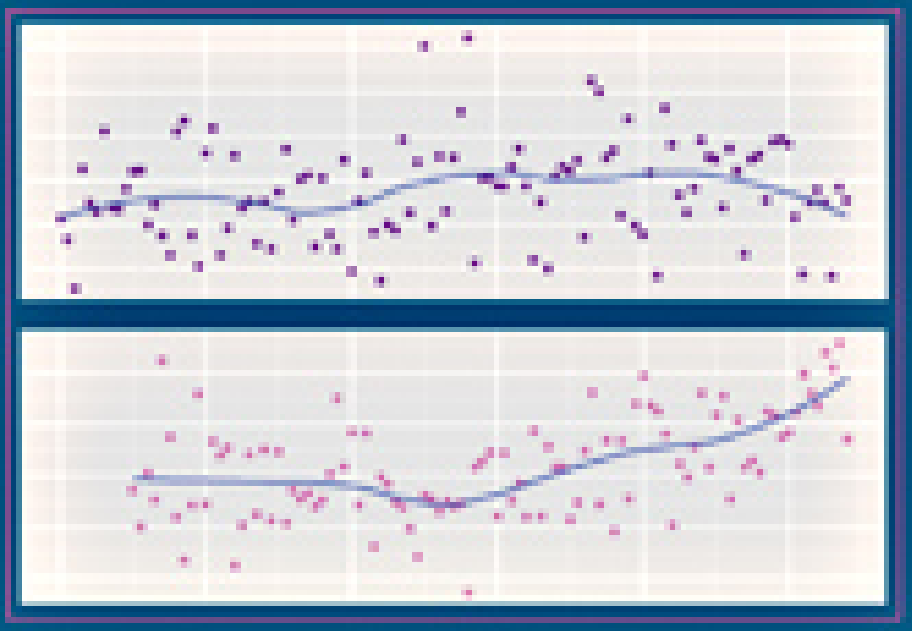
Cambridge Series in Statistical and Probabilistic Mathematics



Data Analysis and Graphics Using R

An Example-Based Approach

Third Edition

John Maindonald and W. John Braun

CAMBRIDGE

CAMBRIDGE www.cambridge.org/9780521762939

This page intentionally left blank

Data Analysis and Graphics Using R, Third Edition

Discover what you can do with R! Introducing the R system, covering standard regression methods, then tackling more advanced topics, this book guides users through the practical, powerful tools that the R system provides. The emphasis is on hands-on analysis, graphical display, and interpretation of data. The many worked examples, from real-world research, are accompanied by commentary on what is done and why. The companion website has code and data sets, allowing readers to reproduce all analyses, along with solutions to selected exercises and updates. Assuming basic statistical knowledge and some experience with data analysis (but not R), the book is ideal for research scientists, final-year undergraduate or graduate-level students of applied statistics, and practicing statisticians. It is both for learning and for reference.

This third edition takes into account recent changes in R, including advances in graphical user interfaces (GUIs) and graphics packages. The treatments of the random forests methodology and one-way analysis have been extended. Both text and code have been revised throughout, and where possible simplified. New graphs and examples have been added.

JOHN MAINDONALD is Visiting Fellow at the Mathematical Sciences Institute at the Australian National University. He has collaborated extensively with scientists in a wide range of application areas, from medicine and public health to population genetics, machine learning, economic history, and forensic linguistics.

W. JOHN BRAUN is Professor in the Department of Statistical and Actuarial Sciences at the University of Western Ontario. He has collaborated with biostatisticians, biologists, psychologists, and most recently has become involved with a network of forestry researchers.

Data Analysis and Graphics Using R – an Example-Based Approach

Third Edition

CAMBRIDGE SERIES IN STATISTICAL AND PROBABILISTIC MATHEMATICS

Editorial Board

Z. Ghahramani (Department of Engineering, University of Cambridge)
R. Gill (Mathematical Institute, Leiden University)
F. P. Kelly (Department of Pure Mathematics and Mathematical Statistics, University of Cambridge)
B. D. Ripley (Department of Statistics, University of Oxford)
S. Ross (Department of Industrial and Systems Engineering, University of Southern California)
B. W. Silverman (St Peter's College, Oxford)
M. Stein (Department of Statistics, University of Chicago)

This series of high quality upper-division textbooks and expository monographs covers all aspects of stochastic applicable mathematics. The topics range from pure and applied statistics to probability theory, operations research, optimization, and mathematical programming. The books contain clear presentations of new developments in the field and also of the state of the art in classical methods. While emphasizing rigorous treatment of theoretical methods, the books also contain applications and discussions of new techniques made possible by advances in computational practice.

A complete list of books in the series can be found at
http://www.cambridge.org/uk/series/sSeries.asp?code=CSPM
Recent titles include the following:

7. *Numerical Methods of Statistics,* by John F. Monahan
8. *A User's Guide to Measure Theoretic Probability,* by David Pollard
9. *The Estimation and Tracking of Frequency,* by B. G. Quinn and E. J. Hannan
10. *Data Analysis and Graphics Using R,* by John Maindonald and John Braun
11. *Statistical Models,* by A. C. Davison
12. *Semiparametric Regression,* by David Ruppert, M. P. Wand and R. J. Carroll
13. *Exercises in Probability,* by Loïc Chaumont and Marc Yor
14. *Statistical Analysis of Stochastic Processes in Time,* by J. K. Lindsey
15. *Measure Theory and Filtering,* by Lakhdar Aggoun and Robert Elliott
16. *Essentials of Statistical Inference,* by G. A. Young and R. L. Smith
17. *Elements of Distribution Theory,* by Thomas A. Severini
18. *Statistical Mechanics of Disordered Systems,* by Anton Bovier
19. *The Coordinate-Free Approach to Linear Models,* by Michael J. Wichura
20. *Random Graph Dynamics,* by Rick Durrett
21. *Networks,* by Peter Whittle
22. *Saddlepoint Approximations with Applications,* by Ronald W. Butler
23. *Applied Asymptotics,* by A. R. Brazzale, A. C. Davison and N. Reid
24. *Random Networks for Communication,* by Massimo Franceschetti and Ronald Meester
25. *Design of Comparative Experiments,* by R. A. Bailey
26. *Symmetry Studies,* by Marlos A. G. Viana
27. *Model Selection and Model Averaging,* by Gerda Claeskens and Nils Lid Hjort
28. *Bayesian Nonparametrics,* edited by Nils Lid Hjort *et al*
29. *From Finite Sample to Asymptotic Methods in Statistics,* by Pranab K. Sen, Julio M. Singer and Antonio C. Pedrosa de Lima
30. *Brownian Motion,* by Peter Mörters and Yuval Peres

Data Analysis and Graphics Using R – an Example-Based Approach

Third Edition

John Maindonald
Mathematical Sciences Institute, Australian National University
and
W. John Braun
Department of Statistical and Actuarial Sciences, University of Western Ontario

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,
São Paulo, Delhi, Dubai, Tokyo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521762939

© Cambridge University Press 2003
Second and third editions © John Maindonald and W. John Braun 2007, 2010

This publication is in copyright. Subject to statutory exception and to the
provision of relevant collective licensing agreements, no reproduction of any part
may take place without the written permission of Cambridge University Press.

First published in print format 2010

ISBN-13 978-0-511-71286-9 eBook (NetLibrary)

ISBN-13 978-0-521-76293-9 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy
of urls for external or third-party internet websites referred to in this publication,
and does not guarantee that any content on such websites is, or will remain,
accurate or appropriate.

For Edward, Amelia and Luke
also Shireen, Peter, Lorraine, Evan and Winifred

For Susan, Matthew and Phillip

Contents

Preface *page* **xix**

Content – how the chapters fit together **xxv**

1 A brief introduction to R 1
1.1 An overview of R 1
1.1.1 A short R session 1
1.1.2 The uses of R 6
1.1.3 Online help 7
1.1.4 Input of data from a file 8
1.1.5 R packages 9
1.1.6 Further steps in learning R 9
1.2 Vectors, factors, and univariate time series 10
1.2.1 Vectors 10
1.2.2 Concatenation – joining vector objects 10
1.2.3 The use of relational operators to compare vector elements 11
1.2.4 The use of square brackets to extract subsets of vectors 11
1.2.5 Patterned data 11
1.2.6 Missing values 12
1.2.7 Factors 13
1.2.8 Time series 14
1.3 Data frames and matrices 14
1.3.1 Accessing the columns of data frames – `with()` and `attach()` 17
1.3.2 Aggregation, stacking, and unstacking 17
1.3.3* Data frames and matrices 18
1.4 Functions, operators, and loops 19
1.4.1 Common useful built-in functions 19
1.4.2 Generic functions, and the class of an object 21
1.4.3 User-written functions 22
1.4.4 `if` Statements 23
1.4.5 Selection and matching 23
1.4.6 Functions for working with missing values 24
1.4.7* Looping 24

1.5 Graphics in R 25
1.5.1 The function `plot( )` and allied functions 25
1.5.2 The use of color 27
1.5.3 The importance of aspect ratio 28
1.5.4 Dimensions and other settings for graphics devices 28
1.5.5 The plotting of expressions and mathematical symbols 29
1.5.6 Identification and location on the figure region 29
1.5.7 Plot methods for objects other than vectors 30
1.5.8 Lattice (trellis) graphics 30
1.5.9 Good and bad graphs 32
1.5.10 Further information on graphics 33
1.6 Additional points on the use of R 33
1.7 Recap 35
1.8 Further reading 36
1.9 Exercises 37

2 Styles of data analysis 43
2.1 Revealing views of the data 43
2.1.1 Views of a single sample 44
2.1.2 Patterns in univariate time series 47
2.1.3 Patterns in bivariate data 49
2.1.4 Patterns in grouped data – lengths of cuckoo eggs 52
2.1.5* Multiple variables and times 53
2.1.6 Scatterplots, broken down by multiple factors 56
2.1.7 What to look for in plots 58
2.2 Data summary 59
2.2.1 Counts 59
2.2.2 Summaries of information from data frames 63
2.2.3 Standard deviation and inter-quartile range 65
2.2.4 Correlation 67
2.3 Statistical analysis questions, aims, and strategies 69
2.3.1 How relevant and how reliable are the data? 70
2.3.2 How will results be used? 70
2.3.3 Formal and informal assessments 71
2.3.4 Statistical analysis strategies 72
2.3.5 Planning the formal analysis 72
2.3.6 Changes to the intended plan of analysis 73
2.4 Recap 73
2.5 Further reading 74
2.6 Exercises 74

3 Statistical models 77
3.1 Statistical models 77
3.1.1 Incorporation of an error or noise component 78
3.1.2 Fitting models – the model formula 80

3.2 Distributions: models for the random component 81
3.2.1 Discrete distributions – models for counts 82
3.2.2 Continuous distributions 84
3.3 Simulation of random numbers and random samples 86
3.3.1 Sampling from the normal and other continuous distributions 87
3.3.2 Simulation of regression data 88
3.3.3 Simulation of the sampling distribution of the mean 88
3.3.4 Sampling from finite populations 90
3.4 Model assumptions 91
3.4.1 Random sampling assumptions – independence 91
3.4.2 Checks for normality 92
3.4.3 Checking other model assumptions 95
3.4.4 Are non-parametric methods the answer? 95
3.4.5 Why models matter – adding across contingency tables 96
3.5 Recap 97
3.6 Further reading 98
3.7 Exercises 98

4 A review of inference concepts 102
4.1 Basic concepts of estimation 102
4.1.1 Population parameters and sample statistics 102
4.1.2 Sampling distributions 102
4.1.3 Assessing accuracy – the standard error 103
4.1.4 The standard error for the difference of means 103
4.1.5* The standard error of the median 104
4.1.6 The sampling distribution of the t-statistic 105
4.2 Confidence intervals and tests of hypotheses 106
4.2.1 A summary of one- and two-sample calculations 109
4.2.2 Confidence intervals and tests for proportions 112
4.2.3 Confidence intervals for the correlation 113
4.2.4 Confidence intervals versus hypothesis tests 113
4.3 Contingency tables 114
4.3.1 Rare and endangered plant species 116
4.3.2 Additional notes 119
4.4 One-way unstructured comparisons 119
4.4.1 Multiple comparisons 122
4.4.2 Data with a two-way structure, i.e., two factors 123
4.4.3 Presentation issues 124
4.5 Response curves 125
4.6 Data with a nested variation structure 126
4.6.1 Degrees of freedom considerations 127
4.6.2 General multi-way analysis of variance designs 127
4.7 Resampling methods for standard errors, tests, and confidence intervals 128
4.7.1 The one-sample permutation test 128
4.7.2 The two-sample permutation test 129

4.7.3* Estimating the standard error of the median: bootstrapping 130
4.7.4 Bootstrap estimates of confidence intervals 131
4.8 Theories of inference* 132
4.8.1 Maximum likelihood estimation 133
4.8.2 Bayesian estimation 133
4.8.3 If there is strong prior information, use it! 135
4.9 Recap 135
4.10 Further reading 136
4.11 Exercises 137

5 Regression with a single predictor 142
5.1 Fitting a line to data 142
5.1.1 Summary information – lawn roller example 143
5.1.2 Residual plots 143
5.1.3 Iron slag example: is there a pattern in the residuals? 145
5.1.4 The analysis of variance table 147
5.2 Outliers, influence, and robust regression 147
5.3 Standard errors and confidence intervals 149
5.3.1 Confidence intervals and tests for the slope 150
5.3.2 SEs and confidence intervals for predicted values 150
5.3.3* Implications for design 151
5.4 Assessing predictive accuracy 152
5.4.1 Training/test sets and cross-validation 153
5.4.2 Cross-validation – an example 153
5.4.3* Bootstrapping 155
5.5 Regression versus qualitative anova comparisons – issues of power 158
5.6 Logarithmic and other transformations 160
5.6.1* A note on power transformations 160
5.6.2 Size and shape data – allometric growth 161
5.7 There are two regression lines! 162
5.8 The model matrix in regression 163
5.9 Bayesian regression estimation using the* MCMCpack *package* 165
5.10 Recap 166
5.11 Methodological references 167
5.12 Exercises 167

6 Multiple linear regression 170
6.1 Basic ideas: a book weight example 170
6.1.1 Omission of the intercept term 172
6.1.2 Diagnostic plots 173
6.2 The interpretation of model coefficients 174
6.2.1 Times for Northern Irish hill races 174
6.2.2 Plots that show the contribution of individual terms 177
6.2.3 Mouse brain weight example 179
6.2.4 Book dimensions, density, and book weight 181

6.3 Multiple regression assumptions, diagnostics, and efficacy measures 183
6.3.1 Outliers, leverage, influence, and Cook's distance 183
6.3.2 Assessment and comparison of regression models 186
6.3.3 How accurately does the equation predict? 187
6.4 A strategy for fitting multiple regression models 189
6.4.1 Suggested steps 190
6.4.2 Diagnostic checks 191
6.4.3 An example – Scottish hill race data 191
6.5 Problems with many explanatory variables 196
6.5.1 Variable selection issues 197
6.6 Multicollinearity 199
6.6.1 The variance inflation factor 201
6.6.2 Remedies for multicollinearity 203
6.7 Errors in x 203
6.8 Multiple regression models – additional points 208
6.8.1 Confusion between explanatory and response variables 208
6.8.2 Missing explanatory variables 208
6.8.3* The use of transformations 210
6.8.4* Non-linear methods – an alternative to transformation? 210
6.9 Recap 212
6.10 Further reading 212
6.11 Exercises 214

7 Exploiting the linear model framework 217
7.1 Levels of a factor – using indicator variables 217
7.1.1 Example – sugar weight 217
7.1.2 Different choices for the model matrix when there are factors 220
7.2 Block designs and balanced incomplete block designs 222
7.2.1 Analysis of the rice data, allowing for block effects 222
7.2.2 A balanced incomplete block design 223
7.3 Fitting multiple lines 224
7.4 Polynomial regression 228
7.4.1 Issues in the choice of model 229
7.5 Methods for passing smooth curves through data* 231
7.5.1 Scatterplot smoothing – regression splines 232
7.5.2* Roughness penalty methods and generalized additive models 235
7.5.3 Distributional assumptions for automatic choice of roughness penalty 236
7.5.4 Other smoothing methods 236
7.6 Smoothing with multiple explanatory variables 238
7.6.1 An additive model with two smooth terms 238
7.6.2* A smooth surface 240
7.7 Further reading 240
7.8 Exercises 240

8 Generalized linear models and survival analysis 244
8.1 Generalized linear models 244
8.1.1 Transformation of the expected value on the left 244
8.1.2 Noise terms need not be normal 245
8.1.3 Log odds in contingency tables 245
8.1.4 Logistic regression with a continuous explanatory variable 246
8.2 Logistic multiple regression 249
8.2.1 Selection of model terms, and fitting the model 252
8.2.2 Fitted values 254
8.2.3 A plot of contributions of explanatory variables 255
8.2.4 Cross-validation estimates of predictive accuracy 255
8.3 Logistic models for categorical data – an example 256
8.4 Poisson and quasi-Poisson regression 258
8.4.1 Data on aberrant crypt foci 258
8.4.2 Moth habitat example 261
8.5 Additional notes on generalized linear models 266
8.5.1* Residuals, and estimating the dispersion 266
8.5.2 Standard errors and z- or t-statistics for binomial models 267
8.5.3 Leverage for binomial models 268
8.6 Models with an ordered categorical or categorical response 268
8.6.1 Ordinal regression models 269
8.6.2* Loglinear models 272
8.7 Survival analysis 272
8.7.1 Analysis of the `Aids2` data 273
8.7.2 Right-censoring prior to the termination of the study 275
8.7.3 The survival curve for male homosexuals 276
8.7.4 Hazard rates 276
8.7.5 The Cox proportional hazards model 277
8.8 Transformations for count data 279
8.9 Further reading 280
8.10 Exercises 281

9 Time series models 283
9.1 Time series – some basic ideas 283
9.1.1 Preliminary graphical explorations 283
9.1.2 The autocorrelation and partial autocorrelation function 284
9.1.3 Autoregressive models 285
9.1.4* Autoregressive moving average models – theory 287
9.1.5 Automatic model selection? 288
9.1.6 A time series forecast 289
9.2 Regression modeling with ARIMA errors* 291
9.3 Non-linear time series* 298
9.4 Further reading 300
9.5 Exercises 301

10 Multi-level models and repeated measures 303
10.1 A one-way random effects model 304
10.1.1 Analysis with `aov()` 305
10.1.2 A more formal approach 308
10.1.3 Analysis using `lmer()` 310
10.2 Survey data, with clustering 313
10.2.1 Alternative models 313
10.2.2 Instructive, though faulty, analyses 318
10.2.3 Predictive accuracy 319
10.3 A multi-level experimental design 319
10.3.1 The anova table 321
10.3.2 Expected values of mean squares 322
10.3.3* The analysis of variance sums of squares breakdown 323
10.3.4 The variance components 325
10.3.5 The mixed model analysis 326
10.3.6 Predictive accuracy 328
10.4 Within- and between-subject effects 329
10.4.1 Model selection 329
10.4.2 Estimates of model parameters 331
10.5 A generalized linear mixed model 332
10.6 Repeated measures in time 334
10.6.1 Example – random variation between profiles 336
10.6.2 Orthodontic measurements on children 340
10.7 Further notes on multi-level and other models with correlated errors 344
10.7.1 Different sources of variance – complication or focus of interest? 344
10.7.2 Predictions from models with a complex error structure 345
10.7.3 An historical perspective on multi-level models 345
10.7.4 Meta-analysis 347
10.7.5 Functional data analysis 347
10.7.6 Error structure in explanatory variables 347
10.8 Recap 347
10.9 Further reading 348
10.10 Exercises 349

11 Tree-based classification and regression 351
11.1 The uses of tree-based methods 352
11.1.1 Problems for which tree-based regression may be used 352
11.2 Detecting email spam – an example 353
11.2.1 Choosing the number of splits 356
11.3 Terminology and methodology 356
11.3.1 Choosing the split – regression trees 357
11.3.2 Within and between sums of squares 357
11.3.3 Choosing the split – classification trees 358
11.3.4 Tree-based regression versus loess regression smoothing 359

11.4 Predictive accuracy and the cost–complexity trade-off 361
11.4.1 Cross-validation 361
11.4.2 The cost–complexity parameter 362
11.4.3 Prediction error versus tree size 363
11.5 Data for female heart attack patients 363
11.5.1 The one-standard-deviation rule 365
11.5.2 Printed information on each split 366
11.6 Detecting email spam – the optimal tree 366
11.7 The randomForest *package* 369
11.8 Additional notes on tree-based methods 372
11.9 Further reading and extensions 373
11.10 Exercises 374

12 Multivariate data exploration and discrimination 377
12.1 Multivariate exploratory data analysis 378
12.1.1 Scatterplot matrices 378
12.1.2 Principal components analysis 379
12.1.3 Multi-dimensional scaling 383
12.2 Discriminant analysis 385
12.2.1 Example – plant architecture 386
12.2.2 Logistic discriminant analysis 387
12.2.3 Linear discriminant analysis 388
12.2.4 An example with more than two groups 390
12.3 High-dimensional data, classification, and plots* 392
12.3.1 Classifications and associated graphs 394
12.3.2 Flawed graphs 394
12.3.3 Accuracies and scores for test data 398
12.3.4 Graphs derived from the cross-validation process 404
12.4 Further reading 406
12.5 Exercises 407

13 Regression on principal component or discriminant scores 410
13.1 Principal component scores in regression 410
13.2 Propensity scores in regression comparisons – labor training data* 414
13.2.1 Regression comparisons 417
13.2.2 A strategy that uses propensity scores 419
13.3 Further reading 426
13.4 Exercises 426

14 The R system – additional topics 427
14.1 Graphical user interfaces to R 427
14.1.1 The R Commander's interface – a guide to getting started 428
14.1.2 The *rattle* GUI 429
14.1.3 The creation of simple GUIs – the *fgui* package 429
14.2 Working directories, workspaces, and the search list 430

14.2.1* The search path 430
14.2.2 Workspace management 430
14.2.3 Utility functions 431
14.3 R *system configuration* 432
14.3.1 The R Windows installation directory tree 432
14.3.2 The library directories 433
14.3.3 The startup mechanism 433
14.4 Data input and output 433
14.4.1 Input of data 434
14.4.2 Data output 437
14.4.3 Database connections 438
14.5 Functions and operators – some further details 438
14.5.1 Function arguments 439
14.5.2 Character string and vector functions 440
14.5.3 Anonymous functions 441
14.5.4 Functions for working with dates (and times) 441
14.5.5 Creating groups 443
14.5.6 Logical operators 443
14.6 Factors 444
14.7 Missing values 446
14.8 Matrices and arrays* 448
14.8.1 Matrix arithmetic 450
14.8.2 Outer products 451
14.8.3 Arrays 451
14.9 Manipulations with lists, data frames, matrices, and time series 452
14.9.1 Lists – an extension of the notion of "vector" 452
14.9.2 Changing the shape of data frames (or matrices) 454
14.9.3* Merging data frames – `merge()` 455
14.9.4 Joining data frames, matrices, and vectors – `cbind()` 455
14.9.5 The `apply` family of functions 456
14.9.6 Splitting vectors and data frames into lists – `split()` 457
14.9.7 Multivariate time series 458
14.10 Classes and methods 458
14.10.1 Printing and summarizing model objects 459
14.10.2 Extracting information from model objects 460
14.10.3 S4 classes and methods 460
14.11 Manipulation of language constructs 461
14.11.1 Model and graphics formulae 461
14.11.2 The use of a list to pass arguments 462
14.11.3 Expressions 463
14.11.4 Environments 463
14.11.5 Function environments and lazy evaluation 464
14.12 Creation of* R *packages* 465
14.13 Document preparation – `Sweave()` *and* `xtable()` 467
14.14 Further reading 468
14.15 Exercises 469

15 Graphs in R 472
15.1 Hardcopy graphics devices 472
15.2 Plotting characters, symbols, line types, and colors 472
15.3 Formatting and plotting of text and equations 474
15.3.1 Symbolic substitution of symbols in an expression 475
15.3.2 Plotting expressions in parallel 475
15.4 Multiple graphs on a single graphics page 476
15.5 Lattice graphics and the grid *package* 477
15.5.1 Groups within data, and/or columns in parallel 478
15.5.2 Lattice parameter settings 480
15.5.3 Panel functions, strip functions, strip labels, and other annotation 483
15.5.4 Interaction with lattice (and other) plots – the *playwith* package 485
15.5.5 Interaction with lattice plots – focus, interact, unfocus 485
15.5.6 Overlaid plots with different scales 486
15.6 An implementation of Wilkinson's Grammar of Graphics 487
15.7 Dynamic graphics – the rgl *and* `rggobi` *packages* 491
15.8 Further reading 492

Epilogue **493**

References **495**

Index of R symbols and functions **507**

Index of terms **514**

Index of authors **523**

The color plates will be found between pages 328 and 329.

Preface

This book is an exposition of statistical methodology that focuses on ideas and concepts, and makes extensive use of graphical presentation. It avoids, as much as possible, the use of mathematical symbolism. It is particularly aimed at scientists who wish to do statistical analyses on their own data, preferably with reference as necessary to professional statistical advice. It is intended to complement more mathematically oriented accounts of statistical methodology. It may be used to give students with a more specialist statistical interest exposure to practical data analysis.

While no prior knowledge of specific statistical methods or theory is assumed, there is a demand that readers bring with them, or quickly acquire, some modest level of statistical sophistication. Readers should have some prior exposure to statistical methodology, some prior experience of working with real data, and be comfortable with the typing of analysis commands into the computer console. Some prior familiarity with regression and with analysis of variance will be helpful.

We cover a range of topics that are important for many different areas of statistical application. As is inevitable in a book that has this broad focus, there will be investigators working in specific areas – perhaps epidemiology, or psychology, or sociology, or ecology – who will regret the omission of some methodologies that they find important.

We comment extensively on analysis results, noting inferences that seem well-founded, and noting limitations on inferences that can be drawn. We emphasize the use of graphs for gaining insight into data – in advance of any formal analysis, for understanding the analysis, and for presenting analysis results.

The data sets that we use as a vehicle for demonstrating statistical methodology have been generated by researchers in many different fields, and have in many cases featured in published papers. As far as possible, our account of statistical methodology comes from the coalface, where the quirks of real data must be faced and addressed. Features that may challenge the novice data analyst have been retained. The diversity of examples has benefits, even for those whose interest is in a specific application area. Ideas and applications that are useful in one area often find use elsewhere, even to the extent of stimulating new lines of investigation. We hope that our book will stimulate such cross-fertilization.

To summarize: The strengths of this book include the directness of its encounter with research data, its advice on practical data analysis issues, careful critiques of analysis results, the use of modern data analysis tools and approaches, the use of simulation and other computer-intensive methods – where these provide insight or give results that are not otherwise available, attention to graphical and other presentation issues, the use of

examples drawn from across the range of statistical applications, and the inclusion of code that reproduces analyses.

A substantial part of the book was derived, initially, from John Maindonald's lecture notes of courses for researchers, at the University of Newcastle (Australia) over 1996–1997 and at The Australian National University over 1998–2001. Both of us have worked extensively over the material in these chapters.

The R *system*

We use the R system for computations. It began in the early 1990s as a project of Ross Ihaka and Robert Gentleman, who were both at the time working at the University of Auckland (New Zealand). The R system implements a dialect of the influential S language, developed at AT&T Bell Laboratories by Rick Becker, John Chambers, and Allan Wilks, which is the basis for the commercial S-PLUS system. It follows S in its close linkage between data analysis and graphics. Versions of R are available, at no charge, for 32-bit versions of Microsoft Windows, for Linux and other Unix systems, and for the Macintosh. It is available through the Comprehensive R Archive Network (CRAN). Go to `http://cran.r-project.org/`, and find the nearest mirror site.

The development model used for R has proved highly effective in marshalling high levels of computing expertise for continuing improvement, for identifying and fixing bugs, and for responding quickly to the evolving needs and interests of the statistical community. Oversight of "base R" is handled by the R Core Team, whose members are widely drawn internationally. Use is made of code, bug fixes, and documentation from the wider R user community. Especially important are the large number of packages that supplement base R, and that anyone is free to contribute. Once installed, these attach seamlessly into the base system.

Many of the analyses offered by R's packages were not, 20 years ago, available in any of the standard statistical packages. What did data analysts do before we had such packages? Basically, they adapted more simplistic (but not necessarily simpler) analyses as best they could. Those whose skills were unequal to the task did unsatisfactory analyses. Those with more adequate skills carried out analyses that, even if not elegant and insightful by current standards, were often adequate. Tools such as are available in R have reduced the need for the adaptations that were formerly necessary. We can often do analyses that better reflect the underlying science. There have been challenging and exciting changes from the methodology that was typically encountered in statistics courses 15 or 20 years ago.

In the ongoing development of R, priorities have been: the provision of good data manipulation abilities; flexible and high-quality graphics; the provision of data analysis methods that are both insightful and adequate for the whole range of application area demands; seamless integration of the different components of R; and the provision of interfaces to other systems (editors, databases, the web, etc.) that R users may require. Ease of use is important, but not at the expense of power, flexibility, and checks against answers that are potentially misleading.

Depending on the user's level of skill with R, there will be some tasks where another system may seem simpler to use. Note however the availability of interfaces, notably John Fox's *Rcmdr*, that give a graphical user interface (GUI) to a limited part of R. Such

interfaces will develop and improve as time progresses. They may in due course, for many users, be the preferred means of access to R. Be aware that the demand for simple tools will commonly place limitations on the tasks that can, without professional assistance, be satisfactorily undertaken.

Primarily, R is designed for scientific computing and for graphics. Among the packages that have been added are many that are not obviously statistical – for drawing and coloring maps, for map projections, for plotting data collected by balloon-borne weather instruments, for creating color palettes, for working with bitmap images, for solving sudoko puzzles, for creating magic squares, for reading and handling shapefiles, for solving ordinary differential equations, for processing various types of genomic data, and so on. Check through the list of R packages that can be found on any of the CRAN sites, and you may be surprised at what you find!

The citation for John Chambers' 1998 Association for Computing Machinery Software award stated that S has "forever altered how people analyze, visualize and manipulate data." The R project enlarges on the ideas and insights that generated the S language. We are grateful to the R Core Team, and to the creators of the various R packages, for bringing into being the R system – this marvellous tool for scientific and statistical computing, and for graphical presentation. We give a list at the end of the reference section that cites the authors and compilers of packages that have been used in this book.

Influences on the modern practice of statistics

The development of statistics has been motivated by the demands of scientists for a methodology that will extract patterns from their data. The methodology has developed in a synergy with the relevant supporting mathematical theory and, more recently, with computing. This has led to methodologies and supporting theory that are a radical departure from the methodologies of the pre-computer era.

Statistics is a young discipline. Only in the 1920s and 1930s did the modern framework of statistical theory, including ideas of hypothesis testing and estimation, begin to take shape. Different areas of statistical application have taken these ideas up in different ways, some of them starting their own separate streams of statistical tradition. See, for example, the comments in Gigerenzer *et al.* (1989) on the manner in which differences of historical development have influenced practice in different research areas.

Separation from the statistical mainstream, and an emphasis on "black-box" approaches, have contributed to a widespread exaggerated emphasis on tests of hypotheses, to a neglect of pattern, to the policy of some journal editors of publishing only those studies that show a statistically significant effect, and to an undue focus on the individual study. Anyone who joins the R community can expect to witness, and/or engage in, lively debate that addresses these and related issues. Such debate can help ensure that the demands of scientific rationality do in due course win out over influences from accidents of historical development.

New computing tools

We have drawn attention to advances in statistical computing methodology. These have made possible the development of new powerful tools for exploratory analysis of regression

data, for choosing between alternative models, for diagnostic checks, for handling non-linearity, for assessing the predictive power of models, and for graphical presentation. In addition, we have new computing tools that make it straightforward to move data between different systems, to keep a record of calculations, to retrace or adapt earlier calculations, and to edit output and graphics into a form that can be incorporated into published documents.

New traditions of data analysis have developed – data mining, machine learning, and analytics. These emphasize new types of data, new data analysis demands, new data analysis tools, and data sets that may be of unprecedented size. Textual data and image data offer interesting new challenges for data analysis. The traditional concerns of professional data analysts remain as important as ever. Size of data set is not a guarantee of quality and of relevance to issues that are under investigation. It does not guarantee that the source population has been adequately sampled, or that the results will generalize as required to the target population.

The best any analysis can do is to highlight the information in the data. No amount of statistical or computing technology can be a substitute for good design of data collection, for understanding the context in which data are to be interpreted, or for skill in the use of statistical analysis methodology. Statistical software systems are one of several components of effective data analysis.

The questions that statistical analysis is designed to answer can often be stated simply. This may encourage the layperson to believe that the answers are similarly simple. Often, they are not. Be prepared for unexpected subtleties. Effective statistical analysis requires appropriate skills, beyond those gained from taking one or two undergraduate courses in statistics. There is no good substitute for professional training in modern tools for data analysis, and experience in using those tools with a wide range of data sets. No-one should be embarrassed that they have difficulty with analyses that involve ideas that professional statisticians may take 7 or 8 years of professional training and experience to master.

Third edition changes and additions

The second edition added new material on survival analysis, random coefficient models, the handling of high-dimensional data, and extended the account of regression methods. This third edition has a more adequate account of errors in predictor variables, extends the treatment and use of random forests, and adds a brief account of generalized linear mixed models. The treatment of one-way analysis of variance, and a major part of the chapter on regression, have been rewritten.

Two areas of especially rapid advance have been graphical user interfaces (GUIs), and graphics. There are now brief introductions to two popular GUIs for R – the R Commander (*Rcmdr*) and *rattle*. The sections on graphics have been substantially extended. There is a brief account of the *latticist* and associated *playwith* GUIs for interfacing with R graphics.

Code has again been extensively revised, simplifying it wherever possible. There are changes to some graphs, and new graphs have been added.

Acknowledgments

Many different people have helped with this project. Winfried Theis (University of Dortmund, Germany) and Detlef Steuer (University of the Federal Armed Forces, Hamburg, Germany) helped with technical LaTeX issues, with a cvs archive for manuscript files, and with helpful comments. Lynne Billard (University of Georgia, USA), Murray Jorgensen (University of Waikato, NZ), and Berwin Turlach (University of Western Australia) gave highly useful comment on the manuscript. Susan Wilson (Australian National University) gave welcome encouragement. Duncan Murdoch (University of Western Ontario) helped with technical advice. Cath Lawrence (Australian National University) wrote a Python program that allowed us to extract the R code from our LaTeX files; this has now at length become an R function.

For the second edition, Brian Ripley (University of Oxford) made extensive comments on the manuscript, leading to important corrections and improvements. We are most grateful to him, and to others who have offered comments. Alan Welsh (Australian National University) has helped work through points where it has seemed difficult to get the emphasis right. Once again, Duncan Murdoch has given much useful technical advice. Others who made helpful comments and/or pointed out errors include Jeff Wood (Australian National University), Nader Tajvidi (University of Lund), Paul Murrell (University of Auckland, on Chapter 15), Graham Williams (`http://www.togaware.com`, on Chapter 1), and Yang Yang (University of Western Ontario, on Chapter 10). Comment that has contributed to this edition has come from Ray Balise (Stanford School of Medicine), Wenqing He and Lengyi Han (University of Western Ontario), Paul Murrell, Andrew Robinson (University of Melbourne, on Chapter 10), Phil Kokic (Australian National University, on Chapter 9), and Rob Hyndman (Monash University, on Chapter 9). Readers who have made relatively extensive comments include Bob Green (Queensland Health) and Zander Smith (SwissRe). Additionally, discussions on the R-help and R-devel email lists have been an important source of insight and understanding. The failings that remain are, naturally, our responsibility.

A strength of this book is the extent to which it has drawn on data from many different sources. Following the references is a list of data sources (individuals and/or organizations) that we wish to thank and acknowledge. We are grateful to those who have allowed us to use their data. At least these data will not, as often happens once data have become the basis for a published paper, gather dust in a long-forgotten folder! We are grateful, also, to the many researchers who, in their discussions with us, have helped stimulate our thinking and understanding. We apologize if there is anyone that we have inadvertently failed to acknowledge.

Diana Gillooly of Cambridge University Press, taking over from David Tranah for the second and third editions, has been a marvellous source of advice and encouragement.

Conventions

Text that is R code, or output from R, is printed in a verbatim text style. For example, in Chapter 1 we will enter data into an R object that we call `austpop`. We will use the

`plot()` function to plot these data. The names of R packages, including our own *DAAG* package, are printed in italics.

Starred exercises and sections identify more technical items that can be skipped at a first reading.

Solutions to exercises

Solutions to selected exercises, R scripts that have all the code from the book, and other supplementary materials are available via the link given at `http://www.maths.anu.edu.au/~johnm/r-book`

Content – how the chapters fit together

Chapter 1 is a brief introduction to R. Readers who are new to R should as a minimum study Section 1.1, or an equivalent, before moving on to later chapters. In later study, refer back as needed to Chapter 1, or forward to Chapter 14.

Chapters 2–4: Exploratory data analysis and review of elementary statistical ideas

Chapters 2–4 cover, at greater depth and from a more advanced perspective, topics that are common in introductory courses. Different readers will use these chapters differently, depending on their statistical preparedness.

Chapter 2 (*Styles of data analysis*) places data analysis in the wider context of the research study, commenting on some of the types of graphs that may help answer questions that are commonly of interest and that will be used throughout the remainder of the text. Subsections 2.1.7, 2.2.3 and 2.2.4 introduce terminology that will be important in later chapters.

Chapter 3 (*Statistical models*) introduces the *signal* + *noise* form of regression model. The different models for the signal component are too varied to describe in one chapter! Coverage of models for the *noise* (random component) is, relative to their use in remaining chapters, more complete.

Chapter 4 (*A review of inference concepts*) describes approaches to generalizing from data. It notes the limitations of the formal hypothesis testing methodology, arguing that a less formal approach is often adequate. It notes also that there are contexts where a Bayesian approach is essential, in order to take account of strong prior information.

Chapters 5–13: Regression and related methodology

Chapters 5–13 are designed to give a sense of the variety and scope of methods that come, broadly, under the heading of *regression*. In Chapters 5 and 6, the models are linear in the explanatory variable(s) as well as in the parameters. A wide range of issues affect the practical use of these models: influence, diagnostics, robust and resistant methods, AIC and other model comparison measures, interpretation of coefficients, variable selection, multicollinearity, and errors in x. All these issues are relevant, in one way or another, throughout later chapters. Chapters 5 and 6 provide relatively straightforward contexts in which to introduce them.

The models of Chapters 5–13 give varying combinations of answers to the questions:

1. What is the *signal* term? Is it in some sense linear? Can it be described by a simple form of mathematical equation?
2. Is the *noise* term *normal*, or are there other possibilities?
3. Are the noise terms independent between observations?
4. Is the model specified in advance? Or will it be necessary to choose the model from a potentially large number of possible models?

In Chapters 5–8, the models become increasingly general, but always with a model that is linear in the coefficients as a starting point. In Chapters 5–7, the noise terms are normal and independent between observations. The *generalized linear models* of Chapter 8 allow non-normal noise terms. These are still assumed independent.[1] Chapter 9 (*Time series models*) and Chapter 10 (*Multilevel models and repeated measures*) introduce models that allow, in their different ways, for dependence between observations. In Chapter 9 the correlation is with observations at earlier points in time, while in Chapter 10 the correlation might for example be between different students in the same class, as opposed to different students in different classes. In both types of model, the noise term is constructed from normal components – there are normality assumptions.

Chapters 6–10 allowed limited opportunity for the choice of model and/or explanatory variables. Chapter 11 (*Tree-based classification and regression*) introduces models that are suited to a *statistical learning* approach, where the model is chosen from a large portfolio of possibilities. Moreover, these models do not have any simple form of equation. Note the usual implicit assumption of independence between observations – this imposes limitations that, depending on the context, may or may not be important for practical use.

Chapter 12 (*Multivariate data exploration and discrimination*) begins with methods that may be useful for multivariate data exploration – principal components, the use of distance measures, and multi-dimensional scaling. It describes dimension reduction approaches that allow low-dimensional views of the data. Subsection 12.2 moves to discriminant methods – i.e., to regression methods in which the outcome is categorical. Subsection 12.3 identifies issues that arise when the number of variables is large relative to the number of observations. Such data is increasingly common in many different application areas.

It is sometimes possible to replace a large number of explanatory variables by one, or a small number, of *scoring* variables that capture the relevant information in the data. Chapter 13 investigates two different ways to create scores that may be used as explanatory variables in regression. In the first example, the principal component scores are used. The second uses *propensity* scores to summarize information on a number of covariates that are thought to explain group differences that are, for the purposes of the investigation, nuisance variables.

[1] Note, however, the extension to allow models with a variance that, relative to the binomial or Poisson, is inflated.

1

A brief introduction to R

This first chapter introduces readers to the basics of R. It provides the minimum of information that is needed for running the calculations that are described in later chapters. The first section may cover most of what is immediately necessary. The rest of the chapter may be used as a reference. Chapter 14 extends this material considerably.

Most of the R commands will run without change in S-PLUS.

1.1 An overview of R

1.1.1 A short R *session*

R *must be installed!*

An up-to-date version of R may be downloaded from a Comprehensive R Archive Network (CRAN) mirror site. There are links at `http://cran.r-project.org/`. Installation instructions are provided at the web site for installing R in Windows, Unix, Linux, and version 10 of the Macintosh operating system.

For most Windows users, R can be installed by clicking on the icon that appears on the desktop once the Windows setup program has been downloaded from CRAN. An installation program will then guide the user through the process. By default, an R icon will be placed on the user's desktop. The R system can be started by double-clicking on that icon.

Various contributed packages extend the capabilities of R. A number of these are a part of the standard R distribution, but a number are not. Many data sets that are mentioned in this book have been collected into our *DAAG* package that is available from CRAN sites. This and other such packages can be readily installed, from an R session, via a live internet connection. Details are given below, immediately prior to Subsection 1.1.2.

Using the console (or command line) window

The command line prompt (>) is an invitation to type commands or expressions. Once the command or expression is complete, and the **Enter** key is pressed, R evaluates and prints the result in the console window. This allows the use of R as a calculator. For example, type `2+2` and press the **Enter** key. Here is what appears on the screen:

```
> 2+2
[1] 4
>
```

The first element is labeled [1] even when, as here, there is just one element! The final > prompt indicates that R is ready for another command.

In a sense this chapter, and much of the rest of the book, is a discussion of what is possible by typing in statements at the command line. Practice in the evaluation of arithmetic expressions will help develop the needed conceptual and keyboard skills. For example:

```
> 2*3*4*5           # * denotes 'multiply'
[1] 120
> sqrt(10)          # the square root of 10
[1] 3.162278
> pi                # R knows about pi
[1] 3.141593
> 2*pi*6378         # Circumference of earth at equator (km)
                    # (radius at equator is 6378 km)
[1] 40074.16
```

Anything that follows a # on the command line is taken as comment and ignored by R.

A continuation prompt, by default +, appears following a carriage return when the command is not yet complete. For example, an interruption of the calculation of 3*4^2 by a carriage return could appear as

```
> 3*4^
+ 2
[1] 48
```

In this book we will omit both the command prompt (>) and the continuation prompt whenever command line statements are given separately from output.

Multiple commands may appear on one line, with a semicolon (;) as the separator. For example,

```
> 3*4^2; (3*4)^2
[1] 48
[1] 144
```

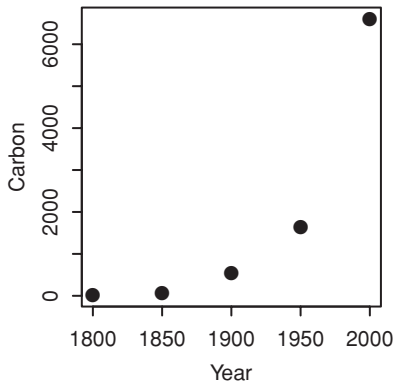
Entry of data at the command line

Figure 1.1 gives, for each of the years 1800, 1850, . . . , 2000, estimated worldwide totals of carbon emissions that resulted from fossil fuel use. To enter the columns of data from the table, and plot Carbon against Year as in Figure 1.1, proceed thus:

```
Year <- c(1800, 1850, 1900, 1950, 2000)
Carbon <- c(8, 54, 534, 1630, 6611)
## Now plot Carbon as a function of Year
plot(Carbon ~ Year, pch=16)
```

Note the following:

- The <- is a left angle bracket (<) followed by a minus sign (-). It means "the values on the right are assigned to the name on the left".



	year	carbon
1	1800	8
2	1850	54
3	1900	534
4	1950	1630
5	2000	6611

Figure 1.1 Estimated worldwide annual totals of carbon emissions from fossil fuel use, in millions of tonnes. Data are due to Marland *et al.* (2003).

- The objects `Year` and `Carbon` are vectors which were each formed by joining (concatenating) separate numbers together. Thus `c(8, 54, 534, 1630, 6611)` joined the numbers 8, 54, 534, 1630, 6611 together to form the vector `Carbon`. See Subsection 1.2.2 for further details.
- The construct `Carbon ~ Year` is a graphics formula. The `plot()` function interprets this formula to mean "Plot `Carbon` as a function of `Year`" or "Plot `Carbon` on the y-axis against `Year` on the x-axis".
- The setting `pch=16` (where `pch` is "plot character") gives a solid black dot.
- Case is significant for names of R objects or commands. Thus, `Carbon` is different from `carbon`.

This basic plot could be improved by adding more informative axis labels, changing sizes of the text and/or the plotting symbol, adding a title, and so on. See Section 1.5.

Once created, the objects `Year` and `Carbon` are stored in the *workspace*, as part of the user's working collection of R objects. The workspace lasts only until the end of a session. In order that the session can be resumed later, a copy or "image" must be kept. Upon typing `q()` to quit the session, you will be asked if you wish to save the workspace.

Collection of vectors into a data frame

The two vectors `Year` and `Carbon` created earlier are matched, element for element. It is convenient to group them together into an object that has the name *data frame*, thus:

```
> fossilfuel <- data.frame(year=Year, carbon=Carbon)
> fossilfuel         # Display the contents of the data frame.
  year carbon
1 1800      8
2 1850     54
3 1900    534
4 1950   1630
5 2000   6611
```

The vector objects `Year` and `Carbon` become, respectively, the columns `year` and `carbon` in the data frame. The vector objects `Year` and `Carbon` are then redundant, and can be removed.

```
rm(Year, Carbon)     # The rm() function removes unwanted objects
```

Figure 1.1 can now be reproduced, with a slight change in the x- and y-labels, using

```
plot(carbon ~ year, data=fossilfuel, pch=16)
```

The `data=fossilfuel` argument instructs `plot()` to start its search for each of `carbon` and `year` by looking among the columns of `fossilfuel`.

There are several ways to identify columns by name. Here, note that the second column can be referred to as `fossilfuel[, 2]`, or as `fossilfuel[, "carbon"]`, or as `fossilfuel$carbon`.

Data frames are the preferred way to organize data sets that are of modest size. For now, think of data frames as a rectangular row by column layout, where the rows are observations and the columns are variables. Section 1.3 has further discussion of data frames. Subsection 1.1.4 will demonstrate input of data from a file, into a data frame.

The R *Commander Graphical User Interface (GUI) to* R

Our discussion will usually assume use of the command line. Excellent GUI interfaces, such as the R Commander, are also available.

Data input is very convenient with the R Commander. When importing data, a window pops up offering a choice of common data formats. Data can be input from a text file, the clipboard, URL, an Excel spreadsheet, or one of several statistical package formats (SPSS, Stata, Minitab, SAS, . . .). Refer to Section 14.1 for more details.

The working directory and the contents of the workspace

Each R session has a working directory. Within a session, the workspace is the default place where R looks for files that are read from disk, or written to disk. In between sessions, it is usual for the working directory to keep a workspace copy or "image" from which the session can be restarted.

For a session that is started from a Windows icon, the initial working directory is the Start in directory that appears by right clicking on the icon and then on Properties. Users of the MacOS X GUI can change the default startup directory from within an R session by clicking on the R menu item, then on Preferences, then making the necessary change in the panel Initial working directory. On Unix or Linux sessions that are started from the command line, the working directory is the directory in which R was started. In the event of uncertainty, type `getwd()` to display the name of the working directory:

```
getwd()
```

Objects that the user creates or copies from elsewhere go into the user workspace. To list the workspace contents, type:

```
ls()
```

The only object left over from the computations above should be `fossilfuel`. There may additionally be objects that are left over from previous sessions (if any) in the same directory, and that were loaded when the session started.

Quitting R

Use the `q()` function to quit (exit) from R:

```
q()
```

There will be a message asking whether to save the workspace image. Clicking **Yes** has the effect that, before quitting, all the objects that remain in the workspace are saved in a file that has the name `.RData`. Because it is a copy or "image" of the workspace, this file is known as an *image* file. (Note that while delaying the saving of important objects until the end of the session is acceptable when working in a learning mode, it is not in general a good strategy when using R in production mode. Section 1.6 has advice on saving and backing up through the course of a session. See also the more extended comments in Subsection 14.2.2.)

Depending on the implementation, alternatives to typing `q()` may be to click on the **File** menu and then on **Exit**, or to click on the × in the top right-hand corner of the R window. (Under Linux, depending on the window manager that is used, clicking on × may exit from the program, but without asking whether to save the workshop image. Check the behavior on your installation.)

Note: The round brackets, when using `q()` to quit the session, are necessary because `q` is a function. Typing `q` on its own, without the brackets, displays the text of the function on the screen. Try it!

Installation of packages

Assuming access to a live internet connection, packages can be installed pretty much automatically. Thus, for installation of the *DAAG* package under Windows, start R and click on the Packages menu. From that menu, choose Install packages. If a mirror site has not been set earlier, this gives a pop-up menu from which a site must be chosen. Once this choice is made, a new pop-up window appears with the entire list of available R packages. Click on DAAG to select it for installation. Control-click to select additional packages. Click on OK to start downloading and installation.

For installation from the command line, enter, for example

```
install.packages("DAAG")
install.packages(c("magic", "schoolmath"), dependencies=TRUE)
```

A further possibility, convenient if packages are to be installed onto a number of local systems, is to download the files used for the installation onto a local directory or onto a CD or DVD, and install from there.

1.1.2 The uses of R

R has extensive capabilities for statistical analysis, that will be used throughout this book. These are embedded in an interactive computing environment that is suited to many different uses, some of which we now demonstrate.

R offers an extensive collection of functions and abilities

Most calculations that users may wish to perform, beyond simple command line computations, involve explicit use of functions. There are of course functions for calculating the sum (`sum()`), mean (`mean()`), range (`range()`), and length of a vector (`length()`), for sorting values into order (`sort()`), and so on. For example, the following calculates the range of the values in the vector `carbon`:

```
> range(fossilfuel$carbon)
[1]    8 6611
```

Here are examples that manipulate character strings:

```
> ## 4 cities
> fourcities <- c("Toronto", "Canberra", "New York", "London")
> ## display in alphabetical order
> sort(fourcities)
[1] "Canberra" "London"   "New York" "Toronto"
> ## Find the number of characters in "Toronto"
> nchar("Toronto")
[1] 7
>
> ## Find the number of characters in all four city names at once
> nchar(fourcities)
[1] 7 8 8 6
```

R will give numerical or graphical data summaries

The data frame `cars` (*datasets* package) has columns (variables) `speed` and `dist`. Typing `summary(cars)` gives summary information on its columns:

```
> summary(cars)
     speed           dist
 Min.   : 4.0   Min.   :  2.00
 1st Qu.:12.0   1st Qu.: 26.00
 Median :15.0   Median : 36.00
 Mean   :15.4   Mean   : 42.98
 3rd Qu.:19.0   3rd Qu.: 56.00
 Max.   :25.0   Max.   :120.00
```

Thus, the range of speeds (first column) is from 4 mph to 25 mph, while the range of distances (second column) is from 2 feet to 120 feet.

Graphical summaries, including histograms and boxplots, are discussed and demonstrated in Section 2.1. Try, for example:

```
hist(cars$speed)
```

R *is an interactive programming language*

The following calculates the Fahrenheit temperatures that correspond to Celsius temperatures 0, 10, . . . , 40:

```
> celsius <- (0:4)*10
> fahrenheit <- 9/5*celsius+32
> conversion <- data.frame(Celsius=celsius, Fahrenheit=fahrenheit)
> print(conversion)
  Celsius Fahrenheit
1       0         32
2      10         50
3      20         68
4      30         86
5      40        104
```

1.1.3 Online help

Familiarity with R's help facilities will quickly pay dividends. R's help files are comprehensive, and are frequently upgraded. Type `help(help)` or `?help` to get information on the help features of the system that is in use. To get help on, e.g., `plot()`, type:

```
?plot                # Equivalent to help(plot)
```

The functions `apropos()` and `help.search()` search for functions that perform a desired task. Examples are:

```
apropos("sort")         # Try, also, apropos ("sor")
 # List all functions where "sort" is part of the name
help.search("sort")     # Note that the argument is 'sort'
 # List functions with 'sort' in the help page title or as an alias
```

Users are encouraged to experiment with R functions, perhaps starting by using the function `example()` to run the examples on the relevant help page. Be warned however that, even for basic functions, some examples may illustrate relatively advanced uses.

Thus, to run the examples from the help page for the function `image()`, type:

```
example(image)
par(ask=FALSE)        # turn off the prompts
```

Press the return key to see each new plot. The `par(ask=FALSE)` on the second line of code stops the prompts that will otherwise continue to appear, prior to the plotting of any subsequent graph.

In learning to use a new function, it may be helpful to create a simple artificial data set, or to extract a small subset from a larger data set, and use this for experimentation. For

extensive experimentation, consider moving to a new working directory and working with copies of any user data sets and functions.

The help pages, while not an encyclopedia on statistical methodology, have very extensive useful information. They include: insightful and helpful examples, references to related functions, and references to papers and books that give the relevant theory. Some abilities will bring pleasant surprises. It can help enormously, before launching into the use of an R function, to check the relevant help page!

Wide-ranging information access and searches

The function `help.start()` opens a browser interface to help information, manuals, and helpful links. It may take practice, and time, to learn to navigate the wealth of information that is on offer.

The function `RSiteSearch()` initiates (assuming a live internet connection) a search of R manuals and help pages, and of the R-help mailing list archives, for key words or phrases. The argument `restrict` allows some limited targeting of the search. See `help(RSiteSearch)` for details.

Help in finding the right package

The CRAN Task Views can be a good place to start when looking for abilities of a particular type. The 23 Task Views that are available at the time of writing include, for example: Bayesian inference, Cluster analysis, Finance, Graphics, and Time series. Go to `http://cran.r-project.org/web/views/`

1.1.4 Input of data from a file

Code that will take data from the file `fuel.txt` that is in the working directory, entering them into the data frame `fossilfuel` in the workspace is:

```
fossilfuel <- read.table("fuel.txt", header=TRUE)
```

Note the use of `header=TRUE` to ensure that R uses the first line to get header information for the columns, usually in the form of column names.

Type `fossilfuel` at the command line prompt, and the data will be displayed almost as they appear in Figure 1.1 (the only difference is the introduction of row labels in the R output).

The function `read.table()` has the default argument `sep=""`, implying that the fields of the input file are separated by spaces and/or tabs. Other settings are sometimes required. In particular:

```
fossilfuel <- read.table("fuel.csv", header=TRUE, sep=",")
```

reads data from a file `fuel.csv` where fields are separated by commas. For other options, consult the help page for `read.table()`. See also Subsection 14.4.1.

On Microsoft Windows systems, it is immaterial whether this file is called `fuel.txt` or `Fuel.txt`. Unix file systems may, depending on the specific file system in use, treat letters that have a different case as different.

1.1.5 R packages

This chapter and Chapter 2 will make frequent use of data from the *MASS* package (Venables and Ripley, 2002) and from our own *DAAG* package. Various further packages will be used in later chapters.

The packages *base*, *stats*, *datasets*, and several other packages, are automatically attached at the beginning of a session. Other installed packages must be explicitly attached prior to use. Use `sessionInfo()` to see which packages are currently attached. To attach any further installed package, use the `library()` function. For example,

```
> library(DAAG)
Loading required package: MASS
. . . .
> sessionInfo()
R version 2.9.0 (2009-04-17)
i386-apple-darwin8.11.1
. . . .
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] DAAG_0.98   MASS_7.2-46
```

Data sets that accompany R packages

Type `data()` to get a list of data sets (mostly data frames) in all packages that are in the current search path. For information on the data sets in the *datasets* package, type

```
data(package="datasets")  # Specify 'package', not 'library'.
```

Replace `"datasets"` by the name of any other installed package, as required (type `library()` to get the names of the installed packages). In most packages, these data sets automatically become available once the package is attached. They will be brought into the workspace when and if required. (A few packages do not implement the *lazy data* mechanism. Explicit use of a command of the form `data(airquality)` is then necessary, bringing the data object into the user's workspace.)

1.1.6 Further steps in learning R

Readers who have followed the discussion thus far and worked through the examples may at this point have learned enough to start on Chapter 2, referring as necessary to later sections of this chapter, to R's help pages, and to Chapter 14. The remaining sections of this chapter cover the following topics:

- Numeric, character, logical, and complex vectors (Section 1.2).
- Factors (Subsection 1.2.7).
- Data frames and matrices (Section 1.3).
- Functions for calculating data summaries (Section 1.4).
- Graphics and lattice graphics (Sections 1.5 and 15.5).

1.2 Vectors, factors, and univariate time series

Vectors, factors, and univariate time series are all univariate objects that can be included as columns in a data frame. The vector modes that will be noted here (there are others) are "numeric", "logical", and "character".

1.2.1 Vectors

Examples of vectors are

```
> c(2, 3, 5, 2, 7, 1)
[1] 2 3 5 2 7 1

> c(T, F, F, F, T, T, F)
[1]  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE

> c("Canberra", "Sydney", "Canberra", "Sydney")
[1] "Canberra" "Sydney"   "Canberra" "Sydney"
```

The first of these is numeric, the second is logical, and the third is a character. The global variables `F (=FALSE)` and `T (=TRUE)` can be a convenient shorthand when logical values are entered.

1.2.2 Concatenation – joining vector objects

The function `c()`, used in Subsection 1.1.1 to join numbers together to form a vector, is more widely useful. It may be used to concatenate any combination of vectors and vector elements. In the following, we form numeric vectors `x` and `y`, that we then concatenate to form a vector `z`:

```
> x <- c(2, 3, 5, 2, 7, 1)  # x then holds values 2, 3, 5, 2, 7, 1
> x
[1] 2 3 5 2 7 1
> y <- c(10, 15, 12)
> y
[1] 10 15 12
> z <- c(x, y)
> z
[1] 2 3 5 2 7 1 10 15 12
```

1.2.3 The use of relational operators to compare vector elements

Relational operators are <, <=, >, >=, ==, and !=. For example, consider the `carbon` and `year` columns of the data frame `fossilfuel`. For example:

```
> x <- c(3, 11, 8, 15, 12)
> x > 8
[1] FALSE  TRUE FALSE  TRUE  TRUE
> x != 8
[1]  TRUE  TRUE FALSE  TRUE  TRUE
```

For further information on relational operators consult `help(Comparison)`, `help(Logic)`, and `help(Syntax)`.

1.2.4 The use of square brackets to extract subsets of vectors

Note three common ways to extract elements of vectors. In each case, the identifying information (in the simplest case, a vector of subscript indices) is enclosed in square brackets.

1. Specify the indices of the elements that are to be extracted, e.g.,
   ```
   > x <- c(3, 11, 8, 15, 12)
   > x[c(2,4)]                 # Elements in positions 2
   [1] 11 15                   # and 4 only
   ```
2. Use negative subscripts to omit the elements in nominated subscript positions (take care not to mix positive and negative subscripts):
   ```
   > x[-c(2,3)]          # Remove the elements in positions 2 and 3
   [1] 3 15 12
   ```
3. Specify a vector of logical values. This extracts elements for which the logical value is `TRUE`. The following extracts values of `x` that are greater than 10:
   ```
   > x > 10
   [1] FALSE TRUE FALSE TRUE TRUE
   > x[x > 10]
   [1] 11 15 12
   ```

Elements of vectors can be given names. Elements can then be extracted by name:

```
> heights <- c(Andreas=178, John=185, Jeff=183)
> heights[c("John","Jeff")]
John Jeff
 185  183
```

1.2.5 Patterned data

Use, for example, `5:15` to generate all integers in a range, here between 5 and 15 inclusive:

```
> 5:15
[1]  5  6  7  8  9 10 11 12 13 14 15
```

Conversely, `15:5` will generate the sequence in the reverse order.

The function `seq()` allows a wider range of possibilities. For example:

```
> seq(from=5, to=22, by=3)  # The first value is 5.  The final
                            # value is <= 22
[1] 5 8 11 14 17 20
## The above can be abbreviated to seq(5, 22, 3)
```

To repeat the sequence (2, 3, 5) four times over, enter

```
> rep(c(2,3,5), 4)
[1] 2 3 5 2 3 5 2 3 5 2 3 5
```

Patterned character vectors are also possible:

```
> c(rep("female", 3), rep("male", 2))
[1] "female" "female" "female" "male" "male"
```

1.2.6 Missing values

The missing value symbol is `NA`. As an example, consider the column `branch` of the data set `rainforest`:

```
> library(DAAG)
> nbranch <- subset(rainforest, species=="Acacia mabellae")$branch
> nbranch            # Number of small branches (2cm or less)
 [1] NA 35 41 50 NA NA NA NA NA  4 30 13 10 17 46 92
```

Any arithmetic expression that involves an `NA` generates `NA` as its result. Functions such as `mean()` allow the argument `na.rm=TRUE`, so that `NA`s are omitted before proceeding with the calculation. For example:

```
> mean(nbranch)
[1] NA
> mean(nbranch, na.rm=TRUE)
[1] 33.8
```

Other functions that behave similarly are `sum()`, `median()`, `range()`, and `sd`.

Arithmetic and logical expressions in which `NA`s appear return `NA`, thus:

```
> NA == 35
[1] NA
```

The unknown value might just possibly equal 35. This is a matter of strict logic, not probability. Thus, the result is `NA`.

To replace all `NA`s by `-999` (in most circumstances a bad idea) use the function `is.na()`, thus:

```
> ## Replace all NAs by -999
> nbranch[is.na(nbranch)] <- -999
> nbranch
 [1] -999   35   41   50 -999 -999 -999 -999 -999    4   30   13
[13]   10   17   46   92
> ## There is now no protection against use of the -999 values as
```

```
> ## if they were legitimate numeric values
> mean(nbranch)
[1] -353.5           # Illegitimate calculation
```

Using a code such as `-999` for missing values requires continual watchfulness to ensure that it is never treated as a legitimate numeric value.

Missing values are discussed further in Subsection 1.4.6 and Section 14.7. For vectors of mode numeric, other legal values that may require special attention are `NaN` (not a number; e.g., 0/0), `Inf` (e.g., 1/0), and `-Inf`.

1.2.7 Factors

A factor is stored internally as a numeric vector with values $1, 2, 3, \ldots, k$. The value k is the number of levels. The levels are character strings.

Consider a survey that has data on 691 females and 692 males. If the first 691 are females and the next 692 males, we can create a vector of strings that holds the values, then turning this vector into a factor, thus:

```
> ## Create character vector
> gender <- c(rep("female",691), rep("male",692))
> levels(gender)     # For a character vector, this returns NULL
NULL
> ## From character vector, create factor
> gender <- factor(gender)
> levels(gender)
[1] "female" "male"
```

Internally, the factor `gender` is stored as 691 `1`s, followed by 692 `2`s. It has stored with it a table that holds the information

1	female
2	male

In most contexts that seem to demand a character string, the 1 is translated into `female` and the 2 into `male`. The values `female` and `male` are the levels of the factor. By default, the levels are in sorted order for the data type from which the factor was formed, so that `female` precedes `male`. Hence:

```
> levels(gender)
[1] "female" "male"
```

Note that if `gender` had been an ordinary character vector, the outcome of the above `levels` command would have been `NULL`.

The order of the factor levels is used, in graphs and tables, to determine the order in which the levels will appear. To cause `male` to come before `female`, use

```
gender <- factor(gender, levels=c("male", "female"))
```

This syntax is available both when the factor is first created, and later to change the order in an existing factor. Take care that the level names are correctly spelled. For example,

specifying `"Male"` in place of `"male"` in the `levels` argument will cause all values that were `"male"` to be coded as missing.

Note finally the function `ordered()`, which generates factors whose values can be compared using the relational operators `<`, `<=`, `>`, `>=`, `==`, and `!=`. Ordered factors are appropriate for use with ordered categorical data. See Section 14.6 for further details.

1.2.8 Time series

The following are the numbers of workers (in 1000s) in the Canadian prairies for each month from January 1995 through December 1996:[1]

```
numjobs <- c(982,981,984,982,981,983,983,983,983,979,973,979,
             974,981,985,987,986,980,983,983,988,994,990,999)
```

The function `ts()` converts numeric vectors into time series objects. Frequently used arguments of `ts()` are `start`, `frequency`, and `end`. The following turns `numjobs` into a time series, which can then be plotted:

```
numjobs <- ts(numjobs, start=1995, frequency = 12)
plot(numjobs)
```

Use the function `window()` to extract a subset of the time series. For example, the following extracts the last quarter of 1995 and the first few months of 1996:

```
first15 <- window(numjobs, start=1995.75, end=1996.25)
```

Multivariate time series can also be handled. See Subsections 2.1.5 and 14.9.7.

1.3 Data frames and matrices

Data frames are fundamental to the use of the R modeling and graphics functions. A data frame is a more general object than a matrix, in the sense that different columns may have different modes. All elements of any column must, however, have the same mode, i.e., all numeric, or all factor, or all character, or all logical.

Included in the *DAAG* package is `Cars93.summary`, created from the `Cars93` data set in the *MASS* package. Its contents are:

```
> Cars93.summary
        Min.passengers Max.passengers No.of.cars abbrev
Compact              4              6         16      C
Large                6              6         11      L
Midsize              4              6         22      M
Small                4              5         21     Sm
Sporty               2              4         14     Sp
Van                  7              8          9      V
```

[1] `## Alternatively, obtain from data frame jobs (DAAG)`
`library(DAAG)`
`numjobs <- jobs$Prairies`

The first three columns are numeric, and the fourth is a factor. Use the function `class()` to check this, e.g., enter `class(Cars93.summary$abbrev)`. (The classification of objects into classes is discussed in Subsection 1.4.2.)

On most systems, use of `edit()` allows access to a spreadsheet-like display of a data frame or of a vector, where entries can be edited or new data added. For example,

```
Cars93.summary <- edit(Cars93.summary)
```

To close the spreadsheet, click on the **File** menu and then on **Close**. On Linux systems, click on **Quit** to exit.

Displaying the first few, or last few, rows of a data frame

When used with a data frame (other possible arguments include vectors and functions), the `head()` function displays the first lines of a data frame, while `tail()` displays the last lines. For example,

```
> head(Cars93.summary, n=3)  # Display the first 3 rows
                               (the default is 6)
        Min.passengers Max.passengers No.of.cars abbrev
Compact              4              6         16      C
Large                6              6         11      L
Midsize              4              6         22      M
>  # . . .
```

Note also the functions `str()` and `summary()`, both of which can be used to get summary information on data frames, different in the two cases.

Column and row names

The function `rownames()` extracts the names of rows, while `colnames()` extracts column names, thus:

```
rownames(Cars93.summary)  # Extract row names
colnames(Cars93.summary)  # Extract column names
```

For use with data frames `row.names()` is an alternative to `rownames()`, while `names()` is an alternative to `colnames()`.

The functions `names()` (or `colnames()`) and `rownames()` can also be used to assign new names. For example:

```
names(Cars93.summary)[3] <- "numCars"
names(Cars93.summary) <- c("minPass","maxPass","numCars","code")
```

Subsets of data frames

Data frames are indexed by row and column number. Thus to extract the element in the 4th row and 2nd column, specify `Cars93.summary[4, 2]`. Here are additional examples:

```
Cars93.summary[1:3, 2:3]   # Rows 1-3 and columns 2-3
Cars93.summary[, 2:3]      # Columns 2-3 (all rows)
Cars93.summary[, c("No.of.cars", "abbrev")]   # Cols 2-3, by name
Cars93.summary[, -c(2,3)]  # omit columns 2 and 3
```

The `subset()` function offers an alternative way to extract rows and columns. For example, the following extracts the first two rows:

```
subset(Cars93.summary,
       subset=c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE))
```

Use the argument `select` to specify a subset of columns. See `help(subset)` for details.

Use of the subscript notation to extract a column, as in `Cars93.summary[, 1]`, returns a vector. By contrast, extraction of the raw `Cars93.summary[1, ]` returns a data frame, necessary because this allows different elements (columns) to retain their existing classes. Note also

- Use of `unlist(Cars93.summary[1, ])` returns a vector, but with the side-effect that the factor value in the final column is coerced to numeric. Such side-effects are usually undesirable, with a result that may be meaningless.
- Avoid `Cars93.summary[4]`, at least until the subtleties of its use are understood. See Subsection 14.9.1. If used where `Cars93.summary[, 4]` was intended, the calculation may fail or give an erroneous result.

Data frames are a specialized type of list

A list is an arbitrary collection of R objects. Here is a simple example, containing two character vectors of differing lengths and a numeric vector:

```
> ## Cities with more than 2.5 million inhabitants
> USACanada <- list(USACities=c("NY", "LA", "Chicago"),
+                   CanadaCities=c("Toronto", "Montreal"),
+                   millionsPop=c(USA=305.9, Canada=31.6))
>
> USACanada
$USACities
[1] "NY"      "LA"      "Chicago"

$CanadaCities
[1] "Toronto"  "Montreal"

$millionsPop
   USA Canada
 305.9   31.6
```

Many of R's modeling functions return their output as a list. Lists can be joined using the function `c()`; in this and in several other respects they are "vectors". Important aspects of the syntax for working with data frames apply also to lists. Obviously, however,

notions of "row" and "column" have no relevance to lists. See Subsection 14.9.1 for further commentary.

1.3.1 Accessing the columns of data frames – `with()` and `attach()`

In repeated computations with the same data frame, it is tiresome to keep repeating the name of the data frame. The function `with()` is often helpful in this connection. Thus, an alternative to `c(mean(cfseal$weight), median(cfseal$weight))` is:

```
> ## cfseal (DAAG) has data on Cape Fur seals
> with(cfseal, c(mean(weight), median(weight)))
[1] 54.8 46.2
```

Curly brackets (braces) can be used to extend the scope of `with()` over several lines of code:

```
> with(pair65,        # stretch of rubber bands, from DAAG
+   {lenchange <- heated-ambient
+    c(mean(lenchange), median(lenchange))
+ })
[1] 6.33 6.00
```

An alternative is `attach()`. Once a data frame has been attached, its columns can be referred to by name, without further need to give the name of the data frame. For example:

```
> year
Error: Object "year" not found
> attach(fossilfuel)  # Attach data frame fossilfuel
> year
[1] 1800 1850 1900 1950 2000
> detach(fossilfuel)  # Detach data frame
```

Be sure to detach data frames that are no longer in use. If more than one data frame is attached that has the column `year`, there is obvious scope for confusion.

What happens if there is an object `year` in the workspace? References to `year` will then take the object that is in the workspace, ignoring the column `year` in the attached data frame. By contrast, use of `with()` ensures that the column, if present, is from the specified data frame.

The attaching of a data frame extends the search list, which is the list of "databases" where R looks for objects. See Section 14.2 for more details on this and other uses of `attach()`.

1.3.2 Aggregation, stacking, and unstacking

The `aggregate()` function yields a data frame that has the mean or value of another specified function for each combination of factor levels. As an example, consider the `chickwts` data frame which contains observations on the weights of 71 six-week-old chicks who have been fed one of six kinds of feed. The columns of `chickwts` are named `weight` and `feed`. To find the average weights for the different feed groups, type

```
> chickwtAvs <- with(chickwts,
+                    aggregate(weight, by=list(feed), mean))
> names(chickwtAvs) <- c("Feed Group", "Mean Weight")
> chickwtAvs
  Feed Group Mean Weight
1     casein    323.5833
2  horsebean    160.2000
3    linseed    218.7500
. . . .
```

See Subsection 14.9.5 for more information on the `aggregate()` function.

For stacking columns of a data frame, i.e., placing successive columns one under the other, the function `stack()` is available. The following use of `stack()`, with data from the data frame `jobs`, will be required for the use of these data in Subsection 2.1.5.

```
> library(DAAG)
> head(jobs,3)
    BC Alberta Prairies Ontario Quebec Atlantic     Date
1 1752    1366      982    5239   3196      947 95.00000
2 1737    1369      981    5233   3205      946 95.08333
3 1765    1380      984    5212   3191      954 95.16667
>   # . . .
> Jobs <- stack(jobs, select = 1:6)
>   # stack() concatenates selected data frame columns into a
>   # single column named "values", & adds a factor named "ind"
>   # that has the names of the concatenated columns as levels.
> head(Jobs,3)
  values ind
1   1752  BC
2   1737  BC
3   1765  BC
>   # . . .
```

For a further example, see Exercise 19.

The `unstack()` function reverses the stacking operation. For example, `unstack(Jobs)` (or more generally, `unstack(Jobs, values ~ ind)`) recovers the original data frame.

1.3.3 Data frames and matrices*

The numeric values in the data frame `fossilfuel` might alternatively be stored in a matrix with the same dimensions, i.e., 5 rows × 2 columns. The following enters these same data as a matrix:

```
fossilfuelmat <- matrix(c(1800, 1850, 1900, 1950, 2000,
                          8, 54, 534, 1630, 6611), nrow=5)
colnames(fossilfuel) <- c("year", "carbon")
```

Another possibility is the use of the function `cbind()` to combine two or more vectors of the same length and type together into a matrix, thus:

```
fossilfuelmat <- cbind(year=c(1800, 1850, 1900, 1950, 2000),
                       carbon=c(8, 54, 534, 1630, 6611))
```

More generally, any data frame where all columns hold data that is all of the same type, i.e., all numeric or all character or all logical, can alternatively be stored as a matrix. Storage of numeric data in matrix rather than data frame format can speed up some mathematical and other manipulations when the number of elements is large, e.g., of the order of several hundreds of thousands. For further details, see Section 14.8.

Note that:

- Matrix elements are stored in column order in one long vector, i.e., columns are stacked one above the other, with the first column first. Section 14.8 describes how to change between a matrix with m rows and n columns, and a vector of length mn.
- The extraction of submatrices has the same syntax as for data frames. Thus, `fossilfuelmat[2:3,]` extracts rows 2 and 3 of the matrix `fossilfuelmat`. (Be careful not to omit the comma, causing the matrix to be treated as one long vector.)
- The `names()` function returns `NULL` when the argument is a matrix. Note however `rownames()` and `colnames()`, which can be used either with data frames or matrices.
- The function `nrow()` (e.g. `nrow(fossilfuel)` or `nrow(fossilfuelmat)`) returns the number of rows, while `ncol()` returns the number of columns.

1.4 Functions, operators, and loops

Functions are integral to the use of the R language. User-written functions are used in exactly the same way as built-in functions. Examples will appear from time to time through the book. An incidental advantage of putting code into functions is that the workspace is not then cluttered with objects that are local to the function.

1.4.1 Common useful built-in functions

```
all()        # returns TRUE if all values are TRUE
any()        # returns TRUE if any values are TRUE
args()       # information on the arguments to a function
cat()        # prints multiple objects, one after the other
cumprod()    # cumulative product
cumsum()     # cumulative sum
diff()       # form vector of first differences
             # N. B. diff(x) has one less element than x
history()    # displays previous commands used
is.factor()  # returns TRUE if the argument is a factor
is.na()      # returns TRUE if the argument is an NA
             # NB also is.logical(), is.matrix(), etc.
length()     # number of elements in a vector or of a list
ls()         # list names of objects in the workspace
```

```
mean()       # mean of the elements of a vector
median()     # median of the elements of a vector
order()      # x[order(x)] sorts x (by default, NAs are last)
print()      # prints a single R object
range()      # minimum and maximum value elements of vector
sort()       # sort elements into order, by default omitting NAs
rev()        # reverse the order of vector elements
str()        # information on an R object
unique()     # form the vector of distinct values
which()      # locates 'TRUE' indices of logical vectors
which.max()  # locates (first) maximum of a numeric vector
which.min()  # locates (first) minimum of a numeric vector
with()       # do computation using columns of specified data frame
```

Be sure to check, where this is relevant, the handling of missing values. In case of doubt, consult the relevant help page. Refer back to Subsection 1.2.6.

The `print()` *function*

This is perhaps R's most pervasive function. It is invoked whenever an object, or the result of a computation, has its value returned to the command line. For example:

```
> x <- 2     # Assign to x the value 2; nothing is printed
> x          # Equivalent to print(x)
[1] 2
> x*5        # Equivalent to print(x*5)
[1] 10
```

It can be convenient to make an assignment and print the value. For this, enclose the assignment in parentheses, i.e., in round brackets:

```
> (x <- 2)  # Equivalent to: x <- 2; print(x)
[1] 2
```

Calculations in parallel across all elements of a vector

Subsection 1.1.2 gave an example in which arithmetic was carried out in parallel across all elements of a vector. Many of R's functions likewise operate in parallel on all elements of arrays, matrices, and data frames.

Data summary functions – `table()` *and* `sapply()`

Data summary functions that create tables of counts are:

```
table()              # Form a table of counts
xtabs()              # Form a table of totals
```

For example, the `tinting` data frame in *DAAG* contains columns specifying the sex (`sex`, levels are `f` and `m`) and age group (`agegp`, levels are `younger` and `older`) of

participants in a study. The `table()` function can be used to count up the numbers of observations in each sex–age group combination:

```
> library(DAAG)       # tinting is from DAAG
> table(Sex=tinting$sex, AgeGroup=tinting$agegp)
   AgeGroup
Sex younger older
  f      63    28
  m      28    63
```

By default, `table()` ignores `NA`s. For further details of `table()`, and for an example of the use of `xtabs()`, see Subsection 2.2.1.

The function `sapply()` applies a function to each column of a data frame, or to each element of a list. The following demonstrates its use to give the range, for all columns of the data frame `jobs` (*DAAG*):

```
> sapply(jobs[, -7], range)
       BC Alberta Prairies Ontario Quebec Atlantic
[1,] 1737    1366      973    5212   3167      941
[2,] 1840    1436      999    5360   3257      968
```

Utility functions

Type `ls()` (or `objects()`) to see the names of all objects in the workspace. One can restrict the names to those with a defined pattern, e.g., starting with the letter `p`:[2]

```
ls(pattern="p")     # List object names that include the letter "p"
ls(pattern="^p")    # List object names that start with "p"
```

Type `help(ls)`, `help(grep)`, and `help(glob2rx)` for more details.

Various packages will add hidden files, whose first character is a full stop, to the workspace. To see these files, type `ls(all=TRUE)`. To clear the workspace completely, type `rm(list=ls(all=TRUE))`.

By default, the function `dir()` lists the contents of the working directory. See Subsection 14.2.3 for further details on this and other utility functions.

1.4.2 Generic functions, and the class of an object

The printing of a data frame requires steps that are different from those for the printing of a vector of numbers. Yet, in R, the same `print()` function handles both tasks. In order to make this possible, all objects in R have a *class*, which can be used to decide how the printing should be handled.

The `print()` function does not itself attend to the printing. Instead, if `print()` is called with a factor argument, `print.factor()` is used. For a data frame `print.data.frame()` is used, and so on. Section 14.10 gives further details.

[2] More generally, the pattern-matching conventions are the same as for `grep()`, which is modeled on the Unix `grep` command.

For objects (such as numeric vectors) that do not otherwise have a print method, `print.default()` handles the printing.

For simple objects such as numbers and text strings, the class is determined informally. More complex objects such as data frames carry a tag (an *attribute*) that specifies the class. In either case, the function `class()` can be used to determine the class. See Section 14.10 for further details.

1.4.3 User-written functions

Here is a function that returns the mean and standard deviation of a vector of numbers:

```
mean.and.sd <- function(x){
    av <- mean(x)
    sdev <- sd(x)
    c(mean=av, SD=sdev)
    }
```

Having constructed the function, we can apply it to a numeric vector, as in the following:

```
> distance <- c(148,182,173,166,109,141,166)
> mean.and.sd(distance)
  mean     SD
155.00  24.68
```

The variables `av` and `sdev` are local to the function. They cannot be accessed outside of the internal function environment.

Many functions have default arguments which make it possible to run them without specifying any data. We can modify the above function to have the default argument `x = rnorm(10)`. This generates a vector of 10 random numbers to which the function is then applied.

```
mean.and.sd <- function(x = rnorm(10)){
    av <- mean(x)
    sdev <- sd(x)
    c(mean=av, SD=sdev)
    }
```

Here is the result of one execution of the modified function:

```
> mean.and.sd()
     mean        SD
0.6576272 0.8595572
```

The structure of functions

The function `mean.and.sd()` has the following structure:

```
function name                    argument(s)
mean.and.sd <- function(x=rnorm(10))
       {
function av <- mean(x)
 body  sdev <- sd(x)
return c(av = av, sd = sdev)
value  }
```

If the function body consists of just one statement that gives the return value, the curly braces ({ }) are unnecessary. The return value, which must be a single object, is given by the final statement of the function body. In the example above, the return value was the vector consisting of the two named elements `mean` and `sdev`. For returning several objects that are of different types, join them into a list.[3]

1.4.4 `if` Statements

Subsection 1.2.4 introduced the use of relational operators to create particular subsets of a given vector.

The R system also has the flow control capabilities of traditional programming languages, including `if` statements. The `if` function tests the truth of a given statement; if the statement is true, the succeeding expression is evaluated. An `else` can be added to provide an alternative expression to be evaluated in the case where the given statement is false. For example, the following checks whether the mean for the carbon emissions exceeds the median:

```
Carbon <- fossilfuel$carbon
> if (mean(Carbon) > median(Carbon)) print("Mean > Median") else
+    print("Median <= Mean")
[1] "Mean > Median"
```

Here is another example:

```
> dist <- c(148, 182, 173, 166, 109, 141, 166)
> dist.sort <- if (dist[1] < 150)
+                  sort(dist, decreasing=TRUE) else sort(dist)
> dist.sort
[1] 182 173 166 166 148 141 109
```

1.4.5 Selection and matching

A highly useful operator is `%in%`, used for testing set membership. For example:

```
> x <- rep(1:5, rep(3,5))
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
> x[x %in% c(2,4)]
[1] 2 2 2 4 4 4
```

[3] `## Thus, to return the mean, SD and name of the input vector`
`## replace c(mean=av, SD=sdev) by`
`list(mean=av, SD=sdev, dataset = deparse(substitute(x)))`

We have picked out those elements of x that are either 2 or 4. To find which elements of x are 2s, which 4s, and which are neither, use match(). Thus:

```
> match(x, c(2,4), nomatch=0)
 [1] 0 0 0 1 1 1 0 0 0 2 2 2 0 0 0
```

The nomatch argument specifies the symbol to be used for elements that do not match. Specifying nomatch=0 is often preferable to the default, which is NA.

1.4.6 Functions for working with missing values

Recall the use of the function is.na(), discussed in Subsection 1.2.6, to identify NAs. Testing for equality with NAs does not give useful information.

Identification of rows that include missing values

Many of the modeling functions will fail unless action is taken to handle missing values. Two functions that are useful for identifying or handling missing values are complete.cases() and na.omit(). Applying the complete.cases() function to a data frame returns a logical vector whose length is the number of rows and whose TRUE values correspond to rows which do not contain any missing values. Thus, the following identifies rows that hold one or more missing values:

```
> ## Which rows have missing values: data frame science (DAAG)
> science[!complete.cases(science), ]
     State PrivPub school class  sex like Class
671    ACT  public     19     1 <NA>    5  19.1
672    ACT  public     19     1 <NA>    5  19.1
```

The function na.omit() omits any rows that contain missing values. For example,

```
> dim(science)
[1] 1385    7
> Science <- na.omit(science)
> dim(Science)
[1] 1383    7
```

It should be noted that there may be better alternatives to omitting missing values. There is an extensive discussion in Harrell (2001, pp. 43–51). Often, the preferred approach is to estimate the values that are missing as part of any statistical analysis. It is important to consider why values are missing – is the probability of finding a missing value independent of the values of variables that appear in the analysis?

1.4.7 Looping*

A simple example of a for loop is[4]

[4] Other looping constructs are

```
repeat <expression>          # Place break somewhere inside
while (x > 0) <expression>   # Or (x < 0), or etc.
```

Here <expression> is an R statement, or a sequence of statements that are enclosed within braces.

```
> for (i in 1:3) print(i)
[1] 1
[1] 2
[1] 3
```

Here is a way to estimate the increase in population for each of the Australian states and territories between 1917 and 1997, relative to 1917, using the data frame `austpop`. Columns are 1: census year (by decade from 1917 through 1997); 2–9: the state and territory populations that are of interest here; and 10: the national population.

```
> ## Relative population increase in Australian states: 1917-1997
> ## Data frame austpop (DAAG)
> relGrowth <- numeric(8)   # numeric(8) creates a numeric vector
>                           # with 8 elements, all set equal to 0
> for (j in seq(from=2, to=9)) {
+     relGrowth[j-1] <- (austpop[9, j]-austpop[1, j])/
+     austpop[1, j]}
> names(relGrowth) <- names(austpop[c(-1,-10)])
>   # We have used names() to name the elements of relGrowth
> relGrowth          # Output is with options(digits=3)
   NSW    Vic    Qld     SA     WA    Tas     NT    ACT
  2.30   2.27   3.98   2.36   4.88   1.46  36.40 102.33
```

Often, there is a better alternative to the use of a loop. See Subsection 14.5.3.

1.5 Graphics in R

Later chapters will make extensive use both of base graphics (using `plot()`, etc.) and of the more stylized graphs provided by *lattice* graphics. This section is a brief introduction to `plot()` and allied functions that are included in R's base graphics. Subsection 1.5.8 is a brief introduction to the more stylized graphical functions in the *lattice* package. Note also the carefully structured abilities of the *ggplot2* package, described in Section 15.2.

Base graphics are provided by the *graphics* package that is automatically attached at startup. It includes the function `plot()` for creating scatterplots, and the functions `points()`, `lines()`, `text()`, `mtext()`, and `axis()` that add to existing plots.

There is a wide range of other functions. To see some of the possibilities, enter

```
demo(graphics)
```

Press the **Enter** key to move to each new graph.

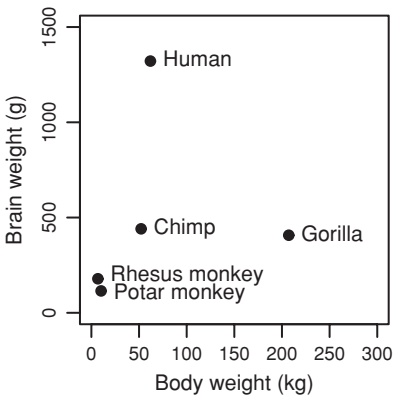
1.5.1 The function `plot( )` and allied functions

The data frame `primates` gives `Bodywt` and `Brainwt`, for five primate species. A basic plot of `Brainwt` against `Bodywt` can be obtained thus:

```
plot(Brainwt ~ Bodywt, data=primates)  # plot(y ~ x) syntax
```

or

```
with(primates, plot(Bodywt, Brainwt))    # plot(x, y) syntax
```



	Bodywt	Brainwt
Potar monkey	10.0	115
Gorilla	207.0	406
Human	62.0	1320
Rhesus monkey	6.8	179
Chimp	52.2	440

Figure 1.2 Brain weight (g) versus body weight (kg). Data are from the `primates` data frame.

There are many possible refinements. A number of the most important are illustrated in Plate 1, which supplements the discussion that follows.

Adding points, lines, text, and axis annotation

Figure 1.2 puts labels on the points.

```
## Place labels on points
plot(Brainwt ~ Bodywt, xlim=c(0, 300), data=primates)
  # Specify xlim so that there is room for the labels
with(primates,
     text(Brainwt ~ Bodywt, labels=row.names(primates), pos=4))
  # pos=4 places text to the right of the points. Other
  # possibilities are: 1: below; 2: to the left; 3: above
```

Figure 1.2 has two further refinements. The *y*-axis limits were extended slightly. Small vertical offsets were incorporated that raised the label `Rhesus monkey` and lowered the label `Potar monkey`, avoiding overlap.[5]

Use `points()` to add points to a plot. Use `lines()` to add lines. Actually these are aliases, differing only in the default for the parameter `type`; `points()` has `type = "p"`, while `lines()` has `type = "l"`.

The function `mtext(text, side, line, ...)` adds text in the margin of the current plot. The sides are numbered 1 (*x*-axis), 2 (*y*-axis), 3 (top), and 4 (right vertical axis). By default, `adj=0.5`, which centers the text at the axis midpoint. Specify `adj=0` to position the left extreme of the text at the left margin, and `adj=1` to position its right extreme at the right margin.

The `axis()` function gives fine control over axis ticks and labels. To use for the *x*-axis, plot the initial graph with `xaxt="n"`. Then call `axis()` with the argument `side=1`, and with other arguments as required. See `help(axis)` for details.

[5]
```
## Plot Brainwt vs Bodywt, primates data frame
plot(Brainwt ~ Bodywt, xlim=c(0, 300), ylim=c(0,1500), data=primates)
yoff <- c(-.125,0,0,.125,0)*par()$cxy[2]
with(primates, text(x=Bodywt, y=Brainwt+yoff, labels=row.names(primates), pos=4))
```

Fine control – parameter settings

Here are some of the parameters that commonly require attention:

- Plotting symbols: `pch` (choice of symbol); `cex` ("character expansion"); `col` (color). Thus `par(cex=1.2)` increases the plot symbol size 20% above the default.
- Lines: `lty` (line type); `lwd` (line width); `col` (color).
- Axis limits: `xlim`; `ylim`. (Assuming `xaxs="r"`, x-axis limits are by default extended by 4% relative to the data limits. Specify `xaxs="i"` to make the default an exact fit to the data limits. For the y-axis, replace `xaxs` by `yaxs`.)
- Axis annotation and labels: `cex.axis` (character expansion for axis annotation, independently of `cex`); `cex.labels` (size of the axis labels); `mgp` (margin line for the axis title, axis labels, and axis line; default is `mgp=c(3, 1, 0)`).
- Graph margins: `mar` (inner margins, clockwise from the bottom; the out-of-the-box default is `mar=c(5.1, 4.1, 4.1, 2.1)`, in lines out from the axis); `oma` (outer margins, relevant when there are multiple graphs on the one graphics page).
- Plot shape: `pty="s"` gives a square plot (must be set using `par()`).
- Multiple graphs on the one graphics page: Specify `par(mfrow=c(m,n))` to get an m rows by n columns layout of graphs on a page. The 1 by 4 layout of plots in Figure 2.1 of Chapter 2 was obtained using `par(mfrow=c(1,4))`.

Type `help(par)` to get a (very extensive) complete list. Figure 15.1 and Plate 1 demonstrate some of the possibilities.

In most (not all) instances, the change can be made either in a call to a plotting function (e.g., `plot()`, `points()`), or using `par()`. If made in a call to a plotting function, the change applies only to that call. If made using `par()`, changes remain in place until changed again, or until a new device is opened.

It can be helpful to store the existing settings, so that they can be restored later. For this, specify, for example:

```
oldpar <- par(cex=1.25)
# Use par(oldpar) to restore previous settings
```

1.5.2 The use of color

The default palette, which can be changed, has eight colors including "white". These are a small selection from the built-in colors. The function `colors()` returns the 657 names of the built-in colors, some of them aliases for the same color.

In the following, points are in the colors of the current palette. These are recycled as necessary.

```
theta <- (1:50)*0.92
plot(theta, sin(theta), col=1:50, pch=16, cex=4)
points(theta, cos(theta), col=51:100, pch=15, cex=4)
palette()            # Names of the colors in the current palette
```

The following repeats the plot, but now using the function `colors()` to supply two sets of 50 (mostly) different colors:

```
plot(theta, sin(theta), col=colors()[1:50], pch=16, cex=4)
points(theta, cos(theta), col=colors()[51:100], pch=15, cex=4)
```

Where data from a two-way layout are presented on the one panel, different symbols can be used for the different levels of one of the classifying factors, with different colors used for the different levels of the other classifying factor. Care may be required in the choice of colors, so that the colors show with clarity the distinctions that are required, and do not clash. Section 15.2 has further discussion of color palettes.

1.5.3 The importance of aspect ratio

Attention to aspect ratio is often crucial for creating graphs that reveal important features of the data. The following simple graphs highlight this point:

```
## Plot sin(theta) vs theta, at regularly spaced values of theta
## sin() expects angles to be in radians
 # multiply angles in degrees by pi/180 to get radians
plot((0:20)*pi/10, sin((0:20)*pi/10))
plot((1:50)*0.92, sin((1:50)*0.92))
```

Readers might show the second of the graphs that now follows to their friends, asking them to identify the pattern!

By holding with the left mouse button on the lower border until a double-sided arrow appears and dragging upwards, the vertical dimension of the graph sheet can be shortened. If sufficiently shortened, the pattern becomes obvious. The eye has difficulty in detecting slope patterns where the slope is close to the horizontal or to the vertical.

Then try this:

```
par(mfrow=c(3,1))    # Gives a 3 by 1 layout of plots
plot((1:50)*0.92, sin((1:50)*0.92))
par(mfrow=c(1,1))
```

See Section 2.1 for further examples.

1.5.4 Dimensions and other settings for graphics devices

The shape of the graph sheet can be set when a new graphics page is started. On Microsoft Windows systems, the function `windows()` (or `win.graph()`) starts a new graphics page on the screen display. On Unix X11 systems, specify `x11()`. Under Macintosh OS X, use `quartz()`. Available arguments include `height` (in inches), `width` (in inches), and `pointsize` (there are 72.27 to an inch). The choice of `pointsize`, with a default that varies between devices, affects character heights.[6] See `help(Devices)` for a full list of the devices, including hardcopy devices, that are available on the particular system that is in use.

[6] Note that once a graph has been pasted (from the clipboard) or imported into Microsoft Word or Open Office or another similar word processor, it can be enlarged or shrunk by pointing at one corner, holding down the left mouse button, and pulling.

1.5.5 The plotting of expressions and mathematical symbols

In commands such as `text()` and `mtext()`, character strings can be replaced by expressions. For this purpose an expression is more general than an algebraic or mathematical expression. Thus, the following code gives a grayed out circle, overlaid as in Plate 1B with the formula for the area of a circle:

```
symbols(x=1.5, y=0, circles=1.2, xlim=c(0,3), ylim=c(-1.5,1.5),
        bg="gray", inches=FALSE)
  # inches=FALSE ensures that radius is in x-axis units
text(1.5, 0.5, expression("Area" == pi*phantom("'")*italic(r)^2))
  # Use '==' to insert '='.
  # Text or symbols that appear either side of '*' are juxtaposed.
  # Notice the use of phantom("'") to insert a small space.
```

By default, `symbols()`, like `plot()`, starts a new graphics frame. Various alternatives to circles are available; see `help(symbols)` for details.[7]

Type `help(plotmath)` to get details of available forms of expression. Run `demo(plotmath)` to see some of the possibilities for plotting mathematical symbols. There are further brief details in Section 15.3. Figures 5.3, 10.7, and 15.2 will demonstrate the use of expressions in annotation and/or labeling.

1.5.6 Identification and location on the figure region

Following the drawing of the initial graph, the two functions that may be used are:

- `identify()` labels points;
- `locator()` prints the co-ordinates of points.

In either case, the user positions the cursor at the location for which co-ordinates are required, and clicks the left mouse button. Depending on the platform, the identification or labeling of points may be terminated by pointing outside of the graphics area and clicking, or by clicking with a button other than the first. If continued, the process will terminate after some default number `n` of points, which the user can set. (For `identify()` the default setting is the number of data points, while for `locator()` the default is 500.)

As an example, identify two of the plotted points on the `primates` scatterplot:

```
plot(Brainwt ~ Bodywt, data=primates)
with(primates,
     identify(Brainwt ~ Bodywt, labels=row.names(primates), n=2))
  # Now click near 2 plotted points
```

[7] `## To add the double-headed arrow and associated label, specify:`
`arrows(1.5, 0, 2.7, 0, length=.1, code=3)   # code=3: arrows at both ends`
`  # length is the length of the arrow head (in inches!)`
`text(2.1, -strheight("R"), expression(italic(r) == 1.2))`

1.5.7 Plot methods for objects other than vectors

We have seen how to plot a numeric vector `y` against a numeric vector `x`. The plot function is a generic function that also has special methods for "plotting" various different classes of object. For example, `plot()` accepts a data frame as argument. Try

```
## Use plot() with data frame trees (datasets)
plot(trees)          # Gives a 3 x 3 layout of pairwise
                     # scatterplots among the three variables
```

This has the same effect as the function call `pairs(trees)`.

The scatterplot matrix will be used extensively in Chapter 6 for scrutiny of regression data. See, for example, Subsection 6.2.3. It will be an important tool, also, in the account of multivariate methods in Chapter 12.

1.5.8 Lattice (trellis) graphics

Many of the analyses in later chapters compare different groups within the data. Visual assessments that complement the analysis are indispensable. The *lattice* package has abilities that are suited to such use. The layout on the page, the choice of plotting symbols and colors, and the distinctions within panels, can be used to represent important aspects of data structure. The syntax and graphics conventions are highly consistent across all lattice functions. Lattice's relatively automated provision of highly structured graphical layouts has a cost – changes to the basic layout and structure may be complicated.

Lattice graphics versus base graphics – `xyplot()` *versus* `plot()`

A `Brainwt` versus `Bodywt` scatterplot for the `primates` data, such as was given earlier, might alternatively have been obtained using the function `xyplot()` from the *lattice* package. The following, when typed on the command line, give a plot on the graphics device:

```
## Plot Brainwt vs Bodywt, data frame primates (DAAG)
plot(Brainwt ~ Bodywt, data=primates)       # base graphics
 # 'base' graphics use the abilities of the graphics package
library(lattice)
xyplot(Brainwt ~ Bodywt, data=primates)     # lattice
```

The mechanism that yields the plot is different in the two cases:

- `plot()` gives a graph as a side-effect of the command.
- `xyplot()` generates a graphics object. As this is output to the command line, the object is "printed", i.e., a graph appears.

The following illustrates the difference between the two functions:

```
invisible(plot(Brainwt ~ Bodywt, data=primates))   # Graph appears
invisible(xyplot(Brainwt ~ Bodywt, data=primates)) # No graph
```

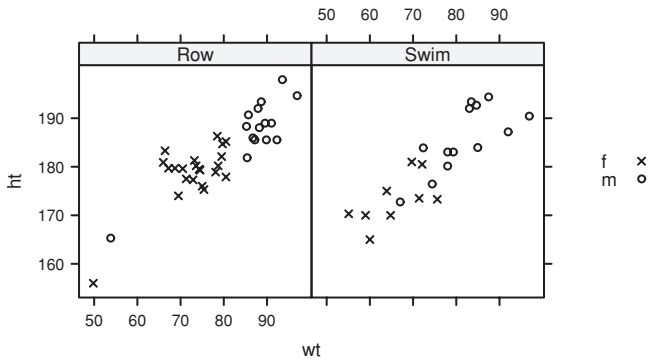


Figure 1.3 Height (ht) versus weight (wt), for two categories of athlete. The different plotting symbols distinguish males from females. The data relate to Telford and Cunningham (1991).

The wrapper function `invisible()` suppresses command line printing, so that `invisible(xyplot(...))` does not yield a graph.

Inside a function, `xyplot(...)` prints a graph only if it is the return value from the function, i.e., usually, is on the final line. In a file that is sourced (use `source()`), no graph will appear. Inside a function (except as mentioned), or in a file that is sourced, there must be an explicit `print()`, i.e.,

```
print(xyplot(ACT ~ year, data=austpop))
```

Panels of scatterplots – the use of `xyplot()`

Graphics functions in the *lattice* package are designed to allow row by column layouts of panels. Different panels are for different subsets of the data. Additionally, points can be distinguished, within panels, according to some further grouping within the data. Chapter 2 will make extensive use of *lattice* functions.

Figure 1.3 demonstrates the use of `xyplot()` with the `ais` data set (*DAAG*) that has data on elite Australian athletes who trained at the Australian Institute of Sport. The plot is restricted to rowers and swimmers. The two panels distinguish the two sports, while different plotting symbols (on a color device, different colors will be used) distinguish females from males. Here is suitable code:

```
trellis.device(color=FALSE)
xyplot(ht ~ wt | sport, groups=sex, pch=c(4,1), aspect=1, data=ais,
       auto.key=list(columns=2), subset=sport%in%c("Row","Swim"))
dev.off()            # Close device
trellis.device()     # Start new device, by default with color=TRUE
```

In the graphics formula `ht ~ wt | sport`, the vertical bar indicates that what follows, in this case `sport`, is a conditioning variable or factor. The graphical information is broken down according to the factor levels or distinct values. The parameter `aspect` controls the ratio of dimensions in the y and x directions.

The setting `auto.key=list(columns=2)` generates a simple key, with the two key items side by side in two columns rather than one under another in a single column as happens with the default setting `columns=1`.

Plotting columns in parallel

Variables and/or factors can be plotted in parallel, on the same (`outer=FALSE`) or different (`outer=TRUE`) panel(s). Separate the names with "+". The following gives a simplified version of Figure 2.10 in Subsection 2.1.5:

```
xyplot(Prairies+Atlantic ~ Date, outer=TRUE, data=jobs)
```

The data frame `jobs` has changes in number of jobs in different regions of Canada over the period January 1995 to December 1996. Subsection 15.5.1 has further discussion on the plotting of columns in parallel.

Selected lattice functions

```
dotplot(factor ~ numeric,..)    # 1-dim. Display
stripplot(factor ~ numeric,..) # 1-dim. Display
barchart(character ~ numeric,..)
histogram( ~ numeric,..)
densityplot( ~ numeric,..)      # Density plot
bwplot(factor ~ numeric,..)     # Box and whisker plot
qqmath(factor ~ numeric,..)     # normal probability plots
splom( ~ dataframe,..)          # Scatterplot matrix
parallel( ~ dataframe,..)       # Parallel coordinate plots
cloud(numeric ~ numeric * numeric, ...)      # 3D surface
wireframe(numeric ~ numeric * numeric, ...)  # 3D scatterplot
```

In each instance, users can add conditioning variables.

Further points to note about the *lattice* package are:

- Because the *lattice* package implements the trellis style of graphics, several of the functions that control stylistic features (color, plot characters, line type, etc.) have *trellis* (where *lattice* might have seemed more natural) as part of their name.
- Lattice graphics functions cannot be mixed (or not easily) with the graphics functions discussed earlier in Section 1.5. It is not possible to use `points()`, `lines()`, `text()`, etc., to add features to a plot that has been created using a *lattice* graphics function. Instead, it is necessary to use functions that are special to *lattice* – `lpoints()`, `llines()`, `ltext()`, `larrows()`, and `lsegments()`.

Subsection 15.5.5 describes a mechanism for interacting with lattice plots.

1.5.9 Good and bad graphs

There is a difference!

Draw graphs so that they are unlikely to mislead. Ensure that they focus the eye on features that are important, and avoid distracting features. Lines that are intended to attract attention can be thickened.

In scatterplots, the intention is typically to draw attention to the points. If there are not too many of them, the use of heavy black dots or other filled symbols will focus attention

on the points, rather than on a fitted line or curve or on the axes. If they are numerous and there is substantial overlap, it then makes better sense to use open symbols. Where there is extensive overlap, ink will fill that region more densely. If there is so much overlap that the use of black symbols would merge most points into a dense black mass, use of a shade of gray may be helpful.[8]

Where the horizontal scale is continuous, patterns of change that are important to identify should bank at an angle of roughly 45° above or below the horizontal. Depending on the context, angles in the approximate range 20° to 70° may be satisfactory, and the aspect ratio should be chosen accordingly. (This was the point of the sine curve example in Subsection 1.5.3.) See Cleveland (1994) for further commentary.

Colors, or gray scales, can often be used to distinguish groupings in the data. Bear in mind that the eye has difficulty in focusing simultaneously on widely separated colors that are close together on the same graph.

1.5.10 Further information on graphics

Several further graphics functions will be introduced in Section 2.1. Note especially `hist()` and `boxplot()`. See also Murrell (2005), Sarkar (2002, 2007).

Note the more detailed information in Chapter 15. Section 15.5 has an extended discussion of the abilities of the *lattice* package. There is brief reference to the relatively specialist abilities of the *grid* package, or which *lattice* is built. Section 15.6 discusses the *ggplot2* package.

1.6 Additional points on the use of R

**Workspace management strategies*

The default choice of working directory, which may be an R installation directory, is not a good choice for long-term use, and should be changed. Subsection 1.1.1 explained how to set the startup choice of working directory.

The working directory can be changed and a new workspace loaded in the course of a session, either using the menu system, if available, or using command line instructions. See Subsection 14.2.2.

In a session where there are extensive calculations, cautious users will from time to time save the current workspace, perhaps first using `rm()` to remove objects that are no longer required. The command `save.image()` will save everything in the workspace, by default into the file `.RData` in the working directory. This can alternatively be done by clicking on the relevant menu item, where such a menu is available. (The file that is saved holds an *image* of the workspace at that point.)

It is good practice to use a separate working directory for each different project. The ability to keep multiple image files in the one directory adds further flexibility. Use the extension `.RData` for such files.[9]

[8] `## Example of plotting with different shades of gray`
`plot(1:4, 1:4, pch=16, col=c("gray20","gray40","gray60","gray80"), cex=3)`

[9] In older versions of R for Windows, `.rda` was an alternative extension.

Forward slashes and backslashes

Note that R syntax follows the Unix conventions and uses forward slashes, where Windows expects backslashes. Thus to read in the file `fuel.txt` from the directory `c:\data`, type

```
fossilfuel <- read.table("c:/data/fuel.txt")
  # Alternative: Replace each "/" by "\\", ie, 2 backslashes
```

Setting the number of decimal places in output

Often, calculations will, by default, give more decimal places of output than are useful. In the output that we give, we often reduce the number of decimal places below what R gives by default. The `options()` function can be used to make a global change to the number of significant digits that are printed. For example:

```
> sqrt(10)
[1] 3.162278
> options(digits=2)  # Change until further notice,
                     # or until end of session.
> sqrt(10)
[1] 3.2
```

Note that `options(digits=2)` expresses a wish, which R will not always obey!

Rounding will sometimes introduce small inconsistencies. For example, in the calculations of Section 4.4:

$$\sqrt{\frac{372}{12}} = 5.57$$

$$\sqrt{2} \times \sqrt{\frac{372}{12}} = 7.88.$$

Note however that $\sqrt{2} \times 5.57 = 7.87$.

Other option settings

Type `help(options)` to get further details. We will meet another important option setting in Chapter 5. (Most of the output that we present uses the setting `options(show.signif.stars=FALSE)`, where the default is `TRUE`. This affects output in Chapter 5 and later chapters.)

Cosmetic issues

In our R code, we write, e.g., `a <- b` rather than `a<-b`, and `y ~ x` rather than `y~x`. This is intended to help readability, perhaps a small step on the way to literate programming. Such presentation details can make a large difference when others use the code.

Where output is obtained with the simple use of `print()` or `summary()`, we have in general included this as the first statement in the output.

*Common sources of difficulty

- It is important to tune the parameter settings of `read.table()` to the input data set. See `help(read.table)` and Subsection 14.4.1 for further details.
- Character vectors that are included as columns in data frames become, by default, factors. There are implications for the use of `read.table()`. See Subsection 14.4.1 and Section 14.6.
- In most contexts, factors are treated as vectors of character strings, with values given by the factor levels. Use `unclass()` to extract the integer values if these, rather than the levels are required. See Section 14.6.
- Keep in mind Section 14.7's comments on the handling of missing values.
- The syntax `fossilfuel[, 2]` extracts the second column from the data frame `fossilfuel`, yielding a numeric vector. Observe however that `fossilfuel[2, ]` yields a data frame, rather than the numeric vector that the user may require. Specify `unlist(fossilfuel[2, ])` to obtain the vector of numeric values in the second row of the data frame. See Subsection 14.9.1.
- The function `sapply()` is commonly used to carry out a computation across all elements of a data frame. If used with a matrix, the computation will be carried out on all matrix elements. See Subsection 14.9.5.
- Once a data frame has been attached, take care with assignments to either the name of the data frame, or the name of a column. Assignment to the name of the data frame will create a new local copy, while assigment to a column name will create a new object in the workspace with that name. Later references to those names will then access the new local copies.
- Data objects that individually or in combination occupy a large part of the available computer memory can slow down all memory-intensive computations. See Subsection 14.2.2 for comment on associated workspace management issues. See also the opening comments in Section 14.8. Note that most of the data objects that are used for our examples are small and thus will not, except where memory is very small, make much individual contribution to demands on memory.

Variable names in data sets

We will refer to a number of different data sets, many of them data frames in our *DAAG* package. When we first introduce the data set, we will give both a general description of the columns of values that we will use, and the names used in the data frame. In later discussion, we will use the name that appears in the data frame whenever the reference is to the particular values that appear in the column.

1.7 Recap

- One use of R is as a calculator, to evaluate arithmetic expressions. Calculations can be carried out in parallel, across all elements of a vector at once.
- Use `q()` to quit from R, usually taking care to save the workspace.

- Help functions include `help()` (help page of a known function), `help.search()` (search for a specified word in the help page header), `apropos()` (search for function names that include a specified character string), and `help.start` (start a browser interface to help information).
- The function `c()` (concatenate) joins vector elements into vectors. It may be used for logical and character vectors, as well as for numeric vectors.
- For simple forms of scatterplot use `plot()`, or the *lattice* function `xyplot()`.
- Important R data structures are vectors, factors, lists, and data frames. Vectors may be of mode numeric, or logical, or character. Factors have mode "numeric" and class "factor".
- Data frames use a list structure to group columns, which must all have the same length, together into a single R object. The different columns may be any mix of logical, numeric, character, or factor.
- Contrast data frames with matrices. All matrix elements have the same mode. A matrix is stored as one long vector that is formatted to appear in a row by column layout.
- Use `is.na()` to identify elements that are `NA`s.
- The R system has extensive abilities for inputting data from rectangular files (see `help(read.table)`), from spreadsheets, and from a variety of statistical package formats. The R Commander GUI offers an easy means to access these abilities.
- Use `attach()` or `with()` (temporary attachment) to give access to the columns of a data frame, without the need to name the data frame whenever a column is accessed.
- The search path determines the order of search for objects that are accessed from the command line, or that are not found in the enclosing environment of a function that accesses them.
- Factors, used for categorical data, are fundamental to the use of many of the R modeling functions. Ordered factors are appropriate for use with ordered categorical data.
- Option settings, which users can change at their discretion, control such matters as the number of significant digits that will be displayed in output.
- Commonly used generic functions include `print()`, `plot()`, and `summary()`. For such functions, the result depends on the class of object that is given as argument.
- To make an assignment and print the value that is assigned, enclose the assigment statement in round brackets. For example:

```
(x <- 2)      # Equivalent to: x <- 2; print(x)
```

1.8 Further reading

Note that the exercises have various hints that extend the discussion in the body of the chapter.

A version of *An Introduction to* R (R Development Core Team, 2009a), current at the time of release, is included with the R distributions. It is available from the CRAN sites as an independent document. (For a list of sites, go to `http://cran.r-project.org`.) Books that include an introduction to R include Dalgaard (2008), Fox (2002).

At a more advanced level note Venables and Ripley (2002), which covers both S-PLUS and R. This will be an important reference throughout this book.

See also documents, including Maindonald (2008), that are listed under **Contributed Documentation** on the CRAN sites. For careful detailed accounts of the R language, see Chambers (2007), Gentleman (2008).

Books and papers that set out principles of good graphics include Cleveland (1993, 1994), Tufte (1997), Wainer (1997), and Wilkinson and Task Force on Statistical Inference (1999). See also the imaginative uses of R's graphical abilities that are demonstrated in Murrell (2005). Maindonald (1992) comments very briefly on graphical design.

References for further reading

Chambers, J. M. 2007. *Software for Data Analysis: Programming with* R.

Cleveland, W. S. 1993. *Visualizing Data.*

Cleveland, W. S. 1994. *The Elements of Graphing Data*, revised edn.

Dalgaard, P. 2008. *Introductory Statistics with* R.

Fox, J. 2002. *An* R *and* S-PLUS *Companion to Applied Regression.*

Gentleman, R. 2008. R *Programming for Bioinformatics*.

Maindonald, J. H. 1992. Statistical design, analysis and presentation issues. *New Zealand Journal of Agricultural Research* 35: 121–41.

Maindonald, J. H. 2008. *Using* R *for Data Analysis and Graphics.* Available as a pdf file at `http://www.maths.anu.edu.au/~johnm/r/usingR.pdf`

Murrell, P. 2005. R *Graphics*.
`http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html`

R Development Core Team. 2009a. *An Introduction to* R.

Tufte, E. R. 1997. *Visual Explanations.*

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

Wainer, H. 1997. *Visual Revelations.*

Wilkinson, L. and Task Force on Statistical Inference. 1999. Statistical methods in psychology journals: guidelines and explanation. *American Psychologist* 54: 594–604.

See the references at the end of the book for fuller bibliographic details.

1.9 Exercises

1. The following table gives the size of the floor area (ha) and the price ($A000), for 15 houses sold in the Canberra (Australia) suburb of Aranda in 1999.

```
   area sale.price
1   694 192.0
2   905 215.0
3   802 215.0
4  1366 274.0
5   716 112.7
6   963 185.0
7   821 212.0
8   714 220.0
9  1018 276.0
10  887 260.0
```

```
11  790 221.5
12  696 255.0
13  771 260.0
14 1006 293.0
15 1191 375.0
```

Type these data into a data frame with column names `area` and `sale.price`.

(a) Plot `sale.price` versus `area`.
(b) Use the `hist()` command to plot a histogram of the sale prices.
(c) Repeat (a) and (b) after taking logarithms of sale prices.
(d) The two histograms emphasize different parts of the range of sale prices. Describe the differences.

2. The `orings` data frame gives data on the damage that had occurred in US space shuttle launches prior to the disastrous Challenger launch of 28 January 1986. The observations in rows 1, 2, 4, 11, 13, and 18 were included in the pre-launch charts used in deciding whether to proceed with the launch, while remaining rows were omitted.

   Create a new data frame by extracting these rows from `orings`, and plot `total` incidents against `temperature` for this new data frame. Obtain a similar plot for the full data set.

3. For the data frame `possum` (*DAAG* package)

   (a) Use the function `str()` to get information on each of the columns.
   (b) Using the function `complete.cases()`, determine the rows in which one or more values is missing. Print those rows. In which columns do the missing values appear?

4. For the data frame `ais` (*DAAG* package)

   (a) Use the function `str()` to get information on each of the columns. Determine whether any of the columns hold missing values.
   (b) Make a table that shows the numbers of males and females for each different sport. In which sports is there a large imbalance (e.g., by a factor of more than 2:1) in the numbers of the two sexes?

5. Create a table that gives, for each species represented in the data frame `rainforest`, the number of values of `branch` that are `NA`s, and the total number of cases.
   [*Hint:* Use either `!is.na()` or `complete.cases()` to identify `NA`s.]

6. Create a data frame called `Manitoba.lakes` that contains the lake's `elevation` (in meters above sea level) and `area` (in square kilometers) as listed below. Assign the names of the lakes using the `row.names()` function.

```
                elevation  area
Winnipeg              217 24387
Winnipegosis          254  5374
Manitoba              248  4624
SouthernIndian        254  2247
Cedar                 253  1353
Island                227  1223
Gods                  178  1151
Cross                 207   755
Playgreen             217   657
```

(a) Use the following code to plot `log2(area)` versus `elevation`, adding labeling information (there is an extreme value of `area` that makes a logarithmic scale pretty much essential):

```
attach(Manitoba.lakes)
plot(log2(area) ~ elevation, pch=16, xlim=c(170,280))
  # NB: Doubling the area increases log2(area) by 1.0
text(log2(area) ~ elevation,
     labels=row.names(Manitoba.lakes), pos=4)
text(log2(area) ~ elevation, labels=area, pos=2)
title("Manitoba's Largest Lakes")
detach(Manitoba.lakes)
```

Devise captions that explain the labeling on the points and on the y-axis. It will be necessary to explain how distances on the scale relate to changes in area.

(b) Repeat the plot and associated labeling, now plotting `area` versus `elevation`, but specifying `log="y"` in order to obtain a logarithmic y-scale. [*Note:* The `log="y"` setting carries across to the subsequent `text()` commands. See Subsection 2.1.5 for an example.]

7. Look up the help page for the R function `dotchart()`. Use this function to display the areas of the Manitoba lakes (a) on a linear scale, and (b) on a logarithmic scale. Add, in each case, suitable labeling information.

8. Using the `sum()` function, obtain a lower bound for the area of Manitoba covered by water.

9. The second argument of the `rep()` function can be modified to give different patterns. For example, to get four 2s, then three 3s, then two 5s, enter

```
rep(c(2,3,5), c(4,3,2))
```

(a) What is the output from the following command?

```
rep(c(2,3,5), 4:2)
```

(b) Obtain a vector of four 4s, four 3s, and four 2s.

(c) The argument `length.out` can be used to create a vector whose length is `length.out`. Use this argument to create a vector of length 50 that repeats, as many times as necessary, the sequence: `3 1 1 5 7`

(d) The argument `each` can be used to form a vector in which each element in the first argument is replaced by the specified number of repeats of itself. Use this to create a vector in which each of `3 1 1 5 7` is replaced by four repeats of itself. Show, also, how this can be done without use of the argument `each`.

10. The ^ symbol denotes exponentiation. Consider the following:

```
1000*((1+0.075)^5 - 1)  # Interest on $1000, compounded
                        # annually at 7.5% p.a. for five years
```

(a) Evaluate the above expression.

(b) Modify the expression to determine the amount of interest paid if the rate is 3.5% p.a.

(c) Explain the result obtained when the exponent `5` is changed to `seq(1, 10)`.

11. Run the following code:

```
gender <- factor(c(rep("female", 91), rep("male", 92)))
table(gender)
gender <- factor(gender, levels=c("male", "female"))
table(gender)
```

```
gender <- factor(gender, levels=c("Male", "female"))
                      # Note the mistake: "Male" should be "male"
table(gender)
table(gender, exclude=NULL)
rm(gender)            # Remove gender
```

Explain the output from the successive uses of `table()`.

12. Write a function that calculates the proportion of values in a vector `x` that exceed some value `cutoff`.

    (a) Use the sequence of numbers 1, 2, . . . , 100 to check that this function gives the result that is expected.
    (b) Obtain the vector `ex01.36` from the *Devore6* (or *Devore7*) package. These data give the times required for individuals to escape from an oil platform during a drill. Use `dotplot()` to show the distribution of times. Calculate the proportion of escape times that exceed 7 minutes.

13. The following plots four different transformations of the `Animals` data from the *MASS* package. What different aspects of the data do these different graphs emphasize? Consider the effect on low values of the variables, as contrasted with the effect on high values.

```
par(mfrow=c(2,2))     # 2 by 2 layout on the page
library(MASS)         # Animals is in the MASS package
plot(brain ~ body, data=Animals)
plot(sqrt(brain) ~ sqrt(body), data=Animals)
plot(I(brain^0.1) ~ I(body^0.1), data=Animals)
  # I() forces its argument to be treated "as is"
plot(log(brain) ~ log(body), data=Animals)
par(mfrow=c(1,1))     # Restore to 1 figure per page
```

14. Use the function `abbreviate()` to obtain six-character abbreviations for the row names in the data frame `cottonworkers` (*DAAG* package). Plot `survey1886` against `census1886`, and plot `avwage*survey1886` against `avwage*census1886`, in each case using the six-letter abbreviations to label the points. How should each of these graphs be interpreted? [*Hint:* Be sure to specify `I(avwage*survey1886)` and `I(avwage*census1886)` when plotting the second of these graphs.]

15. The data frame `socsupport` (*DAAG*) has data from a survey on social and other kinds of support, for a group of university students. It includes Beck Depression Inventory (`BDI`) scores. The following are two alternative plots of `BDI` against `age`:

```
plot(BDI ~ age, data=socsupport)
plot(BDI ~ unclass(age), data=socsupport)
```

    For examination of cases where the score seems very high, which plot is more useful? Explain. Why is it necessary to be cautious in making anything of the plots for students in the three oldest age categories (`25-30`, `31-40`, `40+`)?

16. Functions that can be useful for labeling points on graphs are `abbreviate()` (create abbreviated names), and `paste()` (create composite labels). A composite label might, for the data from `socsupport`, give information about `gender`, `country`, and row number. Try the following:

```
gender1 <- with(socsupport, abbreviate(gender, 1))
table(gender1)        # Examine the result
```

```
country3 <-  with(socsupport, abbreviate(country, 3))
table(country3)       # Examine the result
```
Now use the following to create a label that can be used with `text()` or with `identify()`:
```
num <- with(socsupport, seq(along=gender))   # Generate row numbers
lab <- paste(gender1, country3, num, sep=":")
```
Use `identify()` to place labels on all the points that the boxplots have identified as "outliers".

17. Given a vector `x`, the following demonstrates alternative ways to create a vector of numbers from 1 through n, where n is the length of the vector:
```
x <-  c(8, 54, 534, 1630, 6611)
seq(1, length(x))
seq(along=x)
```
Now set `x <- NULL` and repeat each of the calculations `seq(1, length(x))` and `seq(along=x)`. Which version of the calculation should be used in order to return a vector of length 0 in the event that the supplied argument is `NULL`.

18. The `Rabbit` data frame in the *MASS* library contains blood pressure change measurements on five rabbits (labeled as `R1`, `R2`, ..., `R5`) under various control and treatment conditions. Read the help file for more information. Use the `unstack()` function (three times) to convert `Rabbit` to the following form:
```
   Treatment    Dose    R1    R2    R3    R4   R5
1    Control    6.25  0.50  1.00  0.75  1.25  1.5
2    Control   12.50  4.50  1.25  3.00  1.50  1.5
. . . .
6    Control  200.00 32.00 29.00 24.00 33.00 18.0
7        MDL    6.25  1.25  1.40  0.75  2.60  2.4
8        MDL   12.50  0.75  1.70  2.30  1.20  2.5
. . . .
12       MDL  200.00 37.00 28.00 25.00 22.00 19.0
```

19. The data frame `vlt` (*DAAG*) consists of observations taken on a video lottery terminal during a two-day period. Eight different objects can appear in each of three windows. Here, they are coded from 0 through 7. Different combinations of the objects give prizes (although with small probability). The first four rows are:
```
> head(vlt, 4)       # first few rows of vlt
  window1 window2 window3 prize night
1       2       0       0     0     1
2       0       5       1     0     1
3       0       0       0     0     1
4       2       0       0     0     1
>   # . . .
```
Use `stack()` to convert the first three columns of this data set to a case-by-variable format, then creating a tabular summary of the results, broken down by window.
```
vltcv <- stack(vlt[, 1:3])
head(vltcv)          # first few rows of vltcv
table(vltcv$values, vltcv$ind)
  # More cryptically, table(vltcv) gives the same result.
```
Does any window stand out as different?

20.* The help page for `iris` (type `help(iris)`) gives code that converts the data in `iris3` (*datasets* package) to case-by-variable format, with column names "Sepal.Length",

"Sepal.Width", "Petal.Length", "Petal.Width", and "Species". Look up the help pages for the functions that are used, and make sure that you understand them. Then add annotation to this code that explains each step in the computation.

21.* The following uses the `for()` looping function to plot graphs that compare the relative population growth (here, by the use of a logarithmic scale) for the Australian states and territories.

```
oldpar <- par(mfrow=c(2,4))
for (i in 2:9){
plot(austpop[, 1], log(austpop[, i]), xlab="Year",
     ylab=names(austpop)[i], pch=16, ylim=c(0,10))}
par(oldpar)
```

Find a way to do this without looping. [*Hint:* Use the function `sapply()`, with `austpop[,2:9]` as the first argument.]

2

Styles of data analysis

What is the best way to begin investigation of a new set of data? What forms of data exploration will draw attention to obvious errors or quirks in the data, or to obvious clues that the data contain? What checks are desirable before proceeding with an intended formal analysis, or to help decide what formal analysis may be appropriate? What can be learned from investigations that other researchers have done with similar data?

Competent statisticians have always used graphs to check their data. Numerical summaries, such as an average, can be very useful, but important features of the data may be missed without a glance at an appropriate graph. Careful consideration may be needed to choose a graph that will be effective for the purpose in hand.

We will see in Chapter 3 that an integral part of statistical analysis is the development of a model that accurately describes the data, clarifies what the data say, and makes prediction possible. Without model assumptions, there cannot be a meaningful formal analysis! As assumptions are strengthened, the chances of getting clear results improve. The price for stronger assumptions is that, if wrong, the results may be wrong. Graphical techniques have been developed for checking, to the extent possible, many of the assumptions that must be made in practice.

Preliminary scrutiny of the data can readily degenerate into data snooping, so that the analysis is unduly attuned to statistical artefacts of the particular data that are to be analyzed. Under torture, the data readily yield false confessions. To avoid this, strict limits must be placed on the extent to which the data are allowed to influence the choice of model for the formal analysis.

Even if data have not been collected in a way that makes them suitable for formal statistical analysis, exploratory techniques can often glean clues from them. However, it is unwise, as too often happens, to rely on this possibility!

2.1 Revealing views of the data

The use of graphs to display and help understand data has a long tradition. John W. Tukey formalized and extended this tradition, giving it the name *Exploratory Data Analysis* (EDA). Tukey has had a huge influence on data analysis practices; see Hoaglin (2003). A key concern is that data should, as far as possible, have the opportunity to speak for themselves, prior to or as part of a formal analysis.

A use of graphics that is broadly in an EDA tradition continues to develop and evolve. Statistical theory has an important role in suggesting forms of display that may be helpful

and interpretable. Advances in computing have been important, facilitating the development and use of many of the graphical tools now available. The best modern statistical software makes a strong connection between data analysis and graphics, combining the computer's ability to crunch numbers and present graphs with the ability of a trained human eye to detect pattern.

