y})$.
2. Compute the t -statistic.
3. Compute the p -value and reject the null hypothesis at the 5% level of significance if the p -value is smaller than 0.05 or, equivalently, if

$$|t^{act}| > 1.96.$$

One-sided Alternatives

Sometimes we are interested in testing if the mean is bigger or smaller than some value hypothesized under the null. To stick to the book, take the presumed wage gap between well and less educated working individuals. Since we anticipate that such a differential exists, a relevant alternative (to the null hypothesis that there is no wage differential) is that well educated individuals earn more, i.e., that the average hourly wage for this group, μ_Y is *bigger* than $\mu_{Y,0}$, the average wage of less educated workers which we assume to be known here for simplicity (Section 3.5 discusses how to test the equivalence of to unknown population means).

This is an example of a *right-sided test* and the hypotheses pair is chosen to be

$$H_0 : \mu_Y = \mu_{Y,0} \text{ vs } H_1 : \mu_Y > \mu_{Y,0}.$$

We reject the null hypothesis if the computed test-statistic is larger than the critical value 1.64, the 0.95-quantile of the $\mathcal{N}(0, 1)$ distribution. This ensures that $1 - 0.95 = 5\%$ probability mass remains in the area to the right of the critical value. As before, we can visualize this in R using the function `polygon()`.

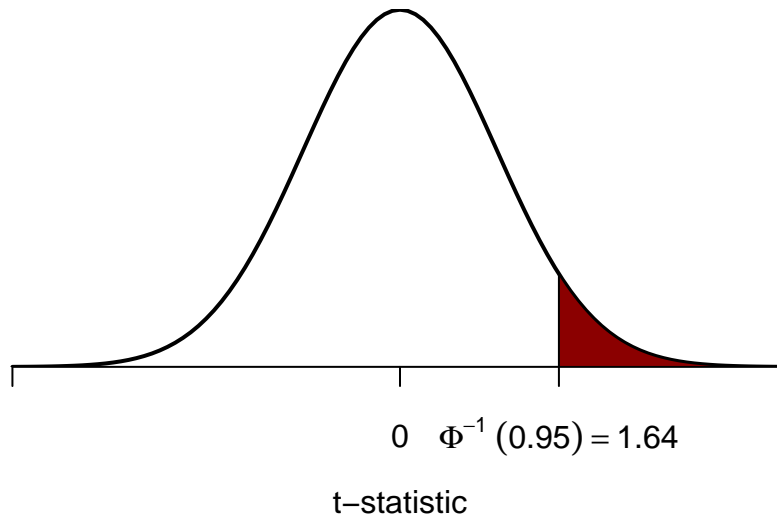
```
# plot the standard normal density on the domain [-4,4]
curve(dnorm(x),
      xlim = c(-4, 4),
      main = "Rejection Region of a Right-Sided Test",
      yaxs = "i",
      xlab = "t-statistic",
      ylab = "",
      lwd = 2,
      axes = "F")

# add the x-axis
axis(1,
     at = c(-4, 0, 1.64, 4),
     padj = 0.5,
     labels = c("", 0, expression(Phi^-1~(.95)==1.64), ""))
```



```
# shade the rejection region in the left tail
polygon(x = c(1.64, seq(1.64, 4, 0.01), 4),
        y = c(0, dnorm(seq(1.64, 4, 0.01)), 0),
        col = "darkred")
```

Rejection Region of a Right-Sided Test



Analogously, for the left-sided test we have

$$H_0 : \mu_Y = \mu_{Y,0} \text{ vs. } H_1 : \mu_Y < \mu_{Y,0}.$$

The null is rejected if the observed test statistic falls short of the critical value which, for a test at the 0.05 level of significance, is given by -1.64 , the 0.05-quantile of the $\mathcal{N}(0, 1)$ distribution. 5% probability mass lies to the left of the critical value.

It is straightforward to adapt the code chunk above to the case of a left-sided test. We only have to adjust the color shading and the tick marks.

```
# plot the the standard normal density on the domain [-4,4]
curve(dnorm(x),
      xlim = c(-4, 4),
      main = "Rejection Region of a Left-Sided Test",
      yaxs = "i",
      xlab = "t-statistic",
      ylab = "",
      lwd = 2,
      axes = "F")

# add x-axis
axis(1,
     at = c(-4, 0, -1.64, 4),
     padj = 0.5,
     labels = c("", 0, expression(Phi^-1~(.05)===-1.64), ""))

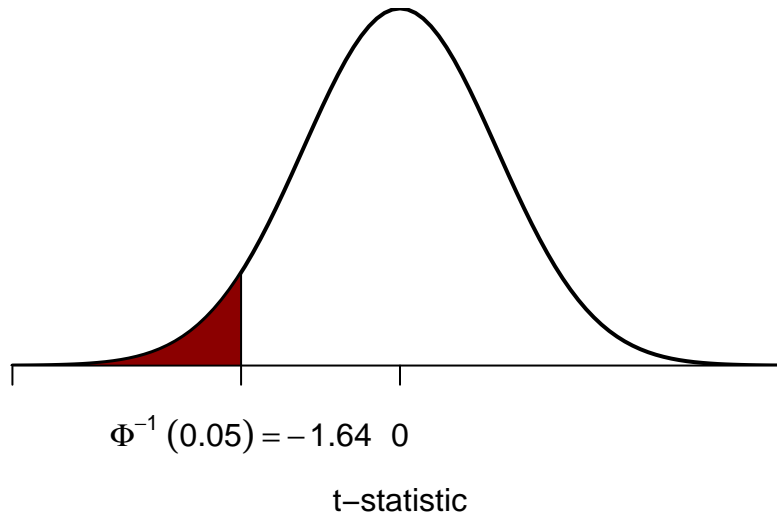
# shade rejection region in right tail
```

```

polygon(x = c(-4, seq(-4, -1.64, 0.01), -1.64),
        y = c(0, dnorm(seq(-4, -1.64, 0.01)), 0),
        col = "darkred")

```

Rejection Region of a Left-Sided Test



3.4 Confidence Intervals for the Population Mean

As mentioned before, we will never estimate the *exact* value of the population mean of Y using a random sample. However, we can compute confidence intervals for the population mean. In general, a confidence interval for an unknown parameter is a recipe that, in repeated samples, yields intervals that contain the true parameter with a prespecified probability, the *confidence level*. Confidence intervals are computed using the information available in the sample. Since this information is the result of a random process, confidence intervals are random variables themselves.

Key Concept 3.7 shows how to compute confidence intervals for the unknown population mean $E(Y)$.

Key Concept 3.7 Confidence Intervals for the Population Mean

A 95% confidence interval for μ_Y is a **random variable** that contains the true μ_Y in 95% of all possible random samples. When n is large we can use the normal approximation. Then, 99%, 95%, 90% confidence intervals are

$$99\% \text{ confidence interval for } \mu_Y = [\bar{Y} \pm 2.58 \times SE(\bar{Y})], \quad (3.9)$$

$$95\% \text{ confidence interval for } \mu_Y = [\bar{Y} \pm 1.96 \times SE(\bar{Y})], \quad (3.10)$$

$$90\% \text{ confidence interval for } \mu_Y = [\bar{Y} \pm 1.64 \times SE(\bar{Y})]. \quad (3.11)$$

These confidence intervals are sets of null hypotheses we cannot reject in a two-sided hypothesis test at the given level of confidence.

Now consider the following statements.

1. In repeated sampling, the interval

$$[\bar{Y} \pm 1.96 \times SE(\bar{Y})]$$

covers the true value of μ_Y with a probability of 95%.

2. We have computed $\bar{Y} = 5.1$ and $SE(\bar{Y}) = 2.5$ so the interval

$$[5.1 \pm 1.96 \times 2.5] = [0.2, 10]$$

covers the true value of μ_Y with a probability of 95%.

While 1. is right (this is in line with the definition above), 2. is wrong and none of your lecturers wants to read such a sentence in a term paper, written exam or similar, believe us. The difference is that, while 1. is the definition of a random variable, 2. is one possible *outcome* of this random variable so there is no meaning in making any probabilistic statement about it. Either the computed interval *does cover* μ_Y or it *does not*!

In R, testing of hypotheses about the mean of a population on the basis of a random sample is very easy due to functions like `t.test()` from the `stats` package. It produces an object of type `list`. Luckily, one of the most simple ways to use `t.test()` is when you want to obtain a 95% confidence interval for some population mean. We start by generating some random data and calling `t.test()` in conjunction with `ls()` to obtain a breakdown of the output components.

```
# set seed
set.seed(1)

# generate some sample data
sampledata <- rnorm(100, 10, 10)

# check the type of the outcome produced by t.test
typeof(t.test(sampledata))
#> [1] "list"

# display the list elements produced by t.test
```

```
ls(t.test(sampledata))
#> [1] "alternative" "conf.int"    "data.name"    "estimate"    "method"
#> [6] "null.value"   "p.value"     "parameter"   "statistic"   "stderr"
```

Though we find that many items are reported, at the moment we are only interested in computing a 95% confidence set for the mean.

```
t.test(sampledata)$"conf.int"
#> [1] 9.306651 12.871096
#> attr(,"conf.level")
#> [1] 0.95
```

This tells us that the 95% confidence interval is

$$[9.31, 12.87].$$

In this example, the computed interval obviously does cover the true μ_Y which we know to be 10.

Let us have a look at the whole standard output produced by `t.test()`.

```
t.test(sampledata)
#>
#> One Sample t-test
#>
#> data:  sampledata
#> t = 12.346, df = 99, p-value < 2.2e-16
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  9.306651 12.871096
#> sample estimates:
#> mean of x
#> 11.08887
```

We see that `t.test()` does not only compute a 95% confidence interval but automatically conducts a two-sided significance test of the hypothesis $H_0 : \mu_Y = 0$ at the level of 5% and reports relevant parameters thereof: the alternative hypothesis, the estimated mean, the resulting t -statistic, the degrees of freedom of the underlying t -distribution (note that `t.test()` performs the normal approximation) and the corresponding p -value. This is very convenient!

In this example, we come to the conclusion that the population mean is significantly different from 0 (which is correct) at the level of 5%, since $\mu_Y = 0$ is not an element of the 95% confidence interval

$$0 \notin [9.31, 12.87].$$

We come to an equivalent result when using the p -value rejection rule since

$$p\text{-value} = 2.2 \cdot 10^{-16} \ll 0.05.$$

3.5 Comparing Means from Different Populations

Suppose you are interested in the means of two different populations, denote them μ_1 and μ_2 . More specifically, you are interested in whether these population means are different from each other and plan

to use a hypothesis test to verify this on the basis of independent sample data from both populations. A suitable pair of hypotheses is

$$H_0 : \mu_1 - \mu_2 = d_0 \quad \text{vs.} \quad H_1 : \mu_1 - \mu_2 \neq d_0, \quad (3.12)$$

where d_0 denotes the hypothesized difference in means (so $d_0 = 0$ when the means are equal, under the null hypothesis). The book teaches us that H_0 can be tested with the t -statistic

$$t = \frac{(\bar{Y}_1 - \bar{Y}_2) - d_0}{SE(\bar{Y}_1 - \bar{Y}_2)} \quad (3.13)$$

where

$$SE(\bar{Y}_1 - \bar{Y}_2) = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}. \quad (3.14)$$

This is called a two sample t -test. For large n_1 and n_2 , (3.13) is standard normal under the null hypothesis. Analogously to the simple t -test we can compute confidence intervals for the true difference in population means:

$$(\bar{Y}_1 - \bar{Y}_2) \pm 1.96 \times SE(\bar{Y}_1 - \bar{Y}_2)$$

is a 95% confidence interval for d . In R, hypotheses as in (3.12) can be tested with `t.test()`, too. Note that `t.test()` chooses $d_0 = 0$ by default. This can be changed by setting the argument `mu` accordingly.

The subsequent code chunk demonstrates how to perform a two sample t -test in R using simulated data.

```
# set random seed
set.seed(1)

# draw data from two different populations with equal mean
sample_pop1 <- rnorm(100, 10, 10)
sample_pop2 <- rnorm(100, 10, 20)

# perform a two sample t-test
t.test(sample_pop1, sample_pop2)
#>
#> Welch Two Sample t-test
#>
#> data: sample_pop1 and sample_pop2
#> t = 0.872, df = 140.52, p-value = 0.3847
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#> -2.338012 6.028083
#> sample estimates:
#> mean of x mean of y
#> 11.088874 9.243838
```

We find that the two sample t -test does not reject the (true) null hypothesis that $d_0 = 0$.

3.6 An Application to the Gender Gap of Earnings

This section discusses how to reproduce the results presented in the box *The Gender Gap of Earnings of College Graduates in the United States* of the book.

In order to reproduce Table 3.1 of the book you need to download the replication data which are hosted by Pearson and can be downloaded here. This file contains data that range from 1992 to 2008 and earnings are reported in prices of 2008.

There are several ways to import the `.xlsx`-files into R. Our suggestion is the function `read_excel()` from the `readxl` package (Wickham and Bryan, 2023). The package is not a part of R's base version and has to be installed manually.

```
# load the 'readxl' package
library(readxl)
```

You are now ready to import the dataset. Make sure you use the correct path to import the downloaded file! In our example, the file is saved in a subfolder of the working directory named `data`. If you are not sure what your current working directory is, use `getwd()`, see also `?getwd`. This will give you the path that points to the place R is currently looking for files to work with.

```
# import the data into R
cps <- read_excel(path = "data/cps_ch3.xlsx")
```

Next, install and load the package `dplyr` (Wickham et al., 2023a). This package provides some handy functions that simplify data wrangling a lot. It makes use of the `%>%` operator.

```
# load the 'dplyr' package
library(dplyr)
```

First, get an overview over the dataset. Next, use `%>%` and some functions from the `dplyr` package to group the observations by gender and year and compute descriptive statistics for both groups.

```
# get an overview of the data structure
head(cps)
#> # A tibble: 6 x 3
#>   a_sex year ahe08
#>   <dbl> <dbl> <dbl>
#> 1     1  1992  17.2
#> 2     1  1992  15.3
#> 3     1  1992  22.9
#> 4     2  1992  13.3
#> 5     1  1992  22.1
#> 6     2  1992  12.2

# group data by gender and year and compute the mean, standard deviation
# and number of observations for each group
avgs <- cps %>%
  group_by(a_sex, year) %>%
  summarise(mean(ahe08),
            sd(ahe08),
            n())

# print the results to the console
print(avgs)
#> # A tibble: 10 x 5
```

```
#> # Groups:   a_sex [2]
#>   a_sex year `mean(ahe08)` `sd(ahe08)` `n()`
#>   <dbl> <dbl>      <dbl>      <dbl> <int>
#> 1     1  1992      23.3      10.2  1594
#> 2     1  1996      22.5      10.1  1379
#> 3     1  2000      24.9      11.6  1303
#> 4     1  2004      25.1      12.0  1894
#> 5     1  2008      25.0      11.8  1838
#> 6     2  1992      20.0       7.87  1368
#> 7     2  1996      19.0       7.95  1230
#> 8     2  2000      20.7       9.36  1181
#> 9     2  2004      21.0       9.36  1735
#> 10    2  2008      20.9       9.66  1871
```

With the pipe operator `%>`, we simply chain different R functions that produce compatible input and output. In the code above, we take the dataset `cps` and use it as an input for the function `group_by()`. The output of `group_by` is subsequently used as an input for `summarise()` and so forth.

Now that we have computed the statistics of interest for both genders, we can investigate how the gap in earnings between both groups evolves over time.

```
# split the dataset by gender
male <- avgs %>% dplyr::filter(a_sex == 1)

female <- avgs %>% dplyr::filter(a_sex == 2)

# rename columns of both splits
colnames(male) <- c("Sex", "Year", "Y_bar_m", "s_m", "n_m")
colnames(female) <- c("Sex", "Year", "Y_bar_f", "s_f", "n_f")

# estimate gender gaps
gap <- male$Y_bar_m - female$Y_bar_f

# compute standard errors
gap_se <- sqrt(male$s_m^2 / male$n_m + female$s_f^2 / female$n_f)

# Compute confidence intervals for all dates
gap_ci_l <- gap - 1.96 * gap_se

gap_ci_u <- gap + 1.96 * gap_se

result <- cbind(male[, -1], female[, -(1:2)], gap, gap_se, gap_ci_l, gap_ci_u)

# print the results to the console
print(result, digits = 3)
#>   Year Y_bar_m s_m n_m Y_bar_f s_f n_f gap gap_se gap_ci_l gap_ci_u
#> 1 1992   23.3 10.2 1594   20.0 7.87 1368 3.23 0.332   2.58   3.88
#> 2 1996   22.5 10.1 1379   19.0 7.95 1230 3.49 0.354   2.80   4.19
#> 3 2000   24.9 11.6 1303   20.7 9.36 1181 4.14 0.421   3.32   4.97
#> 4 2004   25.1 12.0 1894   21.0 9.36 1735 4.10 0.356   3.40   4.80
#> 5 2008   25.0 11.8 1838   20.9 9.66 1871 4.10 0.354   3.41   4.80
```

We observe virtually the same results as the ones presented in the book. The computed statistics

suggest that there is a gender gap in earnings. Note that we can reject the null hypothesis that the gap is zero for all periods. Further, estimates of the gap and bounds of the 95% confidence intervals indicate that the gap has been quite stable in the recent past.

3.7 Scatterplots, Sample Covariance and Sample Correlation

A scatter plot represents two dimensional data, for example n observations on X_i and Y_i , by points in a coordinate system. It is very easy to generate scatter plots using the `plot()` function in R. Let us generate some artificial data on age and earnings of workers and plot it.

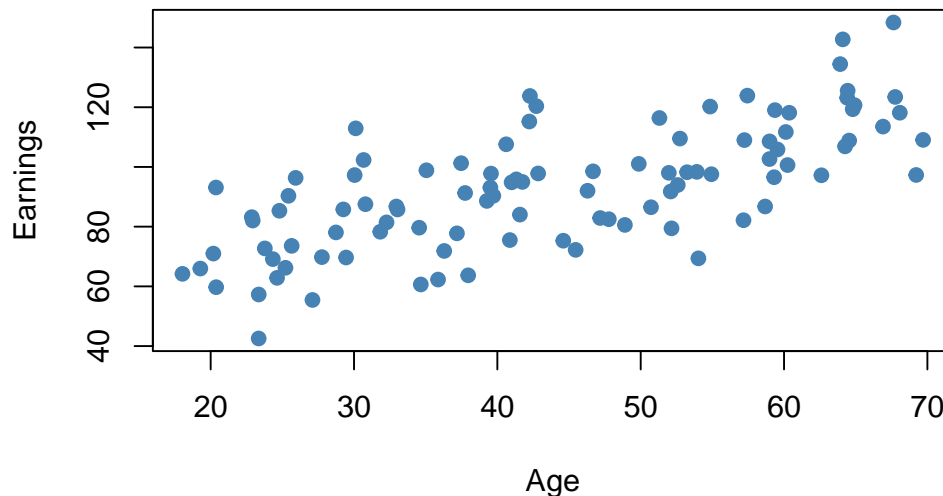
```
# set random seed
set.seed(123)

# generate dataset
X <- runif(n = 100,
           min = 18,
           max = 70)

Y <- X + rnorm(n=100, 50, 15)

# plot observations
plot(X,
     Y,
     type = "p",
     main = "A Scatterplot of X and Y",
     xlab = "Age",
     ylab = "Earnings",
     col = "steelblue",
     pch = 19)
```

A Scatterplot of X and Y



The plot shows positive correlation between age and earnings. This is in line with the notion that older workers earn more than those who joined the working population recently.

Sample Covariance and Correlation

By now you should be familiar with the concepts of variance and covariance. If not, we recommend you to work your way through Chapter 2 of the book.

Just like the variance, covariance and correlation of two variables are properties that relate to the (unknown) joint probability distribution of these variables. We can estimate covariance and correlation by means of suitable estimators using a sample (X_i, Y_i) , $i = 1, \dots, n$.

The sample covariance

$$s_{XY} = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})$$

is an estimator for the population variance of X and Y whereas the sample correlation

$$r_{XY} = \frac{s_{XY}}{s_X s_Y}$$

can be used to estimate the population correlation, a standardized measure for the strength of the linear relationship between X and Y . See Chapter 3.7 in the book for a more detailed treatment of these estimators.

As for variance and standard deviation, these estimators are implemented as R functions in the **stats** package. We can use them to estimate population covariance and population correlation of the artificial data on age and earnings.

```
# compute sample covariance of X and Y
cov(X, Y)
#> [1] 213.934

# compute sample correlation between X and Y
cor(X, Y)
#> [1] 0.706372

# an equivalent way to compute the sample correlation
cov(X, Y) / (sd(X) * sd(Y))
#> [1] 0.706372
```

The estimates indicate that X and Y are moderately correlated.

The next code chunk uses the function `mvrnorm()` from package **MASS** (Ripley, 2023) to generate bivariate sample data with different degrees of correlation.

```
library(MASS)

# set random seed
set.seed(1)

# positive correlation (0.81)
example1 <- mvrnorm(100,
  mu = c(0, 0),
  Sigma = matrix(c(2, 2, 2, 3), ncol = 2),
  empirical = TRUE)

# negative correlation (-0.81)
```

```
example2 <- mvrnorm(100,
                   mu = c(0, 0),
                   Sigma = matrix(c(2, -2, -2, 3), ncol = 2),
                   empirical = TRUE)

# no correlation
example3 <- mvrnorm(100,
                   mu = c(0, 0),
                   Sigma = matrix(c(1, 0, 0, 1), ncol = 2),
                   empirical = TRUE)

# no correlation (quadratic relationship)
X <- seq(-3, 3, 0.01)
Y <- - X^2 + rnorm(length(X))

example4 <- cbind(X, Y)

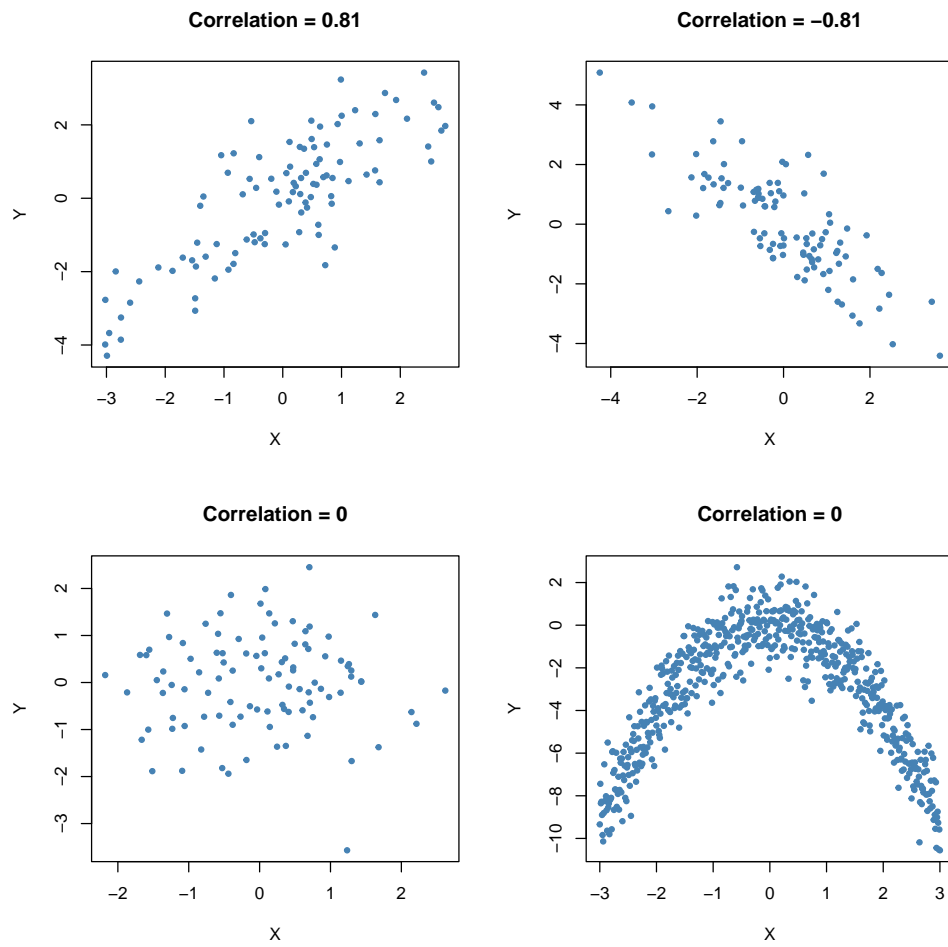
# divide plot area as 2-by-2 array
par(mfrow = c(2, 2))

# plot datasets
plot(example1, col = "steelblue", pch = 20, xlab = "X", ylab = "Y",
     main = "Correlation = 0.81")

plot(example2, col = "steelblue", pch = 20, xlab = "X", ylab = "Y",
     main = "Correlation = -0.81")

plot(example3, col = "steelblue", pch = 20, xlab = "X", ylab = "Y",
     main = "Correlation = 0")

plot(example4, col = "steelblue", pch = 20, xlab = "X", ylab = "Y",
     main = "Correlation = 0")
```



3.8 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 4

Linear Regression with One Regressor

This chapter introduces the basics in linear regression and shows how to perform regression analysis in **R**. In linear regression, the aim is to model the relationship between a dependent variable Y and one or more explanatory variables denoted by X_1, X_2, \dots, X_k . Following the book we will focus on the concept of simple linear regression throughout the whole chapter. In simple linear regression, there is just one explanatory variable X_1 . If, for example, a school cuts its class sizes by hiring new teachers, that is, the school lowers X_1 , the student-teacher ratios of its classes, how would this affect Y , the performance of the students involved in a standardized test? With linear regression we can not only examine whether the student-teacher ratio *does have* an impact on the test results but we can also learn about the *direction* and the *strength* of this effect.

The following packages are needed for reproducing the code presented in this chapter:

- **AER** - accompanies the Book *Applied Econometrics with R* Kleiber and Zeileis (2008a) and provides useful functions and data sets.
- **MASS** - a collection of functions for applied statistics.

Make sure these are installed before you go ahead and try to replicate the examples. The safest way to do so is by checking whether the following code chunk executes without any errors.

```
library(AER)
library(MASS)
```

4.1 Simple Linear Regression

To start with an easy example, consider the following combinations of average test score and the average student-teacher ratio in some fictional school districts.

To work with these data in **R** we begin by generating two vectors: one for the student-teacher ratios

	1	2	3	4	5	6	7
TestScore	680	640	670	660	630	660.0	635
STR	15	17	19	20	22	23.5	25

(STR) and one for test scores (TestScore), both containing the data from the table above.

```
# Create sample data
STR <- c(15, 17, 19, 20, 22, 23.5, 25)
TestScore <- c(680, 640, 670, 660, 630, 660, 635)

# Print out sample data
STR
#> [1] 15.0 17.0 19.0 20.0 22.0 23.5 25.0
TestScore
#> [1] 680 640 670 660 630 660 635
```

To build simple linear regression model, we hypothesize that the relationship between dependent and independent variable is linear, formally:

$$Y = b \cdot X + a.$$

For now, let us suppose that the function which relates test score and student-teacher ratio to each other is

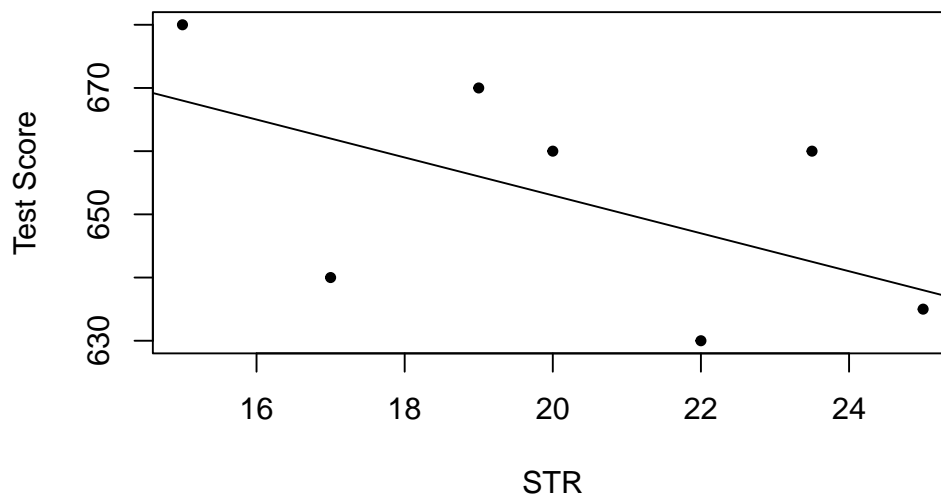
$$TestScore = 713 - 3 \times STR.$$

It is always a good idea to visualize the data you work with. Here, it is suitable to use `plot()` to produce a scatterplot with STR on the x -axis and TestScore on the y -axis. Just call `plot(y_variable ~ x_variable)` whereby `y_variable` and `x_variable` are placeholders for the vectors of observations we want to plot. Furthermore, we might want to add a systematic relationship to the plot. To draw a straight line, R provides the function `abline()`. We just have to call this function with arguments `a` (representing the intercept) and `b` (representing the slope) after executing `plot()` in order to add the line to our plot.

The following code reproduces Figure 4.1 from the textbook.

```
# create a scatterplot of the data
plot(TestScore ~ STR, ylab="Test Score", pch=20)

# add the systematic relationship to the plot
abline(a = 713, b = -3)
```



We find that the line does not touch any of the points although we claimed that it represents the systematic relationship. The reason for this is randomness. Most of the time there are additional influences which imply that there is no bivariate relationship between the two variables.

In order to account for these differences between observed data and the systematic relationship, we extend our model from above by an *error term* u which captures additional random effects. Put differently, u accounts for all the differences between the regression line and the actual observed data. Beside pure randomness, these deviations could also arise from measurement errors or, as will be discussed later, could be the consequence of leaving out other factors that are relevant in explaining the dependent variable.

Which other factors are plausible in our example? For one thing, the test scores might be driven by the teachers' quality and the background of the students. It is also possible that in some classes, the students were lucky on the test days and thus achieved higher scores. For now, we will summarize such influences by an additive component:

$$\text{TestScore} = \beta_0 + \beta_1 \times \text{STR} + \text{other factors.}$$

Of course this idea is very general as it can be easily extended to other situations that can be described with a linear model. Hence, the basic linear regression model we will work with is

$$Y_i = \beta_0 + \beta_1 X_i + u_i.$$

Key Concept 4.1 summarizes the linear regression model and its terminology.

Key Concept 4.1

Terminology for the Linear Regression Model with a Single Regressor

The linear regression model is

$$Y_i = \beta_0 + \beta_1 X_i + u_i,$$

where

- the index i runs over the observations, $i = 1, \dots, n$;
- Y_i is the *dependent variable*, the *regressand*, or simply the *left-hand variable*;
- X_i is the *independent variable*, the *regressor*, or simply the *right-hand variable*;
- $Y = \beta_0 + \beta_1 X$ is the *population regression line* also called the *population regression function*;
- β_0 is the *intercept* of the population regression line;
- β_1 is the *slope* of the population regression line;
- u_i is the *error term*.

4.2 Estimating the Coefficients of the Linear Regression Model

In practice, the intercept β_0 and slope β_1 of the population regression line are unknown. Therefore, we must employ data to estimate both unknown parameters. In the following, a real world example will be used to demonstrate how this is achieved. We want to relate test scores to student-teacher ratios measured in Californian schools. The test score is the district-wide average of reading and math scores for fifth graders. Again, the class size is measured as the number of students divided by the number of teachers (the student-teacher ratio). As for the data, the California School data set (`CASchools`) comes with an R package called `AER`, an acronym for Applied Econometrics with R (Kleiber and Zeileis, 2008b). After installing the package with `install.packages("AER")` and attaching it with `library(AER)` the data set can be loaded using the function `data()`.

```
## # install the AER package (once)
## install.packages("AER")
```

```
##
## # load the AER package
library(AER)

# load the the data set in the workspace
data(CASchools)
```

Once a package has been installed it is available for use at further occasions when invoked with `library()` — there is no need to run `install.packages()` again!

It is interesting to know what kind of object we are dealing with. `class()` returns the class of an object. Depending on the class of an object some functions (for example `plot()` and `summary()`) behave differently.

Let us check the class of the object `CASchools`.

```
class(CASchools)
#> [1] "data.frame"
```

It turns out that `CASchools` is of class `data.frame` which is a convenient format to work with, especially for performing regression analysis.

With help of `head()` we get a first overview of our data. This function shows only the first 6 rows of the data set which prevents an overcrowded console output.

Press `ctrl + L` to clear the console. This command deletes any code that has been typed in and executed by you or printed to the console by R functions. The good news is that anything else is left untouched. You neither lose defined variables etc. nor the code history. It is still possible to recall previously executed R commands using the up and down keys. If you are working in *RStudio*, press `ctrl + Up` on your keyboard (`CMD + Up` on a Mac) to review a list of previously entered commands.

```
head(CASchools)
#>   district          school county grades students teachers
#> 1   75119      Sunol Glen Unified Alameda KK-08      195    10.90
#> 2   61499      Manzanita Elementary  Butte KK-08      240    11.15
#> 3   61549      Thermalito Union Elementary Butte KK-08    1550    82.90
#> 4   61457 Golden Feather Union Elementary Butte KK-08      243    14.00
#> 5   61523      Palermo Union Elementary  Butte KK-08    1335    71.50
#> 6   62042      Burrel Union Elementary  Fresno KK-08      137     6.40
#>   calworks  lunch  computer expenditure  income  english  read  math
#> 1  0.5102  2.0408     67  6384.911 22.690001 0.000000 691.6 690.0
#> 2 15.4167 47.9167    101  5099.381  9.824000 4.583333 660.5 661.9
#> 3 55.0323 76.3226    169  5501.955  8.978000 30.000002 636.3 650.9
#> 4 36.4754 77.0492     85  7101.831  8.978000 0.000000 651.9 643.5
#> 5 33.1086 78.4270    171  5235.988  9.080333 13.857677 641.8 639.9
#> 6 12.3188 86.9565     25  5580.147 10.415000 12.408759 605.7 605.4
```

We find that the data set consists of plenty of variables and that most of them are numeric.

By the way: an alternative to `class()` and `head()` is `str()` which is deduced from ‘structure’ and gives a comprehensive overview of the object. Try!

Turning back to `CASchools`, the two variables we are interested in (i.e., average test score and the

student-teacher ratio) are *not* included. However, it is possible to calculate both from the provided data. To obtain the student-teacher ratios, we simply divide the number of students by the number of teachers. The average test score is the arithmetic mean of the test score for reading and the score of the math test. The next code chunk shows how the two variables can be constructed as vectors and how they are appended to `CASchools`.

```
# compute STR and append it to CASchools
CASchools$STR <- CASchools$students/CASchools$teachers

# compute TestScore and append it to CASchools
CASchools$score <- (CASchools$read + CASchools$math)/2
```

If we ran `head(CASchools)` again we would find the two variables of interest as additional columns named `STR` and `score` (check this!).

Table 4.1 from the textbook summarizes the distribution of test scores and student-teacher ratios. There are several functions which can be used to produce similar results, e.g.,

- `mean()` (computes the arithmetic mean of the provided numbers),
- `sd()` (computes the sample standard deviation),
- `quantile()` (returns a vector of the specified sample quantiles for the data).

The next code chunk shows how to achieve this. First, we compute summary statistics on the columns `STR` and `score` of `CASchools`. In order to get nice output we gather the measures in a `data.frame` named `DistributionSummary`.

```
# compute sample averages of STR and score
avg_STR <- mean(CASchools$STR)
avg_score <- mean(CASchools$score)

# compute sample standard deviations of STR and score
sd_STR <- sd(CASchools$STR)
sd_score <- sd(CASchools$score)

# set up a vector of percentiles and compute the quantiles
quantiles <- c(0.10, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9)
quant_STR <- quantile(CASchools$STR, quantiles)
quant_score <- quantile(CASchools$score, quantiles)

# gather everything in a data.frame
DistributionSummary <- data.frame(Average = c(avg_STR, avg_score),
                                  StandardDeviation = c(sd_STR, sd_score),
                                  quantile = rbind(quant_STR, quant_score))

# print the summary to the console
DistributionSummary
#>           Average StandardDeviation quantile.10. quantile.25. quantile.40.
#> quant_STR    19.64043           1.891812    17.3486    18.58236    19.26618
#> quant_score  654.15655           19.053347    630.3950    640.05000    649.06999
#>           quantile.50. quantile.60. quantile.75. quantile.90.
#> quant_STR     19.72321           20.0783    20.87181    21.86741
#> quant_score   654.45000           659.4000    666.66249    678.85999
```

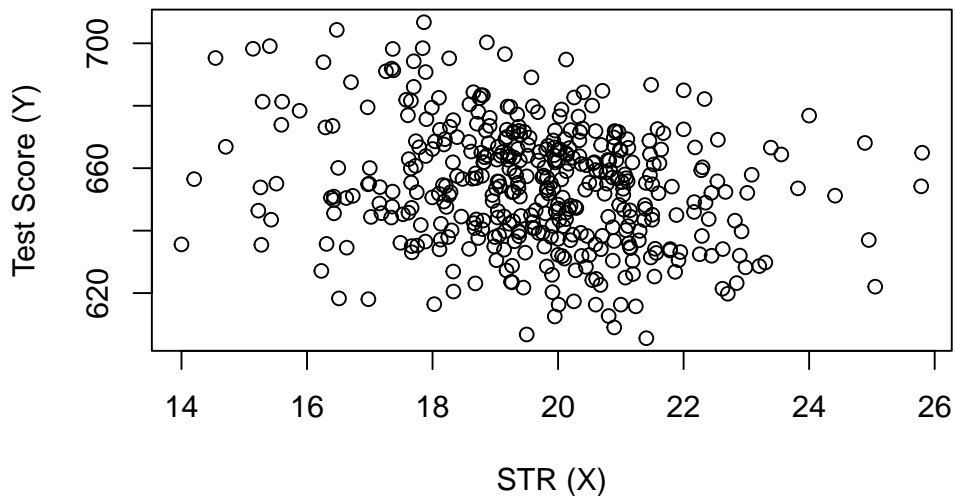
As for the sample data, we use `plot()`. This allows us to detect characteristics of our data, such as

outliers which are harder to discover by looking at mere numbers. This time we add some additional arguments to the call of `plot()`.

The first argument in our call of `plot()`, `score ~ STR`, is again a formula that states variables on the y- and the x-axis. However, this time the two variables are not saved in separate vectors but are columns of `CASchools`. Therefore, R would not find them without the argument `data` being correctly specified. `data` must be in accordance with the name of the `data.frame` to which the variables belong to, in this case `CASchools`. Further arguments are used to change the appearance of the plot: `main` adds a title, `xlab` and `ylab` add custom labels to both axes.

```
plot(score ~ STR,
      data = CASchools,
      main = "Scatterplot of Test Score and STR",
      xlab = "STR (X)",
      ylab = "Test Score (Y)")
```

Scatterplot of Test Score and STR



The plot (Figure 4.2 in the book) shows the scatterplot of all observations on the student-teacher ratio and test score. We see that the points are strongly scattered, and that the variables are negatively correlated. That is, we expect to observe lower test scores in bigger classes.

The function `cor()` (see `?cor` for further info) can be used to compute the correlation between two *numeric* vectors.

```
cor(CASchools$STR, CASchools$score)
#> [1] -0.2263627
```

As the scatterplot already suggests, the correlation is negative but rather weak.

The task we are currently facing is to find a line that best fits the data. We could opt for graphical inspection and correlation analysis and then select the best fitting line by eyeballing. However, this would be rather subjective: different observers would draw different regression lines. On this account, we are interested in techniques that are less arbitrary. Such a technique is given by ordinary least squares (OLS) estimation.

The Ordinary Least Squares Estimator

The OLS estimator chooses the regression coefficients such that the estimated regression line is as “close” as possible to the observed data points. Here, closeness is measured by the sum of the squared mistakes made in predicting Y given X . Let b_0 and b_1 be some estimators of β_0 and β_1 . Then the sum of squared estimation mistakes can be expressed as

$$\sum_{i=1}^n (Y_i - b_0 - b_1 X_i)^2.$$

The OLS estimator in the simple regression model is the pair of estimators for intercept and slope that minimizes the expression above. The derivation of the OLS estimators for both parameters are presented in Appendix 4.1 of the book. The results are summarized in Key Concept 4.2.

Key Concept 4.2

The OLS Estimator, Predicted Values, and Residuals

The OLS estimators of the slope β_1 and the intercept β_0 in the simple linear regression model are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2},$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}.$$

The OLS predicted values \hat{Y}_i and residuals \hat{u}_i are

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i,$$

$$\hat{u}_i = Y_i - \hat{Y}_i.$$

The estimated intercept $\hat{\beta}_0$, the slope parameter $\hat{\beta}_1$ and the residuals (\hat{u}_i) are computed from a sample of n observations of X_i and Y_i , $i=1, \dots, n$. These are *estimates* of the unknown true population intercept (β_0), slope (β_1), and error term (u_i).

There are many possible ways to compute $\hat{\beta}_0$ and $\hat{\beta}_1$ in R. For example, we could implement the formulas presented in Key Concept 4.2 with two of R’s most basic functions: `mean()` and `sum()`. Before doing so we *attach* the `CASchools` dataset.

```
attach(CASchools) # allows to use the variables contained in CASchools directly

# compute beta_1_hat
beta_1 <- sum((STR - mean(STR)) * (score - mean(score))) / sum((STR - mean(STR))^2)

# compute beta_0_hat
beta_0 <- mean(score) - beta_1 * mean(STR)

# print the results to the console
beta_1
#> [1] -2.279808
beta_0
#> [1] 698.9329
```

Calling `attach(CASchools)` enables us to address a variable contained in `CASchools` by its name: it is no longer necessary to use the `$` operator in conjunction with the dataset: R may evaluate the variable name directly.

R uses the object in the user environment if this object shares the name of variable contained in an attached database. However, it is a better practice to always use distinctive names in order to avoid such (seeming) ambivalences!

Notice that we address variables contained in the attached dataset `CASchools` directly for the rest of this chapter!

Of course, there are even more manual ways to perform these tasks. With OLS being one of the most widely-used estimation techniques, R of course already contains a built-in function named `lm()` (linear model) which can be used to carry out regression analysis.

The first argument of the function to be specified is, similar to `plot()`, the regression formula with the basic syntax `y ~ x` where `y` is the dependent variable and `x` the explanatory variable. The argument `data` determines the data set to be used in the regression. We now revisit the example from the book where the relationship between the test scores and the class sizes is analyzed. The following code uses `lm()` to replicate the results presented in figure 4.3 of the book.

```
# estimate the model and assign the result to linear_model
linear_model <- lm(score ~ STR, data = CASchools)

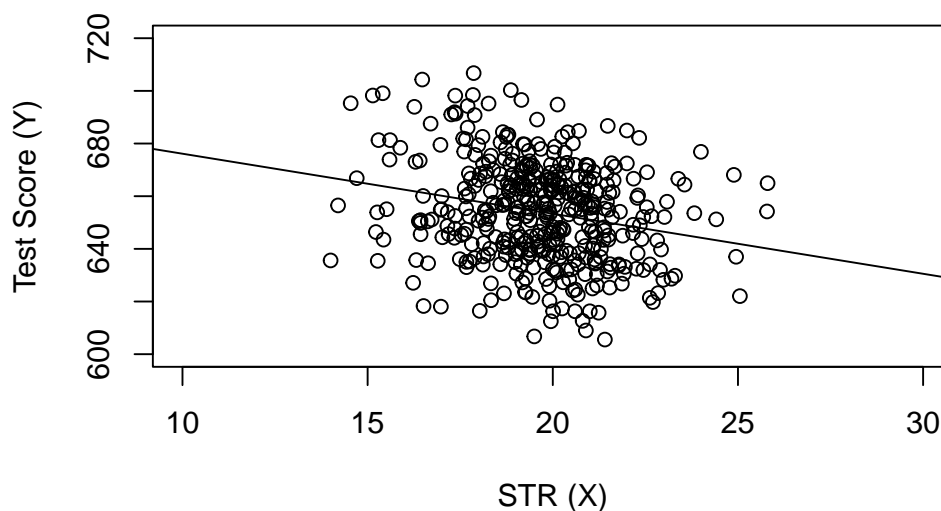
# print the standard output of the estimated lm object to the console
linear_model
#>
#> Call:
#> lm(formula = score ~ STR, data = CASchools)
#>
#> Coefficients:
#> (Intercept)          STR
#>    698.93         -2.28
```

Let us add the estimated regression line to the plot. This time we also enlarge the ranges of both axes by setting the arguments `xlim` and `ylim`.

```
# plot the data
plot(score ~ STR,
      data = CASchools,
      main = "Scatterplot of Test Score and STR",
      xlab = "STR (X)",
      ylab = "Test Score (Y)",
      xlim = c(10, 30),
      ylim = c(600, 720))

# add the regression line
abline(linear_model)
```

Scatterplot of Test Score and STR



Did you notice that this time, we did not pass the intercept and slope parameters to `abline`? If you call `abline()` on an object of class `lm` which only contains a single regressor, R draws the regression line automatically!

4.3 Measures of Fit

After fitting a linear regression model, a natural question is how well the model describes the data. Visually, this amounts to assessing whether the observations are tightly clustered around the regression line. Both the *coefficient of determination* and the *standard error of the regression* measure how well the OLS Regression line fits the data.

The Coefficient of Determination

R^2 , the *coefficient of determination*, is the fraction of the sample variance of Y_i that is explained by X_i . Mathematically, the R^2 can be written as the ratio of the explained sum of squares to the total sum of squares. The *explained sum of squares (ESS)* is the sum of squared deviations of the predicted values \hat{Y}_i , from the average of the Y_i . The *total sum of squares (TSS)* is the sum of squared deviations of the Y_i from their average. Thus we have

$$ESS = \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2, \quad (4.1)$$

$$TSS = \sum_{i=1}^n (Y_i - \bar{Y})^2, \quad (4.2)$$

$$R^2 = \frac{ESS}{TSS}. \quad (4.3)$$

Since $TSS = ESS + SSR$ we can also write

$$R^2 = 1 - \frac{SSR}{TSS},$$

where SSR is the sum of squared residuals, a measure for the errors made when predicting Y by X . The SSR is defined as

$$SSR = \sum_{i=1}^n \hat{u}_i^2.$$

R^2 lies between 0 and 1. It is easy to see that a perfect fit, i.e., no errors made when fitting the regression line, implies $R^2 = 1$ then we have $SSR = 0$. On the contrary, if our estimated regression line does not explain any variation in the Y_i , we have $ESS = 0$ and consequently $R^2 = 0$.

The Standard Error of the Regression

The *Standard Error of the Regression* (SER) is an estimator of the standard deviation of the residuals \hat{u}_i . As such it measures the magnitude of a typical deviation from the regression line, i.e., the magnitude of a typical residual.

$$SER = s_{\hat{u}} = \sqrt{s_{\hat{u}}^2} \quad \text{where} \quad s_{\hat{u}}^2 = \frac{1}{n-2} \sum_{i=1}^n \hat{u}_i^2 = \frac{SSR}{n-2}$$

Remember that the u_i are *unobserved*. This is why we use their estimated counterparts, the residuals \hat{u}_i , instead. See Chapter 4.3 of the book for a more detailed comment on the SER .

Application to the Test Score Data

Both measures of fit can be obtained by using the function `summary()` with an `lm` object provided as the only argument. While the function `lm()` only prints out the estimated coefficients to the console, `summary()` provides additional predefined information such as the regression's R^2 and the SER .

```
mod_summary <- summary(linear_model)
mod_summary
#>
#> Call:
#> lm(formula = score ~ STR, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -47.727 -14.251   0.483  12.822  48.540
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  698.9329     9.4675   73.825 < 2e-16 ***
#> STR          -2.2798     0.4798  -4.751 2.78e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 18.58 on 418 degrees of freedom
#> Multiple R-squared:  0.05124,    Adjusted R-squared:  0.04897
#> F-statistic: 22.58 on 1 and 418 DF,  p-value: 2.783e-06
```

The R^2 in the output is called *Multiple R-squared* and has a value of 0.051. Hence, 5.1% of the variance of the dependent variable *score* is explained by the explanatory variable *STR*. That is, the regression

explains little of the variance in *score*, and much of the variation in test scores remains unexplained (cf. Figure 4.3 of the book).

The *SER* is called *Residual standard error* and equals 18.58. The unit of the *SER* is the same as the unit of the dependent variable. That is, on average the deviation of the actual achieved test score and the regression line is 18.58 points.

Now, let us check whether `summary()` uses the same definitions for R^2 and *SER* as we do when computing them manually.

```
# compute R^2 manually
SSR <- sum(mod_summary$residuals^2)
TSS <- sum((score - mean(score))^2)
R2 <- 1 - SSR/TSS

# print the value to the console
R2
#> [1] 0.05124009

# compute SER manually
n <- nrow(CASchools)
SER <- sqrt(SSR / (n-2))

# print the value to the console
SER
#> [1] 18.58097
```

We find that the results coincide. Note that the values provided by `summary()` are rounded to two decimal places.

4.4 The Least Squares Assumptions

OLS performs well under a quite broad variety of different circumstances. However, there are some assumptions which need to be satisfied in order to ensure that the estimates are normally distributed in large samples (we discuss this in Chapter 4.5).

Key Concept 4.3 The Least Squares Assumptions

$$Y_i = \beta_0 + \beta_1 X_i + u_i, i = 1, \dots, n$$

where

1. The error term u_i has conditional mean zero given X_i : $E(u_i|X_i) = 0$.
2. $(X_i, Y_i), i = 1, \dots, n$ are independent and identically distributed (i.i.d.) draws from their joint distribution.
3. Large outliers are unlikely: X_i and Y_i have nonzero finite fourth moments.

Assumption 1: The Error Term has Conditional Mean of Zero

This means that no matter which value we choose for X , the error term u must not show any systematic pattern and must have a mean of 0. Consider the case that, unconditionally, $E(u) = 0$, but for low and high values of X , the error term tends to be positive and for midrange values of X the error tends to

be negative. We can use R to construct such an example. To do so we generate our own data using R's built-in random number generators.

We will use the following functions:

- `runif()` - generates uniformly distributed random numbers.
- `rnorm()` - generates normally distributed random numbers.
- `predict()` - does predictions based on the results of model fitting functions like `lm()`.
- `lines()` - adds line segments to an existing plot.

We start by creating a vector containing values that are uniformly distributed on the interval $[-5, 5]$. This can be done with the function `runif()`. We also need to simulate the error term. For this we generate normally distributed random numbers with a mean of 0 and a variance of 1 using `rnorm()`. The Y values are obtained as a quadratic function of the X values and the error.

After generating the data we estimate both a simple regression model and a quadratic model that also includes the regressor X^2 (this is a multiple regression model, see Chapter 6). Finally, we plot the simulated data and add the estimated regression line of a simple regression model as well as the predictions made with a quadratic model to compare the fit graphically.

```
# set a seed to make the results reproducible
set.seed(321)

# simulate the data
X <- runif(50, min = -5, max = 5)
u <- rnorm(50, sd = 1)

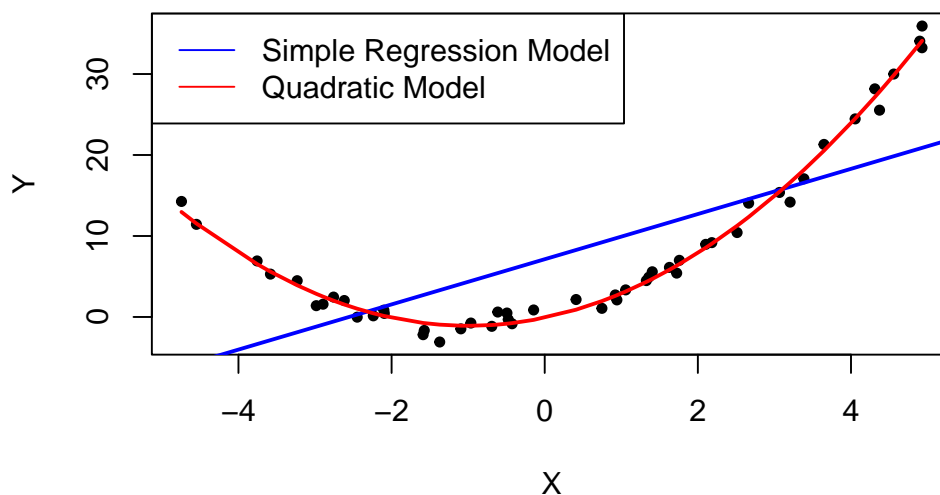
# the true relation
Y <- X^2 + 2 * X + u

# estimate a simple regression model
mod_simple <- lm(Y ~ X)

# estimate a quadratic regression model
mod_quadratic <- lm( Y ~ X + I(X^2))

# predict using a quadratic model
prediction <- predict(mod_quadratic, data.frame(X = sort(X)))

# plot the results
plot( Y ~ X, col = "black", pch = 20, xlab = "X", ylab = "Y")
abline( mod_simple, col = "blue",lwd=2)
#red line = incorrect linear regression (this violates the first OLS assumption)
lines( sort(X), prediction,col="red",lwd=2)
legend("topleft",
      legend = c("Simple Regression Model",
                 "Quadratic Model"),
      cex = 1,
      lty = 1,
      col = c("blue","red"))
```



The plot above shows what is meant by $E(u_i|X_i) = 0$ and why it does not hold for the linear model:

Using the quadratic model (represented by the black curve) we see that there are no systematic deviations of the observation from the predicted relation. It is credible that the assumption is not violated when such a model is employed. However, using a simple linear regression model we see that the assumption is probably violated as $E(u_i|X_i)$ varies with the X_i .

Assumption 2: Independently and Identically Distributed Data

Most sampling schemes used when collecting data from populations produce i.i.d.-samples. For example, we can use R's random number generator to randomly select student IDs from a university's enrollment list and record age X and earnings Y of the corresponding students. This is a typical example of simple random sampling and ensures that all the (X_i, Y_i) are drawn randomly from the same population.

A prominent example where the i.i.d. assumption is not fulfilled is time series data where we have observations on the same unit over time. For example, take X as the number of workers in a production company over time. Due to business transformations, the company cuts jobs periodically by a specific share but there are also some non-deterministic influences that relate to economics, politics etc. Using R we can easily simulate such a process and plot it.

We start the series with a total of 5000 workers and simulate the reduction of employment with an autoregressive process that exhibits a downward movement in the long-run and has normally distributed errors:¹

$$employment_t = -50 + 0.98 \cdot employment_{t-1} + u_t.$$

```
# set seed
set.seed(123)

# generate a date vector
Date <- seq(as.Date("1951/1/1"), as.Date("2000/1/1"), "years")

# initialize the employment vector
X <- c(5000, rep(NA, length(Date)-1))

# generate time series observations with random influences
```

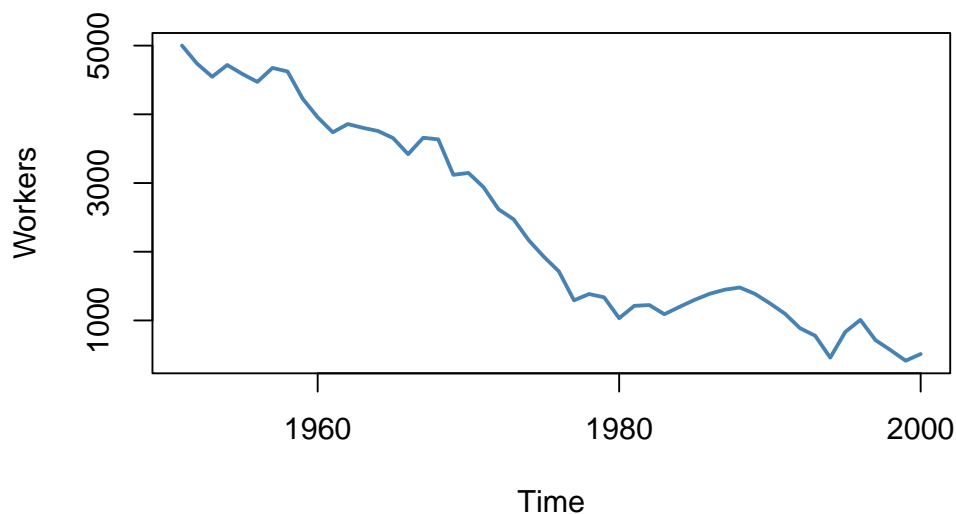
¹See Chapter 14 for more on autoregressive processes and time series analysis in general.


```

for (t in 2:length(Date)) {
  X[t] <- -50 + 0.98 * X[t-1] + rnorm(n = 1, sd = 200)
}

#plot the results
plot(x = Date,
     y = X,
     type = "l",
     col = "steelblue",
     ylab = "Workers",
     xlab = "Time",
     lwd=2)

```



It is evident that the observations on the number of employees cannot be independent in this example: the level of today's employment is correlated with tomorrow's employment level. Thus, the i.i.d. assumption is violated.

Assumption 3: Large Outliers are Unlikely

It is easy to come up with situations where extreme observations, i.e., observations that deviate considerably from the usual range of the data, may occur. Such observations are called outliers. Technically speaking, assumption 3 requires that X and Y have a finite kurtosis.²

Common cases where we want to exclude or (if possible) correct such outliers is when they are apparently typos, conversion errors or measurement errors. Even if it seems like extreme observations have been recorded correctly, it is advisable to exclude them before estimating a model since OLS suffers from *sensitivity to outliers*.

What does this mean? One can show that extreme observations receive heavy weighting in the estimation of the unknown regression coefficients when using OLS. Therefore, outliers can lead to strongly distorted estimates of regression coefficients. To get a better impression of this issue, consider the following application where we have placed some sample data on X and Y which are highly correlated. The

²See Chapter 4.4 of the book.

relation between X and Y seems to be explained pretty well by the plotted regression line: all of the white data points lie close to the red regression line and we have $R^2 = 0.92$.

Now go ahead and add a further observation at, say, (18.2). This observation clearly is an outlier. The result is quite striking: the estimated regression line differs greatly from the one we adjudged to fit the data well. The slope is heavily downward biased and R^2 decreased to a mere 29%! Double-click inside the coordinate system to reset the app. Feel free to experiment. Choose different coordinates for the outlier or add additional ones.

The following code roughly reproduces what is shown in figure 4.5 in the book. As done above we use sample data generated using R's random number functions `rnorm()` and `runif()`. We estimate two simple regression models, one based on the original data set and another using a modified set where one observation is changed to be an outlier and then plot the results. In order to understand the complete code you should be familiar with the function `sort()` which sorts the entries of a numeric vector in ascending order.

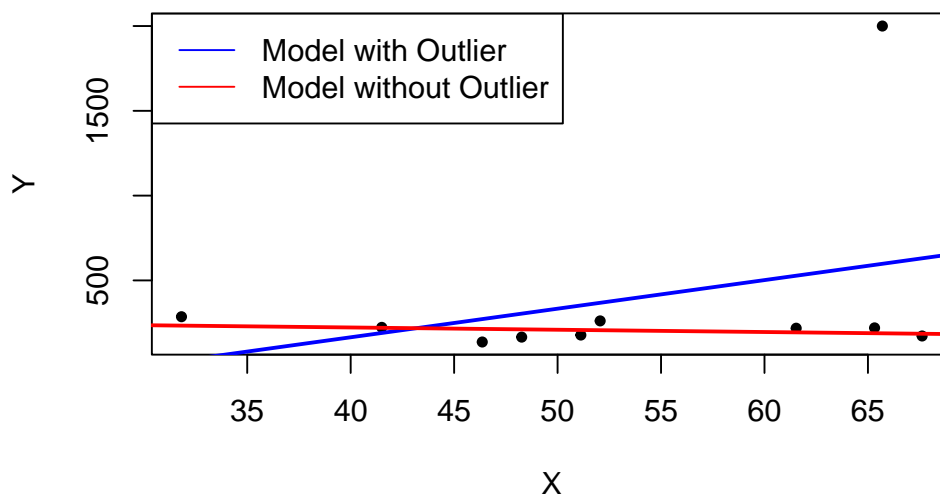
```
# set seed
set.seed(123)

# generate the data
X <- sort(runif(10, min = 30, max = 70))
Y <- rnorm(10, mean = 200, sd = 50)
Y[9] <- 2000

# fit model with outlier
fit <- lm(Y ~ X)

# fit model without outlier
fitWithoutOutlier <- lm(Y[-9] ~ X[-9])

# plot the results
plot(Y ~ X, pch=20)
abline(fit, lwd=2, col="blue")
abline(fitWithoutOutlier, col = "red", lwd=2)
legend("topleft",
      legend = c("Model with Outlier",
                 "Model without Outlier"),
      cex = 1,
      lty = 1,
      col = c("blue", "red"))
```



4.5 The Sampling Distribution of the OLS Estimator

Because $\hat{\beta}_0$ and $\hat{\beta}_1$ are computed from a sample, the estimators themselves are random variables with a probability distribution — the so-called sampling distribution of the estimators — which describes the values they could take on over different samples. Although the sampling distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$ can be complicated when the sample size is small and generally changes with the number of observations, n , it is possible, provided the assumptions discussed in the book are valid, to make certain statements about it that hold for all n . In particular

$$E(\hat{\beta}_0) = \beta_0 \quad \text{and} \quad E(\hat{\beta}_1) = \beta_1,$$

that is, $\hat{\beta}_0$ and $\hat{\beta}_1$ are unbiased estimators of β_0 and β_1 , the true parameters. If the sample is sufficiently large, by the central limit theorem the *joint* sampling distribution of the estimators is well approximated by the bivariate normal distribution (2.1). This implies that the marginal distributions are also normal in large samples. Core facts on the large-sample distributions of $\hat{\beta}_0$ and $\hat{\beta}_1$ are presented in Key Concept 4.4.

Key Concept 4.4**Large Sample Distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$**

If the least squares assumptions in Key Concept 4.3 hold, then in large samples $\hat{\beta}_0$ and $\hat{\beta}_1$ have a joint normal sampling distribution. The large sample normal distribution of $\hat{\beta}_1$ is $\mathcal{N}(\beta_1, \sigma_{\hat{\beta}_1}^2)$, where the variance of the distribution, $\sigma_{\hat{\beta}_1}^2$, is

$$\sigma_{\hat{\beta}_1}^2 = \frac{1}{n} \frac{\text{Var}[(X_i - \mu_X) u_i]}{[\text{Var}(X_i)]^2}. \quad (4.4)$$

The large sample normal distribution of $\hat{\beta}_0$ is $\mathcal{N}(\beta_0, \sigma_{\hat{\beta}_0}^2)$ with

$$\sigma_{\hat{\beta}_0}^2 = \frac{1}{n} \frac{\text{Var}(H_i u_i)}{[E(H_i^2)]^2}, \text{ where } H_i = 1 - \left[\frac{\mu_X}{E(X_i^2)} \right] X_i. \quad (4.5)$$

The interactive simulation below continuously generates random samples (X_i, Y_i) of 200 observations where $E(Y|X) = 100 + 3X$, estimates a simple regression model, stores the estimate of the slope β_1 and visualizes the distribution of the $\hat{\beta}_1$ s observed so far using a histogram. The idea here is that for a large number of $\hat{\beta}_1$ s, the histogram gives a good approximation of the sampling distribution of the estimator. By decreasing the time between two sampling iterations, it becomes clear that the shape of the histogram approaches the characteristic bell shape of a normal distribution centered at the true slope of 3.

This interactive part of the book is only available in the HTML version.

Simulation Study 1

Whether the statements of Key Concept 4.4 really hold can also be verified using R. For this we first build our own population of 100000 observations in total. To do this we need values for the independent variable X , for the error term u , and for the parameters β_0 and β_1 . With these combined in a simple regression model, we compute the dependent variable Y . In our example we generate the numbers X_i , $i = 1, \dots, 100000$ by drawing a random sample from a uniform distribution on the interval $[0, 20]$. The realizations of the error terms u_i are drawn from a standard normal distribution with parameters $\mu = 0$ and $\sigma^2 = 100$ (note that `rnorm()` requires σ as input for the argument `sd`, see `?rnorm`). Furthermore we chose $\beta_0 = -2$ and $\beta_1 = 3.5$ so the true model is

$$Y_i = -2 + 3.5 \cdot X_i.$$

Finally, we store the results in a data.frame.

```
# simulate data
N <- 100000
X <- runif(N, min = 0, max = 20)
u <- rnorm(N, sd = 10)

# population regression
Y <- -2 + 3.5 * X + u
population <- data.frame(X, Y)
```

From now on we will consider the previously generated data as the true population (which of course would be *unknown* in a real world application, otherwise there would be no reason to draw a random sample in the first place). The knowledge about the true population and the true relationship between

Y and X can be used to verify the statements made in Key Concept 4.4.

First, let us calculate the true variances $\sigma_{\hat{\beta}_0}^2$ and $\sigma_{\hat{\beta}_1}^2$ for a randomly drawn sample of size $n = 100$.

```
# set sample size
n <- 100

# compute the variance of beta_hat_0
H_i <- 1 - mean(X) / mean(X^2) * X
var_b0 <- var(H_i * u) / (n * mean(H_i^2)^2)

# compute the variance of hat_beta_1
var_b1 <- var( ( X - mean(X) ) * u ) / (n * var(X)^2)

# print variances to the console
var_b0
#> [1] 4.045066
var_b1
#> [1] 0.03018694
```

Now let us assume that we do not know the true values of β_0 and β_1 and that it is not possible to observe the whole population. However, we can observe a random sample of n observations. Then, it would not be possible to compute the true parameters but we could obtain estimates of β_0 and β_1 from the sample data using OLS. However, we know that these estimates are outcomes of random variables themselves since the observations are randomly sampled from the population. Key Concept 4.4 describes their distributions for large n . When drawing a single sample of size n it is not possible to make any statement about these distributions. Things change if we repeat the sampling scheme many times and compute the estimates for each sample: using this procedure we simulate outcomes of the respective distributions.

To achieve this in R, we employ the following approach:

- We assign the number of repetitions, say 10000, to `reps` and then initialize a matrix `fit` where the estimates obtained in each sampling iteration shall be stored row-wise. Thus `fit` has to be a matrix of dimensions `reps`×2.
- In the next step we draw `reps` random samples of size `n` from the population and obtain the OLS estimates for each sample. The results are stored as row entries in the outcome matrix `fit`. This is done using a `for()` loop.
- At last, we estimate variances of both estimators using the sampled outcomes and plot histograms of the latter. We also add a plot of the density functions belonging to the distributions that follow from Key Concept 4.4. The function `bquote()` is used to obtain math expressions in the titles and labels of both plots. See `?bquote`.

```
# set repetitions and sample size
n <- 100
reps <- 10000

# initialize the matrix of outcomes
fit <- matrix(ncol = 2, nrow = reps)

# loop sampling and estimation of the coefficients
for (i in 1:reps){

  sample <- population[sample(1:N, n), ]
  fit[i, ] <- lm(Y ~ X, data = sample)$coefficients
```

```
}

# compute variance estimates using outcomes
var(fit[, 1])
#> [1] 4.186832
var(fit[, 2])
#> [1] 0.03096199

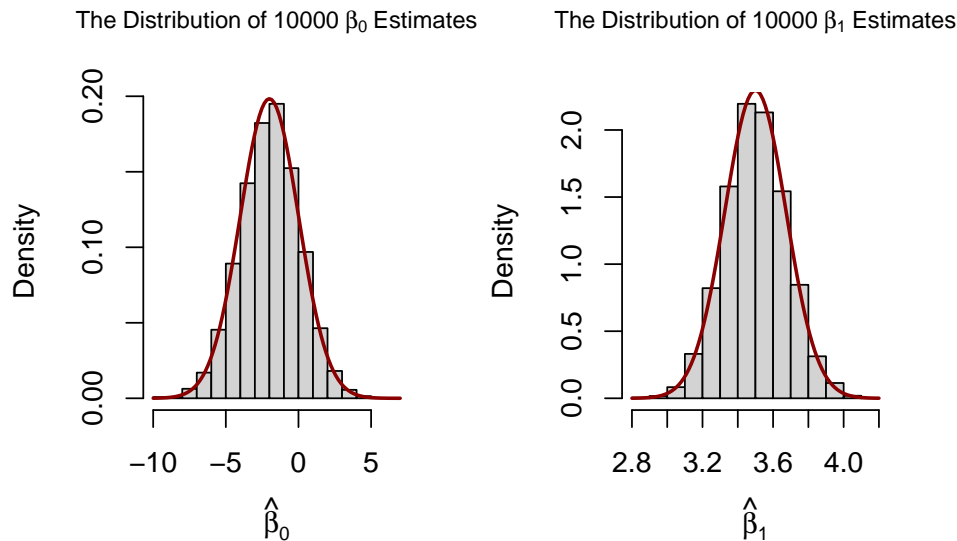
# divide plotting area as 1-by-2 array
par(mfrow = c(1, 2))

# plot histograms of beta_0 estimates
hist(fit[, 1],
     cex.main = 0.8,
     main = bquote(The ~ Distribution ~ of ~ 10000 ~ beta[0] ~ Estimates),
     xlab = bquote(hat(beta)[0]),
     freq = F)

# add true distribution to plot
curve(dnorm(x,
            -2,
            sqrt(var_b0)),
      add = T,
      col = "darkred", lwd=2)

# plot histograms of beta_hat_1
hist(fit[, 2],
     cex.main = 0.8,
     main = bquote(The ~ Distribution ~ of ~ 10000 ~ beta[1] ~ Estimates),
     xlab = bquote(hat(beta)[1]),
     freq = F)

# add true distribution to plot
curve(dnorm(x,
            3.5,
            sqrt(var_b1)),
      add = T,
      col = "darkred", lwd=2)
```



Our variance estimates support the statements made in Key Concept 4.4, coming close to the theoretical values. The histograms suggest that the distributions of the estimators can be well approximated by the respective theoretical normal distributions stated in Key Concept 4.4.

Simulation Study 2

A further result implied by Key Concept 4.4 is that both estimators are consistent, i.e., they converge in probability to the true parameters we are interested in. This is because they are asymptotically unbiased and their variances converge to 0 as n increases. We can check this by repeating the simulation above for a sequence of increasing sample sizes. This means we no longer assign the sample size but a *vector* of sample sizes: `n <- c(...)`. Let us look at the distributions of β_1 . The idea here is to add an additional call for `for()` to the code. This is done in order to loop over the vector of sample sizes n . For each of the sample sizes we carry out the same simulation as before but plot a density estimate for the outcomes of each iteration over n . Notice that we have to change n to $n[j]$ in the inner loop to ensure that the j^{th} element of n is used. In the simulation, we use sample sizes of 100, 250, 1000 and 3000. Consequently we have a total of four distinct simulations using different sample sizes.

```
# set seed for reproducibility
set.seed(1)

# set repetitions and the vector of sample sizes
reps <- 1000
n <- c(100, 250, 1000, 3000)

# initialize the matrix of outcomes
fit <- matrix(ncol = 2, nrow = reps)

# divide the plot panel in a 2-by-2 array
par(mfrow = c(2, 2))

# loop sampling and plotting

# outer loop over n
for (j in 1:length(n)) {
```

```

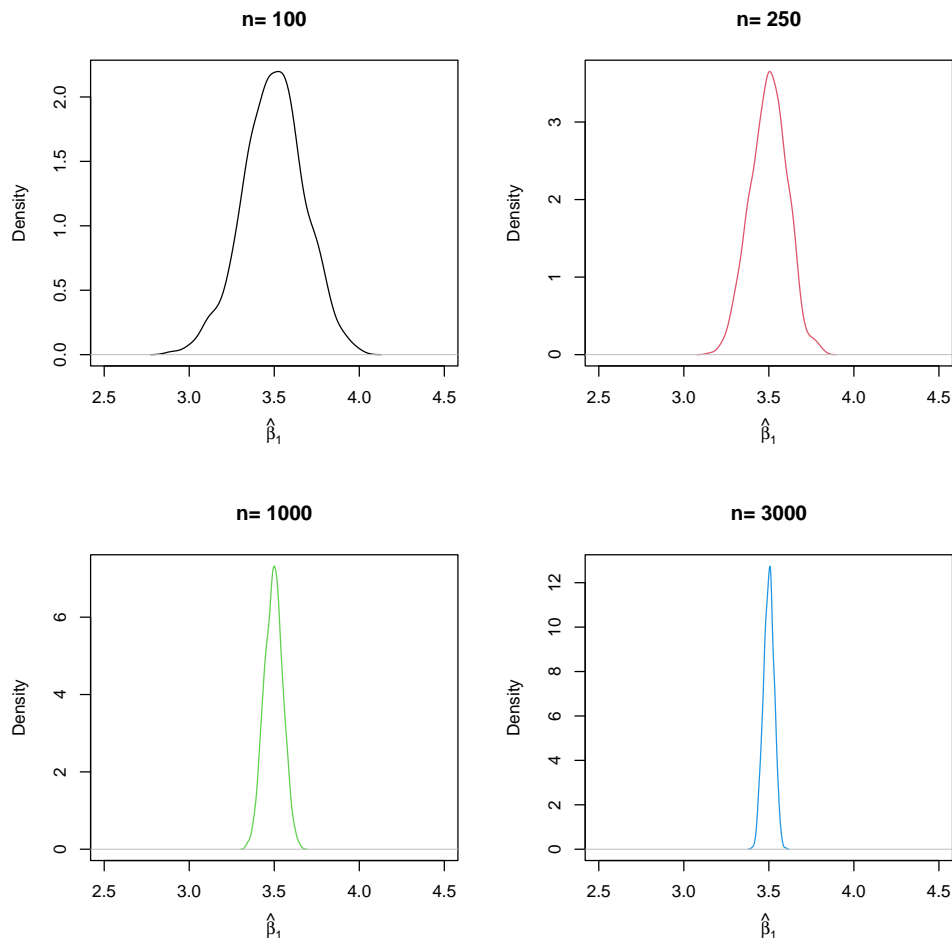
# inner loop: sampling and estimating of the coefficients
for (i in 1:reps){

  sample <- population[sample(1:N, n[j]), ]
  fit[i, ] <- lm(Y ~ X, data = sample)$coefficients

}

# draw density estimates
plot(density(fit[,2]), xlim=c(2.5, 4.5),
     col = j,
     main = paste("n=", n[j]),
     xlab = bquote(hat(beta)[1]))
}

```



We find that, as n increases, the distribution of $\hat{\beta}_1$ concentrates around its mean, i.e., its variance decreases. Put differently, the likelihood of observing estimates close to the true value of $\beta_1 = 3.5$ grows as we increase the sample size. The same behavior can be observed if we analyze the distribution of $\hat{\beta}_0$ instead.

Simulation Study 3

Furthermore, (4.1) reveals that the variance of the OLS estimator for β_1 decreases as the variance of the X_i increases. In other words, as we increase the amount of information provided by the regressor, that is, increasing $Var(X)$, which is used to estimate β_1 , we become more confident that the estimate is close to the true value (i.e., $Var(\hat{\beta}_1)$ decreases). We can visualize this by reproducing Figure 4.6 from the book. To do this, we sample observations (X_i, Y_i) , $i = 1, \dots, 100$ from a bivariate normal distribution with

$$\begin{aligned} E(X) &= E(Y) = 5, \\ Var(X) &= Var(Y) = 5, \end{aligned}$$

and

$$Cov(X, Y) = 4.$$

Formally, this is written down as

$$\begin{pmatrix} X \\ Y \end{pmatrix} \stackrel{i.i.d.}{\sim} \mathcal{N} \left[\begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix} \right]. \quad (4.3)$$

To carry out the random sampling, we make use of the function `mvrnorm()` from the package `MASS` (Ripley, 2023) which allows to draw random samples from multivariate normal distributions, see `?mvtnorm`. Next, we use `subset()` to split the sample into two subsets such that the first set, `set1`, consists of observations that fulfill the condition $|X - \bar{X}| > 1$ and the second set, `set2`, includes the remainder of the sample. We then plot both sets and use different colors to distinguish the observations.

```
# load the MASS package
library(MASS)

# set seed for reproducibility
set.seed(4)

# simulate bivariate normal data
bvndata <- mvrnorm(100,
                  mu = c(5, 5),
                  Sigma = cbind(c(5, 4), c(4, 5)))

# assign column names / convert to data.frame
colnames(bvndata) <- c("X", "Y")
bvndata <- as.data.frame(bvndata)

# subset the data
set1 <- subset(bvndata, abs(mean(X) - X) > 1)
set2 <- subset(bvndata, abs(mean(X) - X) <= 1)

# plot both data sets
plot(set1,
     xlab = "X",
     ylab = "Y",
     pch = 19)

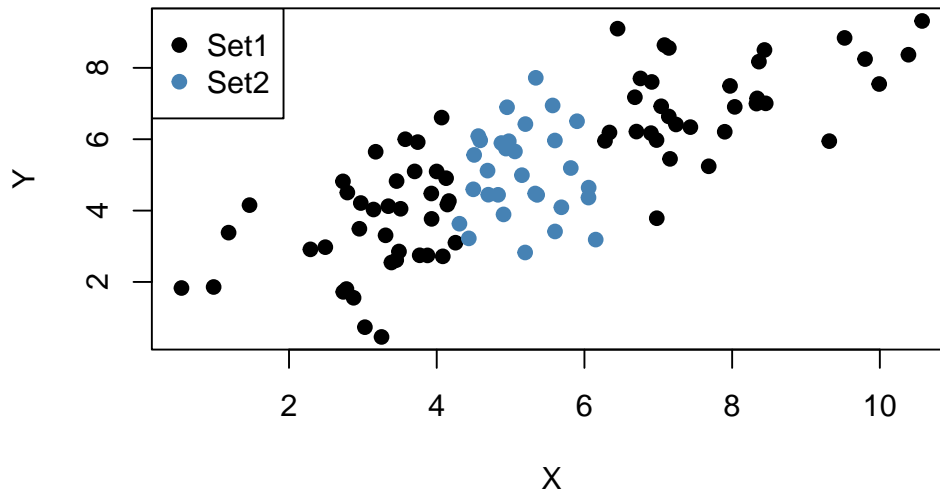
points(set2,
```

```

col = "steelblue",
pch = 19)
legend("topleft",
      legend = c("Set1",
                 "Set2"),

cex = 1,
pch = 19,
col = c("black","steelblue"))

```



It is clear that observations that are close to the sample average of the X_i have less variance than those that are farther away. Now, if we were to draw a line as accurately as possible through either of the two sets it is intuitive that choosing the observations indicated by the black dots, i.e., using the set of observations which has larger variance than the blue ones, would result in a more precise line. Now, let us use OLS to estimate slope and intercept for both sets of observations. We then plot the observations along with both regression lines.

```

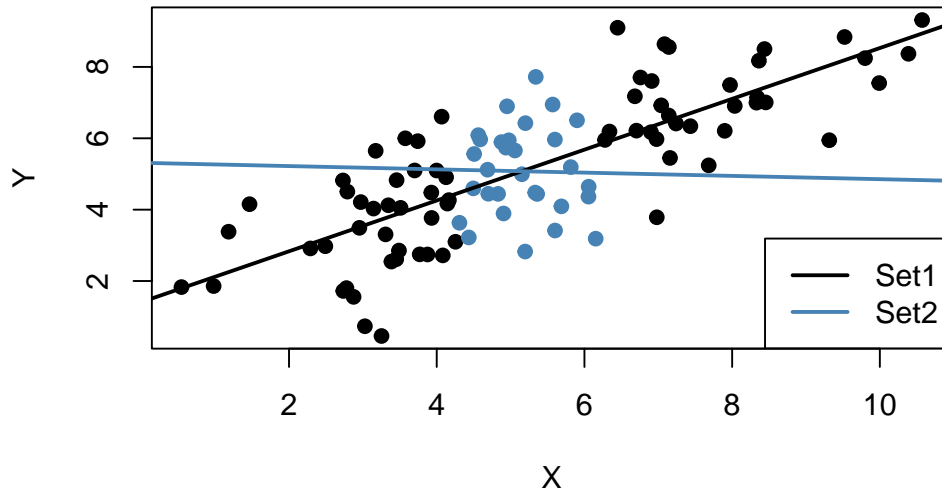
# estimate both regression lines
lm.set1 <- lm(Y ~ X, data = set1)
lm.set2 <- lm(Y ~ X, data = set2)

# plot observations
plot(set1, xlab = "X", ylab = "Y", pch = 19)
points(set2, col = "steelblue", pch = 19)

# add both lines to the plot
abline(lm.set1, col = "black", lwd=2)
abline(lm.set2, col = "steelblue", lwd=2)
legend("bottomright",
      legend = c("Set1",
                 "Set2"),

cex = 1,
lwd=2,
col = c("black","steelblue"))

```



Evidently, the green regression line does far better in describing data sampled from the bivariate normal distribution stated in (4.3) than the red line. This is a nice example for demonstrating why we are interested in a high variance of the regressor X : more variance in the X_i means more information from which the precision of the estimation benefits.

4.6 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 5

Hypothesis Tests and Confidence Intervals in SLR Model

This chapter continues our treatment of the simple linear regression model. The following subsections discuss how we may use our knowledge about the sampling distribution of the OLS estimator in order to make statements regarding its uncertainty.

These subsections cover the following topics:

- Testing Hypotheses regarding regression coefficients.
- Confidence intervals for regression coefficients.
- Regression when X is a dummy variable.
- Heteroskedasticity and Homoskedasticity.

The packages `AER` (Kleiber and Zeileis, 2008b) and `scales` (Wickham and Seidel, 2022) are required for reproduction of the code chunks presented throughout this chapter. The package `scales` provides additional generic plot scaling methods. Make sure both packages are installed before you proceed. The safest way to do so is by checking whether the following code chunk executes without any errors.

```
library(AER)
library(scales)
```

5.1 Testing Two-Sided Hypotheses concerning the Slope Coefficient

Using the fact that $\hat{\beta}_1$ is approximately normally distributed in large samples (see Key Concept 4.4), testing hypotheses about the true value β_1 can be done as in Chapter 3.2.

Key Concept 5.1 General Form of the t -Statistic

Remember from Chapter 3 that a general t -statistic has the form

$$t = \frac{\text{estimated value} - \text{hypothesized value}}{\text{standard error of the estimator}}.$$

Key Concept 5.2 Testing Hypotheses regarding β_1

For testing the hypothesis $H_0 : \beta_1 = \beta_{1,0}$, we need to perform the following steps:

1. Compute the standard error of $\hat{\beta}_1$, $SE(\hat{\beta}_1)$

$$SE(\hat{\beta}_1) = \sqrt{\hat{\sigma}_{\hat{\beta}_1}^2}, \quad \hat{\sigma}_{\hat{\beta}_1}^2 = \frac{1}{n} \times \frac{\frac{1}{n-2} \sum_{i=1}^n (X_i - \bar{X})^2 \hat{u}_i^2}{\left[\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2\right]^2}.$$

2. Compute the t -statistic

$$t = \frac{\hat{\beta}_1 - \beta_{1,0}}{SE(\hat{\beta}_1)}.$$

3. Given a two sided alternative ($H_1 : \beta_1 \neq \beta_{1,0}$) we reject at the 5% level if $|t^{act}| > 1.96$ or, equivalently, if the p -value is less than 0.05.

Recall the definition of the p -value:

$$\begin{aligned} p\text{-value} &= \Pr_{H_0} \left[\left| \frac{\hat{\beta}_1 - \beta_{1,0}}{SE(\hat{\beta}_1)} \right| > \left| \frac{\hat{\beta}_1^{act} - \beta_{1,0}}{SE(\hat{\beta}_1)} \right| \right] \\ &= \Pr_{H_0} (|t| > |t^{act}|) \\ &= 2 \cdot \Phi(-|t^{act}|). \end{aligned}$$

The last transformation is due to the normal approximation for large samples.

Consider again the OLS regression stored in `linear_model` from Chapter 4 that gave us the regression line

$$\widehat{TestScore} = 698.9 - 2.28 \times STR, \quad R^2 = 0.051, \quad SER = 18.6.$$

(9.47) (0.49)

Copy and execute the following code chunk if the above model object is not available in your working environment.

```
# load the `CASchools` dataset
data(CASchools)

# add student-teacher ratio
CASchools$STR <- CASchools$students/CASchools$teachers

# add average test-score
CASchools$score <- (CASchools$read + CASchools$math)/2

# estimate the model
linear_model <- lm(score ~ STR, data = CASchools)
```

For testing a hypothesis concerning the slope parameter (the coefficient on STR), we need $SE(\hat{\beta}_1)$, the standard error of the respective point estimator. As is common in the literature, standard errors are presented in parentheses below the point estimates.

Key Concept 5.1 reveals that it is rather cumbersome to compute the standard error and thereby the t -statistic by hand. The question you should be asking yourself right now is: can we obtain these

values with minimum effort using R? Yes, we can. Let us first use `summary()` to get a summary on the estimated coefficients in `linear_model`.

Note: Throughout the textbook, robust standard errors are reported. We consider it instructive to keep things simple at the beginning and thus start out with simple examples that do not allow for robust inference. Standard errors that are robust to heteroskedasticity are introduced in Chapter 5.4 where we also demonstrate how they can be computed using R. A discussion of heteroskedasticity-autocorrelation robust standard errors takes place in Chapter 15.

```
# print the summary of the coefficients to the console
summary(linear_model)$coefficients
#>           Estimate Std. Error  t value    Pr(>|t|)
#> (Intercept) 698.932949  9.4674911 73.824516 6.569846e-242
#> STR          -2.279808  0.4798255 -4.751327 2.783308e-06
```

The second column of the coefficients' summary reports $SE(\hat{\beta}_0)$ and $SE(\hat{\beta}_1)$. Also, in the third column `t value`, we find t -statistics t^{act} suitable for tests of the separate hypotheses $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 0$. Furthermore, the output provides us with p -values corresponding to both tests against the two-sided alternatives $H_1 : \beta_0 \neq 0$ respectively, $H_1 : \beta_1 \neq 0$ in the fourth column of the table.

Let us have a closer look at the test of

$$H_0 : \beta_1 = 0 \quad \text{vs.} \quad H_1 : \beta_1 \neq 0.$$

We have

$$t^{act} = \frac{-2.279808 - 0}{0.4798255} \approx -4.75.$$

What does this tell us about the significance of the estimated coefficient? We reject the null hypothesis at the 5% level of significance since $|t^{act}| > 1.96$. That is, the observed test statistic falls into the rejection region as $p\text{-value} = 2.78 \cdot 10^{-6} < 0.05$. We conclude that the coefficient is significantly different from zero. In other words, we reject the hypothesis that the class size *has no influence* on the students test scores at the 5% level.

Note that although the difference is negligible in the present case as we will see later, `summary()` does not perform the normal approximation but calculates p -values using the t -distribution instead. Generally, the degrees of freedom of the assumed t -distribution are determined in the following manner:

$$\text{DF} = n - k - 1,$$

where n is the number of observations used to estimate the model and k is the number of regressors, excluding the intercept. In our case, we have $n = 420$ observations and the only regressor is `STR` so $k = 1$. The simplest way to determine the model degrees of freedom is:

```
# determine residual degrees of freedom
linear_model$df.residual
#> [1] 418
```

Hence, for the assumed sampling distribution of $\hat{\beta}_1$ we have

$$\hat{\beta}_1 \sim t_{418}$$

such that the p -value for a two-sided significance test can be obtained by executing the following code:

```
2 * pt(-4.751327, df = 418)
#> [1] 2.78331e-06
```

The result is very close to the value provided by `summary()`. However since n is sufficiently large one could just as well use the standard normal density to compute the p -value:

```
2 * pnorm(-4.751327)
#> [1] 2.02086e-06
```

The difference is indeed negligible. These findings tell us that, if $H_0 : \beta_1 = 0$ is true and we were to repeat the whole process of gathering observations and estimating the model, observing a $\hat{\beta}_1 \geq |-2.28|$ is very unlikely!

Using R we may visualize how such a statement is made when using the normal approximation. This reflects the principles depicted in figure 5.1 in the book. Do not let the following code chunk deter you: the code is somewhat longer than the usual examples and looks unappealing but there is a lot of repetition since color shadings and annotations are added on both tails of the normal distribution. We recommend to execute the code step by step in order to see how the graph is augmented with the annotations.

```
# Plot the standard normal on the support [-6,6]
t <- seq(-6, 6, 0.01)

plot(x = t,
     y = dnorm(t, 0, 1),
     type = "l",
     col = "steelblue",
     lwd = 2,
     yaxp = "i",
     axes = F,
     ylab = "",
     main = expression("Calculating the p-value of a Two-sided Test when"
                       ~ t^act ~ "=-4.75"),
     cex.lab = 0.7,
     cex.main = 1)

tact <- -4.75

axis(1, at = c(0, -1.96, 1.96, -tact, tact), cex.axis = 0.7)

# Shade the critical regions using polygon():

# critical region in left tail
polygon(x = c(-6, seq(-6, -1.96, 0.01), -1.96),
       y = c(0, dnorm(seq(-6, -1.96, 0.01)), 0),
       col = 'orange')

# critical region in right tail

polygon(x = c(1.96, seq(1.96, 6, 0.01), 6),
       y = c(0, dnorm(seq(1.96, 6, 0.01)), 0),
       col = 'orange')
```

```

# Add arrows and texts indicating critical regions and the p-value
arrows(-3.5, 0.2, -2.5, 0.02, length = 0.1)
arrows(3.5, 0.2, 2.5, 0.02, length = 0.1)

arrows(-5, 0.16, -4.75, 0, length = 0.1)
arrows(5, 0.16, 4.75, 0, length = 0.1)

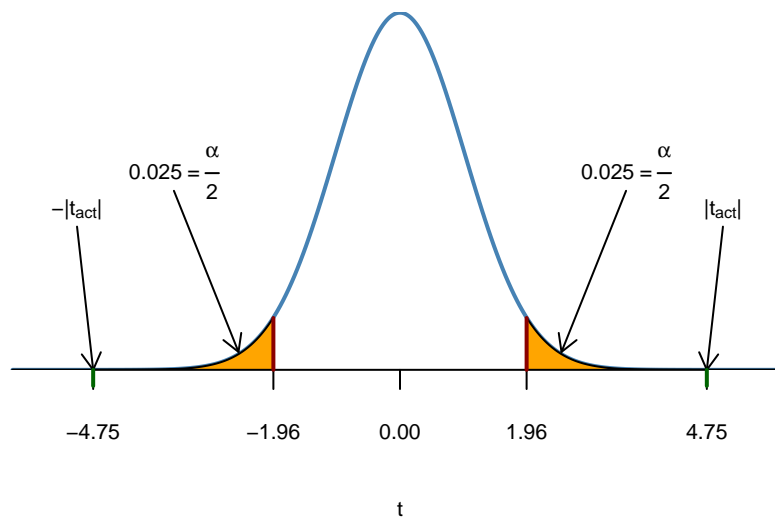
text(-3.5, 0.22,
      labels = expression("0.025"~"="~over(alpha, 2)),
      cex = 0.7)
text(3.5, 0.22,
      labels = expression("0.025"~"="~over(alpha, 2)),
      cex = 0.7)

text(-5, 0.18,
      labels = expression(paste("-",t[act],"|")),
      cex = 0.7)
text(5, 0.18,
      labels = expression(paste("|",t[act],"|")),
      cex = 0.7)

# Add ticks indicating critical values at the 0.05-level, t^act and -t^act
rug(c(-1.96, 1.96), ticksize = 0.145, lwd = 2, col = "darkred")
rug(c(-tact, tact), ticksize = -0.0451, lwd = 2, col = "darkgreen")

```

Calculating the p -value of a Two-sided Test when $t^{\text{act}} = -4.75$



The p -Value is the area under the curve to left of -4.75 plus the area under the curve to the right of 4.75 . As we already know from the calculations above, this value is very small.

5.2 Confidence Intervals for Regression Coefficients

As we already know, estimates of the regression coefficients β_0 and β_1 are subject to sampling uncertainty, see Chapter 4. Therefore, we will *never* exactly estimate the true value of these parameters from sample

data in an empirical application. However, we may construct confidence intervals for the intercept and the slope parameter.

A 95% confidence interval for β_i has two equivalent definitions:

- The interval is the set of values for which a hypothesis test to the level of 5% cannot be rejected.
- The interval has a probability of 95% to contain the true value of β_i . So in 95% of all samples that could be drawn, the confidence interval will cover the true value of β_i .

We also say that the interval has a confidence level of 95%. The idea of the confidence interval is summarized in Key Concept 5.3.

Key Concept 5.3

A Confidence Interval for β_i

Imagine you could draw all possible random samples of given size. The interval that contains the true value β_i in 95% of all samples is given by the expression

$$CI_{0.95}^{\beta_i} = \left[\hat{\beta}_i - 1.96 \times SE(\hat{\beta}_i), \hat{\beta}_i + 1.96 \times SE(\hat{\beta}_i) \right].$$

Equivalently, this interval can be seen as the set of null hypotheses for which a 5% two-sided hypothesis test does not reject.

Simulation Study: Confidence Intervals

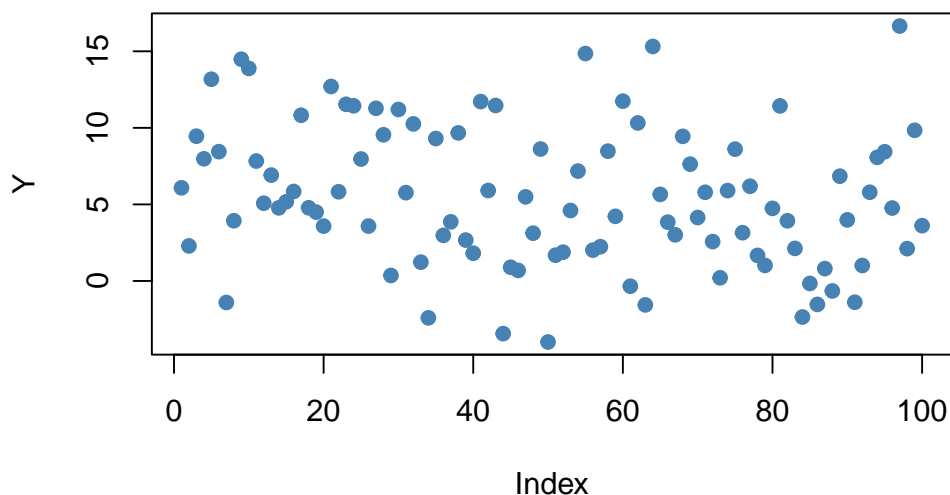
To get a better understanding of confidence intervals we conduct another simulation study. For now, assume that we have the following sample of $n = 100$ observations on a single variable Y where

$$Y_i \stackrel{i.i.d}{\sim} \mathcal{N}(5, 25), \quad i = 1, \dots, 100.$$

```
# set seed for reproducibility
set.seed(4)

# generate and plot the sample data
Y <- rnorm(n = 100,
           mean = 5,
           sd = 5)

plot(Y,
     pch = 19,
     col = "steelblue")
```



We assume that the data is generated by the model

$$Y_i = \mu + \epsilon_i,$$

where μ is an unknown constant and we know that $\epsilon_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 25)$. In this model, the OLS estimator for μ is given by

$$\hat{\mu} = \bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i,$$

i.e., the sample average of the Y_i . It further holds that

$$SE(\hat{\mu}) = \frac{\sigma_\epsilon}{\sqrt{n}} = \frac{5}{\sqrt{100}}$$

(see Chapter 2) a large-sample 95% confidence interval for μ is then given by

$$CI_{0.95}^\mu = \left[\hat{\mu} - 1.96 \times \frac{5}{\sqrt{100}}, \hat{\mu} + 1.96 \times \frac{5}{\sqrt{100}} \right]. \quad (5.1)$$

It is fairly easy to compute this interval in **R** by hand. The following code chunk generates a named vector containing the interval bounds:

```
cbind(CIlower = mean(Y) - 1.96 * 5 / 10, CIupper = mean(Y) + 1.96 * 5 / 10)
#>      CIlower  CIupper
#> [1,] 4.502625 6.462625
```

Knowing that $\mu = 5$ we see that, for our example data, the confidence interval covers the true value.