Graphical exploration after the style of EDA has at least four roles:

- It may suggest ideas and understandings that had not previously been contemplated. This use of EDA fits well with the view of science as inductive reasoning.
- It may challenge the theoretical understanding that guided the initial collection of the data. It then acquires a more revolutionary role. It becomes the catalyst, in the language of Thomas Kuhn, for a paradigm shift.
- It allows the data to criticize an intended analysis and facilitates checks on assumptions. Subsequent formal analysis can then proceed with greater confidence.
- It may reveal additional information, not directly related to the research question. It may, for example, suggest fruitful new lines of research.

The next several subsections will describe the histogram and density plot, the stem-and-leaf display, the boxplot, the scatterplot, the lowess smoother, and the trellis-style graphics that are available in the *lattice* package. The *lattice* functions greatly extend the available styles and layouts.

2.1.1 Views of a single sample

Histograms and density plots

The histogram is a basic (and over-used) EDA tool for displaying the frequency distribution of a set of data. The area of each rectangle of a histogram is proportional to the number of observations whose values lie within the width of the rectangle. A mound-shaped histogram may make it plausible that the data follow a normal distribution (the "bell curve"). In small samples, however, the shape can be highly irregular. In addition, the appearance can depend on the choice of breakpoints, which is a further reason for caution in interpreting the shape. It is often helpful to try more than one set of breakpoints.

The data set `possum` (*DAAG* package) has nine morphometric measurements on each of 104 mountain brushtail possums, trapped at seven sites from southern Victoria to central Queensland (data relate to Lindenmayer *et al.*, 1995). Attention will be limited to the measurements for 43 females, placing them in a subset data frame that will be called `fossum`. The following code creates this subset data frame:

```
library(DAAG)          # Ensure that the DAAG package is attached
## Form the subset of possum that holds data on females only
fossum <- subset(possum, sex=="f")
```

Panels A and B of Figure 2.1 exhibit histogram plots of the frequency distribution of the total lengths of the female possums.[1]

[1] `## To get a 1 by 4 layout, precede with`
`par(mfrow = c(1,4))`

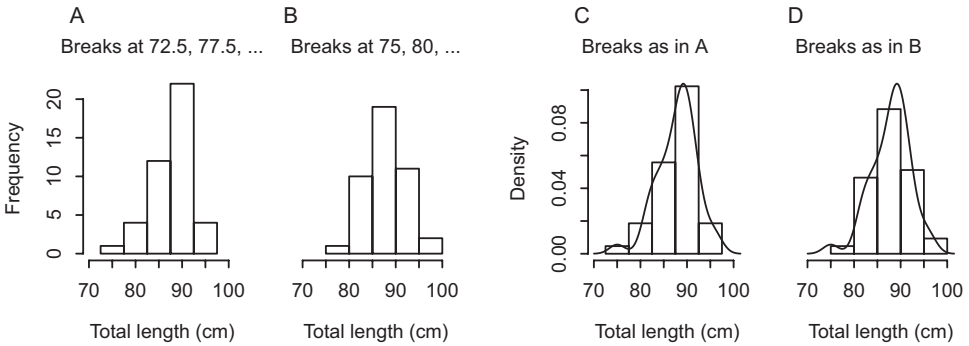


Figure 2.1 The histograms in panels A and B show the same data, but with a different choice of breakpoints. In panels C and D, density plots are overlaid on the histograms from panels A and B, respectively.

```
attach(fossum)
hist(totlngth, breaks = 72.5 + (0:5) * 5, ylim = c(0, 22),
     xlab="Total length (cm)", main ="A: Breaks at 72.5, 77.5, ...")
hist(totlngth, breaks = 75 + (0:5) * 5, ylim = c(0, 22),
     xlab="Total length (cm)", main="B: Breaks at 75, 80, ...")
```

The only difference in the construction of the two plots is the choice of breakpoints, but one plot suggests that the distribution is asymmetric (skewed to the left), while the other suggests symmetry.

A histogram is a crude form of a density estimate. A better alternative is, often, a smooth density estimate, as in Figures 2.1C and D. Whereas the width of histogram bars must be chosen somewhat subjectively, density estimates require the choice of a bandwidth parameter that controls the amount of smoothing. In both cases, the software has default choices that can work reasonably well.

```
dens <- density(totlngth)
xlim <- range(dens$x); ylim <- range(dens$y)
hist(totlngth, breaks = 72.5 + (0:5) * 5, probability = T,
     xlim = xlim, ylim = ylim, xlab="Total length (cm)", main=" ")
lines(dens)
hist(totlngth, breaks = 75 + (0:5) * 5, probability = T,
     xlim = xlim, ylim = ylim, xlab="Total length (cm)", main= " ")
lines(dens)
par(mfrow=c(1,1)); detach(fossum)
```

The height of the density curve at any point is an estimate of the proportion of sample values per unit interval, locally at that point. Observe that in Figures 2.1A and C, the cell of the histogram between the breakpoints (87.5, 92.5] has a frequency of 22. As the total frequency is 43, and the width of the cell is 5, this corresponds to a density of $\frac{22}{43\times 5} = 0.102$, which is just a little smaller than the height of the density curve at its highest point or mode.

Much of the methodology in this book makes assumptions that hold exactly only if the data follow a normal distribution (the "bell curve"), discussed in the next chapter. Density curves are preferable to histograms for drawing attention to particular forms of

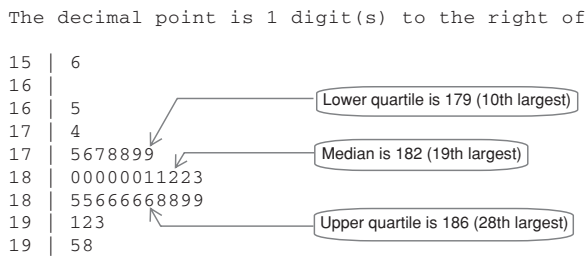


Figure 2.2 Stem-and-leaf display showing the heights of the 37 rowers in the `ais` data set. Annotation has been added that identifies the lower quartile, the median, and the upper quartile.

non-normality, such as that associated with strong skewness in the distribution, but are still not an adequate tool. A more effective way of checking for normality – the normal probability plot – is described in Subsection 3.4.2. Density curves are useful for estimating the population mode, i.e., the value that occurs most frequently.

Where data values have sharp lower and/or upper cutoff limits, use the arguments `from` and `to` to specify those limits. For example, a failure time distribution may have a mode close to zero, with a sharp cutoff at zero.

The stem-and-leaf display

The stem-and-leaf display is a fine-grained alternative to a histogram, for use in displaying a single column of numbers. Figure 2.2 shows a stem-and-leaf plot of the heights of the 37 rowers in the `ais` data set. Code is:

```
with(ais, stem(ht[sport=="Row"]))
```

The data have been rounded to the nearest centimeter. The numbers that are displayed are, in order of magnitude, 156, 165, 174, ... The display has broken these down as $150 + 6$, $160 + 5$, $170 + 4$, ... The column of numbers on the left of the vertical bars (15, 16, . . .) comprises the *stem*; these are the tens of centimeters parts of the numbers. The leaf part for that number (6, 5, 4, . . .) is what remains after removing the stem; these are printed, in order, to the right of the relevant vertical bar.

As there are 37 data values, the median or middle value is the 19th. Starting from the 156 leaf in the first line of the stem-and-leaf diagram and working down, 18 values precede the 19th largest, and 18 values follow. Thus the median (or 50th percentile) is 182. The first and third quartiles (the 25th and 75th percentiles) can be recovered in a similar way, with the exact value depending on the details of the formula used for their calculation. For present purposes the first quartile can be taken as the 10th largest value (= 179), while the third quartile is the 28th largest value (= 186), or the 10th value when starting at the largest and counting down. (The number 10 is the average of the ranks 1 and 19, while 28 is the average of 19 and 39.)[2]

[2] `## Use quantile() to obtain the quartiles of ht: data frame ais (DAAG package)`
`quantile(ais$ht[ais$sport=="Row"], prob=c(.25,.5,.75))`
`  # For the 50th percentile (the 2nd quartile), an alternative is median()`

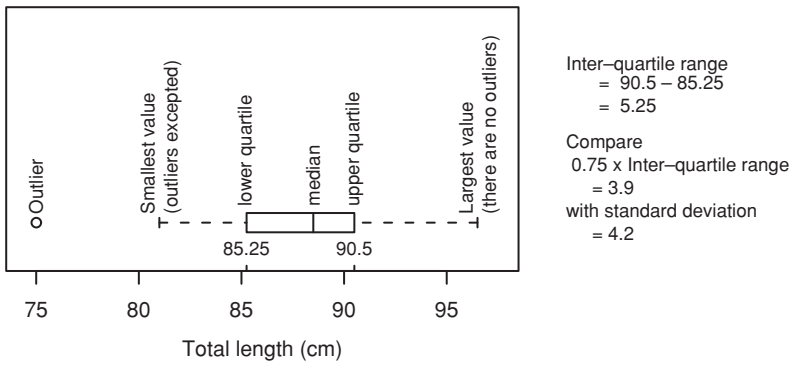


Figure 2.3 Boxplot, with annotation that explains boxplot features.

Boxplots

Like the histogram, the boxplot is a coarse summary. It allows a trained eye to comprehend at a glance specific important features of the data. Figure 2.3 shows a boxplot of total lengths of females in the `possum` data set, with annotation added that explains the interpretation of boxplot features. Code that gives the boxplot, without the annotation, is:

```
## Base graphics boxplot function
with(fossum, boxplot(totlngth, horiz=TRUE))
## Alternative: lattice graphics bwplot function
bwplot(~totlngth, data=fossum)
```

Notice that one point lies outside the "whiskers" to the left, and is thus flagged as a possible outlier. An outlier is a point that, in some sense, lies away from the main body of the data. In identifying points that are flagged as possible outliers, the normal distribution (to be discussed in Subsection 3.2.2) is taken as the standard. Using the default criterion one point in 100 will on average, for data from a normal distribution, be flagged as a possible outlier. Thus, in a boxplot display of 1000 values that are drawn at random from a normal distribution, around 10 will be plotted out beyond the boxplot whiskers and thus flagged as possible outliers. Subsection 2.1.7 has further comment on outliers.

The discussion of the normal and other distributions in Chapter 3 should help clarify these ideas.

2.1.2 Patterns in univariate time series

In Figure 2.4, "measles" includes both what is nowadays called measles and the closely related *rubella* or German measles.[3] Panel A uses a logarithmic vertical scale. Panel B uses an unlogged scale and takes advantage of the fact that deaths from measles are of the order, in any year, of one thousandth of the population. Thus, deaths in thousands and population in millions can be shown on the same scale.

[3] For details of the data, and commentary, see Guy (1882), Stocks (1942), Senn (2003). (Guy's interest was in the comparison with smallpox mortality.) The population estimates (`londonpop`) are from Mitchell (1988).

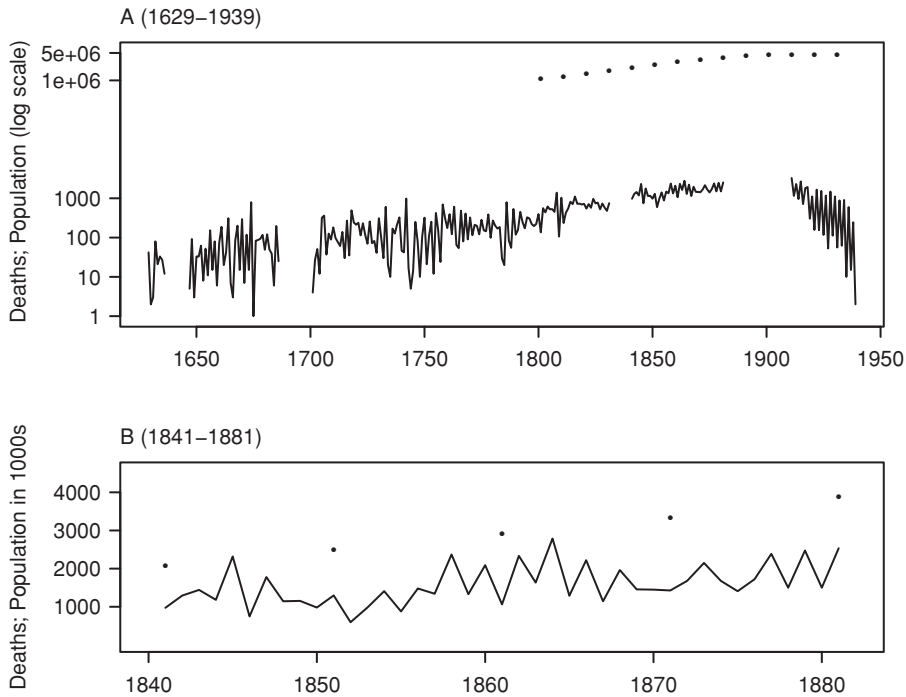


Figure 2.4 The two panels provide different insights into data on mortality from measles, in London over 1629–1939. Panel A shows the numbers of deaths from measles in London for the period from 1629 through 1939 (black curve) and the London population (in thousands, black dots). A log scale has been used (see Subsection 2.1.3 for details). The lower panel B shows the subset of the measles data for the period 1840 through 1882 on the linear scale (black curve), together with the London population (in thousands, black dots).

Simplified code is:

```
## Panel A
plot(log10(measles), xlab="", ylim=log10 (c(1,5000*1000)),
     ylab=" Deaths; Population (log scale)", yaxt="n")
ytiks <- c(1, 10, 100, 1000, 1000000, 5000000)
## London population in thousands
londonpop <-
  ts(c(1088,1258,1504,1778,2073,2491,2921,3336,3881,4266,
       4563,4541,4498,4408), start=1801, end=1931, deltat=10)
points(log10(londonpop*1000), pch=16, cex=.5)
axis(2, at=log10(ytiks), labels=paste(ytiks), las=2)
## Panel B
plot(window(measles, start=1840, end=1882), ylim=c (0, 4600),
     yaxt="n")
axis(2, at=(0:4)* 1000, labels=paste(0:4), las=2)
```

The function `plot()` recognizes that `measles` is a time series object, and calls the plot method `plot.ts()` that is used for time series. For details, see `help(plot.ts)`. Notice the use, for panel B, of the function `window()` that extracts a subseries.

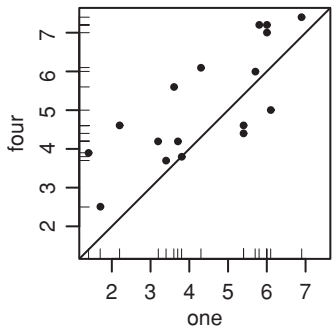


Figure 2.5 Each of 17 panelists compared two milk samples for sweetness. One sample had one unit of additive, while the other had four units of additive.

Panel A shows broad trends over time, but is of no use for identifying changes on the time scale of a year or two. In panel B, the lines that show such changes are, mostly, at an angle that is in the approximate range of 20° to 70° from the horizontal.

A sawtooth pattern, by which years in which there are many deaths are commonly followed by years in which there are fewer deaths, is thus clearly evident. (To obtain this level of detail for the whole period from 1629 until 1939, multiple panels would be necessary.)

The following, with *y*-axis labeling in logarithms of numbers and omitting the population estimates, demonstrates the combining of the two graphs on the one page:

```
## Panel A:
par(fig=c(0, 1, .38, 1))   # 38% to 100% of page, in y-direction
plot(log10(measles), ylab="log10(Deaths)",
     ylim=log10(c(1,5000*1000)))
mtext(side=3, line=0.5, "A (1629-1939)", adj=0)
## Panel B: window from 1840 to 1882; more complete code
par(fig=c(0, 1, 0, .4), new=TRUE)  # 0% to 38% of height of figure region
plot(window(measles, start=1840, end=1882), ylab="Deaths")
mtext(side=3, line=0.5, "B (1841-1881)", adj=0)
par(fig=c(0, 1, 0, 1))     # Restore default figure region
```

2.1.3 Patterns in bivariate data

The scatterplot is a simple but important tool for the examination of pairwise relationships. We will illustrate with specific examples.

Figure 2.5 shows data from a tasting session where each of 17 panelists assessed the sweetness of each of two milk samples, one with four units of additive, and the other with one unit of additive. The line $y = x$ has been added. The function `rug()` adds a "rug", i.e., short bars at right angles to one or other axis that show the distribution values along that axis of the plot. The code is:

```
## Plot four vs one: data frame milk (DAAG)
xyrange <- range(milk)
plot(four ~ one, data = milk, xlim = xyrange, ylim = xyrange,
     pch = 16, pty="s")    # pty="s": square plotting region
```

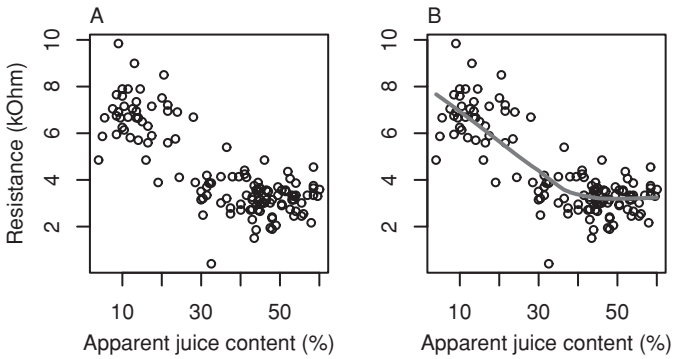


Figure 2.6 Electrical resistance versus apparent juice content. Panel B repeats panel A, but with a smooth curve fitted to the data.

```
rug(milk$one)                # x-axis rug (default is side=1)
rug(milk$four, side = 2)     # y-axis rug
abline(0, 1)
```

There is a positive correlation between assessments for the two samples; if one was rated as sweet, by and large so was the other. The line $y = x$ assists in comparing the two samples. Most panelists (13 out of 17) rated the sample with four units of additive as sweeter than the sample with one unit of additive.

The fitting of a smooth trend curve

Figure 2.6 shows data from a study that measured both electrical resistance and apparent juice content for a number of slabs of kiwifruit. The curve in panel B, obtained using the lowess method that is discussed further in Subsection 7.5.4, estimates the relationship between electrical resistance and apparent juice content. The code is:

```
## Plot ohms vs juice: data frame fruitohms (DAAG)
plot(ohms ~ juice, xlab="Apparent juice content (%)",
     ylab="Resistance (ohms)", data=fruitohms)
## Add a smooth curve, as in Panel B
with(fruitohms, lines(lowess(juice, ohms), lwd=2))
 # With lwd=2, the curve is twice the default thickness
```

The fitted smooth curve shows a form of response that is clearly inconsistent with a straight line. It suggests an approximate linear relationship for juice content up to somewhat over 35%. Once the juice content reaches around 45%, the curve becomes a horizontal line, and there is no evident further change in resistance. There is no obvious simple form of equation that might be used to describe the curve.

A curve fitted using `lowess()` or another such smoothing function can provide a useful benchmark against which to compare the curve given by a theoretical or other mathematical form of equation that the data are thought to follow.

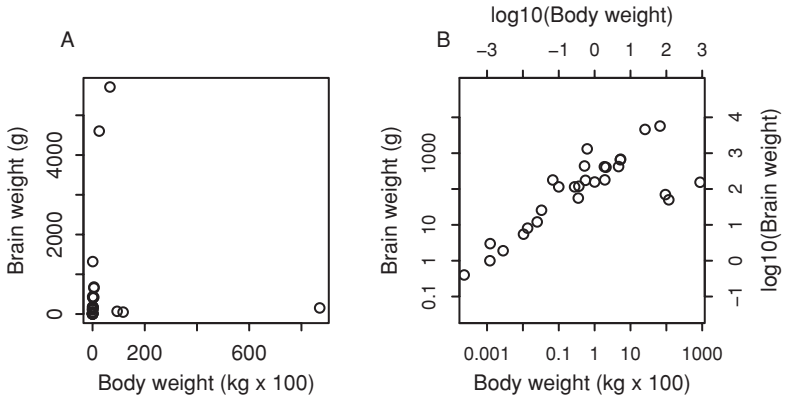


Figure 2.7 Brain weight versus body weight, for 27 animals that vary greatly in size. Panel A uses untransformed scales, while panel B uses logarithmic scales, on both axes.

What is the appropriate scale?

Figures 2.7A and B plot brain weight (g) against body weight (kg), for a number of different animals:

```
## The following omits the labeling information
oldpar <- par(mfrow = c(1,2), pty="s")
## Plot brain vs body: data frame Animals (MASS package)
library(MASS)
plot(brain ~ body, data=Animals)           # Panel A
plot(log(brain) ~ log(body), data=Animals) # Panel B
par(oldpar)
```

Figure 2.7A is almost useless. The axes should be transformed so that the data are spread out more evenly. Here, we can do this by choosing a logarithmic scale. Multiplication by the same factor (e.g., for the tick marks in Figure 2.7B, by a factor of 10) always gives the same distance along the scale. If we marked points 1, 5, 25, 125, . . . along the vertical axis, they would also lie an equal distance apart.

A logarithmic scale is appropriate for quantities that change multiplicatively. For example, if cells in a growing organism divide and produce new cells at a constant rate, then the total number of cells changes in a multiplicative manner, resulting in so-called exponential growth. Growth in the bodily measurements of organisms may similarly be multiplicative, with large organisms increasing in some time interval by the same approximate fraction as smaller organisms. Random changes in the relative growth rate will produce adult organisms whose size (e.g., height) is, on the logarithmic scale, approximately normally distributed. The reason is that growth rate on a natural logarithmic scale ($\log_e$) equals the relative growth rate. Derivation of this result is a straightforward use of the differential calculus.

The logarithmic transformation is so commonly needed that it has seemed necessary to introduce it at this point. Biologists, economists, and others should become comfortable with its use. There is a brief discussion of other transformations in Chapter 5.

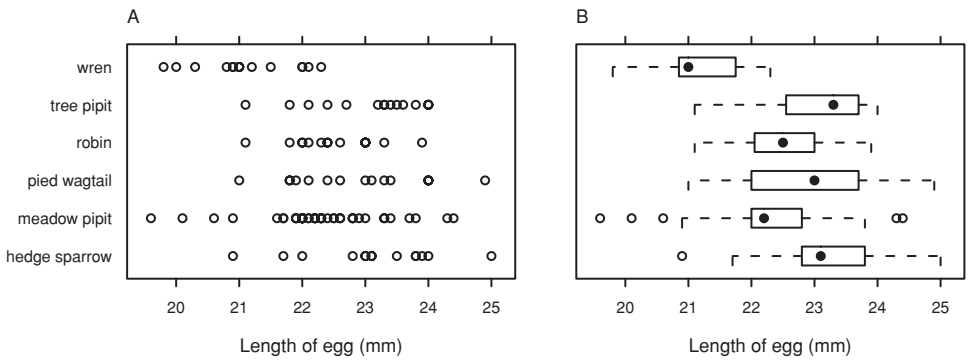


Figure 2.8 Strip plot (panel A) and boxplot (panel B) displays of cuckoo egg lengths. Data, from Latter (1902), are reproduced in summarized form in Tippett (1931).

2.1.4 Patterns in grouped data – lengths of cuckoo eggs

Cuckoos lay eggs in the nests of other birds. The eggs are then unwittingly adopted and hatched by the host birds. In Figure 2.8 the egg lengths are grouped by the species of the host bird, using both a strip plot display (panel A) and boxplot summaries (panel B).

Strip plots and boxplots allow convenient side-by-side comparisons of different groups, here the different host species. The main part of the code used for these plots is:

```
## Compare stripplot() with bwplot(), both from lattice package
stripplot(species ~ length, xlab="Length of egg (mm)", data=cuckoos)
bwplot(species ~ length, xlab="Length of egg (mm)", data=cuckoos,
       scales=list(y=list(alternating=0)))
  # alternating=0; omit y-axis labels
```

Eggs planted in wrens' nests appear smaller than eggs planted in other birds' nests. Apart from several outlying egg lengths in the meadow pipit nests, the length variability within each host species' nest is fairly uniform.

Fuller details of the code are in the footnote.[4]

Comparing densities between groups – lattice style density plots

Lattice-style density plots can be useful for getting an indication of how distributions may differ across different groups of data. Figure 2.9 compares the ear conch measurements

[4]
```
## For tidier labels replace ".", in several of the species names, by a space
specnam <- with(cuckoos, sub(pattern=".", replacement=" ", levels(species), fixed=TRUE))
  # fixed=TRUE: do not interpret "." as a 'regular expression',
## Panel A: Strip plot: data frame cuckoos (DAAG)
plt1 <- stripplot(species ~ length, factor.levels=specnam, data=cuckoos)
print(update(plt1, xlab="Length of egg (mm)"),
      position=c(0,0,0.55,1))  # xmin, ymin, xmax, ymax
  # Use print() to display lattice graphics objects
## Panel B: Box plot
plt2 <- bwplot(species ~ length, factor.levels=specnam, data=cuckoos)
print(update(plt2, xlab="Length of egg (mm)", scales=list(y=list(alternating=0))),
      newpage=FALSE, position=c(0.55,0,1,1))
```

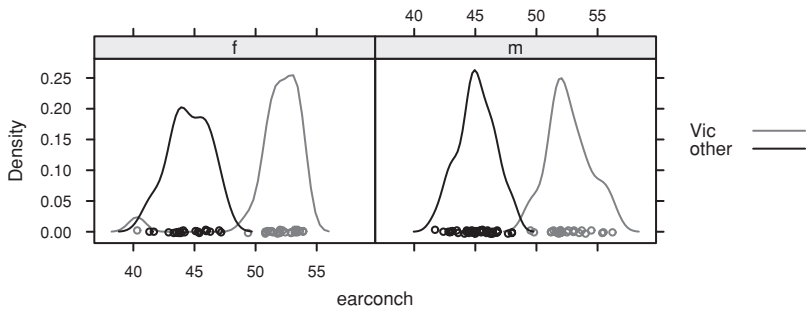


Figure 2.9 Density plot that compares the ear conch measurements for each of the two "populations" of possums, for males and females separately.

of male and female possums, for each of two "populations" (`Vic` and `other`) of possums:

```
## Density plot for earconch: data frame possum (DAAG package)
library(lattice)
densityplot(~earconch | sex, groups=Pop, data=possum,
                        auto.key=list(space="right"))
```

2.1.5* Multiple variables and times

Overlaying plots of several time series (sequences of measurements taken at regular intervals) might seem appropriate for making direct comparisons. However, this approach will only work if the scales are similar for the different series.

The data frame `jobs` (*DAAG*) gives the number of workers (in thousands) in the Canadian labor force, broken down by region (BC, Alberta, Prairies, Ontario, Quebec, Atlantic), for the 24-month period from January 1995 to December 1996. Over this time, Canada was emerging from a deep economic recession. Columns 1–6 have the respective numbers for six different regions. The ranges of values in the columns are:

```
> ## Apply function range to columns of data frame jobs (DAAG)
> sapply(jobs, range)
       BC Alberta Prairies Ontario Quebec Atlantic     Date
[1,] 1737    1366       973    5212   3167       941 95.00000
[2,] 1840    1436       999    5360   3257       968 96.91667
```

In order to see where the economy was taking off most rapidly, it is tempting to plot all of the series on the same graph. In order that similar changes on the scale will correspond to similar proportional changes, a logarithmic scale is used in Figure 2.10A:

```
## Simplified plot; all series in a single panel; use log scale
(simplejobsA.xyplot <-
  xyplot(Ontario+Quebec+BC+Alberta+Prairies+Atlantic ~ Date,
         outer=FALSE, data=jobs, type="b",
         ylab="Number of workers", scales=list(y=list(log="e")),
         auto.key=list(space="right", lines=TRUE)))
```

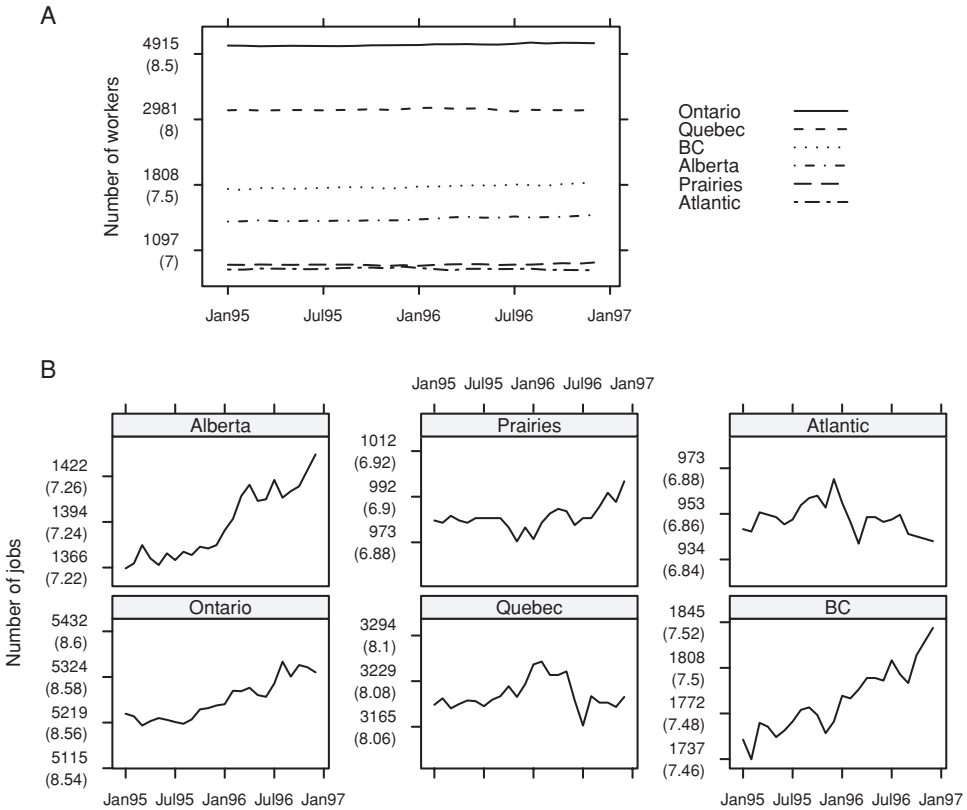


Figure 2.10 Data are numbers in the labor force (thousands) for various regions of Canada, at quarterly intervals over 1995–1996. Panel A uses the same logarithmic scale for all regions. Panel B shows the same data as in panel A, but now with separate ("sliced") logarithmic scales on which the same percentage increase, e.g., by 1%, corresponds to the same distance on the scale, for all plots. Distances between ticks are 0.02 on the $\log_e$ scale, i.e., a change of almost exactly 2%.

The trellis object has been saved so that it can be updated, as demonstrated in the footnote, to give the graph shown in Figure 2.10A.[5]

The use of column names that are joined with `"+"` has the result that the columns are plotted in parallel. The regions have been taken in order of the number of jobs in December 1996 (or, in fact, at any other time). This ensures that the order of the labels in the key matches the positioning of the points for the different regions. Code in the footnote shows how the labeling on the *x*- and *y*-axes was obtained.

[5]
```
##  Panel A: Update trellis object to improve x- and y-axis tick labels
datelabpos <- seq(from=95, by=0.5, length=5)
datelabs <- format(seq(from=as.Date("1Jan1995", format="%d%b%Y"),
                       by="6 month", length=5), "%b%y")
## Now create $y$-labels that have numbers, with log values underneath
ylabpos <- exp(pretty(log(unlist(jobs[,-7])), 5))
ylabels <- paste(round(ylabpos),"\n(", log(ylabpos), ")", sep="")
update(simplejobsA.xyplot, xlab="",
       scales=list(x=list(at=datelabpos, labels=datelabs),
                   y=list(at=ylabpos, labels=ylabels)))
```

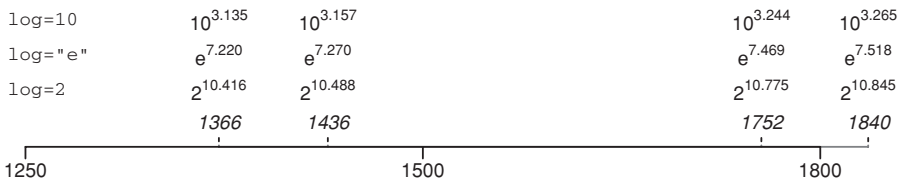


Figure 2.11 Labeling of the values for Alberta (1366, 1436) and Ontario (1752, 1840), with alternative logarithmic scale choices of labeling.

Because the labor forces in the various regions do not have similar sizes, it is impossible to discern any differences among the regions from this plot. Plotting on the logarithmic scale did not remedy this problem.[6]

Figure 2.10B shows a much preferable alternative. The six different panels use different *slices* of the same logarithmic scale. Here is simplified code for Figure 2.9B. The regions are again taken in the order of numbers of jobs in December 1996.[7]

```
## Simplified code for Figure 2.9B
xyplot(Ontario+Quebec+BC+Alberta+Prairies+Atlantic ~ Date,
       data=jobs, type="b", layout=c(3,2), ylab="Number of jobs",
       scales=list(y=list(relation="sliced", log=TRUE)),
       outer=TRUE)
```

Use of `outer=TRUE` ensures that the separate columns (regions) are plotted on separate panels. Equal distances on the scale now correspond to equal relative changes. It is now clear that Alberta and BC experienced the most rapid job growth during the period, and that there was little or no job growth in Quebec and the Atlantic region.

Even better, particularly if ready comprehension is important, would be to standardize by dividing, e.g., by the respective number of persons aged 15 years and over at that time. Exercise 11 at the end of the chapter explores this approach.

*Small proportional changes, on a scale of natural logarithms

Tick marks have been placed a distance 0.02 apart on a scale of natural logarithms or $\log_e$. On a scale of natural logarithms a change of 0.02 is, to a close approximation, a 2% change.

*Tick positions and labeling, on a logarithmic scale

The following are the changes in numbers employed, in each of Alberta and Ontario, from January 1995 to December 1996. The changes are shown in actual numbers, and on scales of $\log_2$, $\log_e$, and $\log_{10}$. Figure 2.11 shows this graphically.

[6] Figure 2.10A might alternatively be plotted as a time series. For details, see Subsection 14.9.7.

[7] Subsection 15.5.2 has code that gives the labeling shown in Figure 2.10B.

		Increase		
	Rel. change	$\log_2$	$\log_e$	$\log_{10}$
Alberta (1366 to 1466; increase = 70)	1.051	0.072	0.050	0.022
Ontario (1752 to 1840; increase = 88)	1.050	0.070	0.049	0.021

From the beginning of 1995 to the end of 1996, Alberta increased by 70 from 1366 to 1436, which is a factor of 1436/1366 $\simeq$ 1.051. Ontario increased by 96 from 5239 to 5335, which is a factor of 1.050. The proper comparison is not between the absolute increases of 70 and 96, but between relative increases by factors of 1.05 and 1.018.

For lattice functions, the arguments `log=2` or `log="e"` or `log=10` are available. These use the relevant logarithmic axis labeling, as in Figure 2.11, for axis labels. In base graphics, with the argument `log="x"`, the default is to label in the original units.

An alternative, both for traditional and lattice graphics, is to enter the logged values, using whatever basis is preferred (2 or `"e"` or 10), into the graphics formula. Unless other tick labels are provided, the tick marks will then be labeled with the logged values for the relevant basis.

2.1.6 Scatterplots, broken down by multiple factors

Data, in the data frame `tinting` (*DAAG*), are from an experiment that examined the effects of the tinting of car windows on visual performance (data relate to Burns *et al.*, 1999). The main focus was on visual recognition tasks, where side window vision is important. Columns are:

- Variables `csoa` (critical stimulus onset asynchrony, i.e., the time in milliseconds required to recognize an alphanumeric target), `it` (inspection time, i.e., the time required for a simple discrimination task), and `age` (age to the nearest year).
- The ordered factor `tint` (levels `no`, `lo`, `hi`).
- Factors `target` (`locon`, i.e., low contrast; `hicon`, i.e., high contrast), `sex` (f = female, m = male), and `agegp` (`younger` = a younger participant, in the 20s; `older` = an older participant, in the 70s).

Each of 28 individuals was tested at each level of `tint`, for each of the two levels of `target`. In all there are four factors (`tint`, `target`, `sex`, and `agegp`) that might influence the values of `csoa` and `it`, and the relationship between them. Two of these (`tint` and `target`) take different values for the same individual, while the other two (`sex` and `agegp`) vary between individuals.

A first step might be to plot `csoa` against `it` for each combination of `sex` and `agegp`. Use of the argument `groups=target` results in the use of different symbols (in a black and white plot) or different colors, depending on whether the target is low contrast or high contrast. Also, we can ask for a key. The code is

```
(target.xyplot <-
  xyplot(csoa ~ it | sex*agegp, data=tinting, groups=target,
         auto.key=list(columns=2)))
```

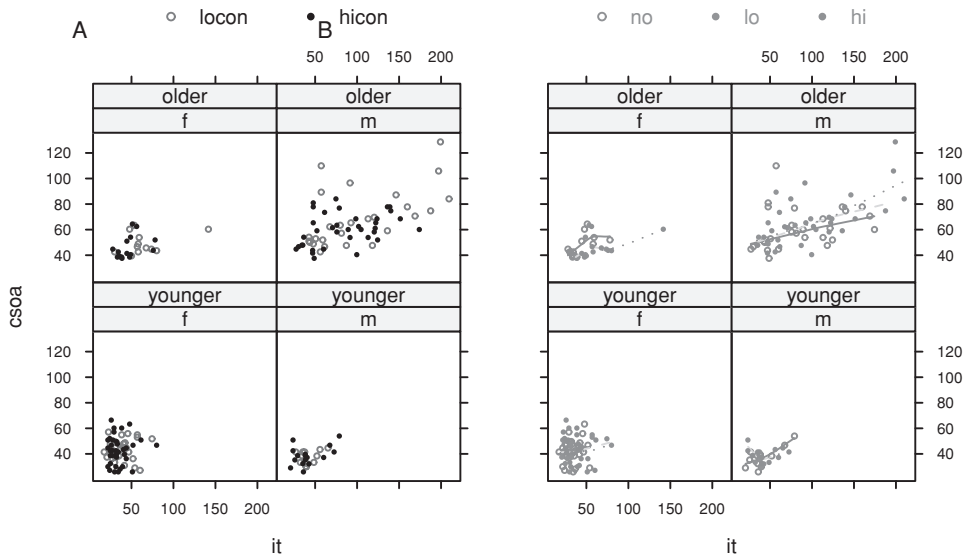


Figure 2.12 Panel A plots `csoa` against `it`, for each combination of `sex` and `agegp`. Different colors (gray and black) and symbols show different levels of `target`. Panel B shows the same points, but different colors (printed in grayscale) now show different levels of `tint`. Notice, also, the addition of smooth curves.

There are further possibilities for refinement. Figure 2.12A has used parameter settings that specify the choice of colors (here gray or black), plotting symbols, and the placement of the key.[8]

Observe that the longest times are for the high level of tinting. The relationship between `csoa` and `it` seems much the same for both levels of contrast. A number of older males have long response times with the low-contrast target. The analysis that will be presented later, in Chapter 10, indicates that within-subject effects – the effect of `tint` and `target` – stand up with greater clarity against the statistical noise than do effects of `sex` and `agegp`. The reason is that `tint` and `target` are effects that can be assessed within subjects, whereas the effects of `sex` and `agegp` involve a comparison across different subjects.

Because there are six points for each subject, Figure 2.12A gives a visual impression that exaggerates the evidence for effects that are associated with `sex` and `agegp`.

Fitting a trend curve, as in Figure 2.12B, makes the relationship clearer. The code for including the smooth curve, without the other refinements of Figure 2.12B, is:

[8]
```
## Settings used for Figure 2.12B (suitable for grayscale on a printed page)
update(target.xyplot,
       par.settings=simpleTheme(col=c("black","gray20"), pch=c(1, 16)))
 # In the above, par.settings changed settings for this use of xyplot()
## Note the use of simpleTheme() for changing settings; see help(simpleTheme)
## Use trellis.par.set() to change settings while the current device is in use
```

```
(tint.xyplot <-
  xyplot(csoa ~ it|sex*agegp, groups=tint, data=tinting,
       type=c("p","smooth"), span=1.25, auto.key=list(columns=3)))
 # "p": points; "smooth": a smooth curve
 # With span=1.25, the smooth curve is close to a straight line
```

The footnote adds the details needed to give Figure 2.12B.[9]

2.1.7 What to look for in plots

This is not a complete account of what plots may reveal! Its purpose is to draw attention to some of the more obvious possibilities.

Outliers

Outliers are points that appear to be isolated from the main body of the data. Such points (whether errors or genuine values) are liable to distort any model that we fit. What appears as an outlier depends, inevitably, on the view that is presented. On a fairly simple level, the view is affected by whether or not, and how, the data are transformed.

Boxplots, and the normal probability plot that will be discussed in Subsection 3.4.2, are useful for highlighting outliers in one dimension. Scatterplots may highlight outliers in two dimensions. Some outliers will be apparent only in three or more dimensions. The presence of outliers can indicate departure from model assumptions.

Asymmetry of the distribution

Most asymmetric distributions can be characterized as either positively skewed or negatively skewed. Positive skewness is the commonest form of asymmetry. There is a long tail to the right, values near the minimum are bunched up together, and the largest values are widely dispersed. Provided that all values are greater than zero, a logarithmic transformation typically makes such a distribution more symmetric. A distribution that is skew cannot be normal. Severe skewness is typically a more serious problem for the validity of results than other types of non-normality.

If values of a variable that takes positive values range by a factor of more than 10:1 then, depending on the application area context, positive skewness is to be expected. A logarithmic transformation should be considered.

Changes in variability

Boxplots and histograms readily convey an impression of the extent of variability or scatter in the data. Side-by-side boxplots such as in Figure 2.8B, or strip charts such as

[9]
```
## Panel B, with refinements
themeB <- simpleTheme(col=c("skyblue1", "skyblue4")[c(2,1,2)], lwd=c(1,1,2),
                      pch=c(1,16,16))  # open, filled, filled
update(tint.xyplot, par.settings=themeB, legend=NULL,
       auto.key=list(columns=3, points=TRUE, lines=TRUE))
 # Set legend=NULL to allow new use of auto.key
```

in Figure 2.8A, allow rough comparisons of the variability across different samples or treatment groups. They provide a visual check on the assumption, common in many uses of statistical models, that variability is constant across treatment groups.

Note, however, that it is easy to over-interpret such plots. Statistical theory offers useful and necessary warnings about the potential for such over-interpretation. (The variability in a sample, typically measured by the variance, is itself highly variable under repeated sampling. Measures of variability will be discussed in Subsection 2.2.3.)

When variability increases as data values increase, the logarithmic transformation will often help. If the variability is constant on a logarithmic scale, then the relative variation on the original scale is constant.

Clustering

Clusters in scatterplots may suggest structure in the data that may or may not have been expected. When we proceed to a formal analysis, this structure must be taken into account. Do the clusters correspond to different values of some relevant variable? Outliers are a special form of clustering.

Non-linearity

We should not fit a linear model to data where relationships are demonstrably non-linear. Often it is possible to transform variables so that terms enter into the model in a manner that is closer to linear. If not, the possibilities are wide-ranging, and we will canvass only a small number of them. See especially Chapter 7.

If there is a theory that suggests the form of model, then this is a good starting point. Available theory may, however, incorporate various approximations, and the data may tell a story that does not altogether match the available theory. The data, unless they are flawed, have the final say!

2.2 Data summary

Data summaries may: (1) be of interest in themselves; (2) give insight into aspects of data structure that may affect further analysis; (3) be used as data for further analysis. In case (3), it is necessary to ensure that important information, relevant to the analysis, is not lost. If no information is lost, the gain in simplicity of analysis can make the use of summary data highly worthwhile.

It is important, when data are summarized, not to introduce distortions that are artefacts of the way that the data have been summarized – examples will be given. The question of whether information in the data may have been lost or obscured has especial importance for the summarizing of counts across the margins of multi-way tables, and for the use of the correlation coefficient.

2.2.1 Counts

Data in the data frame `nswpsid1` (*DAAG* package) are derived from a study (Lalonde, 1986) that compared two groups of individuals with a history of unemployment problems – one an "untreated" control group and the other a "treatment" group whose members were exposed to a labor training program. Are the two groups genuinely comparable? This can

be checked by comparing them with respect to various measures other than their exposure (or not) to the labor training program.

Thus, what are the relative numbers in each of the two groups who had completed high school (`nodeg = 0`), as opposed to those who had not (`nodeg = 1`)?

```
> ## Table of counts example: data frame nswpsid1 (DAAG)
> tab <- with(nswpsid1, table(trt, nodeg, useNA="ifany"))
> dimnames(tab) <- list(trt=c("none", "training"),
+                       educ = c("completed", "dropout"))
> tab
          educ
trt        completed dropout
  none          1730     760
  training        80     217
```

Notice the use of the argument `useNA="ifany"` in the call to `table()`. This ensures that any `NA`s in either of the margins of the table will be tabulated.

The training group has a much higher proportion of dropouts. Similar comparisons are required for other factors and variables, examining joint as well as individual comparisons. These data will be investigated further in Section 13.2.

If `x1`, `x2`,..., `xn` are all columns (factors or vectors) of the same length and each is supplied as an argument to `table()`, the result is an *n*-way table. For example, `table(x1, x2, x3)` gives a three-way table. The first argument defines rows, though it is printed horizontally if there is just one column. The second argument defines columns. The table slices (rows by columns) that correspond to different values of the third argument appear in succession down the page, and so on.

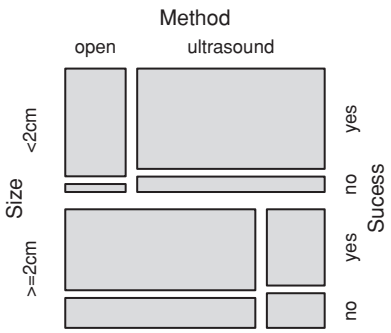
Addition over one or more margins of a table

Figure 2.13 illustrates the possible hazards of adding a multi-way table over one of its margins. Data are from a study (Charig, 1986) that compared outcomes for two different types of surgery for kidney stones; A: `open`, which used open surgery, and B: `ultrasound`, which used a small incision, with the stone destroyed by ultrasound.

Without additional information, the results are impossible to interpret. Different surgeons will have preferred different surgery types, and the prior condition of patients will have affected the choice of surgery type. The consequences of unsuccessful surgery may have been less serious for ultrasound than for open surgery.

Code that gives the mosaic plot is:

```
stones <- array(c(81,6,234,36,192,71,55,25), dim=c(2,2,2),
                dimnames=list(Success=c("yes","no"),
                              Method=c("open","ultrasound"),
                              Size=c("<2cm", ">=2cm")))
  # NB: The margins are 1:Success, 2:Method, 3:Size
library(vcd)
mosaic(stones, sort=3:1)  # c.f. mosaicplot() in base graphics
  # Re-ordering the margins gives a more interpretable plot.
```



		Success		
		yes	no	%Yes
Method	Size			
open	<2cm	81	6	93.1
	>=2cm	192	71	73.0
ultrasound	<2cm	234	36	86.7
	>=2cm	55	25	68.8
Add over Size				
open		273	77	78.0
ultrasound		289	61	82.6

Figure 2.13 Mosaic plot for the kidney stone surgery data that is shown to the right of the figure. Outcomes are for two different types of surgery for kidney stones. The overall (apparent) success rates (78% for open surgery as opposed to 83% for ultrasound) favor ultrasound. The success rate for each size of stone separately favors, in each case, open surgery.

Code that tabulates the data, giving a layout similar to that on the right of the plot, is:

```
## Function to calculate percentage success rates
roundpc <- function(x)round(100*x[1]/sum(x), 1)
## Add "%Yes" to margin 1 (Success) of the table
stonesplus <- addmargins(stones, margin=1, FUN=c("%Yes"=roundpc))
## Print table, use layout similar to that shown alongside plot
ftable(stonesplus, col.vars=1)
## Get sum for each margin 1,2 combination; i.e., sum over margin 3
stones12 <- margin.table(stones, margin=c(1,2))
stones12plus <- addmargins(stones12, margin=1, FUN=c("%Yes"=roundpc))
ftable(stones12plus, col.vars=1) # Table based on sums over Size
```

An alternative to `mosaic()` in the *vcd* package is `mosaicplot()` in base graphics. The function `mosaic()` is more flexible and extensible. A footnote demonstrates how the size of the text in the margins can be modified.[10]

Tabulation that accounts for frequencies or weights – the `xtabs()` *function*

The function `xtabs()` will be illustrated with a further example that, again, demonstrates the hazards of summarizing tabular or other data across factors that affect the frequencies of the margins that are retained.

Each year the National Highway Traffic Safety Administration in the USA uses a random sampling method, with sampling fractions that differ according to class of accident, to collect data from all police-reported crashes in which there is a harmful event (people or

[10]
```
## Add arguments that control size of textual components
mosaic(aperm(stones, 3:1), main=NULL, gp_varnames=gpar(fontsize=8),
       labeling_args=list(gp_labels=gpar(fontsize=7),
                          legend_args=list(fontsize=7)))
```

property), and from which at least one vehicle is towed. The data in `nassCDS` (*DAAG*) are restricted to front-seat occupants (*DAAG*).[11]

Factors whose effect warrant investigation include, as a minimum: `A: airbag` (was an airbag fitted?), `S: seatbelt` (was a seatbelt used?), and `dvcat` (F: a force of impact measure). The letters A, S, and F will be used as abbreviations when tables are generated.

The column `weight` (*national inflation factor*) holds the inverses of the sampling fraction estimates. The weight is designed to be the amount by which the contribution for the relevant row should be multiplied when tables of numbers of deaths and numbers of accidents are created. The following uses `xtabs()` to estimate numbers of front-seat passengers alive and dead, classified by airbag use:

```
> library(DAAG)
> ## NB: The parentheses generate an implicit print(abtab)
> (Atab <- xtabs(weight ~ airbag + dead, data=nassCDS))
        dead
airbag        alive       dead
  none   5445245.90   39676.02
  airbag 6622690.98   25919.11
```

The function `addmargins()` that was introduced above can be used to add the proportion of deaths in the right margin:

```
> roundpc2 <- function(x)round(100*x[2]/sum(x), 2)
> addmargins(Atab, margin=2, FUN=c("%Dead"=roundpc2))
        dead
airbag        alive       dead      %Dead
  none   5445245.90   39676.02       0.72
  airbag 6622690.98   25919.11       0.39
```

The above might suggest that the deployment of an airbag substantially reduces the risk of mortality. Consider, however:

```
> SAtab <- xtabs(weight ~ seatbelt + airbag + dead, data=nassCDS)
> ftable(addmargins(SAtab, margin=3, FUN=c("%Dead"=roundpc2)),
+        col.vars=3)
                dead      alive       dead      %Dead
seatbelt airbag
none     none        1342021.90   24066.65       1.76
         airbag       871875.39   13759.94       1.55
belted   none        4103224.00   15609.36       0.38
         airbag      5750815.59   12159.17       0.21
```

In the earlier table (`Atab`), the results without airbags were mildly skewed (4.12:1.37) to those for `belted`. Results with airbags were strongly skewed (57.6:8.86) to those for `none`, that is, no seatbelt.

[11] They hold a subset of the columns from a corrected version of the data analyzed in Meyer and Finney (2005). See also Farmer (2005) and Meyer (2006). More complete data are available from one of the web pages noted on the help page for `nassCDS`.

The reader may wish to try an analysis that accounts, additionally, for estimated force of impact (`dvcat`):

```
FSAtab <- xtabs(weight ~ dvcat + seatbelt + airbag + dead,
                data=nassCDS)
ftable(addmargins(FSAtab, margin=4, FUN=c("%Dead"=roundpc2)),
       col.vars=4)
```

The small differences that are now apparent, mostly unfavorable to airbags, are below any reasonable threshold of statistical detectability.

Farmer (2005) argues that these data, tabulated as above, have too many uncertainties and potential sources of bias to give reliable results. He presents a different analysis, based on the use of front-seat passenger mortality as a standard against which to compare driver mortality. Farmer's analysis was limited to cars without passenger airbags. In the absence of any effect from airbags, the ratio of driver mortality to passenger mortality should be the same, irrespective of whether or not there was a driver airbag. Farmer found a ratio of driver fatalities to passenger fatalities that was 11% lower in the cars with driver airbags.

In addition to the functions discussed, note the function `CrossTable()` from the *gmodels* package, which offers a choice of SPSS-like and SAS-like output formats.

2.2.2 Summaries of information from data frames

For obtaining summaries at combinations of different factor levels, the `aggregate()` function is often a good recourse. Note also the abilities of `aaply()` and allied functions in Hadley Wickham's *plyr* package.

Summary as a prelude to analysis – `aggregate()`

The data frame `kiwishade` (from *DAAG*) has yield measurements from 48 vines. Plots, made up of four vines each, were the experimental units. The analysis can be simplified by first taking means over plots that had four vines each.

The 12 plots were divided into three blocks of four plots each. One block of four was north-facing, a second block west-facing, and a third block east-facing. (Because the trial was conducted in the Southern hemisphere, there is no south-facing block.) Shading treatments were applied to whole plots, i.e., to groups of four vines, with each treatment occurring once per block. The shading treatments were applied either from August to December, December to February, February to May, or not at all. For more details of the experiment, look ahead to Figure 10.4.

For comparing treatments, there is no loss of information from basing analysis on the plot means. The four individual vine results that are averaged to give the plot mean are multiple measurements on the same experimental unit, here a plot.

Figure 2.14 plots both the aggregated means and the individual vine results. As treatments were applied to whole plots, the graph that shows the individual vine results exaggerates the extent of information that is available, in each block, for comparing treatments. For

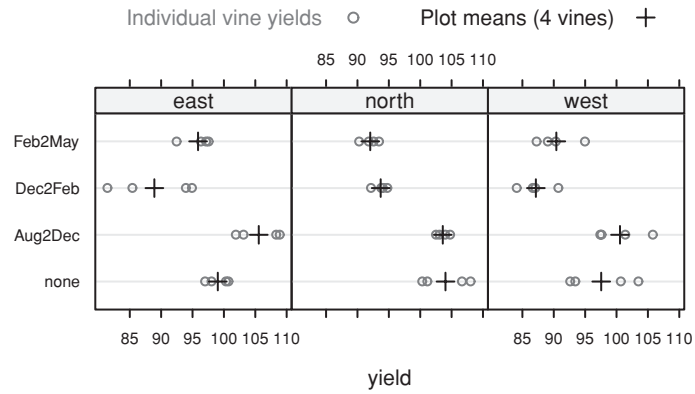


Figure 2.14 The four panels are the four different plots. The solid gray points are plot means. The open gray circles are yields for individual vines in the plot.

gaining a correct impression of the strength of the evidence, it is best to focus the eye on the means, shown as +. The code for Figure 2.14 is given as a footnote.[12]

The first few rows of the data frame are:

```
   yield block shade       plot
1 101.11 north  none north.none
2 108.02 north  none north.none
3 106.67 north  none north.none
4 100.30 north  none north.none
5  92.64  west  none  west.none
```

The `aggregate()` function splits the data frame according to the specified combinations of factor levels, and then applies a specified function to each of the resulting subgroups. Here, it forms a data frame that has the mean for each combination of block and shading treatment. The code, with the first line of output following, is:

```
> ## mean yield by block by shade: data frame kiwishade (DAAG)
> kiwimeans <- with(kiwishade,
+                   aggregate(yield, by=list(block, shade), mean))
> names(kiwimeans) <- c("block","shade","meanyield")
> head(kiwimeans, 4)
  block   shade meanyield
1  east    none   99.0250
> # . . .
```

[12]
```
## Individual vine means, by block and treatment
library(lattice)
## Panel function calls panel.dotplot(), then panel.average()
dotplot(shade ~ yield | block, data=kiwishade, aspect=1,
        panel=function(x,y,...){panel.dotplot(x, y, pch=1, col="gray40")
                                panel.average(x, y, type="p", col="black",
                                              pch=3, cex=1.25)},
        key=list(space="top", columns=2, col=c("gray40", "black"),
          text=list(c("Individual vine yields", "Plot means (4 vines)")),
          points=list(pch=c(1,3), cex=c(1,1.25))), layout=c(3,1))
 # Note that parameter settings were given both in the calls to the
 # panel functions and in the list supplied to key.
```

Use of the aggregated data for analysis commits us to working with plot means. What information is lost? If there were occasional highly aberrant values, use of medians might be preferable. The data should have a say in determining the form of summary.

The benefits of data summary – dengue status example

Hales *et al.* (2002) examined the implications of climate change projections for the world-wide distribution of dengue, a mosquito-borne disease that is a risk in hot and humid regions. Dengue status, i.e., information on whether dengue had been reported during 1965–1973, is available for 2000 administrative regions. Climate information is available on a much finer scale, on a grid of about 80 000 pixels at 0.5° latitude and longitude resolution. Should the analysis work with a data set that consists of 2000 administrative regions, or with the much larger data set that has one row for each of the 80 000 pixels? The following are reasons that might have argued for working with the summarized data:

- Dengue status is a summary figure that is given by administrative region. An analysis that uses the separate data for the 80 000 pixels will, in effect, predict dengue status for climate variable values that are in some sense averages for the administrative region. Explicit averaging, prior to the analysis, gives the user control over the form of averaging that will be used. If, for example, values for some pixels are extreme relative to other pixels in the administrative region, medians may be more appropriate than means. In some regions, the range of climatic variation may be extreme. The mean will give the same weight to sparsely populated cold mountainous locations as to highly populated hot and humid locations on nearby plains.
- Correlation between observations that are close together geographically, though still substantial, will be less of an issue for the data set in which each row is an administrative region. Points that repeat essentially identical information are a problem both for the interpretation of plots and, often, for the analysis. Regions that are geographically close will often have similar climates and the same dengue status.
- Analysis is more straightforward with data sets that are of modest size. It is easier to do standard forms of data checking. The points that appear on plots are more nearly independent. Standard forms of scatterplot less readily degenerate into a dense mass of black ink.

There are many possible ways to calculate a central value, of which the mean and the median are the most common. (In fact, however, the paper used the disaggregated data.)

2.2.3 Standard deviation and inter-quartile range

An important measure of variation in a population is the population standard deviation (often written σ), which is almost always unknown. The variance σ^2, which is the square of the standard deviation, is widely used in formal inference.

The sample standard deviation, used to estimate the population standard deviation when a random sample has been taken, is

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}.$$

Table 2.1 *Standard deviations for cuckoo egg data.*

Hedge sparrow	Meadow pipit	Pied wagtail	Robin	Tree pipit	Wren
1.049	0.920	1.072	0.682	0.880	0.754

In words, take the difference of each data value from the mean, square, add the squared differences together, divide by $n - 1$, and take the square root. In R, use the function `sd()` to calculate the standard deviation, or `var()` to calculate the variance. The standard deviation is in the same units as the original measurements. For s to be an accurate estimate of σ, the sample must be large.

Cuckoo eggs example

Consider again the data on cuckoo eggs that we discussed in Subsection 2.1.4. The group standard deviations are listed in Table 2.1.[13]

The variability in egg length is smallest when the robin is the host.

Degrees of freedom

The denominator $n - 1$ is the number of degrees of freedom remaining after estimating the mean. With one data point, the sum of squares about the mean is zero, the degrees of freedom are zero, and no estimate of the variance is possible. The degrees of freedom are the number of data values, additional to the first data value.

In later chapters, standard deviation calculations will be based on the variation that remains after fitting a model (most simply, a line) to the data. Degrees of freedom are reduced by 1 for each model parameter that is estimated.

Other measures of variability

The standard deviation is similar in concept to the inter-quartile range H, which we saw in Subsection 2.1.1 is the difference between the first and third quartiles. (The region between the lower and upper quartiles takes in 50% of the data.)

For data that are approximately normally distributed, note the approximate relationship

$$s \approx 0.75H.$$

If data are approximately normally distributed, one standard deviation either side of the mean takes in roughly 68% of the data.

Note also the median absolute deviation, calculated using the function `mad()`. This calculates the median of the absolute deviations from the median. By default this is multiplied by 1.4286, to ensure that in a large sample of normally distributed values the value returned should approximately equal the standard deviation.

[13]
```
## SD of length, by species: data frame cuckoos (DAAG)
sapply(split(cuckoos$length, cuckoos$species), sd)
 # Subsection 14.9.6 has information on split()
```

The pooled standard deviation

Consider two independent samples of sizes n_1 and n_2, respectively, randomly selected from populations that have the same amount of variation but for which the means may differ. Thus, two means must be estimated. The number of degrees of freedom remaining for estimating the (common) standard deviation is $n_1 + n_2 - 2$. We compute the so-called pooled standard deviation by summing squares of differences of each data value from their respective sample mean, dividing by the degrees of freedom $n_1 + n_2 - 2$, and taking the square root:

$$s_p = \sqrt{\frac{\sum(x - \bar{x})^2 + \sum(y - \bar{y})^2}{n_1 + n_2 - 2}}.$$

Use of this pooled estimate of the standard deviation is appropriate if variation in the two populations is plausibly similar. The pooled standard deviation is estimated with more degrees of freedom, and therefore, more accurately, than either of the separate standard deviations.

Elastic bands example

Consider data from an experiment in which 21 elastic bands were randomly divided into two groups, one of 10 and one of 11. Bands in the first group were immediately tested for the amount that they stretched under a weight of 1.35 kg. The other group were dunked in hot water at 65°C for four minutes, then left at air temperature for ten minutes, and then tested for the amount that they stretched under the same 1.35 kg weight as before. The results were:

Ambient: 254 252 239 240 250 256 267 249 259 269 (Mean = 253.5)

Heated: 233 252 237 246 255 244 248 242 217 257 254 (Mean = 244.1)

The pooled standard deviation estimate is $s = 10.91$, with 19 ($= 10 + 11 - 2$) degrees of freedom. Since the separate standard deviations ($s_1 = 9.92$; $s_2 = 11.73$) are similar, the pooled standard deviation estimate is an acceptable summary of the variation in the data.

2.2.4 Correlation

The usual Pearson or product–moment correlation is a summary measure of linear relationship. Calculation of a correlation should always be accompanied by a check that the relevant scatterplot shows a linear relationship. Often the addition of a smooth trend line helps the assessment. Check also that the marginal distributions of the two variables are roughly normal, or at least not highly skew. If the relationship is monotonic, but is not linear and/or has asymmetric marginal distributions, it may be appropriate to use a Spearman rank correlation. Examples of the needed code are:

```
> ## Correlation between body and brain: data frame Animals (MASS)
> ## Product--moment correlation
> with(Animals, cor(body, brain))
[1] -0.005341
> ## Product--moment correlation, after log transformation
> with(log(Animals), cor(body, brain))
```

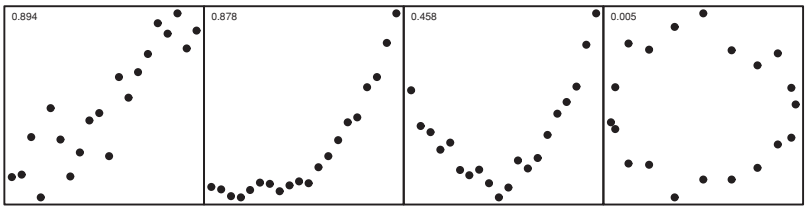


Figure 2.15 Different relationships between y and x. In the second panel, the Pearson correlation is 0.878, while the Spearman rank correlation is 0.928.

```
[1] 0.7795
>> ## Spearman rank correlation
> with(Animals, cor(body, brain, method="spearman"))
[1] 0.7163
```

The function `cor.test()` returns a confidence interval, and tests for no association.

Figure 2.15 gives four graphs to consider. For which does it make sense to calculate

1. A Pearson correlation coefficient?
2. A Spearman rank correlation?

The figure that appears in the upper left in each panel is the Pearson correlation. For the second panel, the Pearson correlation is 0.878, while the Spearman correlation, which better captures the strength of the relationship, is 0.928. Here a linear fit clearly is inadequate. The magnitude of the correlation r, or of the squared correlation r^2, does not of itself indicate whether the fit is adequate.

Note also the Kendall correlation, obtained by specifying `method="kendall"` when `cor.test()` is called. This is often used in contexts where the same individuals are assessed by different judges. It estimates the probability that the two judges will assign the same ranking to an individual.

Here are ways in which the use of correlation may mislead:

- The usual interpretation of the magnitude of the coefficient assumes that sample pairs (x, y) have been taken at random from a bivariate normal distribution. Observations must be independent, and the separate marginal distributions of x and y must be approximately normal. If, for example, the marginal distributions are highly asymmetric, the correlation is likely to be smaller, with increased statistical variability.
- There may be a subgroup structure in the data. If, for example, values of x and/or y are quite different for males and females, then the correlation may only reflect a difference between the sexes. Or, if random samples are taken from each of a number of villages and the data are pooled, then it will be unclear whether any correlation reflects a correlation between village averages or a correlation between individuals within villages, or a bit of each. The interpretation is confused because the two correlations may not be the same, and may even go in different directions. See Cox and Wermuth (1996).
- Any correlation between a constituent and a total amount is likely to be, in part at least, a mathematical artifact. Thus, consider a study of an anti-hypertensive drug that hopes to determine whether the change $y - x$ is larger for those with higher initial blood

pressure. If x and y have similar variances then, unfortunately, $y - x$ will have a negative correlation with x, whatever the influence of the initial blood pressure.

Note that while a correlation coefficient may sometimes be a useful single number summary of the relationship between x and y, regression methods offer a much richer framework for the examination of such relationships.

2.3 Statistical analysis questions, aims, and strategies

Logically, this section might have appeared at the beginning of the chapter, prior to any discussion of analysis methods and approaches. It is here because the reader should by now have a level of familiarity with several data sets that will be used as a basis for discussion.

Different questions, asked of the same data, will demand different analyses. Questions of interest may, given the available data, be unanswerable. Data on house prices in London may not have much relevance, if the interest is in house prices in New York or Paris! Questions should be structured with issues of this type in mind.

Questions should be structured with a view to the intended use of results. Is the aim scientific understanding, perhaps as in the example discussed below to determine whether cuckoos do really match the eggs that they lay in the nests of other birds to the size and color of the host eggs? Or is the aim prediction, perhaps to predict, based on recent prices in the area and on house size, the price that purchasers may be willing to pay?

Effective use of the information in the data

Figure 2.6 was designed to elicit the relationship between electrical resistance and apparent juice content, in kiwifruit. With the data we have, it would be bad practice to do a formal statistical test to compare, for example, juice content of less than 30% with juice content of more than 50%. Such an analysis would miss the relatively rich relationship that exists between the apparent juice content and resistance.

The data are more informative than would be obtained by doing repeated trials, some with a juice content of 30% and some with a juice content of 50%. Even worse would be a study comparing resistance at 40% juice content with resistance at 50% juice content. Such a study would have very little chance of finding anything of consequence.

Observational versus experimental data

Data from experiments appear throughout this book – examples are the data on the tinting of car windows that was used for Figure 2.12 in Subsection 2.1.6, and the kiwifruit shading data that were discussed in Subsection 2.2.2. Data from an experiment can, if well designed with a view to answering the questions of interest, give highly reliable results. With data from carefully designed experiments, perhaps the most serious danger is that the data will be generalized beyond the limits imposed by the experimental conditions.

Observational data are another matter. Section 13.2 will discuss a comparison between results from an experimental study on the effects of a work training program (those enrolled in the study were randomly assigned to training and non-training groups), and results from various sets of matched observational data that have been used in the attempt to answer the

Table 2.2 *Mean lengths of cuckoo eggs, compared with mean lengths of eggs laid by the host bird species. More extensive data that bear on the comparison between cuckoo eggs and host eggs are in the data frame* `cuckoohosts` *(DAAG).*

Host species	Meadow pipit	Hedge sparrow	Robin	Wagtails	Tree pipit	Wren	Yellow hammer
Length (cuckoo)	22.3 (45)	23.1 (14)	22.5 (16)	22.6 (26)	23.1 (15)	21.1 (15)	22.6 (9)
Length (host)	19.7 (74)	20.0 (26)	20.2 (57)	19.9 (16)	20 (27)	17.7 (–)	21.6 (32)

(Numbers in parentheses are numbers of eggs.)

same question. In this instance results from the use of observational data can be compared with those from an experiment in which individuals were randomly assigned either to an experimental or to a control group.

2.3.1 How relevant and how reliable are the data?

Latter (1902) collected the cuckoo egg data presented in Figure 2.8 in order to investigate claims, made in Newton (1893–1896, p. 123), that the eggs that cuckoos lay in the nests of other birds tend to match the eggs of the host bird in size, shape, and color. Figure 2.8 strongly indicated differences, depending on the host bird, in length. A further step is to look for a relationship between the mean size of the cuckoo eggs for each specific host, and the mean size of the host bird's own eggs, using data such as in Table 2.2.

There are various difficulties with the data in Table 2.2. The cuckoo eggs and the host eggs are from different nests, collected in the course of different investigations. Data on the host eggs are from various sources. For the wren, the value is an indicative length from Gordon (1894). There is thus a risk of biases, different for the different sources of data, that limit the inferences that can be drawn.

There is a striking difference between wrens and other species. Not only are their own eggs easily the smallest among the species considered; the eggs of the wren host are easily the smallest, among any of the hosts. Whatever biases may exist in the data, it is unlikely that they would be so large as to affect these major differences. Biases might well affect comparisons between eggs in the nests of species other than wrens.

2.3.2 How will results be used?

Studies may be designed to help scientific understanding. Consider again the data in Table 2.2. The interest of Latter's paper is primarily in establishing whether there is a relationship, and rather less in determining the nature of the relationship. Egg size and shape is one of several pieces of evidence that Latter considers. Uniquely among the birds listed, the architecture of wren nests makes it impossible for the birds to see the eggs. In wren nests, the color of the cuckoo's egg does not match the color of the wren's eggs; for the other species the color does mostly match. Latter concludes that Newton is right, that the eggs that cuckoos lay tend to match the eggs of the host bird in size and shape in ways that will make it difficult for hosts to distinguish their eggs from the cuckoo eggs.

This is very different from the demands of the real-estate agent who may hope, on the basis of last year's prices in a city location, and floor area, to predict the prices that purchasers will be willing to pay. Will a relationship that seems to work in one suburb apply also in another suburb, or in a neighboring city? Accurate prediction is crucial, with a perhaps reduced importance given to understanding the detailed reasons for any relationship that may be apparent. It is important, also, to know the intended use of the data and hence what sort of accuracy is important. Is it accuracy for purposes of making a prediction on one of the suburb(s) used in obtaining the data? Or is it accuracy for making predictions in new suburb(s)? In the latter case, data from multiple suburbs will be needed, and it must be possible to treat the sampled suburbs as a random sample from the suburbs for which predictions are required.

2.3.3 Formal and informal assessments

Statistical data analysis is, often, crucial to the answering of scientific questions. It does not however stand alone, but must be interpreted against a background of subject area knowledge and judgment. Qualitative judgments are inevitable at various points in studies that generate data sets such as are analyzed in this book. Such judgments affect the use of assumed subject area knowledge, the measurements that are taken, the design of data collection, the choice of analysis, and the interpretation of analysis results. These judgments, while they should be as informed as possible, cannot be avoided.

Two examples will now be given:

- In trials that evaluate therapies for conditions that commonly lead to early death, what is the relevant measure? Is it survival time from diagnosis? Or is it more appropriate to use a measure that takes account of quality of life over that time, which differs hugely between different therapies? Two such measures are "Disability Adjusted Life Years" (DALYs) and "Quality Adjusted Life Years" (QALYs).
- The dataset `nswpsid1` (see Subsection 2.2.1 and Section 13.2) allows comparison of two groups of individuals, both with a history of employment and related difficulties. A focus of interest was income in 1978, subsequent to the study. Because the distribution of income is highly skew, comparisons that are based directly on income will be biased towards the experience of those few individuals whose incomes were very large. This effect can be ameliorated by working with the logarithm of income. Or it might be more appropriate to compare the median salaries of the two groups, after adjusting for the effects of other variables.

In neither case is the interpretation of analysis results as simple as might initially appear. There is an inevitable risk that assumed insights and judgments will carry large elements of prejudice or personal bias. A well-designed study will allow some opportunity for study results to challenge the assumed insights and understandings that have motivated the study.

Questionnaires and surveys

The `science` and `socsupport` data frames (*DAAG*) are both from surveys. In both cases, an important question is whether the questions measured what they claimed to measure.

In the `science` data set, a focus of interest is the variable `like`, which measured the students' liking for science. What did students understand by "science"? Was science, for them, a way to gain and test knowledge of the world? Or was it a body of knowledge? Or, more likely, was it a name for their experience of science laboratory classes (smells, bangs, and sparks perhaps) and field trips?

In the `socsupport` data set, an important variable is Beck Depression Index or `BDI`. The Beck Depression Index is a standard psychological measure of depression (see, e.g., Streiner and Norman, 2003), derived from a carefully designed and tested questionnaire.

In either case it is impossible to escape the question: "What was measured?" This question is itself amenable to experimental investigation. For the data frame `science`, answers to other questions included in the survey shed some light. The Beck Depression Index is the result of an extensive process of development and testing that has seemed to validate its results, at least for populations on which it has been tested. Such background evidence helps in assessing what is measured. Finally, however, there must be a qualitative judgment that brings together subject area knowledge, background information and evidence, and the results of the immediate statistical analysis.

2.3.4 Statistical analysis strategies

We have emphasized the importance of careful initial scrutiny of the data. Techniques of a broadly EDA type have, in addition, a role in scrutinizing results from formal analysis, in checking for possible model inadequacies, and perhaps in suggesting remedies. In later chapters, we will discuss the use of diagnostic statistics and graphs in examination both of the model used and of output from the analysis. These are an "after the event" form of EDA. In the course of an analysis, the data analyst may move backwards and forwards between exploratory analysis and more formal analyses.

2.3.5 Planning the formal analysis

Planning the formal analysis is one aspect of planning a research study. Such advance planning should allow for the possibility of limited changes following preliminary investigation.

An ideal is to have available data, and perhaps also analysis results, from a suitably related earlier study. Use of such information to plan the analysis in advance reduces the chance of biasing the new results in a direction that is closest to the analyst's preference! Even so, graphical checks of the data should precede formal analysis. There may be obvious mistakes. The data may have surprises for the analyst.

If available at the beginning of the study, the information from the analysis of earlier data may, additionally, be invaluable in the design of data collection for the new study. When prior data are not available, a pilot study involving a small number of experimental runs can sometimes provide this kind of information.

Where it is not altogether clear what to expect, careful preliminary examination is even more necessary. In particular, the analyst should look for

- outliers,
- clusters in the data,
- unexpected patterns within groups,
- between-group differences in the scatter of the data,
- whether there are unanticipated time trends associated, e.g., with order of data collection.

In all studies, it is necessary to check for obvious data errors or inconsistencies. In addition, there should be checks that the data support the intended form of analysis.

2.3.6 Changes to the intended plan of analysis

What departures from the original plan are acceptable, and what are not? If the exploratory analysis makes it clear that the data should be transformed in order to approximate normality more closely, then use the transformation. It is sometimes useful to do both analyses (with the untransformed as well as with the transformed data) and compare them.

On the other hand, if there are potentially a large number of comparisons that could be made, the comparisons that will be considered should be specified in advance. Prior data, perhaps from a pilot study, can assist in this choice. Any investigation of other comparisons may be undertaken as an exploratory investigation, a preliminary to the next study.

Data-based selection of one or two comparisons from a much larger number is not appropriate, since huge biases may be introduced. Alternatively, there must be allowance for such selection in the assessment of model accuracy. The issues here are non-trivial, and we defer further discussion until later.

2.4 Recap

Exploratory data analysis aims to allow the data to speak for themselves, often prior to or as part of a formal analysis. It gives multiple views of the data that may provide useful insights. Histograms, density plots, stem-and-leaf displays, and boxplots are useful for examining the distributions of individual variables. Scatterplots are useful for looking at relationships two at a time. If there are several variables, the scatterplot matrix provides a compact visual summary of all two-way relationships.

Before analysis, look especially for

- outliers,
- skewness (e.g., a long tail) in the distribution of data values,
- clustering,
- non-linear bivariate relationships,
- indications of heterogeneous variability (i.e., differences in variability across samples),
- whether transformations seem necessary.

After analysis, check residuals for all these same features. Where relationships involve several variables, adequate checks will be possible only after analysis.

Failure of the independence assumption is hard to detect, unless the likely form of dependence is known and the sample is large. Be aware of any structure in the data that may be associated with lack of independence.

Do not allow the initial scrutiny of data to influence the analysis in ways that may lead to over-interpretation.

2.5 Further reading

The books and papers on graphical presentation that were noted in Chapter 1 are equally relevant to this chapter. The books Cleveland (1993, 1994) are especially pertinent to the present chapter. Chatfield (2002) has a helpful and interesting discussion, drawing on consulting experience, of approaches to practical data analysis.

On statistical presentation issues, and deficiencies in the published literature, see Andersen (1990), Chanter (1981), Gardner *et al*. (1983), Maindonald (1992), Maindonald and Cox (1984), Wilkinson and Task Force on Statistical Inference (1999). The Wilkinson *et al*. paper has helpful comments on the planning of data analysis, the role of exploratory data analysis, and so on. Nelder (1999) is forthright and controversial.

Two helpful web pages are

`http://www.math.yorku.ca/SCS/friendly.html#graph` and `http://www.rdg.ac.uk/ssc/publications/publications.html#under`

References for further reading

Andersen, B. 1990. *Methodological Errors in Medical Research: an Incomplete Catalogue.*

Chanter, D. O. 1981. The use and misuse of regression methods in crop modelling. In: *Mathematics and Plant Physiology*, eds. D. A. Rose and D. A. Charles-Edwards.

Chatfield, C. 2002. Confessions of a statistician. *The Statistician* 51: 1–20.

Cleveland, W. S. 1993. *Visualizing Data.*

Cleveland, W. S. 1994. *The Elements of Graphing Data*, revised edn.

Gardner, M. J., Altman, D. G., Jones, D. R. and Machin, D. 1983. Is the statistical assessment of papers submitted to the *British Medical Journal* effective? *British Medical Journal* 286: 1485–8.

Maindonald, J. H. 1992. Statistical design, analysis and presentation issues. *New Zealand Journal of Agricultural Research* 35: 121–41.

Maindonald, J. H. and Cox, N. R. 1984. Use of statistical evidence in some recent issues of DSIR agricultural journals. *New Zealand Journal of Agricultural Research* 27: 597–610.

Nelder, J. A. 1999. From statistics to statistical science. *Journal of the Royal Statistical Society, Series D* 48: 257–67.

Wilkinson, L. and Task Force on Statistical Inference. 1999. Statistical methods in psychology journals: guidelines and explanation. *American Psychologist* 54: 594–604.

2.6 Exercises

1. Use the lattice function `bwplot()` to display, for each combination of `site` and `sex` in the data frame `possum` (*DAAG* package), the distribution of ages. Show the different sites on the same panel, with different panels for different sexes.

2. Do a stem-and-leaf display for the lengths of the female possums. On the display, identify the position of the median, either at one of the leaves or between two leaves. Explain the reasoning used to find the median, and use the function `median()` to check the result.

3. Plot a histogram of the `earconch` measurements for the `possum` data. The distribution should appear *bimodal* (two peaks). This is a simple indication of clustering, possibly due to sex differences. Obtain side-by-side boxplots of the male and female `earconch` measurements. How do these measurement distributions differ? Can you predict what the corresponding histograms would look like? Plot them to check your answer.

4. For the data frame `ais` (*DAAG* package), draw graphs that show how the values of the hematological measures (red cell count, hemoglobin concentration, hematocrit, white cell count, and plasma ferritin concentration) vary with the sport and sex of the athlete.

5. Using the data frame `cuckoohosts`, plot `clength` against `cbreadth`, and `hlength` against `hbreadth`, all on the same graph and using a different color to distinguish the first set of points (for the cuckoo eggs) from the second set (for the host eggs). Join the two points that relate to the same host species with a line. What does a line that is long, relative to other lines, imply? Here is code that you may wish to use:

```
attach(cuckoohosts)
plot(c(clength, hlength), c(cbreadth, hbreadth),
     col=rep(1:2,c(12,12)))
for(i in 1:12)lines(c(clength[i], hlength[i]),
                    c(cbreadth[i], hbreadth[i]))
text(hlength, hbreadth, abbreviate(rownames(cuckoohosts),8))
detach(cuckoohosts)
```

6. Enter and run the following code. Annotate it, describing each function and each function argument:

```
deathrate <- c(40.7, 36,27,30.5,27.6,83.5)
hosp <- c("Cliniques of Vienna (1834-63)\n(> 2000 cases pa)",
    "Enfans Trouves at Petersburg\n(1845-59, 1000-2000 cases pa)",
    "Pesth (500-1000 cases pa)",
    "Edinburgh (200-500 cases pa)",
    "Frankfort (100-200 cases pa)", "Lund (< 100 cases pa)")
hosp <- factor(hosp, levels=hosp[order(deathrate)])
dotplot(hosp~deathrate, xlim=c(0,95), xlab="Death rate per 1000 ",
        type=c("h","p"))
## Source: \cite{Nightingale}. Data are ascribed to Dr Le Fort
```

7. The dataset `ex10.22`, on tomato yields, is available from the package *Devore6* (or in wide format from *Devore7*). Type

```
library(Devore6)      # ex10.22 is from Devore6
tomatoes <- ex10.22
```

This data frame gives tomato yields at four levels of salinity, as measured by electrical conductivity (`EC`, in nmho/cm).

   (a) Obtain a scatterplot of `yield` against `EC`.
   (b) Obtain side-by-side boxplots of `yield` for each level of `EC`.
   (c) The third column of the data frame is a factor representing the four different levels of `EC`. Comment upon whether the yield data are more effectively analyzed using `EC` as a quantitative or qualitative factor.

8. Examine the help for the function `mean()`, and use it to learn about the trimmed mean. For the total lengths of female possums, calculate the mean, the median, and the 10% trimmed

mean. How does the 10% trimmed mean differ from the mean for these data? Under what circumstances will the trimmed mean differ substantially from the mean?

9. Assuming that the variability in egg length for the cuckoo eggs data is the same for all host birds, obtain an estimate of the pooled standard deviation as a way of summarizing this variability. [*Hint:* Remember to divide the appropriate sums of squares by the number of degrees of freedom remaining after estimating the six different means.]

10. Calculate the following three correlations:
```
with(Animals, cor(brain,body))
with(Animals, cor(log(brain),log(body)))
with(Animals, cor(log(brain),log(body), method="spearman"))
```
Comment on the different results. Which is the most appropriate measure of the relationship?

11. Figure 2.10 showed changes in labor force numbers, in six regions of Canada, in successive quarters of 1995–1996. The population (in thousands) aged 15 years and over in each of these regions was, according to the 1996 census: `BC`: 3955; `Alberta`: 2055; `Prairies`: 1604; `Ontario`: 8249; `Quebec`: 5673; `Atlantic`: 1846. Plot a version of Figure 2.10 in which the labor force numbers are standardized by division by the number in the relevant population. Compare a plot that shows all regions in the same panel, with a plot that gives each region its own panel and its own slice of a common scale, commenting on the advantages and disadvantages of each. Is there now any reason to use a logarithmic scale?

12. The following code conveys information that has points of connection with the information in Figure 2.14:
```
bwplot(shade ~ yield|block, data=kiwishade, layout=c(3,1))
```
Compare and contrast the information given by these two plots.

13. The `galaxies` data in the *MASS* library gives speeds on 82 galaxies (see the help file and the references listed there for more information). Obtain a density plot for these data. Is the distribution strongly skewed? Is there evidence of clustering?

14. The `cpus` data frame in the *MASS* library contains information on eight aspects for each of 209 different types of computers. Read the help page for more information.

    (a) Construct a scatterplot matrix for these data. Should any of the variables be transformed before further analysis is conducted?

    (b) How well does estimated performance (`estperf`) predict performance (`perf`)? Study this question by constructing a scatterplot of these two variables, after taking logarithms. Do the plotted points scatter about a straight line or is there an indication of non-linearity? Is variability in performance the same at each level of performance?

3

Statistical models

Many regularities of nature are taken for granted in daily living – the rising and setting of the sun, the effects of fire in burning anyone unfortunate enough to get too near, and so on. Experience of the world, rather than logical deductive argument, has identified these regularities. Scientific investigation, especially in the physical sciences, has greatly extended and systematized awareness of regularities. Mathematical descriptions, i.e., models, have been crucial for describing and quantifying these regularities.

As when any model is pressed into service, it is important to understand which features generalize and which do not. An engineer's scale model of a building may be helpful for checking the routing of the plumbing but may give little indication of the acoustics of seminar rooms that are included in the building. In medical research, mouse responses to disease and to therapeutic agents are widely used as models for human responses. Experimental responses in the mouse may indicate likely responses in humans.

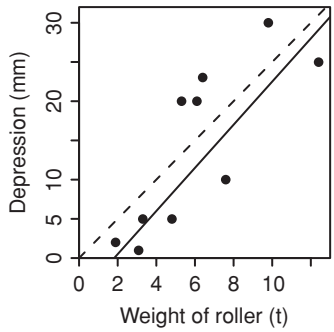
In fundamental research in the physical sciences, deterministic models are often adequate. Statistical variability may be so small that it can, for practical purposes, be ignored. In applications of the physical sciences, variability may more commonly be a serious issue. In studying how buildings respond to a demolition charge, there will be variation from one occasion to another, even for identical buildings and identically placed charges. There will be variation in which parts of the building break first, in what parts remain intact, and in the trajectories of fragments. In the natural sciences, such variability is everywhere.

Statistical models rely on probabilistic forms of description that have wide application over all areas of science. They often consist of a deterministic component, with a random component added that is designed to account for residual variation.

3.1 Statistical models

As we saw in Chapter 2, consideration of a model stays somewhat in the background in initial exploratory data analysis. The choice of model is crucial in formal analysis. The choice may be influenced by previous experience with comparable data, by subject area knowledge, and by cautious use of what may emerge from exploratory analysis.

Models should, wherever possible, be scientifically meaningful, but not at the cost of doing violence to the data. The scientific context includes the analyses, if any, that other researchers have undertaken with related or similar data. It can be important to note and use such analyses critically. While they may give useful leads, there can be serious inadequacies



	weight (t)	depression (mm)	depression / weight
1	1.9	2	1.1
2	3.1	1	0.3
3	3.3	5	1.5
4	4.8	5	1.0
5	5.3	20	3.8
6	6.1	20	3.3
7	6.4	23	3.6
8	7.6	10	1.3
9	9.8	30	3.1
10	12.4	25	2.0

Figure 3.1 Depression in lawn versus roller weight. Both lines were drawn by eye, with the dashed line constrained to go through the origin.

in published analyses. For further detail and discussion, see relevant articles and books in the list of references at the end of Chapter 2.

3.1.1 Incorporation of an error or noise component

Statistical models combine deterministic and random components. The random component is often called *noise* or *error* and the deterministic component is sometimes thought of as the *signal*. It may be helpful to think of the statistical error as the "rough", and of the model prediction as the "smooth".

The error component models variation that cannot be accounted for by given information. The simplest models assume that the elements of the error component are uncorrelated, i.e., the size and sign (negative or positive) of one element give no information on the likely size and sign of any other element.

Both *noise* and *error* are technical terms. Use of the word *error* does not imply that there have been mistakes in the collection of the data, though mistakes can of course contribute to making the variability unnecessarily large.

Figure 3.1 shows data from an experiment where different weights of roller were rolled over different parts of a lawn, and the depression noted (data are from Stewart *et al.*, 1988).[1] The data seem broadly consistent with the assumption of a signal by which depression is proportional to roller weight, as implied by the solid line in Figure 3.1. Variation about this signal is reflected in variation in the values for `depression/weight`. More generally, it might be assumed that the signal is a line that does not necessarily assume strict proportionality between depression and weight. The dashed line in Figure 3.1 is an example. It allows for a systematic error in the measurement of depression.

[1]
```
## Plot depression vs weight: data frame roller (DAAG)
plot(depression ~ weight, data = roller, xlim=c(0,1.04*max(weight)),
     ylim=c(0,1.04*max(depression)),
     xaxs="i", yaxs="i",    # "i"=inner: Fit axes exactly to the limits
     xlab = "Weight of roller (t)", ylab = "Depression(mm)", pch = 16)
abline(0, 2.25)             # A slope of 2.25 looks about right
```

The model has the form:

$$\text{observed value} = \text{model prediction} + \text{statistical error}$$

$$\text{i.e.,} \quad y = \mu + \varepsilon$$

where $\mu = \beta x$ (no intercept), or $\mu = \alpha + \beta x$ (with intercept). The model prediction (μ) is the signal component, while ε is the error component.

Substituting $\mu = \alpha + \beta \times$ weight in the equation:

$$\text{depression} = \alpha + \beta \times \text{weight} + \text{noise}.$$

Here α and β are constants which must be estimated. (For strict proportionality between depression and weight, α will of course be zero.)

Use of subscripts allows identification of the individual points. Given observations (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), we may write

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

Predicting with models

The focus of interest – generally prediction and/or interpretation of model parameters – may be different for different uses of model results. For the lawn roller data of Figure 3.1, one focus of interest is the rate of increase of depression with increasing roller weight, i.e., the slope of the line. Another focus is model prediction, i.e., the fitted values. Models should as far as possible yield inferences that, for their intended use, are acceptably accurate.

Model structure should reflect data structure. The model treats the pattern of change of depression with roller weight as a deterministic or *fixed* effect. The measured values of depression incorporate, in addition, a *random* effect that reflects variation from one part of the lawn to another, differences in the handling of the roller, and measurement error.

Data from multiple lawns are essential, for anything more than informal judgment on how results may generalize to other lawns. Such data would allow use of a model that accounts for between-lawn variation, as well as for the within-lawn variation on which our data give information. Chapter 10 will discuss models that might be tried, if data from multiple lawns were available.

Which model is best?

Figure 3.2 shows two possible models for the lawn roller data, together with information that may be helpful in assessing the adequacy of the model. In Figure 3.2A, a line (with intercept) has been fitted, while Figure 3.2B has used `lowess()` to fit a smooth curve through the data. Sometimes, the fitting of a curve such as in Figure 3.2B helps indicate whether a line really is appropriate. Note that there is just one point that seems to be causing the line, and the fitted curve, to bend down. In any case, there is no statistical justification for fitting a curve rather than a line, as can be verified with a formal analysis; see Exercise 2 in Chapter 7. There is "over-fitting" in Figure 3.2B.

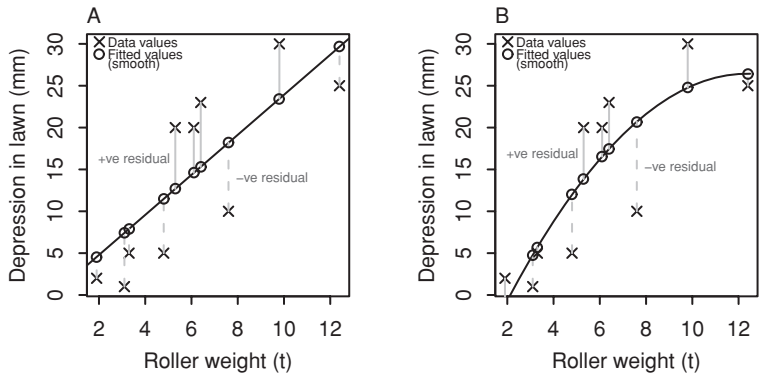


Figure 3.2 In panel A a line has been fitted, while panel B has a smooth curve. Residuals (the "rough") appear as vertical lines. Positive residuals are black lines, while negative residuals are dashed. Figures 3.2A and B were created using our function `g3.2()`, which is available from the web page for the book. Interested readers can check the code.

3.1.2 Fitting models – the model formula

Formulae have already been used extensively to describe graphs that will be plotted using `plot()` or another such function. Modeling functions likewise use formulae to describe the role of variables and factors in models. Thus, the following model statement, with model formula `depression ~ weight`, fits a line to the data of Figure 3.1:

```
## Fit line - by default, this fits intercept & slope.
## requires data frame roller (DAAG)
roller.lm <- lm(depression ~ weight, data=roller)
## Compare with the code used to plot the data
plot(depression ~ weight, data=roller)
## Add the fitted line to the plot
abline(roller.lm)
```

The name `roller.lm`, used for the output object, was chosen for mnemonic reasons – the object was the result of `lm` calculations on the `roller` data set. In the formula, `weight` is the *predictor* or *explanatory* variable, while `depression` is the *response*.[2]

Fitted values, residuals, and coefficients

Residuals, which are the differences between observed values of `depression`, and predicted values of depression at the respective values of `weight`, are important for assessing the accuracy of the model fit. A large error component generates large residuals, and works against accurate prediction. Figure 3.2A exhibits the residuals for the lawn roller data after fitting a straight line, while Figure 3.2B exhibits the residuals after fitting a smooth curve. Positive residuals are represented by solid lines, while negative residuals are represented by dashed lines.

[2] `## For a model that omits the intercept term, specify`
`lm(depression ~ -1 + weight, data=roller)`

Most of the information that is commonly required from model objects can be obtained by the use of an *extractor* function. For example, fitted values and residuals can be calculated with functions `fitted()` and `resid()`:

```
> round(fitted(roller.lm), 1)
   1    2    3    4    5    6    7    8    9   10
 3.0  6.2  6.7 10.7 12.0 14.2 15.0 18.2 24.0 31.0
> round(resid(roller.lm), 1)
   1    2    3    4    5    6    7    8    9   10
-1.0 -5.2 -1.7 -5.7  8.0  5.8  8.0 -8.2  6.0 -6.0
```

The `coef()` function gives the model coefficients:

```
> coef(roller.lm)
(Intercept)      weight
  -2.087148    2.666746
```

Model objects

The model object, above saved as `roller.lm`, is a list. Although not always recommended, we can access information in this list directly. For example, we can extract element names as follows:

```
> names(roller.lm)     # Get names of list elements
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

We can then extract information directly from a list element, such as the model coefficients:

```
> roller.lm$coef
(Intercept)      weight
  -2.087148    2.666746
```

For further discussion, see Subsection 14.10.2.

Summary information about model objects

To get a summary that includes coefficients, standard errors of coefficients, t-statistics, and p-values, type

```
summary(roller.lm)
```

3.2 Distributions: models for the random component

In this section, we briefly review the concepts of random variables and their distributions. Our discussion will focus on the more commonly used models for count data and continuous measurement data.

3.2.1 Discrete distributions – models for counts

Counts of events or numbers of objects are examples of *discrete random variables*. The possible values with their associated probabilities are referred to as a distribution. We consider three important examples: Bernoulli, binomial, and Poisson distributions.

Bernoulli distribution

Successive tosses of a fair coin come up tails with probability 0.5, and heads with probability 0.5, independently between tosses. If we let X take the value 1 for a head and 0 for a tail, then X is said to have a Bernoulli distribution with parameter $\pi = 0.5$.

More generally, we might consider an experiment or test with an uncertain outcome, but where the possibilities are "success" (or "1") and "failure" (or "0"). Success may occur with probability π, where $0 \leq \pi \leq 1$.

Binomial distribution

The sum of a number of independent Bernoulli random variables is called a binomial random variable. The number of successes in n independent tests (where success at each trial occurs with probability π) has a binomial distribution with parameters n and π.

The total number of heads in two tosses of a fair coin is a binomial random variable with $n = 2$ and $\pi = 0.5$. We can use the function `dbinom()` to determine probabilities of having 0, 1 or 2 heads in two coin tosses:[3]

```
## To get labeled output exactly as below, see the footnote
## dbinom(0:2, size=2, prob=0.5)    # Simple version
   0    1    2
0.25 0.50 0.25
```

On average, 25% of all pairs of coin tosses will result in no heads, 50% will have one head, and 25% will have two heads.

The number of heads in four coin tosses can be modeled as binomial with $n = 4$ and $\pi = 0.5$:

```
## dbinom(0:4, size=4, prob=0.5)
     0      1      2      3      4
0.0625 0.2500 0.3750 0.2500 0.0625
```

To calculate the probability of no more than two heads, add up the probabilities of 0, 1, and 2 ($0.0625 + 0.2500 + 0.3750 = 0.6875$). The function `pbinom()` can be used to determine such cumulative probabilities, thus:

```
pbinom(q=2, size=4, prob=0.5)
```

[3] `## To get the labeling (0, 1, 2) as in the text, specify:`
`probs <- dbinom(0:2, size=2, prob=0.5)`
`names(probs) <- 0:2`
`probs`

For another example, suppose a sample of 50 manufactured items is taken from an assembly line that produces 20% defective items, on average. To find the probability of observing no more than four defectives in a sample, use:

```
> pbinom(q=4, size=50, prob=0.2)
[1] 0.0185
```

The probability of observing fewer than five defectives in the sample is 0.0185.

The function `qbinom()` goes in the other direction, from cumulative probabilities to numbers of events; it is used to compute *quantiles*, a generalization of the more familiar term *percentiles*. To calculate a 70th percentile of the distribution of the number of heads in four coin tosses, type:

```
> qbinom(p = 0.70, size = 4, prob = 0.5)
[1] 3
```

Means and standard deviations

In four fair coin tosses, we *expect* to see two heads on average. In a sample of 50 manufactured items from a population where 20% are defective, we expect to see 10 defectives on average. In general, we can compute the *expected value* or *mean* of a binomial random variable using the formula $n\pi$.

The standard deviation is one way of summarizing the spread of a probability distribution; it relates directly to the degree of uncertainty associated with predicting the value of a random variable. High values reflect more uncertainty than low values. The formula $\sqrt{n\pi(1-\pi)}$ gives the standard deviation for a binomial random variable. The standard deviation of the number of heads in four coin tosses is 1, and for the number of defectives in our sample of 50 items, it is 2.83. In an absolute sense, we will be able to predict the number of heads more precisely than the number of defectives.

The variance is defined as the square of the standard deviation: for the binomial, it is $n\pi(1-\pi)$.

Poisson distribution

The Poisson distribution is often used to model the number of events that occur in a certain time interval or for the numbers of defects that are observed in items such as manufactured products.

The distribution depends on a parameter λ (the Greek letter "lambda"), which happens to coincide with the mean or expected value.

As an example, consider a population of raisin buns for which there are an average of three raisins per bun, i.e., $\lambda = 3$. Because of the mixing process, the number of raisins in a particular bun is uncertain; the possible numbers of raisins are 0, 1, 2, ... Under the Poisson model, we have the following probabilities for 0, 1, 2, 3, or 4 raisins in a bun:

```
## Probabilities of 0, 1, 2, 3, 4 raisins
## mean number of raisins per bun = 3
## dpois(x = 0:4, lambda = 3)
     0      1      2      3      4
0.0498 0.1494 0.2240 0.2240 0.1680
```

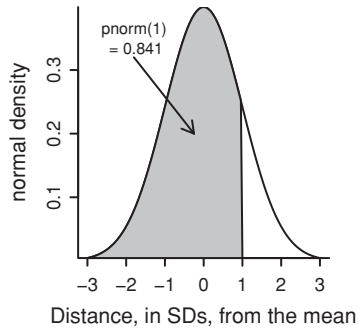


Figure 3.3 A plot of the normal density. The horizontal axis is labeled in standard deviations (SDs) distance from the mean. The area of the shaded region is the probability that a normal random variable has a value less than one standard deviation above the mean.

The cumulative probabilities are:

```
## ppois(q = 0:4, lambda = 3)
     0      1      2      3      4
0.0498 0.1991 0.4232 0.6472 0.8153
```

Thus, for example, the probability of finding two or fewer raisins in a bun is 0.4232.

The variance of a Poisson random variable is equal to its mean, i.e., λ. Thus, the variance of the number of raisins in a bun is 3, and the standard deviation is the square root of λ: 1.73.

3.2.2 Continuous distributions

Models for measurement data are examples of *continuous* distribution. Calculations with continuous distributions are a little different from calculations with discrete distributions. While we can still speak of probabilities of measurements lying in specified intervals, it is no longer useful to consider the probability of a measurement taking on a particular value. A more useful concept is *probability density*. A continuous random variable is summarized by its density function or curve. The area under any density curve between $x = a$ and $x = b$ gives the probability that the random variable lies between those limits.

Normal distribution

The normal distribution, which has the bell-shaped density curve pictured in Figure 3.3, is often used as a model for continuous measurement data (sometimes a transformation of the data is required in order for the normal model to be useful). The height of the curve is a function of the distance from the mean. The area under the density curve is 1.

The density curve plotted in Figure 3.3 corresponds to a normal distribution with a mean of 0 and standard deviation 1. A normal distribution having mean 0 and standard deviation 1 is referred to as the *standard* normal distribution. Multiplying a fixed value σ by a population of such normal variates changes the standard deviation to σ. By adding a fixed value μ, we can change the mean to μ, leaving the standard deviation unchanged.

Here is code that plots the normal density function:[4]

```
## Plot the normal density, in the range -3 to 3
z <- pretty(c(-3,3), 30) # Find ~30 equally spaced points
ht <- dnorm(z)           # By default: mean=0, standard deviation=1
plot(z, ht, type="l",xlab="Normal deviate",ylab="Density",yaxs="i")
 # yaxs="i" locates the axes at the limits of the data
```

The function `pnorm()` calculates the cumulative probability, i.e., the area under the curve up to the specified ordinate or x-value. For example, there is a probability of 0.841 that a normal deviate is less than 1:

```
> pnorm(1.0)          # by default, mean=0 and SD=1
[1] 0.841
```

This corresponds to the area of the shaded region in Figure 3.3.[5] The function `qnorm()` can be used to compute the normal quantiles. For example, the 90th percentile is 1.28:

```
> qnorm(.9)           # 90th percentile; mean=0 and SD=1
[1] 1.28
```

The footnote has additional examples.[6]

Other continuous distributions

There are many other statistical models for continuous observations. The simplest model is the uniform distribution, for which an observation is equally likely to take any value in a given interval; the probability density of values is constant on a fixed interval.

Another model is the exponential distribution that gives high probability density to positive values lying near 0; the density decays exponentially as the values increase. The exponential distribution is commonly used to model times between arrivals of customers to a queue. The exponential distribution is a special case of the chi-squared distribution. The latter distribution arises, for example, when dealing with contingency tables. Details on computing probabilities for these distributions can be found in the exercises.

[4] The following gives a closer approximation to Figure 3.3:

```
## Plot the normal density, in the range -3.25 to 3.25
z <- pretty(c(-3.25,3.25), 30) # Find ~30 equally spaced points
ht <- dnorm(z)                 # By default: mean=0, standard deviation=1
plot(z, ht, type="l", xlab="Normal deviate", ylab="Ordinate", yaxs="i")
polygon(c(z[z <= 1.0], 1.0), c(dnorm(z[z <= 1.0]), 0), col="grey")
 # Around 84.1% of the total area is to the left of the vertical line.
```

[5]

```
## Additional examples:
pnorm(0)             # .5    (exactly half the area is to the left of the mean)
pnorm(-1.96)         # .025
pnorm(1.96)          # .975
pnorm(1.96, mean=2)  # .484  (a normal distribution with mean 2 and SD 1)
pnorm(1.96, sd=2)    # .836  (sd = standard deviation)
```

[6]

```
## Additional examples:
qnorm(0.841)         # 1.0
qnorm(0.5)           # 0
qnorm(0.975)         # 1.96
qnorm(c(.1,.2,.3))   # -1.282 -0.842 -0.524  (10th, 20th and 30th percentiles)
qnorm(.1, mean=100, sd=10)  # 87.2 (10th percentile, mean=100, SD=10)
```

Different ways to describe distributions

In Subsection 2.1.1 it was noted that, with the default boxplot settings, 1% of values that are drawn at random from a normal distribution will on average be flagged as possible outliers. If the distribution is not symmetric, more than 1% of points may lie outside the whiskers, mostly at the lower end if the distribution is skewed (i.e., with a long tail) to the left, and mostly at the upper end if the distribution is skewed to the right. If the distribution is symmetric, but "heavy-tailed", then a higher proportion of values are out beyond the boxplot whiskers on both sides.

3.3 Simulation of random numbers and random samples

In a simulation, repeated random samples are taken from a specified distribution. Statistics, perhaps estimates that are derived from one or other model, can then be calculated for each successive sample. Information is thus obtained on variation under repeated sampling. This allows a check on results predicted by statistical theory. Or it may provide guidance when theoretical results are not available or are of uncertain relevance. This section will begin with simulation of discrete and continuous random variables and will close with a discussion of random sampling from finite populations.

Ordinarily, it is undesirable to use the same random number seed in two or more successive calls to a function that uses the random number generator. However, users will sometimes, for purposes of checking a calculation, wish to repeat calculations with the same sequence of random numbers as was generated in an earlier call. The following uses `set.seed()` to make the call below to `rbinom(10, size=1, p=.5)` thus reproducible:

```
set.seed(23286)  # Use to reproduce the sample below
rbinom(10, size=1, p=.5)
```

The seed for the random number generator is stored in the workspace in a hidden variable (`.Random.seed`) that changes whenever there has been a call to the random number generator. This ensures that any new simulation will be independent of earlier simulations.

When the workspace is saved, `.Random.seed` is stored as part of the workspace. This ensures that, when the workspace is loaded again, the seed will be restored to its value when the workspace was last saved. Any new simulations will then be independent of those prior to the save. In order to take advantage of this feature, be sure to save the workspace at the end of each session.

Sampling from discrete distributions

Values can be simulated from any of many different distributions. We will offer examples of simulated binomial, Poisson, and normal samples.

As a first example, simulate a random sequence of 10 binary digits (0s or 1s) from a population with a specified proportion of 1s, here 50% (i.e., a Bernoulli distribution):

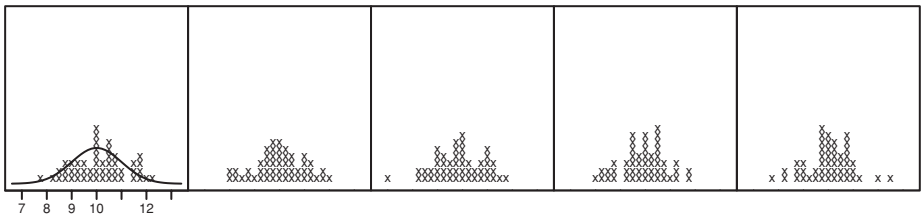


Figure 3.4 Each panel shows a simulated distribution of 50 values from a normal distribution with mean = 10 and sd = 1. The underlying theoretical normal curve is overlaid on the leftmost panel.

```
> rbinom(10, size=1, p=.5)    # 10 Bernoulli trials, prob=0.5
1 0 0 0 1 1 1 0 1 0
```

The random sample is different on each occasion, depending on the *seed*.

To generate the numbers of daughters in a simulated sample of 25 four-child families, assuming that males and females are equally likely, use the `rbinom()` function thus:

```
# For the sequence that follows, precede with set.seed(9388)
> rbinom(25, size=4, prob=0.5)
 [1] 3 1 2 4 1 2 0 3 2 1 2 3 2 4 2 1 1 1 2 2 3 2 0 2 2
```

Now simulate the number of raisins in 20 raisin buns, where the expected number of raisins per bun is 3:

```
> set.seed(9388)
> rpois(20, 3)
 [1] 3 3 4 1 2 2 3 1 1 4 3 0 1 1 3 1 0 4 5 2
```

3.3.1 Sampling from the normal and other continuous distributions

The function `rnorm()` generates random deviates from the normal distribution. To generate 10 random values from a standard normal distribution, we type:

```
> options(digits=2) # Suggest number of digits to display
> rnorm(10)         # 10 random values from the normal distribution
                    # For our sequence, precede with set.seed(3663)
 [1] -0.599 -1.876 1.441 -1.025 0.612 -1.669 0.138 -0.099 1.010 0.013
```

Figure 3.4 demonstrates the use of simulation to indicate the extent of sample-to-sample variation in histogram summaries of the data, when five independent random samples of 50 values are taken from a normal distribution. Figure 3.4 shows histograms from five such samples.[7] Histograms do not discriminate well between sample values that are consistent

[7]
```
## The following gives a rough equivalent of the figure:
set.seed (21)        # Use to reproduce the data in the figure
par(mfrow=c(2,3))
x <- pretty(c(6.5,13.5), 40)
for(i in 1:5){
    y <- rnorm(50, mean=10, sd=1)
    hist(y, prob=TRUE, xlim=c(6.5,13.5), ylim=c(0,0.5), main="")
    lines(x, dnorm(x,10,1))
    }
par(mfrow=c(1,1))
```

with a normal distribution, and sample values that are not. A better tool for assessing normality, the normal probability plot, will be described in Subsection 3.4.2.

Calculations for other distributions, for example `runif()` to generate uniform random numbers or `rexp()` to generate exponential random numbers, follow the same pattern.

```
runif(n = 20, min=0, max=1) # 20 numbers, uniform distn on (0, 1)
rexp(n=10, rate=3)          # 10 numbers, exponential, mean 1/3.
## Exercises at the end of this chapter explore further possibilities.
```

3.3.2 Simulation of regression data

The following code shows how to simulate a sample of n observations from the model:

$$y = b_0 + b_1 x + \varepsilon$$

where ε is normally distributed with standard deviation σ. We take $n = 8$, the intercept to be 2, the slope to be 3, and σ to be 2.5 in our simulation, and we use a fixed equally spaced design for the predictor values:

```
> options(digits=3)
> n <- 8;  x <- seq(1,n); sigma <- 2.5; b0 <- 2;  b1 <- 3
> error <- rnorm(n, sd=sigma)
> y <- b0 + b1*x + error
>
> t(data.frame(x,y))
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
x 1.00 2.00 3.00  4.0  5.0    6  7.0  8.0
y 6.85 8.96 9.49 12.2 19.1   20 26.2 28.5
```

It is often useful to repeatedly simulate data from a fitted model, then re-fitting to each new set of simulated data. This provides a check on variation under such repeated simulation. The function `simulate()` can be used for this purpose.

Thus to do 10 simulations based on the model that was fitted to the `roller` data, do:

```
roller.lm <- lm(depression ~ weight, data=roller)
roller.sim <- simulate(roller.lm, nsim=20)  # 20 simulations
```

The object `roller.sim` is a data frame with 20 columns, i.e., one column for each of the 20 simulations. Each column has values of depression, simulated from the fitted model at each level of weight. To visualize this output, enter

```
with(roller, matplot(weight, roller.sim, pch=1, ylim=range
(depression)) points(roller, pch=16)
```

3.3.3 Simulation of the sampling distribution of the mean

The sampling distribution of the mean is the distribution of the means of repeated random samples of size n. The standard deviation of this sampling distribution has the name *standard error of the mean* (SEM). If the population mean is μ and the standard deviation

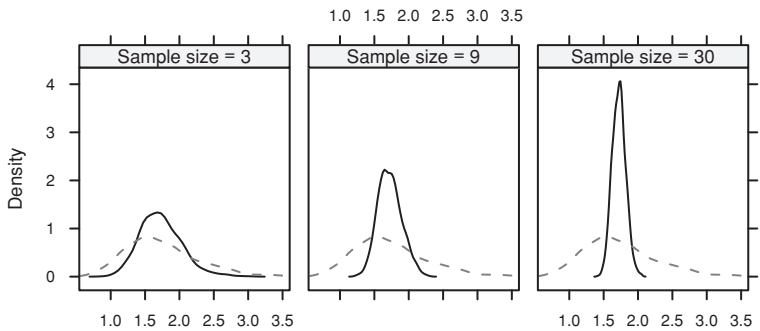


Figure 3.5 The density curves are from simulations of the sampling distribution of the mean, for a distribution that is mildly skew. Each density curve is from 1000 simulations, which is large enough to give an accurate visual indication of the distribution.

is σ, then

$$\text{SEM} = \frac{s}{\sqrt{n}}.$$

The Central Limit Theorem will be important in the discussion of inference in Chapter 4. This theorem implies that, for large enough n, this sampling distribution will closely approximate the normal.[8] The sample size n needed so that the normal is a good approximation will depend on the distribution of the population from which samples are taken.

Figure 3.5 shows the simulated sampling distribution of the mean, for samples from a distribution that is mildly skew. Even for a sample size of 3, much of the skewness has gone. Code that generates samples from the sampling distribution of means of the required sample sizes is:

```
## Function to generate n sample values; skew population
sampvals <- function(n)exp(rnorm(n, mean = 0.5, sd = 0.3))
## Means across rows of a dimension nsamp x sampsize matrix of
## sample values gives nsamp means of samples of size sampsize.
samplingDist <- function(sampsize=3, nsamp=1000, FUN=mean)
  apply(matrix(sampvals(sampsize*nsamp), ncol=sampsize), 1, FUN)
size <- c(3,10,30)
## Simulate means of samples of 3, 9 and 30; place in dataframe
df <- data.frame(y3=samplingDist(sampsize=size[1]),
                 y9=samplingDist(sampsize=size[2]),
                 y30=samplingDist(sampsize=size[3]))
```

The following then gives a slightly simplified version of Figure 3.5:

```
## Simulate source population (sampsize=1)
y <- samplingDist(sampsize=1)
densityplot(~y3+y9+y30, data = df, outer=TRUE, layout = c(3,1),
            plot.points = FALSE, panel = function(x, ...) {
```

[8] More precisely, the distribution of the sample mean approximates the normal distribution with arbitrary accuracy, for a sample that is large enough, provided the measurements are independent, and their standard deviation is finite. There are similar results for a number of other sample statistics.

```
    panel.densityplot(x, ..., col = "black")
    panel.densityplot(y, col = "gray40", lty = 2, ...)
  })
```

Code that will reproduce the strip panel labels is in the footnote.[9] The plots can alternatively be obtained using the function `sampdist()` (*DAAG*), with default arguments. The skewness of the population can be increased by increasing `sd` in the call to `sampvals()`.

3.3.4 Sampling from finite populations

We can use the `sample()` function to generate a simple random sample from a given set of numbers. Suppose, for example, that names on an electoral roll are numbered from 1 to 9384. We can obtain a random sample of 15 individuals as follows:

```
> ## For the sequence below, precede with set.seed(3676)
> sample(1:9384, 15, replace=FALSE)
[1] 9178 2408 8724  173  106 4664 3787 6381 5098 3228 8321
165 7332 9036  540
```

This gives the numerical labels for the 15 individuals that we should include in our sample. The task is then to find them! The option `replace=FALSE` gives a *without-replacement* sample, i.e., it ensures that no one is included more than once.

To randomly assign 10 plants (labeled from 1 to 10, inclusive) to one of two equal-sized groups, control and treatment, the following code could be used:

```
> ## For the sequence below, precede with set.seed(366)
> split(sample(seq(1:10)), rep(c("Control","Treatment"), 5))
>  # sample(1:10) gives a random re-arrangement (permutation)
>  # of 1, 2, ..., 10
$Control
[1] 6 8 3 7 9

$Treatment
[1]  5  4  2  1 10
```

We then assign plants 6, 8, 3, 7, and 9 to the control group. By choosing the plants in such a manner, we avoid biases that could arise, for example, due to choosing healthier-looking plants for the treatment group.

With-replacement samples

We can randomly sample from the set $\{1, 2, \ldots, 10\}$, allowing for repeated observations, by using:

```
> sample(1:10, replace=TRUE)
 [1] 7 5 2 1 2 3 1 5 7 6
```

[9]
```
## Use strip.custom to customize the strip labeling
doStrip <- strip.custom(strip.names = TRUE, factor.levels = as.expression(size),
                        var.name = "Sample size", sep = expression(" = "))
## Then include the argument 'strip=doStrip' in the call to densityplot
```

Cluster sampling

Cluster sampling is one of many different probability-based variants on simple random sampling. See Barnett (2002). In surveys of human populations cluster-based sampling, e.g., samples of households or of localities, with multiple individuals from each chosen household or locality, is likely to introduce a cluster-based form of dependence. The analysis must then take account of this clustering. Standard inferential methods require adaptation to take account of the fact that it is the clusters that are independent, not the individuals within the clusters. Donner and Klar (2000) describe methods that are designed for use in health research.

Simulation in teaching and research

The R package *animation* (Xie and Cheng, 2008) has a number of simulations that are intended for use in teaching or self-instruction. In statistical theory and practice, simulation is widely used to determine the statistical properties of models and/or of model statistics in cases where it has not been possible to derive analytical results.

3.4 Model assumptions

Common model assumptions are normality, independence of the elements of the *error* term, and homogeneity of variance (i.e., the standard deviations of all measurements are the same).

If certain assumptions fail to hold, a statistical method may be invalid. Other assumptions may not be as important; we say that the method used is *robust* against those assumptions. Much of the art of applied statistics comes from knowing which assumptions are important and need careful checking. There are few hard and fast rules.

3.4.1 Random sampling assumptions – independence

Typically, the data analyst has a sample of values that will be used as a window into a wider population. Ideally, data should be gathered in such a way that the independence assumption is guaranteed. This is why randomization is so important in designed experiments, and why random sampling is so important in designed sample surveys.

Elementary analysis methods can be modified or extended in various ways to handle modifications of the simple independent random sampling scheme. For example, we can modify the methodology to handle analyses of data from a random sample of clusters of individuals.

In practice, analysts may make the random sampling assumption when the selection mechanism does not guarantee randomness. Inferences from data that are chosen haphazardly are inevitably less secure than when we have random samples. Random selection avoids the conscious or unconscious biases that result when survey or other samplers make their own selection, or take whatever items seem suitable.

Where there has not been explicit use of a random sampling mechanism, it is necessary to consider carefully how this may have affected the data. Is some form of dependence structure likely? Temporal and spatial dependence arise because values that are close together in time

or space are relatively more similar. Is there clustering that arises because all individuals within chosen streets, or within chosen families, have been included. Two individuals in the same family or in the same street may be more similar than two individuals chosen at random from the same city suburb.

Often samples are chosen haphazardly, e.g., an experimenter may pick a few plants from several different parts of a plot. Or a survey interviewer may, in a poor-quality survey, seek responses from individuals who can be found in a shopping center. Self-selected samples can be particularly unsatisfactory, e.g., those readers of a monthly magazine who are sufficiently motivated to respond to a questionnaire that is included with the magazine.

Failure of the independence assumption is a common reason for wrong statistical inferences. Detecting failure of the independence assumption is often difficult. Tests for independence are at best an occasionally useful guide. They are of little use unless we have some idea how the assumption may have failed, and the sample is large! It is in general better to try to identify the nature of any possible dependence, and use a form of analysis that allows for it.

Models that do not obviously reflect mechanisms that generated the data can sometimes be useful for prediction. They can also, if their deficiencies are not understood or if they are used inappropriately, be misleading. Careful checking that the model is serving its intended purpose, and caution, are necessary.

3.4.2 Checks for normality

Many data analysis methods rest on the assumption that the data are normally distributed. Real data are unlikely to be exactly normally distributed.

Broadly, gross departures from normality are a cause for concern. Small departures are of no consequence. Check especially for data that are skew. Check also for data that take a small number of discrete values, perhaps as a result of excessive rounding. Whether a specific form of departure will matter depends on the use made of the data.

For modest-sized samples, only gross departures will be detectable. For small samples (e.g., less than about 10), it is typically necessary to rely on sources of evidence that are external to the data, e.g., previous experience with similar data.

Graphical tools for checking for normality

As noted in Subsection 2.1.1, histograms are not an effective means for assessing whether the distribution is plausibly normal. Refer back to the five histograms shown in Figure 3.4, from five independent random samples of 50 values from a normal distribution. None of these histograms showed a close resemblance to a theoretical normal distribution.

The normal probability plot

A better tool for assessing normality is the normal probability (or quantile–quantile) plot. The data values are sorted, then plotted against the ordered values that might be expected if the data really were from a normal distribution. If the data are from a normal distribution, the plot should approximate a straight line. Figure 3.6 shows normal probability plots for

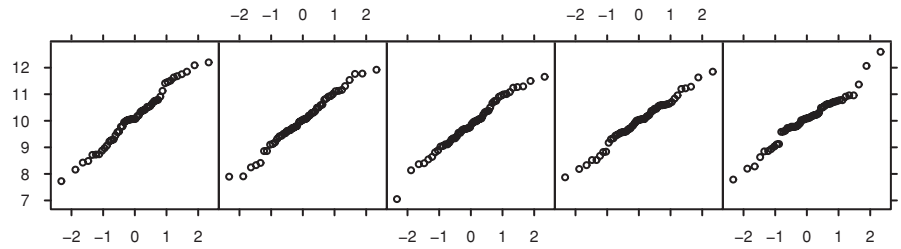


Figure 3.6 Normal probability plots for the same random normal data as in Figure 3.4.

the same five sets of 50 normally distributed values as were displayed in Figure 3.4. The code is:

```
## Use qreference() (DAAG)
## With seed=21, the random numbers are as in the previous figure
qreference(m=50, seed=21, nrep=5, nrows=1)  # 50 values per panel
```

An alternative is to use the lattice function `qqmath()`.[10] To obtain a single plot of this type, the function `qqnorm()`, which relies on functions from base graphics, may be used. Specify, e.g., `qqnorm(rnorm(50))`.

Displays such as Figure 3.6 help the data analyst to calibrate the eye, to get a feel for the nature and extent of departures from linearity that are to be expected in random normal samples of the specified size, here 50. It is useful to repeat the process several times. Such plots give a standard against which to compare the normal probability plot for the sample.

Note that, by plotting against the ordered values that might be expected from the relevant distribution, the methodology allows a comparison with any distribution that is of interest.

The sample plot, set alongside plots for random normal data

Consider data from an experiment that tested the effect of heat on the stretchiness of elastic bands. Eighteen bands were first tested for amount of stretch under a load that stretched the bands by a small amount (the actual load was 425 g, thought small enough not to interfere with the elastic qualities of the bands). This information was used to arrange bands into nine pairs, such that the two members of a pair had similar initial stretch. One member of each pair, chosen at random, was placed in hot water (60–65 °C) for four minutes. The other member of the pair remained at ambient temperature. All bands were then measured for amount of stretch under a load of 1.35 kg weight. Table 3.1 shows the results.

In the next chapter, the heated–ambient differences will be the basis for various statistical calculations. Is the distribution consistent with the assumption of normality? The normal probability plot for these data is in the lower left panel of Figure 3.7. The other seven plots are for samples (all of size 9) of simulated random normal values. They give a standard against which to compare the plot for the experimental data. There is no obvious feature that distinguishes the plot in the lower left panel from the seven reference plots. The code is:

[10]
```
## Set seed to get the same data as earlier
library(lattice)
qqmath(~rnorm(50*5)|rep(1:5,rep(50,5)), layout=c(5,1), aspect=1)
```

Table 3.1 *Eighteen elastic bands were divided into nine pairs, with bands of similar stretchiness placed in the same pair. One member of each pair was placed in hot water (60–65 °C) for four minutes, while the other was left at ambient temperature. After a wait of about 10 minutes, the amounts of stretch, under a 1.35 kg weight, were recorded.*

	Pair #								
	1	2	3	4	5	6	7	8	9
Heated (mm)	244	255	253	254	251	269	248	252	292
Ambient	225	247	249	253	245	259	242	255	286
Difference	19	8	4	1	6	10	6	−3	6

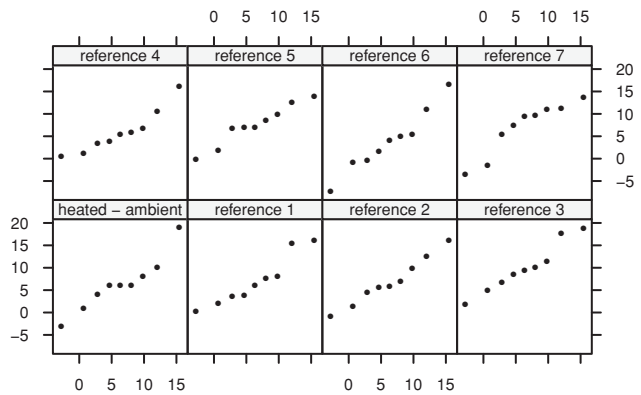


Figure 3.7 The lower left panel is the normal probability plot for heated–ambient differences. Remaining panels show plots for samples of nine numbers from a normal distribution.