As opposed to real world examples, we can use **R** to get a better understanding of confidence intervals by repeatedly sampling data, estimating μ and computing the confidence interval for μ as in (5.1).

The procedure is as follows:

- We initialize the vectors **lower** and **upper** in which the simulated interval limits are to be saved. We want to simulate 10000 intervals so both vectors are set to have this length.
- We use a **for()** loop to sample 100 observations from the $\mathcal{N}(5, 25)$ distribution and compute $\hat{\mu}$ as well as the boundaries of the confidence interval in every iteration of the loop.

- At last we join `lower` and `upper` in a matrix.

```
# set seed
set.seed(1)

# initialize vectors of lower and upper interval boundaries
lower <- numeric(10000)
upper <- numeric(10000)

# loop sampling / estimation / CI
for(i in 1:10000) {

  Y <- rnorm(100, mean = 5, sd = 5)
  lower[i] <- mean(Y) - 1.96 * 5 / 10
  upper[i] <- mean(Y) + 1.96 * 5 / 10

}

# join vectors of interval bounds in a matrix
CIs <- cbind(lower, upper)
```

According to Key Concept 5.3 we expect that the fraction of the 10000 simulated intervals saved in the matrix `CIs` that contain the true value $\mu = 5$ should be roughly 95%. We can easily check this using logical operators.

```
mean(CIs[, 1] <= 5 & 5 <= CIs[, 2])
#> [1] 0.9487
```

The simulation shows that the fraction of intervals covering $\mu = 5$, i.e., those intervals for which $H_0 : \mu = 5$ cannot be rejected is close to the theoretical value of 95%.

Let us draw a plot of the first 100 simulated confidence intervals and indicate those which *do not* cover the true value of μ . We do this via horizontal lines representing the confidence intervals on top of each other.

```
# identify intervals not covering mu
# (4 intervals out of 100)
ID <- which(!(CIs[1:100, 1] <= 5 & 5 <= CIs[1:100, 2]))

# initialize the plot
plot(0,
     xlim = c(3, 7),
     ylim = c(1, 100),
     ylab = "Sample",
     xlab = expression(mu),
     main = "Confidence Intervals")

# set up color vector
colors <- rep(gray(0.6), 100)
colors[ID] <- "red"

# draw reference line at mu=5
abline(v = 5, lty = 2)
```

```

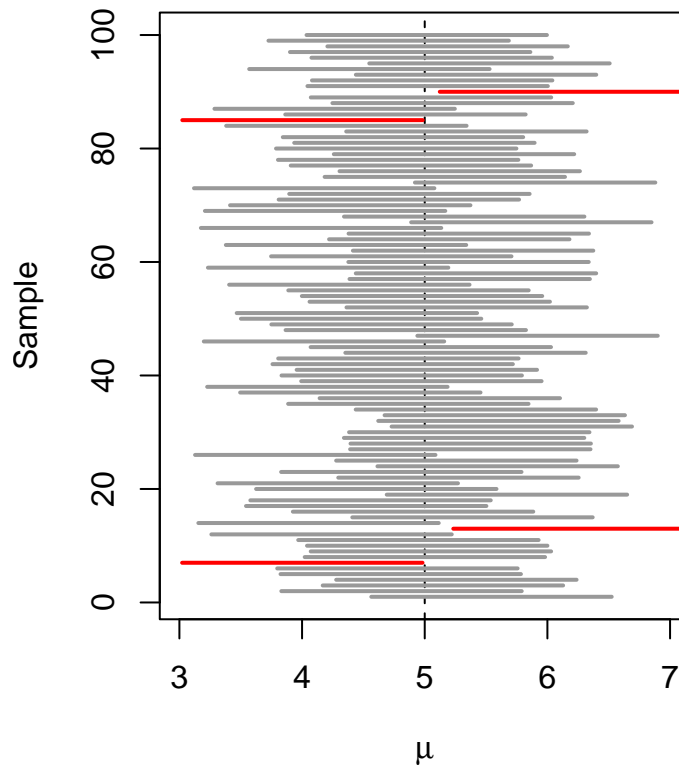
# add horizontal bars representing the CIs
for(j in 1:100) {

  lines(c(CIs[j, 1], CIs[j, 2]),
        c(j, j),
        col = colors[j],
        lwd = 2)

}

```

Confidence Intervals



For the first 100 samples, the true null hypothesis is rejected in four cases so these intervals do not cover $\mu = 5$. We have indicated the intervals which lead to a rejection of the null red.

Let us now come back to the example of test scores and class sizes. The regression model from Chapter 4 is stored in `linear_model`. An easy way to get 95% confidence intervals for β_0 and β_1 , the coefficients on (intercept) and STR, is to use the function `confint()`. We only have to provide a fitted model object as an input to this function. The confidence level is set to 95% by default but can be modified by setting the argument `level`, see `?confint`.

```

# compute 95% confidence interval for coefficients in 'linear_model'
confint(linear_model)
#>           2.5 %      97.5 %
#> (Intercept) 680.32312 717.542775
#> STR         -3.22298 -1.336636

```

Let us check if the calculation is done as we expect it to be for β_1 , the coefficient on STR.

```
# compute 95% confidence interval for coefficients in 'linear_model' by hand
lm_sum <- summary(linear_model)

c("lower" = lm_sum$coef[2,1] - qt(0.975, df = lm_sum$df[2]) * lm_sum$coef[2, 2],
  "upper" = lm_sum$coef[2,1] + qt(0.975, df = lm_sum$df[2]) * lm_sum$coef[2, 2])
#>      lower      upper
#> -3.222980 -1.336636
```

The upper and the lower bounds coincide. We have used the 0.975-quantile of the t_{418} distribution to get the exact result reported by `confint`. Obviously, this interval *does not* contain the value zero which, as we have already seen in the previous section, leads to the rejection of the null hypothesis $\beta_{1,0} = 0$.

5.3 Regression when X is a Binary Variable

Instead of using a continuous regressor X , we might be interested in running the regression

$$Y_i = \beta_0 + \beta_1 D_i + u_i, \quad (5.2)$$

where D_i is a binary variable, a so-called *dummy variable*. For example, we may define D_i as follows:

$$D_i = \begin{cases} 1 & \text{if STR in } i^{\text{th}} \text{ school district} < 20 \\ 0 & \text{if STR in } i^{\text{th}} \text{ school district} \geq 20. \end{cases} \quad (5.3)$$

The regression model now is

$$\text{TestScore}_i = \beta_0 + \beta_1 D_i + u_i. \quad (5.4)$$

Let us see how these data look like in a scatter plot:

```
# Create the dummy variable as defined above
CASchools$D <- CASchools$STR < 20

# Compute the average score when D=1 (low STR)
mean.score.for.D.1 <- mean(CASchools$score[CASchools$D == TRUE])

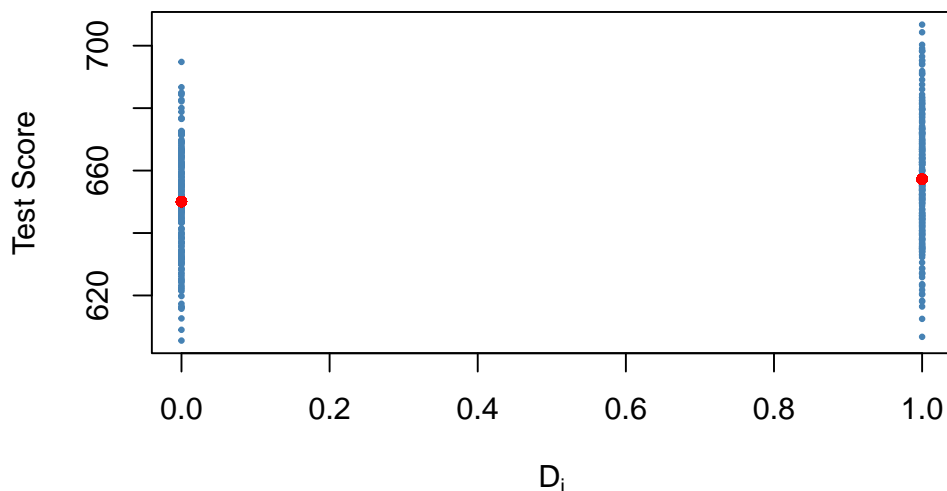
# Compute the average score when D=0 (high STR)
mean.score.for.D.0 <- mean(CASchools$score[CASchools$D == FALSE])

plot( CASchools$score ~ CASchools$D,          # provide the data to be plotted
      pch = 19,                               # use filled circles as plot symbols
      cex = 0.5,                              # set size of plot symbols to 0.5
      col = "Steelblue",                     # set the symbols' color to "Steelblue"
      xlab = expression(D[i]),               # Set title and axis names
      ylab = "Test Score",
      main = "Dummy Regression")

# Add the average for each group
```

```
points( y = mean.score.for.D.0, x = 0, col="red", pch = 19)
points( y = mean.score.for.D.1, x = 1, col="red", pch = 19)
```

Dummy Regression



With D as the regressor, it is not useful to think of β_1 as a slope parameter since $D_i \in \{0, 1\}$, i.e., we only observe two discrete values instead of a continuum of regressor values. There is no continuous line depicting the conditional expectation function $E(\text{TestScore}_i|D_i)$ since this function is solely defined for x -positions 0 and 1.

Therefore, the interpretation of the coefficients in this regression model is as follows:

- $E(Y_i|D_i = 0) = \beta_0$, so β_0 is the expected test score in districts where $D_i = 0$ and STR is above 20.
- $E(Y_i|D_i = 1) = \beta_0 + \beta_1$ or, using the result above, $\beta_1 = E(Y_i|D_i = 1) - E(Y_i|D_i = 0)$. Thus, β_1 is the difference in group-specific expectations, i.e., the difference in expected test score between districts with $STR < 20$ and those with $STR \geq 20$.

We will now use R to estimate the dummy regression model as defined by the equations (5.2) and (5.3).

```
# estimate the dummy regression model
dummy_model <- lm(score ~ D, data = CASchools)
summary(dummy_model)
#>
#> Call:
#> lm(formula = score ~ D, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -50.496 -14.029  -0.346  12.884  49.504
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  650.077       1.393  466.666 < 2e-16 ***
#> DTRUE        7.169        1.847   3.882 0.00012 ***
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 18.74 on 418 degrees of freedom
#> Multiple R-squared:  0.0348, Adjusted R-squared:  0.0325
#> F-statistic: 15.07 on 1 and 418 DF,  p-value: 0.0001202
```

summary() function indicates that the p -value for testing whether the coefficient on (Intercept) is zero is reported to be $< 2e-16$. This scientific notation implies that the p -value is extremely tiny, smaller than $\frac{2}{10^{16}}$. This small value is a result of the limitation of computers to handle extremely minute numbers. In fact, $\frac{2}{10^{16}}$ is the smallest possible number R can work with.

The vector `CASchools$D` has the type `logical` (to see this, use `typeof(CASchools$D)`) which is shown in the output of `summary(dummy_model)`: the label `DTRUE` states that all entries `TRUE` are coded as 1 and all entries `FALSE` are coded as 0. Thus, the interpretation of the coefficient `DTRUE` is same as stated above for β_1 .

One can see that the expected test score in districts with $STR < 20$ ($D_i = 1$) is predicted to be $650.1 + 7.17 = 657.27$ while districts with $STR \geq 20$ ($D_i = 0$) are expected to have an average test score of only 650.1.

Group specific predictions can be added to the plot by execution of the following code chunk.

```
# add group specific predictions to the plot
points(x = CASchools$D,
       y = predict(dummy_model),
       col = "red",
       pch = 20)
```

Here we use the function `predict()` to obtain estimates of the group specific means. The red dots represent these sample group averages. Accordingly, $\hat{\beta}_1 = 7.17$ can be seen as the difference in group averages.

`summary(dummy_model)` also answers the question whether there is a statistically significant difference in group means. This in turn would support the hypothesis that students perform differently when they are taught in small classes. We can assess this by a two-tailed test of the hypothesis $H_0 : \beta_1 = 0$. Conveniently, the t -statistic and the corresponding p -value for this test are computed by `summary()`.

Since `t value = 3.88 > 1.96`, we reject the null hypothesis at the 5% level of significance. The same conclusion results when using the p -value, which reports significance up to the 0.00012% level.

As done with `linear_model`, we may alternatively use the function `confint()` to compute a 95% confidence interval for the true difference in means and see if the hypothesized value is an element of this confidence set.

```
# confidence intervals for coefficients in the dummy regression model
confint(dummy_model)
#>           2.5 %      97.5 %
#> (Intercept) 647.338594 652.81500
#> DTRUE       3.539562  10.79931
```

We reject the hypothesis that there is no difference between group means at the 5% significance level since $\beta_{1,0} = 0$ lies outside of $[3.54, 10.8]$, the 95% confidence interval for the coefficient on D .

5.4 Heteroskedasticity and Homoskedasticity

All inference made in the previous chapters relies on the assumption that the error variance does not vary as regressor values change. But this will often not be the case in empirical applications.

Key Concept 5.4 Heteroskedasticity and Homoskedasticity

- The error term of our regression model is homoskedastic if the variance of the conditional distribution of u_i given X_i , $\text{Var}(u_i|X_i = x)$, is constant *for all* observations in our sample:

$$\text{Var}(u_i|X_i = x) = \sigma^2 \quad \forall i = 1, \dots, n.$$

- If instead there is dependence of the conditional variance of u_i on X_i , the error term is said to be heteroskedastic. We then write

$$\text{Var}(u_i|X_i = x) = \sigma_i^2 \quad \forall i = 1, \dots, n.$$

- Homoskedasticity is a *special case* of heteroskedasticity.

For a better understanding of heteroskedasticity, we generate some bivariate heteroskedastic data, estimate a linear regression model and then use box plots to depict the conditional distributions of the residuals.

```
# load scales package for adjusting color opacities
library(scales)

# generate some heteroskedastic data:

# set seed for reproducibility
set.seed(123)

# set up vector of x coordinates
x <- rep(c(10, 15, 20, 25), each = 25)

# initialize vector of errors
e <- c()

# sample 100 errors such that the variance increases with x
e[1:25] <- rnorm(25, sd = 10)
e[26:50] <- rnorm(25, sd = 15)
e[51:75] <- rnorm(25, sd = 20)
e[76:100] <- rnorm(25, sd = 25)

# set up y
y <- 720 - 3.3 * x + e

# Estimate the model
mod <- lm(y ~ x)

# Plot the data
plot(x = x,
```



```

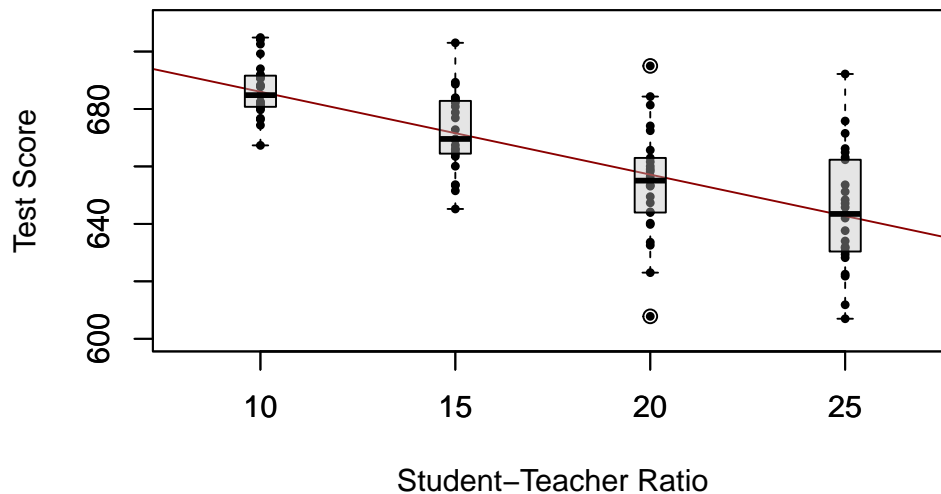
y = y,
main = "An Example of Heteroskedasticity",
xlab = "Student-Teacher Ratio",
ylab = "Test Score",
cex = 0.5,
pch = 19,
xlim = c(8, 27),
ylim = c(600, 710))

# Add the regression line to the plot
abline(mod, col = "darkred")

# Add boxplots to the plot
boxplot(formula = y ~ x,
        add = TRUE,
        at = c(10, 15, 20, 25),
        col = alpha("gray", 0.4),
        border = "black")

```

An Example of Heteroskedasticity



We have used the `formula` argument `y ~ x` in `boxplot()` to specify that we want to split up the vector `y` into groups according to `x`. `boxplot(y ~ x)` generates a boxplot for each of the groups in `y` defined by `x`.

For this artificial data it is clear that the conditional error variances differ. Specifically, we observe that the variance in test scores (and therefore the variance of the errors committed) *increases* with the student teacher ratio.

A Real-World Example for Heteroskedasticity

Think about the economic value of education: if there were no expected economic value-added to receiving university education, you probably would not be reading this script right now. A starting point to empirically verify such a relation is to have data on working individuals. More precisely, we need data on wages and education of workers in order to estimate a model like

$$wage_i = \beta_0 + \beta_1 \cdot education_i + u_i.$$

What can be presumed about this relation? It is likely that, on average, higher educated workers earn more than workers with less education, so we expect to estimate an upward sloping regression line. Also, it seems plausible that earnings of better educated workers have a higher dispersion than those of low-skilled workers: solid education is not a guarantee for a high salary so even highly qualified workers take on low-income jobs. However, they are more likely to meet the requirements for the well-paid jobs than workers with less education for whom opportunities in the labor market are much more limited.

To verify this empirically we may use real data on hourly earnings and the number of years of education of employees. Such data can be found in `CPSSWEducation`. This data set is part of the package `AER` and comes from the Current Population Survey (CPS) which is conducted periodically by the Bureau of Labor Statistics in the United States.

The subsequent code chunks demonstrate how to import the data into R and how to produce a plot in the fashion of Figure 5.3 in the book.

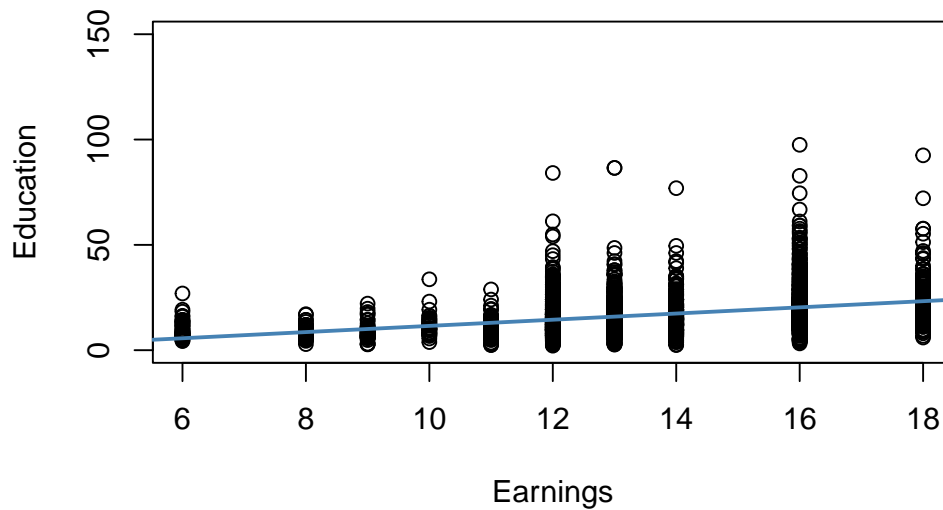
```
# load package and attach data
library(AER)
data("CPSSWEducation")
attach(CPSSWEducation)

# get an overview
summary(CPSSWEducation)
#>      age      gender      earnings      education
#> Min.   :29.0  female:1202  Min.    : 2.137  Min.    : 6.00
#> 1st Qu.:29.0  male  :1748  1st Qu.:10.577  1st Qu.:12.00
#> Median :29.0                      Median :14.615  Median :13.00
#> Mean   :29.5                      Mean   :16.743  Mean   :13.55
#> 3rd Qu.:30.0                      3rd Qu.:20.192  3rd Qu.:16.00
#> Max.   :30.0                      Max.   :97.500  Max.   :18.00

# estimate a simple regression model
labor_model <- lm(earnings ~ education)

# plot observations and add the regression line
plot(education,
     earnings,
     xlab="Earnings",
     ylab="Education",
     ylim = c(0, 150))

abline(labor_model,
       col = "steelblue",
       lwd = 2)
```



The plot reveals that the mean of the distribution of earnings increases with the level of education. This is also supported by a formal analysis: the estimated regression model stored in `labor_model` shows that there is a positive relation between years of education and earnings.

```
# print the contents of labor_model to the console
labor_model
#>
#> Call:
#> lm(formula = earnings ~ education)
#>
#> Coefficients:
#> (Intercept)    education
#>    -3.134      1.467
```

The estimated regression equation states that, on average, an additional year of education increases a worker's hourly earnings by about \$1.47. Once more we use `confint()` to obtain a 95% confidence interval for both regression coefficients.

```
# compute a 95% confidence interval for the coefficients in the model
confint(labor_model)
#>           2.5 %    97.5 %
#> (Intercept) -5.015248 -1.253495
#> education   1.330098  1.603753
```

Since the interval is $[1.33, 1.60]$ we can reject the hypothesis that the coefficient on `education` is zero at the 5% level.

Furthermore, the plot indicates that there is heteroskedasticity: if we assume the regression line to be a reasonably good representation of the conditional mean function $E(\text{earnings}_i | \text{education}_i)$, the dispersion of hourly earnings around that function clearly increases with the level of education, i.e., the variance of the distribution of earnings increases. In other words: the variance of the errors (the errors made in explaining earnings by education) increases with education so that the regression errors are heteroskedastic.

This example makes a case that the assumption of homoskedasticity is doubtful in economic applications. Should we care about heteroskedasticity? Yes, we should. As explained in the next section, heteroskedasticity can have serious negative consequences in hypothesis testing, if we ignore it.

Should We Care About Heteroskedasticity?

To answer the question whether we should worry about heteroskedasticity being present, consider the variance of $\hat{\beta}_1$ under the assumption of homoskedasticity. In this case we have

$$\sigma_{\hat{\beta}_1}^2 = \frac{\sigma_u^2}{n \cdot \sigma_X^2}, \quad (5.5)$$

which is a simplified version of the general equation (4.1) presented in Key Concept 4.4. See Appendix 5.1 of the book for details on the derivation. `summary()` estimates (5.5) by

$$\hat{\sigma}_{\hat{\beta}_1}^2 = \frac{SER^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \quad \text{where} \quad SER = \frac{1}{n-2} \sum_{i=1}^n \hat{u}_i^2.$$

Thus `summary()` estimates the *homoskedasticity-only* standard error

$$\sqrt{\hat{\sigma}_{\hat{\beta}_1}^2} = \sqrt{\frac{SER^2}{\sum_{i=1}^n (X_i - \bar{X})^2}}.$$

This is in fact an estimator for the standard deviation of the estimator $\hat{\beta}_1$ that is *inconsistent* for the true value $\sigma_{\hat{\beta}_1}^2$ when there is heteroskedasticity. The implication is that *t*-statistics computed in the manner of Key Concept 5.1 do not follow a standard normal distribution, even in large samples. This issue may invalidate inference when using the previously treated tools for hypothesis testing: we should be cautious when making statements about the significance of regression coefficients on the basis of *t*-statistics as computed by `summary()` or confidence intervals produced by `confint()` if it is doubtful for the assumption of homoskedasticity to hold!

We will now use R to compute the homoskedasticity-only standard error for $\hat{\beta}_1$ in the regression model `labor_model` by hand and see that it matches the value produced by `summary()`.

```
# Store model summary in 'model'
model <- summary(labor_model)

# Extract the standard error of the regression from model summary
SER <- model$sigma

# Compute the variation in 'education'
V <- (nrow(CPSSWEducation)-1) * var(education)

# Compute the standard error of the slope parameter's estimator and print it
SE.beta_1.hat <- sqrt(SER^2/V)
SE.beta_1.hat
#> [1] 0.06978281

# Use logical operators to see if the value computed by hand matches the one provided
# in mod$coefficients. Round estimates to four decimal places
round(model$coefficients[2, 2], 4) == round(SE.beta_1.hat, 4)
#> [1] TRUE
```

Indeed, the estimated values are equal.

Computation of Heteroskedasticity-Robust Standard Errors

Consistent estimation of $\sigma_{\hat{\beta}_1}$ under heteroskedasticity is granted when the following *robust* estimator is used.

$$SE(\hat{\beta}_1) = \sqrt{\frac{1}{n} \cdot \frac{\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2 \hat{u}_i^2}{\left[\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2\right]^2}}. \quad (5.6)$$

Standard error estimates computed this way are also referred to as Eicker-Huber-White standard errors, the most frequently cited paper on this is White (1980).

It can be quite cumbersome to do this calculation by hand. Luckily certain R functions exist, serving this purpose. An example of such function is `vcovHC()` is part of the package `sandwich`.¹ This function can compute a variety of standard errors. The one brought forward in (5.6) is computed when the argument `type` is set to "HC0". Most of the examples presented in the book rely on a slightly different formula which is the default in the statistics package *STATA*:

$$SE(\hat{\beta}_1)_{HC1} = \sqrt{\frac{1}{n} \cdot \frac{\frac{1}{n-2} \sum_{i=1}^n (X_i - \bar{X})^2 \hat{u}_i^2}{\left[\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2\right]^2}}. \quad (5.2)$$

The difference is that we multiply by $\frac{1}{n-2}$ in the numerator of (5.2). This is a degrees of freedom correction and was considered by MacKinnon and White (1985). To get `vcovHC()` to use (5.2), we have to set `type = "HC1"`.

Let us now compute robust standard error estimates for the coefficients in `linear_model`.

```
# compute heteroskedasticity-robust standard errors
vcov <- vcovHC(linear_model, type = "HC1")
vcov
#>           (Intercept)          STR
#> (Intercept) 107.419993 -5.3639114
#> STR        -5.363911  0.2698692
```

The output of `vcovHC()` is the variance-covariance matrix of coefficient estimates. We are interested in the square root of the diagonal elements of this matrix, i.e., the standard error estimates.

When we have $k > 1$ regressors, writing down the equations for a regression model becomes very messy. A more convenient way to denote and estimate so-called multiple regression models (see Chapter 6) is by using matrix algebra. This is why functions like `vcovHC()` produce matrices. In the simple linear regression model, the variances and covariances of the estimators can be gathered in the symmetric variance-covariance matrix

$$\text{Var} \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} = \begin{pmatrix} \text{Var}(\hat{\beta}_0) & \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \\ \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) & \text{Var}(\hat{\beta}_1) \end{pmatrix}, \quad (5.3)$$

so `vcovHC()` gives us $\widehat{\text{Var}}(\hat{\beta}_0)$, $\widehat{\text{Var}}(\hat{\beta}_1)$ and $\widehat{\text{Cov}}(\hat{\beta}_0, \hat{\beta}_1)$, but most of the time we are interested in the diagonal elements of the estimated matrix.

¹The package `sandwich` is a dependency of the package `AER`, meaning that it is attached automatically if you load `AER`.

```
# compute the square root of the diagonal elements in vcov
robust_se <- sqrt(diag(vcov))
robust_se
#> (Intercept)          STR
#> 10.3643617  0.5194893
```

Now assume we want to generate a coefficient summary as provided by `summary()` but with *robust* standard errors of the coefficient estimators, robust *t*-statistics and corresponding *p*-values for the regression model `linear_model`. This can be done using `coefTest()` from the package `lmtest`, see `?coefTest`. Further we specify in the argument `vcov` that `vcov`, the Eicker-Huber-White estimate of the variance matrix we have computed before, should be used.

```
# we invoke the function `coefTest()` on our model
coefTest(linear_model, vcov. = vcov)
#>
#> t test of coefficients:
#>
#>          Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 698.93295   10.36436  67.4362 < 2.2e-16 ***
#> STR          -2.27981    0.51949  -4.3886 1.447e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that the values reported in the column `Std.Error` are equal those from `sqrt(diag(vcov))`.

How severe are the implications of using homoskedasticity-only standard errors in the presence of heteroskedasticity? The answer is: it depends. As mentioned above we face the risk of drawing wrong conclusions when conducting significance tests. Let us illustrate this by generating another example of a heteroskedastic data set and using it to estimate a simple regression model. We take

$$Y_i = \beta_1 \cdot X_i + u_i, \quad u_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 0.36 \cdot X_i^2)$$

with $\beta_1 = 1$ as the data generating process. Clearly, the assumption of homoskedasticity is violated here since the variance of the errors is a nonlinear, increasing function of X_i but the errors have zero mean and are i.i.d. such that the assumptions made in Key Concept 4.3 are not violated. As before, we are interested in estimating β_1 .

```
set.seed(905)

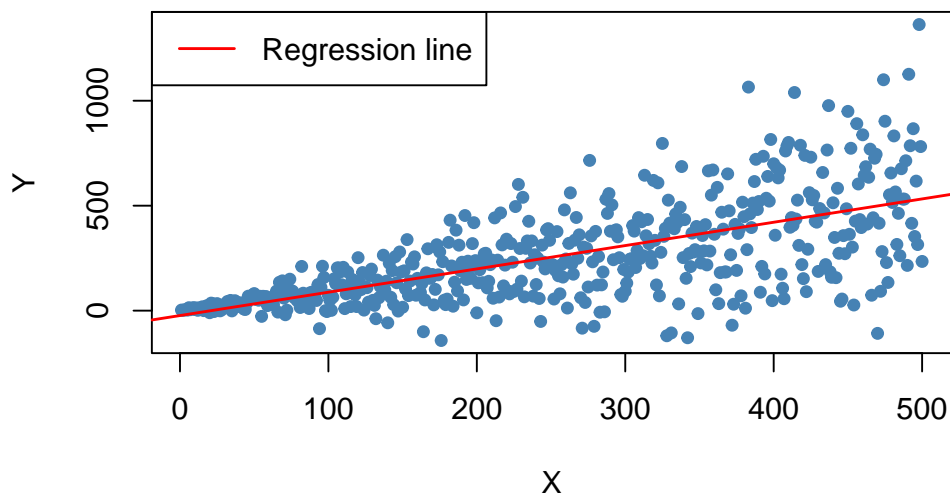
# generate heteroskedastic data
X <- 1:500
Y <- rnorm(n = 500, mean = X, sd = 0.6 * X)

# estimate a simple regression model
reg <- lm(Y ~ X)
```

We plot the data and add the regression line.

```
# plot the data
plot(x = X, y = Y,
     pch = 19,
     col = "steelblue",
     cex = 0.8)
```

```
# add the regression line to the plot
abline(reg,
       col = "red",
       lwd = 1.5)
legend("topleft", "Regression line", col="red", lwd=1.5)
```



The plot shows that the data are heteroskedastic as the variance of Y grows with X . We next conduct a significance test of the (true) null hypothesis $H_0 : \beta_1 = 1$ twice, once using the homoskedasticity-only standard error formula and once with the robust version (5.6). An easy way to do this in R is the function `linearHypothesis()` from the package `car`, see `?linearHypothesis`. It allows to test linear hypotheses about parameters in linear models in a similar way as done with a t -statistic and offers various robust covariance matrix estimators. We test by comparing the tests' p -values to the significance level of 5%.

`linearHypothesis()` computes a test statistic that follows an F -distribution under the null hypothesis. We will not focus on the details of the underlying theory. In general, the idea of the F -test is to compare the fit of different models. When testing a hypothesis about a *single* coefficient using an F -test, one can show that the test statistic is simply the square of the corresponding t -statistic:

$$F = t^2 = \left(\frac{\hat{\beta}_i - \beta_{i,0}}{SE(\hat{\beta}_i)} \right)^2 \sim F_{1,n-k-1}.$$

In `linearHypothesis()`, there are different ways to specify the hypothesis to be tested, e.g., using a vector of the type character (as done in the next code chunk), see `?linearHypothesis` for alternatives. The function returns an object of class `anova` which contains further information on the test that can be accessed using the `$` operator.

```
# test hypothesis using the default standard error formula
linearHypothesis(reg, hypothesis.matrix = "X = 1")$'Pr(>F)'[2] < 0.05
#> [1] TRUE

# test hypothesis using the robust standard error formula
linearHypothesis(reg, hypothesis.matrix = "X = 1",
                 white.adjust = "hc1")$'Pr(>F)'[2] < 0.05
```

```
#> [1] FALSE
```

This is a good example of what can go wrong if we ignore heteroskedasticity: for the data set at hand the default method rejects the null hypothesis $\beta_1 = 1$ although it is true. When using the robust standard error formula the test does not reject the null. Of course, we could think this might just be a coincidence and both tests do equally well in maintaining the type I error rate of 5%. This can be further investigated by computing *Monte Carlo* estimates of the rejection frequencies of both tests on the basis of a large number of random samples. We proceed as follows:

- initialize vectors `t` and `t.rob`.
- Using a `for()` loop, we generate 10000 heteroskedastic random samples of size 1000, estimate the regression model and check whether the tests falsely reject the null at the level of 5% using comparison operators. The results are stored in the respective vectors `t` and `t.rob`.
- After the simulation, we compute the fraction of false rejections for both tests.

```
# initialize vectors t and t.rob
t <- c()
t.rob <- c()

# loop sampling and estimation
for (i in 1:10000) {

  # sample data
  X <- 1:1000
  Y <- rnorm(n = 1000, mean = X, sd = 0.6 * X)

  # estimate regression model
  reg <- lm(Y ~ X)

  # homoskedasticity-only significance test
  t[i] <- linearHypothesis(reg, "X = 1")$'Pr(>F)'[2] < 0.05

  # robust significance test
  t.rob[i] <- linearHypothesis(reg, "X = 1", white.adjust = "hc1")$'Pr(>F)'[2] < 0.05

}

# compute the fraction of false rejections
round(cbind(t = mean(t), t.rob = mean(t.rob)), 3)
#>           t t.rob
#> [1,] 0.073 0.05
```

These results reveal the increased risk of falsely rejecting the null using the homoskedasticity-only standard error for the testing problem at hand: with the common standard error, 7.28% of all tests falsely reject the null hypothesis. In contrast, with the robust test statistic we are closer to the nominal level of 5%.

5.5 The Gauss-Markov Theorem

When estimating regression models, we know that the results of the estimation procedure are random. However, when using unbiased estimators, at least on average, we estimate the true parameter. When comparing different unbiased estimators, it is therefore interesting to know which one has the highest

precision: being aware that the likelihood of estimating the *exact* value of the parameter of interest is 0 in an empirical application, we want to make sure that the likelihood of obtaining an estimate very close to the true value is as high as possible. This means we want to use the estimator with the lowest variance of all unbiased estimators, provided we care about unbiasedness. The Gauss-Markov theorem states that, in the class of conditionally unbiased linear estimators, the OLS estimator has this property under certain conditions.

Key Concept 5.5

The Gauss-Markov Theorem for $\hat{\beta}_1$

Suppose that the assumptions made in Key Concept 4.3 hold and that the errors are *homoskedastic*. The OLS estimator is the best (in the sense of smallest variance) linear conditionally unbiased estimator (BLUE) in this setting.

Let us have a closer look at what this means:

- Estimators of β_1 that are linear functions of the Y_1, \dots, Y_n and that are unbiased conditionally on the regressor X_1, \dots, X_n can be written as

$$\tilde{\beta}_1 = \sum_{i=1}^n a_i Y_i.$$

where the a_i are weights that are allowed to depend on the X_i but *not* on the Y_i .

- We already know that $\tilde{\beta}_1$ has a sampling distribution: $\tilde{\beta}_1$ is a linear function of the Y_i which are random variables. If now

$$E(\tilde{\beta}_1 | X_1, \dots, X_n) = \beta_1,$$

$\tilde{\beta}_1$ is a linear unbiased estimator of β_1 , conditionally on the X_1, \dots, X_n .

- We may ask if $\tilde{\beta}_1$ is also the *best* estimator in this class, i.e., the most efficient one of all linear conditionally unbiased estimators where most efficient means smallest variance. The weights a_i play an important role here and it turns out that OLS uses just the right weights to have the BLUE property.

Simulation Study: BLUE Estimator

Consider the case of a regression of Y_1, \dots, Y_n only on a constant. Here, the Y_i are assumed to be a random sample from a population with mean μ and variance σ^2 . The OLS estimator in this model is simply the sample mean, see Chapter 3.2.

$$\hat{\beta}_1 = \sum_{i=1}^n \underbrace{\frac{1}{n}}_{=a_i} Y_i. \quad (5.4)$$

Clearly, each observation is weighted by

$$a_i = \frac{1}{n},$$

and we also know that $\text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{n}$.

We now use R to conduct a simulation study that demonstrates what happens to the variance of (5.4) if different weights

$$w_i = \frac{1 \pm \epsilon}{n}$$

are assigned to either half of the sample Y_1, \dots, Y_n instead of using $\frac{1}{n}$, the OLS weights.

```
# set sample size and number of repetitions
n <- 100
reps <- 1e5

# choose epsilon and create a vector of weights as defined above
epsilon <- 0.8
w <- c(rep((1 + epsilon) / n, n / 2),
       rep((1 - epsilon) / n, n / 2))

# draw a random sample y_1, ..., y_n from the standard normal distribution,
# use both estimators 1e5 times and store the result in the vectors 'ols' and
# 'weightedestimator'

ols <- rep(NA, reps)
weightedestimator <- rep(NA, reps)

for (i in 1:reps) {

  y <- rnorm(n)
  ols[i] <- mean(y)
  weightedestimator[i] <- crossprod(w, y)

}

# plot kernel density estimates of the estimators' distributions:

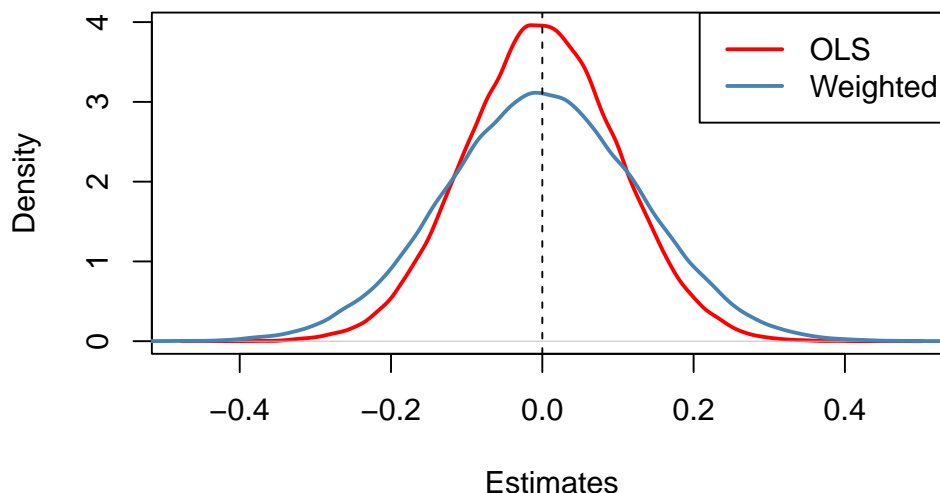
# OLS
plot(density(ols),
     col = "red",
     lwd = 2,
     main = "Density of OLS and Weighted Estimator",
     xlab = "Estimates")

# weighted
lines(density(weightedestimator),
      col = "steelblue",
      lwd = 2)

# add a dashed line at 0 and add a legend to the plot
abline(v = 0, lty = 2)

legend('topright',
      c("OLS", "Weighted"),
      col = c("red", "steelblue"),
      lwd = 2)
```

Density of OLS and Weighted Estimator



What conclusion can we draw from the result?

- Both estimators seem to be unbiased: the means of their estimated distributions are zero.
- The estimator using weights that deviate from those implied by OLS is less efficient than the OLS estimator: there is higher dispersion when weights are $w_i = \frac{1 \pm 0.8}{100}$ instead of $w_i = \frac{1}{100}$ as required by the OLS solution.

Hence, the simulation results support the Gauss-Markov Theorem.

5.6 Using the t -Statistic in Regression when the Sample Size Is Small

The three OLS assumptions discussed in Chapter 4 (see Key Concept 4.3) are the foundation for the results on the large sample distribution of the OLS estimators in the simple regression model. What can be said about the distribution of the estimators and their t -statistics when the sample size is small and the population distribution of the data is unknown? Provided that the three least squares assumptions hold and the errors are normally distributed and homoskedastic (we refer to these conditions as the homoskedastic normal regression assumptions), we have normally distributed estimators and t -distributed test statistics in small samples.

Recall the definition of a t -distributed variable

$$\frac{Z}{\sqrt{W/M}} \sim t_M,$$

where Z is a standard normal random variable, W is χ^2 distributed with M degrees of freedom and Z and W are independent. See section 5.6 in the book for a more detailed discussion of the small sample distribution of t -statistics in regression methods.

Let us simulate the distribution of regression t -statistics based on a large number of small random samples, say $n = 20$, and compare the simulated distributions to the theoretical distributions which should be t_{18} , the t -distribution with 18 degrees of freedom (recall that $DF = n - k - 1$).

```

# initialize two vectors
beta_0 <- c()
beta_1 <- c()

# loop sampling / estimation / t statistics
for (i in 1:10000) {

  X <- runif(20, 0, 20)
  Y <- rnorm(n = 20, mean = X)
  reg <- summary(lm(Y ~ X))
  beta_0[i] <- (reg$coefficients[1, 1] - 0)/(reg$coefficients[1, 2])
  beta_1[i] <- (reg$coefficients[2, 1] - 1)/(reg$coefficients[2, 2])

}

# plot the distributions and compare with t_18 density:

# divide plotting area
par(mfrow = c(1, 2))

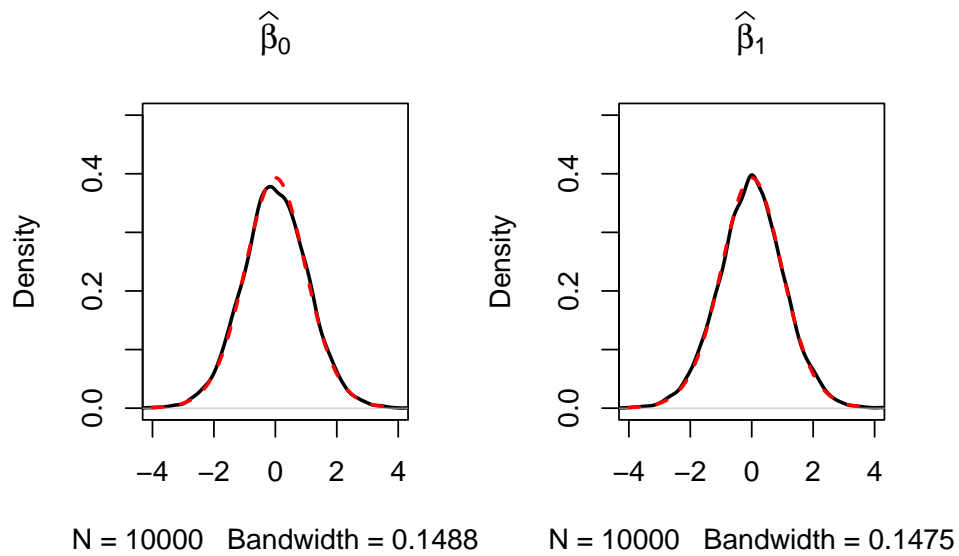
# plot the simulated density of beta_0
plot(density(beta_0),
     lwd = 2,
     main = expression(widehat(beta)[0]),
     xlim = c(-4, 4),
     ylim=c(0,0.5))

# add the t_18 density to the plot
curve(dt(x, df = 18),
      add = T,
      col = "red",
      lwd = 2,
      lty = 2)

# plot the simulated density of beta_1
plot(density(beta_1),
     lwd = 2,
     main = expression(widehat(beta)[1]),
     xlim = c(-4, 4),
     ylim=c(0,0.5))

# add the t_18 density to the plot
curve(dt(x, df = 18),
      add = T,
      col = "red",
      lwd = 2,
      lty = 2)

```



The outcomes are consistent with our expectations: the empirical distributions of both the estimators seem to track the theoretical t_{18} distribution quite closely.

5.7 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 6

Regression Models with Multiple Regressors

In what follows we introduce linear regression models that use more than just one explanatory variable and discuss important key concepts in multiple regression. As we broaden our scope beyond the relationship of only two variables (the dependent variable and a single regressor) some potential new issues arise such as *multicollinearity* and *omitted variable bias* (OVB). In particular, this chapter deals with omitted variables and its implication for causal interpretation of OLS-estimated coefficients.

Naturally, we will discuss estimation of multiple regression models using R. We will also illustrate the importance of thoughtful usage of multiple regression models via simulation studies that demonstrate the consequences of using highly correlated regressors or misspecified models.

The packages `AER` (Kleiber and Zeileis, 2008b) and `MASS` (Ripley, 2023) are needed for reproducing the code presented in this chapter. Make sure that the following code chunk executes without any errors.

```
library(AER)
library(MASS)
```

6.1 Omitted Variable Bias

The previous analysis of the relationship between test score and class size discussed in Chapters 4 and 5 has a major flaw: we ignored other determinants of the dependent variable (test score) that correlate with the regressor (class size). Remember that influences on the dependent variable which are not captured by the model are collected in the error term, which we so far assumed to be uncorrelated with the regressor. However, this assumption is violated if we exclude determinants of the dependent variable which vary with the regressor. This might induce an estimation bias, i.e., the mean of the OLS estimator's sampling distribution is no longer equals to the true mean. In our example we therefore wrongly estimate the causal effect on test scores of a unit change in the student-teacher ratio, on average. This issue is called *omitted variable bias* (OVB) and is summarized by Key Concept 6.1.

Key Concept 6.1 Omitted Variable Bias in Regression with a Single Regressor

Omitted variable bias is the bias in the OLS estimator that arises when the regressor, X , is *correlated* with an omitted variable. For omitted variable bias to occur, two conditions must be fulfilled:

1. X is correlated with the omitted variable.
2. The omitted variable is a determinant of the dependent variable Y .

Together, 1. and 2. result in a violation of the first OLS assumption $E(u_i|X_i) = 0$. Formally, the resulting bias can be expressed as

$$\hat{\beta}_1 \xrightarrow{p} \beta_1 + \rho_{Xu} \frac{\sigma_u}{\sigma_X}. \quad (6.1)$$

See Appendix 6.1 of the book for a detailed derivation. (6.1) states that OVB is a problem that cannot be solved by increasing the number of observations used to estimate β_1 , as $\hat{\beta}_1$ is inconsistent: OVB prevents the estimator from converging in probability to the true parameter value. Strength and direction of the bias are determined by ρ_{Xu} , the correlation between the error term and the regressor.

In the example of test score and class size, it is easy to come up with variables that may cause such a bias, if omitted from the model. As mentioned in the book, a highly relevant variable could be the percentage of English learners in the school district: it is plausible that the ability to speak, read and write English is an important factor for successful learning. Therefore, students that are still learning English are likely to perform worse in tests than native speakers. Also, it is conceivable that the share of English learning students is bigger in school districts where class sizes are relatively large: think of poor urban districts where a lot of immigrants live.

Let us think about a possible bias induced by omitting the share of English learning students ($PctEL$) in view of (6.1). When the estimated regression model does not include $PctEL$ as a regressor although the true data generating process (DGP) is

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times PctEL \quad (6.2)$$

where STR and $PctEL$ are correlated, we have

$$\rho_{STR, PctEL} \neq 0.$$

Let us investigate this using R. After defining our variables we may compute the correlation between STR and $PctEL$ as well as the correlation between STR and $TestScore$.

```
# load the AER package
library(AER)

# load the data set
data(CASchools)

# define variables
CASchools$STR <- CASchools$students/CASchools$teachers
```

```

CASchools$score <- (CASchools$read + CASchools$math)/2

# compute correlations
cor(CASchools$STR, CASchools$score)
#> [1] -0.2263627
cor(CASchools$STR, CASchools$english)
#> [1] 0.1876424

```

The fact that $\hat{\rho}_{STR, Testscore} = -0.2264$ is cause for concern that Omitting *PctEL* leads to a negatively biased estimate $\hat{\beta}_1$ which is $\hat{\rho}_{STR, Testscore} = -0.2264$ since this indicates that $\rho_{Xu} < 0$. As a consequence we expect $\hat{\beta}_1$, the coefficient on *STR*, to be too large in absolute value. Put differently, the OLS estimate of $\hat{\beta}_1$ suggests that small classes improve test scores, but that the effect of small classes is overestimated as it captures the effect of having fewer English learners, too.

What happens to the magnitude of $\hat{\beta}_1$ if we add the variable *PctEL* to the regression, that is, if we estimate the model

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times PctEL + u$$

instead? And what do we expect about the sign of $\hat{\beta}_2$, the estimated coefficient on *PctEL*? Following the reasoning above we should still end up with a negative but larger coefficient estimate $\hat{\beta}_1$ than before and a negative estimate $\hat{\beta}_2$.

Let us estimate both regression models and compare. Performing a multiple regression in R is straightforward. One can simply add additional variables to the right hand side of the formula argument of the function `lm()` by using their names and the `+` operator.

```

# estimate both regression models
mod <- lm(score ~ STR, data = CASchools)
mult.mod <- lm(score ~ STR + english, data = CASchools)

# print the results to the console
mod
#>
#> Call:
#> lm(formula = score ~ STR, data = CASchools)
#>
#> Coefficients:
#> (Intercept)          STR
#>    698.93         -2.28
mult.mod
#>
#> Call:
#> lm(formula = score ~ STR + english, data = CASchools)
#>
#> Coefficients:
#> (Intercept)          STR          english
#>    686.0322     -1.1013     -0.6498

```

We find the outcomes to be consistent with our expectations.

The following section discusses some theory on multiple regression models.

6.2 The Multiple Regression Model

The multiple regression model extends the basic concept of the simple regression model discussed in chapters 4 and 5. A multiple regression model enables us to estimate the effect on Y_i of changing a regressor X_{1i} if the remaining regressors $X_{2i}, X_{3i}, \dots, X_{ki}$ do not vary. In fact we already have performed estimation of the multiple regression model (6.2) using **R** in the previous section. The interpretation of the coefficient on student-teacher ratio is the effect on test scores of a one unit change student-teacher ratio if the percentage of English learners is kept constant.

Just like in the simple regression model, we assume the true relationship between Y and $X_{1i}, X_{2i}, \dots, X_{ki}$ to be linear. On average, this relation is given by the population regression function

$$E(Y_i | X_{1i} = x_1, X_{2i} = x_2, X_{3i} = x_3, \dots, X_{ki} = x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_k x_k. \quad (6.3)$$

As in the simple regression model, the relation

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \dots + \beta_k X_{ki}$$

does not hold exactly since there are disturbing influences to the dependent variable Y we cannot observe as explanatory variables. Therefore we add an error term u which represents deviations of the observations from the population regression line to (6.3). This yields the population multiple regression model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \dots + \beta_k X_{ki} + u_i, \quad i = 1, \dots, n. \quad (6.4)$$

Key Concept 6.2 summarizes the core concepts of the multiple regression model.

Key Concept 6.2 The Multiple Regression Model

- Y_i is the i^{th} observation in the dependent variable. Observations on the k regressors are denoted by $X_{1i}, X_{2i}, \dots, X_{ki}$ and u_i is the error term.
- The average relationship between Y and the regressors is given by the population regression line

$$E(Y_i | X_{1i} = x_1, X_{2i} = x_2, X_{3i} = x_3, \dots, X_{ki} = x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_k x_k.$$

- β_0 is the intercept; it is the expected value of Y when all X s equal 0. β_j , $j = 1, \dots, k$ are the coefficients on X_j , $j = 1, \dots, k$. β_1 measures the expected change in Y_i that results from a one unit change in X_{1i} while holding all other regressors constant.

How can we estimate the coefficients of the multiple regression model (6.4)? We will not go too much into detail on this issue as our focus is on using **R**. However, it should be pointed out that, similarly to the simple regression model, the coefficients of the multiple regression model can be estimated using OLS. As in the simple model, we seek to minimize the sum of squared mistakes by choosing estimates b_0, b_1, \dots, b_k for the coefficients $\beta_0, \beta_1, \dots, \beta_k$ such that

$$\sum_{i=1}^n (Y_i - b_0 - b_1 X_{1i} - b_2 X_{2i} - \dots - b_k X_{ki})^2 \quad (6.5)$$

is minimized. Note that (6.5) is simply an extension of SSR in the case with just one regressor and a constant. The estimators that minimize (6.5) are hence denoted by $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ and, as in the simple

regression model, we call them the ordinary least squares estimators of $\beta_0, \beta_1, \dots, \beta_k$. For the predicted value of Y_i given the regressors and the estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ we have

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \dots + \hat{\beta}_k X_{ki}.$$

The difference between Y_i and its predicted value \hat{Y}_i is called the OLS residual of observation i : $\hat{u} = Y_i - \hat{Y}_i$.

For further information regarding the theory behind multiple regression, see Chapter 18.1 of the book which inter alia presents a derivation of the OLS estimator in the multiple regression model using matrix notation.

Now let us jump back to the example of test scores and class sizes. The estimated model object is `mult.mod`. As for simple regression models we can use `summary()` to obtain information on estimated coefficients and model statistics.

```
summary(mult.mod)$coef
#>               Estimate Std. Error   t value    Pr(>|t|)
#> (Intercept) 686.0322445  7.41131160  92.565565 3.871327e-280
#> STR          -1.1012956  0.38027827  -2.896026 3.978059e-03
#> english     -0.6497768  0.03934254 -16.515882 1.657448e-47
```

So the estimated multiple regression model is

$$\widehat{TestScore} = 686.03 - 1.10 \times STR - 0.65 \times PctEL. \quad (6.6)$$

Unlike in the simple regression model where the data can be represented by points in the two-dimensional coordinate system, we now have three dimensions. Hence observations can be represented by points in three-dimensional space. Therefore (6.6) is now no longer a regression line but a *regression plane*. This idea extends to higher dimensions when we further expand the number of regressors k . We then say that the regression model can be represented by a hyperplane in the $k + 1$ dimensional space. It is already hard to imagine such a space if $k = 3$ and we best stick with the general idea that, in the multiple regression model, the dependent variable is explained by a *linear combination of the regressors*. However, in the present case we are able to visualize the situation. The following figure is an interactive 3D visualization of the data and the estimated regression plane (6.6).

This interactive part of the book is only available in the HTML version.

We observe that the estimated regression plane fits the data reasonably well — at least with regard to the shape and spatial position of the points. The color of the markers is an indicator for the absolute deviation from the predicted regression plane. Observations that are colored more reddish lie close to the regression plane while the color shifts to blue with growing distance. An anomaly that can be seen from the plot is that there might be heteroskedasticity: we see that the dispersion of regression errors made, i.e., the distance of observations to the regression plane tends to decrease as the share of English learning students increases.

6.3 Measures of Fit in Multiple Regression

In multiple regression, common summary statistics are *SER*, R^2 and the adjusted R^2 .

Taking the code from Section 6.2, simply use `summary(mult.mod)` to obtain the *SER*, R^2 and adjusted- R^2 . For multiple regression models the *SER* is computed as

$$SER = s_{\hat{u}} = \sqrt{s_{\hat{u}}^2},$$

where modify the denominator of the premultiplied factor in s_u^2 in order to accommodate for additional regressors. Thus,

$$s_u^2 = \frac{1}{n - k - 1} SSR$$

with k denoting the number of regressors *excluding* the intercept.

While `summary()` computes the R^2 just as in the case of a single regressor, it is no reliable measure for multiple regression models. This is due to R^2 increasing whenever an additional regressor is added to the model. Adding a regressor decreases the SSR — at least unless the respective estimated coefficient is exactly zero, which practically never happens (see Chapter 6.4 of the book for a detailed argument). The adjusted R^2 takes this into consideration by “punishing” the addition of regressors using a correction factor. So the adjusted R^2 , or simply \bar{R}^2 , is a modified version of R^2 . It is defined as

$$\bar{R}^2 = 1 - \frac{n - 1}{n - k - 1} \frac{SSR}{TSS}.$$

As you may have already suspected, `summary()` adjusts the formula for SEr and it computes \bar{R}^2 and of course R^2 by default, thereby leaving the decision of which measure to rely on to the user.

You can find both measures at the bottom of the output produced by calling `summary(mult.mod)`.

```
summary(mult.mod)
#>
#> Call:
#> lm(formula = score ~ STR + english, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -48.845 -10.240  -0.308   9.815  43.461
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  686.03224     7.41131   92.566 < 2e-16 ***
#> STR          -1.10130     0.38028   -2.896  0.00398 **
#> english     -0.64978     0.03934  -16.516 < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.46 on 417 degrees of freedom
#> Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
#> F-statistic:  155 on 2 and 417 DF,  p-value: < 2.2e-16
```

We can also compute the measures by hand using the formulas above. Let us check that the results coincide with the values provided by `summary()`.

```
# define the components
n <- nrow(CASchools)           # number of observations (rows)
k <- 2                          # number of regressors

y_mean <- mean(CASchools$score) # mean of avg. test-scores

SSR <- sum(residuals(mult.mod)^2) # sum of squared residuals
```

```

TSS <- sum((CASchools$score - y_mean )^2)      # total sum of squares
ESS <- sum((fitted(mult.mod) - y_mean)^2)      # explained sum of squares

# compute the measures

SER <- sqrt(1/(n-k-1) * SSR)                  # standard error of the regression
Rsq <- 1 - (SSR / TSS)                        # R^2
adj_Rsq <- 1 - (n-1)/(n-k-1) * SSR/TSS       # adj. R^2

# print the measures to the console
c("SER" = SER, "R2" = Rsq, "Adj.R2" = adj_Rsq)
#>      SER      R2      Adj.R2
#> 14.4644831 0.4264315 0.4236805

```

Now, what can we say about the fit of our multiple regression model for test scores with the percentage of English learners as an additional regressor? Does it improve on the simple model including only an intercept and a measure of class size? The answer is yes: compare \bar{R}^2 with that obtained for the simple regression model `mod`.

Including *PctEL* as a regressor improves the \bar{R}^2 , which we deem to be more reliable in view of the above discussion. Notice that the difference between R^2 and \bar{R}^2 is small since $k = 2$ and n is large. In short, the fit of (6.6) improves vastly on the fit of the simple regression model with *STR* as the only regressor. Comparing regression errors we find that the precision of the multiple regression model (6.6) improves upon the simple model as adding *PctEL* lowers the *SER* from 18.6 to 14.5 units of test score.

As already mentioned, \bar{R}^2 may be used to quantify how good a model fits the data. However, it is rarely a good idea to maximize these measures by stuffing the model with regressors. You will not find any serious study that does so. Instead, it is more useful to include regressors that improve the estimation of the causal effect of interest which is *not* assessed well by R^2 . The issue of variable selection is covered in Chapter 8.

6.4 OLS Assumptions in Multiple Regression

In the multiple regression model we extend the three least squares assumptions of the simple regression model (see Chapter 4) and add a fourth assumption. These assumptions are presented in Key Concept 6.4. We will not go into the details of assumptions 1-3 since their ideas generalize easy to the case of multiple regressors. We will focus on the fourth assumption. This assumption rules out perfect correlation between regressors.

Key Concept 6.4**The Least Squares Assumptions in the Multiple Regression Model**

The multiple regression model is given by

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + u_i, \quad i = 1, \dots, n.$$

The OLS assumptions in the multiple regression model are an extension of the ones made for the simple regression model:

1. Regressors $(X_{1i}, X_{2i}, \dots, X_{ki}, Y_i)$, $i = 1, \dots, n$, are drawn such that the i.i.d. assumption holds.
2. u_i is an error term with conditional mean zero given the regressors, i.e.,

$$E(u_i | X_{1i}, X_{2i}, \dots, X_{ki}) = 0.$$

3. Large outliers are unlikely, formally X_{1i}, \dots, X_{ki} and Y_i have finite fourth moments.
4. No perfect multicollinearity.

Multicollinearity

Multicollinearity means that two or more regressors in a multiple regression model are *strongly* correlated. If the correlation between two or more regressors is perfect, that is, one regressor can be written as a linear combination of the other(s), we have *perfect multicollinearity*. While strong multicollinearity in general is unpleasant as it causes the variance of the OLS estimator to be large (we will discuss this in more detail later), the presence of perfect multicollinearity makes it impossible to solve for the OLS estimator, i.e., the model cannot be estimated in the first place.

The next section presents some examples of perfect multicollinearity and demonstrates how `lm()` deals with them.

Examples of Perfect Multicollinearity

How does R react if we try to estimate a model with perfectly correlated regressors?

`lm` will produce a warning in the first line of the coefficient section of the output (`1 not defined because of singularities`) and ignore the regressor(s) which is (are) assumed to be a linear combination of the other(s). Consider the following example where we add another variable `FracEL`, the fraction of English learners, to `CASchools` where observations are scaled values of the observations for `english` and use it as a regressor together with `STR` and `english` in a multiple regression model. In this example `english` and `FracEL` are perfectly collinear. The R code is as follows.

```
# define the fraction of English learners
CASchools$FracEL <- CASchools$english / 100

# estimate the model
mult.mod <- lm(score ~ STR + english + FracEL, data = CASchools)

# obtain a summary of the model
summary(mult.mod)
#>
#> Call:
#> lm(formula = score ~ STR + english + FracEL, data = CASchools)
#>
```

```

#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -48.845 -10.240  -0.308   9.815  43.461
#>
#> Coefficients: (1 not defined because of singularities)
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 686.03224    7.41131  92.566 < 2e-16 ***
#> STR          -1.10130    0.38028  -2.896  0.00398 **
#> english      -0.64978    0.03934 -16.516 < 2e-16 ***
#> FracEL              NA              NA      NA      NA
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.46 on 417 degrees of freedom
#> Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
#> F-statistic:  155 on 2 and 417 DF,  p-value: < 2.2e-16

```

The row `FracEL` in the coefficients section of the output consists of `NA` entries since `FracEL` was excluded from the model.

If we were to compute OLS by hand, we would run into the same problem but no one would be helping us out! The computation simply fails. Why is this? Take the following example:

Assume you want to estimate a simple linear regression model with a constant and a single regressor X . As mentioned above, for perfect multicollinearity to be present X has to be a linear combination of the other regressors. Since the only other regressor is a constant (think of the right hand side of the model equation as $\beta_0 \times 1 + \beta_1 X_i + u_i$ so that β_1 is always multiplied by 1 for every observation), X has to be constant as well. For $\hat{\beta}_1$ we have

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} = \frac{\widehat{Cov}(X, Y)}{\widehat{Var}(X)}. \quad (6.7)$$

The variance of the regressor X is in the denominator. Since the variance of a constant is zero, we are not able to compute this fraction and $\hat{\beta}_1$ is undefined.

Note: In this special case the denominator in (6.7) equals zero, too. Can you show that?

Let us consider two further examples where our selection of regressors induces perfect multicollinearity. First, assume that we intend to analyze the effect of class size on test score by using a dummy variable that identifies classes which are not small (NS). We define that a school has the NS attribute when the school's average student-teacher ratio is at least 12,

$$NS = \begin{cases} 0, & \text{if STR} < 12 \\ 1 & \text{otherwise.} \end{cases}$$

We add the corresponding column to `CASchools` and estimate a multiple regression model with covariates `computer` and `english`.

```

# if STR smaller 12, NS = 0, else NS = 1
CASchools$NS <- ifelse(CASchools$STR < 12, 0, 1)

# estimate the model
mult.mod <- lm(score ~ computer + english + NS, data = CASchools)

```

```

# obtain a model summary
summary(mult.mod)
#>
#> Call:
#> lm(formula = score ~ computer + english + NS, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -49.492  -9.976  -0.778   8.761  43.798
#>
#> Coefficients: (1 not defined because of singularities)
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 663.704837   0.984259  674.319 < 2e-16 ***
#> computer     0.005374    0.001670   3.218 0.00139 **
#> english     -0.708947    0.040303  -17.591 < 2e-16 ***
#> NS              NA              NA      NA      NA
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.43 on 417 degrees of freedom
#> Multiple R-squared:  0.4291, Adjusted R-squared:  0.4263
#> F-statistic: 156.7 on 2 and 417 DF,  p-value: < 2.2e-16

```

Again, the output of `summary(mult.mod)` tells us that inclusion of `NS` in the regression would render the estimation infeasible. What happened here? This is an example where we made a logical mistake when defining the regressor `NS`: taking a closer look at `NS`, the redefined measure for class size, reveals that there is not a single school with `STR < 12` hence `NS` equals one for all observations. We can check this by printing the contents of `CASchools$NS` or by using the function `table()`, see `?table`.

```

table(CASchools$NS)
#>
#> 1
#> 420

```

`CASchools$NS` is a vector of 420 ones and our data set includes 420 observations. This obviously violates assumption 4 of Key Concept 6.4: the observations for the intercept are always 1,

$$\text{intercept} = \lambda \cdot NS.$$

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\ \Leftrightarrow \lambda = 1.$$

Since the regressors can be written as a linear combination of each other, we face perfect multicollinearity and `R` excludes `NS` from the model. Thus the take-away message is: think carefully about how the regressors in your models relate to each other!

Another example of perfect multicollinearity is known as the *dummy variable trap*. This may occur when multiple dummy variables are used as regressors. A common case for this is when dummies are used to sort the data into mutually exclusive categories. For example, suppose we have spatial information that indicates whether a school is located in the North, West, South or East of the U.S. This allows us to create the dummy variables

$$\begin{aligned} North_i &= \begin{cases} 1 & \text{if located in the north} \\ 0 & \text{otherwise} \end{cases} \\ West_i &= \begin{cases} 1 & \text{if located in the west} \\ 0 & \text{otherwise} \end{cases} \\ South_i &= \begin{cases} 1 & \text{if located in the south} \\ 0 & \text{otherwise} \end{cases} \\ East_i &= \begin{cases} 1 & \text{if located in the east} \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Since the regions are mutually exclusive, for every school $i = 1, \dots, n$ we have

$$North_i + West_i + South_i + East_i = 1.$$

We run into problems when trying to estimate a model that includes a constant and *all four* direction dummies in the model, e.g.,

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_3 \times North_i + \beta_4 \times West_i + \beta_5 \times South_i + \beta_6 \times East_i + u_i \quad (6.8)$$

since then for all observations $i = 1, \dots, n$ the constant term is a linear combination of the dummies:

$$intercept = \lambda_1 \cdot (North + West + South + East) \quad (6.2)$$

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda_1 \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \quad (6.3)$$

$$\Leftrightarrow \lambda_1 = 1 \quad (6.4)$$

and we have perfect multicollinearity. Thus the “dummy variable trap” means not paying attention and falsely including exhaustive dummies *and* a constant in a regression model.

How does `lm()` handle a regression like (6.8)? Let us first generate some artificial categorical data and append a new column named `directions` to `CASchools` and see how `lm()` behaves when asked to estimate the model.

```
# set seed for reproducibility
set.seed(1)

# generate artificial data on location
CASchools$direction <- sample(c("West", "North", "South", "East"),
                             420,
                             replace = T)
```



```

# estimate the model
mult.mod <- lm(score ~ STR + english + direction, data = CASchools)

# obtain a model summary
summary(mult.mod)
#>
#> Call:
#> lm(formula = score ~ STR + english + direction, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -49.603 -10.175  -0.484   9.524  42.830
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  684.80477    7.54130  90.807 < 2e-16 ***
#> STR          -1.08873    0.38153  -2.854  0.00454 **
#> english      -0.65597    0.04018 -16.325 < 2e-16 ***
#> directionNorth  1.66314    2.05870   0.808  0.41964
#> directionSouth  0.71619    2.06321   0.347  0.72867
#> directionWest  1.79351    1.98174   0.905  0.36598
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.5 on 414 degrees of freedom
#> Multiple R-squared:  0.4279, Adjusted R-squared:  0.421
#> F-statistic: 61.92 on 5 and 414 DF,  p-value: < 2.2e-16

```

Notice that R solves the problem on its own by generating and including the dummies `directionNorth`, `directionSouth` and `directionWest` but omitting `directionEast`. Of course, the omission of every other dummy instead would achieve the same. Another solution would be to exclude the constant and to include all dummies instead.

Does this mean that the information on schools located in the East is lost? Fortunately, this is not the case: exclusion of `directionEast` just alters the interpretation of coefficient estimates on the remaining dummies from absolute to relative. For example, the coefficient estimate on `directionNorth` states that, on average, test scores in the North are about 1.61 points higher than in the East.

A last example considers the case where a perfect linear relationship arises from redundant regressors. Suppose we have a regressor `PctES`, the percentage of English speakers in the school, which is given by

$$PctES = 100 - PctEL$$

and both `PctES` and `PctEL` are included in a regression model. One regressor is redundant since the other one conveys the same information. Since this obviously is a case where the regressors can be written as linear combination, we end up with perfect multicollinearity, again.

Let us do this in R.

```

# Percentage of english speakers
CASchools$PctES <- 100 - CASchools$english

# estimate the model

```

```

mult.mod <- lm(score ~ STR + english + PctES, data = CASchools)

# obtain a model summary
summary(mult.mod)
#>
#> Call:
#> lm(formula = score ~ STR + english + PctES, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -48.845 -10.240  -0.308   9.815  43.461
#>
#> Coefficients: (1 not defined because of singularities)
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  686.03224    7.41131   92.566 < 2e-16 ***
#> STR          -1.10130    0.38028   -2.896  0.00398 **
#> english      -0.64978    0.03934  -16.516 < 2e-16 ***
#> PctES                NA              NA      NA      NA
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.46 on 417 degrees of freedom
#> Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
#> F-statistic:  155 on 2 and 417 DF,  p-value: < 2.2e-16

```

Once more, `lm()` refuses to estimate the full model using OLS and excludes `PctES`.

For more explanation of perfect multicollinearity and its impact on the OLS estimator in general multiple regression models using matrix notation, refer to Chapter 18.1 of the book.

Imperfect Multicollinearity

As opposed to perfect multicollinearity, imperfect multicollinearity is — to a certain extent — less of a problem. In fact, imperfect multicollinearity is the reason why we are interested in estimating multiple regression models in the first place: the OLS estimator allows us to *isolate* influences of *correlated* regressors on the dependent variable. If it was not for these dependencies, there would not be a reason to resort to a multiple regression approach and we could simply work with a single-regressor model. However, this is rarely the case in applications. We already know that ignoring dependencies among regressors which influence the outcome variable has an adverse effect on estimation results.

So when and why is imperfect multicollinearity a problem? Suppose you have the regression model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + u_i \quad (6.9)$$

and you are interested in estimating β_1 , the effect on Y_i of a one unit change in X_{1i} , while holding X_{2i} constant. You do not know that the true model indeed includes X_2 . You follow some reasoning and add X_2 as a covariate to the model in order to address a potential omitted variable bias. You are confident that $E(u_i | X_{1i}, X_{2i}) = 0$ and that there is no reason to suspect a violation of the assumptions 2 and 3 made in Key Concept 6.4. If X_1 and X_2 are highly correlated, OLS struggles to precisely estimate β_1 . That means that although $\hat{\beta}_1$ is a consistent and unbiased estimator for β_1 , it has a large variance due to X_2 being included in the model. If the errors are homoskedastic, this issue can be better understood from the formula for the variance of $\hat{\beta}_1$ in the model (6.9) (see Appendix 6.2 of the book):

$$\sigma_{\hat{\beta}_1}^2 = \frac{1}{n} \left(\frac{1}{1 - \rho_{X_1, X_2}^2} \right) \frac{\sigma_u^2}{\sigma_{X_1}^2}. \quad (6.10)$$

First, if $\rho_{X_1, X_2} = 0$, i.e., if there is no correlation between both regressors, including X_2 in the model has no influence on the variance of $\hat{\beta}_1$. Secondly, if X_1 and X_2 are correlated, $\sigma_{\hat{\beta}_1}^2$ is inversely proportional to $1 - \rho_{X_1, X_2}^2$ so the stronger the correlation between X_1 and X_2 , the smaller is $1 - \rho_{X_1, X_2}^2$ and thus the bigger is the variance of $\hat{\beta}_1$. Thirdly, increasing the sample size helps to reduce the variance of $\hat{\beta}_1$. Of course, this is not limited to the case with two regressors: in multiple regressions, imperfect multicollinearity inflates the variance of one or more coefficient estimators. It is an empirical question which coefficient estimates are severely affected by this and which are not. When the sample size is small, one often faces the decision whether to accept the consequence of adding a large number of covariates (higher variance) or to use a model with only few regressors (possible omitted variable bias). This is called *bias-variance trade-off*.

In sum, undesirable consequences of imperfect multicollinearity are generally not the result of a logical error made by the researcher (as is often the case for perfect multicollinearity) but are rather a problem that is linked to the data used, the model to be estimated and the research question at hand.

Simulation Study: Imperfect Multicollinearity

Let us conduct a simulation study to illustrate the issues sketched above.

1. We use (6.9) as the data generating process and choose $\beta_0 = 5$, $\beta_1 = 2.5$ and $\beta_2 = 3$ and u_i is an error term distributed as $\mathcal{N}(0, 5)$. In the first step, we sample the regressor data from a bivariate normal distribution:

$$X_i = (X_{1i}, X_{2i}) \stackrel{i.i.d.}{\sim} \mathcal{N} \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 10 & 2.5 \\ 2.5 & 10 \end{pmatrix} \right].$$

It is straightforward to see that the correlation between X_1 and X_2 in the population is rather low:

$$\rho_{X_1, X_2} = \frac{\text{Cov}(X_1, X_2)}{\sqrt{\text{Var}(X_1)}\sqrt{\text{Var}(X_2)}} = \frac{2.5}{10} = 0.25.$$

2. Next, we estimate the model (6.9) and save the estimates for β_1 and β_2 . This is repeated 10000 times with a `for` loop so we end up with a large number of estimates that allow us to describe the distributions of $\hat{\beta}_1$ and $\hat{\beta}_2$.
3. We repeat steps 1 and 2 but increase the covariance between X_1 and X_2 from 2.5 to 8.5 such that the correlation between the regressors is high:

$$\rho_{X_1, X_2} = \frac{\text{Cov}(X_1, X_2)}{\sqrt{\text{Var}(X_1)}\sqrt{\text{Var}(X_2)}} = \frac{8.5}{10} = 0.85.$$

4. In order to assess the effect on the precision of the estimators of increasing the collinearity between X_1 and X_2 we estimate the variances of $\hat{\beta}_1$ and $\hat{\beta}_2$ and compare.

```
# load packages
library(MASS)
library(mvtnorm)

# set number of observations
n <- 50
```

```

# initialize vectors of coefficients
coefs1 <- cbind("hat_beta_1" = numeric(10000), "hat_beta_2" = numeric(10000))
coefs2 <- coefs1

# set seed
set.seed(1)

# loop sampling and estimation
for (i in 1:10000) {

  # for cov(X_1,X_2) = 0.25
  X <- rmvnorm(n, c(0, 0), sigma = cbind(c(10, 2.5), c(2.5, 10)))
  u <- rnorm(n, sd = 5)
  Y <- 5 + 2.5 * X[, 1] + 3 * X[, 2] + u
  coefs1[i, ] <- lm(Y ~ X[, 1] + X[, 2])$coefficients[-1]

  # for cov(X_1,X_2) = 0.85
  X <- rmvnorm(n, c(0, 0), sigma = cbind(c(10, 8.5), c(8.5, 10)))
  Y <- 5 + 2.5 * X[, 1] + 3 * X[, 2] + u
  coefs2[i, ] <- lm(Y ~ X[, 1] + X[, 2])$coefficients[-1]

}

# obtain variance estimates
diag(var(coefs1))
#> hat_beta_1 hat_beta_2
#> 0.05674375 0.05712459
diag(var(coefs2))
#> hat_beta_1 hat_beta_2
#> 0.1904949 0.1909056

```

We are interested in the variances which are the diagonal elements. We see that due to the high collinearity, the variances of $\hat{\beta}_1$ and $\hat{\beta}_2$ have more than tripled, meaning it is more difficult to precisely estimate the true coefficients.

6.5 The Distribution of the OLS Estimators in Multiple Regression

As in simple linear regression, different samples will produce different values of the OLS estimators in the multiple regression model. Again, this variation leads to uncertainty of those estimators which we seek to describe using their sampling distribution(s). In short, if the assumption made in Key Concept 6.4 hold, the large sample distribution of $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ become multivariate normal such that the individual estimators themselves are also normally distributed. Key Concept 6.5 summarizes the corresponding statements made in Chapter 6.6 of the book. A more technical derivation of these results can be found in Chapter 18 of the book.

Key Concept 6.5**Large-sample distribution of $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$**

If the least squares assumptions in the multiple regression model (see Key Concept 6.4) hold, then, in large samples, the OLS estimators $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ are jointly normally distributed. We also say that their joint distribution is *multivariate* normal. Further, each $\hat{\beta}_j$ is distributed as $\mathcal{N}(\beta_j, \sigma_{\beta_j}^2)$.

Essentially, Key Concept 6.5 states that, if the sample size is large, we can approximate the individual sampling distributions of the coefficient estimators by specific normal distributions and their joint sampling distribution by a multivariate normal distribution.

How can we use R to get an idea of what the joint PDF of the coefficient estimators in multiple regression model looks like? When estimating a model on some data, we end up with a set of point estimates that do not reveal much information on the *joint* density of the estimators. However, with a large number of estimations using repeatedly randomly sampled data from the same population, we can generate a large set of point estimates that allows us to plot an *estimate* of the joint density function.

The approach to do this in R is as follows:

- Generate 10000 random samples of size 50 using the DGP

$$Y_i = 5 + 2.5 \cdot X_{1i} + 3 \cdot X_{2i} + u_i$$

where the regressors X_{1i} and X_{2i} are sampled for each observation as

$$X_i = (X_{1i}, X_{2i}) \sim \mathcal{N} \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 10 & 2.5 \\ 2.5 & 10 \end{pmatrix} \right]$$

and

$$u_i \sim \mathcal{N}(0, 5)$$

is an error term.

- For each of the 10000 simulated sets of sample data, we estimate the model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + u_i$$

and save the coefficient estimates $\hat{\beta}_1$ and $\hat{\beta}_2$.

- We compute a density estimate of the joint distribution of $\hat{\beta}_1$ and $\hat{\beta}_2$ in the model above using the function `kde2d()` from the package `MASS`, see `?MASS`. This estimate is then plotted using the function `persp()`.

```
# load packages
library(MASS)
library(mvtnorm)

# set sample size
n <- 50

# initialize vector of coefficients
coefs <- cbind("hat_beta_1" = numeric(10000), "hat_beta_2" = numeric(10000))

# set seed for reproducibility
set.seed(1)
```

```

# loop sampling and estimation
for (i in 1:10000) {

  X <- rmvnorm(n, c(50, 100), sigma = cbind(c(10, 2.5), c(2.5, 10)))
  u <- rnorm(n, sd = 5)
  Y <- 5 + 2.5 * X[, 1] + 3 * X[, 2] + u
  coefs[i,] <- lm(Y ~ X[, 1] + X[, 2])$coefficients[-1]

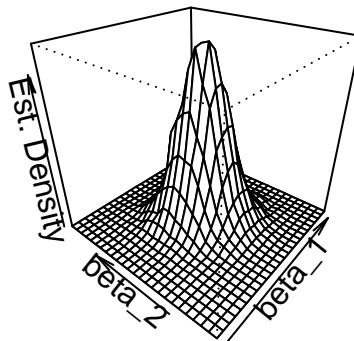
}

# compute density estimate
kde <- kde2d(coefs[, 1], coefs[, 2])

# plot density estimate
persp(kde,
      theta = 310,
      phi = 30,
      xlab = "beta_1",
      ylab = "beta_2",
      zlab = "Est. Density",
      main = "2D Kernel Density Estimate")

```

2D Kernel Density Estimate



From the plot above we can see that the density estimate has some similarity to a bivariate normal distribution (see Chapter 2) though it is not very pretty and probably a little rough. Furthermore, there is a correlation between the estimates such that $\rho \neq 0$ in (2.1). Also, the distribution's shape deviates from the symmetric bell shape of the bivariate standard normal distribution and has an elliptical surface area instead.

```

# estimate the correlation between estimators
cor(coefs[, 1], coefs[, 2])
#> [1] -0.2503028

```

Where does this correlation come from? Notice that, due to the way we generated the data, correlation between the regressors X_1 and X_2 appears. Correlation between the regressors in a multiple regression model always translates into correlation between the estimators (see Appendix 6.2 of the book). In our case, the positive correlation between X_1 and X_2 translates to negative correlation between $\hat{\beta}_1$ and $\hat{\beta}_2$. To get a better idea of the distribution you can vary the point of view in the subsequent smooth

interactive 3D plot of the same density estimate used for plotting with `persp()`. Here you can see that the shape of the distribution is somewhat stretched due to $\rho = -0.20$ and it is also apparent that both estimators are unbiased since their joint density seems to be centered close to the true parameter vector $(\beta_1, \beta_2) = (2.5, 3)$.

This interactive part of the book is only available in the HTML version.

6.6 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 7

Hypothesis Tests and Confidence Intervals in MR Models

This chapter discusses methods that allow to quantify the sampling uncertainty in the OLS estimator of the coefficients in multiple regression models. The basis for this are hypothesis tests and confidence intervals which, just as for the simple linear regression model, can be computed using basic R functions. We will also tackle the issue of testing joint hypotheses on these coefficients.

Make sure the packages **AER** (Kleiber and Zeileis, 2008b) and **stargazer** (Hlavac, 2022) are installed before you go ahead and replicate the examples. The safest way to do so is by checking whether the following code chunk executes without any issues.

```
library(AER)
library(stargazer)
```

7.1 Hypothesis Tests and Confidence Intervals for a Single Coefficient

We will first discuss how to compute standard errors, test hypotheses and construct confidence intervals for a single regression coefficient β_j in a multiple regression model. The basic idea is summarized in Key Concept 7.1.

Key Concept 7.1**Testing the Hypothesis $\beta_j = \beta_{j,0}$ Against the Alternative $\beta_j \neq \beta_{j,0}$**

1. Compute the standard error of $\hat{\beta}_j$.
2. Compute the t -statistic,

$$t^{act} = \frac{\hat{\beta}_j - \beta_{j,0}}{SE(\hat{\beta}_j)}$$

3. Compute the p -value,

$$p\text{-value} = 2\Phi(-|t^{act}|)$$

where t^{act} is the value of the t -statistic actually computed. Reject the hypothesis at the 5% significance level if the p -value is less than 0.05 or, equivalently, if $|t^{act}| > 1.96$.

The standard error and (typically) the t -statistic and the corresponding p -value for testing $\beta_j = 0$ are computed automatically by suitable R functions, e.g., by `summary()`.

Testing a single hypothesis about the significance of a coefficient in the multiple regression model proceeds similarly to the process in the simple regression model.

You can easily see this by inspecting the coefficient summary of the regression model

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + u$$

already discussed in Chapter 6. Let us review this:

```
model <- lm(score ~ STR + english, data = CASchools)
coefTest(model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error  t value Pr(>|t|)
#> (Intercept) 686.032245   8.728225   78.5993 < 2e-16 ***
#> STR          -1.101296   0.432847  -2.5443 0.01131 *
#> english      -0.649777   0.031032 -20.9391 < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can verify that these quantities are computed as in the simple regression model by manually calculating the t -statistics or p -values using the provided output above and using R as a calculator.

For example, using the definition of the p -value for a two-sided test as given in Key Concept 7.1, we can confirm the p -value for a test of the hypothesis that the coefficient β_1 , the coefficient on `size`, to be approximately zero.

```
# compute two-sided p-value
2 * (1 - pt(abs(coefTest(model, vcov. = vcovHC, type = "HC1")[2, 3]),
            df = model$df.residual))
#> [1] 0.01130921
```

Key Concept 7.2**Confidence Intervals for a Single Coefficient in Multiple Regression**

A 95% two-sided confidence interval for the coefficient β_j is an interval that contains the true value of β_j with a 95% probability; that is, it contains the true value of β_j in 95% of repeated samples. Equivalently, it is the set of values of β_j that cannot be rejected by a 5% two-sided hypothesis test. When the sample size is large, the 95% confidence interval for β_j is

$$\left[\hat{\beta}_j - 1.96 \times SE(\hat{\beta}_j), \hat{\beta}_j + 1.96 \times SE(\hat{\beta}_j) \right].$$

7.2 An Application to Test Scores and the Student-Teacher Ratio

Let us take a look at the regression from Section 6.3 again.

Computing confidence intervals for individual coefficients in the multiple regression model proceeds as in the simple regression model using the function `confint()`.

```
model <- lm(score ~ STR + english, data = CASchools)
confint(model)
#>                2.5 %      97.5 %
#> (Intercept) 671.4640580 700.6004311
#> STR          -1.8487969  -0.3537944
#> english     -0.7271113  -0.5724424
```

To obtain confidence intervals at another level, say 90%, just set the argument `level` in our call of `confint()` accordingly.

```
confint(model, level = 0.9)
#>                5 %      95 %
#> (Intercept) 673.8145793 698.2499098
#> STR          -1.7281904  -0.4744009
#> english     -0.7146336  -0.5849200
```

The output now reports the desired 90% confidence intervals for all coefficients.

One drawback of using `confint()` is that it doesn't utilize robust standard errors for calculating the confidence interval. To create large-sample confidence intervals that account for robust standard errors, you can easily do manually using the following approach.

```
# compute robust standard errors
rob_se <- diag(vcovHC(model, type = "HC1"))^0.5

# compute robust 95% confidence intervals
rbind("lower" = coef(model) - qnorm(0.975) * rob_se,
      "upper" = coef(model) + qnorm(0.975) * rob_se)
#>      (Intercept)      STR      english
#> lower    668.9252 -1.9496606 -0.7105980
#> upper    703.1393 -0.2529307 -0.5889557

# compute robust 90% confidence intervals
rbind("lower" = coef(model) - qnorm(0.95) * rob_se,
```

```

"upper" = coef(model) + qnorm(0.95) * rob_se)
#>      (Intercept)      STR      english
#> lower    671.6756 -1.8132659 -0.7008195
#> upper    700.3889 -0.3893254 -0.5987341

```

Knowing how to use R to make inference about the coefficients in multiple regression models, you can now answer the following question:

Can the null hypothesis that a change in the student-teacher ratio, *STR*, has no significant influence on test scores, *scores*, — if we control for the percentage of students learning English in the district, *english*, — be rejected at the 10% and the 5% level of significance?

The output above shows that zero is not an element of the confidence interval for the coefficient on *STR* such that we can reject the null hypothesis at significance levels of 5% and 10%. The same conclusion can be made via the *p*-value for *STR*: $0.00398 < 0.05 = \alpha$.

Note that rejection at the 5%-level implies rejection at the 10% level (why?).

Recall from Chapter 5.2 the 95% confidence interval computed above *does not* tell us that a one-unit decrease in the student-teacher ratio has an effect on test scores that lies in the interval with a lower bound of -1.9497 and an upper bound of -0.2529 . Once a confidence interval has been computed, a probabilistic statement like this is wrong: either the interval contains the true parameter or it does not. We do not know which is true.

Another Augmentation of the Model

What is the average effect on test scores of reducing the student-teacher ratio when the expenditures per pupil and the percentage of english learning pupils are held constant?

Let us augment our model by an additional regressor that is a measure for expenditure per pupil. Using `?CASchools` we find that `CASchools` contains the variable `expenditure`, which provides expenditure per student.

Our model now is

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_3 \times expenditure + u$$

with *expenditure* being the total amount of expenditure per pupil in the district (thousands of dollars).

Let us now estimate the model:

```

# scale expenditure to thousands of dollars
CASchools$expenditure <- CASchools$expenditure/1000

# estimate the model
model <- lm(score ~ STR + english + expenditure, data = CASchools)
coeftest(model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>      Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 649.577947  15.458344  42.0212 < 2e-16 ***
#> STR         -0.286399   0.482073  -0.5941 0.55277
#> english     -0.656023   0.031784 -20.6398 < 2e-16 ***
#> expenditure  3.867901    1.580722   2.4469 0.01482 *

```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated effect of a one unit change in the student-teacher ratio on test scores with expenditure and the share of english learning pupils held constant is -0.29 , which is rather small. What is more, the coefficient on STR is not significantly different from zero anymore even at 10% since $p\text{-value} = 0.55$. Can you come up with an interpretation for these findings (see Chapter 7.1 of the book)? The insignificance of $\hat{\beta}_1$ could be due to a larger standard error of $\hat{\beta}_1$ resulting from adding *expenditure* to the model so that we estimate the coefficient on *size* less precisely. This illustrates the issue of strongly correlated regressors (imperfect multicollinearity). The correlation between STR and *expenditure* can be computed using `cor()`.

```
# compute the sample correlation between 'STR' and 'expenditure'
cor(CASchools$STR, CASchools$expenditure)
#> [1] -0.6199822
```

Altogether, we conclude that the new model provides no evidence that changing the student-teacher ratio, e.g., by hiring new teachers, has any effect on the test scores while keeping expenditures per student and the share of English learners constant.

7.3 Joint Hypothesis Testing using the F-Statistic

The estimated model is

$$\widehat{TestScore} = 649.58 - 0.29 \times STR - 0.66 \times english + 3.87 \times expenditure.$$

(15.21) (0.48) (0.04) (1.41)

Now, can we reject the hypothesis that the coefficient on *size* and the coefficient on *expenditure* are zero? To answer this, we have to resort to joint hypothesis tests. A joint hypothesis imposes restrictions on multiple regression coefficients. This is different from conducting individual t -tests where a restriction is imposed on a single coefficient. Chapter 7.2 of the book explains why testing hypotheses about the model coefficients one at a time is different from testing them jointly.

The homoskedasticity-only F -Statistic is given by

$$F = \frac{(SSR_{restricted} - SSR_{unrestricted})/q}{SSR_{unrestricted}/(n - k - 1)}$$

with $SSR_{restricted}$ being the sum of squared residuals from the restricted regression, i.e., the regression where we impose the restriction. $SSR_{unrestricted}$ is the sum of squared residuals from the full model, q is the number of restrictions under the null and k is the number of regressors in the unrestricted regression.

It is fairly easy to conduct F -tests in R. We can use the function `linearHypothesis()` contained in the package `car`.

```
# estimate the multiple regression model
model <- lm(score ~ STR + english + expenditure, data = CASchools)

# execute the function on the model object and provide both linear restrictions
# to be tested as strings
linearHypothesis(model, c("STR=0", "expenditure=0"))
#> Linear hypothesis test
```

```

#>
#> Hypothesis:
#> STR = 0
#> expenditure = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + english + expenditure
#>
#>   Res.Df  RSS Df Sum of Sq    F  Pr(>F)
#> 1     418 89000
#> 2     416 85700  2   3300.3 8.0101 0.000386 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The output reveals that the F -statistic for this joint hypothesis test is about 8.01 and the corresponding p -value is 0.0004. Thus, we can reject the null hypothesis that both coefficients are zero at any level of significance commonly used in practice.

A heteroskedasticity-robust version of this F -test (which leads to the same conclusion) can be conducted as follows:

```

# heteroskedasticity-robust F-test
linearHypothesis(model, c("STR=0", "expenditure=0"), white.adjust = "hc1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> STR = 0
#> expenditure = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + english + expenditure
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df    F  Pr(>F)
#> 1     418
#> 2     416  2 5.4337 0.004682 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The standard output of a model summary also reports an F -statistic and the corresponding p -value. The null hypothesis belonging to this F -test is that *all* of the population coefficients in the model except for the intercept are zero, so the hypotheses are

$$H_0 : \beta_1 = 0, \beta_2 = 0, \beta_3 = 0 \quad \text{vs.} \quad H_1 : \beta_j \neq 0 \text{ for at least one } j = 1, 2, 3.$$

This is also called the *overall regression F-statistic* and the null hypothesis is obviously different from testing if only β_1 and β_3 are zero.

We now check whether the F -statistic belonging to the p -value listed in the model's summary coincides with the result reported by `linearHypothesis()`.

```

# execute the function on the model object and provide the restrictions
# to be tested as a character vector

```

```

linearHypothesis(model, c("STR=0", "english=0", "expenditure=0"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> STR = 0
#> english = 0
#> expenditure = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + english + expenditure
#>
#>   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
#> 1     419 152110
#> 2     416  85700  3     66410 107.45 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Access the overall F-statistic from the model's summary
summary(model)$fstatistic
#>   value  numdf  dendf
#> 107.4547  3.0000 416.0000

```

The entry `value` is the overall F -statistics and it equals the result of `linearHypothesis()`. The F -test rejects the null hypothesis that the model has no power in explaining test scores. It is important to know that the F -statistic reported by `summary` is *not* robust to heteroskedasticity.

7.4 Confidence Sets for Multiple Coefficients

Based on the F -statistic that we have previously encountered, we can specify confidence sets. Confidence sets are analogous to confidence intervals for single coefficients. As such, confidence sets consist of *combinations* of coefficients that contain the true combination of coefficients in, say, 95% of all cases if we could repeatedly draw random samples, just like in the univariate case. Put differently, a confidence set is the set of all coefficient combinations for which we cannot reject the corresponding joint null hypothesis tested using an F -test.

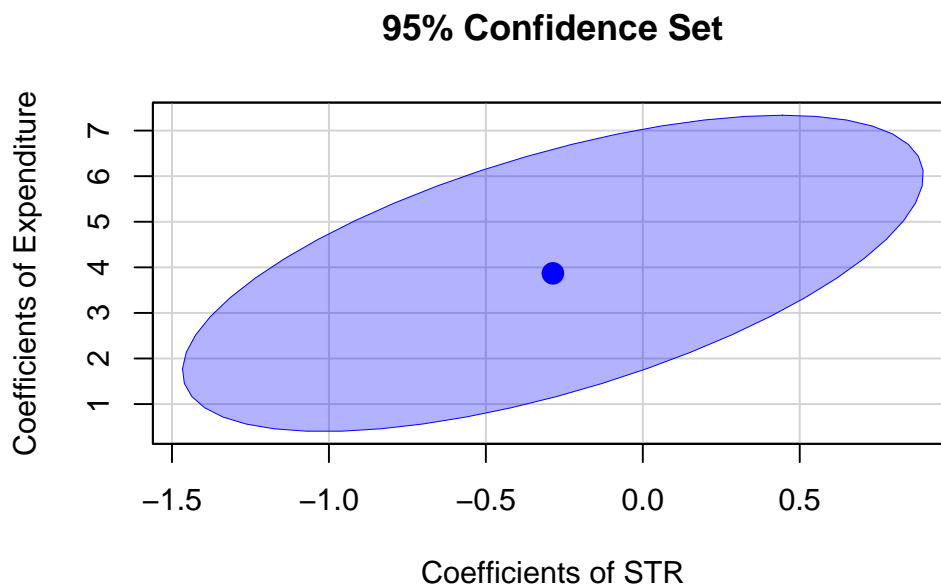
The confidence set for two coefficients is an ellipse which is centered around the point defined by both coefficient estimates. Again, there is a very convenient way to plot the confidence set for two coefficients of model objects, namely the function `confidenceEllipse()` from the `car` package.