```
## Compare normal probability plot for normal-ambient difference
## with simulated normal values: data frame pair65 (DAAG)
qreference(pair65$heated - pair65$ambient, nrep=8)
```

The function `qreference()` is from the *DAAG* package.

How close to normal is the sampling distribution of the mean?

Often, the interest is in the normality of the sampling distribution of a mean or other statistic. Figure 3.8 simulates repeated sampling from the same mildly skew distribution as in Figure 3.5. Instead of density curves, the normal probability plots are shown. The normal probability plot for a sample from the population is the gray dashed line. This plot can be obtained by replacing `densityplot()` by `qqmath()` in the code for Figure 3.5. Alternatively, use the function `sampdist()` (*DAAG*), with the argument `plot.type="qq"`.

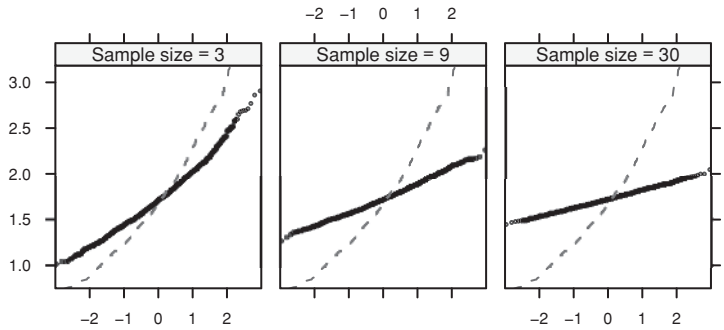


Figure 3.8 The normal probability plots are from simulations of the sampling distribution of the mean, for the same mildly skew distribution as in Figure 3.5. The plot for the population is shown in gray. The panels show the plots for samples of respective sizes 3, 9, and 30. Each is from 1000 simulations.

Notice that the plot for the sampling distribution is increasingly linear, with a reduced slope, as one goes from $n = 3$ to $n = 9$ to $n = 30$. The reduced slope reflects the reduced SEM, which goes from $\frac{\sigma}{\sqrt{3}}$ to $\frac{\sigma}{\sqrt{9}}$ to $\frac{\sigma}{\sqrt{30}}$.

Formal statistical testing for normality?

Both formal statistical tests for normality and less formal graphical checks are of limited usefulness. With small or modest-sized samples, only gross departures are likely to be detected. Large samples will show departures from normality that may be too small to have any practical consequence for standard forms of statistical analysis. For a statistic such as a mean or a regression slope, the effects of averaging may give a close approximation to normality, even when the underlying population is clearly not normally distributed. (This is a consequence of the "Central Limit Theorem" that was discussed in Subsection 3.3.3.)

Depending then on the specific context, normality may not be an important issue for analyses where samples are large. Tests for normality will detect non-normality in contexts where there is the least reason to be concerned about it.

3.4.3 Checking other model assumptions

In Chapter 2, we discussed a number of exploratory techniques that can aid in checking whether the standard deviation is the same for all observations in a data set. Following analysis, a plot of residuals against fitted values may give useful indications. For example, residuals may tend to fan out as fitted values increase, giving a "funnel" effect, a fairly sure sign that the standard deviation is increasing. Alternatively, or additionally, there may be evidence of outliers – one or more unusually large residuals. The major concern may however be to identify points, whether or not outliers, that have such high *influence* that they distort model estimates.

3.4.4 Are non-parametric methods the answer?

Non-parametric methods have been developed to handle situations where normality or other model assumptions are in question, and where it might be difficult to pose an alternative

model. These methods are only sometimes useful, and they still depend on assumptions, and we still need assurance that these assumptions are realistic. If used in a way that ignores structure in the data that we should be modeling, we risk missing insights that parametric methods may provide. Building too little structure into a model can be just as bad as building in too much structure.

There is a trade-off between the strength of model assumptions and the ability to find effects. Newer methodologies such as lowess smoothing are welcome and useful additions to the statistical toolbox. However, if we assume a linear relationship, we may be able to find it, where we will find nothing if we look for a general form of smooth curve or a completely arbitrary relationship. This is why simple non-parametric approaches are often unsatisfactory – they assume too little. Often they assume much less than we know to be true. Johnson (1995) has useful comments on the role of non-parametric tests. In part, the objection is to a view of non-parametric modeling that is too limited.

3.4.5 Why models matter – adding across contingency tables

The multi-way table `UCBAdmissions` (*datasets* package) has admission frequencies, by sex, for the six largest departments at the University of California at Berkeley in 1973 (Bickel *et al.*, 1975). Do the data provide evidence, across the University as a whole, of sex-based discrimination? Note the margins of the table:

```
> str(UCBAdmissions)
 table [1:2, 1:2, 1:6] 512 313 89 19 353 207 17 8 120 205 ...
 - attr(*, "dimnames")=List of 3
  ..$ Admit : chr [1:2] "Admitted" "Rejected"
  ..$ Gender: chr [1:2] "Male" "Female"
  ..$ Dept  : chr [1:6] "A" "B" "C" "D" ...
```

First, calculate overall admission rates (percentages) for females and males.

```
> ## Tabulate by Admit and Gender
> byGender <- margin.table(UCBAdmissions, margin=1:2)
> round(100*prop.table(byGender, margin=2)["Admitted", ], 1)
  Male Female
  44.5   30.4
```

Admission rates will now be calculated for individual departments:

```
> ## Admission rates, by department
> round(100*prop.table(UCBAdmissions,
+                      margin=2:3)["Admitted", , ], 1)
        Dept
Gender      A  B    C    D    E   F
  Male   62.1 63 36.9 33.1 27.7 5.9
  Female 82.4 68 34.1 34.9 23.9 7.0
```

As a fraction of those who applied, females were strongly favored in department A, and males somewhat favored in departments C and E.

Look now, for each department, at the numbers of males applying as a proportion of the total number of male applicants, and similarly for females:

```
> ## Calculate totals, by department, of males & females applying
> (applicants <- margin.table(UCBAdmissions, margin=2:3))
        Dept
Gender     A   B   C   D   E   F
  Male   825 560 325 417 191 373
  Female 108  25 593 375 393 341
> ## Calculate proportions of male & female applicants
> round(100*prop.table(applicants, margin=1), 1)
        Dept
Gender      A    B    C    D    E    F
  Male   30.7 20.8 12.1 15.5  7.1 13.9
  Female  5.9  1.4 32.3 20.4 21.4 18.6
```

Relatively few females (5.9%) applied to department A, while a high proportion (32.3% and 21.4% respectively) applied to departments C and E where admission rates were relatively low. The very high number of males applying to departments A and B has biased the male rates towards the relatively high admission rates in those departments, while the relatively high number of females applying to departments C, D and F biased the overall female rates towards the low admission rates in those departments. The overall bias arose because males favored departments where there were a relatively larger number of places.

What model is in mind? Is the aim to compare the chances of admission for a randomly chosen female with the chances of admission for a randomly chosen male? The relevant figure is then the overall admission rate of 30.4% for females, as against 44.5% for males. Or, is the interest in the chances of a particular student who has decided on a department? A female had a much better chance than a male in department A, while a male had a slightly better chance in departments C and E.

Here, information was available on the classifying factor on which it was necessary to condition. This will not always be the case. In any such tabulation, it is always possible that there is some further variable that, when conditioned on, can reverse or otherwise affect an observed association.

The results that give the overall proportions are, for these data and depending on the intended use, an unsatisfactory and potentially misleading summary. The phenomenon that they illustrate, known as Simpson's paradox or as the Yule–Simpson effect, is discussed in Aldrich (1995), Simpson (1951).

In any overall analysis, the effect of the classifying (or *conditioning*) factor `sex` must be explicitly incorporated in the model. There are various ways to do this. Section 8.3 demonstrates one suitable approach. See also Exercise 11 in Chapter 4, and the references given there.

3.5 Recap

Statistical models have both deterministic and random error components, or *signal* components and *noise* components. In simpler cases, which include most of the cases we

consider,

$$\text{observation} = \text{signal} + \text{noise}.$$

After fitting a model, we have

$$\text{observation} = \text{fitted value} + \text{residual}$$

which we can think of as

$$\text{observation} = \text{smooth component} + \text{rough component}.$$

The hope is that the fitted value will recapture most of the signal, and that the residual will contain mostly noise. Unfortunately, as the relative contribution of the noise increases,

- it becomes harder to distinguish between signal and noise,
- it becomes harder to decide between competing models.

Model assumptions, such as normality, independence, and constancy of the variance, should be checked, to the extent that this is possible.

3.6 Further reading

Finding the right statistical model is an important part of statistical problem-solving. Chatfield (2002, 2003b) has helpful comments. Clarke (1968) has a useful discussion of the use of models in archaeology. See also the very different points of view of Breiman and Cox (as discussant) in Breiman (2001). Our stance is much closer to Cox than to Breiman. See also our brief comments on Bayesian modeling in Section 4.10.

Johnson (1995) comments critically on the limitations of widely used non-parametric methods. See Hall (2001) for an overview of non-parametrics from a modern perspective.

References for further reading

Breiman, L. 2001. Statistical modeling: the two cultures. *Statistical Science* 16: 199–215.

Chatfield, C. 2002. Confessions of a statistician. *The Statistician* 51: 1–20.

Chatfield, C. 2003b. *Problem Solving. A Statistician's Guide*, 2nd edn.

Clarke, D. 1968. *Analytical Archaeology.*

Hall, P. 2001. Biometrika centenary: non-parametrics. *Biometrika* 88: 143–65.

Johnson, D. H. 1995. Statistical sirens: the allure of non-parametrics. *Ecology* 76: 1998–2000.

3.7 Exercises

1. The distance that a body, starting at rest, falls under gravity in t seconds is commonly given as $d = \frac{1}{2}gt^2$, where $g \simeq 9.8$ msec^{-2}. The equation can be modified to take account of the effects of air resistance, which will vary with barometric pressure and other atmospheric conditions. Will a time–distance relationship that is obtained for a human dummy that falls from a height of some thousands of meters be useful in predicting the time–distance relationship for another dummy,

or for a human, falling at another time from a similar height? Or is the situation comparable to that for the lawn roller data in Subsection 3.1.1, where the relationship is likely to be different for different lawns? [Humans have very occasionally survived falls from such heights. See http://www.greenharbor.com/fffolder/ffresearch.html]

2. Hooke's law of elasticity is an approximation which states that the amount by which a spring or other elastic body deforms is proportional to the applied force. Data are obtained for one spring. Can those data be used to make predictions for another spring that has been manufactured in the same way? How can the accuracy of such predictions be tested?

3. An experimenter intends to arrange experimental plots in four blocks. In each block there are seven plots, one for each of seven treatments. Use the function `sample()` to find four random permutations of the numbers 1 to 7 that will be used, one set in each block, to make the assignments of treatments to plots.

4. Use `y <- rnorm(100)` to generate a random sample of size 100 from a normal distribution.

   (a) Calculate the mean and standard deviation of `y`.
   (b) Use a loop to repeat the above calculation 25 times. Store the 25 means in a vector named `av`. Calculate the standard deviation of the values in `av`.
   (c) Create a function that performs the calculations described in (b). Run the function several times, showing each of the distributions of 25 means in a density plot.

5. To simulate samples from normal populations having different means and standard deviations, the `mean` and `sd` arguments can be used in `rnorm()`. Simulate a random sample of size 20 from a normal population having a mean of 100 and a standard deviation of 10.

6. Use `mfrow` to set up the layout for a 3 by 4 array of plots. In the top 4 panels, show normal probability plots for 4 separate "random" samples of size 10, all from a normal distribution. In the middle 4 panels, display plots for samples of size 100. In the bottom 4 panels, display plots for samples of size 1000. Comment on how the appearance of the plots changes as the sample size changes.

7. The function `runif()` generates a sample from a uniform distribution, by default on the interval 0 to 1. Try `x <- runif(10)`, and print out the resulting numbers. Then repeat Exercise 5 above, but taking samples from a uniform distribution rather than from a normal distribution. What shape do the plots follow?

8. The function `pexp(x, rate=r)` can be used to compute the probability that an exponential variable is less than `x`. Suppose the time between accidents at an intersection can be modeled by an exponential distribution with a rate of 0.05 per day. Find the probability that the next accident will occur during the next three weeks.

9. Use the function `rexp()` to simulate 100 exponential random numbers with rate 0.2. Obtain a density plot for the observations. Find the sample mean of the observations. Compare with the population mean (the mean for an exponential population is 1/rate).

10.* This exercise investigates simulation from other distributions. The statement `x <- rchisq(10, 1)` generates 10 random values from a chi-squared distribution with one degree of freedom. The statement `x <- rt(10, 1)` generates 10 random values from a t-distribution with one degree of freedom. Make normal probability plots for samples of various sizes from each of these distributions. How large a sample is necessary, in each instance, to obtain a consistent shape?

11. The following data represent the total number of aberrant crypt foci (abnormal growths in the colon) observed in seven rats that had been administered a single dose of the carcinogen azoxymethane and sacrificed after six weeks (thanks to Ranjana Bird, Faculty of Human Ecology, University of Manitoba for the use of these data):

```
87 53 72 90 78 85 83
```

Enter these data and compute their sample mean and variance. Is the Poisson model appropriate for these data? To investigate how the sample variance and sample mean differ under the Poisson assumption, repeat the following simulation experiment several times:

```
x <- rpois(7, 78.3)
mean(x); var(x)
```

12.* A Markov chain is a data sequence which has a special kind of dependence. For example, a fair coin is tossed repetitively by a player who begins with \$2. If "heads" appear, the player receives one dollar; otherwise, she pays one dollar. The game stops when the player has either \$0 or \$5. The amount of money that the player has before any coin flip can be recorded – this is a Markov chain. A possible sequence of plays is as follows:

Player's fortune:	2	1	2	3	4	3	2	3	2	3	2	1	0
Coin Toss result:	T	H	H	H	T	T	H	T	H	T	T	T	

Note that all we need to know in order to determine the player's fortune at any time is the fortune at the previous time as well as the coin flip result at the current time. The probability of an increase in the fortune is 0.5 and the probability of a decrease in the fortune is 0.5. The transition probabilities can be summarized in a transition matrix:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The (i, j) entry of this matrix is the probability of making a change from the value i to the value j. Here, the possible values of i and j are $0, 1, 2, \ldots, 5$. According to the matrix, there is a probability of 0 of making a transition from \$2 to \$4 in one play, since the (2, 4) element is 0; the probability of moving from \$2 to \$1 in one transition is 0.5, since the (2, 1) element is 0.5.

The following function can be used to simulate N values of a Markov chain sequence, with transition matrix P:

```
Markov <- function (N=100, initial.value=1, P)
{
    X <- numeric(N)
    X[1] <- initial.value + 1  # States 0:5; subscripts 1:6
    n <- nrow(P)
    for (i in 2:N){
    X[i] <- sample(1:n, size=1, prob=P[X[i-1], ])}
    X - 1
}
```

Simulate 15 values of the coin flip game, starting with an initial value of \$2. Repeat the simulation several times.

13. A Markov chain for the weather in a particular season of the year has the transition matrix, from one day to the next:

$$Pb = \begin{bmatrix} & Sun & Cloud & Rain \\ Sun & 0.6 & 0.2 & 0.2 \\ Cloud & 0.2 & 0.4 & 0.4 \\ Rain & 0.4 & 0.3 & 0.3 \end{bmatrix}$$

It can be shown, using linear algebra, that in the long run this Markov chain will visit the states according to the *stationary* distribution:

Sun	Cloud	Rain
0.641	0.208	0.151

A result called the *ergodic* theorem allows us to estimate this distribution by simulating the Markov chain for a long enough time.

(a) Simulate 1000 values, and calculate the proportion of times the chain visits each of the states. Compare the proportions given by the simulation with the above theoretical proportions.

(b) Here is code that calculates rolling averages of the proportions over a number of simulations and plots the result. It uses the function `rollmean()` from the *zoo* package.

```
plotmarkov <-
  function(n=10000, start=0, window=100, transition=Pb, npanels=5){
    xc2 <- Markov(n, start, transition)
    mav0 <- rollmean(as.integer(xc2==0), window)
    mav1 <- rollmean(as.integer(xc2==0), window)
    npanel <- cut(1:length(mav0), breaks=seq(from=1, to=length(mav0),
                  length=npanels+1), include.lowest=TRUE)
    df <- data.frame(av0=mav0, av1=mav1, x=1:length(mav0),
                     gp=npanel)
    print(xyplot(av0+av1 ~ x | gp, data=df, layout=c(1,npanels),
                 type="l", par.strip.text=list(cex=0.65),
                 scales=list(x=list(relation="free"))))
  }
```

Try varying the number of simulations and the width of the window. How wide a window is needed to get a good sense of the stationary distribution? This series settles down rather quickly to its stationary distribution (it "burns in" quite quickly). A reasonable width of window is, however, needed to give an accurate indication of the stationary distribution.

4

A review of inference concepts

A random sample is a set of values drawn independently from a larger population. A (uniform) random sample has the characteristic that all members of the population have an equal chance of being drawn. In the previous chapter, we discussed the implications of drawing repeated random samples from a normally distributed population, where the probability that a value lies in a given interval is governed by the normal density. This chapter will expand upon that discussion by using the idea of a sampling distribution, with its associated standard error, to assess estimation accuracy. Confidence intervals and tests of hypotheses offer a formal basis for inference, based on the sampling distribution. We will comment on weaknesses in the hypothesis testing framework.

4.1 Basic concepts of estimation

This section will introduce material that is fundamental to inference.

4.1.1 Population parameters and sample statistics

Parameters, such as the mean (μ) or standard deviation (σ), numerically summarize various aspects of a population. Such parameters are usually unknown and are estimated using *statistics* calculated using a random sample taken from the population. The sample mean is an example of a statistic, and it is used to estimate the population mean.

Other commonly used statistics are the proportion, standard deviation, variance, median, the quartiles, the slope of a regression line, and the correlation coefficient. Each may be used as an estimate of the corresponding population parameter.

4.1.2 Sampling distributions

Subsection 3.3.3 introduced the sampling distribution of the mean: the distribution of sample means, under repeated random sampling. The standard deviation of this sampling distribution is called the *standard error of the mean* (SEM). The SEM is a measure of the accuracy of the sample mean, as an estimate of the population mean.

The challenge is to use the one sample that is available, together with the assumption of independent and identically distributed sample values, to infer the sampling distribution of the mean. Two approaches will be described. The first, used in the main part of this chapter, relies on statistical theory – the Central Limit Theorem. The second, relying on repeated

resampling from the one available sample, will be the subject of Subsections 4.7.3 and 4.7.4.

Reliance on the Central Limit Theorem

As a consequence of the Central Limit Theorem, the sampling distribution of the mean can, for a population with mean μ and standard deviation σ, often be well approximated by a normal distribution with mean μ and standard deviation $\sigma/\sqrt{n}$. An estimate of the SEM is thus

$$\text{SEM} = \frac{s}{\sqrt{n}}$$

where s is an estimator of the population standard deviation σ. Refer back to Figure 3.5 (Subsection 3.3.3).

This deceivingly simple formula, relating the SEM to the standard deviation, hides quite complex mathematical ideas. Note that if the data are not independent, then the formula does not apply.

Other statistics, such as the sample proportion, have their own sampling distributions. Often, these sampling distributions are also reasonably well approximated by a normal distribution.

4.1.3 Assessing accuracy – the standard error

A small SEM suggests that the sample mean is close to the population mean, while a large SEM allows for the possibility that the sample and population means may differ widely.

The data frame `pair65`, shown earlier in Table 3.1, has information on nine sets of paired comparisons, leading to nine differences in the amount of stretch under a 1.35 kg weight. These were:

Difference	19	8	4	1	6	10	6	−3	6

The mean is 6.33, the SD is $s = 6.10$, and SEM = $6.10/\sqrt{9} = 2.03$.[1] We may report: "The mean change is 6.33 [SEM 2.03], based on $n = 9$ values", or "The mean change is 6.10/2.03 (= 3.11) times the standard error".

4.1.4 The standard error for the difference of means

Where there are two independent samples of size n_1 and n_2, the comparison is usually in the form of a difference:

$$\bar{x}_1 - \bar{x}_2$$

where $\bar{x}_1$ and $\bar{x}_2$ denote the respective sample means. If the corresponding standard errors are denoted by SEM_1 and SEM_2, then the standard error of the difference (SED) is

$$\text{SED} = \sqrt{\text{SEM}_1^2 + \text{SEM}_2^2}.$$

[1]
```
## Calculate heated-ambient; take heated & ambient from columns of pair65
test <- with(pair65, heated-ambient)
c(mean = mean(test), SD = sd(test), SEM = sd(test)/sqrt(length(test)))
```

If all SEMs are the same, then for all comparisons,

$$\text{SED} = \sqrt{2} \times \text{SEM}.$$

It is sometimes reasonable to assume equality of the standard deviations in the populations from which the samples are drawn. Then

$$\text{SEM}_1 = \frac{s}{\sqrt{n_1}}, \quad \text{SEM}_2 = \frac{s}{\sqrt{n_2}}$$

and the formula can be written as

$$\text{SED} = s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

where s is the pooled standard deviation estimate described in Subsection 2.2.3.

As an example, consider the unpaired elastic band experiment data of Subsection 2.2.3. The pooled standard deviation estimate is 10.91. Hence, the SED is $10.91 \times \sqrt{\frac{1}{10} + \frac{1}{11}} = 4.77$.[2]

4.1.5 The standard error of the median*

For data from a normal distribution, there is a similarly simple formula for the standard error of the median. It is

$$\text{SE}_{\text{median}} = \sqrt{\frac{\pi}{2}}\frac{s}{\sqrt{n}} \approx 1.25\frac{s}{\sqrt{n}}.$$

The standard error of the median is thus about 25% greater than the standard error of the mean. For data from a normal distribution, the population mean can be estimated more precisely than can the population median.

Consider again the `cuckoos` data. The median and standard error for the median of the egg lengths in the wrens' nests are 21.0 and 0.244, respectively.[3]

A different formula for the standard error of the median, one that depends on the distribution, must be used when the data cannot reasonably be approximated by a normal model.

[2]
```
## Heated vs ambient; unpaired elastic band data
heated <- c(254, 252, 239, 240, 250, 256, 267, 249, 259, 269)
ambient <- c(233, 252, 237, 246, 255, 244, 248, 242, 217, 257, 254)
v1 <- var(heated)           # 10 numbers; 10-1 = 9 d.f.
v2 <- var(ambient)          # 11 numbers; 11-1 = 10 d.f.
v <- (9*v1 + 10*v2)/(9+10)  # Pooled estimate of variance
# Estimate SED; variances may not be equal
c(sem1 = sqrt(v1/10), sem2 = sqrt(v2/11), sed = sqrt(v1/10 + v2/11))
# Estimate SED; use pooled estimate
c(sd = sqrt(v), sed = sqrt(v1/10 + v2/11))
```

[3]
```
## median and SD for length, by species: data frame cuckoos (DAAG)
wren <- split(cuckoos$length, cuckoos$species)$wren
median(wren)
n <- length(wren)
sqrt(pi/2)*sd(wren)/sqrt(n)  # this SE computation assumes normality
```

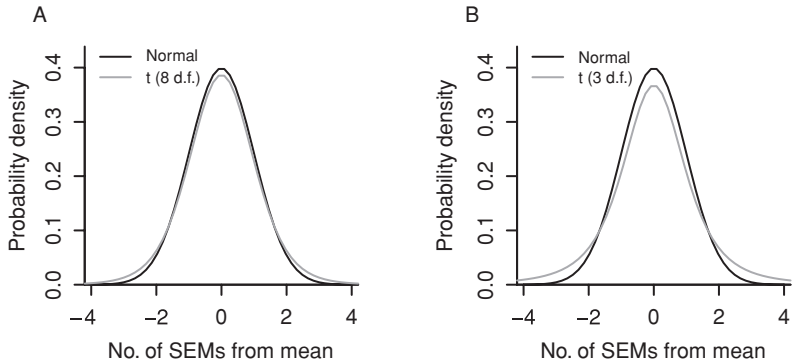


Figure 4.1 Panel A overlays the density for a normal distribution with the density for a t-distribution with 8 d.f. Panel B overlays the density for a t-distribution with 3 d.f.

4.1.6 The sampling distribution of the t-statistic

The formula

$$t = \frac{\bar{x} - \mu}{\text{SEM}}$$

counts the number of standard error units between the true value μ and the sample estimate $\bar{x}$. It can be thought of as a standardized distance between the true mean and the sample mean.

The variability in t has two sources: the sampling variability of $\bar{x}$ and the sampling variability of SEM. The replacing of σ by s introduces an uncertainty that is larger as the degrees of freedom $n - 1$ in s are smaller. Hence the use of a t-distribution with $n - 1$ degrees of freedom (d.f.), where if σ was known precisely a normal distribution would be used. The t-statistic becomes more and more like a standard normal random variable as the sample size increases.

Figure 4.1B shows the density curve for a normal distribution overlaid with those for t-distributions with 8 and 3 d.f. respectively. The main difference, in each case, is in the tails, and much larger for the t-distribution with the smaller d.f., 3 as opposed to 8. In the terminology of Subsection 3.2.2, the t-distribution is heavy-tailed – heavier for smaller than for larger degrees of freedom.

For the data in the data frame `pair65`, the relevant inference is suitably based on the mean $\bar{d}$ of the differences that were observed when the bands were heated. In order to standardize this mean difference, it needs to be divided by its standard error SE[$\bar{d}$], i.e., the relevant statistic is

$$t = \frac{\bar{d}}{\text{SE}[\bar{d}]} = \frac{\bar{d}}{s/\sqrt{9}} = \frac{6.33}{2.47} = 3.11.$$

The mean is 3.11 times the magnitude of the standard error.

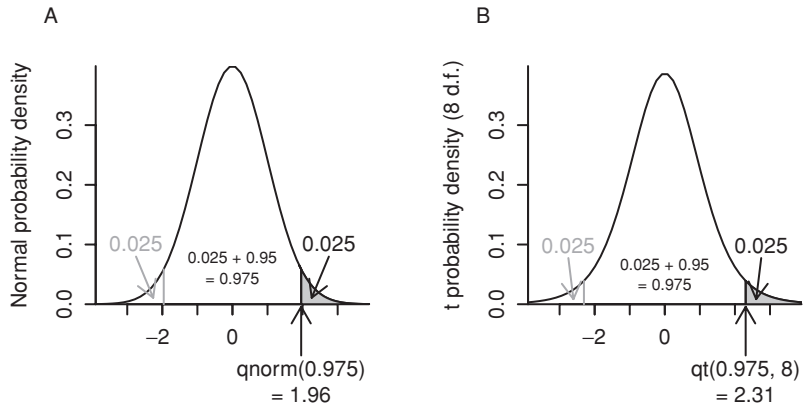


Figure 4.2 Calculation of the endpoints of the symmetrically placed region that encloses 95% of the probability: (A) for a normal distribution, and (B) for a t-distribution with 8 d.f. In each panel, the upper 2.5% of the area under the curve is shaded in gray.

4.2 Confidence intervals and tests of hypotheses

Calculations with the t-distribution

Calculations for the t-distribution follow the same pattern as those shown for the normal distribution in Subsection 3.2.2, but now with a distribution whose standard deviation is the SEM, which has to be estimated. There are two sorts of calculation that may be useful, both of which can be related to Figure 4.2:

- **Given the distance from the mean, calculate the area under the curve.** Thus, calculate the area under the density curve within some specified number of standard errors either side of the mean. For this, use functions that have p as their initial letter, here `pnorm()` and `pt`:

```
> #  Plus or minus 1.96SE normal distribution limits, e.g.
> pnorm(1.96) - pnorm(-1.96)
[1] 0.95
> #  Plus or minus 2.31SE t distribution (8 df) limits, e.g.
> pt(2.31, 8) - pt(-2.31,8)  # 2.31 SEs either side
[1] 0.95
```

- **Given an area under the curve, calculate the limit or limits.** Thus, what distance from the mean gives an area under the curve, up to and including that point, that takes some specified value? For this, use functions that have q as their initial letter, here `qnorm()` and `qt()`:

```
> qnorm(0.975)          # normal distribution
[1] 1.96
> qt(0.975, 8)          # t-distribution with 8 d.f.
[1] 2.31
```

Confidence intervals of 95% or 99%

The second of these statements makes it possible to say that in sampling from the sampling distribution of $t_8 = \frac{\bar{d}-\mu}{s/\sqrt{9}}$, 95% of the values of t_8 will lie between −2.31 and 2.31, i.e., that

Table 4.1 *Comparison of normal distribution endpoints (multipliers for the SEM) with the corresponding t-distribution endpoints on 8 d.f.*

Probability enclosed between limits	Cumulative probability	Number of SEMs	
		Normal distribution	t-Distribution (8 d.f.)
68.3%	84.1%	1.0	1.07
95%	97.5%	1.96	2.31
99%	99.5%	2.58	3.36
99.9%	99.95%	3.29	5.04

$\bar{d} - \mu$ will lie between $-2.31s$ and $2.31s$. In other words, in 95% of such samples $\bar{d}$ will lie within a distance $2.31s$ of μ. Furthermore (see Table 4.1, or enter `qt(0.995, 8)`), in 99% of such samples $\bar{d}$ will lie within a distance $3.36s$ of μ. This leads immediately to the following "confidence" (or coverage) interval for μ:

95% CI: $(6.33 - 2.03 \times 2.31, 6.33 + 2.03 \times 2.31) = (1.64, 11.02)$
99% CI: $(6.33 - 2.03 \times 3.36, 6.33 + 2.03 \times 3.36) = (-0.49, 13.15)$

Code that may be used (here, for a 95% confidence interval) is:[4]

```
## 95% CI for mean of heated-ambient: data frame pair65 (DAAG)
with(pair65, t.test(heated, ambient, paired=TRUE,
                    conf.level=0.95)$conf.int)
```

The confidence interval has been constructed so that most often, when the sample is taken in the way that the one available sample has been taken, it will include the population mean. The two common choices for the long-run proportion of similar samples for which the corresponding intervals should contain the population mean are 95% and 99%.

Tests of hypotheses

If the confidence interval for the population mean does not contain zero, this is equivalent to rejection of the hypothesis that the population mean is zero. Starting from a 95% confidence interval, the "significance level" for the test is $p = 1 - 0.95 = 0.05$.

In the example just considered, the 95% confidence interval does not contain zero, while the (wider) 99% confidence interval does contain zero. Thus the hypothesis that the population mean is zero is rejected for $p = 0.05$, but not for $p = 0.01$. The value of p that is on the borderline between rejection and non-rejection is termed the p-value.

This value can be obtained by doubling the probability that the t-statistic is less than

$$-\text{mean/SEM} = -6.33/2.03$$

```
## Probability that t-statistic (8 d.f.) is less than -6.33/2.03
> 1-pt(6.33/2.03, 8)    # Equals pt(-6.33/2.03, 8)
[1] 0.00713
```

[4] `## Detailed calculations; 95% CI for mean of heated-ambient`
`pair65.diff <- with(pair65, heated-ambient)`
`pair65.n <- length(pair65.diff)`
`pair65.se <- sd(pair65.diff)/sqrt(pair65.n)`
`mean(pair65.diff) + qt(c(.025,.975),8)*pair65.se`

Doubling 0.00713 to determine the sum of the probabilities in the two tails yields $p = 0.014$. The result may be summarized in the statement: "Based on the sample mean of 6.33, the population mean is greater than zero ($p = 0.014$)".

Formal hypothesis testing requires the statement of *null* and alternative hypotheses. Taking the population mean to be μ, the *null hypothesis* is

$$\mathrm{H}_0 : \ \mu = 0$$

while the alternative hypothesis is $\mu \neq 0$.

The formal methodology of hypothesis testing may seem contorted. A small p-value makes the null hypothesis appear implausible. It is not a probability statement about the null hypothesis itself, or for that matter about its alternative. All it offers is an assessment of implications that flow from accepting the null hypothesis. A straw man is set up, the statement that $\mu = 0$. The typical goal is to knock down this straw man. By its very nature, hypothesis testing lends itself to various abuses.

What is a small p-value?

At what point is a p-value small enough to be convincing? The conventional $p = 0.05$ ($= 5\%$) cutoff is too large, if results from the experiment are to be made the basis for a recommendation for changes to farming practice or to medical treatment. It may be too small when the interest is in deciding which effects merit further experimental or other investigation. There must be a careful balancing of the likely costs and benefits of any such recommendation, having regard to the statistical evidence. In any particular case, consider carefully:

- Is there other relevant evidence, additional to that summarized in a p-value or confidence interval?
- What is the most helpful way to present results: a p-value, or a confidence interval, or something else again?

t-Distribution versus the normal distribution

Table 4.1 compares the normal distribution multipliers with those for a t-distribution with 8 d.f., for several different choices of area under the curve. Changing from a normal distribution to a t-distribution with 8 d.f. led to a small change, from 1.0 to 1.07, for enclosing the central 68.3% of the area. There is a substantial difference, giving an increase from 1.96 to 2.31, for enclosing 95% of the area.

How good is the normal theory approximation?

For random samples from a distribution that is close to symmetric, the approximation is often adequate, even for samples as small as 3 or 4. In practice, we may know little about the population from which we are sampling. Even if the main part of the population distribution is symmetric, occasional aberrant values are to be expected. Such aberrant

Table 4.2 *Formulae for confidence intervals and tests of hypothesis based on the t-distribution.*

	Confidence interval	Test statistic	d.f.
One-sample t	$\bar{d} \pm t_{\text{crit}}\text{SE}[\bar{d}]$	$t = \dfrac{\bar{d}}{\text{SE}[\bar{d}]}$	$n-1$
e.g.	$6.33 \pm 2.306 \times \frac{6.10}{\sqrt{9}}$	$t = \frac{6.33}{6.10/\sqrt{9}}$	8
Two-sample t	$\bar{x}_2 - \bar{x}_1 \pm t_{\text{crit}}\text{SE}[\bar{x}_2 - \bar{x}_1]$	$t = \dfrac{\bar{x}_2 - \bar{x}_1}{\text{SE}[\bar{x}_2 - \bar{x}_1]}$	$n_1 + n_2 - 2$
e.g.	$253.5 - 244.1 \pm 2.09 \times 10.91\sqrt{\frac{1}{10} + \frac{1}{11}}$ $= 253.5 - 244.12.09 \times 4.77 = (-0.6, 19.4)$	$t = \dfrac{253.5 - 244.1}{10.91 \times \sqrt{\frac{1}{10} + \frac{1}{11}}}$	19

Here, t_{crit} is the 97.5th percentile of a t-statistic with 8 d.f. (one-sample example) or 19 d.f. (two-sample example). (The 97.5th percentile is the same as the two-sided 5% critical value.)

values do, perhaps fortunately, work in a conservative direction – they make it more difficult to detect genuine differences. The take-home message is that, especially in small samples, the probabilities and quantiles can be quite imprecise. They are rough guides, intended to assist researchers in making a judgment.

4.2.1 A summary of one- and two-sample calculations

Confidence intervals for a mean difference, or for a difference of means, have the form

$$\text{difference} \ \pm \ t\text{-critical value} \ \times \ \text{standard error of difference}.$$

The t-statistic has the form

$$t = \frac{\text{difference}}{\text{standard error of difference}}.$$

Given t, the p-value for a (two-sided) test is defined as

$$P(T > t) + P(T < -t)$$

where T has a t-distribution with the appropriate number of degrees of freedom. A small p-value corresponds to a large value of $|t|$, regarded as evidence that the true difference is non-zero and leading to the rejection of the null hypothesis.

Table 4.2 lists confidence intervals and tests in the one- and two-sample cases.[5] The single-sample example is for the paired elastic band data that we discussed at the beginning of this section. The example that we use for the two-sample calculations was discussed in Subsection 2.2.3.

[5]
```
## t-test and confidence interval calculations
heated <- c(254, 252, 239, 240, 250, 256, 267, 249, 259, 269)
ambient <- c(233, 252, 237, 246, 255, 244, 248, 242, 217, 257, 254)
t.test(heated, ambient, var.equal=TRUE)
```

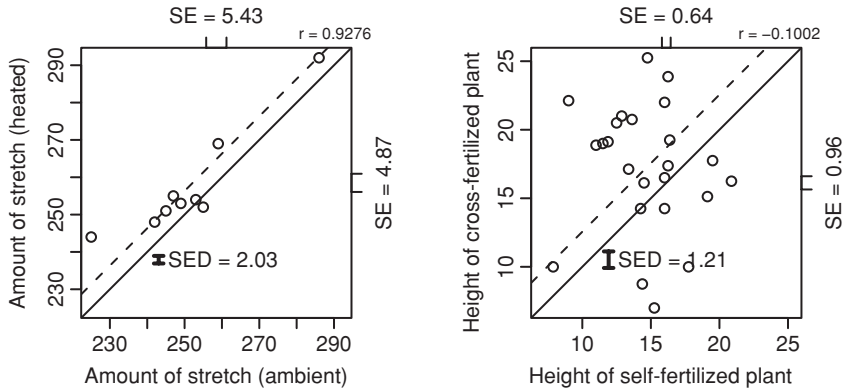


Figure 4.3 Second versus first member, for each pair. The first panel is for the ambient/heated elastic band data from Subsection 4.1.6, while the second is for Darwin's plants.

When is pairing helpful?

Figure 4.3 shows, for two different sets of paired data, a plot of the second member of the pair against the first.[6] The first panel is for the paired elastic band data of Subsection 4.1.6, while the second panel (for the data set `mignonette`) is from the biologist Charles Darwin's experiments that compared the heights of crossed plants with the heights of self-fertilized plants (data, for the wild mignonette *Reseda lutea*, are from p. 118 of Darwin, 1877). Plants were paired within the pots in which they were grown, with one plant on one side and one on the other.

For the paired elastic band data there is a clear correlation, and the standard error of the difference is much less than the root mean square of the two separate standard errors. For Darwin's data there is little evidence of correlation. The standard error of differences of pairs is about equal to the root mean square of the two separate standard errors. For the elastic band data, the pairing was helpful; it led to a low SED. The pairing was not helpful for Darwin's data (note that Darwin (cited above) gives other data sets where the pairing was helpful, in the sense of allowing a more accurate comparison).

If the data are paired, then the two-sample t-test corresponds to the wrong model! It is appropriate to use the one-sample approach, whether or not there is evidence of correlation between members of the same pair.

What if the standard deviations are unequal?

If variances are heterogeneous (unequal variances or standard deviations), the t-statistic based on the pooled variance estimate is inappropriate. The Welch procedure gives an adequate approximation, unless degrees of freedom are very small. The Welch statistic is

[6]
```
## heated vs ambient: pair65 (DAAG); and cross vs self: mignonette (DAAG)
par(mfrow=c(1,2))
plot(heated ~ ambient, data=pair65); abline(0, 1)    # left panel
with(pair65, abline(mean(heated-ambient), 1, lty=2))
plot(cross ~ self, data=mignonette); abline(0, 1)    # right panel
with(mignonette, abline(mean(cross-self), 1, lty=2))
par(mfrow = c(1,1))
```

the difference in means divided by a standard error of difference that allows for unequal variances, i.e.,

$$t = \frac{\bar{x}_2 - \bar{x}_1}{\text{SED}},$$

where

$$\text{SED} = \sqrt{\frac{s_2^2}{n_2} + \frac{s_1^2}{n_1}}.$$

If the two variances are unequal this does not have a t-distribution. However, critical values are quite well approximated by the critical values of a t-distribution with degrees of freedom given by a readily calculated function of the observed sample variances and sample sizes. The most commonly used approximation is that of Welch (1949), leading to the name *Welch test*. For details, see Miller (1986). The function `t.test()` has the Welch test as its default; unequal variances are assumed unless the argument `var.equal=TRUE` is given.

Note that if $n_1 = n_2$ then the statistic is the same as for the t-test that is based on the pooled estimate of variance. However, the degrees of freedom are likely to be reduced.

Different ways to report results

For the paired elastic band data of Table 3.1, the mean difference in amount of stretch before and after heating is 6.33, with a standard deviation of 6.10. The standard error of this difference (SED) is thus $6.10/\sqrt{9} = 2.03$. The bare minimum of information that should be reported is: "The mean change is 6.33 [SED 2.03, $n = 9$]". In engineering and physical science contexts where the aim is to accompany a report of the mean with a statement of its precision, this may be enough. It is most appropriate when differences are large, of the order of more than five times the SEM for any individual treatment or four times the SED for comparing two means.

Confidence intervals and hypothesis testing give this form of report a more interpretive twist. Here are some of the various alternatives:[7]

1. The mean change is 6.33 [SED 2.03, $n = 9$].
2. The t-statistic is $t = 6.333/2.034 = 3.11$, on 8 $(= 9 - 1)$ degrees of freedom. In other words, the difference is 3.11 times the standard error.
3. A 95% confidence interval for the change is

   $$(6.33 - 2.306 \times 2.034, 6.33 + 2.306 \times 2.034), \text{ i.e., } (1.64, 11.02).$$

   [The multiplier, equal to 2.306, is the 5% two-sided critical value for a t-statistic on 8 $(= 9 - 1)$ d.f.]

[7]
```
## Different ways to report results: calculations
pair65.diff <- with(pair65, heated-ambient)
n <- length(pair65.diff)
av <- mean(pair65.diff); sd <- sqrt(var(pair65.diff)); se <- sd/sqrt(n)
print(c(mean=av, SED=se, "mean/SED"=av/se))   # Items 1 and 2
t.test(pair65.diff)                           # Items 3 and 4
```

Table 4.3 *Approximate 95% confidence interval, assuming $0.35 \leq \pi \leq 0.65$.*

n	Approximate 95% confidence interval
25	$p \pm 20\%$
100	$p \pm 10\%$
400	$p \pm 5\%$
1000	$p \pm 3.1\%$

4. We reject the null hypothesis that the true mean difference is 0 ($p = 0.014$) – see Subsection 4.2.1 for definitions.
   [The two-sided p-value for $t = 3.11$ on 8 d.f. is 0.014.]

Alternative 1 is straightforward. The t-statistic (alternative 2) expresses the change as a multiple of its standard error. The conventional wisdom is that the change is worthy of note if the p-value is less than 0.05 or, equivalently, if the 95% confidence interval does not contain 0. For this, the t-statistic must be somewhat greater than 1.96, i.e., for all practical purposes >2.0. For small degrees of freedom, the t-statistic must be substantially greater than 2.0.

Readers who have difficulty with alternatives 3 and 4 may find it helpful to note that these restate and interpret the information in alternatives 1 and 2. If standard errors are not enough and formal inferential information is required, confidence intervals may be preferable to formal tests of hypotheses.

4.2.2 Confidence intervals and tests for proportions

We assume that individuals are drawn independently and at random from a binomial population where individuals are in one of two categories – male as opposed to female, a favorable treatment outcome as opposed to an unfavorable outcome, survival as opposed to non-survival, defective as opposed to non-defective, Democrat as opposed to Republican, etc. Let π be the population proportion. In a sample of size n, the proportion in the category of interest is denoted by p. Then,

$$\mathrm{SE}[p] = \sqrt{\pi(1-\pi)/n}.$$

An upper bound for SE[p] is $1/(2\sqrt{n})$. If π is between about 0.35 and 0.65, the inaccuracy in taking SE[p] as $1/(2\sqrt{n})$ is small.

This approximation leads to the confidence intervals shown in Table 4.3. Note again that the approximation is poor if π is outside the range 0.35 to 0.65.

An alternative is to use the estimator

$$\widehat{\mathrm{SE}[p]} = \sqrt{\frac{p(1-p)}{n}}.$$

An approximate 95% confidence bound for the proportion π is then

$$p \pm 1.96\sqrt{\frac{p(1-p)}{n}}.$$

4.2.3 Confidence intervals for the correlation

The correlation measure that we discuss here is the Pearson or product–moment correlation, which measures linear association.

The standard error of the correlation coefficient is typically not a useful statistic. The distribution of the sample correlation, under the usual assumptions (e.g., bivariate normality), is too skew. The function `cor.test()` may be used to test the null hypothesis that the sample has been drawn from a population in which the correlation ρ is zero. For given x, the distribution of y is assumed normal, independently for different ys and with mean given by a linear function of x.

Classical methods for comparing the magnitudes of correlations, or for calculation of a confidence interval for the correlation, rely on the assumption that the joint distribution of (x, y) is bivariate normal. In addition to the assumption for the test that $\rho = 0$, we need to know that x is normally distributed, independently between (x, y) pairs. This assumption is required for the default confidence interval that `cor.test()` outputs. In practice, it may be enough to check that both x and y have normal distributions.

4.2.4 Confidence intervals versus hypothesis tests

Those who have problems with confidence intervals and (especially) tests of hypotheses (often called significance tests) are in good company. There is increasing support for the view that they should play a relatively minor role in statistical analysis or be eliminated altogether.

The methodology is too often abused. Papers that present a large number of significance tests are, typically, not making good use of the data. It becomes difficult to know what to make of the results. Among a large number of tests, some will be significant as a result of chance.

Misunderstandings are common in the literature, even among mature researchers. A p-value does not allow the researcher to say anything about the probability that either hypothesis, the null or its alternative, is true. Then why use them? Perhaps the best that can be said is that hypothesis tests often provide a convenient and quick answer to questions about whether effects seem to stand out above background noise. However if all that emerges from an investigation are a few p-values, we have to wonder what has been achieved.

Because of these problems, there are strong moves away from hypothesis testing and towards confidence intervals. Tukey (1991) argues strongly, and cogently, that confidence intervals are more informative and more honest than p-values. He argues

> Statisticians classically asked the wrong question – and were willing to answer with a lie, one that was often a downright lie. They asked "Are the effects of A and B different?" and they were willing to answer "no".

All we know about the world teaches us that the effects of A and B are always different – in some decimal place – for every A and B. Thus asking "Are the effects different?" is foolish. What we should be answering first is "Can we tell the direction in which the effects of A differ from the effects of B?" In other words, can we be confident about the direction from A to B? Is it "up", "down", or "uncertain"? [Tukey, 1991]

Tukey argues that we should never conclude that we "accept the null hypothesis". Rather, our uncertainty is about the direction in which A may differ from B. Confidence intervals do much better at capturing the nature of this uncertainty.

Guidelines for significance testing

Few scientific papers make more than half-a-dozen points that are of consequence. Any significance tests should be closely tied to these main points, preferably with just one or two tests for each point that is made. Keep any significance tests and p-values in the background. Once it is apparent that an effect is statistically significant, the focus of interest should shift to its pattern and magnitude, and to its scientific significance.

It is poor practice to perform t-tests for each comparison between treatments when the real interest is (or should be) in the overall pattern of response. Where the response depends on a continuous variable, it is often pertinent to ask whether, e.g., the response keeps on rising (falling), or whether it rises (falls) to a maximum (minimum) and then falls (rises).

Significance tests should give the researcher, and the reader of the research paper, confidence that the effects that are discussed are real! The focus should then move to the substantive scientific issues. Statistical modeling can be highly helpful for this. The interest is often, finally, in eliciting the patterns of response that the data present.

4.3 Contingency tables

Table 4.4 is from US data that were used in the evaluation of labor training programs, aimed at individuals who had experienced economic and social difficulties. The table shows numbers of high school graduates and dropouts who had participated in a labour training program (NSW group) and those who had not participated (PSID3 group).[8] These data will be discussed further in Section 13.2.

A glance at the table suggests that the proportion of high school dropouts in the NSW group is much higher than in the PSID3 group. The chi-squared test for no association is described in the next subsection; it can be used to check this formally:

```
> # To agree with hand calculation below, specify correct=FALSE
> chisq.test(with(nswpsid3, table(trt, nodeg)), correct=FALSE)
. . . .
X-squared = 19.9, df = 1, p-value = 8.189e-06
```

[8]
```
## Compare number with a high school qualification, between 'untreated' rows
## from data frame psid3 and 'treated' rows from nswdemo
library(DAAG)   # Data are from DAAG
nswpsid3 <- rbind(psid3, subset(nswdemo, trt==1))
table(nswpsid3$trt, nswpsid3$nodeg)
  # PSID3 males are coded 0; NSW male trainees are coded 1.
```

Table 4.4 *Contingency table derived from data that relates to the Lalonde (1986) paper.*

	High school graduate certificate	
	Yes	No
PSID3 males	63	65
NSW male trainees	80	217

Table 4.5 *The calculated* expected values *for the contingency table in Table 4.4.*

	High school graduate			
	Yes	No	Total	Row proportion
PSID3	63 (115)	65 (12.95)	128	$128/425 = 0.301$
NSW74 trainees	80 (267.0)	217 (30.05)	297	$217/425 = 0.699$
Total	143	282	425	
Column proportion	$143/425 = 0.336$	$282/425 = 0.664$		

The small p-value confirms that high school dropouts are more strongly represented in the NSW data.

The mechanics of the chi-squared test

The null hypothesis is that the proportion of the total in each cell is, to within random error, the result of multiplying a row proportion by a column proportion. The independence assumption, i.e., the assumption of independent allocation to the cells of the table, is crucial.

Assume there are I rows and J columns. The expected value in cell (i, j) is calculated as

$$E_{ij} = \text{(proportion for row } i\text{)} \times \text{(proportion for column } j\text{)} \times \text{total.}$$

We can then obtain a score for each cell of the table by computing the absolute value of the difference between the expected value and the observed value (with the continuity correction that is the default, 0.5 would be subtracted at this point), squaring, dividing the result by the expected value, and replacing any negative scores by zero. Summing over all scores gives the chi-squared statistic.

Under the null hypothesis the chi-squared statistic has an approximate chi-squared distribution with $(I - 1)(J - 1)$ degrees of freedom. In Table 4.5, the values in parentheses are the expected values E_{ij}.

Table 4.6 *Contingency table compiled from Hobson (1988, Table 12.1, p. 248).*

	Object moves	
Dreamer moves	Yes	No
Yes	5	17
No	3	85

The expected values are found by multiplying the column totals by the row proportions. (Alternatively, the row totals can be multiplied by the column proportions.) Thus $117 \times 0.591 = 69.15$, $196 \times 0.591 = 115.85$, etc.

An example where a chi-squared test may not be valid

In Table 4.6 we summarize information that Hobson (1988) derived from drawings of dreams, made by an unknown person that he calls "The Engine Man". Among other information Hobson notes, for each of 110 drawings of dreams made, whether the dreamer moves, and whether an object moves. Dreamer movement may occur if an object moves, but is relatively rare if there is no object movement. (Note that Hobson does not give the form of summary that we present in Table 4.6.)

It may also seem natural to do a chi-squared test for no association.[9] This gives $\chi^2 = 7.1$ (1 d.f.), $p = 0.008$.

A reasonable rule, for the use of the chi-squared approximation, may be that all expected values should be at least 2 (Miller, 1986), a requirement that is satisfied for this application of the test. A check is to do a Fisher exact test. In this instance the Fisher exact test[10] gives, surprisingly, exactly the same result as the chi-squared test, i.e., $p = 0.008$.

A more serious concern is that there is a time sequence to the dreams. Thus, there could well be runs of dreams of the same type. Hobson gives the numbers of the dreams in sequence. Assuming these represent the sequence in time, this would allow a check of the strength of any evidence for runs. Hobson's table has information that our tabular summary (Table 4.6) has not captured.

4.3.1 Rare and endangered plant species

The calculations for a test for no association in a two-way table can sometimes give useful insight, even where a formal test of statistical significance would be invalid. The example that now follows (Table 4.7) illustrates this point. Data are from species lists for various regions of Australia. Species were classified CC, CR, RC and RR, with C denoting common and R denoting rare. The first code letter relates to South Australia and Victoria, and the

[9] `## Engine man data`
`engineman <- matrix(c(5,3,17,85), 2,2)`
`chisq.test(engineman)`
[10] `fisher.test(engineman)`

Table 4.7 *Cross-classification of species occurring in South Australia/ Victoria and in Tasmania.*

Common/rare classification	Habitat type		
	D	W	WD
CC	37	190	94
CR	23	59	23
RC	10	141	28
RR	15	58	16

second to Tasmania. They were further classified by habitat according to the Victorian register, where D = dry only, W = wet only, and WD = wet or dry.[11]

We use a chi-squared calculation to check whether the classification into the different habitats is similar for the different rows. Details of the calculations are:

```
> (x2 <- chisq.test(rareplants))

        Pearson's Chi-squared test

data:  rareplants
X-squared = 35, df = 6, p-value = 4.336e-06
```

This low p-value should attract a level of skepticism. We do not have a random sample from some meaningful larger population. Suppose that there is clustering, so that species come in closely related pairs, with both members of the pair always falling into the same cell of the table. This will inflate the chi-squared statistic by a factor of 2 (the net effect of inflating the numerator by 2^2, and the denominator by 2). There probably is some such clustering, though different from that of this simplistic example. Such clustering will inflate the chi-squared statistic by an amount that the available information does not allow us to estimate.

The standard Pearson chi-squared tests rely on multinomial sampling assumptions, with counts entering independently into the cells. Where it is possible to form replicate tables, the assumption should be tested.

Figure 4.4 shows expected number of species, by habitat.[12]

[11]
```
## Enter the data thus:
rareplants <- matrix(c(37,190,94,23,59,23,10,141,28,15,58,16), ncol=3,
     byrow=TRUE, dimnames=list(c("CC","CR","RC","RR"), c("D","W","WD")))
```

[12]
```
## Expected number of species, by habitat (calculate x2 as above)
x2E <- stack(data.frame(t(x2$expected)))
habitat <- rep(c(1,2,3), 4)
plot(x2E$values ~ habitat, axes=FALSE, xlim=c(0.5, 3.5), pch=16,
     xlab="habitat", ylab="Expected Number of Species")
text(x2E$values ~ habitat, labels=x2E$ind, pos=rep(c(4,4,2,2),3))
axis(1, at=seq(1,3), labels=c("D", "W", "WD"))
axis(2); box()
```

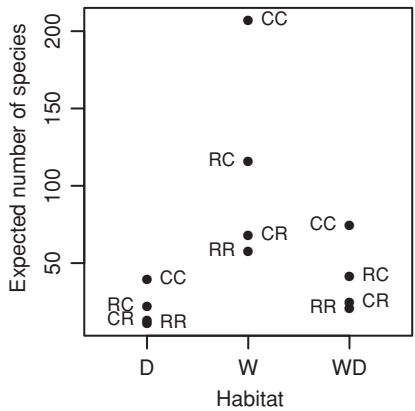


Figure 4.4 Expected number of species, by habitat, for the `rareplants` data.

Examination of departures from a consistent overall row pattern

The investigator then needs to examine the nature of variation with the row classification. For this, it is helpful to look at the residuals; these are calculated as (observed − expected)/expected$^{0.5}$:

```
> x2 <- chisq.test(rareplants)
> ## Standardized residuals
> residuals(x2)
        D       W     WD
CC -0.369 -1.1960  2.263
CR  2.828 -1.0666 -0.275
RC -2.547  2.3675 -2.099
RR  1.242  0.0722 -1.023
```

The null hypothesis implies that the expected relative numbers in different columns are the same in every row. The chi-squared residuals show where there may be departures from this pattern. In large tables these will, under the null hypothesis, behave like random normal deviates with mean zero and variance one. The values that should be noted, if the assumptions required for a chi-squared test are satisfied, are those whose absolute value is somewhat greater than 2.0. For the present table, there are five standardized residuals whose value is substantially greater than 2.0. It is these, and especially the two that are largest, that should perhaps attract attention.

Notice that the `CC` species are, relative to the overall average, over-represented in the `WD` classification, the `CR` species are over-represented in the `D` classification, while the `RC` species are under-represented in `D` and `WD` and over-represented in `W`.

For reference, here is the table of expected values:

```
> x2$expected
      D     W   WD
CC 39.3 207.2 74.5
CR 12.9  67.8 24.4
RC 21.9 115.6 41.5
RR 10.9  57.5 20.6
```

4.3.2 Additional notes

Interpretation issues

Having found an association in a contingency table, what does it mean? The interpretation will differ depending on the context. The incidence of gastric cancer is relatively high in Japan and China. Do screening programs help? Here are two ways in which the problem has been studied:

- In a long-term follow-up study, patients who have undergone surgery for gastric cancer may be classified into two groups – a "screened" group whose cancer was detected by mass screening, and an "unscreened" group who presented at a clinic or hospital with gastric cancer. The death rates over the subsequent five- or ten-year period are then compared. For example, the five-year mortality may be around 58% in the unscreened group, compared with 72% in the screened group, out of approximately 300 patients in each group.
- In a prospective cohort study, two populations – a screened population and an unscreened population – may be compared. The death rates in the two populations over a ten-year period may then be compared. For example, the annual death rate may be of the order of 60 per 100 000 for the unscreened group, compared with 30 per 100 000 for the screened group, in populations of several thousand individuals.

In the long-term follow-up study, the process that led to the detection of cancer was different between the screened and unscreened groups. The screening may lead to surgery for some cancers that would otherwise lie dormant long enough that they would never attract clinical attention. The method of detection is a confounding factor. It is necessary, as in the prospective cohort study, to compare all patients in a screened group with all patients in an unscreened group. Even so, it is necessary, in a study where assignment of participants is not random, to be sure that the two populations are comparable.

Modeling approaches

Modeling approaches typically work with data that record information on each case separately. Data where there is a binary (yes/no) outcome, and where logistic regression may be appropriate, are an important special case. Chapter 8 gives further details.

4.4 One-way unstructured comparisons

Figure 4.5 displays data from a one-way unstructured comparison between three treatments. The weights of the plants were measured after two months on respective treatments: water concentrated nutrient, and concentrated nutrient plus the selective herbicide 2,4-D. Data are:

```
tomato <-
  data.frame(weight=
              c(1.5, 1.9, 1.3, 1.5, 2.4, 1.5,    # water
                1.5, 1.2, 1.2, 2.1, 2.9, 1.6,    # Nutrient
                1.9, 1.6, 0.8, 1.15, 0.9, 1.6), # Nutrient+24D
              trt = rep(c("water", "Nutrient", "Nutrient+24D"),
                        c(6, 6, 6)))
```

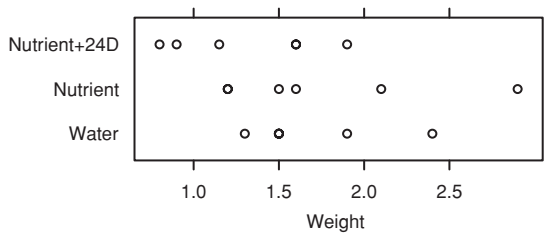


Figure 4.5 Weights of tomato plants, after two months of the three treatments.

```
## Make water the first level of trt.  It will then appear as
## the initial level in the graphs. In aov or lm calculations,
## it will be the baseline or reference level.
tomato$trt <- relevel(tomato$trt, ref="water")
```

Figure 4.5 can be obtained with:

```
stripplot(trt~weight, aspect=0.6, data=tomato)
```

The strip plots display "within-group" variability, as well as giving an indication of differences among the group means. Variances seem similar for the three treatments.

There is a single explanatory factor (`trt`), with one level for each of the different treatments that were applied. A simple-minded approach is to calculate the means for each of the three treatments, and then examine all three pairwise comparisons. With three comparisons this is, arguably, a reasonable strategy.

If there were four treatments, there would be six comparisons, and it really would become desirable to do one analysis rather than six. We would certainly not want to draw six graphs, one for each pair of treatments that are compared. Hence the use of an analysis of variance (really, as noted above, the fitting of a linear model) to do an overall analysis. We will first examine an overall visual summary of the analysis results, then examine the analysis.

The tomato data – a visual summary

The function `onewayPlot()`, from the *DAAG* package, provides a convenient visual summary of results, shown in Figure 4.6. The code is:

```
## Do analysis of variance calculations
tomato.aov <- aov(weight ~ trt, data=tomato)
## Summarize results graphically
oneway.plot(obj=tomato.aov)
```

Notice that the graph gives two different assessments of the least difference that should be treated as "significant". These use different criteria:

- The 5% least significant difference (LSD) is designed so that, under the null model (no differences), significant differences will be found in 5% of comparisons.
- The 5% honest significant difference (HSD) is designed so that, under the null model, the maximum difference will be significant in 5% of experiments.

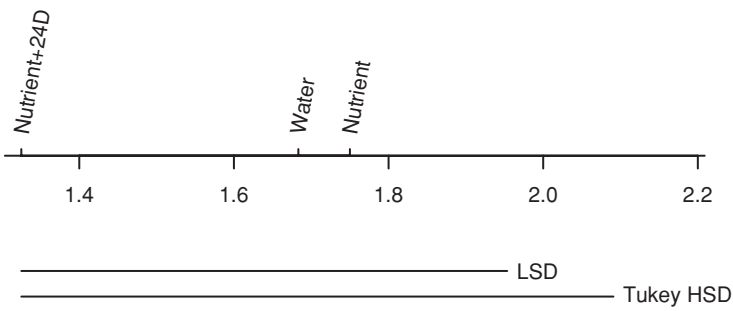


Figure 4.6 Graphical presentation of results from the analysis of the tomato weight data. Means that differ by more than the LSD (least significant difference) are different, at the 5% level, in a t-test that compares the two means. Tukey's honest significant difference (HSD) takes into account the number of means that are compared. See the text for details.

The LSD is commonly regarded as overly lax, while the HSD may be overly conservative. There are a variety of alternatives to the HSD that are less conservative; see the more detailed discussion of multiple comparisons in the next subsection.

The simplicity of Figure 4.6 is appealing, but it is important to note the assumption that the standard error of difference is the same for all treatment comparisons. As all three treatments have the same number of observations, it is enough for the variance to be the same for all treatments.

The analysis of variance table

The analysis of variance table is given by the `anova()` function, thus:

```
> anova(tomato.aov)
Analysis of Variance Table

Response: weight
           Df Sum Sq Mean Sq F value Pr(>F)
trt         2   0.63    0.31     1.2   0.33
Residuals 15   3.91    0.26
```

Observe that the residual mean squared error is 3.21. Note that two degrees of freedom are associated with estimating the variance of the three group means. Each treatment contributes $6 - 1 = 5$ d.f. to the pooled or residual sum of squares, giving $3 \times 5 = 15$ d.f. in all. Note that 2 (for `trt`) plus 15 (for `Residuals`) equals 17, which is one less than the number of observations. Estimation of the overall mean accounts for the remaining degree of freedom.

The `Mean Sq` ("mean square") column has estimates of between-sample (`trt`) and within-sample variability (`Residuals`). The between-sample variance can be calculated by applying the function `var()` to the vector of treatment means, then multiplying by the common sample size, in this case 6. The within-sample variability estimate is, effectively, a pooled variance estimate for the three treatments. Each mean square is the result from dividing the `Sum Sq` ("sum of squares") column by the appropriate degrees of freedom.

In the absence of systematic differences between the sample means, the two mean squares will have the same expected value, and their ratio (the F-statistic) will be near 1. Systematic differences between the sample means will add extra variation into the treatment mean square, with no effect on the residual mean square, resulting in an expected F-statistic that is larger than 1. In the output above, the F-statistic is 0.33, on 3 and 8 degrees of freedom, with $p = 0.33$. There is no convincing indication that there are indeed differences among the treatment means. Interest then turns to teasing out the nature of those differences.

The one-way analysis of variance formally tests whether the variation among the means is greater than what might occur simply because of the natural variation within each group. This comparison is based on the F-statistic, which is given in the output column headed `F value`. An F-statistic that is much larger than 1 points to the conclusion that the means are different. The p-value is designed to assist this judgment.

Figure 4.6 is one of a number of graphical presentation possibilities for a one-way layout. Others are (1) a side-by-side comparison of the histograms – but there are too few values for that; (2) density plots – again there are too few values; and (3) a comparison of the boxplots – this works quite well with 12 values for each treatment.

4.4.1 Multiple comparisons

In Figure 4.6 we gave two "yardsticks" – the LSD and the HSD – against which to compare differences between means. Because neither of these suggested a difference between treatments, the choice between them was not of great consequence. Also, there were just three treatment levels, so that the difference between the LSD and the HSD is not large.

The 5% HSD is designed so that, under the null model (no difference between treatments), the maximum difference will be greater than the HSD in 5% of experiments. In other words, the 5% relates to an experiment-wise error rate, defined as just described. The HSD is an appropriate yardstick against which to compare treatment differences if the demand is for a 5% or other specified probability of finding a difference between the largest and smallest means when there was no difference in the populations from which they were drawn.

Contrast this with the 5% least significant difference (LSD). This is designed, if used without a preliminary F-test, to give a 5% comparison-wise error rate.

A reasonable practical strategy is to do a preliminary analysis of variance F-test. If that suggests differences between the means, then it is of interest to use both yardsticks in comparing them. The LSD gives an anti-conservative yardstick, i.e., one that, in the absence of the preliminary F-test, would be somewhat biased towards finding differences. Tukey's HSD gives a stricter conservative yardstick, i.e., one that is somewhat biased against finding differences. Ignoring changes in degrees of freedom and possible associated changes in the standard error, the HSD will increase as the number of treatment groups that are to be compared increases.

**Microarray data – severe multiplicity*

Multiple tests are a serious issue in the analysis of microarray data, where an individual slide (or sometimes, as for Plate 2, half-slide) may yield information on some thousands

of genes. Each slide (or, here, half-slide) is commonly used to compare, for each of a large number of genes, the gene expression in two samples of genetic material.

The experiment that led to Plate 2 was designed to investigate changes in gene expression between the pre-settlement free-swimming stage of coral, and the post-settlement stage. For 3042 genes (one for each of 3042 spots), which showed an increase in gene expression and which a decrease? Note that each panel in Plate 2 has 3072 spots; this includes 30 blanks. Where there was an increase, the spot should be fairly consistently blue, or bluish, over all six panels. Where there was a decrease, the spot should be fairly consistently yellow, or yellowish.

Here, all that will be attempted is to give broad indications of the experimental procedure, and subsequent processing, that led to the plots shown in Plate 2. The slides are first printed with probes, with one probe per spot. Each probe is designed to check for evidence of the expression of one gene. The two samples are separately labeled so that when later a spot "lights up" under a scanner, it will be possible to check for differences in the signal intensity.

After labeling the separate samples, mixing them, and wiping the mixture over the slide or half-slide, and various laboratory processing steps, a scanner was used to determine, for each spot, the intensities generated from the two samples. Various corrections are then necessary, leading finally to the calculation of logarithms of intensity ratios. Essentially, it is logarithms of intensity ratios that are shown in Plate 2.

For these data there are, potentially, 3042 t-statistics. This is small, by the standards of microarray experiments. There are severe problems of multiplicity to address. Details of a defensible approach to analyzing the data shown in Plate 2 will be posted on the web site for the book.

For further information on the analysis of microarray data, see Smyth (2004). For background on the coral data, see Grasso *et al.* (2008).

4.4.2 Data with a two-way structure, i.e., two factors

Consider now data from an experiment that compared wild type (`wt`) and genetically modified rice plants (`ANU843`), each with three different chemical treatments. A first factor relates to whether `F10` or `NH4Cl` or `NH4NO3` is applied. A second factor relates to whether the plant is wild type (`wt`) or `ANU843`.

There are 72 sets of results, i.e., two types (`variety`) × three chemical treatments (`fert`) × 6 replicates, with the experimental setup repeated across each of two blocks (`Block`). Figures 4.7A and B show alternative perspectives on these data.[13]

[13]
```
## Simplified version of code
library(lattice)
## Panel A
dotplot(trt ~ ShootDryMass, data=rice, aspect=1,
        panel=function(x,y,...){panel.dotplot(x, y, pch=1, col="gray40")
                                panel.average(x, y, type="p", col="black",
                                              pch=3, cex=1.25)},
        xlab="Shoot dry mass (g)")
## Panel B
with(rice, interaction.plot(fert, variety, ShootDryMass,
                            xlab="Level of first factor"))
```

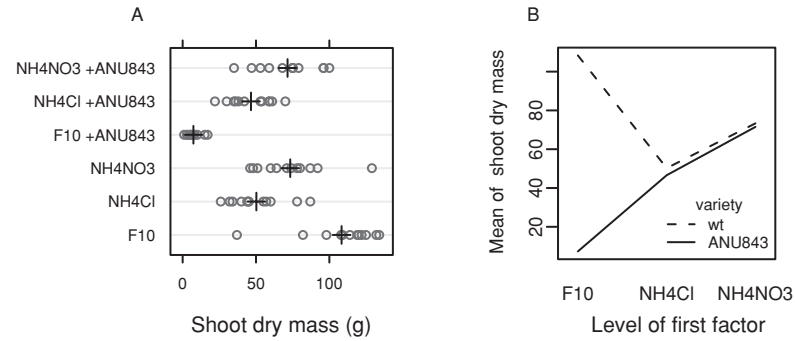


Figure 4.7 Both panels are for rice shoot dry mass data. Panel A shows a one-way strip plot, with different strips for different treatment regimes. Treatment means are shown with a large +. The interaction plot in panel B shows how the effect of fertilizer (the first factor) changes with variety (the second factor). Data relate to Perrine *et al.* (2001).

Figure 4.7B shows a large difference between `ANU843` and wild type (`wt`) for the `F10` treatment. For the other treatments, there is no detectable difference. A two-way analysis will show a large interaction.

Note, finally, that the treatments were arranged in two blocks. In general, this has implications for the analysis. This example will be discussed again in Chapter 7, where block effects will be taken into account.

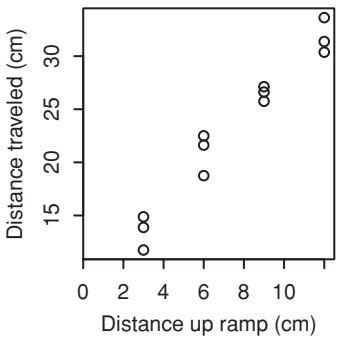
4.4.3 Presentation issues

The discussion so far has treated all comparisons as of equal interest. Often they are not. There are several possibilities:

- Interest may be in comparing treatments with a control, with comparisons between treatments of lesser interest.
- Interest may be in comparing treatments with one another, with any controls used as a check that the order of magnitude of the treatment effect is pretty much what was expected.
- There may be several groups of treatments, with the chief interest in comparisons between the different groups.

Any of these situations should lead to specifying in advance the specific treatment comparisons that are of interest.

Often, however, scientists prefer to regard all treatments as of equal interest. Results may be presented in a graph that displays, for each factor level, the mean and its associated standard error. Alternatives to displaying bars that show the standard error may be to show a 95% confidence interval for the mean, or to show the standard deviation. Displaying or quoting the standard deviation may be appropriate when the interest is not in comparing level means, but in obtaining an idea of the extent to which the different levels are clearly separated.



Starting point	Distance traveled		
3	31.38	30.38	33.63
6	26.63	25.75	27.13
9	18.75	22.50	21.63
12	13.88	11.75	14.88

Figure 4.8 Distance traveled (`distance.traveled`) by model car, as a function of starting point (`starting.point`), up a 20° ramp.

In any case:

- For graphical presentation, use a layout that reflects the data structure, i.e., a one-way layout for a one-way data structure, and a two-way layout for a two-way data structure.
- Explain clearly how error bars should be interpreted – ± SE limits, ± 95% confidence interval, ± SED limits, or whatever. Or if the intention is to indicate the variation in observed values, the SD (standard deviation) may be more appropriate.
- Where there is more than one source of variation, explain what source(s) of "error" is/are represented. It is pointless and potentially misleading to present information on a source of error that is of little or no interest, e.g., on analytical error when the relevant "error" for the treatment comparisons that are of interest arises from fruit-to-fruit variation.

4.5 Response curves

The table shown to the right of Figure 4.8 exhibits data that are strongly structured. A model car was released three times at each of four different distances (`starting.point`) up a 20° ramp. The experimenter recorded distances traveled from the bottom of the ramp across a concrete floor. Figure 4.8 shows a plot of these data.[14] What is the pattern of the response? This should be handled as a regression problem rather than as an analysis of variance problem. It would be particularly nonsensical to examine all pairwise comparison, thus doing violence to the treatment structure, and confusing interpretation. Response curve analyses should be used whenever appropriate in preference to comparison of individual pairs of means.

For these data, the physics can be used to suggest the likely form of response. Where no such help is available, careful examination of the graph, followed by systematic examination of plausible forms of response, may suggest a suitable form of response curve.

[14]
```
## Data frame modelcars (DAAG)
plot(distance.traveled ~ starting.point, data=modelcars,
       xlim=c(0,12.5), xaxs="i", xlab = "Distance up ramp (cm)",
       ylab="Distance traveled (cm)")
```

Table 4.8 *Each tester made two firmness tests on each of five fruit.*

Fruit	Tester	Firmness	Mean
1	1	6.8, 7.3	*7.05*
2	1	7.2, 7.3	*7.25*
3	1	7.4, 7.3	*7.35*
4	1	6.8, 7.6	*7.2*
5	1	7.2, 6.5	*6.85*
6	2	7.7, 7.7	*7.7*
7	2	7.4, 7.0	*7.2*
8	2	7.2, 7.6	*7.4*
9	2	6.7, 6.7	*6.7*
10	2	7.2, 6.8	*7.0*

Steps that are suitable for use with data that appear to follow a relatively simple form of response are:

1. Does a straight line explain the data better than assuming a random scatter about a horizontal line?
2. Does a quadratic response curve offer any improvement?
3. Would a cubic curve do better still?

Notice that at this stage we are not concerned to say that a quadratic or cubic curve is a good description of the data. All we are examining is whether such a curve captures an important part of the pattern of change. If it does, but the curve is still not quite right, it may be worthwhile to look for a different form of curve that does fit the data adequately.

A representation of the response curve in terms of coefficients of orthogonal polynomials provides information that makes it relatively easy to address questions 1–3. Consider, for example, a model that has terms in x and x^2. Orthogonal polynomials re-express this combination of terms in such a way that the coefficient of the "linear" term is independent of the coefficient of the "quadratic" term. Higher-order (cubic, . . .) orthogonal polynomial terms can of course be fitted, and it remains the case that the coefficients are mutually independent. There is some further limited discussion of orthogonal polynomials in Section 7.4. Steel *et al.* (1993) discuss the practical use of orthogonal polynomials in some detail.

4.6 Data with a nested variation structure

Ten apples are taken from a box. A randomization procedure assigns five to one tester, and the other five to another tester. Each tester makes two firmness tests on each of their five fruit. Firmness is measured by the pressure needed to push the flat end of a piece of rod through the surface of the fruit. Table 4.8 gives the results, in N/m^2.

For comparing the testers, we have five experimental units for each tester, not ten. One way to do a t-test is to take means for each fruit. We then have five values (means, italicized) for one treatment, that we can compare with the five values for the other treatment.

What happens if we ignore the data structure, and compare ten values for one tester with ten values for the other tester? This pretends that we have ten experimental units for each tester. The analysis will suggest that the treatment means are more accurate than is really the case. We obtain a pretend standard error that is not the correct standard error of the mean. We are likely to under-estimate the standard error of the treatment difference.

4.6.1 Degrees of freedom considerations

For comparison of two means when the sample sizes n_1 and n_2 are small, it is important to have as many degrees of freedom as possible for the denominator of the t-test. It is worth tolerating possible bias in some of the calculated SEDs in order to gain extra degrees of freedom.

The same considerations arise in the one-way analysis of variance, and we pursue the issue in that context. It is illuminating to plot out, side by side, say 10 SEDs based on randomly generated normal variates, first for a comparison based on 2 d.f., then 10 SEDs for a comparison based on 4 d.f., etc.

A formal statistical test is thus unlikely, unless the sample is large, to detect differences in variance that may have a large effect on the result of the test. It is therefore necessary to rely on judgment. Both past experience with similar data and subject area knowledge may be important. In comparing two treatments that are qualitatively similar, differences in the population variance may be unlikely, unless the difference in means is at least of the same order of magnitude as the individual means. If the means are not much different then it is reasonable, though this is by no means inevitable, to expect that the variances will not be much different.

If the treatments are qualitatively different, then differences in variance may be expected. Experiments in weed control provide an example where it would be surprising to find a common variance. There will be few weeds in all plots where there is effective weed control, and thus little variation. In control plots, or for plots given ineffective treatments, there may be huge variation.

If there do seem to be differences in variance, it may be possible to model the variance as a function of the mean. It may be possible to apply a variance-stabilizing transformation. Or the variance may be a smooth function of the mean. Otherwise, if there are just one or two degrees of freedom per mean, use a pooled estimate of variance unless the assumption of equal variance seems clearly unacceptable.

4.6.2 General multi-way analysis of variance designs

Generalization to multi-way analysis of variance raises a variety of new issues. If each combination of factor levels has the same number of observations, and if there is no structure in the *error* (or *noise*), the extension is straightforward. The extension is less straightforward when one or both of these conditions are not met. For unbalanced data from designs with a simple error structure, it is necessary to use the `lm()` (linear model) function. The `lme()` function in the *nlme* package, or alternatively `lmer()` in the *lme4* package, is able to handle problems where there is structure in the error term, including data from unbalanced designs. See Chapter 9 for further details.

Table 4.9 *These are the same data as in Table 3.1.*

	Pair #								
	1	2	3	4	5	6	7	8	9
Heated (mm)	244	255	253	254	251	269	248	252	292
Ambient	225	247	249	253	245	259	242	255	286
Difference	19	8	4	1	6	10	6	−3	6

4.7 Resampling methods for standard errors, tests, and confidence intervals

There are many different resampling methods. All rely on the selection of repeated samples from a "population" that is constructed using the sample data. In general, there are too many possible samples to take them all, and we therefore rely on repeated random samples. In this section, we demonstrate permutation and bootstrap methods. We start with permutation tests, illustrating their use for the equivalent of one-sample and two-sample t-tests.

4.7.1 The one-sample permutation test

Consider the paired elastic band data of Table 3.1 again, reproduced here as Table 4.9.

If the treatment has made no difference, then an outcome of 244 for the heated band and 225 for the ambient band might equally well have been 225 for the heated band and 244 for the ambient band. A difference of 19 becomes a difference of −19. There are $2^9 = 512$ permutations, and a mean difference associated with each permutation. We then locate the mean difference for the data that we observed within this permutation distribution. The p-value is the proportion of values that are as large in absolute value as, or larger than, the mean for the data.

The absolute values of the nine differences are

Difference	19	8	4	1	6	10	6	3	6

In the permutation distribution, these each have an equal probability of taking a positive or a negative sign. There are 2^9 possibilities, and hence $2^9 = 512$ different values for $\bar{d}$. The possibilities that give a mean difference that is as large as or larger than in the actual sample, where the value for pair 8 has a negative sign, are

Difference	19	8	4	1	6	10	6	3	6
	19	8	4	−1	6	10	6	3	6
	19	8	4	1	6	10	6	−3	6

There are another three possibilities that give a mean difference that is of the same absolute value, but negative. Hence $p = 6/512 = 0.0117$.

In general, when the number of pairs is large, it will not be feasible to use such an enumeration approach to get information on relevant parts of the upper and lower tails of the distribution. We therefore take repeated random samples from the permutation

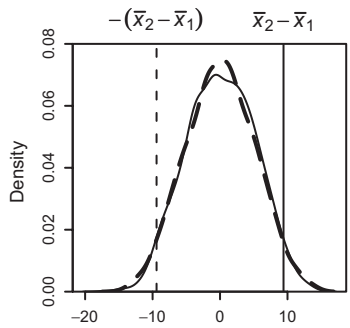


Figure 4.9 Density curves for two samples of 2000 each from the permutation distribution of the difference in means, for the two-sample elastic band data.

distribution. The function `onetPermutation()` in our *DAAG* package may be used for this.

4.7.2 The two-sample permutation test

Suppose we have n_1 values in one group and n_2 in a second, with $n = n_1 + n_2$. The permutation distribution results from taking all possible samples of n_2 values from the total of n values. For each such sample, we calculate

mean of the n_2 values that are selected – mean of remaining n_1 values.

The permutation distribution is the distribution of all such differences of means. We locate the differences of means for the actual samples within this permutation distribution.

The calculation of the full permutation distribution is not usually feasible. We therefore take perhaps 1000 samples from this distribution. The function `twot.permutation()` that is in our *DAAG* package may be used for this repeated sampling.

Thus consider the data from Subsection 4.2.2:

Ambient: 254 252 239 240 250 256 267 249 259 269 (mean = 253.5)
Heated: 233 252 237 246 255 244 248 242 217 257 254 (mean = 244.1)

Figure 4.9 shows two estimates of the permutation distribution that were obtained by taking, in each instance, 2000 random samples from this distribution. The point where the difference in means falls with respect to this distribution (253.5 − 244.1 = 9.4) has been marked, as has minus this difference.[15]

[15]
```
## Draw one curve only; permutation distribution of difference in means
x1 <- two65$ambient; x2 <- two65$heated; x <- c(x1, x2)
n1 <- length(x1); n2 <- length(x2); n <- n1+n2
dbar <- mean(x2) - mean(x1)
z <- numeric(2000)
for(i in 1:2000){
    mn <- sample(n, n2, replace=FALSE)
    dbardash <- mean(x[mn]) - mean(x[-mn])
    z[i]<- dbardash
    }
plot(density(z), yaxs="i")
abline(v = dbar)
abline(v = -dbar, lty=2)
signif((sum(z > abs(dbar)) + sum (z< -abs(dbar)))/length(z), 3)
```

The density estimate corresponding to the solid line gave a p-value of 0.051. The density estimate corresponding to the dashed line gave a p-value of 0.060. Use of a larger sample size will of course lead to more accurate p-values.

4.7.3 Estimating the standard error of the median: bootstrapping*

The formula given in Subsection 4.1.3 for the SEM has the same form, irrespective of the distribution, providing that the sample is chosen randomly. By contrast, the formula for the standard error of the median (Subsection 4.1.5) applies only when data are normally distributed.

The bootstrap estimate of the standard error of the median avoids this requirement, and avoids also the need to look for some alternative distribution that may be a better fit to the data. A comparison between the bootstrap estimate and the normal theory estimate allows an assessment of the seriousness of any bias that may result from non-normality. We proceed to calculate the bootstrap estimate of the standard error for the median length for the eggs that were in wrens' nests. (The *boot* package (Canty, 2002) is needed for all bootstrap examples.) We will use the result as a check on our earlier computation.

The idea is as follows. In estimating the standard error of the median, we are seeking the standard deviation of medians that could be obtained for all possible samples of egg lengths in wrens' nests. Of course, we have access to one sample only, but if our sample is of reasonable size and has been collected properly it should give us a good approximation to the entire population.

We estimate the standard deviation of the median by computing sample medians for each of the resamples and taking the standard deviation of all of these medians. Even though the resamples are not genuine new samples, this estimate for the standard error of the median has good statistical properties, for purposes of estimating the standard error of the median.

Here is the output from R for the egg lengths from the wrens' nests:

```
> ## bootstrap estimate of median of wren length: data frame cuckoos
(DAAG)
> wren <- split(cuckoos$length, cuckoos$species)$wren
> library(boot)
> ## First define median.fun(), with two required arguments:
> ##         data specifies the data vector,
> ##         indices selects vector elements for a each resample
> median.fun <- function(data, indices){median(data[indices])}
> ## Call boot(), with statistic=median.fun, R = # of resamples
> (wren.boot <- boot(data = wren, statistic = median.fun, R = 999))
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = wren, statistic = median.fun, R = 999)

Bootstrap Statistics :
    original  bias    std. error
t1*     21  0.061       0.225
```

The original estimate of the median was 21. The bootstap estimate of the standard error is 0.225, based on 999 resamples. Compare this with the slightly larger standard error estimate of 0.244 given by the normal theory formula in Subsection 4.1.5. The bootstrap estimate of the standard error will of course differ somewhat between different runs of the calculation. Also given is an estimate of the bias, i.e., of the tendency to under- or over-estimate the median.

4.7.4 Bootstrap estimates of confidence intervals

The usual approach to constructing confidence intervals is based on a statistical theory that relies, in part, on the assumption of normally distributed observations. Sometimes this theory is too complicated to work out, and/or the normal assumption is not applicable. In such cases, the bootstrap may be helpful. We demonstrate the use of the methodology to calculate confidence intervals for the median and for the correlation.

Several different confidence intervals can be calculated using bootstrap replicates of the data. The function `boot.ci()` handles five of these. The `perc` (percentile) type is probably the most commonly used method; it is not the most accurate. The `bca` type (bias corrected accelerated or BC_a) will often give a substantial improvement. Efron and Tibshirani (1993) give a clear description of these methods, together with theoretical justification for the use of the BC_a method.

Bootstrap 95% confidence intervals for the median

As when computing bootstrap standard errors, we calculate sample medians for a large number of resamples. The endpoints for the 95% percentile confidence interval are calculated as the 2.5 and 97.5 percentiles of the resulting distribution of medians. The endpoints for the BC_a confidence interval are calculated in a more complicated way; Efron and Tibshirani (1993) can be consulted for the details.

```
> median.fun <- function(data, indices){median(data[indices])}
> ## Call the boot() function, with statistic=median.fun
> wren <- cuckoos[cuckoos$species=="wren", "length"]
> (wren.boot <- boot(data=wren, statistic=median.fun, R=9999))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 9999 bootstrap replicates

CALL :
boot.ci(boot.out = wren.boot, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   (20.9, 22.0 )   (20.0, 21.0 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

Interestingly, the BC_a interval is slightly narrower than the cruder percentile interval in this example. The warning may be taken as an indication that the calculation should be

run again, with a larger number (perhaps 99 999) of resamples. Such warnings may arise because of outliers in the data. Use `qqnorm(wren)` to check that this is not an issue for these data.

The correlation coefficient

Bootstrap methods do not require bivariate normality. Independence between observations, i.e., between (x, y) pairs, is as important as ever. Note however that a correlation of, e.g., 0.75 for a non-normal distribution may have quite different implications from a correlation of 0.75 when normality assumptions apply.

We will compute 95% confidence intervals for the correlation between `chest` and `belly` for the `possum` data frame:[16]

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 9999 bootstrap replicates

CALL :
boot.ci(boot.out = possum.boot, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   ( 0.476,  0.709 )   ( 0.468,  0.704 )
Calculations and Intervals on Original Scale
```

The bootstrap – parting comments

Bootstrap methods are not a panacea. We must respect the structure of the data; any form of dependence in the data must be taken into account. There are contexts where the bootstrap is invalid and will mislead. As a rough guideline, the bootstrap is unlikely to be satisfactory for statistics, including maximum, minimum and range, that are functions of sample extremes. The bootstrap is usually appropriate for statistics from regression analysis – means, variances, coefficient estimates, and correlation coefficients. It also works reasonably well for medians and quartiles, and other such statistics. See Davison and Hinkley (1997), Efron and Tibshirani (1993). See also the references in the help page for the `boot()` function in the *boot* package.

4.8* Theories of inference

Formal statistical methodologies are of two broad types: frequentist and Bayesian. The frequentist approach is usually based on the concept of likelihood; given the model, what is the probability of obtaining a sample similar to that observed? Parameter values are

[16]
```
## Bootstrap estimate of 95% CI of cor(chest, belly): data frame possum (DAAG)
possum.fun <- function(data, indices) {
    chest <- data$chest[indices]
    belly <- data$belly[indices]
    cor(belly, chest)
    }
possum.boot <- boot(possum, possum.fun, R=9999)
boot.ci(possum.boot, type=c("perc", "bca"))
```

assumed to be unknown constants, and estimates are chosen to maximize this likelihood. This has been the approach that we have followed for most of this chapter.

Another type of methodology, broadly known as "Bayesian" uses Bayes' theorem. The essential idea is that we might have prior information (knowledge or belief) about the distribution of a parameter value before taking a sample of observations. This prior information can be updated using the sample and the rules of probability.

4.8.1 Maximum likelihood estimation

Consider the model

$$y_i = \mu + \varepsilon_i, \quad i = 1, 2, \ldots, n$$

where μ is an unknown constant, and where the errors ε are assumed to be independent and normally distributed with mean 0 and variance σ^2.

The probability density for the ith y-value is normal with mean μ and variance σ^2. Because of the independence assumption, the probability density of the entire sample of ys is simply the product of these normal densities. This product is the likelihood. The maximum likelihood estimates are the values of μ and σ which maximize this function. A calculus argument can be used to see that the estimates are $\bar{y}$ and $s\sqrt{(n-1)/n}$.

Note that the usual estimator of the standard deviation differs slightly from the maximum likelihood estimator; the denominator in the usual variance estimate is the number of degrees of freedom ($n - 1$ in this case), while it is n for the maximum likelihood estimate; this difference is negligible in large samples.

For an example, consider the observed differences between `heated` and `ambient`, assuming an independent normal errors model:

```
funlik <- function(mu, sigma, x=with(pair65, heated-ambient))
  prod(dnorm(x, mu, sigma))
```

In practice, it is more convenient to work with the loglikelihood, rather than the likelihood. Maximizing on the log scale leads to exactly the same estimates as on the original scale. Try the following:

```
> log(funlik(6.3, 6.1))     # Close to estimated mean and SD
[1] -28.549
> log(funlik(6.33, 5.75))   # Close to the actual mle's
[1] -28.520
> log(funlik(7, 5.75))
[1] -28.580
```

4.8.2 Bayesian estimation

As noted earlier, the Bayesian methodology provides a way to update our prior information about the model parameters using sample information.

Usually, the prior information is summarized in the form of a probability law called the *prior distribution* of the model parameters. Interest usually centers on the *posterior distribution* of the parameters, which is proportional to the product of the likelihood and the prior distribution.

A simple application of Bayes' theorem is as follows. The incidence of HIV in adult Australian males (15–49 years) who do not have any known risk factor may be of the order of 1 in 100 000, i.e., the prior probability of infection is 0.00001. A person in this group has an initial test (for example, it may be required in order to obtain a US green card) that has a specificity of 0.01%, i.e., for every 10 000 people tested, there will on average be one false positive. How should such an individual interpret the result? If 100 000 individuals take the test, one will on average have AIDS and will almost certainly return a positive test. On the other hand there will, on average, be close to 10 false positives (0.1% of 99 999).

Not infected	Infected
$10000 \times 0.001 = 10$ (false) positives	1 true positive

The posterior odds that the person has AIDS are thus close to 1:10, certainly a narrowing from the prior odds of 1:99 999.

Note that, as often happens when Bayesian calculations are used, the prior information is not very precise. What we can say is that the prior probability is, in the case mentioned, very low.

Bayesian estimation with normal prior and normal likelihood

A relatively simple example is that of a normal likelihood (as considered in the previous section) where the unobserved true mean is now also assumed to have a normal distribution, but this time with mean μ_0 and variance σ_0^2. The posterior density of the mean is then normal with mean

$$\frac{n\bar{y} + \mu_0\sigma^2/\sigma_0^2}{n + \sigma^2/\sigma_0^2}$$

and variance

$$\frac{\sigma^2}{n + \sigma^2/\sigma_0^2}.$$

This assumes that σ^2 is actually known; an estimate can be obtained using the sample variance. Alternatively, we could put a prior distribution on this parameter as well.

In problems where the model contains many parameters, each with its own prior distribution, the exact calculation of the posterior distribution for each parameter can be quite involved. Fortunately, in recent years, a simulation technique (called Markov Chain Monte Carlo, or MCMC) has been shown to give very effective approximations to these posterior distributions. Calculations must run for long enough that the posterior distribution reaches a steady state that is independent of the starting values of parameters. The steady state or stationary distribution is designed to be the posterior distribution of the parameter(s) of interest.

Exercise 12 in Chapter 3, and the two following exercises, demonstrated the simulation of finite state Markov chains. Section 5.9 will demonstrate the use of Bayesian MCMC for straight line regression.

4.8.3 *If there is strong prior information, use it!*

Any methodology that ignores strong prior information is inappropriate, and may be highly misleading. Diagnostic testing (the AIDS test example mentioned above) and criminal investigations provide cogent examples.

Using the hypothesis testing framework, we take the null hypothesis H_0, in the AIDS test example, to be the hypothesis that the individual does not have HIV. Given this null hypothesis, the probability of a positive result is 0.0001. Therefore the null hypothesis is rejected. As we saw, the prior information makes such a conclusion entirely inappropriate.

In a serious criminal case, the police might scrutinize a large number of potential perpetrators. A figure of 10 000 or more is entirely within the range of possibility. Suppose there is a form of incriminating evidence that is found in one person in 1000.

Scrutiny of 10 000 potential perpetrators will on average net 10 suspects. Suppose one of these is later charged. The probability of such incriminating evidence, assuming that the defendant is innocent, is indeed 0.001. The police screening will net around 10 innocent people along with, perhaps, the one perpetrator. The following summarizes the expected result of the police search for a suspect. It is optimistic in its assumption that the perpetrator will be among those netted.

Not the perpetrator	The perpetrator
$10000 \times 0.001 = 10$ (false) positives	1 true positive

This evidence leads to odds of 1:10 or worse, i.e., less than 10%, that the defendant is guilty. On its own, it should be discounted.

The interpretation of results from medical tests for AIDS is discussed in detail in Gigerenzer (2002). The calculations just given made an informal use of Bayesian methodology. Such an approach is essential when, as here, there is strong prior knowledge. Where there is no strong prior knowledge and the prior assessment of probabilities is little more than a guess, a Bayesian analysis may nevertheless be insightful.

4.9 Recap

Dos and don'ts

- Do examine appropriate plots.
- Ensure that the analysis and graphs reflect any important structure in the data.
- Always present means, standard errors, and numbers for each group. Results from formal significance tests have secondary usefulness.
- The use of a large number of significance tests readily leads to data summaries that lack coherence and insight. Consider whether there is an alternative and more coherent analysis that would provide better insight.
- Reserve multiple range tests for unstructured data.
- Think about the science behind the data. What analysis will best reflect that science?

- The aim should be an insightful and coherent account of the data, placing it in the context of what is already known. Ensure that the statistical analysis assists this larger purpose.

4.10 Further reading

On general issues of style and approach, see Wilkinson and Task Force on Statistical Inference (1999), Maindonald (1992), Krantz (1999) and Gigerenzer (1998, 2002). See also the statistical good practice guidelines at the web site `http://www.ssc.rdg.ac.uk/publications/publications.html`. Miller (1986) has extensive comment on consequences of failure of assumptions, and on how to handle such failures. On the design of experiments, and on analysis of the resulting data, see Cox (1958), Cox and Reid (2000). We include further brief discussion of the design and analysis of experiments in Chapter 10.

Formal hypothesis testing, which at one time had become almost a ritual among researchers in psychology, is now generating extensive controversy, reflected in the contributions to Harlow *et al*. (1997). The review of the Harlow *et al*. book in Krantz (1999) is a good guide to the controversy. See also Gigerenzer (1998), Wilkinson and Task Force on Statistical Inference (1999), Nicholls (2000).

Gigerenzer *et al*. (1989) give interesting historical background to different styles and approaches to inference that have grown up in one or other area of statistical application. Wonnacott and Wonnacott (1990) have an elementary account of Bayesian methodology. See also Gelman *et al*. (2003), Gill (2008). There is a helpful brief summary of Bayesian methodology, including Bayesian modeling, in Chapters 4, 6 and 7 of Bolker (2008). Gigerenzer (2002) demonstrates the use of Bayesian arguments in several important practical contexts, including AIDS testing and screening for breast cancer.

Chapter 4 of Senn (2003) has interesting comments on competing theories of statistical inference. Young and Smith (2005) give a terse and remarkably comprehensive exposition of competing theories, which does however make relatively severe technical demands.

References for further reading

Bolker, B. M. 2008. *Ecological Models and Data in R*.

Cox, D. R. 1958. *Planning of Experiments.*

Cox, D. R. and Reid, N. 2000. *Theory of the Design of Experiments.*

Gelman, A. B., Carlin, J. S., Stern, H. S. and Rubin, D. B. 2003. *Bayesian Data Analysis*, 2nd edn.

Gigerenzer, G. 1998. We need statistical thinking, not statistical rituals. *Behavioural and Brain Sciences* 21: 199–200.

Gigerenzer, G. 2002. *Reckoning with Risk: Learning to Live with Uncertainty*.

Gigerenzer, G., Swijtink, Z., Porter, T., Daston, L., Beatty, J. and Krüger, L. 1989. *The Empire of Chance.*

Gill, J. 2008. *Bayesian Methods: A Social and Behavioral Sciences Approach*, 2nd edn.

Harlow, L. L., Mulaik, S. A. and Steiger, J. H. (eds). 1997. *What If There Were No Significance Tests?*

Krantz, D. H. 1999. The null hypothesis testing controversy in psychology. *Journal of the American Statistical Association* 44: 1372–81.

Maindonald, J. H. 1992. Statistical design, analysis and presentation issues. *New Zealand Journal of Agricultural Research* 35: 121–41.

Miller, R. G. 1986. *Beyond ANOVA, Basics of Applied Statistics.*

Nicholls, N. 2000. The insignificance of significance testing. *Bulletin of the American Meteorological Society* 81: 981–6.

Senn, S. 2003. *Dicing with Death: Chance, Risk and Health.*

Wilkinson, L. and Task Force on Statistical Inference. 1999. Statistical methods in psychology journals: guidelines and explanation. *American Psychologist* 54: 594–604.

Wonnacott, T. H. and Wonnacott, R. 1990. *Introductory Statistics*, 5th edn.

Young, G. and Smith, R. L. 2005. *Essentials of Statistical Inference*.

4.11 Exercises

1. Using the data set `nswdemo` (*DAAG*), determine 95% confidence intervals for: (a) the 1975 mean incomes of each group; (b) the 1978 mean incomes of each group. Finally, calculate a 95% confidence interval for the difference in mean income between treated and controls in 1978.

2. Draw graphs that show, for degrees of freedom between 1 and 100, the change in the 5% critical value of the t-statistic. Compare a graph on which neither axis is transformed with a graph on which the respective axis scales are proportional to log(t-statistic) and log(degrees of freedom). Which graph gives the more useful visual indication of the change in the 5% critical value of the t-statistic with increasing degrees of freedom?

3. Generate a random sample of 10 numbers from a normal distribution with mean 0 and standard deviation 2. Use `t.test()` to test the null hypothesis that the mean is 0. Now generate a random sample of 10 numbers from a normal distribution with mean 1.5 and standard deviation 2. Again use `t.test()` to test the null hypothesis that the mean is 0. Finally write a function that generates a random sample of n numbers from a normal distribution with mean μ and standard deviation 1, and returns the p-value for the test that the mean is 0.

4. Use the function that was created in Exercise 3 to generate 50 independent p-values, all with a sample size $n = 10$ and with mean $\mu = 0$. Use `qqplot()`, with the argument setting `x = qunif(ppoints(50))`, to compare the distribution of the p-values with that of a uniform random variable, on the interval [0, 1]. Comment on the plot.

5. The following code draws, in a 2×2 layout, 10 boxplots of random samples of 1000 from a normal distribution, 10 boxplots of random samples of 1000 from a t-distribution with 7 d.f., 10 boxplots of random samples of 200 from a normal distribution, and 10 boxplots of random samples of 200 from a t-distribution with 7 d.f.:

```
oldpar <- par(mfrow=c(2,2))
tenfold1000 <- rep(1:10, rep(1000,10))
boxplot(split(rnorm(1000*10), tenfold1000), ylab="normal - 1000")
boxplot(split(rt(1000*10, 7), tenfold1000),
        ylab=expression(t[7]*" - 1000"))
tenfold100 <- rep(1:10, rep(100, 10))
boxplot(split(rnorm(100*10), tenfold100), ylab="normal - 100")
boxplot(split(rt(100*10, 7), tenfold100),
        ylab=expression(t[7]*" - 100"))
par(oldpar)
```

Refer back to the discussion of heavy-tailed distributions in Subsection 3.2.2, and comment on the different numbers and configurations of points that are flagged as possible outliers.

6. Here we generate random normal numbers with a sequential dependence structure:
```
y1 <- rnorm(51)
y <- y1[-1] + y1[-51]
acf(y1)               # acf is  'autocorrelation function'
                      # (see Chapter 9)
acf(y)
```
Repeat this several times. There should be no consistent pattern in the `acf` plot for different random samples `y1`. There will be a fairly consistent pattern in the `acf` plot for `y`, a result of the correlation that is introduced by adding to each value the next value in the sequence.

7. Create a function that does the calculations in the first two lines of the previous exercise. Put the calculation in a loop that repeats 25 times. Calculate the mean and variance for each vector `y` that is returned. Store the 25 means in the vector `av`, and store the 25 variances in the vector `v`. Calculate the variance of `av`.

8. The following use the data frame `nswpsid3`, created as in footnote 8:

   (a) For each column of the data set `nswpsid3` after the first, compare the control group (`trt==0`) with the treatment group (`trt==1`). Use overlaid density plots to compare the continuous variables, and two-way tables to compare the binary (0/1) variables. Where are the greatest differences?
   (b) Repeat the comparison, but now for the data set `nswdemo`.
   (c) Compare and contrast the two sets of results. Read carefully the help pages for `psid3` and for `nswdemo`, and comment on why the different thrust of the two sets of results is perhaps not surprising.

9. In a study that examined the use of acupuncture to treat migraine headaches, consenting patients on a waiting list for treatment for migraine were randomly assigned in a 2:1:1 ratio to acupuncture treatment, a "sham" acupuncture treatment in which needles were inserted at non-acupuncture points, and waiting-list patients whose only treatment was self-administered (Linde *et al*., 2005). (The "sham" acupuncture treatment was described to trial participants as an acupuncture treatment that did not follow the principles of Chinese medicine.) Analyze the following two tables. What, in each case, are the conclusions that should be drawn from the analyses? Comment on implications for patient treatment and for further research:

   (a) Outcome is classified according to numbers of patients who experienced a greater than 50% reduction in headaches over a four-week period, relative to a pre-randomization baseline:

	Acupuncture	Sham acupuncture	Waiting list
$\geq$ 50% reduction	74	43	11
$<$ 50% reduction	71	38	65

   (b) Patients who received the acupuncture and sham acupuncture treatments were asked to guess their treatment. Results were:

	Acupuncture	Sham acupuncture
Chinese	82	30
Other	17	26
Don't know	30	16

10. Use `mosaicplot()` to display the table `rareplants` (Subsection 4.3.1) that was created using code in footnote 11. Annotate the mosaic plot to draw attention to the results that emerged from the analysis in Subsection 4.3.1.

11. The table `UCBAdmissions` was discussed in Subsection 2.2.1. The following gives a table that adds the 2×2 tables of admission data over all departments:

```
## UCBAdmissions is in the datasets package
## For each combination of margins 1 and 2, calculate the sum
UCBtotal <- apply(UCBAdmissions, c(1,2), sum)
```

What are the names of the two dimensions of this table?

(a) From the table `UCBAdmissions`, create mosaic plots for each faculty separately. (If necessary refer to the code given in the help page for `UCBAdmissions`.)

(b) Compare the information in the table `UCBtotal` with the result from applying the function `mantelhaen.test()` to the table `UCBAdmissions`. Compare the two sets of results, and comment on the difference.

(c) The Mantel–Haenzel test is valid only if the male-to-female odds ratio for admission is similar across departments. The following code calculates the relevant odds ratios:

```
apply(UCBAdmissions, 3, function(x)
      (x[1,1]*x[2,2])/(x[1,2]*x[2,1]))
```

Is the odds ratio consistent across departments? Which department(s) stand(s) out as different? What is the nature of the difference?

[For further information on the Mantel–Haenszel test, see the help page for `mantelhaen.test` and Agresti (2002, pp. 287f).]

12. Tables 4.10A and B contain fictitious data that illustrate issues that arise in combining data across tables. To enter the data for Table 4.10A, type:

```
admissions <- array(c(30,30,10,10,15,5,30,10),
                    dim=c(2,2,2))
```

and similarly for Table 4.10B. The third dimension in each table is faculty, as required for using faculty as a stratification variable for the Mantel–Haenzel test. From the help page for `mantelhaen.test()`, extract and enter the code for the function `woolf()`. Apply the function `woolf()`, followed by the function `mantelhaen.test()`, to the data of each of Tables 4.10A and B. Explain, in words, the meaning of each of the outputs. Then apply the Mantel–Haenzel test to each of these tables.

13. The function `overlapDensity()` in the *DAAG* package can be used to visualize the unpaired version of the *t*-test. Type in

```
## Compare densities for ambient & heated: list two65 (DAAG)
with(two65, overlapDensity(ambient, heated))
  # Do overlapDensity(ambient, heated) with ambient and heated
  # taken, if not found elsewhere, from the columns of two65
```

Table 4.10 *These illustrate the dangers of adding over contingency tables. In B, biases that go in different directions in the two faculties have canceled in the table of totals.*

A:	Engineering			Sociology			Total	
	Male	Female		Male	Female		Male	Female
Admit	30	10	Admit	15	30	Admit	45	40
Deny	30	10	Deny	5	10	Deny	35	20
B:								
Admit	30	20	Admit	10	20	Admit	40	40
Deny	30	10	Deny	5	25	Deny	35	35

in order to observe estimates of the stretch distributions of the ambient (control) and heated (treatment) elastic bands.

14.* For constructing bootstrap confidence intervals for the correlation coefficient, it is advisable to work with the Fisher z-transformation of the correlation coefficient. The following lines of R code show how to obtain a bootstrap confidence interval for the z-transformed correlation between `chest` and `belly` in the `possum` data frame. The last step of the procedure is to apply the inverse of the z-transformation to the confidence interval to return it to the original scale. Run the following code and compare the resulting interval with the one computed without transformation. Is the z-transform necessary here?

```
z.transform <- function(r) .5*log((1+r)/(1-r))
z.inverse <- function(z) (exp(2*z)-1)/(exp(2*z)+1)
possum.fun <- function(data, indices) {
     chest <- data$chest[indices]
     belly <- data$belly[indices]
     z.transform(cor(belly, chest))}
possum.boot <- boot(possum, possum.fun, R=999)
z.inverse(boot.ci(possum.boot, type="perc")$percent[4:5])
 # See help(bootci.object). The 4th and 5th elements of
 # the percent list element hold the interval endpoints.
```

15. The 24 paired observations in the data set `mignonette` were from five pots. The observations are in order of pot, with the numbers 5, 5, 5, 5, 4 in the respective pots. Plot the data in a way that shows the pot to which each point belongs. Also do a plot that shows, by pot, the differences between the two members of each pair. Do the height differences appear to be different for different pots?

16. Add code to the function `mean.and.sd()`, defined in Subsection 1.4.3 for calculation of a 95% confidence interval for the mean. Recall that the multiplier for the standard error is `qt(0.975, nu)`, where `nu` is the number of degrees of freedom for the standard deviation estimate.

17. Use the function `rexp()` to simulate 100 random observations from an exponential distribution with rate 1. Use the bootstrap (with 99 999 replications) to estimate the standard error

of the median. Repeat several times. Compare with the result that would be obtained using the normal approximation, i.e., $\sqrt{\pi/(2n)}$.

18. Low doses of the insecticide toxaphene may cause weight gain in rats (Chu *et al.*, 1988). A sample of 20 rats are given toxaphene in their diet, while a control group of 8 rats are not given toxaphene. Assume further that weight gain among the treated rats is normally distributed with a mean of 60 g and a standard deviation of 30 g, while weight gain among the control rats is normally distributed with a mean of 10 g and a standard deviation of 50 g. Using simulation, compare confidence intervals for the difference in mean weight gain, using the pooled variance estimate and the Welch approximation. Which type of interval is correct more often?

    Repeat the simulation experiment under the assumption that the standard deviations are 40 g for both samples. Is there a difference between the two types of interval now? *Hint:* Is one of the methods giving systematically larger confidence intervals? Which type of interval do you think is best?

19. The following simulation experiment investigates least-squares estimation of the mean:

```
set.seed(32083)
x <- rnorm(100, mean=3, sd=7)
```

    Next, confirm that the sample mean of the values in `x` is near 4.642. Now plot the sum of squared deviations from μ:

$$\sum_{i=1}^{100}(x_i - \mu)^2$$

    as a function of μ, using the following code:

```
lsfun <- function(mu) apply(outer(x, mu, "-")^2, 2, sum)
curve(lsfun, from=4.6, to=4.7)
```

    Repeat this experiment for different samples, noting where the minimum of the sum of the squares is located each time.

20.* Experiment with the `pair65` example and plot various views of the likelihood function, either as a surface using the `persp()` function or as one-dimensional profiles using the `curve()` function. Is there a single maximizer? Where does it occur?

21.* Suppose the mean reaction time to a particular stimulus has been estimated in several previous studies, and it appears to be approximately normally distributed with mean 0.35 seconds with standard deviation 0.1 seconds. On the basis of 10 new observations, the mean reaction time is estimated to be 0.45 seconds with an estimated standard deviation of 0.15 seconds. Based on the sample information, what is the maximum likelihood estimator for the true mean reaction time? What is the Bayes' estimate of the mean reaction time?

22.* Plot the likelihood function for the mean in the normal errors model for the `pair65` differences. Assume a normal prior distribution with mean 6.0 and standard deviation 5.0. Plot the prior distribution as well as the posterior distribution of the mean. Is it reasonable to view the posterior distribution as a compromise between the prior distribution and the likelihood?

5

Regression with a single predictor

Data for which the models of this chapter may be appropriate can be displayed as a scatterplot. The focus will be on the straight line model, though the use of transformations makes it possible to accommodate specific forms of non-linear relationship within this framework. By convention, the x-variable, plotted on the horizontal axis, has the role of explanatory variable. The y-variable, plotted on the vertical axis, has the role of response or outcome variable.

Many of the issues that arise for these simple regression models are fundamental to any study of regression methods. Various special applications of linear regression raise their own specific issues. One such special application, discussed in Subsection 5.6.2, is to size and shape data.

Scrutiny of the scatterplot should precede regression calculations. Such a plot may indicate that the intended regression is plausible, or it may reveal unexpected features.

If there are many observations, it is often useful to compare the fitted line with a fitted smooth curve. If this differs substantially from an intended line, then straight line regression may be inappropriate, as in Figure 2.6. The fitting of such smooth curves will be a major focus of Chapter 7.

5.1 Fitting a line to data

How accurate is the line?

Application of the summary function to an `lm` object from a straight line regression, as in Subsection 5.1.1 following, gives a standard error for each of a and b. Standard errors of predicted values may also or alternatively be of interest; we defer discussion of these until Section 5.3. Determination of these standard errors requires the specific statistical assumptions that will now be noted.

In the model

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

the assumptions are that given x_i, the response y_i is from a normal distribution with mean $\alpha + \beta x_i$, and that the y_i are sampled independently. Equivalently, the ε_i are independently and identically distributed as normal variables with mean 0 and variance σ^2.

With different assumptions (e.g., a sequential correlation between successive data points), the standard errors will be different.

5.1.1 Summary information – lawn roller example

As described in Subsection 3.1.2, we use the R model formula `depression ~ weight` to supply the model information to the function `lm()`. We then use `summary()` to display the output:[1]

```
> library(DAAG)
> ## Fit lm model: data from roller (DAAG); output in roller.lm
> roller.lm <- lm(depression ~ weight, data = roller)
> ## Use the extractor function summary() to summarize results
> summary(roller.lm)

Call:
lm(formula = depression ~ weight, data = roller)

Residuals:
   Min     1Q Median     3Q    Max
 -8.18  -5.58  -1.35   5.92   8.02

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2.09       4.75   -0.44   0.6723
weight          2.67       0.70    3.81   0.0052

Residual standard error: 6.74 on 8 degrees of freedom
Multiple R-Squared: 0.644,      Adjusted R-squared:  0.6
F-statistic: 14.5 on 1 and 8 DF,  p-value: 0.00518
```

Following a numerical summary of the residuals, there is a table of the estimated regression coefficients and their standard errors. The intercept of the fitted line is $a = -2.09$ (SE = 4.75), while the estimated slope is $b = 2.67$ (SE = 0.70).

The p-value for the slope (testing the null hypothesis that $\beta =$ true slope $= 0$) is small, consistent with the evident linear trend. The p-value for the intercept (testing $\alpha = 0$) is 0.67, i.e., the difference from zero may well be random sampling error. Thus, consistently with the intuition that depression should be proportional to weight, it would be reasonable to fit a model that lacks an intercept term. We leave this as an exercise for the reader.[2]

The standard deviation of the noise term, here identified as the residual standard error, is 6.735. We defer comment on R^2 and the F-statistic until Subsection 5.1.4.

5.1.2 Residual plots

The residuals provide information about the noise term in the model, and allow limited checks on model assumptions. Note however that, in such a small data set, departures from assumptions will be hard to detect.

[1] `## Global options used for this and most later output`
`options(show.signif.stars=FALSE, digits=4)`
`  # show.signif.stars=FALSE suppresses interpretive features that,`
`  # for our use of the output, are unhelpful.`

[2] `## Fit model that omits intercept term; i.e., y = bx`
`lm(depression ~ -1 + weight, data=roller)`

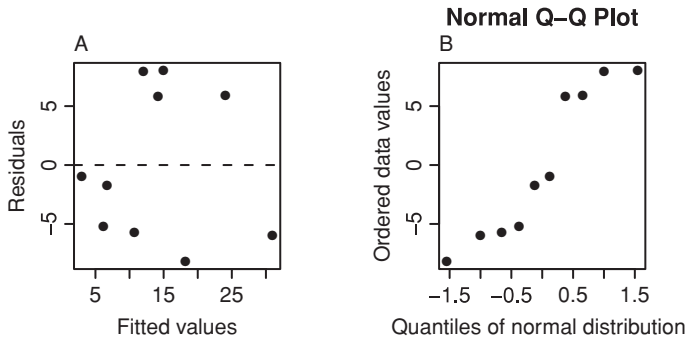


Figure 5.1 Diagnostic plots for the regression of Figure 3.2A. Panel A plots residuals against fitted values. Panel B is a normal probability plot of residuals, giving a visual check whether data are consistent with a normal error distribution.

Two common checks, both available by using the `plot()` function with an `lm` object, are:

- A plot of residuals versus fitted values, as in Figure 5.1A. This allows a visual check for any pattern in the residuals that might suggest a curve rather than a line.
- A normal probability plot of residuals, as in Figure 5.1B. If residuals are from a normal distribution points should lie, to within statistical error, close to a line.

Code for these plots is:[3]

```
## A: Plot residuals vs fitted values; B: normal probability plot
plot(roller.lm, which = 1:2)
```

Note that if the argument `which` is left at its default, i.e., type `plot(roller.lm)`, there are two further plots. These further default plots will be demonstrated later.

In Figure 5.1A, there is a suggestion of clustering in the residuals, but no clear indication that there should be a curve rather than a line.

For interpreting the normal probability plot in Figure 5.1B, the eye needs a reference standard. It is useful to compare a plot such as Figure 5.1B against a number of independent plots from computer-generated normal data with the same number of observations, as in Figure 5.2, thus in effect calibrating the eye. The function `qreference()` (*DAAG*) may be used to generate suitable plots:[4]

```
## Normal probability plot, plus 7 reference plots
qreference(residuals(roller.lm), nrep=8, nrows=2)
```

[3] For side-by-side plots, precede with `par(mfrow=c(1,2))`.

[4]
```
## Alternatively, use the following code:
test <- residuals(roller.lm); n <- length(test)
av <- mean(test); sdev <- sd(test)
y <- c(test, rnorm(7*n, av, sdev))
fac <- c(rep("residuals(roller.lm)",n), paste("reference", rep(1:7, rep(n,7))))
fac <- factor(fac, levels=unique(fac))
library(lattice)
qqmath(~ y|fac, aspect=1, layout=c(4,2))
```

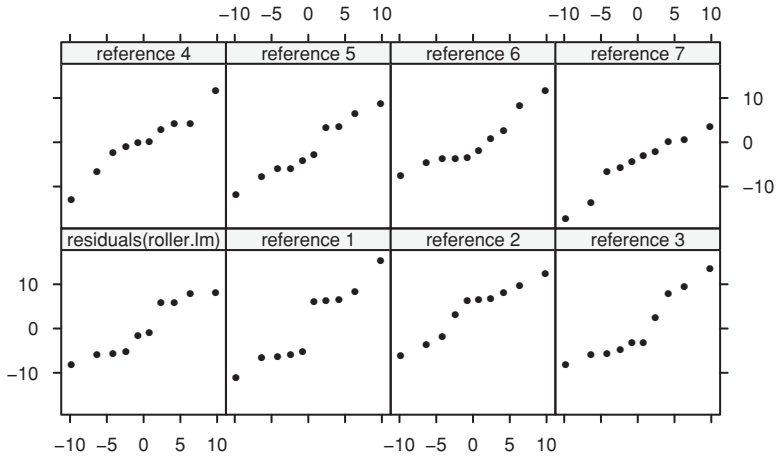


Figure 5.2 The normal probability plot for the regression of Figure 3.2A is shown in the lower left panel. Other panels show normal probability plots for computer-generated normal data.

5.1.3 Iron slag example: is there a pattern in the residuals?

Now consider an example where there is an evident pattern in residuals from a straight line regression. The data compare two methods for measuring the iron content in slag – a magnetic method and a chemical method (data are from Roberts, 1974, p. 126). The chemical method requires greater effort and is presumably expensive, while the magnetic method is quicker and easier.

Figure 5.3A suggests that the straight line model is wrong.[5] The smooth curve (shown with a dashed line) gives a better indication of the pattern in the data. Panel B shows the residuals from the straight line fit.[6] The non-linearity is now more evident. Panel C plots observed values against predicted values.[7] Panel D allows a visual check on whether the error variance is constant.[8] There are theoretical reasons for plotting the square root of absolute values of the residuals on the vertical axis in panel D.

Taken at face value, Figure 5.3D might seem to indicate that the variance decreases with increasing value of `magnetic`. Note, however, that the residuals are from the straight line model, which Figures 5.3A and B suggested was inappropriate. Thus the plot in panel D

[5]
```
## Panel A: chemical vs magnetic (Data frame ironslag from DAAG)
plot(chemical ~ magnetic, data=ironslag)
ironslag.lm <- lm(chemical ~ magnetic, data=ironslag)
abline(ironslag.lm)
with(ironslag, lines(lowess(chemical ~ magnetic, f=.9), lty=2))
```
[6]
```
## Panel B: Residuals from straight line fit, vs magnetic
res <- residuals(ironslag.lm)
plot(res ~ magnetic, xlab="Residual", data=ironslag)
with(ironslag, lines(lowess(res ~ magnetic, f=.9), lty=2))
```
[7]
```
## Panel C: Observed vs predicted
yhat <- fitted(ironslag.lm)
plot(chemical ~ yhat, data=ironslag, xlab="Predicted chemical", ylab="Chemical")
with(ironslag, lines(lowess(chemical ~ yhat, f=.9), lty=2))
```
[8]
```
## Panel D: Check whether error variance seems constant
sqrtabs <- sqrt(abs(res))
plot(sqrtabs ~ yhat, data=ironslag, xlab = "Predicted chemical",
     ylab =  expression(sqrt(abs(residual))), type = "n")
panel.smooth(yhat, sqrtabs, span = 0.95)
```

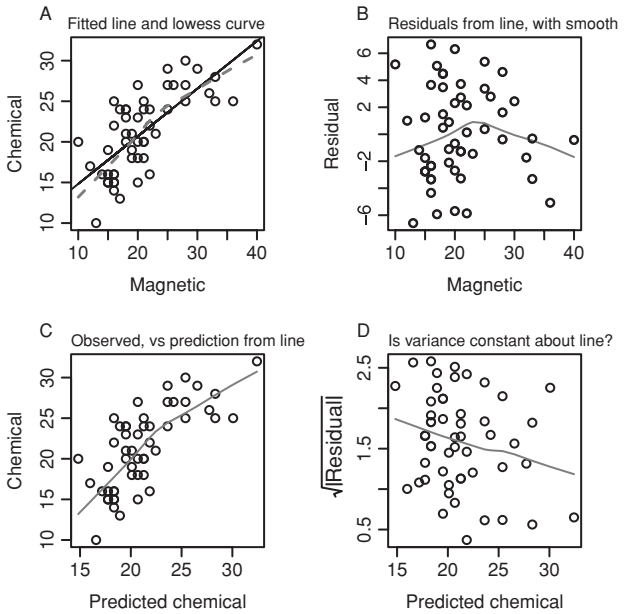


Figure 5.3 Chemical test of iron content in slag versus magnetic test. The fitted curves used the lowess smooth. In panel D, the downward slope suggests lower variance for larger fitted values. See however Figure 5.4.

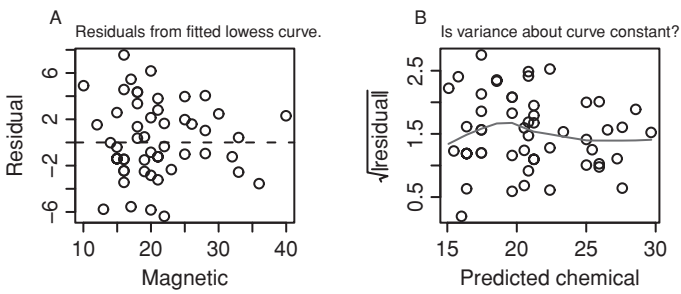


Figure 5.4 Residuals (panel A), and square root of absolute values of residuals (panel B), for the lowess smooth for the data of Figure 5.3. (Exercise 8 at the end of the chapter has the R code.)

may be misleading. For an accurate assessment, it is necessary to examine the equivalent plot for residuals from the smooth curve.

Figure 5.4 shows the plot of residuals from the smooth curve versus `magnetic`, together with the plot of square root of absolute values of residuals against predicted chemical test result. Any suggestion of heterogeneity is now of small consequence.

Where there is genuine heterogeneity of variance, and an accurate estimate of the variance at each data point is available, data points should be weighted proportionately to the reciprocal of the variance. Getting an estimate to within some constant of proportionality is enough. It may be possible to guess at a suitable functional form for the change in variance with x or (equivalently, since $y = a + bx$) with y. For example, the variance may be proportional to y.

5.1.4 The analysis of variance table

The analysis of variance table breaks the sum of squares about the mean, for the y-variable, into two parts: a part that is accounted for by the deterministic component of the model, i.e., by a linear function of `weight`, and a part attributed to the noise component or residual. For the lawn roller example, the analysis of variance table is:

```
> anova(roller.lm)
Analysis of Variance Table

Response: depression
          Df Sum Sq Mean Sq F value   Pr(>F)
weight     1 657.97  657.97  14.503 0.005175
Residuals  8 362.93   45.37
```

The total sum of squares (about the mean) for the 10 observations is 1020.9 (= 658.0 + 362.9; we round to one decimal place). Including weight reduced this by 658.0, giving a residual sum of squares equal to 362.9. For comparing the reduction with the residual, it is best to look at the column headed `Mean Sq`, i.e., *mean square*. The mean square for `weight` was 658.0; this compares with a mean square of 45.4 for the residual.

The degrees of freedom can be understood thus: Two points determine a line. With just two observations, both residuals would be zero, yielding no information about the noise. Every additional observation beyond two yields one additional degree of freedom for estimating the noise variance. Thus with 10 points, $10 - 2$ (= 8) degrees of freedom are available (in the residuals) for estimating the noise variance. (Where a line is constrained to pass through the origin, one point is enough to determine the line, and with 10 points the variance would be estimated with 9 degrees of freedom.)

This table has the information needed for calculating R^2 (also known as the "coefficient of determination") and adjusted R^2. The R^2 statistic is the square of the correlation coefficient, and is the sum of squares due to weight divided by the total sum of squares:

$$R^2 = \frac{658.0}{1020.9} = 0.64,$$

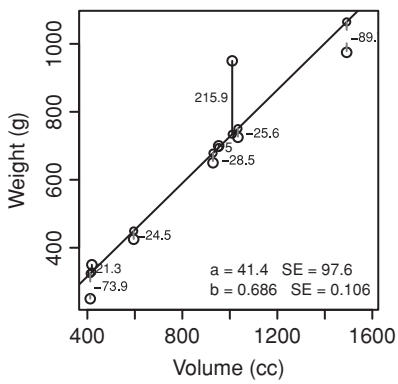
while

$$\text{adjusted } R^2 = 1 - \frac{362.9/8}{1020.9/9} = 0.60.$$

Adjusted R^2 takes into account the number of degrees of freedom, and is in general preferable to R^2. Neither statistic gives any direct indication of how well the regression equation will predict when applied to a new data set. Subsection 6.3.2 will argue against use of these statistics, and suggest alternatives. See also Section 5.4.

5.2 Outliers, influence, and robust regression

The data displayed in Figure 5.5, with data shown on the right, are for a collection of eight softback books. Additionally, the figure shows the fitted regression line, with information on the residuals from the line.



volume	weight
412	250
953	700
929	650
1492	975
419	350
1010	950
595	425
1034	725

Figure 5.5 Volumes ($\leq$m^3) and weights (g), for eight softback books. The figure also shows the fitted regression line, with information on the residuals from the line.

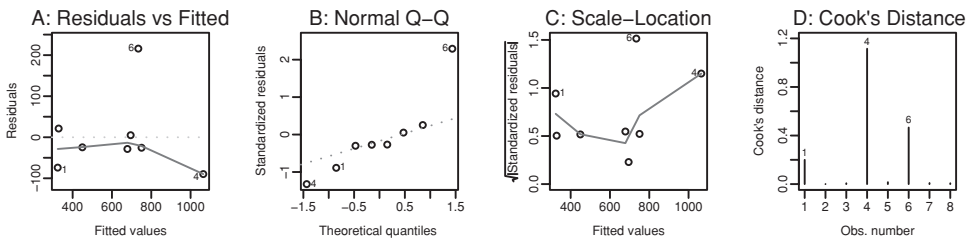


Figure 5.6 Diagnostic plots for Figure 5.5.

Here is the output from the regression calculations:

```
## Fit lm model: data frame softbacks (DAAG)
> softbacks.lm <- lm(weight ~ volume, data=softbacks)
> summary(softbacks.lm)
. . . .
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   41.372     97.559    0.42  0.68629
volume         0.686      0.106    6.47  0.00064
. . . .
```

Figure 5.6 shows regression diagnostics. Suitable code is:

```
par(mfrow=c(2,2))    # Use par(mfrow=c(1,4)) for layout in figure
plot(softbacks.lm, which=1:4)
 # By default, plots 1:3 and 5 [which=c(1:3,5)] are given
par(mfrow=c(1,1))
```

We should already be familiar with plots A and B from Figure 5.6. For regression with one explanatory variable, plot A is equivalent to a plot of residuals against the explanatory variable. Plot C, which is not of much interest for the present data, is designed for examining the constancy of the variance. Plot D identifies residuals that are *influential* in determining

the form of the regression line. Points with Cook's distances that are greater than one, or whose Cook's distances are substantially larger than for other points, may warrant investigation. The largest two residuals in Figure 5.6D, for observations 4 and 6, are identified with their row labels. For the `softbacks` data, the row labels are the observation numbers.

Cook's distance is a measure of *influence*; it measures the extent to which the line would change if the point were omitted. Although observation 6 has the largest residual, its Cook's distance is relatively small. Observation 4 has the largest Cook's distance. In part, the value is large because this point is at the extreme end of the range of x-values. It is a high *leverage* point that exerts a greater pull on the regression line than observations closer to the center of the range. Since its y-value is lower than would be predicted by the line, it pulls the line downward.

Diagnostic plots, such as Figure 5.6, are not definitive. Rather, they draw attention to points that require further investigation. Here, with only eight points, it would not make sense to omit any of them, especially as points 4 and 6 are both, for different reasons, candidates for omission.

It may, on checking, turn out that an outlier has arisen from a recording or similar error. Where an outlier seems a genuine data value, it is good practice to do the analysis both with and without the outlier. If retention of an apparent outlier makes little difference to the practical use and interpretation of the results, it is usually best to retain it in the main analysis. If an outlier that seems a genuine data value is omitted from the main analysis, it should be reported along with the main analysis, and included in graphs.

Robust regression

Robust regression offers a half-way house between including outliers and omitting them entirely. Rather than omitting outliers, it downweights them, reducing their influence on the fitted regression line. This has the additional advantage of making outliers stand out more strongly against the line. The *MASS* package has the robust regression function `rlm()`. Note also the *resistant* regression function `lqs()`, also in the *MASS* package. Resistant regression methods aim to ensure that outliers do not contribute to the regression fit. For both robust and resistant methods, it is important that residuals have an approximately symmetric distribution. See further, Section 6.3 and Exercise 14 at the end of Chapter 6.

5.3 Standard errors and confidence intervals

Recall that since two parameters (the slope and intercept) have been estimated, the error mean square is calculated with $n - 2$ degrees of freedom. As a consequence, the standard errors for the slope and for predicted values are calculated with $n - 2$ degrees of freedom. Both involve the use of the square root of the error mean square.

Additionally, as will be demonstrated in Subsection 5.3.2, the `predict()` function can be used to obtain standard errors and/or confidence intervals for predicted values. A wide confidence interval for the regression slope implies that intervals for predicted values will likewise be wide.

5.3.1 Confidence intervals and tests for the slope

A 95% confidence interval for the regression slope is

$$b \pm t_{.975}\text{SE}_b$$

where $t_{.975}$ is the 97.5% point of the t-distribution with $n-2$ degrees of freedom, and SE_b is the standard error of b.

We demonstrate the calculation for the `roller` data. From the second row of the `Coefficients` table in the summary output of Subsection 5.1.1, the slope estimate is 2.67, with SE = 0.70. The t-critical value for a 95% confidence interval on $10-2=8$ degrees of freedom is $t_{.975}=2.30$. Therefore, the 95% confidence interval is:[9]

$$2.67 \pm 2.3 \times 0.7 = (1.1, 4.3).$$

If the 95% confidence interval for b contains 0, the null hypothesis that the slope is zero will be rejected at the 95% significance level. (For the testing of such hypotheses, refer back to Subsection 5.1.1.)

5.3.2 SEs and confidence intervals for predicted values

There are two types of predictions: prediction of points on the line, and prediction of a new data value. The SE estimates of predictions for new data values take account both of uncertainty in the line and of the variation of individual points about the line. Thus the SE for prediction of a new data value is larger than that for prediction of points on the line.

Table 5.1 shows expected values of the depression, with SEs, for various roller weights.[10] The column headed SE indicates the precision of points on the line. The column headed SE.OBS indicates the precision with which new observations can be predicted. For determining SE.OBS, there are two sources of uncertainty: the standard error for the fitted value (estimated at 3.6 in row 1) and the noise standard error (estimated at 6.74) associated with a new observation.

Figure 5.7 shows 95% pointwise confidence bounds for the fitted line.[11] It bears emphasizing that the validity of these calculations depends crucially on the appropriateness of the fitted model for the given data.

[9]
```
## Code for confidence interval calculations
SEb <- summary(roller.lm)$coefficients[2, 2]
coef(roller.lm)[2] + qt(c(0.025,.975), 8)*SEb
```

[10]
```
## Code to obtain fitted values and standard errors (SE, then SE.OBS)
fit.with.se <- predict(roller.lm, se.fit=TRUE)
fit.with.se$se.fit                                          # SE
sqrt(fit.with.se$se.fit^2+fit.with.se$residual.scale^2)  # SE.OBS
```

[11]
```
## Plot depression vs weight, with 95\% pointwise bounds for the fitted line
plot(depression ~ weight, data=roller, xlab = "Weight of Roller (tonnes)",
     ylab = "Depression in Lawn (mm)", pch = 16)
roller.lm <- lm(depression ~ weight, data = roller)
abline(roller.lm$coef, lty = 1)
xy <- data.frame(weight = pretty(roller$weight, 20))
yhat <- predict(roller.lm, newdata = xy, interval="confidence")
ci <- data.frame(lower=yhat[, "lwr"], upper=yhat[, "upr"])
lines(xy$weight, ci$lower, lty = 2, lwd=2, col="grey")
lines(xy$weight, ci$upper, lty = 2, lwd=2, col="grey")
```

Table 5.1 *Observed and fitted values of* `depression` *at the given* `weight` *values, together with two different types of SE. The column headed SE gives the precision of the predicted value. The column headed SE.OBS gives the precision of a new observation.*

	Predictor `weight`	Observed `depression`	Fitted	SE	SE.OBS	
1	1.9	2	3.0	3.6	7.6	$\sqrt{3.6^2 + 6.74^2}$
2	3.1	1	6.2	3.0	7.4	$\sqrt{3.0^2 + 6.74^2}$
3	3.3	5	6.7	2.9	7.3	
... 10	12.4	25	31.0	4.9	8.3	

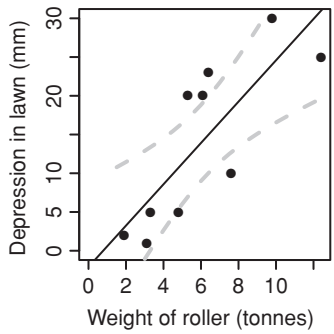


Figure 5.7 Lawn depression, for various weights of roller, with fitted line and showing 95% pointwise confidence bounds for points on the fitted line.

5.3.3* *Implications for design*

An emphasis of this subsection is that the choice of location of the x-values, which is a design issue, is closely connected with sample size considerations. Increasing the sample size is not the only, or necessarily the best, way to improve precision.

The estimated variance of the slope estimate is

$$\mathrm{SE}_b^2 = \frac{s^2}{n s_x^2},$$

where we define

$$s_x^2 = \frac{\sum_i (x_i - \bar{x})^2}{n}.$$

Here s^2 is the error mean square, i.e., s is the estimated SD for the population from which the residuals are taken. The expected value of SE_b^2 is

$$\mathrm{E}[\mathrm{SE}_b^2] = \frac{\sigma^2}{n s_x^2}.$$

Now consider two alternative ways to reduce SE_b by a factor of 2:

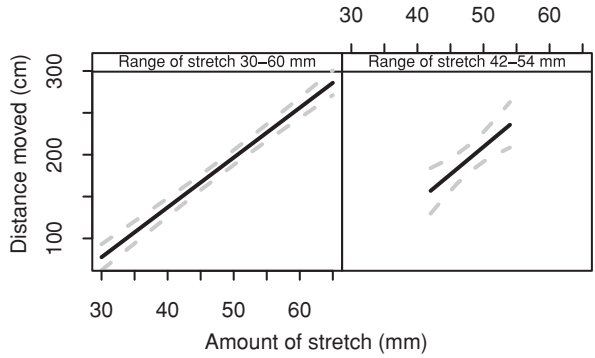


Figure 5.8 Two rubber band experiments, with different ranges of x-values. The dashed curves are pointwise 95% confidence bounds for points on the fitted line. Note that, for the panel on the right, the axis labels appear above the panel, as is done for *lattice* plots. Data are from `elastic1` (left panel, 7 points) and `elastic2` (right panel, 9 points), both from *DAAG*. Even with 9 points as against 7, the right panel has much wider pointwise bounds.

- By fixing the configuration of x-values, but multiplying by 4 the number of values at each discrete x-value, s_x is unchanged. As n increases by a factor of 4, the expected value of SE_b^2 reduces by a factor of 4, and SE_b by a factor of 2.
- Alternatively, increasing the average separation between x-values by a factor of 2 will reduce SE_b by a factor of 2.

Spreading out the x-values achieves the same reduction in SE_b, as increasing the number of points. Checking for linearity over the extended range of x-values is, however, important.

Reducing SE_b reduces the standard error of the fitted values as well. Figure 5.8 shows the effect of increasing the range of x-values (the code for both panels is a ready adaptation of the code for Figure 5.7). Both experiments used the same rubber band. The first experiment used a much wider range of values of x (= amount by which the rubber band was stretched). For the left panel of Figure 5.8, $s_x = 10.8$, while for the right panel, $s_x = 4.3$.

5.4 Assessing predictive accuracy

The *training data* estimate of predictive accuracy, derived by direct application of model predictions to the data from which the regression relationship was derived, gives in general an optimistic assessment. There is a mutual dependence between the model prediction and the data used to derive that prediction. It is because of this dependence that degrees of freedom for the variance are adjusted to take account of the number of parameters estimated.

The issue becomes more important in contexts, such as the classification models that will be discussed in Chapter 11 and Section 12.2, where no satisfactory theoretical adjustment for the dependence is available. The simple models discussed in the present chapter are a good context in which to demonstrate general approaches that address this issue.

Table 5.2 *Floor area and sale price, for 15 houses in Aranda, a suburb of Canberra, Australia.*

Row number	area	bedrooms	sale.price
1	694	4	192.0
2	905	4	215.0
	...	...	...
15	1191	6	375.0

5.4.1 Training/test sets and cross-validation

An ideal is to assess the performance of the model on a new data set. It is good practice to split the data into two sets: the *training* set is for developing the model, and the *test* set is for testing predictions. This is a valid procedure, if the test set can be regarded as a random sample of the population to which predictions will be applied. If there are too few data to make it reasonable to divide data into training and test sets, then the method of cross-validation can be used.

Cross-validation extends the training/test set approach. As with that approach, it estimates predictive accuracy for data that are sampled from the population in the same way as the existing data. The data are divided into k sets (or *folds*), where k is typically in the range 3 to 10. Each of the k sets becomes in turn the test set, with the remaining data forming the training set. The predictive accuracy assessments from the k folds are combined to give a measure of the predictive performance of the model. This may be done for several different measures of predictive performance.

5.4.2 Cross-validation – an example

We present an example of the use of cross-validation with a small data set. In order to simplify the discussion, we will use threefold validation only.

Table 5.2 shows data on floor area and sale price for 15 houses in a suburb of Canberra, in 1999. Rows of data have been numbered from 1 to 15. For demonstrating cross-validation, we use a random number sampling system to divide the data up into three equal groups.[12]

The observation numbers for the three groups we obtain are:

```
2 3 12 13 15
1 5 7 8 14
4 6 9 10 11
```

Rerunning the calculations will of course lead to a different division into three groups.

At the first pass (fold 1) the first set of rows will be set aside as the test data, with remaining rows making up the training data. Each such division between training data and

[12]
```
## Split row numbers randomly into 3 groups
rand <- sample(1:15)%%3 + 1
  # a%%3 is the remainder of a, modulo 3
  # Subtract from a the largest multiple of 3 that is <= a; take remainder
(1:15)[rand == 1]    # Observation numbers for the first group
(1:15)[rand == 2]    # Observation numbers for the second group
(1:15)[rand == 3]    # Observation numbers for the third group.
```

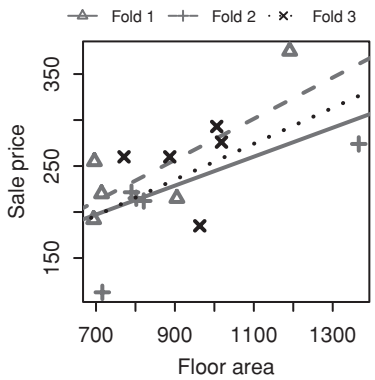


Figure 5.9 Graphical summary of threefold cross-validation for the house sale data (Table 5.2). The line is fitted leaving out the corresponding "test" set of points. Predictions for these omitted points are used to assess predictive accuracy.

test data is known as a *fold*. At the second pass (fold 2) the second set of rows will be set aside as the test data, while at the third pass (fold 3) the third set of rows will be set aside as the test data. A crucial point is that at each pass the data that are used for testing are separate from the data used for prediction. Figure 5.9 is a visual summary, obtained by using the function `CVlm()` (*DAAG*) with the default setting `plotit=TRUE`.

The following summary of the cross-validation results includes, for each fold, estimates of the mean square error.

```
> ## Cross-validate lm calculations: data frame houseprices (DAAG)
> houseprices.lm <- lm(sale.price ~ area, data=houseprices)
> CVlm(houseprices, houseprices.lm, plotit=TRUE)

fold 1
Observations in test set: 2 3 12 13 15
Floor area      905.0 802.00 696.0 771.0 1191
Predicted price 225.9 208.63 190.9 203.4  274
Observed price  215.0 215.00 255.0 260.0  375
Residual        -10.9   6.37  64.1  56.6  101

Sum of squares = 17719    Mean square = 3544    n = 5

fold 2
Observations in test set: 1 5 7 8 14

Floor area      694.0  716 821.0 714.00 1006.0
Predicted price 222.4  225 238.6 224.97  262.2
Observed price  192.0  113 212.0 220.00  293.0
Residual        -30.4 -113 -26.6  -4.97   30.8

Sum of squares = 15269    Mean square = 3054    n = 5

fold 3
```

```
Observations in test set: 4 6 9 10 11

Floor area      1366 963.0 1018.0 887.00 790.00
Predicted price  412 278.4  296.6 253.28 221.22
Observed price   274 185.0  276.0 260.00 221.50
Residual        -138 -93.4  -20.6   6.72   0.28

Sum of squares = 28127    Mean square = 5625    n = 5
Overall ms
      4074
```

At each fold, the training set consists of the remaining rows of data.

To obtain the estimate of the error mean square, take the total of the sums of squares and divide by 15. This gives

$$s^2 = (17\,719 + 15\,269 + 28\,127)/15 = 4074.$$

Actually, what we have is an estimate of the error mean square when we use only two-thirds of the data. Thus we expect the cross-validated error to be larger than the error if all the data could be used. We can reduce the error by doing 10-fold rather than threefold cross-validation. Or we can do leave-one-out cross-validation, which for these data is 15-fold cross-validation.

Contrast $s^2 = 4074$ with the estimate $s^2 = 2323$ that we obtained from the model-based estimate in the regression output for the total data.[13]

5.4.3 Bootstrapping*

We first indicate how resampling methods can be used to estimate the standard error of slope of a regression line. Recalling that the standard error of the slope is the standard deviation of the sampling distribution of the slope, we need a way of approximating this sampling distribution. One way of obtaining such an approximation is to resample the observations or cases directly. For example, suppose five observations have been taken on a predictor x and response y:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5).$$

Generate five random numbers with replacement from the set $\{1, 2, 3, 4, 5\}$: 3, 5, 5, 1, 2, say. The corresponding resample is then

$$(x_3, y_3), (x_5, y_5), (x_5, y_5), (x_1, y_1), (x_2, y_2).$$

Note we are only demonstrating the so-called case-resampling approach. Another approach involves fitting a model and resampling the residuals. Details for both methods are in Davison and Hinkley (1997, Chapter 6). A regression line can be fit to the resampled observations, yielding a slope estimate. Repeatedly taking such resamples, we obtain a distribution of slope estimates, the bootstrap distribution.

[13] `## Estimate of sigma^2 from regression output`
`summary(houseprices.lm)$sigma^2`

As an example, consider the regression relating `sale.price` to `area` in the `houseprices` data. We will compute a bootstrap estimate of the standard error of the slope. For comparison purposes, note first the estimate given by `lm()`: 0.0664.

```
> houseprices.lm <- lm(sale.price ~ area, data=houseprices)
> summary(houseprices.lm)$coef

             Estimate Std. Error t value Pr(>|t|)
(Intercept)   70.750    60.3477    1.17   0.2621
area           0.188     0.0664    2.83   0.0142
```

In order to use the `boot()` function, we need a function that will evaluate the slope for each of the bootstrap resamples:

```
houseprices.fn <- function (houseprices, index){
    house.resample <- houseprices[index, ]
    house.lm <- lm(sale.price ~ area, data=house.resample)
    coef(house.lm)[2]    # slope estimate for resampled data
    }
```

We then use the `boot()` function to make 999 calls to the `houseprices.fn()` function with different randomly generated resamples from the data frame `houseprices`.

```
> set.seed(1028)     # use to replicate the exact results below
> library(boot)      # ensure that the boot package is loaded
> ## requires the data frame houseprices (DAAG)
> (houseprices.boot <- boot(houseprices, R=999, statistic=houseprices.fn))
. . . .
Bootstrap Statistics :
    original  bias    std. error
t1*    0.188  0.0169      0.0916
```

The output shows us the original slope estimate, a bootstrap estimate of the bias of this estimate, and the standard error estimate: 0.0916. This standard error was computed from the standard deviation of the 999 resampled slope estimates.

By changing the statistic argument in the `boot()` function appropriately, we can compute standard errors and confidence intervals for fitted values. Here we use the `predict()` function to obtain predictions for the given area:

```
housepred.fn <- function(houseprices, index){
    house.resample <- houseprices[index, ]
    house.lm <- lm(sale.price ~ area, data=house.resample)
    predict(house.lm, newdata=data.frame(area=1200))
    }
```

For example, a 95% confidence interval for the expected sale price of a house (in Aranda) having an area of 1200 square feet is (249 000, 363 000).[14]

[14] `## 95% CI for predicted price of 1200 square foot house`
`housepred.boot <- boot(houseprices, R=999, statistic=housepred.fn)`
`boot.ci(housepred.boot, type="perc") # "basic" is an alternative to "perc"`

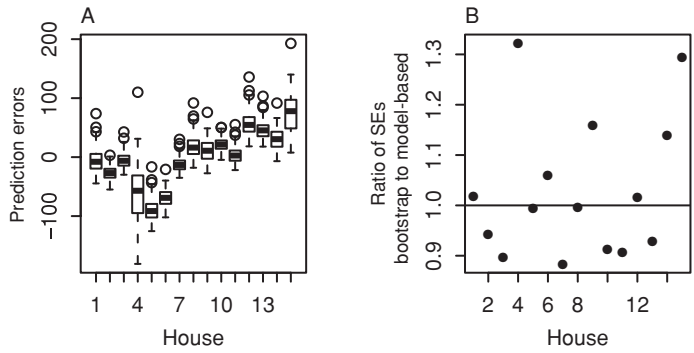


Figure 5.10 (A) Plot of bootstrap distributions of prediction errors for regression relating `sale.price` to `area`, each based on 200 bootstrap estimates of the prediction error. (B) Ratios of bootstrap prediction standard errors to model-based prediction standard errors.

The bootstrap procedure can be used to gain additional insight into how well a regression model is making predictions. Regression estimates for each resample are used to compute predicted values at all of the original values of the predictor. The differences (i.e., the prediction errors) between the observed responses and these resampled predictions can be plotted against observation number. Repeating this procedure a number of times gives a distribution of the prediction errors at each observation. Figure 5.10 A displays a prediction error plot for the `houseprices` data.[15] Note the large variability in the prediction error associated with observation 4. We can use the same bootstrap output to estimate the standard errors. These can be compared with the usual estimates obtained by `lm`. Figure 5.10B displays ratios of the bootstrap standard errors to the model-based standard errors.[16] In this case, the model-based standard errors are generally smaller than the bootstrap standard errors. A cautious data analyst might prefer the bootstrap standard errors.

We can also compute an estimate of the aggregate prediction error, as an alternative to the cross-validation estimate obtained in the previous subsection. There are a number of ways to do this, and some care should be taken. We refer the interested reader to Davison and Hinkley (1997, Section 6.4).

[15]
```
## Bootstrap estimates of prediction errors of house prices
houseprices2.fn <- function (houseprices, index)
{
  house.resample <- houseprices[index, ]
  house.lm <- lm(sale.price ~ area, data=house.resample)
  houseprices$sale.price - predict(house.lm, houseprices)  # resampled prediction
                                                           # errors
}
n <- length(houseprices$area);  R <- 200
houseprices2.boot <- boot(houseprices, R=R, statistic=houseprices2.fn)
house.fac <- factor(rep(1:n, rep(R, n)))
plot(house.fac, as.vector(houseprices2.boot$t), ylab="Prediction Errors",
     xlab="House")
```

[16]
```
## Ratios of bootstrap to model-based standard errors
bootse <- apply(houseprices2.boot$t, 2, sd)
usualse <- predict.lm(houseprices.lm, se.fit=TRUE)$se.fit
plot(bootse/usualse, ylab="Ratio of Bootstrap SE's to Model-Based SE's",
     xlab="House", pch=16)
abline(1, 0)
```

Commentary

The cross-validation and bootstrap estimates of mean square error are valid, provided we can assume a homogeneous variance. This is true even if data values are not independent. However, the estimate of predictive error applies only to data that have been sampled in the same way as the data that are used as the basis for the calculations. They assume that the `target` population will be highly comparable to the `source` population that generated the data. In the present instance, the estimate of predictive accuracy applies only to 1999 house prices in the same city suburb.

Such standard errors may have little relevance to the prediction of house prices in another suburb, even if thought to be comparable, or to prediction for more than a very short period of time into the future. This point has relevance to the use of regression methods in business "data mining" applications. A prediction that a change will make cost savings of $500 000 in the current year may have little relevance to subsequent years. The point has special force if changes will take years rather than months to implement.

A realistic, though still not very adequate, assessment of accuracy may be derived by testing a model that is based on data from previous years on a test set that is formed from the current year's data. Predictions based on the current year's data may, if other features of the business environment do not change, have a roughly comparable accuracy for prediction a year into the future. If the data series is long enough, we might, starting at a point part-way through the series, compare predictions one year into the future with data for that year. The estimated predictive accuracy would be the average accuracy for all such predictions. A more sophisticated approach might involve incorporation of temporal components into the model, i.e., use of a time series model. See Maindonald (2003) for more extended commentary on such issues.

5.5 Regression versus qualitative anova comparisons – issues of power

An analysis that fails to take advantage of structure in the data may fail to find what is there. Figure 5.11 shows six sets of data that have been simulated to follow a linear trend. Simulation of regression models was discussed in Subsection 3.3.2.

The first p-value tests for linear trend, while the second p-value tests for qualitative differences between treatment effects, ignoring the fact that the levels are quantitative (note that the test for linear trend is equivalent to the test for a linear contrast from the `aov()` function that is available when the explanatory term is an ordered factor). A test for linear trend is more powerful than an analysis of variance that treats the five levels as qualitatively different levels. In repeated simulations of Figure 5.11, the p-values in the test for linear trend will on average be smaller than in the analysis of variance that makes qualitative comparisons between the five levels.

To get a clear indication of the effect, we need a more extensive simulation. Figure 5.12 plots results from 200 simulations. Both axes use a scale of $\log(p/(1-p))$. On the vertical axis are the p-values for a test for linear trend, while the horizontal axis plots p-values for an aov test for qualitative differences. The majority of points (for this simulation, 91%) lie below the line $y = x$.

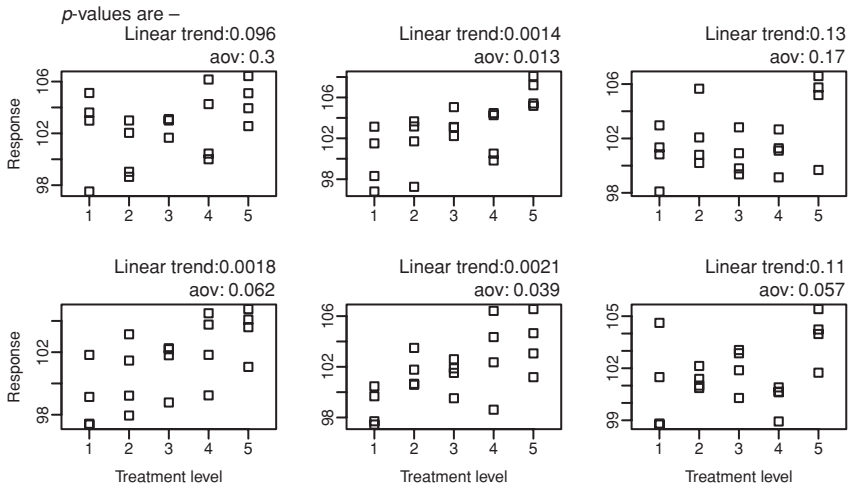


Figure 5.11 Test for linear trend, versus analysis of variance comparison that treats the levels as qualitatively different. The six panels are six different simulations from the straight line model with slope 0.8, SD = 2, and 4 replications per level.

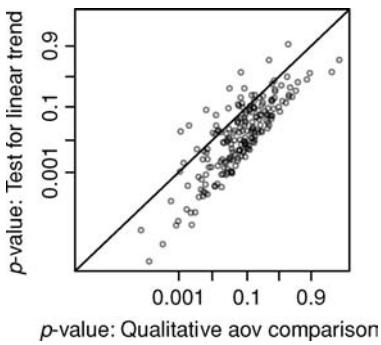


Figure 5.12 This plot compares p-values from a test for linear trend with p-values from an analysis of variance test for qualitative differences, in each of 200 sets of simulated results. The line $y = x$ is superimposed.

The function `simulateLinear()` in our *DAAG* package allows readers to experiment with such simulations. Write the p-values for a test for linear trend as p_l, and the p-values for the analysis of variance test for qualitative differences as p_a. Specifying `type="density"` gives overlaid plots of the densities for the two sets of p-values, both on a scale of $\log(p/(1-p))$, together with a plot of the density of $\log(p_l/(1-p_l)) - \log(p_a/(1-p_a))$. As the data are paired, this last plot is the preferred way to make the comparison.

The pattern of change

There are other reasons for fitting a line or curve, where this is possible, rather than fitting an analysis of variance model that has a separate parameter for each separate level of

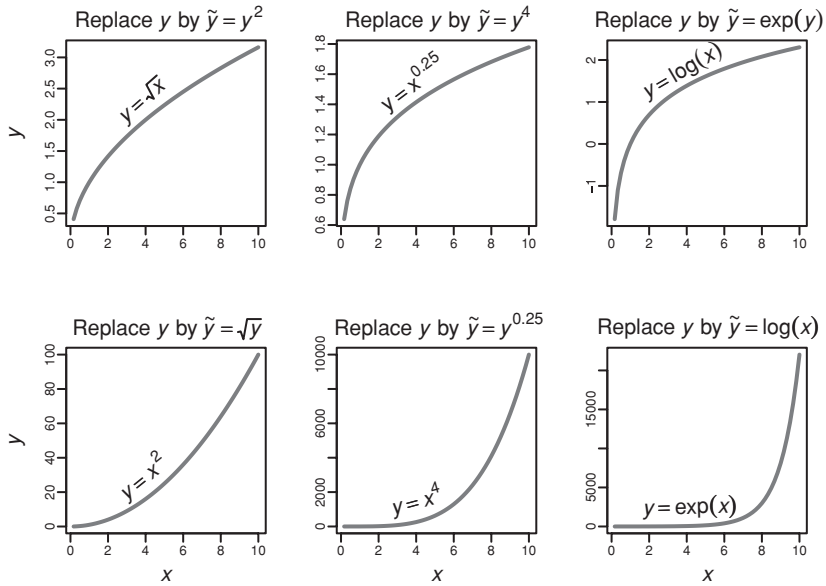


Figure 5.13 The above panels show some alternative response curves. The formula for $\tilde{y}$ gives the power family transformation of y that will make $\tilde{y}$ a linear function of x. Thus, if $y = \log(x)$, then the transformation $\tilde{y} = \exp(y)$ will make $\tilde{y}$ a linear function of x.

the explanatory variable. Fitting a line (or a curve) allows interpolation between successive levels of the explanatory variable. It may be reasonable to hazard prediction a small distance beyond the range of the data. The pattern of response may give scientific insight.

5.6 Logarithmic and other transformations

5.6.1 A note on power transformations*

Among the more common transformations for continuous data are:

- **Logarithmic.** This is often the right transformation for size measurements (linear, surface, volume or weight) of biological organisms. Some data may be too skewed even for a logarithmic transformation. For example, counts of insects on leaves may have this character.
- **Square root** or **cube root.** These are milder than the logarithmic transformation. If linear measurements on insects are normally distributed, then we might expect the cube root of weight to be approximately normally distributed. The square root is useful for data for counts of "rare events". The power transformation generalizes the transformations that we have just discussed. Examples of power transformations are y^2, $y^{0.5}$, y^3, etc. Figure 5.13 shows a number of response curves, and describes the particular power transformation that would make the relationship linear.

If the ratio of largest to smallest data value is greater than 10, and especially if it is more than 100, then the logarithmic transformation should be tried as a matter of course. Check this advice against the response curves shown in Figure 5.13.

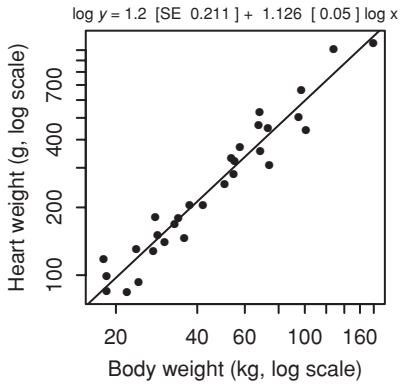


Figure 5.14 Heart weight versus body weight, for 30 Cape fur seals.

We have so far mentioned only transformation of y. We might alternatively transform x, or transform both x and y.

*General power transformations

For $\lambda \neq 0$, the power transformation replaces a value y by y^λ. The logarithmic transformation corresponds to $\lambda = 0$. In order to make this connection, a location and scale correction is needed. The transformation is then

$$y(\lambda) = \frac{y^\lambda - 1}{\lambda}, \quad \text{if } \lambda \neq 0,$$
$$y(\lambda) = \log(y), \quad \text{if } \lambda = 0.$$

- If the small values of a variable need to be spread, make λ smaller.
- If the large values of a variable need to be spread, make λ larger.

This is called the Box–Cox transformation, as proposed in Box and Cox (1964).

The function `boxcox()` (*MASS*), whose syntax is similar to that of `lm()`, can be used to obtain data-driven estimates of λ. An exercise at the end of the chapter pursues investigation of `boxcox()`.

5.6.2 Size and shape data – allometric growth

The logarithmic transformation is commonly important for morphometric data, i.e., for data on the size and shape of organisms. Figure 5.14 uses logarithmic scales to plot heart weight against body weight, for 30 seals that had been snared in trawl nets as an unintended consequence of commercial fishing (Stewardson *et al.*, 1999).

For each animal, the data provide information at just one point in time, when they died. The data thus have limited usefulness for the study of growth profiles through time. At best, if conditions have not changed too much over the lifetimes of the animals in the sample, the data may provide an indication of the average of the population growth profiles. If, e.g., sample ages range from 1 to 10 years, it is pertinent to ask how food availability may have

changed over the past 10 years, and whether this may have had differential effects on the different ages of animal in the sample.

The allometric growth equation

The allometric growth equation is

$$y = ax^b$$

where x may, e.g., be body weight and y heart weight. It may alternatively be written

$$\log y = \log a + b \log x,$$

i.e.,

$$Y = A + bX,$$

where

$$Y = \log y, \quad A = \log a, \quad \text{and } X = \log x.$$

Thus, we have an equation that can be fitted by linear regression methods, allowing prediction of values of Y given a value for X. If $b = 1$, then the two organs (e.g., heart and body weight) grow at the same rate.

Here is the R output for the calculations that fit the regression line in Figure 5.14:

```
> summary(cfseal.lm <- lm(log(heart) ~ log(weight), data=cfseal))
. . . .
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.2043     0.2113     5.7  4.1e-06
log(weight)   1.1261     0.0547    20.6  < 2e-16

Residual standard error: 0.18 on 28 degrees of freedom
Multiple R-Squared: 0.938,      Adjusted R-squared: 0.936
F-statistic:  424 on 1 and 28 DF,  p-value: <2e-16
```

Note that the estimate of the exponent b ($= 1.126$) differs from 1.0 by 2.3 ($= 0.126/0.0547$) times its standard error. Thus for these data, the relative rate of increase seems slightly greater for heart weight than for body weight. We have no interest in the comparison between b and zero, for which the t-statistic and p-value in the regression output are appropriate (authors sometimes present p-values that focus on the comparison with zero, even though their interest is in the comparison with 1.0. See Table 10 and other similar tables in Gihr and Pilleri (1969, p. 43)). For an elementary discussion of allometric growth, see Schmidt-Nielsen (1984).

5.7 There are two regression lines!

At this point, we note that there are two regression lines – a regression line for y on x, and a regression line for x on y. It makes a difference which is the explanatory variable, and

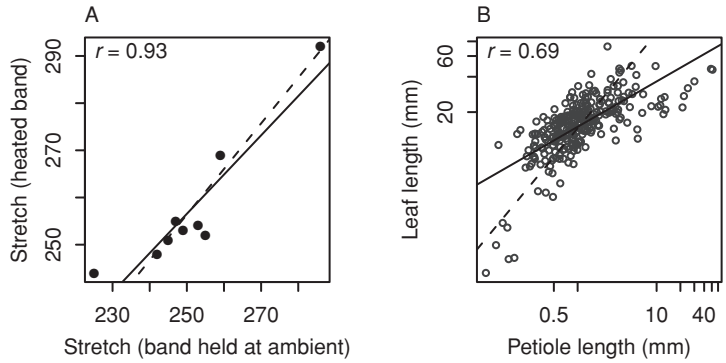


Figure 5.15 Each plot shows both the regression line for y on x (solid line), and the regression line for x on y (dotted line). In panel A the lines are quite similar, while in panel B where the correlation is smaller, the lines are quite different. Plot A is for the data of Table 3.1, while B is for a leaf data set.

which the dependent variable. The two lines are quite different if the correlation is small. Figure 5.15 illustrates the point for two other data sets.

An alternative to a regression line

There are yet other possibilities. A perspective that makes good sense for the seal organ growth data is that there is an underlying linear functional relationship. The analysis assumes that observed values of log(organ weight) and log(body weight) differ from the values for this underlying functional relationship by independent random amounts. The line that is obtained will lie between the regression line for y on x and the line for x on y. See Sprent (1966). Exercise 12 demonstrates a method for finding such a line.

5.8 The model matrix in regression

For many of the uses of the `lm()` function in later chapters, it will be important to understand the use of the model matrix to structure calculations for practical computation. This is especially true for Chapter 7. Straight line regression is a simple context in which to introduce these ideas.

In straight line regression, the model or X matrix has two columns – a column of 1s and a column that holds values of the explanatory variable x. As fitted, the straight line model is

$$\widehat{y} = a + bx$$

which we can write as

$$\widehat{y} = 1 \times a + x \times b.$$

For an example, we return to the lawn roller data. The model matrix, with the y-vector alongside, is given in Table 5.3. To obtain the model matrix, specify:

```
model.matrix(roller.lm)
```

Table 5.3 *The model matrix, for the lawn roller data, with the vector of observed values in the column to the right.*

X		y
`weight` (t)		`depression` (mm)
1	1.9	2
1	3.1	1
1	3.3	5
1	4.8	5
1	5.3	20
1	6.1	20
1	6.4	23
1	7.6	10
1	9.8	30
1	12.4	25

Table 5.4 *The use of the model matrix for calculation of fitted values and residuals, in fitting a straight line to the lawn roller data.*

Model matrix				
$\times -2.09$	$\times$ 2.67 weight	Multiply and add to yield fitted value $\widehat{y}$	Compare with observed y	Residual = $y - \widehat{y}$
1	1.9	$-2.1 + 2.67 \times 1.9 = 2.98$	2	$2 - 2.98$
1	3.1	$-2.1 + 2.67 \times 3.1 = 6.18$	1	$1 - 6.18$
1	3.3	$-2.1 + 2.67 \times 3.3 = 6.71$	5	$5 - 6.71$
1	4.8	$-2.1 + 2.67 \times 4.8 = 10.71$	5	$5 - 10.71$
1	5.3	$-2.1 + 2.67 \times 5.3 = 12.05$	20	$20 - 12.05$
1	6.1	$-2.1 + 2.67 \times 6.1 = 14.18$	20	$20 - 14.18$
1	6.4	$-2.1 + 2.67 \times 6.4 = 14.98$	23	$23 - 14.98$
1	7.6	$-2.1 + 2.67 \times 7.6 = 18.18$	10	$10 - 18.18$
1	9.8	$-2.1 + 2.67 \times 9.8 = 24.05$	30	$30 - 24.05$
1	12.4	$-2.1 + 2.67 \times 12.4 = 30.98$	25	$25 - 30.98$

For each row, we take some multiple of the value in the first column, another multiple of the value in the second column, and add them. Table 5.4 shows how calculations proceed given the estimates of a and b obtained earlier.

Note also the simpler (no intercept) model. For this:

$$\widehat{y} = bx.$$

In this case the model matrix has only a single column, containing the values of x.

5.9* Bayesian regression estimation using the MCMCpack package

Subsection 4.8.2 discussed ideas of Bayesian estimation, drawing attention to the use of the Markov Chain Monte Carlo (MCMC) simulation technique to generate successive parameter estimates. The simulation process must be allowed to *burn in*, i.e., run for long enough that the posterior distribution reaches a steady state that is independent of the starting values of parameters.

The *MCMCpack* package has the function `MCMCregress()`, with a similar syntax to `lm()`, that can be used for regression calculations. The following is intended as a straightforward demonstration of the methodology, albeit for an example where use of the function `lm()` might in practice be preferable.

The default is to assume independent uniform priors for the regression coefficients, to allow the simulation to run for 10 000 iterations, and to take the first 1000 iterations as burn-in. Here is the code and accompanying output, for the `roller` data:

```
> library(MCMCpack)
> roller.mcmc <- MCMCregress(depression ~ weight, data=roller)
> summary(roller.mcmc)

Iterations = 1001:11000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
(Intercept) -2.00  5.486  0.05486        0.05422
weight       2.65  0.812  0.00812        0.00843
sigma2      60.47 40.610  0.40610        0.57218

2. Quantiles for each variable:

              2.5%   25%   50%   75%  97.5%
(Intercept) -12.80 -5.38 -1.99  1.29   9.25
weight        1.01  2.17  2.66  3.16   4.26
sigma2       21.01 35.40 49.39 71.42 166.23
```

Because estimates from the previous iteration are the starting values for the current iteration, the sequence of estimates is Markovian and there is a lag 1 partial autocorrelation. The time series SE in the final column is designed to adjust for this autocorrelation. (See Section 9.1 for the relevant time series concepts. Specifically, it is assumed that the sequence of estimates follows an autoregressive process of order 1.) The standard error is inflated to take account of the correlation between successive estimates. Notice that the coefficient estimates are very similar to those obtained in Subsection 5.1.1 using `lm()`, while the SEs (both sets) are slightly larger.

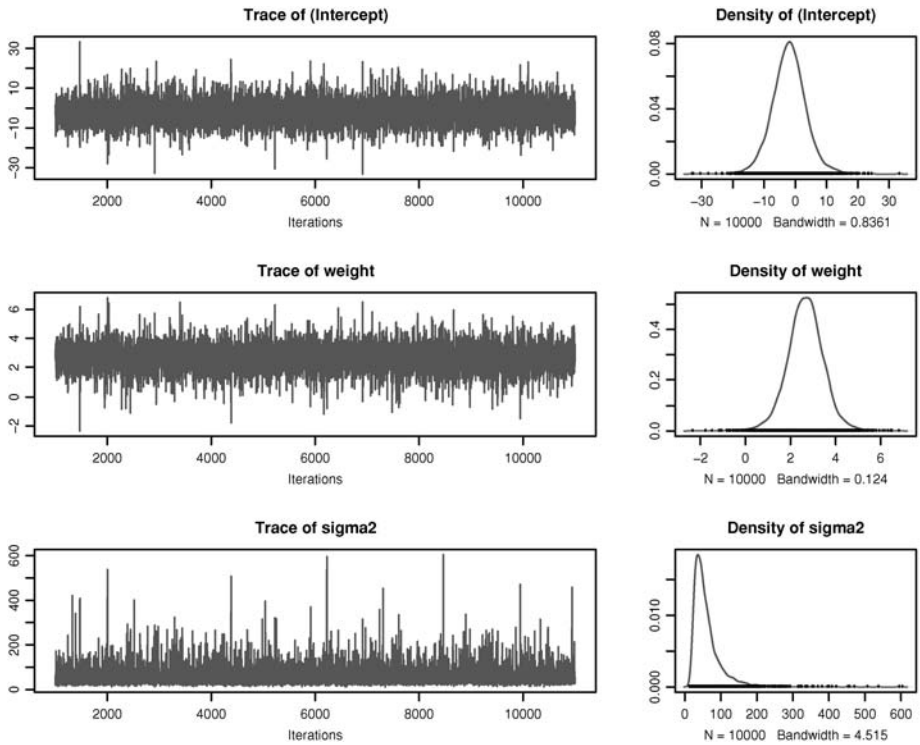


Figure 5.16 Diagnostic plots for the Bayesian analysis that used `MCMCregress()`.

There is a plot method for objects of class `mcmc` that allows a check on whether an adequate number of iterations were allowed for burn-in. Figure 5.16 shows this information. The layout has been changed somewhat from the default. The code is:

```
mat <- matrix(c(1:6), byrow=TRUE, ncol=2)
  # panels are 1, then 2, ... 6. Layout=dim(mat), i.e., 3 by 2
layout(mat, widths=rep(c(2,1),3), heights=rep(1,6))
  # NB: widths & heights are relative
plot(roller.mcmc, auto.layout=FALSE, ask=FALSE, col="gray40")
  # The method is plot.mcmc()
```

These plots are unremarkable. For this very simple model, burn-in occurs quickly, and none of the plots show any indication of a trend. The posterior distributions of the model coefficients all look plausibly normal.

The *coda* package, on which *MCMCpack* depends, has several other functions that give diagnostic information that may be helpful in interpreting the MCMC results. See `help(package="coda")`.

5.10 Recap

In exploring the relationships in bivariate data, the correlation coefficient can be a crude and unduly simplistic summary measure. Keep in mind that it measures linear association. Wherever possible, use the richer and more insightful regression framework.

The model matrix, together with the coefficients, allows calculation of predicted values. The coefficients give the values by which the values in the respective columns must be multiplied. These are then summed over all columns. In later chapters, we will use the model matrix formulation to fit models that are not inherently linear.

In the study of regression relationships, there are many more possibilities than regression lines. If a line is adequate, use that. It is in any case useful to fit a smooth curve, to see whether it suggests systematic departure from a line.

Simple alternatives to straight line regression using the original data are:

- Transform x and/or y.
- Use polynomial regression.
- Fit a smoothing curve.

Following the calculations:

- Plot residuals against fitted values.
- If it seems necessary, do a plot that checks homogeneity of variance.

Use of the function `plot()`, with an `lm` model, gives both these plots, by default as the first and third plots (cf. panels A and C in Figure 5.6.)

For size and shape data, the equation that assumes allometric variation is a good starting point. Relationships between the logarithms of the size variables are linear.

The line for the regression of y on x is different from the line for the regression of x on y. The difference between the two lines is most marked when the correlation is small.

5.11 Methodological references

We refer the reader to the suggestions for further reading at the end of Chapter 6.

5.12 Exercises

1. The data sets `elastic1` and `elastic2` were obtained using the same apparatus, including the same rubber band, as the data frame `elasticband`. Using a different symbol and/or a different color, plot the data from the two data frames `elastic1` and `elastic2` on the same graph. Do the two sets of results appear consistent?

2. For each of the data sets `elastic1` and `elastic2`, determine the regression of `stretch` on `distance`. In each case determine

   (i) fitted values and standard errors of fitted values and
   (ii) the R^2 statistic. Compare the two sets of results. What is the key difference between the two sets of data?

   Use the robust regression function `rlm()` from the *MASS* package to fit lines to the data in `elastic1` and `elastic2`. Compare the results with those from use of `lm()`. Compare regression coefficients, standard errors of coefficients, and plots of residuals against fitted values.

3. Using the data frame `cars` (*datasets*), plot `distance` (i.e., stopping distance) versus `speed`. Fit a line to this relationship, and plot the line. Then try fitting and plotting a quadratic curve. Does the quadratic curve give a useful improvement to the fit? [Readers who have studied the

relevant physics might develop a model for the change in stopping distance with speed, and check the data against this model.]

4. Calculate volumes (`volume`) and page areas (`area`) for the books on which information is given in the data frame `oddbooks` (*DAAG*).

   (a) Plot `log(weight)` against `log(volume)`, and fit a regression line.
   (b) Plot `log(weight)` against `log(area)`, and again fit a regression line.
   (c) Which of the lines (a) and (b) gives the better fit?
   (d) Repeat (a) and (b), now with `log(density)` in place of `log(weight)` as the dependent variable. Comment on how results from these regressions may help explain the results obtained in (a) and (b).

5. In the data set `pressure` (*datasets*), examine the dependence of pressure on temperature. [The relevant theory is that associated with the Claudius–Clapeyron equation, by which the logarithm of the vapor pressure is approximately inversely proportional to the absolute temperature. For further details of the Claudius–Clapeyron equation, search on the internet, or look in a suitable reference text.]

6* Look up the help page for the function `boxcox()` from the *MASS* package, and use this function to determine a transformation for use in connection with Exercise 5. Examine diagnostics for the regression fit that results following this transformation. In particular, examine the plot of residuals against temperature. Comment on the plot. What are its implications for further investigation of these data?

7. Annotate the code that gives panels B and D of Figure 5.3, explaining what each function does, and what the function arguments are.

8. The following is a simplified version of the code used for the two panels of Figure 5.4:

```
## requires the data frame ironslag (DAAG)
xy <- with(ironslag, lowess(chemical ~ magnetic))
chemfit <- approx(xy$x, xy$y, xout=ironslag$magnetic, ties=
"ordered")$y
res2 <- with(ironslag, chemical - chemfit)
plot(res2 ~ magnetic, data=ironslag) # Panel A
abline(v=0, lty=2)
sqrtabs2 <- sqrt(abs(res2))
plot(sqrtabs2 ~ chemfit, type="n")   # Panel B
panel.smooth(chemfit, sqrtabs2)
```

   Examine the help page for `lowess()`, and explain why the call to `approx()` is needed.

9. In the data frame `nswdemo` (*DAAG*), plot 1978 income (`re78`) against 1975 income (`re75`). What features of the plot make the fitting of a regression relationship a challenge?

   (a) Restricting attention to observations for which both `re78` and `re75` are non-zero, plot `log(re78)` against `log(re75)`, and fit a trend curve. Additionally, fit a regression line to the plot. Does the regression line accurately describe the relationship. In what respects is it deficient?
   (b) Now examine the diagnostic plot that is obtained by using `plot()` with the regression object as parameter. What further light does this shed on the regression line model?

10. Write a function which simulates simple linear regression data from the model

$$y = 2 + 3x + \varepsilon$$

    where the noise terms are independent normal random variables with mean 0 and variance 1.

Using the function, simulate two samples of size 10. Consider two designs: first, assume that the x-values are independent uniform variates on the interval $[-1, 1]$; second, assume that half of the x-values are -1s, and the remainder are 1s. In each case, compute slope estimates, standard error estimates, and estimates of the noise standard deviation. What are the advantages and disadvantages of each type of design?

11. For each of the data sets `elastic1` and `elastic2`, simulate artificial data from the model that was fitted.

    (a) Thus, after fitting the `elastic1` data using
    ```
    e1.lm <- lm(distance ~ stretch, data=elastic1)
    ```
    simulate artificial data from this model (conditional on the stretch measurements) using
    ```
    elastic1$newdistance <-
       cbind(rep(1, 7), elastic1$stretch)%*%coef(e1.lm) +
             rnorm(7, sd=summary(e1.lm)$sigma)
    ```
    (b) Investigate the use of the function `simulate()` as an alternative to using the above code.

    (c) Now, regress `newdistance` against `stretch` and obtain side-by-side residual plots for the original data and the artificial data. Repeat this procedure several times. Does it seem that the outliers in the original residual plot are consistent with the fitted model? Apply the same procedure to the `elastic2` data.
    [Technically, the methodology used here is that of a *parametric* bootstrap.]

12.* The following function returns the coefficient of the estimated linear functional relationship between `x` and `y`:

```
"funRel" <-
  function(x=leafshape$logpet, y=leafshape$loglen, scale=c(1,1)){
## Find principal components rotation; see Section 11.1
## Here (unlike 11.1) the interest is in the final component
    xy.prc <- prcomp(cbind(x,y), scale=scale)
    b <- xy.prc$rotation[,2]/scale
    bxy <- -b[1]/b[2]          # slope - functional eqn line
    c(bxy = bxy)
  }
## Try the following:
funRel(scale=c(1,1))     # Take x and y errors as equally important
funRel(scale=c(1,10))    # Error is mostly in y; structural relation
                         # line is close to regression line of y on x
funRel(scale=c(10,1))    # Error is mostly in x ...
## Note that all lines pass through (xbar, ybar)
```

    (a) Note where, for each of the three settings of the argument `scale`, the values of the functional coefficient lie in the range between $b_{y.x}$ and $b_{x.y}^{-1}$, where $b_{y.x}$ is the slope of the regression line of y on x and $b_{x.y}$ is the slope of the regression line of x on y.

    (b) Repeat this for each of the data frames `softbacks` and `elastic2` and (with the variables `logpet` and `loglen`) `leafshape17`.

    (c) Explain the effect of changing the settings of the argument `scale`.

6

Multiple linear regression

In straight line regression, a response variable y is regressed on a single explanatory variable x. Multiple linear regression generalizes this methodology to allow multiple explanatory or predictor variables. The focus may be on accurate prediction. Or it may, alternatively or additionally, be on the regression coefficients themselves. Be warned that interpreting the regression coefficients is not as straightforward as it might appear.

The discussion will emphasize the use of model diagnostics, and of graphs that give insight into the model fit. Diagnostic checks are intended to assist in the tuning of models so that they perform well when used for their intended purpose, and do not give unexpected and perhaps unpleasant surprises. For example, a model fit that is unduly affected by influential outliers may give results that are less than satisfactory with the main body of the data. This is one of several common types of departure from model assumptions that diagnostic checks may bring to attention.

6.1 Basic ideas: a book weight example

The book weight example has two x-variables in the regression equation. In the data shown in Figure 6.1 and printed to the right of the figure, seven books with hardback covers have been added to the eight softbacks. Code for the figure is:

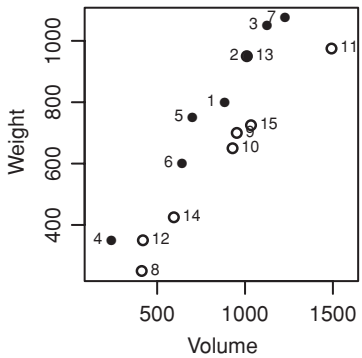
```
## Plot weight vs volume: data frame allbacks (DAAG)
plot(weight ~ volume, data=allbacks, pch=c(16,1)[unclass(cover)])
 # unclass(cover) gives the integer codes that identify levels
with(allbacks, text(weight ~ volume, labels=paste(1:15),
                    pos=c(2,4)[unclass(cover)]))
```

Explanatory variables are the volume of the book ignoring the covers, and the total area of the front and back covers. We might expect that

$$\text{weight of book} = b_0 + b_1 \times \text{volume} + b_2 \times \text{area of covers}.$$

The intercept, b_0, may not be needed. However, we will retain it for the moment. Later, we can decide whether to set it to zero. Here is the regression output:

```
> summary(allbacks.lm <- lm(weight ~ volume+area, data=allbacks))
. . . .
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```



	volume (cm^3)	area (cm^2)	weight (g)	cover
1	885	382	800	hb
...				
7	1228	396	1075	hb
8	412	0	250	pb
...				
15	1034	0	725	pb

Figure 6.1 Weight versus volume, for seven hardback and eight softback books. Filled dots are hardbacks, while open dots are softbacks. Selected data are shown to the right of the graph.

```
(Intercept)  22.4134    58.4025    0.38  0.70786
volume        0.7082     0.0611   11.60    7e-08
area          0.4684     0.1019    4.59  0.00062

Residual standard error: 77.7 on 12 degrees of freedom
Multiple R-Squared: 0.928,      Adjusted R-squared: 0.917
F-statistic: 77.9 on 2 and 12 DF,  p-value: 1.34e-007
> ## coefficient estimates and SEs only: summary(allbacks.lm)$coef
```

The coefficient estimates are $b_0 = 22.4$, $b_1 = 0.708$, and $b_2 = 0.468$. Standard errors and p-values are provided for each estimate. Note that the p-value for the intercept suggests that it cannot be distinguished from 0, as we guessed earlier. The p-value for `volume` tests $b_1 = 0$, in the equation that has both `volume` and `area` as explanatory variables.

The estimate of the noise standard deviation (the residual standard error) is 77.7. There are now $15 - 3 = 12$ degrees of freedom for the residual; we start with 15 observations and estimate three parameters. In addition, there are two versions of R^2.

The output is geared towards various tests of hypotheses. The F-statistic allows an overall test of significance of the regression. The null hypothesis for this test is that all coefficients (other than the intercept) are 0. Here, we obviously reject this hypothesis and conclude that the equation does have explanatory power.

The t-statistics and associated p-values should however be used informally, rather than as a basis for formal tests of significance. Even in this simple example, the output has four p-values. This may not be too bad, but what if there are six or eight p-values? There are severe problems in interpreting results from such a multiplicity of formal tests, with varying amounts of dependence between the various tests.

The information on individual regression coefficients can readily be adapted to obtain a confidence interval for the coefficient. The 5% critical value for a t-statistic with 12 degrees of freedom is 2.18.[1] Thus, a 95% confidence interval for `volume` is $0.708 \pm 2.18 \times 0.0611$,

[1] `## 5% critical value; t-statistic with 12 d.f.`
`qt(0.975, 12)`

i.e., it ranges from 0.575 to 0.841. As for the tests of hypotheses that were noted, these confidence intervals are not independent between parameters.

A sequential analysis of variance table assesses the contribution of each predictor variable to the model in turn, assuming inclusion of previously assessed variables. It can be obtained using the `anova()` function.

```
> anova(allbacks.lm)

Analysis of Variance Table

Response: weight
          Df Sum Sq Mean Sq F value  Pr(>F)
volume     1 812132  812132   134.7   7e-08
area       1 127328  127328    21.1 0.00062
Residuals 12  72373    6031
```

This table gives the contribution of `volume` after fitting the overall mean, then the contribution of `area` after fitting both the overall mean and `volume`. The p-value for area in the anova table must agree with that in the main regression output, since both these p-values test the contribution of `area` after including `volume` in the model. The p-values for `volume` will differ if there is a correlation between `volume` and `area`. Here, the correlation of `volume` with `area` is 0.0015, i.e., small.[2] As a consequence, the p-values for `volume` are very nearly equal.

Finally, note the model matrix that has been used in the least square calculations:

```
> model.matrix(allbacks.lm)

   (Intercept) volume area
1            1    885  382
. . . .
7            1   1228  396
8            1    412    0
. . . .
15           1   1034    0
```

Predicted values are given by multiplying the first column by b_0 (=22.4), the second by b_1 (=0.708), the third by b_2 (=0.468), and adding.

6.1.1 Omission of the intercept term

We now investigate the effect of leaving out the intercept. Here is the regression output:

```
> allbacks.lm0 <- lm(weight ~ -1+volume+area, data=allbacks)
> summary(allbacks.lm0)
 . . .

Coefficients:
```

[2] `## Correlation of volume with area`
`with(allbacks, cor(volume,area))`

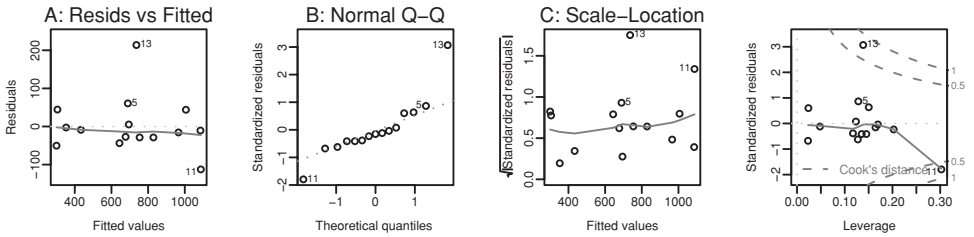


Figure 6.2 Diagnostic plots for the model that fits `weight` as a function of `volume` and `area`, omitting the intercept.

```
        Estimate Std. Error t value Pr(>|t|)
volume    0.7289     0.0277   26.34  1.1e-12
area      0.4809     0.0934    5.15  0.00019

Residual standard error: 75.1 on 13 degrees of freedom
Multiple R-Squared: 0.991,      Adjusted R-squared: 0.99
F-statistic:  748 on 2 and 13 DF,  p-value: 3.8e-014
```

The regression coefficients now have smaller standard errors. The reason is that, in the model that included the intercept, there was a substantial negative correlation between the estimate of the intercept and the coefficient estimates. The reduction in standard error is greater for the coefficient of `volume`, where the correlation was −0.88, than for `area`, where the correlation was −0.32. Correlations between estimates can be obtained by setting `corr=TRUE` in the call to `summary()`:

```
## Display correlations between estimates of model coefficients
summary(allbacks.lm, corr=TRUE)
```

6.1.2 Diagnostic plots

Figure 6.2 displays useful information for checking on the adequacy of the model fit to the `allbacks` data. It used the code:

```
par(mfrow=c(2,2))     # Get all 4 plots on one page
plot(allbacks.lm0)
par(mfrow=c(1,1))
```

Figure 6.2 gives the default set of diagnostic plots. By comparison with Figure 5.6D in Section 5.2, it disentangles the contributions that the residual and the leverage make to the influence. Note the large residual (panel A) for observation 13. Note also that observation 13 lies outside the 0.5 contour of Cook's distance, well out towards the contour for a Cook's distance of 1. Thus it is a (somewhat) influential point. The Cook's distance measure, which was mentioned in Section 5.2, will be discussed in more detail in Subsection 6.3.1.

Should we omit observation 13? The first task is to check the data, which are however correct. The book is a computing book, with a smaller height to width ratio than any of the other books. It has heavier paper, though differences in the paper may not be immediately obvious. It may be legitimate to omit it from the main analysis, but noting the omission

Table 6.1 *Distance* (`dist`)*, height climbed* (`climb`)*, and record times* (`time`)*, for four of the 23 Northern Irish hill races.*

	Name	`dist` (mi)	`climb` (ft)	`time` (h)	`timef` (h)
1	Binevenagh	7.5	1740	0.86	1.06
2	Slieve Gullion	4.2	1110	0.47	0.62
3	Glenariff Mountain	5.9	1210	0.70	0.89
...	...	...	...	...	...
23	BARF Turkey Trot	5.7	1430	0.71	0.94

of this one book that had a much higher weight-to-volume ratio than other books. The following omits observation 13:

```
> allbacks.lm13 <- lm(weight ~ -1+volume+area, data=allbacks[-13, ])
> summary(allbacks.lm13)
 . . .

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
volume   0.6949     0.0163    42.6  1.8e-14
area     0.5539     0.0527    10.5  2.1e-07

Residual standard error: 41 on 12 degrees of freedom
Multiple R-Squared: 0.997,      Adjusted R-squared: 0.997
F-statistic: 2.25e+003 on 2 and 12 DF,  p-value: 3.33e-016
```

The residual standard error is substantially smaller (41 instead of 75.1) in the absence of observation 13. Observation 11 now has a Cook's distance that is close to 1, but does not stand out in the plot of residuals. This is about as far as it is reasonable to go in the investigation of diagnostic plots. With just 15 points, there is a risk of over-interpretation.

6.2 The interpretation of model coefficients

If an aim is a scientific understanding that involves interpretation of model coefficients, then it is important to fit a model whose coefficients are open to the relevant interpretations. Different formulations of the regression model, or different models, may serve different explanatory purposes. Predictive accuracy is in any case a consideration, and is often the main interest.

Three examples will demonstrate issues for the interpretation of model coefficients. The first is a data set on record times, distances, and amounts of climb for Northern Irish hill races. The second has data on mouse brain weight, litter size, and body weight. The third has data on book dimensions and weight, from a highly biased sample of books.

6.2.1 Times for Northern Irish hill races

The data set `nihills` (*DAAG*), from which Table 6.1 has selected observations, gives distances (`dist`), heights climbed (`climb`), male record times (`time`), and female record times (`timef`), for 23 Northern Irish hill races.

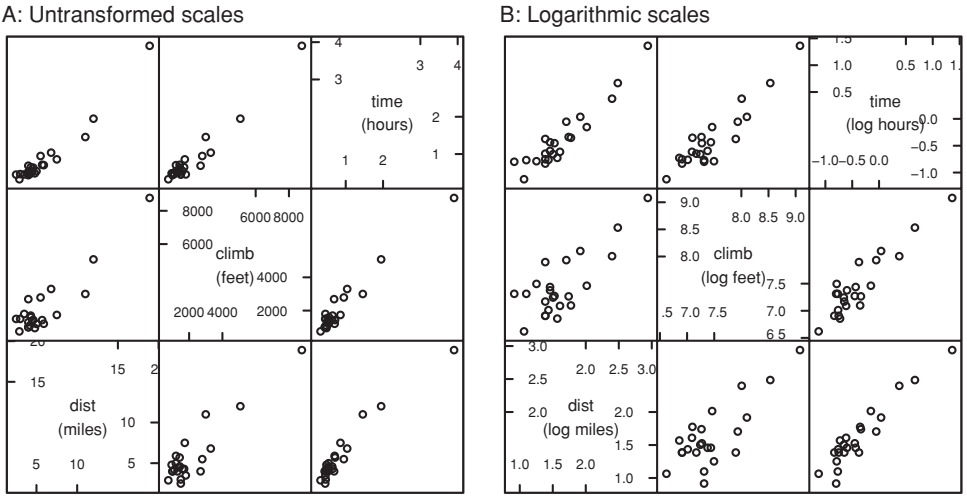


Figure 6.3 Scatterplot matrix for the `nihills` data (Table 6.1). Panel A uses the untransformed scales, while panel B uses logarithmic scales.

We begin with scatterplot matrices, both for the untransformed data (Figure 6.3A), and for the log transformed data (Figure 6.3B). Attention is limited to the male results.[3] The diagonal panels give the x-variable names for all plots in the column above or below, and the y-variable names for all plots in the row to the left or right. Note that the vertical axis labels alternate between the axis on the extreme left and the axis on the extreme right, and similarly for the horizontal axis labels. This avoids a proliferation of axis labels on the extreme left and lower axes.

Apart from a possible outlier, the relationship between `dist` and `climb` seems approximately linear. The same is true when logarithmic scales are used, and the outlier is now much less evident. Nonlinear relationships are undesirable, in part, because they create problems for the interpretation of diagnostic plots.

The following are reasons for investigating the taking of logarithms:

- The range of values of `time` is large (3.9:0.32, i.e., >10:1), and similarly for `dist` and `climb`. The times are bunched up towards zero, with a long tail. In such instances, use of a logarithmic transformation is likely to lead to a more symmetric distribution.
- One point in particular has a time that is more than twice that of the next largest `time`, as is evident in Figure 6.3A. The values of `dist` and `climb` similarly stand out as much larger than for other points. In a regression that uses the untransformed variables, this point will have a much greater say in determining the regression equation than any

[3]
```
## : data frame nihills (DAAG)
## Panel A: Scatterplot matrix, untransformed data, data frame nihills (DAAG)
library(lattice); library(DAAG)
splom(~ nihills[, c("dist","climb","time")], cex.labels=1.2,
      varnames=c("dist\n(miles)","climb\n(feet)", "time\n(hours)"))
## Panel B: log transformed data
splom(~ log(nihills[, c("dist","climb","time")]), cex.labels=1.2,
      varnames=c("dist\n(log miles)", "climb\n(log feet)", "time\n(log hours)"))
```

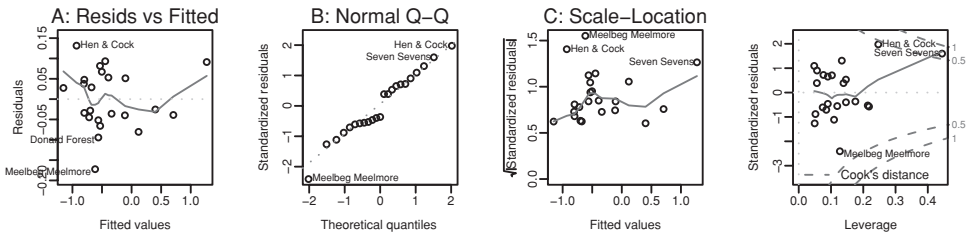


Figure 6.4 Diagnostic plots for the regression of `log(time)` on `log(dist)` and `log(climb)`.

other point. In the terminology of Subsection 6.3.1, it has large *leverage*. Even after taking logarithms (Figure 6.3B), its leverage remains large, but not quite so dominating.

- It can be expected that `time` will increase more than linearly at very long times, and similarly for `climb`, as physiological demands on the human athlete move closer to limits of human endurance.
- Such relationship as is evident between the explanatory variables (*dist* and *climb*) is more nearly linear on the logarithmic scale of Figure 6.3B.

Additionally, use of a logarithmic scale may help stabilize the variance.

These considerations suggest fitting the equation

$$\log(\texttt{time}) = a + b_1 \log(\texttt{dist}) + b_2 \log(\texttt{climb}).$$

This is equivalent to the power relationship

$$\texttt{time} = A(\texttt{dist})^{b_1}(\texttt{climb})^{b_2},$$

where $a = \log(A)$.

Now fit the model and examine the diagnostic plots (shown in Figure 6.4):

```
nihills.lm <- lm(log(time) ~ log(dist) + log(climb), data = nihills)
plot(nihills.lm)
```

The diagnostic plots do not indicate problems, apart from the moderately large residual associated with the Meelbeg Meelmore race.

The estimates of the coefficients (a, b_1 and b_2) are:

```
> summary(nihills.lm)$coef
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -4.961     0.2739   -18.1 7.09e-14
log(dist)       0.681     0.0552    12.3 8.19e-11
log(climb)      0.466     0.0453    10.3 1.98e-09
              Estimate Std. Error   t value     Pr(>|t|)
```

Interpreting the coefficients

The estimated equation is

$$\log(\texttt{time}) = \underset{}{-4.96} + \underset{[\mathrm{SE}=0.27]}{0.68} \times \log(\texttt{dist}) + \underset{[\mathrm{SE}=0.055]}{0.47} \times \log(\texttt{climb}) + \underset{[\mathrm{SE}=0.045]}{}.$$

Exponentiating both sides of this equation, and noting $\exp(-4.96) = 0.0070$, gives

$$\texttt{time} = 0.00070 \times \texttt{dist}^{0.68} \times \texttt{climb}^{0.47}.$$

This equation implies that for a given height of climb, the time taken is smaller for the second three miles than for the first three miles. The relative rate of increase in time is 68% of the relative rate of increase in distance. Is this plausible?

The answer comes from examination of the implications of holding `climb` constant. For a given value of `climb`, short races will be steep while for long races the slope will be relatively gentle. Thus a coefficient that is less than 1.0 is unsurprising.

A meaningful coefficient for `logdist`

The coefficient for `logdist` will be more meaningful if we regress on `logdist` and `log(climb/dist)`. Then we find:

```
> lognihills <- log(nihills)
> names(lognihills) <- paste("log", names(nihills), sep="")
> lognihills$logGrad <- with(nihills, log(climb/dist))
> nihillsG.lm <- lm(logtime ~ logdist + logGrad, data=lognihills)
> summary(nihillsG.lm)$coef
            Estimate Std. Error t value Pr(>|t|)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -4.961     0.2739   -18.1 7.09e-14
logdist        1.147     0.0346    33.2 5.90e-19
logGrad        0.466     0.0453    10.3 1.98e-09
```

The coefficient of `logdist` is now, reassuringly, greater than 1. The coefficient of `logdist` depends, critically, on what other explanatory variables are used!

There is another, related, benefit. The correlation between `logdist` and `logGradient` is −0.065, negligible relative to the correlation of 0.78 between `logdist` and `logclimb`.[4] Because the correlation between `logdist` and `logGradient` is so small, the coefficient of `logdist` (=1.124) in the regression on `logdist` alone is almost identical to the coefficient of `logdist` (=1.147) in the regression on `logdist` and `logGradient`.

The standard error of the coefficient of `logdist` is smaller – 0.035 as against 0.055 – when the second explanatory variable is `logGradient` rather than `logclimb`. Note that the predicted values do not change. The models `nihills.lm` `nihillsG.lm` are different mathematical formulations of the same underlying model.

6.2.2 Plots that show the contribution of individual terms

For simplicity, the discussion will assume just two explanatory variables, x_1 and x_2, with the intention of showing the contribution of each in turn to the model.

[4] `## Correlations of logGrad and logclimb with logdist`
`with(lognihills, cor(cbind(logGrad, logclimb), logdist))`

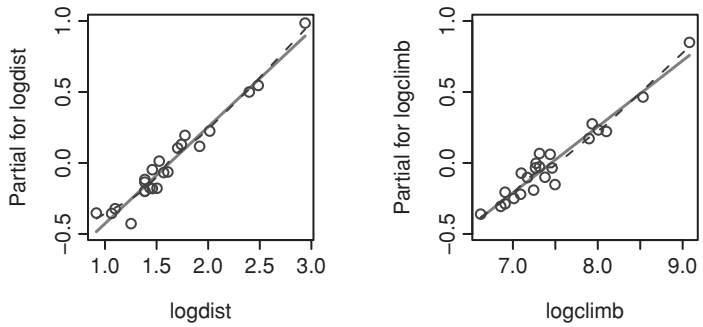


Figure 6.5 The solid lines show the respective contributions of the two model terms, in the regression of `logtime` on `logdist` and `logclimb`. Partial residuals (specify `partial.resid=TRUE`) and an associated smooth curve (specify `smooth=panel.smooth`) have been added.

The fitting of a regression model makes it possible to write:

$$y = b_0 + b_1 x_1 + b_2 x_2 + e \tag{6.1}$$

$$= \widehat{y} + e, \tag{6.2}$$

where $\widehat{y} = b_0 + b_1 x_1 + b_2 x_2$.

Another way to write the model that is to be fitted is:

$$y - \bar{y} = a + b_1(x_1 - \bar{x}_1) + b_2(x_2 - \bar{x}_2) + e.$$

It is a fairly straightforward algebraic exercise to show that $a = 0$. Notice that, for fitting the model in this form:

- The observations are $y - \bar{y}$, with mean zero.
- The first explanatory variable is $x_1 - \bar{x}_1$, with mean zero, and the first term in the model is $b_1(x_1 - \bar{x}_1)$, with mean zero.
- The second explanatory variable is $x_2 - \bar{x}_2$, with mean zero, and the first term in the model is $b_1(x_1 - \bar{x}_1)$, with mean zero.

The residuals e are exactly the same as before, and have mean zero.

The fitted model can then be written:

$$y = \bar{y} + b_1(x_1 - \bar{x}_1) + b_2(x_2 - \bar{x}_2) + e$$

$$= \bar{y} + t_1 + t_2 + e.$$

This neatly splits the response value y into three parts – an overall mean $\bar{y}$, a term that is due to x_1, a term that is due to x_2, and a residual e. Moreover, the values of t_1 and t_2 sum, in each case, to zero.

The `predict()` function has an option (`type="terms"`) that gives t_1 and t_2.

```
yterms <- predict(nihills.lm, type="terms")
```

The first column of `yterms` has the values of $t_1 = b_1(x_1 - \bar{x}_1)$, while the second has the values of t_2. Values in both these columns sum to zero. The solid lines of the *component plus residual* plot in Figure 6.5 show the contributions of the individual terms to the model.

The solid line in the left panel shows a plot of $b_1(x_1 - \bar{x}_1)$ against x_1, while the solid line in the right panel shows a plot of $b_2(x_2 - \bar{x}_2)$ against x_2.

The lines can be obtained directly with the `termplot()` command. As the contributions of the terms are linear on a logarithmic scale, a plot in which the x-axis variables are the log transformed variables will serve the purpose best. We therefore refit the equation before using `termplot()`, thus:

```
lognihills <- log(nihills)
names(lognihills) <- paste("log", names(nihills), sep="")
nihills.lm <- lm(logtime ~ logdist + logclimb, data = lognihills)
## To show points, specify partial.resid=TRUE
## For a smooth curve, specify smooth=panel.smooth
termplot(nihills.lm, partial.resid=TRUE, smooth=panel.smooth,
         col.res="gray30")
```

The plotted points are the *partial residuals*, for the respective term.

- The vector $t_1 + e = \widehat{y} - t_2$ holds the partial residuals for x_1 given x_2, i.e., they account for that part of the response that is not explained by the term in x_2.
- The vector $t_2 + e$ holds the partial residuals for x_2 given x_1.

The smooth curve that has been passed through the partial residuals is designed to help in assessing any departure from the lines. Both plots show a hint of curvature, more pronounced for `logdist` than `logclimb`. The difference from a line is however small, and may not be of much practical consequence.

Each plot indicates the pattern in the residuals when there is no change to the linear pattern of response that is assumed for the other variable. Both plots suggest a slight but noticeable departure from linearity, though in the right panel largely due to a single point. Further investigation, if it should seem warranted, can best proceed using methods for fitting smooth curves that will be described in Chapter 7.

6.2.3 Mouse brain weight example

The `litters` data frame (*DAAG* library) has observations on brain weight, body weight, and litter size of 20 mice. As Figure 6.6 makes clear, the explanatory variables `lsize` and `bodywt` are strongly correlated. Code for obtaining a simplified version of Figure 6.6 is:[5]

```
pairs(litters)       # For improved labeling, see the footnote.
```

Observe now that, in a regression with `brainwt` as the dependent variable, the coefficient for `lsize` has a different sign (−ve versus +ve) depending on whether or not `bodywt`

[5]
```
## Scatterplot matrix for data frame litters (DAAG); labels as in figure
library(lattice)
splom(~litters, varnames=c("lsize\n\n(litter size)", "bodywt\n\n(Body Weight)",
      "brainwt\n\n(Brain Weight)"))
```

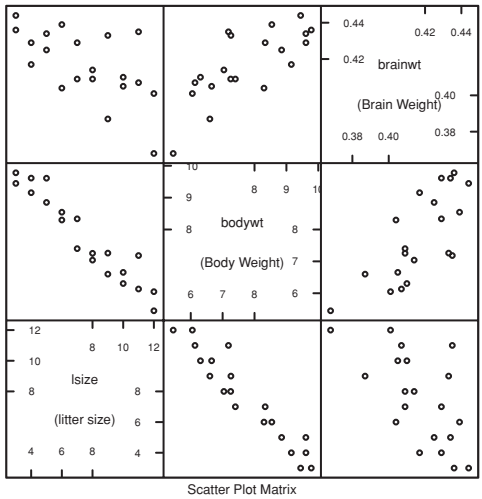


Figure 6.6 Scatterplot matrix for the litters data set. Data relate to Wainright *et al.* (1989).

also appears as an explanatory variable. Both coefficients are significant ($p < 0.05$). Here are the calculations:

```
> ## Regression of brainwt on lsize
> summary(lm(brainwt ~ lsize, data = litters))$coef
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.44700    0.00962   46.44 3.39e-20
lsize       -0.00403    0.00120   -3.37 3.44e-03
> ## Regression of brainwt on lsize and bodywt
> summary(lm(brainwt ~ lsize + bodywt, data = litters))$coef
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.17825    0.07532    2.37  0.03010
lsize        0.00669    0.00313    2.14  0.04751
bodywt       0.02431    0.00678    3.59  0.00228
```

The coefficients have different interpretations in the two cases:

- In the first regression, variation in `brainwt` is being explained only with `lsize`, regardless of `bodywt`. No adjustment has been made for the fact that `bodywt` increases as `lsize` decreases: individuals having small values of `lsize` have `brainwt` values corresponding to large values of `bodywt`, while individuals with large values of `lsize` have `brainwt` values corresponding to low `bodywt` values.
- In the multiple regression, the coefficient for `lsize` is a measure of the change in `brainwt` with `lsize`, when `bodywt` is held constant. For any particular value of `bodywt`, `brainwt` increases with `lsize`. This was a noteworthy finding for the purposes of the study.

The results are consistent with the biological concept of brain sparing, whereby the nutritional deprivation that results from large litter sizes has a proportionately smaller effect on brain weight than on body weight.

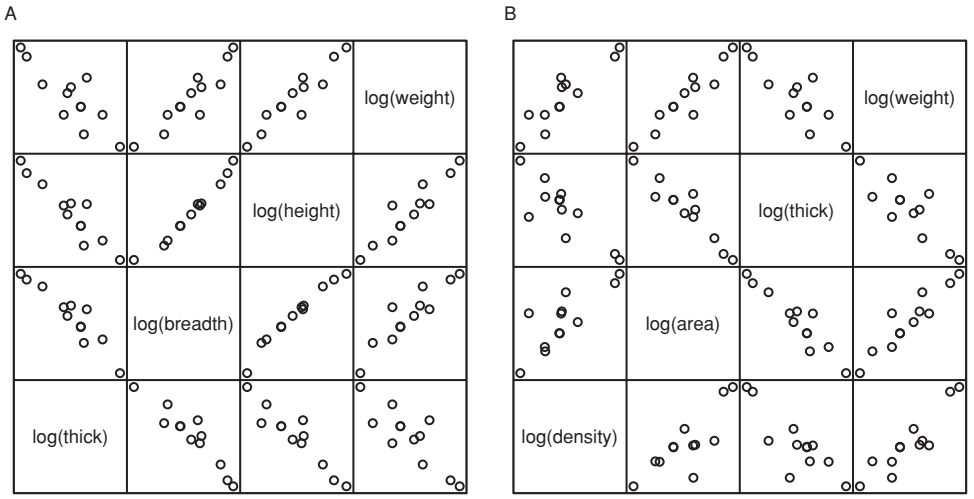


Figure 6.7 Panel A is the scatterplot matrix for the logarithms of the variables in the `oddbooks` data frame. Books were selected in such a way that weight increased with decreasing thickness. Panel B has the scatterplot matrix for the logarithms of the derived variables `density` and `area`, together with `log(thick)` and `log(weight)`.

6.2.4 Book dimensions, density, and book weight

The way that data are sampled can affect the coefficients. This section will examine data, sampled in a deliberately biased way, on the effect of book dimensions (thickness, height, and width) on book weight.

Figure 6.7A shows a scatterplot matrix for logged measurements, from the data frame `oddbooks`, on 12 soft-cover books. Figure 6.7B is for later reference.[6] Books were selected in such a way that weight increased with decreasing thickness, reflected in the strong negative correlation between log(`weight`) and log(`thick`).

It might be expected that weight would be proportional to volume, i.e., $w = tbh$ and

$$\widehat{\log(w)} = \log(t) + \log(b) + \log(h) \tag{6.3}$$

where $w =$ `weight`, $t =$ `thick`, $b =$ `breadth`, and $h =$ `height`.

Although equation (6.3) seems plausible, there can be no guarantee that it will give a result that makes sense for data where there have been strong constraints on the choice of books. We will fit models in which the fitted values take the following forms:

$$\begin{aligned} 1 &: \log(w) = a_0 + a_1(\log(t) + \log(b) + \log(h)) \\ 2 &: \log(w) = a_0 + a_1\log(t) + a_2(\log(b) + \log(h)) \\ 3 &: \log(w) = a_0 + a_1\log(t) + a_2\log(b) + a_3\log(h) \end{aligned}$$

[6]
```
## Code for Panel A
splom(~log(oddbooks), varnames=c("thick\n\nlog(mm)", "breadth\n\nlog(cm)",
      "height\n\nlog(cm)", "weight\n\nlog(g)"), pscales=0)
## Code for Panel B
oddothers <-
  with(oddbooks,
       data.frame(density = weight/(breadth*height*thick),
                  area = breadth*height, thick=thick, weight=weight))
splom(~log(oddothers), pscales=0,
      varnames=c("log(density)", "log(area)", "log(thick)", "log(weight)"))
```

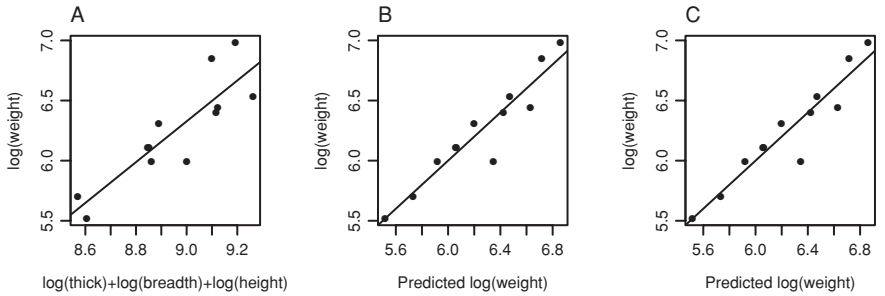


Figure 6.8 Panel A plots log(`weight`) against log(`thick`) + log(`breadth`) + log(`height`). The dashed line shows the fitted values for model 1. Panel B plots observed values against fitted values for model 2, with the line $y = x$ superposed. Panel C plots observed values against fitted values for model 3 and again shows the line $y = x$.

Figure 6.8 assists comparison of these models. The coefficients in the fitted equations, with SEs in square brackets underneath, are:[7]

$$1 : \log(w) = \underset{[\text{SE}=2.7]}{-8.9} + \underset{[0.31]}{1.7} \times (\log(t) + \log(b) + \log(h))$$
$$2 : \log(w) = \underset{[2.9]}{-1.6} + \underset{[0.42]}{0.48}\log(t) + \underset{[0.28]}{1.10}(\log(b) + \log(h))$$
$$3 : \log(w) = \underset{[3.2]}{-0.72} + \underset{[0.43]}{0.46}\log(t) + \underset{[1.07]}{1.88}\log(b) + \underset{[1.27]}{0.15}\log(h))$$

Transforming back to a relationship between weight and volume, the result for model 1 is that `weight` $\propto$ `volume`$^{1.7}$, implying that weight increases faster than volume. As the volume of books increases, their density increases. Almost certainly, the effect has arisen because the books with larger page sizes are printed on heavier paper.

Note that the predicted values in model 2 are very similar to those for model 3. The coefficient of `area` in model 2 indicates that, for a given value of `thick`, weight is very nearly proportional to page area.

Note finally that the regression of `log(weight)` on `log(thick)` yields:

```
> coef(summary(lm(log(weight) ~ log(thick), data=oddbooks)))
             Estimate Std. Error t value Pr(>|t|)
(Intercept)      9.69      0.708    13.7 8.35e-08
log(thick)      -1.07      0.219    -4.9 6.26e-04
```

The implication is that `weight` decreases as `thick` increases. For these data, it does!

The `oddbooks` data were contrived to give a skewed picture of the way that book weight varies with dimensions. Correlations between `area` and `thick`, and between both `area` and `thick` and density of paper, make it impossible to use multiple regression to separate the effects of these different variables. The one fairly solid piece of information

[7]
```
## Details of calculations
volume <- apply(oddbooks[, 1:3], 1, prod)
area <- apply(oddbooks[, 2:3], 1, prod)
lob1.lm <- lm(log(weight) ~ log(volume), data=oddbooks)
lob2.lm <- lm(log(weight) ~ log(thick)+log(area), data=oddbooks)
lob3.lm <- lm(log(weight) ~ log(thick)+log(breadth)+log(height), data=oddbooks)
coef(summary(lob1.lm))
## Similarly for coefficients and SEs for other models
```

that is available from these data is obtained by using our knowledge of what the relationship should be, to indicate how density changes with `area` or `thick`, thus:

```
> book.density <- oddbooks$weight/volume
> bookDensity.lm <- lm(log(book.density) ~ log(area), data=oddbooks)
> coef(summary(bookDensity.lm))
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -5.109     0.5514   -9.27 3.18e-06
log(area)       0.419     0.0958    4.37 1.39e-03
```

Observational data is very susceptible to such bias. For example solar radiation, windspeed, temperature, and rainfall may change systematically with distance up a hillside, and it may be impossible to distinguish the effects of the different factors on plant growth or on the ecology. Worse, effects may be at work that will be discerned only from substantial understanding of the physical processes and which are not obvious from the measured data.

6.3 Multiple regression assumptions, diagnostics, and efficacy measures

At this point, as a preliminary to setting out a general strategy for fitting multiple regression models, we describe the assumptions that underpin multiple regression modeling. Given the explanatory variables $x_1, x_2, \ldots, x_p$, the assumptions are that:

- The expectation $\mathrm{E}[y]$ is some linear combination of $x_1, x_2, \ldots, x_p$:
$$\mathrm{E}[y] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$
- The distribution of y is normal with mean $\mathrm{E}[y]$ and constant variance, independently between observations.

In general, the assumption that $\mathrm{E}[y]$ is a linear combination of the xs is likely to be false. It may, however, be a good approximation. In addition, there are simple checks that, if the assumption fails, may indicate the nature of the failure. The assumption may be hard to fault when the range of variation of each explanatory variable is small relative to the noise component of the variation in y, so that any non-linearity in the effects of explanatory variables is unlikely to show up. Even where there are indications that it is not entirely adequate, a simple form of multiple regression model may be a reasonable starting point for analysis, on which we can then try to improve.

6.3.1 Outliers, leverage, influence, and Cook's distance

This extends earlier discussions of regression diagnostics in Section 5.2 and Subsection 6.1.2.

Detection of outliers

Outliers can be hard to detect. Two (or more) outliers that are influential may mask each other. If this seems a possible issue, it is best to work with residuals from a resistant fit. Resistant fits aim to completely ignore the effect of outliers. Use of this methodology will be demonstrated in Subsection 6.4.3. See also Exercise 14 at the end of the chapter.

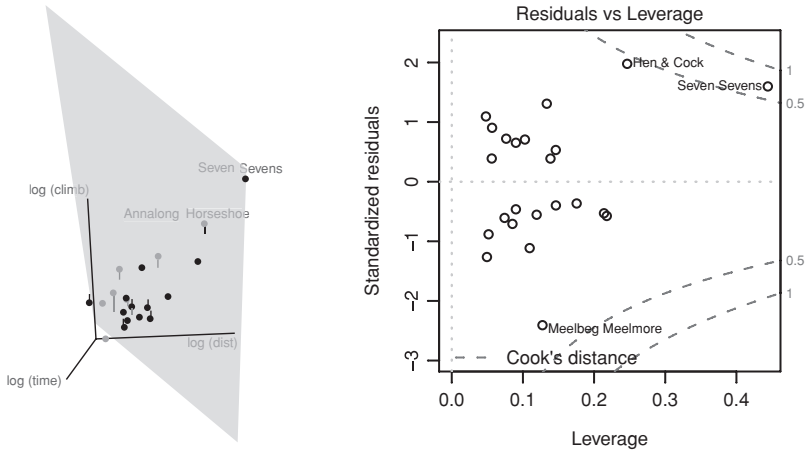


Figure 6.9 The left panel is a snapshot of a three-dimensional dynamic graphic plot. The two points that have the largest *leverage* in the regression of `logtime` on `logdist` and `logclimb` have been labeled. In the right panel, standardized residuals are plotted against leverages. Contours are shown for Cook's distances of 0.5 and 1.0.

This demonstrates the type of aberrant results that may result from resistant regression, if residuals do not have an approximately symmetric distribution.

Dynamic graphic exploration can be helpful. Abilities in the *rgl* package for dynamic three-dimensional plots can be accessed conveniently from the graphics menu of the R Commander GUI. The *rgobi* package (Cook and Weisberg, 1999) has very extensive abilities for dynamic graphic exploration.

*Leverage and the hat matrix

What difference does replacing of y_i by $y_i + \Delta_i$, while leaving other y-values unchanged, make to the fitted surface. There is a straightforward answer; the fitted value changes from $\widehat{y}_i$ to $\widehat{y}_i + h_{ii}\Delta_i$, where h_{ii} is the *leverage* for that point.

The leverage values h_{ii} are the diagonal elements of the so-called hat matrix that can be derived from the model matrix. (The name of the hat matrix H is due to the fact that the vector of fitted values ("hat" values) is related to the observed response vector by $\widehat{y} = Hy$.) Large values represent high leverage. Use the function `hatvalues()` to obtain the leverages, thus:

```
> as.vector(hatvalues(nihills.lm))  # as.vector() strips off names
 [1] 0.119 0.074 0.134 0.109 0.085 0.146 0.052 0.146 0.247 0.218
[11] 0.090 0.057 0.056 0.127 0.049 0.103 0.175 0.214 0.444 0.090
[21] 0.139 0.048 0.077
```

The largest leverage, for observation 19, is 0.44. As this is more than three times the average value of 0.13, it warrants attention. (There are $p = 3$ coefficients and $n = 23$ observations, so that the average is $p/n = 0.13$.)

The left panel of Figure 6.9 is a snapshot of a three-dimensional dynamic graphic plot that shows the regression plane in the regression of `logtime` on `logdist` and `logclimb`.

The two points that have the largest *leverage* have been labeled. The right panel is a plot of residuals against leverage values, in which the Cook's distances are shown as contours.

The snapshot in the left panel can be obtained from the 3D graph submenu on the R Commander's Graphs pulldown menu, then rotating the graph to give the view shown. Alternatively, enter from the command line the code that is given in Section 15.7. To rotate the display, hold down the left mouse button and move the mouse. Try it! The snapshot is a poor substitute for the experience of rotating the display on a computer screen!

The right panel can be obtained thus:

```
## Residuals versus leverages
plot(nihills.lm, which=5, add.smooth=FALSE)
## The points can alternatively be plotted using
## plot(hatvalues(model.matrix(nihills.lm)), residuals(nihills.lm))
```

Influential points and Cook's distance

Data points that distort the fitted response are "influential". Such distortion is a combined effect of the size of the residual, and its leverage. The Cook's distance statistic is a commonly used measure of "influence". It measures, for each observation, the change in model estimates when that observation is omitted. It measures the combined effect of leverage and of the magnitude of the residual. Recall the guideline that was given in Section 5.2, that any Cook's distance of 1.0 or more, or that is substantially larger than other Cook's distances, should be noted.

Any serious distortion of the fitted response may lead to residuals that are hard to interpret or even misleading. Thus it is wise to check the effect of removing any highly influential data points before proceeding far with the analysis.

Influence on the regression coefficients

In addition to the diagnostic plots, it is useful to investigate the effect of each observation on the estimated regression coefficients. One approach is to calculate the difference in the coefficient estimates obtained with and without each observation. The function `dfbetas()` does these calculations and standardizes the resulting differences, i.e., they are divided by a standard error estimate. These are more readily interpretable than absolute differences. For the regression model fit to the `allbacks` data set, without the intercept term, the values are shown in Figure 6.10.

If the distributional assumptions are satisfied, standardized changes that are larger than 2 can be expected, for a specified coefficient, in about 1 observation in 20. Here, the only change that seems worthy of note is for `volume` in observation 13. For absolute changes, should they be required, use the function `lm.influence()`.

Outliers, influential or not, should be taken seriously

Outliers, influential or not, should never be disregarded. Careful scrutiny of the original data may reveal an error in data entry that can be corrected. Alternatively, their exclusion may be a result of use of the wrong model, so that with the right model they can be re-incorporated.

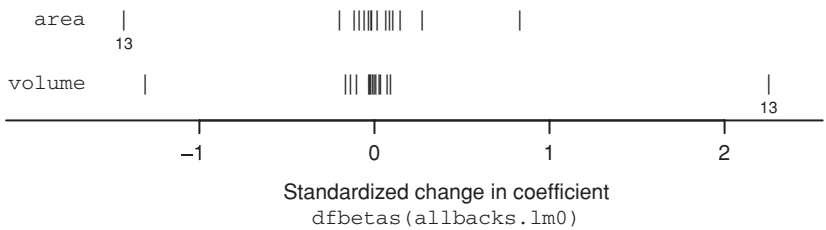


Figure 6.10 Standardized changes in regression coefficients, for the model that was fitted to the `allbacks` data set. The points for the one row (row 13) where the change for one of the coefficients was greater than 2 in absolute value are labeled with the row number.

If apparently genuine outliers remain excluded from the final fitted model, they must be noted in the eventual report or paper. They should be included, separately identified, in graphs.

*Additional diagnostic plots

The functions in the *car* package, designed to accompany Fox (2002), greatly extend the range of diagnostic plots. See the examples and references included on the help pages for this package. As an indication of what is available, try

```
library(car)
leverage.plots(allbacks.lm, term.name="volume",
               identify.points=FALSE)
```

6.3.2 Assessment and comparison of regression models

The measures that will be discussed all focus on predictive accuracy, assuming that data used for fitting the model can be treated as a random sample from the data that will be used in making predictions. This assumption may be incorrect. Also, as pointed out earlier, predictive accuracy may not be the only or the most important consideration. It is nonetheless always an important consideration, even where interpretation of model parameters is the primary focus of interest.

R^2 and adjusted R^2

The R^2 and adjusted R^2 that are included in the default output from `summary.lm()` (cf. Section 6.1) are commonly used to give a rough sense of the adequacy of a model. R^2 is the proportion of the sum of squares about the mean that is explained by the model. Adjusted R^2, which is the proportion of the mean sum of squares that is explained, is preferable to R^2. Neither is appropriate for comparisons between different studies, where the ranges of values of the explanatory variables may be different. Both are likely to be largest in those studies where the range of values of the explanatory variables is greatest.

The R^2 statistic (whether or not adjusted) has a more legitimate use for comparing different models for the same data. For this purpose, however, AIC and related statistics are much preferable. The next subsection will discuss these "information measure" statistics.

AIC and related statistics

These statistics are designed to choose, from among a small number of alternatives, the model with the best predictive power. Statistics that are in use include the Akaike Information Criterion (AIC), Mallows' C_p, and various elaborations of the AIC. The AIC criterion and related statistics can be used for more general models than multiple regression. We give the version that is used in R.

The model that gives the smallest value is preferred, whether for the AIC or for the C_p statistic. The variance estimate $\widehat{\sigma}^2$ is often determined from the full model.

Another statistic is the Bayesian Information Criterion (BIC), which is claimed as an improvement on the AIC. The AIC is inclined to over-fit, i.e., to choose too complex a model.

Calculation of R*'s version of the AIC*

Let n be the number of observations, let p be the number of parameters that have been fitted, and let RSS denote the residual sum of squares. Then, if the variance is known, R takes the AIC statistic as

$$\text{AIC} = \frac{\text{RSS}}{\sigma^2} + 2p + \text{const.}$$

where, here, the constant term arises from the assumption of an i.i.d. normal distribution for the errors. In the more usual situation where the variance is not known, R takes the AIC statistic as

$$\text{AIC} = n \log\left(\frac{\text{RSS}}{n}\right) + 2p + \text{const.}$$

The BIC (Schwarz's Bayesian criterion) statistic, which replaces $2p$ by $\log(n) \times p$, penalizes models with many parameters more strongly.

The C_p statistic differs from the AIC statistic only by subtraction of n, and by omission of the constant term. It is

$$C_p = n \log\left(\frac{\text{RSS}}{n}\right) + 2p - n.$$

6.3.3 How accurately does the equation predict?

The best test of how well an equation predicts is the accuracy of its predictions. We can measure this theoretically, by examining, e.g., 95% confidence intervals for predicted values. Alternatively we can use cross-validation, or another resampling procedure, to assess predictive accuracy. Both methods (theoretical and cross-validation) for assessing predictive accuracy assume that the sample is randomly drawn from the population that is of interest, and that any new sample for which predictions are formed is from that same population.

Earlier, we noted that a quadratic term in `ldist` would probably improve the model slightly. We can accommodate this by replacing `logdist` by `poly(logdist,2)`, or

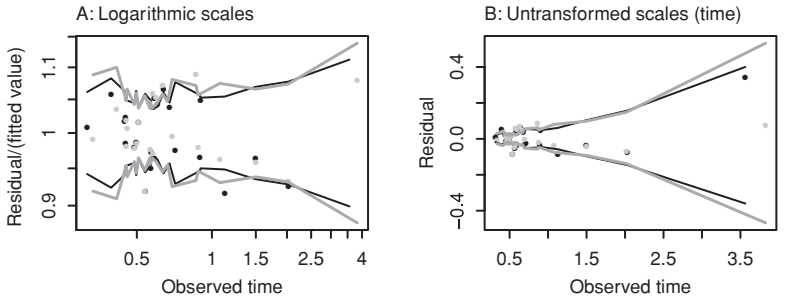


Figure 6.11 Residuals versus predicted values for the hill race data. The vertical limits of the bounding curves are 95% pointwise confidence limits, with the predicted value subtracted off. Black points and 95% curves are for the model that is linear in `logdist`, Gray points and 95% curves are for the model that is quadratic in `logdist`. Panel A uses logarithmic scales, with the points labeled on the untransformed scale.

by `poly(logdist,2,raw=TRUE)` if coefficients are required that are immediately interpretable. Orthogonal polynomials will be discussed in Section 7.4.

The following table gives 95% coverage (confidence) intervals for predicted times in the regression of `log(time)` on `log(climb)` and `log(dist)`, for the first few observations from the data set `nihills`:

```
> lognihills <- log(nihills)
> names(lognihills) <- paste("log", names(nihills), sep="")
> nihills.lm <- lm(logtime ~ logdist + logclimb, data = lognihills)
## Coverage intervals; use exp() to undo the log transformation
> exp(predict(nihills.lm, interval="confidence"))[1:5, ]
                      fit   lwr   upr
Binevenagh          0.893 0.845 0.944
Slieve Gullion      0.488 0.467 0.510
Glenariff Mountain  0.640 0.604 0.679
Donard & Commedagh  1.126 1.068 1.187
McVeigh Classic     0.570 0.544 0.597
```

Prediction intervals from the model that replaces `logGrad` by `logclimb` will be identical.

Figure 6.11 shows these prediction intervals graphically. Panel A plots the points on a scale of `time`, while panel B plots the points on a scale of `log(time)`. The 95% pointwise intervals apply to the fitted values, that is, to the values on the vertical axis. Fitted values and 95% confidence bounds are shown both for the model that is linear in `log(dist)` and for the model that is quadratic in `log(dist)`.[8]

Notice that the bounds for the quadratic model are narrower within the main body of the data, but fan out as time increases. The quadratic bounds better indicate the uncertainty in the predictions for large times. Note the trade-off between bias for the model that lacks the quadratic term, and increased variance for the model that includes the quadratic term.

[8]
```
## Model that is quadratic in logdist
nihills2.lm <- lm(logtime ~ poly(logdist,2) + logclimb, data = lognihills)
citimes2 <- exp(predict(nihills2.lm, interval="confidence"))
```

This assessment of predictive accuracy has important limitations:

1. It is crucially dependent on the model assumptions – independence of the data points, homogeneity of variance, and normality. If the theoretical assumptions are wrong, perhaps because of clustering or other forms of dependence, then these prediction intervals will also be wrong. Departures from normality are commonly of less consequence than the other assumptions.
2. It applies only to the population from which these data have been sampled. If the sample for which predictions are made is really from a different population, or is drawn with a different sampling bias, the assessments of predictive accuracy will be wrong. Thus it might be hazardous to use the above model to predict winning times for hill races in England or Mexico or Tasmania.

Point 2 can be addressed only by testing the model against data from these other locations. Thus, it is interesting to compare the results from the Scottish `hills2000` data with results from the Northern Irish `nihills` data. The results are broadly comparable. We leave further investigation of this comparison to the exercises.

Better still, where this is possible, is to use data from multiple locations to develop a model that allows for variation between locations as well as variation within locations. Chapter 10 discusses models that are, in principle, suitable for such applications.

We might consider cross-validation, or the bootstrap, or another resampling method. Cross-validation and other resampling methods can cope, to a limited extent and depending on how they are used, with lack of independence. Note that neither this nor any other such internal method can address sampling bias. Heterogeneity is just as much an issue as for model-based assessments of accuracy. Thus for the hill race data, if there is no adjustment for changing variability with increasing length of race:

1. If we use the untransformed data, cross-validation will exaggerate the accuracy for long races and be slightly too pessimistic for short races.
2. If we use the log transformed data, cross-validation will exaggerate the accuracy for short races and under-rate the accuracy for long races.

6.4 A strategy for fitting multiple regression models

Careful graphical scrutiny of the explanatory variables is an essential first step. This may lead to any or all of:

- Transformation of some or all variables.
- Replacement of existing variables by newly constructed variables that are a better summary of the information. For example, we might want to replace variables x_1 and x_2 by the new variables $x_1 + x_2$ and $x_1 - x_2$.
- Omission of some variables.

Why are linear relationships between explanatory variables preferable?

Where there are many explanatory variables, the class of models that would result from allowing all possible transformations of explanatory variables is too wide to be a satisfactory starting point for investigation. The following are reasons for restricting attention to

transformations, where available, that lead to scatterplots in which relationships between explanatory variables are approximately linear.

- If relationships between explanatory variables are non-linear, diagnostic plots may be misleading. See Cook and Weisberg (1999).
- Approximately linear relationships ensure that all explanatory variables have similar distributions, preferably distributions that are not asymmetric to an extent that gives the smallest or largest points undue leverage.
- If relationships are linear, it is useful to check the plots of explanatory variables against the response variable for indications of the relationship with the dependent variable. Contrary to what might be expected, this is more helpful than looking at the plots of the response variable against explanatory variables.

Surprisingly often, logarithmic or other standard forms of transformation give more symmetric distributions, lead to scatterplots where the relationships appear more nearly linear, and make it straightforward to identify a regression equation that has good predictive power.

6.4.1 Suggested steps

Here are steps that are reasonable to follow. They involve examination of the distributions of values of explanatory variables, and of the pairwise scatterplots.

- Examine the distribution of each of the explanatory variables, and of the dependent variable. Look for any instances where distributions are highly skew, or where there are outlying values. Check whether any outlying values may be mistakes.
- Examine the scatterplot matrix involving all the explanatory variables. (Including the dependent variable is, at this point, optional.) Look first for evidence of non-linearity in the plots of explanatory variables against each other. Look for values that appear as outliers in any of the pairwise scatterplots.
- Note the ranges of each of the explanatory variables. Do they vary sufficiently to affect values of the dependent variable?
- How accurately are each of the explanatory variables measured? At worst, the inaccuracy may be so serious that it is unlikely that any effect can be detected, and/or that coefficients of other explanatory variables are seriously in error. Section 6.7 has further details.
- If some pairwise plots show evidence of non-linearity, consider use of transformation(s) to give more nearly linear relationships.
- Where the distribution is skew, consider transformations that may lead to a more symmetric distribution.
- Look for pairs of explanatory variables that are so highly correlated that they appear to give the same information. Do scientific considerations help in judging whether both variables should be retained? For example, the two members of the pair may measure what is essentially the same quantity. Note however that there will be instances where the difference, although small, is important. Section 8.2 has an example.

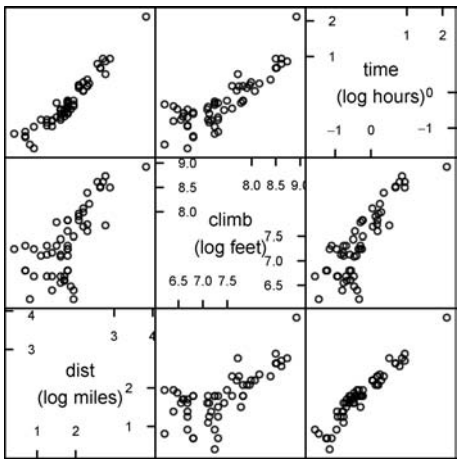


Figure 6.12 Scatterplot matrix for the `hills2000` data (Table 6.1).

6.4.2 *Diagnostic checks*

Checks should include:

- Plot residuals against fitted values. For initial checks, consider the use of residuals from a resistant regression model. Check for patterns in the residuals, and for the fanning out (or in) of residuals as the fitted values change. (Do not plot residuals against observed values. This is potentially deceptive; there is an inevitable positive correlation.)
- Examine the Cook's distance statistics. If it seems helpful, examine standardized versions of the drop-1 coefficients directly, using `dfbetas()`. It may be necessary to delete influential data points and refit the model.
- For each explanatory variable, construct a component plus residual plot, to check whether any of the explanatory variables require transformation.

A further question is whether multiple regression methodology is adequate, or whether a non-linear form of equation may be required. Ideas of "structural dimension", which are beyond the scope of the present text, become important. The package *dr* (dimension reduction) addresses such issues.

6.4.3 *An example – Scottish hill race data*

The data in `hills2000` is comparable to `nihills`, but for Scotland rather than Northern Ireland. Again, we will work on the logarithmic scale and limit attention to the male results. Figure 6.12 shows the scatterplot matrix.[9] Apart from a possible outlier, the relationship between `dist` and `climb` seems approximately linear on the log scale.

[9]
```
## Scatterplot matrix: data frame hills2000 (DAAG), log scales
splom(~ log(hills2000[, c("dist","climb","time")]), cex.labels=1.2,
      varnames=c("dist\n(log miles)", "climb\n(log feet)", "time\n(log hours)"))
```

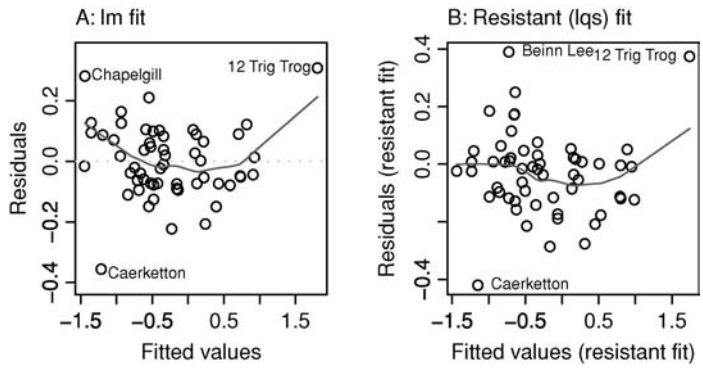


Figure 6.13 Plots of residuals against fitted values from the regression of `log(time)` on `log(climb)` and `log(dist)`. Panel A is from the least squares (`lm`) fit, while panel B is for a resistant fit that uses `lqs` from the *MASS* package. Note that the resistant fit relies on repeated sampling of the data, and will differ slightly from one run to the next.

The help page for `races2000` (`hills2000` is a subset of `races2000`) suggests uncertainty about the distance for the Caerketton race in row 42. We will include Caerketton during our initial analysis and check whether it appears to be an outlier.

Figure 6.13 shows residuals (A) from a least squares (`lm`) fit and (B) from a resistant `lqs` fit, in both cases plotted against fitted values. Resistant fits completely ignore the effects of large residuals. By default, even if almost half the observations are outliers, the effect on the fitted model would be small. See the help page for `lqs` for more information. The code is:

```
## Panel A
lhills2k.lm <- lm(log(time) ~ log(climb) + log(dist),
                  data = hills2000)
plot(lhills2k.lm, caption="", which=1)
## Panel B
library(MASS)         # lqs() is in the MASS package
lhills2k.lqs <- lqs(log(time) ~ log(climb) + log(dist),
                    data = hills2000)
reres <- residuals(lhills2k.lqs)
refit <- fitted(lhills2k.lqs)
big3 <- which(abs(reres) >= sort(abs(reres), decreasing=TRUE)[3])
plot(reres ~ refit, xlab="Fitted values (resistant fit)",
     ylab="Residuals (resistant fit)")
lines(lowess(reres ~ refit), col=2)
text(reres[big3] ~ refit[big3], labels=rownames(hills2000)[big3],
     pos=4-2*(refit[big3] > mean(refit)), cex=0.8)
```

Caerketton shows up rather clearly as an outlier, in both panels. Its residual in panel 1 is -0.356 (visual inspection might suggest -0.35). The predicted log(time) for this race is conveniently written as $\log(\widehat{\text{time}})$. Then

$$\log(\text{time}) - \log(\widehat{\text{time}}) = -0.356.$$

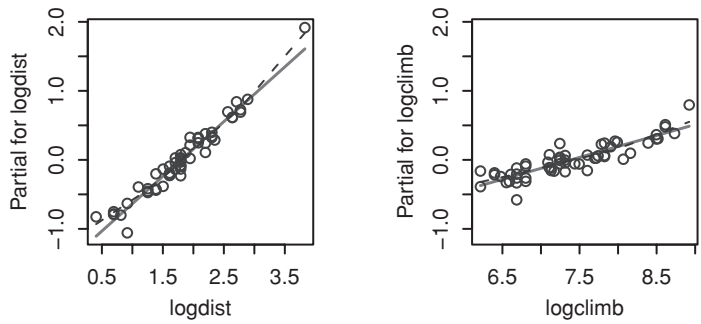


Figure 6.14 The solid lines show the respective contributions of the two model terms, in the regression of `logtime` on `logdist` and `logclimb`. Partial residuals, and an associated smooth curve, have been added.

Thus

$$\log(\frac{\text{time}}{\widehat{\text{time}}}) = -0.356, \text{ i.e., } \frac{\text{time}}{\widehat{\text{time}}} \simeq \exp -0.356 = 0.7.$$

Thus the time given for this race is 70% of that predicted by the regression equation, a very large difference indeed. The standardized difference is -3; this can be seen by use of

```
plot(lhills2k.lm, which=2)
```

If the model is correct and residuals are approximately normally distributed, a residual of this magnitude will occur about 2.6 times in 1000 residuals.[10]

The resistant fit in panel B suggests that Beinn Lee and 12 Trig Trog are also outliers. These outliers may be a result of non-linearity. We next use `termplot()` to check whether `log(dist)` and/or `log(climb)` should enter the model non-linearly.

The contribution of the separate terms

In order to get a plot that is easy to interpret, it is best to transform the variables before entering them into the model, thus:

```
## Create data frame that has logs of time, dist & climb
lhills2k <- log(hills2000[, c("dist", "climb", "time")])
names(lhills2k) <- paste("log", names(lhills2k), sep="")
lhills2k.lm <- lm(logtime ~ logdist+logclimb, data=lhills2k)
termplot(lhills2k.lm, partial.resid=TRUE, smooth=panel.smooth,
         col.res="gray30")
```

Figure 6.14 shows the result. There is a small but clear departure from linearity, most evident for `logdist`. The linear model does however give a good general summary of the indications in the data. If the shortest and longest of the races are left out, it appears entirely adequate.

```
> ## Probability of a value <= -3 or >= 3
> 2 * pnorm(-3)
[1] 0.0027
```
[10]

It is necessary to model the departure from linearity before proceeding further with checking residuals. A quadratic term in `logdist` will work well here.

A resistant fit that has a polynomial term in `logdist`

The quadratic term needs to be entered as `I(logdist^2)`; this ensures that `I(logdist^2)` is taken as the square, rather than as an interaction of `logdist` with itself. It is however better, as will be explained in Section 7.4, to combine `logdist` and `I(logdist^2)` together into a single orthogonal polynomial "term". Subsequent use of the `termplot()` function then shows the combined effects from `logdist` and `I(logdist^2)`. Here is the code:

```
> reres2 <- residuals(lqs(logtime ~ poly(logdist,2)+logclimb, data=lhills2k))
> reres2[order(abs(reres2), decreasing=TRUE)[1:4]]
Caerketton  Beinn Lee    Yetholm Ardoch Rig
    -0.417      0.299      0.200      0.188
```

As might have been expected, Caerketton is now the only large residual. In the sequel, we therefore omit Caerketton.

Refining the model

The component plus residual plot suggested taking a quadratic function of distance. Another possibility might be to add the interaction term `logdist:logclimb` or, equivalently (as these are continuous variables), `I(logdist*logclimb)`. The following compares these possibilities, using AIC as a criterion. (Lower AIC is better.)

```
> add1(lhills2k.lm, ~ logdist+I(logdist^2)+logclimb+logdist:logclimb,
+      test="F")
Single term additions

Model:
logtime ~ logdist + logclimb
                 Df Sum of Sq     RSS     AIC F value    Pr(F)
<none>                         0.77 -233.89
I(logdist^2)      1      0.17    0.61 -245.52  14.331 0.000399
logdist:logclimb  1      0.06    0.71 -236.69   4.651 0.035682
```

On its own, the AIC value is pretty meaningless. What matters is the comparison between the values of the statistic for the different models. A smaller AIC is preferred; it indicates that the model has better predictive power. Clearly the quadratic term in `logdist` is doing the job much better.

We now examine the diagnostic plots for the model that includes `poly(logdist,2)`. The code is:

```
lhills2k.lm2 <- lm(logtime ~ poly(logdist,2)+logclimb, data=lhills2k[-42, ])
plot(lhills2k.lm2)
```

Figure 6.15 shows the result.

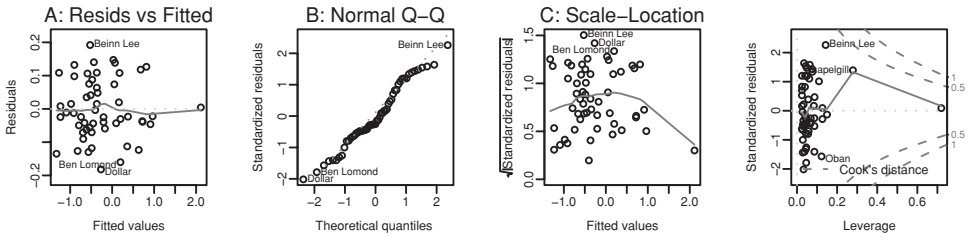


Figure 6.15 Diagnostic plots from the fit of `logtime` to a polynomial of degree 2 in `logdist` and `logclimb`.

Note that `poly(logdist,2)` generates orthogonal polynomial contrasts. To get a result in which the parameters are coefficients of `logdist` and `logdist^2`, repeat the fit specifying the quadratic term as `poly(logdist,2, raw=TRUE)`.

Additionally, the reader may wish to check that adding the interaction term to a model that already has the quadratic term gives no useful improvement.[11] It will be found that use of the interaction term increases the AIC slightly.

The model without the interaction term

As noted earlier, the model that is linear in `logdist` and `logclimb` would be adequate for many practical purposes. The coefficients are:

```
> lhills2k.lm <- lm(logtime ~ logdist+logclimb, data=lhills2k[-42, ])
> summary(lhills2k.lm)$coef
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -4.023     0.1865   -21.6 1.33e-27
logdist         0.778     0.0335    23.2 3.89e-29
logclimb        0.317     0.0302    10.5 1.90e-14
```

The estimated equation is:

$$\log(\texttt{time}) = -4.02 + 0.78\log(\texttt{dist}) + 0.32\log(\texttt{climb}).$$

By noting that $\exp(-4.02) = 0.018$, this can be rewritten as

$$\texttt{time} = 0.018 \times \texttt{dist}^{0.78} \times \texttt{climb}^{0.32}.$$

Are "errors in x" an issue?

Most distances are given to the nearest half mile, and may in any case not be known at all accurately. The error is, however, likely to be small relative to the range of values of distances, so that the attenuation effects that will be discussed in Section 6.7 are likely to be small and of little consequence. See Section 6.7 for the theory that is relevant to the assessment of likely attenuation effects.

[11]
```
## Check addition of interaction term
add1(lhills2k.lm2, ~ poly(logdist,2)+logclimb+logdist:logclimb)
```

What happens if we do not transform?

If we avoid transformation and do not allow for increasing variability for the longer races (see Exercise 6 at the end of the chapter for further development), we find several outlying observations, with the race that has the longest time highly influential.

Venables and Ripley (2002, p. 154) point out that it is reasonable to expect that variances will be larger for longer races. Using `dist` as a surrogate for time, they give observations weights of `1/dist^2`. This is roughly equivalent, in its effect on the variance, to our use of `log(time)` as the dependent variable.

6.5 Problems with many explanatory variables

Variable selection is an issue when the aim is to obtain the best prediction possible. Be sure to distinguish the variable selection problem from that of determining which variables have greatest explanatory power. If the interest is in which variables have useful explanatory power, then the choice of variables will depend on which quantities are to be held constant when the effects of other quantities are estimated. There should in any case be an initial exploratory investigation of explanatory variables, as described in Section 6.4, leading perhaps to transformation of one or more of the variables.

One suggested rule is that there should be at least 10 times as many observations as variables, before any use of variable selection takes place. For any qualitative factor, subtract one from the number of levels, and count this as the number of variables contributed by that factor. This may be a reasonable working rule when working with relatively noisy data where none of the variables have a dominant effect. There are important contexts where it is clearly inapplicable.

For an extended discussion of state-of-the-art approaches to variable selection, we refer the reader to Harrell (2001), Hastie *et al.* (2009, Sections 4.3 and 4.7). The technically demanding paper by Rao and Wu (2001) was, at the time of publication, a good summary of the literature on model selection.

We begin by noting strategies that are designed, broadly, to keep to a minimum the number of different models that will be compared. The following strategies may be used individually, or in combination.

1. A first step may be an informed guess as to what variables/factors are likely to be important. An extension of this approach classifies explanatory variables into a small number of groups according to an assessment of scientific "importance". Fit the most important variables first, then add the next set of variables as a group, checking whether the fit improves from the inclusion of all variables in the new group.
2. Interaction effects are sometimes modeled by including pairwise multiples of explanatory variables, e.g., $x_1 \times x_2$ as well as x_1 and x_2. Use is made of an omnibus check for all interaction terms, rather than checking for interaction effects one at a time.
3. Principal components analysis is one of several methods that may be able to identify a small number of components, i.e., combinations of the explanatory variables, that together account for most of the variation in the explanatory variables. In favorable circumstances, one or more of the first few principal components will prove to be useful explanatory variables, or may suggest useful simple forms of summary of the

original variables. In unfavorable circumstances, the components will prove irrelevant! See Section 13.1 and Harrell (2001, Sections 4.7 and 8.6) for further commentary and examples. See Chapter 13 for examples.

4. Discriminant analysis can sometimes be used to identify a summary variable. There is an example in Chapter 13.

6.5.1 Variable selection issues

We caution against giving much credence to output from conventional automatic variable selection techniques – various forms of stepwise regression, and best subsets regression. The resulting regression equation may have poorer genuine predictive power than the regression that includes all explanatory variables. The standard errors and t-statistics typically ignore the effects of the selection process; estimates of standard errors, p-values, and F-statistics will be optimistic. Estimates of regression coefficients are biased upwards in absolute value – positive coefficients will be larger than they should be, and negative coefficients will be smaller than they should be. See Harrell (2001) for further discussion.

Variable selection – a simulation with random data

Repeated simulation of a regression problem where the data consist entirely of noise will demonstrate the extent of the problem. In each regression there are 41 vectors of 100 numbers that have been generated independently and at random from a normal distribution. In these data:[12]

1. The first vector is the response variable y.
2. The remaining 40 vectors are the variables $x_1, x_2, \ldots, x_{40}$.

If we find any regression relationships in these data, this will indicate faults with our methodology. (In computer simulation, we should not however totally discount the possibility that a quirk of the random number generator will affect results. We do not consider this an issue for the present simulation!)

We perform a best subsets regression in which we look for the three x-variables that best explain y. (This subsection has an example that requires access to the *leaps* package, implemented by Thomas Lumley using Fortran code by Alan Miller. Before running the code, be sure that *leaps* is installed.)[13]

```
Call:
lm(formula = y ~ -1 + xx[, subvar])
. . . .
```

[12]
```
## Generate a 100 by 40 matrix of random normal data
y <- rnorm(100)
xx <- matrix(rnorm(4000), ncol = 40)
dimnames(xx)<- list(NULL, paste("X",1:40, sep=""))
```

[13]
```
## Find the best fitting model
library(leaps)
xx.subsets <- regsubsets(xx, y, method = "exhaustive", nvmax = 3, nbest = 1)
subvar <- summary(xx.subsets)$which[3,-1]
best3.lm <- lm(y ~ -1+xx[, subvar])
print(summary(best3.lm, corr = FALSE))
## The following call to bestsetNoise() (from DAAG) achieves the same purpose
bestsetNoise(m=100, n=40)
```

```
Coefficients:
                 Estimate Std. Error t  value Pr(>|t|)
xx[, subvar]X12   0.2204     0.0896     2.46   0.0156
xx[, subvar]X23  -0.1431     0.0750    -1.91   0.0593
xx[, subvar]X28   0.2529     0.0927     2.73   0.0076

Residual standard error: 0.892 on 97 degrees of freedom
Multiple R-Squared: 0.132,      Adjusted R-squared: 0.105
F-statistic: 4.93 on 3 and 97 DF,  p-value: 0.00314
```

Note that two of the three variables selected have p-values less than 0.05.

When we repeated this experiment 10 times, the outcomes were as follows. Categories are exclusive:

	Instances
All three variables selected were significant at $p < 0.01$	1
All three variables had $p < 0.05$	3
Two out of three variables had $p < 0.05$	3
One variable with $p < 0.05$	3
Total	10

In the modeling process there are two steps:

1. Select variables.
2. Do a regression and determine SEs and p-values, etc.

The p-value calculations have taken no account of step 1. Our ability to find "significance" in data sets that consist only of noise is evidence of a large bias.

Cross-validation that accounts for the variable selection process

Cross-validation is one way to determine realistic standard errors and p-values. At each cross-validation step, we repeat both of steps 1 and 2 above, i.e., both the variable selection step and the comparison of predictions from the regression equation with data different from that used in forming the regression equation.

Estimates of regression coefficients and standard errors should similarly be based on fitting regressions, at each fold, to the test data for that fold. The issue here is that the equation must be fitted to data that were not used for variable selection. The estimates from the separate folds must then be combined.

Regression on principal components

Regression on principal components, which we discussed briefly in the preamble to this section and will demonstrate in Section 13.1, may sometimes be a useful recourse. Hastie *et al.* (2009) discuss principal components regression alongside a number of other methods that are available. Note especially the "shrinkage" methods, which directly shrink the

coefficients. The function `lars()`, in the package with the same name, implements a class of methods of this type that has the name least angle regression, allowing also for variable selection. See Efron *et al.* (2003).

6.6 Multicollinearity

Some explanatory variables may be linearly related to combinations of one or more of the other explanatory variables. Technically, this is known as multicollinearity. For each multicollinear relationship, there is one redundant variable.

The approaches that we have advocated – careful thinking about the background science, careful initial scrutiny of the data, and removal of variables whose effect is already accounted for by other variables – will generally avoid the more extreme effects of multicollinearity that we will illustrate. Milder consequences are pervasive, especially for observational data.

An example – compositional data

The data set `Coxite`, in the `compositions` package, has the mineral compositions of 25 rock specimens of coxite type. Each composition consists of the percentage by weight of five minerals, the depth of location, and porosity. The names of the minerals are abbreviated to `A` = albite, `B` = blandite, `C` = cornite, `D` = daubite, and `E` = endite. The analysis that follows is a relatively crude use of these data. For an analysis that uses a method that is designed for compositional data, see Aitchison (2003).

Figure 6.16 shows the scatterplot matrix. Notice that the relationship between `D` and `E` is close to linear. Also, the percentages of the five minerals sum, in each row, to 100. The code for Figure 6.16 is:

```
library(compositions)
data(Coxite)          # For pkg compositions, needed to access data
coxite <- as.data.frame(Coxite)   # From matrix, create data frame
## Scatterplot matrix for data frame coxite
pairs(coxite)
```

We will look for a model that explains `porosity` as a function of mineral composition. Here is a model that tries to use all six explanatory variables:

```
> coxiteAll.lm <- lm(porosity ~ A+B+C+D+E+depth, data=coxite)
> summary(coxiteAll.lm)
. . . .

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -217.7466   253.4439   -0.86     0.40
A              2.6486     2.4825    1.07     0.30
B              2.1915     2.6015    0.84     0.41
C              0.2113     2.2271    0.09     0.93
D              4.9492     4.6720    1.06     0.30
E                  NA         NA      NA       NA
depth          0.0145     0.0333    0.44     0.67
```

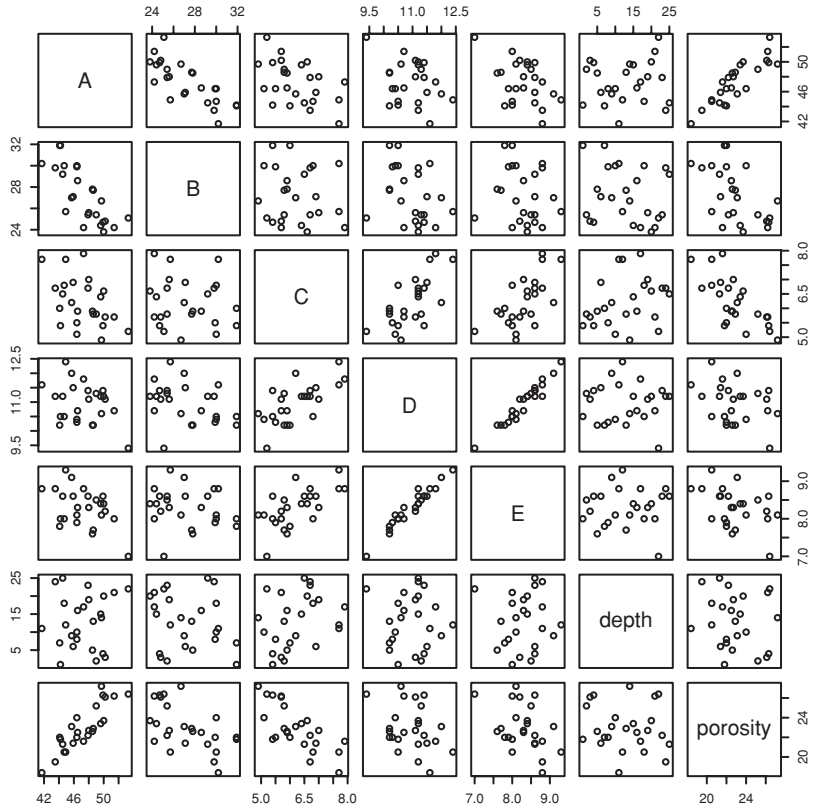


Figure 6.16 Scatterplot matrix for the variables in the constructed `Coxite` data set.

```
Residual standard error: 0.649 on 19 degrees of freedom
Multiple R-squared: 0.936,        Adjusted R-squared: 0.919
F-statistic: 55.1 on 5 and 19 DF,  p-value: 1.18e-10
```

Notice that:

- The variable `E`, because it is a linear combination of earlier variables, adds no information additional to those variables. Effectively, its coefficient has been set to zero.
- None of the individual coefficients comes anywhere near the usual standards of statistical significance.
- The overall regression fit, with a p-value of 1.18×10^{-10}, is highly significant.

The overall regression fit has good predictive power, notwithstanding the inability to tease out the contributions of the individual coefficients. Figure 6.17 shows 95% pointwise confidence intervals for fitted values at several points within the range. Pointwise confidence bounds can be obtained thus:

```
hat <- predict(coxiteAll.lm, interval="confidence", level=0.95)
```

The object that is returned is a matrix, with columns `fit` (fitted values), `lwr` (lower confidence limits), and `upr` (upper confidence limits).

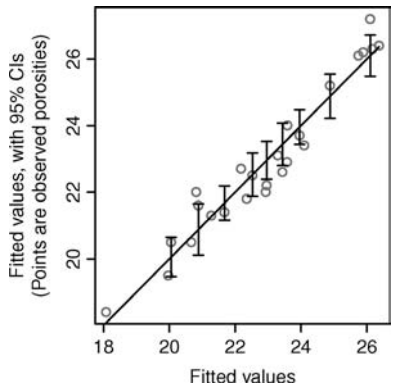


Figure 6.17 Line $y = x$, with 95% pointwise confidence bounds for fitted values shown at several locations along the range of fitted values. The points show the observed porosities at each of the fitted values.

6.6.1 The variance inflation factor

The variance inflation factor (VIF) measures the effect of correlation with other variables in increasing the standard error of a regression coefficient. If x_j, with values x_{ij} $(i = 1, \ldots, n)$ is the only variable in a straight line regression model, and b_j is the estimated coefficient, then:

$$\text{var}[b_j] = \frac{\sigma^2}{s_{jj}}, \quad \text{where } s_{jj} = \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2$$

and σ^2 is the variance of the error term in the model. When further terms are included in the regression model, this variance is inflated, as a multiple of σ^2, by the variance inflation factor. Notice that the VIF depends only on the model matrix. It does not reflect changes in the residual variance.