We now plot the 95% confidence ellipse for the coefficients on `STR` and `expenditure` from the regression conducted above. By specifying the additional argument `fill`, the confidence set is colored.

```

# draw the 95% confidence set for coefficients on STR and expenditure
confidenceEllipse(model,
  fill = T,
  lwd = 0,
  which.coef = c("STR", "expenditure"),
  main = "95% Confidence Set",
  ylab="Coefficients of Expenditure",
  xlab="Coefficients of STR")

```

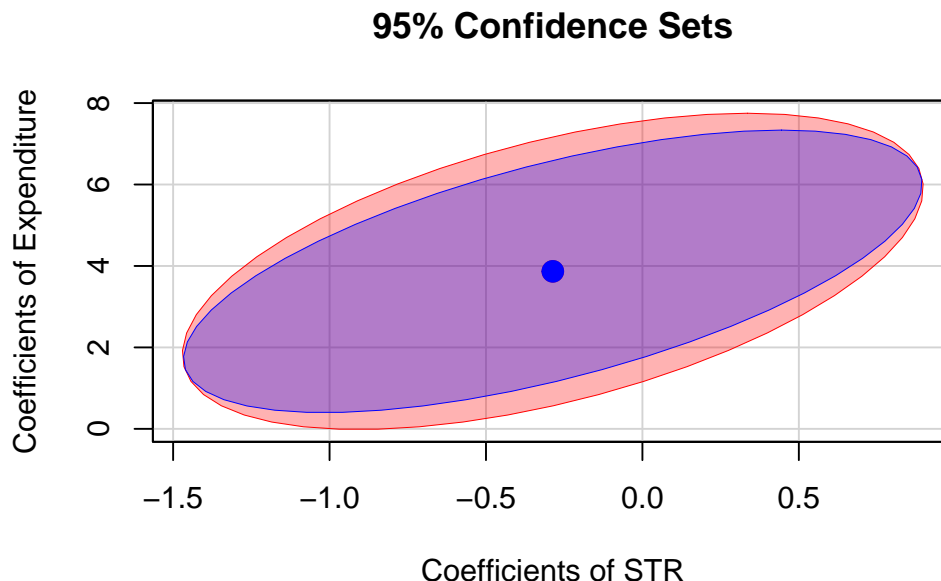


We see that the ellipse is centered around $(-0.29, 3.87)$, the pair of coefficients estimates on *STR* and *expenditure*. What is more, $(0, 0)$ is not element of the 95% confidence set so that we can reject $H_0 : \beta_1 = 0, \beta_3 = 0$.

By default, `confidenceEllipse()` uses homoskedasticity-only standard errors. The following code chunk shows how to compute a robust confidence ellipse and how to overlay it with the previous plot.

```
# draw the robust 95% confidence set for coefficients on STR and expenditure
confidenceEllipse(model,
  fill = T,
  lwd = 0,
  which.coef = c("STR", "expenditure"),
  main = "95% Confidence Sets",
  vcov. = vcovHC(model, type = "HC1"),
  col = "red",
  ylab="Coefficients of Expenditure",
  xlab="Coefficients of STR")

# draw the 95% confidence set for coefficients on STR and expenditure
confidenceEllipse(model,
  fill = T,
  lwd = 0,
  which.coef = c("STR", "expenditure"),
  add = T)
```



As the robust standard errors are slightly larger than those valid under homoskedasticity only in this case, the robust confidence set is slightly larger. This is analogous to the confidence intervals for the individual coefficients.

7.5 Model Specification for Multiple Regression

Choosing a regression specification, i.e., selecting the variables to be included in a regression model, is a difficult task. However, there are some guidelines on how to proceed. The goal is clear: obtaining an unbiased and precise estimate of the causal effect of interest. As a starting point, think about omitted variables, that is, to avoid possible bias by using suitable control variables. Omitted variables bias in the context of multiple regression is explained in Key Concept 7.3. A second step could be to compare different specifications by measures of fit. However, as we shall see one should not rely solely on \bar{R}^2 .

Key Concept 7.3

Omitted Variable Bias in Multiple Regression

Omitted variable bias is the bias in the OLS estimator that arises when regressors correlate with an omitted variable. For omitted variable bias to arise, two things must be true:

1. At least one of the included regressors must be correlated with the omitted variable.
2. The omitted variable must be a determinant of the dependent variable, Y .

We will now discuss an example where we may encounter potential omitted variable bias in a multiple regression model:

Consider again the estimated regression equation

$$\widehat{TestScore} = 686.0 - 1.10 \times STR - 0.650 \times english.$$

(8.7) (0.43) (0.031)

We are interested in estimating the causal effect of class size on test score. There might be a bias due to omitting “outside learning opportunities” from our regression since such a measure could be

a determinant of the students' test scores and could also be correlated with both regressors already included in the model (so that both conditions of Key Concept 7.3 are fulfilled). “Outside learning opportunities” are a complicated concept that is difficult to quantify. A surrogate we can consider instead is the students' economic background which are strongly related to outside learning opportunities: think of wealthy parents that are able to provide time and/or money for private tuition of their children. We thus augment the model with the variable `lunch`, the percentage of students that qualify for a free or subsidized lunch in school due to family incomes below a certain threshold, and reestimate the model.

```
# estimate the model and print the summary to console
model <- lm(score ~ STR + english + lunch, data = CASchools)
coeftest(model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 700.149957  5.568453 125.7351 < 2.2e-16 ***
#> STR          -0.998309  0.270080  -3.6963 0.0002480 ***
#> english      -0.121573  0.032832  -3.7029 0.0002418 ***
#> lunch        -0.547345  0.024107 -22.7046 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, the estimated regression line is

$$\widehat{TestScore} = 700.15 - 1.00 \times STR - 0.12 \times english - 0.55 \times lunch.$$

(5.56) (0.27) (0.03) (0.02)

We observe no substantial changes in the conclusion about the effect of `STR` on `TestScore`: the coefficient on `STR` changes by only 0.1 and retains its significance.

Although the difference in estimated coefficients is not big in this case, it is useful to keep `lunch` to make the assumption of conditional mean independence more credible (see Chapter 7.5 of the book).

Model Specification in Theory and in Practice

Key Concept 7.4 lists some common pitfalls when using R^2 and \bar{R}^2 to evaluate the predictive ability of regression models.

Key Concept 7.4 **R^2 and \bar{R}^2 : What They Tell You — and What They Do not**

The R^2 and \bar{R}^2 tell you whether the regressors are good at explaining the variation of the independent variable in the sample. If the R^2 (or \bar{R}^2) is nearly 1, then the regressors produce a good prediction of the dependent variable in that sample, in the sense that the variance of OLS residuals is small compared to the variance of the dependent variable. If the R^2 (or \bar{R}^2) is nearly 0, the opposite is true.

The R^2 and \bar{R}^2 do *not* tell you whether:

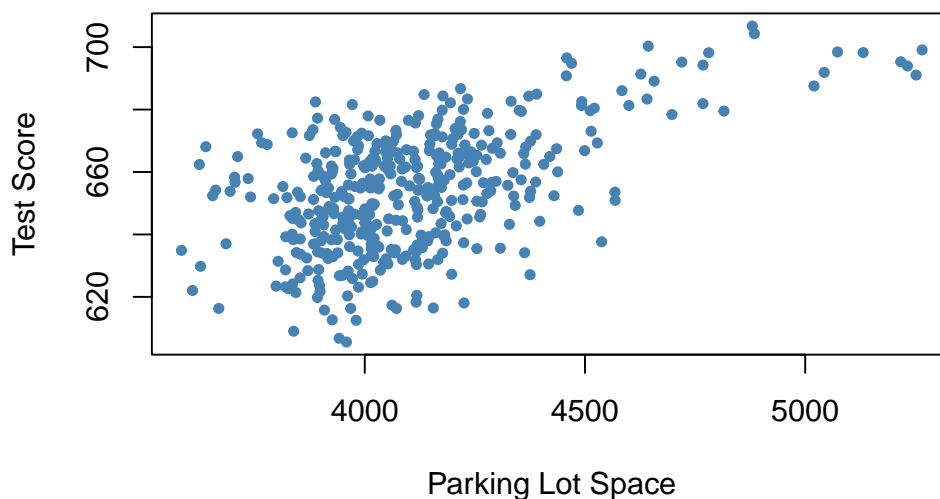
1. An included variable is statistically significant.
2. The regressors are the true cause of the movements in the dependent variable.
3. There is omitted variable bias.
4. You have chosen the most appropriate set of regressors.

For example, think of regressing *TestScore* on *PLS* which measures the available parking lot space in thousand square feet. You are likely to observe a significant coefficient of reasonable magnitude and moderate to high values for R^2 and \bar{R}^2 . The reason for this is that parking lot space is correlated with many determinants of the test score like location, class size, financial endowment and so on. Although we do not have observations on *PLS*, we can use R to generate some relatively realistic data.

```
# set seed for reproducibility
set.seed(1)

# generate observations for parking lot space
CASchools$PLS <- c(22 * CASchools$income
                  - 15 * CASchools$STR
                  + 0.2 * CASchools$expenditure
                  + rnorm(nrow(CASchools), sd = 80) + 3000)

# plot parking lot space against test score
plot(CASchools$PLS,
     CASchools$score,
     xlab = "Parking Lot Space",
     ylab = "Test Score",
     pch = 20,
     col = "steelblue")
```



```
# regress test score on PLS
summary(lm(score ~ PLS, data = CASchools))
#>
#> Call:
#> lm(formula = score ~ PLS, data = CASchools)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -42.608 -11.049   0.342  12.558  37.105
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 4.897e+02  1.227e+01   39.90  <2e-16 ***
#> PLS          4.002e-02  2.981e-03   13.43  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 15.95 on 418 degrees of freedom
#> Multiple R-squared:  0.3013, Adjusted R-squared:  0.2996
#> F-statistic: 180.2 on 1 and 418 DF,  p-value: < 2.2e-16
```

PLS is generated as a linear function of $expenditure$, $income$, STR and a random disturbance. Therefore the data suggest that there is some positive relationship between parking lot space and test score. In fact, when estimating the model

$$TestScore = \beta_0 + \beta_1 \times PLS + u. \quad (7.1)$$

using `lm()` we find that the coefficient on PLS is positive and significantly different from zero. Also R^2 and \bar{R}^2 are about 0.3 which is a lot more than the roughly 0.05 observed when regressing the test scores on the class sizes only. This suggests that increasing the parking lot space boosts a school's test scores and that model (7.1) does even better in explaining heterogeneity in the dependent variable than a model with STR as the only regressor. Keeping in mind how PLS is constructed this comes as no surprise. It is evident that the high R^2 cannot be used to conclude that the estimated relation between parking lot space and test scores is causal: the (relatively) high R^2 is due to correlation between PLS and other determinants and/or control variables. Increasing parking lot space is *not* an appropriate measure to generate more learning success!

7.6 Analysis of the Test Score Data Set

Chapter 6 and some of the previous sections have stressed that it is important to include control variables in regression models if it is plausible that there are omitted factors. In our example of test scores, we want to estimate the causal effect of a change in the student-teacher ratio on test scores. We now provide an example of how to use multiple regression in order to alleviate omitted variable bias and demonstrate how to report this results using R.

So far we have considered two variables that control for unobservable student characteristics which correlate with the student-teacher ratio *and* are assumed to have an impact on test scores:

- *English*, the percentage of English learning students.
- *lunch*, the share of students that qualify for a subsidized or even a free lunch at school.

Another new variable provided with `CASchools` is `calworks`, the percentage of students that qualify for the *CalWorks* income assistance program. Students eligible for *CalWorks* live in families with a total income below the threshold for the subsidized lunch program so both variables are indicators for the share of economically disadvantaged children. Both indicators are highly correlated.

```
# estimate the correlation between 'calworks' and 'lunch'
cor(CASchools$calworks, CASchools$lunch)
#> [1] 0.7394218
```

There is no unambiguous way to proceed when deciding which variable to use. In any case it may not be a good idea to use both variables as regressors in view of collinearity. Therefore, we also consider alternative model specifications.

For a start, we plot student characteristics against test scores.

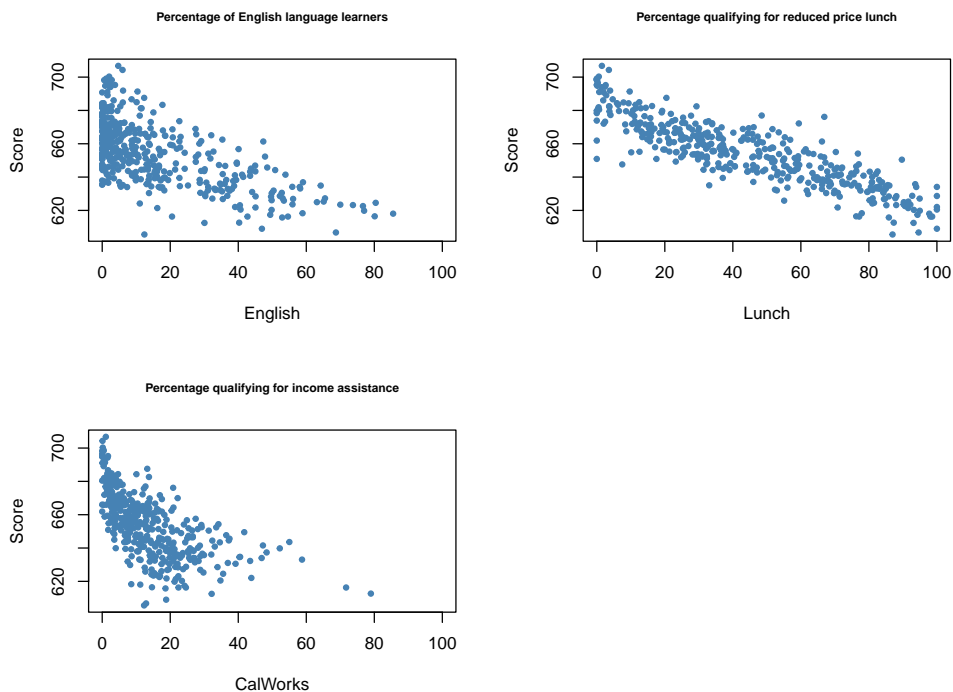
```
# set up arrangement of plots
m <- rbind(c(1, 2), c(3, 0))
graphics::layout(mat = m)

# scatterplots
plot(score ~ english,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     xlim = c(0, 100),
     cex.main = 0.7,
     xlab="English",
     ylab="Score",
     main = "Percentage of English language learners")

plot(score ~ lunch,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     cex.main = 0.7,
     xlab="Lunch",
     ylab="Score",
     main = "Percentage qualifying for reduced price lunch")

plot(score ~ calworks,
     data = CASchools,
```

```
col = "steelblue",
pch = 20,
xlim = c(0, 100),
cex.main = 0.7,
xlab="CalWorks",
ylab="Score",
main = "Percentage qualifying for income assistance")
```



We divide the plotting area up using `layout()`. The matrix `m` specifies the location of the plots, see `?layout`.

We see that all relationships are negative. Here are the correlation coefficients.

```
# estimate correlation between student characteristics and test scores
cor(CASchools$score, CASchools$english)
#> [1] -0.6441238
cor(CASchools$score, CASchools$lunch)
#> [1] -0.868772
cor(CASchools$score, CASchools$calworks)
#> [1] -0.6268533
```

We shall consider five different model equations:

- (I) $TestScore = \beta_0 + \beta_1 \times STR + u$,
- (II) $TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + u$,
- (III) $TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_3 \times lunch + u$,
- (IV) $TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_4 \times calworks + u$,
- (V) $TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_3 \times lunch + \beta_4 \times calworks + u$.

The best way to communicate regression results is in a table. The `stargazer` package is very convenient for this purpose. It provides a function that generates professionally looking HTML and LaTeX tables that satisfy scientific standards. One simply has to provide one or multiple object(s) of class `lm`. The rest is done by the function `stargazer()`.

```
# load the stargazer library
library(stargazer)

# estimate different model specifications
spec1 <- lm(score ~ STR, data = CASchools)
spec2 <- lm(score ~ STR + english, data = CASchools)
spec3 <- lm(score ~ STR + english + lunch, data = CASchools)
spec4 <- lm(score ~ STR + english + calworks, data = CASchools)
spec5 <- lm(score ~ STR + english + lunch + calworks, data = CASchools)

# gather robust standard errors in a list
rob_se <- list(sqrt(diag(vcovHC(spec1, type = "HC1"))),
               sqrt(diag(vcovHC(spec2, type = "HC1"))),
               sqrt(diag(vcovHC(spec3, type = "HC1"))),
               sqrt(diag(vcovHC(spec4, type = "HC1"))),
               sqrt(diag(vcovHC(spec5, type = "HC1"))))

# generate a LaTeX table using stargazer
stargazer(spec1, spec2, spec3, spec4, spec5,
          se = rob_se,
          digits = 3,
          header = F,
          column.labels = c("(I)", "(II)", "(III)", "(IV)", "(V)"))
```

Table 7.1 states that *score* is the dependent variable and that we consider five models. We see that the columns of Table 7.1 contain most of the information provided by `coefest()` and `summary()` for the regression models under consideration: the coefficients estimates equipped with significance codes (the asterisks) and standard errors in parentheses below. Although there are no *t*-statistics, it is straightforward for the reader to compute them simply by dividing a coefficient estimate by the corresponding standard error. The bottom of the table reports summary statistics for each model and a legend. For an in-depth discussion of the tabular presentation of regression results, see Chapter 7.6 of the book.

What can we conclude from the model comparison?

1. We see that adding control variables roughly halves the coefficient on `STR`. Also, the estimate is not sensitive to the set of control variables used. The conclusion is that decreasing the student-teacher ratio *ceteris paribus* by one unit leads to an estimated average increase in test scores of about 1 point.
2. Adding student characteristics as controls increases R^2 and \bar{R}^2 from 0.049 (`spec1`) up to 0.773 (`spec3` and `spec5`), so we can consider these variables as suitable predictors for test scores. Moreover, the estimated coefficients on all control variables are consistent with the impressions gained from Figure 7.2 of the book.
3. We see that the control variables are not statistically significant in all models. For example in `spec5`, the coefficient on *calworks* is not significantly different from zero at 5% since $|-0.048/0.059| = 0.81 < 1.64$. We also observe that the effect on the estimate (and its standard error) of the coefficient on *size* of adding *calworks* to the base specification `spec3` is negligible. We can therefore consider *calworks* as a superfluous control variable, given the inclusion of *lunch* in this

Table 7.1: Regressions of Test Scores on the Student-Teacher Ratio and Control Variables

	Dependent Variable: Test Score				
	(I) spec1	(II) spec2	score (III) spec3	(IV) spec4	(V) spec5
STR	-2.280*** (0.519)	-1.101** (0.433)	-0.998*** (0.270)	-1.308*** (0.339)	-1.014*** (0.269)
english		-0.650*** (0.031)	-0.122*** (0.033)	-0.488*** (0.030)	-0.130*** (0.036)
lunch			-0.547*** (0.024)		-0.529*** (0.038)
calworks				-0.790*** (0.068)	-0.048 (0.059)
Constant	698.933*** (10.364)	686.032*** (8.728)	700.150*** (5.568)	697.999*** (6.920)	700.392*** (5.537)
Observations	420	420	420	420	420
R ²	0.051	0.426	0.775	0.629	0.775
Adjusted R ²	0.049	0.424	0.773	0.626	0.773
Residual Std. Error	18.581 (df = 418)	14.464 (df = 417)	9.080 (df = 416)	11.654 (df = 416)	9.084 (df = 415)
F Statistic	22.575*** (df = 1; 418)	155.014*** (df = 2; 417)	476.306*** (df = 3; 416)	234.638*** (df = 3; 416)	357.054*** (df = 4; 415)

Note: *p<0.1; **p<0.05; ***p<0.01

model.

7.7 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 8

Nonlinear Regression Functions

Until now we assumed the regression function to be linear, i.e., we have treated the slope parameter of the regression function as a constant. This implies that the effect on Y of a one unit change in X does not depend on the level of X . If, however, the effect of a change in X on Y does depend on the value of X , we should use a nonlinear regression function.

Just like for the previous chapter, the packages `AER` (Kleiber and Zeileis, 2008b) and `stargazer` (Hlavac, 2022) are required for reproduction of the code presented in this chapter. Check whether the code chunk below executes without any error messages.

```
library(AER)
library(stargazer)
```

8.1 A General Strategy for Modelling Nonlinear Regression Functions

Let us have a look at an example where using a nonlinear regression function is better suited for estimating the population relationship between the regressor, X , and the regressand, Y : the relationship between the income of schooling districts and their test scores.

```
# prepare the data
library(AER)
data(CASchools)
CASchools$size <- CASchools$students/CASchools$teachers
CASchools$score <- (CASchools$read + CASchools$math) / 2
```

We start our analysis by computing the correlation between both variables.

```
cor(CASchools$income, CASchools$score)
#> [1] 0.7124308
```

Here, income and test scores are positively related: school districts with above average income tend to achieve above average test scores. Does a linear regression function model the data adequately? Let us plot the data and add a linear regression line.

```
# fit a simple linear model
linear_model <- lm(score ~ income, data = CASchools)
```

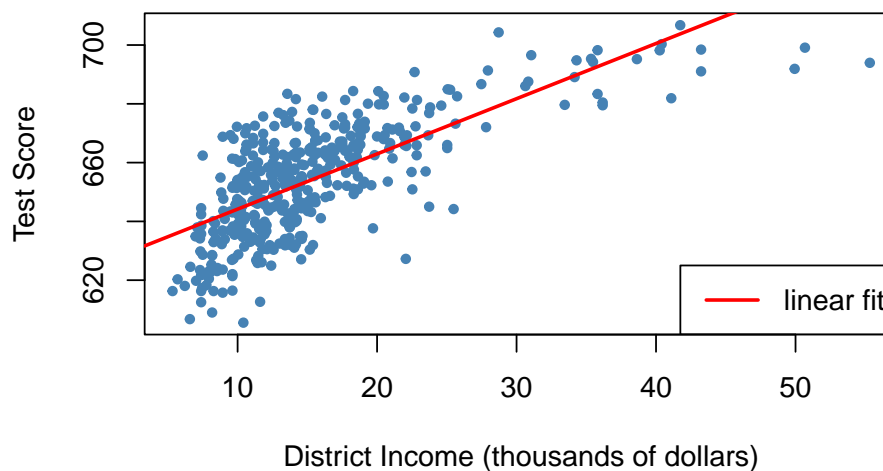
```

# plot the observations
plot(CASchools$income, CASchools$score,
     col = "steelblue",
     pch = 20,
     xlab = "District Income (thousands of dollars)",
     ylab = "Test Score",
     cex.main = 0.9,
     main = "Test Score vs. District Income and a Linear OLS Regression Function")

# add the regression line to the plot
abline(linear_model,
       col = "red",
       lwd = 2)
legend("bottomright", legend="linear fit", lwd=2, col="red")

```

Test Score vs. District Income and a Linear OLS Regression Function



As pointed out in the book, the linear regression line seems to overestimate the true relationship when income is very high or very low and underestimates it for the middle income group.

Fortunately, OLS does not only handle linear functions of the regressors. We can, for example, model test scores as a function of income and the square of income. The corresponding regression model is

$$TestScore_i = \beta_0 + \beta_1 \times income_i + \beta_2 \times income_i^2 + u_i,$$

called a *quadratic regression model*. That is, $income^2$ is treated as an additional explanatory variable. Hence, the quadratic model is a special case of a multivariate regression model. When fitting the model with `lm()` we have to use the `^` operator in conjunction with the function `I()` to add the quadratic term as an additional regressor to the argument `formula`. This is because the regression formula we pass to `formula` is converted to an object of the class `formula`. For objects of this class, the operators `+`, `-`, `*` and `^` have a nonarithmetic interpretation. `I()` ensures that they are used as arithmetical operators, see `?I`,

```

# fit the quadratic Model
quadratic_model <- lm(score ~ income + I(income^2), data = CASchools)

# obtain the model summary

```

```
coeftest(quadratic_model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 607.3017435   2.9017544 209.2878 < 2.2e-16 ***
#> income      3.8509939   0.2680942  14.3643 < 2.2e-16 ***
#> I(income^2) -0.0423084   0.0047803  -8.8505 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output tells us that the estimated regression function is

$$\widehat{TestScore}_i = 607.3 + \underset{(2.90)}{3.85} \times income_i - \underset{(0.0048)}{0.0423} \times income_i^2.$$

This model allows us to test the hypothesis that the relationship between test scores and district income is linear against the alternative that it is quadratic. This corresponds to testing

$$H_0 : \beta_2 = 0 \text{ vs. } H_1 : \beta_2 \neq 0,$$

since $\beta_2 = 0$ corresponds to a simple linear equation and $\beta_2 \neq 0$ implies a quadratic relationship. We find that $t = (\hat{\beta}_2 - 0)/SE(\hat{\beta}_2) = -0.0423/0.0048 = -8.81$ so the null is rejected at any common level of significance and we conclude that the relationship is nonlinear. This is consistent with the impression gained from the plot.

We now draw the same scatter plot as for the linear model and add the regression line for the quadratic model. Because `abline()` can only draw straight lines, it cannot be used here. `lines()` is a function which allows to draw nonstraight lines, see `?lines`. The most basic call of `lines()` is `lines(x_values, y_values)` where `x_values` and `y_values` are vectors of the same length that provide coordinates of the points to be *sequentially* connected by a line. This makes it necessary to sort the coordinate pairs according to the X-values. Here we use the function `order()` to sort the fitted values of `score` according to the observations of `income`.

```
# draw a scatterplot of the observations for income and test score
plot(CASchools$income, CASchools$score,
     col = "steelblue",
     pch = 20,
     xlab = "District Income (thousands of dollars)",
     ylab = "Test Score",
     main = "Estimated Linear and Quadratic Regression Functions")

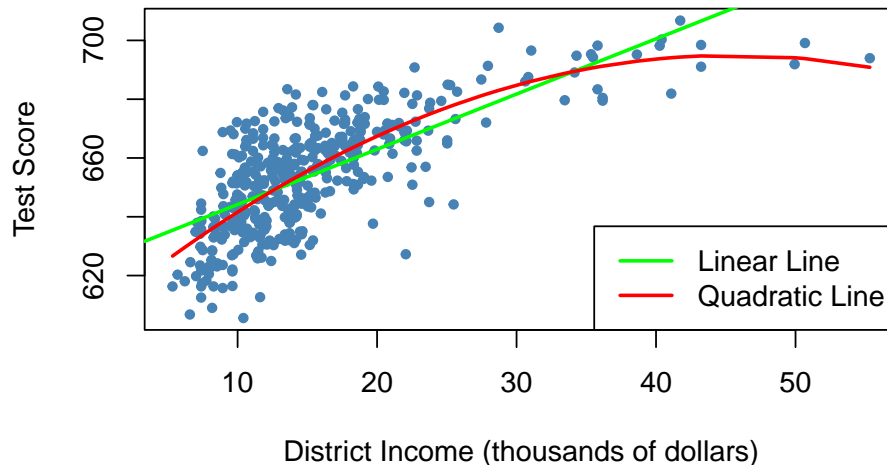
# add a linear function to the plot
abline(linear_model, col = "green", lwd = 2)

# add quadratic function to the plot
order_id <- order(CASchools$income)

lines(x = CASchools$income[order_id],
      y = fitted(quadratic_model)[order_id],
      col = "red",
      lwd = 2)
```

```
legend("bottomright", legend=c("Linear Line", "Quadratic Line"),
      lwd=2, col=c("green", "red"))
```

Estimated Linear and Quadratic Regression Functions



We see that the quadratic function does fit the data much better than the linear function.

8.2 Nonlinear Functions of a Single Independent Variable

Polynomials

The approach used to obtain a quadratic model can be generalized to polynomial models of arbitrary degree r ,

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \cdots + \beta_r X_i^r + u_i.$$

A cubic model for instance can be estimated in the same way as the quadratic model; we just have to use a polynomial of degree $r = 3$ in `income`. This is conveniently done using the function `poly()`.

```
# estimate a cubic model
cubic_model <- lm(score ~ poly(income, degree = 3, raw = TRUE), data = CASchools)
```

`poly()` generates orthogonal polynomials which are orthogonal to the constant by default. Here, we set `raw = TRUE` such that raw polynomials are evaluated, see `?poly`.

In practice the question will arise which polynomial order should be chosen. First, similarly as for $r = 2$, we can test the null hypothesis that the true relation is linear against the alternative hypothesis that the relationship is a polynomial of degree r :

$$H_0 : \beta_2 = 0, \beta_3 = 0, \dots, \beta_r = 0 \quad \text{vs.} \quad H_1 : \text{at least one } \beta_j \neq 0, j = 2, \dots, r.$$

This is a joint null hypothesis with $r - 1$ restrictions so it can be tested using the F -test presented in Chapter 7. `linearHypothesis()` can be used to conduct such tests. For example, we may test the null of a linear model against the alternative of a polynomial of a maximal degree $r = 3$ as follows.

```
# test the hypothesis of a linear model against quadratic or polynomial
# alternatives
```

```

# set up hypothesis matrix
R <- rbind(c(0, 0, 1, 0),
           c(0, 0, 0, 1))

# do the test
linearHypothesis(cubic_model,
                 hypothesis.matrix = R,
                 white.adj = "hc1")

#> Linear hypothesis test
#>
#> Hypothesis:
#> poly(income, degree = 3, raw = TRUE)2 = 0
#> poly(income, degree = 3, raw = TRUE)3 = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ poly(income, degree = 3, raw = TRUE)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F    Pr(>F)
#> 1      418
#> 2      416  2 37.691 9.043e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

We provide a hypothesis matrix as the argument `hypothesis.matrix`. This is useful when the coefficients have long names, as is the case here due to using `poly()`, or when the restrictions include multiple coefficients. How the hypothesis matrix \mathbf{R} is interpreted by `linearHypothesis()` is best seen using matrix algebra:

For the two linear constraints above, we have

$$\mathbf{R}\boldsymbol{\beta} = \mathbf{s}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

`linearHypothesis()` uses the zero vector for \mathbf{s} by default, see `?linearHypothesis`.

The p -value for the test is very small so that we reject the null hypothesis. However, this does not tell us *which* r to choose. In practice, one approach to determine the degree of the polynomial is to use *sequential testing*:

1. Estimate a polynomial model for some maximum value r .
2. Use a t -test to test $\beta_r = 0$. *Rejection* of the null means that X^r belongs in the regression equation.
3. *Acceptance* of the null in step 2 means that X^r can be eliminated from the model. Continue by repeating step 1 with order $r - 1$ and test whether $\beta_{r-1} = 0$. If the test rejects, use a polynomial model of order $r - 1$.
4. If the test from step 3 rejects, continue with the procedure until the coefficient on the highest power is statistically significant.

There is no unambiguous guideline on how to choose r in step one. However, as pointed out in Stock

and Watson (2015), economic data is often smooth such that it is appropriate to choose small orders like 2, 3, or 4.

We will demonstrate how to apply sequential testing by the example of the cubic model.

```
summary(cubic_model)
#>
#> Call:
#> lm(formula = score ~ poly(income, degree = 3, raw = TRUE), data = CASchools)
#>
#> Residuals:
#>   Min       1Q   Median       3Q      Max
#> -44.28  -9.21   0.20   8.32  31.16
#>
#> Coefficients:
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)          6.001e+02  5.830e+00 102.937 < 2e-16
#> poly(income, degree = 3, raw = TRUE)1  5.019e+00  8.595e-01   5.839 1.06e-08
#> poly(income, degree = 3, raw = TRUE)2 -9.581e-02  3.736e-02  -2.564  0.0107
#> poly(income, degree = 3, raw = TRUE)3  6.855e-04  4.720e-04   1.452  0.1471
#>
#> (Intercept)                ***
#> poly(income, degree = 3, raw = TRUE)1 ***
#> poly(income, degree = 3, raw = TRUE)2 *
#> poly(income, degree = 3, raw = TRUE)3
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 12.71 on 416 degrees of freedom
#> Multiple R-squared:  0.5584, Adjusted R-squared:  0.5552
#> F-statistic: 175.4 on 3 and 416 DF,  p-value: < 2.2e-16
```

The estimated cubic model stored in `cubic_model` is

$$\widehat{TestScore}_i = \underset{(5.83)}{600.1} + \underset{(0.86)}{5.02} \times income - \underset{(0.037)}{0.096} \times income^2 - \underset{(0.00047)}{0.00069} \times income^3.$$

The t -statistic on $income^3$ is 1.45 so the null that the relationship is quadratic cannot be rejected, even at the 10% level. This is contrary to the result presented in the book which reports robust standard errors throughout, so we will also use robust variance-covariance estimation to reproduce these results.

```
# test the hypothesis using robust standard errors
coeftest(cubic_model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>               Estimate Std. Error t value
#> (Intercept)          6.0008e+02  5.1021e+00 117.6150
#> poly(income, degree = 3, raw = TRUE)1  5.0187e+00  7.0735e-01   7.0950
#> poly(income, degree = 3, raw = TRUE)2 -9.5805e-02  2.8954e-02  -3.3089
#> poly(income, degree = 3, raw = TRUE)3  6.8549e-04  3.4706e-04   1.9751
#>
#>               Pr(>|t|)
#> (Intercept)                < 2.2e-16 ***
#> poly(income, degree = 3, raw = TRUE)1 5.606e-12 ***
```

```
#> poly(income, degree = 3, raw = TRUE)2 0.001018 **
#> poly(income, degree = 3, raw = TRUE)3 0.048918 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The reported standard errors have changed. Furthermore, the coefficient for income^3 is now significant at the 5% level. This means we reject the hypothesis that the regression function is quadratic against the alternative that it is cubic. Furthermore, we can also test if the coefficients for income^2 and income^3 are jointly significant using a robust version of the F -test.

```
# perform robust F-test
linearHypothesis(cubic_model,
                 hypothesis.matrix = R,
                 vcov. = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> poly(income, degree = 3, raw = TRUE)2 = 0
#> poly(income, degree = 3, raw = TRUE)3 = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ poly(income, degree = 3, raw = TRUE)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F    Pr(>F)
#> 1     418
#> 2     416  2 29.678 8.945e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With a p -value of $8.945e - 13$, i.e., much less than 0.05, the null hypothesis of linearity is rejected in favor of the alternative that the relationship is quadratic or cubic.

Interpretation of Coefficients in Nonlinear Regression Models

The coefficients in the polynomial regressions do not have a simple interpretation. Why? Think of a quadratic model: it is not helpful to think of the coefficient on X as the expected change in Y associated with a change in X holding the other regressors constant because X^2 changes as X varies. This is also the case for other deviations from linearity, for example in models where regressors and/or the dependent variable are log-transformed. A way to approach this is to calculate the estimated effect on Y associated with a change in X for one or more values of X . This idea is summarized in Key Concept 8.1.

Key Concept 8.1**The Expected Effect on Y for a Change in X_1 in a Nonlinear Regression Model**

Consider the nonlinear population regression model

$$Y_i = f(X_{1i}, X_{2i}, \dots, X_{ki}) + u_i, \quad i = 1, \dots, n,$$

where $f(X_{1i}, X_{2i}, \dots, X_{ki})$ is the population regression function and u_i is the error term. Denote by ΔY the expected change in Y associated with ΔX_1 , the change in X_1 while holding X_2, \dots, X_k constant. That is, the expected change in Y is the difference

$$\Delta Y = f(X_1 + \Delta X_1, X_2, \dots, X_k) - f(X_1, X_2, \dots, X_k).$$

The estimator of this unknown population difference is the difference between the predicted values for these two cases. Let $\hat{f}(X_1, X_2, \dots, X_k)$ be the predicted value of Y based on the estimator \hat{f} of the population regression function. Then the predicted change in Y is

$$\Delta \hat{Y} = \hat{f}(X_1 + \Delta X_1, X_2, \dots, X_k) - \hat{f}(X_1, X_2, \dots, X_k).$$

For example, we may ask the following: what is the predicted change in test scores associated with a one unit change (i.e., \$1000) in income, based on the estimated quadratic regression function

$$\widehat{TestScore} = 607.3 + 3.85 \times income - 0.0423 \times income^2 ?$$

Because the regression function is quadratic, this effect depends on the *initial* district income. We therefore consider two cases:

1. An increase in district income from 10 to 11 (from \$10000 per capita to \$11000).
2. An increase in district income from 40 to 41 (that is from \$40000 to \$41000).

In order to obtain the $\Delta \hat{Y}$ associated with a change in income from 10 to 11, we use the following formula:

$$\Delta \hat{Y} = \left(\hat{\beta}_0 + \hat{\beta}_1 \times 11 + \hat{\beta}_2 \times 11^2 \right) - \left(\hat{\beta}_0 + \hat{\beta}_1 \times 10 + \hat{\beta}_2 \times 10^2 \right).$$

To compute \hat{Y} using R we may use `predict()`.

```
# compute and assign the quadratic model
quadratic_model <- lm(score ~ income + I(income^2), data = CASchools)

# set up data for prediction
new_data <- data.frame(income = c(10, 11))

# do the prediction
Y_hat <- predict(quadratic_model, newdata = new_data)

# compute the difference
diff(Y_hat)
#>      2
#> 2.962517
```

Analogously we can compute the effect of a change in district income from 40 to 41:


```
# set up data for prediction
new_data <- data.frame(income = c(40, 41))

# do the prediction
Y_hat <- predict(quadratic_model, newdata = new_data)

# compute the difference
diff(Y_hat)
#>      2
#> 0.4240097
```

So for the quadratic model, the expected change in *TestScore* induced by an increase in *income* from 10 to 11 is about 2.96 points but an increase in *income* from 40 to 41 increases the predicted score by only 0.42. Hence, the slope of the estimated quadratic regression function is *steeper* at low levels of income than at higher levels.

Logarithms

Another way to specify a nonlinear regression function is to use the natural logarithm of Y and/or X . Logarithms convert changes in variables into percentage changes. This is convenient as many relationships are naturally expressed in terms of percentages.

There are three different cases in which logarithms might be used.

1. Transform X to its logarithm, but not Y .
2. Analogously we can transform Y to its logarithm but leave X at level.
3. Both Y and X are transformed to their logarithms.

The interpretation of the regression coefficients is different in each case.

Case I: X is in Logarithm, Y is not.

The regression model then is

$$Y_i = \beta_0 + \beta_1 \times \ln(X_i) + u_i, \quad i = 1, \dots, n.$$

Similar as for polynomial regressions, we do not have to create a new variable before using `lm()`. We can simply adjust the formula argument of `lm()` to tell R that the log-transformation of a variable should be used.

```
# estimate a level-log model
LinearLog_model <- lm(score ~ log(income), data = CASchools)

# compute robust summary
coefTest(LinearLog_model,
         vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 557.8323      3.8399 145.271 < 2.2e-16 ***
#> log(income)  36.4197      1.3969  26.071 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hence, the estimated regression function is

$$\widehat{TestScore} = 557.8 + 36.42 \times \ln(\text{income}).$$

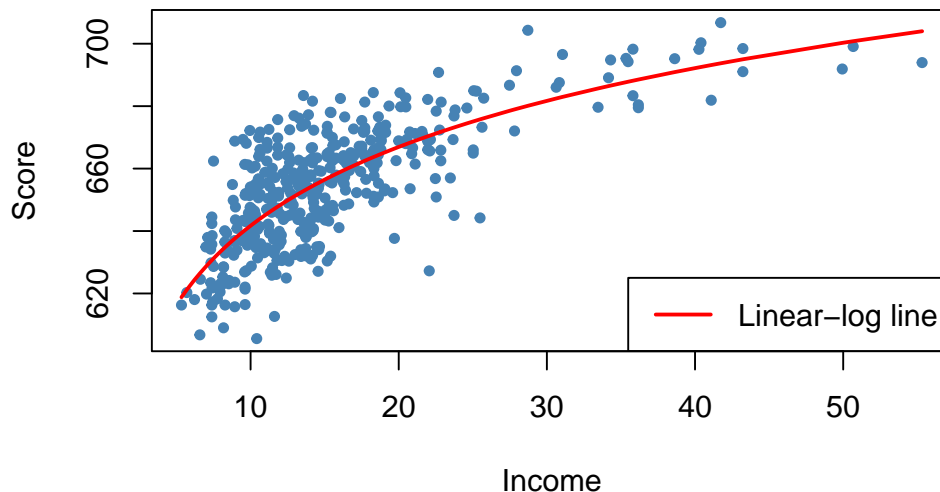
Let us draw a plot of this function.

```
# draw a scatterplot
plot(score ~ income,
     col = "steelblue",
     pch = 20,
     data = CASchools,
     ylab="Score",
     xlab="Income",
     main = "Linear-Log Regression Line")

# add the linear-log regression line
order_id <- order(CASchools$income)

lines(CASchools$income[order_id],
      fitted(LinearLog_model)[order_id],
      col = "red",
      lwd = 2)
legend("bottomright", legend = "Linear-log line", lwd = 2, col = "red")
```

Linear-Log Regression Line



We can interpret $\hat{\beta}_1$ as follows: a 1% increase in income is associated with an increase in test scores of $0.01 \times 36.42 = 0.36$ points. In order to get the estimated effect of a one unit change in income (that is, a change in the original units, thousands of dollars) on test scores, the method presented in Key Concept 8.1 can be used.

```
# set up new data
new_data <- data.frame(income = c(10, 11, 40, 41))

# predict the outcomes
Y_hat <- predict(LinearLog_model, newdata = new_data)
```

```
# compute the expected difference
Y_hat_matrix <- matrix(Y_hat, nrow = 2, byrow = TRUE)
Y_hat_matrix[, 2] - Y_hat_matrix[, 1]
#> [1] 3.471166 0.899297
```

By setting `nrow = 2` and `byrow = TRUE` in `matrix()`, we ensure that `Y_hat_matrix` is a 2×2 matrix filled row-wise with the entries of `Y_hat`.

The estimated model states that for an income increase from \$10000 to \$11000, test scores increase by an expected amount of 3.47 points. When income increases from \$40000 to \$41000, the expected increase in test scores is only about 0.90 points.

Case II: Y is in Logarithm, X is not

There are cases where it is useful to regress $\ln(Y)$.

The corresponding regression model then is

$$\ln(Y_i) = \beta_0 + \beta_1 \times X_i + u_i, \quad i = 1, \dots, n.$$

```
# estimate a log-linear model
LogLinear_model <- lm(log(score) ~ income, data = CASchools)

# obtain a robust coefficient summary
coefest(LogLinear_model,
        vcov = vcovHC, type = "HC1")

#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 6.43936234 0.00289382 2225.210 < 2.2e-16 ***
#> income      0.00284407 0.00017509   16.244 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression function is

$$\ln(\widehat{TestScore}) = 6.439 + 0.00284 \times income.$$

An increase in district income by \$1000 is expected to increase test scores by $100 \times 0.00284\% = 0.284\%$.

When the dependent variable is in logarithm, one cannot simply use $e^{\log(\cdot)}$ to transform predictions back to the original scale, see page of the book.

Case III: X and Y are in Logarithms

The log-log regression model is

$$\ln(Y_i) = \beta_0 + \beta_1 \times \ln(X_i) + u_i, \quad i = 1, \dots, n.$$

```
# estimate the log-log model
LogLog_model <- lm(log(score) ~ log(income), data = CASchools)
```

```

# print robust coefficient summary to the console
coeftest(LogLog_model,
          vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 6.3363494  0.0059246 1069.501 < 2.2e-16 ***
#> log(income) 0.0554190  0.0021446   25.841 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The estimated regression function hence is

$$\ln(\widehat{TestScore}) = 6.336 + 0.0554 \times \ln(\text{income}).$$

In a log-log model, a 1% change in X is associated with an estimated $\hat{\beta}_1\%$ change in Y .

We now reproduce Figure 8.5 of the book.

```

# generate a scatterplot
plot(log(score) ~ income,
     col = "steelblue",
     pch = 20,
     data = CASchools,
     ylab="log(Score)",
     xlab="Income",
     main = "Log-Linear Regression Function")

# add the log-linear regression line
order_id <- order(CASchools$income)

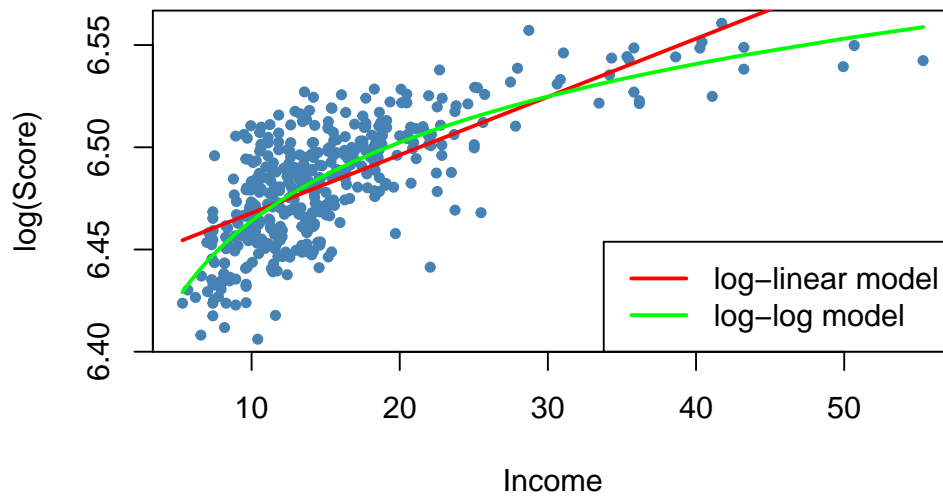
lines(CASchools$income[order_id],
      fitted(LogLinear_model)[order_id],
      col = "red",
      lwd = 2)

# add the log-log regression line
lines(sort(CASchools$income),
      fitted(LogLog_model)[order(CASchools$income)],
      col = "green",
      lwd = 2)

# add a legend
legend("bottomright",
      legend = c("log-linear model", "log-log model"),
      lwd = 2,
      col = c("red", "green"))

```

Log-Linear Regression Function



Key Concept 8.2 summarizes the three logarithmic regression models.

Key Concept 8.2 Logarithms in Regression: Three Cases

Case	Model Specification	Interpretation of β_1
(I)	$Y_i = \beta_0 + \beta_1 \ln(X_i) + u_i$	A 1% change in X is associated with a change in Y of $0.01 \times \beta_1$.
(II)	$\ln(Y_i) = \beta_0 + \beta_1 X_i + u_i$	A change in X by one unit ($\Delta X = 1$) is associated with a $100 \times \beta_1\%$ change in Y .
(III)	$\ln(Y_i) = \beta_0 + \beta_1 \ln(X_i) + u_i$	A 1% change in X is associated with a $\beta_1\%$ change in Y , so β_1 is the elasticity of Y with respect to X .

Of course we can also estimate a *polylog* model like

$$TestScore_i = \beta_0 + \beta_1 \times \ln(income_i) + \beta_2 \times \ln(income_i)^2 + \beta_3 \times \ln(income_i)^3 + u_i,$$

which models the dependent variable *TestScore* by a third-degree polynomial of the log-transformed regressor *income*.

```
# estimate the polylog model
polyLog_model <- lm(score ~ log(income) + I(log(income)^2) + I(log(income)^3),
  data = CASchools)

# print robust summary to the console
coefTest(polyLog_model,
  vcov = vcovHC, type = "HC1")

#>
#> t test of coefficients:
#>
```

```
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    486.1341    79.3825  6.1239 2.115e-09 ***
#> log(income)    113.3820    87.8837  1.2901  0.1977
#> I(log(income)^2) -26.9111    31.7457 -0.8477  0.3971
#> I(log(income)^3)  3.0632     3.7369  0.8197  0.4128
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparing by \bar{R}^2 we find that, leaving out the log-linear model, all models have a similar adjusted fit. In the class of polynomial models, the cubic specification has the highest \bar{R}^2 whereas the linear-log specification is the best of the log-models.

```
# compute the adj. R^2 for the nonlinear models
adj_R2 <- rbind("quadratic" = summary(quadratic_model)$adj.r.squared,
              "cubic" = summary(cubic_model)$adj.r.squared,
              "LinearLog" = summary(LinearLog_model)$adj.r.squared,
              "LogLinear" = summary(LogLinear_model)$adj.r.squared,
              "LogLog" = summary(LogLog_model)$adj.r.squared,
              "polyLog" = summary(polyLog_model)$adj.r.squared)

# assign column names
colnames(adj_R2) <- "adj_R2"

adj_R2
#>               adj_R2
#> quadratic 0.5540444
#> cubic     0.5552279
#> LinearLog 0.5614605
#> LogLinear 0.4970106
#> LogLog    0.5567251
#> polyLog   0.5599944
```

Let us now compare the cubic and the linear-log model by plotting the corresponding estimated regression functions.

```
# generate a scatterplot
plot(score ~ income,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     ylab="Score",
     xlab="Income",
     main = "Linear-Log and Cubic Regression Functions")

# add the linear-log regression line
order_id <- order(CASchools$income)

lines(CASchools$income[order_id],
      fitted(LinearLog_model)[order_id],
      col = "darkgreen",
      lwd = 2)

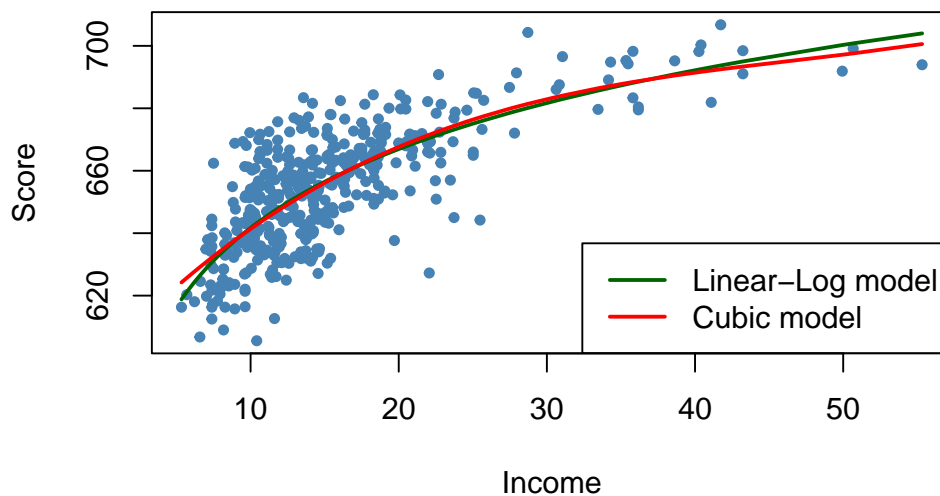
# add the cubic regression line
```

```

lines(x = CASchools$income[order_id],
      y = fitted(cubic_model)[order_id],
      col = "red",
      lwd = 2)
# add a legend
legend("bottomright",
      legend = c("Linear-Log model", "Cubic model"),
      lwd = 2,
      col = c("darkgreen", "red"))

```

Linear-Log and Cubic Regression Functions



Both regression lines look nearly identical. Altogether the linear-log model may be preferable since it is more parsimonious in terms of regressors: it does not include higher-degree polynomials.

8.3 Interactions between Independent Variables

There are research questions where it is interesting to learn how the effect on Y of a change in an independent variable depends on the value of another independent variable. For example, we may ask if districts with many English learners benefit differentially from a decrease in class sizes to those with few English learning students. To assess this using a multiple regression model, we include an interaction term. We consider three cases:

1. Interactions between two binary variables.
2. Interactions between a binary and a continuous variable.
3. Interactions between two continuous variables.

The following subsections discuss these cases briefly and demonstrate how to perform such regressions in R.

Interactions Between Two Binary Variables

Take two binary variables D_1 and D_2 and the population regression model

$$Y_i = \beta_0 + \beta_1 \times D_{1i} + \beta_2 \times D_{2i} + u_i.$$

Now assume that

$$Y_i = \ln(\text{Earnings}_i),$$

$$D_{1i} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ person has a college degree,} \\ 0 & \text{else.} \end{cases}$$

$$D_{2i} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ person is female,} \\ 0 & \text{if } i^{\text{th}} \text{ person is male.} \end{cases}$$

We know that β_1 measures the average difference in $\ln(\text{Earnings})$ between individuals with and without a college degree and β_2 is the gender differential in $\ln(\text{Earnings})$, ceteris paribus. This model *does not* allow us to determine if there is a gender specific effect of having a college degree and, if so, *how strong* this effect is. It is easy to come up with a model specification that allows to investigate this:

$$Y_i = \beta_0 + \beta_1 \times D_{1i} + \beta_2 \times D_{2i} + \beta_3 \times (D_{1i} \times D_{2i}) + u_i,$$

$(D_{1i} \times D_{2i})$ is called an interaction term and β_3 measures the difference in the effect of having a college degree for women versus men.

Key Concept 8.3

A Method for Interpreting Coefficients in Regression with Binary Variables

Compute expected values of Y for each possible set described by the set of binary variables. Compare the expected values. The coefficients can be expressed either as expected values or as the difference between at least two expected values.

Following Key Concept 8.3 we have

$$\begin{aligned} E(Y_i | D_{1i} = 0, D_{2i} = d_2) &= \beta_0 + \beta_1 \times 0 + \beta_2 \times d_2 + \beta_3 \times (0 \times d_2) \\ &= \beta_0 + \beta_2 \times d_2. \end{aligned}$$

If D_{1i} switches from 0 to 1 we obtain

$$\begin{aligned} E(Y_i | D_{1i} = 1, D_{2i} = d_2) &= \beta_0 + \beta_1 \times 1 + \beta_2 \times d_2 + \beta_3 \times (1 \times d_2) \\ &= \beta_0 + \beta_1 + \beta_2 \times d_2 + \beta_3 \times d_2. \end{aligned}$$

Hence, the overall effect is

$$E(Y_i | D_{1i} = 1, D_{2i} = d_2) - E(Y_i | D_{1i} = 0, D_{2i} = d_2) = \beta_1 + \beta_3 \times d_2,$$

so the effect is a difference of expected values.

Application to the Student-Teacher Ratio and the Percentage of English Learners

Now let

$$HiSTR = \begin{cases} 1, & \text{if } STR \geq 20 \\ 0, & \text{else,} \end{cases}$$

$$HiEL = \begin{cases} 1, & \text{if } PctEL \geq 10 \\ 0, & \text{else.} \end{cases}$$

We may use R to construct the variables above as follows.

```
# append HiSTR to CASchools
CASchools$HiSTR <- as.numeric(CASchools$size >= 20)

# append HiEL to CASchools
CASchools$HiEL <- as.numeric(CASchools$english >= 10)
```

We proceed by estimating the model

$$TestScore = \beta_0 + \beta_1 \times HiSTR + \beta_2 \times HiEL + \beta_3 \times (HiSTR \times HiEL) + u_i. \quad (8.1)$$

There are several ways to add the interaction term to the formula argument when using `lm()` but the most intuitive way is to use `HiEL * HiSTR`.¹

```
# estimate the model with a binary interaction term
bi_model <- lm(score ~ HiSTR * HiEL, data = CASchools)

# print a robust summary of the coefficients
coefest(bi_model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 664.1433      1.3881 478.4589 < 2.2e-16 ***
#> HiSTR       -1.9078      1.9322  -0.9874  0.3240
#> HiEL       -18.3155      2.3340  -7.8472 3.634e-14 ***
#> HiSTR:HiEL  -3.2601      3.1189  -1.0453  0.2965
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression model is

$$\widehat{TestScore} = 664.1 - \underset{(1.39)}{1.9} \times HiSTR - \underset{(2.33)}{18.3} \times HiEL - \underset{(3.12)}{3.3} \times (HiSTR \times HiEL),$$

and it predicts that the effect of moving from a school district with a low student-teacher ratio to a district with a high student-teacher ratio, depending on high or low percentage of english learners is

¹Appending `HiEL * HiSTR` to the formula will add `HiEL`, `HiSTR` and their interaction as regressors while `HiEL:HiSTR` only adds the interaction term.

$-1.9 - 3.3 \times HiEL$. So for districts with a low share of english learners ($HiEL = 0$), the estimated effect is a decrease of 1.9 points in test scores while for districts with a large fraction of English learners ($HiEL = 1$), the predicted decrease in test scores amounts to $1.9 + 3.3 = 5.2$ points.

We can also use the model to estimate the mean test score for each possible combination of the included binary variables.

```
# estimate means for all combinations of HiSTR and HiEL

# 1.
predict(bi_model, newdata = data.frame("HiSTR" = 0, "HiEL" = 0))
#>      1
#> 664.1433

# 2.
predict(bi_model, newdata = data.frame("HiSTR" = 0, "HiEL" = 1))
#>      1
#> 645.8278

# 3.
predict(bi_model, newdata = data.frame("HiSTR" = 1, "HiEL" = 0))
#>      1
#> 662.2354

# 4.
predict(bi_model, newdata = data.frame("HiSTR" = 1, "HiEL" = 1))
#>      1
#> 640.6598
```

We now verify that these predictions are differences in the coefficient estimates presented in equation (8.1):

$$\begin{aligned} \widehat{TestScore} &= \hat{\beta}_0 = 664.1 && \Leftrightarrow HiSTR = 0, HiEL = 0. \\ \widehat{TestScore} &= \hat{\beta}_0 + \hat{\beta}_2 = 664.1 - 18.3 = 645.8 && \Leftrightarrow HiSTR = 0, HiEL = 1. \\ \widehat{TestScore} &= \hat{\beta}_0 + \hat{\beta}_1 = 664.1 - 1.9 = 662.2 && \Leftrightarrow HiSTR = 1, HiEL = 0. \\ \widehat{TestScore} &= \hat{\beta}_0 + \hat{\beta}_1 + \hat{\beta}_2 + \hat{\beta}_3 = 664.1 - 1.9 - 18.3 - 3.3 = 640.6 && \Leftrightarrow HiSTR = 1, HiEL = 1. \end{aligned}$$

Interactions Between a Continuous and a Binary Variable

Now let X_i denote the years of working experience of person i , which is a continuous variable. We have

$$Y_i = \ln(Earnings_i),$$

$$X_i = \text{working experience of person } i,$$

$$D_i = \begin{cases} 1, & \text{if } i^{th} \text{ person has a college degree} \\ 0, & \text{else.} \end{cases}$$

The baseline model thus is

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + u_i,$$

a multiple regression model that allows us to estimate the average benefit of having a college degree holding working experience constant as well as the average effect on earnings of a change in working experience holding college degree constant.

By adding the interaction term $X_i \times D_i$ we allow the effect of an additional year of work experience to differ between individuals with and without college degree,

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3 (X_i \times D_i) + u_i.$$

Here, β_3 is the expected difference in the effect of an additional year of work experience for college graduates versus non-graduates. Another possible specification is

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i \times D_i) + u_i.$$

This model states that the expected impact of an additional year of work experience on earnings differs for college graduates and non-graduates but that graduating on its own does not increase earnings.

All three regression functions can be visualized by straight lines. Key Concept 8.4 summarizes the differences.

Key Concept 8.4 Interactions Between Binary and Continuous Variables

An interaction term like $X_i \times D_i$ (where X_i is continuous and D_i is binary) allows for the slope to depend on the binary variable D_i . There are three possibilities:

1. Different intercept and same slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + u_i.$$

2. Different intercept and different slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3 \times (X_i \times D_i) + u_i.$$

3. Same intercept and different slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i \times D_i) + u_i.$$

The following code chunk demonstrates how to replicate the results shown in Figure 8.8 of the book using artificial data.

```
# generate artificial data
set.seed(1)

X <- runif(200,0, 15)
D <- sample(0:1, 200, replace = T)
Y <- 450 + 150 * X + 500 * D + 50 * (X * D) + rnorm(200, sd = 300)

# divide plotting area accordingly
```

```
m <- rbind(c(1, 2), c(3, 0))
graphics::layout(m)

# estimate the models and plot the regression lines
# 1. (baseline model)
plot(X, log(Y),
     pch = 20,
     col = "steelblue",
     main = "Different Intercepts, Same Slope",
     cex.main=1.2)

mod1_coef <- lm(log(Y) ~ X + D)$coefficients

abline(coef = c(mod1_coef[1], mod1_coef[2]),
       col = "red",
       lwd = 1.5)

abline(coef = c(mod1_coef[1] + mod1_coef[3], mod1_coef[2]),
       col = "green",
       lwd = 1.5)

# 2. (baseline model + interaction term)
plot(X, log(Y),
     pch = 20,
     col = "steelblue",
     main = "Different Intercepts, Different Slopes",
     cex.main=1.2)

mod2_coef <- lm(log(Y) ~ X + D + X:D)$coefficients

abline(coef = c(mod2_coef[1], mod2_coef[2]),
       col = "red",
       lwd = 1.5)

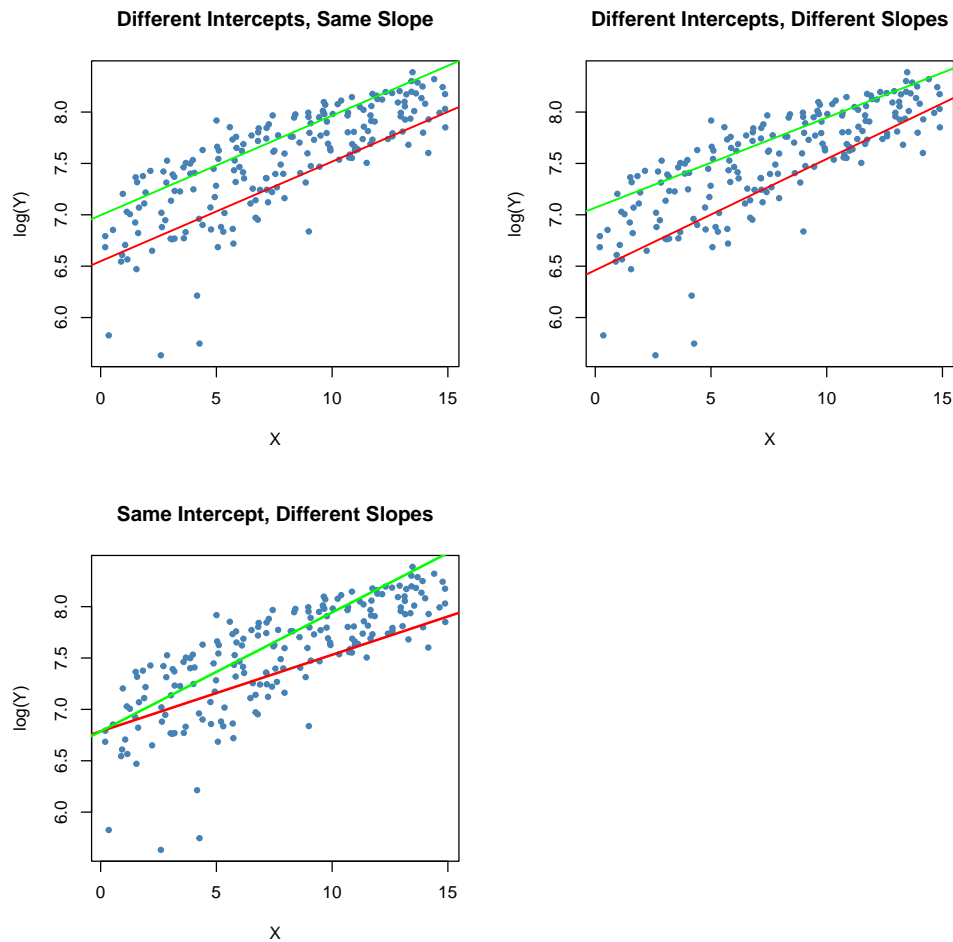
abline(coef = c(mod2_coef[1] + mod2_coef[3], mod2_coef[2] + mod2_coef[4]),
       col = "green",
       lwd = 1.5)

# 3. (omission of D as regressor + interaction term)
plot(X, log(Y),
     pch = 20,
     col = "steelblue",
     main = "Same Intercept, Different Slopes",
     cex.main=1.2)

mod3_coef <- lm(log(Y) ~ X + X:D)$coefficients

abline(coef = c(mod3_coef[1], mod3_coef[2]),
       col = "red",
       lwd = 2)
```

```
abline(coef = c(mod3_coef[1], mod3_coef[2] + mod3_coef[3]),
       col = "green",
       lwd = 2)
```



Application to the Student-Teacher Ratio and the Percentage of English Learners

Using a model specification like the second one discussed in Key Concept 8.3 (different slope, different intercept) we may answer the question whether the effect on test scores of decreasing the student-teacher ratio depends on whether there are many or few English learners. We estimate the regression model

$$\widehat{TestScore}_i = \beta_0 + \beta_1 \times size_i + \beta_2 \times HiEL_i + \beta_3(size_i \times HiEL_i) + u_i.$$

```
# estimate the model
bci_model <- lm(score ~ size + HiEL + size * HiEL, data = CASchools)

# print robust summary of coefficients to the console
coeftest(bci_model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept) 682.24584 11.86781 57.4871 <2e-16 ***
#> size -0.96846 0.58910 -1.6440 0.1009
#> HiEL 5.63914 19.51456 0.2890 0.7727
#> size:HiEL -1.27661 0.96692 -1.3203 0.1875
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression model is

$$\widehat{TestScore} = 682.2 - 0.97 \times size + 5.6 \times HiEL - 1.28 \times (size \times HiEL).$$

(11.87) (0.59) (19.51) (0.97)

The estimated regression line for districts with a low fraction of English learners ($HiEL_i = 0$) is

$$\widehat{TestScore} = 682.2 - 0.97 \times size_i.$$

For districts with a high fraction of English learners we have

$$\begin{aligned} \widehat{TestScore} &= 682.2 + 5.6 - 0.97 \times size_i - 1.28 \times size_i \\ &= 687.8 - 2.25 \times size_i. \end{aligned}$$

The predicted increase in test scores following a reduction of the student-teacher ratio by 1 unit is about 0.97 points in districts where the fraction of English learners is low but 2.25 in districts with a high share of English learners. From the coefficient on the interaction term $size \times HiEL$ we see that the difference between both effects is 1.28 points.

The next code chunk draws both lines belonging to the model. In order to make observations with $HiEL = 0$ distinguishable from those with $HiEL = 1$, we use different colors.

```
# identify observations with PctEL >= 10
id <- CASchools$english >= 10

# plot observations with HiEL = 0 as red dots
plot(CASchools$size[!id], CASchools$score[!id],
     xlim = c(0, 27),
     ylim = c(600, 720),
     pch = 20,
     col = "red",
     main = "",
     xlab = "Class Size",
     ylab = "Test Score")

# plot observations with HiEL = 1 as green dots
points(CASchools$size[id], CASchools$score[id],
       pch = 20,
       col = "green")

# read out estimated coefficients of bci_model
coefs <- bci_model$coefficients

# draw the estimated regression line for HiEL = 0
```

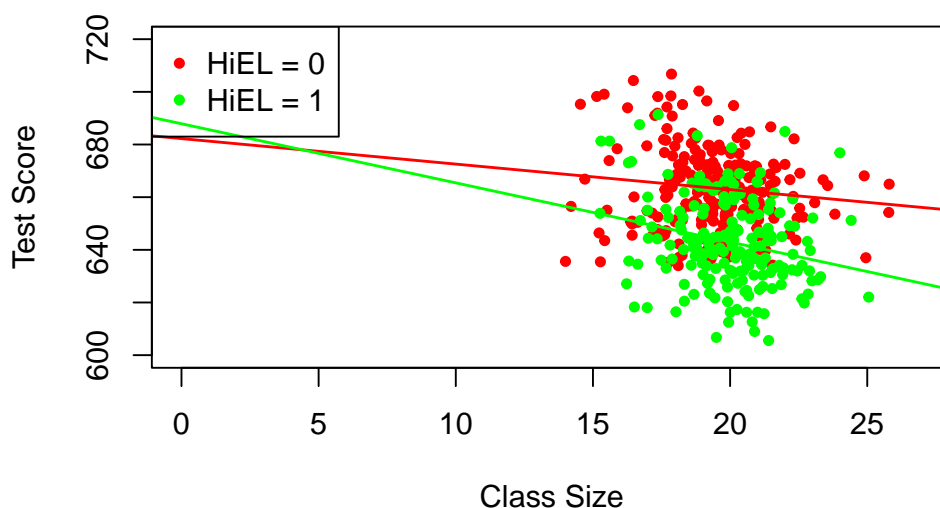
```

abline(coef = c(coefs[1], coefs[2]),
       col = "red",
       lwd = 1.5)

# draw the estimated regression line for HiEL = 1
abline(coef = c(coefs[1] + coefs[3], coefs[2] + coefs[4]),
       col = "green",
       lwd = 1.5 )

# add a legend to the plot
legend("topleft",
      pch = c(20, 20),
      col = c("red", "green"),
      legend = c("HiEL = 0", "HiEL = 1"))

```



Interactions Between Two Continuous Variables

Consider a regression model with Y the log earnings and two continuous regressors X_1 , the years of work experience, and X_2 , the years of schooling. We want to estimate the effect on wages of an additional year of work experience depending on a given level of schooling. This effect can be assessed by including the interaction term ($X_{1i} \times X_{2i}$) in the model:

$$\Delta Y_i = \beta_0 + \beta_1 \times X_{1i} + \beta_2 \times X_{2i} + \beta_3 \times (X_{1i} \times X_{2i}) + u_i.$$

Following Key Concept 8.1, we find that the effect on Y for a change on X_1 given X_2 is

$$\frac{\Delta Y}{\Delta X_1} = \beta_1 + \beta_3 X_2.$$

In the earnings example, a positive β_3 implies that the effect on log earnings of an additional year of work experience grows linearly with years of schooling. In reverse, we have

$$\frac{\Delta Y}{\Delta X_2} = \beta_2 + \beta_3 X_1$$

as the effect on log earnings of an additional year of schooling holding work experience constant.

Altogether we find that β_3 measures the effect of a unit increase in X_1 and X_2 *beyond* the effects of increasing X_1 alone and X_2 alone by one unit. The overall change in Y is thus

$$Y_i = (\beta_1 + \beta_3 X_2)\Delta X_1 + (\beta_2 + \beta_3 X_1)\Delta X_2 + \beta_3 \Delta X_1 \Delta X_2. \quad (8.2)$$

Key Concept 8.5 summarizes interactions between two regressors in multiple regression.

Key Concept 8.5 Interactions in Multiple Regression

The interaction term between the two regressors X_1 and X_2 is given by their product $X_1 \times X_2$. Adding this interaction term as a regressor to the model

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + u_i$$

allows the effect on Y of a change in X_2 to depend on the value of X_1 and vice versa. Thus the coefficient β_3 in the model

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \times X_2) + u_i$$

measures the effect of a one-unit increase in both X_1 and X_2 above and beyond the sum of both individual effects. This holds for continuous *and* binary regressors.

8.3.0.1 Application to the Student-Teacher Ratio and the Percentage of English Learners

We now examine the interaction between the continuous variables student-teacher ratio and the percentage of English learners.

```
# estimate regression model including the interaction between 'PctEL' and 'size'
cci_model <- lm(score ~ size + english + english * size, data = CASchools)
```

```
# print a summary to the console
coeftest(cci_model, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 686.3385268  11.7593466  58.3654 < 2e-16 ***
#> size        -1.1170184   0.5875136  -1.9013  0.05796 .
#> english     -0.6729119   0.3741231  -1.7986  0.07280 .
#> size:english  0.0011618    0.0185357   0.0627  0.95005
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model equation is

$$\widehat{TestScore} = \underset{(11.76)}{686.3} - \underset{(0.59)}{1.12} \times STR - \underset{(0.37)}{0.67} \times PctEL + \underset{(0.02)}{0.0012} \times (STR \times PctEL).$$

For the interpretation, let us consider the quartiles of $PctEL$.


```
summary(CASchools$english)
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.000  1.941   8.778  15.768  22.970  85.540
```

According to (8.2), if $PctEL$ is at its median value of 8.78, the slope of the regression function relating test scores and the student teacher ratio is predicted to be $-1.12 + 0.0012 \times 8.78 = -1.11$. This means that increasing the student-teacher ratio by one unit is expected to deteriorate test scores by 1.11 points. For the 75% quantile, the estimated change on $TestScore$ of a one-unit increase in STR is estimated by $-1.12 + 0.0012 \times 23.0 = -1.09$ so the slope is somewhat lower. The interpretation is that for a school district with a share of 23% English learners, a reduction of the student-teacher ratio by one unit is expected to increase test scores by only 1.09 points.

However, the output of `coefstest()` indicates that the difference of the effect for the median and the 75% quantile is not statistically significant. $H_0 : \beta_3 = 0$ cannot be rejected at the 5% level of significance (the p -value is 0.95).

Example: The Demand for Economic Journals

In this section we replicate the empirical example *The Demand for Economic Journals* presented on pages 336 - 337 of the book. The central question is: how elastic is the demand by libraries for economic journals? The idea here is to analyze the relationship between the number of subscriptions to a journal at U.S. libraries and the journal's subscription price. The study uses the data set `Journals` which is provided with the `AER` package and contains observations for 180 economic journals for the year 2000. You can use the help function (`?Journals`) to get more information on the data after loading the package.

```
# load package and the data set
library(AER)
data("Journals")
```

We measure the price as “price per citation” and compute journal age and the number of characters manually. For consistency with the book we also rename the variables.

```
# define and rename variables
Journals$PricePerCitation <- Journals$price/Journals$citations
Journals$Age <- 2000 - Journals$foundingyear
Journals$Characters <- Journals$charpp * Journals$pages/10^6
Journals$Subscriptions <- Journals$subs
```

The range of “price per citation” is quite large:

```
# compute summary statistics for price per citation
summary(Journals$PricePerCitation)
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.005223  0.464495  1.320513  2.548455  3.440171  24.459459
```

The lowest price observed is a mere 0.5¢ per citation while the highest price is more than 20¢ per citation.

We now estimate four different model specifications. All models are log-log models. This is useful because it allows us to directly interpret the coefficients as elasticities, see Key Concept 8.2. (I) is a linear model. To alleviate a possible omitted variable bias, (II) augments (I) by the covariates $\ln(Age)$ and $\ln(Characters)$. The largest model (III) attempts to capture nonlinearities in the relationship of $\ln(Subscriptions)$ and $\ln(PricePerCitation)$ using a cubic regression function of $\ln(PricePerCitation)$ and also adds the interaction term ($PricePerCitation \times Age$) while specification (IV) does not include

the cubic term.

$$(I) \quad \ln(\text{Subscriptions}_i) = \beta_0 + \beta_1 \ln(\text{PricePerCitation}_i) + u_i.$$

$$(II) \quad \ln(\text{Subscriptions}_i) = \beta_0 + \beta_1 \ln(\text{PricePerCitation}_i) + \beta_4 \ln(\text{Age}_i) + \beta_6 \ln(\text{Characters}_i) + u_i.$$

$$(III) \quad \ln(\text{Subscriptions}_i) = \beta_0 + \beta_1 \ln(\text{PricePerCitation}_i) + \beta_2 \ln(\text{PricePerCitation}_i)^2 \\ + \beta_3 \ln(\text{PricePerCitation}_i)^3 + \beta_4 \ln(\text{Age}_i) \\ + \beta_5 [\ln(\text{Age}_i) \times \ln(\text{PricePerCitation}_i)] + \beta_6 \ln(\text{Characters}_i) + u_i.$$

$$(IV) \quad \ln(\text{Subscriptions}_i) = \beta_0 + \beta_1 \ln(\text{PricePerCitation}_i) + \beta_4 \ln(\text{Age}_i) \\ + \beta_5 [\ln(\text{Age}_i) \times \ln(\text{PricePerCitation}_i)] + \beta_6 \ln(\text{Characters}_i) + u_i.$$

```
# Estimate models (I) - (IV)
J_mod1 <- lm(log(Subscriptions) ~ log(PricePerCitation),
             data = Journals)

J_mod2 <- lm(log(Subscriptions) ~ log(PricePerCitation)
             + log(Age) + log(Characters),
             data = Journals)

J_mod3 <- lm(log(Subscriptions) ~
             log(PricePerCitation) + I(log(PricePerCitation)^2)
             + I(log(PricePerCitation)^3) + log(Age)
             + log(Age):log(PricePerCitation) + log(Characters),
             data = Journals)

J_mod4 <- lm(log(Subscriptions) ~
             log(PricePerCitation) + log(Age)
             + log(Age):log(PricePerCitation) +
             log(Characters),
             data = Journals)
```

Using `summary()`, we obtain the following estimated models:

$$(I) \quad \ln(\widehat{Subscriptions}_i) = 4.77 - 0.53 \ln(\text{PricePerCitation}_i)$$

$$(II) \quad \ln(\widehat{Subscriptions}_i) = 3.21 - 0.41 \ln(\text{PricePerCitation}_i) + 0.42 \ln(\text{Age}_i) + 0.21 \ln(\text{Characters}_i)$$

$$(III) \quad \ln(\widehat{Subscriptions}_i) = 3.41 - 0.96 \ln(\text{PricePerCitation}_i) + 0.02 \ln(\text{PricePerCitation}_i)^2 \\ + 0.004 \ln(\text{PricePerCitation}_i)^3 + 0.37 \ln(\text{Age}_i) \\ + 0.16 [\ln(\text{Age}_i) \times \ln(\text{PricePerCitation}_i)] \\ + 0.23 \ln(\text{Characters}_i)$$

$$(IV) \quad \ln(\widehat{Subscriptions}_i) = 3.43 - 0.90 \ln(\text{PricePerCitation}_i) + 0.37 \ln(\text{Age}_i) \\ + 0.14 [\ln(\text{Age}_i) \times \ln(\text{PricePerCitation}_i)] + 0.23 \ln(\text{Characters}_i)$$

We use an F -Test to test if the transformations of $\ln(\text{PricePerCitation})$ in Model (III) are statistically significant.

```
# F-Test for significance of cubic terms
linearHypothesis(J_mod3,
                 c("I(log(PricePerCitation)^2)=0", "I(log(PricePerCitation)^3)=0"),
                 vcov. = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> I(log(PricePerCitation)^2) = 0
#> I(log(PricePerCitation)^3) = 0
#>
#> Model 1: restricted model
#> Model 2: log(Subscriptions) ~ log(PricePerCitation) + I(log(PricePerCitation)^2) +
#>   I(log(PricePerCitation)^3) + log(Age) + log(Age):log(PricePerCitation) +
#>   log(Characters)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df    F Pr(>F)
#> 1     175
#> 2     173  2 0.1943 0.8236
```

Clearly, we cannot reject the null hypothesis $H_0 : \beta_3 = \beta_4 = 0$ in model (III).

We now demonstrate how the function `stargazer()` can be used to generate a tabular representation of all four models.

```
# load the stargazer package
library(stargazer)

# gather robust standard errors in a list
rob_se <- list(sqrt(diag(vcovHC(J_mod1, type = "HC1"))),
               sqrt(diag(vcovHC(J_mod2, type = "HC1"))),
               sqrt(diag(vcovHC(J_mod3, type = "HC1"))),
```

```

sqrt(diag(vcovHC(J_mod4, type = "HC1"))))

# generate a Latex table using stargazer
stargazer(J_mod1, J_mod2, J_mod3, J_mod4,
          se = rob_se,
          digits = 3,
          column.labels = c("(I)", "(II)", "(III)", "(IV)"))

```

The subsequent code chunk reproduces Figure 8.9 of the book.

```

# divide plotting area
m <- rbind(c(1, 2), c(3, 0))
graphics::layout(m)

# scatterplot
plot(Journals$PricePerCitation,
     Journals$Subscriptions,
     pch = 20,
     col = "steelblue",
     ylab = "Subscriptions",
     xlab = "Price per citation",
     main = "(a)")

# log-log scatterplot and estimated regression line (I)
plot(log(Journals$PricePerCitation),
     log(Journals$Subscriptions),
     pch = 20,
     col = "steelblue",
     ylab = "ln(Subscriptions)",
     xlab = "ln(Price per citation)",
     main = "(b)")

abline(J_mod1,
       lwd = 1.5)

# log-log scatterplot and regression lines (IV) for Age = 5 and Age = 80
plot(log(Journals$PricePerCitation),
     log(Journals$Subscriptions),
     pch = 20,
     col = "steelblue",
     ylab = "ln(Subscriptions)",
     xlab = "ln(Price per citation)",
     main = "(c)")

JM4C <- J_mod4$coefficients

# Age = 80
abline(coef = c(JM4C[1] + JM4C[3] * log(80),
               JM4C[2] + JM4C[5] * log(80)),
       col = "darkred",
       lwd = 1.5)

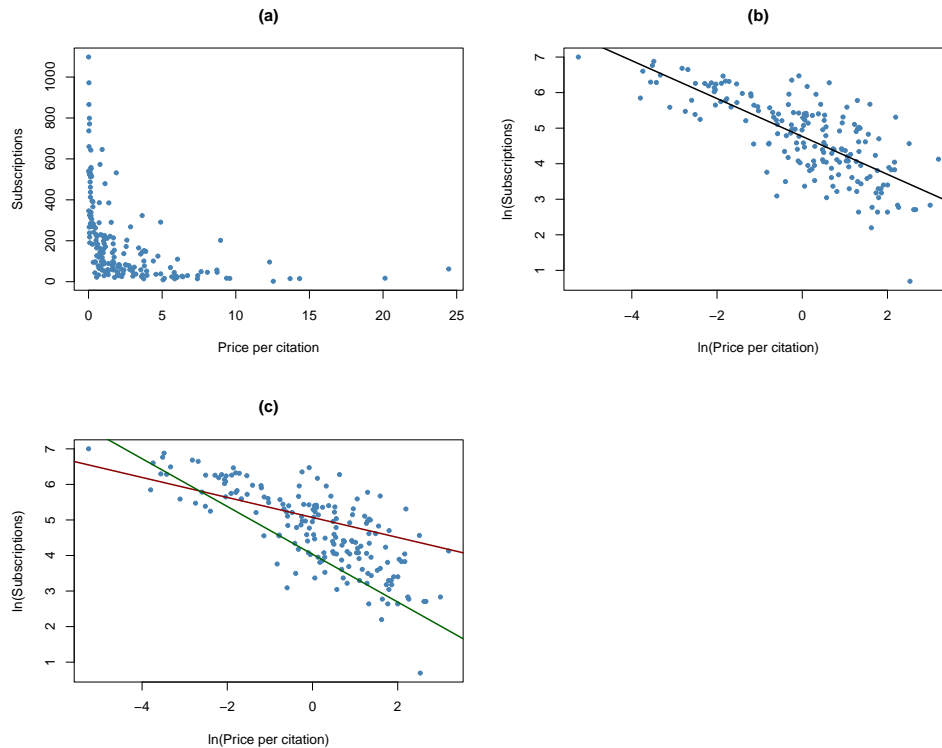
```

Table 8.1: Nonlinear Regression Models of Journal Subscriptions

	Dependent Variable: Logarithm of Subscriptions			
	(I)	(II)	(III)	(IV)
log(PricePerCitation)	-0.533*** (0.034)	-0.408*** (0.044)	-0.961*** (0.160)	-0.899*** (0.145)
I(log(PricePerCitation)^2)			0.017 (0.025)	
I(log(PricePerCitation)^3)			0.004 (0.006)	
log(Age)		0.424*** (0.119)	0.373*** (0.118)	0.374*** (0.118)
log(Characters)		0.206** (0.098)	0.235** (0.098)	0.229** (0.096)
log(PricePerCitation):log(Age)			0.156*** (0.052)	0.141*** (0.040)
Constant	4.766*** (0.055)	3.207*** (0.380)	3.408*** (0.374)	3.434*** (0.367)
Observations	180	180	180	180
R ²	0.557	0.613	0.635	0.634
Adjusted R ²	0.555	0.607	0.622	0.626
Residual Std. Error	0.750 (df = 178)	0.705 (df = 176)	0.691 (df = 173)	0.688 (df = 175)
F Statistic	224.037*** (df = 1; 178)	93.009*** (df = 3; 176)	50.149*** (df = 6; 173)	75.749*** (df = 4; 175)

* p<0.1; ** p<0.05; *** p<0.01

```
# Age = 5
abline(coef = c(JM4C[1] + JM4C[3] * log(5),
               JM4C[2] + JM4C[5] * log(5)),
       col = "darkgreen",
       lwd = 1.5)
```



As can be seen from plots (a) and (b), the relation between subscriptions and the citation price is adverse and nonlinear. Log-transforming both variables makes it approximately linear. Plot (c) shows that the price elasticity of journal subscriptions depends on the journal's age: the red line shows the estimated relationship for $Age = 80$ while the green line represents the prediction from model (IV) for $Age = 5$.

Which conclusion can be drawn?

1. We conclude that the demand for journals is more elastic for young journals than for old journals.
2. For model (III) we cannot reject the null hypothesis that the coefficients on $\ln(\text{PricePerCitation})^2$ and $\ln(\text{PricePerCitation})^3$ are both zero using an F -test. This is evidence compatible with a linear relation between log-subscriptions and log-price.
3. Demand is greater for Journals with more characters, holding price and age constant.

Altogether our estimates suggest that the demand is very inelastic, i.e., the libraries' demand for economic journals is quite insensitive to the price: using model (IV), even for a young journal ($Age = 5$) we estimate the price elasticity to be $-0.899 + 0.374 \times \ln(5) + 0.141 \times [\ln(1) \times \ln(5)] \approx -0.3$ so a one percent increase in price is predicted to reduce the demand by only 0.3 percent.

This finding comes at no surprise since providing the most recent publications is a necessity for libraries.

8.4 Nonlinear Effects on Test Scores of the Student-Teacher Ratio

In this section we will discuss three specific questions about the relationship between test scores and the student-teacher ratio:

1. Does the effect on test scores of decreasing the student-teacher ratio depend on the fraction of English learners when we control for economic idiosyncrasies of the different districts?
2. Does this effect depend on the student-teacher ratio?
3. *How strong* is the effect of decreasing the student-teacher ratio (by two students per teacher) if we take into account economic characteristics and nonlinearities?

To answer these questions we consider a total of seven models, some of which are nonlinear regression specifications of the types that have been discussed before. As measures for the students' economic backgrounds, we additionally consider the regressors *lunch* and $\ln(\textit{income})$. We use the logarithm of *income* because the analysis in Chapter 8.2 showed that the nonlinear relationship between *income* and *TestScores* is approximately logarithmic. We do not include expenditure per pupil (*expenditure*) because doing so would imply that expenditure varies with the student-teacher ratio (see Chapter 7.2 of the book for a detailed argument).

Nonlinear Regression Models of Test Scores

The considered model specifications are:

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_4 \textit{english}_i + \beta_9 \textit{lunch}_i + u_i. \quad (8.3)$$

(8.4)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_4 \textit{english}_i + \beta_9 \textit{lunch}_i + \beta_{10} \ln(\textit{income}_i) + u_i. \quad (8.5)$$

(8.6)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_5 \textit{HiEL}_i + \beta_6 (\textit{HiEL}_i \times \textit{size}_i) + u_i. \quad (8.7)$$

(8.8)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_5 \textit{HiEL}_i + \beta_6 (\textit{HiEL}_i \times \textit{size}_i) + \beta_9 \textit{lunch}_i + \beta_{10} \ln(\textit{income}_i) + u_i. \quad (8.9)$$

(8.10)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_2 \textit{size}_i^2 + \beta_5 \textit{HiEL}_i + \beta_9 \textit{lunch}_i + \beta_{10} \ln(\textit{income}_i) + u_i. \quad (8.11)$$

(8.12)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_2 \textit{size}_i^2 + \beta_3 \textit{size}_i^3 + \beta_5 \textit{HiEL}_i + \beta_6 (\textit{HiEL} \times \textit{size}) \quad (8.13)$$

$$+ \beta_7 (\textit{HiEL}_i \times \textit{size}_i^2) + \beta_8 (\textit{HiEL}_i \times \textit{size}_i^3) + \beta_9 \textit{lunch}_i + \beta_{10} \ln(\textit{income}_i) + u_i. \quad (8.14)$$

(8.15)

$$\textit{TestScore}_i = \beta_0 + \beta_1 \textit{size}_i + \beta_2 \textit{size}_i^2 + \beta_3 \textit{size}_i^3 + \beta_4 \textit{english} + \beta_9 \textit{lunch}_i + \beta_{10} \ln(\textit{income}_i) + u_i. \quad (8.16)$$

```
# estimate all models
```

```
TS_mod1 <- lm(score ~ size + english + lunch, data = CASchools)
```

```
TS_mod2 <- lm(score ~ size + english + lunch + log(income), data = CASchools)
```

```
TS_mod3 <- lm(score ~ size + HiEL + HiEL:size, data = CASchools)
```

```

TS_mod4 <- lm(score ~ size + HiEL + HiEL:size + lunch + log(income), data = CASchools)

TS_mod5 <- lm(score ~ size + I(size^2) + I(size^3) + HiEL + lunch + log(income),
  data = CASchools)

TS_mod6 <- lm(score ~ size + I(size^2) + I(size^3) + HiEL + HiEL:size + HiEL:I(size^2) +
  HiEL:I(size^3) + lunch + log(income), data = CASchools)

TS_mod7 <- lm(score ~ size + I(size^2) + I(size^3) + english + lunch + log(income),
  data = CASchools)

```

We may use `summary()` to assess the models' fit. Using `stargazer()` we may also obtain a tabular representation of all regression outputs and which is more convenient for comparison of the models.

```

# gather robust standard errors in a list
rob_se <- list(sqrt(diag(vcovHC(TS_mod1, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod2, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod3, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod4, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod5, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod6, type = "HC1"))),
  sqrt(diag(vcovHC(TS_mod7, type = "HC1"))))

```

```

# generate a LaTeX table of regression outputs
stargazer(TS_mod1,
  TS_mod2,
  TS_mod3,
  TS_mod4,
  TS_mod5,
  TS_mod6,
  TS_mod7,
  digits = 3,
  dep.var.caption = "Dependent Variable: Test Score",
  se = rob_se,
  column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)", "(7)"))

```

Let us summarize what can be concluded from the results presented in Table 8.2.

First of all, the coefficient on *size* is statistically significant in all seven models. Adding $\ln(\text{income})$ to model (1) we find that the corresponding coefficient is statistically significant at 1% while all other coefficients remain at their significance level. Furthermore, the estimate for the coefficient on *size* is roughly 0.27 points larger, which may be a sign of attenuated omitted variable bias. We consider this a reason to include $\ln(\text{income})$ as a regressor in other models, too.

Regressions (3) and (4) aim to assess the effect of allowing for an interaction between *size* and *HiEL*, without and with economic control variables. In both models, both the coefficient on the interaction term and the coefficient on the dummy are not statistically significant. Thus, even with economic controls we cannot reject the null hypotheses, that the effect of the student-teacher ratio on test scores is the same for districts with high and districts with low share of English learning students.

Regression (5) includes a cubic term for the student-teacher ratio and omits the interaction between *size* and *HiEL*. The results indicate that there is a nonlinear effect of the student-teacher ratio on test scores (Can you verify this using an F -test of $H_0 : \beta_2 = \beta_3 = 0$?)

Table 8.2: Nonlinear Models of Test Scores

	Dependent Variable: Test Score						
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
size	-0.998*** (0.270)	-0.734*** (0.257)	-0.968 (0.589)	-0.531 (0.342)	64.339*** (24.861)	83.702*** (28.497)	65.285*** (25.259)
english	-0.122*** (0.033)	-0.176*** (0.034)					-0.166*** (0.034)
I(size^2)					-3.424*** (1.250)	-4.381*** (1.441)	-3.466*** (1.271)
I(size^3)					0.059*** (0.021)	0.075*** (0.024)	0.060*** (0.021)
lunch	-0.547*** (0.024)	-0.398*** (0.033)		-0.411*** (0.029)	-0.420*** (0.029)	-0.418*** (0.029)	-0.402*** (0.033)
log(income)		11.569*** (1.819)		12.124*** (1.798)	11.748*** (1.771)	11.800*** (1.778)	11.509*** (1.806)
HiEL			5.639 (19.515)	5.498 (9.795)	-5.474*** (1.034)	816.076** (327.674)	
size:HiEL			-1.277 (0.967)	-0.578 (0.496)		-123.282** (50.213)	
I(size^2):HiEL						6.121** (2.542)	
I(size^3):HiEL						-0.101** (0.043)	
Constant	700.150*** (5.568)	658.552*** (8.642)	682.246*** (11.868)	653.666*** (9.869)	252.050 (163.634)	122.353 (185.519)	244.809 (165.722)
Observations	420	420	420	420	420	420	420
R ²	0.775	0.796	0.310	0.797	0.801	0.803	0.801
Adjusted R ²	0.773	0.794	0.305	0.795	0.798	0.799	0.798

Note: *p<0.1; **p<0.05; ***p<0.01

Consequently, regression (6) further explores whether the fraction of English learners impacts the student-teacher ratio by using $HiEL \times size$ and the interactions $HiEL \times size^2$ and $HiEL \times size^3$. All individual t -tests indicate that there are significant effects. We check this using a robust F -test of $H_0 : \beta_6 = \beta_7 = \beta_8 = 0$.

```
# check joint significance of the interaction terms
linearHypothesis(TS_mod6,
  c("size:HiEL=0", "I(size^2):HiEL=0", "I(size^3):HiEL=0"),
  vcov. = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> size:HiEL = 0
#> I(size^2):HiEL = 0
#> I(size^3):HiEL = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ size + I(size^2) + I(size^3) + HiEL + HiEL:size + HiEL:I(size^2) +
#>   HiEL:I(size^3) + lunch + log(income)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     413
#> 2     410  3 2.1885 0.08882 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that the null can be rejected at the level of 5% and conclude that the regression function differs for districts with high and low percentage of English learners.

Specification (7) uses a continuous measure for the share of English learners instead of a dummy variable (and thus does not include interaction terms). We observe only small changes to the coefficient estimates on the other regressors and thus conclude that the results observed for specification (5) are not sensitive to the way the percentage of English learners is measured.

We continue by reproducing Figure 8.10 of the book for interpretation of the nonlinear specifications (2), (5) and (7).

```
# scatterplot
plot(CASchools$size,
  CASchools$score,
  xlim = c(12, 28),
  ylim = c(600, 740),
  pch = 20,
  col = "gray",
  xlab = "Student-Teacher Ratio",
  ylab = "Test Score")

# add a legend
legend("top",
  legend = c("Linear Regression (2)",
    "Cubic Regression (5)",
    "Cubic Regression (7)"),
```

```
cex = 0.6,
ncol = 3,
lty = c(1, 1, 2),
col = c("blue", "red", "black"))

# data for use with predict()
new_data <- data.frame("size" = seq(16, 24, 0.05),
                      "english" = mean(CASchools$english),
                      "lunch" = mean(CASchools$lunch),
                      "income" = mean(CASchools$income),
                      "HiEL" = mean(CASchools$HiEL))

# add estimated regression function for model (2)
fitted <- predict(TS_mod2, newdata = new_data)

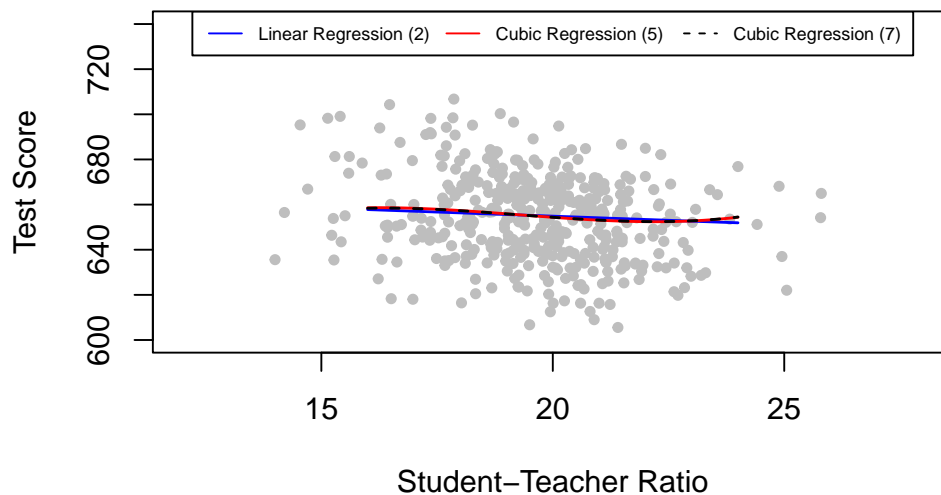
lines(new_data$size,
      fitted,
      lwd = 1.5,
      col = "blue")

# add estimated regression function for model (5)
fitted <- predict(TS_mod5, newdata = new_data)

lines(new_data$size,
      fitted,
      lwd = 1.5,
      col = "red")

# add estimated regression function for model (7)
fitted <- predict(TS_mod7, newdata = new_data)

lines(new_data$size,
      fitted,
      col = "black",
      lwd = 1.5,
      lty = 2)
```



For the above figure all regressors except *size* are set to their sample averages. We see that the cubic regressions (5) and (7) are almost identical. They indicate that the relation between test scores and the student-teacher ratio only has a small amount of nonlinearity since they do not deviate much from the regression function of (2).

The next code chunk reproduces Figure 8.11 of the book. We use `plot()` and `points()` to color observations depending on *HiEL*. Again, the regression lines are drawn based on predictions using average sample averages of all regressors except for *size*.

```
# draw scatterplot

# observations with HiEL = 0
plot(CASchools$size[CASchools$HiEL == 0],
     CASchools$score[CASchools$HiEL == 0],
     xlim = c(12, 28),
     ylim = c(600, 730),
     pch = 20,
     col = "gray",
     xlab = "Student-Teacher Ratio",
     ylab = "Test Score")

# observations with HiEL = 1
points(CASchools$size[CASchools$HiEL == 1],
       CASchools$score[CASchools$HiEL == 1],
       col = "steelblue",
       pch = 20)

# add a legend
legend("top",
      legend = c("Regression (6) with HiEL=0", "Regression (6) with HiEL=1"),
      cex = 0.7,
      ncol = 2,
      lty = c(1, 1),
      col = c("green", "red"))

# data for use with 'predict()'
new_data <- data.frame("size" = seq(12, 28, 0.05),
```

```

    "english" = mean(CASchools$english),
    "lunch" = mean(CASchools$lunch),
    "income" = mean(CASchools$income),
    "HiEL" = 0)

# add estimated regression function for model (6) with HiEL=0
fitted <- predict(TS_mod6, newdata = new_data)

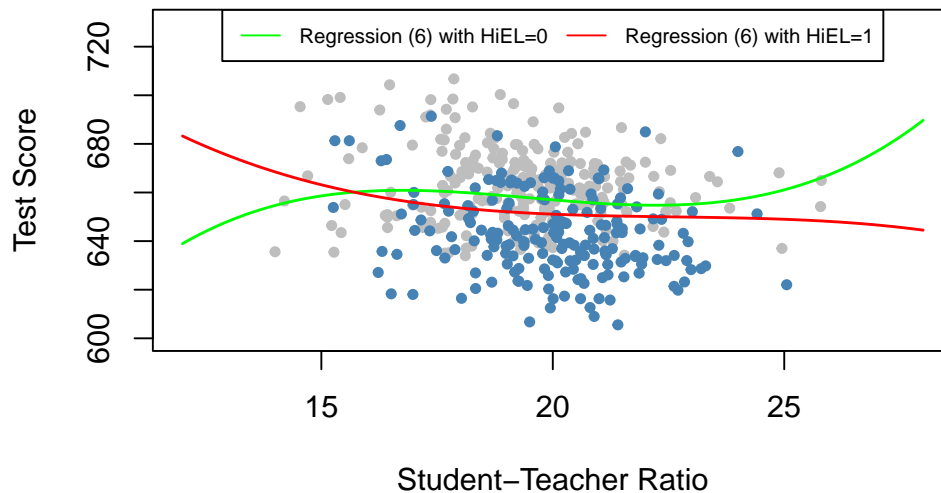
lines(new_data$size,
      fitted,
      lwd = 1.5,
      col = "green")

# add estimated regression function for model (6) with HiEL=1
new_data$HiEL <- 1

fitted <- predict(TS_mod6, newdata = new_data)

lines(new_data$size,
      fitted,
      lwd = 1.5,
      col = "red")

```



The regression output shows that model (6) finds statistically significant coefficients on the interaction terms $HiEL : size$, $HiEL : size^2$ and $HiEL : size^3$, i.e., there is evidence that the nonlinear relationship connecting test scores and student-teacher ratio depends on the fraction of English learning students in the district. However, the above figure shows that this difference is not of practical importance and is a good example for why one should be careful when interpreting nonlinear models: although the two regression functions look different, we see that the slope of both functions is almost identical for student-teacher ratios between 17 and 23. Since this range includes almost 90% of all observations, we can be confident that nonlinear interactions between the fraction of English learners and the student-teacher ratio can be neglected.

One might be tempted to object since both functions show opposing slopes for student-teacher ratios below 15 and beyond 24. There are at least two possible objections:

1. There are only few observations with low and high values of the student-teacher ratio, so there is

only little information to be exploited when estimating the model. This means the estimated function is less precise in the tails of the data set.

2. The above described behavior of the regression function, is a typical caveat when using cubic functions since they generally show extreme behavior for extreme regressor values. Think of the graph of $f(x) = x^3$.

We thus find no clear evidence for a relation between class size and test scores on the percentage of English learners in the district.

Summary

We are now able to answer the three questions posed at the beginning of this section.

1. In the linear models, the percentage of English learners has only little influence on the effect on test scores from changing the student-teacher ratio. This result stays valid if we control for economic background of the students. While the cubic specification (6) provides evidence that the effect of the student-teacher ratio on test score depends on the share of English learners, the strength of this effect is negligible.
2. When controlling for the students' economic background we find evidence of nonlinearities in the relationship between student-teacher ratio and test scores.
3. The linear specification (2) predicts that a reduction of the student-teacher ratio by two students per teacher leads to an improvement in test scores of about $-0.73 \times (-2) = 1.46$ points. Since the model is linear, this effect is independent of the class size. Assume that the student-teacher ratio is 20. For example, the nonlinear model (5) predicts that the reduction increases test scores by

$$64.33 \cdot 18 + 18^2 \cdot (-3.42) + 18^3 \cdot (0.059) - (64.33 \cdot 20 + 20^2 \cdot (-3.42) + 20^3 \cdot (0.059)) \approx 3.3$$

points. If the ratio is 22, a reduction to 20 leads to a predicted improvement in test scores of

$$64.33 \cdot 20 + 20^2 \cdot (-3.42) + 20^3 \cdot (0.059) - (64.33 \cdot 22 + 22^2 \cdot (-3.42) + 22^3 \cdot (0.059)) \approx 2.4$$

points. This suggests that the effect is stronger in smaller classes.

8.5 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 9

Assessing Studies Based on Multiple Regression

The majority of Chapter 9 of the book is of a theoretical nature. Therefore this section briefly reviews the concepts of internal and external validity in general and discusses examples of threats to internal and external validity of multiple regression models. We discuss consequences of

- misspecification of the functional form of the regression function
- measurement errors
- missing data and sample selection
- simultaneous causality

as well as sources of inconsistency of OLS standard errors. We also review concerns regarding internal validity and external validity in the context of forecasting using regression models.

The chapter closes with an application in R where we assess whether results found by multiple regression using the `CASchools` data can be generalized to school districts of another federal state of the United States.

For a more detailed treatment of these topics we encourage you to work through Chapter 9 of the book.

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter:

- `AER`
- `mvtnorm`
- `stargazer`

```
library(AER)
library(mvtnorm)
library(stargazer)
```

9.1 Internal and External Validity

Key Concept 9.1 Internal and External Validity

A statistical analysis has *internal* validity if the statistical inference made about causal effects are valid for the considered population.

An analysis is said to have *external* validity if inferences and conclusion are valid for the studies' population and can be generalized to other populations and settings.

Threats to Internal Validity

There are two conditions for internal validity to exist:

- I. The estimator of the causal effect, which is measured by the coefficient(s) of interest, should be unbiased and consistent.
- II. Statistical inference is valid, that is, hypothesis tests should have the desired size and confidence intervals should have the desired coverage probability.

In multiple regression, we estimate the model coefficients using OLS. Thus for condition I to be fulfilled we need the OLS estimator to be unbiased and consistent. For the second condition to be valid, the standard errors must be valid such that hypothesis testing and computation of confidence intervals yield results that are trustworthy. Remember that a sufficient condition for conditions 1. and 2. to be fulfilled is that the assumptions of Key Concept 6.4 hold.

Threats to External Validity

External validity might be invalid

- if there are differences between the population studied and the population of interest.
- if there are differences in the *settings* of the considered populations, e.g., the legal framework or the time of the investigation.

9.2 Threats to Internal Validity of Multiple Regression Analysis

This section treats five sources that cause the OLS estimator in (multiple) regression models to be biased and inconsistent for the causal effect of interest and discusses possible remedies. All five sources imply a violation of the first least squares assumption presented in Key Concept 6.4.

This sections treats:

- omitted variable bias,
- misspecification of the functional form,
- measurement errors,
- missing data and sample selection,
- simultaneous causality bias.

Beside these threats for consistency of the estimator, we also briefly discuss causes of inconsistent estimation of OLS standard errors.

Omitted Variable Bias**Key Concept 9.2****Omitted Variable Bias: Should I include More Variables in My Regression?**

Inclusion of additional variables reduces the risk of omitted variable bias but may increase the variance of the estimator of the coefficient of interest.

We present some guidelines that help deciding whether to include an additional variable:

- Specify the coefficient(s) of interest.
- Identify the most important potential sources of omitted variable bias by using knowledge available *before* estimating the model. You should end up with a base specification and a set of regressors that are questionable.
- Use different model specifications to test whether questionable regressors have coefficients different from zero.
- Use tables to provide full disclosure of your results, i.e., present different model specifications that both support your argument and enable the reader to see the effect of including questionable regressors.

By now you should be aware of omitted variable bias and its consequences. Key Concept 9.2 gives some guidelines on how to proceed if there are control variables that possibly allow to reduce omitted variable bias. If including additional variables to mitigate the bias is not an option because there are no adequate controls, there are different approaches to solve the problem:

- usage of panel data methods (discussed in Chapter 10),
- usage of instrumental variables regression (discussed in Chapter 12),
- usage of a randomized control experiment (discussed in Chapter 13).

Misspecification of the Functional Form of the Regression Function

If the population regression function is nonlinear but the regression function is linear, the functional form of the regression model is misspecified. This leads to a bias of the OLS estimator.

Key Concept 9.3**Functional Form Misspecification**

We say a regression suffers from misspecification of the functional form when the functional form of the estimated regression model differs from the functional form of the population regression function. Functional form misspecification leads to biased and inconsistent coefficient estimators. A way to detect functional form misspecification is to plot the estimated regression function and the data. This may also be helpful to choose the correct functional form.

It is easy to come up with examples of misspecification of the functional form: consider the case where the population regression function is

$$Y_i = X_i^2,$$

but the model used is

$$Y_i = \beta_0 + \beta_1 X_i + u_i.$$

Clearly, the regression function is misspecified here. We now simulate data and visualize this.

```
# set seed for reproducibility
set.seed(3)

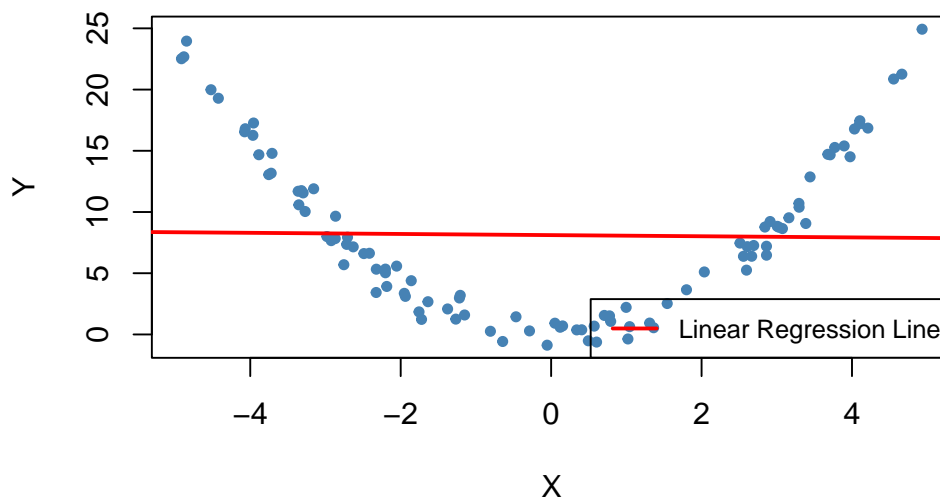
# simulate data set
X <- runif(100, -5, 5)
Y <- X^2 + rnorm(100)

# estimate the regression function
ms_mod <- lm(Y ~ X)
ms_mod
#>
#> Call:
#> lm(formula = Y ~ X)
#>
#> Coefficients:
#> (Intercept)          X
#>   8.11363    -0.04684

# plot the data
plot(X, Y,
      main = "Misspecification of Functional Form",
      pch = 20,
      col = "steelblue")

# plot the linear regression line
abline(ms_mod,
       col = "red",
       lwd = 2)
legend("bottomright",
      bg = "transparent",
      cex = 0.8,
      lwd = 2,
      col = "red",
      legend = "Linear Regression Line")
```

Misspecification of Functional Form



It is evident that the regression errors are relatively small for observations close to $X = -3$ and $X = 3$ but that the errors increase for X values closer to zero and even more for values beyond -4 and 4 . Consequences are drastic: the intercept is estimated to be 8.1 and for the slope parameter, we obtain an estimate obviously very close to zero. This issue does not disappear as the number of observations is increased because OLS is biased *and* inconsistent due to the misspecification of the regression function.

Measurement Error and Errors-in-Variables Bias

Key Concept 9.4 Errors-in-Variable Bias

When independent variables are measured imprecisely, we speak of errors-in-variables bias. This bias does not disappear if the sample size is large. If the measurement error has mean zero and is independent of the affected variable, the OLS estimator of the respective coefficient is biased towards zero.

Suppose you are incorrectly measuring the single regressor X_i so that there is a measurement error and you observe \tilde{X}_i instead of X_i . Then, instead of estimating the regression model for the population, which is :

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

, we end up estimating the regression model below:

$$Y_i = \beta_0 + \beta_1 \tilde{X}_i + \underbrace{\beta_1 (X_i - \tilde{X}_i)}_{=v_i} + u_i.$$

$$Y_i = \beta_0 + \beta_1 \tilde{X}_i + v_i.$$

where \tilde{X}_i and the error term v_i are correlated. Thus OLS would be biased and inconsistent for the true β_1 in this example. One can show that direction and strength of the bias depend on the correlation between the observed regressor, \tilde{X}_i , and the measurement error, $w_i = X_i - \tilde{X}_i$. This correlation in turn depends on the type of the measurement error made.

The classical measurement error model assumes that the measurement error, w_i , has zero mean and that it is uncorrelated with the variable, X_i , and the error term of the population regression model, u_i :

$$\tilde{X}_i = X_i + w_i, \quad \rho_{w_i, u_i} = 0, \quad \rho_{w_i, X_i} = 0. \quad (9.1)$$

Then it holds that

$$\hat{\beta}_1 \xrightarrow{p} \frac{\sigma_X^2}{\sigma_X^2 + \sigma_w^2} \beta_1, \quad (9.2)$$

which implies inconsistency as $\sigma_X^2, \sigma_w^2 > 0$ such that the fraction in (9.2) is smaller than 1. Note that there are two extreme cases:

1. If there is no measurement error, $\sigma_w^2 = 0$ such that $\hat{\beta}_1 \xrightarrow{p} \beta_1$.
2. If $\sigma_w^2 \gg \sigma_X^2$ we have $\hat{\beta}_1 \xrightarrow{p} 0$. This is the case if the measurement error is so large that there essentially is no information on X in the data that can be used to estimate β_1 .

The most obvious way to deal with errors-in-variables bias is to use an accurately measured X . If this is not possible, instrumental variables regression is an option. One might also deal with the issue by using a mathematical model of the measurement error and adjust the estimates appropriately: if it is plausible that the classical measurement error model applies and if there is information that can be used to estimate the ratio in equation (9.2), one could compute an estimate that corrects for the downward bias.

For example, consider two bivariate normally distributed random variables X, Y . It is a well known result that the conditional expectation function of Y given X has the form

$$E(Y|X) = E(Y) + \rho_{X,Y} \frac{\sigma_Y}{\sigma_X} [X - E(X)]. \quad (9.3)$$

Thus for

$$(X, Y) \sim \mathcal{N} \left[\begin{pmatrix} 50 \\ 100 \end{pmatrix}, \begin{pmatrix} 10 & 5 \\ 5 & 10 \end{pmatrix} \right] \quad (9.4)$$

according to (9.3), the population regression function is

$$\begin{aligned} Y_i &= 100 + 0.5(X_i - 50) \\ &= 75 + 0.5X_i. \end{aligned}$$

Now suppose you gather data on X and Y , but that you can only measure $\tilde{X}_i = X_i + w_i$ with $w_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 10)$. Since the w_i are independent of the X_i , there is no correlation between the X_i and the w_i so that we have a case of the classical measurement error model. We now illustrate this example in R using the package `mvtnorm` (Genz et al., 2023).

```
# set seed
set.seed(1)

# load the package 'mvtnorm' and simulate bivariate normal data
library(mvtnorm)
dat <- data.frame(
  rmvnorm(1000, c(50, 100),
    sigma = cbind(c(10, 5), c(5, 10))))

# set columns names
colnames(dat) <- c("X", "Y")
```

We now estimate a simple linear regression of Y on X using this sample data and run the same regression again but this time we add i.i.d. $\mathcal{N}(0, 10)$ errors added to X .

```
# estimate the model (without measurement error)
noerror_mod <- lm(Y ~ X, data = dat)

# estimate the model (with measurement error in X)
dat$X <- dat$X + rnorm(n = 1000, sd = sqrt(10))
error_mod <- lm(Y ~ X, data = dat)

# print estimated coefficients to console
noerror_mod$coefficients
#> (Intercept)          X
#> 76.3002047  0.4755264
error_mod$coefficients
#> (Intercept)          X
#> 87.276004  0.255212
```

Next, we visualize the results and compare with the population regression function.

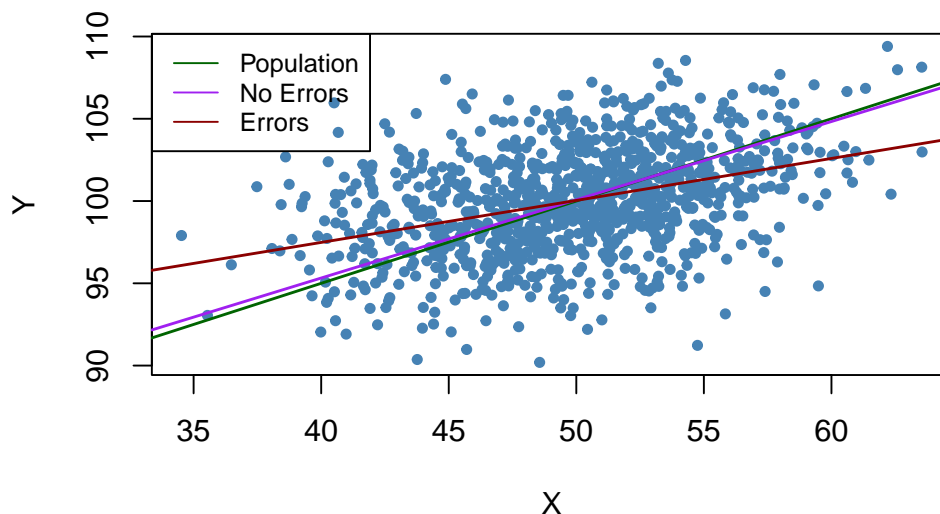
```
# plot sample data
plot(dat$X, dat$Y,
     pch = 20,
     col = "steelblue",
     xlab = "X",
     ylab = "Y")

# add population regression function
abline(coef = c(75, 0.5),
       col = "darkgreen",
       lwd = 1.5)

# add estimated regression functions
abline(noerror_mod,
       col = "purple",
       lwd = 1.5)

abline(error_mod,
       col = "darkred",
       lwd = 1.5)

# add legend
legend("topleft",
      bg = "transparent",
      cex = 0.8,
      lty = 1,
      col = c("darkgreen", "purple", "darkred"),
      legend = c("Population", "No Errors", "Errors"))
```



In the situation without measurement error, the estimated regression function is close to the population regression function. Things are different when we use the mismeasured regressor X : both the estimate for the intercept and the estimate for the coefficient on X differ considerably from results obtained using the “clean” data on X . In particular $\hat{\beta}_1 = 0.255$, so there is a downward bias. We are in the comfortable situation to know σ_X^2 and σ_w^2 . This allows us to correct for the bias using (9.2). Using this information we obtain the biased-corrected estimate

$$\frac{\sigma_X^2 + \sigma_w^2}{\sigma_X^2} \cdot \hat{\beta}_1 = \frac{10 + 10}{10} \cdot 0.255 = 0.51$$

which is quite close to $\beta_1 = 0.5$, the true coefficient from the population regression function.

Bear in mind that the above analysis uses a single sample. Thus one may argue that the results are just a coincidence. Can you show the contrary using a simulation study?

Missing Data and Sample Selection

Key Concept 9.5 Sample Selection Bias

When the sampling process influences the availability of data and when there is a relation of this sampling process to the dependent variable that goes beyond the dependence on the regressors, we say that there is a sample selection bias. This bias is due to correlation between one or more regressors and the error term. Sample selection implies both bias and inconsistency of the OLS estimator.

There are three cases of sample selection. Only one of them poses a threat to internal validity of a regression study. The three cases are:

1. Data are missing at random.
2. Data are missing based on the value of a regressor.
3. Data are missing due to a selection process which is related to the dependent variable.

Let us jump back to the example of variables X and Y distributed as stated in equation (9.4) and illustrate all three cases using R.

If data are missing at random, this is nothing but losing the observations. For example, losing 50% of the sample would be the same as never having seen the (randomly chosen) half of the sample observed. Therefore, missing data do not introduce an estimation bias and “only” lead to less efficient estimators.

```
# set seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50, 100),
    sigma = cbind(c(10, 5), c(5, 10)))

colnames(dat) <- c("X", "Y")

# mark 500 randomly selected observations
id <- sample(1:1000, size = 500)

plot(dat$X[-id],
  dat$Y[-id],
  col = "steelblue",
  pch = 20,
  cex = 0.8,
  xlab = "X",
  ylab = "Y")

points(dat$X[id],
  dat$Y[id],
  cex = 0.8,
  col = "gray",
  pch = 20)

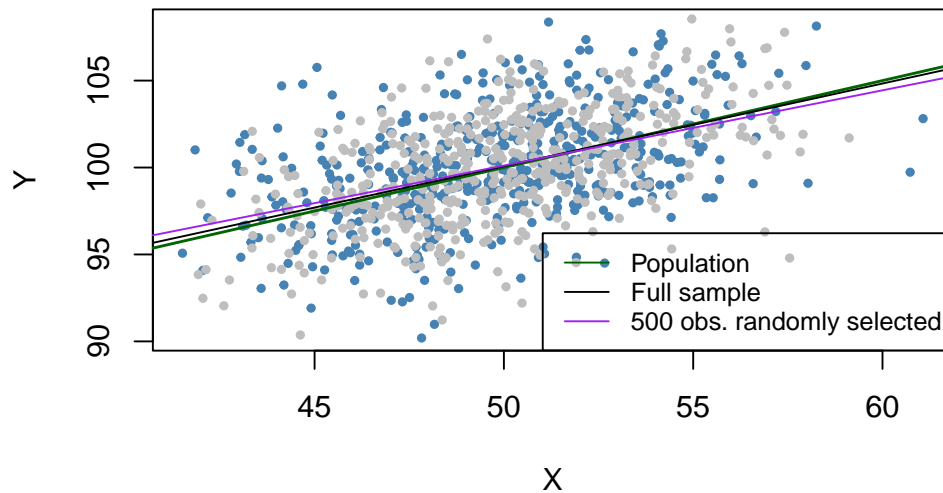
# add the population regression function
abline(coef = c(75, 0.5),
  col = "darkgreen",
  lwd = 1.5)

# add the estimated regression function for the full sample
abline(noerror_mod)

# estimate model case 1 and add the regression line
dat <- dat[-id, ]

c1_mod <- lm(dat$Y ~ dat$X, data = dat)
abline(c1_mod, col = "purple")

# add a legend
legend("bottomright",
  lty = 1,
  bg = "transparent",
  cex = 0.8,
  col = c("darkgreen", "black", "purple"),
  legend = c("Population", "Full sample", "500 obs. randomly selected"))
```



The gray dots represent the 500 discarded observations. When using the remaining observations, the estimation results deviate only marginally from the results obtained using the full sample.

Selecting data randomly based on the value of a regressor has also the effect of reducing the sample size and does not introduce estimation bias. We will now drop all observations with $X > 45$, estimate the model again and compare.

```
# set random seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50, 100),
    sigma = cbind(c(10, 5), c(5, 10))))

colnames(dat) <- c("X", "Y")

# mark observations
id <- dat$X >= 45

plot(dat$X[-id],
  dat$Y[-id],
  col = "steelblue",
  cex = 0.8,
  pch = 20,
  xlab = "X",
  ylab = "Y")

points(dat$X[id],
  dat$Y[id],
  col = "gray",
  cex = 0.8,
  pch = 20)

# add population regression function
abline(coef = c(75, 0.5),
  col = "darkgreen",
```



```

    lwd = 1.5)

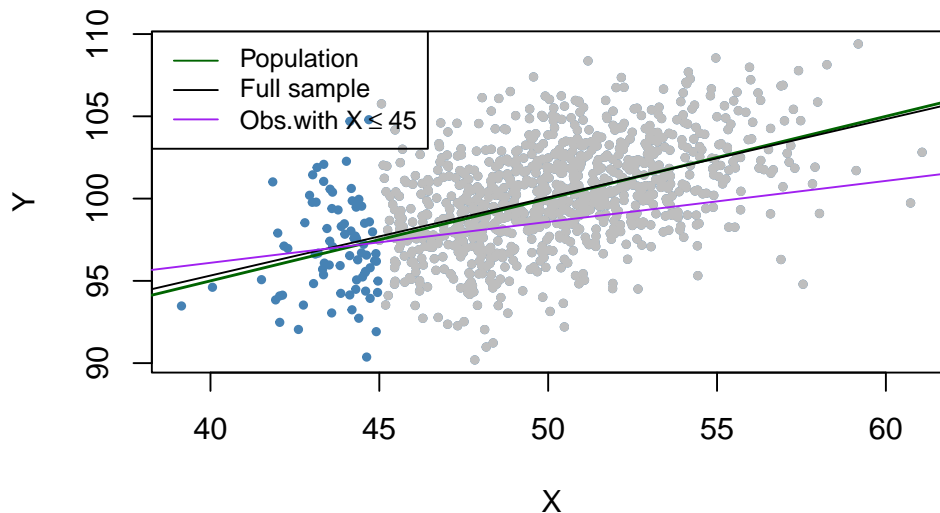
# add estimated regression function for full sample
abline(noerror_mod)

# estimate model case 1, add regression line
id <- which(dat$X >= 45)
dat <- dat[-id, ]

c2_mod <- lm(dat$Y ~ dat$X, data = dat)
abline(c2_mod, col = "purple")

# add legend
legend("topleft",
      lty = 1,
      bg = "transparent",
      cex = 0.8,
      col = c("darkgreen", "black", "purple"),
      legend = c("Population", "Full sample", expression(paste("Obs.with ",
                                                                X <= 45))))

```



Note that although we dropped more than 90% of all observations, the estimated regression function is very close to the line estimated based on the full sample.

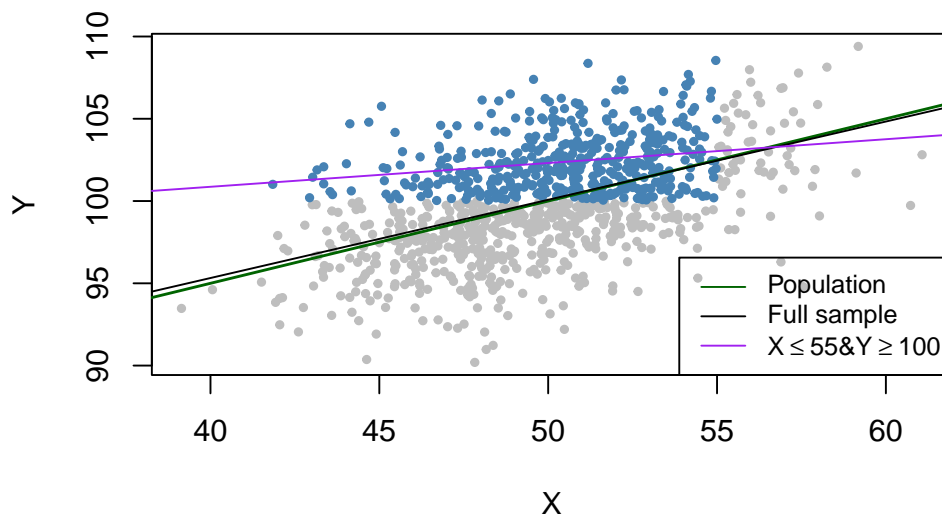
In the third case we face sample selection bias. We can illustrate this by using only observations with $X_i < 55$ and $Y_i > 100$. These observations are easily identified using the function `which()` and logical operators: `which(dat$X < 55 & dat$Y > 100)`.

```

# set random seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50,100),
          sigma = cbind(c(10,5), c(5,10))))

```

We see that the selection process leads to biased estimation results.

There are methods that allow to correct for sample selection bias. However, these methods are beyond the scope of the book and are therefore not considered here. The concept of sample selection bias is summarized in Key Concept 9.5.

Simultaneous Causality

Key Concept 9.6 Simultaneous Causality Bias

So far we have assumed that the changes in the independent variable X are responsible for the changes in the dependent variable Y . When the reverse is also true, we say that there is *simultaneous causality* between X and Y . This reverse causality leads to correlation between X and the error in the population regression of interest such that the coefficient on X is estimated with a bias.

Suppose we are interested in estimating the effect of a 20% increase in cigarette prices on cigarette consumption in the United States using a multiple regression model. This may be investigated using the dataset `CigarettesSW` which is a part of the `AER` package. `CigarettesSW` is a panel data set on cigarette consumption for all 48 continental U.S. federal states from 1985-1995 and provides data on economic indicators and average local prices, taxes and per capita pack consumption.

After loading the data set, we pick observations for the year 1995 and plot logarithms of the per pack price, `price`, against pack consumption, `packs`, and estimate a simple linear regression model.

```
# load the data set
library(AER)
data("CigarettesSW")
c1995 <- subset(CigarettesSW, year == "1995")

# estimate the model
cigcon_mod <- lm(log(packs) ~ log(price), data = c1995)
cigcon_mod
#>
#> Call:
```

```

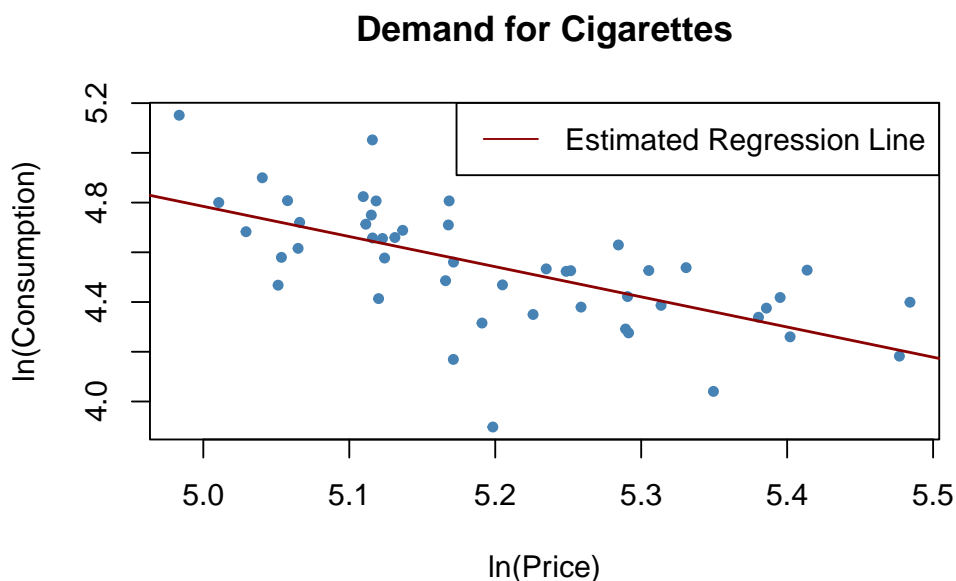
#> lm(formula = log(packs) ~ log(price), data = c1995)
#>
#> Coefficients:
#> (Intercept)    log(price)
#>      10.850      -1.213

# plot the estimated regression line and the data
plot(log(c1995$price), log(c1995$packs),
     xlab = "ln(Price)",
     ylab = "ln(Consumption)",
     main = "Demand for Cigarettes",
     pch = 20,
     col = "steelblue")

abline(cigcon_mod,
       col = "darkred",
       lwd = 1.5)

# add legend
legend("topright", lty=1, col= "darkred", "Estimated Regression Line")

```



Remember from Chapter 8 that, due to the log-log specification, in the population regression the coefficient on the logarithm of price is interpreted as the price elasticity of consumption. The estimated coefficient suggests that a 1% increase in cigarette prices reduces cigarette consumption by about 1.2%, on average. Have we estimated a demand curve? The answer is no: this is a classic example of simultaneous causality, see Key Concept 9.6. The observations are market equilibrium which are determined by both changes in supply and changes in demand. Therefore the price is correlated with the error term and the OLS estimator is biased. We can neither estimate a demand nor a supply curve consistently using this approach.

We will return to this issue in Chapter 12 which treats instrumental variables regression, an approach that allows consistent estimation when there is simultaneous causality.

Sources of Inconsistency of OLS Standard Errors

There are two central threats to computation of consistent OLS standard errors:

1. Heteroskedasticity: implications of heteroskedasticity have been discussed in Chapter 5. Heteroskedasticity-robust standard errors as computed by the function `vcovHC()` from the package `sandwich` produce valid standard errors under heteroskedasticity.
2. Serial correlation: if the population regression error is correlated across observations, we have serial correlation. This often happens in applications where repeated observations are used, e.g., in panel data studies. As for heteroskedasticity, `vcovHC()` can be used to obtain valid standard errors when there is serial correlation.

Inconsistent standard errors will produce invalid hypothesis tests and wrong confidence intervals. For example, when testing the null that some model coefficient is zero, we cannot trust the outcome anymore because the test may fail to have a size of 5% due to the wrongly computed standard error.

Key Concept 9.7 summarizes all threats to internal validity discussed above.

Key Concept 9.7 Threats to Internal Validity of a Regression Study

The five primary threats to internal validity of a multiple regression study are:

1. Omitted variables.
2. Misspecification of functional form.
3. Errors in variables (measurement errors in the regressors).
4. Sample selection.
5. Simultaneous causality.

All these threats lead to failure of the first least squares assumption

$$E(u_i | X_{1i}, \dots, X_{ki}) \neq 0,$$

so that the OLS estimator is biased *and* inconsistent.

Furthermore, if one does not adjust for heteroskedasticity *and/or* serial correlation, incorrect standard errors may be a threat to internal validity of the study.

9.3 Internal and External Validity when the Regression is used for Forecasting

Recall the regression of test scores on the student-teacher ratio (*STR*) performed in Chapter 4:

```
linear_model <- lm(score ~ STR, data = CASchools)
linear_model
#>
#> Call:
#> lm(formula = score ~ STR, data = CASchools)
#>
#> Coefficients:
#> (Intercept)          STR
#>    698.93         -2.28
```

The estimated regression function was

$$\widehat{TestScore} = 698.9 - 2.28 \times STR.$$

The book discusses the example of a parent moving to a metropolitan area who plans to choose where to live based on the quality of local schools: a school district's average test score is an adequate measure for the quality. However, the parent has information on the student-teacher ratio only such that test scores need to be predicted. Although we have established that there is omitted variable bias in this model due to omission of variables like student learning opportunities outside school, the share of English learners and so on, `linear_model` may in fact be useful for the parent:

The parent need not care if the coefficient on *STR* has causal interpretation, she wants *STR* to explain as much variation in test scores as possible. Therefore, despite the fact that `linear_model` cannot be used to estimate the *causal effect* of a change in *STR* on test scores, it can be considered a *reliable predictor* of test scores in general.

Thus, the threats to internal validity as summarized in Key Concept 9.7 are negligible for the parent. This is, as instanced in the book, different for a superintendent who has been tasked to take measures that increase test scores: she requires a more reliable model that does not suffer from the threats listed in Key Concept 9.7.

Consult Chapter 9.3 of the book (Stock and Watson) for the corresponding discussion.

9.4 Example: Test Scores and Class Size

This section discusses internal and external validity of the results gained from analyzing the California test score data using multiple regression models.

External Validity of the Study

External validity of the California test score analysis means that its results can be generalized. Whether this is possible depends on the population and the setting. Following the book we conduct the same analysis using data for fourth graders in 220 public school districts in Massachusetts in 1998. Like `CASchools`, the data set `MASchools` is part of the `AER` package (Kleiber and Zeileis, 2008b). Use the help function (`?MASchools`) to get information on the definitions of all the variables contained.

We start by loading the data set and proceed by computing some summary statistics.

```
# attach the 'MASchools' dataset
data("MASchools")
summary(MASchools)
#>   district      municipality      expreg      expspecial
#> Length:220      Length:220      Min.   :2905      Min.   : 3832
#> Class :character Class :character 1st Qu.:4065      1st Qu.: 7442
#> Mode  :character Mode  :character Median :4488      Median : 8354
#>
#>      Mean   :4605      Mean   : 8901
#>      3rd Qu.:4972      3rd Qu.: 9722
#>      Max.   :8759      Max.   :53569
#>
#>      expbil      expocc      exptot      scratio
#> Min.   : 0      Min.   : 0      Min.   :3465      Min.   : 2.300
#> 1st Qu.: 0      1st Qu.: 0      1st Qu.:4730      1st Qu.: 6.100
#> Median : 0      Median : 0      Median :5155      Median : 7.800
#> Mean   :3037      Mean   :1104      Mean   :5370      Mean   : 8.107
```

```

#> 3rd Qu.: 0 3rd Qu.: 0 3rd Qu.:5789 3rd Qu.: 9.800
#> Max. :295140 Max. :15088 Max. :9868 Max. :18.400
#> NA's :9
#> special lunch stratio income
#> Min. : 8.10 Min. : 0.40 Min. :11.40 Min. : 9.686
#> 1st Qu.:13.38 1st Qu.: 5.30 1st Qu.:15.80 1st Qu.:15.223
#> Median :15.45 Median :10.55 Median :17.10 Median :17.128
#> Mean :15.97 Mean :15.32 Mean :17.34 Mean :18.747
#> 3rd Qu.:17.93 3rd Qu.:20.02 3rd Qu.:19.02 3rd Qu.:20.376
#> Max. :34.30 Max. :76.20 Max. :27.00 Max. :46.855
#>
#> score4 score8 salary english
#> Min. :658.0 Min. :641.0 Min. :24.96 Min. : 0.0000
#> 1st Qu.:701.0 1st Qu.:685.0 1st Qu.:33.80 1st Qu.: 0.0000
#> Median :711.0 Median :698.0 Median :35.88 Median : 0.0000
#> Mean :709.8 Mean :698.4 Mean :35.99 Mean : 1.1177
#> 3rd Qu.:720.0 3rd Qu.:712.0 3rd Qu.:37.96 3rd Qu.: 0.8859
#> Max. :740.0 Max. :747.0 Max. :44.49 Max. :24.4939
#> NA's :40 NA's :25

```

It is fairly easy to replicate key components of Table 9.1 of the book using R. To be consistent with variable names used in the CASchools data set, we do some formatting beforehand.

```

# Customized variables in MASchools
MASchools$score <- MASchools$score4
MASchools$STR <- MASchools$stratio

# Reproduce Table 9.1 of the book
vars <- c("score", "STR", "english", "lunch", "income")

cbind(CA_mean = sapply(CASchools[, vars], mean),
      CA_sd = sapply(CASchools[, vars], sd),
      MA_mean = sapply(MASchools[, vars], mean),
      MA_sd = sapply(MASchools[, vars], sd))
#> CA_mean CA_sd MA_mean MA_sd
#> score 654.15655 19.053347 709.827273 15.126474
#> STR 19.64043 1.891812 17.344091 2.276666
#> english 15.76816 18.285927 1.117676 2.900940
#> lunch 44.70524 27.123381 15.315909 15.060068
#> income 15.31659 7.225890 18.746764 5.807637

```

The summary statistics reveal that the average test score is higher for school districts in Massachusetts. The test used in Massachusetts is somewhat different from the one used in California (the Massachusetts test score also includes results for the school subject “Science”), therefore a direct comparison of test scores is not appropriate. We also see that, on average, classes are smaller in Massachusetts than in California and that the average district income, average percentage of English learners as well as the average share of students receiving subsidized lunch differ considerably from the averages computed for California. There are also notable differences in the observed dispersion of the variables.

Following the book we examine the relationship between district income and test scores in Massachusetts as we have done before in Chapter 8 for the California data and reproduce Figure 9.2 of the book.

```

# estimate linear model
Linear_model_MA <- lm(score ~ income, data = MASchools)
Linear_model_MA
#>
#> Call:
#> lm(formula = score ~ income, data = MASchools)
#>
#> Coefficients:
#> (Intercept)      income
#>    679.387      1.624

# estimate linear-log model
Linearlog_model_MA <- lm(score ~ log(income), data = MASchools)
Linearlog_model_MA
#>
#> Call:
#> lm(formula = score ~ log(income), data = MASchools)
#>
#> Coefficients:
#> (Intercept)  log(income)
#>    600.80      37.71

# estimate Cubic model
cubic_model_MA <- lm(score ~ I(income) + I(income^2) + I(income^3), data = MASchools)
cubic_model_MA
#>
#> Call:
#> lm(formula = score ~ I(income) + I(income^2) + I(income^3), data = MASchools)
#>
#> Coefficients:
#> (Intercept)  I(income)  I(income^2)  I(income^3)
#>  600.398531  10.635382  -0.296887   0.002762

# plot data
plot(MASchools$income, MASchools$score,
     pch = 20,
     col = "steelblue",
     xlab = "District income",
     ylab = "Test score",
     xlim = c(0, 50),
     ylim = c(620, 780))

# add estimated regression line for the linear model
abline(Linear_model_MA, lwd = 2)

# add estimated regression function for Linear-log model
order_id <- order(MASchools$income)

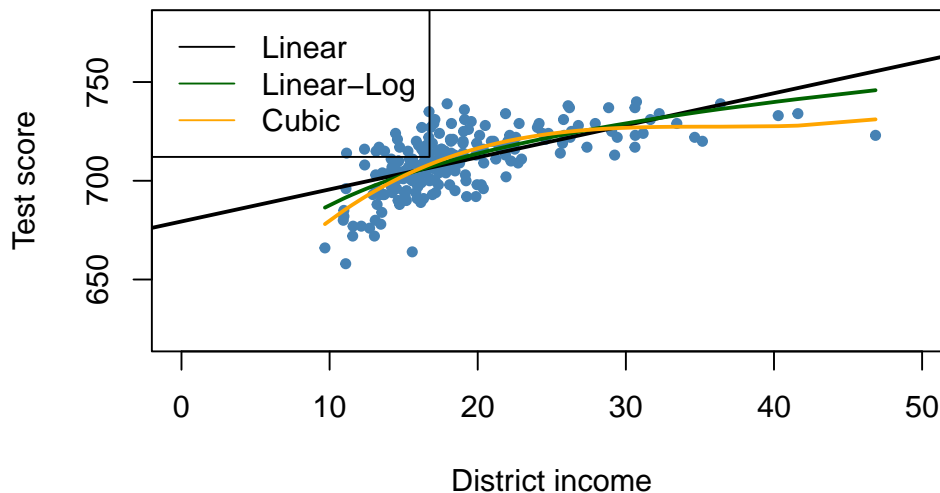
lines(MASchools$income[order_id],
      fitted(Linearlog_model_MA)[order_id],
      col = "darkgreen",
      lwd = 2)

```



```
# add estimated cubic regression function
lines(x = MASchools$income[order_id],
      y = fitted(cubic_model_MA)[order_id],
      col = "orange",
      lwd = 2)

# add a legend
legend("topleft",
      legend = c("Linear", "Linear-Log", "Cubic"),
      lty = 1,
      col = c("Black", "darkgreen", "orange"))
```



The plot indicates that the cubic specification fits the data best. Interestingly, this is different from the CASchools data where the pattern of nonlinearity is better described by the linear-log specification.

We continue by estimating most of the model specifications used for analysis of the CASchools data set in Chapter 8 and use `stargazer()` (Hlavac, 2022) to generate a tabular representation of the regression results.

```
# add 'HiEL' to 'MASchools'
MASchools$HiEL <- as.numeric(MASchools$english > median(MASchools$english))

# estimate the model specifications from Table 9.2 of the book
TSM1_mod1 <- lm(score ~ STR, data = MASchools)

TSM1_mod2 <- lm(score ~ STR + english + lunch + log(income),
                data = MASchools)

TSM1_mod3 <- lm(score ~ STR + english + lunch + income + I(income^2)
                + I(income^3), data = MASchools)

TSM1_mod4 <- lm(score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income
                + I(income^2) + I(income^3), data = MASchools)

TSM1_mod5 <- lm(score ~ STR + HiEL + I(income^2) + I(income^3) + HiEL:STR
                + lunch + income, data = MASchools)
```

```

TSMa_mod6 <- lm(score ~ STR + I(income^2) + I(income^3) + lunch
               + income, data = MASchools)

# gather robust standard errors
rob_se <- list(sqrt(diag(vcovHC(TSMa_mod1, type = "HC1"))),
               sqrt(diag(vcovHC(TSMa_mod2, type = "HC1"))),
               sqrt(diag(vcovHC(TSMa_mod3, type = "HC1"))),
               sqrt(diag(vcovHC(TSMa_mod4, type = "HC1"))),
               sqrt(diag(vcovHC(TSMa_mod5, type = "HC1"))),
               sqrt(diag(vcovHC(TSMa_mod6, type = "HC1"))))

# generate a table with 'stargazer()'
library(stargazer)

stargazer(TSMa_mod1, TSMa_mod2, TSMa_mod3,
          TSMa_mod4, TSMa_mod5, TSMa_mod6,
          title = "Regressions Using Massachusetts Test Score Data",
          type = "latex",
          digits = 3,
          header = FALSE,
          se = rob_se,
          object.names = TRUE,
          model.numbers = FALSE,
          column.labels = c("(I)", "(II)", "(III)", "(IV)", "(V)", "(VI)"))

```

% Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-mail: marek.hlavac@gmail.com % Date and time: Tue, Feb 13, 2024 - 10:19:10 % Requires LaTeX packages: rotating

Next we reproduce the F -statistics and p -values for testing exclusion of groups of variables.

```

# F-test model (3)
linearHypothesis(TSMa_mod3,
                 c("I(income^2)=0", "I(income^3)=0"),
                 vcov. = vcovHC(TSMa_mod3, type = "HC1"))

#> Linear hypothesis test
#>
#> Hypothesis:
#> I(income^2) = 0
#> I(income^3) = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + english + lunch + income + I(income^2) + I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F    Pr(>F)
#> 1      215
#> 2      213  2 7.7448 0.0005664 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# F-tests model (4)

```

Table 9.1: Regressions Using Massachusetts Test Score Data

	<i>Dependent variable:</i>					
	(I)	(II)	(III)	(IV)	(V)	(VI)
	score					
STR	-1.718*** (0.499)	-0.689** (0.270)	-0.641** (0.268)	12.426 (14.010)	-1.018*** (0.370)	-0.672** (0.271)
I(STR ²)				-0.680 (0.737)		
I(STR ³)				0.011 (0.013)		
english		-0.411 (0.306)	-0.437 (0.303)	-0.434 (0.300)		
HiEL					-12.561 (9.793)	
lunch		-0.521*** (0.078)	-0.582*** (0.097)	-0.587*** (0.104)	-0.709*** (0.091)	-0.653*** (0.073)
log(income)		16.529*** (3.146)				
income			-3.067 (2.353)	-3.382 (2.491)	-3.867 (2.488)	-3.218 (2.306)
I(income ²)			0.164* (0.085)	0.174* (0.089)	0.184** (0.090)	0.165* (0.085)
I(income ³)			-0.002** (0.001)	-0.002** (0.001)	-0.002** (0.001)	-0.002** (0.001)
STR:HiEL					0.799 (0.555)	
Constant	739.621*** (8.607)	682.432*** (11.497)	744.025*** (21.318)	665.496*** (81.332)	759.914*** (23.233)	747.364*** (20.278)
Observations	220	220	220	220	220	220
R ²	0.067	0.676	0.685	0.687	0.686	0.681
Adjusted R ²	0.063	0.670	0.676	0.675	0.675	0.674
Residual Std. Error	14.646 (df = 218)	8.686 (df = 215)	8.607 (df = 213)	8.626 (df = 211)	8.621 (df = 212)	8.637 (df = 214)

*p<0.1; **p<0.05; ***p<0.01

```

linearHypothesis(TSMA_mod4,
                  c("STR=0", "I(STR^2)=0", "I(STR^3)=0"),
                  vcov. = vcovHC(TSMA_mod4, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> STR = 0
#> I(STR^2) = 0
#> I(STR^3) = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
#>   I(income^2) + I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     214
#> 2     211  3 2.8565 0.03809 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

linearHypothesis(TSMA_mod4,
                  c("I(STR^2)=0", "I(STR^3)=0"),
                  vcov. = vcovHC(TSMA_mod4, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> I(STR^2) = 0
#> I(STR^3) = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
#>   I(income^2) + I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     213
#> 2     211  2 0.4463 0.6406

linearHypothesis(TSMA_mod4,
                  c("I(income^2)=0", "I(income^3)=0"),
                  vcov. = vcovHC(TSMA_mod4, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> I(income^2) = 0
#> I(income^3) = 0
#>
#> Model 1: restricted model

```

```

#> Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
#>   I(income^2) + I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F    Pr(>F)
#> 1     213
#> 2     211  2 7.7487 0.0005657 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#F-tests model (5)
linearHypothesis(TSMA_mod5,
                 c("STR=0", "STR:HiEL=0"),
                 vcov. = vcovHC(TSMA_mod5, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> STR = 0
#> STR:HiEL = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
#>   I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     214
#> 2     212  2 4.0062 0.0196 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
linearHypothesis(TSMA_mod5,
                 c("I(income^2)=0", "I(income^3)=0"),
                 vcov. = vcovHC(TSMA_mod5, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> I(income^2) = 0
#> I(income^3) = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
#>   I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F    Pr(>F)
#> 1     214
#> 2     212  2 5.8468 0.003375 **
#> ---

```

```

#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
  linearHypothesis(TSMA_mod5,
                   c("HiEL=0", "STR:HiEL=0"),
                   vcov. = vcovHC(TSMA_mod5, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> HiEL = 0
#> STR:HiEL = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
#>           I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     214
#> 2     212  2 1.5835 0.2077

# F-tests model (6)
linearHypothesis(TSMA_mod6,
                 c("I(income^2)=0", "I(income^3)=0"),
                 vcov. = vcovHC(TSMA_mod6, type = "HC1"))
#> Linear hypothesis test
#>
#> Hypothesis:
#> I(income^2) = 0
#> I(income^3) = 0
#>
#> Model 1: restricted model
#> Model 2: score ~ STR + lunch + income + I(income^2) + I(income^3)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F  Pr(>F)
#> 1     216
#> 2     214  2 6.5479 0.001737 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

We see that, in terms of \bar{R}^2 , specification (3) which uses a cubic to model the relationship between district income and test scores indeed performs better than the linear-log specification (2). Using different F -tests on models (4) and (5), we cannot reject the hypothesis that there is no nonlinear relationship between student teacher ratio and test score and also that the share of English learners has an influence on the relationship of interest. Furthermore, regression (6) shows that the percentage of English learners can be omitted as a regressor. Because of the model specifications made in (4) to (6) do not lead to substantially different results than those of regression (3), we choose model (3) as the most suitable specification.

In comparison to the California data, we observe the following results:

1. Controlling for the students' background characteristics in model specification (2) reduces the coefficient of interest (student-teacher ratio) by roughly 60%. The estimated coefficients are close to each other.
2. The coefficient on student-teacher ratio is always significantly different from zero at the level of 1% for both data sets. This holds for all considered model specifications in both studies.
3. In both studies, the share of English learners in a school district is of little importance for the estimated impact of a change in the student-teacher ratio on test score.

The biggest difference is that, in contrast to the California results, we do not find evidence of a nonlinear relationship between test scores and the student teacher ratio for the Massachusetts data since the corresponding F -tests for model (4) do not reject.

As pointed out in the book, the test scores for California and Massachusetts have different units because the underlying tests are different. Thus the estimated coefficients on the student-teacher ratio in both regressions cannot be compared before standardizing test scores to the same units as

$$\frac{\overline{Testscore} - \overline{TestScore}}{\sigma_{TestScore}}$$

for all observations in both data sets and running the regressions of interest using the standardized data again. One can show that the coefficient on student-teacher ratio in the regression using standardized test scores is the coefficient of the original regression divided by the standard deviation of test scores.

For model (3) of the Massachusetts data, the estimated coefficient on the student-teacher ratio is -0.64 . A reduction of the student-teacher ratio by two students is predicted to increase test scores by $-2 \cdot (-0.64) = 1.28$ points. Thus we can compute the effect of a reduction of student-teacher ratio by two students on the standardized test scores as follows:

```
TSMa_mod3$coefficients[2] / sd(MASchools$score) * (-2)
#>          STR
#> 0.08474001
```

For Massachusetts, the predicted increase of test scores due to a reduction of the student-teacher ratio by two students is 0.085 standard deviations of the distribution of the observed distribution of test scores.

Using the linear specification (2) for California, the estimated coefficient on the student-teacher ratio is -0.73 so the predicted increase of test scores induced by a reduction of the student-teacher ratio by two students is $-0.73 \cdot (-2) = 1.46$. We use R to compute the predicted change in standard deviation units:

```
TestScore_mod2$coefficients[2] / sd(CASchools$score) * (-2)
#>          STR
#> 0.07708103
```

This shows that the predicted increase of test scores due to a reduction of the student-teacher ratio by two students is 0.077 standard deviation of the observed distribution of test scores for the California data.

In terms of standardized test scores, the predicted change is essentially the same for school districts in California and Massachusetts.

Altogether, the results support external validity of the inferences made using data on Californian elementary school districts — at least for Massachusetts.

Internal Validity of the Study

External validity of the study *does not* ensure their internal validity. Although the chosen model specification improves upon a simple linear regression model, internal validity may still be violated due to some of the threats listed in Key Concept 9.7. These threats are:

- omitted variable bias
- misspecification of functional form
- errors in variables
- sample selection issues
- simultaneous causality
- heteroskedasticity
- correlation of errors across observations

Consult the book for an in-depth discussion of these threats in view of both test score studies.

Summary

We have found that there is a small but statistically significant effect of the student-teacher ratio on test scores. However, it remains unclear if we have indeed estimated the causal effect of interest since — despite that our approach including control variables, taking into account nonlinearities in the population regression function and statistical inference using robust standard errors — the results might still be biased for example if there are omitted factors which we have not considered. Thus *internal validity* of the study remains questionable. As we have concluded from comparison with the analysis of the Massachusetts data set, this result may be *externally valid*.

The following chapters address techniques that can be remedies to all the threats to internal validity listed in Key Concept 9.7 if multiple regression alone is insufficient. This includes regression using panel data and approaches that employ instrumental variables.

9.5 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 10

Regression with Panel Data

Regression using panel data may mitigate omitted variable bias when there is no information on variables that correlate with both the regressors of interest and the dependent variable and if these variables are constant in the time dimension or across entities. When panel data is available, panel regression methods can be used to improve upon multiple regression models. This is because multiple regression models may produce results that lack internal validity in such a setting, as discussed in Chapter 9.

This chapter covers the following topics:

- notation for panel data,
- fixed effects regression using time and/or entity fixed effects,
- computation of standard errors in fixed effects regression models.

Following the book, for applications we make use of the dataset `Fatalities` from the `AER` package (Kleiber and Zeileis, 2008b) which is a panel dataset reporting annual state level observations on U.S. traffic fatalities for the period 1982 through 1988. The applications analyze if there are effects of alcohol taxes and drunk driving laws on road fatalities and, if present, *how strong* these effects are.

We introduce `plm()`, a convenient R function that enables us to estimate linear panel regression models which comes with the package `plm` (Croissant et al., 2023). Usage of `plm()` is very similar as for the function `lm()` which we have used throughout the previous chapters for estimation of simple and multiple regression models.

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter on your computer:

- `AER`
- `plm`
- `stargazer`

Check whether the following code chunk runs without any errors.

```
library(AER)
library(plm)
library(stargazer)
```

10.1 Panel Data

Key Concept 10.1 Notation for Panel Data

In contrast to cross-section data where we have observations on n subjects (entities), panel data has observations on n entities at $T \geq 2$ time periods. This is denoted as

$$(X_{it}, Y_{it}), i = 1, \dots, n \quad \text{and} \quad t = 1, \dots, T$$

where the index i refers to the entity while t refers to the time period.

Sometimes panel data is also called longitudinal data as it adds a temporal dimension to cross-sectional data. Let us have a look at the dataset `Fatalities` by checking its structure and listing the first few observations.

```
# load the packages and the dataset
library(AER)
library(plm)
data(Fatalities)
# pdata.frame() declares the data as panel data.
Fatalities<- pdata.frame(Fatalities, index = c("state", "year"))

# obtain the dimension and inspect the structure
is.data.frame(Fatalities)
#> [1] TRUE
dim(Fatalities)
#> [1] 336 34

str(Fatalities)
#> Classes 'pdata.frame' and 'data.frame': 336 obs. of 34 variables:
#> $ state      : Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ year       : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ spirits    : 'pseries' Named num 1.37 1.36 1.32 1.28 1.23 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ unemp     : 'pseries' Named num 14.4 13.7 11.1 8.9 9.8 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ income    : 'pseries' Named num 10544 10733 11109 11333 11662 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
```

```

#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ emppop : 'pseries' Named num 50.7 52.1 54.2 55.3 56.5 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ beertax : 'pseries' Named num 1.54 1.79 1.71 1.65 1.61 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ baptist : 'pseries' Named num 30.4 30.3 30.3 30.3 30.3 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ mormon : 'pseries' Named num 0.328 0.343 0.359 0.376 0.393 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ drinkage : 'pseries' Named num 19 19 19 19 19 19 21 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ dry : 'pseries' Named num 25 23 24 23.6 23.5 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ youngdrivers: 'pseries' Named num 0.212 0.211 0.211 0.211 0.213 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ miles : 'pseries' Named num 7234 7836 8263 8727 8953 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ breath : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ jail : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:

```

```

#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ service   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ fatal     : 'pseries' Named int 839 930 932 882 1081 1110 1023 724 675 869 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ nfatal    : 'pseries' Named int 146 154 165 146 172 181 139 131 112 149 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ sfatal    : 'pseries' Named int 99 98 94 98 119 114 89 76 60 81 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ fatal1517 : 'pseries' Named int 53 71 49 66 82 94 66 40 40 51 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ nfatal1517 : 'pseries' Named int 9 8 7 9 10 11 8 7 7 8 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ fatal1820 : 'pseries' Named int 99 108 103 100 120 127 105 81 83 118 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ nfatal1820 : 'pseries' Named int 34 26 25 23 23 31 24 16 19 34 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ fatal2124 : 'pseries' Named int 120 124 118 114 119 138 123 96 80 123 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ nfatal2124 : 'pseries' Named int 32 35 34 45 29 30 25 36 17 33 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...

```

```

#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ afatal      : 'pseries' Named num 309 342 305 277 361 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ pop        : 'pseries' Named num 3942002 3960008 3988992 4021008 4049994 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ pop1517    : 'pseries' Named num 209000 202000 197000 195000 204000 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ pop1820    : 'pseries' Named num 221553 219125 216724 214349 212000 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ pop2124    : 'pseries' Named num 290000 290000 288000 284000 263000 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ milestot   : 'pseries' Named num 28516 31032 32961 35091 36259 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ unempus    : 'pseries' Named num 9.7 9.6 7.5 7.2 7 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ emppopus   : 'pseries' Named num 57.8 57.9 59.5 60.1 60.7 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> $ gsp        : 'pseries' Named num -0.0221 0.0466 0.0628 0.0275 0.0321 ...
#> ..- attr(*, "names")= chr [1:336] "al-1982" "al-1983" "al-1984" "al-1985" ...
#> ..- attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> .. ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> .. ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...
#> - attr(*, "index")=Classes 'pindex' and 'data.frame': 336 obs. of 2 variables:
#> ..$ state: Factor w/ 48 levels "al","az","ar",...: 1 1 1 1 1 1 1 2 2 2 ...
#> ..$ year : Factor w/ 7 levels "1982","1983",...: 1 2 3 4 5 6 7 1 2 3 ...

```

```

# list the first few observations
head(Fatalities)
#>      state year spirits unemp  income  emppop  beertax  baptist  mormon
#> al-1982  al 1982   1.37  14.4 10544.15 50.69204 1.539379 30.3557 0.32829
#> al-1983  al 1983   1.36  13.7 10732.80 52.14703 1.788991 30.3336 0.34341
#> al-1984  al 1984   1.32  11.1 11108.79 54.16809 1.714286 30.3115 0.35924
#> al-1985  al 1985   1.28   8.9 11332.63 55.27114 1.652542 30.2895 0.37579
#> al-1986  al 1986   1.23   9.8 11661.51 56.51450 1.609907 30.2674 0.39311
#> al-1987  al 1987   1.18   7.8 11944.00 57.50988 1.560000 30.2453 0.41123
#>      drinkage  dry youngdrivers  miles breath jail service fatal nfatal
#> al-1982  19.00 25.0063  0.211572 7233.887  no  no  no  839  146
#> al-1983  19.00 22.9942  0.210768 7836.348  no  no  no  930  154
#> al-1984  19.00 24.0426  0.211484 8262.990  no  no  no  932  165
#> al-1985  19.67 23.6339  0.211140 8726.917  no  no  no  882  146
#> al-1986  21.00 23.4647  0.213400 8952.854  no  no  no 1081  172
#> al-1987  21.00 23.7924  0.215527 9166.302  no  no  no 1110  181
#>      sfatal fatal1517 nfatal1517 fatal1820 nfatal1820 fatal2124 nfatal2124
#> al-1982  99 53 9 99 34 120 32
#> al-1983  98 71 8 108 26 124 35
#> al-1984  94 49 7 103 25 118 34
#> al-1985  98 66 9 100 23 114 45
#> al-1986 119 82 10 120 23 119 29
#> al-1987 114 94 11 127 31 138 30
#>      afatal  pop  pop1517  pop1820  pop2124  milestot  unempus  emppopus
#> al-1982 309.438 3942002 208999.6 221553.4 290000.1 28516 9.7 57.8
#> al-1983 341.834 3960008 202000.1 219125.5 290000.2 31032 9.6 57.9
#> al-1984 304.872 3988992 197000.0 216724.1 288000.2 32961 7.5 59.5
#> al-1985 276.742 4021008 194999.7 214349.0 284000.3 35091 7.2 60.1
#> al-1986 360.716 4049994 203999.9 212000.0 263000.3 36259 7.0 60.7
#> al-1987 368.421 4082999 204999.8 208998.5 258999.8 37426 6.2 61.5
#>      gsp
#> al-1982 -0.02212476
#> al-1983 0.04655825
#> al-1984 0.06279784
#> al-1985 0.02748997
#> al-1986 0.03214295
#> al-1987 0.04897637

# summarize the variables 'state' and 'year'
summary(Fatalities[, c(1, 2)])
#>      state      year
#> al      : 7  1982:48
#> az      : 7  1983:48
#> ar      : 7  1984:48
#> ca      : 7  1985:48
#> co      : 7  1986:48
#> ct      : 7  1987:48
#> (Other):294 1988:48

```

We find that the dataset consists of 336 observations on 34 variables. Notice that the variable `state` is a factor variable with 48 levels (one for each of the 48 contiguous federal states of the U.S.). The variable `year` is also a factor variable that has 7 levels identifying the time period when the observation

was made. This gives us $7 \times 48 = 336$ observations in total. Since all variables are observed for all entities and over all time periods, the panel is *balanced*. If there were missing data for at least one entity in at least one time period we would call the panel *unbalanced*.

Example: Traffic Deaths and Alcohol Taxes

We start by reproducing Figure 10.1 of the book. To this end we estimate simple regressions using data for years 1982 and 1988 that model the relationship between beer tax (adjusted for 1988 dollars) and the traffic fatality rate, measured as the number of fatalities per 10000 inhabitants. Afterwards, we plot the data and add the corresponding estimated regression functions.

```
# define the fatality rate
Fatalities$fatal_rate <- Fatalities$fatal / Fatalities$pop * 10000

# subset the data
Fatalities1982 <- subset(Fatalities, year == "1982")
Fatalities1988 <- subset(Fatalities, year == "1988")

# estimate simple regression models using 1982 and 1988 data
fatal1982_mod <- lm(fatal_rate ~ beertax, data = Fatalities1982)
fatal1988_mod <- lm(fatal_rate ~ beertax, data = Fatalities1988)

coeftest(fatal1982_mod, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.01038    0.14957 13.4408 <2e-16 ***
#> beertax      0.14846    0.13261  1.1196  0.2687
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coeftest(fatal1988_mod, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.85907    0.11461 16.2205 < 2.2e-16 ***
#> beertax      0.43875    0.12786  3.4314  0.001279 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression functions are

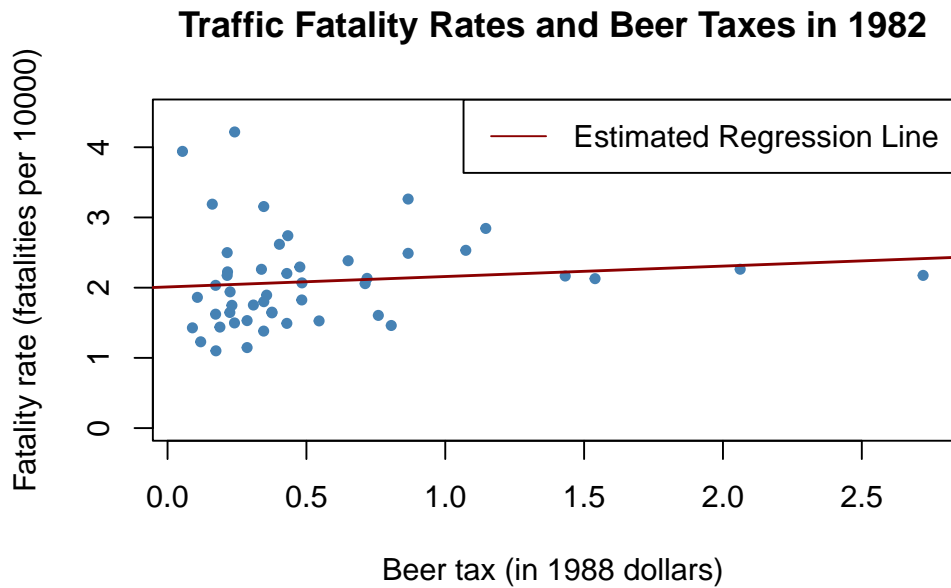
$$\widehat{FatalityRate} = \underset{(0.15)}{2.01} + \underset{(0.13)}{0.15} \times BeerTax \quad (1982 \text{ data}),$$

$$\widehat{FatalityRate} = \underset{(0.11)}{1.86} + \underset{(0.13)}{0.44} \times BeerTax \quad (1988 \text{ data}).$$

```
# plot the observations and add the estimated regression line for 1982 data
plot(x = as.double(Fatalities1982$beertax),
     y = as.double(Fatalities1982$fatal_rate),
     xlab = "Beer tax (in 1988 dollars)",
     ylab = "Fatality rate (fatalities per 10000)",
     main = "Traffic Fatality Rates and Beer Taxes in 1982",
```

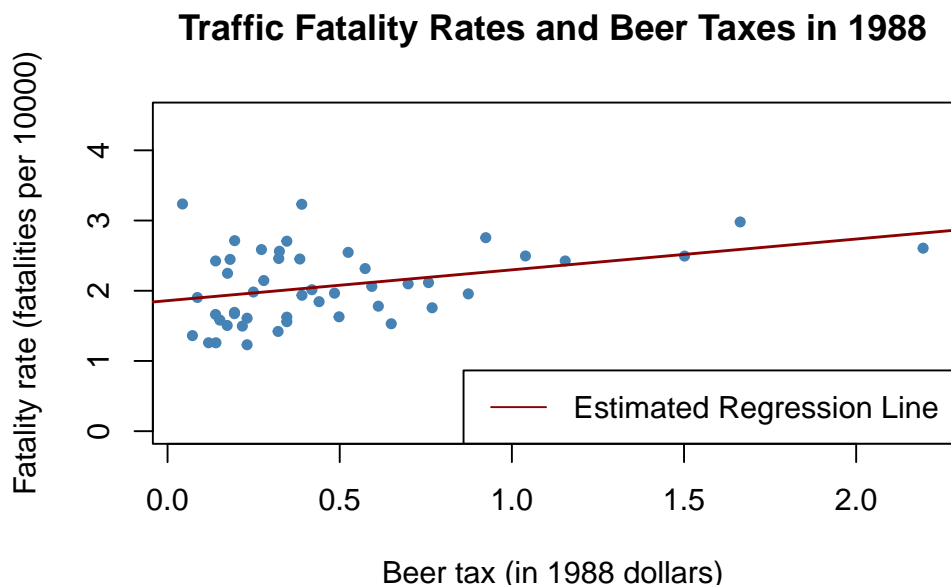
```
ylim = c(0, 4.5),
pch = 20,
col = "steelblue")
```

```
abline(fatal1982_mod, lwd = 1.5, col="darkred")
legend("topright", lty=1, col="darkred", "Estimated Regression Line")
```



```
# plot observations and add estimated regression line for 1988 data
plot(x = as.double(Fatalities1988$beertax),
     y = as.double(Fatalities1988$fatal_rate),
     xlab = "Beer tax (in 1988 dollars)",
     ylab = "Fatality rate (fatalities per 10000)",
     main = "Traffic Fatality Rates and Beer Taxes in 1988",
     ylim = c(0, 4.5),
     pch = 20,
     col = "steelblue")
```

```
abline(fatal1988_mod, lwd = 1.5, col="darkred")
legend("bottomright", lty=1, col="darkred", "Estimated Regression Line")
```

In both plots, each point represents observations of beer tax and fatality rate for a given state in the respective year. The regression results indicate a positive relationship between the beer tax and the fatality rate for both years. The estimated coefficient on beer tax for the 1988 data is almost three times as large as for the 1982 dataset. This is contrary to our expectations: alcohol taxes are supposed to *lower* the rate of traffic fatalities. As we know from Chapter 6, this is possibly due to omitted variable bias, since both models do not include any covariates, e.g., economic conditions. This could be corrected by using a multiple regression approach. However, this cannot account for omitted *unobservable* factors that differ from state to state but can be assumed to be constant over the observation span, e.g., the populations' attitude towards drunk driving. As shown in the next section, panel data allow us to hold such factors constant.

10.2 Panel Data with Two Time Periods: “Before and After” Comparisons

Suppose there are only $T = 2$ time periods $t = 1982, 1988$. This allows us to analyze differences in changes of the fatality rate from year 1982 to 1988. We start by considering the population regression model

$$FatalityRate_{it} = \beta_0 + \beta_1 BeerTax_{it} + \beta_2 Z_i + u_{it}$$

where the Z_i are state specific characteristics that differ between states but are *constant over time*. For $t = 1982$ and $t = 1988$ we have

$$FatalityRate_{i1982} = \beta_0 + \beta_1 BeerTax_{i1982} + \beta_2 Z_i + u_{i1982},$$

$$FatalityRate_{i1988} = \beta_0 + \beta_1 BeerTax_{i1988} + \beta_2 Z_i + u_{i1988}.$$

We can eliminate the Z_i by regressing the difference in the fatality rate between 1988 and 1982 on the difference in beer tax between those years:

$$FatalityRate_{i1988} - FatalityRate_{i1982} = \beta_1 (BeerTax_{i1988} - BeerTax_{i1982}) + u_{i1988} - u_{i1982}.$$

This regression model, where the difference in fatality rate between 1988 and 1982 is regressed on the difference in beer tax between those years, yields an estimate for β_1 that is robust to a possible bias due to omission of Z_i , as these influences are eliminated from the model. Next we will use R to estimate a regression based on the differenced data and to plot the estimated regression function.

```

# compute the differences
diff_fatal_rate <- Fatalities1988$fatal_rate - Fatalities1982$fatal_rate
diff_beertax <- Fatalities1988$beertax - Fatalities1982$beertax

# estimate a regression using differenced data
fatal_diff_mod <- lm(diff_fatal_rate ~ diff_beertax)

coeftest(fatal_diff_mod, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.072037  0.065355 -1.1022 0.276091
#> diff_beertax -1.040973  0.355006 -2.9323 0.005229 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Including the intercept allows for a change in the mean fatality rate in the time between 1982 and 1988 in the absence of a change in the beer tax.

We obtain the OLS estimated regression function

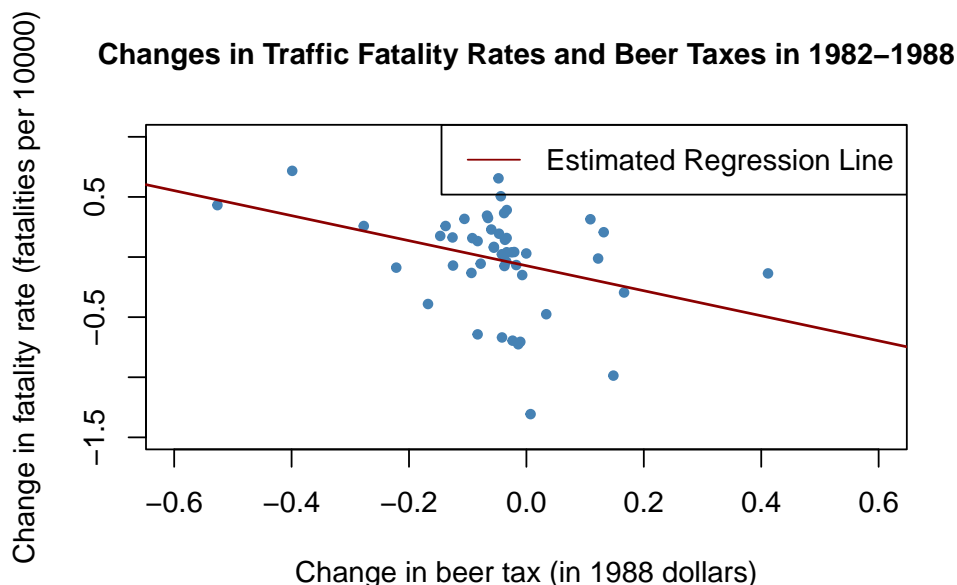
$$\widehat{FatalityRate}_{i1988} - \widehat{FatalityRate}_{i1982} = \underset{(0.065)}{-0.072} - \underset{(0.36)}{1.04} \times (BeerTax_{i1988} - BeerTax_{i1982}).$$

```

# plot the differenced data
plot(x = as.double(diff_beertax),
     y = as.double(diff_fatal_rate),
     xlab = "Change in beer tax (in 1988 dollars)",
     ylab = "Change in fatality rate (fatalities per 10000)",
     main = "Changes in Traffic Fatality Rates and Beer Taxes in 1982-1988",
     cex.main=1,
     xlim = c(-0.6, 0.6),
     ylim = c(-1.5, 1),
     pch = 20,
     col = "steelblue")

# add the regression line to plot
abline(fatal_diff_mod, lwd = 1.5, col="darkred")
#add legend
legend("topright", lty=1, col="darkred", "Estimated Regression Line")

```



The estimated coefficient on beer tax is now negative and significantly different from zero at 5%. Its interpretation is that raising the beer tax by \$1 causes traffic fatalities to decrease by 1.04 per 10000 people. This is rather large as the average fatality rate is approximately 2 persons per 10000 people.

```
# compute mean fatality rate over all states for all time periods
mean(Fatalities$fatal_rate)
#> [1] 2.040444
```

Once more this outcome is likely to be a consequence of omitting factors in the single-year regression that influence the fatality rate and are correlated with the beer tax *and* change over time. The message is that we need to be more careful and control for such factors before drawing conclusions about the effect of a raise in beer taxes.

The approach presented in this section discards information for years 1983 to 1987. The fixed effects method that allows us to use data for more than $T = 2$ time periods and enables us to add control variables to the analysis.

10.3 Fixed Effects Regression

Consider the panel regression model

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \beta_2 Z_i + u_{it}$$

where the Z_i are unobserved time-invariant heterogeneities across the entities $i = 1, \dots, n$. We aim to estimate β_1 , the effect on Y_i of a change in X_i holding constant Z_i . Letting $\alpha_i = \beta_0 + \beta_2 Z_i$ we obtain the model

$$Y_{it} = \alpha_i + \beta_1 X_{it} + u_{it}. \quad (10.1)$$

Having individual specific intercepts α_i , $i = 1, \dots, n$, where each of these can be understood as the fixed effect of entity i , this model is called the *fixed effects model*. The variation in the α_i , $i = 1, \dots, n$ comes from the Z_i . (10.1) can be rewritten as a regression model containing $n - 1$ dummy regressors and a constant:

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \gamma_2 D2_i + \gamma_3 D3_i + \dots + \gamma_n Dn_i + u_{it}. \quad (10.2)$$

Model (10.2) has n different intercepts — one for every entity. (10.1) and (10.2) are equivalent representations of the fixed effects model (Note: β_0 is intercept of the fixed effect model in equation 10.2).

The fixed effects model can be generalized to contain more than just one determinant of Y that is correlated with X and changes over time. Key Concept 10.2 presents the generalized fixed effects regression model.

Key Concept 10.2 The Fixed Effects Regression Model

The fixed effects regression model is

$$Y_{it} = \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \alpha_i + u_{it} \quad (10.3)$$

with $i = 1, \dots, n$ and $t = 1, \dots, T$. The α_i are entity-specific intercepts that capture heterogeneities across entities. An equivalent representation of this model is given by

$$Y_{it} = \beta_0 + \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \gamma_2 D2_i + \gamma_3 D3_i + \cdots + \gamma_n Dn_i + u_{it} \quad (10.4)$$

where the $D2_i, D3_i, \dots, Dn_i$ are dummy variables.

Estimation and Inference

Software packages use a so-called “entity-demeaned” OLS algorithm which is computationally more efficient than estimating regression models with $k + n$ regressors as needed for models (10.3) and (10.4).

Taking averages across time on both sides of (10.1) we obtain

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^T Y_{it} &= \beta_1 \frac{1}{T} \sum_{t=1}^T X_{it} + \alpha_i + \frac{1}{T} \sum_{t=1}^T u_{it} \\ \bar{Y}_i &= \beta_1 \bar{X}_i + \alpha_i + \bar{u}_i. \end{aligned}$$

Subtraction from (10.1) yields

$$\begin{aligned} Y_{it} - \bar{Y}_i &= \beta_1 (X_{it} - \bar{X}_i) + (u_{it} - \bar{u}_i) \\ \tilde{Y}_{it} &= \beta_1 \tilde{X}_{it} + \tilde{u}_{it}. \end{aligned} \quad (10.5)$$

In this model, the OLS estimate of the parameter of interest β_1 is equal to the estimate obtained using (10.2) — without the need to estimate $n - 1$ dummies and an intercept.

We conclude that there are two ways of estimating β_1 in the fixed effects regression:

1. OLS of the dummy regression model as shown in (10.2).
2. OLS using the entity demeaned data as in (10.5).

Provided that the fixed effects regression assumptions stated in Key Concept 10.3 hold, the sampling distribution of the OLS estimator in the fixed effects regression model is normal in large samples. The variance of the estimates can be estimated and we can compute standard errors, t -statistics and confidence intervals for coefficients. In the next section, we see how to estimate a fixed effects model using R and how to obtain a model summary that reports heteroskedasticity-robust standard errors. We leave aside complicated formulas of the estimators. See Chapter 10.5 and Appendix 10.2 of the book for a discussion of theoretical aspects.

Application to Traffic Deaths

Following Key Concept 10.2, the simple fixed effects model for estimation of the relation between traffic fatality rates and the beer taxes is

$$FatalityRate_{it} = \beta_1 BeerTax_{it} + StateFixedEffects + u_{it}, \quad (10.6)$$

a regression of the traffic fatality rate on beer tax and 48 binary regressors — one for each federal state.

We can simply use the function `lm()` to obtain an estimate of β_1 .

```
fatal_fe_lm_mod <- lm(fatal_rate ~ beertax + state - 1, data = Fatalities)
fatal_fe_lm_mod
#>
#> Call:
#> lm(formula = fatal_rate ~ beertax + state - 1, data = Fatalities)
#>
#> Coefficients:
#> beertax  stateal  stateaz  statear  stateca  stateco  statect  statede
#> -0.6559  3.4776  2.9099  2.8227  1.9682  1.9933  1.6154  2.1700
#> statefl  statega  stateid  stateil  statein  stateia  stateks  stateky
#>  3.2095  4.0022  2.8086  1.5160  2.0161  1.9337  2.2544  2.2601
#> statela  stateme  statemd  statema  statemi  statemn  statems  statemo
#>  2.6305  2.3697  1.7712  1.3679  1.9931  1.5804  3.4486  2.1814
#> statemt  statene  statenv  statenh  statenj  statenm  stateny  statenc
#>  3.1172  1.9555  2.8769  2.2232  1.3719  3.9040  1.2910  3.1872
#> statend  stateoh  stateok  stateor  statepa  stateri  statesc  statesd
#>  1.8542  1.8032  2.9326  2.3096  1.7102  1.2126  4.0348  2.4739
#> statetn  statetx  stateut  statevt  stateva  statewa  statewv  statewi
#>  2.6020  2.5602  2.3137  2.5116  2.1874  1.8181  2.5809  1.7184
#> statewy
#>  3.2491
```

As discussed in the previous section, it is also possible to estimate β_1 by applying OLS to the demeaned data, that is, to run the regression

$$\tilde{FatalityRate} = \beta_1 \tilde{BeerTax}_{it} + u_{it}.$$

```
# obtain demeaned data
fatal_demeaned <- with(Fatalities,
  data.frame(fatal_rate = fatal_rate - ave(fatal_rate, state),
    beertax = beertax - ave(beertax, state)))

# estimate the regression
summary(lm(fatal_rate ~ beertax - 1, data = fatal_demeaned))
```

The function `ave` is convenient for computing group averages. We use it to obtain state specific averages of the fatality rate and the beer tax.

Alternatively one may use `plm()` from the package with the same name.

```
# install and load the 'plm' package
## install.packages("plm")
library(plm)
```

As for `lm()` we have to specify the regression formula and the data to be used in our call of `plm()`. Additionally, it is required to pass a vector of names of entity and time ID variables to the argument `index`. For `Fatalities`, the ID variable for entities is named `state` and the time id variable is `year`. Since the fixed effects estimator is also called the *within estimator*, we set `model = "within"`. Finally, the function `coefstest()` allows to obtain inference based on robust standard errors.

```
# estimate the fixed effects regression with plm()
fatal_fe_mod <- plm(fatal_rate ~ beertax,
  data = Fatalities,
  index = c("state", "year"),
  model = "within")

coefstest(fatal_fe_mod, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>      Estimate Std. Error t value Pr(>|t|)
#> beertax -0.65587    0.28880  -2.271  0.02388 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated coefficient is again -0.6559 . Note that `plm()` uses the entity-demeaned OLS algorithm and thus does not report dummy coefficients. The estimated regression function is

$$\widehat{FatalityRate} = -0.66 \underset{(0.29)}{\times} BeerTax + StateFixedEffects. \quad (10.7)$$

The coefficient on *BeerTax* is negative and significant. The interpretation is that the estimated reduction in traffic fatalities due to an increase in the real beer tax by \$1 is 0.66 per 10000 people, which is still pretty high. Although including state fixed effects eliminates the risk of a bias due to omitted factors that vary across states but not over time, we suspect that there are other omitted variables that vary over time and thus cause a bias.

10.4 Regression with Time Fixed Effects

Controlling for variables that are constant across entities but vary over time can be done by including time fixed effects. If there are *only* time fixed effects, the fixed effects regression model becomes

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \delta_2 B2_t + \cdots + \delta_T B T_t + u_{it},$$

where only $T - 1$ dummies are included ($B1$ is omitted) since the model includes an intercept. This model eliminates omitted variable bias caused by excluding unobserved variables that evolve over time but are constant across entities.

In some applications it is meaningful to include both entity and time fixed effects. The *entity and time fixed effects model* is

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \gamma_2 D2_i + \cdots + \gamma_n D T_i + \delta_2 B2_t + \cdots + \delta_T B T_t + u_{it}.$$

The combined model allows to eliminate bias from unobservables that change over time but are constant over entities and it controls for factors that differ across entities but are constant over time. Such models can be estimated using the OLS algorithm that is implemented in R.

The following code chunk shows how to estimate the combined entity and time fixed effects model of the relation between fatalities and beer tax:

$$FatalityRate_{it} = \beta_1 BeerTax_{it} + StateEffects + TimeFixedEffects + u_{it},$$

using both `lm()` and `plm()`. It is straightforward to estimate this regression with `lm()` since it is just an extension of (10.6) so we only have to adjust the `formula` argument by adding the additional regressor `year` for time fixed effects. In our call of `plm()` we set another argument `effect = "twoways"` for inclusion of entity *and* time dummies.

```
# estimate a combined time and entity fixed effects regression model

# via lm()
fatal_tefe_lm_mod <- lm(fatal_rate ~ beertax + state + year - 1, data = Fatalities)
fatal_tefe_lm_mod
#>
#> Call:
#> lm(formula = fatal_rate ~ beertax + state + year - 1, data = Fatalities)
#>
#> Coefficients:
#> beertax stateal stateaz statear stateca stateco statect statede
#> -0.63998 3.51137 2.96451 2.87284 2.02618 2.04984 1.67125 2.22711
#> statefl statega stateid stateil statein stateia stateks stateky
#> 3.25132 4.02300 2.86242 1.57287 2.07123 1.98709 2.30707 2.31659
#> statela stateme statemd statema statemi statemm statems statemo
#> 2.67772 2.41713 1.82731 1.42335 2.04488 1.63488 3.49146 2.23598
#> statemt statene statenv statenh statenj statenm stateny statenc
#> 3.17160 2.00846 2.93322 2.27245 1.43016 3.95748 1.34849 3.22630
#> statend stateoh stateok stateor statepa stateri statesc statesd
#> 1.90762 1.85664 2.97776 2.36597 1.76563 1.26964 4.06496 2.52317
#> statetn statetx stateut statevt stateva statewa statewv statewi
#> 2.65670 2.61282 2.36165 2.56100 2.23618 1.87424 2.63364 1.77545
#> statewy year1983 year1984 year1985 year1986 year1987 year1988
#> 3.30791 -0.07990 -0.07242 -0.12398 -0.03786 -0.05090 -0.05180

# via plm()
fatal_tefe_mod <- plm(fatal_rate ~ beertax,
                     data = Fatalities,
                     index = c("state", "year"),
                     model = "within",
                     effect = "twoways")

coeftest(fatal_tefe_mod, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#> Estimate Std. Error t value Pr(>|t|)
#> beertax -0.63998 0.35015 -1.8277 0.06865 .
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Before discussing the outcomes we convince ourselves that `state` and `year` are of the class `factor` .

```
# check the class of 'state' and 'year'
class(Fatalities$state)
#> [1] "pseries" "factor"
class(Fatalities$year)
#> [1] "pseries" "factor"
```

The `lm()` functions converts factors into dummies automatically. Since we exclude the intercept by adding `-1` to the right-hand side of the regression formula, `lm()` estimates coefficients for $n + (T - 1) = 48 + 6 = 54$ binary variables (6 year dummies and 48 state dummies). Again, `p1m()` only reports the estimated coefficient on *BeerTax*.

The estimated regression function is

$$\widehat{FatalityRate} = -0.64 \times BeerTax + StateEffects + TimeFixedEffects. \quad (10.8)$$

(0.35)

The result -0.66 is close to the estimated coefficient for the regression model including only entity fixed effects. Unsurprisingly, the coefficient is less precisely estimated but significantly different from zero at 10%.

In view of (10.7) and (10.8) we conclude that the estimated relationship between traffic fatalities and the real beer tax is not affected by omitted variable bias due to factors that are constant either over time or across states..

10.5 The Fixed Effects Regression Assumptions and Standard Errors for Fixed Effects Regression

This section focuses on the entity fixed effects model and presents model assumptions that need to hold in order for OLS to produce unbiased estimates that are normally distributed in large samples. These assumptions are an extension of the assumptions made for the multiple regression model (see Key Concept 6.4) and are given in Key Concept 10.3. We also briefly discuss standard errors in fixed effects models which differ from standard errors in multiple regression as the regression error can exhibit serial correlation in panel models.

Key Concept 10.3 The Fixed Effects Regression Assumptions

In the fixed effects regression model

$$Y_{it} = \beta_1 X_{it} + \alpha_i + u_{it} \quad , \quad i = 1, \dots, n, \quad t = 1, \dots, T,$$

we assume the following:

1. The error term u_{it} has conditional mean zero, that is, $E(u_{it} | X_{i1}, X_{i2}, \dots, X_{iT})$.
2. $(X_{i1}, X_{i2}, \dots, X_{iT}, u_{i1}, \dots, u_{iT})$, $i = 1, \dots, n$ are i.i.d. draws from their joint distribution.
3. Large outliers are unlikely, i.e., (X_{it}, u_{it}) have nonzero finite fourth moments.
4. There is no perfect multicollinearity.

When there are multiple regressors, X_{it} is replaced by $X_{1,it}, X_{2,it}, \dots, X_{k,it}$.

The first assumption is that the error is uncorrelated with *all* observations of the variable X for the entity i over time. If this assumption is violated, we face omitted variables bias. The second assumption

ensures that variables are i.i.d. *across* entities $i = 1, \dots, n$. This does not require the observations to be uncorrelated *within* an entity. The X_{it} are allowed to be *autocorrelated* within entities. This is a common property of time series data. The same is allowed for errors u_{it} . Consult Chapter 10.5 (Stock and Watson) of the book for a detailed explanation for why autocorrelation is plausible in panel applications. The second assumption is justified if the entities are selected by simple random sampling. The third and fourth assumptions are analogous to the multiple regression assumptions made in Key Concept 6.4.

Standard Errors for Fixed Effects Regression

Similar as for heteroskedasticity, autocorrelation invalidates the usual standard error formulas as well as heteroskedasticity-robust standard errors since these are derived under the assumption that there is no autocorrelation. When there is both heteroskedasticity *and* autocorrelation, the so-called *heteroskedasticity and autocorrelation-consistent (HAC) standard errors* need to be used. *Clustered standard errors* belong to these type of standard errors. They allow for heteroskedasticity and autocorrelated errors within an entity but *not* correlation across entities.

As shown in the examples throughout this chapter, it is fairly easy to specify usage of clustered standard errors in regression summaries produced by functions like `coefTest()` in conjunction with `vcovHC()` from the package `sandwich`. Conveniently, `vcovHC()` recognizes panel model objects (objects of class `plm`) and computes clustered standard errors by default.

The regressions conducted in this chapter are good examples for why usage of clustered standard errors is crucial in empirical applications of fixed effects models. For example, consider the entity and time fixed effects model for fatalities. Since `fatal_tefe_lm_mod` is an object of class `lm`, `coefTest()` does not compute clustered standard errors but uses robust standard errors that are only valid in the absence of autocorrelated errors.

```
# check class of the model object
class(fatal_tefe_lm_mod)
#> [1] "lm"

# obtain a summary based on heteroskedasticity-robust standard errors
# (no adjustment for heteroskedasticity only)
coefTest(fatal_tefe_lm_mod, vcov = vcovHC, type = "HC1")[1, ]
#> Estimate Std. Error t value Pr(>|t|)
#> -0.6399800 0.2547149 -2.5125346 0.0125470

# check class of the (plm) model object
class(fatal_tefe_mod)
#> [1] "plm" "panelmodel"

# obtain a summary based on clustered standard errors
# (adjustment for autocorrelation + heteroskedasticity)
coefTest(fatal_tefe_mod, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#> Estimate Std. Error t value Pr(>|t|)
#> beertax -0.63998 0.35015 -1.8277 0.06865 .
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The outcomes differ rather strongly: imposing no autocorrelation we obtain a standard error of 0.25

which implies significance of $\hat{\beta}_1$, the coefficient on *BeerTax* at the level of 5%. On the contrary, using the clustered standard error 0.35 results in a failure to reject the null hypothesis $H_0 : \beta_1 = 0$ at the same level, see equation (10.8). Consult Appendix 10.2 of the book (Stock and Watson) for insights on the computation of clustered standard errors.

10.6 Drunk Driving Laws and Traffic Deaths

There are two major sources of omitted variable bias that are not accounted for by all of the models of the relation between traffic fatalities and beer taxes that we have considered so far: economic conditions and driving laws. Fortunately, *Fatalities* has data on state-specific legal drinking age (*drinkage*), punishment (*jail*, *service*) and various economic indicators like unemployment rate (*unemp*) and per capita income (*income*). We may use these covariates to extend the preceding analysis.

These covariates are defined as follows:

- *unemp*: a numeric variable stating the state specific unemployment rate.
- $\log(\text{income})$: the logarithm of real per capita income (in 1988 dollars).
- *miles*: the state average miles per driver.
- *drinkage*: the state specific minimum legal drinking age.
- *drinkagec*: a discretized version of *drinkage* that classifies states into four categories of minimal drinking age; 18, 19, 20, 21 and older. R denotes this as [18,19), [19,20), [20,21) and [21,22]. These categories are included as dummy regressors where [21,22] is chosen as the reference category.
- *punish*: a dummy variable with levels *yes* and *no* that measures if drunk driving is severely punished by mandatory jail time or mandatory community service (first conviction).

At first, we define the variables according to the regression results presented in Table 10.1 of the book.