In order to obtain VIFs, we need to explicitly omit `E` (or one of `A`, `B`, `C`, or `D`) from the model, thus:

```
> vif(lm(porosity ~ A+B+C+D+depth, data=coxite))
      A       B       C       D   depth
2717.82 2484.98  192.59  566.14    3.42
```

Given the size of these factors, it is unsurprising that none of the individual coefficients can be estimated meaningfully.

It is reasonable to try a model that uses those variables that, individually, correlate most strongly with porosity. Here are the correlations:

```
> cor(coxite$porosity, coxite)
         A      B      C     D      E  depth porosity
[1,] 0.869 -0.551 -0.723 -0.32 -0.408 -0.147        1
```

Thus we try:

```
> summary(coxiteABC.lm <- lm(porosity ~ A+B+C, data=coxite))
. . . .
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  53.3046    13.2219    4.03  0.00060
A            -0.0125     0.1558   -0.08  0.93700
B            -0.5867     0.1513   -3.88  0.00087
C            -2.2188     0.3389   -6.55  1.7e-06
. . . .
> vif(coxiteABC.lm)
    A     B     C
10.94  8.59  4.56
```

It is then reasonable to simplify this to:

```
> summary(coxiteBC.lm <- lm(porosity ~ B+C, data=coxite))
. . . .
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  52.2571     1.7831    29.3  < 2e-16
B            -0.5753     0.0508   -11.3  1.2e-10
C            -2.1949     0.1562   -14.1  1.8e-12
> vif(coxiteBC.lm)
   B    C
1.01 1.01
```

Using the AIC statistic to compare this model with the model that used all six explanatory variables, we have:

```
> AIC(coxiteAll.lm, coxiteBC.lm)
             df  AIC
coxiteAll.lm  7 56.5
coxiteBC.lm   4 52.5
```

The simpler model wins. Predictions from this simpler model should have slightly narrower confidence bounds than those shown in Figure 6.17. Verification of this is left as an exercise.

Numbers that do not quite add up

Now round all the percentages to whole numbers, and repeat the analysis that uses all six available explanatory variables.

```
> coxiteR <- coxite
> coxiteR[, 1:5] <- round(coxiteR[, 1:5])
> summary(lm(porosity ~ ., data=coxiteR))
. . . .
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.4251    23.4043   -0.06  0.95212
A             0.5597     0.2457    2.28  0.03515
B             0.0157     0.2403    0.07  0.94865
C            -1.3180     0.2940   -4.48  0.00029
```

```
D             0.9748     0.4222    2.31  0.03305
E            -0.5530     0.5242   -1.05  0.30543
depth        -0.0149     0.0223   -0.67  0.51310

Residual standard error: 0.708 on 18 degrees of freedom
Multiple R-squared: 0.927,        Adjusted R-squared: 0.903
F-statistic: 38.3 on 6 and 18 DF,  p-value: 2.69e-09

> vif(lm(porosity ~ .-E, data=coxiteR))
    A     B     C     D depth
16.96 16.49  3.28  4.66  1.25
```

This result may seem surprising. Noise has been introduced that has removed some of the correlation, so that C now appears significant even when all other explanatory variables are included. Perhaps more surprising is the $p = 0.033$ for D.

While this is contrived, we have from time to time seen comparable effects in computer output that researchers have brought us for scrutiny.

6.6.2 *Remedies for multicollinearity*

As noted at the beginning of the section, careful initial choice of variables, based on scientific knowledge and careful scrutiny of relevant exploratory plots of explanatory variables, will often avert the problem. Occasionally, it may be possible to find or collect additional data that will reduce correlations among the explanatory variables.

Ridge regression is one of several approaches that may be used to alleviate the effects of multicollinearity, in inflating coefficients and their standard errors. We refer the reader to the help page for the function `lm.ridge()` in the *MASS* package and to the discussions in Berk (2008), Myers (1990), Harrell (2001). For a less elementary and more comprehensive account, see Hastie *et al.* (2009). Use of the function `lm.ridge()` requires the user to choose the tuning parameter `lambda`. Typically, the analyst tries several settings and chooses the value that, according to one of several available criteria, is optimal. Principal components regression is another possible remedy.

6.7 Errors in *x*

The issues that are canvassed here have large practical importance. It can be difficult to assess the implications of the theoretical results for particular practical circumstances.

The discussion so far has been assumed: either that the explanatory variables are measured with negligible error, or that the interest is in the regression relationship given the observed values of explanatory variables. This subsection draws attention to the effect that errors in the explanatory variables can have on regression slope. Discussion is mainly on the relatively simple "classical" errors in x model.

With a single explanatory variable, the effect under the classical "errors in x" model is to reduce the expected magnitude of the slope, that is, the slope is attenuated. Furthermore, the estimated slope is less likely to be distinguishable from statistical noise. For estimating the magnitude of the error and consequent attenuation of the slope, there must be information

additional to that shown in a scatterplot of y versus x. There must be a direct comparison with values that are measured with negligible error.

The study of the measurement of dietary intake that is reported in Schatzkin *et al.* (2003) illustrates some of the key points. The error in the explanatory variable, as commonly measured, was shown to be larger and of greater consequence than most researchers had been willing to contemplate.

Measurement of dietary intake

The 36-page Diet History Questionnaire is a Food Frequency Questionnaire (FFQ) that was developed and evaluated at the US National Cancer Institute. In large-scale trials that look for dietary effects on cancer and on other diseases, it has been important to have an instrument for measuring food intake that is relatively cheap and convenient. (Some trials have cost US$100 000 000 or more.)

The FFQ asks for details of food intake over the previous year for 124 food items. It queries frequency of intake and, for most items, portion sizes. Supplementary questions query such matters as seasonal intake and food type. More detailed food records may be collected at specific times, which can then be used to calibrate the FFQ results. One such instrument is a 24-hour dietary recall, which questions participants on their dietary intake in the previous 24 hours.

Schatzkin *et al.* (2003) compared FFQ measurements with those from Doubly Labeled Water, used as an accurate but highly expensive biomarker. They conclude that the FFQ is too inaccurate for its intended purpose. The 24-hour dietary recall, although better, was still seriously inaccurate.

In some instances, the standard deviation for estimated energy intake was seven times the standard deviation, between different individuals, of the reference. There was a bias in the relationship between FFQ and reference that further reduced the attenuation factor, to 0.04 for women and to 0.08 for men. For the relationship between the 24-hour recalls and the reference, the attenuation factors were 0.1 for women and 0.18 for men, though these can be improved by use of repeated 24-hour recalls.

These results raise serious questions about what such studies can achieve, using presently available instruments that are sufficiently cheap and convenient that they can be used in large studies. Carroll (2004) gives an accessible summary of the issues. Subsection 10.7.6 has further brief comment on the modeling issues.

Simulations of the effect of measurement error

Suppose that the underlying regression relationship that is of interest is:

$$y_i = \alpha + \beta x_i + \varepsilon_i, \text{ where } \mathrm{var}[\varepsilon_i] = \sigma^2 \ (i = 1, \ldots, n).$$

Let $s_x = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)}$ be the standard deviation of the values that are measured without error.

Take the measured values as

$$w_i = x_i + \eta_i, \text{ where } \mathrm{var}[\eta_i] = s_x^2\,\tau^2.$$

The η_i are assumed independent of the ε_i.

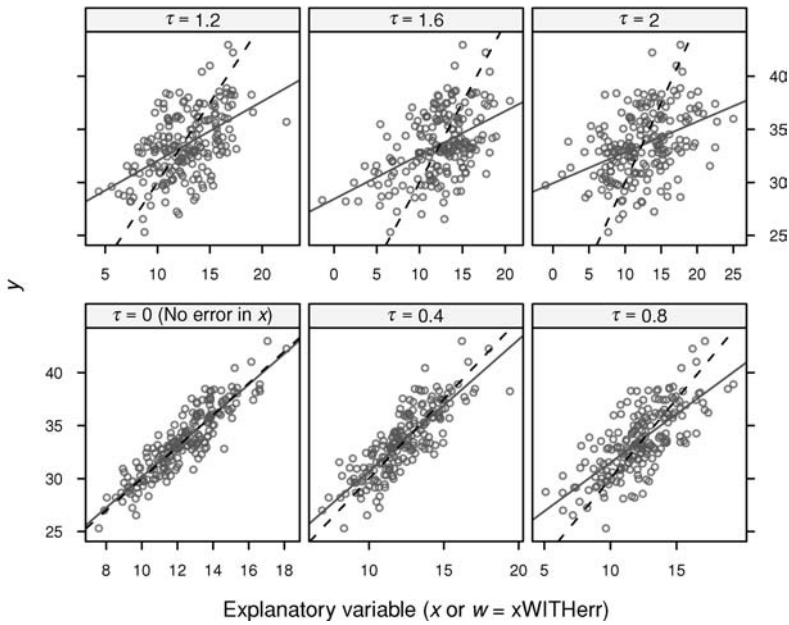


Figure 6.18 The fitted solid lines show the change in the regression line as the error in x changes. The underlying relationship, shown with the dashed line, is in each instance $y = 15 + 1.5x$. For the definition of τ, see the text. This figure was obtained by use of the *DAAG* function `errorsINx()`, with default arguments.

Figure 6.18 shows results from a number of simulations that use the w_i as the explanatory values. If $\tau = 0.4$ (the added error has a variance that is 40% of s_x), the effect on the slope is modest. If $\tau = 2$, the attenuation is severe. The function `errorsINx()` (*DAAG*) can be used for additional simulations such as are shown in Figure 6.18.

An estimate of the attenuation in the slope is, to a close approximation:

$$\lambda = \frac{1}{1 + \tau^2}.$$

Here, λ has the name *reliability ratio*.

If, for example, $\tau = 0.4$, then $\lambda \simeq 0.86$. Whether a reduction in the estimated slope by a factor of 0.86 is of consequence will depend on the application. Often there will be more important concerns. Very small attenuation factors (large attenuations), e.g., less than 0.1 such as were found in the Schatzkin *et al.* (2003) study, are likely to have serious consequences for the use of analysis results.

Points to note are:

- From the data used in the panels of Figure 6.18, it is impossible to estimate τ, or to know the underlying x_i values. This can be determined only from an investigation that compares the w_i with an accurate, i.e., for all practical purposes error-free, determination of the x_i.
- A test for $\beta = 0$ can be undertaken in the usual way, but with reduced power to detect an effect that may be of interest.

- The t-statistic for testing $\beta = 0$ is affected in two ways; the numerator is reduced by an expected factor of λ, while the standard error that appears in the numerator increases. Thus if $\lambda = 0.1$, the sample size required to detect a non-zero slope is inflated by more than the factor of 100 that is suggested by the effect on the slope alone.

*Two explanatory variables

Consider first the case where one predictor is measured with error, and others without error. The coefficient of the variable that is measured with error is attenuated, as in the single variable case. The coefficients of other variables may be reversed in sign, or show an effect when there is none. See Carroll *et al.* (2006, pp. 52–55) for summary comment.

Suppose that

$$y = \beta_1 x_1 + \beta_2 x_2 + \varepsilon.$$

If w_1 is unbiased for x_1 and the measurement error η is independent of x_1 and x_2, then least squares regression with explanatory variables w_1 and x_2 yields an estimate of $\lambda\beta_1$, where if ρ is the correlation between x_1 and x_2:

$$\lambda = \frac{1 - \rho^2}{1 - \rho^2 + \tau^2}.$$

A new feature is the bias in the least squares estimate of β_2. The naive least squares estimator estimates

$$\beta_2 + \beta_1(1 - \lambda)\,\gamma_{12}, \text{ where } \gamma_{12} = \rho\frac{s_1}{s_2}. \tag{6.4}$$

Here, γ_{12} is the coefficient of x_2 in the least squares regression of x_1 on x_2, $s_1 = \text{SD}[x_1]$ and $s_2 = \text{SD}[x_2]$. The estimate of β_2 may be substantially different from zero, even though $\beta_2 = 0$. Where $\beta_2 \neq 0$, the least squares estimate can be reversed in sign from β_2. Some of the effect of x_1 is transferred to the estimate of the effect of x_2.

Two explanatory variables, one measured without error – a simulation

The function `errorsINx()` (*DAAG*), when supplied with a non-zero value for the argument `gpdiff`, simulates the effect when the variable that is measured without error codes for a categorical effect. Figure 6.19 had `gpdiff=1.5`. Two lines appear, suggesting a "treatment" effect where there was none.

The function `errorsINseveral()` simulates a model where there are two continuous variables x_1 and x_2. The default choice of arguments has

$$\beta_1 = 1.5, \beta_2 = 0, \rho = -0.5, s_1 = s_2 = 2, \tau = 1.5, \text{var}[\varepsilon] = 0.25.$$

Measurement error variances are x_1: $s_1^2\tau^2$, x_2: 0. Then $\lambda = 0.25$, $\gamma_{12} = -0.5$, and the expected value for the naive least squares estimator of β_2 is

$$\beta_2 + \beta_1(1 - \lambda)\gamma_{12} = 0 + 1.5 \times 0.75 \times (-0.5) = -0.5675.$$

Here is a simulation result, with default arguments as noted:

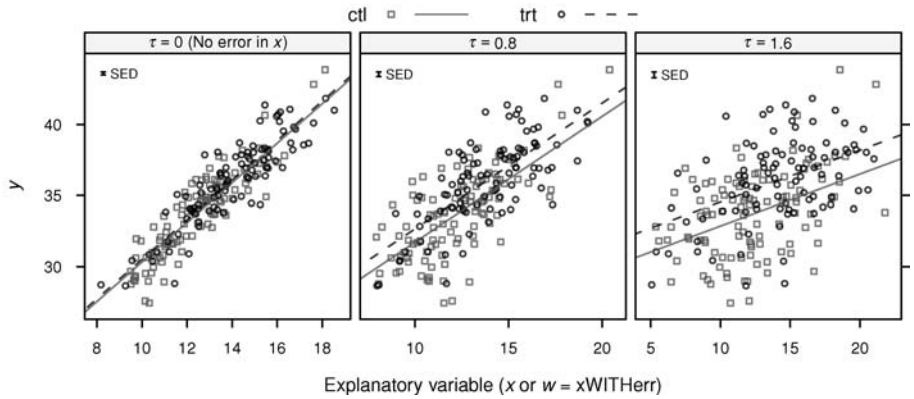


Figure 6.19 In the simulations whose results are shown here, y is a linear function of x. The mean value of x is 12.5 for the first level ("ctl") of a grouping variable, and 14.0 for the second level ("trt") of the grouping variable. In the panel on the left, the values of x are measured without error. In the middle and right panels, independent errors have been added to x from distributions with SDs that are 0.8 and 1.6 times that of the within-group standard deviation of x. The SEDs are conditional on $w =$ xWITHerr.

```
> errorsINseveral()
                            Intercept    b1     b2
Values for simulation           2.482 1.500  0.000
Estimates: no error in x1       2.554 1.494  0.009
LS Estimates: error in x1      35.448 0.400 -0.597
```

An arbitrary number of variables

Where two or more variables are measured with substantial error, there is an increased range of possibilities for transferring some part or parts of effects between variables. By specifying the arguments V and xerrV, the function errorsINseveral() can be used for simulations with an arbitrary correlation structure for the explanatory variables, and with an arbitrary variance–covariance matrix for the added errors.[14]

**The classical error model versus the Berkson error model*

In the classical model, $\mathrm{E}[w_i|x_i] = x_i$. In the Berkson model, $\mathrm{E}[x_i|w_i] = w_i$. This may be a realistic model in an experiment where w_i is an instrument setting, but the true value varies randomly about the instrument setting. For example, the temperature in an oven or kiln may be set to w_i, but the resulting (and unknown) actual temperature is x_i. In straight line regression, the coefficient is then unbiased, but the variance of the estimate of the coefficient increases.

[14] If β is the vector of coefficients in the model without errors in the measured values, V corresponds in the obvious way to V, and U to xerrV, then an estimate for the resulting least squares estimates for regression on the values that are measured with error is $\beta' V(V+U)^{-1}$. Note that Zeger *et al.* (2000, p. 421) have an initial T (our U), where V is required.

Zeger *et al.* (2000) discuss the practical consequences of both types of error, though giving most of their attention to the classical model. In their context, realistic models may have elements of both the classical and Berkson models.

6.8 Multiple regression models – additional points

The following notes should help dispel any residual notion that this chapter's account of multiple regression models has covered everything of importance.

6.8.1 Confusion between explanatory and response variables

As an example, we return to the `allbacks` data. We compare the coefficients in the equation for predicting `area` given `volume` and `weight` with the rearranged coefficients from the equation that predicts `weight` given `volume` and `area`:

```
> coef(lm(area ~ volume + weight, data=allbacks))
(Intercept)      volume      weight
     35.459      -0.964       1.361
> b <- as.vector(coef(lm(weight ~ volume + area, data=allbacks)))
> c("_Intercept_"=-b[1]/b[3], volume=-b[2]/b[3], weight=1/b[3])
_Intercept_      volume      weight
     -47.85       -1.51        2.13
```

Only if the relationship is exact, so that predicted time is the same as observed time, will the equations be exactly the same. For examples from the earth sciences literature, see Williams (1983).

Unintended correlations

Suppose that x_i $(i = 1, 2, \ldots, n)$ are results from a series of controls, while y_i $(i = 1, 2, \ldots, n)$ are results from the corresponding treated group. It is tempting to plot $y - x$ versus x. Unfortunately, there is likely to be a negative correlation between $y - x$ and x, though this is not inevitable. This emphasizes the desirability of maintaining a clear distinction between explanatory and response variables. See the example in Sharp *et al.* (1996).

6.8.2 Missing explanatory variables

Here the issue is use of the wrong model for the expected value. With the right "balance" in the data, the expected values are unbiased or nearly unbiased. Where there is serious imbalance, the bias may be huge.

Figure 6.20 relates to data collected in an experiment on the use of painkillers (Gordon *et al.*, 1995). Pain was measured as a VAS (Visual-Analogue Scale) score. Researchers were investigating differences in the pain score between the two analgesic treatments, without and with baclofen.

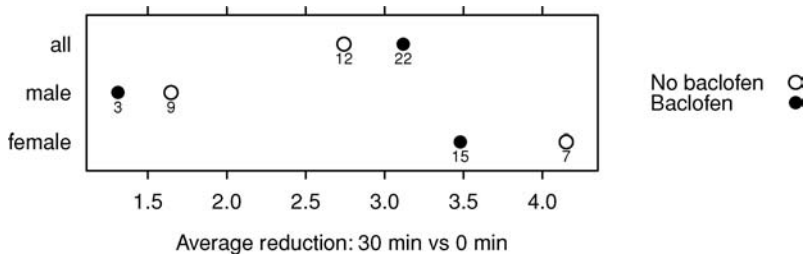


Figure 6.20 Does baclofen, following operation (additional to earlier painkiller), reduce pain? Subgroup numbers, shown below each point in the graph, weight the overall averages when sex is ignored.

Notice that the overall comparison (average for baclofen versus average for no baclofen) goes in a different direction from the comparison for the two sexes separately. As the two treatment groups had very different numbers of men and women, and as there was a strong sex effect, an analysis that does not account for the sex effect gives an incorrect estimate of the treatment effect (Cohen, 1996).

The overall averages in Figure 6.20 reflect the following subgroup weighting effects (f is shorthand for female and m for male):

Baclofen: 15f to 3m, i.e., $\frac{15}{18}$ to $\frac{3}{18}$ (a little less than f average)
No baclofen: 7f to 9m, i.e., $\frac{7}{16}$ to $\frac{9}{16}$ ($\approx \frac{1}{2}$-way between m & f)

There is a sequel. More careful investigation revealed that the response to pain has a different pattern over time. For males, the sensation of pain declined more rapidly over time.

Strategies

(i) Simple approach. Calculate means for each subgroup separately.
Overall treatment effect is average of subgroup differences.
Effect of baclofen (reduction in pain score from time 0) is:

Females: $3.479 - 4.151 = -0.672$ ($-$ve, therefore an increase)
Males: $1.311 - 1.647 = -0.336$
Average over male and female $= -0.5 \times (0.672 + 0.336) = -0.504$

(ii) Fit a model that accounts for sex and baclofen effects. y = overall mean + sex effect + baclofen effect + interaction.
(At this point, we are not including an error term.)

When variables or factors are omitted from models, values of the outcome variable are as far as possible accounted for using those that remain. The mouse brain weight example in Subsection 6.2.3 can be understood in this way. Bland and Altman (2005) give several examples of published results where conclusions have been vitiated by effects of this type.

Another example of this same type, albeit in the context of contingency tables, was discussed in Subsection 3.4.5. The analysis of the UCB admissions data in Section 8.3 formulates the analysis of contingency table data as a regression problem.

6.8.3 The use of transformations*

Often there are scientific reasons for transformations. Thus, suppose we have weights w of individual apples, but the effects under study are more likely to be related to surface area. We should consider using $x = w^{\frac{2}{3}}$ as the explanatory variable. If the interest is in studying relative, rather than absolute, changes, it may be best to work with the logarithms of measurements.

Statisticians use transformations for one or more of the following reasons:

1. To form a straight line or other simple relationship.
2. To ensure that the "scatter" of the data is similar for all categories, i.e., to ensure that the boxplots all have a similar shape. In more technical language, the aim is to achieve homogeneity of variance.
3. To make the distribution of data more symmetric and closer to normal.

If there is a transformation that deals with all these issues at once, we are fortunate. It may greatly simplify the analysis.

A log transformation may both remove an interaction and give more nearly normal data. It may, on the other hand, introduce an interaction where there was none before. Or a transformation may reduce skewness while increasing heterogeneity. The availability of direct methods for fitting special classes of model with non-normal errors, for example the generalized linear models that we will discuss in Chapter 8, has reduced the need for transformations.

6.8.4 Non-linear methods – an alternative to transformation?*

This is a highly important area for which, apart from the present brief discussion, we have not found room in the present book. We will investigate the use of the R `nls()` function (*stats* package) to shed light on the loglinear model that we used for the hill race data in Subsection 6.2.1.

The analysis of Subsection 6.2.1 assumed additive errors on the transformed logarithmic scale. This implies, on the untransformed scale, multiplicative errors. We noted that the assumption of homogeneity of errors was in doubt.

One might alternatively assume that the noise term is additive on the untransformed scale, leading to the non-linear model

$$y = x_1^{\alpha} x_2^{\beta} + \varepsilon$$

where $y =$ `time`, $x_1 =$ `dist`, and $x_2 =$ `climb`.

We will use the `nls()` non-linear least squares function to estimate α and β. The procedure used to solve the resulting non-linear equations is iterative, requiring starting values for α and β. We use the estimates from the earlier loglinear regression as starting

values. Because we could be taking a square or cube of the `climb` term, we prefer to work with the variable `climb.mi` that is obtained by dividing `climb` by 5280, so that numbers are of modest size:

```
nihills$climb.mi <- nihills$climb/5280
nihills.nls0 <- nls(time ~ (dist^alpha)*(climb.mi^beta), start =
                  c(alpha = 0.68, beta = 0.465), data = nihills)
plot(residuals(nihills.nls0) ~ log(predict(nihills.nls0)))
```

Output from the `summary()` function includes the following:

```
Parameters:
      Estimate Std. Error t value Pr(>|t|)
alpha  0.31516    0.00806    39.1   <2e-16
beta   0.81429    0.02949    27.6   <2e-16
```

These parameter estimates differ substantially from those obtained under the assumption of multiplicative errors. This is not an unusual occurrence; the non-linear least squares problem has an error structure that is different from the linearized least-squares problem solved earlier. Residuals suggest a non-linear pattern.

Another possibility, that allows `time` to increase non-linearly with `climb.mi`, is

$$y = \alpha + \beta x_1 + \gamma x_2^{\delta} + \varepsilon.$$

We then fit the model, using an arbitrary starting guess:

```
nihills.nls <- nls(time ~ gamma + delta1*dist^alpha +
                   delta2*climb.mi^beta,
                   start=c(gamma = .045, delta1 = .09, alpha = 1,
                   delta2=.9, beta = 1.65), data=nihills)
plot(residuals(nihills.nls) ~ log(predict(nihills.nls)))
```

The starting values were obtained by fitting an initial model in which `alpha` was constrained to equal 1. The result is:

```
Parameters:
       Estimate Std. Error t value Pr(>|t|)
gamma    0.1522     0.0714    2.13  0.04708
delta1   0.0399     0.0284    1.41  0.17694
alpha    1.3102     0.2709    4.84  0.00013
delta2   0.8657     0.0922    9.39  2.3e-08
beta     1.5066     0.1810    8.32  1.4e-07
       Estimate Std. Error t value Pr(>|t|)
```

There are no obvious outliers in the residual plot. In addition, there is no indication of an increase in the variance as the fitted values increase. Thus, a variance-stabilizing transformation or the use of weighted least squares is unnecessary.

6.9 Recap

A coefficient in a multiple regression equation predicts the effect of a variable when other variables are held constant. Coefficients can thus be different, sometimes dramatically, for a different choice of explanatory variables.

Regression equation predictions are for the data used to derive the equation, and reflect any sampling biases that affect that data. Biases that arise because data were not randomly sampled from the population can lead to predictions that, for the population as a whole, are seriously astray.

Plots that can be useful for checking regression assumptions and/or for checking whether results may be unduly influenced by individual data points include: scatterplot matrices of the variables in the regression equation, plots of residuals against fitted values, partial residual plots such as are provided by the `termplot()` function (these are a better guide than plots of residuals against individual explanatory variables), normal probability plots of residuals, scale–location plots, and Cook's distance and related plots.

Robust methods downweight points that may be outliers. Resistant regression methods are designed to completely remove the contribution of outliers to the regression fit.

6.10 Further reading

Faraway (2004) is a wide-ranging account that covers many of the important practical issues. Harrell (2001) is likewise wide-ranging, with an emphasis on biostatistical applications. Again it has a great deal of useful practical advice. Weisberg (1985) offers a relatively conventional approach. Cook and Weisberg (1999) rely heavily on graphical explorations to uncover regression relationships. Venables and Ripley (2002) is a basic reference for using R for regression calculations. See also Fox (2002). Hastie *et al.* (2009) offer wide-ranging challenges for the reader who would like to explore beyond the forms of analysis that we have described.

On variable selection, which warrants more attention than we have given it, see Bolker (2008), Harrell (2001), Hastie *et al.* (2009), Venables (1998). There is certain to be, in the next several years, substantial enhancement to what R packages offer in this area. Bolker's account extends (p. 215) to approaches that weight and average models.

Rosenbaum (2002) is required reading for anyone who wishes to engage seriously with the analysis of data from observational studies. Results can rarely be interpreted with the same confidence as for a carefully designed experimental study. There are however checks and approaches that, depending on the context, can be helpful in assessing the credence that should be given one or other interpretation of analysis results.

On errors in variables, see Carroll *et al.* (2006). Linear models are a special case of non-linear models.

Several of the studies that are discussed in Leavitt and Dubner (2005), some with major public policy relevance, relied to a greater or lesser extent on regression methods. References in the notes at the end of their book allow interested readers to pursue technical details of the statistical and other methodology. The conflation of multiple sources of insight and evidence is invariably necessary, in such studies, if conclusions are to carry conviction.

Especially hazardous is the use of analyses where there are multiple potential confounding variables, i.e., variables whose effects must be accounted for if coefficients for remaining

variables are to be genuinely suggestive of a causal link. Not only must confounders be included; their effects, including possible interaction effects, must be correctly modeled. Controversy over studies on the health effects of moderate alcohol consumption provide a good example; see, for example, Jackson *et al*. (2005).

For commentary on the use of regression and other models for predictive purposes, see Maindonald (2003).

Structural equation models allow, in addition to explanatory variables and dependent variables, intermediate variables that are dependent with respect to one or more of the explanatory variables, and explanatory with respect to one or more of the dependent variables. Cox and Wermuth (1996) and Edwards (2000) describe approaches that use regression methods to elucidate the relationships. Cox and Wermuth is useful for the large number of examples and for its illuminating comments on practical issues, while Edwards is more up-to-date in its account of the methodology.

Bates and Watts (1988) discuss non-linear models in detail. A more elementary presentation is given in one of the chapters of Myers (1990).

References for further reading

Bates, D. M. and Watts, D. G. 1988. *Non-linear Regression Analysis and Its Applications.*

Bolker, B. M. 2008. *Ecological Models and Data in R.*

Carroll, R. J., Ruppert, D. and Stefanski, L. A. 2006. *Measurement Error in Nonlinear Models: A Modern Perspective*, 2nd edn.

Cook, R. D. and Weisberg, S. 1999. *Applied Regression Including Computing and Graphics.*

Cox, D. R. and Wermuth, N. 1996. *Multivariate Dependencies: Models, Analysis and Interpretation.*

Edwards, D. 2000. *Introduction to Graphical Modelling*, 2nd edn.

Faraway, J. J. 2004. *Linear Models with R.*

Fox, J. 2002. *An R and S-PLUS Companion to Applied Regression*.

Harrell, F. E. 2001. *Regression Modelling Strategies, with Applications to Linear Models, Logistic Regression and Survival Analysis.*

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd edn.

Jackson, R., Broad, J., Connor, J. and Wells, S. 2005. Alcohol and ischaemic heart disease: probably no free lunch. *The Lancet* 366: 1911–12.

Leavitt, S. D. and Dubner, S. J. 2005. *Freakonomics. A Rogue Economist Explores the Hidden Side of Everything.*

Maindonald, J. H. 2003. The role of models in predictive validation. Invited Paper, ISI Meeting, Berlin.

Myers, R. H. 1990. *Classical and Modern Regression with Applications*, 2nd edn.

Rosenbaum, P. R. 2002. *Observational Studies*, 2nd edn.

Venables, W. N. 1998. Exegeses on linear models. Proceedings of the 1998 International S-PLUS User Conference.

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with S*, 4th edn.

Weisberg, S. 1985. *Applied Linear Regression*, 2nd edn.

6.11 Exercises

1. The data set `cities` lists the populations (in thousands) of Canada's largest cities over 1992 to 1996. There is a division between Ontario and the West (the so-called "have" regions) and other regions of the country (the "have-not" regions) that show less rapid growth. To identify the "have" cities we can specify

```
## Set up factor that identifies the 'have' cities
## Data frame cities (DAAG)
cities$have <- factor((cities$REGION=="ON") |
                      (cities$REGION=="WEST"))
```

   Plot the 1996 population against the 1992 population, using different colors to distinguish the two categories of city, both using the raw data and taking logarithms of data values, thus:

```
plot(POP1996 ~ POP1992, data=cities,
     col=as.integer(cities$have))
plot(log(POP1996) ~ log(POP1992), data=cities,
                       col=as.integer(cities$have))
```

   Which of these plots is preferable? Explain.

   Now carry out the regressions

```
cities.lm1 <- lm(POP1996 ~ have+POP1992, data=cities)
cities.lm2 <- lm(log(POP1996) ~ have+log(POP1992),
                 data=cities)
```

   and examine diagnostic plots. Which of these seems preferable? Interpret the results.

2. In the data set `cement` (*MASS* package), examine the dependence of `y` (amount of heat produced) on `x1`, `x2`, `x3` and `x4` (which are proportions of four constituents). Begin by examining the scatterplot matrix. As the explanatory variables are proportions, do they require transformation, perhaps by taking $\log(x/(100-x))$? What alternative strategies might be useful for finding an equation for predicting heat?

3. Use the model that was fitted to the data in `nihills` to give predicted values for the data in `hills2000`. Plot these against predicted values from the model fitted to `hills2000`, and use differences from observed values of `log(time)` to estimate a prediction variance that is relevant when Northern Irish data are used to predict Scottish times. Would you expect this variance to be larger or smaller than the estimated error variance from the `hills2000` model fit? Is this expectation born out?

4. The data frame `hills2000` (*DAAG*) updates the 1984 information in the data set `hills`. Fit regression models, for men and women separately, based on the data in `hills`. Check whether they fit satisfactorily over the whole range of race times. Compare the two equations.

5. Section 6.1 used `lm()` to analyze the allbacks data that are presented in Figure 6.1. Repeat the analysis using (1) the function `rlm()` in the *MASS* package, and (2) the function `lqs()` in the *MASS* package. Compare the two sets of results with the results in Section 6.1.

6. The following investigates the consequences of not using a logarithmic transformation for the `nihills` data analysis. The second differs from the first in having a `dist` × `climb` interaction term, additional to linear terms in `dist` and `climb`.

   (a) Fit the two models:

```
nihills.lm <- lm(time ~ dist+climb, data=nihills)
nihills2.lm <- lm(time ~ dist+climb+dist:climb, data=nihills)
anova(nihills.lm, nihills2.lm)
```

(b) Using the F-test result, make a tentative choice of model, and proceed to examine diagnostic plots. Are there any problematic observations? What happens if these points are removed? Refit both of the above models, and check the diagnostics again.

7. Check the variance inflation factors for `bodywt` and `lsize` for the model `brainwt ~ bodywt + lsize`, fitted to the `litters` data set. Comment.

8. Apply the `lm.ridge()` function to the `litters` data, using the generalized cross-validation (GCV) criterion to choose the tuning parameter. (GCV is an approximation to cross-validation.)

   (a) In particular, estimate the coefficients of the model relating `brainwt` to `bodywt` and `lsize` and compare with the results obtained using `lm()`.
   (b) Using both ridge and ordinary regression, estimate the mean brain weight when litter size is 10 and body weight is 7. Use the bootstrap, with case-resampling, to compute approximate 95% percentile confidence intervals using each method. Compare with the interval obtained using `predict.lm()`.

9.* Compare the ranges of `dist` and `climb` in the data frames `nihills` and `hills2000`. In which case would you expect it to be more difficult to find a model that fits well? For each of these data frames, fit both the model based on the formula

```
log(time) ~ log(dist) + log(climb)
```

and the model based on the formula

```
time ~ alpha*dist + beta*I(climb^2)
```

Is there one model that gives the best fit in both cases?

10. The data frame `table.b3` in the *MPV* package contains data on gas mileage and 11 other variables for a sample of 32 automobiles.

    (a) Construct a scatterplot of `y` (mpg) versus `x1` (displacement). Is the relationship between these variables non-linear?
    (b) Use the `xyplot()` function, and `x11` (type of transmission) as a `group` variable. Is a linear model reasonable for these data?
    (c) Fit the model relating `y` to `x1` and `x11` which gives two lines having possibly different slopes and intercepts. Check the diagnostics. Are there any influential observations? Are there any influential outliers?
    (d) Plot the residuals against the variable `x7` (number of transmission speeds), again using `x11` as a `group` variable. Is there anything striking about this plot?

11. The following code is designed to explore effects that can result from the omission of explanatory variables:

```
> x1 <- runif(10)             # predictor which will be missing
> x2 <- rbinom(10, 1, 1-x1)   # observed predictor which depends
>                             # on missing predictor
> y <- 5*x1 + x2 + rnorm(10,sd=.1)    # simulated model; coef
>                                     # of x2 is positive
> y.lm <- lm(y ~ factor(x2)) # model fitted to observed data
> coef(y.lm)
(Intercept) factor(x2)1
  2.8224119  -0.6808925      # effect of missing variable:
                             # coefficient of x2 has wrong sign
> y.lm2 <- lm(y ~ x1 + factor(x2))    # correct model
> coef(y.lm2)
```

```
(Intercept)          x1 factor(x2)1
 0.06654892  4.91216206  0.92489061  # coef estimates are now OK
```

What happens if `x2` is generated according to `x2 <- rbinom(10, 1, x1)`? `x2 <- rbinom(10, 1, .5)`?

12. Fit the model investigated in Subsection 6.8.4, omitting the parameter α. Investigate and comment on changes in the fitted coefficients, standard errors, and fitted values.

13. Figure 6.19 used the function `errorsINx()`, with the argument `gpdiff=1.5`, to simulate data in which the regression relationship $y = 15 + 1.5x$ is the same in each of two groups (called `ctl` and `trt`). The left panel identifies the two fitted lines when the explanatory variable is measured without error. These are, to within statistical error, identical. The right panel shows the fitted regression lines when random error of the same order of magnitude as the within-groups variation in x is added to x, giving the column of values `zWITHerr`.

    (a) Run the function for several different values of `gpdiff` in the interval (0, 1.5), and plot the estimate of the treatment effect against `gpdiff`.
    (b) Run the function for several different values of `timesSDz` in the interval (0, 1.5), and plot the estimate of the treatment effect against `gpdiff`.
    (c) Run the function with `beta = c(-1.5,0)`. How does the estimate of the treatment effect change, as compared with `b = c(1.5,0)`? Explain the change.

14. Fit the following two resistant regressions, in each case plotting the residuals against `Year`.

```
library(MASS}
nraw.lqs <- lqs(northRain ~ SOI + CO2, data=bomregions)
north.lqs <- lqs(I(northRain^(1/3)) ~ SOI + CO2, data=bomregions)
par(mfrow=c(2,1))
plot(residuals(nraw.lqs) ~ Year, data=bomregions)
plot(residuals(north.lqs) ~ Year, data=bomregions)
par(mfrow=c(1,1))
```

    Compare, also, normal probably plots for the two sets of residuals.

    (a) Repeat the calculations several times. Comment on the extent of variation, from one run to the next, in the regression coefficients.
    (b) Based on examination of the residuals, which regression model seems more acceptable: `nraw.lqs or north.lqs`?
    (c) Compare the two sets of regression coefficients. Can you explain why they are so very different?

    (More careful modeling will take into account the temporal sequence in the observations. See Section 9.2 for an analysis that does this.)

7

Exploiting the linear model framework

The model matrix X is fundamental to all calculations for a linear model. The model matrix carries the information needed to calculate the fitted values that correspond to any particular choice of coefficients. There is a one-to-one correspondence between columns of X and regression coefficients.

In Chapter 6, the columns of the model matrix contained the observed values of the explanatory variables, perhaps after transformation. Fitted values were obtained by multiplying the first column by the first coefficient (usually the intercept), the second column by the second coefficient, and so on across all columns. The sum of the products in any row is the fitted value for that row.

This chapter will explore new ways to relate the columns of the model matrix to the explanatory variables, where a *variable* may be either a vector of numeric values, or a factor. Vectors of zeros and ones (columns of "dummy" variables) can be used to handle factor levels, but as noted below there are other possibilities. For modeling a quadratic form of response, we take values of x as one of the columns and values of x^2 as another. The model matrix framework also allows the modeling of many other forms of non-linear response. As before, the regression calculations find the set of coefficients that best predicts the observed responses, in the sense of minimizing the sum of squares of the residuals.

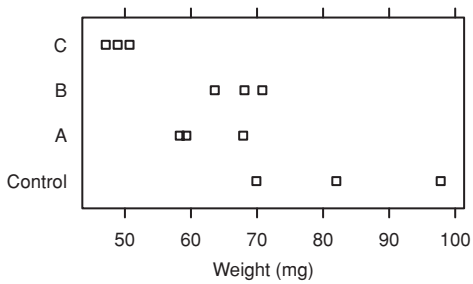
The latter part of this chapter will describe the fitting of smooth curves and surfaces that do not necessarily have a simple parametric form of mathematical description. Although not as convenient as linear or other simple parametric relationships, such curves and surfaces do allow the calculation of predicted values and associated standard error estimates. For many purposes, this is all that is needed.

7.1 Levels of a factor – using indicator variables

7.1.1 Example – sugar weight

Figure 7.1 displays data from an experiment that compared an unmodified wild-type plant with three different genetically modified (GM) forms (data are in the data set `sugar` in our *DAAG* package). The measurements are weights (mg) of sugar that were obtained by breaking down the cellulose. There is a single explanatory factor (treatment), with one level for each of the different control agents. For convenience, we will call the factor levels Control, A (GM1), B (GM2), and C (GM3).[1]

[1] `stripplot(trt ~ weight, pch=0, xlab="Weight (mg)", data=sugar, aspect=0.5))`



Control (WT[a])	A (GM1)	B (GM2)	C (GM3)
82.0	58.3	68.1	50.7
97.8	67.9	70.8	47.1
69.9	59.3	63.6	48.9
Mean =			
83.2	61.8	67.5	48.9

[a] WT = Wild Type; GM = Genetically Modified

Figure 7.1 Weights (`weight`) of sugar extracted from a control (wild-type) plant, and from three different genetically modified plant types.

We could reduce the apparent difference in variability between treatments by working with `log(weight)`. For present illustrative purposes, we will however work with the variable `weight`, leaving as an exercise for the reader the analysis that works with `log(weight)`.

The model can be fitted either using the function `lm()` or using the function `aov()`. The two functions give different default output.

For any problem that involves factor(s), there are several different ways to set up the model matrix. A common strategy (on a vanilla setup, the default) is to set up one of the treatment levels as a baseline or reference, with the effects of other treatment levels then measured from the baseline. Here it makes sense to set `Control` (Wild) as the baseline.

Before proceeding, it may pay to do the following check:

```
> options()$contrasts  # Check the default factor contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"
> ## If your output does not agree with the above, then enter
> options(contrasts=c("contr.treatment", "contr.poly"))
```

The reason for this check will become apparent shortly.

With `Control` as baseline, a one-way analysis of variance of the data in Figure 7.1 has the model matrix that is shown in Table 7.1; values of the response (`sugar$weight`) have been added in the final column. The following checks the order of the levels in the column `trt` and prints the model matrix:

```
> levels(sugar$trt)  #  Note the order of the levels
[1] "Control" "A"       "B"       "C"
> sugar.aov <- aov(weight ~ trt, data=sugar)
> model.matrix(sugar.aov)
```

Here are the results from the least squares calculations:

```
> summary.lm(sugar.aov)    # NB: summary.lm(),
+                          # not summary() or summary.aov()

Call:
aov(formula = weight ~ trt, data = sugar)
Residuals:
```

Table 7.1 *Model matrix for the analysis of variance calculation for the data in Figure 7.1. The values of the response are in the final column.*

Control (baseline)	A	B	C	`weight`
1	0	0	0	82.0
1	0	0	0	97.8
1	0	0	0	69.9
1	1	0	0	58.3
1	1	0	0	67.9
1	1	0	0	59.3
1	0	1	0	68.1
1	0	1	0	70.8
1	0	1	0	63.6
1	0	0	1	50.7
1	0	0	1	47.1
1	0	0	1	48.9

```
    Min      1Q  Median      3Q     Max
-13.333  -2.783  -0.617   2.175  14.567
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    83.23       4.47   18.61  7.2e-08
trtA          -21.40       6.33   -3.38  0.00960
trtB          -15.73       6.33   -2.49  0.03768
trtC          -34.33       6.33   -5.43  0.00062

Residual standard error: 7.75 on 8 degrees of freedom
Multiple R-Squared: 0.791,      Adjusted R-squared: 0.713
F-statistic: 10.1 on 3 and
8 degrees of freedom,           p-value: 0.00425
```

The row labeled `(Intercept)` gives the estimate (= 83.23) for the baseline, i.e., `Control`. The remaining coefficients (differences from the baseline) are:

A: weight differs by −21.40.
B: weight differs by −15.73.
C: weight differs by −34.33.

All three differences from the control are significant at the conventional 5% level.

In Table 7.2, the multiples determined by the least squares calculations are shown above each column. Also shown is $\widehat{y}$, which is the fitted or predicted value, calculated either as `fitted(sugar.aov)` or as `predict(sugar.aov)`.

Residuals can be obtained by subtracting the predicted values ($\widehat{y}$) in Table 7.2 from the observed values (y) in Table 7.1.

In this example, the estimate for each treatment is the treatment mean. Regression calculations have given us a complicated way to compute averages! The methodology

Table 7.2 *At the head of each column is the multiple, as determined by least squares, that is taken in forming the fitted values.*

Control: 83.2	A: −21.4	B: −15.7	C: −34.3	Fitted value
1	0	0	0	83.2
1	0	0	0	83.2
1	0	0	0	83.2
1	1	0	0	61.8
1	1	0	0	61.8
1	1	0	0	61.8
1	0	1	0	67.5
1	0	1	0	67.5
1	0	1	0	67.5
1	0	0	1	48.9
1	0	0	1	48.9
1	0	0	1	48.9

shows its power to better effect in more complex models, where there is no such simple alternative.

Use of the overall analysis of variance F-test, prior to these individual comparisons, is often a sufficient safeguard against over-interpretation of the results of such comparisons. Nevertheless, insight may be gained from assessing differences against Tukey's experimentwise HSD criterion that was discussed in Subsection 4.4.1.

```
> sem <- summary.lm(sugar.aov)$sigma/sqrt(3)  # 3 results/trt
> # Alternatively, sem <- 6.33/sqrt(2)
> qtukey(p=.95, nmeans=4, df=8) * sem
1] 20.26
```

Using this stricter criterion, B cannot be distinguished from the control, and A, B and C cannot be distinguished from each other.

7.1.2 Different choices for the model matrix when there are factors

In the language used in the R help pages, different choices of *contrasts* are available, with each different choice leading to a different model matrix. The different choices thus give different mathematical descriptions for the same model. The coefficients (parameters) change and must be interpreted differently. The fitted values and the analysis of variance table do not change. The choice of contrasts may call for careful consideration, in order to obtain coefficient estimates with the interpretation that is most helpful for the problem in hand. Or, more than one run of the analysis may be necessary, to gain information on all effects of interest.

The default (*treatment*) choice of *contrasts* uses the initial factor level as baseline, as we have noted. Different choices of the baseline or reference level lead to different versions of the model matrix. The other common choice, i.e., *sum* contrasts, uses the average of treatment effects as the baseline.

In order to use sum contrasts in place of treatment contrasts, specify `options(contrasts=c("contr.sum", "contr.poly"))`. It is also possible to set contrasts separately for each factor. See `help(C)`.

Here is the output when the baseline is the average of the treatment effects, i.e., from using the *sum* contrasts:[2]

```
> oldoptions <- options(contrasts=c("contr.sum", "contr.poly"))
>  # The mean over all treatment levels is now the baseline.
>  # (The second setting ("contr.poly") is for ordered factors.)
> summary.lm(aov(weight ~ trt, data = sugar))
. . . .
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    65.37       2.24   29.23  2.0e-09
trt1           17.87       3.87    4.61   0.0017
trt2           -3.53       3.87   -0.91   0.3883
trt3            2.13       3.87    0.55   0.5968

Residual standard error: 7.75 on 8 degrees of freedom
Multiple R-Squared: 0.791,      Adjusted R-squared: 0.713
F-statistic: 10.1 on 3 and
8 degrees of freedom,           p-value: 0.00425

> options(oldoptions)  # Restore default treatment contrasts
```

Note the differences from the output from the default choice of contrasts. The baseline, labeled `(Intercept)`, is now the treatment mean. This equals 65.37. Remaining coefficients are differences, for `Control` and for treatment levels A and B, from this mean. The sum of the differences for all three treatments is zero. Thus the difference for C is (rounding up)

$$-(17.87 - 3.53 + 2.13) = -16.5.$$

The estimates (means) are:

Control: $65.37 + 17.87 = 83.2$.
A: $65.37 - 3.53 = 61.8$.
B: $65.37 + 2.13 = 67.5$.
C: $65.37 - 16.5 = 48.9$.

Note also the possibility of using *helmert* contrasts. For a factor that has two levels, helmert contrasts lead to a parameter estimate that is just half that for treatment contrasts. For factors with more than two levels, the parameter estimates that are associated with helmert contrasts rarely correspond to the scientific questions that are of interest.

For ordered factors, *polynomial* contrasts are the default. There is a brief discussion of contrasts in Section 14.6.

[2] The first vector element specifies the choice of contrasts for factors (i.e., unordered factors), while the second specifies the choice for ordered factors.

7.2 Block designs and balanced incomplete block designs

Data in the data frame `rice` (*DAAG*) were displayed in Figure 4.7. They were from an experiment where the plants were laid out in blocks, with each treatment combination occurring once in each block. As all combinations of factors occur equally often in each block, the experimental design is a complete block design.

The data in `appletaste` are from a *balanced incomplete block design* (BIBD). In this particular BIBD, one treatment is left out of each block, but in such a way that the number of blocks in which a treatment is left out is the same for all treatments. (More generally, the requirement for a BIBD is that all treatments must occur together equally often in the same block.)

Blocks should be chosen so that conditions are as uniform as possible within each block. In a glasshouse (or greenhouse) experiment all plants in a single block should be in a similar position in the glasshouse, with a similar exposure to light.

7.2.1 Analysis of the rice data, allowing for block effects

In general, there should be allowance for block differences when data from block designs are analyzed. Otherwise, if there are substantial differences between blocks, treatment effects are likely to be masked by these substantial block differences. The interest is in knowing the extent to which treatment differences are consistent across blocks, irrespective of block-to-block differences that affect all plants in a block pretty much equally.

The analysis of variance table is a useful first point of reference, for examining results:

```
> ricebl.aov <- aov(ShootDryMass ~ Block + variety * fert,
                    data=rice)
> summary(ricebl.aov)
             Df Sum Sq Mean Sq F value  Pr(>F)
Block         1   3528    3528    10.9  0.0016
variety       1  22685   22685    70.1 6.4e-12
fert          2   7019    3509    10.8 8.6e-05
variety:fert  2  38622   19311    59.7 1.9e-15
Residuals    65  21034     324
```

This makes it clear that there are substantial differences between blocks.

Use `summary.lm()` to obtain details of the effects:

```
> summary.lm(ricebl.aov)
. . . .
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                 115.33       5.61   20.56  < 2e-16
Block2                      -14.00       4.24   -3.30   0.0016
varietyANU843              -101.00       7.34  -13.75  < 2e-16
fertNH4Cl                   -58.08       7.34   -7.91  4.2e-11
fertNH4NO3                  -35.00       7.34   -4.77  1.1e-05
varietyANU843:fertNH4Cl      97.33      10.39    9.37  1.1e-13
varietyANU843:fertNH4NO3     99.17      10.39    9.55  5.4e-14
```

```
Residual standard error: 18 on 65 degrees of freedom
Multiple R-Squared: 0.774,      Adjusted R-squared: 0.753
F-statistic:   37 on 6 and 65 DF,  p-value: <2e-16
```

The above residual standard error, i.e., 18.0 on 65 degrees of freedom, may be compared with a standard error of 19.3 on 66 degrees of freedom when there is no allowance for block effects.[3]

Because this was a complete balanced design, the function `model.tables()` can be used to obtain a summary of treatment effects in a pleasantly laid out form. Any visual summary of results should, at a minimum, include the information given in Figure 4.7. (Do not try to use `model.tables()` for anything other than complete balanced designs. Even for balanced "incomplete" results such as will now be discussed, results will be incorrect.)

7.2.2 A balanced incomplete block design

In tasting experiments, a number of different products, e.g., wines, are to be compared. If presented with too many different specimens to test, tasters can become confused, even when precautions are taken (including washing the palette) to minimize carry-over effects from one product to another. Hence it is usual to limit the number of products given to any one taster.

In the example that will now be given, the products were different varieties of apple, identified by the numerical codes `298`, `493`, `649`, and `937`. The 20 tasters were divided into four groups of five. For each group of five tasters, a different product was omitted. Panelists made a mark on a line that gave their rating of `aftertaste` (`0` for extreme dislike; `150` for extreme approval). The following is a summary of the experimental design:

```
> table(appletaste$product, appletaste$panelist)

     a b c d e f g h i j k l m n o p q r s t
298  1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
493  1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1
649  0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
937  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
```

The tasters play the role that blocks would play in a field design. In spite of differences in the way that different raters use the scale (some will tend to score low and some high), there may be acceptable consistency in their comparative ratings of the products.

For analysis, it is necessary only to specify factors `panelist` and `product` as explanatory factors.

```
> sapply(appletaste, is.factor)  # panelist & product are factors
aftertaste   panelist    product
     FALSE       TRUE       TRUE
> summary(appletaste.aov <- aov(aftertaste ~ panelist + product,
+                               data=appletaste))
```

[3] `## AOV calculations, ignoring block effects`
`rice.aov <- aov(ShootDryMass ~ variety * fert, data=rice)`
`summary.lm(rice.aov)$sigma`

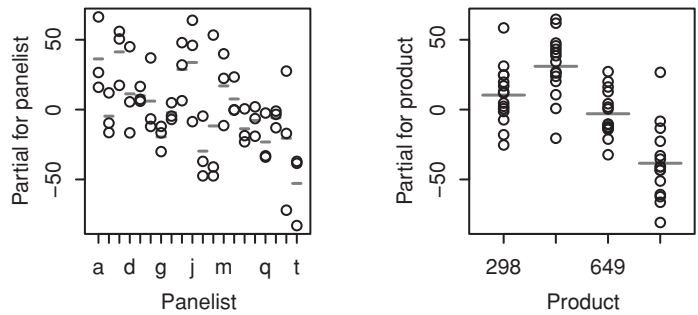


Figure 7.2 These plots show the respective contributions of the factors `panelist` and `product` to `aftertaste` scores, in an apple tasting experiment.

```
            Df Sum Sq Mean Sq F value  Pr(>F)
panelist    19  30461    1603    2.21   0.019
product      3  34014   11338   15.60 1.0e-06
Residuals   37  26892     727
```

There are differences between the ratings of different panelists, but they are of minor consequence relative to differences between products.

Figure 7.2 shows the partial residual plot, obtained using the function `termplot()` with the argument `partial=TRUE`; this gives a useful summary of the results. Notice that ratings seem generally lower for the final few raters. Had product quality deteriorated over time?

In its present form, this function is useful only for displaying the effects of factors for which no interaction terms are present. As noted above, do not try to use `model.tables()` to obtain estimates of effects; the results, for this incomplete block design, will be incorrect.

7.3 Fitting multiple lines

Multiple regression can be used to fit multiple lines. In the example that follows (Table 7.3), there are measurements of vapor pressure (`vapPress`) and of the difference between leaf and air temperature (`tempDiff`), for three different levels of carbon dioxide.

Possibilities we may want to consider are:

- Model 1 (constant response): $y = a$.
- Model 2 (a single line): $y = a + bx$.
- Model 3 (three parallel lines): $y = a_1 + a_2 z_2 + a_3 z_3 + bx$.
  (For the low CO_2 group ($z_2 = 0$ and $z_3 = 0$) the constant term is a_1; for the medium CO_2 group ($z_2 = 1$ and $z_3 = 0$) the constant term is $a_1 + a_2$; while for the high CO_2 group ($z_2 = 0$ and $z_3 = 1$) the constant term is $a_1 + a_3$.)
- Model 4 (three separate lines): $y = a_1 + a_2 z_2 + a_3 z_3 + b_1 x + b_2 z_2 x + b_3 z_3 x$.
  (Here, z_2 and z_3 are as in model 3 (panel B). For the low CO_2 group ($z_2 = 0$ and $z_3 = 0$) the slope is b_1; for the medium CO_2 group ($z_2 = 1$ and $z_3 = 0$) the slope is $b_1 + b_2$; while for the high CO_2 group ($z_2 = 0$ and $z_3 = 1$) the slope is $b_1 + b_3$.)

Table 7.3 *Selected rows, showing values of* `CO2level`, `vapPress` *and* `tempDiff`, *from the data set* `leaftemp`.

`CO2level`	`vapPress`	`tempDiff`
low	1.88	1.36
low	2.20	0.60
...	...	...
medium	2.38	1.94
medium	2.72	0.83
...	...	...
high	2.56	1.50
high	2.55	0.85
...	...	...

Table 7.4 *Model matrix for fitting three parallel lines (model 3) to the data of Table 7.3. The y-values are in the separate column to the right.*

`(Inter-cept)`	Medium	High	`vapPress`	`tempDiff`
1	0	0	1.88	1.36
1	0	0	2.2	0.6
...	...	...	...	...
1	1	0	2.38	1.94
1	1	0	2.72	0.83
...	...	...	...	...
1	0	1	2.56	1.5
1	0	1	2.55	0.85
...	...	...	...	...

Table 7.5 *Model matrix for fitting three separate lines (model 4), with y-values in the separate column to the right.*

`(Intercept)`	Medium	High	`vapPress`	Medium: `vapPress`	High: `vapPress`	`tempDiff`
1	0	0	1.88	0	0	1.36
1	0	0	2.2	0	0	0.6
...	...	...	...	...	...	...
1	1	0	2.38	2.38	0	1.94
1	1	0	2.72	2.72	0	0.83
...	...	...	...	...	...	...
1	0	1	2.56	0	2.56	1.5
1	0	1	2.55	0	2.55	0.85
...	...	...	...	...	...	...

Selected rows from the model matrices for model 3 and model 4 are displayed in Tables 7.4 and 7.5, respectively.

The statements used to fit the four models are:

```
## Fit various models to columns of data frame leaftemp (DAAG)
leaf.lm1 <- lm(tempDiff ~ 1 , data = leaftemp)
leaf.lm2 <- lm(tempDiff ~ vapPress, data = leaftemp)
leaf.lm3 <- lm(tempDiff ~ CO2level + vapPress, data = leaftemp)
leaf.lm4 <- lm(tempDiff ~ CO2level + vapPress
               + vapPress:CO2level, data = leaftemp)
```

Recall that `CO2level` is a factor and `vapPress` is a variable. Technically, `vapPress:CO2level` is an interaction. The effect of an interaction between a factor and a variable is to allow different slopes for different levels of the factor.

The analysis of variance table is helpful in making a choice between these models:

Table 7.6 *Analysis of variance information. The starting point is a model that has only an intercept or "constant" term. The entries in rows 1–3 of the* `Df` *column and of the* `Sum of Sq` *column are then sequential decreases from fitting, in turn,* `vapPress`*, then three parallel lines, and then finally three separate lines.*

	`Df`	`Sum of Sq`	Mean square	`F`	`Pr(<F)`	
`vapPress` (variable)	1	5.272	5.272	11.3	0.0014	Reduction in SS due to fitting one line
Three parallel lines	2	6.544	3.272	7.0	0.0019	Additional reduction in SS due to fitting two parallel lines
Three different lines	2	2.126	1.063	2.3	0.1112	Additional reduction in SS due to fitting two separate lines
Residuals	61	40.000	0.656			

```
> anova(leaf.lm1, leaf.lm2, leaf.lm3, leaf.lm4)

Analysis of Variance Table

Model 1: tempDiff ~ 1
Model 2: tempDiff ~ vapPress
Model 3: tempDiff ~ CO2level + vapPress
Model 4: tempDiff ~ CO2level + vapPress + CO2level:vapPress
  Res.Df    RSS Df Sum of Sq       F   Pr(>F)
1     61  40.00
2     60  34.73  1     5.272   11.33   0.0014
3     58  28.18  2     6.544    7.03   0.0019
4     56  26.06  2     2.126    2.28   0.1112
```

This is a sequential analysis of variance table. Thus, the quantity in the sum of squares column (`Sum of Sq`) is the reduction in the residual sum of squares due to the inclusion of that term, given that earlier terms had already been included. The `Df` (degrees of freedom) column gives the change in the degrees of freedom due to the addition of that term. Table 7.6 explains this in detail.

The analysis of variance table suggests use of the parallel line model, shown in panel B of Figure 7.3. The reduction in the mean square from model 3 (panel B in Figure 7.3) to model 4 (panel C) in the analysis of variance table has a p-value equal to 0.1112. The coefficients and standard errors for model 3 are:

```
> summary(leaf.lm3)

Call:
lm(formula = tempDiff ~ CO2level + vapPress,
   data = leaftemp)
. . . .
```

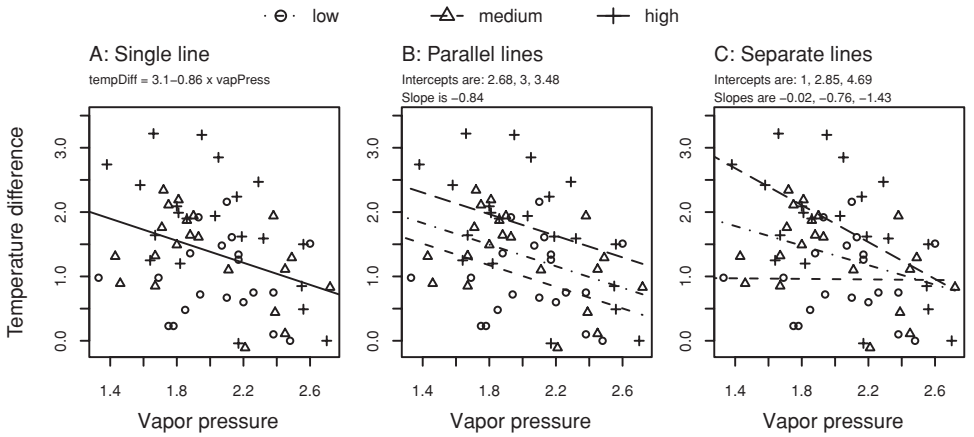


Figure 7.3 A sequence of models fitted to the plot of `tempDiff` versus `vapPress`, for low, medium and high levels of `CO2level`. Panel A relates to model 2, panel B to model 3, and panel C to model 4.

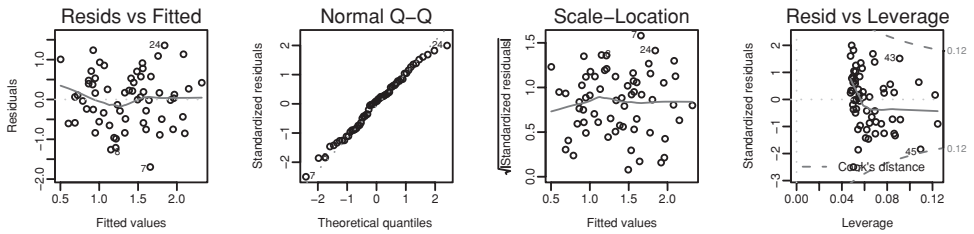


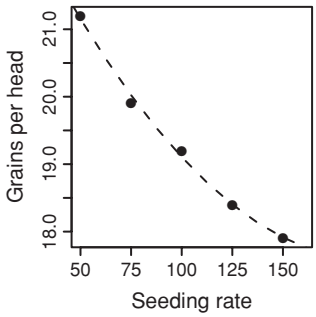
Figure 7.4 Diagnostic plots for the parallel line model of Figure 7.3.

```
Coefficients:
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)     2.685      0.560    4.80  1.16e-05
CO2levelmedium  0.320      0.219    1.46  0.14861
CO2levelhigh    0.793      0.218    3.64  0.00058
vapPress       -0.839      0.261   -3.22  0.00213

Residual standard error: 0.69707 on 58 degrees of freedom
Multiple R-Squared: 0.295,    Adjusted R-squared: 0.259
F-statistic: 8.106 on 3 and
58 degrees of freedom,        p-value: 0.000135
```

The coefficients in the equations for this parallel line model are given in the annotation for Figure 7.3B. For the first equation (low CO_2), the constant term is 2.685; for the second equation (medium CO_2), the constant term is $2.685 + 0.320 = 3.005$; while for the third equation, the constant term is $2.685 + 0.793 = 3.478$.

In addition, we examine a plot of residuals against fitted values, and a normal probability plot of residuals (Figure 7.4). These plots seem unexceptional.



Data

	rate	grain
1	50	21.2
2	75	19.9
3	100	19.2
4	125	18.4
5	150	17.9

Model matrix – quadratic fit

(Intercept)	rate	rate^2
1	50	2 500
1	75	5 625
1	100	10 000
1	125	15 625
1	150	22 500

Figure 7.5 Plot of number of grains per head versus seeding rate, for the barley seeding rate data shown to the right of the figure, with fitted quadratic curve. The model matrix for fitting a quadratic curve is shown on the far right. Data relate to McLeod (1982).

7.4 Polynomial regression

Polynomial regression provides a straightforward way to model simple forms of departure from linearity. The simplest case is where the response curve has a simple cup-up or cup-down shape. For a cup-down shape, the curve has some part of the profile of a path that follows the steepest slope up a rounded hilltop towards the summit and down over the other side. For a cup-up shape the curve passes through a valley. Such cup-down or cup-up shapes can often be modeled quite well using quadratic, i.e., polynomial with degree 2, regression. For this the analyst uses x^2 as well as x as explanatory variables. If a straight line is not adequate, and the departure from linearity suggests a simple cup-up or cup-down form of response, then it is reasonable to try a quadratic regression. The calculations are formally identical to those for multiple regression.

To avoid numerical problems, it is often preferable to use orthogonal polynomial regression. Interested readers may wish to pursue for themselves the use of orthogonal polynomial regression, perhaps using as a starting point Exercise 18 at the end of the chapter. Orthogonal polynomials have the advantage that the coefficient(s) of lower-order terms (linear, . . .) do(es) not change when higher-order terms are added. One model fit, with the highest-order term present that we wish to consider, provides the information needed to make an assessment about the order of polynomial that is required. The orthogonal polynomial coefficients must be translated back into coefficients of powers of x (these are not of course independent), if these are required.

Figure 7.5 shows number of grains per head (averaged over eight replicates), for different seeding rates of barley. A quadratic curve has been fitted. The code is:

```
## Fit quadratic curve: data frame seedrates (DAAG)
seedrates.lm2 <- lm(grain ~ rate + I(rate^2), data = seedrates)
  # The wrapper function I() ensures that the result from
  # calculating rate^2 is treated as a variable in its own right.
plot(grain ~ rate, data = seedrates, pch = 16,
     xlim = c(50, 160),  cex=1.4)
new.df <- data.frame(rate = (1:14) * 12.5) # for plotting the fitted curve
hat2 <- predict(seedrates.lm2, newdata = new.df, interval="predict",
                coverage = 0.95)
lines(new.df$rate, hat2[, "fit"], lty = 2, lwd=2)
```

The quadratic regression appears, from visual inspection, a good fit to the data. The fitted model may be written

$$\widehat{y} = a + b_1 x_1 + b_2 x_2$$

where $x_1 = x$, and $x_2 = x^2$. Thus, the model matrix has a column of 1s, a column of values of x, and a column that has values of x^2.

Here is the output from R:

```
> summary(seedrates.lm2, corr=TRUE)

Call:
lm(formula = grain ~ rate + I(rate^2), data = seedrates)

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept)  24.060    0.456     52.799   0.000
       rate  -0.067    0.010     -6.728   0.021
  I(rate^2)   0.000    0.000      3.497   0.073

Residual standard error: 0.115 on 2 degrees of freedom
Multiple R-Squared: 0.996
F-statistic: 256 on 2 and 2 degrees of freedom,
the p-value is 0.0039

Correlation of Coefficients:
          (Intercept)   rate
     rate -0.978
I(rate^2)  0.941      -0.989
```

(In a model formula, `rate^2` will be interpreted as `rate:rate` = `rate`. Hence the use of `I(rate^2)` to denote the square of `rate`.) Observe the high correlations between the coefficients. Note in particular the large negative correlation between the coefficients for `rate` and `I(rate^2)`. Forcing the coefficient for `rate` to be high would lead to a low coefficient for `I(rate^2)`, and so on.

In orthogonal polynomial regression, the separate terms `rate` and `I(rate^2)` are replaced by the single term `poly(rate,2)`, i.e., an orthogonal polynomial of degree 2 in `rate`. The fitted values will be identical, but the coefficients are then coefficients of the orthogonal polynomials, not coefficients of `rate` and `I(rate^2)`.

7.4.1 Issues in the choice of model

The coefficient of the x^2 term in the quadratic model fell short of statistical significance at the 5% level. Fitting the x^2 leaves only two degrees of freedom for error. For prediction, our interest is likely to be in choosing the model that is on balance likely to give the more accurate predictions; for this, use of the model that includes the quadratic term may be preferred.

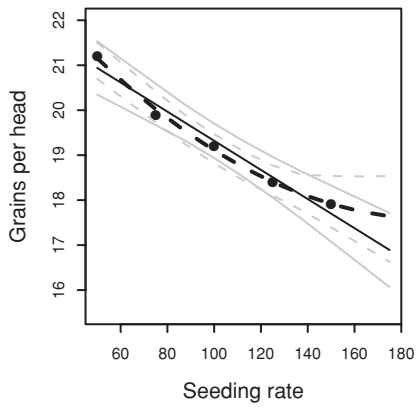


Figure 7.6 Number of grains per head versus seeding rate, with fitted line (solid) and fitted quadratic curve (dashed). Also shown are 95% pointwise confidence bounds.

Figure 7.6 shows both a fitted line and a fitted curve, in both cases with 95% confidence bounds.[4] It shows a quadratic curve (dashed line) as well as a line (solid line). In addition, the graph shows 95% pointwise confidence bounds for the expected number of grains per head, both about the line and about the curve.

The curve is a better fit to the data than the line. For a short distance beyond the final data point, it almost certainly gives a better estimate than does the line. Notice that the confidence bounds for the curve are much wider, beyond a rate of about 160, than the line. Not only does the line almost certainly give a biased estimate, it also gives unrealistically narrow bounds for that estimate. If the model is wrong, it will give wrong estimates of predictive accuracy. This is especially serious for extrapolation beyond the limits of the data.

Beyond the limits of the data, it would be unwise to put much trust in either the line or the curve. Our point is that the bounds for the quadratic curve do better reflect uncertainty in the curve that ought to be fitted. We could try other, single-parameter, models. Selecting a model from a number of choices that allow for the curvature may not however be much different, in its effect on the effective degrees of freedom, from adding an x^2 term. The wider confidence bounds for the quadratic model reflect this uncertainty in choice of model, better than results from any individual model that has one parameter additional to the intercept.

We can in fact fit the data well by modeling `grain` as a linear function of `log(rate)`. This model seems intuitively more acceptable; the fitted value of `grain` continues to

[4]
```
## Fit line, fit curve, determine pointwise bounds, and create the plots
CIcurves <-
  function(form=grain~rate, data=seedrates, lty=1, col=3,
           newdata=data.frame(rate=seq(from=50, to=175, by=25))){
    seedrates.lm <- lm(form, data=data)
    x <- newdata[, all.vars(form)[2]]
    hat <- predict(seedrates.lm, newdata=newdata, interval="confidence")
    lines(spline(x, hat[, "fit"]))
    lines(spline(x, hat[, "lwr"]), lty=lty, col=col)
    lines(spline(x, hat[, "upr"]), lty=lty, col=col)
  }
plot(grain ~ rate, data=seedrates, xlim=c(50,175), ylim=c(15.5,22))
CIcurves()
CIcurves(form=grain~rate+I(rate^2), lty=2)
```

decrease as the `rate` increases beyond the highest rate used in the experiment. This is perhaps the model that we should have chosen initially on scientific grounds.

7.5* Methods for passing smooth curves through data

In the previous section, we used the linear model framework to fit a curve that had x and x^2, etc. terms. This framework can be adapted to fit higher-order polynomial curves to regression data. For a polynomial of degree m, the model matrix must have, in addition to a column of 1s, columns that hold values of $x, x^2, \ldots, x^m$. Polynomials can be effective when a curve of degree m equal to 2 or 3 is appropriate. Polynomial curves where m is greater than 3 can be problematic. High-degree polynomials tend to move up and down between the data values in a snake-like manner. Splines, or piecewise polynomials, which we now consider, are usually preferable to polynomials of degree greater than 3.

A spline curve joins two or more polynomial curves, and is sometimes called a piecewise polynomial curve. The locations of the joints are called *knots*.

The following is a simple piecewise linear spline function, with a knot at $x = 2$:

$$y = 3 + 4x - 5(x - 2)I_{\{x \geq 2\}}. \tag{7.1}$$

The indicator I takes the value 1, when $x \geq 2$ and the value 0, when $x < 2$. For values of x less than 2, this spline function behaves as a straight line with slope 4, but for x greater than 2, the slope switches to $4 - 5 = -1$.

More generally, spline functions take the form

$$y = b_0 P_0(x) + b_1 P_1(x) + \cdots + b_k P_k(x)$$

where the b_is are constant coefficients, and the functions $P_i(x)$ are either polynomial functions like the first two terms of equation (7.1), or polynomial functions multiplied by indicators like the last term in equation (7.1).

Given a set of knots, there are many different ways to choose the $P_i(x)$. Some choices that are useful in practice may involve relatively complicated forms of algebraic expression that are different from those used in our definition. The $P_i(x)$ are known as *basis* functions.

A particularly convenient set of basis functions for splines is referred to as B-splines. Given data of the form $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we can write a regression model in terms of these B-spline functions as:

$$y = b_0 B_0(x) + b_1 B_1(x) + \cdots + b_k B_k(x) + \varepsilon$$

where ε represents an error term as in the previous chapter. Written in this form, it is clear that a spline regression model can be fit using the same methods as for fitting multiple regression models. The columns of the model matrix are constructed from evaluating each of the B-splines at each of the values of x. Once the basis functions have been evaluated, the `lm()` function can be used to carry out the estimation of the regression coefficients.

Natural splines, implemented using the function `ns()` from the *splines* package, are an alternative to B-splines. For natural splines, the slope of the curve is constrained to be constant at and beyond the boundary knots. For B-splines, there are no boundary constraints, though there are boundary knots that anchor the spline basis. By default, these are placed at the limits of the data.

Table 7.7 *Resistance (ohms) versus apparent juice content. The table shows a selection of the data.*

	Juice (%)	Ohms		Juice (%)	Ohms		Juice (%)	Ohms		Juice (%)	Ohms
1	4	4860	33	20	7500	65	41.5	3350	123	58.5	3650
2	5	5860	34	20.5	8500	66	42.5	2700	124	58.5	3750
3	5.5	6650	35	21.5	5600	67	43	2750	125	58.5	4550
4	7.5	7050	36	21.5	6950	68	43	3150	126	59.5	3300
5	8.5	5960	37	21.5	7200	69	43	3250	127	60	3600
...	...	...	...	...	...	...	...	...	128	9	9850

The number of degrees of freedom (`df`) additional to the intercept will be specified for the spline curve, with the software then allowed to determine the number and location of internal join points (=knots). For B-splines, the number of internal knots is `df - degree`, where `degree` is the degree (by default 3) of the piecewise polynomial. For natural splines, the number of internal knots is `df - degree` + 2, where `degree` must be 3. (The difference in `df` is a result of the boundary constraints.) In either case, internal knots are by default placed at equally spaced quantiles of the data.

The use of regression splines, using B-spline or N-spline bases, will be demonstrated in the next subsection. The discussion will then move to more general types of smoothing terms. We will take a simple example where there is just one explanatory variable, and try several different methods on it.

7.5.1 Scatterplot smoothing – regression splines

We have (in Table 7.7) the apparent juice content and resistance (in ohms) for 128 slabs of fruit (these data relate to Harker and Maindonald, 1994). Figure 7.7 shows four different curves fitted to these data:[5] Figures 7.7A and B show spline curves, the first with one knot and the second with two knots. Figures 7.7C and D show, for comparison, third- and fourth-degree polynomials. The polynomials do quite well here relative to the splines. Also shown are 95% pointwise confidence intervals for the fitted curves.

Diagnostic plots can be used, just as for the models considered in earlier chapters, to highlight points that are associated with large residuals, or that are having a strong influence

[5]
```
## Fit various models to columns of data frame fruitohms (DAAG)
library(splines)
## Panel A
plot(ohms ~ juice, cex=0.8, xlab="Apparent juice content (%)",
     ylab="Resistance (ohms)", data=fruitohms)
CIcurves(form=ohms ~ ns(juice, 2), data=fruitohms,
        newdata=data.frame(juice=pretty(fruitohms$juice,20)))
## For panels B, C, D replace form = ohms ~ ns(juice,2) by:
## form = ohms ~ ns(juice,3)   # panel B: nspline, df = 4
## ohms ~ poly(juice,2)        # panel C: polynomial, df = 3
## ohms ~ poly(juice,3)        # panel D: polynomial, df = 4
  # For more information on poly(), see help(poly) and Exercise 15.
```

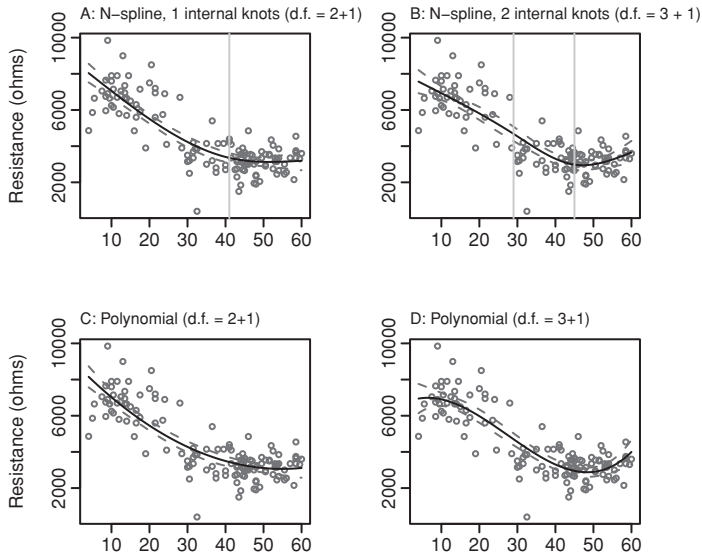


Figure 7.7 Different smooth curves fitted to the data of Table 7.7. The dashed lines show 95% pointwise confidence bounds for the fitted curve. In panels A and B, vertical lines show the locations of the knots. The degrees of freedom ("df" or "degree") shown are those supplied to `ns()` or `poly()`. These must in each case be increased by one to allow for the intercept.

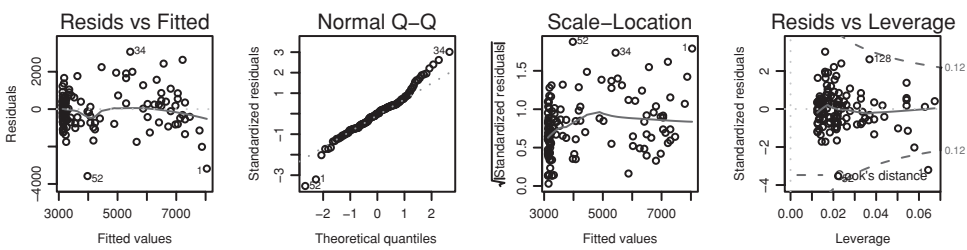


Figure 7.8 Diagnostic plots for the fitted model given in Figure 7.7A.

on the curve. Figure 7.8 shows the default diagnostic plots for the fitted model shown in Figure 7.7A.[6]

Apart from the large residual associated with point 52 (at 32.5% apparent juice content), these plots show nothing of note. The curves have bent to accommodate points near the extremes that might otherwise have appeared as outliers.

Here is the summary information:

```
> summary(fruit.lm2)
. . . .
```

[6]
```
## Fit degree 2 normal spline, plot diagnostics
par(mfrow = c(2,2))
fruit.lm2 <- lm(ohms ~ ns(juice,2), data=fruitohms)
  # for panel B: ns(juice,3)
plot(fruit.lm2)
par(mfrow = c(1,1))
```

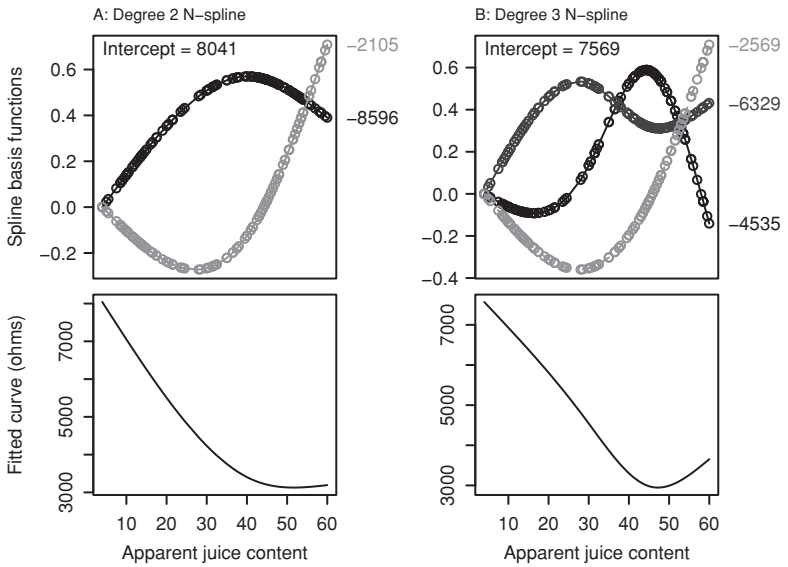


Figure 7.9 Panel A shows the N-spline basis curves (one knot) fitted in Figure 7.7A. The fitted curve, obtained by multiplying the values by −8596 and −2108, respectively, summing, and adding an intercept of 8041, is shown below. Panel B (two knots) relates, similarly, to Figure 7.7B.

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)        8041        260    30.9  < 2e-16
ns(juice, 2)1     -8596        578   -14.9  < 2e-16
ns(juice, 2)2     -2105        357    -5.9  3.3e-08

Residual standard error: 1030 on 125 degrees of freedom
Multiple R-Squared: 0.701,        Adjusted R-squared: 0.696
F-statistic:  146 on 2 and 125 DF,  p-value: <2e-16
```

Attention is best focused on the fitted curve, ignoring the fact that the curve can be constructed by the smooth joining of separate cubic curves, or as a linear combination of basis functions. Nevertheless, comments that will help make sense of the coefficients and standard errors in the R output may be helpful. To help understand how the curve has been formed as a linear combination of basis functions, we plot graphs that show the curves for which these are the coefficients. For this, we plot the relevant column of the X-matrix against x, and join up the points, as in Figure 7.9.[7]

[7]
```
## Display the basis curves as in panel A
mm2 <- model.matrix(fruit.lm2)
ylim <- range(mm2[, -1])
plot(mm2[, 2] ~ juice, ylim=ylim, xlab="Apparent Juice Content (%)",
     ylab="Spline basis functions", type="l", data=fruitohms)
lines(fruitohms$juice, mm2[, 3], col="gray")
  # NB: Values of juice are already ordered
## For panel B basis curves: mm3 <- model.matrix(fruit.lm3), etc.
```

Looking back again at the coefficients, basis curve 2 (with a coefficient of −8596) seems more strongly represented than the other basis curve.

Compare sets of N-spline basis functions with sets of polynomial basis functions. An orthogonal polynomial basis gives successively more accurate approximations to the data. A set of polynomial basis functions of degree 3 is the set of degree 2 with another basis function added. For the splines this is not the case. All the basis functions change if another degree of freedom is added. For given degrees of freedom, the prediction that they combine to give is accurate. If one basis function is dropped from the set of basis functions, predictions will be disastrously inaccurate. Instead, it is necessary to find a complete new set of basis functions for a spline curve with the reduced degrees of freedom.

The regression splines that we described above are attractive because they fit easily within a linear model framework, i.e., we can fit them by specifying an appropriate X-matrix. There are a wide variety of other methods, most of which do not fit within the linear model framework required for use of `lm()`.

7.5.2* Roughness penalty methods and generalized additive models

In the use of regression splines in the previous subsection, the approach was to specify the degrees of freedom for the spline curve; knots were then located at equal quantiles of the data. This is unlikely to be optimal. Use of too few knots can lead to a curve that fails to capture all of the nuances of the regression function while choosing too many can result in excessive bumpiness, i.e., the details of the curve capture noise.

The function `gam()` in the *mgcv* package implements generalized additive models (GAMs). The function `s()` is used to generate smoothing terms. A brief explanation will now be provided of some of the methods that are available, in the *mgcv* package, for generating smoothing terms. For more extended information, attach the *mgcv* package, and type `help(smooth.terms)`.

The roughness penalty approach is designed to reduce or remove arbitrariness that can result from the choice of the number and placement of knots. Cubic smoothing spline methods assign a knot to each predictor value, while applying a roughness penalty that constrains the fitted spline to smoothly pass through the cloud of observations. Note that without such a constraint, the spline would interpolate the observations, usually rendering a rough curve.

In order to reduce the time and memory requirements, knots can be placed at a subset of the predictor values. Even so, there will be many more knots than would be used in the absence of a roughness penalty. The "cubic regression splines" that are provided in the *mgcv* package place knots at equally spaced quantiles of the data, as does the function `smooth.spline()` (*stats* package). Penalized splines, as implemented by `pspline()` in the *survival* package, are another approach that places knots at values that are evenly spaced through the data.

Thin plate splines circumvent the overt choice of knots. The roughness penalty is, on its own, enough to determine a set of basis functions. The basis functions do, moreover, plausibly give successively more accurate approximations. They generalize in a natural manner to the fitting of smooth surfaces, in an arbitrary number of dimensions. In practice, in order to keep computational demands within reason, a low rank approximation is used,

resulting in a much reduced number of basis functions. Wood (2006) calls these thin plate regression splines.

The `fruitohms` data set furnishes examples:

```
library(mgcv)
## Thin plate regression splines (bs="tp")
fruit.tp <- gam(ohms ~ s(juice, bs="tp"), data=fruitohms)
## Plot points, fitted curve, and +/- 1SE limits
plot(fruit.tp, residuals=TRUE)
## Cubic regression splines (bs="cr")
fruit.cr <- gam(ohms ~ s(juice, bs="cr"), data=fruitohms)
## Plot points, fitted curve, and +/- 1SE limits
plot(fruit.cr, residuals=TRUE)
```

Note also the *gam* package, which ports to R the code used for the `gam()` function in S-PLUS.

7.5.3 Distributional assumptions for automatic choice of roughness penalty

The residual sum of squares is increased by a roughness (or wiggliness) penalty that is a multiple λ of the integral of the squared second derivative, to give a *penalized* sum of squares. Parameter estimates (multiples of the basis terms) are chosen to minimize this penalized residual sum of squares.

The `gam()` function's default method for choosing λ is a variant of generalized cross-validation. This relies on an analytical approximation to the expected value of the cross-validation estimate of the residual sum of squares. All automatic methods for choosing the penalty assume that errors are independently and identically distributed (i.i.d.), with implications that users will do well to keep in mind.

Sequential correlation structures, which are the major focus of Chapter 9, are a common type of departure from i.i.d. errors. Where there is a sequential correlation structure in the data, the methodology will – if possible – use a smooth curve to account for it. The pattern that is thus extracted will not be reproducible under a re-run of the process. Exercise 19 at the end of the chapter is designed to illustrate this point.

7.5.4 Other smoothing methods

Lowess curves are a popular alternative to spline curves. Figure 2.6 showed a curve that was fitted to the `fruitohms` data using the function `lowess()`, which always uses a resistant form of smoothing. The curve is thus relatively insensitive to large residuals. Special steps are taken to avoid distortions due to end effects. As implemented in R, `lowess()` is not available for use when there are multiple explanatory variables, and there is no mechanism (or theory) for calculating pointwise confidence bounds.

The `loess()` function is an alternative to `lowess()` that is able to handle multi-dimensional smoothing. The default for `loess()` is a non-resistant smooth; for a resistant smooth, specify `family=symmetric`.

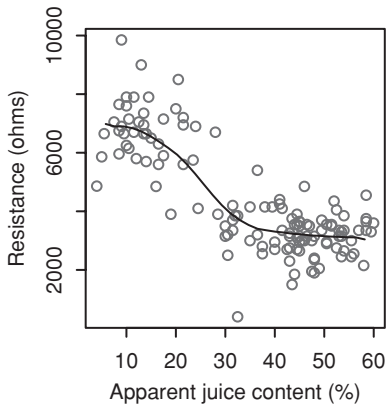


Figure 7.10 A monotonic decreasing spline curve has been fitted to the points shown in Figures 7.7 and 7.9.

Both `lowess()` and `loess()` implement a *locally weighted regression* methodology. The following description is directly relevant to `lowess()`; calculations for `loess()` follow the same general pattern. The method is said to be local, because the fitted value $m(x)$ at a point x uses only the data within a specified neighborhood of x. Points nearest to x are given highest weight. Those farther away are given little or no weight. Outlier resistance is achieved by assigning low weight to observations which generate large residuals; this allows for curves which are relatively unaffected by the presence of outliers. An iterative method is used, with the residual at the previous iteration determining the weight at the current iteration.

Kernel smoothing methods further widen the range of possibilities. For example, see the documentation for the function `locpoly()` in the *KernSmooth* package.

On lowess smoothing, see Cleveland (1981). There is a useful brief discussion of smoothing methods in Venables and Ripley (2002) and a fuller discussion of kernel smoothing, splines, and lowess in Fan and Gijbels (1996). See also Hall (2001).

*Monotone curves

Constraints can be included that force curves to be monotone increasing or monotone decreasing. The function `monoproc()` in the *monoProc* package can be used, starting with a fit using `loess()` or another function whose output follows the same conventions, to create a monotone fit, as in Figure 7.10.

The code is:

```
library(monoProc)
fit.mono <- monoproc(loess(ohms~juice, data=fruitohms),
                     bandwidth=0.1, mono1="decreasing",
                     gridsize=30)
plot(ohms ~ juice, data=fruitohms,
     xlab="Apparent juice content (%)", ylab="Resistance (ohms)")
lines(fit.mono)
```

Table 7.8 *Average dewpoint* (`dewpt`)*, for available combinations of monthly averages of minimum temperature* (`mintemp`) *and maximum temperature* (`maxtemp`)*. The table shows a selection of the data.*

	maxtemp	mintemp	dewpt		maxtemp	mintemp	dewpt
1	18	8	7	67	38	26	20
2	18	10	10	68	40	18	5
3	20	6	5	69	40	20	8
4	20	8	7	70	40	22	11
5	20	10	9	71	40	24	14
...	...	...	...	72	40	26	17

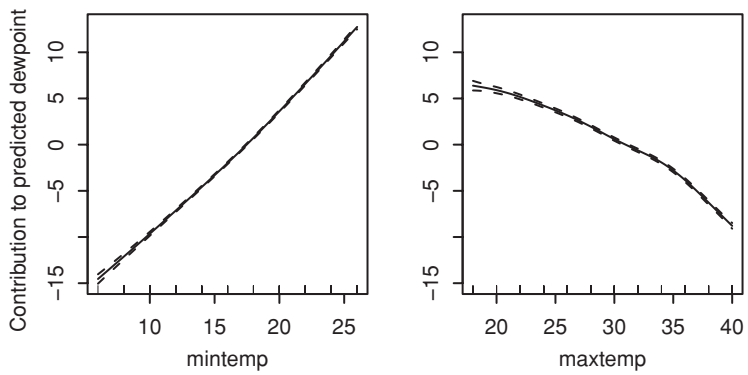


Figure 7.11 Representation of average dewpoint (`dewpt`) as the sum of an effect due to minimum temperature (`mintemp`), and an effect due to maximum temperature (`maxtemp`). (Data are from Table 7.8.) The dashed lines are 95% pointwise confidence bounds.

7.6 Smoothing with multiple explanatory variables

Attention will now move to models with multiple explanatory variables. Table 7.8 has data on monthly averages of minimum temperature, maximum temperature, and dewpoint. For the background to these data, see Linacre (1992), Linacre and Geerts (1997). The dewpoint is the maximum temperature at which the relative humidity reaches 100%. Monthly data were obtained for a large number of sites worldwide. For each combination of minimum and maximum temperature, the average dewpoint was then determined.

7.6.1 An additive model with two smooth terms

Figure 7.11 shows a representation of these data using an additive model with two spline-smoothing terms. A simplified version of the code used for the fit and for the graph is:

```
## Regression of dewpt vs maxtemp: data frame dewpoint (DAAG)
library(mgcv)
ds.gam <- lm(dewpt ~ s(mintemp) + s(maxtemp), data=dewpoint)
oldpar <- par(mfrow = c(1,2), pty="s")
```

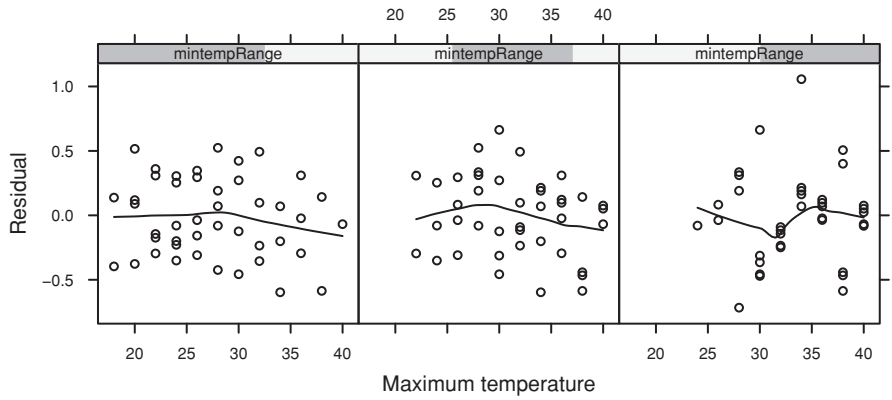


Figure 7.12 Plot of residuals against maximum temperature, for three different ranges of values of minimum temperature. Panel strips are shaded to show the range of values of the conditioning variable.

```
plot(ds.gam, se=2)    # se=2: Show 2SE limits
## Try also: plot(ds.gam, se=2, residuals=TRUE, pch=1, cex=0.4)
par(oldpar)
```

We can write the model as

$$y = \mu + f_1(x_1) + f_2(x_2) + \varepsilon$$

where $y = \texttt{dewpt}$, $x_1 = \texttt{maxtemp}$, and $x_2 = \texttt{mintemp}$.

Here μ is estimated by the mean of y, so that the estimates of $f_1(x_1)$ and $f_2(x_2)$ give differences from this overall mean. In Figure 7.11, both $f_1(x_1)$ and $f_2(x_2)$ are modeled by spline functions with five degrees of freedom. The left panel is a plot of the estimate of $f_1(x_1)$ against x_1, while the right panel plots $f_2(x_2)$ against x_2.

There is no obvious reason why the additive model should work so well. In general, we might expect an interaction term, i.e., we might expect that $f_1(x_1)$ would be different for different values of x_2, or equivalently that $f_2(x_2)$ would be different for different values of x_1. Even where the effects are not additive, an additive model is often a good starting approximation. We can fit the additive model, and then check whether there are departures from it that require examination of the dependence of y upon x_1 and x_2 jointly.

One check is to take, e.g., for $x_1 = \texttt{mintemp}$, three perhaps overlapping ranges of values, which we might call "low", "medium", and "high". For this purpose we are then treating `mintemp` as a *conditioning* variable. We then plot residuals against $5u = \texttt{maxtemp}$ for each range of values of x_1, as in Figure 7.12. If there is a pattern in these plots that changes with the range of the conditioning variable, this is an indication that there are non-additive effects that require attention.[8]

[8]
```
## Residuals vs maxtemp, for different mintemp ranges
library(lattice)
mintempRange <- equal.count(dewpoint$mintemp, number=3)
xyplot(residuals(ds.lm) ~ maxtemp | mintempRange, data=dewpoint, aspect=1,
       layout=c(3,1), type=c("p","smooth"),
       xlab="Maximum temperature", ylab="Residual")
```

7.6.2 A smooth surface*

If it is suspected that the additive model is inappropriate, an alternative is to use a thin plate regression spline basis for `mintemp` and `maxtemp` jointly. For this, specify:

```
ds.tp <- gam(dewpt ~ s(mintemp, maxtemp), data=dewpoint)
vis.gam(ds.tp, plot.type="contour")  # gives a contour plot of the
                                     # fitted regression surface
vis.gam(ds.gam, plot.type="contour") # cf. model with 2 smooth terms
```

Three-dimensional perspective plots can also be obtained with the argument `plot.type="persp"`.

7.7 Further reading

There is a review of the methodologies we have described, and of extensions, in Venables and Ripley (2002). See also references on the help pages for functions in the *locfit* and *mgcv* packages. Eubank (1999) gives a comprehensive and readable introduction to the use of splines in non-parametric regression. Maindonald (1984) has an elementary introduction to B-splines, starting with piecewise linear functions. Faraway (2006) is a wide-ranging and practically oriented account that starts with Generalized Linear Models. Wood (2006) is even more wide-ranging, with stronger technical demands. It has extensive coverage of Linear Models and Generalized Linear Models, then proceeding to an account of Generalized Additive Models and Generalized Additive Mixed Models that makes heavy use of various forms of spline bases.

References for further reading

Eubank, R. L. 1999. *Nonparametric Regression and Spline Smoothing,* 2nd edn.

Faraway, J. J. 2006. *Extending the Linear Model with R. Generalized Linear, Mixed Effects and Nonparametric Regression Models.*

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd edn.

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

Wood, S. N. 2006. *Generalized Additive Models. An Introduction with R.*

7.8 Exercises

1. Reanalyze the sugar weight data of Subsection 7.1.1 using `log(weight)` in place of `weight`.
2. Use `anova()` to compare the two models:
   ```
   roller.lm <- lm(depression~weight, data=roller)
   roller.lm2 <- lm(depression~weight+I(weight^2), data=roller)
   ```
   Is there any justification for including the squared term?
3. Use the method of Section 7.3 to compare, formally, the regression lines for the two data frames `elastic1` and `elastic2` from Exercise 1 in Chapter 5.

4. The data frame `toycars` consists of 27 observations on the distance (in meters) traveled by one of three different toy cars on a smooth surface, starting from rest at the top of a 16-inch-long ramp tilted at varying angles (measured in degrees). Because of differing frictional effects for the three different cars, we seek three regression lines that relate distance traveled to angle.

   (a) As a first try, fit three lines that have the same slope but different intercepts.
   (b) Note the value of R^2 from the summary table. Examine the diagnostic plots carefully. Is there an influential outlier? How should it be treated?
   (c) The physics of the problem actually suggests that the three lines should have the same intercept (very close to 0, in fact), and possibly differing slopes, where the slopes are inversely related to the coefficient of dynamic friction for each car. Fit the model, and note that the value of R^2 is slightly lower than that for the previously fitted model. Examine the diagnostic plots. What has happened to the influential outlier? In fact, this is an example where it is inadvisable to take R^2 too seriously; in this case, a more carefully considered model can accommodate all of the data satisfactorily. Maximizing R^2 does not necessarily give the best model!

5. The data frame `cuckoos` holds data on the lengths and breadths of eggs of cuckoos, found in the nests of six different species of host birds. Fit models for the regression of length on breadth that have:

   A: a single line for all six species.
   B: different parallel lines for the different host species.
   C: separate lines for the separate host species.

   Use the `anova()` function to print out the sequential analysis of variance table. Which of the three models is preferred? Print out the diagnostic plots for this model. Do they show anything worthy of note? Examine the output coefficients from this model carefully, and decide whether the results seem grouped by host species. How might the results be summarized for reporting purposes?

6. Fit the three models A, B and C from the previous exercise, but now using the robust regression function `rlm()` from the *MASS* package. Do the diagnostic plots look any different from those from the output from `lm()`? Is there any substantial change in the regression coefficients?

7. Apply polynomial regression to the seismic timing data in the data frame `geophones`. Specifically, check the fits of linear, quadratic, cubic, and quartic (degree $= 4$) polynomial estimates of the expected thickness as a function of distance. What do you observe about the fitted quartic curve? Do any of the fitted curves capture the curvature of the data in the region where `distance` is large?

8. Apply spline regression to the `geophones` data frame. Specifically, regress thickness against distance, and check the fits of 4-, 5- and 6-degree-of-freedom cases. Which case gives the best fit to the data? How does this fitted curve compare with the polynomial curves obtained in the previous exercise? Calculate pointwise confidence bounds for the 5-degree-of-freedom case.

9. In the data frame `worldRecords` (*DAAG*): (i) fit `log(Time)` as a linear function of `log(Distance)`; (ii) fit `log(Time)` as a polynomial of degree 4 in `log(Distance)`; (iii) fit `log(Time)` as a natural spline function of degree 4 in `log(Distance)`.

   (a) Use `anova()` to compare the fits (i) and (ii).
   (b) Compare the R^2 statistics from the fits (i), (ii) and (iii). Do they convey useful information about the adequacy of the models?

(c) For each of (i), (ii) and (iii), plot residuals against log(*Distance*). Which model best accounts for the pattern of change of time with log(*Distance*)? For what range(s) of distances does there seem, for all three models, to be some apparent residual bias?

10. Apply `lowess()` to the `geophones` data as in the previous two exercises. You will need to experiment with the `f` argument, since the default value oversmooths this data. Small values of `f` (less than 0.2) give a very rough plot, while larger values give a smoother plot. A value of about 0.25 seems a good compromise.

11. Check the diagnostic plots for the results of Exercise 8 for the 5-degree-of-freedom case. Are there any influential outliers?

12. Continuing to refer to Exercise 8, obtain plots of the spline basis curves for the 5-degree-of-freedom case. That is, plot the relevant column of the model matrix against y.

13. Apply the penalized spline to the `geophones` data using the default arguments in *mgcv*'s `s()` function. Is this a satisfactory fit to the data? The argument `k` controls the number of knots. Try setting `k` to 20, and examine the fit. How might an "optimal" value of `k` be selected?

14. The `ozone` data frame holds data, for nine months only, on ozone levels at the Halley Bay station between 1956 and 2000. (See Christie (2000), Shanklin (2001) for the scientific background.) Up-to-date data are available from the web site given under `help(ozone, package="DAAG")`. Replace zeros by missing values. Determine, for each month, the number of missing values. Plot the October levels against Year, and fit a smooth curve. At what point does there seem to be clear evidence of a decline? Plot the data for other months also. Do other months show a similar pattern of decline?

15. The `wages1833` data frame holds data on the wages of Lancashire cotton factory workers in 1833. Plot male wages against age and fit a smooth curve. Repeat using the numbers of male workers as weights. Do the two curves seem noticeably different? Repeat the exercise for female workers. [See Boot and Maindonald (2008) for background information on these data.]

16. Clutton-Brock *et al.* (1999) studied how the time that adult meerkats spent on guarding varied with the size of the group. (Studies were conducted in the Kalahari Gemsbok Park in South Africa.) Approximate percentages of time were:
Group size 1: 50,47; 2: 26; 3: 26; 4: 24,23; 5: 19; 6: 13; 7: 3.
(NB: These numbers were read off from a graph.)
Model the percentage of time as a function of group size.

17. From the data set `cricketers` extract the subset for which `year` (year of birth) is in the range 1840 to 1960, inclusive. Fit the following:

    (a) A polynomial of degree 2 in year.
    (b) A polynomial of degree 3 in year.

    Plot the fitted curves on a graph of proportion left-handed versus year of birth. Does the polynomial of degree 3 give any worthwhile improvement over a polynomial of degree 2? Compare also with a regression B-spline of degree 3; i.e., `bs(year, 3)`.

18.* Compare the two results:

```
seedrates.lm <- lm(grain ~ rate + I(rate^2), data=seedrates)
seedrates.pol <- lm(grain ~ poly(rate,2), data=seedrates)
```

Check that the fitted values and residuals from the two calculations are the same, and that the t-statistic and p-value are the same for the coefficient labeled `poly(rate, 2)2` in the

polynomial regression as for the coefficient labeled `I(rate^2)` in the regression on `rate` and `rate^2`.

Check that the coefficients remain the same if, in the calculation of `seedrates.lm`, `rate + I(rate^2)` is replaced by `poly(rate, 2, raw=TRUE)`.

Regress the second column of `model.matrix(seedrates.pol)` on `rate` and `I(rate^2)`, and similarly for the third column of `model.matrix(seedrates.pol)`. Hence, express the first and second orthogonal polynomial terms as functions of `rate` and `rate^2`.

19.* The following fits a `gam` model to data that have a strong sequential correlation (see Section 9.1 for the basic time series concepts assumed in this exercise):

```
library(mgcv)
xy <- data.frame(x=1:200, y=arima.sim(list(ar=0.75), n=200))
df.gam <- gam(y ~ s(x), data=xy)
plot(df.gam, residuals=TRUE)
```

(a) Run the code several times. (Be sure, on each occasion, to simulate a new data frame `xy`.) Is the function `gam()` overfitting? What is overfitting in this context? Compare with the result from re-running the code with `ar=0`.

(b) Repeat, now with `ar=-0.75` in the code that generates the sequentially correlated series. Why is the result so very different?

8

Generalized linear models and survival analysis

The straight line regression model we considered in Chapter 5 had the form

$$y = \alpha + \beta x + \varepsilon$$

where, if we were especially careful, we would add a subscript i to each of y, x, and ε. In this chapter, we will resume with models where there is just one x, as in Chapter 5, in order to keep the initial discussion simple. Later, we will add more predictor variables, as required.

The regression model can be written

$$\mathrm{E}[y] = \alpha + \beta x$$

where E is *expectation*. This form of the equation is a convenient point of departure for moving to generalized linear models, abbreviated to GLMs. This class of models, first introduced in the 1970s, gives a unified theoretical and computational approach to models that had previously been treated as distinct. They have been a powerful addition to the data analyst's armory of statistical tools.

The present chapter will limit attention to a few important special cases. The chapter will end with a discussion of survival methods. Survival methods, while having important theoretical connections with generalized linear models, require a distinct theoretical and computational treatment.

8.1 Generalized linear models

Generalized linear models (GLMs) differ in two ways from the models used in earlier chapters. They allow a more general form of expression for the expectation, and they allow various types of non-normal error terms. Logistic regression models are perhaps the most widely used GLM.

8.1.1 Transformation of the expected value on the left

GLMs allow a transformation $f()$ to the left-hand side of the regression equation, i.e., to $\mathrm{E}[y]$. The result specifies a linear relation with x. In other words,

$$f(\mathrm{E}[y]) = \alpha + \beta x$$

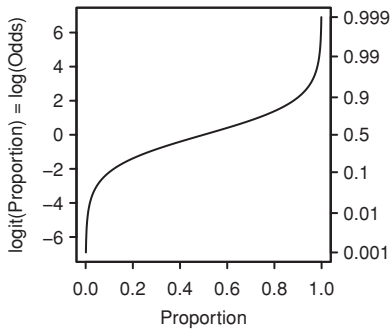


Figure 8.1 The logit or log(odds) transformation. Shown here is a plot of log(odds) versus proportion. Notice how the range is stretched out at both ends.

where $f()$ is a function, which is usually called the *link* function. In the fitted model, we call $\alpha + \beta x$ the linear predictor, while $\mathrm{E}[y]$ is the expected value of the response. The function $f()$ transforms from the scale of the response to the scale of the linear predictor.

Some common examples of link functions are: $f(x) = x$, $f(x) = 1/x$, $f(x) = \log(x)$, and $f(x) = \log(x/(1-x))$. The last, shown in Figure 8.1, is the logit link that is the link function for logistic regression.[1] Observe that these functions are all monotonic, i.e., they increase or (in the case of $1/x$) decrease with increasing values of x.

8.1.2 Noise terms need not be normal

We may write

$$y = \mathrm{E}[y] + \varepsilon.$$

Here, the elements of y may have a distribution different from the normal. Common distributions are the binomial where y is the number responding out of a given total n, and the Poisson where y is a count.

Even more common may be models where the random component differs from the binomial or Poisson by having a variance that is larger than the mean. The analysis proceeds as though the distribution were binomial or Poisson, but the theoretical binomial or Poisson variance estimates are replaced by a variance that is estimated from the data. Such models are called, respectively, quasi-binomial models and quasi-Poisson models.

8.1.3 Log odds in contingency tables

With proportions that range from less than 0.1 to greater than 0.9, it is not reasonable to expect that the expected proportion will be a linear function of x. A transformation (link function) such as the logit is required. A good way to think about logit models is that they

[1]
```
## Simplified plot showing the logit link function
p <- (1:999)/1000
gitp <- log(p/(1 - p))
plot(p, gitp, xlab = "Proportion", ylab = "", type = "l", pch = 1)
```

Table 8.1 *Terminology used for logistic regression (or more generally for generalized linear models), compared with multiple regression terminology.*

Regression	Logistic regression
Degrees of freedom	Degrees of freedom
Sum of squares (SS)	Deviance (D)
Mean sum of squares (divide by degrees of freedom)	Mean deviance (divide by degrees of freedom)
Fit models by minimizing the residual sum of squares.	Fit models by minimizing the deviance.

work on a log(odds) scale. If p is a probability (e.g., that horse A will win the race), then the corresponding odds are $p/(1-p)$, and

$$\log(\text{odds}) = \log(p/(1-p)) = \log(p) - \log(1-p).$$

Logistic regression provides a framework for analyzing contingency table data. Let us now recall the fictitious admissions data presented in Table 4.10. The observed proportion of students (male and female) admitted into Engineering is $40/80 = 0.5$. For Sociology, the admission proportion is $15/60 = 0.25$. Thus, we have

$$\log(\text{odds}) = \log(0.5/0.5) = 0 \text{ for Engineering},$$

$$\log(\text{odds}) = \log(0.75/0.25) = 1.0986 \text{ for Sociology}.$$

What determines whether a student will be admitted to Engineering? What determines whether a student will be admitted to Sociology? Is age a factor? Logistic regression allows us to model log(odds of admission) as a function of age, or as a function of any other predictor that we may wish to investigate.

For such data, we may write

$$\log(\text{odds}) = \text{constant} + \text{effect due to faculty} + \text{effect due to gender}.$$

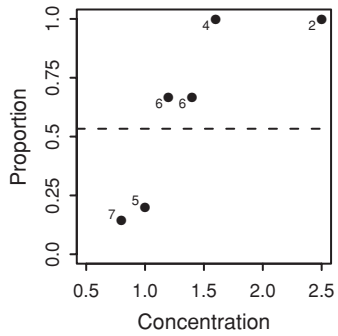
This now has the form of a linear model.

8.1.4 Logistic regression with a continuous explanatory variable

The likelihood is the joint probability of the observed data values, given the model parameters. It is thus a function of the model parameters. The deviance is minus twice the logarithm of the likelihood. Maximizing the likelihood is equivalent to minimizing the deviance.

The fitting of the logistic model is accomplished by minimizing *deviances*. A deviance has a role very similar to a sum of squares in regression (in fact, if the data are normally distributed, the two quantities are equivalent). This aspect of the analogy between regression and logistic regression is furnished by Table 8.1.

Data for the example that now follows are in the data frame `anesthetic` (*DAAG*). Thirty patients were given an anesthetic agent that was maintained at a predetermined (alveolar) concentration for 15 minutes before making an incision. It was then noted whether the patient moved, i.e., jerked or twisted. The interest is in estimating how the



	Value of nomove		
conc	0 (move)	1 (no move)	prop
0.8	6	1	0.17
1	4	1	0.20
1.2	2	4	0.67
1.4	2	4	0.67
1.6	0	4	1.00
2.5	0	2	1.00

Figure 8.2 Plot, versus concentration, of proportion of patients not moving, for each of six different alveolar concentrations. The horizontal line is the proportion of no-moves over the data as a whole. Data are displayed to the right of the plot.

probability of jerking or twisting varies with increasing concentration of the anesthetic agent.

We take the response as nomove, because the proportion then increases with increasing concentration. The totals and proportions, for each of the six concentrations, can be calculated thus:

```
library(DAAG)
anestot <- aggregate(anesthetic[, c("move","nomove")],
                     by=list(conc=anesthetic$conc), FUN=sum)
## The column 'conc', because from the 'by' list, is then a factor.
## The next line recovers the numeric values
anestot$conc <- as.numeric(as.character(anestot$conc))
anestot$total <- apply(anestot[, c("move","nomove")], 1 , sum)
anestot$prop <- anestot$nomove/anestot$total
```

Figure 8.2 plots the proportions. The table that is shown to the right of Figure 8.2 gives the information in the data frame anestot that has just been calculated. It gives, for each concentration, the respective numbers with nomove equal to 0 (i.e., movement) and nomove equal to 1 (i.e., no movement).[2]

We can fit the logit model either directly to the 0/1 data, or to the proportions in the table that appears to the right of Figure 8.2:

```
## Fit model directly to the 0/1 data in nomove
anes.logit <- glm(nomove ~ conc, family=binomial(link="logit"),
                  data=anesthetic)
## Fit model to the proportions; supply total numbers as weights
anes1.logit <- glm(prop ~ conc, family=binomial(link="logit"),
                   weights=total, data=anestot)
```

[2] ## Plot proportion moving vs conc: data frame anesthetic (DAAG)
plot(prop ~ conc, data=anestot, xlab = "Concentration",
ylab = "Proportion", xlim = c(.5,2.5), ylim = c(0, 1), pch = 16)
with(anestot,
{text(conc, prop, paste(total), pos=2)
abline(h=sum(nomove)/sum(total), lty=2)})

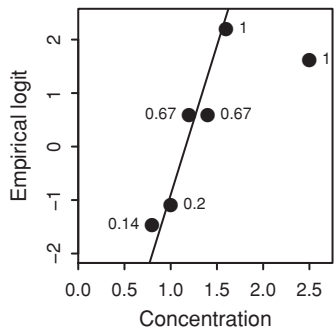


Figure 8.3 Plot, on an empirical logit scale, of log(odds) = logit(proportion) of patients not moving, versus concentration. If x is the number of patients not moving, and n is the total, then the empirical logit is $\log(\frac{x+0.5}{n-x+0.5})$. Notice that the order changes from the order of the untransformed proportions. Labels on the points show the observed proportions. The line gives the estimate of the proportion of moves, based on the fitted logit model.

The analysis assumes that individuals respond independently with a probability, estimated from the data, that on a logistic scale is a linear function of the concentration. For any fixed concentration, the assumption is that we have Bernoulli trials, i.e., that individual responses are drawn at random from the same population.

Output from the `summary()` function (see also Figure 8.3[3]) is:

```
> summary(anes.logit)
. . . .
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.47       2.42   -2.67   0.0075
conc            5.57       2.04    2.72   0.0064

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 41.455  on 29  degrees of freedom
Residual deviance: 27.754  on 28  degrees of freedom
AIC: 31.75

Number of Fisher Scoring iterations: 5
```

With such a small sample size, a convincing check of the adequacy of the model is impossible.

[3]
```
## Graphical summary of logistic regression results
anestot$emplogit <- with(anestot, log((nomove+0.5)/(move+0.5)))
plot(emplogit ~ conc, data=anestot,
     xlab = "Concentration", xlim = c(0, 2.75), xaxs="i",
     ylab = "Empirical logit", ylim=c(-2, 2.4), cex=1.5, pch = 16)
with(anestot, text(conc, emplogit, paste(round(prop,2)), pos=c(2,4,2,4,4,4)))
abline(anes.logit)
```

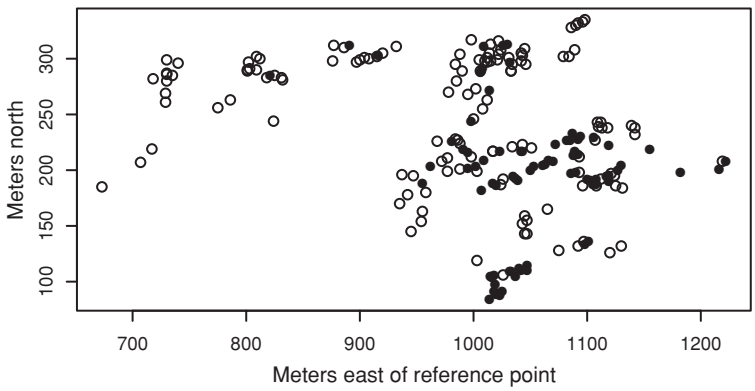


Figure 8.4 Location of sites, relative to reference point, that were examined for frogs. The sites are all in the Snowy Mountains region of New South Wales, Australia. The filled points are for sites where frogs were found.

8.2 Logistic multiple regression

We now consider data on the distribution of the Southern Corroboree frog, that occurs in the Snowy Mountains area of New South Wales, Australia (data are from Hunter, 2000). In all, 212 sites were surveyed. In Figure 8.4 filled circles denote sites where the frogs were found; open circles denote the absence of frogs.[4]

The variables in the data set are `pres.abs` (were frogs found?), `easting` (reference point), `northing` (reference point), `altitude` (in meters), `distance` (distance in meters to nearest extant population), `NoOfPools` (number of potential breeding pools), `NoOfSites` (number of potential breeding sites within a 2 km radius), `avrain` (mean rainfall for Spring period), `meanmin` (mean minimum Spring temperature), and `meanmax` (mean maximum Spring temperature).

As with multiple linear regression, a desirable first step is to check relationships among explanatory variables. Where possible, we will transform so that those relationships are made linear, as far as we can tell from the scatterplot matrix. If we can so transform the variables, this gives, as noted earlier in connection with classical multiple regression, access to a theory that can be highly helpful in guiding the process of regression modeling.

We wish to explain frog distribution as a function of the other variables. Because we are working within a very restricted geographic area, we do not expect that the distribution will change as a function of latitude and longitude *per se*, so that `easting` and `northing` will not be used as explanatory variables. Figure 8.5 shows the scatterplot matrix for the remaining explanatory variables.[5]

Notice that the relationships between `altitude`, `avrain`, `meanmin`, and `meanmax` are close to linear. For the remaining variables, the distributions are severely skewed, to the extent that it is difficult to tell whether the relationship is linear. For a distance, the logarithmic transformation is often reasonable. Two of the variables are counts, so that

[4]
```
## Presence/absence information: data frame frogs (DAAGS)
plot(northing ~ easting, data=frogs, pch=c(1,16)[frogs$pres.abs+1],
     xlab="Meters east of reference point", ylab="Meters north")
```
[5]
```
## Pairs plot; frogs data
pairs(frogs[, c(5:10,4)], oma=c(2,2,2,2), cex=0.5)
```



Figure 8.5 Scatterplot matrix for predictor variables in the `frogs` data set.

a square root transformation might be appropriate. Figure 8.6 shows three density plots for each of these variables: untransformed, after a square root, and after a logarithmic transformation.[6] For `NoOfSites`, a large number of values of `NoOfSites` are zero. Hence we plot log(`NoOfSites`+1).

For `distance` and `NoOfPools`, we prefer a logarithmic transformation, while `NoOfSites` is best not transformed at all.

Figure 8.7 makes it clear that `altitude` and `meanmax`, with a correlation of -0.9966, tell essentially the same story.[7]

[6]
```
## Here is code for the top row of plots
par(mfrow=c(3,3))
for(nam in c("distance","NoOfPools","NoOfSites")){
    y <- frogs[, nam]
    plot(density(y), main="", xlab=nam)
    }
  # The other rows can be obtained by replacing y by
  # sqrt(y) or log(y) (or, for NoOfSites, by log(y+1))
par(mfrow=c(1,1))
```

[7]
```
## Correlation between altitude and meanmax
with(frogs, cor(altitude,meanmax))
```

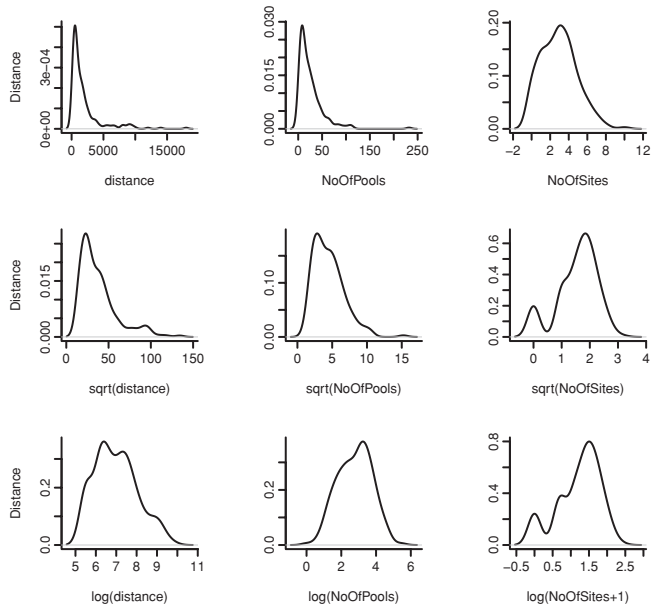


Figure 8.6 Density plots for distance (`distance`), number of pools (`NoOfPools`), and number of sites (`NoOfSites`), before and after transformation.

Rather than working with `meanmin` and `meanmax`, it is in principle better to work with `meanmax-meanmin` and `meanmax+meanmin` (or `0.5*(meanmax+meanmin)`), which are much less strongly correlated. Here is the correlation matrix:

```
> with(frogs, cor(cbind(altitude,
+                       "meanmax+meanmin" = meanmin+meanmax,
+                       "meanmax-meanmin" = meanmax-meanmin)))
                altitude meanmax+meanmin meanmax-meanmin
altitude          1.0000         -0.9933         -0.9095
meanmax+meanmin  -0.9933          1.0000          0.8730
meanmax-meanmin  -0.9095          0.8730          1.0000
```

The variance inflation factor (VIF) for `altitude` and for `meanmax+meanmin` will be at least $(1 - 0.9933^2)^{-1} \simeq 74.9$ (the squared correlation between, e.g., `altitude` and any one other explanatory variable is a lower bound for the squared multiple correlation with all other explanatory variables). It is pointless to include both of these as explanatory variables; the standard errrors will be so large as to render the calculated coefficients meaningless.

We now (using Figure 8.7) investigate the scatterplot matrix for the variables that are transformed as above. We replace `meanmin` and `meanmax` by `meanmax-meanmin` and `meanmax+meanmin`.[8]

[8]
```
with(frogs,
   pairs(cbind(log(distance), log(NoOfPools), NoOfSites, avrain,
            altitude, meanmax+meanmin, meanmax-meanmin), col="gray",
         labels=c("log(distance)", "log(NoOfPools)", "NoOfSites",
                  "Av. rainfall", "altitude", "meanmax+meanmin",
                   "meanmax-meanmin"),
         panel=panel.smooth))
```

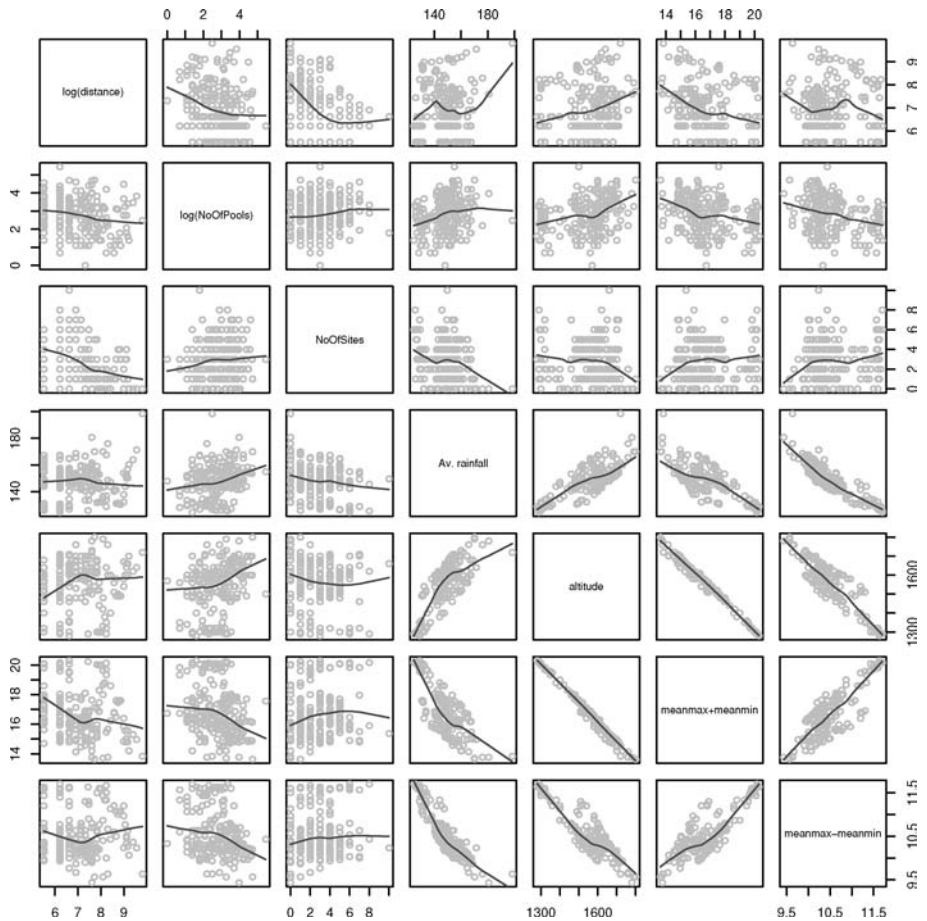


Figure 8.7 Scatterplot matrix for altitude, transformed versions of `distance` and `NoOfPools`, and `NoOfSites`. We are particularly interested in whether the relationship with `altitude` is plausibly linear. If the relationship with `altitude` is linear, then it will be close to linear also with the temperature and rain variables.

The smoothing curves for log(`distance`) and log(`NoOfPools`) are nearly linear. There appears to be some non-linearity associated with `NoOfSites`, so that we might consider including `NoOfSites2` (the square of `NoOfSites`) as well as `NoOfSites` in the regression equation.

8.2.1 Selection of model terms, and fitting the model

For prediction, `meanmax+meanmin` may be preferable to `altitude`. A change in temperature at a given altitude may well influence the choice of sites.

We try the simpler model (without `NoOfSites2`) first:

```
> summary(frogs.glm0 <- glm(formula = pres.abs ~ log(distance) +
+                           log(NoOfPools) + NoOfSites + avrain +
+                           I(meanmax+meanmin)+I(meanmax-meanmin),
```

```
+                           family = binomial, data = frogs))
. . . .
Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           18.26890   16.13819    1.13   0.2576
log(distance)         -0.75832    0.25581   -2.96   0.0030
log(NoOfPools)         0.57090    0.21533    2.65   0.0080
NoOfSites             -0.00362    0.10615   -0.03   0.9728
avrain                 0.00070    0.04117    0.02   0.9864
I(meanmax + meanmin)   1.49581    0.31532    4.74  2.1e-06
I(meanmax - meanmin)  -3.85827    1.27839   -3.02   0.0025

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 279.99  on 211  degrees of freedom
Residual deviance: 197.65  on 205  degrees of freedom
AIC: 211.7

Number of Fisher Scoring iterations: 5
```

Because we have taken no account of spatial clustering, and because there are questions about the adequacy of the asymptotic approximation that is required for their calculation, the p-values should be used with caution. We do however note the clear separation of the predictor variables into two groups – in one group the p-values are 0.5 or more, while for those in the other group the p-values are all very small.

Note that `NoOfSites` has a coefficient with a large p-value. Replacing it with `NoOfSites2` makes little difference. We can almost certainly, without loss of predictive power, omit `NoOfSites` and `avrain`. The regression equation becomes:

```
> frogs.glm <- glm(pres.abs ~ log(distance) + log(NoOfPools) +
+                  I(meanmax+meanmin)+ I(meanmax-meanmin),
+                  family = binomial, data = frogs)
> summary(frogs.glm)
. . . .
Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)             18.527      5.267    3.52  0.00044
log(distance)           -0.755      0.226   -3.34  0.00084
log(NoOfPools)           0.571      0.215    2.65  0.00800
I(meanmax + meanmin)     1.498      0.309    4.85  1.2e-06
I(meanmax - meanmin)    -3.881      0.900   -4.31  1.6e-05

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 279.99  on 211  degrees of freedom
Residual deviance: 197.66  on 207  degrees of freedom
AIC: 207.7

Number of Fisher Scoring iterations: 5
```

The residual deviance is almost unchanged. The coefficients that are retained have scarecely changed. Thus, conditioning on the variables that have been left out of the regression equation makes no difference of consequence to the estimates of the effects of variables that have been included.

8.2.2 Fitted values

The function `fitted()` calculates fitted values that are on the scale of the response. The values that are returned by the linear predictor are back-transformed to give the fitted values. For logistic regression, the link function is the logit, i.e.:

$$f(x) = \log\left(\frac{x}{1-x}\right), \text{ where } 0 < x < 1.$$

The inverse of the logit, used for the back transformation, is:

$$g(u) = \frac{\exp(u)}{1+\exp(u)}.$$

An alternative is to calculate fitted values on the scale of the linear predictor. The default action of the function `predict()` is to return fitted values on the scale of the linear predictor. The possibilities are:

```
fitted(frogs.glm)    # Fitted values' scale of response
predict(frogs.glm, type="response")  # Same as fitted(frogs.glm)
predict(frogs.glm, type="link")      # Scale of linear predictor
## For approximate SEs, specify
predict(frogs.glm, type="link", se.fit=TRUE)
```

Plate 3 allows a comparison of model predictions with observed occurrences of frogs, with respect to geographical co-ordinates. Because our modeling has taken no account of spatial correlation, examination of such a plot is more than ordinarily desirable. We should be concerned if we see any patterns in the spatial arrangement of points where frogs were predicted with high probability but few frogs were found; or where frogs were not expected but appeared with some frequency. There is little overt evidence of such clusters.

For models with a binary outcome, plots of residuals are not, in general, very useful. At each point the residual is obtained by subtracting the fitted value from either 0 or 1. Given the fitted model, the residual can take just one of two possible values. For the `frogs.glm` model it makes sense, as noted above, to look for patterns in the spatial arrangement of points where there seems a discrepancy between observations of frogs and predicted probabilities of occurrence.

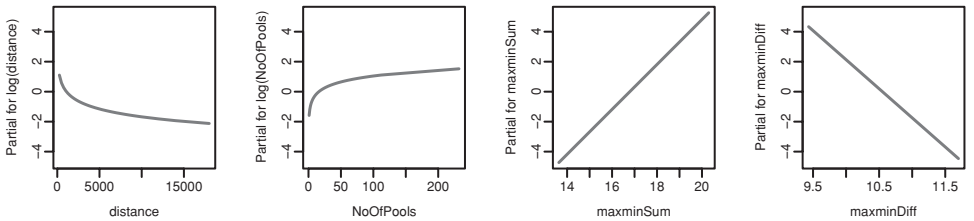


Figure 8.8 Plots showing the contributions of the explanatory variables to the fitted values, on the scale of the linear predictor.

8.2.3 A plot of contributions of explanatory variables

The equation that we have just fitted adds together the effects of `log(distance)`, `log(NoOfPools)`, `meanmin`, and `meanmax`. Figure 8.8 shows the change due to each of these variables in turn, when other terms are held at their means. The code is:[9]

```
par(mfrow=c(1,4), pty="s")
frogs$maxminSum <- with(frogs, meanmax+meanmin)
frogs$maxminDiff <- with(frogs, meanmax-meanmin)
frogs.glm <- glm(pres.abs ~ log(distance) + log(NoOfPools) +
                  maxminSum + maxminDiff, family = binomial,
                  data = frogs)
termplot(frogs.glm)
par(mfrow=c(1,1))
```

The y-scale on these plots is that of the linear predictor. Under the logit link, zero corresponds to a probability of 0.5. Probabilities for other values are -4: 0.02, -2: 0.12, 2: 0.88, and 4: 0.98. The ranges on the vertical scale indicate that easily the biggest effects are for `meanmax+meanmin` and `meanmax-meanmin`. A high value of `meanmax+meanmin` greatly increases the probability of finding frogs, while a large difference `meanmax-meanmin` greatly reduces that probability. The contributions from log(`distance`) and log(`NoOfPools`) are much smaller.

8.2.4 Cross-validation estimates of predictive accuracy

Our function `CVbinary()` calculates cross-validation estimates of predictive accuracy for these models. Folds are numbered according to the part of the data that is used, at that fold, for testing. The resubstitution measure of accuracy makes predictions for the data used to derive the model, and calculates the proportion of correct predictions. By the time the cross-validation calculations are complete, each observation has been a member of a test set at one of the folds only; thus predictions are available for all observations that are derived independently of those observations. The cross-validation measure is the proportion that are correct:

[9] `## For binary response data, plots of residuals are not insightful`
`## To see what they look like, try:`
`termplot(frogs.glm, data=frogs, partial.resid=TRUE)`

```
> CVbinary(frogs.glm0)

Fold:  5 1 3 8 7 10 2 4 9 6
Training data estimate of accuracy = 0.778
Cross-validation estimate of accuracy = 0.769
> CVbinary(frogs.glm)

Fold:  8 5 6 10 7 4 9 1 2 3
Training data estimate of accuracy = 0.778
Cross-validation estimate of accuracy = 0.774
```

The training estimate assesses the accuracy of prediction for the data used in deriving the model. Cross-validation estimates the accuracy that might be expected for a new sample.

The cross-validation estimates can be highly variable; it is best to run the cross-validation routine a large number of times and to make comparisons in as accurate a manner as possible. When assessing the appropriateness of the proposed model for the `frogs` data, we computed four separate cross-validation estimates of prediction accuracy using the initial model, and we repeated these computations independently using the reduced model with variables `log(distance)`, `log(NoOfPools)`, `meanmax+meanmin`, and `meanmax-meanmin`. The accuracy estimates for the initial model came to 78.8%, 77.8%, 77.4%, and 78.8%, while the accuracy estimates for the reduced model were 76.9%, 77.8%, 78.3%, and 77.8%. The variability of these estimates makes it difficult to discern a difference.

A more accurate comparison can be obtained by matching the cross-validation runs for each model. We do this by making a random assignment of observations to folds before we call the cross-validation function, first with the full model and then with the reduced model. The first four cross-validation comparisons are as follows:[10]

```
Initial: 0.783   Reduced: 0.783
Initial: 0.774   Reduced: 0.774
Initial: 0.774   Reduced: 0.774
Initial: 0.774   Reduced: 0.774
```

The pairwise comparisons show no difference.

8.3 Logistic models for categorical data – an example

The array `UCBAdmissions` in the *datasets* package is a 2×2×6 array, with dimensions: `Admit` (Admitted/Rejected) × `Gender` (Male/Female) × `Dept` (A, B, C, D, E, F).

[10]
```
## The cross-validated estimate of accuracy is stored in the list
## element acc.cv, in the output from the function CVbinary(), thus:
for (j in 1:4){
     randsam <- sample(1:10, 212, replace=TRUE)
     initial.acc <- CVbinary(frogs.glm0, rand=randsam,
                             print.details=FALSE)$acc.cv
     reduced.acc <- CVbinary(frogs.glm, rand=randsam,
                             print.details=FALSE)$acc.cv
     cat("Initial:", round(initial.acc,3), "  Reduced:", round(reduced.acc,3), "\n")
     }
```

A first step is to create, from this table, a data frame whose columns hold the information in the observations by variables format that will be required for the use of `glm()`:

```
## Create data frame from multi-way table UCBAdmissions (datasets)
dimnames(UCBAdmissions)  # Check levels of table margins
UCB <- as.data.frame.table(UCBAdmissions["Admitted", , ])
names(UCB)[3] <- "admit"
UCB$reject <- as.data.frame.table(UCBAdmissions["Rejected", , ])$Freq
UCB$Gender <- relevel(UCB$Gender, ref="Male")
## Add further columns total and p (proportion admitted)
UCB$total <- UCB$admit + UCB$reject
UCB$p <- UCB$admit/UCB$total
```

We use a loglinear model to model the probability of admission of applicants. It is important, for present purposes, to fit `Dept`, thus adjusting for different admission rates in different departments, before fitting `Gender`:

```
UCB.glm <- glm(p ~ Dept*Gender, family=binomial,
               data=UCB, weights=total)
anova(UCB.glm, test="Chisq")
```

The output is:

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: p

Terms added sequentially (first to last)


            Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           11        877
Dept         5      855          6         22  1.2e-182
Gender       1        2          5         20   2.2e-01
Dept:Gender  5       20          0   -2.6e-13   1.1e-03
>
```

After allowance for overall departmental differences in admission rate (but not for the `Dept:Gender` interaction; this is a sequential table), there is no detectable main effect of `Gender`. The significant interaction term suggests that there are department-specific gender biases, which average out to reduce the main effect of `Gender` to close to zero.