```
# discretize the minimum legal drinking age
Fatalities$drinkagec <- cut(Fatalities$drinkage,
                           breaks = 18:22,
                           include.lowest = TRUE,
                           right = FALSE)

# set minimum drinking age [21, 22] to be the baseline level
Fatalities$drinkagec <- relevel(Fatalities$drinkagec, "[21,22]")

# mandatory jail or community service?
Fatalities$punish <- with(Fatalities, factor(jail == "yes" | service == "yes",
                                             labels = c("no", "yes")))

# the set of observations on all variables for 1982 and 1988
fatal_1982_1988 <- Fatalities[with(Fatalities, year == 1982 | year == 1988), ]
```

Next, we estimate all seven models using `plm()`.

```
# estimate all seven models
fat_mod1 <- lm(fatal_rate ~ beertax, data = Fatalities)

fat_mod2 <- plm(fatal_rate ~ beertax + state, data = Fatalities)

fat_mod3 <- plm(fatal_rate ~ beertax + state + year,
               index = c("state", "year"),
               model = "within",
```

```

        effect = "twoways",
        data = Fatalities)

fat_mod4 <- plm(fatal_rate ~ beertax + state + year + drinkagec
               + punish + miles + unemp + log(income),
               index = c("state", "year"),
               model = "within",
               effect = "twoways",
               data = Fatalities)

fat_mod5 <- plm(fatal_rate ~ beertax + state + year + drinkagec
               + punish + miles,
               index = c("state", "year"),
               model = "within",
               effect = "twoways",
               data = Fatalities)

fat_mod6 <- plm(fatal_rate ~ beertax + year + drinkage
               + punish + miles + unemp + log(income),
               index = c("state", "year"),
               model = "within",
               effect = "twoways",
               data = Fatalities)

fat_mod7 <- plm(fatal_rate ~ beertax + state + year + drinkagec
               + punish + miles + unemp + log(income),
               index = c("state", "year"),
               model = "within",
               effect = "twoways",
               data = fatal_1982_1988)

```

We again use `stargazer()` (Hlavac, 2022) to generate a comprehensive tabular presentation of the results.

```

library(stargazer)

# gather clustered standard errors in a list
rob_se <- list(sqrt(diag(vcovHC(fat_mod1, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod2, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod3, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod4, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod5, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod6, type = "HC1"))),
               sqrt(diag(vcovHC(fat_mod7, type = "HC1"))))

# generate the table
stargazer(fat_mod1,
          fat_mod2,
          fat_mod3,
          fat_mod4,
          fat_mod5,
          fat_mod6,

```

```

fat_mod7,
digits = 3,
header = FALSE,
type = "latex",
se = rob_se,
title = "Linear Panel Regression Models of Traffic Fatalities
        due to Drunk Driving",
model.numbers = FALSE,
column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)", "(7)")

```

While columns (2) and (3) recap the results (10.7) and (10.8), column (1) presents an estimate of the coefficient of interest in the naive OLS regression of the fatality rate on beer tax without any fixed effects. We obtain a *positive* estimate for the coefficient on beer tax that is likely to be upward biased. The model fit is rather bad, too ($\bar{R}^2 = 0.091$). The sign of the estimate changes as we extend the model by both entity and time fixed effects in models (2) and (3). Furthermore \bar{R}^2 increases substantially as fixed effects are included in the model equation. Nonetheless, as discussed before, the magnitudes of both estimates may be too large.

The model specifications (4) to (7) include covariates that shall capture the effect of economic conditions overall state as well as the legal framework. Considering (4) as the baseline specification, we observe four interesting results:

1. Including the covariates does not lead to a major reduction of the estimated effect of the beer tax. The coefficient is not significantly different from zero at the level of 5% as the estimate is rather imprecise.
2. The minimum legal drinking age *does not* have an effect on traffic fatalities: none of the three dummy variables are significantly different from zero at any common level of significance. Moreover, an *F*-Test of the joint hypothesis that all three coefficients are zero does not reject the null hypothesis. The next code chunk shows how to test this hypothesis.

```

# test if legal drinking age has no explanatory power
linearHypothesis(fat_mod4,
                 test = "F",
                 c("drinkagec[18,19]=0", "drinkagec[19,20]=0", "drinkagec[20,21)"),
                 vcov. = vcovHC, type = "HC1")

#> Linear hypothesis test
#>
#> Hypothesis:
#> drinkagec[18,19) = 0
#> drinkagec[19,20) = 0
#> drinkagec[20,21) = 0
#>
#> Model 1: restricted model
#> Model 2: fatal_rate ~ beertax + state + year + drinkagec + punish + miles +
#>      unemp + log(income)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     276
#> 2     273  3 0.3782 0.7688

```

3. There is no evidence that punishment for first offenders has a deterring effects on drunk driving:

Table 10.1: Linear Panel Regression Models of Traffic Fatalities due to Drunk Driving

	<i>Dependent variable:</i>						
	fatal_rate						
	Linear Panel Regression						
OLS	(1)	(2)	(3)	(4)	(5)	(6)	(7)
beertax	0.365*** (0.053)	-0.656** (0.289)	-0.640* (0.350)	-0.445 (0.291)	-0.690** (0.345)	-0.456 (0.301)	-0.926*** (0.337)
drinkagec[18,19)				0.028 (0.068)	-0.010 (0.081)		0.037 (0.101)
drinkagec[19,20)				-0.018 (0.049)	-0.076 (0.066)		-0.065 (0.097)
drinkagec[20,21)				0.032 (0.050)	-0.100* (0.055)		-0.113 (0.123)
drinkage						-0.002 (0.021)	
punishyes				0.038 (0.101)	0.085 (0.109)	0.039 (0.101)	0.089 (0.161)
miles				0.00001 (0.00001)	0.00002* (0.00001)	0.00001 (0.00001)	0.0001*** (0.00005)
unemp				-0.063*** (0.013)		-0.063*** (0.013)	-0.091*** (0.021)
log(income)				1.816*** (0.624)		1.786*** (0.631)	0.996 (0.666)
Constant	1.853*** (0.047)						
Observations	336	336	336	335	335	335	95
R ²	0.093	0.041	0.036	0.360	0.066	0.357	0.659
Adjusted R ²	0.091	-0.120	-0.149	0.217	-0.134	0.219	0.157
Residual Std. Error	0.544 (df = 334)						

Note: *p<0.1; **p<0.05; ***p<0.01

the corresponding coefficient is not significant at the 10% level.

4. The economic variables significantly explain traffic fatalities. We can check that the employment rate and per capita income are jointly significant at the level of 0.1%.

```
# test if economic indicators have no explanatory power
linearHypothesis(fat_mod4,
                 test = "F",
                 c("log(income)", "unemp"),
                 vcov. = vcovHC, type = "HC1")

#> Linear hypothesis test
#>
#> Hypothesis:
#> log(income) = 0
#> unemp = 0
#>
#> Model 1: restricted model
#> Model 2: fatal_rate ~ beertax + state + year + drinkagec + punish + miles +
#>   unemp + log(income)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df    F   Pr(>F)
#> 1     275
#> 2     273  2 31.577 4.609e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model (5) omits the economic factors. The result supports the notion that economic indicators should remain in the model as the coefficient on beer tax is sensitive to the inclusion of the latter.

Results for model (6) demonstrate that the legal drinking age has little explanatory power and that the coefficient of interest is not sensitive to changes in the functional form of the relation between drinking age and traffic fatalities.

Specification (7) reveals that reducing the amount of available information (we only use 95 observations for the period 1982 to 1988 here) inflates standard errors but does not lead to drastic changes in coefficient estimates.

Summary

We have not found evidence that severe punishments and increasing the minimum drinking age reduce traffic fatalities due to drunk driving. Nonetheless, there seems to be a negative effect of alcohol taxes on traffic fatalities which, however, is estimated imprecisely and cannot be interpreted as the causal effect of interest as there still may be a bias. The issue is that there may be omitted variables that differ across states *and* change over time and this bias remains even though we use a panel approach that controls for entity specific and time invariant unobservables.

A powerful method that can be used if common panel regression approaches fail is instrumental variables regression. We will return to this concept in Chapter 12.

10.7 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 11

Regression with a Binary Dependent Variable

This chapter, we discuss a special class of regression models that aim to explain a limited dependent variable. In particular, we consider models where the dependent variable is binary. We will see that in such models, the regression function can be interpreted as a conditional probability function of the binary dependent variable.

We review the following concepts:

- the linear probability model,
- the Probit model,
- the Logit model,
- maximum likelihood estimation of nonlinear regression models.

Of course, we will also see how to estimate above models using R and discuss an application where we examine the question whether there is racial discrimination in the U.S. mortgage market.

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter on your computer:

- `AER` (Kleiber and Zeileis, 2008b),
- `stargazer` (Hlavac, 2022).

Check whether the following code chunk runs without any errors.

```
library(AER)
library(stargazer)
```

11.1 Binary Dependent Variables and the Linear Probability Model

Key Concept 11.1 The Linear Probability Model

The linear regression model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + u_i$$

with a binary dependent variable Y_i is called the linear probability model. In the linear probability model we have

$$E(Y|X_1, X_2, \dots, X_k) = P(Y = 1|X_1, X_2, \dots, X_k)$$

where

$$P(Y = 1|X_1, X_2, \dots, X_k) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k.$$

Thus, β_j can be interpreted as the change in the probability that $Y_i = 1$, holding constant the other $k - 1$ regressors. Just as in common multiple regression, the β_j can be estimated using OLS and the robust standard error formulas can be used for hypothesis testing and computation of confidence intervals.

In most linear probability models, R^2 has no meaningful interpretation since the regression line can never fit the data perfectly if the dependent variable is binary and the regressors are continuous. This can be seen in the application below.

It is *essential* to use robust standard errors since the u_i in a linear probability model are always heteroskedastic.

Linear probability models are easily estimated in R using the function `lm()`.

Mortgage Data

Following the book, we start by loading the data set `HMDA` which provides data that relate to mortgage applications filed in Boston in the year of 1990.

```
# load `AER` package and attach the HMDA data
library(AER)
data(HMDA)
```

We continue by inspecting the first few observations and compute summary statistics afterwards.

```
# inspect the data
head(HMDA)
#>   deny pirat hirat      lvrat chist mhist phist unemp selfemp insurance condomin
#> 1   no  0.221 0.221 0.8000000     5     2   no    3.9      no      no      no
#> 2   no  0.265 0.265 0.9218750     2     2   no    3.2      no      no      no
#> 3   no  0.372 0.248 0.9203980     1     2   no    3.2      no      no      no
#> 4   no  0.320 0.250 0.8604651     1     2   no    4.3      no      no      no
#> 5   no  0.360 0.350 0.6000000     1     1   no    3.2      no      no      no
#> 6   no  0.240 0.170 0.5105263     1     1   no    3.9      no      no      no
#>   afam single hschool
#> 1   no      no      yes
```



```

#> 2   no   yes   yes
#> 3   no   no   yes
#> 4   no   no   yes
#> 5   no   no   yes
#> 6   no   no   yes
summary(HMDA)
#>   deny          pirat          hirat          lvrat          chist
#> no :2095   Min.   :0.0000   Min.   :0.0000   Min.   :0.0200   1:1353
#> yes: 285   1st Qu.:0.2800   1st Qu.:0.2140   1st Qu.:0.6527   2: 441
#>           Median :0.3300   Median :0.2600   Median :0.7795   3: 126
#>           Mean   :0.3308   Mean   :0.2553   Mean   :0.7378   4:  77
#>           3rd Qu.:0.3700   3rd Qu.:0.2988   3rd Qu.:0.8685   5: 182
#>           Max.   :3.0000   Max.   :3.0000   Max.   :1.9500   6: 201
#>   mhist   phist          unemp          selfemp   insurance   condomin
#> 1: 747   no :2205   Min.   : 1.800   no :2103   no :2332   no :1694
#> 2:1571   yes: 175   1st Qu.: 3.100   yes: 277   yes:  48   yes: 686
#> 3:  41           Median : 3.200
#> 4:  21           Mean   : 3.774
#>           3rd Qu.: 3.900
#>           Max.   :10.600
#>   afam          single   hschool
#> no :2041   no :1444   no :  39
#> yes: 339   yes: 936   yes:2341
#>
#>
#>
#>

```

The variable we are interested in modelling is `deny`, an indicator for whether an applicant's mortgage application has been accepted (`deny = no`) or denied (`deny = yes`). A regressor that ought to have power in explaining whether a mortgage application has been denied is `pirat`, the size of the anticipated total monthly loan payments relative to the the applicant's income. It is straightforward to translate this into the simple regression model

$$\text{deny} = \beta_0 + \beta_1 \times P/I \text{ ratio} + u. \quad (11.1)$$

We estimate this model just as any other linear regression model using `lm()`. Before we do so, the variable `deny` must be converted to a numeric variable using `as.numeric()` as `lm()` does not accept the *dependent variable* to be of class `factor`. Note that `as.numeric(HMDA$deny)` will turn `deny = no` into `deny = 1` and `deny = yes` into `deny = 2`, so using `as.numeric(HMDA$deny)-1` we obtain the values 0 and 1.

```

# convert 'deny' to numeric
HMDA$deny <- as.numeric(HMDA$deny) - 1

# estimate a simple linear probability model
denymod1 <- lm(deny ~ pirat, data = HMDA)
denymod1
#>
#> Call:
#> lm(formula = deny ~ pirat, data = HMDA)

```

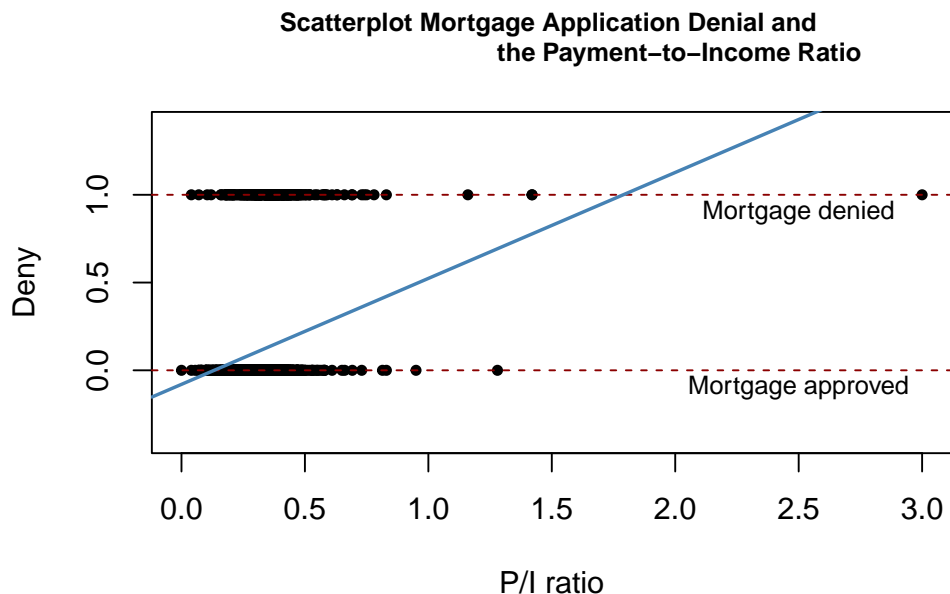
```
#>
#> Coefficients:
#> (Intercept)      pirat
#> -0.07991      0.60353
```

Next, we plot the data and the regression line to reproduce Figure 11.1 of the book.

```
# plot the data
plot(x = HMDA$pirat,
     y = HMDA$deny,
     main = "Scatterplot Mortgage Application Denial and
           the Payment-to-Income Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.8)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex = 0.8, "Mortgage approved")

# add the estimated regression line
abline(denymod1,
       lwd = 1.8,
       col = "steelblue")
```



According to the estimated model, a payment-to-income ratio of 1 is associated with an expected probability of mortgage application denial of roughly 50%. The model indicates that there is a positive relation between the payment-to-income ratio and the probability of a denied mortgage application so individuals with a high ratio of loan payments to income are more likely to be rejected.

We may use `coeftest()` to obtain robust standard errors for both coefficient estimates.

```
# print robust coefficient summary
coeftest(denymod1, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.079910  0.031967 -2.4998  0.01249 *
#> pirat       0.603535  0.098483  6.1283 1.036e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression line is

$$\widehat{deny} = -0.080 + 0.604P/I \text{ ratio.} \quad (11.2)$$

(0.032) (0.098)

The true coefficient on *P/I ratio* is statistically different from 0 at the 1% level. Its estimate can be interpreted as follows: a 1 percentage point increase in *P/I ratio* leads to an increase in the probability of a loan denial by $0.604 \cdot 0.01 = 0.00604 \approx 0.6\%$.

Following the book we augment the simple model (11.1) by an additional regressor *black* which equals 1 if the applicant is an African American and equals 0 otherwise. Such a specification is the baseline for investigating if there is racial discrimination in the mortgage market: if being black has a significant (positive) influence on the probability of a loan denial when we control for factors that allow for an objective assessment of an applicant's creditworthiness, this is an indicator for discrimination.

```
# rename the variable 'afam' for consistency
colnames(HMDA)[colnames(HMDA) == "afam"] <- "black"

# estimate the model
denymod2 <- lm(deny ~ pirat + black, data = HMDA)
coeftest(denymod2, vcov. = vcovHC)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.090514  0.033430 -2.7076  0.006826 **
#> pirat       0.559195  0.103671  5.3939 7.575e-08 ***
#> blackyes    0.177428  0.025055  7.0815 1.871e-12 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression function is

$$\widehat{deny} = -0.091 + 0.559P/I \text{ ratio} + 0.177black. \quad (11.3)$$

(0.029) (0.089) (0.025)

The coefficient on *black* is positive and significantly different from zero at the 0.01% level. The interpretation is that, holding the *P/I ratio* constant, being black increases the probability of a mortgage application denial by about 17.7%. This finding is compatible with racial discrimination. However, it might be distorted by omitted variable bias so discrimination could be a premature conclusion.

11.2 Probit and Logit Regression

The linear probability model has a major flaw: it assumes the conditional probability function to be linear. This does not restrict $P(Y = 1|X_1, \dots, X_k)$ to lie between 0 and 1. We can easily see this in our reproduction of Figure 11.1 of the book: for P/I ratio ≥ 1.75 , (11.2) predicts the probability of a mortgage application denial to be bigger than 1. For applications with P/I ratio close to 0, the predicted probability of denial is even negative so that the model has no meaningful interpretation here.

This circumstance calls for an approach that uses a nonlinear function to model the conditional probability function of a binary dependent variable. Commonly used methods are Probit and Logit regression.

Probit Regression

In Probit regression, the cumulative standard normal distribution function $\Phi(\cdot)$ is used to model the regression function when the dependent variable is binary, that is, we assume

$$E(Y|X) = P(Y = 1|X) = \Phi(\beta_0 + \beta_1 X). \quad (11.4)$$

$\beta_0 + \beta_1 X$ in (11.4) plays the role of a quantile z . Remember that

$$\Phi(z) = P(Z \leq z), \quad Z \sim \mathcal{N}(0, 1),$$

such that the Probit coefficient β_1 in (11.4) is the change in z associated with a one unit change in X . Although the effect on z of a change in X is linear, the link between z and the dependent variable Y is nonlinear since Φ is a nonlinear function of X .

Since the dependent variable is a nonlinear function of the regressors, the coefficient on X has no simple interpretation. According to Key Concept 8.1, the expected change in the probability that $Y = 1$ due to a change in P/I ratio can be computed as follows:

1. Compute the predicted probability that $Y = 1$ for the original value of X .
2. Compute the predicted probability that $Y = 1$ for $X + \Delta X$.
3. Compute the difference between both the predicted probabilities.

Of course, we can generalize (11.4) to Probit regression with multiple regressors to mitigate the risk of facing omitted variable bias. Probit regression essentials are summarized in Key Concept 11.2.

Key Concept 11.2 Probit Model, Predicted Probabilities and Estimated Effects

Assume that Y is a binary variable. The model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + u$$

with

$$P(Y = 1 | X_1, X_2, \dots, X_k) = \Phi(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)$$

is the population Probit model with multiple regressors X_1, X_2, \dots, X_k and $\Phi(\cdot)$ is the cumulative standard normal distribution function.

The predicted probability that $Y = 1$ given X_1, X_2, \dots, X_k can be calculated in two steps:

1. Compute $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$,
2. Look up $\Phi(z)$ by calling `pnorm()`.

β_j is the effect on z of a one unit change in regressor X_j , holding constant all other $k-1$ regressors.

The effect on the predicted probability of a change in a regressor can be computed as in Key Concept 8.1.

In R, Probit models can be estimated using the function `glm()` from the package `stats`. Using the argument `family` we specify that we want to use a Probit link function.

We now estimate a simple Probit model of the probability of a mortgage denial.

```
# estimate the simple probit model
denyprobit <- glm(deny ~ pirat,
                 family = binomial(link = "probit"),
                 data = HMDA)

coefest(denyprobit, vcov. = vcovHC, type = "HC1")
#>
#> z test of coefficients:
#>
#>           Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -2.19415    0.18901 -11.6087 < 2.2e-16 ***
#> pirat        2.96787    0.53698   5.5269 3.259e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model is

$$P(\widehat{\text{deny}} | P/I \text{ ratio}) = \Phi\left(\underset{(0.19)}{-2.19} + \underset{(0.54)}{2.97} P/I \text{ ratio}\right). \quad (11.5)$$

Just as in the linear probability model we find that the relation between the probability of denial and the payments-to-income ratio is positive and that the corresponding coefficient is highly significant.

The following code chunk reproduces Figure 11.2 of the book.

```

# plot data
plot(x = HMDA$pirat,
     y = HMDA$deny,
     main = "Probit Model of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.85)

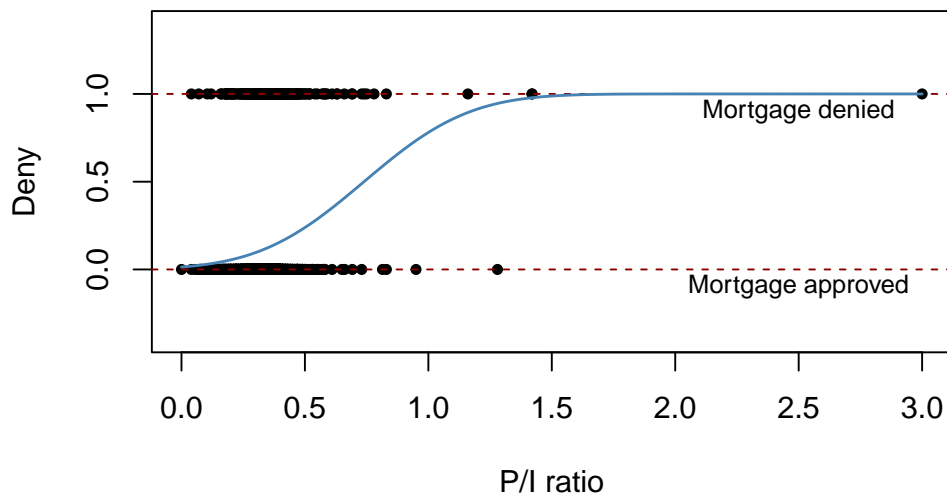
# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex = 0.8, "Mortgage approved")

# add estimated regression line
x <- seq(0, 3, 0.01)
y <- predict(denyprobit, list(pirat = x), type = "response")

lines(x, y, lwd = 1.5, col = "steelblue")

```

Probit Model of the Probability of Denial, Given P/I Ratio



The estimated regression function has a stretched “S”-shape which is typical for the CDF of a continuous random variable with symmetric PDF like that of a normal random variable. The function is clearly nonlinear and flattens out for large and small values of P/I ratio. The functional form thus also ensures that the predicted conditional probabilities of a denial lie between 0 and 1.

We use `predict()` to compute the predicted change in the denial probability when P/I ratio is increased from 0.3 to 0.4.

```

# 1. compute predictions for P/I ratio = 0.3, 0.4
predictions <- predict(denyprobit,
                       newdata = data.frame("pirat" = c(0.3, 0.4)),
                       type = "response")

```

```
# 2. Compute difference in probabilities
diff(predictions)
#>      2
#> 0.06081433
```

We find that an increase in the payment-to-income ratio from 0.3 to 0.4 is predicted to increase the probability of denial by approximately 6.1%.

We continue by using an augmented Probit model to estimate the effect of race on the probability of a mortgage application denial.

```
#As a remainder, we have renamed the variable "afam" to "black" in Chapter 11.1
#colnames(HMDA)[colnames(HMDA) == "afam"] <- "black"
#We continue by using an augmented
denyprobit2 <- glm(deny ~ pirat + black,
                  family = binomial(link = "probit"),
                  data = HMDA)

coeftest(denyprobit2, vcov. = vcovHC, type = "HC1")
#>
#> z test of coefficients:
#>
#>           Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -2.258787  0.176608 -12.7898 < 2.2e-16 ***
#> pirat       2.741779  0.497673  5.5092 3.605e-08 ***
#> blackyes    0.708155  0.083091  8.5227 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model equation is

$$P(\text{deny} | \widehat{P/I \text{ ratio}}, \text{black}) = \Phi\left(\underset{(0.18)}{-2.26} + \underset{(0.50)}{2.74} P/I \text{ ratio} + \underset{(0.08)}{0.71} \text{black}\right). \quad (11.6)$$

While all coefficients are highly significant, both the estimated coefficients on the payments-to-income ratio and the indicator for African American descent are positive. Again, the coefficients are difficult to interpret but they indicate that, first, African Americans have a higher probability of denial than white applicants, holding the payments-to-income ratio constant and second, applicants with a high payments-to-income ratio face a higher risk of being rejected.

How big is the estimated difference in denial probabilities between two hypothetical applicants with the same payments-to-income ratio? As before, we may use `predict()` to compute this difference.

```
# 1. Compute predictions for P/I ratio = 0.3
predictions <- predict(denyprobit2,
                      newdata = data.frame("black" = c("no", "yes"),
                                           "pirat" = c(0.3, 0.3)),
                      type = "response")

# 2. Compute difference in probabilities
diff(predictions)
#>      2
#> 0.1578117
```

In this case, the estimated difference in denial probabilities is about 15.8%.

Logit Regression

Key Concept 11.3 summarizes the Logit regression function.

Key Concept 11.3 Logit Regression

The population Logit regression function is

$$P(Y = 1|X_1, X_2, \dots, X_k) = F(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k) \\ = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k)}}$$

The idea is similar to Probit regression except that a different CDF is used:

$$F(x) = \frac{1}{1 + e^{-x}}$$

is the CDF of a standard logistic distribution.

As for Probit regression, there is no simple interpretation of the model coefficients and it is best to consider predicted probabilities or differences in predicted probabilities. Here again, t -statistics and confidence intervals based on large sample normal approximations can be computed as usual.

It is fairly easy to estimate a Logit regression model using R.

```
denylogit <- glm(deny ~ pirat,
                 family = binomial(link = "logit"),
                 data = HMDA)

coeftest(denylogit, vcov. = vcovHC, type = "HC1")
#>
#> z test of coefficients:
#>
#>           Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -4.02843    0.35898 -11.2218 < 2.2e-16 ***
#> pirat        5.88450    1.00015   5.8836 4.014e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The subsequent code chunk reproduces Figure 11.3 of the book.

```
# plot data
plot(x = HMDA$pirat,
     y = HMDA$deny,
     main = "Probit and Logit Models of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.9)
```



```

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex = 0.8, "Mortgage approved")

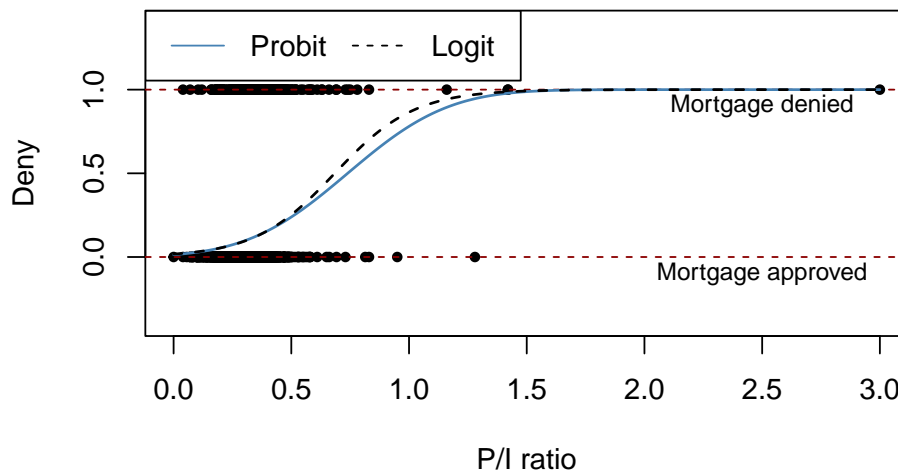
# add estimated regression line of Probit and Logit models
x <- seq(0, 3, 0.01)
y_probit <- predict(denyprobit, list(pirat = x), type = "response")
y_logit <- predict(denylogit, list(pirat = x), type = "response")

lines(x, y_probit, lwd = 1.5, col = "steelblue")
lines(x, y_logit, lwd = 1.5, col = "black", lty = 2)

# add a legend
legend("topleft",
      horiz = TRUE,
      legend = c("Probit", "Logit"),
      col = c("steelblue", "black"),
      lty = c(1, 2))

```

Probit and Logit Models of the Probability of Denial, Given P/I Ratio



Both models produce very similar estimates of the probability that a mortgage application will be denied depending on the applicants payment-to-income ratio.

Following the book we extend the simple Logit model of mortgage denial with the additional regressor *black*.

```

# estimate a Logit regression with multiple regressors
denylogit2 <- glm(deny ~ pirat + black,
                 family = binomial(link = "logit"),
                 data = HMDA)

coeftest(denylogit2, vcov. = vcovHC, type = "HC1")
#>

```

```
#> z test of coefficients:
#>
#>           Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -4.12556    0.34597 -11.9245 < 2.2e-16 ***
#> pirat       5.37036    0.96376  5.5723 2.514e-08 ***
#> blackyes    1.27278    0.14616  8.7081 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain

$$P(\text{deny} = 1 | \widehat{P/I \text{ ratio}}, \text{black}) = F\left(-\frac{4.13}{(0.35)} + \frac{5.37}{(0.96)} P/I \text{ ratio} + \frac{1.27}{(0.15)} \text{black}\right). \quad (11.7)$$

As for the Probit model (11.6) all model coefficients are highly significant and we obtain positive estimates for the coefficients on *P/I ratio* and *black*. For comparison we compute the predicted probability of denial for two hypothetical applicants that differ in race and have a *P/I ratio* of 0.3.

```
# 1. compute predictions for P/I ratio = 0.3
predictions <- predict(denylogit2,
                       newdata = data.frame("black" = c("no", "yes"),
                                             "pirat" = c(0.3, 0.3)),
                       type = "response")

predictions
#>           1           2
#> 0.07485143 0.22414592

# 2. Compute difference in probabilities
diff(predictions)
#>           2
#> 0.1492945
```

We find that the white applicant faces a denial probability of only 7.5%, while the African American is rejected with a probability of 22.4%, a difference of 14.9% points.

Comparison of the Models

The Probit model and the Logit model deliver only approximations to the unknown population regression function $E(Y|X)$. It is not obvious how to decide which model to use in practice. The linear probability model has the clear drawback of not being able to capture the nonlinear nature of the population regression function and it may predict probabilities to lie outside the interval $[0, 1]$. Probit and Logit models are harder to interpret but capture the nonlinearities better than the linear approach: both models produce predictions of probabilities that lie inside the interval $[0, 1]$. Predictions of all three models are often close to each other. The book suggests to use the method that is easiest to use in the statistical software of choice. As we have seen, it is equally easy to estimate Probit and Logit model using R. We can therefore give no general recommendation which method to use.

11.3 Estimation and Inference in the Logit and Probit Models

So far nothing has been said about *how* Logit and Probit models are estimated by statistical software. The reason why this is interesting is that both models are *nonlinear in the parameters* and thus cannot

be estimated using OLS. Instead one relies on *maximum likelihood estimation* (MLE). Another approach is estimation by *nonlinear least squares* (NLS).

Nonlinear Least Squares

Consider the multiple regression Probit model

$$E(Y_i|X_{1i}, \dots, X_{ki}) = P(Y_i = 1|X_{1i}, \dots, X_{ki}) = \Phi(\beta_0 + \beta_1 X_{1i} + \dots + \beta_k X_{ki}). \quad (11.8)$$

Similarly to OLS, NLS estimates the parameters $\beta_0, \beta_1, \dots, \beta_k$ by minimizing the sum of squared mistakes

$$\sum_{i=1}^n [Y_i - \Phi(b_0 + b_1 X_{1i} + \dots + b_k X_{ki})]^2.$$

NLS estimation is a consistent approach that produces estimates which are normally distributed in large samples. In R there are functions like `nls()` from package `stats` which provide algorithms for solving nonlinear least squares problems. However, NLS is inefficient, meaning that there are estimation techniques that have a smaller variance which is why we will not dwell any further on this topic.

Maximum Likelihood Estimation

In MLE we seek to estimate the unknown parameters choosing them such that the likelihood of drawing the sample observed is maximized. This probability is measured by means of the likelihood function, the joint probability distribution of the data treated as a function of the unknown parameters. Put differently, the maximum likelihood estimates of the unknown parameters are the values that result in a model which is most likely to produce the observed data. It turns out that MLE is more efficient than NLS.

As maximum likelihood estimates are normally distributed in large samples, statistical inference for coefficients in nonlinear models like Logit and Probit regression can be made using the same tools that are used for linear regression models: we can compute *t*-statistics and confidence intervals.

Many software packages use an MLE algorithm for estimation of nonlinear models. The function `glm()` uses an algorithm named *iteratively reweighted least squares*.

Measures of Fit

It is important to be aware that the usual R^2 and \bar{R}^2 are *invalid* for nonlinear regression models. The reason for this is simple: both measures assume that the relation between the dependent and the explanatory variable(s) is linear. This obviously does not hold for Probit and Logit models. Thus R^2 need not lie between 0 and 1 and there is no meaningful interpretation. However, statistical software sometimes reports these measures anyway.

There are many measures of fit for nonlinear regression models and there is no consensus which one should be reported. The situation is even more complicated because there is no measure of fit that is generally meaningful. For models with a binary response variable like *deny*, one could use the following rule:

If $Y_i = 1$ and $P(Y_i|X_{i1}, \dots, X_{ik}) > 0.5$ or if $Y_i = 0$ and $P(Y_i|X_{i1}, \dots, X_{ik}) < 0.5$, consider the Y_i as correctly predicted. Otherwise Y_i is said to be incorrectly predicted. The measure of fit is the share of correctly predicted observations. The downside of such an approach is that it does not mirror the quality of the prediction: whether $P(Y_i = 1|X_{i1}, \dots, X_{ik}) = 0.51$ or $P(Y_i = 1|X_{i1}, \dots, X_{ik}) = 0.99$ is not reflected, we just predict $Y_i = 1$.¹

¹This is in contrast to the case of a numeric dependent variable where we use the squared errors for assessment of the quality of the prediction.

An alternative to the latter are so called pseudo- R^2 measures. In order to measure the quality of the fit, these measures compare the value of the maximized (log-)likelihood of the model with all regressors (the *full model*) to the likelihood of a model with no regressors (*null model*, regression on a constant).

For example, consider a Probit regression. The pseudo- R^2 is given by

$$\text{pseudo-}R^2 = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}$$

where $f_j^{max} \in [0, 1]$ denotes the maximized likelihood for model j .

The reasoning behind this is that the maximized likelihood increases as additional regressors are added to the model, similarly to the decrease in SSR when regressors are added in a linear regression model. If the full model has a similar maximized likelihood as the null model, the full model does not really improve upon a model that uses only the information in the dependent variable, so pseudo- $R^2 \approx 0$. If the full model fits the data very well, the maximized likelihood should be close to 1 such that $\ln(f_{full}^{max}) \approx 0$ and pseudo- $R^2 \approx 1$. See Appendix 11.2 of the book for more on MLE and pseudo- R^2 measures.

`summary()` does not report pseudo- R^2 for models estimated by `glm()` but we can use the entries *residual deviance* (`deviance`) and *null deviance* (`null.deviance`) instead. These are computed as

$$\text{deviance} = -2 \times [\ln(f_{saturated}^{max}) - \ln(f_{full}^{max})]$$

and

$$\text{null deviance} = -2 \times [\ln(f_{saturated}^{max}) - \ln(f_{null}^{max})]$$

where $f_{saturated}^{max}$ is the maximized likelihood for a model which assumes that each observation has its own parameter (there are $n + 1$ parameters to be estimated which leads to a perfect fit). For models with a binary dependent variable, it holds that

$$\text{pseudo-}R^2 = 1 - \frac{\text{deviance}}{\text{null deviance}} = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}.$$

We now compute the pseudo- R^2 for the augmented Probit model of mortgage denial.

```
# compute pseudo-R2 for the probit model of mortgage denial
pseudoR2 <- 1 - (denyprobit2$deviance) / (denyprobit2>null.deviance)
pseudoR2
#> [1] 0.08594259
```

Another way to obtain the pseudo- R^2 is to estimate the null model using `glm()` and extract the maximized log-likelihoods for both the null and the full model using the function `logLik()`.

```
# compute the null model
denyprobit_null <- glm(formula = deny ~ 1,
                      family = binomial(link = "probit"),
                      data = HMDA)

# compute the pseudo-R2 using 'logLik'
1 - logLik(denyprobit2)[1] / logLik(denyprobit_null)[1]
#> [1] 0.08594259
```

11.4 Application to the Boston HMDA Data

Models (11.6) and (11.7) indicate that denial rates are higher for African American applicants holding constant the payment-to-income ratio. Both results could be subject to omitted variable bias. In order to obtain a more trustworthy estimate of the effect of being black on the probability of a mortgage application denial we estimate a linear probability model as well as several Logit and Probit models. We thereby control for financial variables and additional applicant characteristics which are likely to influence the probability of denial and differ between black and white applicants.

Sample averages as shown in Table 11.1 of the book can be easily reproduced using the functions `mean()` (as usual for numeric variables) and `prop.table()` (for factor variables).

```
# Mean P/I ratio
mean(HMDA$pirat)
#> [1] 0.3308136

# inhouse expense-to-total-income ratio
mean(HMDA$hirat)
#> [1] 0.2553461

# loan-to-value ratio
mean(HMDA$lvrat)
#> [1] 0.7377759

# consumer credit score
mean(as.numeric(HMDA$chist))
#> [1] 2.116387

# mortgage credit score
mean(as.numeric(HMDA$mhist))
#> [1] 1.721008

# public bad credit record
mean(as.numeric(HMDA$phist)-1)
#> [1] 0.07352941

# denied mortgage insurance
prop.table(table(HMDA$insurance))
#>
#>      no      yes
#> 0.97983193 0.02016807

# self-employed
prop.table(table(HMDA$selfemp))
#>
#>      no      yes
#> 0.8836134 0.1163866

# single
prop.table(table(HMDA$single))
#>
#>      no      yes
#> 0.6067227 0.3932773
```

```

# high school diploma
prop.table(table(HMDA$hschool))
#>
#>      no      yes
#> 0.01638655 0.98361345

# unemployment rate
mean(HMDA$unemp)
#> [1] 3.774496

# condominium
prop.table(table(HMDA$condomin))
#>
#>      no      yes
#> 0.7117647 0.2882353

# black
prop.table(table(HMDA$black))
#>
#>      no      yes
#> 0.857563 0.142437

# deny
prop.table(table(HMDA$deny))
#>
#>      0      1
#> 0.8802521 0.1197479

```

See Chapter 11.4 of the book or use R's help function for more on variables contained in the HMDA dataset.

Before estimating the models we transform the loan-to-value ratio (*lvrat*) into a factor variable, where

$$lvrat = \begin{cases} \text{low} & \text{if } lvrat < 0.8, \\ \text{medium} & \text{if } 0.8 \leq lvrat \leq 0.95, \\ \text{high} & \text{if } lvrat > 0.95 \end{cases}$$

and convert both credit scores to numeric variables.

```

# define low, medium and high loan-to-value ratio
HMDA$lvrat <- factor(
  ifelse(HMDA$lvrat < 0.8, "low",
  ifelse(HMDA$lvrat >= 0.8 & HMDA$lvrat <= 0.95, "medium", "high")),
  levels = c("low", "medium", "high"))

# convert credit scores to numeric
HMDA$mhist <- as.numeric(HMDA$mhist)
HMDA$chist <- as.numeric(HMDA$chist)

```

Next we reproduce the estimation results presented in Table 11.2 of the book.

```

# estimate all 6 models for the denial probability
lpm_HMDA <- lm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
              + insurance + selfemp, data = HMDA)

logit_HMDA <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                 + insurance + selfemp,
                 family = binomial(link = "logit"),
                 data = HMDA)

probit_HMDA1 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                  + insurance + selfemp,
                  family = binomial(link = "probit"),
                  data = HMDA)

probit_HMDA2 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                  + insurance + selfemp + single + hschool + unemp,
                  family = binomial(link = "probit"),
                  data = HMDA)

probit_HMDA3 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist
                  + phist + insurance + selfemp + single + hschool + unemp
                  +condomin + I(mhist==3) + I(mhist==4) + I(chist==3)
                  + I(chist==4) + I(chist==5)+ I(chist==6),
                  family = binomial(link = "probit"),
                  data = HMDA)

probit_HMDA4 <- glm(deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist
                  + insurance + selfemp + single + hschool + unemp,
                  family = binomial(link = "probit"),
                  data = HMDA)

```

Just as in previous chapters, we store heteroskedasticity-robust standard errors of the coefficient estimators in a list which is then used as the argument `se` in `stargazer()`.

```

rob_se <- list(sqrt(diag(vcovHC(lpm_HMDA, type = "HC1"))),
              sqrt(diag(vcovHC(logit_HMDA, type = "HC1"))),
              sqrt(diag(vcovHC(probit_HMDA1, type = "HC1"))),
              sqrt(diag(vcovHC(probit_HMDA2, type = "HC1"))),
              sqrt(diag(vcovHC(probit_HMDA3, type = "HC1"))),
              sqrt(diag(vcovHC(probit_HMDA4, type = "HC1"))))

stargazer(lpm_HMDA, logit_HMDA, probit_HMDA1,
          probit_HMDA2, probit_HMDA3, probit_HMDA4,
          digits = 3,
          type = "latex",
          header = FALSE,
          se = rob_se,
          model.numbers = FALSE,
          column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)"))

```

In Table 11.1, models (1), (2) and (3) are baseline specifications that include several financial control variables. They differ only in the way they model the denial probability. Model (1) is a linear probability

Table 11.1: HMDA Data: LPM, Probit and Logit Models

	<i>Dependent variable:</i>					
	<i>OLS</i> (1)	<i>logistic</i> (2)	<i>deny</i> (3)	(4)	(5)	(6)
blackyes	0.084*** (0.023)	0.688*** (0.183)	0.389*** (0.099)	0.371*** (0.100)	0.363*** (0.101)	0.246 (0.479)
pirat	0.449*** (0.114)	4.764*** (1.332)	2.442*** (0.673)	2.464*** (0.654)	2.622*** (0.665)	2.572*** (0.728)
hirat	-0.048 (0.110)	-0.109 (1.298)	-0.185 (0.689)	-0.302 (0.689)	-0.502 (0.715)	-0.538 (0.755)
lvratmedium	0.031** (0.013)	0.464*** (0.160)	0.214*** (0.082)	0.216*** (0.082)	0.215** (0.084)	0.216*** (0.083)
lvrathigh	0.189*** (0.050)	1.495*** (0.325)	0.791*** (0.183)	0.795*** (0.184)	0.836*** (0.185)	0.788*** (0.185)
chist	0.031*** (0.005)	0.290*** (0.039)	0.155*** (0.021)	0.158*** (0.021)	0.344*** (0.108)	0.158*** (0.021)
mhist	0.021* (0.011)	0.279** (0.138)	0.148** (0.073)	0.110 (0.076)	0.162 (0.104)	0.111 (0.077)
phistyes	0.197*** (0.035)	1.226*** (0.203)	0.697*** (0.114)	0.702*** (0.115)	0.717*** (0.116)	0.705*** (0.115)
insuranceyes	0.702*** (0.045)	4.548*** (0.576)	2.557*** (0.305)	2.585*** (0.299)	2.589*** (0.306)	2.590*** (0.299)
selfempyes	0.060*** (0.021)	0.666*** (0.214)	0.359*** (0.113)	0.346*** (0.116)	0.342*** (0.116)	0.348*** (0.116)
singleyes				0.229*** (0.080)	0.230*** (0.086)	0.226*** (0.081)
hschoolyes				-0.613*** (0.229)	-0.604** (0.237)	-0.620*** (0.229)
unemp				0.030* (0.018)	0.028 (0.018)	0.030 (0.018)
condominyes				-0.055 (0.096)	-0.055 (0.096)	
I(mhist == 3)				-0.107 (0.301)	-0.107 (0.301)	
I(mhist == 4)				-0.383 (0.427)	-0.383 (0.427)	
I(chist == 3)				-0.226 (0.248)	-0.226 (0.248)	
I(chist == 4)				-0.251 (0.338)	-0.251 (0.338)	
I(chist == 5)				-0.789* (0.412)	-0.789* (0.412)	
I(chist == 6)				-0.905* (0.515)	-0.905* (0.515)	
blackyes:pirat						-0.579 (1.550)
blackyes:hirat						1.232 (1.709)
Constant	-0.183*** (0.028)	-5.707*** (0.484)	-3.041*** (0.250)	-2.575*** (0.350)	-2.896*** (0.404)	-2.543*** (0.370)
Observations	2,380	2,380	2,380	2,380	2,380	2,380
R ²	0.266					
Adjusted R ²	0.263					
Log Likelihood		-635.637	-636.847	-628.614	-625.064	-628.332
Akaike Inf. Crit.		1,293.273	1,295.694	1,285.227	1,292.129	1,288.664
Residual Std. Error	0.279 (df = 2369)					
F Statistic	85.974*** (df = 10; 2369)					

Note: * p<0.1; ** p<0.05; *** p<0.01

model, model (2) is a Logit regression and model (3) uses the Probit approach.

In the linear model (1), the coefficients have direct interpretation. For example, an increase in the consumer credit score by 1 unit is estimated to increase the probability of a loan denial by about 0.031 percentage points. Having a high loan-to-value ratio is detrimental for credit approval: the coefficient for a loan-to-value ratio higher than 0.95 is 0.189 so clients with this property are estimated to face an almost 19% larger risk of denial than those with a low loan-to-value ratio, *ceteris paribus*. The estimated coefficient on the race dummy is 0.084, which indicates the denial probability for African Americans is 8.4% larger than for white applicants with the same characteristics except for race. Apart from the housing-expense-to-income ratio and the mortgage credit score, all coefficients are significant.

Models (2) and (3) provide similar evidence that there is racial discrimination in the U.S. mortgage market. All coefficients except for the housing expense-to-income ratio (which is not significantly different from zero) are significant at the 1% level. As discussed above, the nonlinearity makes the interpretation of the coefficient estimates more difficult than for model (1). In order to make a statement about the effect of being black, we need to compute the estimated denial probability for two individuals that differ only in race. For the comparison we consider two individuals that share mean values for all numeric regressors. For the qualitative variables we assign the property that is most representative for the data at hand. For example, consider self-employment: we have seen that about 88% of all individuals in the sample are not self-employed such that we set `selfemp = no`. Using this approach, the estimate for the effect on the denial probability of being African American of the Logit model (2) is about 4%. The next code chunk shows how to apply this approach for models (1) to (7) using R.

```
# compute regressor values for an average black person
new <- data.frame(
  "pirat" = mean(HMDA$pirat),
  "hirat" = mean(HMDA$hirat),
  "lvrat" = "low",
  "chist" = mean(HMDA$chist),
  "mhist" = mean(HMDA$mhist),
  "phist" = "no",
  "insurance" = "no",
  "selfemp" = "no",
  "black" = c("no", "yes"),
  "single" = "no",
  "hschool" = "yes",
  "unemp" = mean(HMDA$unemp),
  "condomin" = "no")

# difference predicted by the LPM
predictions <- predict(lpm_HMDA, newdata = new)
diff(predictions)
#>      2
#> 0.08369674

# difference predicted by the logit model
predictions <- predict(logit_HMDA, newdata = new, type = "response")
diff(predictions)
#>      2
#> 0.04042135

# difference predicted by probit model (3)
predictions <- predict(probit_HMDA1, newdata = new, type = "response")
```

```

diff(predictions)
#>      2
#> 0.05049716

# difference predicted by probit model (4)
predictions <- predict(probit_HMDA2, newdata = new, type = "response")
diff(predictions)
#>      2
#> 0.03978918

# difference predicted by probit model (5)
predictions <- predict(probit_HMDA3, newdata = new, type = "response")
diff(predictions)
#>      2
#> 0.04972468

# difference predicted by probit model (6)
predictions <- predict(probit_HMDA4, newdata = new, type = "response")
diff(predictions)
#>      2
#> 0.03955893

```

The estimates of the impact on the denial probability of being black are similar for models (2) and (3). It is interesting that the magnitude of the estimated effects is much smaller than for Probit and Logit models that do not control for financial characteristics (see section 11.2). This indicates that these simple models produce biased estimates due to omitted variables.

Regressions (4) to (6) use regression specifications that include different applicant characteristics and credit rating indicator variables as well as interactions. However, most of the corresponding coefficients are not significant and the estimates of the coefficient on `black` obtained for these models as well as the estimated difference in denial probabilities do not differ much from those obtained for the similar specifications (2) and (3).

An interesting question related to racial discrimination can be investigated using the Probit model (6) where the interactions `blackyes:pirat` and `blackyes:hirat` are added to model (4). If the coefficient on `blackyes:pirat` was different from zero, the effect of the payment-to-income ratio on the denial probability would be different for black and white applicants. Similarly, a non-zero coefficient on `blackyes:hirat` would indicate that loan officers weight the risk of bankruptcy associated with a high loan-to-value ratio differently for black and white mortgage applicants. We can test whether these coefficients are jointly significant at the 5% level using an F -Test.

```

linearHypothesis(probit_HMDA4,
                 test = "F",
                 c("blackyes:pirat=0", "blackyes:hirat=0"),
                 vcov = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> blackyes:pirat = 0
#> blackyes:hirat = 0
#>
#> Model 1: restricted model
#> Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +

```

```
#> insurance + selfemp + single + hschool + unemp
#>
#> Note: Coefficient covariance matrix supplied.
#>
#> Res.Df Df      F Pr(>F)
#> 1 2366
#> 2 2364 2 0.2473 0.7809
```

Since p -value ≈ 0.77 for this test, the null cannot be rejected. Nonetheless, we can reject the hypothesis that there is no racial discrimination at all since the corresponding F -test has a p -value of about 0.002.

```
linearHypothesis(probit_HMDA4,
                 test = "F",
                 c("blackyes=0", "blackyes:pirat=0", "blackyes:hirat=0"),
                 vcov = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> blackyes = 0
#> blackyes:pirat = 0
#> blackyes:hirat = 0
#>
#> Model 1: restricted model
#> Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +
#> insurance + selfemp + single + hschool + unemp
#>
#> Note: Coefficient covariance matrix supplied.
#>
#> Res.Df Df      F Pr(>F)
#> 1 2367
#> 2 2364 3 4.7774 0.002534 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary

Models (1) to (6) provide evidence that there is an effect of being African American on the probability of a mortgage application denial: in all specifications, the effect is estimated to be positive (ranging from 4% to 5%) and is significantly different from zero at the 1% level. While the linear probability model seems to slightly overestimate this effect, it still can be used as an approximation to an intrinsically nonlinear relationship.

See Chapters 11.4 and 11.5 of the book for a discussion of external and internal validity of this study and some concluding remarks on regression models where the dependent variable is binary.

11.5 Exercises

This interactive part of the book is only available in the HTML version.

Chapter 12

Instrumental Variables Regression

As discussed in Chapter 9, regression models may suffer from problems like omitted variables, measurement errors and simultaneous causality. If so, the error term is correlated with the regressor of interest and so that the corresponding coefficient is estimated inconsistently. So far we have assumed that we can add the omitted variables to the regression to mitigate the risk of biased estimation of the causal effect of interest. However, if omitted factors cannot be measured or are not available for other reasons, multiple regression cannot solve the problem. The same issue arises if there is simultaneous causality. When causality runs from X to Y and vice versa, there will be an estimation bias that cannot be corrected for by multiple regression.

A general technique for obtaining a consistent estimator of the coefficient of interest is instrumental variables (IV) regression. In this chapter we focus on the IV regression tool called *two-stage least squares* (TSLS). The first sections briefly recap the general mechanics and assumptions of IV regression and show how to perform TSLS estimation using R. Next, IV regression is used for estimating the elasticity of the demand for cigarettes — a classical example where multiple regression fails to do the job because of simultaneous causality.

Just like for the previous chapter, the packages `AER` (Kleibler and Zeileis, 2008b) and `stargazer` (Hlavac, 2022) are required for reproducing the code presented in this chapter. Check whether the code chunk below executes without any error messages.

```
library(AER)
library(stargazer)
```

12.1 The IV Estimator with a Single Regressor and a Single Instrument

Consider the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i \quad , \quad i = 1, \dots, n \quad (12.1)$$

where the error term u_i is correlated with the regressor X_i (X is *endogenous*) such that OLS is inconsistent for the true β_1 . In the most simple case, IV regression uses a single instrumental variable Z to obtain a consistent estimator for β_1 .

Z must satisfy two conditions to be a valid instrument:

1. Instrument relevance condition:

X and its instrument Z must be correlated: $\rho_{Z_i, X_i} \neq 0$.

2. Instrument exogeneity condition:

The instrument Z must not be correlated with the error term u : $\rho_{Z_i, u_i} = 0$.

The Two-Stage Least Squares Estimator

As can be guessed from its name, TSLS proceeds in two stages. In the first stage, the variation in the endogenous regressor X is decomposed into a problem-free component that is explained by the instrument Z and a problematic component that is correlated with the error u_i . The second stage uses the problem-free component of the variation in X to estimate β_1 .

The first stage regression model is

$$X_i = \pi_0 + \pi_1 Z_i + \nu_i,$$

where $\pi_0 + \pi_1 Z_i$ is the component of X_i that is explained by Z_i while ν_i is the component that cannot be explained by Z_i and exhibits correlation with u_i .

Using the OLS estimates $\hat{\pi}_0$ and $\hat{\pi}_1$ we obtain predicted values \hat{X}_i , $i = 1, \dots, n$. If Z is a valid instrument, the \hat{X}_i are problem-free in the sense that \hat{X} is exogenous in a regression of Y on \hat{X} which is done in the second stage regression. The second stage produces $\hat{\beta}_0^{TSLS}$ and $\hat{\beta}_1^{TSLS}$, the TSLS estimates of β_0 and β_1 .

For the case of a single instrument one can show that the TSLS estimator of β_1 is

$$\hat{\beta}_1^{TSLS} = \frac{s_{ZY}}{s_{ZX}} = \frac{\frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})(Z_i - \bar{Z})}{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Z_i - \bar{Z})}, \quad (12.2)$$

which is nothing but the ratio of the sample covariance between Z and Y to the sample covariance between Z and X .

As shown in Appendix 12.3 of the book, (12.2) is a consistent estimator for β_1 in (12.1) under the assumption that Z is a valid instrument. Just as for every other OLS estimator we have considered so far, the CLT implies that the distribution of $\hat{\beta}_1^{TSLS}$ can be approximated by a normal distribution if the sample size is large. This allows us to use t -statistics and confidence intervals which are also computed by certain R functions. A more detailed argument on the large-sample distribution of the TSLS estimator is sketched in Appendix 12.3 of the book.

Application to the Demand For Cigarettes

The relation between the demand for and the price of commodities is a simple yet widespread problem in economics. Health economics is concerned with the study of how health-affecting behavior of individuals is influenced by the health-care system and regulation policy. Probably the most prominent example in public policy debates is smoking as it is related to many illnesses and negative externalities.

It is plausible that cigarette consumption can be reduced by taxing cigarettes more heavily. The question is by *how much* taxes must be increased to reach a certain reduction in cigarette consumption. Economists use elasticities to answer this kind of question. Since the price elasticity for the demand of cigarettes is unknown, it must be estimated. As discussed in the box *Who Invented Instrumental Variables Regression* presented in Chapter 12.1 of the book, an OLS regression of log quantity on log price cannot be used to estimate the effect of interest since there is simultaneous causality between demand and supply. Instead, IV regression can be used.

We use the data set `CigarettesSW` which comes with the package `AER`. It is a panel data set that contains observations on cigarette consumption and several economic indicators for all 48 continental federal states of the U.S. from 1985 to 1995. Following the book, we consider data for the cross section of states in 1995 only.

We start by loading the package, attaching the data set and getting an overview.

```
# load the data set and get an overview
library(AER)
data("CigarettesSW")
summary(CigarettesSW)
#>      state      year      cpi      population      packs
#> AL       : 2   1985:48   Min.    :1.076   Min.    : 478447   Min.    : 49.27
#> AR       : 2   1995:48   1st Qu.:1.076   1st Qu.: 1622606   1st Qu.: 92.45
#> AZ       : 2                Median :1.300   Median : 3697472   Median :110.16
#> CA       : 2                Mean    :1.300   Mean    : 5168866   Mean    :109.18
#> CO       : 2                3rd Qu.:1.524   3rd Qu.: 5901500   3rd Qu.:123.52
#> CT       : 2                Max.    :1.524   Max.    :31493524   Max.    :197.99
#> (Other):84
#>      income      tax      price      taxes
#> Min.    : 6887097   Min.    :18.00   Min.    : 84.97   Min.    : 21.27
#> 1st Qu.: 25520384   1st Qu.:31.00   1st Qu.:102.71   1st Qu.: 34.77
#> Median : 61661644   Median :37.00   Median :137.72   Median : 41.05
#> Mean    : 99878736   Mean    :42.68   Mean    :143.45   Mean    : 48.33
#> 3rd Qu.:127313964   3rd Qu.:50.88   3rd Qu.:176.15   3rd Qu.: 59.48
#> Max.    :771470144   Max.    :99.00   Max.    :240.85   Max.    :112.63
#>
```

Use `?CigarettesSW` for a detailed description of the variables.

We are interested in estimating β_1 in

$$\log(Q_i^{\text{cigarettes}}) = \beta_0 + \beta_1 \log(P_i^{\text{cigarettes}}) + u_i, \quad (12.3)$$

where $Q_i^{\text{cigarettes}}$ is the number of cigarette packs per capita sold and $P_i^{\text{cigarettes}}$ is the after-tax average real price per pack of cigarettes in state i .

The instrumental variable we are going to use for instrumenting the endogenous regressor $\log(P_i^{\text{cigarettes}})$ is `SalesTax`, the portion of taxes on cigarettes arising from the general sales tax. `SalesTax` is measured in dollars per pack. The idea is that `SalesTax` is a relevant instrument as it is included in the after-tax average price per pack. Also, it is plausible that `SalesTax` is exogenous since the sales tax does not influence quantity sold directly but indirectly through the price.

We perform some transformations in order to obtain deflated cross section data for the year 1995.

We also compute the sample correlation between the sales tax and price per pack. The sample correlation is a consistent estimator of the population correlation. The estimate of approximately 0.614 indicates that `SalesTax` and $P_i^{\text{cigarettes}}$ exhibit positive correlation which meets our expectations: higher sales taxes lead to higher prices. However, a correlation analysis like this is not sufficient for checking whether the instrument is relevant. We will later come back to the issue of checking whether an instrument is relevant and exogenous.

```
# compute real per capita prices
CigarettesSW$rprice <- with(CigarettesSW, price / cpi)
```

```
# compute the sales tax
CigarettesSW$salestax <- with(CigarettesSW, (taxes - tax) / cpi)

# check the correlation between sales tax and price
cor(CigarettesSW$salestax, CigarettesSW$price)
#> [1] 0.6141228

# generate a subset for the year 1995
c1995 <- subset(CigarettesSW, year == "1995")
```

The first stage regression is

$$\log(P_i^{cigarettes}) = \pi_0 + \pi_1 SalesTax_i + \nu_i.$$

We estimate this model in R using `lm()`. In the second stage we run a regression of $\log(Q_i^{cigarettes})$ on $\log(\widehat{P}_i^{cigarettes})$ to obtain $\widehat{\beta}_0^{TOLS}$ and $\widehat{\beta}_1^{TOLS}$.

```
# perform the first stage regression
cig_s1 <- lm(log(rprice) ~ salestax, data = c1995)

coeftest(cig_s1, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 4.6165463  0.0289177 159.6444 < 2.2e-16 ***
#> salestax    0.0307289  0.0048354   6.3549 8.489e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first stage regression is

$$\log(\widehat{P}_i^{cigarettes}) = \underset{(0.03)}{4.62} + \underset{(0.005)}{0.031} SalesTax_i,$$

which predicts the relation between sales tax price per cigarettes to be positive. How much of the observed variation in $\log(P_i^{cigarettes})$ is explained by the instrument *SalesTax*? This can be answered by looking at the regression's R^2 which states that about 47% of the variation in after tax prices is explained by the variation of the sales tax across states.

```
# inspect the R^2 of the first stage regression
summary(cig_s1)$r.squared
#> [1] 0.4709961
```

We next store $\log(\widehat{P}_i^{cigarettes})$, the fitted values obtained by the first stage regression `cig_s1`, in the variable `lcigp_pred`.

```
# store the predicted values
lcigp_pred <- cig_s1$fitted.values
```

Next, we run the second stage regression which gives us the TOLS estimates we seek.

```
# run the stage 2 regression
cig_s2 <- lm(log(c1995$packs) ~ lcigp_pred)
coeftest(cig_s2, vcov = vcovHC)
```

```
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  9.71988    1.70304  5.7074 7.932e-07 ***
#> lcigp_pred  -1.08359    0.35563 -3.0469 0.003822 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus estimating the model (12.3) using TSLS yields

$$\log(\widehat{Q}_i^{\text{cigarettes}}) = \underset{(1.70)}{9.72} - \underset{(0.36)}{1.08} \log(P_i^{\text{cigarettes}}), \quad (12.4)$$

where we write $\log(P_i^{\text{cigarettes}})$ instead of $\log(\widehat{P}_i^{\text{cigarettes}})$ for consistency with the book.

The function `ivreg()` from the package `AER` carries out TSLS procedure automatically. It is used similarly as `lm()`. Instruments can be added to the usual specification of the regression formula using a vertical bar separating the model equation from the instruments. Thus, for the regression at hand the correct formula is `log(packs) ~ log(rprice) | salestax`.

```
# perform TSLS using 'ivreg()'
cig_ivreg <- ivreg(log(packs) ~ log(rprice) | salestax, data = c1995)

coeftest(cig_ivreg, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  9.71988    1.52832  6.3598 8.346e-08 ***
#> log(rprice) -1.08359    0.31892 -3.3977 0.001411 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that the coefficient estimates coincide for both approaches.

Two notes on the computation of TSLS standard errors

1. We have demonstrated that running the individual regressions for each stage of TSLS using `lm()` leads to the same coefficient estimates as when using `ivreg()`. However, the standard errors reported for the second-stage regression (e.g., `coeftest()` or `summary()`) are *invalid* because they do not account for the use of predictions from the first-stage regression as regressors in the second-stage regression. Fortunately, `ivreg()` performs the necessary adjustment automatically. This is another advantage over manual step-by-step estimation which we have done above for demonstrating the mechanics of the procedure.
2. Just like in multiple regression it is important to compute heteroskedasticity-robust standard errors as we have done above using `vcovHC()`.

The TSLS estimate for β_1 in (12.4) suggests that an increase in cigarette prices by one percent reduces cigarette consumption by roughly 1.08 percentage points, which is fairly elastic. However, we should keep in mind that this estimate might not be trustworthy even though we used IV estimation: there still might be a bias due to omitted variables. Thus a multiple IV regression approach is needed.

12.2 The General IV Regression Model

The simple IV regression model is easily extended to a multiple regression model which we refer to as the general IV regression model. In this model we distinguish between four types of variables: the dependent variable, included exogenous variables, included endogenous variables and instrumental variables. Key Concept 12.1 summarizes the model and the common terminology. See Chapter 12.2 of the book for a more comprehensive discussion of the individual components of the general model.

Key Concept 12.1

The General Instrumental Variables Regression Model and Terminology

$$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \beta_{k+1} W_{1i} + \cdots + \beta_{k+r} W_{ri} + u_i, \quad (12.5)$$

with $i = 1, \dots, n$ is the general instrumental variables regression model where

- Y_i is the dependent variable,
- $\beta_0, \dots, \beta_{k+r}$ are $1 + k + r$ unknown regression coefficients,
- X_{1i}, \dots, X_{ki} are k endogenous regressors ,
- W_{1i}, \dots, W_{ri} are r exogenous regressors which are uncorrelated with u_i ,
- u_i is the error term,
- Z_{1i}, \dots, Z_{mi} are m instrumental variables.

The coefficients are overidentified if $m > k$. If $m < k$, the coefficients are underidentified and when $m = k$ they are exactly identified. For estimation of the IV regression model we require exact identification or overidentification.

While the computation of both stages of TSLS individually in the context of a simple regression model with a single endogenous regressor (12.1) is not a big deal, Key Concept 12.2 clarifies why employing TSLS functions such as `ivreg()` becomes more advantageous when dealing with a larger set of potentially endogenous regressors and instruments.

Estimating regression models with TSLS using multiple instruments by means of `ivreg()` is straightforward. There are, however, some subtleties in correctly specifying the regression formula.

Assume that you want to estimate the model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + W_{1i} + u_i,$$

where X_{1i} and X_{2i} are endogenous regressors that shall be instrumented by Z_{1i} , Z_{2i} and Z_{3i} and W_{1i} is an exogenous regressor. The corresponding data is available in a `data.frame` with column names `y`, `x1`, `x2`, `w1`, `z1`, `z2` and `z3`. It might be tempting to specify the argument `formula` in your call of `ivreg()` as `y ~ x1 + x2 + w1 | z1 + z2 + z3` which is wrong. As explained in the documentation of `ivreg()` (see `?ivreg`), it is necessary to list *all* exogenous variables as instruments too, that is joining them by '+'s on the right of the vertical bar: `y ~ x1 + x2 + w1 | w1 + z1 + z2 + z3` where `w1` is "instrumenting itself".

If there is a large number of exogenous variables it may be convenient to provide an update formula with a `.` (this includes all variables except for the dependent variable) right after the `|` and to exclude all endogenous variables using a `-`. For example, if there is one exogenous regressor `w1` and one endogenous regressor `x1` with instrument `z1`, the appropriate formula would be `y ~ w1 + x1 | w1 + z1` which is equivalent to `y ~ w1 + x1 | . - x1 + z1`.

Key Concept 12.2 Two-Stage Least Squares

Similarly to the simple IV regression model, the general IV model (12.5) can be estimated using the two-stage least squares estimator:

- **First-stage regression(s):**
Run an OLS regression for each of the endogenous variables (X_{1i}, \dots, X_{ki}) on all instrumental variables (Z_{1i}, \dots, Z_{mi}) , all exogenous variables (W_{1i}, \dots, W_{ri}) and an intercept. Compute the fitted values $(\hat{X}_{1i}, \dots, \hat{X}_{ki})$.
- **Second-stage regression:**
Regress the dependent variable on the predicted values of all endogenous regressors, all exogenous variables and an intercept using OLS. This gives $\hat{\beta}_0^{TOLS}, \dots, \hat{\beta}_{k+r}^{TOLS}$, the TSLS estimates of the model coefficients.

In the general IV regression model, the instrument relevance and instrument exogeneity assumptions are the same as in the simple regression model with a single endogenous regressor and only one instrument. See Key Concept 12.3 for a recap using the terminology of general IV regression.

Key Concept 12.3 Two Conditions for Valid Instruments

For Z_{1i}, \dots, Z_{mi} to be a set of valid instruments, the following two conditions must be fulfilled:

1. Instrument Relevance

If there are k endogenous variables, r exogenous variables and $m \geq k$ instruments Z and the $\hat{X}_{1i}^*, \dots, \hat{X}_{ki}^*$ are the predicted values from the k population first stage regressions, it must hold that

$$(\hat{X}_{1i}^*, \dots, \hat{X}_{ki}^*, W_{1i}, \dots, W_{ri}, 1)$$

are not perfectly multicollinear. 1 denotes the constant regressor which equals 1 for all observations.

Note: If there is only one endogenous regressor X_i , there must be at least one non-zero coefficient on the Z and the W in the population regression for this condition to be valid: if all of the coefficients are zero, all the \hat{X}_i^* are just the mean of X such that there is perfect multicollinearity.

2. Instrument Exogeneity

All m instruments must be uncorrelated with the error term,

$$\rho_{Z_{1i}, u_i} = 0, \dots, \rho_{Z_{mi}, u_i} = 0.$$

One can show that if the IV regression assumptions presented in Key Concept 12.4 hold, the TSLS estimator in (12.5) is consistent and normally distributed when the sample size is large. Appendix 12.3 of the book deals with a proof in the special case with a single regressor, a single instrument and no exogenous variables. The reasoning behind this carries over to the general IV model. Chapter 18 of the book proves a more complicated explanation for the general case.

For our purposes it is sufficient to bear in mind that validity of the assumptions stated in Key Concept

12.4 allows us to obtain valid statistical inference using R functions which compute t -Tests, F -Tests and confidence intervals for model coefficients.

Key Concept 12.4 The IV Regression Assumptions

For the general IV regression model in Key Concept 12.1 we assume the following:

1. $E(u_i | W_{1i}, \dots, W_{ri}) = 0$.
2. $(X_{1i}, \dots, X_{ki}, W_{1i}, \dots, W_{ri}, Z_{1i}, \dots, Z_{mi})$ are i.i.d. draws from their joint distribution.
3. All variables have nonzero finite fourth moments, i.e., outliers are unlikely.
4. The Z s are valid instruments (see Key Concept 12.3).

Application to the Demand for Cigarettes

The estimated elasticity of the demand for cigarettes in (12.1) is 1.08. Although (12.1) was estimated using IV regression it is plausible that this IV estimate is biased: in this model, the TSLS estimator is inconsistent for the true β_1 if the instrument (the real sales tax per pack) correlates with the error term. This is likely to be the case since there are economic factors, like state income, which impact the demand for cigarettes and correlate with the sales tax. States with high personal income tend to generate tax revenues by income taxes and less by sales taxes. Consequently, state income should be included in the regression model.

$$\log(Q_i^{\text{cigarettes}}) = \beta_0 + \beta_1 \log(P_i^{\text{cigarettes}}) + \beta_2 \log(\text{income}_i) + u_i \quad (12.6)$$

Before estimating (12.6) using `ivreg()` we define `income` as real per capita income `rincome` and append it to the data set `CigarettesSW`.

```
# add rincome to the dataset
CigarettesSW$rincome <- with(CigarettesSW, income / population / cpi)

c1995 <- subset(CigarettesSW, year == "1995")

# estimate the model
cig_ivreg2 <- ivreg(log(packs) ~ log(rprice) + log(rincome) | log(rincome) +
  salestax, data = c1995)

coeftest(cig_ivreg2, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  9.43066    1.25939   7.4883 1.935e-09 ***
#> log(rprice) -1.14338    0.37230  -3.0711 0.003611 **
#> log(rincome) 0.21452    0.31175   0.6881 0.494917
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain

$$\log(\widehat{Q}_i^{\text{cigarettes}}) = \underset{(1.26)}{9.42} - \underset{(0.37)}{1.14} \log(P_i^{\text{cigarettes}}) + \underset{(0.31)}{0.21} \log(\text{income}_i). \quad (12.7)$$

Following the book we add the cigarette-specific taxes (cigtax_i) as a further instrumental variable and estimate again using TSLS.

```
# add cigtax to the data set
CigarettesSW$cigtax <- with(CigarettesSW, tax/cpi)

c1995 <- subset(CigarettesSW, year == "1995")

# estimate the model
cig_ivreg3 <- ivreg(log(packs) ~ log(rprice) + log(rincome) |
                  log(rincome) + salestax + cigtax, data = c1995)

coeftest(cig_ivreg3, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  9.89496    0.95922  10.3157 1.947e-13 ***
#> log(rprice) -1.27742    0.24961  -5.1177 6.211e-06 ***
#> log(rincome) 0.28040    0.25389   1.1044 0.2753
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the two instruments salestax_i and cigtax_i we have $m = 2$ and $k = 1$ so the coefficient on the endogenous regressor $\log(P_i^{\text{cigarettes}})$ is *overidentified*. The TSLS estimate of (12.6) is

$$\log(\widehat{Q}_i^{\text{cigarettes}}) = \underset{(0.96)}{9.89} - \underset{(0.25)}{1.28} \log(P_i^{\text{cigarettes}}) + \underset{(0.25)}{0.28} \log(\text{income}_i). \quad (12.8)$$

Should we trust the estimates presented in (12.7) or rather rely on (12.8)? The estimates obtained using both instruments are more precise since in (12.8) all standard errors reported are smaller than in (12.7). In fact, the standard error for the estimate of the demand elasticity is only two thirds of the standard error when the sales tax is the only instrument used. This is due to more information being used in estimation (12.8). *If* the instruments are valid, (12.8) can be considered more reliable.

However, without insights regarding the validity of the instruments it is not sensible to make such a statement. This stresses why checking instrument validity is essential. Chapter 12.3 briefly discusses guidelines in checking instrument validity and presents approaches that allow to test for instrument relevance and exogeneity under certain conditions. These are then used in an application to the demand for cigarettes in Chapter 12.4.

12.3 Checking Instrument Validity

Instrument Relevance

Instruments that explain little variation in the endogenous regressor X are called *weak instruments*. Weak instruments provide little information about the variation in X that is exploited by IV regression

to estimate the effect of interest: the estimate of the coefficient on the endogenous regressor is estimated inaccurately. Moreover, weak instruments cause the distribution of the estimator to deviate considerably from a normal distribution even in large samples such that the usual methods for obtaining inference about the true coefficient on X may produce wrong results. See Chapter 12.3 and Appendix 12.4 of the book for a more detailed argument on the undesirable consequences of using weak instruments in IV regression.

Key Concept 12.5

A Rule of Thumb for Checking for Weak Instruments

Consider the case of a single endogenous regressor X and m instruments Z_1, \dots, Z_m . If the coefficients on all instruments in the population first-stage regression of a TSLS estimation are zero, the instruments do not explain any of the variation in the X which clearly violates assumption 1 of Key Concept 12.2. Although the latter case is unlikely to be encountered in practice, we should ask ourselves to what extent the assumption of instrument relevance should be fulfilled.

While this is hard to answer for general IV regression, in the case of a *single* endogenous regressor X one may use the following rule of thumb:

Compute the F -statistic which corresponds to the hypothesis that the coefficients on Z_1, \dots, Z_m are all zero in the first-stage regression. If the F -statistic is less than 10, the instruments are weak such that the TSLS estimate of the coefficient on X is biased and no valid statistical inference about its true value can be made. See also Appendix 12.5 of the book.

The thumb rule of Key Concept 12.5 is easily implemented in R. Run the first-stage regression using `lm()` and subsequently compute the heteroskedasticity-robust F -statistic by means of `linearHypothesis()`. This is part of the application to the demand for cigarettes discussed in Chapter 12.4.

If Instruments are Weak

There are two ways to proceed if instruments are weak:

1. Discard the weak instruments and/or find stronger instruments. While the former is only an option if the unknown coefficients remain identified when the weak instruments are discarded, the latter can be very difficult and even may require a redesign of the whole study.
2. Stick with the weak instruments but use methods that improve upon TSLS in this scenario, for example limited information maximum likelihood estimation, see Appendix 12.5 of the book.

When the Assumption of Instrument Exogeneity is Violated

If there is correlation between an instrument and the error term, IV regression is not consistent (this is shown in Appendix 12.4 of the book). The overidentifying restrictions test (also called the J -test) is an approach to test the hypothesis that *additional* instruments are exogenous. For the J -test to be applicable there need to be *more* instruments than endogenous regressors. The J -test is summarized in Key Concept 12.5.

Key Concept 12.6***J*-Statistic / Overidentifying Restrictions Test**

Take \hat{u}_i^{TOLS} , $i = 1, \dots, n$, the residuals of the TOLS estimation of the general IV regression model 12.5. Run the OLS regression

$$\hat{u}_i^{TOLS} = \delta_0 + \delta_1 Z_{1i} + \dots + \delta_m Z_{mi} + \delta_{m+1} W_{1i} + \dots + \delta_{m+r} W_{ri} + e_i \quad (12.9)$$

and test the joint hypothesis

$$H_0 : \delta_1 = 0, \dots, \delta_m = 0,$$

which states that all instruments are exogenous. This can be done using the corresponding *F*-statistic by computing

$$J = mF.$$

This test is the overidentifying restrictions test and the statistic is called the *J*-statistic with

$$J \sim \chi_{m-k}^2$$

in large samples under the null and the assumption of homoskedasticity. The degrees of freedom $m - k$ state the degree of overidentification since this is the number of instruments m minus the number of endogenous regressors k .

It is important to note that the *J*-statistic discussed in Key Concept 12.6 is only χ_{m-k}^2 distributed when the error term e_i in the regression (12.9) is homoskedastic. A discussion of the heteroskedasticity-robust *J*-statistic is beyond the scope of this chapter. We refer to Section 18.7 of the book for a theoretical argument.

As for the procedure shown in Key Concept 12.6, the application in the next section shows how to apply the *J*-test using `linearHypothesis()`.

12.4 Application to the Demand for Cigarettes

Are the general sales tax and the cigarette-specific tax valid instruments? If not, TOLS is not helpful to estimate the demand elasticity for cigarettes discussed in Chapter 12.2. As discussed in Chapter 12.1, both variables are likely to be relevant but whether they are exogenous is a different question.

The book argues that cigarette-specific taxes could be endogenous because there might be state specific historical factors like economic importance of the tobacco farming and cigarette production industry that lobby for low cigarette specific taxes. Since it is plausible that tobacco growing states have higher rates of smoking than others, this would lead to endogeneity of cigarette specific taxes. If we had data on the size of the tobacco and cigarette industry, we could solve this potential issue by including the information in the regression. Unfortunately, this is not the case.

However, since the role of the tobacco and cigarette industry is a factor that can be assumed to differ across states but not over time we may exploit the panel structure of `CigarettesSW` instead: as shown in Chapter 10.2, regression using data on *changes* between two time periods eliminates such state specific and time invariant effects. Following the book we consider changes in variables between 1985 and 1995. That is, we are interested in estimating the *long-run elasticity* of the demand for cigarettes.

The model to be estimated by TOLS using the general sales tax and the cigarette-specific sales tax as instruments hence is

$$\log(Q_{i,1995}^{cigarettes}) - \log(Q_{i,1985}^{cigarettes}) = \beta_0 + \beta_1 \left[\log(P_{i,1995}^{cigarettes}) - \log(P_{i,1985}^{cigarettes}) \right] + \beta_2 [\log(income_{i,1995}) - \log(income_{i,1985})] + u_i. \quad (12.10)$$

We first create differences from 1985 to 1995 for the dependent variable, the regressors and both instruments.

```
# subset data for year 1985
c1985 <- subset(CigarettesSW, year == "1985")

# define differences in variables
packsdiff <- log(c1995$packs) - log(c1985$packs)

pricediff <- log(c1995$price/c1995$cpi) - log(c1985$price/c1985$cpi)

incomediff <- log(c1995$income/c1995$population/c1995$cpi) -
log(c1985$income/c1985$population/c1985$cpi)

salestaxdiff <- (c1995$taxes - c1995$tax)/c1995$cpi -
(c1985$taxes - c1985$tax)/c1985$cpi

cigtaxdiff <- c1995$tax/c1995$cpi - c1985$tax/c1985$cpi
```

We now perform three different IV estimations of (12.10) using `ivreg()`:

1. TSLS using only the difference in the sales taxes between 1985 and 1995 as the instrument.
2. TSLS using only the difference in the cigarette-specific sales taxes 1985 and 1995 as the instrument.
3. TSLS using both the difference in the sales taxes 1985 and 1995 and the difference in the cigarette-specific sales taxes 1985 and 1995 as instruments.

```
# estimate the three models
cig_ivreg_diff1 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff +
salestaxdiff)

cig_ivreg_diff2 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff +
cigtaxdiff)

cig_ivreg_diff3 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff +
salestaxdiff + cigtaxdiff)
```

As usual we use `coeftest()` in conjunction with `vcovHC()` to obtain robust coefficient summaries for all models.

```
# robust coefficient summary for 1.
coeftest(cig_ivreg_diff1, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>
#> Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.117962 0.068217 -1.7292 0.09062 .
#> pricediff -0.938014 0.207502 -4.5205 4.454e-05 ***
#> incomediff 0.525970 0.339494 1.5493 0.12832
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# robust coefficient summary for 2.
coeftest(cig_ivreg_diff2, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.017049  0.067217 -0.2536  0.8009
#> pricediff   -1.342515  0.228661 -5.8712 4.848e-07 ***
#> incomediff   0.428146  0.298718  1.4333  0.1587
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# robust coefficient summary for 3.
coeftest(cig_ivreg_diff3, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.052003  0.062488 -0.8322  0.4097
#> pricediff   -1.202403  0.196943 -6.1053 2.178e-07 ***
#> incomediff   0.462030  0.309341  1.4936  0.1423
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We proceed by generating a tabulated summary of the estimation results using `stargazer()`.

```
# gather robust standard errors in a list
rob_se <- list(sqrt(diag(vcovHC(cig_ivreg_diff1, type = "HC1"))),
               sqrt(diag(vcovHC(cig_ivreg_diff2, type = "HC1"))),
               sqrt(diag(vcovHC(cig_ivreg_diff3, type = "HC1"))))

# generate table
stargazer(cig_ivreg_diff1, cig_ivreg_diff2, cig_ivreg_diff3,
          header = FALSE,
          type = "html",
          omit.table.layout = "n",
          digits = 3,
          column.labels = c("IV: salestax", "IV: cigtax", "IVs: salestax, cigtax"),
          dep.var.labels.include = FALSE,
          dep.var.caption = "Dependent Variable: 1985-1995 Difference in Log per Pack Price",
          se = rob_se)
```

Table 12.1 reports negative estimates of the coefficient on `pricediff` that are quite different in magnitude. Which one should we trust? This hinges on the validity of the instruments used. To assess this we compute F -statistics for the first-stage regressions of all three models to check instrument relevance.

```
# first-stage regressions
mod_relevance1 <- lm(pricediff ~ salestaxdiff + incomediff)
mod_relevance2 <- lm(pricediff ~ cigtaxdiff + incomediff)
```


Table 12.1: TSLS Estimates of the Long-Term Elasticity of the Demand for Cigarettes using Panel Data

	Dependent variable: 1985-1995 difference in log per pack price		
	IV: salestax	IV: cigtax	IVs: salestax, cigtax
	(1)	(2)	(3)
pricediff	-0.938***	-1.343	-1.202
incomediff	0.526***	0.428	0.462
Constant	-0.118*** (0.028)	-0.017 (0.484)	-0.052 (0.250)
Observations	48	48	48
R ²	0.550	0.520	0.547
Adjusted R ²	0.530	0.498	0.526
Residual Std. Error (df = 45)	0.091	0.094	0.091

```
mod_relevance3 <- lm(pricediff ~ incomediff + salestaxdiff + cigtaxdiff)
```

```
# check instrument relevance for model (1)
linearHypothesis(mod_relevance1,
                  "salestaxdiff = 0",
                  vcov = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> salestaxdiff = 0
#>
#> Model 1: restricted model
#> Model 2: pricediff ~ salestaxdiff + incomediff
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df    F   Pr(>F)
#> 1     46
#> 2     45  1 28.445 3.009e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# check instrument relevance for model (2)
linearHypothesis(mod_relevance2,
                  "cigtaxdiff = 0",
                  vcov = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> cigtaxdiff = 0
#>
#> Model 1: restricted model
```

```

#> Model 2: pricediff ~ cigtaxdiff + incomediff
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F   Pr(>F)
#> 1      46
#> 2      45  1 98.034 7.09e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# check instrument relevance for model (3)
linearHypothesis(mod_relevance3,
                 c("salestaxdiff = 0", "cigtaxdiff = 0"),
                 vcov = vcovHC, type = "HC1")
#> Linear hypothesis test
#>
#> Hypothesis:
#> salestaxdiff = 0
#> cigtaxdiff = 0
#>
#> Model 1: restricted model
#> Model 2: pricediff ~ incomediff + salestaxdiff + cigtaxdiff
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F   Pr(>F)
#> 1      46
#> 2      44  2 76.916 4.339e-15 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

We also conduct the overidentifying restrictions test for model three which is the only model where the coefficient on the difference in log prices is overidentified ($m = 2$, $k = 1$) such that the J -statistic can be computed. To do this we take the residuals stored in `cig_ivreg_diff3` and regress them on both instruments and the presumably exogenous regressor `incomediff`. We again use `linearHypothesis()` to test whether the coefficients on both instruments are zero which is necessary for the exogeneity assumption to be fulfilled. Note that with `test = "Chisq"` we obtain a chi-squared distributed test statistic instead of an F -statistic.

```

# compute the J-statistic
cig_iv_OR <- lm(residuals(cig_ivreg_diff3) ~ incomediff + salestaxdiff + cigtaxdiff)

cig_OR_test <- linearHypothesis(cig_iv_OR,
                              c("salestaxdiff = 0", "cigtaxdiff = 0"),
                              test = "Chisq")

cig_OR_test
#> Linear hypothesis test
#>
#> Hypothesis:
#> salestaxdiff = 0
#> cigtaxdiff = 0
#>

```

```
#> Model 1: restricted model
#> Model 2: residuals(cig_ivreg_diff3) ~ incomediff + salestaxdiff + cigtaxdiff
#>
#>   Res.Df    RSS Df Sum of Sq Chisq Pr(>Chisq)
#> 1     46 0.37472
#> 2     44 0.33695  2  0.037769 4.932   0.08492 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Caution: In this case the p -Value reported by `linearHypothesis()` is wrong because the degrees of freedom are set to 2. This differs from the degree of overidentification ($m - k = 2 - 1 = 1$) so the J -statistic is χ_1^2 distributed instead of following a χ_2^2 distribution as assumed by default in `linearHypothesis()`. We may compute the correct p -Value using `pchisq()`.

```
# compute correct p-value for J-statistic
pchisq(cig_OR_test[2, 5], df = 1, lower.tail = FALSE)
#> [1] 0.02636406
```

Since this value is smaller than 0.05 we reject the hypothesis that both instruments are exogenous at the level of 5%. This means one of the following:

1. The sales tax is an invalid instrument for the per-pack price.
2. The cigarettes-specific sales tax is an invalid instrument for the per-pack price.
3. Both instruments are invalid.

The book argues that the assumption of instrument exogeneity is more likely to hold for the general sales tax (see Chapter 12.4 of the book) such that the IV estimate of the long-run elasticity of demand for cigarettes we consider the most trustworthy is -0.94 , the TSLS estimate obtained using the general sales tax as the only instrument.

The interpretation of this estimate is that over a 10-year period, an increase in the average price per package by one percent is expected to decrease consumption by about 0.94 percentage points. This suggests that, in the long run, price increases can reduce cigarette consumption considerably.

12.5 Where Do Valid Instruments Come From?

Chapter 12.5 of the book presents a comprehensive discussion of approaches to find valid instruments in practice by the example of three research questions:

- Does putting criminals in jail reduce crime?
- Does cutting class sizes increase test scores?
- Does aggressive treatment of heart attacks prolong lives?

This section is not directly related to applications in R which is why we do not discuss the contents here. We encourage you to work through this on your own.

Summary

`ivreg()` from the package `AER` provides convenient functionalities to estimate IV regression models in R. It is an implementation of the TSLS estimation approach.

Besides treating IV estimation, we have also discussed how to test for weak instruments and how to conduct an overidentifying restrictions test when there are more instruments than endogenous regressors using R.

An empirical application has shown how `ivreg()` can be used to estimate the long-run elasticity of demand for cigarettes based on `CigarettesSW`, a panel data set on cigarette consumption and economic indicators for all 48 continental U.S. states for 1985 and 1995. Different sets of instruments were used and it has been argued why using the general sales tax as the only instrument is the preferred choice. The estimate of the demand elasticity deemed the most trustworthy is -0.94 . This estimate suggests that there is a remarkable negative long-run effect on cigarette consumption of increasing prices.

12.6 Exercises

Chapter 13

Experiments and Quasi-Experiments

This chapter discusses statistical tools that are commonly applied in program evaluation, where interest lies in measuring the causal effects of programs, policies or other interventions. An optimal research design for this purpose is what statisticians call an ideal randomized controlled experiment. The basic idea is to randomly assign subjects to two different groups, one that receives the treatment (the treatment group) and one that does not (the control group) and to compare outcomes for both groups in order to get an estimate of the average treatment effect.

Such *experimental* data is fundamentally different from *observational* data. For example, one might use a randomized controlled experiment to measure how much the performance of students in a standardized test differs between two classes where one has a “regular” student-teacher ratio and the other one has fewer students. The data produced by such an experiment would be different from, e.g., the observed cross-section data on the students’ performance used throughout Chapters 4 to 8 where class sizes are not randomly assigned to students but instead are the results of an economic decision where educational objectives and budgetary aspects were balanced.

For economists, randomized controlled experiments are often difficult or even infeasible to implement. For example, due to ethic, moral and legal reasons it is practically impossible for a business owner to estimate the causal effect on the productivity of workers of setting them under psychological stress using an experiment where workers are randomly assigned either to the treatment group that is under time pressure or to the control group where work is under regular conditions, at best without knowledge of being in an experiment (see the box *The Hawthorne Effect* on p. 528 of the book).

However, sometimes external circumstances produce what is called a *quasi-experiment* or *natural experiment*. This “as if” randomness allows for estimation of causal effects that are of interest for economists using tools which are very similar to those valid for ideal randomized controlled experiments. These tools draw heavily on the theory of multiple regression and also on IV regression (see Chapter 12). We will review the core aspects of these methods and demonstrate how to apply them in R using the STAR data set (see the description of the data set).

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter:

- `AER` (Kleiber and Zeileis, 2008b),
- `dplyr` (Wickham et al., 2023a),
- `MASS` (Ripley, 2023),
- `mvtnorm` (Genz et al., 2023),
- `rddtools` (Stigler and Quast, 2022),
- `scales` (Wickham and Seidel, 2022),

- `stargazer` (Hlavac, 2022),
- `tidyr` (Wickham et al., 2023b).

Make sure the following code chunk runs without any errors.

```
library(AER)
library(dplyr)
library(MASS)
library(mvtnorm)
library(rddtools)
library(scales)
library(stargazer)
library(tidyr)
```

13.1 Potential Outcomes, Causal Effects and Idealized Experiments

We now briefly recap the idea of the average causal effect and how it can be estimated using the *differences estimator*. We advise you to work through Chapter 13.1 of the book for a better understanding.

Potential Outcomes and the average causal effect

A *potential outcome* is the outcome for an individual under a potential treatment. For this individual, the causal effect of the treatment is the difference between the potential outcome if the individual receives the treatment and the potential outcome if she does not. Since this causal effect may be different for different individuals and it is not possible to measure the causal effect for a single individual, one is interested in studying the *average causal effect* of the treatment, hence also called the *average treatment effect*.

In an ideal randomized controlled experiment the following conditions are fulfilled:

1. The subjects are selected at random from the population.
2. The subjects are randomly assigned to treatment and control group.

Condition 1 guarantees that the subjects' potential outcomes are drawn randomly from the same population distribution such that the expected value of the causal effect in the sample is equal to the average causal effect in the population. Condition 2 ensures that the receipt of treatment is independent from the subjects' potential outcomes. If both conditions are fulfilled, the expected causal effect is the expected outcome in the treatment group minus the expected outcome in the control group. Using conditional expectations we have

$$\text{Average causal effect} = E(Y_i | X_i = 1) - E(Y_i | X_i = 0),$$

where X_i is a binary treatment indicator.

The average causal effect can be estimated using the *differences estimator*, which is nothing but the OLS estimator in the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i \quad , \quad i = 1, \dots, n, \quad (13.1)$$

where random assignment ensures that $E(u_i | X_i) = 0$.

The OLS estimator in the regression model

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_{1i} + \dots + \beta_{1+r} W_{ri} + u_i \quad , \quad i = 1, \dots, n \quad (13.2)$$

with additional regressors W_1, \dots, W_r is called the *differences estimator with additional regressors*. It is assumed that treatment assignment X_i is random and independent of the pretreatment characteristic W_i . This assumption is called *conditional mean independence* and implies

$$E(u_i|X_i, W_i) = E(u_i|W_i) = 0,$$

so the conditional expectation of the error u_i given the treatment indicator X_i and the pretreatment characteristic W_i does not depend on the X_i . Conditional mean independence replaces the first least squares assumption in Key Concept 6.4 and thus ensures that the differences estimator of β_1 is unbiased. The *differences estimator with additional regressors* is more efficient than the *differences estimator* if the additional regressors explain some of the variation in the Y_i .

13.2 Threats to Validity of Experiments

The concepts of internal and external validity discussed in Key Concept 9.1 are also applicable for studies based on experimental and quasi-experimental data. Chapter 13.2 of the book provides a thorough explanation of the particular threats to internal and external validity of experiments including examples. We limit ourselves to a short repetition of the threats listed there. Consult the book for a more detailed explanation.

Threats to Internal Validity

1. Failure to Randomize

If the subjects are not randomly assigned to the treatment group, then the outcomes will be contaminated with the effect of the subjects' individual characteristics or preferences and it is not possible to obtain an unbiased estimate of the treatment effect. One can test for nonrandom assignment using a significance test (*F*-Test) on the coefficients in the regression model

$$X_i = \beta_0 + \beta_1 W_{1i} + \dots + \beta_r W_{ri} + u_i, \quad i = 1, \dots, n.$$

2. Failure to Follow the Treatment Protocol

If subjects do not follow the treatment protocol, i.e., some subjects in the treatment group manage to avoid receiving the treatment and/or some subjects in the control group manage to receive the treatment (*partial compliance*), there is correlation between X_i and u_i such that the OLS estimator of the average treatment effect will be biased. If there are data on *both* treatment actually received (X_i) and initial random assignment (Z_i), IV regression of the models (13.1) and (13.2) is a remedy.

3. Attrition

Attrition may result in a nonrandomly selected sample. If subjects systematically drop out of the study after being assigned to the control or the treatment group (systematic means that the reason of the dropout is related to the treatment) there will be correlation between X_i and u_i and hence bias in the OLS estimator of the treatment effect.

4. Experimental Effects

If human subjects in treatment group and/or control group know that they are in an experiment, they might adapt their behaviour in a way that prevents unbiased estimation of the treatment effect.

5. Small Sample Sizes

As we know from the theory of linear regression, small sample sizes lead to imprecise estimation of the coefficients and thus imply imprecise estimation of the causal effect. Furthermore, confidence intervals and hypothesis test may produce wrong inference when the sample size is small.

Threats to External Validity

1. Nonrepresentative Sample

If the population studied and the population of interest are not sufficiently similar, there is no justification in generalizing the results.

2. Nonrepresentative Program or Policy

If the program or policy for the population studied differs considerably from the program (to be) applied to population(s) of interest, the results cannot be generalized. For example, a small-scale program with low funding might have different effects than a widely available scaled-up program that is actually implemented. There are other factors like duration and the extent of monitoring that should be considered here.

3. General Equilibrium Effects

If market and/or environmental conditions cannot be kept constant when an internally valid program is implemented broadly, external validity may be doubtful.

13.3 Experimental Estimates of the Effect of Class Size Reductions

Experimental Design and the Data Set

The Project *Student-Teacher Achievement Ratio* (STAR) was a large randomized controlled experiment with the aim of asserting whether a class size reduction is effective in improving education outcomes. It was conducted in 80 Tennessee elementary schools over a period of four years during the 1980s by the State Department of Education.

In the first year, about 6400 students were randomly assigned into one of three interventions: small class (13 to 17 students per teacher), regular class (22 to 25 students per teacher), and regular-with-aide class (22 to 25 students with a full-time teacher's aide). Teachers were also randomly assigned to the classes they taught. The interventions were initiated as the students entered school in kindergarten and continued through to third grade. Control and treatment groups across grades are summarized in Table 13.1.

Table 13.1: Control and treatment groups in the STAR experiment

	K	1	2	3
Treatment 1	Small class	Small class	Small class	Small class
Treatment 2	Regular class + aide	Regular class + aide	Regular class + aide	Regular class + aide
Control	Regular class	Regular class	Regular class	Regular class

Each year, the students' learning progress was assessed using the sum of the points scored on the math and reading parts of a standardized test (the Stanford Achievement Test).

The STAR data set is part of the package **AER**.


```
# load the package AER and the STAR dataset
library(AER)
data(STAR)
```

`head(STAR)` shows that there is a variety of factor variables that describe student and teacher characteristics as well as various school indicators, all of which are separately recorded for the four different grades. The data is in *wide format*. That is, each variable has its own column and for each student, the rows contain observations on these variables. Using `dim(STAR)` we find that there are a total of 11598 observations on 47 variables.

```
# get an overview
head(STAR, 2)
#>   gender ethnicity  birth stark star1 star2  star3 readk read1 read2 read3
#> 1122 female      afam 1979 Q3 <NA> <NA> <NA> regular NA NA NA 580
#> 1137 female      cauc 1980 Q1 small small small small 447 507 568 587
#>   mathk math1 math2 math3 lunchk lunch1 lunch2 lunch3 schoolk school1
#> 1122 NA NA NA 564 <NA> <NA> <NA> free <NA> <NA>
#> 1137 473 538 579 593 non-free free non-free free rural rural
#>   school2 school3 degreek degree1 degree2 degree3 ladderk ladder1
#> 1122 <NA> suburban <NA> <NA> <NA> bachelor <NA> <NA>
#> 1137 rural rural bachelor bachelor bachelor bachelor level1 level1
#>   ladder2 ladder3 experiencek experience1 experience2 experience3
#> 1122 <NA> level1 NA NA NA 30
#> 1137 apprentice apprentice 7 7 3 1
#>   tethnicityk tethnicity1 tethnicity2 tethnicity3 systemk system1 system2
#> 1122 <NA> <NA> <NA> cauc <NA> <NA> <NA>
#> 1137 cauc cauc cauc cauc 30 30 30
#>   system3 schoolidk schoolid1 schoolid2 schoolid3
#> 1122 22 <NA> <NA> <NA> 54
#> 1137 30 63 63 63 63
dim(STAR)
#> [1] 11598 47
```

```
# get variable names
names(STAR)
#> [1] "gender" "ethnicity" "birth" "stark" "star1"
#> [6] "star2" "star3" "readk" "read1" "read2"
#> [11] "read3" "mathk" "math1" "math2" "math3"
#> [16] "lunchk" "lunch1" "lunch2" "lunch3" "schoolk"
#> [21] "school1" "school2" "school3" "degreek" "degree1"
#> [26] "degree2" "degree3" "ladderk" "ladder1" "ladder2"
#> [31] "ladder3" "experiencek" "experience1" "experience2" "experience3"
#> [36] "tethnicityk" "tethnicity1" "tethnicity2" "tethnicity3" "systemk"
#> [41] "system1" "system2" "system3" "schoolidk" "schoolid1"
#> [46] "schoolid2" "schoolid3"
```

A majority of the variable names contain a suffix (k, 1, 2 or 3) stating the grade which the respective variable is referring to. This facilitates regression analysis because it allows to adjust the `formula` argument in `lm()` for each grade by simply changing the variables' suffixes accordingly.

The outcome produced by `head()` shows that some recorded values are `NA` and thus, there is no data on this variable for the student under consideration. This lies in the nature of the data: for example, take the first observation `STAR[1,]`.

In the output of `head(STAR, 2)` we find that the student entered the experiment in third grade in a regular class, which is why the class size is recorded in `star3` and the other class type indicator variables are NA. Her math and reading scores for the third grade are available; however, recordings for other grades are not present for the same reason. Obtaining only her non-missing (non-NA) recordings is straightforward: simply eliminate the NAs using the `!is.na()` function.

```
# drop NA recordings for the first observation and print to the console
STAR[1, !is.na(STAR[1, ])]
#>   gender ethnicity  birth  star3 read3 math3 lunch3  school3  degree3
#> 1122 female      afam 1979 Q3 regular  580  564   free suburban bachelor
#>      ladder3 experience3 tethnicity3 system3 schoolid3
#> 1122 level1      30      cauc      22      54
```

`is.na(STAR[1,])` returns a logical vector with TRUE at positions that correspond to <NA> entries for the first observation. The `!` operator is used to invert the result such that we obtain only non-<NA> entries for the first observations.

In general it is not necessary to remove rows with missing data because `lm()` does so by default. Missing data may imply a small sample size and thus may lead to imprecise estimation and wrong inference. This is, however, not an issue for the study at hand since, as we will see below, sample sizes lie beyond 5000 observations for each regression conducted.

Analysis of the STAR Data

As can be seen from Table 13.1 that there are two treatment groups in each grade, small classes with only 13 to 17 students and regular classes with 22 to 25 students and a teaching aide. Thus, two binary variables, each being an indicator for the respective treatment group, are introduced for the differences estimator to capture the treatment effect for each treatment group separately. This yields the population regression model

$$Y_i = \beta_0 + \beta_1 \text{SmallClass}_i + \beta_2 \text{RegAide}_i + u_i, \quad (13.3)$$

with test score Y_i which is the small class indicator SmallClass_i and RegAide_i which is the indicator for a regular class with aide.

We reproduce the results presented in Table 13.1 of the book by performing the regression (13.3) for each grade separately. For each student, the dependent variable is simply the sum of the points scored in the math and reading parts, constructed using `I()`.

```
# compute differences Estimates for each grades
fmk <- lm(I(readk + mathk) ~ stark, data = STAR)
fm1 <- lm(I(read1 + math1) ~ star1, data = STAR)
fm2 <- lm(I(read2 + math2) ~ star2, data = STAR)
fm3 <- lm(I(read3 + math3) ~ star3, data = STAR)

# obtain coefficient matrix using robust standard errors
coeftest(fmk, vcov = vcovHC, type= "HC1")
#>
#> t test of coefficients:
#>
#>              Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)    918.04289    1.63339  562.0473 < 2.2e-16 ***
#> starksmall      13.89899    2.45409   5.6636 1.554e-08 ***
#> starkregular+aide  0.31394    2.27098   0.1382  0.8901
#> ---
```

```

#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coeftest(fm1, vcov = vcovHC, type= "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)    1039.3926    1.7846 582.4321 < 2.2e-16 ***
#> star1small      29.7808     2.8311  10.5190 < 2.2e-16 ***
#> star1regular+aide  11.9587     2.6520   4.5093 6.62e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coeftest(fm2, vcov = vcovHC, type= "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)    1157.8066    1.8151 637.8820 < 2.2e-16 ***
#> star2small      19.3944     2.7117   7.1522 9.55e-13 ***
#> star2regular+aide  3.4791     2.5447   1.3672  0.1716
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coeftest(fm3, vcov = vcovHC, type= "HC1")
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)    1228.50636    1.68001 731.2483 < 2.2e-16 ***
#> star3small      15.58660     2.39604   6.5051 8.393e-11 ***
#> star3regular+aide -0.29094     2.27271  -0.1280  0.8981
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

We gather the results and present them in a table using `stargazer()`.

```

# compute robust standard errors for each model and gather them in a list
rob_se_1 <- list(sqrt(diag(vcovHC(fmk, type = "HC1"))),
                 sqrt(diag(vcovHC(fm1, type = "HC1"))),
                 sqrt(diag(vcovHC(fm2, type = "HC1"))),
                 sqrt(diag(vcovHC(fm3, type = "HC1"))))

```

```
library(stargazer)
```

```

stargazer(fmk, fm1, fm2, fm3,
  title = "Project STAR: Differences Estimates",
  header = FALSE,
  type = "latex",
  model.numbers = F,
  omit.table.layout = "n",
  digits = 3,
  column.labels = c("K", "1", "2", "3"),
  dep.var.caption = "Dependent Variable: Grade",
  dep.var.labels.include = FALSE,

```

```
se = rob_se_1)
```

The estimates presented in Table 13.2 suggest that the class size reduction improves student performance. Except for grade 1, the estimates of the coefficient on *SmallClass* are roughly of the same magnitude (the estimates lie between 13.90 and 19.39 points) and they are statistically significant at 1%. Furthermore, a teaching aide has little, possibly zero, effect on the performance of the students.

Following the book, we augment the regression model (13.3) by different sets of regressors for two reasons:

1. If the additional regressors explain some of the observed variation in the dependent variable, we obtain more efficient estimates of the coefficients of interest.
2. If the treatment is not received at random due to failures to follow the treatment protocol (see Chapter 13.3 of the book), the estimates obtained using (13.3) may be biased. Adding additional regressors may solve or mitigate this problem.

In particular, we consider the following student and teacher characteristics

- *experience* — Teacher's years of experience
- *boy* — Student is a boy (dummy)
- *lunch* — Free lunch eligibility (dummy)
- *black* — Student is African-American (dummy)
- *race* — Student's race is other than black or white (dummy)
- *schoolid* — School indicator variables

in the four population regression specifications

$$Y_i = \beta_0 + \beta_1 \text{SmallClass}_i + \beta_2 \text{RegAide}_i + u_i, \quad (13.4)$$

$$Y_i = \beta_0 + \beta_1 \text{SmallClass}_i + \beta_2 \text{RegAide}_i + \beta_3 \text{experience}_i + u_i, \quad (13.5)$$

$$Y_i = \beta_0 + \beta_1 \text{SmallClass}_i + \beta_2 \text{RegAide}_i + \beta_3 \text{experience}_i + \text{schoolid} + u_i, \quad (13.6)$$

and

$$Y_i = \beta_0 + \beta_1 \text{SmallClass}_i + \beta_2 \text{RegAide}_i + \beta_3 \text{experience}_i + \beta_4 \text{boy} + \beta_5 \text{lunch} \quad (13.7)$$

$$+ \beta_6 \text{black} + \beta_7 \text{race} + \text{schoolid} + u_i. \quad (13.8)$$

Prior to estimation, we perform some subsetting and data wrangling using functions from the packages `dplyr` and `tidyr`. These are both part of `tidyverse`, a collection of R packages designed for data science and handling big datasets (see the official site for more on `tidyverse` packages). The functions `%>%`, `transmute()` and `mutate()` are sufficient for us here:

- `%>%` allows to chain function calls.
- `transmute()` allows to subset the data set by naming the variables to be kept.
- `mutate()` is convenient for adding new variables based on existing ones while preserving the latter.

The regression models (13.4) to (13.8) require the variables `gender`, `ethnicity`, `stark`, `readk`, `mathk`, `lunchk`, `experiencek` and `schoolidk`. After dropping the remaining variables using `transmute()`, we use `mutate()` to add three additional binary variables which are derivatives of existing ones: `black`, `race` and `boy`. They are generated using logical statements within the function `ifelse()`.

```
# load packages 'dplyr' and 'tidyr' for data wrangling functionalities
library(dplyr)
library(tidyr)
```

Table 13.2: Project STAR - Differences Estimates

	Dependent Variable: Grade			
	K	1	2	3
starksmall	13.899*** (2.454)			
starkregular+aide	0.314 (2.271)			
star1small		29.781*** (2.831)		
star1regular+aide		11.959*** (2.652)		
star2small			19.394*** (2.712)	
star2regular+aide			3.479 (2.545)	
star3small				15.587*** (2.396)
star3regular+aide				-0.291 (2.273)
Constant	918.043*** (1.633)	1,039.393*** (1.785)	1,157.807*** (1.815)	1,228.506*** (1.680)
Observations	5,786	6,379	6,049	5,967
R ²	0.007	0.017	0.009	0.010
Adjusted R ²	0.007	0.017	0.009	0.010
Residual Std. Error	73.490 (df = 5783)	90.501 (df = 6376)	83.694 (df = 6046)	72.910 (df = 5964)
F Statistic	21.263*** (df = 2; 5783)	56.341*** (df = 2; 6376)	28.707*** (df = 2; 6046)	30.250*** (df = 2; 5964)

```

# generate subset with kindergarten data
STARK <- STAR %>%
  transmute(gender,
            ethnicity,
            stark,
            readk,
            mathk,
            lunchk,
            experiencek,
            schoolidk) %>%
  mutate(black = ifelse(ethnicity == "afam", 1, 0),
         race = ifelse(ethnicity == "afam" | ethnicity == "cauc", 1, 0),
         boy = ifelse(gender == "male", 1, 0))

# estimate the models
gradeK1 <- lm(I(mathk + readk) ~ stark + experiencek,
             data = STARK)

gradeK2 <- lm(I(mathk + readk) ~ stark + experiencek + schoolidk,
             data = STARK)

gradeK3 <- lm(I(mathk + readk) ~ stark + experiencek + boy + lunchk
             + black + race + schoolidk,
             data = STARK)

```

For brevity, we exclude the coefficients for the indicator dummies in the output of the `coeftest()` by subsetting the matrices.

```

# obtain robust inference on the significance of coefficients
coeftest(gradeK1, vcov. = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>
#>           Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)    904.72124    2.22235  407.1020 < 2.2e-16 ***
#> starksmall      14.00613    2.44704   5.7237 1.095e-08 ***
#> starkregular+aide -0.60058    2.25430  -0.2664  0.7899
#> experiencek     1.46903    0.16929   8.6778 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coeftest(gradeK2, vcov. = vcovHC, type = "HC1")[1:4, ]
#>
#>           Estimate Std. Error  t value  Pr(>|t|)
#> (Intercept)    925.6748750    7.6527218 120.9602155 0.000000e+00
#> starksmall      15.9330822    2.2411750   7.1092540 1.310324e-12
#> starkregular+aide  1.2151960    2.0353415   0.5970477 5.504993e-01
#> experiencek     0.7431059    0.1697619   4.3773429 1.222880e-05
coeftest(gradeK3, vcov. = vcovHC, type = "HC1")[1:7, ]
#>
#>           Estimate Std. Error  t value  Pr(>|t|)
#> (Intercept)    937.6831330   14.3726687  65.2407117 0.000000e+00
#> starksmall      15.8900507    2.1551817   7.3729516 1.908960e-13
#> starkregular+aide  1.7869378    1.9614592   0.9110247 3.623211e-01
#> experiencek     0.6627251    0.1659298   3.9940097 6.578846e-05

```

```
#> boy          -12.0905123  1.6726331  -7.2284306  5.533119e-13
#> lunchkfree   -34.7033021  1.9870366  -17.4648529  1.437931e-66
#> black        -25.4305130  3.4986918  -7.2685776  4.125252e-13
```

We now use `stargazer()` to gather all relevant information in a structured table.

```
# compute robust standard errors for each model and gather them in a list
rob_se_2 <- list(sqrt(diag(vcovHC(fmk, type = "HC1"))),
                 sqrt(diag(vcovHC(gradeK1, type = "HC1"))),
                 sqrt(diag(vcovHC(gradeK2, type = "HC1"))),
                 sqrt(diag(vcovHC(gradeK3, type = "HC1"))))
```

```
stargazer(fmk, gradeK1, gradeK2, gradeK3,
          title = "Project STAR - Differences Estimates with
                  Additional Regressors for Kindergarten",
          header = FALSE,
          type = "latex",
          model.numbers = F,
          omit.table.layout = "n",
          digits = 3,
          column.labels = c("(1)", "(2)", "(3)", "(4)"),
          dep.var.caption = "Dependent Variable: Test Score in Kindergarten",
          dep.var.labels.include = FALSE,
          se = rob_se_2)
```

The results in column (1) of Table 13.3 same as the results obtained for (13.3). Columns (2) to (4) reveal that adding student characteristics and school fixed effects does not lead to substantially different estimates of the treatment effects. This result makes it more plausible that the estimates of the effects obtained using model (13.3) do not suffer from failure of random assignment. There is some decrease in the standard errors and some increase in \bar{R}^2 , implying that the estimates are more precise.

Because teachers were randomly assigned to classes, inclusion of school fixed effect allows us to estimate the causal effect of a teacher's experience on test scores of students in kindergarten. Regression (3) predicts the average effect of 10 years experience on test scores to be $10 \cdot 0.74 = 7.4$ points. Be aware that the other estimates on student characteristics in regression (4) *do not* have causal interpretation due to nonrandom assignment (see Chapter 13.3 of the book for a detailed discussion).

Are the estimated effects presented in Table 13.3 large or small in a practical sense? Let us translate the predicted changes in test scores to units of standard deviation in order to allow for a comparison (see Section 9.4 for a similar argument).

```
# compute the sample standard deviations of test scores
SSD <- c("K" = sd(na.omit(STAR$readk + STAR$mathk)),
         "1" = sd(na.omit(STAR$read1 + STAR$math1)),
         "2" = sd(na.omit(STAR$read2 + STAR$math2)),
         "3" = sd(na.omit(STAR$read3 + STAR$math3)))

# translate the effects of small classes to standard deviations
Small <- c("K" = as.numeric(coef(fmk)[2]/SSD[1]),
          "1" = as.numeric(coef(fm1)[2]/SSD[2]),
          "2" = as.numeric(coef(fm2)[2]/SSD[3]),
          "3" = as.numeric(coef(fm3)[2]/SSD[4]))

# adjust the standard errors
```

Table 13.3: Project STAR - Differences Estimates with Additional Regressors for Kindergarten

	Dependent Variable: Test Score in Kindergarten			
	(1)	(2)	(3)	(4)
starksmall	13.899*** (2.454)	14.006*** (2.447)	15.933*** (2.241)	15.890*** (2.155)
starkregular+aide	0.314 (2.271)	-0.601 (2.254)	1.215 (2.035)	1.787 (1.961)
experiencek		1.469*** (0.169)	0.743*** (0.170)	0.663*** (0.166)
boy				-12.091*** (1.673)
lunchfree				-34.703*** (1.987)
black				-25.431*** (3.499)
race				8.501 (12.520)
Constant	918.043*** (1.633)	904.721*** (2.222)	925.675*** (7.653)	937.683*** (14.373)
School indicators?				
Observations	no 5,786	no 5,766	yes 5,766	yes 5,748
R ²	0.007	0.020	0.234	0.291
Adjusted R ²	0.007	0.020	0.223	0.281
Residual Std. Error	73.490 (df = 5783)	73.085 (df = 5762)	65.075 (df = 5684)	62.663 (df = 5662)
F Statistic	21.263*** (df = 2; 5783)	39.861*** (df = 3; 5762)	21.413*** (df = 81; 5684)	27.364*** (df = 85; 5662)


```

SmallSE <- c("K" = as.numeric(rob_se_1[[1]][2]/SSD[1]),
            "1" = as.numeric(rob_se_1[[2]][2]/SSD[2]),
            "2" = as.numeric(rob_se_1[[3]][2]/SSD[3]),
            "3" = as.numeric(rob_se_1[[4]][2]/SSD[4]))

# translate the effects of regular classes with aide to standard deviations
RegAide<- c("K" = as.numeric(coef(fmk)[3]/SSD[1]),
           "1" = as.numeric(coef(fm1)[3]/SSD[2]),
           "2" = as.numeric(coef(fm2)[3]/SSD[3]),
           "3" = as.numeric(coef(fm3)[3]/SSD[4]))

# adjust the standard errors
RegAideSE <- c("K" = as.numeric(rob_se_1[[1]][3]/SSD[1]),
              "1" = as.numeric(rob_se_1[[2]][3]/SSD[2]),
              "2" = as.numeric(rob_se_1[[3]][3]/SSD[3]),
              "3" = as.numeric(rob_se_1[[4]][3]/SSD[4]))

# gather the results in a data.frame and round
df <- t(round(data.frame(
          Small, SmallSE, RegAide, RegAideSE, SSD),
          digits = 2))

```

It is fairly easy to turn the `data.frame` `df` into a table.

```

# generate a simple table using stargazer
stargazer(df,
          title = "Estimated Class Size Effects
                  (in Units of Standard Deviations)",
          type = "html",
          summary = FALSE,
          header = FALSE
          )

```

Table 13.4: Estimated Class Size Effects (in Units of Standard Deviations)

	K	1	2	3
Small	0.190	0.330	0.230	0.210
SmallSE	0.030	0.030	0.030	0.030
RegAide	0	0.130	0.040	0
RegAideSE	0.030	0.030	0.030	0.030
SSD	73.750	91.280	84.080	73.270

The estimated effect of a small class is largest for grade 1. As pointed out in the book, this is probably because students in the control group for grade 1 did poorly on the test for some unknown reason or simply due to random variation. The difference between the estimated effect of being in a small class and being in a regular class with an aide is roughly 0.2 standard deviations for all grades. This leads to the conclusion that the effect of being in a regular sized class with an aide is zero and the effect of being in a small class is roughly the same for all grades.

The remainder of Chapter 13.3 in the book discusses to what extent these experimental estimates are comparable with observational estimates obtained using data on school districts in California and

Massachusetts in Chapter 9. It turns out that the estimates are indeed very similar. Please refer to the aforementioned section in the book for a more detailed discussion.

13.4 Quasi Experiments

In quasi-experiments, “as if” randomness is exploited to use methods similar to those that have been discussed in the previous chapter. There are two types of quasi-experiments:¹

1. Random variations in individual circumstances allow to view the treatment “as if” it was randomly determined.
2. The treatment is only partially determined by “as if” random variation.

The former allows to estimate the effect using either model (13.2), i.e., the *difference estimator with additional regressors*, or, if there is doubt that the “as if” randomness does not entirely ensure that there are no systematic differences between control and treatment group, using the *differences-in-differences* (DID) estimator. In the latter case, an IV approach for estimation of a model like (13.2) which uses the source of “as if” randomness in treatment assignment as the instrument may be applied.

Some more advanced techniques that are helpful in settings where the treatment assignment is (partially) determined by a threshold in a so-called running variable are *sharp regression discontinuity design* (RDD) and *fuzzy regression discontinuity design* (FRDD).

We briefly review these techniques and, since the book does not provide any empirical examples in this section, we will use our own simulated data in a minimal example to discuss how DID, RDD and FRDD can be applied in R.

The Differences-in-Differences Estimator

In quasi-experiments the source of “as if” randomness in treatment assignment can often not entirely prevent systematic differences between control and treatment groups. This problem was encountered by Card and Krueger (1994) who use geography as the “as if” random treatment assignment to study the effect on employment in fast-food restaurants caused by an increase in the state minimum wage in New Jersey in the year of 1992. Their idea was to use the fact that the increase in minimum wage applied to employees in New Jersey (treatment group) but not to those living in neighboring Pennsylvania (control group).

It is quite conceivable that such a wage hike is not correlated with other determinants of employment. However, there still might be some state-specific differences and thus differences between control and treatment group. This would render the *differences estimator* biased and inconsistent. Card and Krueger (1994) solved this by using a DID estimator: they collected data in February 1992 (before the treatment) and November 1992 (after the treatment) for the same restaurants and estimated the effect of the wage hike by analyzing differences in the differences in employment for New Jersey and Pennsylvania before and after the increase.² The DID estimator is

$$\hat{\beta}_1^{\text{diffs-in-diffs}} = (\bar{Y}^{\text{treatment,after}} - \bar{Y}^{\text{treatment,before}}) - (\bar{Y}^{\text{control,after}} - \bar{Y}^{\text{control,before}}) \quad (13.9)$$

$$= \Delta \bar{Y}^{\text{treatment}} - \Delta \bar{Y}^{\text{control}}, \quad (13.10)$$

with

- $\bar{Y}^{\text{treatment,before}}$ - the sample average in the treatment group before the treatment

¹See Chapter 13.4 of the book for some example studies that are based on quasi-experiments.

²Also see the box *What is the Effect on Employment of the Minimum Wage?* in Chapter 13.4 of the book.

- $\bar{Y}^{\text{treatment,after}}$ - the sample average in the treatment group after the treatment
- $\bar{Y}^{\text{treatment,before}}$ - the sample average in the control group before the treatment
- $\bar{Y}^{\text{control,after}}$ - the sample average in the control group after the treatment.

We now use R to reproduce Figure 13.1 of the book.

```
# initialize plot and add control group
plot(c(0, 1), c(6, 8),
     type = "p",
     ylim = c(5, 12),
     xlim = c(-0.3, 1.3),
     main = "The Differences-in-Differences Estimator",
     xlab = "Period",
     ylab = "Y",
     col = "steelblue",
     pch = 20,
     xaxt = "n",
     yaxt = "n")

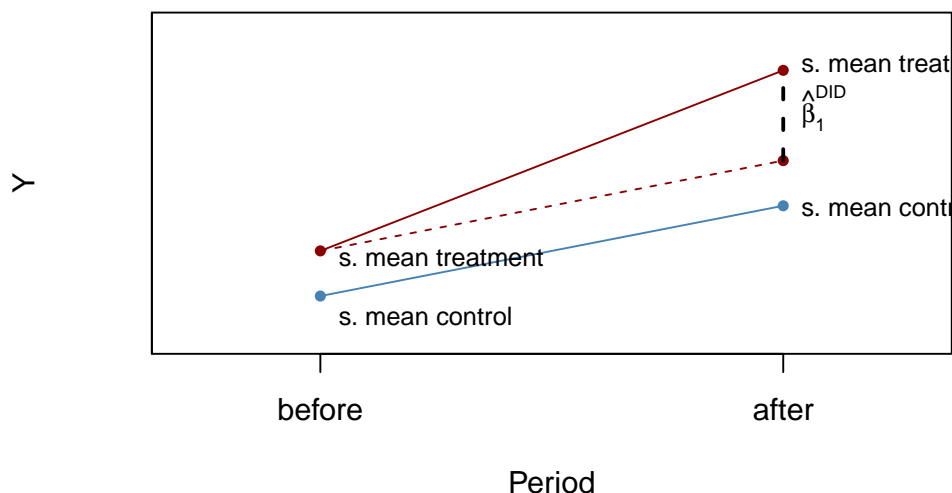
axis(1, at = c(0, 1), labels = c("before", "after"))
axis(2, at = c(0, 13))

# add treatment group
points(c(0, 1, 1), c(7, 9, 11),
       col = "darkred",
       pch = 20)

# add line segments
lines(c(0, 1), c(7, 11), col = "darkred")
lines(c(0, 1), c(6, 8), col = "steelblue")
lines(c(0, 1), c(7, 9), col = "darkred", lty = 2)
lines(c(1, 1), c(9, 11), col = "black", lty = 2, lwd = 2)

# add annotations
text(1, 10, expression(hat(beta)[1]^{DID}), cex = 0.8, pos = 4)
text(0, 5.5, "s. mean control", cex = 0.8, pos = 4)
text(0, 6.8, "s. mean treatment", cex = 0.8, pos = 4)
text(1, 7.9, "s. mean control", cex = 0.8, pos = 4)
text(1, 11.1, "s. mean treatment", cex = 0.8, pos = 4)
```

The Differences-in-Differences Estimator



The DID estimator (13.10) can also be written in regression notation: $\widehat{\beta}_1^{\text{DID}}$ is the OLS estimator of β_1 in

$$\Delta Y_i = \beta_0 + \beta_1 X_i + u_i, \quad (13.11)$$

where ΔY_i denotes the difference in pre- and post-treatment outcomes of individual i and X_i is the treatment indicator.

Adding additional regressors that measure pre-treatment characteristics to (13.11) we obtain

$$\Delta Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_{1i} + \cdots + \beta_{1+r} W_{ri} + u_i, \quad (13.12)$$

the *difference-in-differences estimator* with additional regressors. The additional regressors may lead to a more precise estimate of β_1 .

We keep things simple and focus on estimation of the treatment effect using DID in the simplest case, that is a control and a treatment group observed for two time periods — one before and one after the treatment. In particular, we will see that there are three different ways to proceed.

First, we simulate pre- and post-treatment data using R.

```
# set sample size
n <- 200

# define treatment effect
TEffect <- 4

# generate treatment dummy
TDummy <- c(rep(0, n/2), rep(1, n/2))

# simulate pre- and post-treatment values of the dependent variable
y_pre <- 7 + rnorm(n)
y_pre[1:n/2] <- y_pre[1:n/2] - 1
y_post <- 7 + 2 + TEffect * TDummy + rnorm(n)
y_post[1:n/2] <- y_post[1:n/2] - 1
```

Next we plot the data. The function `jitter()` is used to add some artificial dispersion in the horizontal component of the points so that there is less overplotting. The function `alpha()` from the package `scales` allows to adjust the opacity of colors used in plots.

```
library(scales)

pre <- rep(0, length(y_pre[TDummy==0]))
post <- rep(1, length(y_pre[TDummy==0]))

# plot control group in t=1
plot(jitter(pre, 0.6),
     y_pre[TDummy == 0],
     ylim = c(0, 16),
     col = alpha("steelblue", 0.3),
     pch = 20,
     xlim = c(-0.5, 1.5),
     ylab = "Y",
     xlab = "Period",
     xaxt = "n",
     main = "Artificial Data for DID Estimation")

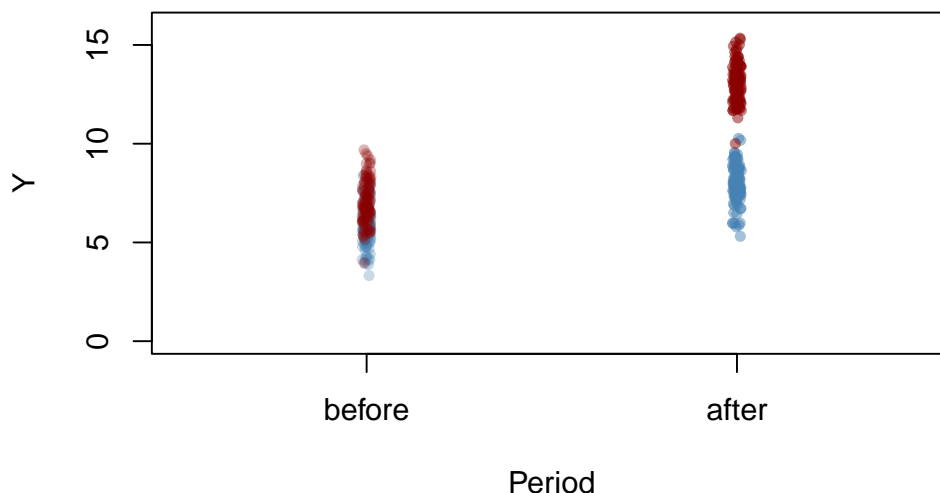
axis(1, at = c(0, 1), labels = c("before", "after"))

# add treatment group in t=1
points(jitter(pre, 0.6),
       y_pre[TDummy == 1],
       col = alpha("darkred", 0.3),
       pch = 20)

# add control group in t=2
points(jitter(post, 0.6),
       y_post[TDummy == 0],
       col = alpha("steelblue", 0.5),
       pch = 20)

# add treatment group in t=2
points(jitter(post, 0.6),
       y_post[TDummy == 1],
       col = alpha("darkred", 0.5),
       pch = 20)
```

Artificial Data for DID Estimation



After the treatment, both the control and treatment groups have a higher mean, but the increase is stronger in the treatment group. Using DID we may estimate how much of that difference is due to the treatment.

It is straightforward to compute the DID estimate in the fashion of (13.10).

```
# compute the DID estimator for the treatment effect 'by hand'
mean(y_post[TDummy == 1]) - mean(y_pre[TDummy == 1]) -
(mean(y_post[TDummy == 0]) - mean(y_pre[TDummy == 0]))
#> [1] 3.960268
```

Notice that the estimate is close to 4, the value chosen as the treatment effect `TEffect` above. Since (13.11) is a simple linear model, we may perform OLS estimation of this regression specification using `lm()`.

```
# compute the DID estimator using a linear model
lm(I(y_post - y_pre) ~ TDummy)
#>
#> Call:
#> lm(formula = I(y_post - y_pre) ~ TDummy)
#>
#> Coefficients:
#> (Intercept)      TDummy
#>      2.104         3.960
```

We find that the estimates coincide. Furthermore, one can show that the DID estimate obtained by estimating specification (13.11) is the same as the OLS estimate of β_{TE} in

$$Y_i = \beta_0 + \beta_1 D_i + \beta_2 Period_i + \beta_{TE}(Period_i \times D_i) + \varepsilon_i, \quad (13.13)$$

where D_i is the binary treatment indicator, $Period_i$ is a binary indicator for the after-treatment period and the $Period_i \times D_i$ is the interaction of both.

As for (13.11), estimation of (13.13) using R is straightforward. See Chapter 8 for a discussion of interaction terms.

```

# prepare data for DID regression using the interaction term
d <- data.frame("Y" = c(y_pre,y_post),
               "Treatment" = TDummy,
               "Period" = c(rep("1", n), rep("2", n)))

# estimate the model
lm(Y ~ Treatment * Period, data = d)
#>
#> Call:
#> lm(formula = Y ~ Treatment * Period, data = d)
#>
#> Coefficients:
#>      (Intercept)      Treatment      Period2 Treatment:Period2
#>           5.858            1.197            2.104            3.960

```

As expected, the estimate of the coefficient on the interaction of the treatment dummy and the time dummy coincide with the estimates obtained using (13.10) and OLS estimation of (13.11).

Regression Discontinuity Estimators

Consider the model

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_i + u_i \quad (13.14)$$

and let

$$X_i = \begin{cases} 1, & W_i \geq c \\ 0, & W_i < c, \end{cases}$$

so that the receipt of treatment, X_i , is determined by some threshold c of a continuous variable W_i , the so called running variable. The idea of *regression discontinuity design* is to use observations with a W_i close to c for estimation of β_1 . β_1 is the average treatment effect for individuals with $W_i = c$ which is assumed to be a good approximation to the treatment effect in the population. (13.14) is called a *sharp regression discontinuity design* because treatment assignment is deterministic and discontinuous at the cutoff: all observations with $W_i < c$ do not receive treatment and all observations where $W_i \geq c$ are treated.

The subsequent code chunks show how to estimate a linear SRDD using R and how to produce plots in the way of Figure 13.2 of the book.

```

# generate some sample data
W <- runif(1000, -1, 1)
y <- 3 + 2 * W + 10 * (W>=0) + rnorm(1000)

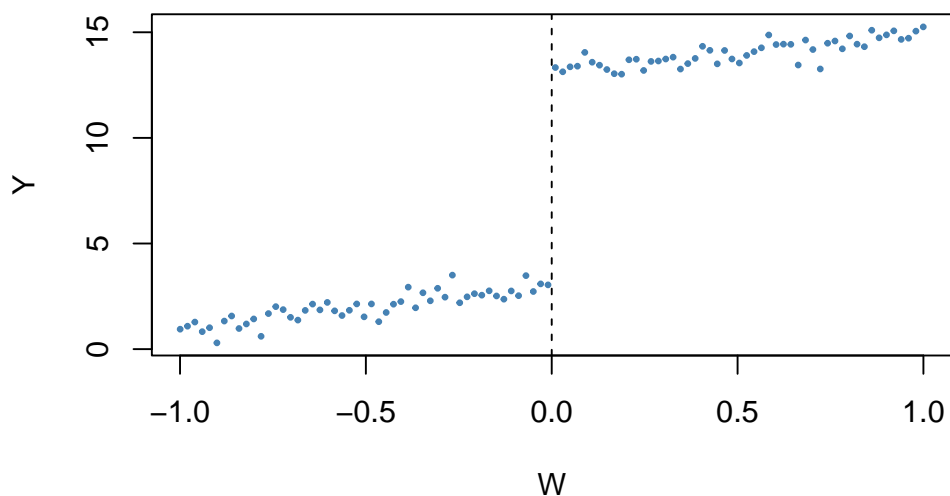
# load the package 'rddtools'
library(rddtools)

# construct rdd_data
data <- rdd_data(y, W, cutpoint = 0)

# plot the sample data
plot(data,
     col = "steelblue",
     cex = 0.35,

```

```
xlab = "W",
ylab = "Y")
```



W
h=0.0198/0.0198, n bins=102 (51/51)

The argument `nbins` sets the number of bins the running variable is divided into for aggregation. The dots represent bin averages of the outcome variable.

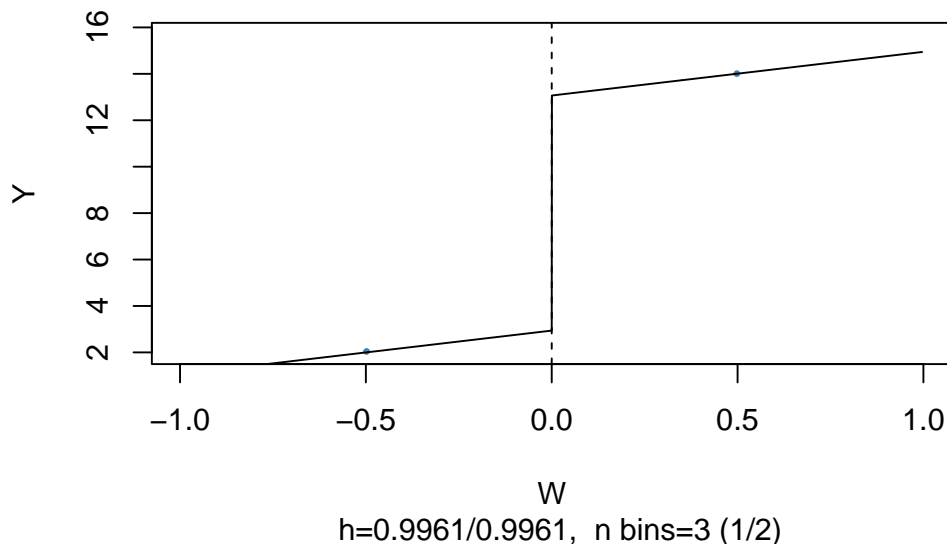
We may use the function `rdd_reg_lm()` to estimate the treatment effect using model (13.14) for the artificial data generated above. By choosing `slope = "same"` we restrict the slopes of the estimated regression function to be the same on both sides of the jump at the cutpoint $W = 0$.

```
# estimate the sharp RDD model
rdd_mod <- rdd_reg_lm(rdd_object = data,
                     slope = "same")
summary(rdd_mod)
#>
#> Call:
#> lm(formula = y ~ ., data = dat_step1, weights = weights)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -3.2361 -0.6779 -0.0039  0.7113  3.0096
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.93889    0.07082   41.50  <2e-16 ***
#> D            10.12692    0.12631   80.18  <2e-16 ***
#> x             1.88249    0.11074   17.00  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.019 on 997 degrees of freedom
#> Multiple R-squared:  0.972, Adjusted R-squared:  0.972
#> F-statistic: 1.732e+04 on 2 and 997 DF, p-value: < 2.2e-16
```

The coefficient estimate of interest is labeled D. The estimate is very close to the treatment effect chosen in the DGP above.

It is easy to visualize the result: simply call `plot()` on the estimated model object.

```
# plot the RDD model along with binned observations
plot(rdd_mod,
     cex = 0.35,
     col = "steelblue",
     xlab = "W",
     ylab = "Y")
```



As above, the dots represent averages of binned observations.

So far we assumed that crossing of the threshold determines receipt of treatment so that the jump of the population regression functions at the threshold can be regarded as the causal effect of the treatment.

When crossing the threshold c is not the only cause for receipt of the treatment, treatment is no a deterministic function of W_i . Instead, it is useful to think of c as a threshold where the *probability* of receiving the treatment jumps.

This jump may be due to unobservable variables that have impact on the probability of being treated. Thus, X_i in (13.14) will be correlated with the error u_i and it becomes more difficult to consistently estimate the treatment effect. In this setting, using a *fuzzy regression discontinuity design* which is based an IV approach may be a remedy: take the binary variable Z_i as an indicator for crossing of the threshold,

$$Z_i = \begin{cases} 1, & W_i \geq c \\ 0, & W_i < c, \end{cases}$$

and assume that Z_i relates to Y_i only through the treatment indicator X_i . Then Z_i and u_i are uncorrelated but Z_i influences receipt of treatment so it is correlated with X_i . Thus, Z_i is a valid instrument for X_i and (13.14) can be estimated using TSLS.

The following code chunk generates sample data where observations with a value of the running variable W_i below the cutoff $c = 0$ do not receive treatment and observations with $W_i \geq 0$ do receive treatment with a probability of 80% so that treatment status is only partially determined by the running variable and the cutoff. Treatment leads to an increase in Y by 2 units. Observations with $W_i \geq 0$ that do not receive treatment are called *no-shows*: think of an individual that was assigned to receive the treatment but somehow manages to avoid it.

```

library(MASS)

# generate sample data
mu <- c(0, 0)
sigma <- matrix(c(1, 0.7, 0.7, 1), ncol = 2)

set.seed(1234)
d <- as.data.frame(mvnorm(2000, mu, sigma))
colnames(d) <- c("W", "Y")

# introduce fuzziness
d$treatProb <- ifelse(d$W < 0, 0, 0.8)

fuzz <- sapply(X = d$treatProb, FUN = function(x) rbinom(1, 1, prob = x))

# treatment effect
d$Y <- d$Y + fuzz * 2

```

`sapply()` applies the function provided to `FUN` to every element of the argument `X`. Here, since `d$treatProb` is a vector, the result is a vector, too.

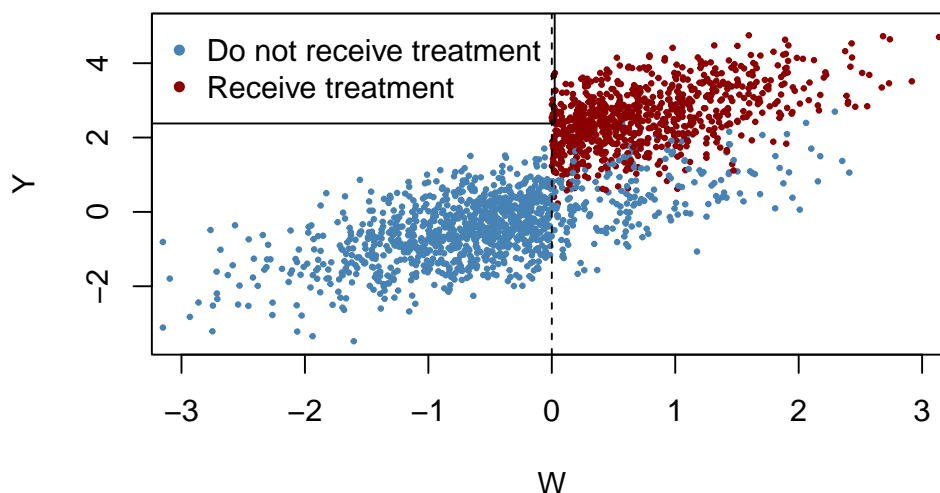
We plot all observations and use blue color to mark individuals that did not receive the treatment and use red color for those who received the treatment.

```

# generate a colored plot of treatment and control group
plot(d$W, d$Y,
     col = c("steelblue", "darkred")[factor(fuzz)],
     pch= 20,
     cex = 0.5,
     xlim = c(-3, 3),
     ylim = c(-3.5, 5),
     xlab = "W",
     ylab = "Y")

# add a dashed vertical line at cutoff
abline(v = 0, lty = 2)
#add legend
legend("topleft", pch=20, col=c("steelblue", "darkred"),
      legend=c("Do not receive treatment", "Receive treatment"))

```



Obviously, receipt of treatment is no longer a deterministic function of the running variable W . Some observations with $W \geq 0$ *did not* receive the treatment. We may estimate a FRDD by additionally setting `treatProb` as the assignment variable z in `rdd_data()`. Then `rdd_reg_lm()` applies the following TSLS procedure: treatment is predicted using W_i and the cutoff dummy Z_i , the instrumental variable, in the first stage regression. The fitted values from the first stage regression are used to obtain a consistent estimate of the treatment effect using the second stage where the outcome Y is regressed on the fitted values and the running variable W .

```
# estimate the Fuzzy RDD
data <- rdd_data(d$Y, d$W,
               cutpoint = 0,
               z = d$treatProb)

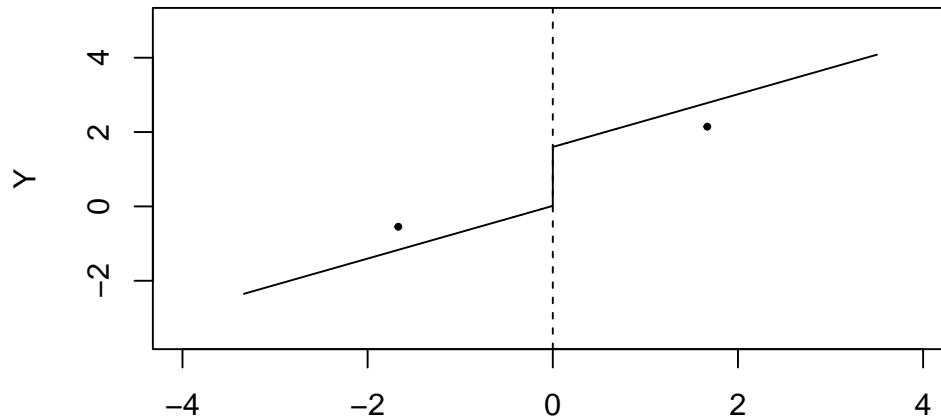
frdd_mod <- rdd_reg_lm(rdd_object = data,
                     slope = "same")

frdd_mod
#> ### RDD regression: parametric ###
#> Polynomial order: 1
#> Slopes: same
#> Number of obs: 2000 (left: 999, right: 1001)
#>
#> Coefficient:
#> Estimate Std. Error t value Pr(>|t|)
#> D 1.981297 0.084696 23.393 < 2.2e-16 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimate is close to 2, the population treatment effect. We may call `plot()` on the model object to obtain a figure consisting of binned data and the estimated regression function.

```
# plot estimated FRDD function
plot(frdd_mod,
     cex = 0.5,
     lwd = 0.4,
     xlim = c(-4, 4),
     ylim = c(-3.5, 5),
     xlab = "W",
```

```
ylab = "Y")
```



W
h=3.3374/3.3374, n bins=4 (2/2)

What if we used a SRDD instead, thereby ignoring the fact that treatment is not perfectly determined by the cutoff in W ? We may get an impression of the consequences by estimating an SRDD using the previously simulated data.

```
# estimate SRDD
data <- rdd_data(d$Y,
                d$W,
                cutpoint = 0)

srdd_mod <- rdd_reg_lm(rdd_object = data,
                      slope = "same")

srdd_mod
#> ### RDD regression: parametric ###
#> Polynomial order: 1
#> Slopes: same
#> Number of obs: 2000 (left: 999, right: 1001)
#>
#> Coefficient:
#> Estimate Std. Error t value Pr(>|t|)
#> D 1.585038 0.067756 23.393 < 2.2e-16 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimate obtained using a SRDD is suggestive of a substantial downward bias. In fact, this procedure is inconsistent for the true causal effect so increasing the sample would not alleviate the bias.

The book continues with a discussion of potential problems with quasi-experiments. As for all empirical studies, these potential problems are related to internal and external validity. This part is followed by a technical discussion of treatment effect estimation when the causal effect of treatment is heterogeneous in the population. We encourage you to work on these sections on your own.

Summary

This chapter has introduced the concept of causal effects in randomized controlled experiments and quasi-experiments where variations in circumstances or accidents of nature are treated as sources of “as if” random assignment to treatment. We have also discussed methods that allow for consistent estimation of these effects in both settings. These included the *differences estimator*, the *differences-in-differences estimator* as well as *sharp* and *fuzzy regression discontinuity design* estimators. It was shown how to apply these estimation techniques in R.

In an empirical application we have shown how to replicate the results of the analysis of the STAR data presented in Chapter 13.3 of the book using R. This study uses a randomized controlled experiment to assess whether smaller classes improve students’ performance on standardized tests. Being related to a randomized controlled experiment, the data of this study is fundamentally different to those used in the cross-section studies in Chapters 4 to 8. We therefore have motivated usage of a *differences estimator*.

Chapter 12.4 demonstrated how estimates of treatment effects can be obtained when the design of the study is a quasi-experiment that allows for *differences-in-differences* or *regression discontinuity design* estimators. In particular, we have introduced functions of the package `rddtools` that are convenient for estimation as well as graphical analysis when estimating a regression discontinuity design.

13.5 Exercises

The subsequent exercises guide you in reproducing some of the results presented in one of the most famous DID studies by Card and Krueger (1994). The authors use geography as the “as if” random treatment assignment to study the effect on employment in fast food restaurants caused by an increase in the state minimum wage in New Jersey in the year of 1992, see Chapter 13.4.

The study is based on survey data collected in February 1992 and in November 1992, after New Jersey’s minimum wage rose by \$0.80 from \$4.25 to \$5.05 in April 1992.

Estimating the effect of the wage increase simply by computing the change in employment in New Jersey (as you are asked to do in Exercise 3) would fail to control for omitted variables. By using Pennsylvania as a control in a difference-in-differences (DID) model one can control for variables with a common influence on New Jersey (treatment group) and Pennsylvania (control group). This reduces the risk of omitted variable bias enormously and even works when these variables are unobserved.

For the DID approach to work we must assume that New Jersey and Pennsylvania have parallel trends over time, i.e., we assume that the (unobserved) factors influence employment in Pennsylvania and New Jersey in the same manner. This allows to interpret an observed change in employment in Pennsylvania as the change New Jersey would have experienced if there was no increase in minimum wage (and vice versa).

Against to what standard economic theory would suggest, the authors did not find evidence that the increased minimum wage induced an increase in unemployment in New Jersey using the DID approach: quite the contrary, their results suggest that the \$0.80 minimum wage increase in New Jersey led to a 2.75 full-time equivalent (FTE) increase in employment.

Chapter 14

Introduction to Time Series Regression and Forecasting

Time series data is collected for a single entity over time. This is fundamentally different from cross-section data which is data on multiple entities at the same point in time. Time series data allows estimation of the effect on Y of a change in X *over time*. This is what econometricians call a *dynamic causal effect*. Let us go back to the application of cigarette consumption of Chapter 12 where we were interested in estimating the effect on cigarette demand of a price increase caused by a raise of the general sales tax. One might use time series data to assess the causal effect of a tax increase on smoking both, initially and in subsequent periods.

Another application of time series data is forecasting. For example, weather services use time series data to predict tomorrow's temperature by, inter alia, using today's temperature and temperatures of the past. To motivate an economic example, central banks are interested in forecasting next month's unemployment rates.

The remaining chapters in the book deal with econometric techniques for analyzing time series data, as well as their applications in forecasting and estimating dynamic causal effects. This section covers the basic concepts presented in Chapter 14 of the book, explains how to visualize time series data and demonstrates how to estimate simple autoregressive models, where the regressors are past values of the dependent variable or other variables. In this context we also discuss the concept of stationarity, an important property which has far-reaching consequences.

Most empirical applications in this chapter are concerned with forecasting and use data on U.S. macroeconomic indicators or financial time series like Gross Domestic Product (GDP), the unemployment rate or excess stock returns.

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter:

- `AER` (Kleiber and Zeileis, 2008b),
- `dynlm` (Zeileis, 2019),
- `forecast` (Hyndman et al., 2023),
- `readxl` (Wickham and Bryan, 2023),
- `stargazer` (Hlavac, 2022),
- `scales` (Wickham and Seidel, 2022),
- `quantmod` (Ryan and Ulrich, 2023),
- `urca` (Pfaff, 2008).

Please verify that the following code chunk runs on your machine without any errors.

```
library(AER)
library(dynlm)
library(forecast)
library(readxl)
library(stargazer)
library(scales)
library(quantmod)
library(urca)
```

14.1 Using Regression Models for Forecasting

What is the difference between estimating models for assessment of causal effects and forecasting? Consider again the simple example of estimating the causal effect of the student-teacher ratio on test scores introduced in Chapter 4.

```
library(AER)
data(CASchools)
CASchools$STR <- CASchools$students/CASchools$teachers
CASchools$score <- (CASchools$read + CASchools$math)/2

mod <- lm(score ~ STR, data = CASchools)
mod
#>
#> Call:
#> lm(formula = score ~ STR, data = CASchools)
#>
#> Coefficients:
#> (Intercept)          STR
#>    698.93         -2.28
```

As has been stressed in Chapter 6, the estimate of the coefficient on the student-teacher ratio does not have a causal interpretation due to omitted variable bias. However, in terms of deciding which school to send her child to, it might nevertheless be appealing for a parent to use `mod` for forecasting test scores in schooling districts where no public data on scores are available.

As an example, assume that the average class in a district has 25 students. This is not a perfect forecast but the following one-liner might be helpful for the parent to decide.

```
predict(mod, newdata = data.frame("STR" = 25))
#>          1
#> 641.9377
```

In a time series context, the parent could use data on present and past years test scores to forecast next year's test scores — a typical application for an autoregressive model.

14.2 Time Series Data and Serial Correlation

GDP is commonly defined as the total value of goods and services produced during a given time period. The dataset `us_macro_quarterly.xlsx`, provided by the authors, can be downloaded [here](#). It contains quarterly data on U.S. real GDP (i.e., inflation-adjusted) from 1947 to 2004.

As before, a good starting point is to plot the data. The package `quantmod` provides some convenient functions for plotting and computing with time series data. We also load the package `readxl` to read the data into R.

```
# attach the package 'quantmod'
library(quantmod)
library(readxl)
```

We begin by importing the data set.

```
# load US macroeconomic data
USMacroSWQ <- read_xlsx("Data/us_macro_quarterly.xlsx",
                        sheet = 1,
                        col_types = c("text", rep("numeric", 9)))

# format date column
USMacroSWQ$...1 <- as.yearqtr(USMacroSWQ$...1, format = "%Y:0%q")

# adjust column names
colnames(USMacroSWQ) <- c("Date", "GDPC96", "JAPAN_IP", "PCEPTI",
                          "GS10", "GS1", "TB3MS", "UNRATE", "EXUSUK", "CPIAUCSL")
```

The first column of `us_macro_quarterly.xlsx` contains text and the remaining ones are numeric. Using `col_types = c("text", rep("numeric", 9))`, we instruct `read_xlsx()` to consider this format when importing the data.

It is useful to work with time-series objects that keep track of the frequency of the data and are extensible. In what follows we will use objects of the class `xts`, see `?xts`. Since the data in `USMacroSWQ` are in quarterly frequency we convert the first column to `yearqtr` format before generating the `xts` object `GDP`.

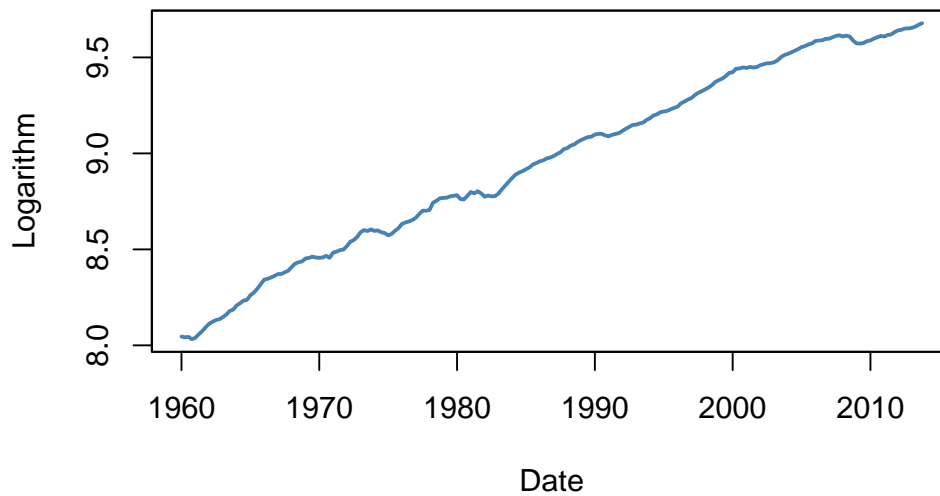
```
# GDP series as xts object
GDP <- xts(USMacroSWQ$GDPC96, USMacroSWQ$Date) ["1960:2013"]

# GDP growth series as xts object
GDPGrowth <- xts(400 * log(GDP/lag(GDP)))
```

The following code chunks reproduce Figure 14.1 of the book.

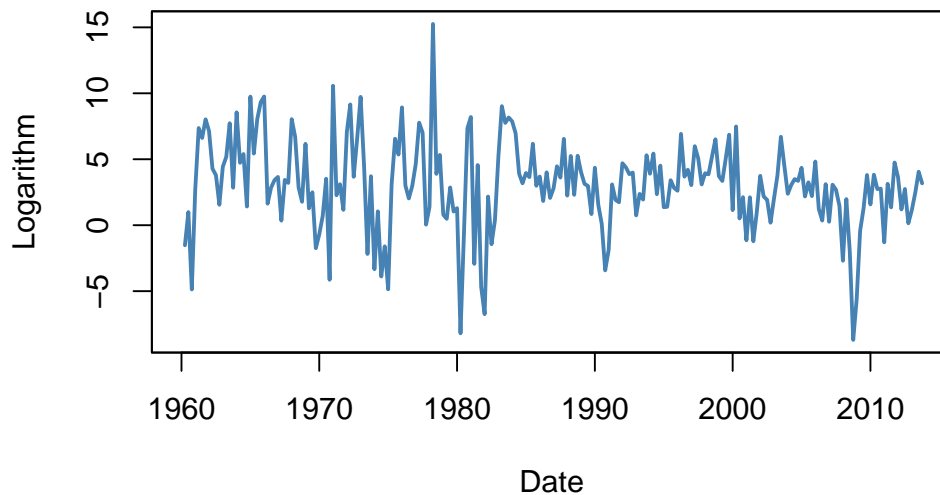
```
# reproduce Figure 14.1 (a) of the book
plot(log(as.zoo(GDP)),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "U.S. Quarterly Real GDP")
```


U.S. Quarterly Real GDP



```
# reproduce Figure 14.1 (b) of the book
plot(as.zoo(GDPGrowth),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "U.S. Real GDP Growth Rates")
```

U.S. Real GDP Growth Rates



Notation, Lags, Differences, Logarithms and Growth Rates

For observations of a variable Y recorded over time, Y_t denotes the value observed at time t . The period between two sequential observations Y_t and Y_{t-1} is a unit of time: hours, days, weeks, months, quarters, years, etc. Key Concept 14.1 introduces the essential terminology and notation for time series data we use in the subsequent sections.

Key Concept 14.1**Lags, First Differences, Logarithms and Growth Rates**

- Previous values of a time series are called *lags*. The first lag of Y_t is Y_{t-1} . The j^{th} lag of Y_t is Y_{t-j} . In R, lags of univariate or multivariate time series objects are conveniently computed by `lag()`, see `?lag`.
- Sometimes we work with differenced series. The first difference of a series is $\Delta Y_t = Y_t - Y_{t-1}$, the difference between periods t and $t - 1$. If Y is a time series, the series of first differences is computed as `diff(Y)`.
- It may be convenient to work with the first difference in logarithms of a series. We denote this by $\Delta \log(Y_t) = \log(Y_t) - \log(Y_{t-1})$. For a time series Y , this is obtained using `log(Y/lag(Y))`.
- $100\Delta \log(Y_t)$ is an approximation for the percentage change between Y_t and Y_{t-1} .

The definitions made in Key Concept 14.1 are useful because of two properties that are common to many economic time series:

- Exponential growth: some economic series grow approximately exponentially such that their logarithm is approximately linear.
- The standard deviation of many economic time series is approximately proportional to their level. Therefore, the standard deviation of the logarithm of such a series is approximately constant.

Furthermore, it is common to report growth rates in macroeconomic series which is why log-differences are often used.

Table 14.1 of the book presents the quarterly U.S. GDP time series, its logarithm, the annualized growth rate and the first lag of the annualized growth rate series for the period 2012:Q1 - 2013:Q1. The following simple function can be used to compute these quantities for a quarterly time series `series`.

```
# compute logarithms, annual growth rates and 1st lag of growth rates
quants <- function(series) {
  s <- series
  return(
    data.frame("Level" = s,
              "Logarithm" = log(s),
              "AnnualGrowthRate" = 400 * log(s / lag(s)),
              "1stLagAnnualGrowthRate" = lag(400 * log(s / lag(s))))
  )
}
```

The annual growth rate is computed using the approximation

$$\text{AnnualGrowth}Y_t = 400 \cdot \Delta \log(Y_t)$$

since $100 \cdot \Delta \log(Y_t)$, is an approximation of the quarterly percentage changes, see Key Concept 14.1.

We call `quants()` on observations for the period 2011:Q3 - 2013:Q1.

```
# obtain a data.frame with level, logarithm, annual growth rate and its 1st lag of GDP
quants(GDP["2011-07::2013-01"])
#>      Level Logarithm AnnualGrowthRate X1stLagAnnualGrowthRate
#> 2011 Q3 15062.14  9.619940                NA                NA
#> 2011 Q4 15242.14  9.631819                4.7518062           NA
#> 2012 Q1 15381.56  9.640925                3.6422231           4.7518062
```

```
#> 2012 Q2 15427.67 9.643918 1.1972004 3.6422231
#> 2012 Q3 15533.99 9.650785 2.7470216 1.1972004
#> 2012 Q4 15539.63 9.651149 0.1452808 2.7470216
#> 2013 Q1 15583.95 9.653997 1.1392015 0.1452808
```

Autocorrelation

Observations of a time series are typically correlated. This type of correlation is called *autocorrelation* or *serial correlation*. Key Concept 14.2 summarizes the concepts of population autocovariance and population autocorrelation and shows how to compute their sample equivalents.

Key Concept 14.2 Autocorrelation and Autocovariance

The covariance between Y_t and its j^{th} lag, Y_{t-j} , is called the j^{th} *autocovariance* of the series Y_t . The j^{th} *autocorrelation coefficient*, also called the *serial correlation coefficient*, measures the correlation between Y_t and Y_{t-j} . We thus have

$$j^{\text{th}} \text{ autocovariance} = \text{Cov}(Y_t, Y_{t-j}),$$

$$j^{\text{th}} \text{ autocorrelation} = \rho_j = \rho_{Y_t, Y_{t-j}} = \frac{\text{Cov}(Y_t, Y_{t-j})}{\sqrt{\text{Var}(Y_t)\text{Var}(Y_{t-j})}}.$$

Population autocovariance and population autocorrelation can be estimated by $\widehat{\text{Cov}}(Y_t, Y_{t-j})$, the sample autocovariance, and $\widehat{\rho}_j$, the sample autocorrelation:

$$\widehat{\text{Cov}}(Y_t, Y_{t-j}) = \frac{1}{T} \sum_{t=j+1}^T (Y_t - \bar{Y}_{j+1:T})(Y_{t-j} - \bar{Y}_{1:T-j}),$$

$$\widehat{\rho}_j = \frac{\widehat{\text{Cov}}(Y_t, Y_{t-j})}{\widehat{\text{Var}}(Y_t)}.$$

$\bar{Y}_{j+1:T}$ denotes the average of $Y_{j+1}, Y_{j+2}, \dots, Y_T$.

In R the function `acf()` from the package `stats` computes the sample autocovariance or the sample autocorrelation function.

Using `acf()` it is straightforward to compute the first four sample autocorrelations of the series `GDPGrowth`.

```
acf(na.omit(GDPGrowth), lag.max = 4, plot = F)
#>
#> Autocorrelations of series 'na.omit(GDPGrowth)', by lag
#>
#> 0.00 0.25 0.50 0.75 1.00
#> 1.000 0.352 0.273 0.114 0.106
```

This is the evidence that there is mild positive autocorrelation in the growth of GDP: if GDP grows faster than average in one period, there is a tendency for it to grow faster than average in the following periods.

Other Examples of Economic Time Series

Figure 14.2 of the book presents four plots: the U.S. unemployment rate, the U.S. Dollar / British Pound exchange rate, the logarithm of the Japanese industrial production index as well as daily changes in the Wilshire 5000 stock price index, a financial time series. The next code chunk reproduces the plots of the three macroeconomic series and adds percentage changes in the daily values of the New York Stock Exchange Composite index as a fourth one (the data set NYSESW comes with the AER package).

```
# define series as xts objects
USUnemp <- xts(USMacroSWQ$UNRATE, USMacroSWQ$Date) ["1960::2013"]

DollarPoundFX <- xts(USMacroSWQ$EXUSUK, USMacroSWQ$Date) ["1960::2013"]

JPIndProd <- xts(log(USMacroSWQ$JAPAN_IP), USMacroSWQ$Date) ["1960::2013"]

# attach NYSESW data
data("NYSESW")
NYSESW <- xts(Delt(NYSESW))

# divide plotting area into 2x2 matrix
par(mfrow = c(2, 2))

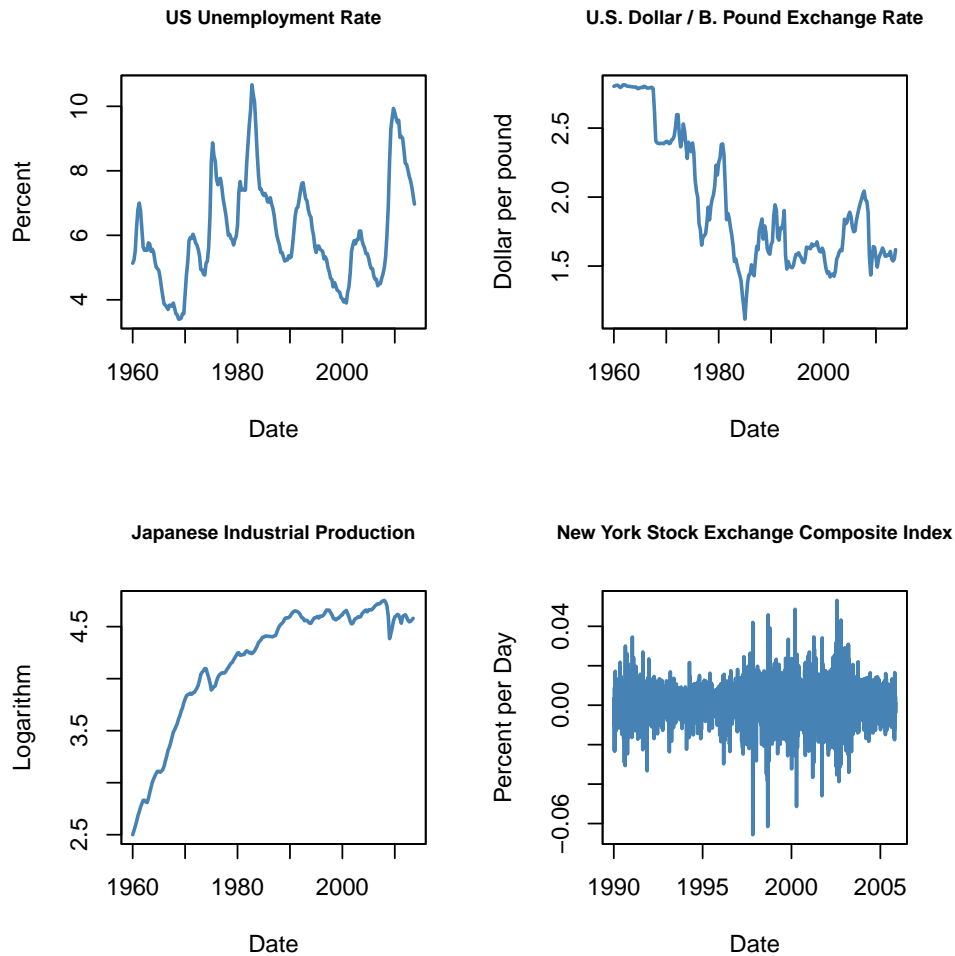
# plot the series
plot(as.zoo(USUnemp),
     col = "steelblue",
     lwd = 2,
     ylab = "Percent",
     xlab = "Date",
     main = "US Unemployment Rate",
     cex.main = 0.8)

plot(as.zoo(DollarPoundFX),
     col = "steelblue",
     lwd = 2,
     ylab = "Dollar per pound",
     xlab = "Date",
     main = "U.S. Dollar / B. Pound Exchange Rate",
     cex.main = 0.8)

plot(as.zoo(JPIndProd),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "Japanese Industrial Production",
     cex.main = 0.8)

plot(as.zoo(NYSESW),
     col = "steelblue",
     lwd = 2,
     ylab = "Percent per Day",
     xlab = "Date",
     main = "New York Stock Exchange Composite Index",
```

```
cex.main = 0.8)
```



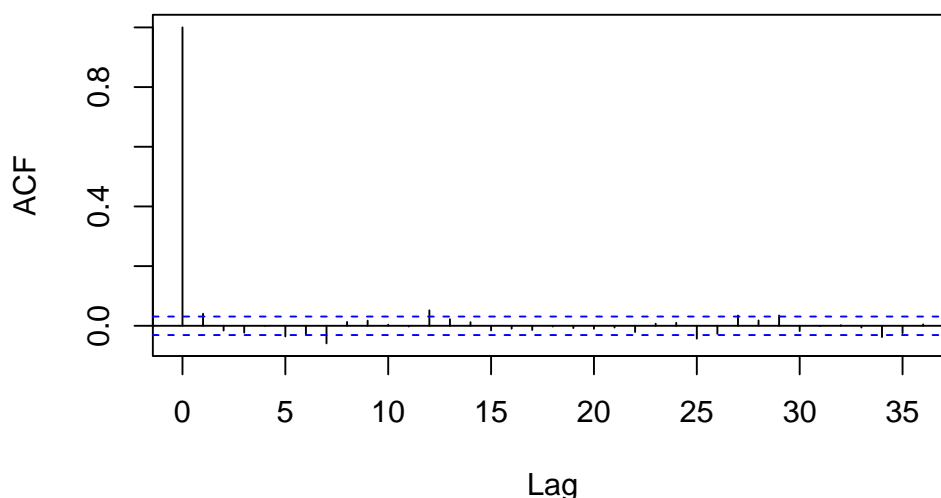
The series show quite different characteristics. The unemployment rate increases during recessions and declines during economic recoveries and growth. The Dollar/Pound exchange rates shows a deterministic pattern until the end of the Bretton Woods system. Japan's industrial production exhibits an upward trend and decreasing growth. Daily changes in the New York Stock Exchange composite index seem to fluctuate randomly around the zero line. The sample autocorrelations support this conjecture.

```
# compute sample autocorrelation for the NYSESW series
acf(na.omit(NYSESW), plot = F, lag.max = 10)
#>
#> Autocorrelations of series 'na.omit(NYSESW)', by lag
#>
#>      0      1      2      3      4      5      6      7      8      9     10
#> 1.000 0.040 -0.016 -0.023 0.000 -0.036 -0.027 -0.059 0.013 0.017 0.004
```

The first 10 sample autocorrelation coefficients are very close to zero. The default plot generated by `acf()` provides further evidence.

```
# plot sample autocorrelation for the NYSESW series
acf(na.omit(NYSESW), main = "Sample Autocorrelation for NYSESW Data")
```

Sample Autocorrelation for NYSESW Data



The blue dashed bands represent values beyond which the autocorrelations are significantly different from zero at 5% level. Even when the true autocorrelations are zero, we need to expect a few exceedences — recall the definition of a type-I-error from Key Concept 3.5. For most lags we see that the sample autocorrelation does not exceed the bands and there are only a few cases that lie marginally beyond the limits.

Furthermore, the NYSESW series exhibits what econometricians call *volatility clustering*: there are periods of high and periods of low variance. This is common for many financial time series.

14.3 Autoregressions

Autoregressive models are heavily used in economic forecasting. An autoregressive model relates a time series variable to its past values. This section discusses the basic ideas of autoregressions models, shows how they are estimated and discusses an application to forecasting GDP growth using R.

The First-Order Autoregressive Model

It is intuitive that the immediate past of a variable should have power to predict its near future. The simplest autoregressive model uses only the most recent outcome of the time series observed to predict future values. For a time series Y_t such a model is called a first-order autoregressive model, often abbreviated as AR(1), where the 1 indicates that the order of autoregression is one:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t$$

is the AR(1) population model of a time series Y_t .

For the GDP growth series, an autoregressive model of order one uses only the information on GDP growth observed in the last quarter to predict a future growth rate. The first-order autoregression model of GDP growth can be estimated by computing OLS estimates in the regression of $GDPGR_t$ on $GDPGR_{t-1}$,

$$\widehat{GDPGR}_t = \hat{\beta}_0 + \hat{\beta}_1 GDPGR_{t-1}. \quad (14.1)$$

Following the book we use data from 1962 to 2012 to estimate (14.1). This is easily done with the function `ar.ols()` from the package `stats`.

```

# subset data
GDPGRSub <- GDPGrowth["1962::2012"]

# estimate the model
ar.ols(GDPGRSub,
       order.max = 1,
       demean = F,
       intercept = T)

#>
#> Call:
#> ar.ols(x = GDPGRSub, order.max = 1, demean = F, intercept = T)
#>
#> Coefficients:
#>      1
#> 0.3384
#>
#> Intercept: 1.995 (0.2993)
#>
#> Order selected 1 sigma^2 estimated as 9.886

```

We can check that the computations done by `ar.ols()` are the same as done by `lm()`.

```

# length of data set
N <-length(GDPGRSub)

GDPGR_level <- as.numeric(GDPGRSub[-1])
GDPGR_lags <- as.numeric(GDPGRSub[-N])

# estimate the model
armod <- lm(GDPGR_level ~ GDPGR_lags)
armod

#>
#> Call:
#> lm(formula = GDPGR_level ~ GDPGR_lags)
#>
#> Coefficients:
#> (Intercept)  GDPGR_lags
#>      1.9950      0.3384

```

As usual, we may use `coefTest()` to obtain a robust summary on the estimated regression coefficients.

```

# robust summary
coefTest(armod, vcov. = vcovHC, type = "HC1")

#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 1.994986  0.351274  5.6793 4.691e-08 ***
#> GDPGR_lags  0.338436  0.076188  4.4421 1.470e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Thus the estimated model is

$$\widehat{GDPGR}_t = 1.995 + 0.338GDPGR_{t-1}. \quad (14.2)$$

(0.351) (0.076)

We omit the first observation for $GDPGR_{1962:Q1}$ from the vector of the dependent variable since $GDPGR_{1962:Q1} = GDPGR_{1961:Q4}$, is not included in the sample. Similarly, the last observation, $GDPGR_{2012:Q4}$, is excluded from the predictor vector since the data does not include $GDPGR_{2012:Q4+1} = GDPGR_{2013:Q1}$. Put differently, when estimating the model, one observation is lost because of the time series structure of the data.

Forecasts and Forecast Errors

Suppose Y_t follows an AR(1) model with an intercept and that you have an OLS estimate of the model on the basis of observations for T periods. Then you may use the AR(1) model to obtain $\widehat{Y}_{T+1|T}$, a forecast for Y_{T+1} using data up to period T where

$$\widehat{Y}_{T+1|T} = \hat{\beta}_0 + \hat{\beta}_1 Y_T.$$

The forecast error is

$$\text{Forecast error} = Y_{T+1} - \widehat{Y}_{T+1|T}.$$

Forecasts and Forecasted Values

Forecasted values of Y_t are *not* what we refer to as *OLS predicted values* of Y_t . Also, the forecast error is *not* an OLS residual. Forecasts and forecast errors are obtained using *out-of-sample* values while predicted values and residuals are computed for *in-sample* values that were actually observed and used in estimating the model.

The root mean squared forecast error (RMSFE) measures the typical size of the forecast error and is defined as

$$RMSFE = \sqrt{E \left[\left(Y_{T+1} - \widehat{Y}_{T+1|T} \right)^2 \right]}.$$

The *RMSFE* is composed of the future errors u_t and the error made when estimating the coefficients. When the sample size is large, the former may be much larger than the latter so that $RMSFE \approx \sqrt{\text{Var}(u_t)}$ which can be estimated by the standard error of the regression.

Application to GDP Growth

Using (14.2), the estimated AR(1) model of GDP growth, we perform the forecast for GDP growth for 2013:Q1 (remember that the model was estimated using data for periods 1962:Q1 - 2012:Q4, so 2013:Q1 is an out-of-sample period). Plugging $GDPGR_{2012:Q4} \approx 0.15$ into (14.2),

$$\widehat{GDPGR}_{2013:Q1} = 1.995 + 0.348 \cdot 0.15 = 2.047.$$

The function `forecast()` from the `forecast` package has some useful features for forecasting time series data.


```

library(forecast)

# assign GDP growth rate in 2012:Q4
new <- data.frame("GDPGR_lags" = GDPGR_level[N-1])

# forecast GDP growth rate in 2013:Q1
forecast(armod, newdata = new)
#>   Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
#> 1      2.044155 -2.036225  6.124534 -4.213414  8.301723

```

Using `forecast()` produces the same point forecast of about 2.0, along with 80% and 95% forecast intervals, see section 14.5. We conclude that our AR(1) model forecasts GDP growth to be 2% in 2013:Q1.

How accurate is this forecast? The forecast error is quite large: $GDPGR_{2013:Q1} \approx 1.1\%$ while our forecast is 2%. Second, by calling `summary(armod)` shows that the model explains only little of the variation in the growth rate of GDP and the *SER* is about 3.16. Leaving aside forecast uncertainty due to estimation of the model coefficients β_0 and β_1 , the *RMSFE* must be at least 3.16%, the estimate of the standard deviation of the errors. We conclude that this forecast is pretty inaccurate.

```

# compute the forecast error
forecast(armod, newdata = new)$mean - GDPGrowth["2013"][1]
#>           x
#> 2013 Q1 0.9049532

# R^2
summary(armod)$r.squared
#> [1] 0.1149576

# SER
summary(armod)$sigma
#> [1] 3.15979

```

Autoregressive Models of Order p

For forecasting GDP growth, the AR(1) model (14.2) disregards any information in the past of the series that is more distant than one period. An AR(p) model incorporates the information of p lags of the series. The idea is explained in Key Concept 14.3.

Key Concept 14.3 Autoregressions

An AR(p) model assumes that a time series Y_t can be modeled by a linear function of the first p of its lagged values.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + u_t$$

is an autoregressive model of order p where $E(u_t | Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}) = 0$.

Following the book, we estimate an AR(2) model of the GDP growth series from 1962:Q1 to 2012:Q4.

```
# estimate the AR(2) model
GDPGR_AR2 <- dynlm(ts(GDPGR_level) ~ L(ts(GDPGR_level)) + L(ts(GDPGR_level), 2))

coeftest(GDPGR_AR2, vcov. = sandwich)
#>
#> t test of coefficients:
#>
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      1.631747    0.402023   4.0588 7.096e-05 ***
#> L(ts(GDPGR_level)) 0.277787    0.079250   3.5052 0.0005643 ***
#> L(ts(GDPGR_level), 2) 0.179269    0.079951   2.2422 0.0260560 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimation yields

$$\widehat{GDPGR}_t = \underset{(0.40)}{1.63} + \underset{(0.08)}{0.28}GDPGR_{t-1} + \underset{(0.08)}{0.18}GDPGR_{t-1}. \quad (14.3)$$

We see that the coefficient on the second lag is significantly different from zero. The fit improves slightly: \bar{R}^2 grows from 0.11 for the AR(1) model to about 0.14 and the *SER* reduces to 3.13.

```
# R^2
summary(GDPGR_AR2)$r.squared
#> [1] 0.1425484

# SER
summary(GDPGR_AR2)$sigma
#> [1] 3.132122
```

We may use the AR(2) model to obtain a forecast for GDP growth in 2013:Q1 in the same manner as for the AR(1) model.

```
# AR(2) forecast of GDP growth in 2013:Q1
forecast <- c("2013:Q1" = coef(GDPGR_AR2) %*% c(1, GDPGR_level[N-1],
                                             GDPGR_level[N-2]))
```

This leads to a forecast error of roughly -1% .

```
# compute AR(2) forecast error
GDPGrowth["2013"] [1] - forecast
#>
#> x
#> 2013 Q1 -1.025358
```

14.4 Can You Beat the Market? (Part I)

The theory of efficient capital markets states that stock prices embody all currently available information. If this hypothesis holds, it should not be possible to estimate a useful model for forecasting future stock returns using publicly available information on past returns (this is also referred to as the weak-form efficiency hypothesis): if it was possible to forecast the market, traders would be able to arbitrage, e.g., by relying on an AR(2) model, they would use information that is not already priced-in which would push prices until the expected return is zero.

This idea is presented in the box *Can You Beat the Market? (Part I)* on p. 582 of the book. This section reproduces the estimation results.

We start by importing monthly data from 1931:1 to 2002:12 on excess returns of a broad-based index of stock prices, the CRSP value-weighted index. The data are provided by the authors of the book as an excel sheet which can be downloaded [here](#).

```
# read in data on stock returns
SReturns <- read_xlsx("Data/Stock>Returns_1931_2002.xlsx",
                    sheet = 1,
                    col_types = "numeric")

#Alternatively, ASC can be read directly from internet.

SReturns <- read.csv("https://www.princeton.edu/~mwatson/Stock-Watson_3u/Students/EE_Datasets/Stock_R
                    sep = "\t", header = FALSE )

colnames(SReturns) <- c( "Year", "Month", "ExReturn", "ln_DivYield" )
```

We continue by converting the data to an object of class `ts`.

```
# convert to ts object
StockReturns <- ts(SReturns[, 3:4],
                  start = c(1931, 1),
                  end = c(2002, 12),
                  frequency = 12)
```

Next, we estimate AR(1), AR(2) and AR(4) models of excess returns for the time period 1960:1 to 2002:12.

```
# estimate AR models:

# AR(1)
SR_AR1 <- dynlm(ExReturn ~ L(ExReturn),
               data = StockReturns, start = c(1960, 1), end = c(2002, 12))

# AR(2)
SR_AR2 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2),
               data = StockReturns, start = c(1960, 1), end = c(2002, 12))

# AR(4)
SR_AR4 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2:4),
               data = StockReturns, start = c(1960, 1), end = c(2002, 12))
```

After computing robust standard errors, we gather the results in a table generated by `stargazer()`.

```
# compute robust standard errors
rob_se <- list(sqrt(diag(sandwich(SR_AR1))),
              sqrt(diag(sandwich(SR_AR2))),
              sqrt(diag(sandwich(SR_AR4))))

# generate table using 'stargazer()'
stargazer(SR_AR1, SR_AR2, SR_AR4,
          title = "Autoregressive Models of Monthly Excess Stock Returns",
          header = FALSE,
          model.numbers = F,
          omit.table.layout = "n",
          digits = 3,
```

```

column.labels = c("AR(1)", "AR(2)", "AR(4)"),
dep.var.caption = "Dependent Variable: Excess Returns on the CSRP Value-Weighted Index",
dep.var.labels.include = FALSE,
covariate.labels = c("$excess return_{t-1}$", "$excess return_{t-2}$",
                    "$excess return_{t-3}$", "$excess return_{t-4}$",
                    "Intercept"),
se = rob_se,
omit.stat = "rsq")

```

Table 14.1: Autoregressive Models of Monthly Excess Stock Returns

	Dependent Variable: Excess returns on the CSRP Value-Weighted Index		
	AR(1)	AR(2)	AR(4)
$excessreturn_{t-1}$	0.050 (0.051)	0.053 (0.051)	0.054 (0.051)
$excessreturn_{t-2}$		-0.053 (0.048)	
$excessreturn_{t-3}$			-0.054 (0.048)
$excessreturn_{t-4}$			0.009 (0.050)
Intercept			-0.016 (0.047)
Constant	0.312 (0.197)	0.328* (0.199)	0.331 (0.202)
Observations	516	516	516
Adjusted R ²	0.001	0.001	-0.002
Residual Std. Error	4.334 (df = 514)	4.332 (df = 513)	4.340 (df = 511)
F Statistic	1.306 (df = 1; 514)	1.367 (df = 2; 513)	0.721 (df = 4; 511)

The results are consistent with the hypothesis of efficient financial markets: there are no statistically significant coefficients in any of the estimated models and the hypotheses that all coefficients are zero cannot be rejected. \bar{R}^2 is almost zero in all models and even negative for the AR(4) model. This suggests that none of the models are useful for forecasting stock returns.

14.5 Additional Predictors and The ADL Model

Instead of only using the dependent variable's lags as predictors, an autoregressive distributed lag (ADL) model also uses lags of other variables for forecasting. The general ADL model is summarized in Key Concept 14.4:

Key Concept 14.4 The Autoregressive Distributed Lag Model

An $ADL(p,q)$ model assumes that a time series Y_t can be represented by a linear function of p of its lagged values and q lags of another time series X_t :

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} \\ + \delta_1 X_{t-1} + \delta_2 X_{t-2} + \cdots + \delta_q X_{t-q} + u_t.$$

is an *autoregressive distributed lag model* with p lags of Y_t and q lags of X_t where

$$E(u_t | Y_{t-1}, Y_{t-2}, \dots, X_{t-1}, X_{t-2}, \dots) = 0.$$

Forecasting GDP Growth Using the Term Spread

Interest rates on long-term and short term treasury bonds are closely linked to macroeconomic conditions. While interest rates on both types of bonds have the same long-run tendencies, they behave quite differently in the short run. The difference in interest rates of two bonds with distinct maturity is called the *term spread*.

The following code chunks reproduce Figure 14.3 of the book which displays interest rates of 10-year U.S. Treasury bonds and 3-month U.S. Treasury bills from 1960 to 2012.

```
# 3-month Treasury bills interest rate
TB3MS <- xts(USMacroSWQ$TB3MS, USMacroSWQ$Date) ["1960":2012]

# 10-year Treasury bonds interest rate
TB10YS <- xts(USMacroSWQ$GS10, USMacroSWQ$Date) ["1960":2012]

# term spread
TSpread <- TB10YS - TB3MS

# reproduce Figure 14.2 (a) of the book
plot(merge(as.zoo(TB3MS), as.zoo(TB10YS)),
      plot.type = "single",
      col = c("darkred", "steelblue"),
      lwd = 2,
      xlab = "Date",
      ylab = "Percent per annum",
      main = "Interest Rates")

# define function that transform years to class 'yearqtr'
YToYQTR <- function(years) {
  return(
    sort(as.yearqtr(sapply(years, paste, c("Q1", "Q2", "Q3", "Q4"))))
  )
}

# recessions
recessions <- YToYQTR(c(1961:1962, 1970, 1974:1975, 1980:1982, 1990:1991, 2001,
                       2007:2008))

# add color shading for recessions
```

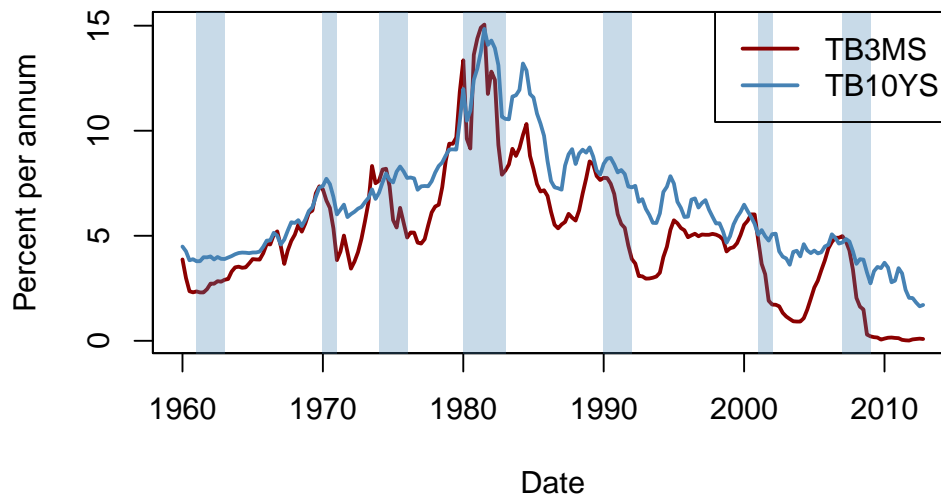
```

xblocks(time(as.zoo(TB3MS)),
        c(time(TB3MS) %in% recessions),
        col = alpha("steelblue", alpha = 0.3))

# add a legend
legend("topright",
      legend = c("TB3MS", "TB10YS"),
      col = c("darkred", "steelblue"),
      lwd = c(2, 2))

```

Interest Rates

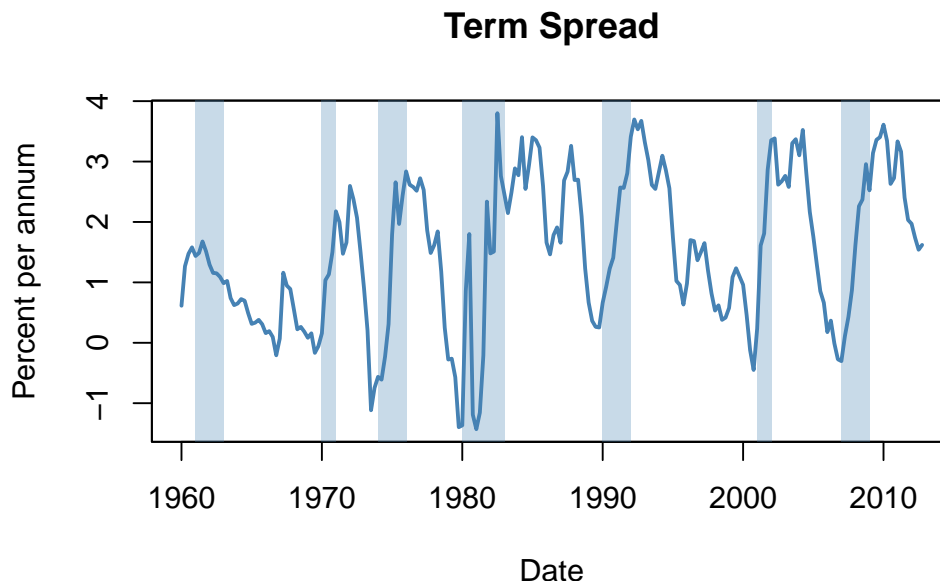


```

# reproduce Figure 14.2 (b) of the book
plot(as.zoo(TSpread),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Percent per annum",
     main = "Term Spread")

# add color shading for recessions
xblocks(time(as.zoo(TB3MS)),
        c(time(TB3MS) %in% recessions),
        col = alpha("steelblue", alpha = 0.3))

```



Before recessions, the gap between interest rates on long-term bonds and short term bills narrows and consequently the term spread declines drastically towards zero or even becomes negative in times of economic stress. This information might be used to improve GDP growth forecasts of future.

We check this by estimating an ADL(2, 1) model and an ADL(2, 2) model of the GDP growth rate using lags of GDP growth and lags of the term spread as regressors. We then use both models for forecasting GDP growth in 2013:Q1.

```
# convert growth and spread series to ts objects
GDPGrowth_ts <- ts(GDPGrowth,
  start = c(1960, 1),
  end = c(2013, 4),
  frequency = 4)

TSpread_ts <- ts(TSpread,
  start = c(1960, 1),
  end = c(2012, 4),
  frequency = 4)

# join both ts objects
ADLdata <- ts.union(GDPGrowth_ts, TSpread_ts)

# estimate the ADL(2,1) model of GDP growth
GDPGR_ADL21 <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) +
  L(TSpread_ts), start = c(1962, 1), end = c(2012, 4))

coeftest(GDPGR_ADL21, vcov. = sandwich)
#>
#> t test of coefficients:
#>
#>
#> Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 0.954990 0.486976 1.9611 0.051260 .
#> L(GDPGrowth_ts) 0.267729 0.082562 3.2428 0.001387 **
#> L(GDPGrowth_ts, 2) 0.192370 0.077683 2.4763 0.014104 *
```

```
#> L(TSpread_ts)      0.444047  0.182637  2.4313  0.015925 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated equation of the ADL(2, 1) model is

$$\widehat{GDPGR}_t = 0.96 + \frac{0.26}{(0.08)} GDPGR_{t-1} + \frac{0.19}{(0.08)} GDPGR_{t-2} + \frac{0.44}{(0.18)} TSpread_{t-1} \quad (14.4)$$

All coefficients are significant at the level of 5%.

```
# 2012:Q3 / 2012:Q4 data on GDP growth and term spread
subset <- window(ADLdata, c(2012, 3), c(2012, 4))

# ADL(2,1) GDP growth forecast for 2013:Q1
ADL21_forecast <- coef(GDPGR_ADL21) %*% c(1, subset[2, 1], subset[1, 1],
                                         subset[2, 2])

ADL21_forecast
#>      [,1]
#> [1,] 2.241689

# compute the forecast error
window(GDPGrowth_ts, c(2013, 1), c(2013, 1)) - ADL21_forecast
#>      Qtr1
#> 2013 -1.102487
```

Model (14.4) predicts the GDP growth in 2013:Q1 to be 2.24% which leads to a forecast error of -1.10%.

We estimate the ADL(2,2) specification to see whether adding additional information on past term spread improves the forecast.

```
# estimate the ADL(2,2) model of GDP growth
GDPGR_ADL22 <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2)
                    + L(TSpread_ts) + L(TSpread_ts, 2),
                    start = c(1962, 1), end = c(2012, 4))

coeftest(GDPGR_ADL22, vcov. = sandwich)
#>
#> t test of coefficients:
#>
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.967967   0.472470   2.0487 0.041800 *
#> L(GDPGrowth_ts)  0.243175   0.077836   3.1242 0.002049 **
#> L(GDPGrowth_ts, 2) 0.177070   0.077027   2.2988 0.022555 *
#> L(TSpread_ts)   -0.139554   0.422162  -0.3306 0.741317
#> L(TSpread_ts, 2)  0.656347   0.429802   1.5271 0.128326
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


We obtain

$$\widehat{GDPGR}_t = 0.98 + 0.24GDPGR_{t-1} + 0.18GDPGR_{t-2} - 0.14TSpread_{t-1} + 0.66TSpread_{t-2}. \quad (14.5)$$

(0.47) (0.08) (0.08) (0.42) (0.43)

The coefficients on both lags of the term spread are not significant at the 10% level.

```
# ADL(2,2) GDP growth forecast for 2013:Q1
ADL22_forecast <- coef(GDPGR_ADL22) %*% c(1, subset[2, 1], subset[1, 1],
                                         subset[2, 2], subset[1, 2])

ADL22_forecast
#>           [,1]
#> [1,] 2.274407

# compute the forecast error
window(GDPGrowth_ts, c(2013, 1), c(2013, 1)) - ADL22_forecast
#>           Qtr1
#> 2013 -1.135206
```

The ADL(2,2) forecast of GDP growth in 2013:Q1 is 2.27% which implies a forecast error of 1.14%.