We now examine the individual coefficients in the model:

```
> summary(UCB.glm)$coef
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)         0.4921     0.0717   6.859 6.94e-12
DeptB               0.0416     0.1132   0.368 7.13e-01
```

```
DeptC               -1.0276     0.1355  -7.584 3.34e-14
DeptD               -1.1961     0.1264  -9.462 3.02e-21
DeptE               -1.4491     0.1768  -8.196 2.49e-16
DeptF               -3.2619     0.2312 -14.110 3.31e-45
GenderFemale         1.0521     0.2627   4.005 6.20e-05
DeptB:GenderFemale  -0.8321     0.5104  -1.630 1.03e-01
DeptC:GenderFemale  -1.1770     0.2995  -3.929 8.52e-05
DeptD:GenderFemale  -0.9701     0.3026  -3.206 1.35e-03
DeptE:GenderFemale  -1.2523     0.3303  -3.791 1.50e-04
DeptF:GenderFemale  -0.8632     0.4026  -2.144 3.20e-02
```

The first six coefficients relate to overall admission rates, for males, in the six departments. The strongly significant positive coefficient for `GenderFemale` indicates that log(odds) is increased by 1.05, in department A, for females relative to males. In departments C, D, E and F, the log(odds) is reduced for females, relative to males.

8.4 Poisson and quasi-Poisson regression

8.4.1 Data on aberrant crypt foci

The data frame `ACF1` consists of two columns: `count` and `endtime`. The first column contains the counts of simple aberrant crypt foci (ACFs) – these are aberrant aggregations of tube-like structures – in the rectal end of 22 rat colons after administration of a dose of the carcinogen azoxymethane. Each rat was sacrificed after 6, 12 or 18 weeks (recorded in the column `endtime`). For further background information, see McLellan *et al.* (1991).

The argument is that there are a large number of sites where ACFs might occur, in each instance with the same low probability. Because “site" does not have a precise definition, the total number of sites is unknown, but it is clearly large. If we can assume independence between sites, then we might expect a Poisson model, for the total number of ACF sites. The output from fitting the model will include information that indicates whether a Poisson model was, after all, satisfactory. If a Poisson model does not seem satisfactory, then a *quasi-Poisson* model may be a reasonable alternative. A model of a quasi-Poisson type is consistent with clustering in the appearance of ACFs. This might happen because some rats are more prone to ACFs than others, or because ACFs tend to appear in clusters within the same rat.

Figure 8.9 provides plots of the data.[11] A logarithmic scale is appropriate on the x-axis because the Poisson model will use a `log` link. Observe that the counts increase with time. For fitting a Poisson regression model, we use the `poisson` family in the `glm()` function. Momentarily ignoring the apparent quadratic relation, we fit a simple linear model:

```
> summary(ACF.glm0 <- glm(formula = count ~ endtime,
+                         family = poisson, data = ACF1))
. . . .
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
```

[11] `## Plot count vs endtime: data frame ACF1 (DAAG)`
`plot(count ~ endtime, data=ACF1, pch=16, log="x")`

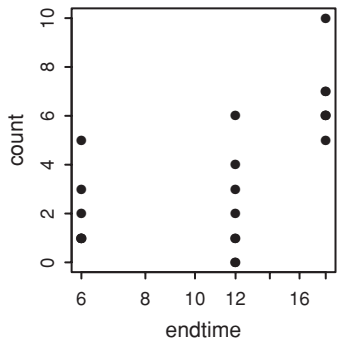


Figure 8.9 Plot of number of simple aberrant crypt foci (`count`) versus `endtime`.

```
(Intercept)  -0.3215     0.4002   -0.80     0.42
endtime       0.1192     0.0264    4.51  6.4e-06

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 51.105  on 21  degrees of freedom
Residual deviance: 28.369  on 20  degrees of freedom
AIC: 92.21

Number of Fisher Scoring iterations: 4
```

We see that the relation between `count` and `endtime` is highly significant. In order to accommodate the apparent quadratic effect, we try adding an `endtime^2` term. The code and output summary is:

```
> summary(ACF.glm <- glm(formula = count ~ endtime + I(endtime^2),
+                        family = poisson, data = ACF1))
. . . .
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.72235    1.09177    1.58    0.115
endtime      -0.26235    0.19950   -1.32    0.188
I(endtime^2)  0.01514    0.00795    1.90    0.057
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 51.105  on 21  degrees of freedom
Residual deviance: 24.515  on 19  degrees of freedom
AIC: 90.35
. . . .
```

The fitted values are:

```
> (etime <- unique(ACF1$endtime))
[1]  6 12 18
> exp(-0.3215 + 0.1192*etime)  # Linear term only
[1] 1.48 3.03 6.20
```

```
> exp(1.72235 - 0.26235*etime + 0.01514*etime^2)  # Quadratic model
[1] 2.00 2.13 6.72
```

These fitted values are alternatively available as `unique(fitted(ACF.glm0))` and `unique(fitted(ACF.glm))`.

Observe that the residual mean deviance is $24.515/19 = 1.29$. For a Poisson model this should, unless a substantial proportion of fitted values are small (e.g., less than about 2), be close to 1. This may be used as an estimate of the *dispersion*, i.e., of the amount by which the variance is increased relative to the expected variance for a Poisson model. A better estimate (it has only a very small bias) is, however, that based on the Pearson chi-squared measure of lack of fit. This can be obtained as follows:

```
> sum(resid(ACF.glm, type="pearson")^2)/19
[1] 1.25
```

This suggests that standard errors should be increased, and t-statistics reduced by $\sqrt{1.25} \simeq 1.12$, relative to those given above for the Poisson model.

The following fits a quasi-Poisson model, which incorporates this adjustment to the variance. As the increased residual variance will take the quadratic term even further from the conventional 5% significance level, we do not bother to fit it:

```
> ACFq.glm <- glm(formula = count ~ endtime,
+                 family = quasipoisson, data = ACF1)
> summary(ACFq.glm)$coef
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.322     0.4553  -0.706 0.488239
endtime        0.119     0.0300   3.968 0.000758
```

Notice that the `z value` has now become a `t value`, suggesting that is should be referred to a t-distribution with 19 degrees of freedom (19 d.f. were available to estimate the dispersion).

There are enough small expected counts that the dispersion estimate and resultant t-statistics should be treated with modest caution. Additionally, there is a question whether the same dispersion estimate should apply to all values of `endtime`. This can be checked with:

```
> sapply(split(residuals(ACFq.glm), ACF1$endtime), var)
    6    12    18
1.096 1.979 0.365
```

Under the assumed model, these should be approximately equal. The differences in variance are, however, nowhere near statistical significance. The following approximate test ignores the small amount of dependence in the residuals:

```
> fligner.test(resid(ACFq.glm) ~ factor(ACF1$endtime))

        Fligner-Killeen test of homogeneity of variances

data:  resid(ACFq.glm) by factor(ACF1$endtime)
Fligner-Killeen:med chi-squared = 2.17, df = 2, p-value = 0.3371
```

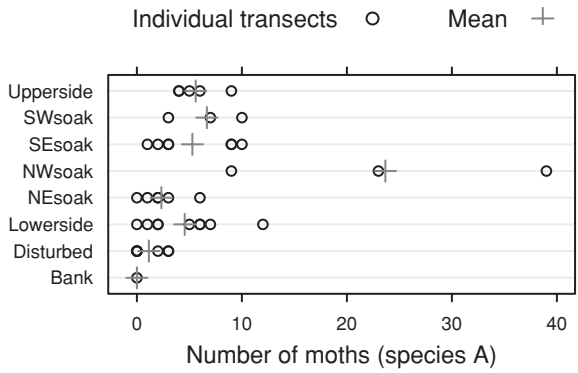


Figure 8.10 Dotplot summary of the numbers of moths of species A, by habitat type.

8.4.2 Moth habitat example

The `moths` data are from a study of the effect of habitat on the densities of two species of moth. Transects were set out across the search area. Within transects, sections were identified according to habitat type. The end result was that there were:

- 41 different lengths (`meters`) of transect ranging from 2 meters to 233 meters,
- grouped into eight habitat types within the search area – `Bank`, `Disturbed`, `Lowerside`, `NEsoak`, `NWsoak`, `SEsoak`, `SWsoak`, `Upperside`,
- with records taken at 20 different times (morning and afternoon) over 16 weeks.

The variables `A` and `P` give the numbers of moths of the two different species. Here is a tabular summary, by habitat:[12]

	Bank	Disturbed	Lowerside	NEsoak	NWsoak	SEsoak	SWsoak	Upperside
Number	1	7	9	6	3	7	3	5
meters	21	49	191	254	65	193	116	952
A	0	8	41	14	71	37	20	28
P	4	33	17	14	19	6	48	8

The `Number` is the total number of transects for that habitat, while `meters` is the total length.

Figure 8.10 gives a visual summary of the data. The code is:

```
library(lattice)
dotplot(habitat ~ A, data=moths, xlab="Number of moths (species A)",
        panel=function(x, y, ...){
          panel.dotplot(x,y, pch=1, col="black", ...)
          panel.average(x, y, pch=3, cex=1.25, type="p", col="gray45")
          },
        key=list(text=list(c("Individual transects", "Mean")),
          points=list(pch=c(1,3), cex=c(1,1.25), col=c("black","gray45")),
          columns=2))
```

[12] `## Number of moths by habitat: data frame moths (DAAG)`
`rbind(Number=table(moths[, 4]), sapply(split(moths[, -4], moths$habitat), apply, 2, sum))`

For data such as these, there is no reason to accept the Poisson distribution assumption that moths appear independently. The `quasipoisson` family, used in place of the `poisson`, will now be the starting point for analysis. The dispersion estimate from the `quasipoisson` model will indicate whether a Poisson distribution assumption might have been adequate.

The model will take the form

$$y = \text{habitat effect} + \beta \log(\text{length of section})$$

where $y = \log(\text{expected number of moths})$.

The use of a logarithmic link function, so that the model works with log(expected number of moths), is the default. With this model, effects are multiplicative. This form of model allows for the possibility that the number of moths per meter might be relatively lower or higher for long than for short sections.

Comparisons are made against a reference (or baseline) factor level. The default reference level determined according to the alphanumeric order of the level names is not always the most useful, so that the `relevel()` function must be used to change it. Before making such a change, let's see what happens if we proceed blindly and use as reference `Bank`, which is the initial level of `habitat` in the data as included in the *DAAG* package.

An unsatisfactory choice of reference level

No moths of the first species (`A`) were found in the `Bank` habitat. This zero count creates problems for the calculation of standard errors, as will now be apparent:

```
> summary(A.glm <- glm(A ~ habitat + log(meters),
+                      family = quasipoisson, data = moths))
. . . .
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -15.696   2096.588   -0.01     0.99
log(meters)         0.129      0.153    0.85     0.40
habitatDisturbed   15.622   2096.588    0.01     0.99
habitatLowerside   16.906   2096.588    0.01     0.99
habitatNEsoak      16.084   2096.588    0.01     0.99
habitatNWsoak      18.468   2096.588    0.01     0.99
habitatSEsoak      16.968   2096.588    0.01     0.99
habitatSWsoak      17.137   2096.588    0.01     0.99
habitatUpperside   16.743   2096.588    0.01     0.99

(Dispersion parameter for quasipoisson family taken to be 2.7)

    Null deviance: 257.108  on 40  degrees of freedom
Residual deviance:  93.991  on 32  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 13
```

The row of the data frame that has the information for `Bank` is:

```
> subset(moths, habitat=="Bank")
   meters A P habitat
40     21 0 4    Bank
```

The estimate for `(Intercept)` is −15.696; note that this is on a logarithmic scale. Thus the expected number of moths, according to the fitted model, is

$$\exp(-15.696 + 0.129\log(21)) = -2.27 \times 10^{-7}.$$

This is an approximation to zero! A tightening of the convergence criteria would give an even smaller intercept and an expected value that is even closer to zero, which is the observed value.[13]

The huge standard errors, and the value of `NA` for the AIC statistic, are an indication that the output should not be taken at face value. A further indication comes from examination of the standard errors of predicted values:

```
> ## SEs of predicted values
> A.se <- predict(A.glm, se=T)$se.fit
> A.se[moths$habitat=="Bank"]
  40
2097
> range(A.se[moths$habitat!="Bank"])
[1] 0.197 0.631
```

The generalized linear model approximation to the standard error breaks down when, as here, the fitted value for the reference level is zero. We cannot, from this output, say anything about the accuracy of comparisons with `Bank` or between other types of habitat.

A more satisfactory choice of reference level

The difficulty can be avoided, for habitats other than `Bank`, by taking `Lowerside` as the reference:

```
> moths$habitat <- relevel(moths$habitat, ref="Lowerside")
> summary(A.glm <- glm(A ~ habitat + log(meters),
+                      family=quasipoisson, data=moths))
. . . .
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.2098     0.4553    2.66    0.012
habitatBank      -16.9057  2096.5884   -0.01    0.994
habitatDisturbed  -1.2832     0.6474   -1.98    0.056
```

[13]
```
> ## Analysis with tighter convergence criterion
> A.glm <- update(A.glm, epsilon=1e-10)
> summary(A.glm)$coef
> head(summary(A.glm)$coefficients, 3)
                 Estimate Std. Error       t value  Pr(>|t|)
(Intercept)       -20.70      25542 -0.0008103   0.9994
habitatDisturbed   20.62      25542  0.0008074   0.9994
habitatLowerside   21.91      25542  0.0008576   0.9993
 # Notice the large increase in standard errors that were already large
```

```
habitatNEsoak      -0.8217     0.5370   -1.53    0.136
habitatNWsoak       1.5624     0.3343    4.67  5.1e-05
habitatSEsoak       0.0621     0.3852    0.16    0.873
habitatSWsoak       0.2314     0.4781    0.48    0.632
habitatUpperside   -0.1623     0.5843   -0.28    0.783
log(meters)         0.1292     0.1528    0.85    0.404

(Dispersion parameter for quasipoisson family taken to be 2.7)

    Null deviance: 257.108  on 40  degrees of freedom
Residual deviance:  93.991  on 32  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 13
```

Note that the dispersion parameter, which for a Poisson distribution should equal 1, is 2.7. (One of several possible explanations for the dispersion is that there is a clustering effect, with an average cluster size of 2.7. Such clustering need not imply that moths will necessarily be seen together – they may appear at different times and places in the same transect.) Thus, standard errors and p-values from a model that assumed Poisson errors would be highly misleading. Use of a quasi-Poisson model, rather than a Poisson model, has increased standard errors by a factor of $\sqrt{2.7} = 1.64$. The ratio of estimate to standard error is shown as `t value`, implying that a t-distribution should be used as the reference. The number of degrees of freedom is that used for estimating the dispersion.

The estimates are on a logarithmic scale. Thus, the model predicts `exp(-1.283)` = 0.277 times as many moths on `Disturbed` as on `Lowerside`. There remains a problem for the comparison between `Bank` and other habitats. We will return to this shortly.

The fitted values remain the same as when `Bank` was the reference. Thus for `Disturbed`, the earlier model calculated the fitted values (on a log scale) as

$$-15.696 + 15.622 + 0.129\log(\text{meters}) = -0.074 + 0.129\log(\text{meters})$$

while the model with `Lowerside` as the reference calculates them as

$$1.2098 - 1.2832 + 0.129\log(\text{meters}) = -0.0734 + 0.129\log(\text{meters}).$$

According to the model:

$$\log(\text{expected number of moths}) = 1.21 + 0.13\log(\texttt{meters}) + \texttt{habitat (change from Lowerside as reference)},$$

i.e., expected number of moths $= 0.93 \times \texttt{meters}^{0.13} \times e^{\texttt{habitat}}$.

The above output indicates that moths are much more abundant in `NWsoak` than in other habitats. Notice that the coefficient of `log(meters)` is not statistically significant. This species (`A`) avoids flying over long open regions. Their preferred habitats were typically at the ends of transects. As a result, the length of the transect made little difference to the number of moths observed.

The comparison between `Bank` *and other habitats*

Enough has been said to indicate that the `t value` or Wald statistic, which is anyway approximate, is meaningless for comparisons that involve `Bank`. The deviances can however be used, in an adaptation of a likelihood ratio test, to make the comparison. (The adaptation is needed to account for the dispersion; standard likelihood theory does not apply.) Here is the comparison between `Bank` and `NWsoak`:

```
> ## Examine change in deviance from merging Bank with NWsoak
> habitatNW <- moths$habitat
> habitatNW[habitatNW=="Bank"] <- "NWsoak"
> habitatNW <- factor(habitatNW)  # NB: Bank and NWsoak
> table(habitatNW)
habitatNW
Lowerside Disturbed    NEsoak    NWsoak    SEsoak    SWsoak Upperside
        9         7         6         4         7         3         5
> ANW.glm <- glm(A ~ habitatNW + log(meters), family = quasipoisson,
+                data=moths)
> anova(A.glm, ANW.glm, test="F")
Analysis of Deviance Table

Model 1: A ~ habitat + log(meters)
Model 2: A ~ habitatNW + log(meters)
  Resid. Df Resid. Dev Df Deviance    F  Pr(>F)
1        32       94.0
2        33      134.9 -1    -40.9 15.1 0.00047
```

An F-test is used because the dispersion estimate is greater than one. The quantity that is labeled `F` is calculated as (Change in deviance)/Dispersion $= 40.9/2.7 \simeq 15.1$. While this scaled change in deviance statistic does not have the problems that can arise in the calculation of the denominator for the Wald statistic, there can be no guarantee that it will be well approximated by an F-distribution, as assumed here. The diagnostic plots that will now be examined shortly are however encouraging.

Code for the comparison between `Bank` and `Disturbed` is:

```
habitatD <- moths$habitat
habitatD[habitatD=="Bank"] <- "Disturbed"
habitatD <- factor(habitatD)
A.glm <- glm(A ~ habitat + log(meters), family = quasipoisson,
             data=moths)
AD.glm <- glm(A ~ habitatD + log(meters), family = quasipoisson,
              data=moths)
anova(A.glm, AD.glm, test="F")
```

Here, the p-value is 0.34, i.e., the two habitats cannot be distinguished.

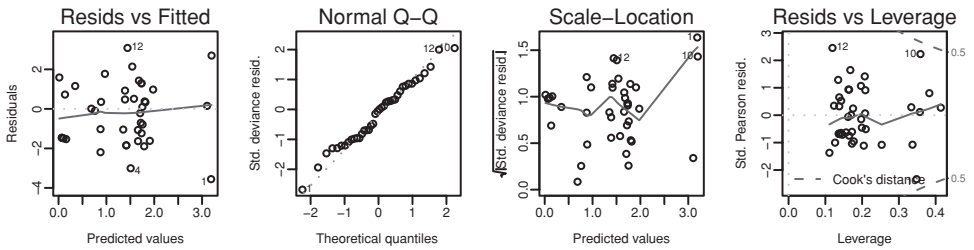


Figure 8.11 Diagnostic plots for modeling `A` (the number of moths of species A) as a function of `habitat` and `log(meters)`, for habitats other than `Bank`.

Diagnostic plots

For the plot now presented (Figure 8.11), the habitat `Bank` was omitted from the analysis. To see why, readers may care to repeat the use of `plot()` with the model that includes `Bank`. The code is:

```
A1.glm <- glm(formula = A ~ habitat + log(meters), family = quasipoisson,
              data = moths, subset=habitat!="Bank")
plot(A1.glm, panel=panel.smooth)
```

Observe that the standardized residuals follow a distribution that seems close to normal. Moreover, the scale–location plot is consistent with the assumption that the dispersion is constant, independent of the predicted value.

8.5 Additional notes on generalized linear models

8.5.1 Residuals, and estimating the dispersion*

The R function `residuals.glm()`, which is called when `residuals()` is used with `glm` object as its argument, offers a choice of three different types of residuals – deviance residuals (`type="deviance"`), Pearson residuals (`type="pearson"`), and working residuals (`type="working"`). For most diagnostic uses, the deviance residuals, which are the default, seem preferable. Deletion residuals, which McCullagh and Nelder (1989) suggest are the preferred residuals to use in checking for outliers, are not, as of version 2.2.0 of R, among the alternatives that are on offer. However, the working residuals may often be a reasonable approximation.

Plots of residuals can be hard to interpret. This is an especially serious problem for models that have a binary (0/1) response, where it is essential to use a suitable form of smooth curve as an aid to visual interpretation.

Other choices of link function for binomial models

Perhaps the most important alternative to the logit is the complementary log–log link; this has $f(x) = \log(-\log(1-x))$. For an argument from extreme value theory that motivates the use of the complementary log–log link, see Maindonald *et al.* (2001). Also used is the probit, which is the normal quantile function, i.e., the inverse of the cumulative normal

distribution. This is hard to distinguish from the logit – extensive data are required, and differences are likely to be evident only in the extreme tails.

Quasi-binomial and quasi-Poisson models

In `quasibinomial` and `quasipoisson` models, a further issue is the estimation of the dispersion. Note that for data with a 0/1 response, no estimate of dispersion is possible, unless there is some form of replication that allows this. The discussion that follows is thus not relevant. See McCullagh and Nelder (1989, Section 4.5.1).

The default is to divide the sum of squares of the Pearson residuals by the degrees of freedom of the residual, and use this as the estimate. This estimate, which appears in the output from `summary()`, has a bias that in most cases is small enough to be ignored, and is in this respect much preferable to the residual mean deviance as an estimate of dispersion. See McCullagh and Nelder (1989, Section 4.5.2).

Note the assumption that the dispersion is constant, independent of the fitted proportion or Poisson rate. This may be inappropriate, especially if the range of fitted values is wide.

In `lm` models, which are equivalent to GLMs with identity link and normal errors, the dispersion and the residual mean deviance both equal the mean residual sum of squares, and have a chi-squared distribution with degrees of freedom equal to the degrees of freedom of the residual. This works quite well as an approximation for quasi-Poisson models whose fitted values are of a reasonable magnitude (e.g., at least 2), and for some binomial models. In other cases, it can be highly unsatisfactory. For models with a binary (0/1) outcome, the residual deviance is a function of the fitted parameters and gives no information on the dispersion.

The difference in deviance between two nested models has a distribution that is often well approximated by a chi-squared distribution. Where there are two such models, with nesting such that the mean difference in deviance (i.e., difference in deviance divided by degrees of freedom) gives a plausible estimate of the source of variation that is relevant to the intended inferences, this may be used to estimate the dispersion. For example, see McCullagh and Nelder (1989, Section 4.5.2). A better alternative may be use of the `lmer()` function in *lme4*, specifying a binomial link and specifying the random part of the model appropriately.

See McCullagh and Nelder (1989, Chapter 9) for a discussion of theoretical issues in the use of ideas of quasi-likelihood.

8.5.2 Standard errors and z- or t-statistics for binomial models

The z- or t-statistics are sometimes known as the Wald statistics. They are based on an asymptotic approximation to the standard error. For binomial models, this approximation can be seriously inaccurate when the fitted proportions are close to 0 or 1, to the extent that an increased difference in an estimated proportion can be associated with a smaller t-statistic. Where it is important to have a reasonably accurate p-value, this is suitably obtained from an analysis of deviance table that compares models with and without the term.

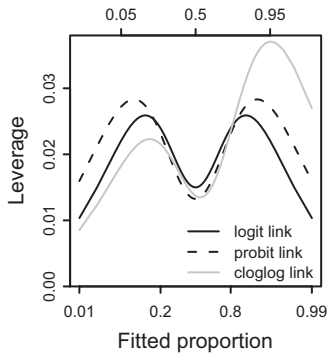


Figure 8.12 Leverage versus fitted proportion, for the three common link functions. Points that all have the same binomial totals were symmetrically placed, on the scale of the response, about a fitted proportion of 0.5. The number and location of points will affect the magnitudes, but for the symmetric links any symmetric configuration of points will give the same pattern of change.

The following demonstrates the effect:

```
> fac <- factor(LETTERS[1:4])
> p <- c(103, 30, 11, 3)/500
> n <- rep(500,4)
> summary(glm(p ~ fac, family=binomial, weights=n))$coef
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    -1.35      0.111  -12.20 3.06e-34
facB           -1.40      0.218   -6.42 1.35e-10
facC           -2.45      0.324   -7.54 4.71e-14
facD           -3.76      0.590   -6.38 1.78e-10
```

Notice that the `z value` for level D is smaller than the `z value` for level C, even though the difference from level A (the reference) is greater.

The phenomenon is discussed in Hauck and Donner (1977).

8.5.3 Leverage for binomial models

In an `lm` model with a single predictor and with homogeneous errors, the points that are furthest away from the mean inevitably have the largest leverages. For models with binomial errors, this is not the case. The leverages are functions of the fitted values. Figure 8.12 shows the pattern of change of the leverage, for the three common link functions, when points are symmetrically placed, on the scale of the response, about a fitted proportion of 0.5. The optimal placement of points for a `probit` link is slightly further apart than for a `logit` link.

8.6 Models with an ordered categorical or categorical response

This section draws attention to an important class of models that further extend the generalized linear model framework. Ordinal regression models will be discussed in modest detail.

Table 8.2 *Data from a randomized trial – assessments of the clarity of the instructions provided to the inhaler.*

	Easy (category 1)	Needed rereading (category 2)	Not clear (category 3)
Inhaler 1	99	41	2
Inhaler 2	76	55	13

Table 8.3 *Odds ratios, and logarithms of odds ratios, for two alternative choices of cutpoint in Table 8.2.*

	Easy versus some degree of difficulty		Clear after study versus not clear	
	Odds	Log odds	Odds	Log odds
Inhaler 1	99/43	$\log(99/43) = 0.83$	140/2	$\log(140/2) = 4.25$
Inhaler 2	76/68	$\log(76/68) = 0.11$	131/13	$\log(131/13) = 2.31$

Loglinear models, which may be appropriate when there is a qualitative (i.e., unordered) categorical response, will get brief mention.

8.6.1 Ordinal regression models

We will demonstrate how logistic and related generalized linear models that assume binomial errors can be used for an initial analysis. The particular form of logistic regression that we will demonstrate is *proportional odds logistic regression*.

Ordinal logistic regression is relevant when there are three or more ordered outcome categories that might, e.g., be (1) complete recovery, (2) continuing illness, (3) death. Here, in Table 8.2, we give an example where patients who were randomly assigned to two different inhalers were asked to compare the clarity of leaflet instructions for their inhaler (data, initially published with the permission of 3M Health Care Ltd, are adapted from Ezzet and Whitehead, 1991).

Exploratory analysis

There are two ways to split the outcomes into two categories: we can contrast "easy" with the remaining two responses (some degree of difficulty), or we can contrast the first two categories (clear, perhaps after study) with "not clear". Table 8.3 presents, side by side in parallel columns, odds based on these two splits. Values for log(odds) are given alongside.

Wherever we make the cut, the comparison favors the instructions for inhaler 1. The picture is not as simple as we might have liked. The log(odds ratio), i.e., the difference on the log(odds) scale, may depend on which cutpoint we choose.

*Proportional odds logistic regression

The function `polr()` in the *MASS* package allows the fitting of a formal model. We begin by fitting a separate model for the two rows of data:

```
library(MASS)
inhaler <-  data.frame(freq=c(99,76,41,55,2,13),
                       choice=rep(c("inh1","inh2"), 3),
                       ease=ordered(rep(c("easy","re-read",
                                          "unclear"), rep(2,3))))
inhaler1.polr <-  polr(ease ~ 1, weights=freq, data=inhaler,
                       Hess=TRUE, subset=inhaler$choice=="inh1")
 # Setting Hess=TRUE at this point averts possible numerical
 # problems if this calculation is deferred until later.
inhaler2.polr <-  polr(ease ~ 1, weights=freq, data=inhaler,
                       Hess=TRUE, subset=inhaler$choice=="inh2")
```

Notice that the dependent variable specifies the categories, while frequencies are specified as weights. The output is:

```
> summary(inhaler1.polr)  # inhaler$choice == "inh1"
. . . .
Intercepts:
                Value Std. Error t value
easy|re-read    0.834 0.183      4.566
re-read|unclear 4.248 0.712      5.966

Residual Deviance: 190.34
AIC: 194.34
> summary(inhaler2.polr)  # inhaler$choice == "inh2"
. . . .
Intercepts:
                Value Std. Error t value
easy|re-read    0.111 0.167      0.666
re-read|unclear 2.310 0.291      7.945

Residual Deviance: 265.54
AIC: 269.54
```

For interpreting the output, observe that the intercepts for the model fitted to the first row are $0.834 = \log(99/43)$ and $4.248 = \log(140/2)$. For the model fitted to the second row, they are $0.111 = \log(76/68)$ and $2.310 = \log(131/13)$.

We now fit the combined model:

```
> summary(inhaler.polr <-  polr(ease ~ choice, weights=freq,
+                               Hess=TRUE, data=inhaler))
. . . .
Coefficients:
           Value Std. Error t value
choiceinh2  0.79      0.245    3.23
```

```
Intercepts:
                 Value  Std. Error t value
easy|re-read      0.863  0.181       4.764
re-read|unclear   3.353  0.307      10.920

Residual Deviance: 459.29
AIC: 465.29
```

The difference in deviance between the combined model and the two separate models is:

```
> deviance(inhaler.polr) - (deviance(inhaler1.polr)
+                           + deviance(inhaler2.polr))
[1] 3.42
```

We compare this with the 5% critical value for a chi-squared deviate on 1 degree of freedom – there are 2 parameters for each of the separate models, making a total of 4, compared with 3 degrees of freedom for the combined model. The difference in deviance is just short of significance at the 5% level.

The parameters for the combined model are:

```
> summary(inhaler.polr)
Call:
polr(formula = ease ~ choice, data = inhaler, weights = freq,
    Hess = T)

Coefficients:
           Value Std. Error t value
choiceinh2  0.79      0.245    3.23

Intercepts:
                 Value  Std. Error t value
easy|re-read      0.863  0.181       4.764
re-read|unclear   3.353  0.307      10.920

Residual Deviance: 459.29
AIC: 465.29
```

The value that appears under the heading "`Coefficients`" is an estimate of the reduction in log(odds) between the first and second rows. Table 8.4 gives the estimates for the combined model.

The fitted probabilities for each row can be derived from the fitted log(odds). Thus for inhaler 1, the fitted probability for the `easy` category is $\exp(0.863)/(1+\exp(0.863)) = 0.703$, while the cumulative fitted probability for `easy` and `re-read` is $\exp(3.353)/(1+\exp(3.353)) = 0.966$.

Table 8.4 *The entries are* log*(odds) and odds estimates for the proportional odds logistic regression model that was fitted to the combined data.*

	log(odds). Odds in parentheses	
	Easy versus some degree of difficulty	Clear after study versus not clear
Inhaler 1	0.863 (exp(0.863) = 2.37)	3.353 (28.6)
Inhaler 2	0.863 − 0.790 (1.08)	3.353 − 0.790 (13.0)

8.6.2 Loglinear models*

Loglinear models specify expected values for the frequencies in a multi-way table. For the model-fitting process, all margins of the table have the same status. However, one of the margins has a special role for interpretative purposes; it is known as the *dependent* margin. For the `UCBAdmissions` data that we discussed in Section 8.3, the interest was in the variation of admission rate with `Dept` and with `Gender`. A loglinear model, with `Admit` as the dependent margin, offers an alternative way to handle the analysis. Loglinear models are however generally reserved for use when the dependent margin has more than two levels, so that logistic regression is not an alternative.

Examples of the fitting of loglinear models are included with the help page for `loglm()`, in the *MASS* package. To run them, type in

```
library(MASS)
example(loglm)
```

8.7 Survival analysis

Survival (or failure) analysis introduces features different from any of those encountered in the regression methods discussed in earlier chapters. It has been widely used for comparing the times of survival of patients suffering a potentially fatal disease who have been subject to different treatments. The computations that follow will use the *survival* package, written for S-PLUS by Terry Therneau, and ported to R by Thomas Lumley.

Other names, mostly used in non-medical contexts, are *Failure Time Analysis* and *Reliability*. Yet another term is *Event History Analysis*. The focus is on time to any event of interest, not necessarily failure. It is an elegant methodology that is too little known outside of medicine and industrial reliability testing.

Applications include:

- The failure time distributions of industrial machine components, electronic equipment, automobile components, kitchen toasters, light bulbs, businesses, etc. (failure time analysis, or reliability).
- The time to germination of seeds, to marriage, to pregnancy, or to getting a first job.
- The time to recurrence of an illness or other medical condition.

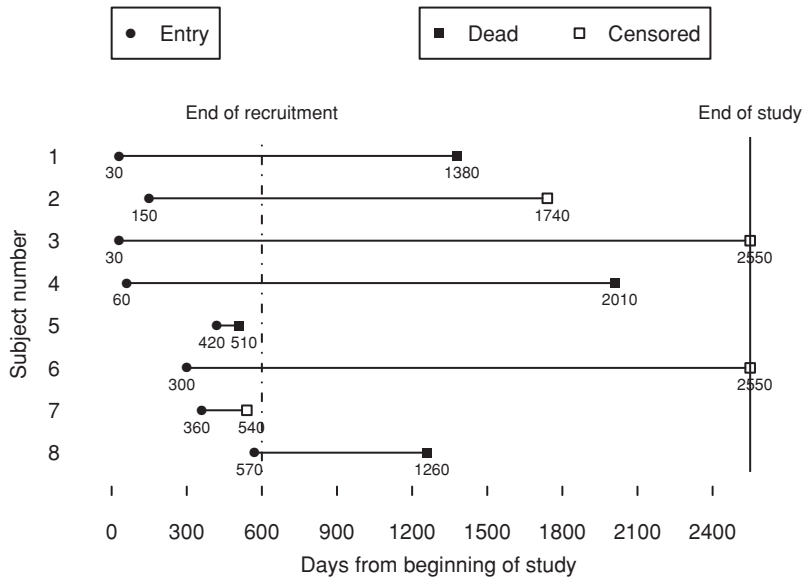


Figure 8.13 Outline of the process of collection of data that will be used in a survival analysis.

The outcomes are survival times, but with a twist. The methodology is able to handle data where failure (or another event of interest) has, for a proportion of the subjects, not occurred at the time of termination of the study. It is not necessary to wait until all subjects have died, or all items have failed, before undertaking the analysis! Censoring implies that information about the outcome is incomplete in some respect, but not completely missing. For example, while the exact point of failure of a component may not be known, it may be known that it did not survive more than 720 hours (= 30 days). In a clinical trial, there may for some subjects be a final time up to which they survived, but no subsequent information. Such observations are said to be right-censored.

Thus, for each observation there are two pieces of information: a time, and censoring information. Commonly the censoring information indicates either right-censoring denoted by a 0, or failure denoted by a 1.

Many of the same issues arise as in more classical forms of regression analysis. One important set of issues has to do with the diagnostics used to check on assumptions. Here there have been large advances in recent years. A related set of issues has to do with model choice and variable selection. There are close connections with variable selection in classical regression. Yet another set of issues has to do with the incomplete information that is available when there is censoring.

Figure 8.13 shows a common pattern for the collection of survival analysis data.

8.7.1 *Analysis of the* `Aids2` *data*

We first examine the data frame `Aids2` (*MASS* package). In the study that provided these data, recruitment continued until the day prior to the end of the study. Once recruited,

subjects were followed until either they were "censored", i.e., were not available for further study, or until they died from an AIDS-related cause. The time from recruitment to death or censoring will be used for analysis.

Observe the variety of different types of right-censoring. Subjects may be removed because they died from some cause that is not AIDS-related, or because they can no longer be traced. Additionally, subjects who are still alive at the end of the study cannot at that point be studied further, and are also said to be censored.

Details of the different columns are:

```
> library(MASS)
> str(Aids2, vec.len=2)
'data.frame':   2843 obs. of  7 variables:
 $ state  : Factor w/ 4 levels "NSW","Other",..: 1 1 1 1 1 ...
 $ sex    : Factor w/ 2 levels "F","M": 2 2 2 2 2 ...
 $ diag   : int  10905 11029 9551 9577 10015 ...
 $ death  : int  11081 11096 9983 9654 10290 ...
 $ status : Factor w/ 2 levels "A","D": 2 2 2 2 2 ...
 $ T.categ: Factor w/ 8 levels "hs","hsid","id",..: 1 1 1 5 1 ...
 $ age    : int  35 53 42 44 39 ...
```

Note that `death` really means "final point in time at which status was known".

The analyses that will be presented will use two different subsets of the data – individuals who contracted AIDS from contaminated blood, and male homosexuals. The extensive data in the second of these data sets makes it suitable for explaining the notion of *hazard*.

A good starting point for any investigation of survival data is the survival curve or (if there are several groups within the data) survival curves. The survival curve estimates the proportion who have survived at any time. The analysis will work with "number of days from diagnosis to death or removal from the study", and this number needs to be calculated.

```
bloodAids <- subset(Aids2, T.categ=="blood")
bloodAids$days <- bloodAids$death-bloodAids$diag
bloodAids$dead <- as.integer(bloodAids$status=="D")
```

For this subset of the data, all subjects were followed right through until the end of the study, and the only patients that were censored were those that were still alive at the end of the study. Thus, Figure 8.14 compares females with males who contracted AIDS from contaminated blood.

The survival package uses the `Surv()` function to package together the time information and the censoring (alive = 0, dead = 1) information. The `survfit()` function then estimates the survival curve. The code is:

```
library(survival)
plot(survfit(Surv(days, dead) ~ sex, data=bloodAids),
     col=c(2,4), conf.int=TRUE)
```

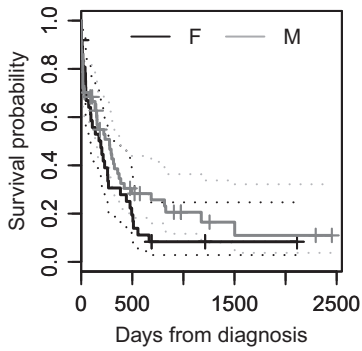


Figure 8.14 Survival curves that compare females with males who contracted AIDS from contaminated blood.

Pointwise 95% confidence intervals have been placed around the estimated survival curves. These assume independence between the individuals in the study group.

8.7.2 Right-censoring prior to the termination of the study

This might happen in (at least) two ways:

- The subject may have died from a cause that is thought unlikely to be AIDS-related, e.g., a traffic accident.
- The subject may have disappeared from the study, with efforts to trace them unsuccessful.

A common assumption is that, subject to any predictor effects, the pattern of risk subsequent to time when the subject was last known to be alive remained the same as for individuals who were not censored. In technical language, the censoring was "non-informative". Censoring would be informative if it gave information on the risk of that patient. For example, some patients may, because of direct or indirect effects of their illness or of the associated medication, be at increased risk of such accidents. Such possibilities make it unlikely that the assumption will be strictly true that censoring was non-informative, though it may be a reasonable working approximation, especially if a relatively small proportion of subjects was censored in this way.

In order to illustrate the point, we will examine the pattern of censoring for males where the mode of transmission (`T.categ`) was homosexual activity. (These data will be used in the later discussion to illustrate the notion of *hazard*.)

```
> hsaids <- subset(Aids2, sex=="M" & T.categ=="hs")
> hsaids$days <- hsaids$death-hsaids$diag
> hsaids$dead <- as.integer(hsaids$status=="D")
> table(hsaids$status,hsaids$death==11504)
   FALSE TRUE
  A   16   916
  D 1531     1
```

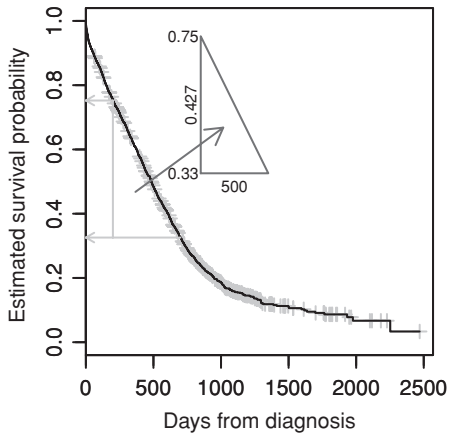


Figure 8.15 Survival curve for males for whom the mode of transmission was homosexual activity. There are two sets of arrows that show how the estimated survival probability decreased over a 400-day period.

Recall that `death` really means "final point in time at which status was known". The interpretation of this output is:

Alive at some time prior to end of study (16 censored at time $< 11\,504$)	Alive at end of study (916 censored on day 11 504)
Dead at some time prior to end of study (1531 not censored)	Dead at end of study (1 not censored)

Just 16 individuals were censored prior to the end of the study, probably not enough to cause a serious bias, even if their censoring was to an extent informative.

8.7.3 The survival curve for male homosexuals

Figure 8.15 shows the survival curve for these data, with 95% pointwise confidence limits. Annotation has been added that will be used to help illustrate the notion of *hazard*. The triangle on the upper left shows the change between 200 days and 700 days, from a survival of 0.75 at day 200 to a survival of 0.33 at day 700. The difference in survival is 0.42, or more precisely 0.427, which is a reduction of 0.000854 per day. The code is:

```
## Survival curve for male homosexuals
hsaids.surv <- survfit(Surv(days, dead) ~ 1, data=hsaids)
plot(hsaids.surv)
```

8.7.4 Hazard rates

The hazard rate is obtained by dividing the mortality per day by the survival to that point. Over a short time interval, multiplication of the hazard by the time interval gives the probability of death over that interval, given survival up to that time. In Figure 8.15, hazard rates at days 200 and 700 are, approximately:

Day 200	$0.00854/0.752 \simeq 0.11$
Day 700	$0.00854/0.326 \simeq 0.026$

More generally, the hazard rate can be determined at any point by fitting a smooth function to the survival curve, and dividing the slope of the tangent by the survival at that point. The above calculations of hazard are rough. More refined estimates can be obtained by using the function `muhaz()` from the *muhaz* package. In order to get precise estimates of the hazard, large sample sizes are needed.

8.7.5 The Cox proportional hazards model

Even where sample sizes are relatively small, and estimates of the individual hazards for two groups that are of interest highly inaccurate, the assumption that the ratio is constant between the two groups may be a reasonable working approximation. For example, it might be assumed that the hazard for males who contracted AIDS from contaminated blood, relative to females, is a constant, i.e., that the hazard for males is directly proportional to the hazard for females. There are two points to keep in mind:

- The assumption can be checked, as will be demonstrated below. Where data are sufficiently extensive to make such a check effective, it often turns out that the ratio is not constant over the whole time period.
- Even if it is suspected that the ratio is not truly constant over the whole time period, the estimated ratio can be treated as an average.

With these points in mind, and using females as the baseline, we now examine the hazard ratios for males who contracted AIDS from contaminated blood. The code is:

```
bloodAids.coxph <- coxph(Surv(days, dead) ~ sex, data=bloodAids)
```

Output is:

```
> summary(bloodAids.coxph)
Call:
coxph(formula = Surv(days, dead) ~ sex, data = bloodAids)

  n= 94
       coef exp(coef) se(coef)     z    p
sexM -0.276     0.759    0.233 -1.18 0.24

     exp(coef) exp(-coef) lower .95 upper .95
sexM     0.759       1.32     0.481      1.20

Rsquare= 0.015   (max possible= 0.998 )
Likelihood ratio test= 1.38  on 1 df,   p=0.239
Wald test            = 1.4  on 1 df,   p=0.236
Score (logrank) test = 1.41  on 1 df,   p=0.235
```

Let $h_0(x)$ be the hazard function for females. Then the model assumes that the hazard function $h_m(x)$ for males can be written:

$$h_m(x) = h_0(x)\exp(\beta).$$

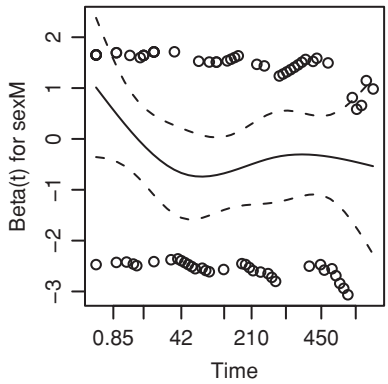


Figure 8.16 Plot of martingale residuals from the Cox proportional hazards model.

In the above output, `coef` is what, in this formula, is β. There are several ways to examine the comparison of males with females:

- The confidence interval for β includes 0. Equivalently, the t-test for the test $\beta = 0$ has a p-value equal to 0.24.
- The confidence interval for $\exp(\beta)$ includes 1.
- A likelihood ratio test has $p = 0.239$.
- A Wald test has $p = 0.236$.
- A log-rank test has $p = 0.235$.

In general, β may be a linear function of predictors, and there may be several groups. Confidence intervals will then be given for the separate coefficients in this model. The likelihood ratio test, the Wald test, and the log-rank test are all then tests for the null hypothesis that the groups have the same hazard function.

The following tests whether the hazard ratio is constant:

```
> cox.zph(bloodAids.coxph)
        rho chisq     p
sexM -0.127  1.21 0.272
```

A plot of *Schoenfeld* residuals, with smooth curves fitted as in Figure 8.16, may however be more informative. The code is:

```
plot(cox.zph(bloodAids.coxph))
```

The Schoenfeld residuals (one for each event, here death) are computed for each predictor in a model and are used to determine whether the coefficients in the model are constant or time-varying. They are essentially differences between the predictor value for each individual and an appropriately weighted average of the corresponding coefficient over all other individuals still at risk at the individual's event time.

The graph suggests that the hazard ratio may have been greater for males for the first 40 days or so after diagnosis. However, the difference from a constant β cannot be distinguished from statistical error. In samples of this size, the difference from a constant ratio must be large to allow much chance of detection.

Also available are the *martingale* residuals. These can be calculated and plotted thus:

```
plot(bloodAids$days, resid(bloodAids.coxph))
lines(lowess(bloodAids$days, resid(bloodAids.coxph)))
```

The martingale residuals are given by either 1− the log of the survival function estimate or the negative log of the survival function estimate, according to whether the observation has been censored or not. These residuals should be approximately exponentially distributed if the model is correct. One way to check this is to plot the residuals against time (or another suitable predictor) and to pass a loess curve through the plotted points. If the result differs substantially from a horizontal line, then the model has not been specified correctly.

8.8 Transformations for count data

Transformations were at one time commonly used to make count data amenable to analysis using normal theory methods. Generalized linear models have largely removed the need for such approaches. They are, however, still sometimes a useful recourse. Here are some of the possibilities:

- **The square root transformation**: $y = \sqrt{n}$. This may be useful for counts. If counts follow a Poisson distribution, this gives values that have constant variance 0.25.
- **The angular transformation**: $y = \arcsin(\sqrt{p})$, where p is a proportion. If proportions have been generated by a binomial distribution, the angular transformation will "stabilize the variance".

  For handling this calculation in R, define `arcsin ← function (p) asin (sqrt(p))`. The values that asin() returns are in radians. To obtain values that run from 0 to 90, multiply by $180/\pi = 57.3$.
- **The probit or normal equivalent deviate**. Again, this is for use with proportions. For this, take a table of areas under the normal curve. We transform back from the area under the curve to the distance from the mean. It is commonly used in bioassay work. This transformation has a severe effect when p is close to 0 or 1. It moves 0 down to $-\infty$ and 1 up to ∞, i.e., these points cannot be shown on the graph. Some authors add 5 on to the normal deviates to get "probits".
- **The logit**. This function is defined by $\text{logit}(p) = \log(p/(1-p))$ – this is a little more severe than the probit transformation. Unless there are enormous amounts of data, it is impossible to distinguish data that require a logit transformation from data that require a probit transformation. Logistic regression uses this transformation on the expected values in the model.
- **The complementary log–log transformation**. This function is defined by $\text{cloglog}(p) = \log(-\log(1-p))$. For p close to 0, this behaves like the logit transformation. For large values of p, it is a much milder transformation than the probit or logit. It is often the appropriate transformation to use when p is a mortality that increases with time.

Proportions usually require transformation unless the range of values is quite small, between about 0.3 and 0.7.

Note that generalized linear models transform, not the observed proportion, but its expected value as estimated by the model.

8.9 Further reading

Dobson (2001) is an elementary introduction to generalized linear models. See also Faraway (2006). Harrell (2001) gives a comprehensive account of both generalized linear models and survival analysis that includes extensive practical advice. Venables and Ripley (2002) give a summary overview of the S-PLUS and R abilities, again covering survival analysis as well as generalized linear models. Chapter 2 of Wood (2006) is a succinct, insightful, and practically oriented summary of the theory of generalized linear models. A classic reference on generalized linear models is McCullagh and Nelder (1989). For quasi-binomial and quasi-Poisson models, see pp. 200–208.

Collett (2003) is a basic introduction to elementary uses of survival analysis. Therneau and Grambsch (2001) describe extensions to the Cox model, with extensive examples that use the abilities of the *survival* package. Bland and Altman (2005) is an interesting example of the application of survival methods.

There are many ways to extend the models that we have discussed. We have not demonstrated models with links other than the logit. Particularly important, for data with binomial or quasi-binomial errors, is the complementary log–log link; for this specify `family = binomial(link=cloglog)`, replacing `binomial` with `quasibinomial` if that is more appropriate. Another type of extension, not currently handled in R, arises when there is a natural mortality that must be estimated. See, e.g., Finney (1978). Multiple levels of variation, such as we will discuss in Chapter 10, are a further potential complication. Maindonald *et al.* (2001) present an analysis where all of these issues arose. In that instance, the binomial totals (n) were large enough that it was possible to work with the normal approximation to the binomial.

References for further reading

Bland, M. and Altman, D. 2005. Do the left-handed die young? *Significance* 2: 166–70.

Collett, D. 2003. *Modelling Survival Data in Medical Research*, 2nd edn.

Dobson, A. J. 2001. *An Introduction to Generalized Linear Models*, 2nd edn.

Faraway, J. J. 2006. Extending the Linear Model with R. *Generalized Linear, Mixed Effects and Nonparametric Regression Models.*

Finney, D. J. 1978. *Statistical Methods in Bioassay*, 3rd edn.

Harrell, F. E. 2001. *Regression Modelling Strategies, with Applications to Linear Models, Logistic Regression and Survival Analysis.*

Maindonald, J. H., Waddell, B. C. and Petry, R. J. 2001. Apple cultivar effects on codling moth (Lepidoptera: Tortricidae) egg mortality following fumigation with methyl bromide. *Postharvest Biology and Technology* 22: 99–110.

McCullagh, P. and Nelder, J. A. 1989. *Generalized Linear Models*, 2nd edn.

Therneau, T. M. and Grambsch, P. M. 2001. *Modeling Survival Data: Extending the Cox Model.*

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

Wood, S. N. 2006. *Generalized Additive Models. An Introduction with R.*

8.10 Exercises

1. The following table shows numbers of occasions when inhibition (i.e., no flow of current across a membrane) occurred within 120 s, for different concentrations of the protein peptide-C (data are used with the permission of Claudia Haarmann, who obtained these data in the course of her PhD research). The outcome `yes` implies that inhibition has occurred.

```
conc 0.1 0.5  1 10 20 30 50 70 80 100 150
no     7   1 10  9  2  9 13  1  1   4   3
yes    0   0  3  4  0  6  7  0  0   1   7
```

   Use logistic regression to model the probability of inhibition as a function of protein concentration.

2. In the data set (an artificial one of 3121 patients, that is similar to a subset of the data analyzed in Stiell *et al*., 2001) `minor.head.injury`, obtain a logistic regression model relating `clinically.important.brain.injury` to other variables. Patients whose risk is sufficiently high will be sent for CT (computed tomography). Using a risk threshold of 0.025 (2.5%), turn the result into a decision rule for use of CT.

3. Consider again the `moths` data set of Section 8.4.

   (a) What happens to the standard error estimates when the `poisson` family is used in `glm()` instead of the `quasipoisson` family?
   (b) Analyze the `P` moths, in the same way as the `A` moths were analyzed. Comment on the effect of transect length.

4*. The factor `dead` in the data set `mifem` (*DAAG* package) gives the mortality outcomes (`live` or `dead`), for 1295 female subjects who suffered a myocardial infarction. (See Section 11.5 for further details.) Determine ranges for `age` and `yronset` (year of onset), and determine tables of counts for each separate factor. Decide how to handle cases for which the outome, for one or more factors, is not known. Fit a logistic regression model, beginning by comparing the model that includes all two-factor interactions with the model that has main effects only.

5. Use the function `logisticsim()` (in the *DAAG* package) to simulate data from a logistic regression model to study the `glm()` function. For example, you might try experiments such as the following:

   (a) Simulate 100 observations from the model

$$\text{logit}(x) = 2 - 4x$$

   for $x = 0, 0.01, 0.02, \ldots, 1.0$. [This is the default setting for `logisticsim()`.]
   (b) Plot the responses (y) against the "dose" (x). Note how the pattern of 0s and 1s changes as x increases.
   (c) Fit the logistic regression model to the simulated data, using the `binomial` family. Compare the estimated coefficients with the true coefficients. Are the estimated coefficients within about 2 standard errors of the truth?
   (d) Compare the estimated logit function with the true logit function. How well do you think the fitted logistic model would predict future observations? For a concrete indication of the difference, simulate a new set of 100 observations at the same x values, using a specified pseudorandom number generator seed and the true model. Then simulate some predicted observations using the estimated model and the same seed.

6. As in the previous exercise, the function `poissonsim()` allows for experimentation with Poisson regression. In particular, `poissonsim()` can be used to simulate Poisson responses with log-rates equal to $a + bx$, where a and b are fixed values by default.

   (a) Simulate 100 Poisson responses using the model

   $$\log \lambda = 2 - 4x$$

   for $x = 0, 0.01, 0.02 \ldots, 1.0$. Fit a Poisson regression model to these data, and compare the estimated coefficients with the true coefficients. How well does the estimated model predict future observations?

   (b) Simulate 100 Poisson responses using the model

   $$\log \lambda = 2 - bx$$

   where b is normally distributed with mean 4 and standard deviation 5. [Use the argument `slope.sd=5` in the `poissonsim()` function.] How do the results using the `poisson` and `quasipoisson` families differ?

7. The data set `cricketer` (*DAAG*) holds information on British first-class cricketers with year of birth 1840–1960. The function `gam()` in the *mgcv* package can be used to fit GLM models that include smoothing terms, with automatic choice of smoothing parameter. Use the following to model changes with year of birth, in the proportion of left-handers in the `cricketer` data:

```
library(mgcv)
hand <- with(cricketer, unclass(left)-1)  # 0 for left; 1 for right
hand.gam <- gam(hand ~ s(year), data=cricketer)
plot(hand.gam)
```

   (a) Is the assumption of independence between cricketers, for these data, reasonable?

   (b) What might explain the clear dip in the proportion of left-handers with birth years in the decade or so leading up to 1934?

   (c) Use the following to get estimates on the scale of the response, and plot the results:

```
ndf <- data.frame(year=seq(from=1840, to=1960, by=2))
fit <- predict(hand.gam, newdata=ndf, type="response", se.fit=TRUE)
plot(fit$fit ~ newdat$year, ylim=range(fit), type="1")
```

   Add confidence bounds to the plot.

8. Fit Cox regressions that compare the hazard rate between right- and left-handers. Experiment with using as a covariate both (i) a quartic polynomial in `year` and (ii) a smooth function of year (e.g., `pspline(year, df=4)`):

   (a) Ignore cause of death; i.e., all still alive in 1992 are censored.

   (b) Censor all accidental deaths; this compares the risk for those who did not die by accident (NB: all still alive in 1992 are in any case "censored").

   (c) Censor all deaths except accidental deaths that were not "killed in action". (NB: "killed in action" is a subset of "accidental death".)

   (d) Censor observations for all deaths other than "killed in action".

   Use code such as the following:

```
library(survival)
coxph(Surv(life, dead) ~ left + poly(year, 4), data = cricketer)
```

   Interpret the results. Aside from accidental deaths (including "killed in action"), are left-handers at greater hazard than right-handers?

9

Time series models

A time series is a sequence of observations that have been recorded over time. Almost invariably, observations that are close together in time are more strongly correlated than observations that are widely separated. The independence assumption of previous chapters is, in general, no longer valid. Variation in a single spatial dimension may have characteristics akin to those of time series, and the same types of models may find application there also.

Many techniques have been developed to deal with the special nature of the dependence that is commonly found in such series. The present treatment will be introductory and restricted in scope, focusing on autoregressive integrated moving average (ARIMA) models.

In the section that follows, annual depth measurements at a specific site on Lake Huron will be modeled directly as an ARIMA process. Section 9.2 will model a regression where the error term has a time series structure. The chapter will close with a brief discussion of "non-linear" time series, such as have been widely used in modeling financial time series.

The analyses will use functions in the *stats* package, which is a recommended package, included with binary distributions. Additionally, there will be use of the *forecast* package. In order to make this available, install the *forecasting* bundle of packages.

The brief discussion of non-linear time series (ARCH and GARCH models) will require access to the *tseries* package, which must be installed.

The interested reader is directed to any of several excellent books listed in the references.

9.1 Time series – some basic ideas

The time series object `LakeHuron` (*datasets*) has annual depth measurements at a specific site on Lake Huron. The discussion of sequential dependence, and of the use of ARIMA-type models of this dependence, will use these data for illustrative purposes.

9.1.1 Preliminary graphical explorations

Figure 9.1 is a trace plot, i.e., an unsmoothed plot of the data against time. The code is:

```
## Plot depth measurements: ts object LakeHuron (datasets)
plot(LakeHuron, ylab="depth (in feet)", xlab = "Time (in years)")
```

There is a slight downward trend for at least the first half of the series. Observe also that depth measurements that are close together in time are often close in value. Occasional exceptions are consistent with this general pattern. Consistently with the sequential

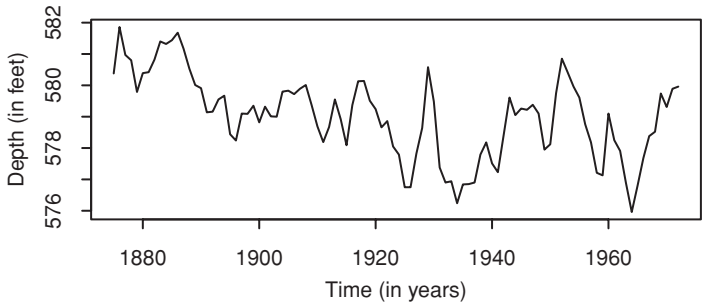


Figure 9.1 A trace plot of annual depth measurements of Lake Huron versus time.

dependence that is typical in time series, these are not independent observations. A key challenge is to find good ways to model this dependence.

Lag plots may give a useful visual impression of the dependence. Suppose that our observations are $x_1, x_2, \ldots, x_n$. Then the lag 1 plot plots x_i against x_{i-1} $(i = 2, \ldots, n)$, thus:

y-value	x_2	x_3	...	x_n
lag 1 (x-axis)	x_1	x_2	...	x_{n-1}

For a lag 2 plot, x_i is plotted against x_{i-2} $(i = 3, \ldots, n)$, and so on for higher lags. Notice that the number of points that are plotted is reduced by the number of lags.

Figure 9.2A shows the first four lag plots for the Lake Huron data. The code is:

```
lag.plot(LakeHuron, lags=3, do.lines=FALSE)
```

The scatter of points about a straight line in the first lag plot suggests that the dependence between consecutive observations is linear. The second, third, and fourth lag plots show increasing scatter. As might be expected, points separated by two or three lags are successively less dependent. Note that the slopes, and therefore the correlations, are all positive.

9.1.2 The autocorrelation and partial autocorrelation function

Informally, the correlation that is evident in each of the lag plots in Figure 9.2A is an *autocorrelation* (literally, self-correlation). The autocorrelation function (ACF), which gives the autocorrelations at all possible lags, often gives useful insight. Figure 9.2B plots the first 19 autocorrelations. The code is:

```
acf(LakeHuron)
## pacf(LakeHuron) gives the plot of partial autocorelations
```

The autocorrelation at lag 0 is included by default; this always takes the value 1, since it is the correlation between the data and themselves. Interest is in the autocorrelations at lag 1 and later lags. As inferred from the lag plots, the largest autocorrelation is at lag 1 (sometimes called the serial correlation), with successively smaller autocorrelations at lags 2, 3, ... There is no obvious linear association among observations separated by lags of more than about 10.

Autocorrelations at lags greater than 1 are in part a flow-on from autocorrelations at earlier lags. The *partial autocorrelation* at a particular lag measures the strength of

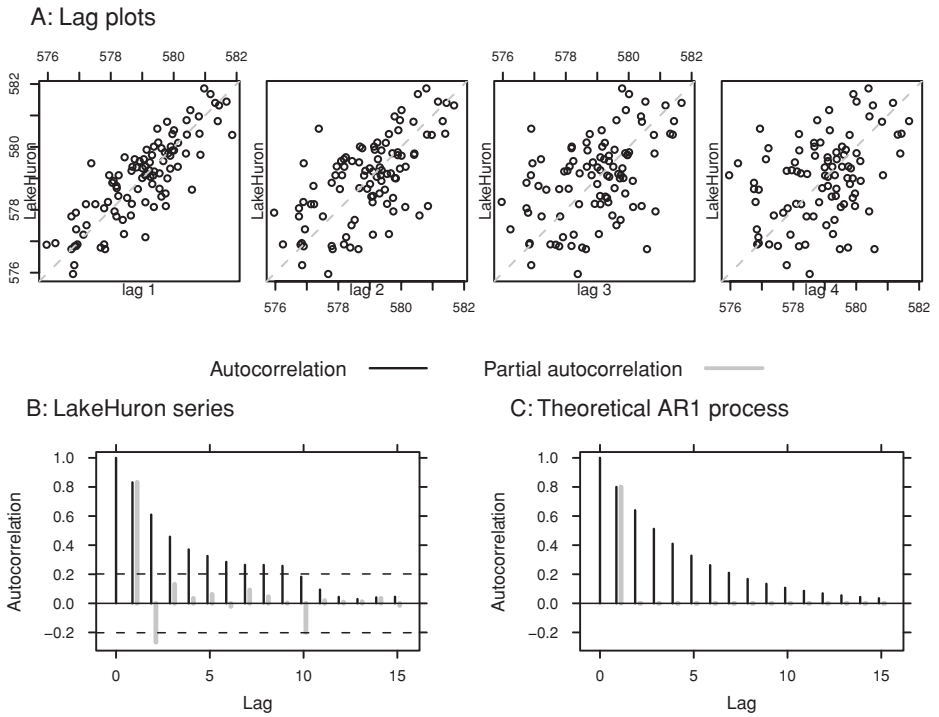


Figure 9.2 The top four panels (A) show the first four lag plots of the annual depth measurements of Lake Huron. Panel B shows the estimated autocorrelations, with the partial autocorrelations shown in gray. The dashed horizontal lines are approximate pointwise 5% critical values for the autocorrelations for a pure noise process, i.e., independent normal data with mean 0. Panel C shows the theoretical autocorrelations for an AR(1) process with $\alpha = 0.8$. Again the partial autocorrelations, with a single spike at lag 1, are shown in gray.

linear correlation between observations separated by that lag, after adjusting for correlations between observations separated by fewer lags. Figure 9.2B indicates partial autocorrelations that may be greater than statistical error at lags 1, 2 and 10. Other autocorrelations are mostly a flow-on effect from the large autocorrelations at lag 1. Figure 9.2C shows the autocorrelation function and partial autocorrelation function for a very simple theoretical model that accounts for most of the correlation structure in Figure 9.2B. Notice however the very clear differences, especially evident in the comparison between the partial autocorrelation plots in Figures 9.2B and C.

As discussed in the next subsection, autocorrelation in a time series complicates the estimation of such quantities as the standard error of the mean. There must be appropriate modeling of the dependence structure. If there are extensive data, it helps to group the data into sets of k successive values, and take averages. The serial correlation then reduces to ρ_1/k approximately. See Miller (1986) for further comment.

9.1.3 Autoregressive models

We have indicated earlier that, whenever possible, the rich regression framework provides additional insights, beyond those available from the study of correlation structure. This is

true also for time series. Autoregressive (AR) models move beyond attention to correlation structure to the modeling of regressions relating successive observations.

The AR(1) *model*

Figure 9.2C showed the theoretical autocorrelation and partial autocorrelation plots for an AR(1) process. The autoregressive model of order 1 (AR(1)) for a time series $X_1, X_2, \ldots$ has the recursive formula

$$X_t = \mu + \alpha(X_{t-1} - \mu) + \varepsilon_t, \quad t = 1, 2, \ldots,$$

where μ and α are parameters. Figure 9.2C had $\alpha = 0.8$.

Usually, α takes values in the interval $(-1, 1)$; this is the so-called *stationary* case. Nonstationarity, in the sense used here, implies that the properties of the series are changing with time. The mean may be changing with time, and/or the variances and covariances may depend on the time lag. Any such nonstationarity must be removed or modeled.

For series of positive values, a logarithmic transformation will sometimes bring the series closer to stationarity and/or make it simpler to model any trend. Discussion of standard ways to handle trends will be deferred to Subsection 9.1.4.

The error term ε_t is the familiar independent noise term with constant variance σ^2. The distribution of X_0 is assumed fixed and will not be of immediate concern.

For the AR(1) model, the ACF at lag i is α^i, for $i = 1, 2, \ldots$. If $\alpha = 0.8$, then the observed autocorrelations should be 0.8, 0.64, 0.512, 0.410, 0.328, ..., a geometrically decaying pattern, as in Figure 9.2C and not too unlike that in Figure 9.2B.

To gain some appreciation for the importance of models like the AR(1) model, we consider estimation of the standard error for the estimate of the mean μ. Under the AR(1) model, a large-sample approximation to the standard error for the mean is:

$$\frac{\sigma}{\sqrt{n}} \frac{1}{(1-\alpha)}.$$

For a sample of size 100 from an AR(1) model with $\sigma = 1$ and $\alpha = 0.5$, the standard error of the mean is 0.2. This is exactly twice the value that results from the use of the usual $\sigma/\sqrt{n}$ formula. Use of the usual standard error formula will result in misleading and biased confidence intervals for time series where there is substantial autocorrelation.

There are several alternative methods for estimating for the parameter α. The *method of moments* estimator uses the autocorrelation at lag 1, here equal to 0.8319. The maximum likelihood estimator, equal to 0.8376, is an alternative.[1]

[1]
```
## Yule-Walker autocorrelation estimate
LH.yw <- ar(x = LakeHuron, order.max = 1, method = "yw")
                          # autocorrelation estimate
                          # order.max = 1 for the AR(1) model
LH.yw$ar                  # autocorrelation estimate of alpha
## Maximum likelihood estimate
LH.mle <- ar(x = LakeHuron, order.max = 1, method = "mle")
LH.mle$ar                 # maximum likelihood estimate of alpha
LH.mle$x.mean             # estimated series mean
LH.mle$var.pred           # estimated innovation variance
```

The general AR(p) *model*

It is possible to include regression terms for X_t against observations at greater lags than one. The autoregressive model of order p (the AR(p) model) regresses X_t against $X_{t-1}, X_{t-2}, \ldots, X_{t-p}$:

$$X_t = \mu + \alpha_1(X_{t-1} - \mu) + \cdots + \alpha_p(X_{t-p} - \mu) + \varepsilon_t, \quad t = 1, 2, \ldots,$$

where $\alpha_1, \alpha_2, \ldots, \alpha_p$ are additional parameters that would need to be estimated. The parameter α_i is the partial autocorrelation at lag i.

Assuming an AR process, how large should p be, i.e., how many AR parameters are required? The function `ar()`, in the *stats* package, can be used to estimate the AR order. This uses the Akaike Information Criterion (AIC), which was introduced in Subsection 6.3.2. Use of this criterion, with models fitted using maximum likelihood, gives:

```
> ar(LakeHuron, method="mle")
. . . .
Coefficients:
      1        2
 1.0437  -0.2496

Order selected 2  sigma^2 estimated as  0.4788
ar(LakeHuron, method="mle")      # AIC is used by default if
                                 # order.max is not specified
```

While the plot of partial autocorrelations in Figure 9.2B suggested that an AR process might be a good first approximation, there is a lag 2 spike that is not consistent with a pure low-order AR process. Moving average models, which will be discussed next, give the needed additional flexibility.

9.1.4 Autoregressive moving average models – theory*

In a moving average (MA) process of order q, the *error* term is the sum of an *innovation* ε_t that is specific to that term, and a linear combination of earlier *innovations* $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$. The equation is

$$X_t = \mu_t + \varepsilon_t + b_1\varepsilon_{t-1} + \cdots + b_q\varepsilon_{t-q} \tag{9.1}$$

where $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_q$ are independent random normal variables, all with mean 0. The autocorrelation between terms that are more than q time units apart is zero. Moving average terms can be helpful for modeling autocorrelation functions that are spiky, or that have a sharp cutoff.

An autoregressive moving average (ARMA) model is an extension of an autoregressive model to include "moving average" terms. Autoregressive integrated moving average (ARIMA) models allow even greater flexibility. (The model (AR or MA or ARMA) is applied, not to the series itself, but to a *differenced* series, where the differencing process may be repeated more than once.) There are three parameters: the order p of the autoregressive component, the order d of differencing (in our example 0), and the order q of the moving average component.

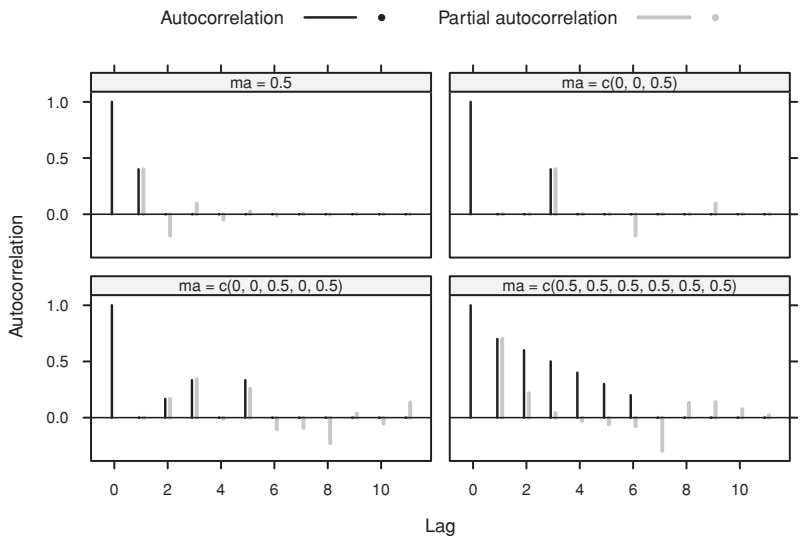


Figure 9.3 Theoretical autocorrelations for a moving average process, with the parameters shown. The thick gray lines are the partial autocorrelations.

Differencing of order 1 removes a linear trend. Differencing of order 2 removes a quadratic trend. The downside of differencing is that it complicates the correlation structure. Thus, it turns an uncorrelated series into an MA series of order 1. Differencing can be done explicitly prior to analysis.[2] However it is easiest to let the R time series modeling functions do any differencing that is required internally, then reversing the process following the fitting of an ARMA model to the differenced series.

Figure 9.3 shows the autocorrelation functions for simulations of several different moving average models. Notice that:

- with $b_1 = 0.5$ ($q = 1$), there is a spike at lag 1;
- with $b_1 = b_2 = 0$, $b_3 = 0.5$ ($q = 3$), there is a spike at lag 3;
- with $b_1 = b_2 = 0$, $b_3 = 0.5$, $b_4 = 0$, $b_5 = 0.5$ ($q = 5$), there are spikes at lags 2, 3 and 5;
- with $b_i = 0.5$ ($i = 1, \ldots, 5$), there are spikes at the first five lags.

9.1.5 Automatic model selection?

The function `auto.arima()` from the *forecast* package uses the AIC in a model selection process that can proceed pretty much automatically. This takes some of the inevitable arbitrariness from the selection process. Fortunately, in applications such as forecasting or the regression example in the next section, what is important is to account for the major part of the correlation structure. A search for finesse in the detail may be pointless, with scant practical consequence.

The algorithm looks for the optimum AR order p, the optimum order of differencing, and the optimum MA order q. Additionally, there is provision for seasonal terms and for

[2] `## series.diff <- diff(series)`

"drift". Drift implies that there is a constant term in a model that has $d > 0$. Seasonal terms, which are important in some applications, will not be discussed further here.

An exhaustive (non-stepwise) search can be very slow. The `auto.arima()` default is a stepwise search. At each iteration, the search is limited to a parameter space that is "close" to the parameter space of the current model. Values of AR and MA parameters are allowed to change by at most one from their current values. There are similar restrictions on the search space for seasonal and other parameters.

Depending on previous experience with comparable series, there may be a case for checking the partial autocorrelation plot for clear "spikes" that may indicate high-order MA terms, before proceeding to a stepwise search. Note however that, in a plot that shows lags 1 to 20 of a pure noise process, one can expect, on average, one spike that extends beyond the 95% pointwise limits. For the `LakeHuron` data, there is no cogent reason to change from the default choice of starting values of AR and MA parameters.

Application of `auto.arima()` to the `LakeHuron` data gives the following:

```
> library(forecast)
## Use auto.arima() to select model
> auto.arima(LakeHuron)
Series: LakeHuron
ARIMA(1,1,2)

Call: auto.arima(x = LakeHuron)

Coefficients:
        ar1     ma1     ma2
      0.648  -0.584  -0.328
s.e.  0.129   0.139   0.108

sigma^2 estimated as 0.482:  log likelihood = -102.6
AIC = 213.1   AICc = 213.6   BIC = 223.4
```

Now try

```
## Check that model removes most of the correlation structure
acf(resid(arima(LakeHuron, order=c(p=1, d=1, q=2))))
## The following achieves the same effect, for these data
acf(resid(auto.arima(LakeHuron)))
```

The function `auto.arima()` chose an ARIMA(1,1,2) model; i.e., the order of the autoregressive terms is $p = 1$, the order of the differencing is $d = 1$, and the order of the moving average terms is $q = 2$.

9.1.6 A time series forecast

The function `forecast()` in the *forecast* package makes it easy to obtain forecasts, as in Figure 9.4. The code is:

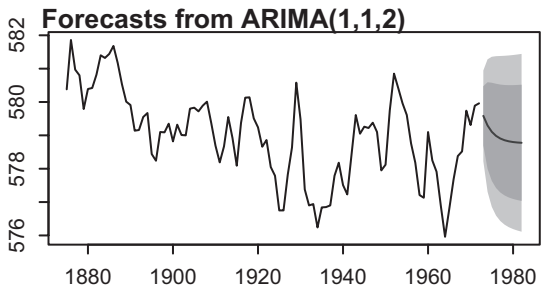


Figure 9.4 Forecast levels for Lake Huron, based on the `LakeHuron` data. The intervals shown are 80% and 95% prediction intervals.

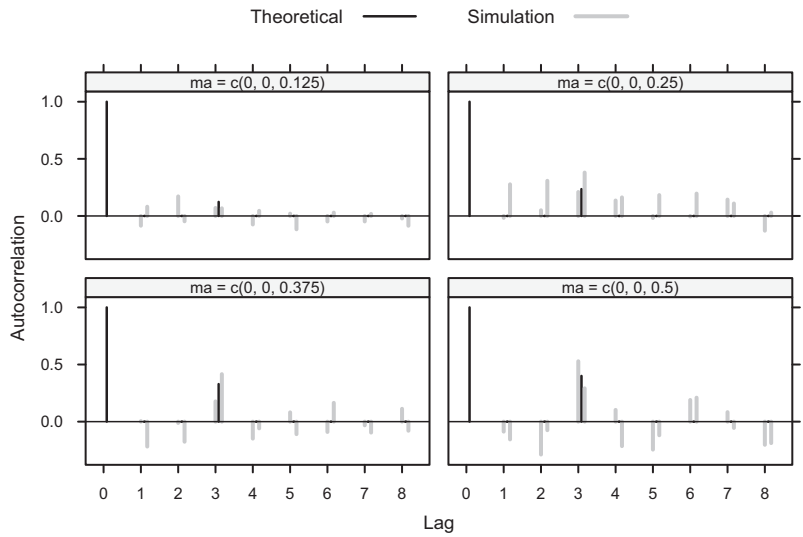


Figure 9.5 Results of two simulation runs (in gray) for an MA process of order 3, with b_3 set in turn to 0.125, 0.25, 0.375 and 0.5. In each case, coefficients other than b_3 were set to zero. The theoretical autocorrelation is shown, in each case, in black.

```
LH.arima <- auto.arima(LakeHuron)
fcast <- forecast(LH.arima)
plot(fcast)
```

Use of simulation as a check

Simulation can be used to assess what magnitude of MA or other coefficients may be detectable. To simplify the discussion, we limit attention to a moving average process of order 3, with $b_1 = 0$, $b_2 = 0$, and b_3 taking one of the values 0.125, 0.25, 0.374 and 0.5. Below we will tabulate, for each of these values, the level of MA process detected by `auto.arima()` over 20 simulation runs.

As an indication of the variation between different simulation runs, Figure 9.5 shows the autocorrelations and partial autocorrelations from two runs. Code that plots results from a single set of simulation runs is:

```
oldpar <- par(mfrow=c(3,2), mar=c(3.1,4.6,2.6, 1.1))
for(i in 1:4){
  ma3 <- 0.125*i
  simts <- arima.sim(model=list(order=c(0,0,3), ma=c(0,0,ma3)), n=98)
  acf(simts, main="", xlab="")
  mtext(side=3, line=0.5, paste("ma3 =", ma3), adj=0)
}
par(oldpar)
```

Now do 20 simulation runs for each of the four values of b_3, recording in each case the order of MA process that is detected:

```
set.seed(29)          # Ensure that results are reproducible
estMAord <- matrix(0, nrow=4, ncol=20)
for(i in 1:4){
  ma3 <- 0.125*i
  for(j in 1:20){
    simts <- arima.sim(n=98, model=list(ma=c(0,0,ma3)))
    estMAord[i,j] <- auto.arima(simts, start.q=3)$arma[2]
}
}
detectedAs <- table(row(estMAord), estMAord)
dimnames(detectedAs) <- list(ma3=paste(0.125*(1:4)),
                             Order=paste(0:(dim(detectedAs)[2]-1)))
```

The following table summarizes the result of this calculation:

```
> print(detectedAs)
       Order
ma3      0  1  2  3  4
  0.125 12  4  3  1  0
  0.25   7  3  2  8  0
  0.375  3  1  2 11  3
  0.5    1  1  0 15  3
```

Even with $b_3 = 0.375$, the chances are only around 50% that an MA component of order 3 will be detected as of order 3.

9.2* Regression modeling with ARIMA errors

The Southern Oscillation Index (SOI) is the difference in barometric pressure at sea level between the Pacific island of Tahiti and Darwin, close to the northernmost tip of Australia. Annual SOI and rainfall data for various parts of Australia, for the years 1900–2005, are in the data frame `bomregions` (*DAAG* package). (See Nicholls *et al.* (1996) for background.) To what extent is the SOI useful for predicting rainfall in one or other region of Australia?

This section will examine the relationship between rainfall in the Murray–Darling basin (`mdbRain`) and `SOI`, with a look also at the relationship in northern Australia (`northRain`). The Murray–Darling basin takes its name from its two major rivers, the

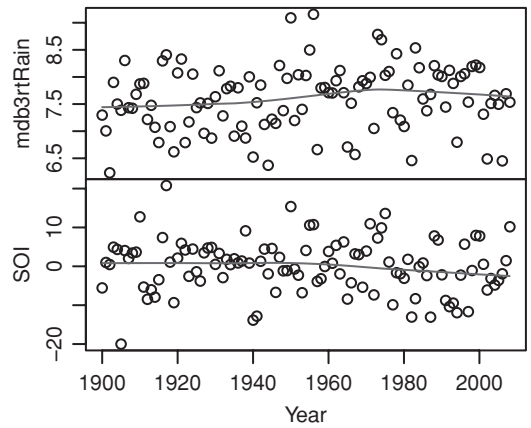


Figure 9.6 Plots of `mdbRain` (Australia's Murray–Darling basin) and `SOI` (Southern Oscillation Index) against year.

Murray and the Darling. Over 70% of Australia's irrigation resources are concentrated there. It is Australia's most significant agricultural area.[3]

Figure 9.6 plots the two series. The code is:

```
library(DAAG)
## Plot time series mdbRain and SOI: ts object bomregions (DAAG)
plot(ts(bomregions[, c("mdbRain","SOI")], start=1900),
     panel=function(y,...)panel.smooth(bomregions$Year, y,...))
```

Trends that are evident in the separate curves are small relative to the variation of the data about the trend curves.

Note also that a cube root transformation has been used to reduce or remove the skewness in the data. Use of a square root or cube root transformation (Stidd, 1953) is common for such data. The following code creates a new data frame `xbomsoi` that has the cube root transformed rainfall data, together with trend estimates (over time) for `SOI` and `mdb3rtRain`.

```
xbomsoi <-
  with(bomregions, data.frame(SOI=SOI, mdbRain=mdbRain,
                              mdb3rtRain=mdbRain^{0.33}))
xbomsoi$trendSOI <- lowess(xbomsoi$SOI, f=0.1)$y
xbomsoi$trendRain <- lowess(xbomsoi$mdb3rtRain, f=0.1)$y
```

For understanding the relationship between `mdb3rtRain` and `SOI`, it can be helpful to distinguish effects that seem independent of time from effects that result from a steady pattern of change over time. Detrended series, for `mdb3rtRain` and for `SOI`, can be obtained thus:

```
xbomsoi$detrendRain <-
  with(xbomsoi, mdb3rtRain - trendRain + mean(trendRain))
```

[3] The help page for `bomregions` gives a link to a map that identifies these regions.

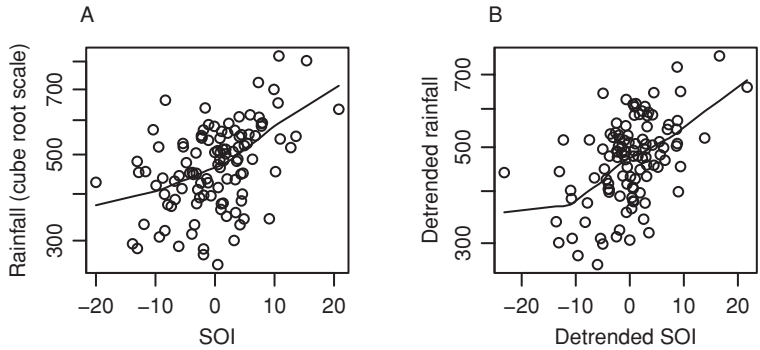


Figure 9.7 Plot of (A) rainfall (cube root scale) against SOI and (B) detrended rainfall against detrended SOI.

```
xbomsoi$detrendSOI <-
  with(xbomsoi, SOI - trendSOI + mean(trendSOI))
```

Figure 9.7A plots `mdb3rtRain` against `SOI`, while Figure 9.7B gives the equivalent plot for the detrended series.[4] Both relationships seem acceptably close to linear. Here, we model the relationship without the detrending. A linear relationship will be assumed. Figures 9.6 and 9.7 might suggest that the relationship for the detrended series will not be much different. We can check this directly. A further matter to check is the suggestion, in Figure 9.7A, of a quadratic relationship.

The time series structure of the data will almost inevitably lead to correlated errors that should be modeled or at least investigated, in order to obtain realistic standard errors for model parameters and to make accurate inferences.

A first step may be to fit a line as for uncorrelated errors; then using the residuals to get an initial estimate of the error structure. Figure 9.8 shows the autocorrelation structure and partial autocorrelation structure of the residuals. The code is:

```
par(mfrow=c(1,2))
acf(resid(lm(mdb3rtRain ~ SOI, data = xbomsoi)), main="")
pacf(resid(lm(mdb3rtRain ~ SOI, data = xbomsoi)), main="")
par(mfrow=c(1,1))
```

Here now is the model fit:

```
> ## Use auto.arima() to fit model
> (mdb.arima <- with(xbomsoi, auto.arima(mdb3rtRain, xreg=SOI)))
Series: mdb3rtRain
```

[4]
```
## Plot cube root of Murray-Darling basin rainfall vs SOI
oldpar <- par(mfrow=c(1,2), pty="s")
plot(mdb3rtRain ~ SOI, data = xbomsoi, ylab = "Rainfall (cube root scale)", yaxt="n")
rainpos <- pretty(xbomsoi$mdbRain, 6)
axis(2, at = rainpos^0.33, labels=paste(rainpos))
with(xbomsoi, lines(lowess(mdb3rtRain ~ SOI, f=0.1)))
plot(detrendRain ~ detrendSOI, data = xbomsoi,
     xlab="Detrended SOI", ylab = "Detrended rainfall", yaxt="n")
axis(2, at = rainpos^0.33, labels=paste(rainpos))
with(xbomsoi, lines(lowess(detrendRain ~ detrendSOI, f=0.1)))
par(oldpar)
```

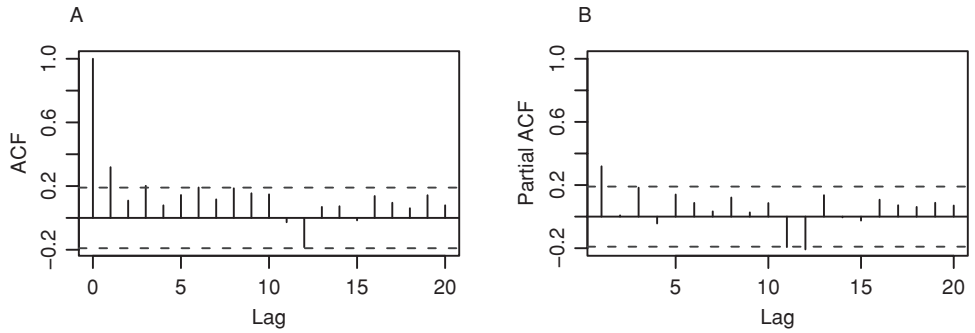


Figure 9.8 Panel A shows the autocorrelation function for the residuals from the regression of `mdb3rtRain` (= `mdbRain^(1/3)`) on `SOI`. Panel B shows the partial autocorrelation function.

```
ARIMA(0,1,1)
. . . .
Coefficients:
          ma1     SOI
      -0.934   0.041
s.e.   0.046   0.007

sigma^2 estimated as 0.274:  log likelihood = -84.37
AIC = 174.7   AICc = 175.0   BIC = 182.8
```

The default search strategy uses starting values of $p = 2$ and $q = 2$. It tests the result of changing `p`, `d` and `q` in turn by 1 at each step. It tests also the inclusion of a drift term. Note that the model selection process had other restrictions; see `help(auto.arima)`.

Now refer back to Figure 9.8. Any suggestion of non-stationarity seems very weak. We therefore fit, for comparison, a model that has `d=0`, thus:

```
> ## Fit undifferenced model
> (mdb0.arima <- with(xbomsoi,
+                     auto.arima(mdb3rtRain, xreg=SOI, d=0)))
Series: mdb3rtRain
ARIMA(0,0,0) with non-zero mean
. . . .
Coefficients:
      intercept    SOI
          7.603  0.040
s.e.      0.050  0.007

sigma^2 estimated as 0.275:  log likelihood = -84.26
AIC = 174.5   AICc = 174.7   BIC = 182.6
Series: mdb3rtRain
```

The AIC statistic is slightly smaller. The two models give very similar regression coefficients (0.041, 0.040), with very similar standard errors. Rainfall increases by about 0.04, in units of $\texttt{mdbRain}^{0.33}$, for each unit increase in `SOI`. At the low end of the scale (a rainfall

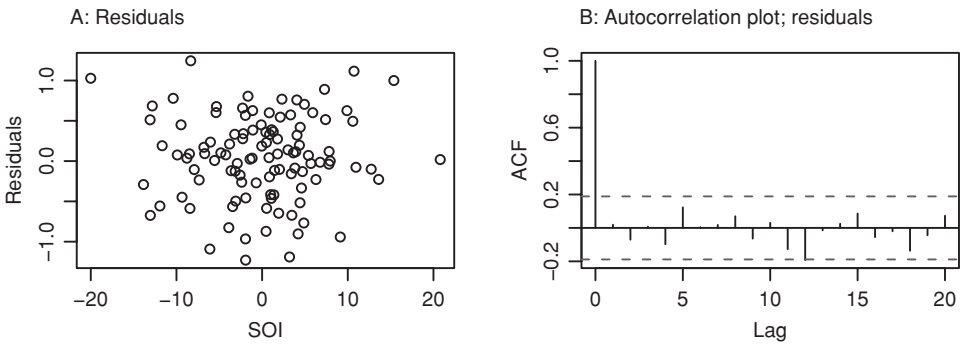


Figure 9.9 Panel A plots residuals from the ARIMA model for `mdbRain` against `SOI`. Panel B shows the autocorrelation plot for the residuals.

of about 256 mm), a unit increase in `SOI` increases the rainfall by about 4.9 mm. At the high end of the scale (821 mm), the estimated increase is about 10.6 mm.[5]

Figure 9.9 plots the residuals from this model against `SOI`, and shows also the autocorrelation plot of the residuals. These plots seem unexceptional.

*The `northRain` series

The plot of `NA3rtRain` (= `northRain^(1/3)`) against `Year`, shown in Figure 9.10A, is flat until about 1950, after which it trends upwards. Figure 9.10B shows a clear relationship between `NA3rtRain` and `SOI`.

Is there any evidence, additionally, for an effect from levels of atmospheric carbon dioxide that have increased over time? Figure 9.10C takes residuals from a resistant regression of `NA3rtRain` against `SOI`, and plots them against residuals from a resistant regression of `CO2` against `SOI`. Figure 9.10C is the appropriate graph for checking whether `CO2` levels explain some further part of the changes in rainfall.

Use of resistant regression avoids potential problems with apparent outliers; these are more likely because the error structure has not at this point been properly modeled. The analyst is in a bind, from which there is no easy escape. Until fixed effects have been accounted for, it is hazardous to try to estimate the error structure. Until the error structure has been accounted for, there is no sound inferential basis for deciding what fixed effects should be included.

Code for Figures 9.10 A, B and C is:

```
par(mfrow=c(2,2))
## Panel A
NA3rtRain <- bomregions$northRain^(1/3)
with(bomregions, plot(NA3rtRain ~ Year, yaxt="n",
                      xlab="", ylab="northRain (mm)"))
with(bomregions, lines(lowess(NA3rtRain ~ Year, f=0.75)))
```

```
> ## Increase in mdbRain, given a unit increase in SOI
> with(xbomsoi, (range(mdbRain)^(1/3) + 0.0427)^3 - range(mdbRain))
[1] 4.9 10.6
```

[5]

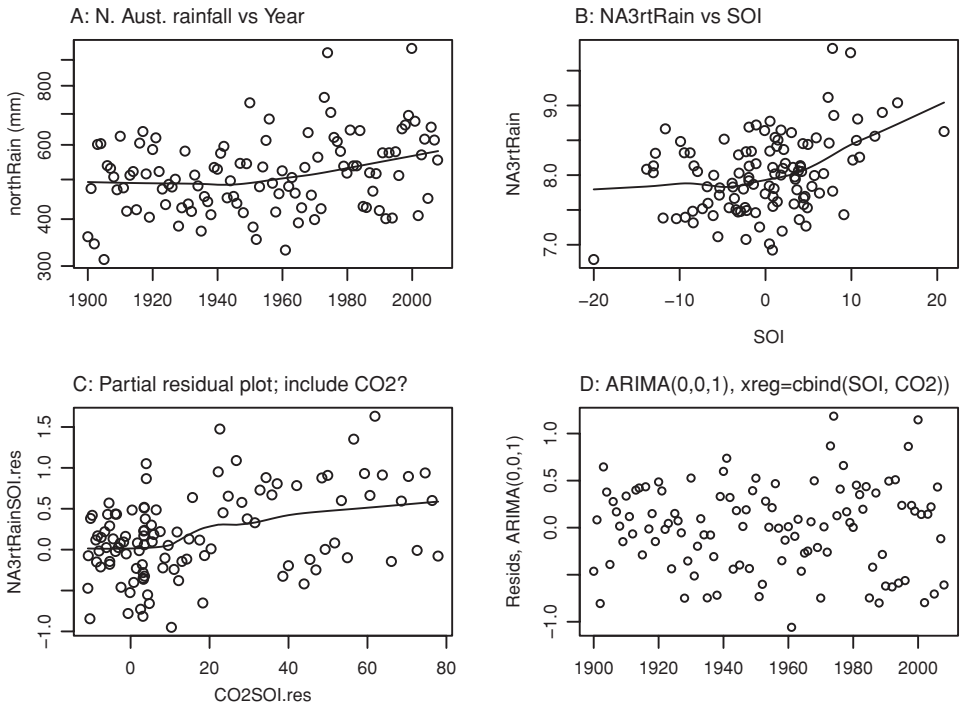


Figure 9.10 The x-ordinate in all four panels is `Year`. Panel A plots northern Australian rainfall on a cube root scale. Panel B plots `NA3rtRain` (= `northRain^(1/3)`) against `SOI`. Panel C is a partial residual plot, suitable for checking whether `CO2` should be included in the regression. Both sets of residuals are from a resistant regression. Panel D plots residuals, from an ARIMA(0,0,1) model for `NA3rtRain`, with `SOI` and `CO2` as regressors.

```
yloc <- pretty(bomregions$northRain)
axis(2, at=yloc^(1/3), labels=paste(yloc))
## Panel B
plot(NA3rtRain ~ SOI, data=bomregions)
lines(lowess(NA3rtRain ~ bomregions$SOI, f=0.75))
## Panel C
NA3rtRainSOI.res <- resid(lm(NA3rtRain ~ SOI, data=bomregions))
CO2SOI.res <- resid(lm(CO2 ~ SOI, data=bomregions))
plot(NA3rtRainSOI.res ~ CO2SOI.res)
lines(lowess(NA3rtRainSOI.res ~ CO2SOI.res, f=0.75))
```

Code for fitting the ARIMA model that includes both `SOI` and `CO2` as explanatory variables, with output, is:

```
(north.arima <- with(bomregions,
+                    auto.arima(NA3rtRain, xreg=cbind(SOI, CO2))))
Series: NA3rtRain
ARIMA(0,0,1) with non-zero mean
. . . .
Coefficients:
```

```
        ma1  intercept    SOI    CO2
      0.204      5.400  0.035  0.008
s.e.  0.102      0.694  0.007  0.002

sigma^2 estimated as 0.206:  log likelihood = -68.46
AIC = 146.9   AICc = 147.5   BIC = 160.4
> ## Now plot the residuals, as in Panel D
> plot(unclass(resid(north.arima)) ~ Year, type="p",
+      data= bomregions)
```

(Note that the result given in the `intercept` column is really an estimate of the series mean and not an intercept in the usual sense.)

Any measure that remained pretty much constant until around 1950, thereafter trending upwards, would do just as well as `CO2`. An obvious such measure is northern Australian annual average temperature (`northAVt`). If it is accepted that changes in `CO2` level are the reason for the temperature increase, then `northAVt` is a proxy for `CO2` concentration.

The variable `SOI` explains around 20% of the variation in `NA3rtRain`, additional to that explained by `CO2`. This is found by taking the estimate $\sigma^2 = 0.206$ from the model that was fitted above, and comparing it with `sigma^2` from the model where `CO2` is the only explanatory variable:

```
> ## Fit a constant mean model with the same correlation structure
> with(bomregions, arima(x = NA3rtRain, order = c(0, 0, 1), xreg=CO2))
. . . .
sigma^2 estimated as 0.258:  log likelihood = -80.8
. . . .
## Now calculate proportion of variance explained
> (.258-.206)/.258
[1] 0.202
```

Other checks and computations

Other checks include: checking for correlations among the residuals, checking an uncorrelated version of the residuals (the "innovations") for departures from normality (including outliers), and checking for possible non-linearity in covariate effects. Thus, consider again the models `north.arima` and (for the non-linearity check) `mdb.arima`.

The `Box.test()` function provides a so-called *portmanteau* test of whiteness (i.e., lack of autocorrelation) of the residuals. That is, it checks whether there is overall, evidence of correlation at one or more lags up to and including the specified lag m. Here, it will be used to check whether our use of the default arguments for `auto.arima()`, and the search strategy used by that function, may have unduly limited the range of models considered.

The test statistic (we use the Ljung–Box version where the default is Box–Pierce) is a weighted sum of squares of the first m sample autocorrelation estimates. It has an approximate chi-squared distribution on m degrees of freedom, provided there really is no autocorrelation among the residuals. For use of this test, be sure to specify the maximum lag m (the default is 1, not usually what is wanted). The output is:

```
> north.arima <- with(bomregions,
+                     auto.arima(NA3rtRain, xreg=cbind(SOI, CO2)))
```

```
> Box.test(resid(north.arima), lag=20, type="Ljung")

        Box-Ljung test

data:  resid(north.arima)
X-squared = 31.5, df = 20, p-value = 0.04866
```

There may be some further autocorrelation structure for which the model chosen by `auto.arima()` has not accounted. The autocorrelation and partial autocorrelation plots of residuals from `north.arima` both show a substantial negative correlation at a lag of 12. See Exercise 10 at the end of the chapter for further development.

Departures from normality (including outliers) have the potential to complicate estimation of the ARIMA modeling structure. A check on normality is therefore very desirable:

```
## Examine normality of estimates of "residuals" ("innovations")
qqnorm(resid(north.arima, type="normalized"))
```

Figure 9.7 suggested that a straight line regression model was adequate for the Murray–Darling basin rainfall data. We may however wish to carry out a formal check: on whether a squared (`SOI^2`) term is justified. The code is:

```
> (mdb2.arima <- with(xbomsoi, auto.arima(mdb3rtRain,
+                     xreg=poly(SOI,2))))
Series: mdb3rtRain
ARIMA(0,1,3)
. . . .
Coefficients:
         ma1     ma2    ma3      1     2
      -0.972  -0.035  0.074  2.991  0.96
s.e.   0.109   0.168  0.115  0.518  0.55

sigma^2 estimated as 0.264:  log likelihood = -81.63
AIC = 175.3   AICc = 176.1   BIC = 191.3
```

The AIC statistic is slightly larger than for the model that had only the linear term. Additionally, a different ARIMA structure has been chosen.

9.3* Non-linear time series

In the ARMA models so far considered, the error structures have been constructed from linear combinations of the innovations, and the innovations have been i.i.d. normal random variables. Such models are unable to capture the behavior of series where the variance fluctuates widely with time, as happens for many financial and economic time series.

ARCH (autoregressive conditionally heteroscedastic) and GARCH (generalized ARCH) models have been developed to meet these requirements. The principal idea behind a GARCH model is that there is an underlying (or hidden) process which governs the variance of the noise term (i.e., ε_t) while ensuring that these noise terms at different times remain uncorrelated. Market participants will, it is argued, quickly identify and

take advantage of any correlation patterns that seem more than transient, hence negating them.

Perhaps the simplest example is a model with normal ARCH(1) errors. In such a model, the error term at the current time step is normally distributed with mean 0 and with a variance linearly related to the square of the error at the previous time step. In other words, squares of noise terms form an autoregressive process of order 1. Thus, an AR(1) process with ARCH(1) errors is given by

$$X_t = \mu + \alpha(X_{t-1} - \mu) + \varepsilon_t$$

where ε_t is normal with mean 0 and variance $\sigma_t^2 = \alpha_0 + \alpha\varepsilon_{t-1}^2$. The error terms ε_t are uncorrelated, while their squares have serial correlations with the squares of historical values of the error term.

GARCH models are an extension of ARCH models. In a GARCH model of order (p, q), σ_t^2 is the sum of two terms: (1) a linear function both of the previous p squares of earlier errors, as for an ARCH model of order p and (2) a linear function of the variances of the previous q error terms.

The function `garch()` (in the *tseries* package; Trapletti, A. and Hornik, K., 2008) allows for estimation of the mean and the underlying process parameters for a given time series by maximum likelihood, assuming normality. Note also the function `white.test()` which can be used to test for non-linearity, either in the dependence of the error term on earlier errors, or in residuals from a regression that includes a time series term. We caution that the results of such tests should be interpreted with care; such non-linearity can manifest itself in innumerable ways, and such tests will not necessarily be sensitive to the particular type of non-linearity present in a particular data set.

The following code may be used to simulate an ARCH(1) process with $\alpha_0 = 0.25$ and $\alpha_1 = 0.95$:

```
x <- numeric(999)  # x will contain the ARCH(1) errors
x0 <- rnorm(1)
for (i in 1:999){
    x0 <- rnorm(1, sd=sqrt(.25 + .95*x0^2))
    x[i] <- x0
    }
```

[Note that because the initial value is not quite set correctly, a "burn-in" period is required for this to settle down to a close approximation to an ARCH series.]

We can use the `garch()` function to estimate the parameters from the simulated data. Note that the `order` argument specifies the number of MA and AR terms, respectively. With a longer time series, we would expect the respective estimates to converge towards 0.25 and 0.95.

```
> library(tseries)
> garch(x, order = c(0, 1), trace=FALSE)

Coefficient(s):
    a0      a1
0.2412  0.8523
```

The *fSeries* package, which is part of the Rmetrics suite (Würtz, 2004), substantially extends the abilities in *tseries*.

9.4 Further reading

Chatfield (2003a) is a relatively elementary introduction to time series. Brockwell and Davis (2002) is an excellent and practically oriented introduction, with more mathematical detail than Chatfield. Diggle (1990) offers a more advanced practically oriented approach. See also Shumway and Stoffer (2006), which uses R for its examples.

As our discussion has indicated, ARIMA processes provide an integrated framework that include AR and ARMA processes as special cases. Their use in time series modeling and forecasting was popularized by Box and Jenkins in the late 1960s; hence they are often called Box–Jenkins models.

State space modeling approaches are an important alternative. They provide a theoretical basis for the widely popular exponential forecasting methodology, and for various extensions of exponential forecasting. Methods of this type are implemented in the *forecast* package. See Ord *et al.* (1997), Hyndman *et al.* (2002, 2008), and Hyndman and Khandakar (2008). These approaches are intuitively appealing because they discount information from the remote past. Hyndman *et al.* (2008) give an interesting and insightful comparison of a number of different forecasting approaches, in which methods of this type do well. Related methods are implemented in the `StructTS()` and `HoltWinters()` functions in the *stats* package.

On ARCH and GARCH models, see Gourieroux (1997), the brief introduction in Venables and Ripley (2002, pp. 414–418), and references listed there. See also the references given on the help page for the `garch()` function.

Spatial statistics

We have noted that time series ideas can be applied without modification to spatial series in one dimension. In fact, it is also possible to handle higher-dimensional spatial data. Geostatistics, and spatial modeling in general, is concerned with notions of spatial autocorrelation. Kriging is a widely used multi-dimensional smoothing method. This gives best linear unbiased estimates in a spatial context. We direct the interested reader to the *spatial* library and the references given there.

Other time series models and packages

Consult the CRAN Task View for time series analysis for summary details of the extensive range of time series packages that are available. The abilities extend far beyond what has been described in this chapter.

The following is a very limited selection: *pear*, for periodic autoregressive models, including methods for plotting periodic time series; *zoo*, for irregular time series; *fracdiff*, for fractionally differenced ARIMA "long memory" processes, where the correlation between time points decays very slowly as the points move apart in time; *strucchange*, for estimating and testing for change points; and the bundle of packages *dse*, for multivariate ARMA, state space modeling, and associated forecasting.

References for further reading

Brockwell, P. and Davis, R. A. 2002. *Time Series: Theory and Methods,* 2nd edn.

Chatfield, C. 2003a. *The Analysis of Time Series: an Introduction,* 2nd edn.

Diggle, P. 1990. *Time Series: a Biostatistical Introduction.*

Gourieroux, C. 1997. *ARCH Models and Financial Applications*.

Hyndman, R. J. and Khandakar, Y. 2008. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* 27(3): 1– 22.

Hyndman, R. J., Koehler, A. B., Snyder, R. D. and Grose, S. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* 18: 439–54.

Hyndman, R. J., Koehler, A. B., Ord, J. K. and Snyder, R. D. 2008. *Forecasting with Exponential Smoothing: The State Space Approach*.

Ord, J. K., Koehler, A. B. and Snyder, R. D. 1997. Estimation and prediction for a class of dynamic nonlinear statistical models. *Journal of the American Statistical Association* 92: 1621–9.

Shumway, R. and Stoffer, D. 2006. *Time Series Analysis and Its Applications: With R Examples*.

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

9.5 Exercises

1. A time series of length 100 is obtained from an AR(1) model with $\sigma = 1$ and $\alpha = -0.5$. What is the standard error of the mean? If the usual $\sigma/\sqrt{n}$ formula were used in constructing a confidence interval for the mean, with σ defined as in Subsection 9.1.3, would it be too narrow or too wide?

2. Use the `ar` function to fit the second-order autoregressive model to the Lake Huron time series.

3. Repeat the analysis of Section 9.2, replacing `avrain` by: (i) `southRain`, i.e., annual average rainfall in southern Australia; (ii) `northRain`, i.e., annual average rainfall in northern Australia.

4. In the calculation
```
Box.test(resid(lm(detrendRain ~ detrendSOI, data = xbomsoi)),
         type="Ljung-Box", lag=20)
```
try the test with `lag` set to values of 1 (the default), 5, 20, 25 and 30. Comment on the different results.

5.* Use the `arima.sim` function in the `ts` library to simulate 100 000 values from an AR(1) process with $\alpha = -0.5$. Now break this up into 1000 series of length 100. If `x` is the series, a straightforward way to do this is to set
```
xx <- matrix(x, ncol=1000)
```
Now use the function `apply()` (Subsection 14.9.5) to find the means for each of these series of length 100. Compare

$$\frac{\sum(X_t - \bar{X})^2}{n-1}$$

with $\text{var}[\bar{X}]$ estimated from the formula $\frac{\sigma^2}{n(1-\alpha)}$.

For comparison, check the effect of using $\frac{\text{var}[X]}{n}$ to estimate the variance. First calculate the ordinary sample variance for each of our 1000 series. Then compute the average of these variance

estimates and divide by the sample size, 100. (This gives a value that is also close to that predicted by the theory, roughly three times larger than the value that was obtained using the correct formula.)

6. Sugar content in cereal is monitored in two ways: a lengthy lab analysis and by using quick, inexpensive high performance liquid chromatography. The data in `frostedflakes` (*DAAG*) come from 101 daily samples of measurements taken using the two methods.

   (a) Obtain a vector of differences between the pairs of measurements.
   (b) Plot the sample autocorrelation function of the vector of differences. Would an MA(1) model be more realistic than independence?
   (c) Compute a confidence interval for the mean difference under the independence assumption and under the MA(1) assumption.

7.* Take first differences of the logarithms of the first component of the time series object `ice.river` (*tseries* package), i.e.,

```
library(tseries)
data(ice.river)
river1 <- diff(log(ice.river[, 1]))
```

Using `arima()`, fit an ARMA(1,2) model to `river1`. Plot the residuals. Do they appear to have a constant variance? Test the residuals for non-linearity.

8. Repeat, for each of the remaining regional rainfall series in the data frame `bomregions`, an analysis of the type presented in Section 9.2.

9. Repeat, for each of the regional temperature series in Section 9.2, an analysis of the type presented in Section 9.2. Is there a comparable dependence on `SOI`?
   [NB: It will be necessary to omit the first 10 rows of the data frame, where the temperature data are absent.]

10

Multi-level models and repeated measures

This chapter further extends the discussion of models that are a marked departure from the independent errors models of Chapters 5 to 8. In the models that will be discussed in this chapter, there is a hierarchy of variation that corresponds to groupings within the data. The groups are nested. For example, students might be sampled from different classes, that in turn are sampled from different schools. Or, crop yields might be measured on multiple parcels of land at each of a number of different sites.

After fitting such models, predictions can be made at any of the given levels. For example, crop yield could be predicted at new sites, or at new parcels. Prediction for a new parcel at one of the existing sites is likely to be more accurate than a prediction for a totally new site. Multi-level models, i.e., models which have multiple *error* (or *noise*) terms, are able to account for such differences in predictive accuracy.

Repeated measures models are multi-level models where measurements consist of multiple profiles in time or space; each profile can be viewed as a time series. Such data may arise in a clinical trial, and animal or plant growth curves are common examples; each "individual" is measured at several different times. Typically, the data exhibit some form of time dependence that the model should accommodate.

By contrast with the data that typically appear in a time series model, repeated measures data consist of multiple profiles through time. Relative to the length of time series that is required for a realistic analysis, each individual repeated measures profile can and often will have values for a small number of time points only. Repeated measures data have, typically, multiple time series that are of short duration.

This chapter will make a foray into the types of models that were discussed above, starting with multi-level models. An introduction to repeated measures modeling then follows. Ideas that will be central to the discussion are:

- fixed and random effects,
- variance components, and their connection, in special cases, with expected values of mean squares,
- the specification of mixed models with a simple error structure,
- sequential correlation in repeated measures profiles.

Multi-level model and repeated measures analyses will use the package *lme4*, which must be installed. The package *lme4* is a partial replacement for the older *nlme* package. The function `lmer()`, from *lme4*, is a more flexible and powerful replacement for `lme()`,

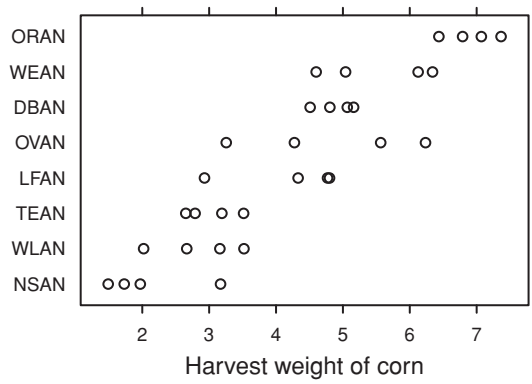


Figure 10.1 Corn yields for four parcels of land in each of eight sites on the Caribbean island of Antigua. Data are in Table 10.1. They are a summarized version (parcel measurements are block means) of a subset of data given in Andrews and Herzberg (1985, pp. 339–353). Sites have been reordered according to the magnitude of the site means.

from *nlme*. The data set `Orthodont` that is used for the analyses of Subsection 10.6.2, and several data sets that appear in the exercises, are in the *MEMSS* package.

10.1 A one-way random effects model

An especially simple multi-level model is the random effects model for the one-way layout. Thus, consider the data frame `ant111b` in the *DAAG* package, based on an agricultural experiment on the Caribbean island of Antigua. Corn yield measurements were taken on four parcels of land within each of eight sites. Figure 10.1 is a visual summary.

Code for Figure 10.1 is:

```
library(lattice); library(DAAG)
Site <- with(ant111b, reorder(site, harvwt, FUN=mean))
stripplot(Site ~ harvwt, data=ant111b, scales=list(tck=0.5),
          xlab="Harvest weight of corn")
```

Figure 10.1 suggests that, as might be expected, parcels on the same site will be relatively similar, while parcels on different sites will be relatively less similar. A farmer whose farm was close to one of the experimental sites might take data from that site as indicative of what he/she might expect. In other cases it may be more appropriate for a farmer to regard his/her farm as a new site, distinct from the experimental sites, so that the issue is one of generalizing to a new site. Prediction for a new parcel at one of the existing sites is more accurate than prediction for a totally new site.

There are two levels of random variation. They are site, and parcel within site. Variation between sites may be due, for example, to differences in elevation or proximity to bodies of water. Within a site, one might expect different parcels to be somewhat similar in terms of elevation and climatic conditions; however, differences in soil fertility and drainage may still have a noticeable effect on yield. (Use of information on such effects, not available as part of the present data, might allow more accurate modeling.)

The model will need: (a) a random term that accounts for variation within sites, and (b) a second superimposed random term that allows variability between parcels that are on different sites to be greater than variation between parcels within sites. The different random terms are known as *random effects*.

The model can be expressed as:

$$\text{yield} = \text{overall mean} + \underset{\text{(random)}}{\text{site effect}} + \underset{\text{(random)}}{\text{parcel effect (within site).}} \tag{10.1}$$

Because of the balance (there are four parcels per site), analysis of variance using `aov()` is entirely satisfactory for these data. It will be instructive, in Subsection 10.1.3 below, to set results from use of `aov()` alongside results from the function `lmer()` in the package *lme4* (Bates, 2005). The comparison is between a traditional analysis of variance approach, which is fine for data from a balanced experimental design, and a general multi-level modeling approach that can in principle handle both balanced and unbalanced designs.

10.1.1 Analysis with `aov()`

In the above model, the overall mean is assumed to be a fixed constant, while the site and parcel effects are both assumed to be random. In order to account for the two levels of variation, the model formula must include an `Error(site)` term, thus:

```
library(DAAG)
ant111b.aov <- aov(harvwt ~ 1 + Error(site), data=ant111b)
```

Explicit mention of the "within-site" level of variation is unnecessary. (Use of the error term `Error(site/parcel)`, which explicitly identifies parcels within sites, is however allowed.) Output is:

```
> summary(ant111b.aov)

Error: site
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  7  70.34   10.05

Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 24 13.861   0.578
```

The analysis of variance (*anova*) table breaks the total sum of squares about the mean into two parts – variation within sites, and variation between site means. Since there are eight sites, the variation between sites is estimated from seven degrees of freedom, after estimating the overall mean. Within each site, estimation of the site mean leaves three degrees of freedom for estimating the variance for that site. Three degrees of freedom at each of eight sites yields 24 degrees of freedom for estimating within-site variation.

Table 10.1 *The leftmost column has harvest weights (*`harvwt`*), for the parcels in each site, for the Antiguan corn data. Each of these harvest weights can be expressed as the sum of the overall mean (= 4.29), site effect (fourth column), and residual from the site effect (final column). The information in the fourth and final columns can be used to generate the sums of squares and mean squares for the analysis of variance table.*

		Site		
Site	Parcel measurements	means	effects	Residuals from site mean
DBAN	5.16, 4.8, 5.07, 4.51	4.88	+0.59	0.28, −0.08, 0.18, −0.38
LFAN	2.93, 4.77, 4.33, 4.8	4.21	−0.08	−1.28, 0.56, 0.12, 0.59
NSAN	1.73, 3.17, 1.49, 1.97	2.09	−2.2	−0.36, 1.08, −0.6, −0.12
ORAN	6.79, 7.37, 6.44, 7.07	6.91	+2.62	−0.13, 0.45, −0.48, 0.15
OVAN	3.25, 4.28, 5.56, 6.24	4.83	+0.54	−1.58, −0.56, 0.73, 1.4
TEAN	2.65, 3.19, 2.79, 3.51	3.03	−1.26	−0.39, 0.15, −0.25, 0.48
WEAN	5.04, 4.6, 6.34, 6.12	5.52	+1.23	−0.49, −0.93, 0.81, 0.6
WLAN	2.02, 2.66, 3.16, 3.52	2.84	−1.45	−0.82, −0.18, 0.32, 0.68

Interpreting the mean squares

The division of the sum of squares into two parts mirrors the two different types of prediction that can be based on these data.

First, suppose that measurements are taken on four new parcels at one of the existing sites. How much might the mean of the four measurements be expected to vary, between one such set of measurements and another. For this, the only source of uncertainty is parcel-to-parcel variation within the existing site. Recall that standard errors of averages can be estimated by dividing the (within) residual mean square by the sample size (in this case, four), and taking the square root. Thus the relevant standard error is $\sqrt{0.578/4} = 0.38$. (Note that this is another form of the pooled variance estimate discussed in Chapter 4.)

Second, for prediction of an average of four parcels at some different site, distinct from the original eight, the relevant standard error can be calculated in the same way, but using the between-site mean square; it is $\sqrt{10.05/4} = 1.6$.

Details of the calculations

This subsection may be omitted by readers who already understand the mean square calculations. Table 10.1 contains the data and gives an indication of the mean square calculations used to produce the anova table.

First, the overall mean is calculated. It is 4.29 for this example. Then site means are calculated using the parcel measurements. Site effects are calculated by subtracting the overall mean from the site means. The parcel effects are the residuals after subtracting the site means from the individual parcel measurements.

The between-site sum of squares is obtained by squaring the site effects, summing, and multiplying by four. This last step reflects the number of parcels per site. Dividing by the degrees of freedom ($8 - 1 = 7$) gives the mean square.

The within-site sum of squares is obtained by squaring the residuals (parcel effects), summing, and dividing by the degrees of freedom ($8 \times (4 - 1) = 24$).

Practical use of the analysis of variance results

Treating site as random when we do the analysis does not at all commit us to treating it as random for purposes of predicting results from a new site. Rather, it allows us this option, if this seems appropriate. Consider how a person who has newly come to the island, and intends to purchase a farming property, might assess the prospects of a farming property that is available for purchase. Two extremes in the range of possibilities are:

1. The property is similar to one of the sites for which data are available, so similar in fact that yields would be akin to those from adding new parcels that together comprise the same area on that site.
2. It is impossible to say with any assurance where the new property should be placed within the range of results from experimental sites. The best that can be done is to treat it as a random sample from the population of all possible sites on the island.

Given adequate local knowledge (and ignoring changes that have taken place since these data were collected!) it might be possible to classify most properties on the island as likely to give yields that are relatively close to those from one or more of the experimental sites. Given such knowledge, it is then possible to give a would-be purchaser advice that is more finely tuned. The standard error (for the mean of four parcels) is likely to be much less than 1.6, and may for some properties be closer to 0.38. In order to interpret analysis results with confidence, and give the would-be purchaser high-quality advice, a fact-finding mission to the island of Antigua may be expedient!

Random effects versus fixed effects

The random effects model bears some resemblance to the one-way model considered in Section 4.4. The important difference is that in Section 4.4 the interest was in differences between the *fixed* levels of the nutrient treatment that were used in the experiment. Generalization to other possible nutrient treatments was not of interest, and would not have made sense. The only predictions that were possible were for nutrient treatments considered in the study.

The random effects model allows for predictions at two levels: (1) for agricultural yield at a new location within an existing site, or (2) for locations in sites that were different from any of the sites that were included in the original study.