Do the ADL models (14.4) and (14.5) improve upon the simple AR(2) model (14.3)? The answer is yes: while SER and \bar{R}^2 improve only slightly, an F -test on the term spread coefficients in (14.5) provides evidence that the model does better in explaining GDP growth than the AR(2) model as the hypothesis that both coefficients are zero can be rejected at the level of 5%.

```
# compare adj. R2
c("Adj.R2 AR(2)" = summary(GDPGR_AR2)$adj.r.squared,
  "Adj.R2 ADL(2,1)" = summary(GDPGR_ADL21)$adj.r.squared,
  "Adj.R2 ADL(2,2)" = summary(GDPGR_ADL22)$adj.r.squared)
#>   Adj.R2 AR(2) Adj.R2 ADL(2,1) Adj.R2 ADL(2,2)
#>   0.1338873   0.1620156   0.1691531

# compare SER
c("SER AR(2)" = summary(GDPGR_AR2)$sigma,
  "SER ADL(2,1)" = summary(GDPGR_ADL21)$sigma,
  "SER ADL(2,2)" = summary(GDPGR_ADL22)$sigma)
#>   SER AR(2) SER ADL(2,1) SER ADL(2,2)
#>   3.132122   3.070760   3.057655

# F-test on coefficients of term spread
linearHypothesis(GDPGR_ADL22,
                 c("L(TSpread_ts)=0", "L(TSpread_ts, 2)=0"),
                 vcov. = sandwich)
#> Linear hypothesis test
#>
#> Hypothesis:
#> L(TSpread_ts) = 0
#> L(TSpread_ts, 2) = 0
#>
#> Model 1: restricted model
#> Model 2: GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) + L(TSpread_ts) +
#>           L(TSpread_ts, 2)
```

```

#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F Pr(>F)
#> 1     201
#> 2     199  2 4.4344 0.01306 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Stationarity

In general, forecasts can be improved by using multiple predictors — just as in cross-sectional regression. When constructing time series models one should take into account whether the variables are *stationary* or *nonstationary*. Key Concept 14.5 explains what stationarity is.

Key Concept 14.5 Stationarity

A time series Y_t is stationary if its probability distribution is time independent, that is, the joint distribution of $Y_{s+1}, Y_{s+2}, \dots, Y_{s+T}$ does not change as s is varied, regardless of T .

Similarly, two time series X_t and Y_t are *jointly stationary* if the joint distribution of $(X_{s+1}, Y_{s+1}, X_{s+2}, Y_{s+2}, \dots, X_{s+T}, Y_{s+T})$ does not depend on s , regardless of T .

In a probabilistic sense, stationarity means that information about how a time series evolves in the future is inherent to its past. If this is not the case, we cannot use the past of a series as a reliable guideline for its future.

Stationarity makes it easier to learn about the characteristics of past data.

Time Series Regression with Multiple Predictors

The concept of stationarity is a key assumption in the general time series regression model with multiple predictors. Key Concept 14.6 lays out this model and its assumptions.

Key Concept 14.6 Time Series Regression with Multiple Predictors

The general time series regression model extends the ADL model such that multiple regressors and their lags are included. It uses p lags of the dependent variable and q_l lags of l additional predictors where $l = 1, \dots, k$:

$$\begin{aligned}
 Y_t = & \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} \\
 & + \delta_{11} X_{1,t-1} + \delta_{12} X_{1,t-2} + \dots + \delta_{1q} X_{1,t-q} \\
 & + \dots \\
 & + \delta_{k1} X_{k,t-1} + \delta_{k2} X_{k,t-2} + \dots + \delta_{kq} X_{k,t-q} \\
 & + u_t.
 \end{aligned} \tag{14.6}$$

For estimation we make the following assumptions:

1. The error term u_t has conditional mean zero given all regressors and their lags:

$$E(u_t | Y_{t-1}, Y_{t-2}, \dots, X_{1,t-1}, X_{1,t-2}, \dots, X_{k,t-1}, X_{k,t-2}, \dots) = 0$$

This assumption is an extension of the conditional mean zero assumption used for AR and ADL models and guarantees that the general time series regression model stated above gives the best forecast of Y_t given its lags, the additional regressors $X_{1,t}, \dots, X_{k,t}$ and their lags.

2. The i.i.d. assumption for cross-sectional data is not (entirely) meaningful for time series data. We replace it by the following assumption which consists of two parts:
 - (a) The $(Y_t, X_{1,t}, \dots, X_{k,t})$ have a stationary distribution (the "identically distributed" part of the i.i.d. assumption for cross-sectional data). If this does not hold, forecasts may be biased and inference can be strongly misleading.
 - (b) $(Y_t, X_{1,t}, \dots, X_{k,t})$ and $(Y_{t-j}, X_{1,t-j}, \dots, X_{k,t-j})$ become independent as j gets large (the "independently" distributed part of the i.i.d. assumption for cross-sectional data). This assumption is also called *weak dependence*. It ensures that the WLLN and the CLT hold in large samples.
3. Large outliers are unlikely: $E(X_{1,t}^4), E(X_{2,t}^4), \dots, E(X_{k,t}^4)$ and $E(Y_t^4)$ have nonzero, finite fourth moments.
4. No perfect multicollinearity.

Since many economic time series appear to be nonstationary, assumption two of Key Concept 14.6 is a crucial one in applied macroeconomics and finance which is why statistical test for stationarity or nonstationarity have been developed. Chapters 14.6 and 14.7 are devoted to this topic.

Statistical inference and the Granger causality test

If a X is a useful predictor for Y , in a regression of Y_t on lags of its own and lags of X_t , not all of the coefficients on the lags on X_t are zero. This concept is called *Granger causality* and is an interesting hypothesis to test. Key Concept 14.7 summarizes the idea.

Key Concept 14.7 Granger Causality Tests

The Granger causality test (Granger, 1969) is an F test of the null hypothesis that *all* lags of a variable X included in a time series regression model do not have predictive power for Y_t . The Granger causality test does not test whether X actually *causes* Y but whether the included lags are informative in terms of predicting Y .

We have already performed a Granger causality test on the coefficients of term spread in (14.5), the ADL(2,2) model of GDP growth and concluded that at least one of the first two lags of term spread has predictive power for GDP growth.

Forecast Uncertainty and Forecast Intervals

In general, it is good practice to report a measure of the uncertainty when presenting results that are affected by the latter. Uncertainty is particularly of interest when forecasting a time series. For example, consider a simple ADL(1, 1) model

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \delta_1 X_{t-1} + u_t,$$

where u_t is a homoskedastic error term. The forecast error is

$$Y_{T+1} - \hat{Y}_{T+1|T} = u_{T+1} - \left[(\hat{\beta}_0 - \beta_0) + (\hat{\beta}_1 - \beta_1) Y_T + (\hat{\delta}_1 - \delta_1) X_T \right].$$

The mean squared forecast error (MSFE) and the RMSFE are

$$\begin{aligned} MSFE &= E \left[(Y_{T+1} - \hat{Y}_{T+1|T})^2 \right] \\ &= \sigma_u^2 + Var \left[(\hat{\beta}_0 - \beta_0) + (\hat{\beta}_1 - \beta_1) Y_T + (\hat{\delta}_1 - \delta_1) X_T \right], \\ RMSFE &= \sqrt{\sigma_u^2 + Var \left[(\hat{\beta}_0 - \beta_0) + (\hat{\beta}_1 - \beta_1) Y_T + (\hat{\delta}_1 - \delta_1) X_T \right]}. \end{aligned}$$

A 95% forecast interval is an interval that covers the true value of Y_{T+1} in 95% of repeated applications. There is a major difference in computing a confidence interval and a forecast interval: when computing a confidence interval of a point estimate we use large sample approximations that are justified by the CLT and thus are valid for a large range of error term distributions. For computation of a forecast interval of Y_{T+1} , however, we must make an additional assumption about the distribution of u_{T+1} , the error term in period $T + 1$. Assuming that u_{T+1} is normally distributed one can construct a 95% forecast interval for Y_{T+1} using $SE(Y_{T+1} - \hat{Y}_{T+1|T})$, an estimate of the RMSFE:

$$\hat{Y}_{T+1|T} \pm 1.96 \cdot SE(Y_{T+1} - \hat{Y}_{T+1|T})$$

Of course, the computation gets more complicated when the error term is heteroskedastic or if we are interested in computing a forecast interval for $T + s, s > 1$.

In some applications it is useful to report multiple forecast intervals for subsequent periods, see the box *The River of Blood* on p. 592 of the book. These can be visualized in a so-called fan chart. We will not replicate the fan chart presented in Figure 14.2 of book because the underlying model is by far more complex than the simple AR and ADL models treated here. Instead, in the example below we use simulated time series data and estimate an AR(2) model which is then used for forecasting the subsequent 25 future outcomes of the series.

```

# set seed
set.seed(1234)

# simulate the time series
Y <- arima.sim(list(order = c(2, 0, 0), ar = c(0.2, 0.2)), n = 200)

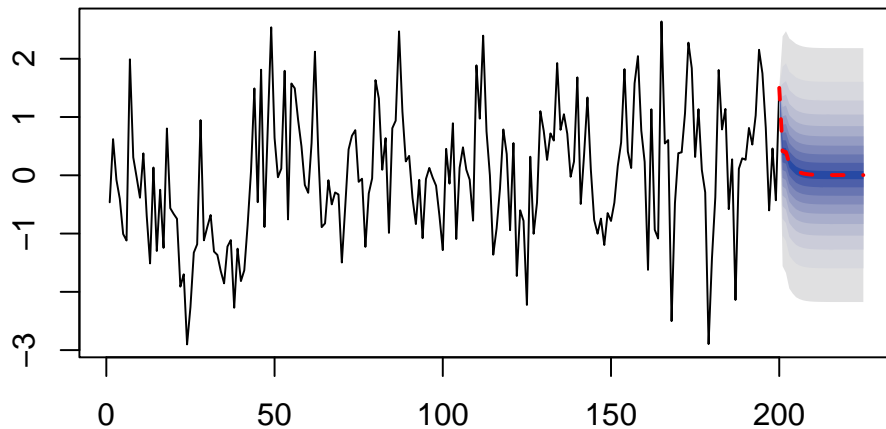
# estimate an AR(2) model using 'arima()', see ?arima
model <- arima(Y, order = c(2, 0, 0))

# compute points forecasts and prediction intervals for the next 25 periods
fc <- forecast(model, h = 25, level = seq(5, 99, 10))

# plot a fan chart
plot(fc,
     main = "Forecast Fan Chart for AR(2) Model of Simulated Data",
     showgap = F,
     fcol = "red",
     flty = 2)

```

Forecast Fan Chart for AR(2) Model of Simulated Data



`arima.sim()` simulates autoregressive integrated moving average (ARIMA) models. AR models belong to this class of models. We use `list(order = c(2, 0, 0), ar = c(0.2, 0.2))` so the DGP is

$$Y_t = 0.2Y_{t-1} + 0.2Y_{t-2} + u_t.$$

We choose `level = seq(5, 99, 10)` in the call of `forecast()` such that forecast intervals with levels 5%, 15%, ..., 95% are computed for each point forecast of the series.

The dashed red line shows point forecasts of the series for the next 25 periods based on an $AR(2)$ model and the shaded areas represent the prediction intervals. The degree of shading indicates the level of the prediction interval. The darkest of the blue bands displays the 5% forecast intervals and the color fades towards grey as the level of the intervals increases.

14.6 Lag Length Selection using Information Criteria

The selection of lag lengths in AR and ADL models can sometimes be guided by economic theory. However, there are statistical methods that are helpful to determine how many lags should be included

as regressors. In general, too many lags inflate the standard errors of coefficient estimates and thus imply an increase in the forecast error while omitting lags that should be included in the model may result in an estimation bias.

The order of an AR model can be determined using two approaches:

1. The F-test approach

Estimate an AR(p) model and test the significance of the largest lag(s). If the test indicates that a particular lag(s) is not significant, we can consider removing it from the model. This approach has the tendency to produce models where the order is too large: in a significance test we always face the risk of rejecting a true null hypothesis!

2. Relying on an information criterion

To circumvent the issue of producing too large models, one may choose the lag order that minimizes one of the following two information criteria:

- The *Bayes information criterion* (BIC):

$$BIC(p) = \log \left(\frac{SSR(p)}{T} \right) + (p + 1) \frac{\log(T)}{T}.$$

- The *Akaike information criterion* (AIC):

$$AIC(p) = \log \left(\frac{SSR(p)}{T} \right) + (p + 1) \frac{2}{T}.$$

Both criteria are estimators of the optimal lag length p . The lag order \hat{p} that minimizes the respective criterion is called the *BIC estimate* or the *AIC estimate* of the optimal model order. The basic idea of both criteria is that the *SSR* decreases as additional lags are added to the model such that the first term decreases whereas the second increases as the lag order grows. One can show that the *BIC* is a consistent estimator of the true lag order while the *AIC* is not which is due to the differing factors in the second addend. Nevertheless, both estimators are used in practice where the *AIC* is sometimes used as an alternative when the *BIC* yields a model with “too few” lags.

The function `dynlm()` does not compute information criteria by default. We will therefore write a short function that reports the *BIC* (along with the chosen lag order p and \bar{R}^2) for objects of class `dynlm`.

```
# compute BIC for AR model objects of class 'dynlm'
BIC <- function(model) {

  ssr <- sum(model$residuals^2)
  t <- length(model$residuals)
  npar <- length(model$coef)

  return(
    round(c("p" = npar - 1,
           "BIC" = log(ssr/t) + npar * log(t)/t,
           "Adj.R2" = summary(model)$adj.r.squared), 4)
  )
}
```

Table 14.3 of the book presents a breakdown of how the *BIC* is computed for AR(p) models of GDP growth with order $p = 1, \dots, 6$. The final result can easily be reproduced using `sapply()` and the function `BIC()` defined above.

```
# apply the BIC() to an intercept-only model of GDP growth
BIC(dynlm(ts(GDPGR_level) ~ 1))
#>      p      BIC Adj.R2
#> 0.0000 2.4394 0.0000

# loop BIC over models of different orders
order <- 1:6

BICs <- sapply(order, function(x)
  "AR" = BIC(dynlm(ts(GDPGR_level) ~ L(ts(GDPGR_level), 1:x))))

BICs
#>      [,1] [,2] [,3] [,4] [,5] [,6]
#> p      1.0000 2.0000 3.0000 4.0000 5.0000 6.0000
#> BIC    2.3486 2.3475 2.3774 2.4034 2.4188 2.4429
#> Adj.R2 0.1099 0.1339 0.1303 0.1303 0.1385 0.1325
```

Note that increasing the lag order increases R^2 because the *SSR* decreases as additional lags are added to the model. However, \bar{R}^2 takes into account the number of parameters in the model and adjusts for the increase in R^2 due to adding more variables, but according to the *BIC*, we should settle for the AR(2) model instead of the AR(5) model. It helps us to decide whether the decrease in *SSR* is enough to justify adding an additional regressor.

If we had to compare a bigger set of models, a convenient way to select the model with the lowest *BIC* is using the function `which.min()`.

```
# select the AR model with the smallest BIC
BICs[, which.min(BICs[2, ])]
#>      p      BIC Adj.R2
#> 2.0000 2.3475 0.1339
```

The *BIC* may also be used to select lag lengths in time series regression models with multiple predictors. In a model with K coefficients, including the intercept, we have

$$BIC(K) = \log\left(\frac{SSR(K)}{T}\right) + K\frac{\log(T)}{T}.$$

Notice that choosing the optimal model according to the *BIC* can be computationally demanding because there may be many different combinations of lag lengths when there are multiple predictors.

To give an example, we estimate ADL(p, q) models of GDP growth where, as above, the additional variable is the term spread between short-term and long-term bonds. We impose the restriction that $p = q_1 = \dots = q_k$ so that only p_{max} models ($p = 1, \dots, p_{max}$) need to be estimated. In the example below we choose $p_{max} = 12$.

```
# loop 'BIC()' over multiple ADL models
order <- 1:12

BICs <- sapply(order, function(x)
  BIC(dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts, 1:x) + L(TSpread_ts, 1:x),
    start = c(1962, 1), end = c(2012, 4))))
```

```

BICs
#>      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]
#> p      2.0000 4.0000 6.0000 8.0000 10.0000 12.0000 14.0000 16.0000 18.0000
#> BIC     2.3411 2.3408 2.3813 2.4181 2.4568 2.5048 2.5539 2.6029 2.6182
#> Adj.R2  0.1332 0.1692 0.1704 0.1747 0.1773 0.1721 0.1659 0.1586 0.1852
#>      [,10]  [,11]  [,12]
#> p      20.0000 22.0000 24.0000
#> BIC     2.6646 2.7205 2.7664
#> Adj.R2  0.1864 0.1795 0.1810

```

From the definition of `BIC()`, for ADL models with $p = q$ it follows that `p` reports the number of estimated coefficients *excluding* the intercept. Thus the lag order is obtained by dividing `p` by 2.

```

# select the ADL model with the smallest BIC
BICs[, which.min(BICs[2, ])]
#>      p      BIC Adj.R2
#> 4.0000 2.3408 0.1692

```

The *BIC* is in favor of the ADL(2,2) model (14.5) we have estimated before.

14.7 Nonstationarity I: Trends

If a series is nonstationary, conventional hypothesis tests, confidence intervals and forecasts can be strongly misleading. The assumption of stationarity is violated if a series exhibits trends or breaks and the resulting complications in an econometric analysis depend on the specific type of the nonstationarity. This section focuses on time series that exhibit trends.

A series is said to exhibit a trend if it has a persistent long-term movement. One can distinguish between *deterministic* and *stochastic* trends as:

- A trend is *deterministic* if it is a nonrandom function of time.
- A trend is said to be *stochastic* if it is a random function of time.

The figures we have produced in Chapter 14.2 reveal that many economic time series show a trending behavior that is probably best modeled by stochastic trends. This is why the book focuses on the treatment of stochastic trends.

The Random Walk Model of a Trend

The simplest way to model a time series Y_t that has stochastic trend is the *random walk*

$$Y_t = Y_{t-1} + u_t, \quad (14.7)$$

where the u_t are i.i.d. errors with $E(u_t|Y_{t-1}, Y_{t-2}, \dots) = 0$. Note that

$$\begin{aligned} E(Y_t|Y_{t-1}, Y_{t-2}, \dots) &= E(Y_{t-1}|Y_{t-1}, Y_{t-2}, \dots) + E(u_t|Y_{t-1}, Y_{t-2}, \dots) \\ &= Y_{t-1} \end{aligned}$$

so the best forecast for Y_t is yesterday's observation Y_{t-1} . Hence the difference between Y_t and Y_{t-1} is unpredictable. The path followed by Y_t consists of random steps u_t , hence it is called a random walk.

Assume that Y_0 , the starting value of the random walk is 0. Another way to write (14.7) is

$$\begin{aligned} Y_0 &= 0 \\ Y_1 &= 0 + u_1 \\ Y_2 &= 0 + u_1 + u_2 \\ &\vdots \\ Y_t &= \sum_{i=1}^t u_i. \end{aligned}$$

Therefore we have

$$\begin{aligned} \text{Var}(Y_t) &= \text{Var}(u_1 + u_2 + \cdots + u_t) \\ &= t\sigma_u^2. \end{aligned}$$

Thus the variance of a random walk depends on t which violates the assumption presented in Key Concept 14.5: a random walk is nonstationary.

Obviously, (14.7) is a special case of an AR(1) model where $\beta_1 = 1$. One can show that a time series that follows an AR(1) model is stationary if $|\beta_1| < 1$. In a general AR(p) model, stationarity is linked to the roots of the polynomial

$$1 - \beta_1 z - \beta_2 z^2 - \beta_3 z^3 - \cdots - \beta_p z^p.$$

If all roots are greater than 1 in absolute value, the AR(p) series is stationary. If at least one root equals 1, the AR(p) is said to have a *unit root* and thus has a stochastic trend.

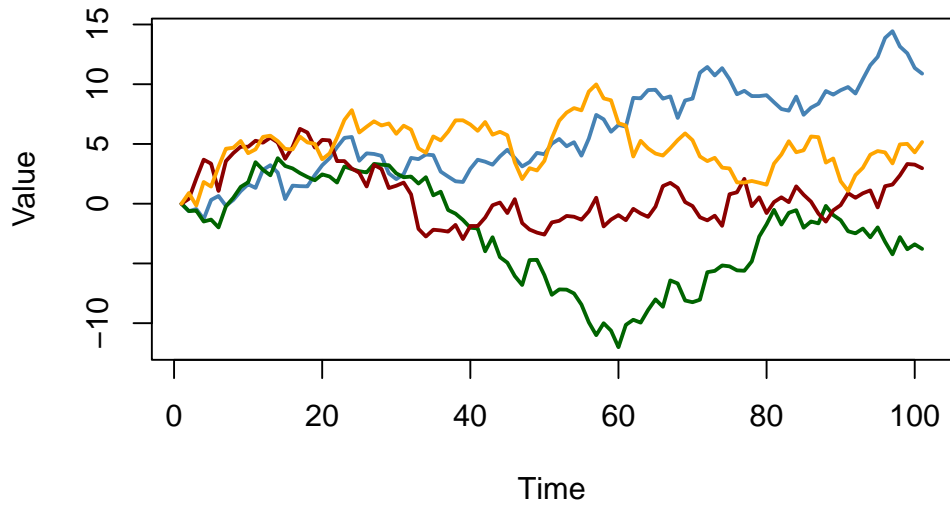
It is straightforward to simulate random walks in R using `arima.sim()`. The function `matplot()` is convenient for simple plots of the columns of a matrix.

```
# simulate and plot random walks starting at 0
set.seed(1)

RWs <- ts(replicate(n = 4,
  arima.sim(model = list(order = c(0, 1, 0)), n = 100)))

matplot(RWs,
  type = "l",
  col = c("steelblue", "darkgreen", "darkred", "orange"),
  lty = 1,
  lwd = 2,
  main = "Four Random Walks",
  xlab = "Time",
  ylab = "Value")
```

Four Random Walks



Adding a constant to (14.7) yields

$$Y_t = \beta_0 + Y_{t-1} + u_t, \quad (14.8)$$

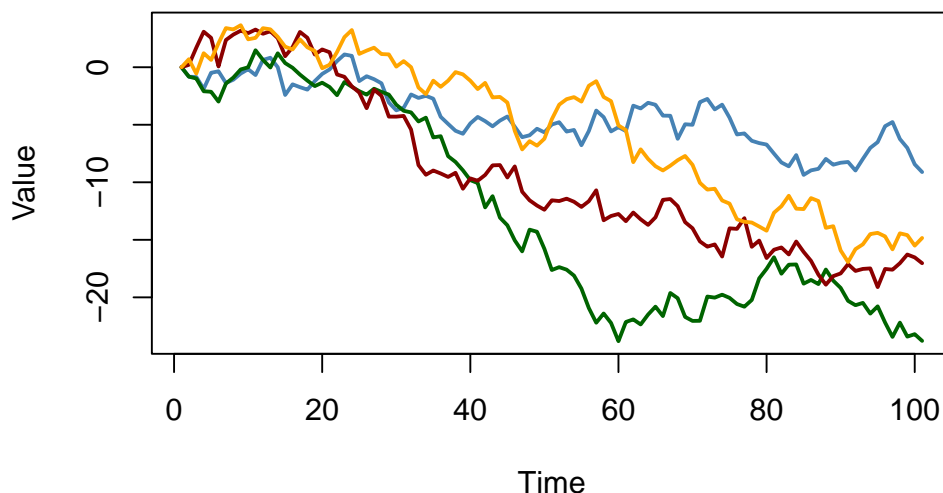
a *random walk model with a drift* which allows to model the tendency of a series to move upwards or downwards. If β_0 is positive, the series drifts upwards and it follows a downward trend if β_0 is negative.

```
# simulate and plot random walks with drift
set.seed(1)

RWsd <- ts(replicate(n = 4,
  arima.sim(model = list(order = c(0, 1, 0)),
    n = 100,
    mean = -0.2)))

matplot(RWsd,
  type = "l",
  col = c("steelblue", "darkgreen", "darkred", "orange"),
  lty = 1,
  lwd = 2,
  main = "Four Random Walks with Drift",
  xlab = "Time",
  ylab = "Value")
```

Four Random Walks with Drift



Problems Caused by Stochastic Trends

OLS estimation of the coefficients on regressors that have a stochastic trend is problematic because the distribution of the estimator and its t -statistic is non-normal, even asymptotically. This has various consequences:

- **Downward bias of autoregressive coefficients:**

If Y_t is a random walk, β_1 can be consistently estimated by OLS but the estimator is biased toward zero. This bias is roughly $E(\hat{\beta}_1) \approx 1 - 5.3/T$ which is substantial for sample sizes typically encountered in macroeconomics. This estimation bias causes forecasts of Y_t to perform worse than a pure random walk model.

- **Non-normally distributed t -statistics:**

The nonnormal distribution of the estimated coefficient of a stochastic regressor translates to a nonnormal distribution of its t -statistic so that normal critical values are invalid and therefore usual confidence intervals and hypothesis tests are invalid, too, and the true distribution of the t -statistic cannot be readily determined.

- **Spurious Regression:**

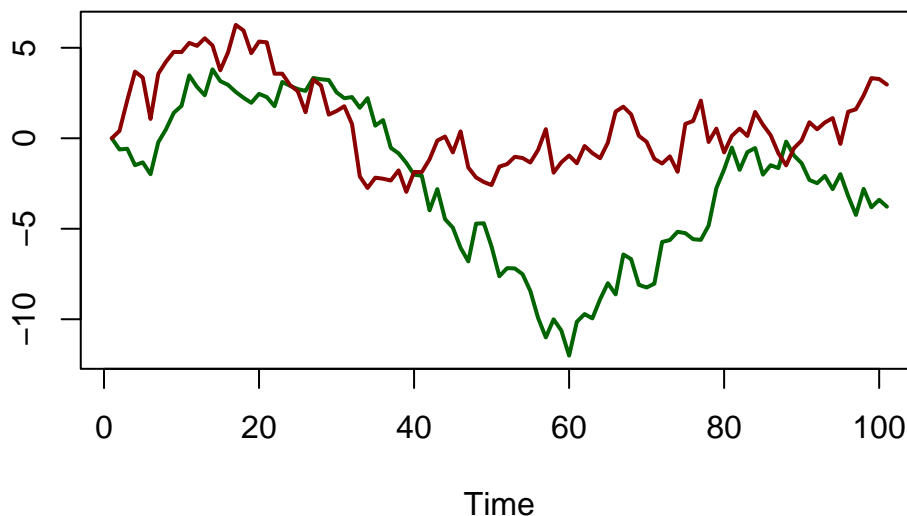
When two stochastically trending time series are regressed onto each other, the estimated relationship may appear highly significant using conventional normal critical values although the series are unrelated. This is what econometricians call a *spurious* relationship.

As an example for spurious regression, consider again the green and the red random walks that we have simulated above. We know that there is no relationship between both series: they are generated independently of each other.

```
# plot spurious relationship
matplotlib(RWs[, c(2, 3)],
  lty = 1,
  lwd = 2,
  type = "l",
  col = c("darkgreen", "darkred"),
  xlab = "Time",
```

```
ylab = "",
main = "A Spurious Relationship")
```

A Spurious Relationship



Imagine we did not have this information and instead conjectured that the green series is useful for predicting the red series and thus end up estimating the ADL(0,1) model

$$Red_t = \beta_0 + \beta_1 Green_{t-1} + u_t.$$

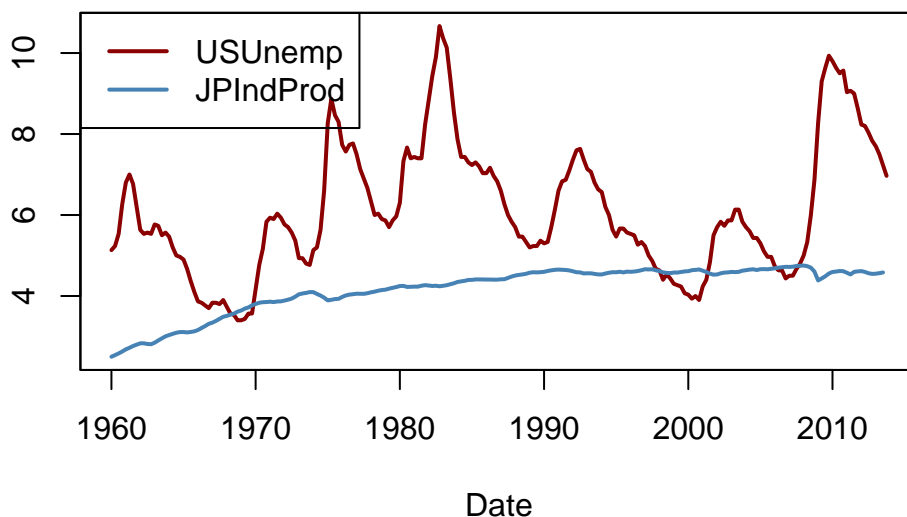
```
# estimate spurious AR model
summary(dynlm(RWs[, 2] ~ L(RWs[, 3]))$coefficients
#>           Estimate Std. Error  t value    Pr(>|t|)
#> (Intercept) -3.459488  0.3635104 -9.516889 1.354156e-15
#> L(RWs[, 3])  1.047195  0.1450874  7.217687 1.135828e-10
```

The result is obviously spurious: the coefficient on $Green_{t-1}$ is estimated to be about 1 and the p -value of $1.14 \cdot 10^{-10}$ of the corresponding t -test indicates that the coefficient is highly significant while its true value is in fact zero.

As an empirical example, consider the U.S. unemployment rate and the Japanese industrial production. Both series show an upward trending behavior from the mid-1960s through the early 1980s.

```
# plot U.S. unemployment rate & Japanese industrial production
plot(merge(as.zoo(USUnemp), as.zoo(JPIndProd)),
      plot.type = "single",
      col = c("darkred", "steelblue"),
      lwd = 2,
      xlab = "Date",
      ylab = "",
      main = "Spurious Regression: Macroeconomic Time series")
# add a legend
legend("topleft",
      legend = c("USUnemp", "JPIndProd"),
      col = c("darkred", "steelblue"),
      lwd = c(2, 2))
```

Spurious Regression: Macroeconomic Time series



```
# estimate regression using data from 1962 to 1985
SR_Unemp1 <- dynlm(ts(USUnemp["1962::1985"]) ~ ts(JPIndProd["1962::1985"]))
coeftest(SR_Unemp1, vcov = sandwich)
#>
#> t test of coefficients:
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      -2.37452   1.12041  -2.1193  0.0367 *
#> ts(JPIndProd["1962::1985"])  2.22057   0.29233   7.5961 2.227e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A simple regression of the U.S. unemployment rate on Japanese industrial production using data from 1962 to 1985 yields

$$\widehat{U.S. UR}_t = -2.37 + \frac{2.22}{(0.29)} \log(\text{Japanese } IP_t). \quad (14.9)$$

This appears to be a significant relationship: the t -statistic of the coefficient on $\log(\text{Japanese } IP_t)$ is bigger than 7.

```
# Estimate regression using data from 1986 to 2012
SR_Unemp2 <- dynlm(ts(USUnemp["1986::2012"]) ~ ts(JPIndProd["1986::2012"]))
coeftest(SR_Unemp2, vcov = sandwich)
#>
#> t test of coefficients:
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         41.7763    5.4066   7.7270 6.596e-12 ***
#> ts(JPIndProd["1986::2012"]) -7.7771    1.1714  -6.6391 1.386e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When estimating the same model, this time with data from 1986 to 2012, we obtain

$$\widehat{U.S.UR}_t = \underset{(5.41)}{41.78} - \underset{(1.17)}{7.78} \log(\text{JapaneseIP})_t \quad (14.10)$$

which surprisingly is quite different. (14.9) indicates a moderate positive relationship, in contrast to the large negative coefficient in (14.10). This phenomenon can be attributed to stochastic trends in the series: since there is no economic reasoning that relates both trends, both regressions may be spurious.

Testing for a Unit AR Root

A formal test for a stochastic trend has been proposed by Dickey and Fuller (1979) which thus is termed the *Dickey-Fuller test*. As discussed above, a time series that follows an AR(1) model with $\beta_1 = 1$ has a stochastic trend. Thus, the testing problem is

$$H_0 : \beta_1 = 1 \quad \text{vs.} \quad H_1 : |\beta_1| < 1.$$

The null hypothesis is that the AR(1) model has a unit root and the alternative hypothesis is that it is stationary. One often rewrites the AR(1) model by subtracting Y_{t-1} on both sides:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t \quad \Leftrightarrow \quad \Delta Y_t = \beta_0 + \delta Y_{t-1} + u_t, \quad (14.11)$$

where $\delta = \beta_1 - 1$. The testing problem then becomes

$$H_0 : \delta = 0 \quad \text{vs.} \quad H_1 : \delta < 0,$$

which is convenient since the corresponding test statistic is reported by many relevant R functions.¹

The Dickey-Fuller test can also be applied in an AR(p) model. The *Augmented Dickey-Fuller (ADF) test* is summarized in Key Concept 14.8.

Key Concept 14.8 The ADF Test for a Unit Root

Consider the regression

$$\Delta Y_t = \beta_0 + \delta Y_{t-1} + \gamma_1 \Delta Y_{t-1} + \gamma_2 \Delta Y_{t-2} + \cdots + \gamma_p \Delta Y_{t-p} + u_t. \quad (14.12)$$

The ADF test for a unit autoregressive root tests the hypothesis $H_0 : \delta = 0$ (stochastic trend) against the one-sided alternative $H_1 : \delta < 0$ (stationarity) using the usual OLS t -statistic.

If it is assumed that Y_t is stationary around a deterministic linear time trend, the model is augmented by the regressor t :

$$\Delta Y_t = \beta_0 + at + \delta Y_{t-1} + \gamma_1 \Delta Y_{t-1} + \gamma_2 \Delta Y_{t-2} + \cdots + \gamma_p \Delta Y_{t-p} + u_t, \quad (14.13)$$

where again $H_0 : \delta = 0$ is tested against $H_1 : \delta < 0$.

The optimal lag length p can be estimated using information criteria. In (14.12), $p = 0$ (no lags of ΔY_t are used as regressors) corresponds to a simple AR(1).

Under the null, the t -statistic corresponding to $H_0 : \delta = 0$ does not have a normal distribution. The critical values can only be obtained from simulation and differ for regressions (14.12) and (14.13) since the distribution of the ADF test statistic is sensitive to the deterministic components included in the regression.

¹The t -statistic of the Dickey-Fuller test is computed using homoskedasticity-only standard errors since under the null hypothesis, the usual t -statistic is robust to conditional heteroskedasticity.

Critical Values for the ADF Statistic

Key Concept 14.8 states that the critical values for the ADF test in the regressions (14.12) and (14.13) can only be determined using simulation. The idea of the simulation study is to simulate a large number of ADF test statistics and use them to estimate quantiles of their *asymptotic* distribution. This section shows how this can be done using R.

First, consider the following AR(1) model with intercept

$$Y_t = \alpha + z_t, \quad z_t = \rho z_{t-1} + u_t.$$

This can be written as

$$Y_t = (1 - \rho)\alpha + \rho y_{t-1} + u_t,$$

i.e., Y_t is a random walk without drift under the null $\rho = 1$. One can show that Y_t is a stationary process with mean α for $|\rho| < 1$.

The procedure for simulating critical values of a unit root test using the t -ratio of δ in (14.11) is as follows:

- Simulate N random walks with n observations using the data generating process

$$Y_t = a + z_t, \quad z_t = \rho z_{t-1} + u_t,$$

$t = 1, \dots, n$ where N and n are large numbers, a is a constant and u is a zero mean error term.

- For each random walk, estimate the regression

$$\Delta Y_t = \beta_0 + \delta Y_{t-1} + u_t$$

and compute the ADF test statistic. Save all N test statistics.

- Estimate quantiles of the distribution of the ADF test statistic using the N test statistics obtained from the simulation.

For the case with drift *and* linear time trend we replace the data generating process by

$$Y_t = a + b \cdot t + z_t, \quad z_t = \rho z_{t-1} + u_t \tag{14.14}$$

where $b \cdot t$ is a linear time trend. Y_t in (14.14) is a random walk with (without) drift if $b \neq 0$ ($b = 0$) under the null of $\rho = 1$ (can you show this?). We estimate the regression

$$\Delta Y_t = \beta_0 + \alpha \cdot t + \delta Y_{t-1} + u_t.$$

Loosely speaking, the precision of the estimated quantiles depends on two factors: n , the length of the underlying series and N , the number of test statistics used. Since we are interested in estimating quantiles of the *asymptotic* distribution (the Dickey-Fuller distribution) of the ADF test statistic both using many observations and large number of simulated test statistics will increase the precision of the estimated quantiles. We choose $n = N = 1000$ as the computational burden grows quickly with n and N .

```
# repetitions
N <- 1000

# observations
n <- 1000
```

```

# define constant, trend and rho
drift <- 0.5
trend <- 1:n
rho <- 1

# function which simulates an AR(1) process
AR1 <- function(rho) {
  out <- numeric(n)
  for(i in 2:n) {
    out[i] <- rho * out[i-1] + rnorm(1)
  }
  return(out)
}

# simulate from DGP with constant
RWD <- ts(replicate(n = N, drift + AR1(rho)))

# compute ADF test statistics and store them in 'ADFD'
ADFD <- numeric(N)

for(i in 1:ncol(RWD)) {
  ADFD[i] <- summary(
    dynlm(diff(RWD[, i], 1) ~ L(RWD[, i], 1)))$coef[2, 3]
}

# simulate from DGP with constant and trend
RWDT <- ts(replicate(n = N, drift + trend + AR1(rho)))

# compute ADF test statistics and store them in 'ADFDT'
ADFDT <- numeric(N)

for(i in 1:ncol(RWDT)) {
  ADFDT[i] <- summary(
    dynlm(diff(RWDT[, i], 1) ~ L(RWDT[, i], 1) + trend(RWDT[, i]))
  )$coef[2, 3]
}

# estimate quantiles for ADF regression with a drift
round(quantile(ADFD, c(0.1, 0.05, 0.01)), 2)
#> 10% 5% 1%
#> -2.62 -2.83 -3.39

# estimate quantiles for ADF regression with drift and trend
round(quantile(ADFDT, c(0.1, 0.05, 0.01)), 2)
#> 10% 5% 1%
#> -3.11 -3.43 -3.97

```

The estimated quantiles are close to the large-sample critical values of the ADF test statistic reported in Table 14.4 of the book.

Table 14.2: Large Sample Critical Values of ADF Test

Deterministic Regressors	10%	5%	1%
Intercept only	-2.57	-2.86	-3.43
Intercept and time trend	-3.12	-3.41	-3.96

The results show that using standard normal critical values is erroneous: the 5% critical value of the standard normal distribution is -1.64 . For the Dickey-Fuller distributions the estimated critical values are -2.87 (drift) and -3.43 (drift and linear time trend). This implies that a true null (the series has a stochastic trend) would be rejected far too often if inappropriate normal critical values were used.

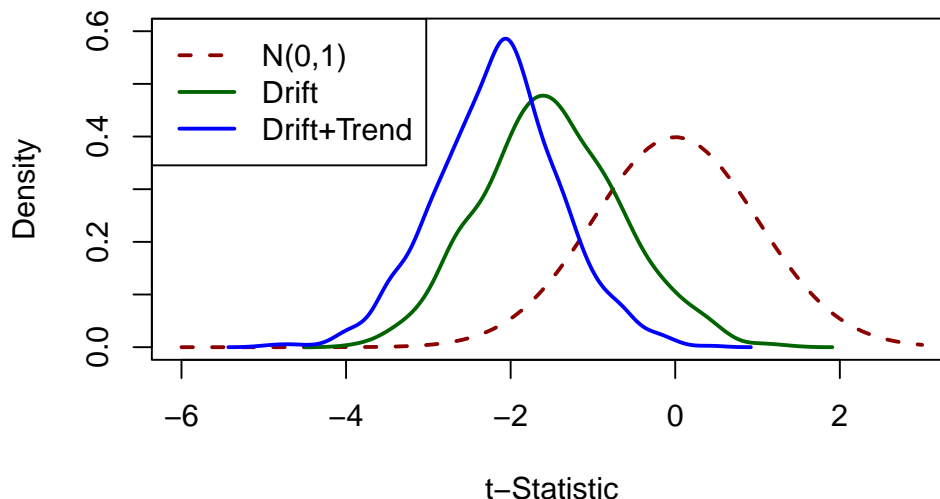
We may use the simulated test statistics for a graphical comparison of the standard normal density and (estimates of) both Dickey-Fuller densities.

```
# plot standard normal density
curve(dnorm(x),
      from = -6, to = 3,
      ylim = c(0, 0.6),
      lty = 2,
      ylab = "Density",
      xlab = "t-Statistic",
      main = "Distributions of ADF Test Statistics",
      col = "darkred",
      lwd = 2)

# plot density estimates of both Dickey-Fuller distributions
lines(density(ADFD), lwd = 2, col = "darkgreen")
lines(density(ADFDT), lwd = 2, col = "blue")

# add a legend
legend("topleft",
      c("N(0,1)", "Drift", "Drift+Trend"),
      col = c("darkred", "darkgreen", "blue"),
      lty = c(2, 1, 1),
      lwd = 2)
```

Distributions of ADF Test Statistics



The deviations from the standard normal distribution are significant: both Dickey-Fuller distributions are skewed to the left and have a heavier left tail than the standard normal distribution.

Does U.S. GDP have a Unit Root?

As an empirical example, we use the ADF test to assess whether there is a stochastic trend in U.S. GDP using the regression

$$\Delta \log(GDP_t) = \beta_0 + \alpha t + \beta_1 \log(GDP_{t-1}) + \beta_2 \Delta \log(GDP_{t-1}) + \beta_3 \Delta \log(GDP_{t-2}) + u_t.$$

```
# generate log GDP series
LogGDP <- ts(log(GDP["1962::2012"]))

# estimate the model
coeftest(
  dynlm(diff(LogGDP) ~ trend(LogGDP, scale = F) + L(LogGDP)
        + diff(L(LogGDP)) + diff(L(LogGDP), 2)))
#>
#> t test of coefficients:
#>
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      0.27877045  0.11793233   2.3638 0.019066 *
#> trend(LogGDP, scale = F) 0.00023818  0.00011090   2.1476 0.032970 *
#> L(LogGDP)        -0.03332452  0.01441436  -2.3119 0.021822 *
#> diff(L(LogGDP))    0.08317976  0.11295542   0.7364 0.462371
#> diff(L(LogGDP), 2)  0.18763384  0.07055574   2.6594 0.008476 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimation yields

$$\begin{aligned} \Delta \log(GDP_t) = & 0.28 + 0.0002t - 0.033 \log(GDP_{t-1}) \\ & \quad \quad \quad (0.118) \quad (0.0001) \quad (0.014) \\ & + 0.083 \Delta \log(GDP_{t-1}) + 0.188 \Delta \log(GDP_{t-2}) + u_t, \\ & \quad \quad \quad (0.113) \quad \quad \quad (0.071) \end{aligned}$$

so the ADF test statistic is $t = -0.033/0.014 = -2.35$. The corresponding 5% critical value from Table 14.2 is -3.41 so we cannot reject the null hypothesis that $\log(GDP)$ has a stochastic trend in favor of the alternative that it is stationary around a deterministic linear time trend.

The ADF test can be done conveniently using `ur.df()` from the package `urca`.

```
# test for unit root in GDP using 'ur.df()' from the package 'urca'
summary(ur.df(LogGDP,
              type = "trend",
              lags = 2,
              selectlags = "Fixed"))

#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root Test #
#> #####
#>
#> Test regression trend
#>
#>
#> Call:
#> lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.025580 -0.004109  0.000321  0.004869  0.032781
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.2790086   0.1180427   2.364 0.019076 *
#> z.lag.1      -0.0333245   0.0144144  -2.312 0.021822 *
#> tt           0.0002382   0.0001109   2.148 0.032970 *
#> z.diff.lag1  0.2708136   0.0697696   3.882 0.000142 ***
#> z.diff.lag2  0.1876338   0.0705557   2.659 0.008476 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.007704 on 196 degrees of freedom
#> Multiple R-squared:  0.1783, Adjusted R-squared:  0.1616
#> F-statistic: 10.63 on 4 and 196 DF,  p-value: 8.076e-08
#>
#>
#> Value of test-statistic is: -2.3119 11.2558 4.267
#>
#> Critical values for test statistics:
#>      1pct  5pct 10pct
#> tau3 -3.99 -3.43 -3.13
#> phi2  6.22  4.75  4.07
#> phi3  8.43  6.49  5.47
```

The first test statistic at the bottom of the output is the one we are interested in. The number of test statistics reported depends on the test regression. For `type = "trend"`, the second statistics corresponds to the test that there is unit root and no time trend while the third one corresponds to a test of the hypothesis that there is a unit root, no time trend and no drift term.

14.8 Nonstationarity II: Breaks

When there are discrete (at a distinct date) or gradual (over time) changes in the population regression coefficients, the series is nonstationary. These changes are called *breaks*. There is a variety of reasons why breaks can occur in macroeconomic time series but most often they are related to changes in economic policy or major changes in the structure of the economy. See Chapter 14.7 of the book for some examples.

If breaks are not accounted for in the regression model, OLS estimates will reflect the average relationship. Since these estimates might be strongly misleading and result in poor forecast quality, we are interested in testing for breaks. One distinguishes between testing for a break when the date is known and testing for a break with an unknown break date.

Let τ denote a known break date and let $D_t(\tau)$ be a binary variable indicating time periods before and after the break. Incorporating the break in an ADL(1,1) regression model yields

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \delta_1 X_{t-1} + \gamma_0 D_t(\tau) + \gamma_1 [D_t(\tau) \cdot Y_{t-1}] \\ + \gamma_2 [D_t(\tau) \cdot X_{t-1}] + u_t,$$

where we allow for discrete changes in β_0 , β_1 and β_2 at the break date τ . The null hypothesis of no break,

$$H_0 : \gamma_0 = \gamma_1 = \gamma_2 = 0,$$

can be tested against the alternative that at least one of the γ 's is not zero using an F -Test. This idea is called a Chow test after Gregory Chow (1960).

When the break date is unknown the *Quandt likelihood ratio (QLR) test* (Quandt, 1960) may be used. It is a modified version of the Chow test which uses the largest of all F -statistics obtained when applying the Chow test for all possible break dates in a predetermined range $[\tau_0, \tau_1]$. The QLR test is summarized in Key Concept 14.9.

Key Concept 14.9 The QLR Test for Coefficient Stability

The QLR test can be used to test for a break in the population regression function if the date of the break is unknown. The QLR test statistic is the largest (Chow) $F(\tau)$ statistic computed over a range of eligible break dates $\tau_0 \leq \tau \leq \tau_1$:

$$QLR = \max [F(\tau_0), F(\tau_0 + 1), \dots, F(\tau_1)]. \quad (14.15)$$

The most important properties are:

- The QLR test can be applied to test whether a subset of the coefficients in the population regression function breaks but the test also rejects if there is a slow evolution of the regression function. When there is a single discrete break in the population regression function that lying at a date within the range tested, the QLR test statistic is $F(\hat{\tau})$ and $\hat{\tau}/T$ is a consistent estimator of the fraction of the sample at which the break is.
- The large-sample distribution of QLR depends on q , the number of restrictions being tested and both ratios of end points to the sample size, $\tau_0/T, \tau_1/T$.
- Similar to the ADF test, the large-sample distribution of QLR is nonstandard. Critical values are presented in Table 14.5 of the book.

Has the Predictive Power of the term spread been stable?

Using the QLR statistic we may test whether there is a break in the coefficients on the lags of the term spread in (14.5), the ADL(2,2) regression model of GDP growth. Following Key Concept 14.9 we modify the specification of (14.5) by adding a break dummy $D(\tau)$ and its interactions with both lags of term spread and choose the range of break points to be tested as 1970:Q1 - 2005:Q2 (these periods are the center 70% of the sample data from 1962:Q2 - 2012:Q4). Thus, the model becomes

$$\begin{aligned} GDPGR_t = & \beta_0 + \beta_1 GDPGR_{t-1} + \beta_2 GDPGR_{t-2} \\ & + \beta_3 TSpread_{t-1} + \beta_4 TSpread_{t-2} \\ & + \gamma_1 D(\tau) + \gamma_2 (D(\tau) \cdot TSpread_{t-1}) \\ & + \gamma_3 (D(\tau) \cdot TSpread_{t-2}) \\ & + u_t. \end{aligned}$$

Next, we estimate the model for each break point and compute the F -statistic corresponding to the null hypothesis $H_0 : \gamma_1 = \gamma_2 = \gamma_3 = 0$. The QLR -statistic is the largest of the F -statistics obtained in this manner.

```
# set up a range of possible break dates
tau <- seq(1970, 2005, 0.25)

# initialize vector of F-statistics
Fstats <- numeric(length(tau))

# estimation loop over break dates
for(i in 1:length(tau)) {

  # set up dummy variable
  D <- time(GDPGrowth_ts) > tau[i]

  # estimate ADL(2,2) model with interactions
  test <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) +
               D*L(TSpread_ts) + D*L(TSpread_ts, 2),
               start = c(1962, 1),
               end = c(2012, 4))

  # compute and save the F-statistic
  Fstats[i] <- linearHypothesis(test,
                               c("DTRUE=0", "DTRUE:L(TSpread_ts)",
                                 "DTRUE:L(TSpread_ts, 2)"),
                               vcov. = sandwich)$F[2]
}
```

We determine the QLR statistic using `max()`.

```
# identify QLR statistic
QLR <- max(Fstats)
QLR
#> [1] 6.651156
```

Let us check that the QLR -statistic is the F -statistic obtained for the regression where 1980:Q4 is chosen as the break date.

```
# identify the time period where the QLR-statistic is observed
as.yearqtr(tau[which.max(Fstats)])
#> [1] "1980 Q4"
```

Since $q = 3$ hypotheses are tested and the central 70% of the sample are considered to contain breaks, the corresponding 1% critical value of the *QLR* test is 6.02. We reject the null hypothesis that all coefficients (the coefficients on both lags of term spread and the intercept) are stable since the computed *QLR*-statistic exceeds this threshold. Thus evidence from the *QLR* test suggests that there is a break in the ADL(2,2) model of GDP growth in the early 1980s.

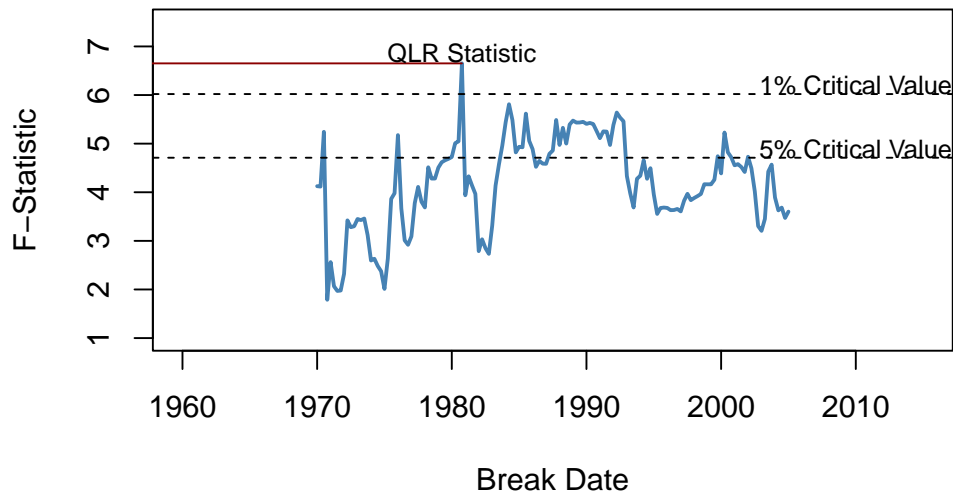
To reproduce Figure 14.5 of the book, we convert the vector of sequential break-point *F*-statistics into a time series object and then generate a simple plot with some annotations.

```
# series of F-statistics
Fstatsseries <- ts(Fstats,
                  start = tau[1],
                  end = tau[length(tau)],
                  frequency = 4)

# plot the F-statistics
plot(Fstatsseries,
     xlim = c(1960, 2015),
     ylim = c(1, 7.5),
     lwd = 2,
     col = "steelblue",
     ylab = "F-Statistic",
     xlab = "Break Date",
     main = "Testing for a Break in GDP ADL(2,2) Regression at Different Dates",
     cex.main=0.8)

# dashed horizontal lines for critical values and QLR statistic
abline(h = 4.71, lty = 2)
abline(h = 6.02, lty = 2)
segments(0, QLR, 1980.75, QLR, col = "darkred")
text(2010, 6.2, "1% Critical Value", cex=0.8)
text(2010, 4.9, "5% Critical Value", cex=0.8)
text(1980.75, QLR+0.2, "QLR Statistic", cex=0.8)
```

Testing for a Break in GDP ADL(2,2) Regression at Different Dates



Pseudo Out-of-Sample Forecasting

Pseudo out-of-sample forecasts are used to simulate the out-of-sample performance (the real time forecast performance) of a time series regression model. In particular, pseudo out-of-sample forecasts allow estimation of the *RMSFE* of the model and enable researchers to compare different model specifications with respect to their predictive power. Key Concept 14.10 summarizes this idea.

Key Concept 14.10
Pseudo Out-of-Sample Forecasting

1. Divide the sample data into $s = T - P$ and P subsequent observations. The P observations are used as pseudo-out-of-sample observations.
2. Estimate the model using the first s observations.
3. Compute the pseudo-forecast $\tilde{Y}_{s+1|s}$.
4. Compute the pseudo-forecast-error $\tilde{u}_{s+1} = Y_{s+1} - \tilde{Y}_{s+1|s}$.
5. Repeat steps 2 through 4 for all remaining pseudo-out-of-sample dates, i.e., reestimate the model at each date.

Did the Predictive Power of the Term Spread Change During the 2000s?

The insight gained in the previous section gives reason to presume that the pseudo-out-of-sample performance of ADL(2,2) models estimated using data *after* the break in the early 1980s should not deteriorate relative to using the whole sample: provided that the coefficients of the population regression function are stable after the potential break in 1980:Q4, these models should have good predictive power. We check this by computing pseudo-out-of-sample forecasts for the period 2003:Q1 - 2012:Q4, a range covering 40 periods, where the forecast for 2003:Q1 is done using data from 1981:Q1 - 2002:Q4, the forecast for 2003:Q2 is based on data from 1981:Q1 - 2003:Q1 and so on.

Similarly as for the *QLR*-test we use a `for()` loop for estimation of all 40 models and gather their *SERs* and the obtained forecasts in a vector which is then used to compute pseudo-out-of-sample forecast errors.

```

# end of sample dates
EndOfSample <- seq(2002.75, 2012.5, 0.25)

# initialize matrix forecasts
forecasts <- matrix(nrow = 1,
                   ncol = length(EndOfSample))

# initialize vector SER
SER <- numeric(length(EndOfSample))

# estimation loop over end of sample dates
for(i in 1:length(EndOfSample)) {

  # estimate ADL(2,2) model
  m <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2)
            + L(TSpread_ts) + L(TSpread_ts, 2),
            start = c(1981, 1),
            end = EndOfSample[i])

  SER[i] <- summary(m)$sigma

  # sample data for one-period ahead forecast
  s <- window(ADLdata, EndOfSample[i] - 0.25, EndOfSample[i])

  # compute forecast
  forecasts[i] <- coef(m) %*% c(1, s[1, 1], s[2, 1], s[1, 2], s[2, 2])
}

# compute pseudo-out-of-sample forecast errors
POOSFCE <- c(window(GDPGrowth_ts, c(2003, 1), c(2012, 4))) - forecasts

```

We next translate the pseudo-out-of-sample forecasts into an object of class `ts` and plot the real GDP growth rate against the forecasted series.

```

# series of pseudo-out-of-sample forecasts
PSOSSFc <- ts(c(forecasts),
             start = 2003,
             end = 2012.75,
             frequency = 4)

# plot the GDP growth time series
plot(window(GDPGrowth_ts, c(2003, 1), c(2012, 4)),
     col = "steelblue",
     lwd = 2,
     ylab = "Percent",
     main = "Pseudo-Out-Of-Sample Forecasts of GDP Growth")

# add the series of pseudo-out-of-sample forecasts
lines(PSOSSFc,
     lwd = 2,
     lty = 2)

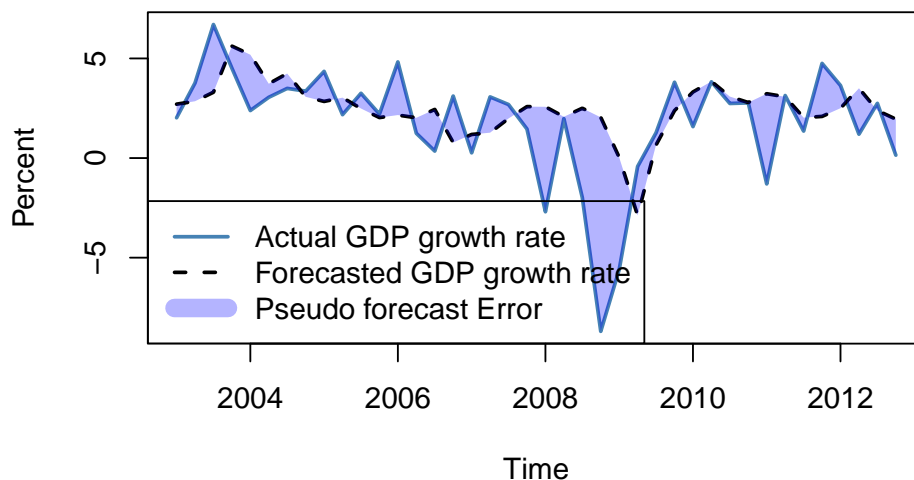
```



```
# shade area between curves (the pseudo forecast error)
polygon(c(time(PSOSSFc), rev(time(PSOSSFc))),
        c(window(GDPGrowth_ts, c(2003, 1), c(2012, 4)), rev(PSOSSFc)),
        col = alpha("blue", alpha = 0.3),
        border = NA)

# add a legend
legend("bottomleft",
       lty = c(1, 2, 1),
       lwd = c(2, 2, 10),
       col = c("steelblue", "black", alpha("blue", alpha = 0.3)),
       legend = c("Actual GDP growth rate",
                  "Forecasted GDP growth rate",
                  "Pseudo forecast Error"))
```

Pseudo-Out-Of-Sample Forecasts of GDP Growth



Apparently, the pseudo forecasts track the actual GDP growth rate quite well, except for the kink in 2009 which can be attributed to the recent financial crisis.

The *SER* of the first model (estimated using data from 1981:Q1 to 2002:Q4) is 2.39 so based on the in-sample fit we would expect the out of sample forecast errors to have mean zero and a root mean squared forecast error of about 2.39.

```
# SER of ADL(2,2) mode using data from 1981:Q1 - 2002:Q4
SER[1]
#> [1] 2.389773
```

The root mean squared forecast error of the pseudo-out-of-sample forecasts is somewhat larger.

```
# compute root mean squared forecast error
sd(POOSFCE)
#> [1] 2.667612
```

An interesting hypothesis is whether the mean forecast error is zero, that is the ADL(2,2) forecasts are right, on average. This hypothesis is easily tested using the function `t.test()`.

```

# test if mean forecast error is zero
t.test(POOSFCE)
#>
#> One Sample t-test
#>
#> data: POOSFCE
#> t = -1.5523, df = 39, p-value = 0.1287
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#> -1.5078876 0.1984001
#> sample estimates:
#> mean of x
#> -0.6547438

```

The hypothesis cannot be rejected at the 10% significance level. Altogether the analysis suggests that the ADL(2,2) model coefficients have been stable since the presumed break in the early 1980s.

14.9 Can You Beat the Market? (Part II)

The dividend yield (the ratio of current dividends to the stock price) can be considered as an indicator of future excess returns: if a stock has a high current dividend yield, it can be considered undervalued and it can be presumed that the price of the stock goes up in the future, meaning that future excess returns go up.

This presumption can be examined using ADL models of excess returns, where lags of the logarithm of the stock's dividend yield serve as additional regressors.

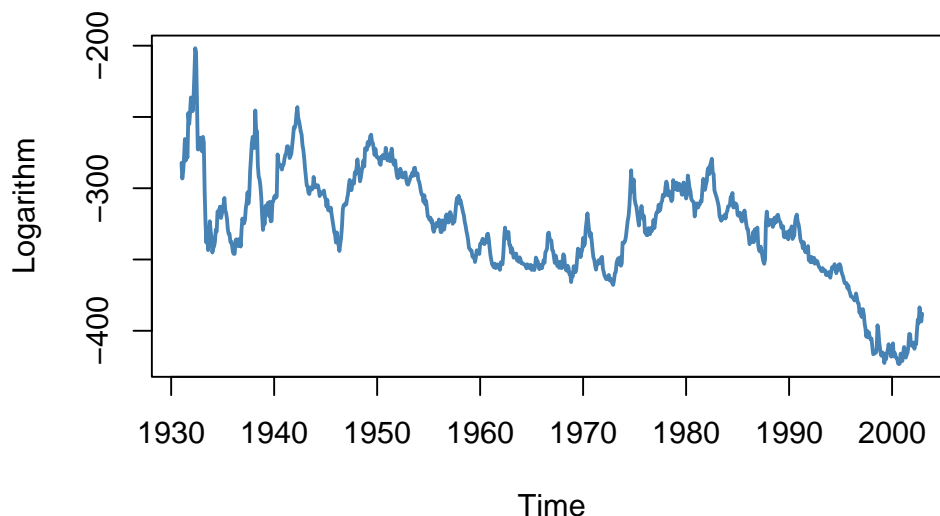
Unfortunately, a graphical inspection of the time series of the logarithm of the dividend yield casts doubt on the assumption that the series is stationary which, as has been discussed in Chapter 14.7, is necessary to conduct standard inference in a regression analysis.

```

# plot logarithm of dividend yield series
plot(StockReturns[, 2],
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     main = "Dividend Yield for CRSP Index")

```

Dividend Yield for CRSP Index



The Dickey-Fuller test statistic for an autoregressive unit root in an AR(1) model with drift provides further evidence that the series might be nonstationary.

```
# test for unit root in GDP using 'ur.df()' from the package 'urca'
summary(ur.df(window(StockReturns[, 2],
                    c(1960,1),
                    c(2002, 12)),
            type = "drift",
            lags = 0))

#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root Test #
#> #####
#>
#> Test regression drift
#>
#>
#> Call:
#> lm(formula = z.diff ~ z.lag.1 + 1)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -14.3540  -2.9118  -0.2953   2.6375  25.5169
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -2.740964   2.080038  -1.318   0.188
#> z.lag.1     -0.007652   0.005989  -1.278   0.202
#>
#> Residual standard error: 4.45 on 513 degrees of freedom
#> Multiple R-squared:  0.003172, Adjusted R-squared:  0.001229
#> F-statistic: 1.633 on 1 and 513 DF, p-value: 0.2019
#>
```

```
#>
#> Value of test-statistic is: -1.2777 0.9339
#>
#> Critical values for test statistics:
#>      1pct  5pct 10pct
#> tau2 -3.43 -2.86 -2.57
#> phi1  6.43  4.59  3.78
```

We use `window()` to get observations from January 1960 to December 2002 only.

Since the t -value for the coefficient on the lagged logarithm of the dividend yield is -1.27 , the hypothesis that the true coefficient is zero cannot be rejected, even at the 10% significance level.

However, it is possible to examine whether the dividend yield has predictive power for excess returns by using its differences in an ADL(1,1) and an ADL(2,2) model (remember that differencing a series with a unit root yields a stationary series), although these model specifications do not correspond to the economic reasoning mentioned above. Thus, we also estimate an ADL(1,1) regression using the level of the logarithm of the dividend yield.

That is we estimate three different specifications:

$$(I) \text{ excess returns}_t = \beta_0 + \beta_1 \text{ excess returns}_{t-1} + \beta_3 \Delta \log(\text{dividendyield}_{t-1}) + u_t$$

$$(II) \text{ excess returns}_t = \beta_0 + \beta_1 \text{ excess returns}_{t-1} + \beta_2 \text{ excess returns}_{t-2} \\ + \beta_3 \Delta \log(\text{dividendyield}_{t-1}) + \beta_4 \Delta \log(\text{dividendyield}_{t-2}) + u_t$$

$$(III) \text{ excess returns}_t = \beta_0 + \beta_1 \text{ excess returns}_{t-1} + \beta_5 \log(\text{dividendyield}_{t-1}) + u_t$$

```
# ADL(1,1) (1st difference of log dividend yield)
CRSP_ADL_1 <- dynlm(ExReturn ~ L(ExReturn) + d(L(ln_DivYield)),
  data = StockReturns,
  start = c(1960, 1), end = c(2002, 12))

# ADL(2,2) (1st & 2nd differences of log dividend yield)
CRSP_ADL_2 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2)
  + d(L(ln_DivYield)) + d(L(ln_DivYield, 2))),
  data = StockReturns,
  start = c(1960, 1), end = c(2002, 12))

# ADL(1,1) (level of log dividend yield)
CRSP_ADL_3 <- dynlm(ExReturn ~ L(ExReturn) + L(ln_DivYield),
  data = StockReturns,
  start = c(1960, 1), end = c(2002, 12))

# gather robust standard errors
rob_se_CRSP_ADL <- list(sqrt(diag(sandwich(CRSP_ADL_1))),
  sqrt(diag(sandwich(CRSP_ADL_2))),
  sqrt(diag(sandwich(CRSP_ADL_3))))
```

A tabular representation of the results can then be generated using `stargazer()`.

```

stargazer(CRSP_ADL_1, CRSP_ADL_2, CRSP_ADL_3,
  title = "ADL Models of Monthly Excess Stock Returns",
  header = FALSE,
  type = "latex",
  column.sep.width = "-5pt",
  no.space = T,
  digits = 3,
  column.labels = c("ADL(1,1)", "ADL(2,2)", "ADL(1,1)"),
  dep.var.caption = "Dependent Variable: Excess returns on the CRSP value-weighted index",
  dep.var.labels.include = FALSE,
  covariate.labels = c("$excess return_{t-1}$",
    "$excess return_{t-2}$",
    "$1^{st}$ diff log(dividend yield_{t-1})$",
    "$1^{st}$ diff log(dividend yield_{t-2})$",
    "$log(dividend yield_{t-1})$",
    "Constant"),
  se = rob_se_CRSP_ADL)

```

Table 14.3: ADL Models of Monthly Excess Stock Returns

	Dependent Variable: Excess returns on the CRSP Value-Weighted Index		
	ADL(1,1)	ADL(2,2)	ADL(1,1)
	(1)	(2)	(3)
$excessreturn_{t-1}$	0.059 (0.158)	0.042 (0.162)	0.055 (0.051)
$excessreturn_{t-2}$		-0.213 (0.193)	
$1^{st} \text{ diff log}(\text{dividend yield}_{t-1})$	0.009 (0.157)	-0.012 (0.163)	
$1^{st} \text{ diff log}(\text{dividend yield}_{t-2})$		-0.161 (0.185)	
$\text{log}(\text{dividend yield}_{t-1})$			0.010 (0.006)
Constant	0.309 (0.199)	0.372* (0.208)	3.892* (2.222)
Observations	516	516	516
R ²	0.003	0.007	0.009
Adjusted R ²	-0.001	-0.001	0.005
Residual Std. Error	4.338 (df = 513)	4.337 (df = 511)	4.325 (df = 513)
F Statistic	0.653 (df = 2; 513)	0.897 (df = 4; 511)	2.234 (df = 2; 513)

Note:

*p<0.1; **p<0.05; ***p<0.01

For models (1) and (2) none of the individual t -statistics suggest that the coefficients are different from zero. Also, we cannot reject the hypothesis that none of the lags have predictive power for excess returns at any common level of significance (an F -test that the lags have predictive power does not reject for both models).

Things are different for model (3). The coefficient on the level of the logarithm of the dividend yield is different from zero at the 5% level and the F -test rejects, too. But we should be suspicious: the high

degree of persistence in the dividend yield series probably renders this inference dubious because t - and F -statistics may follow distributions that deviate considerably from their theoretical large-sample distributions such that the usual critical values cannot be applied.

If model (3) were of use for predicting excess returns, pseudo-out-of-sample forecasts based on (3) should at least outperform forecasts of an intercept-only model in terms of the sample RMSFE. We can perform this type of comparison using R code in the fashion of the applications of Chapter 14.8.

```
# end of sample dates
EndOfSample <- as.numeric(window(time(StockReturns), c(1992, 12), c(2002, 11)))

# initialize matrix forecasts
forecasts <- matrix(nrow = 2,
                    ncol = length(EndOfSample))

# estimation loop over end of sample dates
for(i in 1:length(EndOfSample)) {

  # estimate model (3)
  mod3 <- dynlm(ExReturn ~ L(ExReturn) + L(ln_DivYield), data = StockReturns,
               start = c(1960, 1),
               end = EndOfSample[i])

  # estimate intercept only model
  modconst <- dynlm(ExReturn ~ 1, data = StockReturns,
                  start = c(1960, 1),
                  end = EndOfSample[i])

  # sample data for one-period ahead forecast
  t <- window(StockReturns, EndOfSample[i], EndOfSample[i])

  # compute forecast
  forecasts[, i] <- c(coef(mod3) %*% c(1, t[1], t[2]), coef(modconst))

}

# gather data
d <- cbind("Excess Returns" = c(window(StockReturns[,1], c(1993, 1), c(2002, 12))),
          "Model (3)" = forecasts[1,],
          "Intercept Only" = forecasts[2,],
          "Always Zero" = 0)

# Compute RMSFEs
c("ADL model (3)" = sd(d[, 1] - d[, 2]),
  "Intercept-only model" = sd(d[, 1] - d[, 3]),
  "Always zero" = sd(d[,1] - d[, 4]))
#>      ADL model (3) Intercept-only model      Always zero
#>      4.043754      4.000218      3.995425
```

The comparison indicates that model (3) is not useful since it is outperformed in terms of sample RMSFE by the intercept-only model. A model forecasting excess returns always to be zero has an even lower sample RMSFE. This finding is consistent with the weak-form efficiency hypothesis which states that all publicly available information is accounted for in stock prices such that there is no way

to predict future stock prices or excess returns using past observations, implying that the perceived significant relationship indicated by model (3) is wrong.

Summary

This chapter dealt with introductory topics in time series regression analysis, where variables are generally correlated from one observation to the next, a concept termed serial correlation. We presented several ways of storing and plotting time series data using **R** and used these for informal analysis of economic data.

We have introduced AR and ADL models and applied them in the context of forecasting of macroeconomic and financial time series using **R**. The discussion also included the topic of lag length selection. It was shown how to set up a simple function that computes the BIC for a model object supplied.

We have also seen how to write simple **R** code for performing and evaluating forecasts and demonstrated some more sophisticated approaches to conduct pseudo-out-of-sample forecasts for assessment of a model's predictive power for unobserved future outcomes of a series, to check model stability and to compare different models.

Furthermore, some more technical aspects like the concept of stationarity were addressed. This included applications to testing for an autoregressive unit root with the Dickey-Fuller test and the detection of a break in the population regression function using the *QLR* statistic. For both methods, the distribution of the relevant test statistic is non-normal, even in large samples. Concerning the Dickey-Fuller test we have used **R**'s random number generation facilities to produce evidence for this by means of a Monte-Carlo simulation and motivated usage of the quantiles tabulated in the book.

Also, empirical studies regarding the validity of the weak and the strong form efficiency hypothesis which are presented in the applications *Can You Beat the Market? Part I & II* in the book have been reproduced using **R**.

In all applications of this chapter, the focus was on forecasting future outcomes rather than estimation of causal relationships between time series variables. However, the methods needed for the latter are quite similar. Chapter 15 is devoted to estimation of so called *dynamic causal effects*.

Chapter 15

Estimation of Dynamic Causal Effects

It sometimes is of interest to know the size of current and future reaction of Y to a change in X . This is called the *dynamic causal effect* on Y of a change in X . In this Chapter we discuss how to estimate dynamic causal effects in R applications, where we investigate the dynamic effect of cold weather in Florida on the price of orange juice concentrate.

The discussion covers:

- estimation of distributed lag models,
- heteroskedasticity- and autocorrelation-consistent (HAC) standard errors,
- generalized least squares (GLS) estimation of ADL models.

To reproduce code examples, install the R packages listed below beforehand and make sure that the subsequent code chunk executes without any errors.

- `AER` (Kleiber and Zeileis, 2008b),
- `dynlm` (Zeileis, 2019),
- `nlme` (Pinheiro et al., 2023),
- `orcutt` (Stefano et al., 2018),
- `quantmod` (Ryan and Ulrich, 2023),
- `stargazer` (Hlavac, 2022).

```
library(AER)
library(quantmod)
library(dynlm)
library(orcutt)
library(nlme)
library(stargazer)
```

15.1 The Orange Juice Data

The largest cultivation region for oranges in the U.S. is located in Florida which usually has ideal climate for the fruit growth. It thus is the source of almost all frozen juice concentrate produced in the country. However, from time to time and depending on their severeness, cold snaps cause a loss of harvests such that the supply of oranges decreases and consequently the price of frozen juice concentrate rises. The timing of the price increases is complicated: a cut in today's supply of concentrate influences

not just today's price but also future prices because supply in future periods will decrease, too. Clearly, the magnitude of today's and future prices increases due to freeze is an empirical question that can be investigated using a distributed lag model — a time series model that relates price changes to weather conditions.

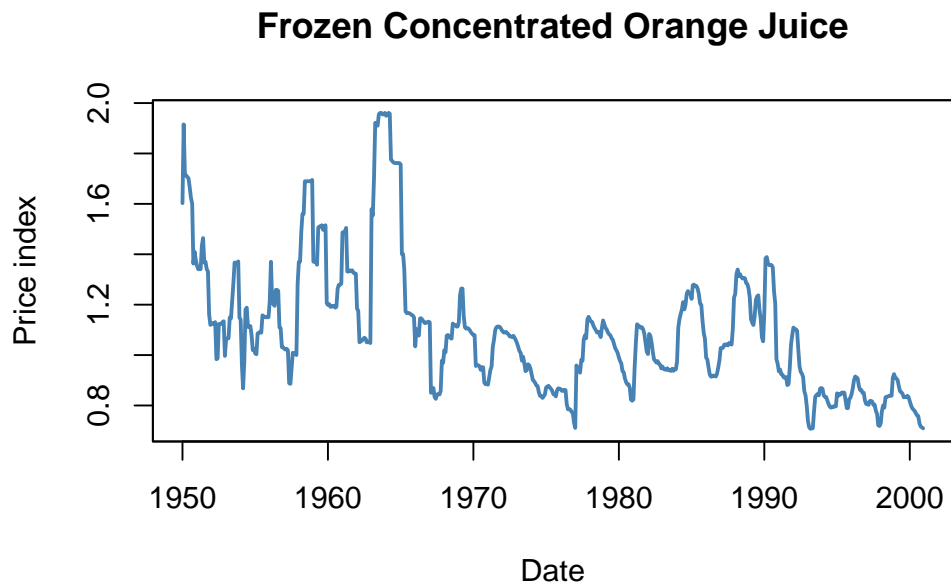
To begin with the analysis, we reproduce Figure 15.1 of the book which displays plots of the price index for frozen concentrated orange juice, percentage changes in the price as well as monthly freezing degree days in Orlando, the center of Florida's orange-growing region.

```
# load the frozen orange juice data set
data("FrozenJuice")

# compute the price index for frozen concentrated juice
FOJCPI <- FrozenJuice[, "price"]/FrozenJuice[, "ppi"]
FOJC_pctc <- 100 * diff(log(FOJCPI))
FDD <- FrozenJuice[, "fdd"]

# convert series to xts objects
FOJCPI_xts <- as.xts(FOJCPI)
FDD_xts <- as.xts(FrozenJuice[, 3])

# Plot orange juice price index
plot(as.zoo(FOJCPI),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Price index",
     main = "Frozen Concentrated Orange Juice")
```

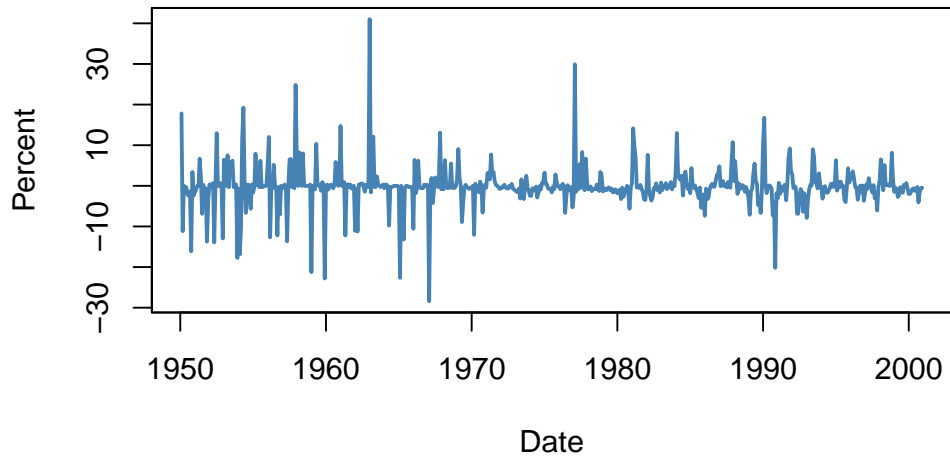


```
# divide plotting area
par(mfrow = c(2, 1))

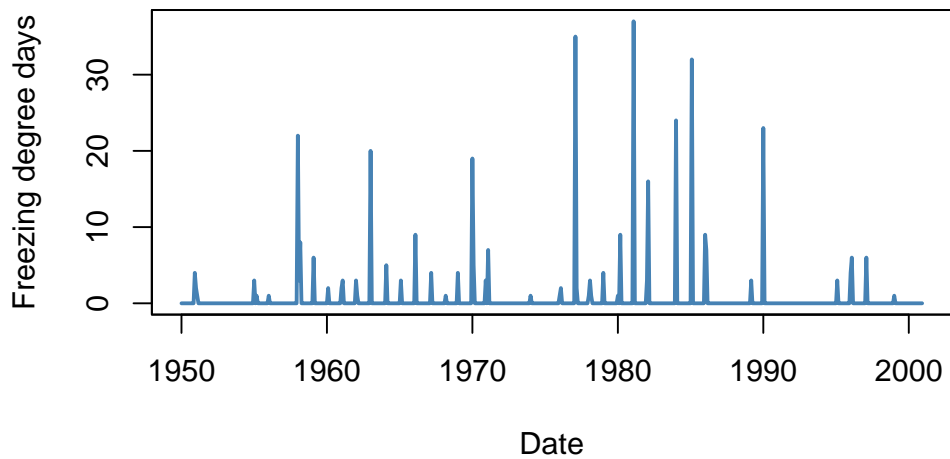
# Plot percentage changes in prices
plot(as.zoo(FOJC_pctc),
     col = "steelblue",
```

```
lwd = 2,  
xlab = "Date",  
ylab = "Percent",  
cex.main=0.8,  
main = "Monthly Changes in the Price of Frozen Concentrated Orange Juice")  
  
# plot freezing degree days  
plot(as.zoo(FDD),  
col = "steelblue",  
lwd = 2,  
xlab = "Date",  
ylab = "Freezing degree days",  
cex.main=0.8,  
main = "Monthly Freezing Degree Days in Orlando, FL")
```

Monthly Changes in the Price of Frozen Concentrated Orange Juice



Monthly Freezing Degree Days in Orlando, FL



Periods with a high amount of freezing degree days are followed by large month-to-month price changes. These coinciding movements motivate a simple regression of price changes ($\%ChgOJC_t$) on freezing degree days (FDD_t) to estimate the effect of an additional freezing degree day on the price in the current month. For this, as for all other regressions in this chapter, we use $T = 611$ observations (January 1950 to December 2000).

```
# simple regression of percentage changes on freezing degree days
orange_SR <- dynlm(FOJC_pctc ~ FDD)
coeftest(orange_SR, vcov. = vcovHAC)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.42095    0.18683 -2.2531 0.0246064 *
#> FDD          0.46724    0.13385  3.4906 0.0005167 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the standard errors are computed using a “HAC” estimator of the variance-covariance matrix, see Chapter 14.5 for a discussion of this estimator.

$$\widehat{\%ChgOJC}_t = -0.42 + 0.47 FDD_t.$$

(0.19) (0.13)

The estimated coefficient on FDD_t has the following interpretation: an additional freezing degree day in month t leads to a price increase of 0.47 percentage points in the same month.

To consider effects of cold snaps on the orange juice price over the subsequent periods, we include lagged values of FDD_t in our model which leads to a *distributed lag regression model*. We estimate a specification using a contemporaneous and six lagged values of FDD_t as regressors.

```
# distributed lag model with 6 lags of freezing degree days
orange_DLM <- dynlm(FOJC_pctc ~ FDD + L(FDD, 1:6))
coeftest(orange_DLM, vcov. = vcovHAC)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.692961    0.212445 -3.2618 0.0011700 **
#> FDD          0.471433    0.135195  3.4871 0.0005242 ***
#> L(FDD, 1:6)1  0.145021    0.081557  1.7782 0.0758853 .
#> L(FDD, 1:6)2  0.058364    0.058911  0.9907 0.3222318
#> L(FDD, 1:6)3  0.074166    0.047143  1.5732 0.1162007
#> L(FDD, 1:6)4  0.036304    0.029335  1.2376 0.2163670
#> L(FDD, 1:6)5  0.048756    0.031370  1.5543 0.1206535
#> L(FDD, 1:6)6  0.050246    0.045129  1.1134 0.2659919
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the result we obtain

$$\begin{aligned} \% \widehat{ChgOJC}_t = & -0.69 + 0.47 FDD_t + 0.15 FDD_{t-1} + 0.06 FDD_{t-2} + 0.07 FDD_{t-3} \\ & \quad (0.21) \quad (0.14) \quad (0.08) \quad (0.06) \quad (0.05) \\ & + 0.04 FDD_{t-4} + 0.05 FDD_{t-5} + 0.05 FDD_{t-6}, \\ & \quad (0.03) \quad (0.03) \quad (0.05) \end{aligned} \quad (15.1)$$

where the coefficient on FDD_{t-1} estimates the price increase in period t caused by an additional freezing degree day in the preceding month, the coefficient on FDD_{t-2} estimates the effect of an additional freezing degree day two month ago and so on. Consequently, the coefficients in (15.1) can be interpreted as price changes in current and future periods due to an unit increase in the current month's freezing degree days.

15.2 Dynamic Causal Effects

This section of the book describes the general idea of a dynamic causal effect and how the concept of a randomized controlled experiment can be translated to time series applications, using several examples.

In general, for empirical attempts to measure a dynamic causal effect, the assumptions of stationarity (see Key Concept 14.5) and exogeneity must hold. In time series applications up until here we have assumed that the model error term has conditional mean zero given current and past values of the regressors. For estimation of a dynamic causal effect using a distributed lag model, assuming a stronger form termed *strict exogeneity* may be useful. Strict exogeneity states that the error term has mean zero conditional on past, present *and future* values of the independent variables.

The two concepts of exogeneity and the distributed lag model are summarized in Key Concept 15.1.

Key Concept 15.1 The Distributed Lag Model and Exogeneity

The general distributed lag model is

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \beta_3 X_{t-2} + \cdots + \beta_{r+1} X_{t-r} + u_t, \quad (15.2)$$

where it is assumed that

1. X is an exogenous variable,

$$E(u_t | X_t, X_{t-1}, X_{t-2}, \dots) = 0.$$

2. (a) X_t, Y_t have a stationary distribution.
(b) (Y_t, X_t) and (Y_{t-j}, X_{t-j}) become independently distributed as j gets large.
3. Large outliers are unlikely. In particular, we need that all the variables have more than eight finite moments — a stronger assumption than before (four finite moments) that is required for computation of the HAC covariance matrix estimator.
4. There is no perfect multicollinearity.

The distributed lag model may be extended to include contemporaneous and past values of additional regressors.

On the assumption of exogeneity

- There is another form of exogeneity termed *strict exogeneity* which assumes

$$E(u_t | \dots, X_{t+2}, X_{t+1}, X_t, X_{t-1}, X_{t-2}, \dots) = 0,$$

that is the error term has mean zero conditional on past, present and future values of X . Strict exogeneity implies exogeneity (as defined in 1. above) but not the other way around. From this point on we will therefore distinguish between exogeneity and strict exogeneity.

- Exogeneity as in 1. suffices for the OLS estimators of the coefficient in distributed lag models to be consistent. However, if the the assumption of strict exogeneity is valid, more efficient estimators can be applied.

15.3 Dynamic Multipliers and Cumulative Dynamic Multipliers

The following terminology regarding the coefficients in the distributed lag model (15.2) is useful for upcoming applications:

- The dynamic causal effect is also called the *dynamic multiplier*. β_{h+1} in (15.2) is the h -period dynamic multiplier.
- The contemporaneous effect of X on Y , β_1 , is termed the *impact effect*.
- The h -period *cumulative dynamic multiplier* of a unit change in X and Y is defined as the

cumulative sum of the dynamic multipliers. In particular, β_1 is the zero-period cumulative dynamic multiplier, $\beta_1 + \beta_2$ is the one-period cumulative dynamic multiplier and so forth.

The cumulative dynamic multipliers of the distributed lag model (15.2) are the coefficients $\delta_1, \delta_2, \dots, \delta_r, \delta_{r+1}$ in the modified regression

$$Y_t = \delta_0 + \delta_1 \Delta X_t + \delta_2 \Delta X_{t-1} + \dots + \delta_r \Delta X_{t-r+1} + \delta_{r+1} X_{t-r} + u_t, \quad (15.3)$$

and thus can be directly estimated using OLS which makes it convenient to compute their HAC standard errors. δ_{r+1} is called the *long-run cumulative dynamic multiplier*.

It is straightforward to compute the cumulative dynamic multipliers for (15.1), the estimated distributed lag regression of changes in orange juice concentrate prices on freezing degree days, using the corresponding model object `orange_DLM` and the function `cumsum()`.

```
# compute cumulative multipliers
cum_mult <- cumsum(orange_DLM$coefficients[-1])

# rename entries
names(cum_mult) <- paste(0:6, sep = "-", "period CDM")

cum_mult
#> 0-period CDM 1-period CDM 2-period CDM 3-period CDM 4-period CDM 5-period CDM
#> 0.4714329 0.6164542 0.6748177 0.7489835 0.7852874 0.8340436
#> 6-period CDM
#> 0.8842895
```

Translating the distributed lag model with six lags of *FDD* to (15.3), we see that the OLS coefficient estimates in this model coincide with the multipliers stored in `cum_mult`.

```
# estimate cumulative dynamic multipliers using the modified regression
cum_mult_reg <- dynlm(FOJC_pctc ~ d(FDD) + d(L(FDD,1:5)) + L(FDD,6))
coef(cum_mult_reg)[-1]
#> d(FDD) d(L(FDD, 1:5))1 d(L(FDD, 1:5))2 d(L(FDD, 1:5))3 d(L(FDD, 1:5))4
#> 0.4714329 0.6164542 0.6748177 0.7489835 0.7852874
#> d(L(FDD, 1:5))5 L(FDD, 6)
#> 0.8340436 0.8842895
```

As noted above, using a model specification as in (15.3) allows to easily obtain standard errors for the estimated dynamic cumulative multipliers.

```
# obtain coefficient summary that reports HAC standard errors
coeftest(cum_mult_reg, vcov. = vcovHAC)
#>
#> t test of coefficients:
#>
#> Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.69296 0.23668 -2.9278 0.0035431 **
#> d(FDD) 0.47143 0.13583 3.4709 0.0005562 ***
#> d(L(FDD, 1:5))1 0.61645 0.13145 4.6896 3.395e-06 ***
#> d(L(FDD, 1:5))2 0.67482 0.16009 4.2151 2.882e-05 ***
#> d(L(FDD, 1:5))3 0.74898 0.17263 4.3387 1.682e-05 ***
#> d(L(FDD, 1:5))4 0.78529 0.17351 4.5260 7.255e-06 ***
#> d(L(FDD, 1:5))5 0.83404 0.18236 4.5737 5.827e-06 ***
#> L(FDD, 6) 0.88429 0.19303 4.5810 5.634e-06 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

15.4 HAC Standard Errors

The error term u_t in the distributed lag model (15.2) may be serially correlated due to serially correlated determinants of Y_t that are not included as regressors. When these factors are not correlated with the regressors included in the model, serially correlated errors do not violate the assumption of exogeneity such that the OLS estimator remains unbiased and consistent.

However, autocorrelated standard errors render the usual homoskedasticity-only *and* heteroskedasticity-robust standard errors invalid and may cause misleading inference. HAC errors are a remedy.

Key Concept 15.2 HAC Standard errors

Problem:

If the error term u_t in the distributed lag model (15.2) is serially correlated, statistical inference that rests on usual (heteroskedasticity-robust) standard errors can be strongly misleading.

Solution:

Heteroskedasticity- and autocorrelation-consistent (HAC) estimators of the variance-covariance matrix circumvent this issue. There are R functions like `vcovHAC()` from the package `sandwich` which are convenient for computation of such estimators.

The package `sandwich` also contains the function `NeweyWest()`, an implementation of the HAC variance-covariance estimator proposed by Newey and West (1987).

Consider the distributed lag regression model with no lags and a single regressor X_t

$$Y_t = \beta_0 + \beta_1 X_t + u_t,$$

with autocorrelated errors. A brief derivation of

$$\tilde{\sigma}_{\hat{\beta}_1}^2 = \hat{\sigma}_{\hat{\beta}_1}^2 \hat{f}_t, \quad (15.4)$$

the so-called *Newey-West variance estimator* for the variance of the OLS estimator of β_1 is presented in Chapter 15.4 of the book. $\hat{\sigma}_{\hat{\beta}_1}^2$ in (15.4) is the heteroskedasticity-robust variance estimate of $\hat{\beta}_1$ and

$$\hat{f}_t = 1 + 2 \sum_{j=1}^{m-1} \left(\frac{m-j}{m} \right) \tilde{\rho}_j \quad (15.5)$$

is a correction factor that adjusts for serially correlated errors and involves estimates of $m - 1$ autocorrelation coefficients $\tilde{\rho}_j$. As it turns out, using the sample autocorrelation as implemented in `acf()` to estimate the autocorrelation coefficients renders (15.4) inconsistent, see pp. 650-651 of the book for a detailed argument. Therefore, we use a somewhat different estimator. For a time series X we have

$$\tilde{\rho}_j = \frac{\sum_{t=j+1}^T \hat{v}_t \hat{v}_{t-j}}{\sum_{t=1}^T \hat{v}_t^2}, \text{ with } \hat{v} = (X_t - \bar{X}) \hat{u}_t.$$

We implement this estimator in the function `acf_c()` below.

m in (15.5) is a truncation parameter to be chosen. A rule of thumb for choosing m is

$$m = \left\lceil 0.75 \cdot T^{1/3} \right\rceil. \quad (15.6)$$

We simulate a time series that, as stated above, follows a distributed lag model with autocorrelated errors and then show how to compute the Newey-West HAC estimate of $SE(\hat{\beta}_1)$ using R. This is done via two separate but, as we will see, identical approaches: at first we follow the derivation presented in the book step-by-step and compute the estimate “manually”. We then show that the result is exactly the estimate obtained when using the function `NeweyWest()`.

```
# function that computes rho tilde
acf_c <- function(x, j) {
  return(
    t(x[-c(1:j)]) %*% na.omit(Lag(x, j)) / t(x) %*% x
  )
}

# simulate time series with serially correlated errors
set.seed(1)

N <- 100

eps <- arima.sim(n = N, model = list(ma = 0.5))
X <- runif(N, 1, 10)
Y <- 0.5 * X + eps

# compute OLS residuals
res <- lm(Y ~ X)$res

# compute v
v <- (X - mean(X)) * res

# compute robust estimate of beta_1 variance
var_beta_hat <- 1/N * (1/(N-2) * sum((X - mean(X))^2 * res^2) ) /
  (1/N * sum((X - mean(X))^2))^2

# rule of thumb truncation parameter
m <- floor(0.75 * N^(1/3))

# compute correction factor
f_hat_T <- 1 + 2 * sum(
  (m - 1:(m-1))/m * sapply(1:(m - 1), function(i) acf_c(x = v, j = i))
)

# compute Newey-West HAC estimate of the standard error
sqrt(var_beta_hat * f_hat_T)
#> [1] 0.04036208
```

For the code to be reusable in other applications, we use `sapply()` to estimate the $m-1$ autocorrelations $\tilde{\rho}_j$.


```
# Using NeweyWest():
NW_VCOV <- NeweyWest(lm(Y ~ X),
                     lag = m - 1, prewhite = F,
                     adjust = T)

# compute standard error
sqrt(diag(NW_VCOV))[2]
#>      X
#> 0.04036208
```

By choosing `lag = m-1` we ensure that the maximum order of autocorrelations used is $m - 1$ — just as in equation (15.5). Notice that we set the arguments `prewhite = F` and `adjust = T` to ensure that the formula (15.4) is used and finite sample adjustments are made.

We find that the computed standard errors coincide. Of course, a variance-covariance matrix estimate as computed by `NeweyWest()` can be supplied as the argument `vcov` in `coeftest()` such that HAC t -statistics and p -values are provided by the latter.

```
example_mod <- lm(Y ~ X)
coeftest(example_mod, vcov = NW_VCOV)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 0.542310  0.235423  2.3036 0.02336 *
#> X           0.423305  0.040362 10.4877 < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

15.5 Estimation of Dynamic Causal Effects with Strictly Exogenous Regressors

In general, the errors in a distributed lag model are correlated which necessitates usage of HAC standard errors for valid inference. If, however, the assumption of exogeneity (the first assumption stated in Key Concept 15.1) is replaced by strict exogeneity, that is,

$$E(u_t | \dots, X_{t+1}, X_t, X_{t-1}, \dots) = 0,$$

more efficient approaches than OLS estimation of the coefficients become available. For a general distributed lag model with r lags and $AR(p)$ errors, these approaches are summarized in Key Concept 15.4.

Key Concept 15.4**Estimation of Dynamic Multipliers Under Strict Exogeneity**

Consider the general distributed lag model with r lags and errors following an AR(p) process,

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \cdots + \beta_{r+1} X_{t-r} + u_t \quad (15.7)$$

$$u_t = \phi_1 u_{t-1} + \phi_2 u_{t-2} + \cdots + \phi_p u_{t-p} + \tilde{u}_t. \quad (15.8)$$

Under strict exogeneity of X_t , one may rewrite the above model in the ADL specification

$$Y_t = \alpha_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \\ + \delta_0 X_t + \delta_1 X_{t-1} + \cdots + \delta_q X_{t-q} + \tilde{u}_t$$

where $q = r + p$ and compute estimates of the dynamic multipliers $\beta_1, \beta_2, \dots, \beta_{r+1}$ using OLS estimates of $\phi_1, \phi_2, \dots, \phi_p, \delta_0, \delta_1, \dots, \delta_q$.

An alternative is to estimate the dynamic multipliers using feasible GLS, that is to apply the OLS estimator to a quasi-difference specification of (??). Under strict exogeneity, the feasible GLS approach is the BLUE estimator for the dynamic multipliers in large samples.

On the one hand, as demonstrated in Chapter 15.5 of the book, OLS estimation of the ADL representation can be beneficial for estimation of the dynamic multipliers in large distributed lag models because it allows for a more parsimonious model that may be a good approximation to the large model. On the other hand, the GLS approach is more efficient than the ADL estimator if the sample size is large.