Nested factors – a variety of applications

Random effects models apply in any situation where there is more than one level of random variability. In many situations, one source of variability is *nested* within the other – thus parcels are nested within sites.

Other examples include: variation between houses in the same suburb, as against variation between suburbs; variation between different clinical assessments of the same patients, as

against variation between patients; variation within different branches of the same business, as against variation between different branches; variations in the bacterial count between subsamples of a sample taken from a lake, as opposed to variation between different samples; variation between the drug prescribing practices of clinicians in a particular specialty in the same hospital, as against variation between different clinicians in different hospitals; and so on. In all these cases, the accuracy with which predictions are possible will depend on which of the two levels of variability is involved. These examples can all be extended in fairly obvious ways to include more than two levels of variability.

Sources of variation can also be *crossed*. For example, different years may be crossed with different sites. Years are not nested in sites, nor are sites nested in years. In agricultural yield trials these two sources of variation may be comparable; see for example Talbot (1984).

10.1.2 A more formal approach

Consider now a formal mathematical description of the model. The model is:

$$y_{ij} = \mu + \underset{\text{(site, random)}}{\alpha_i} + \underset{\text{(parcel, random)}}{\beta_{ij}} \quad (i = 1, \ldots, 8; j = 1, \ldots, 4) \quad (10.2)$$

with $\text{var}[\alpha_i] = \sigma_L^2$, $\text{var}[\beta_{ij}] = \sigma_W^2$. The quantities σ_L^2 (L = location, another term for site) and σ_W^2 (W = within) are referred to as *variance components*.

Variance components allow inferences that are not immediately available from the information in the analysis of variance table. Importantly, the variance components provide information that can help design another experiment.

Relations between variance components and mean squares

The expected values of the mean squares are, in suitably balanced designs such as this, linear combinations of the variance components. The discussion that now follows demonstrates how to obtain the variance components from the analysis of variance calculations. In an unbalanced design, this is not usually possible.

Consider, again, prediction of the average of four parcels within the ith existing site. This average can be written as

$$\bar{y}_i = \mu + \alpha_i + \bar{\beta}_i$$

where $\bar{\beta}_i$ denotes the average of the four parcel effects within the ith site. Since μ and α_i are constant for the ith site (in technical terms, we *condition* on the site being the ith), $\text{var}[\bar{y}_i]$ is the square root of $\text{var}[\bar{\beta}_i]$, which equals $\sigma_W/\sqrt{4}$.

In the `aov()` output, the expected mean square for `Error: Within`, i.e., at the within-site (between packages) level, is σ_W^2. Thus $\widehat{\sigma_W^2} = 0.578$ and $\text{SE}[\bar{y}_i]$ is estimated as $\widehat{\sigma_W}/\sqrt{4} = \sqrt{0.578/4} = 0.38$.

Next, consider prediction of the average yield at four parcels within a new site. The expected mean square at the site level is $4\sigma_L^2 + \sigma_W^2$, i.e., the between-site mean square, which in the `aov()` output is 10.05, estimates $4\sigma_L^2 + \sigma_W^2$. The standard error for the

prediction of the average yield at four parcels within a new site is

$$\sqrt{\sigma_L^2 + \sigma_W^2/4} = \sqrt{(4\sigma_L^2 + \sigma_W^2)/4}.$$

The estimate for this is $\sqrt{10.05/4} = 1.59$.

Finally note how, in this balanced case, σ_L^2 can be estimated from the analysis of variance output. Equating the expected between-site mean square to the observed mean square:

$$4\widehat{\sigma_L^2} + \widehat{\sigma_W^2} = 10.05,$$

i.e.,

$$4\widehat{\sigma_L^2} + 0.578 = 10.05,$$

so that $\widehat{\sigma_L^2} = (10.05 - 0.578)/4 = 2.37$.

Interpretation of variance components

In summary, here is how the variance components can be interpreted, for the Antiguan data. Plugging in numerical values ($\widehat{\sigma_W^2} = 0.578$ and $\widehat{\sigma_L^2} = 2.37$), take-home messages from this analysis are:

- For prediction for a new parcel at one of the existing sites, the standard error is $\widehat{\sigma_W} = \sqrt{0.578} = 0.76$.
- For prediction for a new parcel at a new site, the standard error is $\sqrt{\sigma_L^2 + \sigma_W^2} = \sqrt{2.37 + 0.578} = 1.72$.
- For prediction of the mean of n parcels at a new site, the standard error is $\sqrt{\sigma_L^2 + \sigma_W^2/n} = \sqrt{2.37 + 0.578/n}$.
  [Notice that while σ_W^2 is divided by n, σ_L^2 is not. This is because the site effect is the same for all n parcels.]
- For prediction of the total of n parcels at a new site, the standard error is $\sqrt{\sigma_L^2 n + \sigma_W^2} = \sqrt{2.37n + 0.578}$.

Additionally:

- The variance of the difference between two such parcels at the same site is $2\sigma_W^2$.
  [Both parcels have the same site effect α_i, so that var(α_i) does not contribute to the variance of the difference.]
- The variance of the difference between two parcels that are in different sites is $2(\sigma_L^2 + \sigma_W^2)$:

Thus, where there are multiple levels of variation, the predictive accuracy can be dramatically different, depending on what is to be predicted. Similar issues arise in repeated measures contexts, and in time series.

Intra-class correlation

According to the model, two observations at different sites are uncorrelated. Two observations at the same site are correlated, by an amount that has the name *intra-class correlation.* Here, it equals $\sigma_L^2/(\sigma_L^2 + \sigma_W^2)$. This is the proportion of residual variance explained by differences between sites.

Plugging in the variance component estimates, the intra-class correlation for the corn yield data is $2.37/(2.37 + 0.578) = 0.804$. Roughly 80% of the yield variation is due to differences between sites.

10.1.3 Analysis using `lmer()`

In output from the function `lmer()`, the assumption of two nested random effects, i.e., a hierarchy of three levels of variation, is explicit. Variation between sites (this appeared first in the anova table in Subsection 10.1.1) is the "lower" of the two levels. Here, the *nlme* convention will be followed, and this will be called level 1. Variation between parcels in the same site (this appeared second in the anova table, under "Residuals") is at the "higher" of the two levels, conveniently called level 2.

The modeling command takes the form:

```
library(lme4)
ant111b.lmer <- lmer(harvwt ~ 1 + (1 | site), data=ant111b)
```

The only fixed effect is the overall mean. The `(1 | site)` term fits random variation between sites. Variation between the individual units that are nested within sites, i.e., between parcels, are by default treated as random. Here is the default output:

```
> ## Output is from version 0.999375-28 of lmer
> ## Note that there is no degrees of freedom information.
> ant111b.lmer
Linear mixed model fit by REML
Formula: harvwt ~ 1 + (1 | site)
   Data: ant111b
 AIC BIC logLik deviance REMLdev
 100 105  -47.2       95    94.4
Random effects:
 Groups   Name        Variance Std.Dev.
 site     (Intercept) 2.368    1.54
 Residual             0.578    0.76
Number of obs: 32, groups: site, 8

Fixed effects:
            Estimate Std. Error t value
(Intercept)     4.29       0.56    7.66
```

Observe that, according to `lmer()`, $\widehat{\sigma_W^2} = 0.578$ and $\widehat{\sigma_L^2} = 2.368$. Observe also that $\widehat{\sigma_W^2} = 0.578$ is the mean square from the analysis of variance table. The mean square at level 1 does not appear in the output from the `lmer()` analysis, not even in this balanced case.

Fitted values and residuals in `lmer()`

In hierarchical multi-level models, fitted values can be calculated at each level of variation that the model allows. Corresponding to each level of fitted values, there is a set of residuals that is obtained by subtracting the fitted values from the observed values.

The default, and at the time of writing the only option, is to calculate fitted values and residuals that adjust for all random effects except the residual. Here, these are estimates of the site expected values. They differ slightly from the site means, as will be seen below. Such fitted values are known as BLUPs (Best Linear Unbiased Predictors). Among linear unbiased predictors of the site means, the BLUPs are designed to have the smallest expected error mean square.

Relative to the site means $\bar{y}_{i\cdot}$, the BLUPs are pulled in toward the overall mean $\bar{y}_{\cdot\cdot}$. The most extreme site means will on average, because of random variation, be more extreme than the corresponding "true" means for those sites. For the simple model considered here, the fitted value $\widehat{\alpha_i}$ for the ith site is given by

$$\widehat{\bar{y}_{i\cdot}} = \bar{y}_{\cdot\cdot} + \frac{n\widehat{\sigma_L^2}}{n\widehat{\sigma_L^2} + \widehat{\sigma_W^2}}(\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot}).$$

Shrinkage is substantial, i.e., a shrinkage factor much less than 1.0, when $n^{-1}\widehat{\sigma_W^2}$ is large relative to $\widehat{\sigma_L^2}$. (For the notation, refer back to equation (10.2).)

As a check, compare the BLUPs given by the above formula with the values from `fitted(ant111b.lmer)`:

```
> s2W <- 0.578; s2L <- 2.37; n <- 4
> sitemeans <- with(ant111b, sapply(split(harvwt, site), mean))
> grandmean <- mean(sitemeans)
> shrinkage <- (n*s2L)/(n*s2L+s2W)
> grandmean + shrinkage*(sitemeans - grandmean)
 DBAN  LFAN  NSAN  ORAN  OVAN  TEAN  WEAN  WLAN
4.851 4.212 2.217 6.764 4.801 3.108 5.455 2.925
> ##
> ## More directly, use fitted() with the lmer object
> unique(fitted(ant111b.lmer))
[1] 4.851 4.212 2.217 6.764 4.801 3.108 5.455 2.925
> ##
> ## Compare with site means
> sitemeans
 DBAN  LFAN  NSAN  ORAN  OVAN  TEAN  WEAN  WLAN
4.885 4.207 2.090 6.915 4.832 3.036 5.526 2.841
```

Observe that the fitted values differ slightly from the site means. For site means below the overall mean (4.29), the fitted values are larger (closer to the overall mean), and for site means above the overall mean, the fitted values are smaller.

Notice that `fitted()` has given the fitted values at level 1, i.e., it adjusts for the single random effect. The fitted value at level 0 is the overall mean, given by `fixef(ant111b.lmer)`. Residuals can also be defined on several levels. At level 0, they are the differences between the observed responses and the overall mean. At level 1, they

are the differences between the observed responses and the fitted values at level 1 (which are the BLUPs for the sites).

**Uncertainty in the parameter estimates*

The limits of acceptance of a likelihood ratio test for the null hypothesis of no change in a parameter value can be used as approximate 95% confidence limits for that parameter. The calculations require evaluation of what is termed the profile likelihood, here using the `profile()` method for objects created by `lmer()`:

```
> ## lme4a_0.999375-44 (Development branch); details may change
> CI95 <- confint(profile(ant111b.lmer), level=0.95)
> rbind("sigmaL^2"=CI95[1,]^2, "sigma^2"=exp(CI95[2,])^2,
+         "(Intercept)"=CI95[3,])
             2.5 % 97.5 %
sigmaL^2     0.796   6.94
sigma^2      0.344   1.08
(Intercept) 3.128   5.46
```

The version of the `confint()` method used here returned confidence intervals for σ_L (row label `.sig01`), for $\log(\sigma)$ (row label `.lsig`), and for `(Intercept)`. The `(Intercept)` is the intercept in the fitted model, which estimates the overall mean.

An alternative, once it can be made to work reliably, is a Markov Chain Monte Carlo (MCMC) approach, as implemented in the *lme4* function `mcmcsamp()`. The posterior distribution of the parameters is simulated, using a non-informative prior distribution. Credible intervals can be calculated; these are analogous to confidence intervals, but have an easier interpretation. They are, conditional on the Bayesian assumptions, probability statements. See `help(mcmcsamp)` for more details. Subsection 10.2.1 has an example where the current (at the time of writing) implementation of this methodology delivers acceptable results.

Handling more than two levels of random variation

There can be variation at each of several nested levels. Suppose, for example, that house prices (`price`) were available at samples of three-bedroom bungalows within samples of suburbs (`suburb`) located within a number of different American cities (`city`). Prices differ between cities, between suburbs within cities, and between houses within suburbs.

Using the terminology of the `lme()` function in the *nlme* package, there are three levels of variation: level 3 is house, level 2 is suburb, and level 1 is city. (The `lmer()` function is not limited to the hierarchical models to which this terminology applies, and does not make formal use of the "levels" terminology.)

Since level 1 and 2 variation must be identified in the `lmer()` function call, we would analyze such data using

```
## house.lmer <- lmer(price ~ 1 + (1|city) + (1|city:suburb))
```

Additionally, it may be necessary to replace the intercept term by one or more linear model terms that take account of such explanatory variables (these are "fixed effects") as floor area. In the examples in the next three sections, much of the interest will be on the implications of the random effects for the accuracy of fixed-effect estimates.

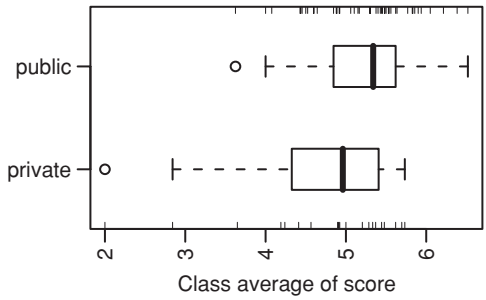


Figure 10.2 Average scores for class, compared between public and private schools.

10.2 Survey data, with clustering

The data that will now be explored are from the data frame `science` (*DAAG*). They are measurements of attitudes to science, from a survey where there were results from 20 classes in 12 private schools and 46 classes in 29 public (i.e., state) schools, all in and around Canberra, Australia. Results are from a total of 1385 year 7 students. The variable `like` is a summary score based on two of the questions. It is on a scale from 1 (dislike) to 12 (like). The number in each class from whom scores were available ranged from 3 to 50, with a median of 21.5. Figure 10.2 compares results for public schools with those for private schools.[1]

10.2.1 Alternative models

Within any one school, we might have

$$y = \text{class effect} + \text{pupil effect}$$

where y represents the attitude measure.

Within any one school, we might use a one-way analysis of variance to estimate and compare class effects. However, this study has the aim of generalizing beyond the classes in the study to all of some wider population of classes, not just in the one school, but in a wider population of schools from which the schools in the study were drawn. In order to be able to generalize in this way, we treat school (`school`), and class (`class`) within school, as random effects. We are interested in possible differences between the sexes (`sex`), and between private and public schools (`PrivPub`). The two sexes are not a sample from some larger population of possible sexes (!), nor are the two types of school (for this study at least) a sample from some large population of types of school. Thus they are fixed effects.

[1]
```
## Means of like (data frame science: DAAG), by class
classmeans <- with(science,
                   aggregate(like, by=list(PrivPub, Class), mean) )
  # NB: Class identifies classes independently of schools
  #     class identifies classes within schools
names(classmeans) <- c("PrivPub", "Class", "avlike")
attach(classmeans)
## Boxplots: class means by Private or Public school
boxplot(split(avlike, PrivPub), horizontal=TRUE, las=2,
  xlab = "Class average of score", boxwex = 0.4)
rug(avlike[PrivPub == "private"], side = 1)
rug(avlike[PrivPub == "public"], side = 3)
detach(classmeans)
```

The interest is in the specific fixed differences between males and females, and between private and public schools.

The preferred approach is a multi-level model analysis. While it is sometimes possible to treat such data using an approximation to the analysis of variance as for a balanced experimental design, it may be hard to know how good the approximation is. We specify sex (`sex`) and school type (`PrivPub`) as fixed effects, while school (`school`) and class (`class`) are specified as random effects. Class is *nested* within school; within each school there are several classes. The model is:

$$\text{y} = + \underset{\text{(fixed)}}{\text{sex effect}} + \underset{\text{(fixed)}}{\text{type (private or public)}} + \underset{\text{(random)}}{\text{school effect}} + \underset{\text{(random).}}{\text{class effect}} + \underset{\text{(random).}}{\text{pupil effect}}$$

Questions we might ask are:

- Are there differences between private and public schools?
- Are there differences between females and males?
- Clearly there are differences among pupils. Are there differences between classes within schools, and between schools, greater than pupil-to-pupil variation within classes would lead us to expect?

The table of estimates and standard errors for the coefficients of the fixed component is similar to that from an `lm()` (single-level) analysis.

```
> science.lmer <- lmer(like ~ sex + PrivPub + (1 | school) +
+                       (1 | school:class), data = science,
+                      na.action=na.exclude)
> summary(science.lmer)
Linear mixed-effects model fit by REML
Formula: like ~ sex + PrivPub + (1 | school) + (1 | school:class)
          Data: science
     AIC     BIC   logLik MLdeviance REMLdeviance
 5556.55 5582.71 -2773.27    5539.14      5546.55
Random effects:
 Groups       Name        Variance  Std.Dev.
 school:class (Intercept) 3.206e-01 5.662e-01
 school       (Intercept) 1.526e-09 3.907e-05
 Residual                 3.052e+00 1.747e+00
number of obs: 1383, groups: school:class, 66; school, 41

Fixed effects:
             Estimate Std. Error t value
(Intercept)     4.7218     0.1624  29.072
sexm            0.1823     0.0982   1.857
PrivPubpublic   0.4117     0.1857   2.217

Correlation of Fixed Effects:
            (Intr) sexm
sexm        -0.309
PrivPubpblc -0.795  0.012
```

Degrees of freedom are as follows:

- Between types of school: 41 (number of schools) $-2 = 39$.
- Between sexes: $1383 - 1$ (overall mean) $- 1$ (differences between males and females) $-$ 65 (differences between the 66 `school:class` combinations) $= 1316$.

The comparison between types of schools compares 12 private schools with 29 public schools, comprising 41 algebraically independent items of information. However because the numbers of classes and class sizes differ between schools, the three components of variance contribute differently to these different accuracies, and the 39 degrees of freedom are for a statistic that has only an approximate t-distribution. On the other hand, schools are made up of mixed male and female classes. The between-pupils level of variation, where the only component of variance is that for the `Residual` in the output above, is thus the relevant level for the comparison between males and females. The t-test for this comparison is, under model assumptions, an exact test with 1316 degrees of freedom.

There are three variance components:[2]

```
Between schools  (school)         0.00000000153
Between classes  (school:class)   0.321
Between students (Residual)       3.052
```

It is important to note that these variances form a nested hierarchy. Variation between students contributes to variation between classes. Variation between students and between classes both contribute to variation between schools. The modest-sized between-class component of variance tells us that differences between classes are greater than would be expected from differences between students alone. This will be discussed further shortly.

The between-schools component of variance is close to zero. There is, moreover, sufficient information on school-to-school variation (20 private schools, 46 public schools) that failure to account for the between-schools component of variation will not much affect inferences that may be drawn. Hence the following simpler analysis that does not account for the schools component of variance. Notice that the variance component estimates are, to two significant digits, the same as before:

```
> science1.lmer <- lmer(like ~ sex + PrivPub + (1 | school:class),
+                       data = science, na.action=na.exclude)
> summary(science1.lmer)
....
Random effects:
 Groups       Name        Variance Std.Dev.
 school:class (Intercept) 0.321    0.566
 Residual                 3.052    1.747
number of obs: 1383, groups: school:class, 66
```

[2] `## The variances are included in the output from VarCorr()`
`VarCorr(science.lmer)  # Displayed output differs slightly`
`# The between students (Residual) component of variance is`
`# attr(VarCorr(science.lmer),"sc")^2`

```
Fixed effects:
              Estimate Std. Error t value
(Intercept)     4.7218     0.1624   29.07
sexm            0.1823     0.0982    1.86
PrivPubpublic   0.4117     0.1857    2.22
....
```

Now use the function `mcmcsamp()` to get approximate 95% confidence intervals (technically, credible intervals) for the random effects:

```
> set.seed(41)
> science1.mcmc <- mcmcsamp(science1.lmer, n=1000)
> samps <- VarCorr(science1.mcmc, type="varcov")
> colnames(samps) <- c("sigma_Classˆ2", "sigmaˆ2")
> signif(HPDinterval(samps, prob=0.95), 3)
              lower upper
sigma_Classˆ2 0.148 0.43
sigmaˆ2       2.840 3.28
attr(,"Probability")
[1] 0.95
```

To what extent do differences between classes affect the attitude to science? A measure of the effect is the *intra-class correlation*, which is the proportion of variance that is explained by differences between classes. Here, it equals $0.321/(0.321 + 3.05) = 0.095$. The main influence comes from outside the class that the pupil attends, e.g., from home, television, friends, inborn tendencies, etc.

Do not be tempted to think that, because 0.321 is small relative to the within-class component variance of 3.05, it is of little consequence. The variance for the mean of a class that is chosen at random is $0.321 + 3.05/n$. Thus, with a class size of 20, the between-class component makes a bigger contribution than the within-class component. If all classes were the same size, then the standard error of the difference between class means for public schools and those for private schools would, as there were 20 private schools and 46 public schools, be

$$\sqrt{(0.321 + 3.05/n)\left(\frac{1}{20} + \frac{1}{46}\right)}.$$

From the output table of coefficients and standard errors, we note that the standard error of difference between public and private schools is 0.1857. For this to equal the expression just given, we require $n = 19.1$. Thus the sampling design is roughly equivalent to a balanced design with 19.1 pupils per class.

Figure 10.3 displays information that may be helpful in the assessment of the model. A simplified version of the code is:

```
science1.lmer <- lmer(like ~ sex + PrivPub + (1 | school:class),
                      data = science, na.action=na.omit)
ranf <- ranef(obj = science1.lmer, drop=TRUE)[["school:class"]]
flist <- science1.lmer@flist[["school:class"]]
```

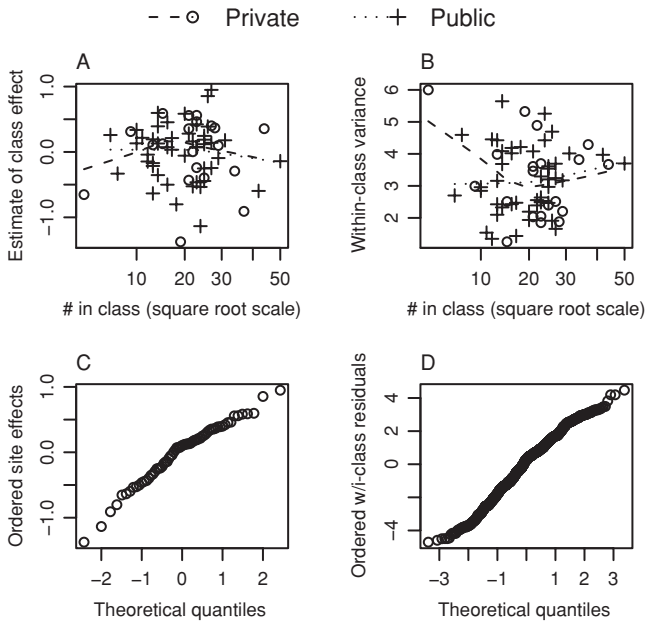


Figure 10.3 Panel A plots class effects against number in the class. Panel B plots within-class variance against number in the class. Panels C and D show normal probability plots, for the class effect and for the level 1 residuals (adjusting for the class effect), respectively.

```
privpub <- science[match(names(ranf), flist), "PrivPub"]
num <- unclass(table(flist)); numlabs <- pretty(num)
par(mfrow=c(2,2))
## Plot effect estimates vs numbers
plot(sqrt(num), ranf, xaxt="n", pch=c(1,3)[unclass(privpub)],
     xlab="# in class (square root scale)",
     ylab="Estimate of class effect")
lines(lowess(sqrt(num[privpub=="private"]),
             ranf[privpub=="private"], f=1.1), lty=2)
lines(lowess(sqrt(num[privpub=="public"]),
             ranf[privpub=="public"], f=1.1), lty=3)
axis(1, at=sqrt(numlabs), labels=paste(numlabs))
res <- residuals(science1.lmer)
vars <- tapply(res, INDEX=list(flist), FUN=var)*(num-1)/(num-2)
## Within plot variance estimates vs numbers
plot(sqrt(num), vars, pch=c(1,3)[unclass(privpub)])
lines(lowess(sqrt(num[privpub=="private"]),
             as.vector(vars)[privpub=="private"], f=1.1), lty=2)
lines(lowess(sqrt(num[privpub=="public"]),
             as.vector(vars)[privpub=="public"], f=1.1), lty=3)
## Normal probability plot of site effects
qqnorm(ranf, ylab="Ordered site effects", main="")
## Normal probability plot of residuals
qqnorm(res, ylab="Ordered w/i class residuals", main="")
par(mfrow=c(1,1))
```

Panel A shows no clear evidence of a trend. Panel B perhaps suggests that variances may be larger for the small number of classes that had more than about 30 students. Panels C and D show distributions that seem acceptably close to normal. The interpretation of panel C is complicated by the fact that the different effects are estimated with different accuracies.

10.2.2 Instructive, though faulty, analyses

Ignoring class as the random effect

It is important that the specification of random effects be correct. It is enlightening to do an analysis that is not quite correct, and investigate the scope that it offers for misinterpretation. We fit `school`, ignoring `class`, as a random effect. The estimates of the fixed effects change little.

```
> science2.lmer <- lmer(like ~ sex + PrivPub + (1 | school),
+                       data = science, na.action=na.exclude)
> science2.lmer
. . . .
Fixed effects:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.738      0.163   29.00   <2e-16
sexm            0.197      0.101    1.96    0.051
PrivPubpublic   0.417      0.185    2.25    0.030
```

This analysis suggests, wrongly, that the between-schools component of variance is substantial. The estimated variance components are:[3]

```
Between schools  0.166
Between students 3.219
```

This is misleading. From our earlier investigation, it is clear that the difference is between classes, not between schools!

Ignoring the random structure in the data

Here is the result from a standard regression (linear model) analysis, with `sex` and `PrivPub` as fixed effects:

```
> ## Faulty analysis, using lm
> science.lm <- lm(like ~ sex + PrivPub, data=science)
> summary(science.lm)$coef

             Estimate Std. Error t value Pr(>|t|)
(Intercept)     4.740     0.0996   47.62 0.000000
sexm            0.151     0.0986    1.53 0.126064
PrivPubpublic   0.395     0.1051    3.76 0.000178
```

[3] `## The numerical values can be extracted from`
`VarCorr(science2.lmer)  # The within schools (Residual) component of variance`
`# is the square of the scale parameter`

Do not believe this analysis! The SEs are too small, and the number of degrees of freedom for the comparison between public and private schools is much too large. The contrast is more borderline than this analysis suggests.

10.2.3 Predictive accuracy

The variance of a prediction of the average for a new class of n pupils, sampled in the same way as existing classes, is $0.32 + 3.05/n$. If classes were of equal size, we could derive an equivalent empirical estimate of predictive accuracy by using a resampling method with the class means. With unequal class sizes, use of the class means in this way will be a rough approximation. There were 60 classes. If the training/test set methodology is used, the 60 class means would be divided between a training set and a test set.

An empirical estimate of the within-class variation can be derived by applying a resampling method (cross-validation, or the bootstrap) to data for each individual class. The variance estimates from the different classes would then be pooled.

The issues here are important. Data do commonly have a hierarchical variance structure comparable with that for the attitudes to science data. As with the Antiguan corn yield data, the population to which results are to be generalized determines what estimate of predictive accuracy is needed. There are some generalizations, e.g., to another state, that the data cannot support.

10.3 A multi-level experimental design

The data in `kiwishade` are from a designed experiment that compared different kiwifruit shading treatments. [These data relate to Snelgar *et al.* (1992). Maindonald (1992) gives the data in Table 10.2, together with a diagram of the field layout that is similar to Figure 10.4. The two papers have different shorthands (e.g., Sept–Nov versus Aug–Dec) for describing the time periods for which the shading was applied.] Figure 10.4 shows the layout. In summary, note also:

- This is a balanced design with four vines per plot, four plots per block, and three blocks.
- There are three levels of variation that will be assumed random – between vines within plots, between plots within blocks, and between blocks.
- The experimental unit is a plot; this is the level at which the treatment was applied. We have four results (vine yields) per plot.
- Within blocks, treatments were randomly allocated to the four plots.

The northernmost plots were grouped together because they were similarly affected by shading from the sun in the north. For the remaining two blocks, shelter effects, whether from the west or from the east, were thought more important. Table 10.2 displays the data.

The `aov()` function allows calculation of an analysis of variance table that accounts for the multiple levels of variation, and allows the use of variation at the plot level to compare treatments. Variance components can be derived, should they be required, from the information in the analysis of variance table. The section will conclude by demonstrating the use of `lmer()` to calculate the variance components directly, and provide information equivalent to that from the use of `aov()`.

Table 10.2 *Data from the kiwifruit shading trial. The level names for the factor* `shade` *are mnemonics for the time during which shading was applied. Thus (*`none`*) implies no shading,* `Aug2Dec` *means "August to December", and similarly for* `Dec2Feb` *and* `Feb2May`*.* The final four columns give yield measurements in kilograms.

Block	Shade	Vine1	Vine2	Vine3	Vine4
east	none	100.74	98.05	97.00	100.31
east	Aug2Dec	101.89	108.32	103.14	108.87
east	Dec2Feb	94.91	93.94	81.43	85.40
east	Feb2May	96.35	97.15	97.57	92.45
north	none	100.30	106.67	108.02	101.11
north	Aug2Dec	104.74	104.02	102.52	103.10
north	Dec2Feb	94.05	94.76	93.81	92.17
north	Feb2May	91.82	90.29	93.45	92.58
west	none	93.42	100.68	103.49	92.64
west	Aug2Dec	97.42	97.60	101.41	105.77
west	Dec2Feb	84.12	87.06	90.75	86.65
west	Feb2May	90.31	89.06	94.99	87.27

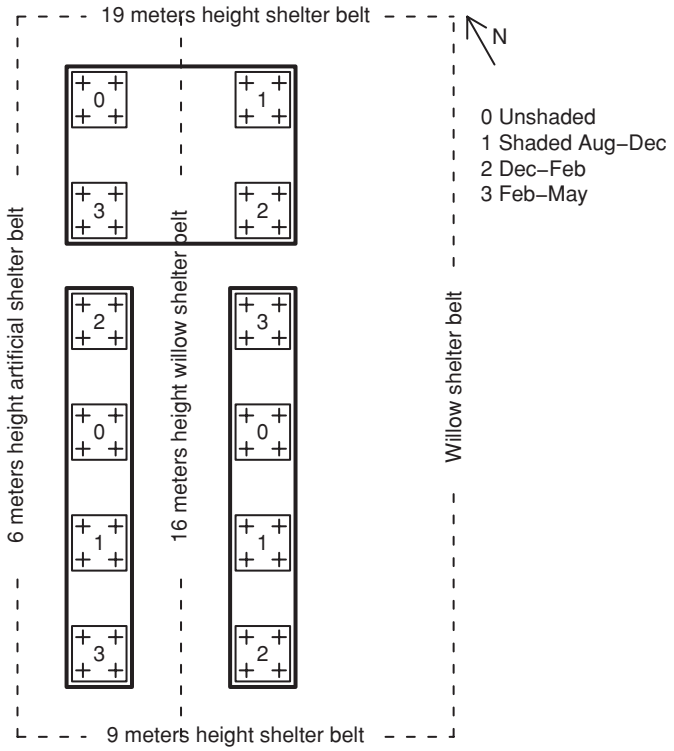


Figure 10.4 The field layout for the kiwifruit shading trial.

The model is:

$$\text{yield} = \text{overall mean} + \underset{\text{(random)}}{\text{block effect}} + \underset{\text{(fixed)}}{\text{shade effect}} + \underset{\text{(random)}}{\text{plot effect}} + \underset{\text{(random)}}{\text{vine effect.}}$$

We characterize the design thus:

Fixed effect: shade (treatment).
Random effect: vine (nested) in plot in block, or block/plot/vine.

Although block is included as a random effect, the estimate of the block component of variance has limited usefulness. On the one hand, the common location of the three blocks has exposed them to similar soil and general climate effects. On the other hand, their different orientations (N, W and E) to sun and prevailing wind will lead to systematic differences. At best, the estimate of the block component of variance will give only the vaguest of hints on the likely accuracy of predictions for other blocks.

There is some limited ability to generalize to other plots and other vines. When horticulturalists apply these treatments in their own orchards, there will be different vines, plots, and blocks. Thus, vines and plots are treated as random effects. A horticulturalist will however reproduce, as far as possible, the same shade treatments as were used in the scientific trial, and these are taken as fixed effects. There is however a caveat. In the different climate, soil, and other conditions of a different site, these effects may well be different.

10.3.1 The anova table

The model formula that is applied to `aov()` is an extension of an `lm()` style of model formula that includes an `Error()` term. Observe that each different plot within a block has a different shading treatment, so that the `block:shade` combination can be used to identify plots. Thus the two error terms that need to be specified can be written `block` and `block:shade`. There is a shorthand way to specify both of these together. Write `block/shade`, which expands into `block+block:shade`. Suitable code for the calculation is:

```
## Analysis of variance: data frame kiwishade (DAAG)
kiwishade.aov <- aov(yield ~ shade + Error(block/shade),
                     data=kiwishade)
```

The output is:

```
> summary(kiwishade.aov)

Error: block
          Df  Sum Sq Mean Sq F value Pr(>F)
Residuals  2 172.348  86.174

Error: block:shade
          Df  Sum Sq Mean Sq F value   Pr(>F)
shade      3 1394.51  464.84  22.211 0.001194
Residuals  6  125.57   20.93
```

Table 10.3 *Mean squares in the analysis of variance table. The final column gives expected values of mean squares, as functions of the variance components.*

	Df	Sum of sq	Mean sq	E[Mean sq]
`block` level	2	172.35	86.17	$16\sigma_B^2 + 4\sigma_P^2 + \sigma_V^2$
`block.plt` level				
`shade`	3	1394.51	464.84	$4\sigma_P^2 + \sigma_V^2 +$ treatment component
`residual`	6	125.57	20.93	$4\sigma_P^2 + \sigma_V^2$
`block.plt.vines` level	36	438.58	12.18	σ_V^2

```
Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 36 438.58   12.18
```

Notice the use of `summary()` to give the information that is required. The function `anova()` is, in this context, unhelpful.

Table 10.3 structures the output, with a view to making it easier to follow. The final column will be discussed later, in Subsection 10.3.2.

10.3.2 Expected values of mean squares

The final column of Table 10.3 shows how the variance components combine to give the expected values of mean squares in the analysis of variance table. In this example, calculation of variance components is not necessary for purposes of assessing the effects of treatments. We compare the `shade` mean square with the `residual` mean square for the `block.plt` level. The ratio is:

$$F\text{-ratio} = \frac{464.84}{20.93} = 22.2, \text{ on 3 and 6 d.f. } (p = 0.0024).$$

As this is a balanced design, the treatment estimates can be obtained using `model.tables()`:

```
> model.tables(kiwishade.aov, type="means")
Tables of means
Grand mean

96.53

 shade
shade
   none Aug2Dec Dec2Feb Feb2May
 100.20  103.23   89.92   92.77
```

The footnote gives an alternative way to calculate these means.[4]

[4] `## Calculate treatment means`
`with(kiwishade, sapply(split(yield, shade), mean))`

Table 10.4 *Plot means, and differences of yields for individual vines from the plot mean.*

block	shade	Mean	vine1	vine2	vine3	vine4
east	none	99.03	1.285	−2.025	−0.975	1.715
east	Aug2Dec	105.55	3.315	−2.415	2.765	−3.665
east	Dec2Feb	88.92	−3.520	−7.490	5.020	5.990
east	Feb2May	95.88	−3.430	1.690	1.270	0.470
north	none	104.03	−2.915	3.995	2.645	−3.725
north	Aug2Dec	103.59	−0.495	−1.075	0.425	1.145
north	Dec2Feb	93.70	−1.528	0.112	1.062	0.352
north	Feb2May	92.03	0.545	1.415	−1.745	−0.215
west	none	97.56	−4.918	5.932	3.123	−4.138
west	Aug2Dec	100.55	5.220	0.860	−2.950	−3.130
west	Dec2Feb	87.15	−0.495	3.605	−0.085	−3.025
west	Feb2May	90.41	−3.138	4.582	−1.347	−0.097

Grand mean = 96.53

Treatment differences are estimated within blocks, so that σ_B^2 does not contribute to the standard error of the differences (SED) between means. The SED is, accordingly, $\sqrt{2} \times$ the square root of the residual mean square divided by the sample size: $\sqrt{2} \times \sqrt{(20.93/12)} = 1.87$. The sample size is 12, since each treatment comparison is based on differences between two different sets of 12 vines.

The next subsection will demonstrate calculation of the sums of squares in the analysis of variance table, based on a breakdown of the observed yields into components that closely reflect the different terms in the model. Readers who do not at this point wish to study Subsection 10.3.3 in detail may nevertheless find it helpful to examine Figures 10.5, taking on trust the scalings used for the effects that they present.

10.3.3 The analysis of variance sums of squares breakdown*

This subsection shows how to calculate the sums of squares and mean squares. These details are not essential to what follows, but do give useful insight. The breakdown extends the approach used in the simpler example of Subsection 10.1.1.

For each plot, we calculate a mean, and differences from the mean. See Table 10.4.[5] Note that whereas we started with 48 observations we have only 12 means. Now we break the means down into overall mean, plus block effect (the average of differences, for that block, from the overall mean), plus treatment effect (the average of the difference, for that treatment, from the overall mean), plus residual.

[5]
```
## For each plot, calculate mean, and differences from the mean
vine <- paste("vine", rep(1:4, 12), sep="")
vine1rows <- seq(from=1, to=45, by=4)
kiwivines <- unstack(kiwishade, yield ~ vine)
kiwimeans <- apply(kiwivines, 1, mean)
kiwivines <- cbind(kiwishade[vine1rows,  c("block","shade")],
                   Mean=kiwimeans, kiwivines-kiwimeans)
kiwivines <- with(kiwivines, kiwivines[order(block, shade), ])
mean(kiwimeans)      # the grand mean
```

Table 10.5 *Each plot mean is expressed as the sum of overall mean (= 96.53), block effect, shade effect, and residual for the* `block:shade` *combination (or* `plt`*).*

`block`	`shade`	Mean	Block effect	Shade effect	`block:shade` residual
east	none	99.02	0.812	3.670	−1.990
east	Aug2Dec	105.56	0.812	6.701	1.509
east	Dec2Feb	88.92	0.812	−6.612	−1.813
east	Feb2May	95.88	0.812	−3.759	2.294
north	none	104.02	1.805	3.670	2.017
north	Aug2Dec	103.60	1.805	6.701	−1.444
north	Dec2Feb	93.70	1.805	−6.612	1.971
north	Feb2May	92.04	1.805	−3.759	−2.545
west	none	97.56	−2.618	3.670	−0.027
west	Aug2Dec	100.55	−2.618	6.701	−0.066
west	Dec2Feb	87.15	−2.618	−6.612	−0.158
west	Feb2May	90.41	−2.618	−3.759	0.251
			square, add, multiply by 4, divide by d.f. = 2, to give ms	square, add, multiply by 4, divide by d.f. = 3, to give ms	square, add, multiply by 4, divide by d.f. = 6, to give ms

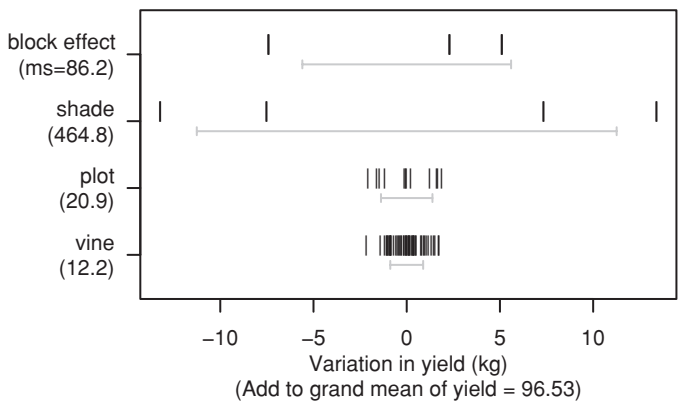


Figure 10.5 Variation at the different levels, for the kiwifruit shading data. The individual data values are given, together with one standard deviation limits either side of zero.

Table 10.5 uses the information from Table 10.4 to express each plot mean as the sum of a block effect, a shade effect, and a residual for the `block:shade` combination. The notes in the last row of each column show how to determine the mean squares that were given in Table 10.3. Moreover, we can scale the values in the various columns so that their standard deviation is the square root of the error mean square, and plot them. Figure 10.5 plots all this information. It shows the individual values, together with one standard deviation limits either side of zero. The chief purpose of these plots is to show the much greater variation at these levels than at the `plt` and vine level.

The estimate of between-plot variance in Table 10.3 was 20.93. While larger than the between-vine mean square of 12.18, it is not so much larger that the evidence from Figure 10.5 can be more than suggestive. Variation between treatments does appear much greater than can be explained from variation between plots, and the same is true for variation between blocks.

We now explain the scaling of effects in Figure 10.5. Consider first the 48 residuals at the vine level. Because 12 degrees of freedom were expended when the 12 plot means were subtracted, the 48 residuals share 36 degrees of freedom and are positively correlated. To enable the residuals to present the appearance of uncorrelated values with the correct variance, we scale the 48 residuals so that the average of their squares is the between-vine estimate of variance σ_V^2; this requires multiplication of each residual by $\sqrt{(48/36)}$.

At the level of plot means, we have 6 degrees of freedom to share between 12 plot effects. In addition, we need to undo the increase in precision that results from taking means of four values. Thus, we multiply each plot effect by $\sqrt{(12/6)} \times \sqrt{4}$. If the between-plot component of variance is zero, the expected value of the average of the square of these scaled effects will be σ_V^2. If the between-plot component of variance is greater than zero, the expected value of the average of these squared effects will be greater than σ_V^2.

In moving from plot effects to treatment effects, we have a factor of $\sqrt{(4/3)}$ that arises from the sharing of 3 degrees of freedom between 4 effects, further multiplied by $\sqrt{12}$ because each treatment mean is an average of 12 vines. For block effects, we have a multiplier of $\sqrt{(3/2)}$ that arises from the sharing of 2 degrees of freedom between 3 effects, further multiplied by $\sqrt{16}$ because each block mean is an average of 16 vines. Effects that are scaled in this way allow visual comparison, as in Figure 10.5.

10.3.4 The variance components

The mean squares in an analysis of variance table for data from a balanced multi-level model can be broken down further, into variance components. The variance components analysis gives more detail about model parameters. Importantly, it provides information that will help design another experiment. Here is the breakdown for the kiwifruit shading data:

- Variation between vines in a plot is made up of one source of variation only. Denote this variance by σ_V^2.
- Variation between vines in different plots is partly a result of variation between vines, and partly a result of additional variation between plots. In fact, if σ_P^2 is the (additional) component of the variation that is due to variation between plots, the expected mean square equals

$$4\sigma_P^2 + \sigma_V^2.$$

  (NB: the 4 comes from 4 vines per plot.)
- Variation between treatments is

$$4\sigma_P^2 + \sigma_V^2 + T$$

  where T (> 0) is due to variation between treatments.

- Variation between vines in different blocks is partly a result of variation between vines, partly a result of additional variation between plots, and partly a result of additional variation between blocks. If σ_B^2 is the (additional) component of variation that is due to differences between blocks, the expected value of the mean square is

$$16\sigma_B^2 + 4\sigma_P^2 + \sigma_V^2$$

(16 vines per block, 4 vines per plot).

We do not need estimates of the variance components in order to do the analysis of variance. The variance components are helpful for designing another experiment. We calculate the estimates thus:

$$\widehat{\sigma}_V^2 = 12.18,$$
$$4\widehat{\sigma}_P^2 + \widehat{\sigma}_V^2 = 20.93, \text{ i.e., } 4\widehat{\sigma}_P^2 + 12.18 = 20.93.$$

This gives the estimate $\widehat{\sigma}_P^2 = 2.19$. We can also estimate $\widehat{\sigma}_B^2 = 4.08$.

We are now in a position to work out how much the precision would change if we had 8 (or, say, 10) vines per plot. With n vines per plot, the variance of the plot mean is

$$(n\widehat{\sigma}_P^2 + \widehat{\sigma}_V^2)/n = \widehat{\sigma}_P^2 + \widehat{\sigma}_V^2/n = 2.19 + 12.18/n.$$

We could also ask how much of the variance, for an individual vine, is explained by vine-to-vine differences. This depends on how much we stretch the other sources of variation. If the comparison is with vines that may be in different plots, the proportion is $12.18/(12.18 + 2.19)$. If we are talking about different blocks, the proportion is $12.18/(12.18 + 2.19 + 4.08)$.

10.3.5 The mixed model analysis

For a mixed model analysis, we specify that treatment (`shade`) is a fixed effect, that `block` and `plot` are random effects, and that `plot` is nested in `block`. The software works out for itself that the remaining part of the variation is associated with differences between vines.

For using `lmer()`, the command is:

```
kiwishade.lmer <- lmer(yield ~ shade + (1|block) + (1|block:plot),
                       data=kiwishade)
  # block:shade is an alternative to block:plot
```

For comparison purposes with the results from the preceding section, consider:

```
> kiwishade.lmer
. . . .
Random effects:
 Groups     Name        Variance Std.Dev.
 block:plot (Intercept)  2.19    1.48
 block      (Intercept)  4.08    2.02
 Residual                12.18    3.49
# of obs: 48, groups: block:plot, 12; block, 3
```

These agree with the estimates that were obtained above.

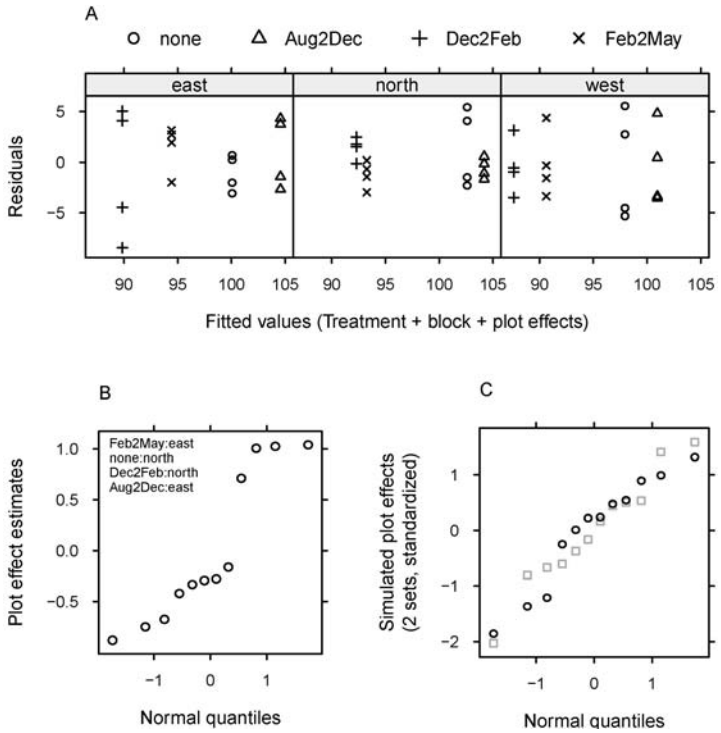


Figure 10.6 Panel A shows standardized residuals after fitting block and plot effects, plotted against fitted values. There are 12 distinct fitted values, one for each plot. Panel B is a normal probability plot that shows the plot effects. The names in the top left-hand corner identify the plots with the largest residuals. Panel C shows overlaid normal probability plots of plot effects from two simulations.

Residuals and estimated effects

In this hierarchical model there are three levels of variation: level 1 is between blocks, level 2 is between plots, and level 3 is between vines. The function `fitted()` adjusts for all levels of random variation except between individual vines, i.e., fitted values are at level 2. Unfortunately, `lmer()`, which was designed for use with crossed as well as hierarchical designs, does not recognize the notion of levels. The function `ranef()` can however be used to extract the relevant random effect estimates.

Figure 10.6A plots residuals after accounting for plot and block effects.[6] Figure 10.6B is a normal probability plot that shows the plot effects. The locations of the four plots that suggest departure from normality are printed in the top left of the panel.[7] The plot

[6]
```
## Simplified version of plot
xyplot(residuals(kiwishade.lmer) ~ fitted(kiwishade.lmer)|block, data=kiwishade,
                groups=shade, layout=c(3,1), par.strip.text=list(cex=1.0),
                xlab="Fitted values (Treatment + block + plot effects)",
                ylab="Residuals", pch=1:4, grid=TRUE,
                scales=list(x=list(alternating=FALSE), tck=0.5),
                key=list(space="top", points=list(pch=1:4),
                         text=list(labels=levels(kiwishade$shade)),columns=4))
```

[7]
```
## Simplified version of graph that shows the plot effects
ploteff <- ranef(kiwishade.lmer, drop=TRUE)[[1]]
qqmath(ploteff, xlab="Normal quantiles", ylab="Plot effect estimates",
       scales=list(tck=0.5))
```

effects are however estimates from a calculation that involves the estimation of a number of parameters. Before deciding that normality assumptions are in doubt, it is necessary to examine normal probability plots from data that have been simulated according to the normality and other model assumptions. Figure 10.6C shows overlaid normal probability plots from two such simulations. As the present interest is in the normality of the effects, not in variation in standard deviation (this would lead, in Figure 10.6C, to wide variation in aspect ratio), the effects are in each case standardized.[8] It is the plot effects that are immediately relevant to assessing the validity of assumptions that underly statistical comparisons between treatment means, not the residuals. The plot effect estimates seem clearly inconsistent with the assumption of normal plot effects. Remember however that each treatment mean is an averaging over three plots. This averaging will take the sampling distribution of the treatment means closer to normality.

It may be relevant to Figure 10.6B to note that the treatment means are, in order,

```
Dec2Feb Feb2May  none    Aug2Dec
89.92   92.77    100.20  103.23
```

Notice that the plot-specific effects go in opposite directions, relative to the overall treatment means, in the `east` and `north` blocks.

10.3.6 Predictive accuracy

We have data for one location on one site only. We thus cannot estimate a between-location component of variance for different locations on the current site, let alone a between-site component of variance. Use of resampling methods will not help; the limitation is inherent in the experimental design.

Where data are available from multiple sites, the site-to-site component of variance will almost inevitably be greater than zero. Given adequate data, the estimate of this component of variance will then also be greater than zero, even in the presence of explanatory variable adjustments that attempt to adjust for differences in rainfall, temperature, soil type, etc. (Treatment differences are often, but by no means inevitably, more nearly consistent across sites than are the treatment means themselves.)

Where two (or more) experimenters use different sites, differences in results are to be expected. Such different results have sometimes led to acriminious exchanges, with each convinced that there are flaws in the other's experimental work. Rather, assuming that both experiments were conducted with proper care, the implication is that both sets of results should be incorporated into an analysis that accounts for site-to-site variation. Better still, plan the experiment from the beginning as a multi-site experiment!

[8]
```
## Overlaid normal probability plots of 2 sets of simulated effects
## To do more simulations, change nsim as required, and re-execute
simvals <- simulate(kiwishade.lmer, nsim=2)
simeff <- apply(simvals, 2, function(y) scale(ranef(refit(kiwishade.lmer, y),
                                                    drop=TRUE)[[1]]))
simeff <- data.frame(v1=simeff[,1], v2=simeff[,2])
qqmath(~ v1+v2, data=simeff, xlab="Normal quantiles",
       ylab="Simulated plot effects\n(2 sets, standardized)",
              scales=list(tck=0.5), aspect=1)
```

A: Plot symbols and text; specify colors and/or character expansion; draw rectangle

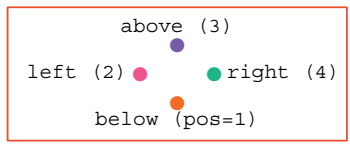
```
par(fig=c(0, 1, 0.415, 1))
plot(0, 0, xlim=c(0, 13), ylim=c(0, 19), type="n")
xpos <- rep((0:12)+0.5, 2);  ypos <- rep(c(14.5,12.75), c(13,13))
points(xpos, ypos, cex=2.5, col=1:26, pch=0:25)
text(xpos, ypos, labels=paste(0:25), cex=0.75)
```




```
## Plot characters, vary cex (expansion)
text((0:4)+0.5, rep(9*ht, 5), letters[1:5], cex=c(2.5,2,1,1.5,2))
```

a b c d e

```
## Position label with respect to point
xmid <- 10.5; xoff <- c(0, -0.5, 0, 0.5)
ymid <- 5.8; yoff <- c(-1,0,1,0)
col4 <- colors()[c(52, 116, 547, 610)]
points(xmid+xoff, ymid+yoff, pch=16, cex=1.5, col=col4)
posText <- c("below (pos=1)", "left (2)", "above (3)", "right (4)")
text(xmid+xoff, ymid+yoff, posText, pos=1:4)
rect(xmid-2.3, ymid-2.3, xmid+2.3, ymid+2.3, border="red")
```



B: Triangles or polygons, circles, and mathematical text

```
par(fig=c(0, 1, 0.01, 0.40), new=TRUE)
plot(0, 0, xlim=c(0, 13), ylim=c(0, 12), type="n")
polygon(x=c(10.7,12.8,12), y=c(7.5,8,11), col="gray", border="red")

## Draw a circle, overlay 2-headed arrow (code=3)
xcenter <- 11.7; ycenter <- 4; r=1.1
symbols(x=xcenter, y=ycenter, circles=r,
        bg="gray", add=TRUE, inches=FALSE)
arrows(x0=xcenter-r, y0=ycenter, x1=xcenter, y1=ycenter,
       length=.05, code=3)

## Use expression() to add labeling information
charht <- strheight("R")
text(x=xcenter-r/2, y=ycenter-charht, expression(italic(r)))
text(xcenter, ycenter+3.5*charht, expression("Area" == pi*italic(r)^2))
```

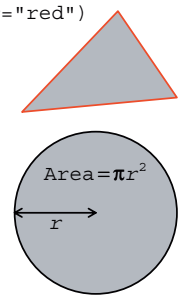


Plate 1 This figure, intended to accompany Section 1.5, demonstrates the use of parameter settings to control various graphical features. (Note that the function `paste()` turns the vector of numerical values `0:12` into a vector of character strings with elements `"0"`, `"1"`, `...`, `"12"`. An alternative to `paste(0:12)` is `as.character(0:12)`.) See also Section 15.2.

.

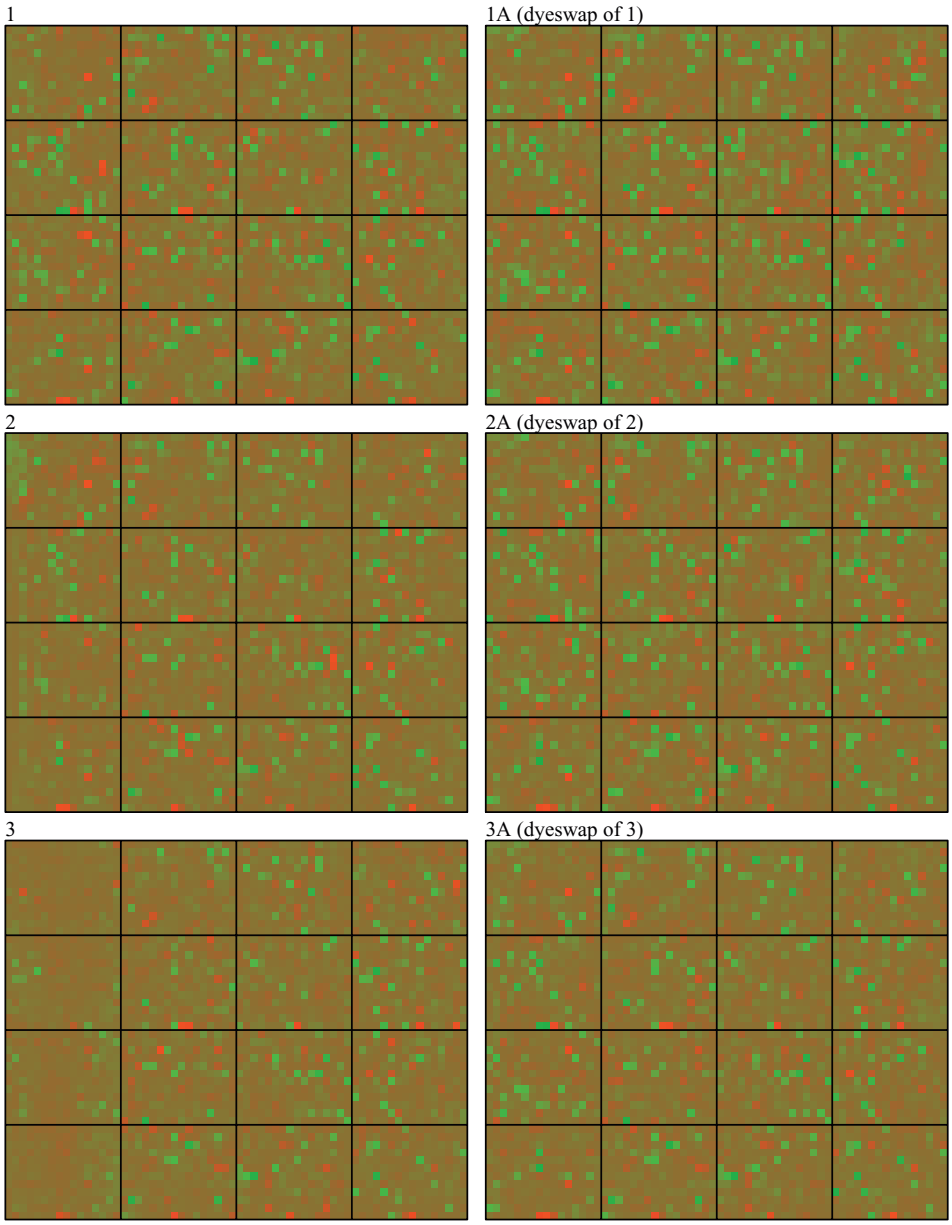


Plate 2 This false color image shows the intensity of the post-signal (red), relative to the pre-signal (green), for each of six half-slides in a two-channel microarray gene-expression experiment. (See Subsection 4.4.1.)

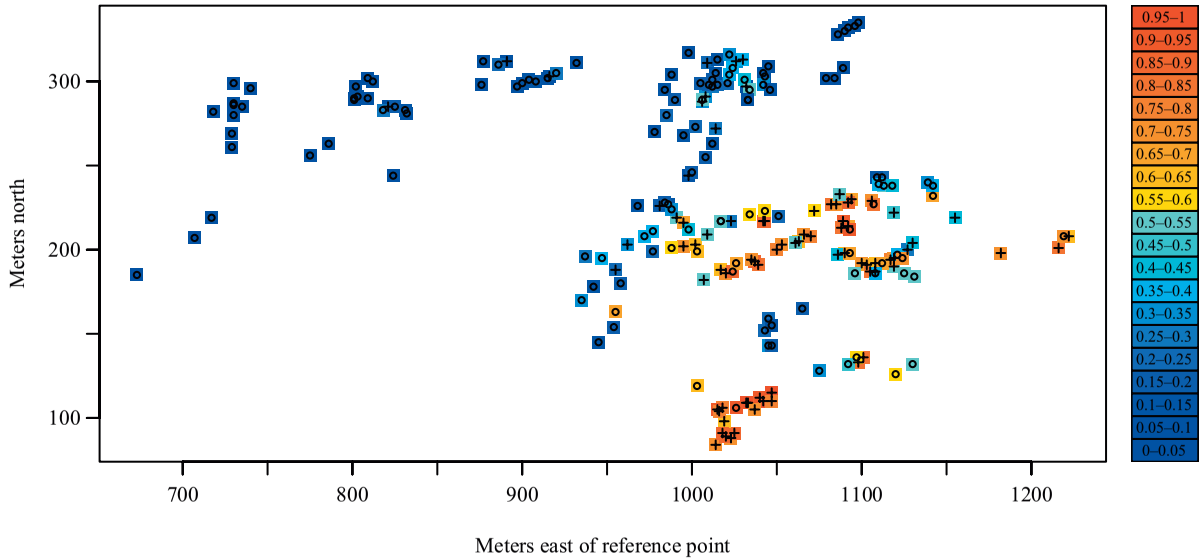


Plate 3 Fitted values (model predictions of the probability of finding a frog) are shown on the `blue2red` color density scale from the *colorRamps* package. Sites are labeled "○" or "+" according to whether frogs were not found or were found. For details of the model, see Subsection 8.2.1.

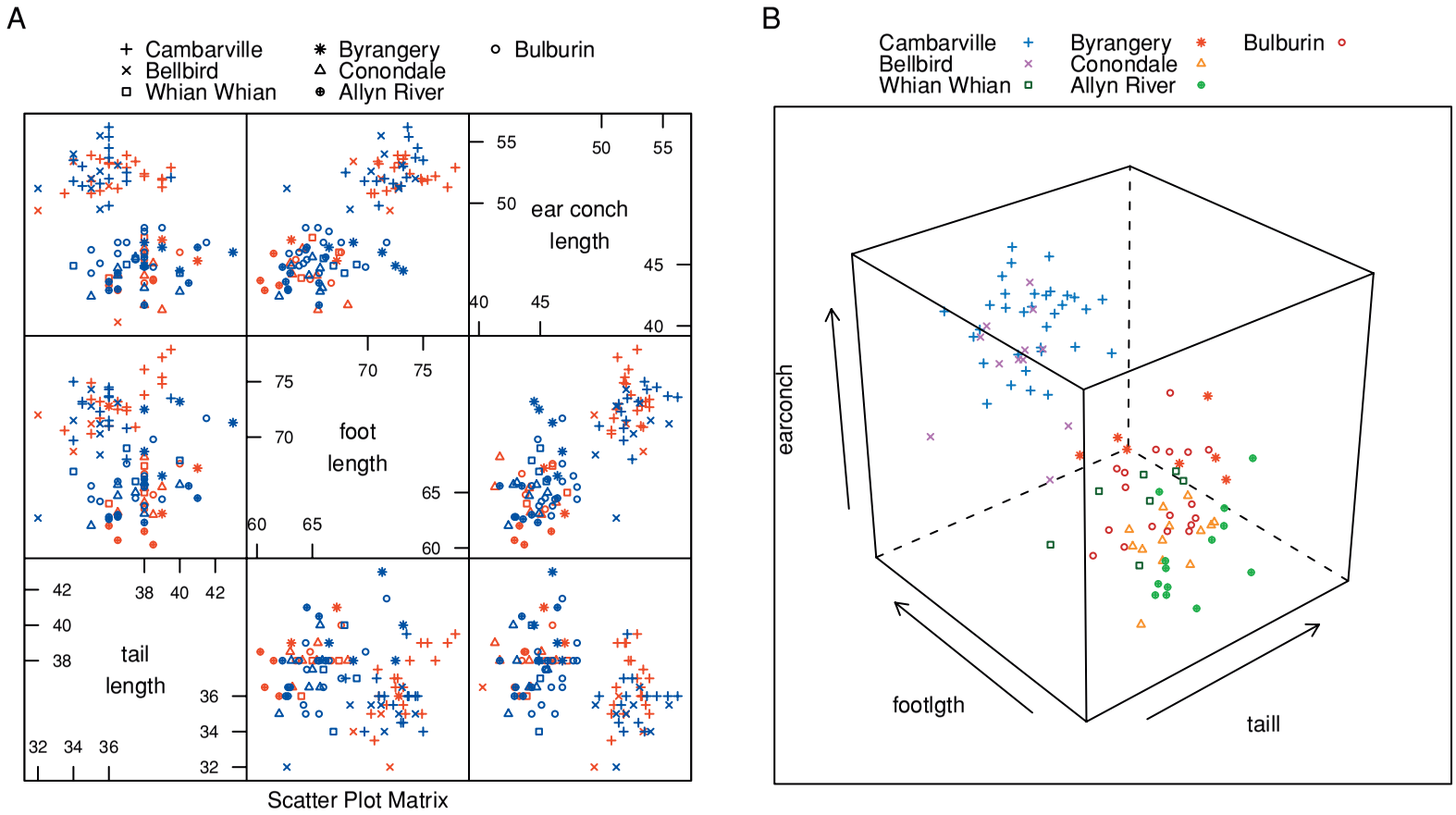


Plate 4 Panel A shows the scatterplot matrix for three morphometric measurements on the mountain brushtail possum. Females are in red; males in blue. Panel B shows a three-dimensional perspective plot (cloud plot) for the same three variables. (See Figure 12.1.)

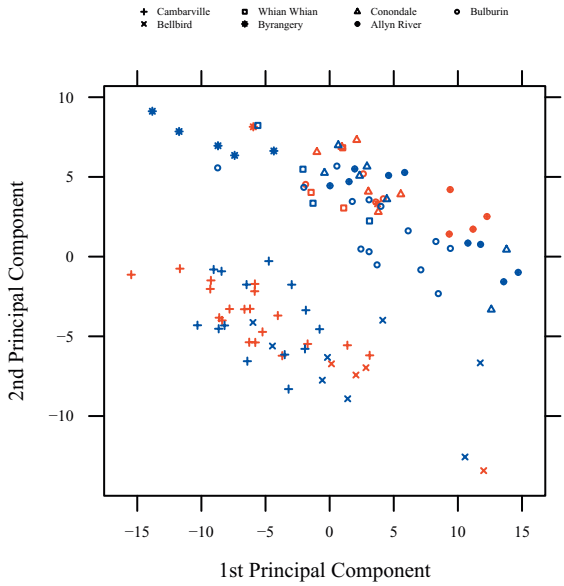


Plate 5 Second principal component versus first principal component, for variables in columns 6–14 of the `possum` data frame. Females are in red; males in blue. (See Figure 12.2.)

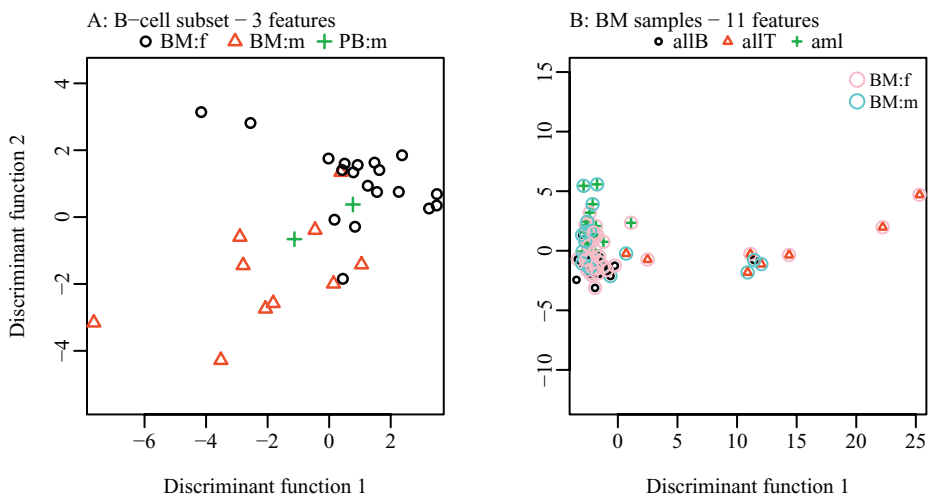


Plate 6 These plots of projections of linear discriminant scores are designed to fairly reflect the performance of a linear discriminant in distinguishing between known groups in the data. The two panels relate to different subsets of the `Golub` data, with different groupings in the two cases. In panel B, for the classification of the 62 bone marrow (`BM`) samples into `allB`, `allT`, and `aml`, points where the sex is known are identified as male or female. (See Figure 12.10.)

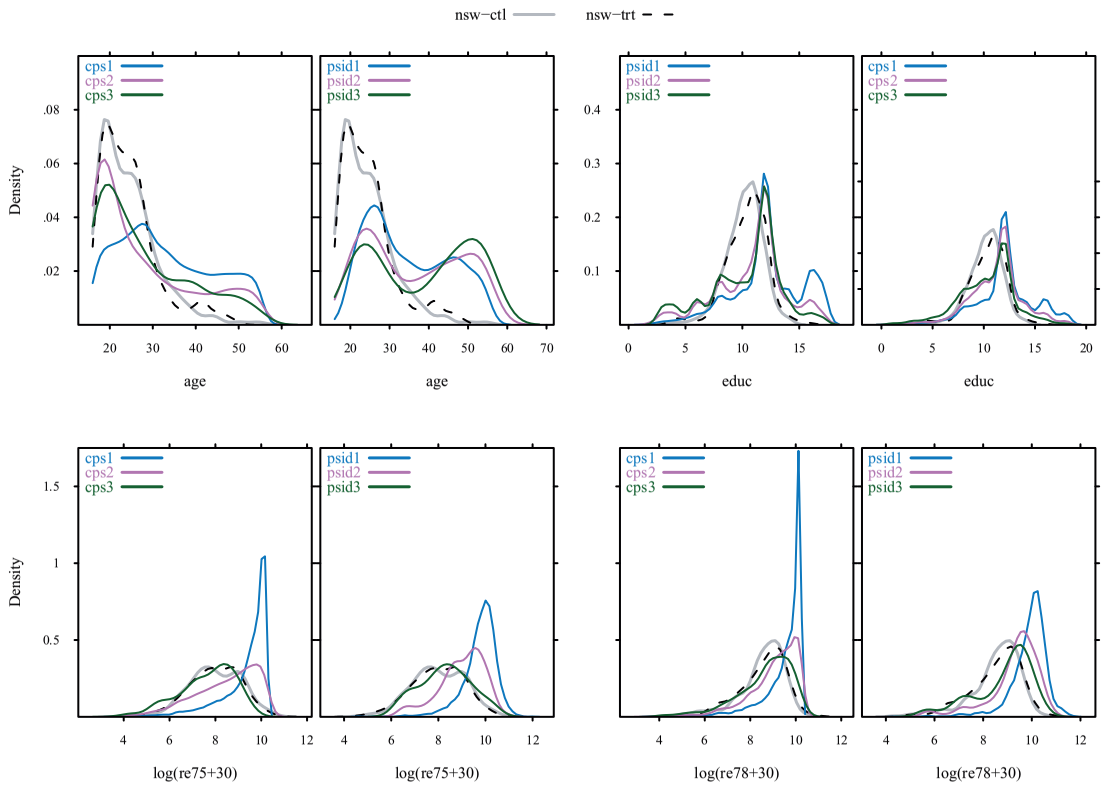


Plate 7 Overlaid density plots, comparing treatment groups with the various alternative choices of control groups, for the variables `age`, `educ`, `log(re75+100)`, and `log(re78+100)`. (See Figure 13.2.)

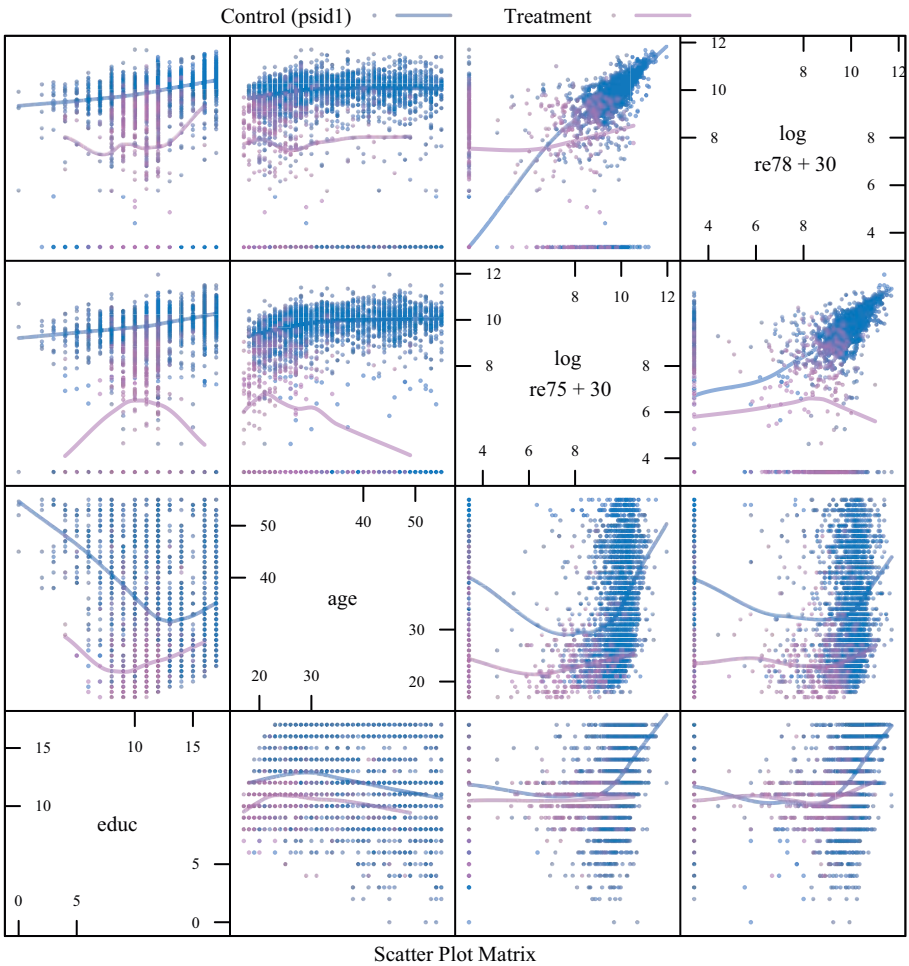


Plate 8 Scatterplot matrix, for non-binary explanatory variables, for the data set that is formed by combining the `psid1` data (non-experimental) with the observations for experimental treatment data in `nswdemo`. For further details, see Section 13.2. Thus, an experimental treatment group is compared with a non-experimental control group. Separate smooth curves have been fitted for the control and treatment groups.

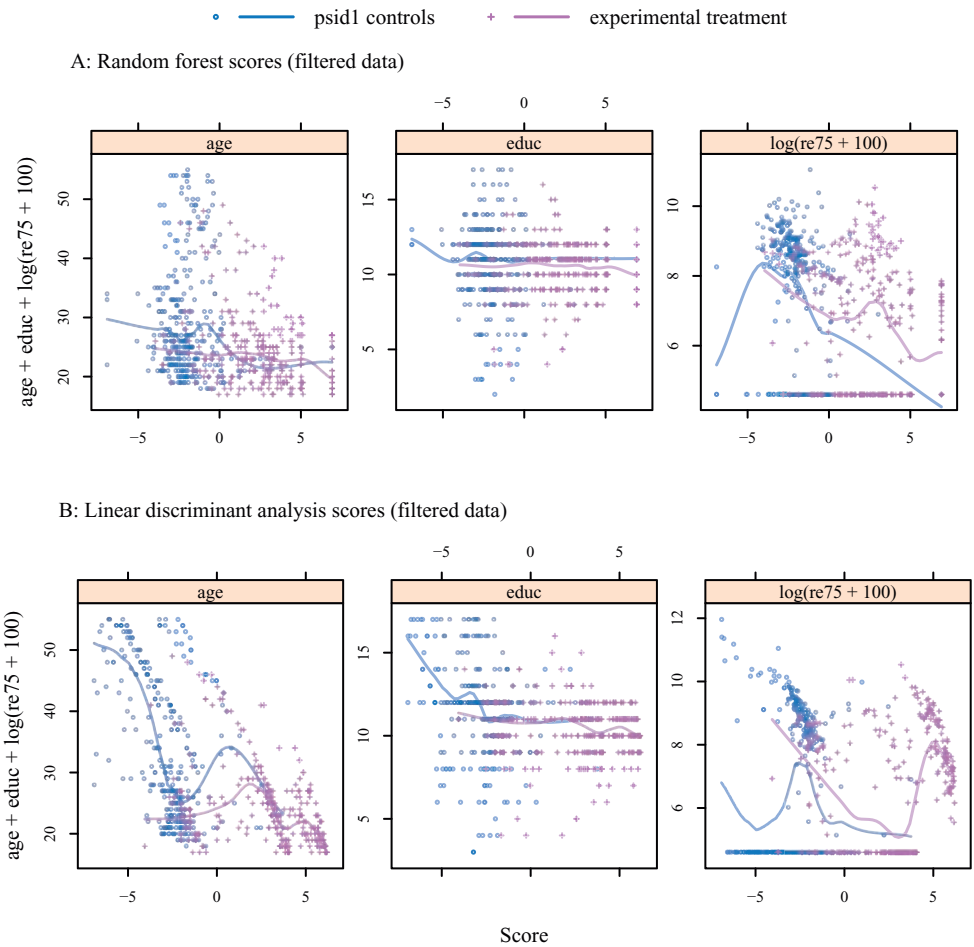


Plate 9 Panel A shows the distribution, for control and treatment data, of `randomForest` scores obtained by refitting the model to data for which the scores, obtained from the total data, were at least −1.5. Panel B is for the `lda` scores, after refitting the model to data for which the scores, obtained from the total data, were at least −4. (See Figure 13.4.)

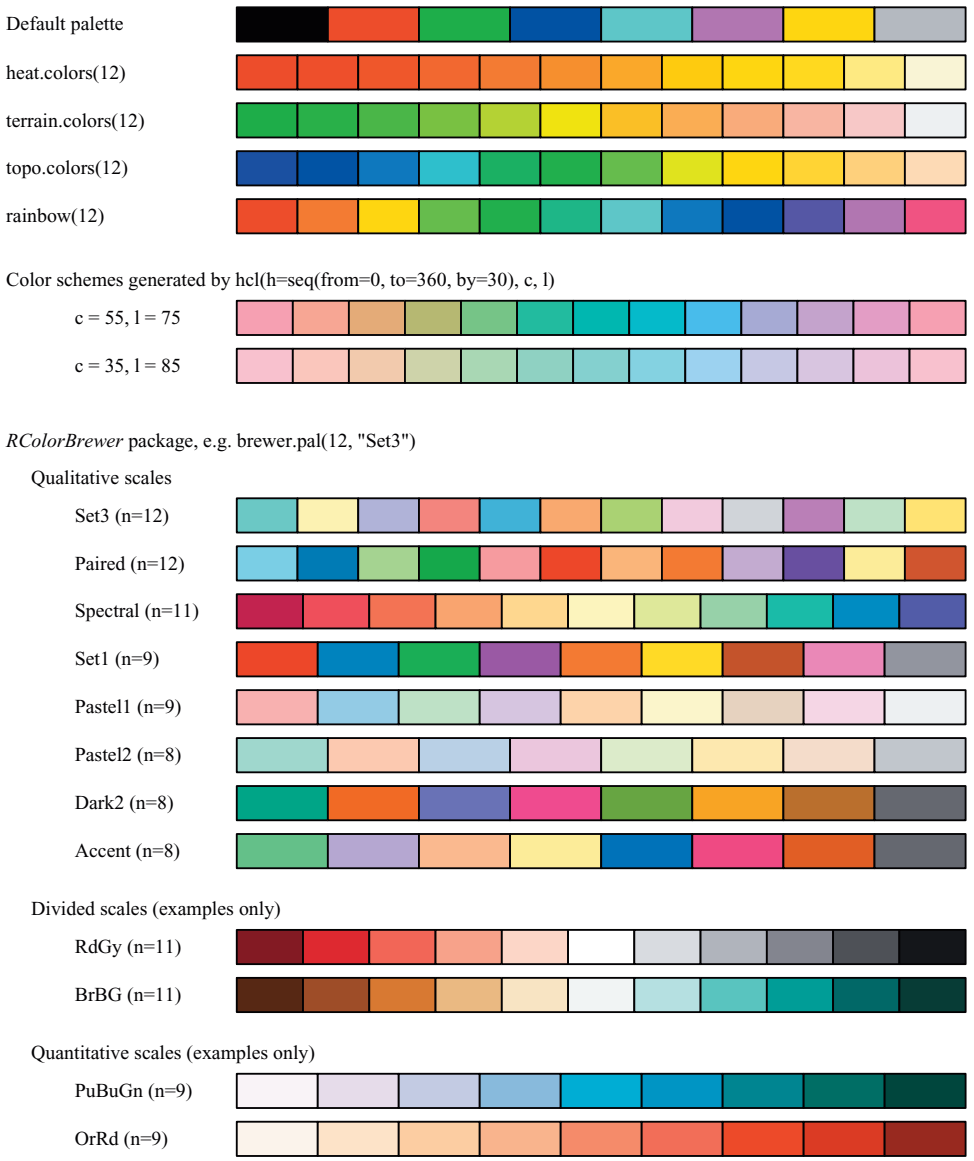
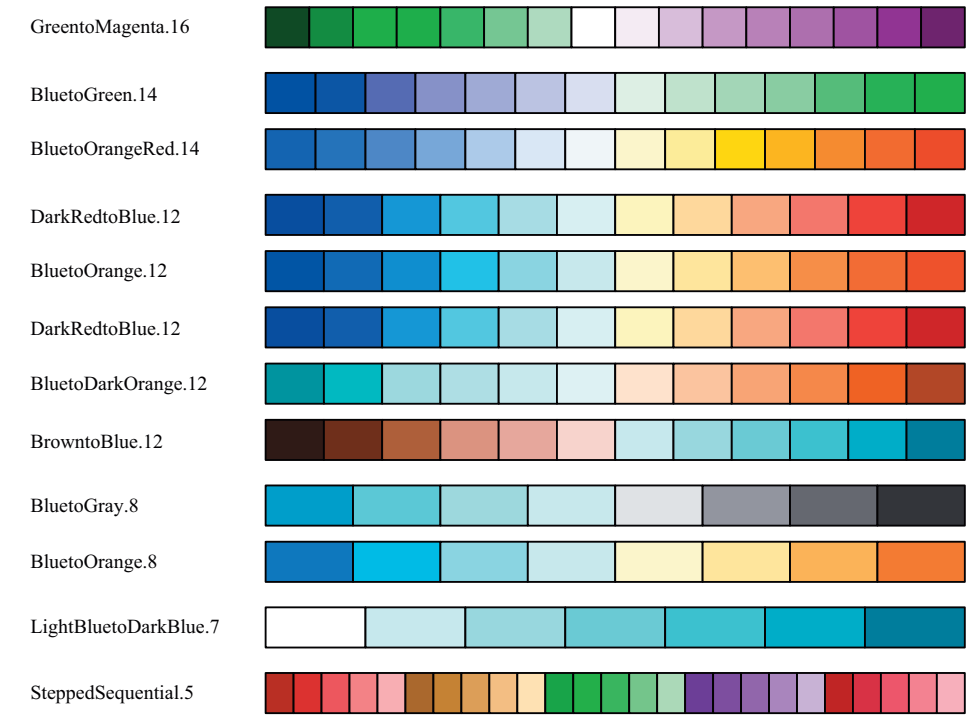


Plate 10 The default palette has the eight colors that are shown. Functions in the default package *grDevices* that can be used to generate color sequences include `heat.colors()`, `terrain.colors()`, `topo.colors()`, `rainbow()`, and `hcl()`. Also shown are a selection of palettes from the *RColorBrewer* package. See Chapter 15 for further details.

Selected schemes from the *dichromat* package, e.g. colorshemes$GreentoMagenta.16



Simulation of Effects of Two Common types of Red–Green Color Blindness

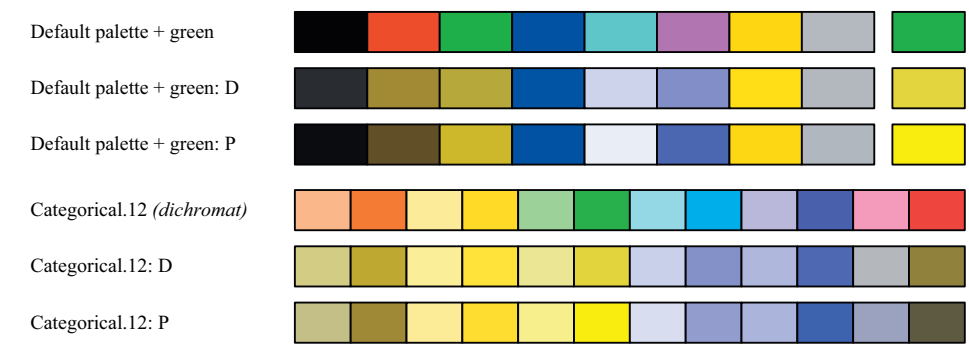
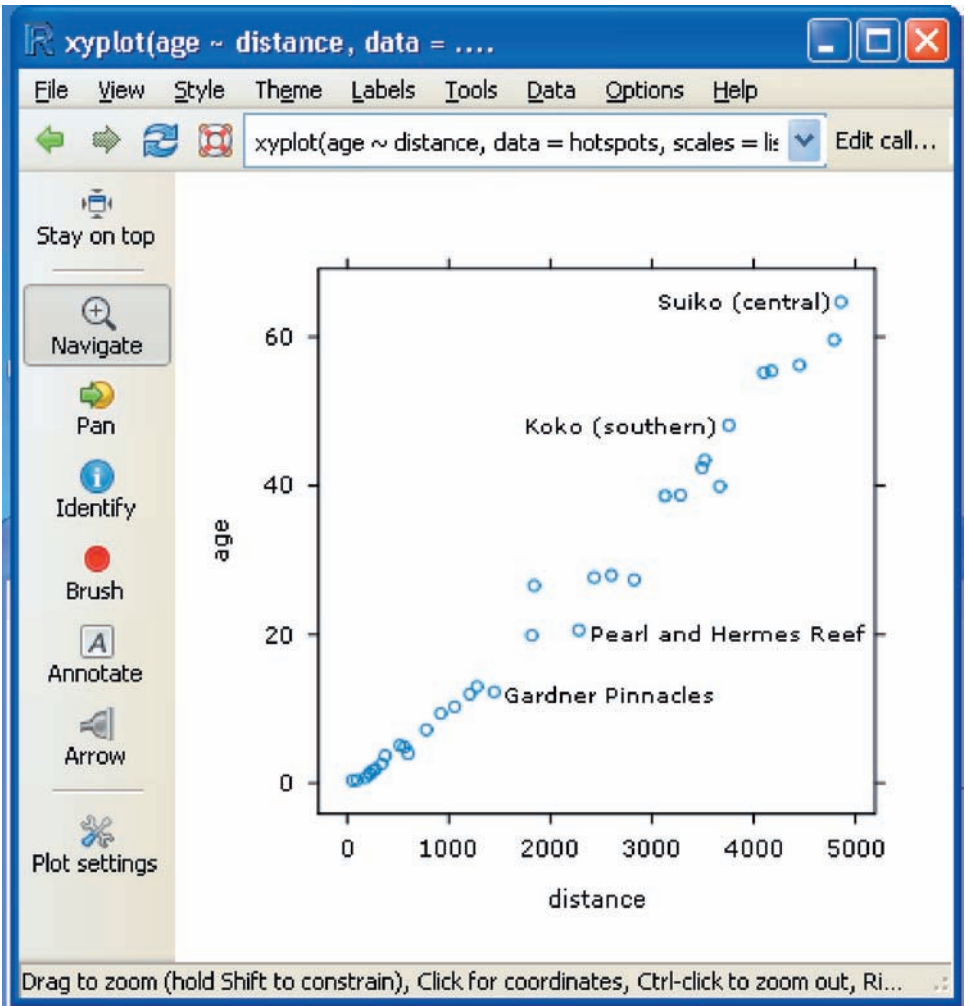


Plate 11 Palettes in the *dichromat* package use colors that can be distinguished by individuals with either of two common forms of red–green color blindness. Also shown are simulations of the effects of these two common forms of red–green color blindness, first for the default palette plus `"green"`, and then for the `Categorical.12` palette from *dichromat*. Simulations for deuteranomia are identified with a D, while simulations for proteranomia are identified with a P. See Chapter 15 for further comment.




```
> playwith(xyplot(age ~ distance, data=hotspots),
+          labels=hotspots$name)
```

Plate 12 This playwith GUI window was generated by wrapping the call to `xyplot()` in the function `playwith()`, then clicking on Identify. Click near to a point to see its label. A second click adds the label to the graph.

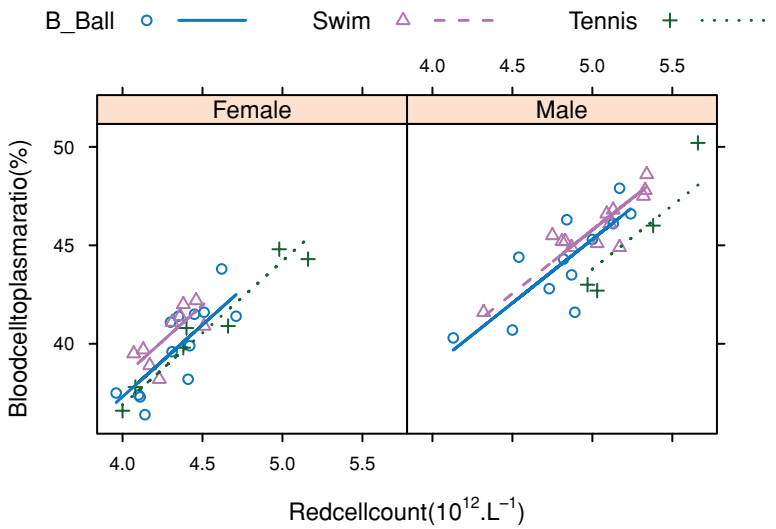


Plate 13 Blood cell to plasma ratio (`hc`) versus red cell count (`rcc`), by `sex` (different panels) and `sport` (distinguished within each panel). A panel function was supplied, as described in Subsection 15.5.5, that fits and draws parallel lines.

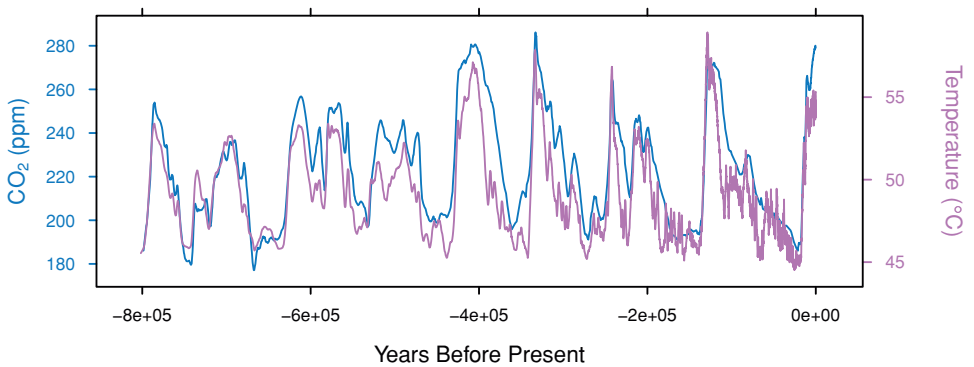


Plate 14 This graph overlays EPICA Dome C ice core 800KYr temperature and CO_2 estimates, showing the strong correlation. Both sets of data have been locally smoothed.

10.4 Within- and between-subject effects

The data frame `tinting` is from an experiment that aimed to model the effects of the tinting of car windows on visual performance. (For more information, see Burns *et al.*, 1999.) Interest is focused on effects on side window vision, and hence on visual recognition tasks that would be performed when looking through side windows.

The variables are `csoa` (critical stimulus onset asynchrony, i.e., the time in milliseconds required to recognize an alphanumeric target), `it` (inspection time, i.e., the time required for a simple discrimination task) and `age`, while `tint` (three levels) and `target` (two levels) are ordered factors. The variable `sex` is coded `f` for females and `m` for males, while the variable `agegp` is coded `Younger` for young people (all in their 20s) and `Older` for older participants (all in their 70s).

Data were collected in two sessions, with half the individuals undertaking the `csoa` task in the first session and the `it` task in the second session, and the other half doing these two types of task in the reverse order. Within each session, the order of presentation of the two levels of target contrast was balanced over participants. For each level of target contrast the levels of `tint` were in the order `no` (100% VLT = visible light transmittance), `lo` (81.3% VLT = visible light transmittance), and `hi` (35% VLT = visible light transmittance). Each participant repeated the combination of high contrast with no tinting (100% VLT) at the end of the session. Comparison with the same task from earlier in the session thus allows a check on any change in performance through the session.

We have two levels of variation – within individuals (who were each tested on each combination of `tint` and `target`), and between individuals. Thus we need to specify `id` (identifying the individual) as a random effect. Plots of the data make it clear that, to have variances that are approximately homogeneous, we need to work with `log(csoa)` and `log(it)`. Here, we describe the analysis for `log(it)`.

Model fitting criteria

The function `lmer()` allows use of one of two criteria: restricted (or residual) maximum likelihood (REML), which is the default, and maximum likelihood (ML). The parameter estimates from the REML method are generally preferred to those from ML, as more nearly unbiased. Comparison of models using `anova()` relies on maximum likelihood theory, and the models should be fitted using ML.

10.4.1 Model selection

A good principle is to limit initial comparisons between models to several alternative models within a hierarchy of increasing complexity. For example, consider main effects only, main effects plus all first-order interactions, main effects plus all first- and second-order interactions, as far on up this hierarchy as seems reasonable. This makes for conceptual clarity, and simplifies inference. (Where a model has been selected from a large number of candidate models, extreme value effects may come into play, and inference must account for this.)

Here, three models will be considered:

1. All possible interactions (this is likely to be more complex than is needed):
```
## Change initial letters of levels of tinting$agegp to upper case
levels(tinting$agegp) <- toupper.initial(levels(tinting$agegp))
## Fit all interactions: data frame tinting (DAAG)
it3.lmer <- lmer(log(it) ~ tint*target*agegp*sex + (1 | id),
                 data=tinting, method="ML")
```
2. All two-factor interactions (this is a reasonable guess; two-factor interactions may be all we need):
```
it2.lmer <- lmer(log(it) ~ (tint+target+agegp+sex)^2 + (1 | id),
                  data=tinting, method="ML")
```
3. Main effects only (this is a very simple model):
```
it1.lmer <- lmer(log(it)~(tint+target+agegp+sex) + (1 | id),
                  data=tinting, method= "ML")
```

Note the use of `method="ML"`; this then allows the equivalent of an analysis of variance comparison:

```
> anova(it1.lmer, it2.lmer, it3.lmer)
Data: tinting
Models:
it1.lmer: log(it) ~ (tint + target + agegp + sex) + (1 | id)
it2.lmer: log(it) ~ (tint + target + agegp + sex)^2 + (1 | id)
it3.lmer: log(it) ~ tint * target * agegp * sex + (1 | id)
          Df  AIC  BIC logLik Chisq Chi Df Pr(>Chisq)
it1.lmer   7 -0.9 21.6    7.4
it2.lmer  16 -5.7 45.5   18.9 22.88      9     0.0065
it3.lmer  25  6.1 86.2   21.9  6.11      9     0.7288
```

Notice that `Df` is now used for degrees of freedom, where `DF` was used in connection with `summary.aov()`. earlier.

The p-value for comparing model 1 with model 2 is 0.73, while that for comparing model 2 with model 3 is 0.0065. This suggests that the model that limits attention to two-factor interactions is adequate. (Note also that the AIC statistic favors model 2. The BIC statistic, which is an alternative to AIC, favors the model that has main effects only.)

Hastie *et al.* (2009, p. 235) suggest, albeit in reference to models with i.i.d. errors, that BIC's penalty for model complexity can be unduly severe when the number of residual degrees of freedom is small. (Note also that the different standard errors are based on variance component information at different levels of the design, so that the critique in Vaida and Blanchard (2005) perhaps makes the use of either of these statistics problematic. See Spiegelhalter *et al.* (2002) for various alternatives to AIC and BIC that may be better suited to use with models with "complex" error structures. Our advice is to use all such statistics with caution, and to consider carefully implications that may arise from the intended use of model results.)

The analysis of variance table indicated that main effects together with two-factor interactions were enough to explain the outcome. Interaction plots, looking at the effects of factors two at a time, are therefore an effective visual summary of the analysis results. In the table

of coefficients that appears below, the highest t-statistics for interaction terms are associated with `tint.L:agegpOlder`, `targethicon:agegpOlder`, `tint.L:targethicon`, and `tint.L:sexm`. It makes sense to look first at those plots where the interaction effects are clearest, i.e., where the t-statistics are largest. The plots may be based on either observed data or fitted values, at the analyst's discretion.[9]

10.4.2 Estimates of model parameters

For exploration of parameter estimates in the model that includes all two-factor interactions, we refit the model used for `it2.lmer`, but now using `method="REML"` (restricted maximum likelihood estimation), and examine the estimated effects. The parameter estimates that come from the `REML` analysis are in general preferable, because they avoid or reduce the biases of maximum likelihood estimates. (See, e.g., Diggle *et al.* (2002). The difference from likelihood can however be of little consequence.)

```
> it2.reml <- update(it2.lmer, method="REML")
> summary(it2.reml)
. . . .
Fixed effects:
                        Estimate Std. Error t value Pr(>|t|)   DF
(Intercept)              3.61907    0.13010   27.82  < 2e-16  145
tint.L                   0.16095    0.04424    3.64  0.00037  145
tint.Q                   0.02096    0.04522    0.46  0.64352  145
targethicon             -0.11807    0.04233   -2.79  0.00590  145
agegpolder               0.47121    0.23294    2.02  0.04469   22
sexm                     0.08213    0.23294    0.35  0.72486   22
tint.L:targethicon      -0.09193    0.04607   -2.00  0.04760  145
tint.Q:targethicon      -0.00722    0.04821   -0.15  0.88107  145
tint.L:agegpolder        0.13075    0.04919    2.66  0.00862  145
tint.Q:agegpolder        0.06972    0.05200    1.34  0.18179  145
tint.L:sexm             -0.09794    0.04919   -1.99  0.04810  145
tint.Q:sexm              0.00542    0.05200    0.10  0.91705  145
targethicon:agegpolder  -0.13887    0.05844   -2.38  0.01862  145
targethicon:sexm         0.07785    0.05844    1.33  0.18464  145
agegpolder:sexm          0.33164    0.32612    1.02  0.31066   22
. . . .
> # NB: The final column, giving degrees of freedom, is not in the
> # summary output for version 0.995-2 of lme4. It is our addition.
```

Because `tint` is an ordered factor with three levels, its effect is split up into two parts. The first, which always carries a `.L` (linear) label, checks if there is a linear change across levels. The second part is labeled `.Q` (quadratic), and as `tint` has only three levels, accounts for all the remaining sum of squares that is due to `tint`. A comparable partitioning of the effect of `tint` carries across to interaction terms also.

[9] `## Code that gives the first four such plots, for the observed data`
`interaction.plot(tinting$tint, tinting$agegp, log(tinting$it))`
`interaction.plot(tinting$target, tinting$sex, log(tinting$it))`
`interaction.plot(tinting$tint, tinting$target, log(tinting$it))`
`interaction.plot(tinting$tint, tinting$sex, log(tinting$it))`

The *t*-statistics are all substantially less than 2.0 in terms that include a `tint.Q` component, suggesting that we could simplify the output by restricting attention to `tint.L` and its interactions.

None of the main effects and interactions involving `agegp` and `sex` are significant at the conventional 5% level, though `agegp` comes close. This may seem inconsistent with Figures 2.12A and B, where it is the older males who seem to have the longer times. On the other hand, the interaction terms (`tint.L:agegpOlder`, `targethicon:agegpOlder`, `tint.L:targethicon`, and `tint.L:sexm`) that are statistically significant stand out much less clearly in Figures 2.12A and B.

To resolve this apparent inconsistency, consider the relative amounts of evidence for the two different sets of comparisons, and the consequences for the standard errors in the computer output.

- **Numbers of individuals**

```
> uid <- unique(tinting$id)
> subs <- match(uid, tinting$id)
> with(tinting, table(sex[subs], agegp[subs]))

   Younger Older
f 9       4
m 4       9
```

  Standard errors in the computer output given above, for comparisons made at the level of individuals and thus with 22 d.f., are in the range 0.23–0.32.
- **Numbers of comparisons between levels of `tint` or `target`**. Each of these comparisons is made at least as many times as there are individuals, i.e., at least 26 times. Standard errors in the computer output on p. 241, for comparisons made at this level, are in the range 0.042–0.058.

Statistical variation cannot be convincingly ruled out as the explanation for the effects that stand out most strongly in the graphs.

10.5 A generalized linear mixed model

Consider again the `moths` data of Subsection 8.4.2. The analysis in Subsection 8.4.2 assumed a quasi-Poisson error, which uses a constant multiplier for the Poisson variance. It may be better to assume a random between-transects error that is additive on the scale of the linear predictor. For this, a random term is associated with each transect. The code is:

```
> moths$transect <- 1:41  # Each row is from a different transect
> moths$habitat <- relevel(moths$habitat, ref="Lowerside")
> (A.glmer <-  glmer(A~habitat+log(meters)+(1|transect),
+                    family=poisson, data=moths))
Generalized linear mixed model fit by the Laplace approximation
Formula: A ~ habitat + log(meters) + (1 | transect)
   Data: moths
 AIC BIC logLik deviance
  95 112  -37.5       75
```

```
Random effects:
 Groups   Name        Variance Std.Dev.
 transect (Intercept) 0.234    0.483
Number of obs: 41, groups: transect, 41

Fixed effects:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)         1.0201     0.4020    2.54   0.0112
habitatBank       -16.9057  2225.4575   -0.01   0.9939
habitatDisturbed   -1.2625     0.4820   -2.62   0.0088
habitatNEsoak      -0.8431     0.4479   -1.88   0.0598
habitatNWsoak       1.5517     0.3956    3.92  8.8e-05
habitatSEsoak       0.0532     0.3549    0.15   0.8808
habitatSWsoak       0.2506     0.4593    0.55   0.5853
habitatUpperside   -0.1707     0.5433   -0.31   0.7534
log(meters)         0.1544     0.1393    1.11   0.2677
```

The variance that is due to the Poisson error is increased, on the scale of the linear predictor, by 0.234. More extreme estimates of treatment differences (but not for `Bank`) are pulled in towards the overall mean. The habitat `Disturbed` now appears clearly different from the reference, which is `Lowerside`.

The reason is that on the scale of the linear predictor, the Poisson variance is largest when the linear predictor is smallest, that is when the expected count is, as for `Disturbed`, close to zero. Addition of an amount that is constant across the range has a relatively smaller effect when the contribution from the Poisson variance is, on this scale, largest.

Residuals should be plotted, both against log(transect length) and against the logarithm of the predicted number of moths:

```
A.glm <- glm(A~habitat+log(meters), data=moths, family=quasipoisson)
fitglm <- fitted(A.glm)
fitglm[fitglm<1e-6] <- NA   # Bank, where no moths were found
fitglmer <- fitted(A.glmer)
fitglmer[fitglmer<1e-6] <- NA
## Plots from quasipoisson analysis
plot(resid(A.glm) ~ moths$meters, log="x")
plot(resid(A.glm) ~ fitglm, log="x")
##
## Plots from glmer mixed model analysis
plot(resid(A.glmer) ~ moths$meters, log="x")
plot(resid(A.glmer) ~ fitglmer, log="x")
```

The residuals do not give any obvious reason to prefer one analysis to the other.

A similar analysis can be obtained using the function `glmmPQL` in the *MASS* package.

Mixed models with a binomial error and logit link

On a logit scale, the binomial contribution to the error increases as the expected value moves away from 0.5. (On the scale of the response, however, error decreases as the expected value

moves away from 0.5.) Thus, relative to a quasi-binomial model, the SED will be reduced for more extreme comparisons, and increased for less extreme comparisons.

10.6 Repeated measures in time

Whenever we make repeated measurements on a treatment unit we are, technically, working with repeated measures. In this sense, both the kiwifruit shading data and the window tinting data sets were examples of repeated measures data sets. Here, our interest is in generalizing the multi-level modeling approach to handle models for longitudinal data, i.e., data where the measurements were repeated at different times. We comment on the principles involved.

In the kiwifruit shading experiment, we gathered data from all vines in a plot at the one time. In principle, we might have taken data from different vines at different points in time. For each plot, there would be data at each of four time points.

There is a close link between a wide class of repeated measures models and time series models. In the time series context, there is usually just one realization of the series, which may however be observed at a large number of time points. In the repeated measures context, there may be a large number of realizations of a series that is typically quite short.

Perhaps the simplest case is where there is no apparent trend with time. Thus consider data from a clinical trial of a drug (progabide) used to control epileptic fits. (For an analysis of data from the study to which this example refers, see Thall and Vail, 1990.) The analysis assumes that patients were randomly assigned to the two treatments – placebo and progabide. After an eight-week run-in period, data were collected, both for the placebo group and for the progabide group, in each of four successive two-week periods. The outcome variable was the number of epileptic fits over that time.

One way to do the analysis is to work with the total number of fits over the four weeks, perhaps adjusted by subtracting the baseline value. It is possible that we might obtain extra sensitivity by taking account of the correlation structure between the four sets of fortnightly results, taking a weighted sum rather than a mean.

Where there is a trend with time, working with a mean over all times will not usually make sense. Any or all of the following can occur, both for the overall pattern of change and for the pattern of difference between one profile and another.

1. There is no trend with time.
2. The pattern with time may follow a simple form, e.g., a line or a quadratic curve.
3. A general form of smooth curve, e.g., a curve fitted using splines, may be required to account for the pattern of change with time.

The theory of repeated measures modeling

For the moment, profiles (or subjects) are assumed independent. The analysis must allow for dependencies between the results of any one subject at different times. For a balanced design, we will assume n subjects ($i = 1, 2, \ldots, n$) and p times ($j = 1, 2, \ldots, p$), though perhaps with missing responses (gaps) for some subjects at some times. The plot of response versus time for any one subject is that subject's *profile*.

A key idea is that there are (at least) two levels of variability – between subjects and within subjects. In addition, there is measurement error.

Repeating the same measurement on the same subject at the same time will not give exactly the same result. The between-subjects component of variation is never observable separately from sources of variation that operate "within subjects". In any data that we collect, measurements are always affected by "within-subjects" variability, plus measurement error. Thus the simplest model that is commonly used has a between-subjects variance component denoted by ν^2, while there is a within-subjects variance at any individual time point that is denoted by σ^2. The measurement error may be bundled in as part of σ^2. The variance of the response for one subject at a particular time point is $\nu^2 + \sigma^2$.

In the special case just considered, the variance of the difference between two time points for one subject is $2\sigma^2$. Comparisons "within subjects" are more accurate than comparisons "between subjects".

*Correlation structure

The time dependence of the data has implications for the correlation structure. The simple model just described takes no account of this structure. Points that are close together in time are in some sense more closely connected than points that are widely separated in time. Often, this is reflected in a correlation between time points that decreases as the time separation increases. The variance for differences between times typically increases as points move further apart in time.

We have seen that correlation structure is also a key issue in time series analysis. A limitation, relative to repeated measures, is that in time series analysis the structure must typically be estimated from just one series, by assuming that the series is in some sense part of a repeating pattern. In repeated measures there may be many realizations, allowing a relatively accurate estimate of the correlation structure. By contrast with time series, the shortness of the series has no effect on our ability to estimate the correlation structure. Multiple realizations are preferable to a single long series.

While we are typically better placed than in time series analysis to estimate the correlation structure there is, for most of the inferences that we commonly wish to make, less need to know the correlation structure. Typically our interest is in the consistency of patterns between individuals. For example, we may want to know: "Do patients on treatment A improve at a greater rate than patients on treatment B?"

There is a broad distinction between approaches that model the profiles, and approaches that focus more directly on modeling the correlation structure. Direct modeling of the profiles leads to random coefficient models, which allow each individual to follow their own profile. Variation between profiles may largely account for the sequential correlation structure. Direct modeling of the correlation is most effective when there are no evident systematic differences between profiles.

For further discussion of repeated measures modeling, see Diggle *et al.* (2002), Pinheiro and Bates (2000). The Pinheiro and Bates book is based around the S-PLUS version of the *nlme* package.

Different approaches to repeated measures analysis

Traditionally, repeated measures models have been analyzed in many different ways. Here is a summary of methods that have been used:

- A simple way to analyze repeated measures data is to form one or more summary statistics for each subject, and then use these summary statistics for further analysis.
- When the variance is the same at all times and the correlation between results is the same for all pairs of times, data can in principle be analyzed using an analysis of variance model. This allows for two components of variance: (1) between subjects and (2) between times. An implication of this model is that the variance of the difference is the same for all pairs of time points, an assumption that is, in general, unrealistic.
- Various adjustments adapt the analysis of variance approach to allow for the possibility that the variances of time differences are not all equal. These should be avoided now that there are good alternatives to the analysis of variance approach.
- Multivariate comparisons accommodate all possible patterns of correlations between time points. This approach accommodates the time series structure, but does not take advantage of it to find an economical parameterization of the correlation structure.
- Repeated measures models aim to reflect the sequential structure, in the fixed effects, in the random effects, and in the correlation structure. They do this in two ways: by modeling the overall pattern of difference between different profiles, and by direct modeling of the correlation structure. This modeling approach often allows insights that are hard to gain from approaches that ignore or do not take advantage of the sequential structure.

10.6.1 Example – random variation between profiles

The data frame `humanpower1` has data from investigations (Bussolari, 1987, Nadel and Bussolari, 1988) designed to assess the feasibility of a proposed 119-km human-powered flight from the island of Crete – in the initial phase of the *Daedalus* project. After an initial 5-minute warm-up period and 5-minute recovery period, the power requirements from the athletes were increased, at 2-minute intervals, in steps of around 30 Watts. Figure 10.7 gives a visual summary of the data.[10]

We leave it as an exercise to verify, using a fixed effects analysis such as was described in Section 7.3, that separate lines are required for the different athletes, and that there is no case for anything more complicated than straight lines. The separate lines fan out at the upper extreme of power output, consistent with predictions from a random slopes model.

[10]
```
## Plot points and fitted lines (panel A)
library(lattice)
xyplot(o2 ~ wattsPerKg, groups=id, data=humanpower1,
        panel=function(x,y,subscripts,groups,...){
           yhat <- fitted(lm(y ~ groups*x))
           panel.superpose(x, y, subscripts, groups, pch=1:5
)
           panel.superpose(x, yhat, subscripts, groups, type="l")
        },
        xlab="Watts per kilogram",
        ylab=expression("Oxygen intake ("*ml.min^{-1}*.kg^{-1}*")"))
```

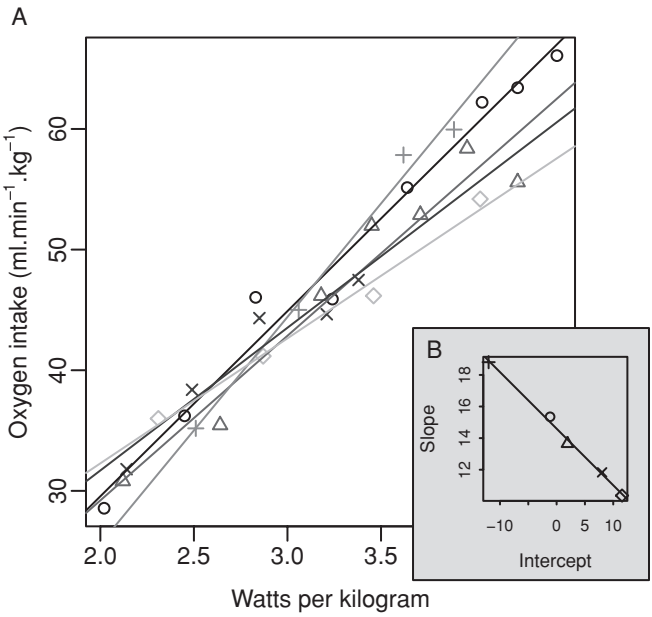


Figure 10.7 Panel A shows oxygen intake, plotted against power output, for each of five athletes who participated in investigations designed to assess the feasibility of a proposed *Daedalus* 119-km human-powered flight. Panel B plots the slopes of these separate lines against the intercepts. A fitted line, with a slope of 2.77, has been added.

Separate lines for different athletes

The model is:

$$y_{ij} = \alpha + \beta x_{ij} + a + b x_{ij} + e_{ij}$$

where i refers to individual, and j to observation j for that individual, α and β are fixed, a and b have a joint bivariate normal distribution, each with mean 0, independently of the e_{ij} which are i.i.d. normal. Each point in Figure 10.7B is a realization of an $(\alpha + a, \beta + b)$ pair.

The following is the code that handles the calculations:

```
## Calculate intercepts and slopes; plot Slopes vs Intercepts
## Uses the function lmList() from the lme4 package
library(lme4)
hp.lmList <- lmList(o2 ~ wattsPerKg | id, data=humanpower1)
coefs <- coef(hp.lmList)
names(coefs) <- c("Intercept", "Slope")
plot(Slope ~ Intercept, data=coefs)
abline(lm(Slope~Intercept, data=coefs))
```

Note the formula `o2 ~ wattsPerKg | id` that is given as argument to the function `lmList()`. For each different level of the factor `id`, there is a regression of `o2` on `wattsPerKg`. Summary information from the calculations is stored in the object `hp.lmList`.

A random coefficients model

Two possible reasons for modeling the variation between slopes as a random effect are:

- There may be an interest in generalizing to further athletes, selected in a similar way – what range of responses is it reasonable to expect?
- The fitted lines from the random slopes model may be a better guide to performance than the fitted "fixed" lines for individual athletes. The range of the slopes for the fixed lines will on average exaggerate somewhat the difference between the smallest and largest slope, an effect which the random effects analysis corrects.

Here, the major reason for working with these data is that they demonstrate a relatively simple application of a random effects model. Depending on how results will be used a random coefficients analysis may well, for these data, be overkill!

The model that will now be fitted allows, for each different athlete, a random slope (for `wattsPerKg`) and random intercept. We expect the correlation between the realizations of the random intercept and the random slope to be close to 1. As it will turn out, this will not create any undue difficulty. Calculations proceed thus:

```
> hp.lmer <- lmer(o2 ~ wattsPerKg + (wattsPerKg | id),
+                 data=humanpower1)
> summary(hp.lmer)
Linear mixed-effects model fit by REML
Formula: o2 ~ wattsPerKg + (wattsPerKg | id)
   Data: humanpower1
    AIC    BIC  logLik MLdeviance REMLdeviance
  134.2  140.9   -62.1      126.9        124.2
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 id       (Intercept) 50.73    7.12
          wattsPerKg   7.15    2.67     -1.000
 Residual              4.13    2.03
# of obs: 28, groups: id, 5

Fixed effects:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     2.09       3.78    0.55     0.58
wattsPerKg     13.91       1.36   10.23  1.3e-10

Correlation of Fixed Effects:
           (Intr)
wattsPerKg -0.992
```

The predicted lines from this random lines model are shown as dashed lines in Figure 10.8A. These are the BLUPs that were discussed earlier in this chapter.

```
hat<-fitted(hp.lmer)
lmhat<-with(humanpower1, fitted(lm(o2 ~ id*wattsPerKg)))
panelfun <-
  function(x, y, subscripts, groups, ...){
```

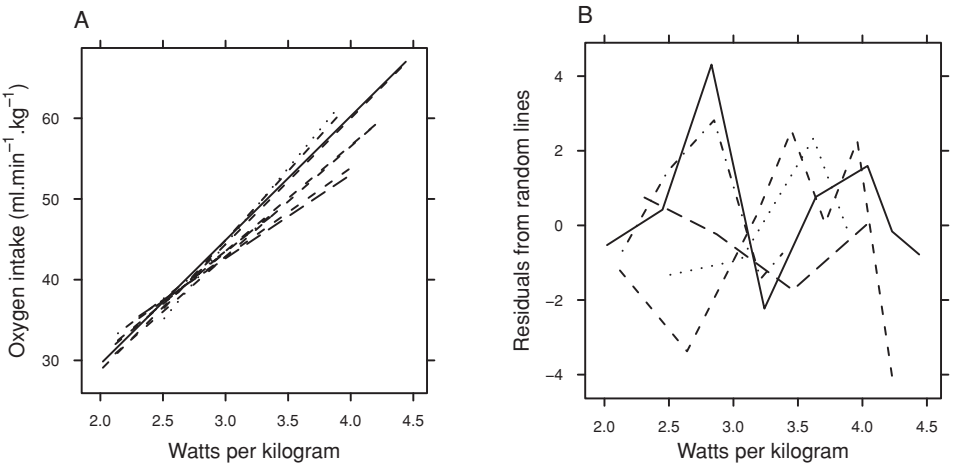


Figure 10.8 In panel A lines have been fitted for each individual athlete, as in Figure 10.7. Also shown, as dashed lines, are the fitted lines from the random lines model. Panel B shows the profiles of residuals from the random lines.

```
            panel.superpose(x, hat, subscripts, groups, type="l",lty=2)
            panel.superpose(x, lmhat, subscripts, groups, type="l",lty=1)
}
xyplot(o2 ~ wattsPerKg, groups=id, data=humanpower1, panel=panelfun,
       xlab="Watts",
       ylab=expression("Oxygen intake ("*ml.min^{-1}*"."*kg^{-1}*")"))
```

Figure 10.8B is a plot of residuals, with the points for each individual athlete connected with broken lines.[11] There is nothing in these residual profiles that obviously calls for attention. For example, none of the athletes shows exceptionally large departures, at one or more points, from the linear trend.

The standard errors relate to the accuracy of prediction of the mean response line for the population from which the athletes were sampled. The slopes are drawn from a distribution with estimated mean 13.9 and standard error $\sqrt{1.36^2 + 2.67^2} = 3.0$. This standard deviation may be compared with the standard deviation (= 3.28) of the five slopes that were fitted to the initial fixed effects model.[12] Standard errors for between-athletes components of variation relate to the particular population from which the five athletes were sampled. Almost certainly, the pattern of variation would be different for five people who were drawn at random from a population of recreational sportspeople.

In this example, the mean response pattern was assumed linear, with random changes, for each individual athlete, in the slope. More generally, the mean response pattern will be non-linear, and random departures from this pattern may be non-linear.

[11] `## Plot of residuals`
`xyplot(resid(hp.lmer) ~ wattsPerKg, groups=id, type="b", data=humanpower1)`

[12] `## Derive the sd from the data frame coefs that was calculated above`
`sd(coefs$Slope)`

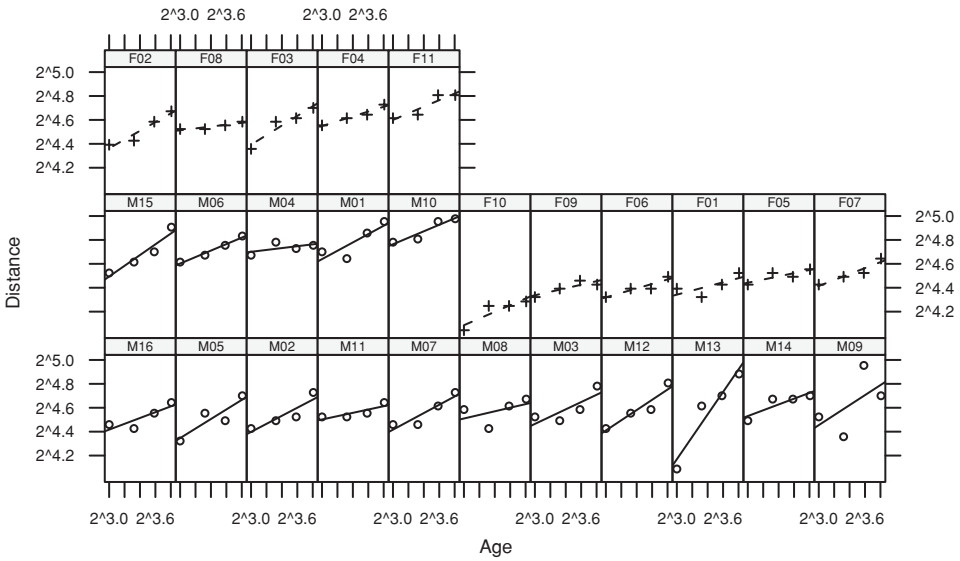


Figure 10.9 Distance between two positions on the skull on a scale of $\log_2$, plotted against age, for each of 27 children.

10.6.2 Orthodontic measurements on children

The `Orthodont` data frame (*MEMSS* package) has measurements on the distance between two positions on the skull, taken every two years from age 8 until age 14, on 16 males and 11 females. Is there a difference in the pattern of growth between males and females?

Preliminary data exploration

Figure 10.9 shows the pattern of change for each of the 25 individuals. Lines have been added; overall the pattern of growth seems close to linear.[13]

A good summary of these data are the intercepts and slopes, as in Figure 10.10. We calculate these both with untransformed distances (panel A) and with distances on a logarithmic scale (panel B). Here is the code:

```
## Use lmList() to find the slopes
ab <- coef(lmList(distance ~ age | Subject, Orthodont))
names(ab) <- c("a", "b")
## Obtain the intercept at x=mean(x)
## (For each subject, this is independent of the slope)
ab$ybar <- ab$a + ab$b*11  # mean age is 11, for each subject.
sex <- substring(rownames(ab), 1 ,1)
plot(ab[, 3], ab[, 2], col=c(F="gray40", M="black")[sex],
     pch=c(F=1, M=3)[sex], xlab="Intercept", ylab="Slope")
```

[13] `## Plot showing pattern of change for each of the 25 individuals`
`library(MEMSS)`
`xyplot(distance ~ age | Subject, groups=Sex, data=Orthodont,`
`scale=list(y=list(log=2)), type=c("p","r"), layout=c(11,3))`

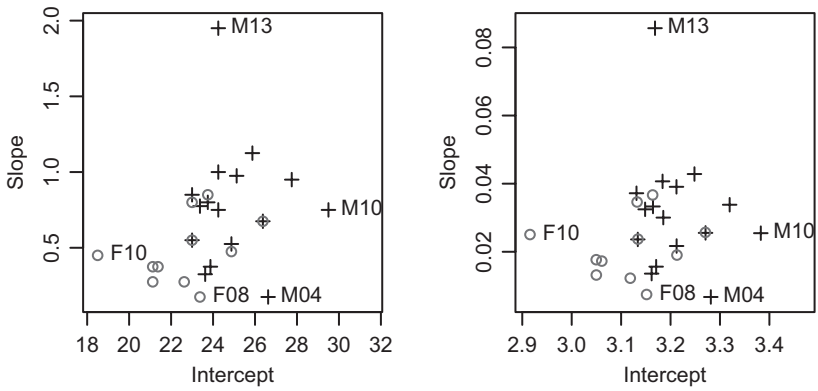


Figure 10.10 Slopes of profiles, plotted against intercepts at age = 11. The females are shown with open circles, and the males with +'s. Panel A is for distances, and panel B is for logarithms of distances.

```
extremes <- ab$ybar %in% range(ab$ybar) |
            ab$b %in% range(ab$b[sex=="M"]) |
            ab$b %in% range(ab$b[sex=="F"])
text(ab[extremes, 3], ab[extremes, 2], rownames(ab)[extremes], pos=4, xpd=TRUE)
## The following makes clear M13's difference from other points
qqnorm(ab$b)
Orthodont$logdist <- log(Orthodont$distance)
## Now repeat, with logdist replacing distance
```

The intercepts for the males are clearly different from the intercepts for the females, as can be verified by a t-test. One slope appears an outlier from the main body of the data. Hence, we omit the largest (M13) and (to make the comparison fair) the smallest (M04) values from the sample of male slopes, before doing a t-test. On the argument that the interest is in relative changes, we will work with logarithms of distances.[14] The output is:

```
        Two Sample t-test

data:  b[sex == "F"] and b[sex == "M" & !extreme.males]
t = -2.32, df = 23, p-value = 0.02957
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.016053 -0.000919
```

[14]
```
## Compare males slopes with female slopes
Orthodont$logdist <- log(Orthodont$distance)
ablog <- coef(lmList(logdist ~ age | Subject, Orthodont))
names(ablog) <- c("a", "b")
## Obtain the intercept at mean age (= 11), for each subject
## (For each subject, this is independent of the slope)
ablog$ybar <- with(ablog, a + b*11)
extreme.males <- rownames(ablog) %in% c("M04","M13")
sex <- substring(rownames(ab), 1, 1)
with(ablog,
  t.test(b[sex=="F"], b[sex=="M" & !extreme.males], var.equal=TRUE))
  # Specify var.equal=TRUE, to allow comparison with anova output
```

```
sample estimates:
mean of x mean of y
   0.0211    0.0296
```

The higher average slope for males is greater than can comfortably be attributed to statistical error.

A random coefficients model

Now consider a random coefficients model. The model will allow different slopes for males and females, with the slope for individual children varying randomly about the slope for their sex. We will omit the same two males as before:

```
> keep <- !(Orthodont$Subject%in%c("M04","M13"))
> orthdiff.lmer <- lmer(logdist ~ Sex * I(age-11) + (I(age-11) | Subject),
+                       data=Orthodont, subset=keep, method="ML")
> orthdiff.lmer
Linear mixed-effects model fit by maximum likelihood
Formula: logdist ~ Sex * I(age - 11) + (I(age - 11) | Subject)
   Data: Orthodont
 Subset: keep
    AIC    BIC logLik MLdeviance REMLdeviance
 -248.9 -230.7  131.5     -262.9       -232.0
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Subject  (Intercept) 6.03e-03 7.77e-02
          I(age - 11) 1.21e-12 1.10e-06 0.000
 Residual             2.42e-03 4.92e-02
# of obs: 100, groups: Subject, 25

Fixed effects:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          3.11451    0.02407  129.39  < 2e-16
SexMale              0.09443    0.03217    2.94   0.0042
I(age - 11)          0.02115    0.00325    6.51  3.4e-09
SexMale:I(age - 11)  0.00849    0.00434    1.96   0.0534
```

Next, we make the fixed slope effect the same for both sexes, and compare the two models:

```
> orthsame.lmer <-
+   lmer(logdist ~ Sex + I(age - 11) + (I(age - 11) | Subject),
+        data=Orthodont, method="ML", subset=keep)
> anova(orthsame.lmer, orthdiff.lmer)
Data: Orthodont
Subset: keep
Models:
orthsame.lmer: logdist ~ Sex + I(age - 11) + (I(age - 11) | Subject)
orthdiff.lmer: logdist ~ Sex * I(age - 11) + (I(age - 11) | Subject)
```

```
              Df    AIC    BIC logLik Chisq Chi Df Pr(>Chisq)
orthsame.lmer  6 -247.2 -231.6  129.6
orthdiff.lmer  7 -248.9 -230.7  131.5  3.73      1     0.0534
```

Note that this compared the two models, while the *t*-test that was carried out above compared the slopes for the males with the slopes for the females.