We shortly review how the different representations of a small distributed lag model can be obtained and show how this specification can be estimated by OLS and GLS using R.

The model is

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + u_t, \quad (15.9)$$

so a change in X is modeled to effect Y contemporaneously (β_1) and in the next period (β_2). The error term u_t is assumed to follow an AR(1) process,

$$u_t = \phi_1 u_{t-1} + \tilde{u}_t,$$

where \tilde{u}_t is serially uncorrelated.

One can show that the ADL representation of this model is

$$Y_t = \alpha_0 + \phi_1 Y_{t-1} + \delta_0 X_t + \delta_1 X_{t-1} + \delta_2 X_{t-2} + \tilde{u}_t, \quad (15.10)$$

with the restrictions

$$\beta_1 = \delta_0, \\ \beta_2 = \delta_1 + \phi_1 \delta_0,$$

see p. 657 of the book.

Quasi-Differences

Another way of writing the ADL(1,2) representation (15.10) is the *quasi-difference model*

$$\tilde{Y}_t = \alpha_0 + \beta_1 \tilde{X}_t + \beta_2 \tilde{X}_{t-1} + \tilde{u}_t, \quad (15.11)$$

where $\tilde{Y}_t = Y_t - \phi_1 Y_{t-1}$ and $\tilde{X}_t = X_t - \phi_1 X_{t-1}$. Notice that the error term \tilde{u}_t is uncorrelated in both models and, as shown in Chapter 15.5 of the book,

$$E(u_t | X_{t+1}, X_t, X_{t-1}, \dots) = 0,$$

which is implied by the assumption of strict exogeneity.

We continue by simulating a time series of 500 observations using the model (15.9) with $\beta_1 = 0.1$, $\beta_2 = 0.25$, $\phi = 0.5$ and $\tilde{u}_t \sim \mathcal{N}(0, 1)$ and estimate the different representations, starting with the distributed lag model (15.9).

```
# set seed for reproducibility
set.seed(1)

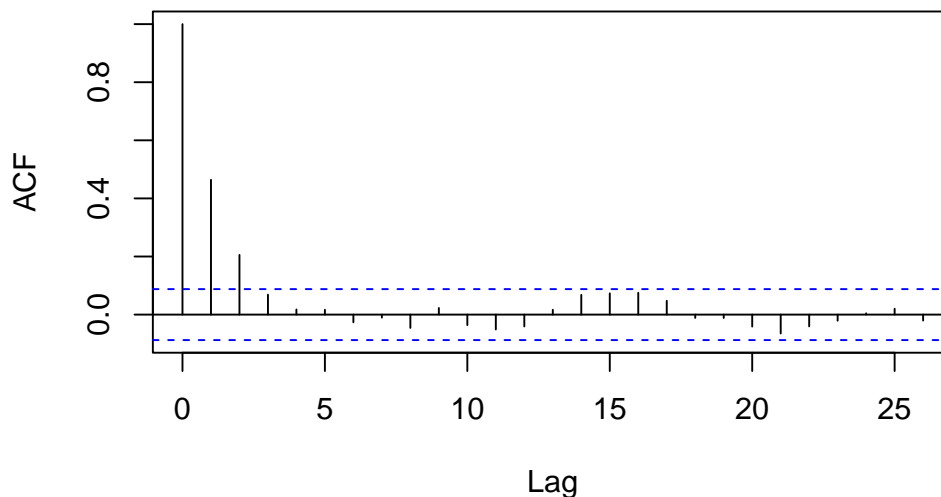
# simulate a time series with serially correlated errors
obs <- 501
eps <- arima.sim(n = obs-1 , model = list(ar = 0.5))
X <- arima.sim(n = obs, model = list(ar = 0.25))
Y <- 0.1 * X[-1] + 0.25 * X[-obs] + eps
X <- ts(X[-1])

# estimate the distributed lag model
dlm <- dynlm(Y ~ X + L(X))
```

Let us check that the residuals of this model exhibit autocorrelation using `acf()`.

```
# check that the residuals are serially correlated
acf(residuals(dlm))
```

Series residuals(dlm)



In particular, the pattern reveals that the residuals follow an autoregressive process, as the sample autocorrelation function decays quickly for the first few lags and is probably zero for higher lag orders. In any case, HAC standard errors should be used.

```
# coefficient summary using the Newey-West SE estimates
coeftest(dlm, vcov = NeweyWest, prewhite = F, adjust = T)
#>
```

```
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 0.038340  0.073411  0.5223  0.601717
#> X           0.123661  0.046710  2.6474  0.008368 **
#> L(X)        0.247406  0.046377  5.3347  1.458e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

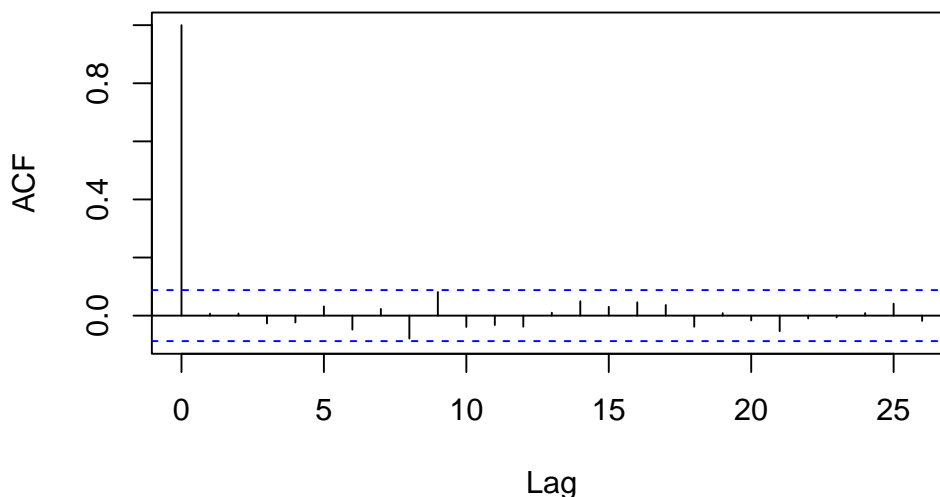
OLS Estimation of the ADL Model

Next, we estimate the ADL(1,2) model (15.10) using OLS. The errors are uncorrelated in this representation of the model. This statement is supported by a plot of the sample autocorrelation function of the residual series.

```
# estimate the ADL(2,1) representation of the distributed lag model
adl21_dynamic <- dynlm(Y ~ L(Y) + X + L(X, 1:2))

# plot the sample autocorrelations of residuals
acf(adl21_dynamic$residuals)
```

Series adl21_dynamic\$residuals



The estimated coefficients of `adl21_dynamic$coefficients` are *not* the dynamic multipliers we are interested in, but instead can be computed according to the restrictions in (15.10), where the true coefficients are replaced by the OLS estimates.

```
# compute estimated dynamic effects using coefficient restrictions
# in the ADL(2,1) representation
t <- adl21_dynamic$coefficients

c("hat_beta_1" = t[3],
  "hat_beta_2" = t[4] + t[3] * t[2])
#>           hat_beta_1.X hat_beta_2.L(X, 1:2)1
#>           0.1176425           0.2478484
```

GLS Estimation

Strict exogeneity allows for OLS estimation of the quasi-difference model (15.11). The idea of applying the OLS estimator to a model where the variables are linearly transformed, such that the model errors are uncorrelated and homoskedastic, is called *generalized least squares* (GLS).

The OLS estimator in (15.11) is called the *infeasible GLS* estimator because \tilde{Y} and \tilde{X} cannot be computed without knowing ϕ_1 , the autoregressive coefficient in the error AR(1) model, which is generally unknown in practice.

Assume we knew that $\phi = 0.5$. We then may obtain the infeasible GLS estimates of the dynamic multipliers in (15.9) by applying OLS to the transformed data.

```
# GLS: estimate quasi-differenced specification by OLS
iGLS_dynamic <- dynlm(I(Y- 0.5 * L(Y)) ~ I(X - 0.5 * L(X)) + I(L(X) - 0.5 * L(X, 2)))

summary(iGLS_dynamic)
#>
#> Time series regression with "ts" data:
#> Start = 3, End = 500
#>
#> Call:
#> dynlm(formula = I(Y - 0.5 * L(Y)) ~ I(X - 0.5 * L(X)) + I(L(X) -
#> 0.5 * L(X, 2)))
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -3.0325 -0.6375 -0.0499  0.6658  3.7724
#>
#> Coefficients:
#>
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      0.01620    0.04564   0.355  0.72273
#> I(X - 0.5 * L(X))  0.12000    0.04237   2.832  0.00481 **
#> I(L(X) - 0.5 * L(X, 2)) 0.25266    0.04237   5.963 4.72e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.017 on 495 degrees of freedom
#> Multiple R-squared:  0.07035,    Adjusted R-squared:  0.0666
#> F-statistic: 18.73 on 2 and 495 DF,  p-value: 1.442e-08
```

The *feasible GLS* estimator uses preliminary estimation of the coefficients in the presumed error term model, computes the quasi-differenced data and then estimates the model using OLS. This idea was introduced by Cochrane and Orcutt (1949) and can be extended by continuing this process iteratively. Such a procedure is implemented in the function `cochrane.orcutt()` from the package `orcutt`.

```
X_t <- c(X[-1])
# create first lag
X_l1 <- c(X[-500])
Y_t <- c(Y[-1])

# iterated cochrane-orcutt procedure
summary(cochrane.orcutt(lm(Y_t ~ X_t + X_l1)))
#> Call:
```

```

#> lm(formula = Y_t ~ X_t + X_l1)
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 0.032885  0.085163  0.386  0.69956
#> X_t         0.120128  0.042534  2.824  0.00493 **
#> X_l1        0.252406  0.042538  5.934 5.572e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.0165 on 495 degrees of freedom
#> Multiple R-squared:  0.0704 , Adjusted R-squared:  0.0666
#> F-statistic: 18.7 on 2 and 495 DF, p-value: < 1.429e-08
#>
#> Durbin-Watson statistic
#> (original):  1.06907 , p-value: 1.05e-25
#> (transformed): 1.98192 , p-value: 4.246e-01

```

Some more sophisticated methods for GLS estimation are provided with the package `nlme`. The function `gls()` can be used to fit linear models by maximum likelihood estimation algorithms and allows to specify a correlation structure for the error term.

```

# feasible GLS maximum likelihood estimation procedure
summary(gls(Y_t ~ X_t + X_l1, correlation = corAR1()))
#> Generalized least squares fit by REML
#> Model: Y_t ~ X_t + X_l1
#> Data: NULL
#>      AIC      BIC    logLik
#> 1451.847 1472.88 -720.9235
#>
#> Correlation Structure: AR(1)
#> Formula: ~1
#> Parameter estimate(s):
#>      Phi
#> 0.4668343
#>
#> Coefficients:
#>           Value Std.Error t-value p-value
#> (Intercept) 0.03929124 0.08530544 0.460595 0.6453
#> X_t         0.11986994 0.04252270 2.818963 0.0050
#> X_l1        0.25287471 0.04252497 5.946500 0.0000
#>
#> Correlation:
#>      (Intr) X_t
#> X_t  0.039
#> X_l1 0.037 0.230
#>
#> Standardized residuals:
#>      Min      Q1      Med      Q3      Max
#> -3.00075518 -0.64255522 -0.05400347 0.69101814 3.28555793
#>
#> Residual standard error: 1.14952
#> Degrees of freedom: 499 total; 496 residual

```

Notice that in this example, the coefficient estimates produced by GLS are somewhat closer to their true values and that the standard errors are the smallest for the GLS estimator.

15.6 Orange Juice Prices and Cold Weather

This section investigates the following two questions using the time series regression methods discussed here:

- How persistent is the effect of a single freeze on orange juice concentrate prices?
- Has the effect been stable over the whole time span?

We start by estimating dynamic causal effects with a distributed lag model where $\%ChgOJC_t$ is regressed on FDD_t and 18 lags. A second model specification considers a transformation of the distributed lag model which allows to estimate the 19 cumulative dynamic multipliers using OLS. The third model, adds 11 binary variables (one for each of the months from February to December) to adjust for a possible omitted variable bias arising from correlation of FDD_t and seasons by adding `season(FDD)` to the right hand side of the formula of the second model.

```
# estimate distributed lag models of frozen orange juice price changes
FOJC_mod_DM <- dynlm(FOJC_pctc ~ L(FDD, 0:18))
FOJC_mod_CM1 <- dynlm(FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18))
FOJC_mod_CM2 <- dynlm(FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD))
```

The above models include a large number of lags with default labels that correspond to the degree of differencing and the lag orders which makes it somewhat cumbersome to read the output. The regressor labels of a model object may be altered by overriding the attribute `names` of the coefficient section using the function `attr()`. Thus, for better readability we use the lag orders as regressor labels.

```
# set lag orders as regressor labels
attr(FOJC_mod_DM$coefficients, "names")[1:20] <- c("(Intercept)", as.character(0:18))
attr(FOJC_mod_CM1$coefficients, "names")[1:20] <- c("(Intercept)", as.character(0:18))
attr(FOJC_mod_CM2$coefficients, "names")[1:20] <- c("(Intercept)", as.character(0:18))
```

Next, we compute HAC standard errors for each model using `NeweyWest()` and gather the results in a list which is then supplied as the argument `se` to the function `stargazer()`, see below. The sample consists of 612 observations:

```
length(FDD)
#> [1] 612
```

According to (15.6), the rule of thumb for choosing the HAC standard error truncation parameter m , we choose

$$m = \left\lceil 0.75 \cdot 612^{1/3} \right\rceil = \lceil 6.37 \rceil = 7.$$

To check for sensitivity of the standard errors to different choices of the truncation parameter in the model that is used to estimate the cumulative multipliers, we also compute the Newey-West estimator for $m = 14$.

```
# gather HAC standard error errors in a list
SEs <- list(sqrt(diag(NeweyWest(FOJC_mod_DM, lag = 7, prewhite = F))),
            sqrt(diag(NeweyWest(FOJC_mod_CM1, lag = 7, prewhite = F))),
            sqrt(diag(NeweyWest(FOJC_mod_CM1, lag = 14, prewhite = F))),
            sqrt(diag(NeweyWest(FOJC_mod_CM2, lag = 7, prewhite = F))))
```

The results are then used to reproduce the outcomes presented in Table 15.1 of the book.

```

stargazer(FOJC_mod_DM , FOJC_mod_CM1, FOJC_mod_CM1, FOJC_mod_CM2,
  title = "Dynamic Effects of a Freezing Degree Day on the Price of Orange Juice",
  header = FALSE,
  digits = 3,
  column.labels = c("Dynamic Multipliers", rep("Dynamic Cumulative Multipliers", 3)),
  dep.var.caption = "Dependent Variable: Monthly Percentage Change in Orange Juice Price",
  dep.var.labels.include = FALSE,
  covariate.labels = as.character(0:18),
  omit = "season",
  se = SEs,
  no.space = T,
  add.lines = list(c("Monthly indicators?", "no", "no", "no", "yes"),
    c("HAC truncation", "7", "7", "14", "7")),
  omit.stat = c("rsq", "f", "ser"))

```

According to column (1) of Table 15.1, the contemporaneous effect of a freezing degree day is an increase of 0.5% in orange juice prices. The estimated effect is only 0.17% for the next month and close to zero for subsequent months. In fact, for all lags larger than 1, we cannot reject the null hypotheses that the respective coefficients are zero using individual t -tests. The model FOJC_mod_DM only explains little of the variation in the dependent variable ($\bar{R}^2 = 0.11$).

Columns (2) and (3) present estimates of the dynamic cumulative multipliers of model FOJC_mod_CM1. Apparently, it does not matter whether we choose $m = 7$ or $m = 14$ when computing HAC standard errors so we stick with $m = 7$ and the standard errors reported in column (2).

If the demand for orange juice is higher in winter, FDD_t would be correlated with the error term since freezing occur rather in winter so we would face omitted variable bias. The fourth model estimate, FOJC_mod_CM2, accounts for this possible issue by using an additional set of 11 monthly dummies. For brevity, estimates of the dummy coefficients are excluded from the output produced by stargazer (this is achieved by setting `omit = 'season'`). We may check that the dummy for January was omitted to prevent perfect multicollinearity.

```

# estimates on mothly dummies
FOJC_mod_CM2$coefficients[-c(1:20)]
#> season(FDD)Feb season(FDD)Mar season(FDD)Apr season(FDD)May season(FDD)Jun
#>   -0.9565759   -0.6358007    0.5006770   -1.0801764    0.3195624
#> season(FDD)Jul season(FDD)Aug season(FDD)Sep season(FDD)Oct season(FDD)Nov
#>    0.1951113    0.3644312   -0.4130969   -0.1566622    0.3116534
#> season(FDD)Dec
#>    0.1481589

```

A comparison of the estimates presented in columns (2) and (4) indicates that adding monthly dummies has a negligible effect. Further evidence for this comes from a joint test of the hypothesis that the 11 dummy coefficients are zero. Instead of using `linearHypothesis()`, we use the function `waldtest()` and supply two model objects instead: `unres_model`, the unrestricted model object which is the same as FOJC_mod_CM2 (except for the coefficient names since we have modified them above) and `res_model`, the model where the restriction that all dummy coefficients are zero is imposed. `res_model` is conveniently obtained using the function `update()`. It extracts the argument `formula` of a model object, updates it as specified and then re-fits the model. By setting `formula = . ~ . - season(FDD)` we impose that the monthly dummies do not enter the model.

```

# test if coefficients on monthly dummies are zero
unres_model <- dynlm(FOJC_ptc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD))

```


Table 15.1: Dynamic Effects of a Freezing Degree Day on the Price of Orange Juice

	Dependent Variable: Monthly Percentage Change in Orange Juice Price			
	Dyn. Mult.	(2) Dyn. Cum. Mult.	(3) Dyn. Cum. Mult.	(4) Dyn. Cum. Mult.
lag 0	0.508*** (0.137)	0.508*** (0.137)	0.508*** (0.139)	0.524*** (0.142)
lag 1	0.172** (0.088)	0.680*** (0.134)	0.680*** (0.130)	0.720*** (0.142)
lag 2	0.068 (0.060)	0.748*** (0.165)	0.748*** (0.162)	0.781*** (0.173)
lag 3	0.070 (0.044)	0.819*** (0.181)	0.819*** (0.181)	0.861*** (0.190)
lag 4	0.022 (0.031)	0.841*** (0.183)	0.841*** (0.184)	0.892*** (0.194)
lag 5	0.027 (0.030)	0.868*** (0.189)	0.868*** (0.189)	0.904*** (0.199)
lag 6	0.031 (0.047)	0.900*** (0.202)	0.900*** (0.208)	0.922*** (0.210)
lag 7	0.015 (0.015)	0.915*** (0.205)	0.915*** (0.210)	0.939*** (0.212)
lag 8	-0.042 (0.034)	0.873*** (0.214)	0.873*** (0.218)	0.904*** (0.219)
lag 9	-0.010 (0.051)	0.862*** (0.236)	0.862*** (0.245)	0.884*** (0.239)
lag 10	-0.116* (0.069)	0.746*** (0.257)	0.746*** (0.262)	0.752*** (0.259)
lag 11	-0.067 (0.052)	0.680** (0.266)	0.680** (0.272)	0.677** (0.267)
lag 12	-0.143* (0.076)	0.537** (0.268)	0.537** (0.271)	0.551** (0.272)
lag 13	-0.083* (0.043)	0.454* (0.267)	0.454* (0.273)	0.491* (0.275)
lag 14	-0.057 (0.035)	0.397 (0.273)	0.397 (0.284)	0.427 (0.278)
lag 15	-0.032 (0.028)	0.366 (0.276)	0.366 (0.287)	0.406 (0.280)
lag 16	-0.005 (0.055)	0.360 (0.283)	0.360 (0.293)	0.408 (0.286)
lag 17	0.003 (0.018)	0.363 (0.287)	0.363 (0.294)	0.395 (0.290)
lag 18	0.003 (0.017)	0.366 (0.293)	0.366 (0.301)	0.386 (0.295)
Constant	-0.343 (0.269)	-0.343 (0.269)	-0.343 (0.256)	-0.241 (0.934)
Monthly indicators?	no	no	no	yes
HAC truncation	7	7	14	7
Observations	594	594	594	594
Adjusted R ²	0.109	0.109	0.109	0.101

Note: *p<0.1; **p<0.05; ***p<0.01

```

res_model <- update(unres_model, formula = . ~ . - season(FDD))

waldtest(unres_model,
         res_model,
         vcov = NeweyWest(unres_model, lag = 7, prewhite = F))
#> Wald test
#>
#> Model 1: FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD)
#> Model 2: FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18)
#>   Res.Df  Df      F Pr(>F)
#> 1     563
#> 2     574 -11 0.9683 0.4743

```

The p -value is 0.47 so we cannot reject the hypothesis that the coefficients on the monthly dummies are zero, even at the 10% level. We conclude that the seasonal fluctuations in demand for orange juice do not pose a serious threat to internal validity of the model.

It is convenient to use plots of dynamic multipliers and cumulative dynamic multipliers. The following two code chunks reproduce Figures 15.2 (a) and 15.2 (b) of the book which display point estimates of dynamic and cumulative multipliers along with upper and lower bounds of their 95% confidence intervals computed using the above HAC standard errors.

```

# 95% CI bounds
point_estimates <- FOJC_mod_DM$coefficients

CI_bounds <- cbind("lower" = point_estimates - 1.96 * SEs[[1]],
                  "upper" = point_estimates + 1.96 * SEs[[1]])[-1, ]

# plot the estimated dynamic multipliers
plot(0:18, point_estimates[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     ylim = c(-0.4, 1),
     xlab = "Lag",
     ylab = "Dynamic multiplier",
     main = "Dynamic Effect of FDD on Orange Juice Price")

# add a dashed line at 0
abline(h = 0, lty = 2)

# add CI bounds
lines(0:18, CI_bounds[,1], col = "darkred")
lines(0:18, CI_bounds[,2], col = "darkred")

```

The 95% confidence intervals plotted in Figure 15.1 indeed include zero for lags larger than 1 such that the null of a zero multiplier cannot be rejected for these lags.

```

# 95% CI bounds
point_estimates <- FOJC_mod_CM1$coefficients

CI_bounds <- cbind("lower" = point_estimates - 1.96 * SEs[[2]],
                  "upper" = point_estimates + 1.96 * SEs[[2]])[-1, ]

```

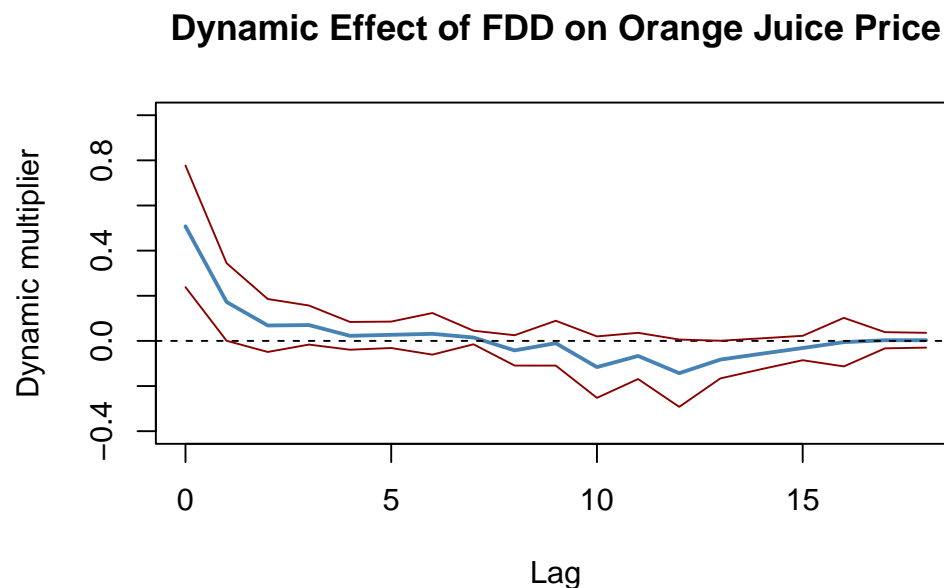


Figure 15.1: Dynamic Multipliers

```
# plot estimated dynamic multipliers
plot(0:18, point_estimates[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     ylim = c(-0.4, 1.6),
     xlab = "Lag",
     ylab = "Cumulative dynamic multiplier",
     main = "Cumulative Dynamic Effect of FDD on Orange Juice Price",
     cex.main=0.8)

# add dashed line at 0
abline(h = 0, lty = 2)

# add CI bounds
lines(0:18, CI_bounds[, 1], col = "darkred")
lines(0:18, CI_bounds[, 2], col = "darkred")
```

As can be seen from Figure 15.2, the estimated dynamic cumulative multipliers grow until the seventh month up to a price increase of about 0.91% and then decrease slightly to the estimated long-run cumulative multiplier of 0.37% which, however, is not significantly different from zero at the 5% level.

Have the dynamic multipliers been stable over time? One way to see this is to estimate these multipliers for different subperiods of the sample span. For example, consider periods 1950 - 1966, 1967 - 1983 and 1984 - 2000. If the multipliers are the same for all three periods the estimates should be close and thus the estimated cumulative multipliers should be similar, too. We investigate this by re-estimating `FOJC_mod_CM1` for the three different time spans and then plot the estimated cumulative dynamic multipliers for the comparison.

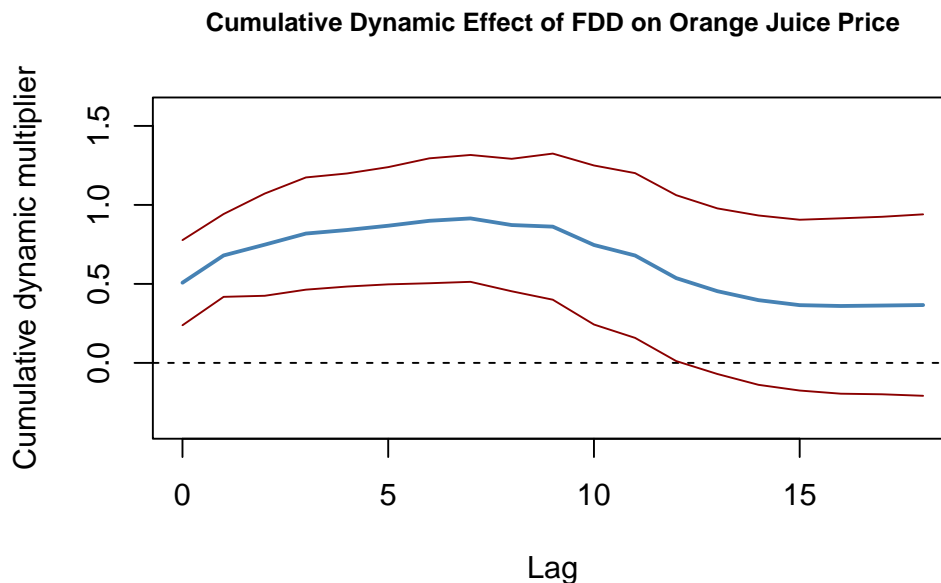


Figure 15.2: Dynamic Cumulative Multipliers

```

# estimate cumulative multipliers using different sample periods
FOJC_mod_CM1950 <- update(FOJC_mod_CM1, start = c(1950, 1), end = c(1966, 12))

FOJC_mod_CM1967 <- update(FOJC_mod_CM1, start = c(1967, 1), end = c(1983, 12))

FOJC_mod_CM1984 <- update(FOJC_mod_CM1, start = c(1984, 1), end = c(2000, 12))

# plot estimated dynamic cumulative multipliers (1950-1966)
plot(0:18, FOJC_mod_CM1950$coefficients[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     xlim = c(0, 20),
     ylim = c(-0.5, 2),
     xlab = "Lag",
     ylab = "Cumulative dynamic multiplier",
     main = "Cumulative Dynamic Effect for Different Sample Periods")

# plot estimated dynamic multipliers (1967-1983)
lines(0:18, FOJC_mod_CM1967$coefficients[-1], lwd = 2)

# plot estimated dynamic multipliers (1984-2000)
lines(0:18, FOJC_mod_CM1984$coefficients[-1], lwd = 2, col = "darkgreen")

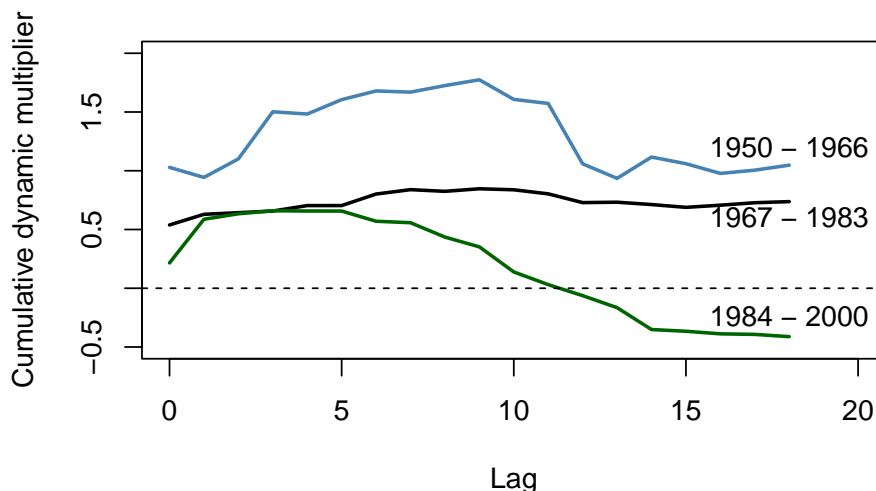
# add dashed line at 0
abline(h = 0, lty = 2)

# add annotations
text(18, -0.24, "1984 - 2000")
text(18, 0.6, "1967 - 1983")

```

```
text(18, 1.2, "1950 - 1966")
```

Cumulative Dynamic Effect for Different Sample Periods



Clearly, the cumulative dynamic multipliers have changed considerably over time. The effect of a freeze was stronger and more persistent in the 1950s and 1960s. For the 1970s the magnitude of the effect was lower but still highly persistent. We observe an even lower magnitude for the final third of the sample span (1984 - 2000) where the long-run effect is much less persistent and essentially zero after a year.

A QLR test for a break in the distributed lag regression of column (1) in Table 15.1 with 15% trimming using a HAC variance-covariance matrix estimate supports the conjecture that the population regression coefficients have changed over time.

```
# set up a range of possible break dates
tau <- c(window(time(FDD),
               time(FDD)[round(612/100*15)],
               time(FDD)[round(612/100*85)]))

# initialize the vector of F-statistics
Fstats <- numeric(length(tau))

# the restricted model
res_model <- dynlm(FOJC_pctc ~ L(FDD, 0:18))

# estimation, loop over break dates
for(i in 1:length(tau)) {

  # set up dummy variable
  D <- time(FOJC_pctc) > tau[i]

  # estimate DL model with interactions
  unres_model <- dynlm(FOJC_pctc ~ D * L(FDD, 0:18))

  # compute and save F-statistic
  Fstats[i] <- waldtest(res_model,
                       unres_model,
                       vcov = NeweyWest(unres_model, lag = 7, prewhite = F))$F[2]
```

```
}

```

Note that this code takes a couple of seconds to run since a total of `length(tau)` regressions with 40 model coefficients each are estimated.

```
# QLR test statistic
max(Fstats)
#> [1] 36.76819

```

The QLR statistic is 36.77. From Table 14.5 of the book we see that the 1% critical value for the QLR test with 15% trimming and $q = 20$ restrictions is 2.43. Since this is a right-sided test, the QLR statistic clearly lies in the region of rejection so we can discard the null hypothesis of no break in the population regression function.

See Chapter 15.7 of the book for a discussion of empirical examples where it is questionable whether the assumption of (past and present) exogeneity of the regressors is plausible.

Summary

- We have seen how R can be used to estimate the time path of the effect on Y of a change in X (the dynamic causal effect on Y of a change in X) using time series data on both. The corresponding model is called the distributed lag model. Distributed lag models are conveniently estimated using the function `dynlm()` from the package `dynlm`.
- The regression error in distributed lag models is often serially correlated such that standard errors which are robust to heteroskedasticity *and* autocorrelation should be used to obtain valid inference. The package `sandwich` provides functions for computation of so-called HAC covariance matrix estimators, for example `vcovHAC()` and `NeweyWest()`.
- When X is *strictly exogeneous*, more efficient estimates can be obtained using an ADL model or by GLS estimation. Feasible GLS algorithms can be found in the R packages `orcutt` and `nlme`. Chapter 15.7 of the book emphasizes that the assumption of strict exogeneity is often implausible in empirical applications.

Chapter 16

Additional Topics in Time Series Regression

This chapter discusses the following advanced topics in time series regression and demonstrates how core techniques can be applied using R:

- *Vector autoregressions* (VARs). We focus on using VARs for forecasting. Another branch of the literature is concerned with so-called *Structural VARs* which are, however, beyond the scope of this chapter.
- Multiperiod forecasts. This includes a discussion of iterated and direct (multivariate) forecasts.
- The DF-GLS test, a modification of the ADF test that has more power than the latter when the series has deterministic components and is close to being nonstationarity.
- Cointegration analysis with an application to short- and long-term interest rates. We demonstrate how to estimate a vector error correction model.
- Autoregressive conditional heteroskedasticity (ARCH) models. We show how a simple generalized ARCH (GARCH) model can be helpful in quantifying the risk associated with investing in the stock market in terms of estimation and forecasting of the volatility of asset returns.

To reproduce the code examples, install the R packages listed below and make sure that the subsequent code chunk executes without any errors.

- `AER` (Kleiber and Zeileis, 2008b),
- `dynlm` (Zeileis, 2019),
- `fGarch` (Wuertz et al., 2023),
- `quantmod` (Ryan and Ulrich, 2023),
- `readxl` (Wickham and Bryan, 2023),
- `scales` (Wickham and Seidel, 2022),
- `vars` (Pfaff, 2023).

```
library(AER)
library(readxl)
library(dynlm)
library(vars)
library(quantmod)
library(scales)
library(fGarch)
```

16.1 Vector Autoregressions

A Vector autoregressive (VAR) model is useful when one is interested in predicting multiple time series variables using a single model. At its core, the VAR model is an extension of the univariate autoregressive model we have dealt with in Chapters 14 and 15. Key Concept 16.1 summarizes the essentials of VAR.

Key Concept 16.1 Vector Autoregressions

The vector autoregression (VAR) model extends the idea of univariate autoregression to k time series regressions, where the lagged values of *all* k series appear as regressors. Put differently, in a VAR model we regress a *vector* of time series variables on lagged vectors of these variables. As for AR(p) models, the lag order is denoted by p so the VAR(p) model of two variables X_t and Y_t ($k = 2$) is given by the equations

$$\begin{aligned} Y_t &= \beta_{10} + \beta_{11}Y_{t-1} + \cdots + \beta_{1p}Y_{t-p} + \gamma_{11}X_{t-1} + \cdots + \gamma_{1p}X_{t-p} + u_{1t}, \\ X_t &= \beta_{20} + \beta_{21}Y_{t-1} + \cdots + \beta_{2p}Y_{t-p} + \gamma_{21}X_{t-1} + \cdots + \gamma_{2p}X_{t-p} + u_{2t}. \end{aligned}$$

The β s and γ s can be estimated using OLS on each equation. The assumptions for VARs are the time series assumptions presented in Key Concept 14.6 applied to each of the equations.

It is straightforward to estimate VAR models in R. A feasible approach is to simply use `lm()` for estimation of the individual equations. Furthermore, the R package `vars` provides standard tools for estimation, diagnostic testing and prediction using this type of models.

When the assumptions of Key Concept 16.1 hold, the OLS estimators of the VAR coefficients are consistent and jointly normal in large samples so that the usual inferential methods such as confidence intervals and t -statistics can be used.

The structure of VARs also allows to jointly test restrictions across multiple equations. For instance, it may be of interest to test whether the coefficients on all regressors of the lag p are zero. This corresponds to testing the null that the lag order $p - 1$ is correct. Large sample joint normality of the coefficient estimates is convenient because it implies that we may simply use an F -test for this testing problem. The explicit formula for such a test statistic is rather complicated but fortunately such computations are easily done using the R functions we work with in this chapter. Another way to determine optimal lag lengths are information criteria like the BIC which we have introduced for univariate time series regressions in Chapter 14.6. Just as in the case of a single equation, for a multiple equation model we choose the specification which has the smallest $BIC(p)$, where

$$BIC(p) = \log \left[\det(\widehat{\Sigma}_u) \right] + k(kp + 1) \frac{\log(T)}{T},$$

with $\widehat{\Sigma}_u$ denoting the estimate of the $k \times k$ covariance matrix of the VAR errors and $\det(\cdot)$ denotes the determinant.

As for univariate distributed lag models, one should think carefully about variables to include in a VAR, as adding unrelated variables reduces the forecast accuracy by increasing the estimation error. This is particularly important because the number of parameters to be estimated grows quadratically to the number of variables modeled by the VAR. In the application below we shall see that economic theory and empirical evidence are helpful for this decision.

A VAR Model of the Growth Rate of GDP and the Term Spread

We now show how to estimate a VAR model of the GDP growth rate, $GDPGR$, and the term spread, TS_{spread} . As following the discussion on nonstationarity of GDP growth in Chapter 14.7 (recall the possible break in the early 1980s detected by the QLR test statistic), we use data from 1981:Q1 to 2012:Q4. The two model equations are:

$$\begin{aligned} GDPGR_t &= \beta_{10} + \beta_{11}GDPGR_{t-1} + \beta_{12}GDPGR_{t-2} + \gamma_{11}TS_{spread}_{t-1} + \gamma_{12}TS_{spread}_{t-2} + u_{1t}, \\ TS_{spread}_t &= \beta_{20} + \beta_{21}GDPGR_{t-1} + \beta_{22}GDPGR_{t-2} + \gamma_{21}TS_{spread}_{t-1} + \gamma_{22}TS_{spread}_{t-2} + u_{2t}. \end{aligned}$$

The data set `us_macro_quarterly.xlsx` is provided on the companion website to Stock and Watson (2015) and can be downloaded here. It contains quarterly data on U.S. real (i.e., inflation adjusted) GDP from 1957 to 2004. We begin by importing the data set and do some formatting (we already worked with this data set in Chapter 14 so you may skip these steps if you have already loaded the data in your working environment).

```
# load the U.S. macroeconomic data set
USMacroSWQ <- read_xlsx("Data/us_macro_quarterly.xlsx",
                      sheet = 1,
                      col_types = c("text", rep("numeric", 9)))

# Change the name of the first column
colnames(USMacroSWQ)[1] <- "Date"

# format the date column
USMacroSWQ$Date <- as.yearqtr(USMacroSWQ$Date, format = "%Y:%0%q")

# define GDP as ts object
GDP <- ts(USMacroSWQ$GDPC96,
         start = c(1957, 1),
         end = c(2013, 4),
         frequency = 4)

# define GDP growth as a ts object
GDPGrowth <- ts(400*log(GDP[-1]/GDP[-length(GDP)]),
               start = c(1957, 2),
               end = c(2013, 4),
               frequency = 4)

# 3-months Treasury bill interest rate as a 'ts' object
TB3MS <- ts(USMacroSWQ$TB3MS,
           start = c(1957, 1),
           end = c(2013, 4),
           frequency = 4)

# 10-years Treasury bonds interest rate as a 'ts' object
TB10YS <- ts(USMacroSWQ$GS10,
            start = c(1957, 1),
            end = c(2013, 4),
            frequency = 4)
```

```
# generate the term spread series
TSpread <- TB10YS - TB3MS
```

We estimate both equations separately by OLS and use `coefstest()` to obtain robust standard errors.

```
# Estimate both equations using 'dynlm()'
VAR_EQ1 <- dynlm(GDPGrowth ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2),
                 start = c(1981, 1),
                 end = c(2012, 4))

VAR_EQ2 <- dynlm(TSpread ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2),
                 start = c(1981, 1),
                 end = c(2012, 4))

# rename regressors for better readability
names(VAR_EQ1$coefficients) <- c("Intercept", "Growth_t-1",
                                "Growth_t-2", "TSpread_t-1", "TSpread_t-2")
names(VAR_EQ2$coefficients) <- names(VAR_EQ1$coefficients)

# robust coefficient summaries
coefstest(VAR_EQ1, vcov. = sandwich)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> Intercept    0.516344   0.524429  0.9846 0.3267616
#> Growth_t-1   0.289553   0.110827  2.6127 0.0101038 *
#> Growth_t-2   0.216392   0.085879  2.5197 0.0130255 *
#> TSpread_t-1 -0.902549   0.358290 -2.5190 0.0130498 *
#> TSpread_t-2  1.329831   0.392660  3.3867 0.0009503 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coefstest(VAR_EQ2, vcov. = sandwich)
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> Intercept    0.4557740  0.1214227  3.7536 0.0002674 ***
#> Growth_t-1   0.0099785  0.0218424  0.4568 0.6485920
#> Growth_t-2  -0.0572451  0.0264099 -2.1676 0.0321186 *
#> TSpread_t-1  1.0582279  0.0983750 10.7571 < 2.2e-16 ***
#> TSpread_t-2 -0.2191902  0.1086198 -2.0180 0.0457712 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We end up with the following results:

$$\begin{aligned}
 GDPGR_t &= 0.52 + 0.29 GDPGR_{t-1} + 0.22 GDPGR_{t-2} - 0.90 TSpread_{t-1} + 1.33 TSpread_{t-2}, \\
 &\quad (0.46) \quad (0.11) \quad (0.09) \quad (0.36) \quad (0.39) \\
 TSpread_t &= 0.46 + 0.01 GDPGR_{t-1} - 0.06 GDPGR_{t-2} + 1.06 TSpread_{t-1} - 0.22 TSpread_{t-2}. \\
 &\quad (0.12) \quad (0.02) \quad (0.03) \quad (0.10) \quad (0.11)
 \end{aligned}$$

The function `VAR()` can be used to obtain the same coefficient estimates as presented above since it applies OLS per equation, too.

```
# set up data for estimation using `VAR()`
VAR_data <- window(ts.union(GDPGrowth, TSpread), start = c(1980, 3), end = c(2012, 4))

# estimate model coefficients using `VAR()`
VAR_est <- VAR(y = VAR_data, p = 2)
VAR_est
#>
#> VAR Estimation Results:
#> =====
#>
#> Estimated coefficients for equation GDPGrowth:
#> =====
#> Call:
#> GDPGrowth = GDPGrowth.l1 + TSpread.l1 + GDPGrowth.l2 + TSpread.l2 + const
#>
#> GDPGrowth.l1  TSpread.l1  GDPGrowth.l2  TSpread.l2      const
#> 0.2895533    -0.9025493    0.2163919    1.3298305    0.5163440
#>
#>
#> Estimated coefficients for equation TSpread:
#> =====
#> Call:
#> TSpread = GDPGrowth.l1 + TSpread.l1 + GDPGrowth.l2 + TSpread.l2 + const
#>
#> GDPGrowth.l1  TSpread.l1  GDPGrowth.l2  TSpread.l2      const
#> 0.009978489  1.058227945 -0.057245123 -0.219190243  0.455773969
```

`VAR()` returns a list of `lm` objects which can be passed to the usual functions, for example `summary()` and so it is straightforward to obtain model statistics for the individual equations.

```
# obtain the adj. R^2 from the output of 'VAR()'
summary(VAR_est$varresult$GDPGrowth)$adj.r.squared
#> [1] 0.2887223
summary(VAR_est$varresult$TSpread)$adj.r.squared
#> [1] 0.8254311
```

We may use the individual model objects to conduct Granger causality tests.

```
# Granger causality tests:

# test if term spread has no power in explaining GDP growth
linearHypothesis(VAR_EQ1,
                 hypothesis.matrix = c("TSpread_t-1", "TSpread_t-2"),
                 vcov. = sandwich)
#> Linear hypothesis test
#>
#> Hypothesis:
#> TSpread_t-1
#> TSpread_t-2
#>
#> Model 1: restricted model
```

```

#> Model 2: GDPGrowth ~ L(GDPGrowth, 1:2) + L(TSPread, 1:2)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F   Pr(>F)
#> 1     125
#> 2     123  2 5.9094 0.003544 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# test if GDP growth has no power in explaining term spread
linearHypothesis(VAR_EQ2,
                 hypothesis.matrix = c("Growth_t-1", "Growth_t-2"),
                 vcov. = sandwich)
#> Linear hypothesis test
#>
#> Hypothesis:
#> Growth_t-1
#> Growth_t-2
#>
#> Model 1: restricted model
#> Model 2: TSPread ~ L(GDPGrowth, 1:2) + L(TSPread, 1:2)
#>
#> Note: Coefficient covariance matrix supplied.
#>
#>   Res.Df Df      F   Pr(>F)
#> 1     125
#> 2     123  2 3.4777 0.03395 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Both Granger causality tests reject at the level of 5%. This is evidence in favor of the conjecture that the term spread has power in explaining GDP growth and vice versa.

Iterated Multivariate Forecasts using an Iterated VAR

The idea of an iterated forecast for period $T + 2$ based on observations up to period T is to use the one-period-ahead forecast as an intermediate step. That is, the forecast for period $T + 1$ is used as an observation when predicting the level of a series for period $T + 2$. This can be generalized to a h -period-ahead forecast where all intervening periods between T and $T + h$ must be forecasted as they are used as observations in the process (see Chapter 16.2 of the book for a more detailed argument on this concept). Iterated multiperiod forecasts are summarized in Key Concept 16.2.

Key Concept 16.2 Iterated Multiperiod Forecasts

The steps for an *iterated multiperiod AR forecast* are:

1. Estimate the $AR(p)$ model using OLS and compute the one-period-ahead forecast.
2. Use the one-period-ahead forecast to obtain the two-period-ahead forecast.
3. Continue by iterating to obtain forecasts farther into the future.

An *iterated multiperiod VAR forecast* is done as follows:

1. Estimate the $VAR(p)$ model using OLS per equation and compute the one-period-ahead forecast for *all* variables in the VAR.
2. Use the one-period-ahead forecasts to obtain the two-period-ahead forecasts.
3. Continue by iterating to obtain forecasts of all variables in the VAR farther into the future.

Since a VAR models all variables using lags of the respective other variables, we need to compute forecasts for *all* variables. It can be cumbersome to do so when the VAR is large but fortunately there are R functions that facilitate this. For example, the function `predict()` can be used to obtain iterated multivariate forecasts for VAR models estimated by the function `VAR()`.

The following code chunk shows how to compute iterated forecasts for GDP growth and the term spread up to period 2015:Q1, that is $h = 10$, using the model object `VAR_est`.

```
# compute iterated forecasts for GDP growth and term spread for the next 10 quarters
forecasts <- predict(VAR_est)
forecasts
#> $GDPGrowth
#>      fcst      lower      upper      CI
#> [1,] 1.738653 -3.006124 6.483430 4.744777
#> [2,] 1.692193 -3.312731 6.697118 5.004925
#> [3,] 1.911852 -3.282880 7.106583 5.194731
#> [4,] 2.137070 -3.164247 7.438386 5.301317
#> [5,] 2.329667 -3.041435 7.700769 5.371102
#> [6,] 2.496815 -2.931819 7.925449 5.428634
#> [7,] 2.631849 -2.846390 8.110088 5.478239
#> [8,] 2.734819 -2.785426 8.255064 5.520245
#> [9,] 2.808291 -2.745597 8.362180 5.553889
#> [10,] 2.856169 -2.722905 8.435243 5.579074
#>
#> $TSpread
#>      fcst      lower      upper      CI
#> [1,] 1.676746 0.708471226 2.645021 0.9682751
#> [2,] 1.884098 0.471880228 3.296316 1.4122179
#> [3,] 1.999409 0.336348101 3.662470 1.6630609
#> [4,] 2.080836 0.242407507 3.919265 1.8384285
#> [5,] 2.131402 0.175797245 4.087008 1.9556052
#> [6,] 2.156094 0.125220562 4.186968 2.0308738
#> [7,] 2.161783 0.085037834 4.238528 2.0767452
#> [8,] 2.154170 0.051061544 4.257278 2.1031082
#> [9,] 2.138164 0.020749780 4.255578 2.1174139
#> [10,] 2.117733 -0.007139213 4.242605 2.1248722
```

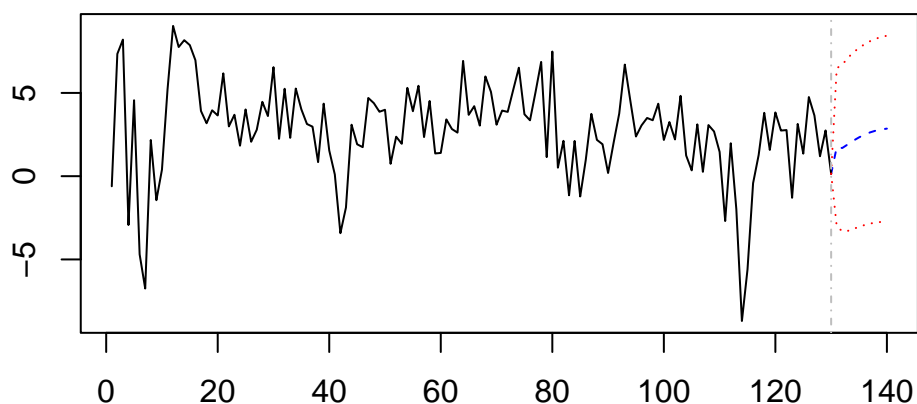
This reveals that the two-quarter-ahead forecast of GDP growth in 2013:Q2 using data through 2012:Q4 is 1.69. For the same period, the iterated VAR forecast for the term spread is 1.88.

The matrices returned by `predict(VAR_est)` also include 95% prediction intervals (however, the function does not adjust for autocorrelation or heteroskedasticity of the errors!).

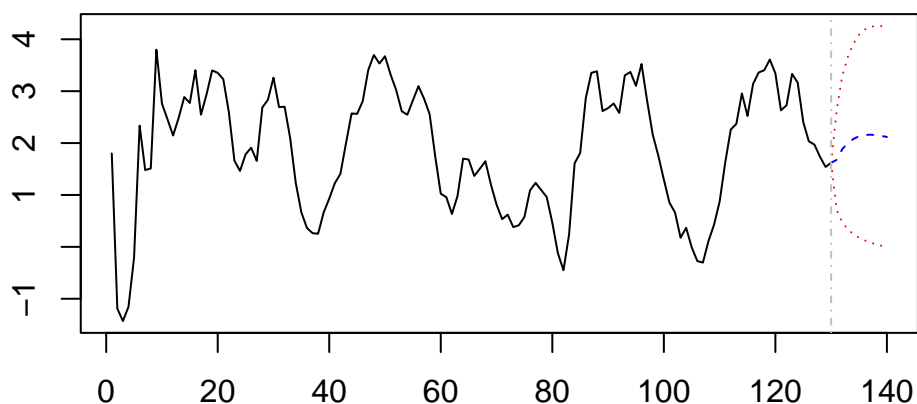
We may also plot the iterated forecasts for both variables by calling `plot()` on the output of `predict(VAR_est)`.

```
# visualize the iterated forecasts
plot(forecasts)
```

Forecast of series GDPGrowth



Forecast of series TSpread



Direct Multiperiod Forecasts

A direct multiperiod forecast uses a model where the predictors are lagged appropriately such that the available observations can be used *directly* to do the forecast. The idea of direct multiperiod forecasting is summarized in Key Concept 16.3.

Key Concept 16.3 Direct Multiperiod Forecasts

A *direct multiperiod forecast* that forecasts h periods into the future using a model of Y_t and an additional predictor X_t with p lags is done by first estimating

$$Y_t = \delta_0 + \delta_1 Y_{t-h} + \cdots + \delta_p Y_{t-p-h+1} + \delta_{p+1} X_{t-h} + \cdots + \delta_{2p} Y_{t-p-h+1} + u_t,$$

which is then used to compute the forecast of Y_{T+h} based on observations through period T .

For example, to obtain two-quarter-ahead forecasts of GDP growth and the term spread we first estimate the equations

$$\begin{aligned} GDPGR_t &= \beta_{10} + \beta_{11} GDPGR_{t-2} + \beta_{12} GDPGR_{t-3} + \gamma_{11} TSpread_{t-2} + \gamma_{12} TSpread_{t-3} + u_{1t}, \\ TSpread_t &= \beta_{20} + \beta_{21} GDPGR_{t-2} + \beta_{22} GDPGR_{t-3} + \gamma_{21} TSpread_{t-2} + \gamma_{22} TSpread_{t-3} + u_{2t} \end{aligned}$$

and then substitute the values of $GDPGR_{2012:Q4}$, $GDPGR_{2012:Q3}$, $TSpread_{2012:Q4}$ and $TSpread_{2012:Q3}$ into both equations. This is easily done manually.

```
# estimate models for direct two-quarter-ahead forecasts
VAR_EQ1_direct <- dynlm(GDPGrowth ~ L(GDPGrowth, 2:3) + L(TSpread, 2:3),
                        start = c(1981, 1), end = c(2012, 4))

VAR_EQ2_direct <- dynlm(TSpread ~ L(GDPGrowth, 2:3) + L(TSpread, 2:3),
                        start = c(1981, 1), end = c(2012, 4))

# compute direct two-quarter-ahead forecasts
coef(VAR_EQ1_direct) %*% c(1, # intercept
                          window(GDPGrowth, start = c(2012, 3), end = c(2012, 4)),
                          window(TSpread, start = c(2012, 3), end = c(2012, 4)))

#>           [,1]
#> [1,] 2.439497

coef(VAR_EQ2_direct) %*% c(1, # intercept
                          window(GDPGrowth, start = c(2012, 3), end = c(2012, 4)),
                          window(TSpread, start = c(2012, 3), end = c(2012, 4)))

#>           [,1]
#> [1,] 1.66578
```

Applied economists often use the iterated method since this forecasts are more reliable in terms of *MSFE*, provided that the one-period-ahead model is correctly specified. If this is not the case, for example because one equation in a VAR is believed to be misspecified, it can be beneficial to use direct forecasts since the iterated method will then be biased and thus have a higher *MSFE* than the direct method. See Chapter 16.2 for a more detailed discussion on advantages and disadvantages of both methods.

16.2 Orders of Integration and the DF-GLS Unit Root Test

Some economic time series have smoother trends than variables that can be described by random walk models. A way to model these time series is

$$\Delta Y_t = \beta_0 + \Delta Y_{t-1} + u_t,$$

where u_t is a serially uncorrelated error term. This model states that the first difference of a series is a random walk. Consequently, the series of second differences of Y_t is stationary. Key Concept 16.4 summarizes the notation.

Key Concept 16.4

Orders of Integration, Differencing and Stationarity

- When a time series Y_t has a unit autoregressive root, Y_t is integrated of order one. This is often denoted by $Y_t \sim I(1)$. We simply say that Y_t is $I(1)$. If Y_t is $I(1)$, its first difference ΔY_t is stationary.
- Y_t is $I(2)$ when Y_t needs to be differenced twice in order to obtain a stationary series. Using the notation introduced here, Y_t is $I(2)$, its first difference ΔY_t is $I(1)$ and its second difference $\Delta^2 Y_t$ is stationary. Y_t is $I(d)$ when Y_t must be differenced d times to obtain a stationary series.
- When Y_t is stationary, it is integrated of order 0 so Y_t is $I(0)$.

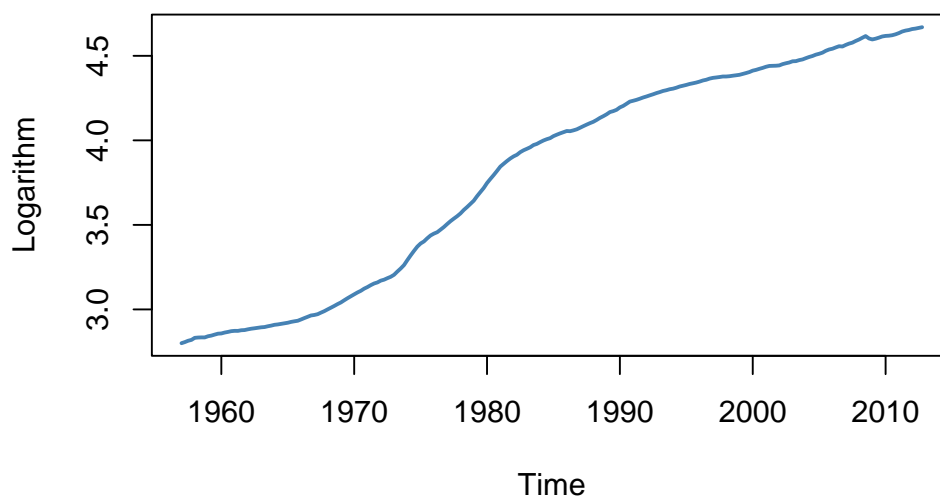
It is fairly easy to obtain differences of time series in R. For example, the function `diff()` returns suitably lagged and iterated differences of numeric vectors, matrices and time series objects of the class `ts`.

Following the book, we take the price level of the U.S. measured by the *Personal Consumption Expenditures Price Index* as an example.

```
# define ts object of the U.S. PCE Price Index
PCECTPI <- ts(log(USMacroSWQ$PCECTPI),
             start = c(1957, 1),
             end = c(2012, 4),
             freq = 4)

# plot logarithm of the PCE Price Index
plot(PCECTPI,
     main = "Log of United States PCE Price Index",
     ylab = "Logarithm",
     col = "steelblue",
     lwd = 2)
```


Log of United States PCE Price Index

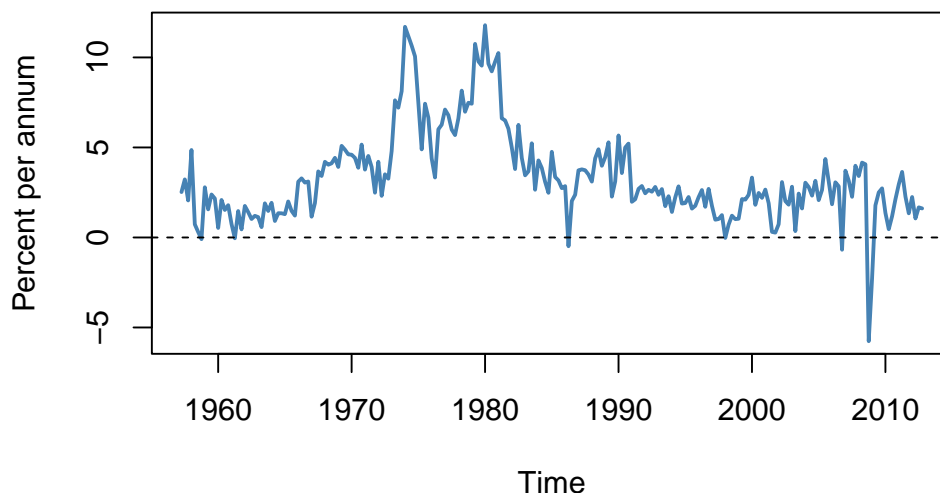


The logarithm of the price level has a smoothly varying trend. This is typical for an $I(2)$ series. If the price level is indeed $I(2)$, the first differences of this series should be $I(1)$. Since we are considering the logarithm of the price level, we obtain growth rates by taking first differences. Therefore, the differenced price level series is the series of quarterly inflation rates. This is quickly done in R using the function `diff()` from the package `base`. As explained in Chapter 14.2, multiplying the quarterly inflation rates by 400 yields the quarterly rate of inflation, measured in percentage points at an annual rate.

```
# plot U.S. Inflation Rate
plot(400 * diff(PCECTPI),
     main = "United States Inflation Rate",
     ylab = "Percent per annum",
     col = "steelblue",
     lwd = 2)

# add a dashed line at y = 0
abline(0, 0, lty = 2)
```

United States Inflation Rate



The inflation rate behaves much more erratically than the smooth graph of the logarithm of the PCE price index.

The DF-GLS Test for a Unit Root

The DF-GLS test for a unit root has been developed by Elliott et al. (1996) and has higher power than the ADF test when the autoregressive root is large but less than one. That is, the DF-GLS has a higher probability of rejecting the false null of a stochastic trend when the sample data stems from a time series that is close to being integrated.

The idea of the DF-GLS test is to test for an autoregressive unit root in the detrended series, whereby GLS estimates of the deterministic components are used to obtain the detrended version of the original series. See Chapter 16.3 of the book for a more detailed explanation of the approach.

A function that performs the DF-GLS test is implemented in the package `urca` (this package is a dependency of the package `vars` so it should be already loaded if `vars` is attached). The function that computes the test statistic is `ur.ers`.

```
# DF-GLS test for unit root in GDP
summary(ur.ers(log(window(GDP, start = c(1962, 1), end = c(2012, 4))),
  model = "trend",
  lag.max = 2))

#>
#> #####
#> # Elliot, Rothenberg and Stock Unit Root Test #
#> #####
#>
#> Test of type DF-GLS
#> detrending of series with intercept and trend
#>
#>
#> Call:
#> lm(formula = dfpls.form, data = data.dfpls)
#>
#> Residuals:
```

```

#>      Min      1Q   Median      3Q      Max
#> -0.025739 -0.004054  0.000017  0.004619  0.033620
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> yd.lag        -0.01213    0.01012  -1.199  0.23207
#> yd.diff.lag1  0.28583    0.07002   4.082 6.47e-05 ***
#> yd.diff.lag2  0.19320    0.07058   2.737  0.00676 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.007807 on 198 degrees of freedom
#> Multiple R-squared:  0.1504, Adjusted R-squared:  0.1376
#> F-statistic: 11.69 on 3 and 198 DF,  p-value: 4.392e-07
#>
#>
#> Value of test-statistic is: -1.1987
#>
#> Critical values of DF-GLS are:
#>              1pct  5pct 10pct
#> critical values -3.48 -2.89 -2.57

```

The summary of the test shows that the test statistic is about -1.2 . The 10% critical value for the DF-GLS test is -2.57 . This is, however, **not** the appropriate critical value for the ADF test when an intercept and a time trend are included in the Dickey-Fuller regression: the asymptotic distributions of both test statistics differ and so do their critical values!

The test is left-sided so we cannot reject the null hypothesis that U.S. inflation is nonstationary, using the DF-GLS test.

16.3 Cointegration

Key Concept 16.5 Cointegration

When X_t and Y_t are $I(1)$ and if there is a θ such that $Y_t - \theta X_t$ is $I(0)$, X_t and Y_t are cointegrated. Put differently, cointegration of X_t and Y_t means that X_t and Y_t have the same or a common stochastic trend and that this trend can be eliminated by taking a specific difference of the series such that the resulting series is stationary.

R functions for cointegration analysis are implemented in the package `urca`.

As an example, reconsider the relation between short-term and long-term interest rates in the example of U.S. 3-month treasury bills, U.S. 10-years treasury bonds and the spread in their interest rates which have been introduced in Chapter 14.4. The next code chunk shows how to reproduce Figure 16.2 of the book.

```

# reproduce Figure 16.2 of the book

# plot both interest series
plot(merge(as.zoo(TB3MS), as.zoo(TB10YS)),

```

```

plot.type = "single",
lty = c(2, 1),
lwd = 2,
xlab = "Date",
ylab = "Percent per annum",
ylim = c(-5, 17),
main = "Interest Rates")

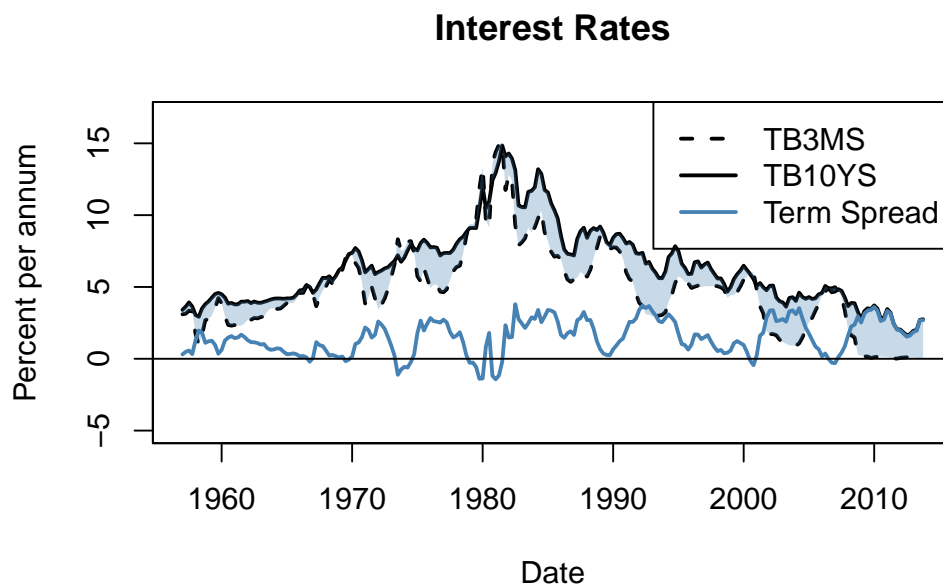
# add the term spread series
lines(as.zoo(TSpread),
      col = "steelblue",
      lwd = 2,
      xlab = "Date",
      ylab = "Percent per annum",
      main = "Term Spread")

# shade the term spread
polygon(c(time(TB3MS), rev(time(TB3MS))),
        c(TB10YS, rev(TB3MS)),
        col = alpha("steelblue", alpha = 0.3),
        border = NA)

# add horizontal line at 0
abline(0, 0)

# add a legend
legend("topright",
      legend = c("TB3MS", "TB10YS", "Term Spread"),
      col = c("black", "black", "steelblue"),
      lwd = c(2, 2, 2),
      lty = c(2, 1, 1))

```



The plot suggests that long-term and short-term interest rates are cointegrated: both interest series

seem to have the same long-run behavior. They share a common stochastic trend. The term spread, which is obtained by taking the difference between long-term and short-term interest rates, seems to be stationary. In fact, the expectations theory of the term structure suggests the cointegrating coefficient θ to be 1. This is consistent with the visual result.

Testing for Cointegration

Following Key Concept 16.5, it seems natural to construct a test for cointegration of two series in the following manner: if two series X_t and Y_t are cointegrated, the series obtained by taking the difference $Y_t - \theta X_t$ must be stationary. If the series are not cointegrated, $Y_t - \theta X_t$ is nonstationary. This is an assumption that can be tested using a unit root test. We have to distinguish between two cases:

1. θ is known.

Knowledge of θ enables us to compute differences $z_t = Y_t - \theta X_t$ so that Dickey-Fuller and DF-GLS unit root tests can be applied to z_t . For these tests, the critical values are the critical values of the ADF or DF-GLS test.

2. θ is unknown.

If θ is unknown, it must be estimated before the unit root test can be applied. This is done by estimating the regression

$$Y_t = \alpha + \theta X_t + z_t$$

using OLS (this is referred to as the first-stage regression). Then, a Dickey-Fuller test is used for testing the hypothesis that z_t is a nonstationary series. This is known as the Engle-Granger Augmented Dickey-Fuller test for cointegration (or *EG-ADF test*) after Engle and Granger (1987). The critical values for this test are special as the associated null distribution is nonnormal and depends on the number of $I(1)$ variables used as regressors in the first stage regression. You may look them up in Table 16.2 of the book. When there are only two presumably cointegrated variables (and thus a single $I(1)$ variable is used in the first stage OLS regression) the critical values for the levels 10%, 5% and 1% are -3.12 , -3.41 and -3.96 .

Application to Interest Rates

As has been mentioned above, the theory of the term structure suggests that long-term and short-term interest rates are cointegrated with a cointegration coefficient of $\theta = 1$. In the previous section we have seen that there is visual evidence for this conjecture since the spread of 10-year and 3-month interest rates looks stationary.

We continue by using formal tests (the ADF and the DF-GLS test) to see whether the individual interest rate series are integrated and if their difference is stationary (for now, we assume that $\theta = 1$ is known). Both is conveniently done by using the functions `ur.df()` for computation of the ADF test and `ur.ers` for conducting the DF-GLS test. Following the book we use data from 1962:Q1 to 2012:Q4 and employ models that include a drift term. We set the maximum lag order to 6 and use the *AIC* for selection of the optimal lag length.

```
# test for nonstationarity of 3-month treasury bills using ADF test
ur.df(window(TB3MS, c(1962, 1), c(2012, 4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")
#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
#> #####
```

```

#>
#> The value of the test statistic is: -2.1004 2.2385

# test for nonstationarity of 10-years treasury bonds using ADF test
ur.df(window(TB10YS, c(1962, 1), c(2012, 4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")
#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -1.0079 0.5501

# test for nonstationarity of 3-month treasury bills using DF-GLS test
ur.ers(window(TB3MS, c(1962, 1), c(2012, 4)),
       model = "constant",
       lag.max = 6)
#>
#> #####
#> # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -1.8042

# test for nonstationarity of 10-years treasury bonds using DF-GLS test
ur.ers(window(TB10YS, c(1962, 1), c(2012, 4)),
       model = "constant",
       lag.max = 6)
#>
#> #####
#> # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -0.942

```

The corresponding 10% critical value for both tests is -2.57 so we cannot reject the null hypotheses of nonstationary for either series, even at the 10% level of significance.¹ We conclude that it is plausible to model both interest rate series as $I(1)$.

Next, we apply the ADF and the DF-GLS test to test for nonstationarity of the term spread series, which means we test for non-cointegration of long- and short-term interest rates.

```

# test if term spread is stationary (cointegration of interest rates) using ADF
ur.df(window(TB10YS, c(1962, 1), c(2012, 4)) - window(TB3MS, c(1962, 1), c(2012, 4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")

```

¹Note: `ur.df()` reports two test statistics when there is a drift in the ADF regression. The first of which (the one we are interested in here) is the t -statistic for the test that the coefficient on the first lag of the series is 0. The second one is the t -statistic for the hypothesis test that the drift term equals 0.

```

#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -3.9308 7.7362

# test if term spread is stationary (cointegration of interest rates) using DF-GLS
ur.ers(window(TB10YS, c(1962, 1), c(2012, 4)) - window(TB3MS, c(1962, 1),c(2012, 4)),
        model = "constant",
        lag.max = 6)
#>
#> #####
#> # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -3.8576

```

Table 16.1 summarizes the results.

Table 16.1: ADF and DF-GLS Test Statistics for Interest Rate Series

Series	ADF Test Statistic	DF-GLS Test Statistic
TB3MS	-2.10	-1.80
TB10YS	-1.01	-0.94
TB10YS - TB3MS	-3.93	-3.86

Both tests reject the hypothesis of nonstationarity of the term spread series at the 1% level of significance, which is strong evidence in favor of the hypothesis that the term spread is stationary, implying cointegration of long- and short-term interest rates.

Since theory suggests that $\theta = 1$, there is no need to estimate θ so it is not necessary to use the EG-ADF test which allows θ to be unknown. However, since it is instructive to do so, we follow the book and compute this test statistic. The first-stage OLS regression is

$$TB10YS_t = \beta_0 + \beta_1 TB3MS_t + z_t.$$

```

# estimate first-stage regression of EG-ADF test
FS_EGADF <- dynlm(window(TB10YS, c(1962, 1), c(2012, 4)) ~ window(TB3MS, c(1962, 1),
                                                                    c(2012, 4)))

FS_EGADF
#>
#> Time series regression with "ts" data:
#> Start = 1962(1), End = 2012(4)
#>
#> Call:
#> dynlm(formula = window(TB10YS, c(1962, 1), c(2012, 4)) ~ window(TB3MS,
#>   c(1962, 1), c(2012, 4)))
#>
#> Coefficients:

```

```
#> (Intercept) window(TB3MS, c(1962, 1), c(2012, 4))
#> 2.4642 0.8147
```

Thus we have

$$T\widehat{B10Y}S_t = 2.46 + 0.81 \cdot TB3MS_t,$$

where $\hat{\theta} = 0.81$. Next, we take the residual series $\{\hat{z}_t\}$ and compute the ADF test statistic.

```
# compute the residuals
z_hat <- resid(FS_EGADF)

# compute the ADF test statistic
ur.df(z_hat, lags = 6, type = "none", selectlags = "AIC")
#>
#> #####
#> # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
#> #####
#>
#> The value of the test statistic is: -3.1935
```

The test statistic is -3.19 which is smaller than the 10% critical value but larger than the 5% critical value (see Table 16.2 of the book). Thus, the null hypothesis of no cointegration can be rejected at the 10% level but not at the 5% level. This indicates lower power of the EG-ADF test due to the estimation of θ : when $\theta = 1$ is the correct value, we expect the power of the ADF test for a unit root in the residuals series $\hat{z}_t = TB10YS - TB3MS$ to be higher than when some estimate $\hat{\theta}$ is used.

A Vector Error Correction Model for $TB10YS_t$ and $TB3MS$ If two $I(1)$ time series X_t and Y_t are cointegrated, their differences are stationary and can be modeled in a VAR which is augmented by the regressor $Y_{t-1} - \theta X_{t-1}$. This is called a *vector error correction model* (VECM) and $Y_t - \theta X_t$ is called the *error correction term*. Lagged values of the error correction term are useful for predicting ΔX_t and/or ΔY_t .

A VECM can be used to model the two interest rates considered in the previous sections. Following the book we specify the VECM to include two lags of both series as regressors and choose $\theta = 1$, as theory suggests (see above).

```
TB10YS <- window(TB10YS, c(1962, 1), c(2012, 4))
TB3MS <- window(TB3MS, c(1962, 1), c(2012, 4))

# set up error correction term
VECM_ECT <- TB10YS - TB3MS

# estimate both equations of the VECM using 'dynlm()'
VECM_EQ1 <- dynlm(d(TB10YS) ~ L(d(TB3MS), 1:2) + L(d(TB10YS), 1:2) + L(VECM_ECT))
VECM_EQ2 <- dynlm(d(TB3MS) ~ L(d(TB3MS), 1:2) + L(d(TB10YS), 1:2) + L(VECM_ECT))

# rename regressors for better readability
names(VECM_EQ1$coefficients) <- c("Intercept", "D_TB3MS_11", "D_TB3MS_12",
                                  "D_TB10YS_11", "D_TB10YS_12", "ect_11")
names(VECM_EQ2$coefficients) <- names(VECM_EQ1$coefficients)

# coefficient summaries using HAC standard errors
coeftest(VECM_EQ1, vcov. = NeweyWest(VECM_EQ1, prewhite = F, adjust = T))
```



```

#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> Intercept    0.1227089  0.0551419  2.2253 0.027205 *
#> D_TB3MS_l1  -0.0016601  0.0727060 -0.0228 0.981807
#> D_TB3MS_l2  -0.0680845  0.0435059 -1.5649 0.119216
#> D_TB10YS_l1 0.2264878  0.0957071  2.3665 0.018939 *
#> D_TB10YS_l2 -0.0734486  0.0703476 -1.0441 0.297740
#> ect_l1      -0.0878871  0.0285644 -3.0768 0.002393 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coefest(VECM_EQ2, vcov. = NeweyWest(VECM_EQ2, prewhite = F, adjust = T))
#>
#> t test of coefficients:
#>
#>           Estimate Std. Error t value Pr(>|t|)
#> Intercept   -0.060746  0.107937 -0.5628 0.57422
#> D_TB3MS_l1  0.240003  0.111611  2.1504 0.03276 *
#> D_TB3MS_l2  -0.155883  0.153845 -1.0132 0.31220
#> D_TB10YS_l1 0.113740  0.125571  0.9058 0.36617
#> D_TB10YS_l2 -0.147519  0.112630 -1.3098 0.19182
#> ect_l1      0.031506  0.050519  0.6236 0.53359
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Thus the two estimated equations of the VECM are

$$\begin{aligned}
 \widehat{\Delta TB3MS}_t &= -0.06 + 0.24 \Delta TB3MS_{t-1} - 0.16 \Delta TB3MS_{t-2} \\
 &\quad \begin{matrix} (0.11) & (0.11) & (0.15) \end{matrix} \\
 &\quad + 0.11 \Delta TB10YS_{t-1} - 0.15 \Delta TB10YS_{t-2} + 0.03 ECT_{t-1}, \text{ and,} \\
 &\quad \begin{matrix} (0.13) & (0.11) & (0.05) \end{matrix} \\
 \widehat{\Delta TB10YS}_t &= 0.12 - 0.00 \Delta TB3MS_{t-1} - 0.07 \Delta TB3MS_{t-2} \\
 &\quad \begin{matrix} (0.06) & (0.07) & (0.04) \end{matrix} \\
 &\quad + 0.23 \Delta TB10YS_{t-1} - 0.07 \Delta TB10YS_{t-2} - 0.09 ECT_{t-1}. \\
 &\quad \begin{matrix} (0.10) & (0.07) & (0.03) \end{matrix}
 \end{aligned}$$

The output produced by `coefest()` shows that there is little evidence that lagged values of the differenced interest series are useful for prediction. This finding is more pronounced for the equation of the differenced series of the 3-month treasury bill rate, where the error correction term (the lagged term spread) is not significantly different from zero at any common level of significance. However, for the differenced 10-years treasury bonds rate the error correction term is statistically significant at 1% with an estimate of -0.09 . This can be interpreted as follows: although both interest rates are nonstationary, their cointegrating relationship allows to predict the *change* in the 10-years treasury bonds rate using the VECM. In particular, the negative estimate of the coefficient on the error correction term indicates that there will be a negative change in the next period's 10-years treasury bonds rate when the 10-years treasury bonds rate is unusually high relative to the 3-month treasury bill rate in the current period.

16.4 Volatility Clustering and Autoregressive Conditional Heteroskedasticity

Financial time series often exhibit a behavior that is known as *volatility clustering*: the volatility changes over time and its degree shows a tendency to persist, i.e., there are periods of low volatility and periods where volatility is high. Econometricians call this *autoregressive conditional heteroskedasticity*. Conditional heteroskedasticity is an interesting property because it can be exploited for forecasting the variance of future periods.

As an example, we consider daily changes in the Wilshire 5000 stock index. The data is available for download at the Federal Reserve Economic Data Base. For consistency with the book we download data from 1989-29-12 to 2013-12-31 (choosing this somewhat larger time span is necessary since later on we will be working with daily changes of the series).

The following code chunk shows how to format the data and how to reproduce Figure 16.3 of the book.

```
# import data on the Wilshire 5000 index
W5000 <- read.csv2("data/Wilshire5000.csv",
                 stringsAsFactors = F,
                 header = T,
                 sep = ",",
                 na.strings = ".")

# transform the columns
W5000$DATE <- as.Date(W5000$DATE)
W5000$WILL5000INDFC <- as.numeric(W5000$WILL5000INDFC)

# remove NAs
W5000 <- na.omit(W5000)

# compute daily percentage changes
W5000_PC <- data.frame("Date" = W5000$DATE,
                      "Value" = as.numeric(Delt(W5000$WILL5000INDFC) * 100))
W5000_PC <- na.omit(W5000_PC)

# plot percentage changes
plot(W5000_PC,
     ylab = "Percent",
     main = "Daily Percentage Changes",
     type = "l",
     col = "steelblue",
     lwd = 0.5)

# add horizontal line at y = 0
abline(0, 0)
```

The series of daily percentage changes in the Wilshire index seems to randomly fluctuate around zero, meaning there is little autocorrelation. This is confirmed by a plot of the sample autocorrelation function.

```
# plot sample autocorrelation of daily percentage changes
acf(W5000_PC$Value, main = "Wilshire 5000 Series")
```

In Figure 16.2 we see that autocorrelations are rather weak so that it is difficult to predict future

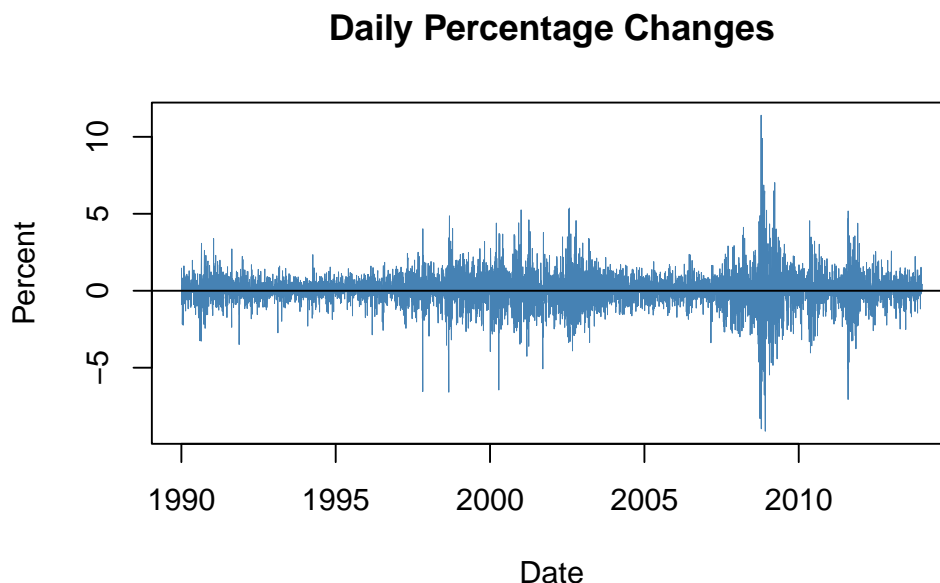


Figure 16.1: Daily Percentage Returns in the Wilshire 5000 Index

outcomes using, e.g., an AR model.

However, there is visual evidence in 16.1 that the series of returns exhibits conditional heteroskedasticity since we observe volatility clustering. For some applications it is useful to measure and forecast these patterns. This can be done using models which assume that the volatility can be described by an autoregressive process.

ARCH and GARCH Models

Consider

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \gamma_1 X_{t-1} + u_t,$$

an ADL(1,1) regression model. The econometrician Robert Engle (1982) proposed to model $\sigma_t^2 = \text{Var}(u_t | u_{t-1}, u_{t-2}, \dots)$, the conditional variance of the error u_t given its past, by an order p distributed lag model,

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \dots + \alpha_p u_{t-p}^2, \quad (16.1)$$

called the *autoregressive conditional heteroskedasticity* (ARCH) model of order p , or short ARCH(p)². We assume $\alpha_0 > 0$ and $\alpha_1, \dots, \alpha_p \geq 0$ to ensure a positive variance $\sigma_t^2 > 0$. The general idea is apparent from the model structure: positive coefficients $\alpha_0, \alpha_1, \dots, \alpha_p$ imply that recent large squared errors lead to a large variance, and thus large squared errors, in the current period.

The generalized ARCH (GARCH) model, developed by Tim Bollerslev (1986), is an extension of the ARCH model, where σ_t^2 is allowed to depend on its own lags and lags of the squared error term. The GARCH(p, q) model is given by

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \dots + \alpha_p u_{t-p}^2 + \phi_1 \sigma_{t-1}^2 + \dots + \phi_q \sigma_{t-q}^2. \quad (16.2)$$

The GARCH model is an ADL(p, q) model and thus can provide more parsimonious parameterizations than the ARCH model (see the discussion in Appendix 15.2 of the book).

²Although we introduce the ARCH model as a component in an ADL(1,1) model, it can be used for modelling the conditional zero-mean error term of any time series model.

Wilshire 5000 Series

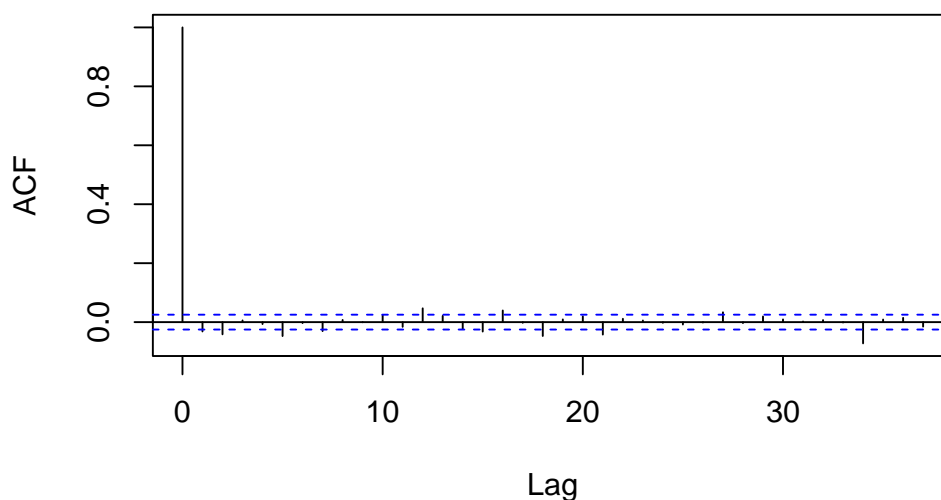


Figure 16.2: Autocorrelation in Daily Price Changes of W5000 Index

Application to Stock Price Volatility

Maximum likelihood estimates of ARCH and GARCH models are efficient and have normal distributions in large samples, such that the usual methods for conducting inference about the unknown parameters can be applied. The R package `fGarch` is a collection of functions for analyzing and modelling the heteroskedastic behavior in time series models. The following application reproduces the results presented in Chapter 16.5 of the book by means of the function `garchFit()`. This function is somewhat sophisticated. It allows for different specifications of the optimization procedure, different error distributions and much more (use `?GarchFit` for a detailed description of the arguments). In particular, the reported standard errors by `garchFit()` are robust.

The GARCH(1,1) model of daily changes in the Wilshire 5000 index we estimate is given by

$$R_t = \beta_0 + u_t, \quad u_t \sim \mathcal{N}(0, \sigma_t^2), \quad (16.3)$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \phi_1 \sigma_{t-1}^2 \quad (16.4)$$

where R_t is the percentage change in period t . β_0 , α_0 , α_1 and ϕ_1 are unknown coefficients and u_t is an error term with conditional mean zero. We do not include any lagged predictors in the equation of R_t because the daily changes in the Wilshire 5000 index reflect daily stock returns which are essentially unpredictable. Note that u_t is assumed to be normally distributed and the variance σ_t^2 depends on t as it follows the GARCH(1,1) recursion (16.4).

It is straightforward to estimate this model using `garchFit()`.

```
# estimate GARCH(1,1) model of daily percentage changes
GARCH_Wilshire <- garchFit(data = W5000_PC$Value, trace = F)
```

We obtain

$$\hat{R}_t = 0.068, \quad (0.010) \quad (16.5)$$

$$\hat{\sigma}_t^2 = 0.011 + 0.081 u_{t-1}^2 + 0.909 \sigma_{t-1}^2, \quad (0.002) \quad (0.007) \quad (0.008) \quad (16.6)$$

so the coefficients on u_{t-1}^2 and σ_{t-1}^2 are statistically significant at any common level of significance. One can show that the persistence of movements in σ_t^2 is determined by the sum of both coefficients, which is 0.99 here. This indicates that movements in the conditional variance are highly persistent, implying long-lasting periods of high volatility which is consistent with the visual evidence for volatility clustering presented above.

The estimated conditional variance $\hat{\sigma}_t^2$ can be computed by plugging the residuals from (16.5) into equation (16.6). This is performed automatically by `garchFit()`, so to obtain the estimated conditional standard deviations $\hat{\sigma}_t$ we only have to read out the values from `GARCH_Wilshire` by appending `\@sigma.t`.

Using the $\hat{\sigma}_t$ we plot bands of \pm one conditional standard deviation along with deviations of the series of daily percentage changes in the Wilshire 5000 index from its mean. The following code chunk reproduces Figure 16.4 of the book.

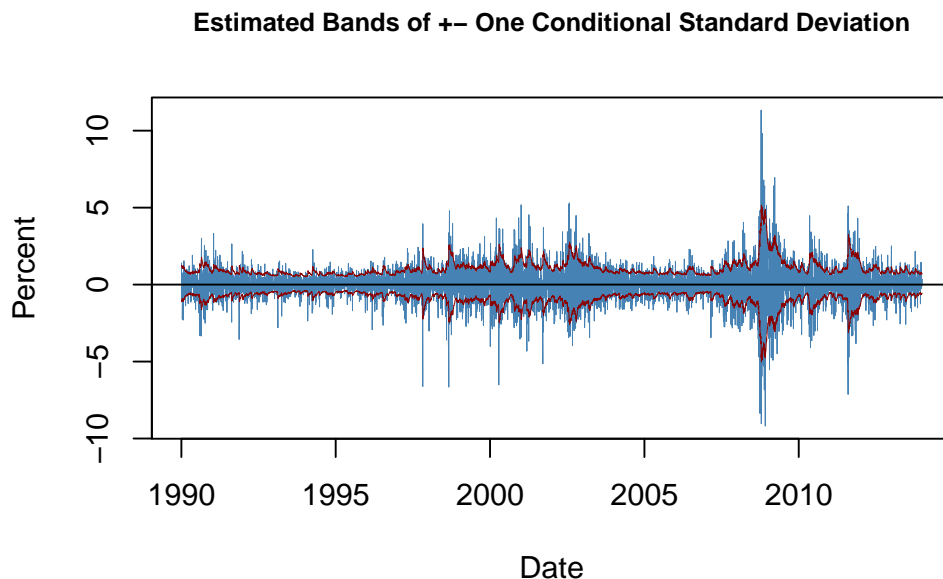
```
# compute deviations of the percentage changes from their mean
dev_mean_W5000_PC <- W5000_PC$Value - GARCH_Wilshire@fit$coef[1]

# plot deviation of percentage changes from mean
plot(W5000_PC$Date, dev_mean_W5000_PC,
     type = "l",
     col = "steelblue",
     ylab = "Percent",
     xlab = "Date",
     main = "Estimated Bands of +- One Conditional Standard Deviation",
     cex.main=0.8,
     lwd = 0.2)

# add horizontal line at y = 0
abline(0, 0)

# add GARCH(1,1) confidence bands (one standard deviation) to the plot
lines(W5000_PC$Date,
      GARCH_Wilshire@fit$coef[1] + GARCH_Wilshire@sigma.t,
      col = "darkred",
      lwd = 0.5)

lines(W5000_PC$Date,
      GARCH_Wilshire@fit$coef[1] - GARCH_Wilshire@sigma.t,
      col = "darkred",
      lwd = 0.5)
```



The bands of the estimated conditional standard deviations track the observed heteroskedasticity in the series of daily changes of the Wilshire 5000 index quite well. This is useful for quantifying the time-varying volatility and the resulting risk for investors holding stocks summarized by the index. Furthermore, this GARCH model may also be used to produce forecast intervals whose widths depend on the volatility of the most recent periods.

Summary

- We have discussed how vector autoregressions are conveniently estimated and used for forecasting in R by means of functions from the `vars` package.
- The package `urca` supplies advanced methods for unit root and cointegration analysis like the DF-GLS and the EG-ADF tests. In an application we have found evidence that 3-months and 10-year interest rates have a common stochastic trend (that is, they are cointegrated) and thus can be modeled using a vector error correction model.
- Furthermore, we have introduced the concept of volatility clustering and demonstrated how the function `garchFit()` from the package `fGarch` can be employed to estimate a GARCH(1,1) model of the conditional heteroskedasticity inherent to returns of the Wilshire 5000 stock index.

Bibliography

- Allaire, J., Xie, Y., Dervieux, C., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2023). *rmarkdown: Dynamic Documents for R*.
- Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31(3):307–327.
- Card, D. and Krueger, A. B. (1994). Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania. *The American Economic Review*, 84(4):772–793.
- Chow, G. C. (1960). Tests of Equality Between Sets of Coefficients in Two Linear Regressions. *Econometrica*, 28(3):591–605.
- Cochrane, D. and Orcutt, G. H. (1949). Application of Least Squares Regression to Relationships Containing Auto-Correlated Error Terms. *Journal of the American Statistical Association*, 44(245):32–61.
- Croissant, Y., Millo, G., and Tappe, K. (2023). *plm: Linear Models for Panel Data*.
- Dickey, D. A. and Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 74(366):pp. 427–431.
- Elliott, G., Rothenberg, T. J., and Stock, J. H. (1996). Efficient Tests for an Autoregressive Unit Root. *Econometrica*, 64(4):813–836.
- Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.
- Engle, R. and Granger, C. (1987). Co-integration and Error Correction: Representation, Estimation and Testing. *Econometrica*, 55(2):251–76.
- Genz, A., Bretz, F., Miwa, T., Mi, X., and Hothorn, T. (2023). *mvtnorm: Multivariate Normal and t Distributions*.
- Granger, C. (1969). Investigating Causal Relations by Econometric Models and Cross-Spectral Methods. *Econometrica*, 37:424–438.
- Heiss, F. (2016). *Using R for Introductory Econometrics*. CreateSpace Independent Publishing Platform.
- Hlavac, M. (2022). *stargazer: Well-Formatted Regression and Summary Statistics Tables*. Social Policy Institute, Bratislava, Slovakia. R package version 5.2.3.
- Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., Kuroptev, K., O’Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. (2023). *forecast: Forecasting Functions for Time Series and Linear Models*.
- Kleiber, C. and Zeileis, A. (2008a). *Applied Econometrics with R*. Springer.

- Kleiber, C. and Zeileis, A. (2008b). *Applied Econometrics with R*. Springer-Verlag, New York. ISBN 978-0-387-77316-2.
- MacKinnon, J. G. and White, H. (1985). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, 29(3):305–325.
- Newey, W. K. and West, K. D. (1987). A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, 55:703–08.
- Pfaff, B. (2008). *Analysis of Integrated and Cointegrated Time Series with R*. Springer, New York, second edition. ISBN 0-387-27960-1.
- Pfaff, B. (2023). *vars: VAR Modelling*.
- Pinheiro, J., Bates, D., and R Core Team (2023). *nlme: Linear and Nonlinear Mixed Effects Models*.
- Quandt, R. E. (1960). Tests of the Hypothesis That a Linear Regression System Obeys Two Separate Regimes. *Journal of the American Statistical Association*, 55(290):324–330.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Ripley, B. (2023). *MASS: Support Functions and Datasets for Venables and Ripley’s MASS*.
- Ryan, J. A. and Ulrich, J. M. (2023). *quantmod: Quantitative Financial Modelling Framework*.
- Stefano, S., Quartagno, M., Tamburini, M., and Robinson, D. (2018). *orcutt: Estimate Procedure in Case of First Order Autocorrelation*.
- Stigler, M. and Quast, B. (2022). *rddtools: Toolbox for Regression Discontinuity Design (RDD)*.
- Stock, J. and Watson, M. (2015). *Introduction to Econometrics, Third Update, Global Edition*. Pearson Education Limited.
- Venables, W. N. and Smith, D. M. (2010). *An Introduction to R*.
- White, H. (1980). A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 48(4):pp. 817–838.
- Wickham, H. and Bryan, J. (2023). *readxl: Read Excel Files*.
- Wickham, H., François, R., Henry, L., Müller, K., and Vaughan, D. (2023a). *dplyr: A Grammar of Data Manipulation*.
- Wickham, H. and Seidel, D. (2022). *scales: Scale Functions for Visualization*.
- Wickham, H., Vaughan, D., and Girlich, M. (2023b). *tidyr: Tidy Messy Data*.
- Wooldridge, J. (2016). *Introductory Econometrics*. Cengage Learning, sixth edition.
- Wuertz, D., Chalabi, Y., Setz, T., Maechler, M., and Boshnakov, G. N. (2023). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*.
- Xie, Y. (2023a). *bookdown: Authoring Books and Technical Documents with R Markdown*.
- Xie, Y. (2023b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*.
- Zeileis, A. (2019). *dynlm: Dynamic Linear Regression*. R package version 0.3-6.