The estimates of fixed effects from the REML model are in general preferable to those from the full maximum likelihood (ML) model.

```
> orthdiffr.lmer <- update(orthdiff.lmer, method="REML")
> summary(orthdiffr.lmer)
Linear mixed-effects model fit by REML
Formula: logdist ~ Sex * I(age - 11) + (I(age - 11) | Subject)
   Data: Orthodont
 Subset: keep
    AIC    BIC logLik MLdeviance REMLdeviance
 -218.1 -199.8  116.0     -262.9       -232.1
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Subject  (Intercept) 6.33e-03 7.96e-02
          I(age - 11) 1.19e-12 1.09e-06 0.000
 Residual             2.38e-03 4.88e-02
number of obs: 100, groups: Subject, 25

Fixed effects:
                    Estimate Std. Error t value
(Intercept)          3.11451    0.02510   124.1
SexMale              0.09443    0.03354     2.8
I(age - 11)          0.02115    0.00329     6.4
SexMale:I(age - 11)  0.00849    0.00440     1.9

Correlation of Fixed Effects:
            (Intr) SexMal I(-11)
SexMale     -0.748
I(age - 11)  0.000  0.000
SxMl:I(-11)  0.000  0.000 -0.748
```

The estimate 1.19e-12 of the slope component of variance is for all practical purposes zero. The variation in the slope of lines is entirely explained by variation of individual points about lines, within and between subjects of the same sex. The output suggests that slopes differ between males and females.

The function `mcmcsamp()` can in principle be used for yet another check on the `Sex:age` term, thus:

```
## At the time of writing, this is not possible
## orth.mcmc <- mcmcsamp(orthdiffr.lmer, n=1000)
## HPDinterval(orth.mcmc)
```

Correlation between successive times

We can calculate the autocorrelations across each subject separately, and check the distribution. The interest is in whether any autocorrelation is consistent across subjects.

```
> res <- resid(orthdiffr.lmer)
> Subject <- factor(Orthodont$Subject[keep])
> orth.acf <- t(sapply(split(res, Subject),
+                      function(x)acf(x, lag=4, plot=FALSE)$acf))
> ## Calculate respective proportions of Subjects for which
> ## autocorrelations at lags 1, 2 and 3 are greater than zero.
> apply(orth.acf[,-1], 2, function(x)sum(x>0)/length(x))
[1] 0.20 0.24 0.40
```

Thus a test for a zero lag 1 autocorrelation has $p = 0.20$. The suggestion of non-zero autocorrelation is very weakly supported.

**The variance for the difference in slopes*

This can be calculated from the components of variance information. The sum of squares about the mean, for one line, is $\sum(x - \bar{x})^2 = 20$. The sum of the two components of variance for an individual line is then: $1.19 \times 10^{-12} + 0.002383/20 = 0.00011915$. The standard error of the difference in slopes is then:

$$\sqrt{0.00011915(1/14 + 1/11)} = 0.00440.$$

Compare this with the value given against the fixed effect `SexMale:I(age - 11)` in the output above. The numbers are, to within rounding error, the same. Degrees of freedom for the comparison are 23 as for the t-test.

10.7 Further notes on multi-level and other models with correlated errors

10.7.1 Different sources of variance – complication or focus of interest?

In the discussion of multi-level models, the main interest was in the parameter estimates. The different sources of variance were a complication. In other applications, the variances may be the focus of interest. Many animal and plant breeding trials are of this type. The aim may be to design a breeding program that will lead to an improved variety or breed. Where there is substantial genetic variability, breeding experiments have a good chance of creating improved varieties.

Investigations into the genetic component of human intelligence have generated fierce debate. Most such studies have used data from identical twins who have been adopted out to different homes, comparing them with non-identical twins and with sibs who have been similarly adopted out. The adopting homes rarely span a large part of a range from extreme social deprivation to social privilege, so that results from such studies may have little or no relevance to investigation of the effects of extreme social deprivation. The discussion in Leavitt and Dubner (2005, Chapter 5) sheds interesting light on these effects.

There has not been, until recently, proper allowance for the substantial effects that arise from simultaneous or sequential occupancy of the maternal womb (Bartholemew, 2004, Daniels *et al*., 1997). Simple forms of components of variance model are unable to account for the Flynn effect (Bartholemew, 2004, pp. 138–140), by which measured IQs in many parts of the world have in recent times increased by about 15 IQ points per generation. The simple model, on which assessments of proportion of variance that is genetic have been based, seems too simplistic to give useful insight.

We have used an analysis of data from a field experimental design to demonstrate the calculation and use of components of variance. Other contexts for multi-level models are the analysis of data from designed surveys, and general regression models in which the "error" term is made up of several components. In all these cases, errors are no longer independently and identically distributed.

10.7.2 Predictions from models with a complex error structure

Here, "complex" refers to models that assume something other than an i.i.d. error structure. Most of the models considered in this chapter can be used for different predictive purposes, and give standard errors for predicted values that differ according to the intended purpose. Accurate modeling of the structure of variation allows, as for the Antiguan corn yield data in Section 10.1, these different inferential uses.

As has been noted, shortcuts are sometimes possible. Thus for using the kiwifruit shading data to predict yields at any level other than the individual vine, there is no loss of information from basing the analysis on plot means.

Consequences from assuming an overly simplistic error structure

In at least some statistical application areas, analyses that assume an overly simplistic error structure (usually, an i.i.d. model) are relatively common in the literature. Inferences may be misleading, or not, depending on how results are used. Where there are multiple levels of variation, all variation that contributes to the sampling error of fixed effects must be modeled correctly. Otherwise, the standard errors of model parameters that appear in computer output will almost inevitably be wrong, and should be ignored.

In data that have appropriate balance, predicted values will ordinarily be unbiased, even if the error structure is not modeled appropriately. The standard errors will almost certainly be wrong, usually optimistic. A good understanding of the structure of variation is typically required in order to make such limited inferences as are available when an overly simplistic error structure is assumed!

10.7.3 An historical perspective on multi-level models

Multi-level models are not new. The inventor of the analysis of variance was R. A. Fisher. Although he would not have described it that way, many of the analysis of variance calculations that he demonstrated were analyses of specific forms of multi-level model. Data have a structure that is imposed by the experimental design. The particular characteristic of the experimental design models that Fisher used was that the analysis could be handled

using analysis of variance methods. The variance estimates that are needed for different comparisons may be taken from different lines of the analysis of variance table. This circumvents the need to estimate the variances of the random effects that appear in a fully general analysis.

Until the modern computing era, multi-level data whose structure did not follow one of the standard designs and thus did not fit the analysis of variance framework required some form of approximate analysis. Such approximate analyses, if they were possible at all, often demanded a high level of skill.

Statistical analysts who used Fisher's experimental designs and methods of analysis followed Fisher's rules, and those of his successors, for the calculations. Each different design had its own recipe. After working through a few such analyses, some of those who followed Fisher's methods began to feel that they understood the rationale fairly well at an intuitive level. Books appeared that gave instructions on how to do the analyses. The most comprehensive of these is Cochran and Cox (1957).

The Genstat system (Payne *et al*., 1997) was the first of the major systems to implement general methods for the analysis of multi-level models that had a suitable "balance". Its coherent and highly structured approach to the analysis of data from suitably balanced designs takes advantage of the balance to simplify and structure the output.

General-purpose software for use with unbalanced data, in the style of the *lme4* package, made its appearance relatively recently. The analyses that resulted from earlier ad hoc approaches were in general less insightful and informative than the more adequate analyses that are available within a multi-level modeling framework.

Regression models are another starting point for consideration of multi-level models. Both the fixed effects parts of the model have a structure, thus moving beyond the models with a single random (or "error") term that have been the stock in trade of courses on regression modeling. Even now, most regression texts give scant recognition of the implications of structure in the random part of the model. Yet data commonly do have structure – students within classes within institutions, nursing staff within hospitals within regions, managers within local organizations within regional groupings, and so on.

As has been noted, models have not always been written down. Once theoretical statisticians did start to write down models, there was a preoccupation with models that had a single error term. Theoretical development, where the discussion centered around models, was disconnected from the practical analysis of experimental designs, where most analysts were content to follow Cochran and Cox and avoid formal mathematical description of the models that underpinned their analyses.

Observational data that have a multi-level structure, which is typically unbalanced, can nowadays be analyzed just as easily as experimental data. It is no longer necessary to look up Cochran and Cox to find how to do an analysis. There are often acceptable alternatives to Cochran and Cox-style experimental designs.

Problems of interpretation and meaningfulness remain, for observational data, as difficult as ever. The power of modern software can become a trap. There may be inadequate care in the design of data collection, in the expectation that computer software will take care of any problems. The result may be data whose results are hard to interpret or cannot be interpreted at all, or that make poor use of resources.

10.7.4 Meta-analysis

Meta-analysis is a name for analyses that bring together into a single analysis framework data from, for example, multiple agricultural trials, or from multiple clinical trials, or from multiple psychological laboratories. Multi-level modeling, and extensions of multi-level modeling such as repeated measures analysis, make it possible to do analyses that take proper account of site-to-site or center-to-center or study-to-study variation. If treatment or other effects are consistent relative to all identifiable major sources of variation, the result can be compelling for practical application.

Meta-analysis is uncontroversial when data are from a carefully planned multi-location trial. More controversial is the bringing together into one analysis of data from quite separate investigations. There may however be little choice; the alternative may be an informal and perhaps unconvincing qualitative evaluation of the total body of evidence. Clearly such analyses challenge the critical acumen of the analyst. A wide range of methodologies have been developed to handle the problems that may arise. Gaver *et al.* (1992) is a useful summary. Turner *et al.* (2009) is an interesting and comprehensive state-of-the-art account.

10.7.5 Functional data analysis

Much of the art of repeated measures modeling lies in finding suitable representations, requiring a small number of parameters, both of the individual profiles and of variation between those profiles. Spline curves are widely used in this context. Chapter 12 will discuss the use of principal components to give a low-dimensional representation of multivariate data. A similar methodology can be used to find representations of curves in terms of a small number of basis functions. Further details are in Ramsay and Silverman (2002).

10.7.6 Error structure in explanatory variables

This chapter has discussed error structure in response variables. There may also be a structure to error in explanatory variables. Studies of the health effects of dietary components, such as were described in Section 6.7, provide an interesting and important example, with major implications for the design of such studies.

10.8 Recap

Multi-level models account for multiple levels of random variation. The random part of the model possesses structure; it is a sum of distinct random components.

In making predictions based on multi-level models, it is necessary to identify precisely the population to which the predictions will apply.

The art in setting up an analysis for these models is in getting the description of the model correct. Specifically it is necessary to

- identify which are fixed and which random effects,
- correctly specify the nesting of the random effects.

In repeated measures designs, it is necessary to specify or otherwise model the pattern of correlation within profiles.

A further generalization is to the modeling of random coefficients, for example, regression lines that vary between different subsets of the data.

Skill and care may be needed to get output into a form that directly addresses the questions that are of interest. Finally, output must be interpreted. Multi-level analyses often require high levels of professional skill.

10.9 Further reading

Fisher (1935) is a non-mathematical account that takes the reader step by step through the analysis of important types of experimental design. It is useful background for reading more modern accounts. Williams *et al.* (2002) is similarly example-based, with an emphasis on tree breeding. See also Cox (1958), Cox and Reid (2000). Cox and Reid is an authoritative up-to-date account of the area, with a more practical focus than its title might seem to imply.

On multi-level and repeated measures models see Gelman and Hill (2007), Snijders and Bosker (1999), Diggle *et al.* (2002), Goldstein (1995), Pinheiro and Bates (2000), Venables and Ripley (2002).

Talbot (1984) is an interesting example of the use of multi-level modeling, with important agricultural and economic implications. It summarizes a large amount of information that is of importance to farmers, on yields for many different crops in the UK, including assessments both of center-to-center and of year-to-year variation.

The relevant chapters in Payne *et al.* (1997), while directed to users of the Genstat system, have helpful commentary on the use of the methodology and on the interpretation of results. Pinheiro and Bates (2000) describes the use of the *nlme* package for handling multi-level analyses.

On meta-analysis see Chalmers and Altman (1995), Gaver *et al.* (1992), Turner *et al.* (2009).

References for further reading

Analysis of variance with multiple error terms

Cox, D. R. 1958. *Planning of Experiments.*

Cox, D. R. and Reid, N. 2000. *Theory of the Design of Experiments.*

Fisher, R. A. 1935 (7th edn, 1960). *The Design of Experiments.*

Payne, R. W., Lane, P. W., Digby, P. G. N., Harding, S. A., Leech, P. K., Morgan, G. W., Todd, A. D., Thompson, R., Tunnicliffe Wilson, G., Welham, S. J. and White, R. P. 1997. *Genstat 5 Release 3 Reference Manual.*

Williams, E. R., Matheson, A. C. and Harwood, C. E. 2002. *Experimental Design and Analysis for Use in Tree Improvement*, revised edn.

Multi-level models and repeated measures

Diggle, P. J., Heagerty, P. J., Liang, K.-Y. and Zeger, S. L. 2002. *Analysis of Longitudinal Data*, 2nd edn.

Gelman, A. and Hill, J. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*.

Goldstein, H. 1995. *Multi-level Statistical Models.*

Payne, R. W., Lane, P. W., Digby, P. G. N., Harding, S. A., Leech, P. K., Morgan, G. W., Todd, A. D., Thompson, R., Tunnicliffe Wilson, G., Welham, S. J. and White, R. P. 1997. *Genstat 5 Release 3 Reference Manual*.

Pinheiro, J. C. and Bates, D. M. 2000. *Mixed Effects Models in* S *and* S-PLUS.

Snijders, T. A. B. and Bosker, R. J. 1999. *Multilevel Analysis. An Introduction to Basic and Advanced Multilevel Modelling*.

Talbot, M. 1984. Yield variability of crop varieties in the UK. *Journal of the Agricultural Society of Cambridge* 102: 315–321. *JRSS* A 172: 21–47.

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

Meta-analysis

Chalmers, I. and Altman, D. G. 1995. *Systematic Reviews*.

Gaver, D. P., Draper, D. P., Goel, K. P., Greenhouse, J. B., Hedges, L. V., Morris, C. N. and Waternaux, C. 1992. *Combining Information: Statistical Issues and Opportunities for Research*.

Turner, R. M., Spiegelhalter, D. J., Smith, G. C. S and Thompson, S. G. 2009. Bias modelling in evidence synthesis. *Journal of the Royal Statistical Society, Series A* 172(1): 21–47.

10.10 Exercises

1. Repeat the calculations of Subsection 10.3.5, but omitting results from two vines at random. Here is code that will handle the calculation:

```
n.omit <- 2
take <- rep(TRUE, 48)
take[sample(1:48,2)] <- FALSE
kiwishade.lmer <- lmer(yield ~ shade + (1|block) + (1|block:plot),
                       data = kiwishade,subset=take)
vcov <- VarCorr(kiwishade.lmer)
vars <- c("(block:plot)^2"=as.vector(vcov[["block:plot"]]),
          "sigma^2"=as.vector(attributes(vcov, "sigmaREML")$sc^2))
print(vars)
```

   Repeat this calculation five times, for each of `n.omit` = 2, 4, 6, 8, 10, 12 and 14. Plot (i) the plot component of variance and (ii) the vine component of variance, against number of points omitted. Based on these results, for what value of `n.omit` does the loss of vines begin to compromise results? Which of the two components of variance estimates is more damaged by the loss of observations? Comment on why this is to be expected.

2. Repeat the previous exercise, but now omitting 1, 2, 3, 4 complete plots at random.

3. The data set `Gun` (*MEMSS* package) reports on the numbers of rounds fired per minute, by each of nine teams of gunners, each tested twice using each of two methods. In the nine teams, three were made of men with slight build, three with average, and three with heavy build. Is there a detectable difference, in number of rounds fired, between build type or between firing methods?

For improving the precision of results, which would be better – to double the number of teams, or to double the number of occasions (from 2 to 4) on which each team tests each method?

4*. The data set `ergoStool` (*MEMSS* package) has data on the amount of effort needed to get up from a stool, for each of nine individuals who each tried four different types of stool. Analyze the data both using `aov()` and using `lme()`, and reconcile the two sets of output. Was there any clear winner among the types of stool, if the aim is to keep effort to a minimum?

5*. In the data set `MathAchieve` (*MEMSS* package), the factors `Minority` (levels `yes` and `no`) and `sex`, and the variable `SES` (socio-economic status) are clearly fixed effects. Discuss how the decision whether to treat `School` as a fixed or as a random effect might depend on the purpose of the study? Carry out an analysis that treats `School` as a random effect. Are differences between schools greater than can be explained by within-school variation?

6*. The data frame `sorption` (*DAAG*) includes columns `ct` (concentration–time sum), `Cultivar` (apple cultivar), `Dose` (injected dose of methyl bromide), and `rep` (replicate number, within `Cultivar` and `year`). Fit a model that allows the slope of the regression of `ct` on `Dose` to be different for different cultivars and years, and to vary randomly with replicate. Consider the two models:

```
cult.lmer <- lmer(ct ~ Cultivar + Dose + factor(year) +
                  (-1 + Dose | gp), data = sorption,
                  REML=TRUE)
cultdose.lmer <- lmer(ct ~ Cultivar/Dose + factor(year) +
                      (-1 + Dose | gp), data = sorption,
                      REML=TRUE)
```

Explain (i) the role of each of the terms in these models, and (ii) how the two models differ. Which model seems preferable? Write a brief commentary on the output from the preferred model.
[NB: The factor `gp`, which has a different level for each different combination of `Cultivar`, `year` and replicate, associates a different random effect with each such combination.]

11

Tree-based classification and regression

Tree-based methods, or *decision tree* methods, may be used for two broad types of problem – classification and regression. These methods may be appropriate when there are extensive data, and there is uncertainty about the form in which explanatory variables ought to enter into the model. They may be useful for initial data exploration. Tree-based methods have been especially popular in the data mining community.

Tree-structured classification has a long history in biology, where informal methods of dendrogram construction have been in use for centuries. Social scientists began automating tree-based procedures for classification in the 1940s and 1950s, using methods which are similar to some of the current partitioning methods; see Belson (1959). Venables and Ripley (2002, Chapter 9) give a short survey of the more recent history.

The tree-based regression and classification methodology is radically different from the methods discussed thus far in this book. The theory that underlies the methods of earlier chapters has limited relevance to tree-based methods. The methodology is relatively easy to use and can be applied to a wide class of problems. It is at the same time insensitive to the nuances of particular problems to which it may be applied.

The methodology makes limited use of the ordering of values of continuous or ordinal explanatory variables. In small data sets, it is unlikely to reveal data structure. Its strength is that, in large data sets, it has the potential to reflect relatively complex forms of structure, of a kind that may be hard to detect with conventional regression modeling.

There are various refinements on tree-based methods that, by building multiple trees, improve on the performance of the single-tree methods that will occupy the first part of this chapter. The chapter will end with a discussion of `randomForest()` and related functions in the *randomForest* package. This takes multiple bootstrap random samples (the default is 500), generating a separate tree for each such sample. The prediction is determined by a simple majority vote across the multiple trees.

There are other refinements on tree-based models that build in less structure than classical regression modeling, but more structure than classification and regression trees. Note also that the inferences that we describe assume that the random part of the model is independent between observations.

Note

This chapter will use both the *rpart* package and (in Section 11.7) the *randomForest* package. The *rpart* package (Therneau and Atkinson, 1997) is one of the recommended

packages, included with all binary distributions of R. It will be necessary to download and install *randomForest* (Liaw and Wiener, 2002).

11.1 The uses of tree-based methods

11.1.1 Problems for which tree-based regression may be used

We will consider four types of problem:

- regression with a continuous outcome variable,
- regression with a binary outcome variable,
- classification with responses which can have one of several possible ordered outcomes,
- classification with responses which can have one of several possible (unordered) outcomes.

There are other possibilities, including tree-based survival analysis, which the *rpart* package implements.

We will use a data set on email spam to motivate the discussion, showing how a classification tree can help distinguish legitimate messages from spam. For simplicity, this initial exploration of these data will limit attention to 6 of the 57 available explanatory variables. The choice is the first author's intuition, educated by his exposure to email spam.

We will next demonstrate and explain the tree-based regression methodology as it applies to data with a continuous outcome variable. Several simple toy data sets, i.e., data sets that have been constructed for demonstration purposes, will be used to help explain the methodology. Also used mainly for illustrating the methodology is a data set that has mileage versus weight, for cars described in US April 1990 *Consumer Reports*.

The main focus of the central part of this chapter will be on classification with two outcomes, i.e., problems that can be treated as binary regression problems. Our primary example, with 11 explanatory variables, examines the mortality of hospitalized female heart attack patients. For such binary outcomes, the use of a classification tree is a competitor for a logistic or related regression. Unordered classification with multiple outcomes is a straightforward extension for which however this chapter has not found room.

When are tree-based methods appropriate?

In small data sets, it may be necessary to use parametric models that build in quite strong assumptions, e.g., no interactions, and the assumption of linear effects for the main explanatory variables, in order to gain adequate predictive power. Results are, as a consequence, likely to contain some bias, but this is often a price worth paying. In larger data sets, weaker assumptions are possible, e.g., allow simple forms of curve rather than lines and include interactions. Tree-based methods may be helpful for data sets that are large enough that very limited assumptions will allow useful inference. Even for such data sets, however, interesting structure may be missed because assumptions have been too weak. There is, additionally, an optimality issue. While each local split is optimal, the overall tree may be far from optimal.

Exploration of a new data set with tree-based regression or classification can sometimes gain a quick handle on which variables have major effects on the outcome. Note also, as

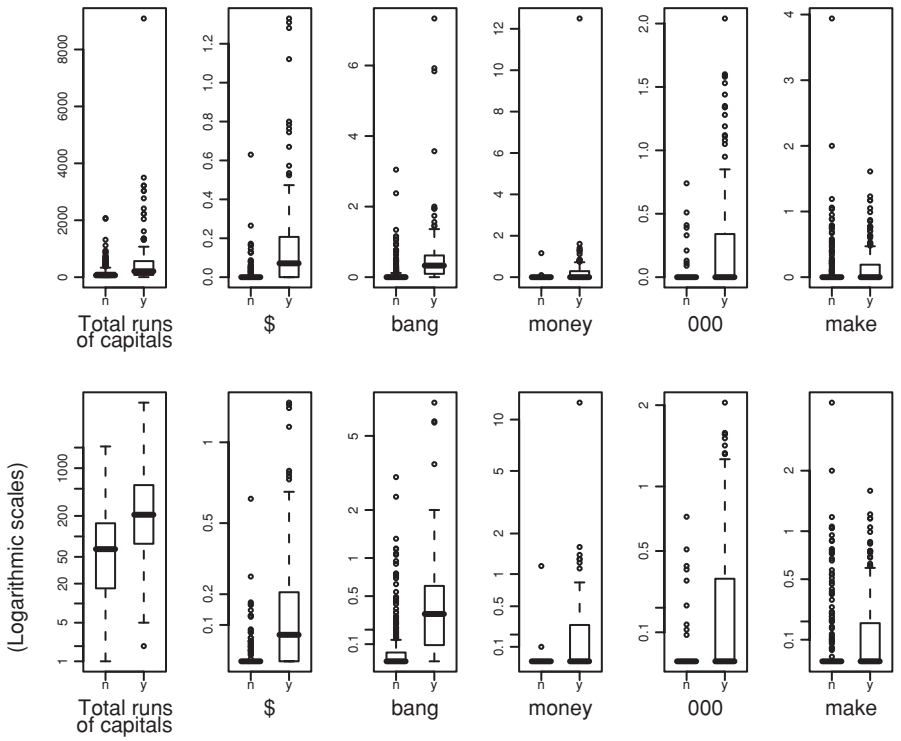


Figure 11.1 Boxplots for the 6 selected explanatory variables, in a random sample of 500 out of 4601 rows (email messages) in the SPAM database. (Presenting data for a subsample makes it easier to see the pattern of outliers.) The upper series of panels are for untransformed data, while in the lower series of panels 0.5 was added to each variable prior to logarithmic transformation.

will be discussed in Section 11.8, that tree-based approaches can sometimes be used in tandem with more classical approaches, in ways that combine the strengths of the different approaches.

11.2 Detecting email spam – an example

The originator of these data, George Forman, of Hewlett–Packard Laboratories, collected 4601 email items, of which 1813 items were identified as spam; these data are available from the web site for the reference Blake and Merz (1998).

Here, for simplicity, we will work with 6 of the 57 explanatory variables on which Forman collected information. In Figure 11.1 (untransformed data in the upper series of panels; log transformed data in the lower series of panels) we show boxplots for 6 of the variables, based in each instance on 500-row samples. In the lower series of panels, we added 0.5 before taking logarithms.[1]

[1]
```
## Obtain 500-row sample; repeat the first plot (of crl.tot)
spam.sample <- spam7[sample(seq(1,4601), 500, replace=FALSE), ]
boxplot(split(spam.sample$crl.tot, spam.sample$yesno))
```

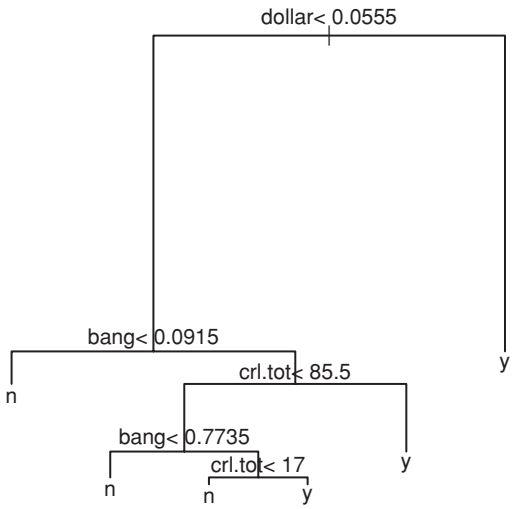


Figure 11.2 Output from the function `rpart()`, with `method="class"`, for classification with the email spam data set, using the 6 explanatory variables noted in the text. At each split, observations for which the condition is satisfied take the branch to the left.

The explanatory variables are:

- `crl.tot`, total length of words that are in capitals;
- `dollar`, the frequency of the $ symbol, as a percentage of all characters;
- `bang`, the frequency of the ! symbol, as a percentage of all characters;
- `money`, frequency of the word "money", as a percentage of all words;
- `n000`, frequency of the character string "000", as a percentage of all words;
- `make`, frequency of the word "make", as a percentage of all words.

The outcome is the factor `yesno`, which is `n` for non-spam and `y` for spam.

For use of logistic regression or another parametric technique with these data, use of transformations, for the final five variables, more severe than the logarithmic transformation used in the lower series of panels, seems required. For tree-based regression, this is not necessary. This is fortunate, because it is not straightforward to find transformations that work well with these data.

Now consider a tree-based classification, using the same 6 variables as predictors. Except for the setting `method="class"` that is anyway the default when the outcome is a factor, we use default settings of control parameters. The code is:

```
library(rpart)
spam.rpart <- rpart(formula = yesno ~ crl.tot + dollar + bang +
                    money + n000 + make,  method="class", data=spam7)
plot(spam.rpart)      # Draw tree
text(spam.rpart)      # Add labeling
```

Figure 11.2 shows the tree.

Table 11.1 *The final row gives information on the performance of the decision tree in Figure 11.2. Earlier rows show the performances of trees with fewer splits.*

```
> printcp(spam.rpart)

Classification tree:
rpart(formula = yesno ~ crl.tot + dollar + bang + money +
    n000 + make, method="class", data = spam7)

Variables actually used in tree construction:
[1] bang    crl.tot dollar

Root node error: 1813/4601 = 0.394

n= 4601

      CP nsplit rel error xerror   xstd
1 0.4766      0     1.000  1.000 0.0183
2 0.0756      1     0.523  0.557 0.0155
3 0.0116      3     0.372  0.386 0.0134
4 0.0105      4     0.361  0.378 0.0133
5 0.0100      5     0.350  0.377 0.0133
```

Reading the tree is done as follows. If the condition that is specified at a node is satisfied, then we take the branch to the left. Thus, dollar (\$) < 0.0555 and bang (!) < 0.0915 leads to the prediction that the email is not spam.

Table 11.1, obtained using `printcp(spam.rpart)`, displays information on the predictive accuracy of the tree. It gives two types of error rate, both expressed as multiples of the root node error rate, here equal to 0.394:

(1) The final row of the column headed `rel error` gives the relative error rate for predictions for the data that generated the tree: 35%. Multiplication by the root node error gives an absolute error rate of $0.394 \times 0.35 = 0.138$, or 13.8%. This is often called the *resubstitution* error rate. It can never increase as tree size increases, gives an optimistic assessment of relative error in a new sample, and is of no use in deciding tree size.
(2) The column headed `xerror` presents the more useful measure of performance. The `x` in `xerror` is an abbreviation for cross-validated. The absolute cross-validated error rate is $0.394 \times 0.377 = 0.149$, or 14.9%.

The cross-validated error rate estimates the expected error rate for use of the prediction tree with new data that are sampled in the same way as the data used to derive Figure 11.2. Examination of the cross-validated error rate suggests that five splits may be marginally better than four splits. The use of six or more splits will be investigated later, in Section 11.6.

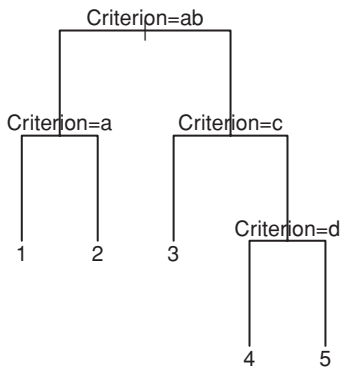


Figure 11.3 Tree labeling. For this illustrative tree, the only term used to determine the splitting is a factor with the name "Criterion". The *split* labels give the factor levels (always coded `a`, `b`, `...`) that lead to choosing the left branch. More generally, different variable or factor names may appear at different nodes. Terminal nodes (*leaves*) are numbered 1, . . . , 5.

11.2.1 Choosing the number of splits

In classical regression, the inclusion of too many explanatory variables may lead to a loss of predictive power, relative to a more parsimonious model. With tree-based methods, the more immediate issue is the number of splits, rather than the number of explanatory variables. Choice of a tree whose cross-validation error is close to the minimum protects against choosing too many splits. The email spam example will be the basis for a later, more extended, investigation.

The next section will explain the terminology and the methodology for forming trees. From there, the discussion will move to showing the use of cross-validation for assessing predictive accuracy and for choosing the optimal size of tree. Calculations are structured to determine a sequence of splits, to get unbiased assessments of predictive accuracy for a range of sizes of tree, and to allow an assessment of optimal tree size.

11.3 Terminology and methodology

The simple tree in Figure 11.3 illustrates basic nomenclature and labeling conventions.[2] In Figure 11.3, there is a single factor (called `Criterion`) with five levels. By default, the function `text.rpart()` labels the levels `a`, `b`, . . . , in order. Each *split* carries a label that gives the decision rule that was used in making the split, e.g., `Criterion=ac`, etc. The label gives the factor levels, or more generally the range of variable values, that lead to choosing the left branch. We have arranged, for this trivial example, that the outcome value is 1 for the level `a`, 2 for level `b`, etc.

[2]
```
## Code to plot tree
Criterion <- factor(paste("Leaf", 1:5))
Node <- c(1,2,3,4,5)
demo.df <- data.frame(Criterion = Criterion, Node = Node)
demo.rpart <- rpart(Node ~ Criterion, data = demo.df,
                    control = list(minsplit = 2, minbucket = 1))
plot(demo.rpart, uniform=TRUE)
text(demo.rpart)
```

There are nine *nodes* in total, made up of four splits and five terminal nodes or *leaves*. In this example, the leaves are labeled 1, 2, . . . , 5. The number of splits is always one less than the number of leaves. Output from the *rpart* package uses the number of splits as a measure of the size of the tree.

11.3.1 Choosing the split – regression trees

The use of `method="anova"` is the default when the outcome variable is a continuous or ordinal variable. The anova splitting rule minimizes the residual sum of squares, called *deviance* in the output from `rpart()`. It is calculated in the manner that will now be described.

Let $\mu_{[j]}$ be the mean for the cell to which y_j is currently assigned. Then the residual sum of squares is

$$D = \sum_j (y_j - \mu_{[j]})^2.$$

Calculations proceed as in forward stepwise regression. At each step, the split is chosen so as to give the maximum reduction in D. Observe that D is the sum of the "within-cells" sums of squares.

Prior to the first split, the deviance is

$$D = \sum_j (y_j - \bar{y})^2.$$

The split will partition the set of subscripts j into two subsets – a set of j_1s (write $\{j_1\}$) and a set of j_2s (write $\{j_2\}$). For any such partition,

$$D = \sum_j (y_j - \bar{y})^2 = \sum_{j_1} (y_{j_1} - \bar{y}_1)^2 + \sum_{j_2} (y_{j_2} - \bar{y}_2)^2 + n_1(\bar{y}_1 - \bar{y})^2 + n_2(\bar{y}_2 - \bar{y})^2$$

where the first two terms comprise the new "within-group" sum of squares, and the final two terms make up the "between-group" sum of squares. If the values y_j are ordered, then the partition between $\{j_1\}$ and $\{j_2\}$ must respect this ordering, i.e., the partitioning is defined by a cutpoint. If the outcome variable is an unordered factor, i.e., if the values are unordered, then every possible split into $\{j_1\}$ and $\{j_2\}$ must in principle be considered.

The split is chosen to make the sum of the first two terms as small as possible, and the sum of the final two terms as large as possible. The approach is equivalent to maximizing the between-groups sum of squares, as in a one-way analysis of variance.

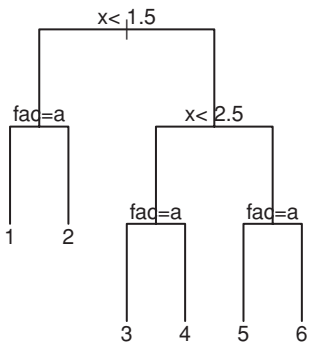
For later splits, each of the current cells is considered as a candidate for splitting, and the split is chosen that gives the maximum reduction in the residual sum of squares.

11.3.2 Within and between sums of squares

Typically there will be several explanatory variables and/or factors. In the toy example of Figure 11.4 there is one explanatory factor (with levels "left" and "right") and one

Table 11.2 *Comparison of candidate splits at the* root, *i.e., at the first split.*

	Overall mean	Cell means	"Within" sum of squares	"Between" sum of squares
`fac=="a"` (`y==1,3,5`) versus `fac=="b"` (`y==2,4,6`)	3.5	3, 4	$8 + 8 = 16$	$3 \times (3 - 3.5)^2 + 3 \times (4 - 3.5)^2 = 1.5$
`x==1` (`y==1,2`) versus `x==2` or `3` (`y==3,4,5,6`)	3.5	1.5, 4.5	$0.5 + 5.0 = 5.5$	$2 \times (1.5 - 3.5)^2 + 4 \times (4.5 - 3.5)^2 = 12$
`x==1` or `2` (`y=1,2,3,4`) versus `x==3` (`y==5, 6`)	3.5	2.5, 5.5	$5.0 + 0.5 = 5.5$	$4 \times (2.5 - 3.5)^2 + 2 \times (5.5 - 3.5)^2 = 12$



fac	x	y
left	1	1
right	1	2
left	2	3
right	2	4
left	3	5
right	3	6

Figure 11.4 This illustrative tree is generated from one explanatory factor (`fac`) and one explanatory variable (`x`). The toy data used to generate the tree are shown to the right of the tree. For factors, the levels that lead to a branch to the left are given. The levels are labeled a, b, ..., rather than with the actual level names. For variables, the range of values is given that leads to taking the left branch.

explanatory variable `x`. The dependent variable is `y`. Observe that, here, $y_j = j$ ($j = 1, 2, \ldots, 6$). Table 11.2 shows the sums of squares calculations for the candidate splits at the root.

The split between `fac=="a"` and `fac=="b"` gives a between sum of squares = 1.5, and a "within" sum of squares = 16. Either of the splits on `x` (between `x = 1` and `x > 1`, or between `x < 3` and `x = 3`) give a within sum of squares equal to 5.5. As the split on `x` leads to the smaller "within" sum of squares, it is the first of the splits on `x` that is chosen. (The second of the splits on `x` might equally well be taken. The tie must be resolved somehow!)

The algorithm then looks at each of the subcells in turn, and looks at options for splits within each subcell. The split chosen is the one that gives the largest "between" sum of squares, for the two new subcells that are formed.

11.3.3 Choosing the split – classification trees

To request a classification tree, include the argument `method="class"` in the `rpart()` call. This setting is the default when the outcome is a factor, but it is best to state it explicitly.

The classes (indexed by k) are the categories of the classification. Then n_{ik} is the number of observations at the ith leaf that are assigned to class k. The n_{ik} are used to estimate the proportions p_{ik}. Each leaf becomes, in turn, a candidate to be the node for a new split.

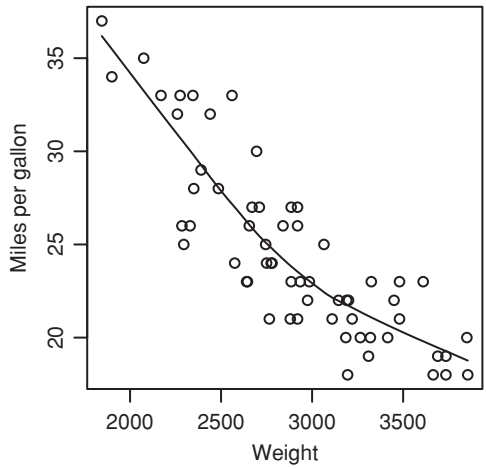


Figure 11.5 `Mileage` versus `Weight`, for cars described in US April 1990 *Consumer Reports*. A loess curve is overlaid.

For classification trees, several different splitting criteria may be used, with different software programs offering different selections of criteria. In *rpart*, the default is `gini`, which uses a modified version of the Gini index

$$\sum_{j \neq k} p_{ij} p_{ik} = 1 - \sum_{k} p_{ik}^2$$

as its default measure of "error", or "impurity". An alternative is `information`, or deviance. The *rpart* documentation and output use the generic term *error* for whatever criterion is used.

The `information` criterion, or deviance, is

$$D_i = \sum_{\text{classes } k} n_{ik} \log(p_{ik}).$$

This differs only by a constant from the entropy measure that is used elsewhere, and thus would give the same tree if the same stopping rule were used. For the two-class problem (a binary classification), the Gini index and the deviance will almost always choose the same split as the deviance or entropy.

The splitting rule, if specified, is set by specifying, e.g., `parms=list(split=gini)` or `parms=list(split=information)`.

11.3.4 Tree-based regression versus loess regression smoothing

The scatterplot of gas mileage versus vehicle weight in Figure 11.5 suggests a nonlinear relationship. Useful insights may be gained from the comparison of predictions from tree-based regression with predictions from the more conventional and (for these data) more appropriate use of a `loess()` or similar regression smoothing approach.

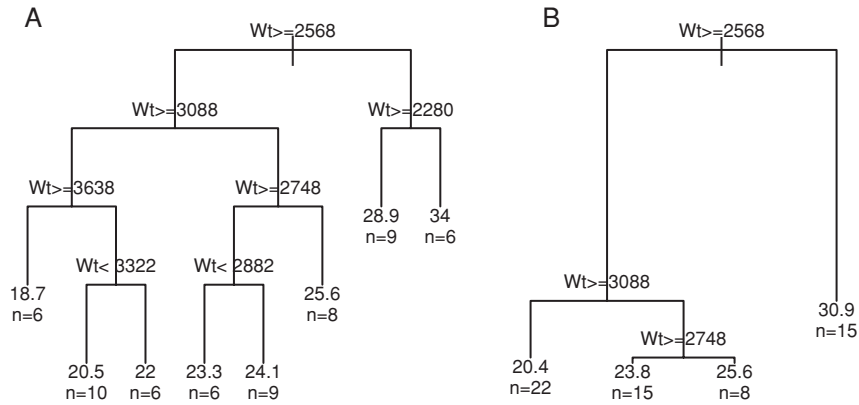


Figure 11.6 Tree-based model for predicting `Mileage` given `Weight`, for cars described in US April 1990 *Consumer Reports*. In panel A, split criteria have for illustrative purposes been changed from the *rpart* defaults, to increase the number of splits. This plot has used uniform vertical spacing between levels of the tree. Panel B used the *rpart* default split criteria, and vertical spacing was set to reflect the change in residual sum of squares. In panel A, such non-uniform spacing would have given splits that were bunched up at the lower nodes.

The code for Figure 11.5 is:

```
## loess fit to Mileage vs Weight: data frame car.test.frame (rpart)
with(car.test.frame, scatter.smooth(Mileage ~ Weight))
```

To fit a regression tree to the car mileage data shown in Figure 11.5, the model formula is `Mileage ~ Weight`, i.e., predict `Mileage` given `Weight`, just as for the use of `lm()` or `loess()`. The code is:

```
car.tree <- rpart(Mileage ~ Weight, data=car.test.frame,
                  control = list(minsplit = 10, minbucket = 5,
                  cp = 0.0001), method="anova")
plot(car.tree, uniform = TRUE)
text(car.tree, digits = 3, use.n = TRUE)
```

Setting `minsplit=10` (the default is 20) allows splitting at any node that has at least ten observations, while `minbucket=5` has reduced to 5 the minimum number of observations at a terminal node. See `help(rpart.control)` for further details.

Figure 11.6A shows the fitted regression decision tree. Table 11.3 compares the predictions of Figure 11.6 with predicted values from the use of `loess` in Figure 11.5. Notice how the tree-based regression has given several anomalous predictions. Later splits have relied on information that is too local to improve predictive ability.

Figure 11.6B is the plot that results when the split criteria are left at their defaults. It used the simpler code:

```
car.tree <- rpart(Mileage ~ Weight, data = car.test.frame)
plot(car.tree, uniform = FALSE)
text(car.tree, digits = 3, use.n = TRUE)
```

Prediction is much coarser. Table 11.4 compares the predictions for this less ambitious tree with the predictions from the loess regression. Beyond a certain point, adding additional leaves reduces genuine predictive power, even though the fit to the data used to develop the predictive model must continue to improve.

Table 11.3 *Predictions from the regression tree of Figure 11.6. For comparison, we give the range of predictions from the loess curve that we fitted in Figure 11.5.*

Range of `Weight`	Predicted `Mileage`	Range of predictions (Figure 11.5)
1845.0 – 2280	34	36.4 – > 30.3
> 2280.0 – 2567.5	28.9	30.3 – > 26.9
> 2567.5 – 2747.5	25.6	26.9 – > 24.9
> 2747.5 – 2882.5	23.3	24.9 – > 23.9
> 2882.5 – 3087.5	24.1	23.9 – > 22.1
> 3087.5 – 3322.5	20.5	22.1 – > 20.7
> 3322.5 – 3637.5	22	20.7 – > 19.7
> 3637.5 – 3855	18.7	19.7 – > 18.9

Table 11.4 *Predictions from the regression tree of Figure 11.6B.*

Range of weights	Predicted mileage	Range of loess predictions (Figure 11.5)
– 2567.5	30.9	36.4 – > 26.9
> 2567.5 – 2747.5	25.6	26.9 – > 24.9
> 2747.5 – 3087.5	23.8	24.9 – > 22.1
> 3087.5	20.4	22.1 – > 18.9

11.4 Predictive accuracy and the cost–complexity trade-off

Realistic estimates of the predictive power of a model must take account of the extent to which the model has been selected from a wide range of candidate models. Two main approaches have been used for determining unbiased assessments of predictive accuracy. These are cross-validation, and the use of training and test sets.

The cross-validation error is relevant to predictions for the population from which the data were sampled. Testing performance under conditions of actual use, which may be different from those that generated the initial data, may require a validation set that is separate from any of the data used to generate the model. See also the further comments, under *Models with a complex error structure*, in Section 11.8.

11.4.1 Cross-validation

As explained in the earlier discussion in Subsection 5.4.1, cross-validation requires the splitting of the data into k subsets, where in *rpart* the default choice is $k = 10$. Each of the k subsets of the data is left out in turn, the model is fitted to the remaining data, and the results used to predict the outcome for the subset that has been left out. There is one such division of the data, known as a *fold*, for each of the k subsets. At the kth fold the kth subset has the role of test data, with the remaining data having the role of training data. The data that are for the time being used as the test data might alternatively be called the

"out-of-bag" data, a name that is more directly appropriate in the context of the bootstrap aggregation approach that will be discussed in Section 11.7.

In a regression model, prediction error is usually taken as the sum of differences between observed and predicted, i.e., the criterion is the same as that used for the splitting rule. In a classification model, prediction error is usually determined by counting 1 for a misclassification and 0 for a correct classification. The crucial feature of cross-validation is that each prediction is independent of the data to which it is applied. As a consequence, cross-validation gives an unbiased estimate of predictive power, albeit for a model that uses, on average, a fraction $(k-1)/k$ of the data. An estimate of average "error" is found by summing up the measure of "error" over all observations and dividing by the number of observations. Once predictions are available in this way for each of the subsets, the average error is taken as (total error)/(total number of observations).

Cross-validation is built into the *rpart* calculations, giving an assessment of the change in prediction error with changing tree size. The usual approach is to build a tree that has more splits than is optimal, i.e., it is over-fitted, and then to prune back to a tree that has close to the minimum cross-validated prediction error.

11.4.2 The cost–complexity parameter

Rather than controlling the number of splits directly, this is controlled indirectly, via a quantity c_p (complexity parameter, `cp` in the `rpart()` output) that puts a cost on each additional split. The increase in "cost" as the tree becomes more complex is traded off against the reduction in the lack-of-fit criterion, so that splitting ceases when the increase in cost outweighs reduction in lack-of-fit. A large value for c_p leads to a small tree (additional cost quickly offsets decrease in lack-of-fit), while a small value leads to a complex tree. The choice of c_p is thus a proxy for the number of splits.

Previous experience in use of tree-based methods with similar problems may suggest a suitable setting for `cp` for an initial run of the calculations. Otherwise, the `rpart` default can be used. If the cross-validation error has not obviously reached a minimum, this will indicate that the tree is too small, and calculations need to be rerun with a smaller `cp`.

Having fitted a tree that is more complex than is optimal, we then plot the cross-validated relative error against c_p (or equivalently, against number of splits), and determine the value of c_p for which the tree seems optimal.

It can be shown that there is a sequence of prunings from the constructed tree back to the root, such that

- at each pruning the complexity reduces,
- each tree has the smallest number of nodes possible for that complexity, given the previous tree.

See for example Ripley (1996). This is true even if the lack-of-fit measure used for pruning is different from that used in the formation of the tree. For classification trees, it is common to use fraction or per cent misclassified when trees are pruned.

For the final choice of tree, an alternative to use of the cross-validation estimate of error is to examine the error rate for a new set of data.

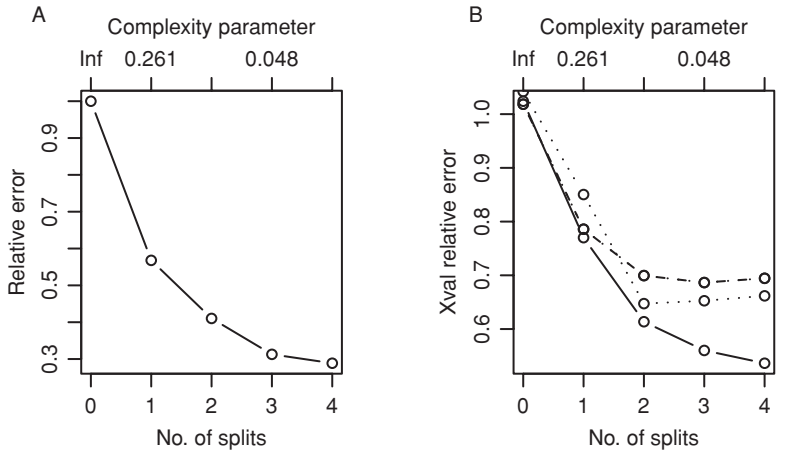


Figure 11.7 While the relative error (panel A) must inevitably decrease as the number of splits increases, the cross-validation relative error (X-val relative error, shown in panel B) is likely, once the number of splits is large enough, to increase. Results from three cross-validation runs are shown. Plots are for the car mileage data of Figure 11.5 and Table 11.4.

In summary:

- The parameter `cp` is a proxy for the number of splits. For the initial `rpart` model, it should be set small enough that the cross-validation error rate achieves a minimum.
- Having identified the optimal tree (with the minimum cross-validation error rate), later splits are then pruned off.

11.4.3 Prediction error versus tree size

In Figure 11.7, we return to the car mileage data of Figure 11.5 and Table 11.4. Figures 11.7A and B plot, against tree size, different assessments of the relative error. As we have a continuous outcome variable (`Mileage`), the relative error is the sum of squares of residual divided by the total sum of squares.

Figure 11.7A plots the resubstitution assessment of relative error, which shows the performance of the tree-based prediction on the data used to form the tree. Figure 11.7B shows estimates from three cross-validation runs, with the data split into $k = 10$ subsets (or *folds*) at each run. The plot gives an indication of the variability that can be expected, for these data, from one cross-validation run to another.

The optimal tree, in Figure 11.7, is the smallest tree that has near minimum cross-validated relative error. In Figure 11.7, the three runs suggest different choices for the optimal tree size. The defaults for the arguments `minsplit` and `minbucket` (see `help(rpart.control)`) have limited the number of splits to 4. Between 2 and 4 splits may be optimal. Note the importance of doing several cross-validation runs.

11.5 Data for female heart attack patients

This and the next section will proceed to look in detail at trees from substantial data sets. This section will examine data on the outcome (`live` or `dead`) for female heart attack

patients, in the data frame `mifem` (*DAAG*). The data are for the mortality of 1295 female heart attack patients. A summary of the data follows.

```
> summary(mifem)     # data frame mifem (DAAG)

 outcome         age           yronset     premi    smstat
 live:974   Min.   :35.0   Min.   :85.0   y :311   c :390
 dead:321   1st Qu.:57.0   1st Qu.:87.0   n :928   x :280
            Median :63.0   Median :89.0   nk: 56   n :522
            Mean   :60.9   Mean   :88.8            nk:103
            3rd Qu.:66.0   3rd Qu.:91.0
            Max.   :69.0   Max.   :93.0
 diabetes highbp   hichol   angina   stroke
 y :248   y :813   y :452   y :472   y : 153
 n :978   n :406   n :655   n :724   n :1063
 nk: 69   nk: 76   nk:188   nk: 99   nk:  79

Notes:

premi = previous myocardial infarction event
For smstat, c = current  x = ex-smoker  n = non-smoker
    nk = not known
```

(Technically, these are patients who have suffered a *myocardial infarction*. Data are from the Newcastle (Australia) center of the Monica project; see the web site given under `help(monica)`.)

In order to fit the tree, we specify

```
mifem.rpart <- rpart(outcome ~ ., method="class",
                     data = mifem, cp = 0.0025)
```

The dot (`.`) on the right-hand side of the formula has the effect of including all available explanatory variables. A choice of `cp` $= 0.0025$ continues splitting to the point where the cross-validated relative error has started to increase. Figure 11.8 shows the change in cross-validated error rate as a function of tree size, while the same information is shown alongside in printed form. The code is:

```
plotcp(mifem.rpart)  # Cross-validated error vs cp
printcp(mifem.rpart) # Tabular version of the same information
```

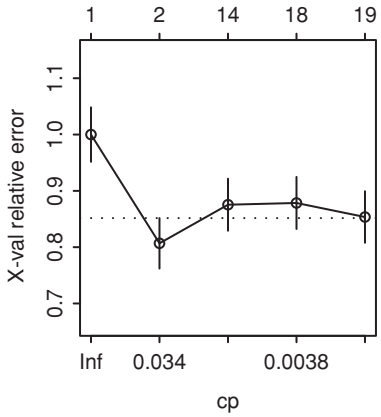
Notice the increase in the cross-validated error rate when there are more than two splits.

For this tree, the optimum is a single split, i.e., two leaves. In order to prune the tree back to this size, specify:

```
mifemb.rpart <- prune(mifem.rpart, cp=0.03)
```

The cross-validated error rate is $0.248 \times 0.832 = 0.21$.

For these data, the optimum tree is very simple indeed, with a single split. Figure 11.9 shows the tree, obtained using:




```
> # Tabular equivalent of graph
> printcp(mifem.rpart)
. . . .
Root node error: 321/1295 = 0.248

n= 1295

        CP nsplit  xerror   xstd
1 0.20249      0   1.000 0.0484
2 0.00561      1   0.832 0.0454
3 0.00467     13   0.844 0.0456
4 0.00312     17   0.838 0.0455
5 0.00250     18   0.850 0.0457
```

Figure 11.8 Cross-validated error versus `cp`, for the female heart attack data. The top side of the bounding box is labeled with the size of tree. The tabular equivalent of the graph is shown alongside. The column that has the (potentially misleading) relative errors has been omitted.

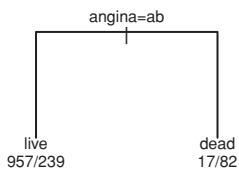


Figure 11.9 Decision tree of size 2, for the data on female myocardial infarction patients.

```
plot(mifemb.rpart)
par(xpd=TRUE)         # May be needed so that labels appear
text(mifemb.rpart, use.n=T, digits=3)
par(xpd=FALSE)
```

Note that the levels of `angina` are `y` (labelled a in the graph), `n` (labeled b in the graph), and `nk`. It is the patients whose angina status was unknown that were most likely to be dead, perhaps because it was not possible to ask them for the information. Use of `use.n` has had the effect of labeling each node with information on outcomes for patients that the decision tree has assigned to that node. For the left node (labeled `live`), 957 patients survive, while 239 die. For the right node, 17 survive, while 82 die.

11.5.1 The one-standard-deviation rule

The function `rpart()` calculates, for each tree, both the cross-validated estimate of "error" and a standard deviation for that error. Where the interest is in which splits are likely to be meaningful, users are advised to choose the smallest tree whose error is less than

$$\text{minimum error} + 1 \text{ standard deviation}.$$

Figure 11.8 has a dotted horizontal line, at a height of `xerror` $= 0.877$ ($= 0.832 + 0.054$), that shows where this error level is attained. Here the one standard error rule leads to the same choice of tree size as the choice of the absolute minimum. The rule is in general

conservative if the interest is in choosing the optimal predictive tree, i.e., the predictive power will on average be slightly less than optimal.

11.5.2 Printed information on each split

We will now examine output that is available from printing the *rpart* object for the very simple tree that is shown in Figure 11.9:

```
> print(mifemb.rpart)
n= 1295

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 1295 321 live ( 0.75 0.25 )
  2) angina:y, n 1196 239 live ( 0.80 0.20 ) *
  3) angina:nk 99  17 dead ( 0.17 0.83 ) *
```

Predictions are:

- At the first split (the *root*) the prediction is `live` (probability 0.752), with the 321 who are `dead` misclassified. Here the `loss` (number misclassified) is 321.
- Take the left branch from node 1 if the person's angina status is `y` or `n`, i.e., if it is known (1196 persons). The prediction is `live` (probability 0.800), with the 239 who die misclassified.
- Take the right branch from node 1 if the angina status is unknown (99 persons). The prediction is `dead` (probability 0.828), with the 17 who are `live` misclassified.

The function `summary.rpart()` gives information on alternative splits.

11.6 Detecting email spam – the optimal tree

In Figure 11.2, where `cp` had its default value of 0.01, splitting did not continue long enough for the cross-validated relative error to reach a minimum. In order to find the minimum, we now repeat the calculation, this time with `cp` = 0.001.

```
spam7a.rpart <- rpart(formula = yesno ~ crl.tot + dollar +
                      bang + money + n000 + make,
                      method="class", data = spam7, cp = 0.001)
```

The choice of `cp` = 0.001 in place of `cp` = 0.01 was, for this initial calculation, a guess. It turns out to carry the splitting just far enough that the cross-validated relative error starts to increase.

In this instance, the output from `plotcp(spam7a.rpart)` is not very useful. The changes are so small that it is hard to determine, from the graph, where the minimum falls. Instead, the printed output will be used:

```
> printcp(spam7a.rpart)
. . . .
Root node error: 1813/4601 = 0.394
```

```
n= 4601

        CP nsplit rel error xerror   xstd
1  0.47656      0     1.000  1.000 0.0183
2  0.07557      1     0.523  0.559 0.0155
3  0.01158      3     0.372  0.386 0.0134
4  0.01048      4     0.361  0.384 0.0134
5  0.00634      5     0.350  0.368 0.0132
6  0.00552     10     0.317  0.352 0.0129
7  0.00441     11     0.311  0.350 0.0129
8  0.00386     12     0.307  0.335 0.0127
9  0.00276     16     0.291  0.330 0.0126
10 0.00221     17     0.288  0.326 0.0125
11 0.00193     18     0.286  0.331 0.0126
12 0.00165     20     0.282  0.330 0.0126
13 0.00100     25     0.274  0.329 0.0126
```

Choice of the tree that minimizes the cross-validated error leads to `nsplit`=17 with `xerror`=0.33. Again, note that different runs of the cross-validation routine will give slightly different results.

Use of the one-standard-deviation rule suggests taking `nsplit`=16. (From the table, minimum + standard error = 0.330 + 0.013 = 0.343. The smallest tree whose `xerror` is less than or equal to this has `nsplit`=16.) Figure 11.10 plots this tree.[3]

The absolute error rate is estimated as $0.336 \times 0.394 = 0.132$ if the one-standard-error rule is used, or $0.330 \times 0.394 = 0.130$ if the tree is chosen that gives the minimum error.

How does the one-standard-error rule affect accuracy estimates?

The function `compareTreecalcs()` (*DAAG*) can be used to assess how accuracies are affected, on average, by use of the one-standard-error rule. See `help(compareTreecalcs)` for further information. An example of its use is:

```
acctree.mat <- matrix(0, nrow=100, ncol=6)
for(i in 1:100)acctree.mat[i,] <- compareTreecalcs(data=spam7,
                                                   fun="rpart")
```

For each of 100 random splits of the data into two nearly equal subsets I and II, the following accuracies are calculated:

1. The estimated cross-validation error rate when `rpart()` is run on the training data (I), and the one-standard-error rule is used;
2. The estimated cross-validation error rate when `rpart()` is run on subset I, and the model used that gives the minimum cross-validated error rate;

[3]
```
## Use prune.rpart() with cp = 0.03 (0.00276 < 0.03 < 0.00386),
## to prune back to nsplit=16.
spam7b.rpart <- prune(spam7a.rpart, cp=0.003)
plot(spam7b.rpart, uniform=TRUE)
text(spam7b.rpart, cex=0.75)
```

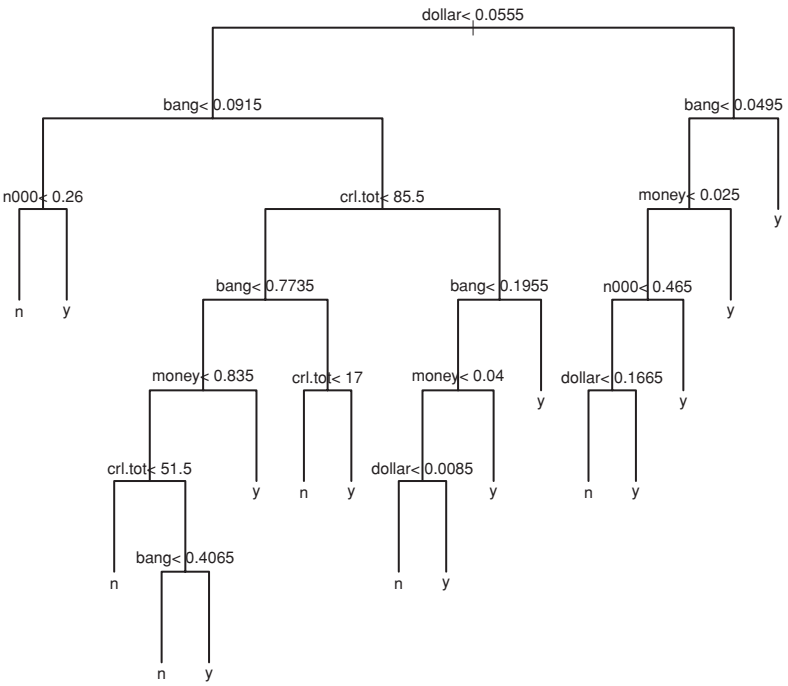


Figure 11.10 Decision tree for a tree size of 16.

3. The error rate when the model fitted in item 1 is used to make predictions for subset II;
4. The error rate when the model fitted in item 2 is used to make predictions for subset II.

In a run of the code given above (results will of course differ from run to run), we obtained the following results. The prediction accuracy for subset II was indeed similar to the cross-validated accuracy when the model was fit to subset I. There was, on average, a small loss of accuracy (0.45%, SE = 0.07%) from use of the one-standard-error rule. Details are in the "Additional Notes" document that can be downloaded from the web page for this book.

How is the standard error calculated?

For classification trees, using number misclassified, the standard deviation is, assuming independent errors, approximately equal to the square root of the number misclassified.

When are tree-based methods appropriate?

Exploration of a new data set with tree-based regression or classification can sometimes yield a quick handle on which variables have large effects on the outcome variable. Such methods may be appropriate when data sets are very large, and very limited assumptions are needed to make useful inferences.

However, there is a risk of missing interesting structure because assumptions have been too weak. Parametric models have a better chance of capturing such structure; they are

almost always more appropriate for small data sets. There is also the issue of optimality; while the local splits are optimal, the overall tree may be far from optimal.

11.7 The *randomForest* package

The function `randomForest()` in the *randomForest* package is an attractive alternative to `rpart()` that, for relatively complex trees, often gives an improved predictive accuracy. For each of a large number of bootstrap samples (by default, 500) trees are independently grown. In addition, a new random sample of variables is chosen for use with each new tree. The "out-of-bag" (OOB) prediction for each observation is determined by a simple majority vote across trees whose bootstrap sample did not include that observation.

Trees are grown to their maximum extent, limited however by `nodesize` (minimum number of trees at a node). Additionally, `maxnodes` can be used to limit the number of nodes. There is no equivalent to the parameter `cp`. The main tuning parameter is the number `mtry` of variables that are randomly sampled at each split. The default is the square root of the total number of variables; this is often satisfactory.

It may seem surprising that it is (usually) beneficial to take a random sample of variables. Essentially, `mtry` controls the trade-off between the amount of information in each individual tree, and the correlation between trees. A very high correlation limits the ability of an individual tree to convey information that is specific to that tree.

The following uses `randomForest()` with the data frame `spam7`:

```
library(randomForest)
spam7.rf <- randomForest(yesno ~ ., data=spam7, importance=TRUE)
```

The output summary (though obtained using `print()`) is:

```
> print(spam7.rf)

Call:
 randomForest(x = yesno ~ ., data = spam7, importance = T)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 11.95%
Confusion matrix:
     n    y class.error
n 2588  200  0.07173601
y  350 1463  0.19305019
```

Use of the OOB error rate is crucial for calculating realistic error rates, unless separate test data are available. Notice that the error rate that was achieved here is slightly better than the 13.0% error rate achieved by *rpart()*.

There is limited scope for tuning, based on performance of "out-of-bag" data. The most important parameter is `mtry`, which controls the number of variables that are sampled for use at each split. Finding and using the optimal value of `mtry` is acceptable because its

only effect is on the way that models are sampled from the class of models that are in contention. The function `tuneRF()` makes it straightforward to find the optimum, thus:

```
> tuneRF(x=spam7[, -7], y=spam7$yesno, trace=FALSE)
-0.08633 0.05
-0.005396 0.05
  mtry OOBError
1    1   0.1313
2    2   0.1208
4    4   0.1215
```

The value of `OOBerror` is smallest for `mtry=2`. The result will vary somewhat from one run of `tuneRF()` to another. For 6 variables, `mtry=2` is in fact the default, obtained by choosing the largest integer whose square is less than or equal to 6.

Two "importance" measures are calculated for each variable. The first is a "leave-one-out" type assessment of its contribution to prediction accuracy, although it is calculated by the average decrease in prediction accuracies in the "out-of-bag" portions of the data from permuting values of the variable. The second is the average decrease in lack-of-fit from splitting on the variable, averaged over all trees. Specifying `importance=TRUE` causes the calculation of both measures of importance, whereas the default is to calculate only the second measure. For the present data, the result values are:

```
> importance(spam7.rf)
            n     y MeanDecreaseAccuracy MeanDecreaseGini
crl.tot 0.631 0.807                0.519            290.1
dollar  0.660 0.798                0.523            411.4
bang    0.683 0.848                0.536            647.1
money   0.562 0.782                0.492            182.2
n000    0.649 0.465                0.510             96.9
make    0.339 0.609                0.418             44.8
```

Note that the first measure gives roughly equal importance to the first three and fifth variables, while the second measure rates `bang` as easily the most "important". Is a variable important? It depends on what is meant by importance.

Comparison between `rpart()` *and* `randomForest()`

In the following, the `spam7` will be repeatedly split at random into two nearly equal subsets: subset I has 2300 observations, and subset II has 2301 observations. Models will be fit (1) using `rpart()` and (2) using `randomForest()`. The calculations are again handled using the function `compareTreecalcs()`.[4]

[4]
```
## Accuracy comparisons
acctree.mat <- matrix(0, nrow=100, ncol=8)
colnames(acctree.mat) <- c("rpSEcvI", "rpcvI", "rpSEtest", "rptest",
                           "n.SErule", "nre.min.12", "rfcvI", "rftest")
for(i in 1:100)acctree.mat[i,] <-
   compareTreecalcs(data=spam7, fun=c("rpart", "randomForest"))
acctree.df <- data.frame(acctree.mat)
lims <- range(acctree.mat[, c(4,7,8)], na.rm=TRUE)
plot(rfcvI ~ rftest, data=acctree.df); abline(0,1)    # Panel A
plot(rptest ~ rftest, data=acctree.df); abline(0,1)   # Panel B
```

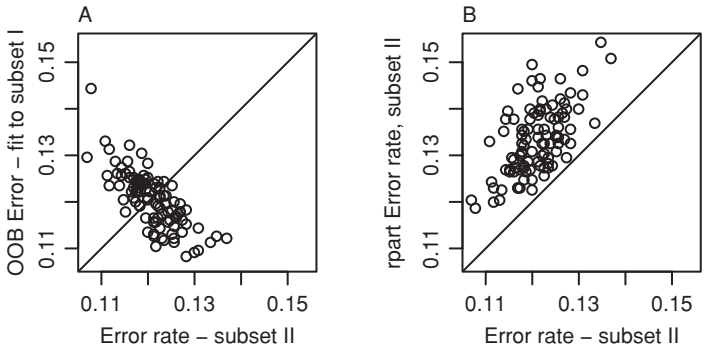


Figure 11.11 For each of 100 random splits of `spam7` into two nearly equal subsets I and II, random forest models were fitted to subset I, keeping a record both of the OOB error rate on subset I and the error rate on predictions for subset II. Panel A compares the two error rates. Panel B compares the error rate from use of `rpart()` with the second of the error rates for the random forest model.

Figure 11.11A is designed to check that `randomForest()` did not over-fit. Figure 11.11B shows that the model fitted using `randomForest()` has given an average reduction in the error rate, relative to `rpart()`, of 0.9% [SE = 0.1%].

The difference between `rpart()` and `randomForest()` is much larger if the full spam database, with 57 explanatory variables, is used. In that case, in three runs, `rpart()` gave error rates of 8.45%, 8.34%, and 8.24%, whereas `randomForest()` gave error rates of 4.56%, 4.41%, and 4.54%.

Efficient computation

Calculations that use `randomForest()` may take a long time when there are extensive data. The following times are for the full spam database, which has dimension 4601×58, on a 2.16GHz MacBook Pro with 2GB of memory. Elapsed times with the full spam database were:

`rpart()` with `cp=0.00025`: 6.4 sec (here there is no alternative to use of a model formula, typically with variables taken from the columns of a data frame)
`randomForest()`: 27.3 sec when a model formula and data frame were specified, as against 27.0 sec when the arguments were a matrix `x` whose columns were the explanatory variables and a response `y`.

Differences between `rpart()` *and* `randomForest()`

Some of the more important differences are:

- `randomForest()` does not give a unique tree.
- Use of `randomForest()` is largely automatic, with limited need or opportunity for tuning.
- For `randomForest()`, there is no direct equivalent of `rpart`'s `cp` parameter, and no possibility or necessity to prune trees.

- `rpart()` requires a model formula, whereas `randomForest()` allows, alternatively, specification of a matrix whose columns are used as predictors.
- `randomForest()` does not have a direct equivalent of `rpart`'s `method` argument, that can be used to distinguish between regression, classification, and other models. Instead, `randomForest` assumes a classification model if the response is a factor, and otherwise assumes a regression model.
- Accuracy is, for some data sets, markedly better for `randomForest()` than for `rpart()`.

The proportion of trees in which both members of a pair of observations appear in the same terminal node gives a measure of proximity, or nearness. Methods that will be described in Subsection 12.1.3 can then be used to derive a low-dimensional representation of the points. See Figure 13.5, and the accompanying discussion, for an example.

11.8 Additional notes on tree-based methods

The combining of tree-based methods with other approaches

Tree-based approaches can sometimes be used in tandem with parametric approaches, in ways that combine the strengths of the different approaches. Thus, tree-based regression may suggest interaction terms that ought to appear in a parametric model. We might also apply tree-based regression analysis to residuals from conventional parametric modeling, in order to check whether there is residual structure that the parametric model has not captured. Another variation is to apply tree-based regression to fitted values of a parametric model, in order to cast predictions in the form of a decision tree.

Models with a complex error structure

Tree-based models, as implemented in current software, are less than ideal for use with data where there is corrrelation in time or space (as in Chapters 9 and 10), or where there is more than one level of variation (as in Chapter 10). If tree-based models are nevertheless used with such data, accuracy should be assessed at the level of variation that is relevant for any use of model predictions. If there is clustering in the data, and the size of the clusters varies widely, predictions may be strongly biased.

Pruning as variable selection

Venables and Ripley (2002) suggest that pruning might be regarded as a method of variable selection. This suggests using an AIC-like criterion as the criterion for optimality. We refer the reader to their discussion for details.

Other types of tree

The *rpart* package allows two other types of tree. The `poisson` splitting method adapts *rpart* models to event rate data. See Therneau and Atkinson (1997, pp. 35–41). The `survival` splitting method is intended for use with survival data, where each subject has either 0 or 1 events. (The underlying model is a special case of the Poisson event rate model.) See again Therneau and Atkinson (1997, pp. 41–46).

Summary of pluses and minuses of tree-based methods

Here is a more detailed comparison that matches tree-based methods against the use of linear models, generalized additive models, and parametric discriminant methods. Strengths of tree-based regression include:

- Results are invariant to a monotone re-expression of explanatory variables.
- The methodology is readily adapted to handle missing values, without omission of complete observations.
- Tree-based methods are adept at capturing non-additive behavior. Interactions are automatically included.
- It handles regression, and in addition unordered and ordered classification.
- Results are in an immediately useful form for classification or diagnosis.

Weaknesses of methods that yield a single tree include:

- The overall tree may not be optimal.
- Large trees make poor intuitive sense; their predictions must be used as black boxes.
- Continuous predictor variables are treated, inefficiently, as discrete categories.
- Assumptions of monotonicity or continuity across category boundaries are lost.
- Low-order interaction effects do not take precedence over higher-order interactions, which may be an issue for ease of interpretation.
- Limited notions of what to look for may result in failure to find useful structure.
- It may obscure insights that are obvious from parametric modeling, e.g., a steadily increasing risk of cancer with increasing exposure to a carcinogen.

Ensemble methods, including random forests, yield more optimal trees. They are rather more open than single trees to the complaint that their predictions must be treated as black boxes. Within the limitations imposed by strongly non-parametric assumptions, they often do better than individual trees. They are equally affected by the final three weaknesses.

11.9 Further reading and extensions

Venables and Ripley (2002) is a good brief overview. Ripley (1996), Berk (2008), Hastie *et al.* (2009) are more comprehensive. Berk (2008) has extensive insightful comments on the properties, practical use, and usefulness of these and related technologies.

Random forests builds on the *bagging* methodology, where multiple bootstrap samples are taken. Boosting is a rather different extension of tree-based methodology. In the boosting methodology, as applied to trees, successive trees give extra weight to points that were incorrectly predicted by earlier trees, with a weighted vote finally taken to get a consensus prediction. Unlike random forests, the statistical properties of boosting methods are not well understood, and they can over-fit. The function `adaboost.M1()` in the *adabag* package implements the Adaboost.M1 algorithm. Note also the *gbm* Generalized Boosted Regression Models package.

Therneau and Atkinson (1997, pp. 50–52) give comparisons between *rpart* methodology and other software. Lim and Loh (2000) investigated 10 decision tree implementations,

not however including *rpart*. Several of these implementations offer a choice of algorithms.

For a biostatistical perspective, including a discussion of survival modeling and analysis of longitudinal data, see Zhang and Singer (1999).

References for further reading

Berk, R. A. 2008. *Statistical Learning from a Regression Perspective.*

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd edn.

Lim, T.-S. and Loh, W.-Y. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning* 40: 203–28.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks.*

Therneau, T. M. and Atkinson, E. J. 1997. *An Introduction to Recursive Partitioning Using the RPART Routines.*

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

Zhang, H. and Singer, B. 1999. *Recursive Partitioning in the Health Sciences.*

11.10 Exercises

1. Refer to the `head.injury` data frame.
   (a) Use the default setting in `rpart()` to obtain a tree-based model for predicting occurrence of clinically important brain injury, given the other variables.
   (b) How many splits gives the minimum cross-validation error?
   (c) Prune the tree using the one-standard-error rule.

2. The data set `mifem` is part of the larger data set in the data frame `monica` that we have included in our *DAAG* package. Use tree-based regression to predict mortality in this larger data set. What is the most immediately striking feature of the analysis output? Should this be a surprise?

3. Use tree-based regression to predict `re78` in the data frame `nswpsid1` that is in the *DAAG* package. Compare the predictions with the multiple regression predictions in Chapter 6.

4. Copy down the email spam data set from the web site given at the start of Section 11.2. Carry out a tree-based regression using all 57 available explanatory variables. Determine the change in the cross-validation estimate of predictive accuracy.

5. This exercise will compare alternative measures of accuracy from `randomForest()` runs. First, 16 rows where data (on `V6`) is missing will be omitted:

```
> library(MASS)
sapply(biopsy, function(x)sum(is.na(x)))
biops <- na.omit(biopsy[,-1])  ## Column 1 is ID
## Examine list element names in randomForest object
names(randomForest(class ~ ., data=biops))
library(MASS)
```

```
> sapply(biopsy, function(x)sum(is.na(x)))
   ID    V1    V2    V3    V4    V5    V6    V7    V8    V9 class
    0     0     0     0     0     0    16     0     0     0     0
> biops <- na.omit(biopsy[,-1])  ## Column 1 is ID
> ## Examine list element names in randomForest object
> names(randomForest(class ~ ., data=biops))
 [1] "call"            "type"            "predicted"       "err.rate"
 [5] "confusion"       "votes"           "oob.times"       "classes"
 [9] "importance"      "importanceSD"    "localImportance" "proximity"
[13] "ntree"           "mtry"            "forest"          "y"
[17] "test"            "inbag"           "terms"
```

(a) Compare repeated `randomForest()` runs:

```
## Repeated runs, note variation in OOB accuracy.
for(i in 1:10) {
  biops.rf <- randomForest(class ~ ., data=biops)
  OOBerr <- mean(biops.rf$err.rate[,"OOB"])
  print(paste(i, ": ", round(OOBerr, 4), sep=""))
  print(round(biops.rf$confusion,4))
}
```

(b) Compare OOB accuracies with test set accuracies:

```
## Repeated train/test splits: OOB accuracy vs test set accuracy.
for(i in 1:10){
  trRows <- sample(1:dim(biops)[1], size=round(dim(biops)[1]/2))
  biops.rf <- randomForest(class ~ ., data=biops[trRows, ],
                           xtest=biops[-trRows,-10],
                           ytest=biops[-trRows,10])
  oobErr <- mean(biops.rf$err.rate[,"OOB"])
  testErr <- mean(biops.rf$test$err.rate[,"Test"])
  print(round(c(oobErr,testErr),4))
}
```

Plot test set accuracies against OOB accuracies. Add the line $y = x$ to the plot. Is there any consistent difference in the accuracies? Given a random training/test split, is there any reason to expect a consistent difference between OOB accuracy and test accuracy?

(c) Calculate the error rate for the training data:

```
randomForest(class ~ ., data=biops, xtest=biops[,-10],
             ytest=biops[,10])
```

Explain why use of the training data for testing leads to an error rate that is zero.

6. Starting in turn with samples of sizes $n = 25, 50, 100, 200, 400, 800$, take a bootstrap sample of the relevant size n. In each case, determine the number of observations that are repeated.

7. The expected number of repeats is $(1 - n^{-1})^n$, with limiting value $\exp(-1)$ for infinite n.[5] Plot the observed number of repeats against the expected number of repeats.

8. Plot, against $\log(n)$, the difference $\exp(-1) - (1 - n^{-1})^n$.

[5] The probability that any particular point (observation) will be excluded is $(1 - n^{-1})^n$, where n is the sample size. Summing over all points, the expected number excluded is $n(1 - n^{-1})^n$, i.e., a fraction $(1 - n^{-1})^n$ of the number of points n.

9. Apply `randomForest()` to the data set `Pima.tr` (*MASS* package), then comparing the error rate with that from use of `randomForest()` with a bootstrap sample of rows of `Pima.tr`:

```
## Run model on total data
randomForest(as.factor(type) ~ ., data=Pima.tr)

rowsamp <- sample(dim(Pima.tr)[1], replace=TRUE)
randomForest(as.factor(type) ~ ., data=Pima.tr[rowsamp, ])
```

Compare the two error rates. Why is the error rate from the fit to the bootstrap sample of rows so (spuriously) low?

12

Multivariate data exploration and discrimination

Earlier chapters have made extensive use of exploratory graphs that have examined variables and their pairwise relationships, as a preliminary to regression modeling. Scatterplot matrices have been an important tool, and will be used in this chapter also. The focus will move from regression to methods that seek insight into the pattern presented by multiple variables. While the methodology has applications in a regression context, this is not a primary focus.

There are a number of methods that project the data on to a low-dimensional space, commonly two dimensions, suggesting "views" of the data that may be insightful. In the absence of other sources of guidance, it is reasonable to begin with views that have been thus suggested. One of the most widely used methods for projecting onto a low-dimensional space is principal components analysis (PCA).

The PCA form of mathematical representation has applications in many contexts beyond those discussed here. As used here, PCA is a special case of a much wider class of multidimensional scaling (MDS) methods. Subsection 12.1.3 is a brief introduction to this wider class of methods.

Principal components analysis replaces the input variables by new derived variables, called *principal components*. The analysis orders the principal components according to the amounts that they contribute to the total of the variances of the original variables. The most insightful plots are often, but by no means inevitably, those that involve the first few principal components.

In the analysis of morphometric data that have not been logarithmically transformed, the first component is often an overall measure of size. This may be of less interest than later principal components, which will often capture differences in relative body dimensions. There is no necessary sense in which the first principal component, explaining the largest proportion of the variance, is the most "important".

As noted, the methods have application in regression and related analyses. A large number of candidate explanatory variables may be replaced by the first few principal components, hoping that they will adequately summarize the information in the candidate explanatory variables. If we are fortunate, simple modifications of the components will give new variables that are readily interpretable.

Discriminant analysis methods are another major focus of this chapter. In discrimination, observations belong to one of several classes or groups. The aim is to find a rule, based on values of explanatory variables, that will, as far as possible, assign observations to their correct classes. This rule may then be used to classify new observations whose class

may be unknown or unclear. An obvious way to evaluate classification rules is to examine their *predictive accuracy*, i.e., the accuracy with which they can be expected to assign new observations.

Discriminant analysis methodology is clearly important in its own right. In addition it may be used, somewhat like PCA, as a data reduction technique. A small number of "discriminant components", i.e., linear combinations of the candidate variables that for discrimination purposes sum up the information in those variables, may replace explanatory variables. Plots of these components can be useful for data exploration. In some applications, the components may become predictors in a regression analysis.

12.1 Multivariate exploratory data analysis

Preliminary exploration of the data is as important for multivariate analysis as for classical regression. Faced with a new set of data, what helpful forms of preliminary graphical exploration are available? Here we illustrate a few of the possibilities.

The data set `possum` was described in Chapter 2. The interest is in finding the morphometric characteristics, if any, that distinguish possums at the different sites. For simplicity, we will limit attention to a subset of the morphometric variables.

12.1.1 Scatterplot matrices

It is good practice to begin by examining relevant scatterplot matrices. This may draw attention to gross errors in the data. A plot in which the sites and/or the sexes are identified will draw attention to any very strong structure in the data. For example, one site may be quite different from the others, for some or all of the variables.

In order to provide a plot in which the features of interest are clear, the scatterplot matrix and cloud plot in Figure 12.1 show three only of the nine variables. The choice of this particular set of three variables anticipates results of a later analysis, designed to identify the variables that discriminate the populations most effectively. Observe that there are two clusters of values, with each of six sites pretty much restricted to one or other of the clusters. We invite readers to create for themselves a scatterplot matrix that has all nine variables. Code for Figure 12.1A is:

```
## Scatterplot matrix, columns 9-11 of possum (DAAG).
## Colors distinguish sexes; symbols distinguish sites
library(lattice)
pchr <- c(3,4,0,8,2,10,1)
colr <- c("red", "blue")
ss <- expand.grid(site=1:7, sex=1:2)           # Site varies fastest
parset <- with(ss, simpleTheme(pch=pchr[site], col=colr[sex]))
sitenames <- c("Cambarville","Bellbird","Whian Whian", "Byrangery",
               "Conondale ","Allyn River", "Bulburin")
## Add column sexsite to possum; will be used again below
possum$sexsite <- paste(possum$sex, possum$site, sep="-")
splom(possum[, c(9:11)], groups = possum$sexsite,
      col = colr[ss$sex], par.settings=parset,
      varnames=c("tail\nlength","foot\nlength","ear conch\nlength"),
      key = list(text=list(sitenames), points=list(pch=pchr), columns=3))
```

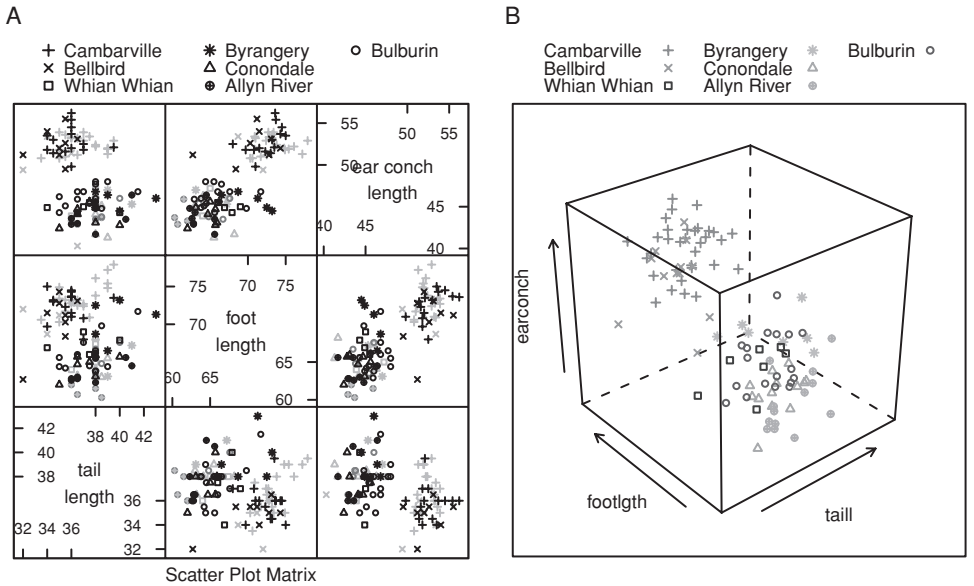


Figure 12.1 Panel A shows the scatterplot matrix for three morphometric measurements on the mountain brushtail possum. In the color version in Plate 4, females are in red (here, gray), males are in blue (here, black). Panel B shows a three-dimensional perspective plot (cloud plot) for the same three variables.

Figure 12.1B gives a three-dimensional representation of the variables shown in Figure 12.1A. The code is:

```
## Cloud plot of earconch, taill and footlgth
cloud(earconch~taill+footlgth, data=possum, pch=pchr, groups=site,
      par.settings=simpleTheme(pch=c(3,4,0,8,2,10,1)),
      auto.key = list(space="top", columns=3, between=1,
                      text=sitenames, between.columns=2))
# auto.key takes its symbols(pch) from par.settings
```

12.1.2 Principal components analysis

As noted in the comments at the beginning of the chapter, the idea is to replace the original variables by a small number of "principal components" – linear combinations of the initial variables, that together may explain most of the variation in the data.

A useful starting point for thinking about principal components analysis is to imagine a two-dimensional scatterplot of data that has, roughly, the shape of an ellipse. Then the first principal component coincides with the longer axis of the ellipse. For example, if we had just the two variables `footlgth` (foot length) and `earconch` (ear conch length) from the `possum` data set, then the first principal component would have the direction of a line from the bottom left to the top right of the scatterplot in the first row and second column of Figure 12.1A. (We might equally well examine the plot in the second row and third column, where the x- and y-axes are interchanged.)

In three dimensions the scatter of data has the shape of an ellipsoidal object, e.g., the shape of a mango or marrow or other similarly shaped fruit or vegetable, but perhaps flattened on one side. The first principal component is the longest axis. If the first principal component accounts for most of the variation, then the object will be long and thin.

Figure 12.1B gave a three-dimensional representation of the variables shown in Figure 12.1A. Notice that the data form into two clusters – the marrow has separated into two parts! One use for principal components and associated plots is to give a visual identification of such clusters.

The scaling of variables is important. If the variables are measured on comparable scales, then unstandardized data may be appropriate. If the scales are not comparable, and especially if the ranges of variables are very different, then standardization may be appropriate. For morphometric (shape and size) data such as in the `possum` data frame, use of the logarithms of variables places all variables on a scale on which equal distances correspond to equal relative changes. Use of the logarithms of measurements is usually for this reason desirable.

Principal components will be formed using the morphometric data in columns 6 to 14 in the `possum` data frame. Interest will be in how principal components differ between sites and sexes. In principle, changes with age should also be investigated. That will not be pursued here, except to compare the distributions of ages between sites.

Preliminary data scrutiny

These morphometric measurements are all essentially linearly related. For these data, taking logarithms makes very little difference to the appearance of the plots. Thus, for simplicity, the present discussion will not use logarithmically transformed variables, instead leaving the analysis of logarithmic transformed data as an exercise.

The reason that taking logarithms makes little difference to the appearance of the plots is that the ratio of largest to smallest value is relatively small, never more than 1.6:

```
## Ratios of largest to smallest values: possum[, 6:14] (DAAG)
sapply(na.omit(possum[, 6:14]), function(x)max(x)/min(x))
 hdlngth   skullw totlngth    taill footlgth earconch      eye
    1.25     1.37     1.29     1.34     1.29     1.36     1.39
   chest   belly
    1.45    1.60
```

In order to make the coefficients in the linear combinations unique, there must be a constraint on their magnitudes. The `princomp()` function makes the squares of the coefficients sum to one. The first principal component is the component that explains the greatest part of the variation. The second principal component is the component that, among linear combinations of the variables that are uncorrelated with the first principal component, explains the greatest part of the remaining variation, and so on.

Different pairs of principal components allow different two-dimensional views of the data. A usual first step is to plot the second principal component against the first principal component. Or, a scatterplot matrix form of presentation can be used to examine all the pairwise plots for the first three or four principal components.

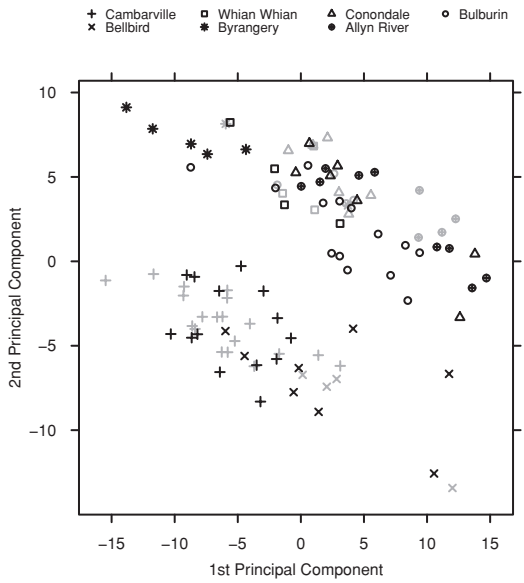


Figure 12.2 Second principal component versus first principal component, for variables in columns 6–14 of the possum data frame. In the color version in Plate 5, females are in red (here, gray), males are in blue (here, black).

The commands used for the analysis were:

```
## Principal components calculations: possum[, 6:14] (DAAG)
possum.prc <- princomp(na.omit(possum[, 6:14]))
```

Notice the use of the na.omit() function to remove rows that contain missing values, prior to the principal components analysis. Note again that, for present tutorial purposes, we have chosen to work with the variables as they stand. Figure 12.2 plots the second principal component against the first, for variables 6 to 14 in the possum data.[1]

Here are further details of the principal components output. We have edited the output so that the "importance of components" information is printed to two decimal places only. By default, blanks appear whenever coefficients are less than 0.1 in absolute value:

```
> summary(possum.prc, loadings=TRUE, digits=2)
Importance of components:
                        Comp.1    2    3    4    5    6    7    8    9
Standard  deviation     6.80  5.03 2.67 2.16 1.74 1.60 1.29 1.11 0.92
```

[1]
```
## Plot of principal components: possum[, 6:14]
here<- complete.cases(possum[, 6:14])
colr <- c("red", "blue")
pchr <- c(3,4,0,8,2,10,1)
ss <- expand.grid(site=1:7, sex=1:2)            # Site varies fastest
xyplot(possum.prc$scores[, 2] ~ possum.prc$scores[, 1],
       groups = possum$sexsite[here], col = colr[ss$sex], pch = pchr[ss$site],
       xlab="1st Principal Component", ylab="2nd Principal Component",
       key=list(points = list(pch=pchr),
                text=list(c("Cambarville", "Bellbird", "Whian Whian",
                            "Byrangery", "Conondale", "Allyn River",
                            "Bulburin" )), columns=4))
```

```
Proportion of Variance  0.50  0.27 0.08 0.05 0.03 0.03 0.02 0.01 0.01
Cumulative Proportion   0.50  0.77 0.85 0.90 0.93 0.96 0.98 0.99 1.00

Loadings:
         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
hdlngth  -0.41   0.28   0.34  -0.19   0.70  -0.28         -0.18
skullw   -0.30   0.27   0.54  -0.34  -0.52   0.28   0.26   0.11
totlngth -0.52   0.31  -0.65  -0.16          0.23  -0.15   0.34
taill            0.25  -0.35         -0.19          0.44  -0.75   0.11
footlgth -0.51  -0.47                -0.34  -0.63
earconch -0.31  -0.65                 0.25   0.58   0.21  -0.17
eye                                                 0.19   0.24   0.94
chest    -0.22          0.17   0.17  -0.18   0.19  -0.76  -0.40   0.27
belly    -0.25   0.18   0.13   0.89          0.10   0.24   0.14
```

Most of the variation is in the first three or four principal components. Notice that components 5 and 6 each explain only 3% of the variance. Later components explain, individually, even less. The variation in these later components is so small that it may well mostly represent noise in the data.

The loadings are the multiples of the original variables that are used in forming the principal components. To a close approximation, the first three components are:

```
Comp.1: -0.42×ldlngth-0.3×skullw-0.52×totlngth-0.51×footgth
                        -0.31×earconch-0.22×chest-0.25×belly
Comp.2: 0.28×ldlngth+0.27×skullw+0.31×totlngth+0.25×tailll
                      -0.47×footgth-0.65×earconch+0.18×belly
Comp.3: 0.34×ldlngth+0.54×skullw-0.65×totlngth-0.35×tailll
                                    +0.17×chest+0.13×belly
```

Notice that the first component is pretty much a size component; the magnitudes of all coefficients are comparable. The negative signs are an artefact of the computations; the signs could all just as well have been switched.

The stability of the principal components plot

Figure 12.2 showed a rather clear separation into two groups, distinguishing Bellbird and Cambarville from the other sites. How stable, relative to statistical variation, are the clusters that were apparent? One way to check this is to use the bootstrap approach.

The population from which the data were taken should not be too unlike data in which each of the sample observations is repeated an infinite number of times. A new sample from this infinite population is obtained by sampling with replacement from the original sample, choosing a bootstrap sample that has the same size as the original sample. This is done for each of the seven sites, thus giving a bootstrap sample data frame `bsample1.possum` that replaces the original data. The process is repeated several times, giving further such data frames `bsample2.possum`, `bsample3.possum`, ...

The steps that led to Figure 12.2 are then repeated for each of these bootstrap sample data frames, and plots obtained that reproduce Figure 12.2 for each of these bootstrap sample

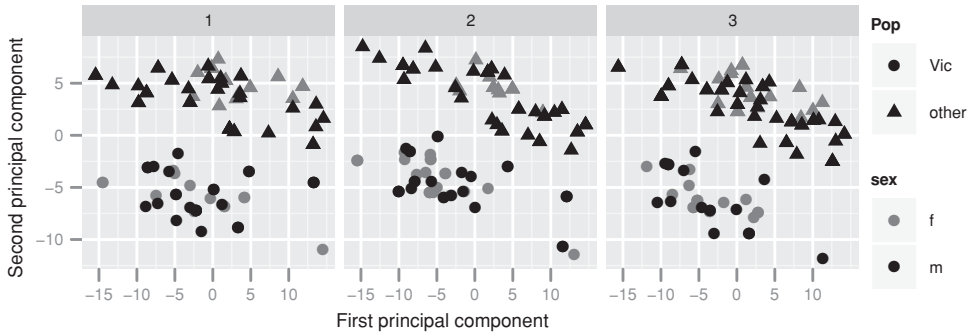


Figure 12.3 Second principal component versus first principal component, for variables in columns 6–14 of the three bootstrap versions of the possum data frame.

data frames. Figure 12.3 presents three such plots. The graphs are simplified so that they distinguish the two Victorian sites (Cambarville and Bellbird) from the other five sites. Use of the quickplot() function from the *ggplot2* package (see Section 15.6) makes it straightforward to distinguish sex as well as Pop within the individual panels.

All three panels show a similar distinction between the two sets of sites. The view given by the principal components may thus be reasonably stable with respect to sampling variation.

The following code was used to give Figure 12.3:

```
## Bootstrap principal components calculations: possum (DAAG)
library(ggplot2)
## Sample from all rows where there are no missing values
rowsfrom <- (1:dim(possum)[1])[complete.cases(possum[, 6:14])]
n <- length(rowsfrom); ntimes <- 3
bootscores <- data.frame(matrix(0, nrow=ntimes*n, ncol=2))
names(bootscores) <- c("scores1","scores2")
allsamp <- numeric(ntimes*n)
for (i in 1:ntimes){
    samprows <- sample(rowsfrom, n, replace=TRUE)
    bootscores[n*(i-1)+(1:n), 1:2] <-
        princomp(possum[samprows, 6:14])$scores[, 1:2]
    allsamp[n*(i-1)+(1:n)] <- samprows
}
bootscores[, c("sex","Pop")] <- possum[allsamp, c("sex","Pop")]
bootscores$sampleID <- factor(rep(1:ntimes, rep(n,ntimes)))
quickplot(x=scores1, y=scores2, colour=sex,
        shape=Pop, facets=.~sampleID, data=bootscores) +
        scale_colour_manual(value=c("m"="black","f"="gray45")) +
        xlab("First Principal Component") +
        ylab("Second Principal Component")
```

12.1.3 Multi-dimensional scaling

We noted above that PCA, in the use made of it here, is a special case of a much wider class of multi-dimensional scaling (MDS) methods. MDS starts with a matrix of "distances"

between points. Classical MDS with Euclidean distance (in three-dimensional space this is just the usual "distance") is equivalent to a principal components representation, working with the unscaled variables. MDS thus allows a choice of a distance measure that can in principle more closely match the scientific requirements. Computational demands can be severe, relative to principal components approaches.

Distance measures

In the best case, a distance measure may arise naturally from the way that the data are believed to have been generated. Thus, for gene sequences, the function `dist.dna()` in the *ape* package has several different measures of distance that arise from different models for the evolutionary process.

For some commonly used non-Euclidean distances, see `help(dist)`, or, for a more extensive range of possibilities, the help page for the function `daisy()` in the *cluster* package. The function `cmdscale()` implements classical MDS.

A particularly simple non-Euclidean distance (or "metric") is the `"manhattan"`. In two dimensions this is the smallest walking distance between street corners, taking a succession of left and/or right turns along streets that are laid out in a two-dimensional grid.

Ordination

In classical ("metric") MDS the distances, however derived, are treated as Euclidean. The function `cmdscale()` (*MASS*) implements this methodology, seeking a configuration of points in a low-dimensional space such that the distances are as far as possible preserved. Other ("non-metric") methods typically use classical MDS to derive a starting configuration.

There are several reasons for moving away from the rigidity of treating the "distances" as distances in a Euclidean space. "Distances" that are restricted to lie between 0 and 1, as in Subsection 13.2.2 (see Figure 13.5), can at best be regarded as relative distances. Small distances may be more accurate than large distances. Or it may be important to reproduce smaller distances more accurately.

The Sammon method (Sammon, 1969) minimizes a weighted sum of squared differences between the supplied and the fitted distance, with weights inversely proportional to the distance. This can be a good compromise between classical MDS and methods such as Kruskal's non-metric MDS. The function `sammon()` (*MASS*) implements Sammon's method, while `isoMDS()` (also *MASS*) implements Kruskal's non-metric MDS. Kruskal's non-metric MDS is a challenge for optimization algorithms, and calculations may take a long time.

The following code demonstrates the use of (1) `sammon()` and (2) `isoMDS()`, using Euclidean distance:

```
library(MASS)
d.possum <- dist(possum[,6:14])  # Euclidean distance matrix
possum.sammon <- sammon(d.possum, k=2)
plot(possum.sammon$points)       # Plot 2nd vs 1st ordinates
possum.isoMDS <- isoMDS(d.possum, k=2)
plot(possum.isoMDS$points)
```

A rotation and/or a reflection may be required to align these plots with each other and with Figure 12.2. After such alignment, the plots are, for these data, remarkably similar.

For further details, see Venables and Ripley (2002), Gordon (1999), Cox and Cox (2001), Izenman (2008).

Binary data

Binary data raises special issues, both for the choice of distance measure and for the choice of MDS methodology. For the "binary" measure, the number of bits where only one is "on" (i.e., equal to one) is divided by the number where at least one is on. If it turns out that distances of 1.0 are a substantial proportion of the total data, they may then not be very informative. Points that are at distances of 1.0 from each other and from most other points cannot, if there is a substantial number of them, be accurately located in the low-dimensional space.

It is then undesirable to give much weight to distances close to or equal to 1.0. Use of `sammon()` or `isoMDS()` is then much preferable to use of `cmdscale()`. The plots may have a striking visual appearance, with points for which distances from most other points are close to 1.0 lying on a circle around the boundary of the total configuration of points.

12.2 Discriminant analysis

Discriminant analysis seeks a rule that will allow prediction of the class to which a new observation may belong. Using language that comes originally from the engineering pattern recognition literature, the interest is in supervised learning. For example, the aim may be to predict, based on prognostic measurements and outcome information for previous patients, which future patients will remain free of disease symptoms for 12 months or more. Two methods will be demonstrated here – logistic discrimination and linear discriminant analysis using the function `lda()` in *MASS*. The tree-based methods that were discussed in Chapter 11 are among other methods that are available.

Cross-validation and the training/test set methodology, introduced in earlier chapters as approaches for assessing predictive accuracy, will have key roles in work with discriminant methods. Note again that resubstitution or training set accuracy, that is, prediction accuracy on the data used to derive the model, must inevitably improve as the prediction model becomes more complex. The cross-validation estimate of predictive accuracy will in general, at some point as model complexity increases, begin to decrease. The same is true for predictive accuracy on test data that are separate from the data used to train the model.

Cross-validation assessments of accuracy, or assessments that are based on test data derived from splitting the sample data into two parts, do however have limitations. Cross-validation gives an estimate of predictive accuracy for the population from which the data have been sampled. It is convenient to call this the "source" population. With observational data, the target to which results will be applied is almost inevitably different, commonly in time and/or space, from the "source". Accuracy estimates that are based on cross-validation, or on a training/test split of the sample data, must then be treated as provisional, until and unless test data become available from the population that is the true target.

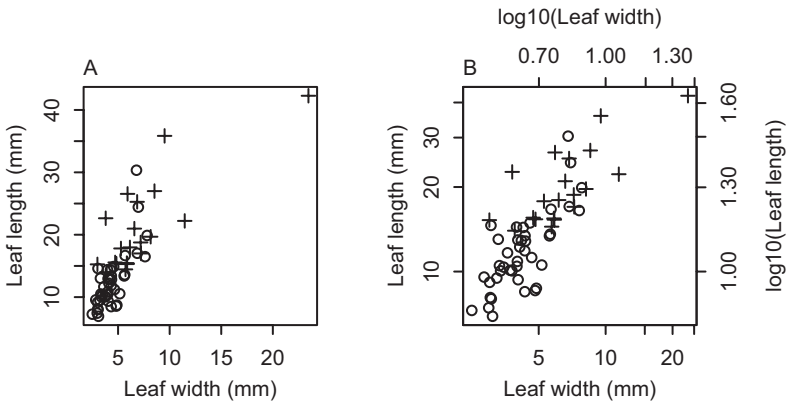


Figure 12.4 Leaf length versus leaf width (A) with untransformed scales; and (B) using logarithmic scales. The symbols are o = plagiotropic, + = orthotropic. Data, in the data frame `leafshape17`, are from a North Queensland site.

12.2.1 Example – plant architecture

Our first example is from data on plant architecture (data relate to (King and Maindonald, 1999). There is a discussion of plant architectures, with diagrams that help make clear the differences between the different architectures, in King (1998). Orthotropic species have steeply angled branches, with leaves coming off on all sides. Plagiotropic species have spreading branches (a few intermediates and a third uncommon branching pattern are excluded). The interest was in comparing the leaf dimensions of species with the two different architectures. Is leaf shape a good discriminator? The interest is not in prediction *per se*, but in the extent to which leaf shape discriminates between the two architectures.

We examine data from a North Queensland site, one of the six sites that provided data. Figure 12.4 is a plot of the data. A logarithmic scale is clearly preferable, as in Figure 12.4B.

With two explanatory variables, there is not much point in a plot of principal components scores. A plot of the second versus first principal components scores from the log transformed data would look like Figure 12.4B, but rotated so that the major axis of variation (running from the lower left to the upper right of Figure 12.4B) is horizontal. We could draw a line by hand through these data at right angles to this major axis, from the lower right to the upper left, that would discriminate orthotropic from plagiotropic species. However, we need a more objective way to discriminate the two classes.

The function `glm()`, and the predict method for `glm` objects, reflect a classical approach to statistical inference and make no explicit assumptions about the prior frequencies of the classes. For making predictions, however, the expected proportions in the two (or more) classes are important in finding a rule that will optimally assign a new sample to a cancer type, with as high a probability of correct classification as possible.

More generally, different types of misclassification (orthotropic for a plagiotropic species, as against plagiotropic for an orthotropic species) may have different costs that should be taken into account. Costs are often important in a medical context. In a screening test for a

disease, some false positives are acceptable, while any false negative (failing to detect an instance of the disease) may have very serious consequences.

Notation

Let π_0 and π_1 be the respective prior probabilities of orthotropic and plagiotropic architectures. For predictions from the logistic regression, π_0 and π_1 are each implicitly assumed equal to 0.5. In the call to `lda()`, π_0 and π_1 can be supplied, using the argument `prior`. They can also be changed, again using an argument `prior`, in a call to the predict method for an `lda` object. By default, `lda()` takes π_0 and π_1 to be the frequencies in the data.

Then the discriminant analysis yields a linear function $f(x, y)$ of $x = \log(\text{leaf width})$ and $y = \log(\text{leaf length})$ such that when

$$f(x, y) < -\log\left(\frac{\pi_1}{\pi_0}\right),$$

the prediction is that the plant will be plagiotropic, while otherwise the plant is predicted to be orthotropic. The function $f(x, y)$ has the form $a(x - \bar{x}) + b(y - \bar{y})$ and is called the discriminant function. There is one value for each observation. The values of the discriminant function are known as scores. The constants a and b must be estimated from the data. Predictions for `glm` objects assume, in effect, that $\log\left(\frac{\pi_1}{\pi_0}\right) = 0$, but results from the output from `glm()` can readily be adapted to accommodate $\pi_1 \neq \pi_0$ and hence $\log\left(\frac{\pi_1}{\pi_0}\right) \neq 0$.

12.2.2 Logistic discriminant analysis

Binary logistic discrimination works with a model for the probability, e.g., that a plant will be orthotropic. To fix attention on this specific example, the model specifies log(odds orthotropic) as a linear function of the explanatory variables for the classification. We get the function from

```
> leaf17.glm <- glm(arch ~ logwid + loglen, family=binomial,
+                   data=leafshape17)
> options(digits=3)
> summary(leaf17.glm)$coef
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -15.286       4.09  -3.735 0.000188
logwid         0.185       1.57   0.118 0.905911
loglen         5.268       1.95   2.704 0.006856
```

Thus we have

$$\log(\text{odds orthotropic}) = -15.3 + 0.185\log(\text{width}) + 5.628\log(\text{length}).$$

Predictive accuracy

The cross-validation estimate is a reasonable assessment of the accuracy that can be expected for a new set of data, sampled in the same way as the existing training data and from the

same source population. Our function `CVbinary()` can conveniently be used to do the cross-validation calculations. It returns the class that is predicted from the cross-validation in the list element `cv`.

```
## Confusion matrix: cross-validation estimate
> leaf17.cv <- CVbinary(leaf17.glm)
> tCV <- table(leafshape17$arch, round(leaf17.cv$cv))        # CV
> cbind(tCV, c(tCV[1,1], class.acc=tCV[2,2])/(tCV[,1]+tCV[,2]))
   0  1
0 36  5 0.878
1  7 13 0.650
```

In the above table, $36/(36 + 5)$ is calculated as 0.878, while $13/(7 + 13)$ is calculated as 0.65. The overall predictive accuracy is $(36 + 13)/(36 + 13 + 5 + 7) = 80.3\%$. We can find this as follows:

```
> sum(tCV[row(tCV)==col(tCV)])/sum(tCV)  # Overall accuracy
[1] 0.803
```

The results will vary slightly from one run of `CVbinary()` to the next.

Accuracies for the data that were used to train the model, here referred to as the *internal* or *resubstitution* accuracy rate, are in general optimistic, and should not be quoted. They are given here for comparative purposes. The function `CVbinary()` returns the resubstitution estimate in the list element `resub`.

```
## Confusion matrix: resubstitution estimate
## This can be grossly optimistic, and should be ignored
> tR <- table(leafshape17$arch, round(leaf17.cv$internal))
> cbind(tR, c(tR[1,1], class.acc=tR[2,2])/(tR[,1]+tR[,2]))
   0  1
0 37  4 0.902
1  7 13 0.650
> sum(tR[row(tR)==col(tR)])/sum(tR)  # Overall accuracy
[1] 0.82
```

The leave-one-out cross-validation accuracy was, for the cross-validation run that is reported here, slightly lower than the resubstitution assessment.

12.2.3 Linear discriminant analysis

The function `lda()`, from the Venables and Ripley *MASS* package, is set in an explicit Bayesian framework, as described in Ripley (1996, p. 36). For two classes it is a logistic regression, but the assumptions (notably, multivariate normal within-group distributions with a variance–covariance matrix that is common across groups) are different from those for `glm()`. It yields posterior probabilities of membership of the several groups. The allocation which makes the smallest expected number of errors chooses, following the *Bayes rule*, the class with the largest posterior probability. Where there are $g > 2$ groups, there are $g - 1$ discriminant axes.

The function `qda()` is an alternative to `lda()` that allows for different variance–covariance matrices in different groups. A restriction is that for use of p features, each group must have at least $p + 1$ observations. Unlike `lda()`, it does not lead to discriminant axes that are common across all groups.

We first extract predictions, and then examine the discriminant function. The code is:

```
library(MASS)
## Discriminant analysis; data frame leafshape17 (DAAG)
leaf17.lda <- lda(arch ~ logwid+loglen, data=leafshape17)
```

Output from the analysis is:

```
> leaf17.lda
Call:
lda.formula(arch ~ logwid + loglen, data = leafshape17)

Prior probabilities of groups:
    0     1
0.672 0.328

Group means:
  logwid loglen
0   1.43   2.46
1   1.87   2.99
Coefficients of linear discriminants:
         LD1
logwid 0.156
loglen 3.066
```

The coefficient estimates are $\mathtt{a} = 0.156$ and $\mathtt{b} = 3.066$.

The prior probabilities should reflect the expected proportions in the population to which the discriminant function will be applied, which may be different from the relative frequencies in the data that were used to make predictions. Both `lda()` and the `predict` method for `lda` objects take the argument `prior`, allowing predictions to use prior probabilities that may be different from those that were assumed in the call to `lda()`.

Assessments of predictive accuracy

Predictions for the data used to train the model can be obtained using `predict(leaf17.lda)`. Such predictions will, here, be ignored. Instead, we will rerun the calculation with the argument `CV=TRUE`. Predictions are then based on leave-one-out cross-validation, i.e., observations are left out one at a time, the model is fitted to the remaining data, and a prediction is made for the omitted observation. Here then are the calculations:

```
> leaf17cv.lda <- lda(arch ~ logwid+loglen, data=leafshape17,
+                     CV=TRUE)
> ## the list element 'class' gives the predicted class
> ## The list element 'posterior' holds posterior probabilities
```

```
> tab <- table(leafshape17$arch, leaf17cv.lda$class)
> cbind(tab, c(tab[1,1], class.acc=tab[2,2])/(tab[,1]+tab[,2]))
   0  1
0 37  4 0.902
1  8 12 0.600
> sum(tab[row(tab)==col(tab)])/sum(tab)
[1] 0.803
```

The function `multinom()`, in the *nnet* package that is supplied as part of the recommended Venables and Ripley *VR* bundle of packages, offers another approach which is however less well adapted for use in prediction.

12.2.4 An example with more than two groups

We present discriminant analysis calculations for the `possum` data frame. The methods followed here are similar to those used in Lindenmayer *et al.* (1995), with these same data, in making a case for the identification of a new possum species. (The species is named *Trichodurus cunninghami* for the statistician whose analysis led to this identification. See Hall (2003).)

We will use the same nine variables as before.[2] The output is:

```
> possum.lda
Call:
lda(site ~ hdlngth + skullw + totlngth + taill + footlgth +
    earconch + eye + chest + belly, data = na.omit(possum))

Prior probabilities of groups:
     1      2      3      4      5      6      7
0.3267 0.0990 0.0693 0.0693 0.1287 0.1287 0.1782

Group means:
  hdlngth skullw totlngth taill footlgth earconch  eye chest belly
1    93.7   57.2     89.7  36.4     73.0     52.6 15.0  27.9  33.3
2    90.2   55.6     82.0  34.5     70.6     52.1 14.4  26.8  31.2
3    94.6   58.9     88.1  37.2     66.6     45.3 16.1  27.6  34.9
4    97.6   61.7     92.2  39.7     68.9     45.8 15.5  29.6  34.6
5    92.2   56.2     86.9  37.7     64.7     43.9 15.4  26.7  32.0
6    89.2   54.2     84.5  37.7     63.1     44.0 15.3  25.2  31.5
7    92.6   57.2     85.7  37.7     65.7     45.9 14.5  26.1  31.9

Coefficients of linear discriminants:
              LD1      LD2      LD3      LD4      LD5      LD6
```

[2]
```
## Linear discriminant calculations for possum data
possum.lda <- lda(site ~ hdlngth + skullw + totlngth + taill + footlgth +
                  earconch + eye + chest + belly, data=na.omit(possum))
  # na.omit() omits any rows that have one or more missing values
options(digits=4)
possum.lda$svd       # Examine the singular values
plot(possum.lda, dimen=3)
  # Scatterplot matrix - scores on 1st 3 canonical variates (Figure 12.5)
```

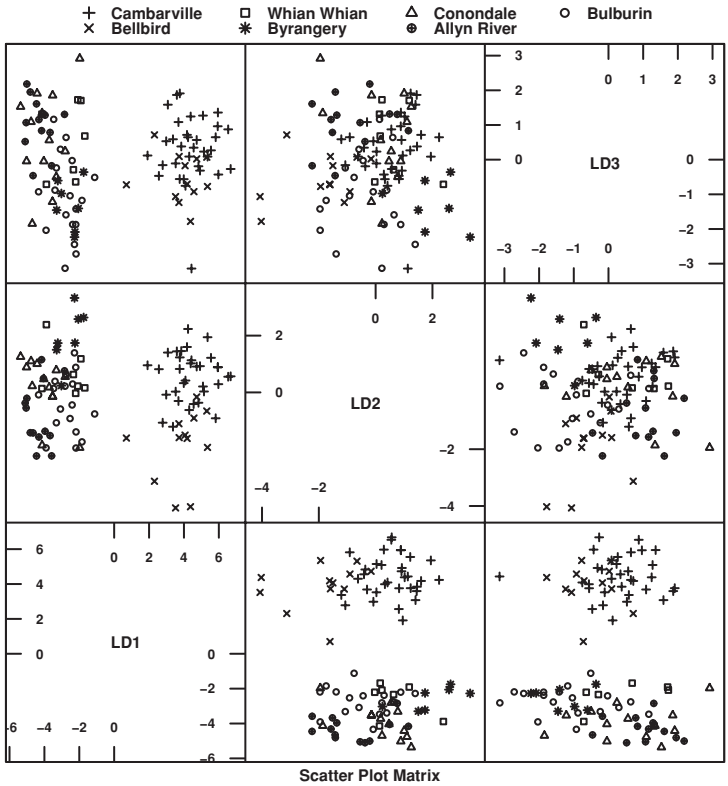


Figure 12.5 Scatterplot matrix for the first three canonical variates based on the linear discriminant analysis of the possum data.

```
hdlngth  -0.1494  0.0848 -0.2427 -0.0272 -0.0842 -0.1867
skullw   -0.0256  0.0624 -0.2422  0.1056 -0.1492  0.1412
totlngth  0.1165  0.2546  0.3257 -0.1795 -0.0816 -0.1164
taill    -0.4577 -0.0690 -0.4532 -0.1835  0.3013  0.5122
footlgth  0.2926 -0.0251 -0.0312  0.0457  0.0619 -0.1047
earconch  0.5809 -0.0626 -0.0876 -0.0783 -0.0311  0.2692
eye      -0.0548  0.0284  0.7763  0.4522  0.2196  0.3348
chest     0.1021  0.0724  0.0216  0.2642  0.6714 -0.0472
belly     0.0073 -0.0333  0.1080  0.1075 -0.3376  0.1813

Proportion of trace:
   LD1    LD2    LD3    LD4    LD5    LD6
0.8953 0.0511 0.0371 0.0087 0.0056 0.0023
```

The "proportion of trace" figures are the proportions of the between-class variance that are explained by the successive linear combinations. For these data, the first linear discriminant does most of the discriminating. The variables that seem important are `hdlngth`, and `taill`, contrasted with `totlngth` and `footlgth`. Figure 12.5 shows the scatterplot matrix for the first three discriminant scores, which together account for 98.4% of the between-class variance.

We invite the reader to repeat the analysis with the argument `CV=TRUE` in the call to `lda()`, in order to obtain a realistic predictive accuracy estimate.

12.3* High-dimensional data, classification, and plots

Data sets that have many more variables than observations are now common in a number of application areas. Here, attention will be limited to data where the observations fall into known previously identified groups.

The data used here were the basis for Golub *et al.* (1999). The *hddplot* package has a processed version of these data, in the matrix `Golub`. These are expression array data, i.e., each of the 7129 variables (or "features") is a measure of the biological activity of a gene. Technically, the values are "gene expression indices". The data are derived from the *golubEsets* package, available on the Bioconductor web site. The *hddplot* version has been subjected to some further preprocessing, beyond the processing that preceded their incorporation into *golubEsets*.

Following a terminology that is common for such data, the variables will be called features. Each of the 72 observations (columns of `Golub`) is from a tissue sample from a cancer patient. The 72 observations are classified into one of the three cancer types ALL B-type (coded `allB`), ALL T-type (coded `allT`), and AML (coded `aml`). ALL is Acute Lymphoblastic Leukemia (lymphoblastic = producing lymph tissue), while AML is Acute Myoblastic Leukemia (myoblastic = producing muscle tissue). Differences in the cancer types are not however the only differences between the samples, which is a complication for analysis.

One use for such data is to find a discrimination rule that, using a small subset of the features, will allow discrimination between the different cancer types. Such a rule might allow the design of a diagnostic device (a "probe") that, given a new sample, could determine the cancer type. (Note however that any classification of cancers is likely to conceal large individual differences that, in many cancers, arise from random differences in the timing and outcome of trigger points in a cascade of genomic damage and disruption.)

As already noted, use of these data for discrimination between cancer types is complicated by the potential effects of other factors. As well as different sexes, there are two different body tissues (bone marrow and peripheral blood). There may also be variation because the tissues came from four different hospitals; this will not be pursued here.

The presence of these other factors makes graphical exploration especially important. Finding suitable views of the data, inevitably low-dimensional, is however a challenge. Views are required that may help reveal subgroups in the data, or points that may have been misclassified, or between-group differences in the variance–covariance structure. Graphs should be revealing, without serious potential to mislead.

The papers Maindonald and Burden (2005), Ambroise and McLachlan (2002), and Zhu *et al.* (2006) are useful background reading for the discussion of this section.

What groups are of interest?

The data frame `golubInfo` has information on the tissue samples, which are the observations. The two classifications that will be investigated are (1) according to tissue type

and sex, given by the factor `tissue.mf`, and (2) according to cancer type (ALL B-type, ALL T-type, AML), given by the factor `cancer`.

The frequencies in the two-way classification by `cancer` and `tissue.mf` are:

```
> library(hddplot)
> data(golubInfo)
> with(golubInfo, table(cancer, tissue.mf))
       tissue.mf
cancer BM:NA BM:f BM:m PB:NA PB:f PB:m
  allB   4    19   10    2     1    2
  allT   0     0    8    0     0    1
  aml   16     2    3    1     1    2
```

For the classification (1) above, according to tissue type and sex (`tissue.mf`), restriction to the `allB` leukemia type and to patients whose sex is known gives a relatively homogeneous set of data. Below, we will define a factor `tissue.mfB` that classifies the `allB` subset of the data for which the sex of the patient is known, and for which at least two samples are available. (The single `allB` observation that is `PB:f` will be omitted.) The levels of `tissue.mfB` will be a subset of those of `tissue.mf`. Restriction to this subset, at least for preliminary investigation, is desirable because the different tissue/sex combinations may bias the attempt to compare cancer types.

(Observe that `allB` is predominantly `BM:f`, while `aml` is predominantly `BM` of unknown sex. If we compare `allB` with `aml` and ignore other factors, will any differences be due to cancer type, or to the sex of the patient, or to the tissue type?)

For the classification (2) above, according to cancer type (`cancer`), some limited homogeneity will be imposed by restricting attention to bone marrow (`BM`) samples. A classifying factor `cancer.BM` will be defined that relates to this reduced subset. Consideration of results for these data should bear in mind that they may be affected by sex differences as well as, possibly, by other unidentified factors (e.g., different types of chromosome damage) where the numbers may differ between the different subgroups.

The following preliminary calculations separate out the `allB` subset (`GolubB`) of the data that will be used (classification 1 above), and derive the factor `tissue.mfB` whose levels are `BM:f`, `BM:m` and `PB:m`:

```
attach(golubInfo)
## Identify allB samples for that are BM:f or BM:m or PB:m
subsetB <- cancer=="allB" & tissue.mf%in%c("BM:f","BM:m","PB:m")
## Form vector that identifies these as BM:f or BM:m or PB:m
tissue.mfB <- tissue.mf[subsetB, drop=TRUE]
## Separate off the relevant columns of the matrix Golub
data(Golub)  # NB: variables(rows) by cases(columns)
GolubB <- Golub[, subsetB]
detach(golubInfo)
```

The argument `drop=TRUE` in `tissue.mf[subsetB, drop=TRUE)` causes the return of a factor that has only the levels that are present in the data subset.

12.3.1 Classifications and associated graphs

In the discussion that now follows, interest will be on the graphical view that can be associated with one or other discriminant rule, rather than in the discriminant rules themselves. The objective is to give a visual representation that shows one or other classification that is of interest. Different classifications will lead to different graphical views – what is seen depends, inevitably, on which clues are pursued. Overly complex classifications may force unsatisfactory compromises in the view that is presented. As noted above, care is required to ensure that graphs present a fair view of the data, not showing spurious differences between groups or exaggerating such differences as may exist.

Discrimination will use the relatively simple and readily understood linear discriminant function methodology that was introduced and used earlier, in Subsection 12.2.3. Analyses will, again, use the `lda()` function (*MASS*). Linear discriminant functions may be as complicated as is sensible, given the substantial noise in current expression array data sets. In any case, the emphasis will be on insight rather than on the use of methods that are arguably optimal.

The statistical information given in the output from the function `lda()` assumes that the variance–covariance matrix is the same in all groups. Even where this is not plausible, a useful graphical is still possible, perhaps showing pronounced between group differences in the variance–covariance structure.

The methodology that will be described is not readily adaptable for use with `qda()`.

Preliminary data manipulation

An observation on a tissue sample comes from a single "chip", possibly leading to systematic differences between observations. Preprocessing is needed to align the feature values for the different observations. For the data set `Golub` in *hddplot*, data have been processed so that, among other things, the median and standard deviation are the same across the different slides. Full details will be included in a Sweave file that will be included with a future version of *hddplot*.

As is standard practice with expression array data, the rows of `Golub` are features (variables), while the columns are observations. Transposition to an observations by features layout will be required for use of `lda()` or other modeling functions. Before proceeding further, the distribution for individual observations across features, and the distribution for a selection of features across observations, should be checked.[3] Both distributions are positively skewed.

12.3.2 Flawed graphs

Figure 12.6A is a flawed attempt at a graph that shows the separation of the 31 observations into the three specified groups. It uses discriminant axes that were determined using 15 features that individually gave the "best" separation into the three groups (see below). It is flawed because no account is taken of the effect of selecting the 15 "best" features, out of 7129. Figure 12.6B, which was obtained by applying the same procedure to random normal

[3]
```
## Display distributions for the first 20 observations
boxplot(data.frame(GolubB[, 1:20]))  # First 20 columns (observations)
## Random selection of 20 rows (features)
boxplot(data.frame(GolubB[sample(1:7129, 20), ]))
```

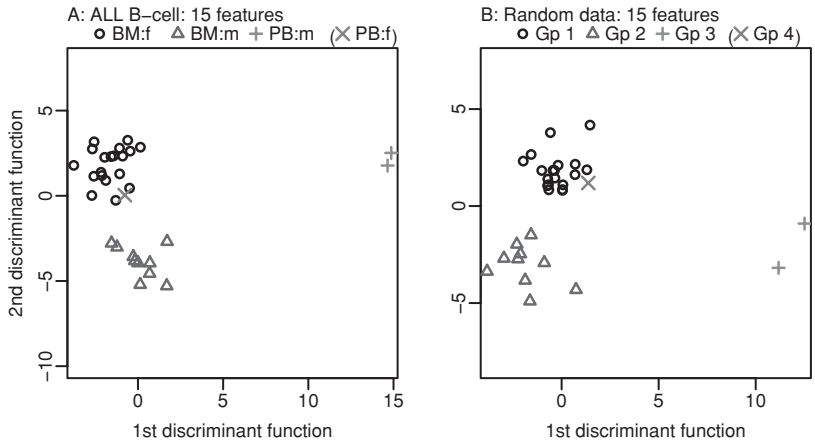


Figure 12.6 The left panel used the subset of ALL B-cell observations for which `Gender` was known. The one `PB:f` observation was excluded for purposes of analysis. An anova F-statistic calculation identified the 15 features that, individually, "best" separated the data into three groups. These 15 features were then used in a linear discriminant analysis. Scores were then determined for each of the two available discriminant axes. Additionally, a predicted score was determined for the `PB:f` observation. For the right panel, the same procedure was followed, but now using a matrix where the "expression values" were random normal data.

data, from 7129 independent normal variables that all had the same mean and variance, shows the potential for getting an entirely spurious separation into groups. In spite of its evident flaws, it is important to understand the procedure that was followed, as the later discussion will extend and adapt it to give graphs that are not similarly flawed.

In summary, Figure 12.6 was obtained as follows:

- The 15 features (from 7129) were selected that, as measured by an analysis of variance F-statistic, gave the best separation of the 31 observations into the three groups `BM:f`, `BM:m`, `PB:f`.[4]
- The two discriminant functions, and their associated discriminant scores, were calculated and the scores plotted.
- Predicted scores were determined for the one `PB:f` sample, allowing its inclusion in the plot.[5]

[4]
```
## Uses orderFeatures() (hddplot); see below
ord15 <- orderFeatures(GolubB, cl=tissue.mfB)[1:15]
```
[5]
```
## Panel A
dfB.ord <- data.frame(t(GolubB[ord15, ]))
## Calculations for the left panel
## Transpose to observations by features
dfB15 <- data.frame(t(GolubB[ord15, ]))
library(MASS)
dfB15.lda <-  lda(dfB15, grouping=tissue.mfB)
scores <- predict(dfB15.lda, dimen=2)$x
## Scores for the single PB:f observation
with(golubInfo, {
     df.PBf <- data.frame(t(Golub[ord15, tissue.mf=="PB:f"
                                  & cancer=="allB", drop=FALSE]))
     scores.PBf <- predict(dfB15.lda, newdata=df.PBf, dimen=2)$x
}
## Warning! The plot that now follows may be misleading!
## Use scoreplot(), from the hddplot package
scoreplot(list(scores=scores, cl=tissue.mfB, other=scores.PBf, cl.other="PB:f"))
```

Figure 12.6B used the same two steps, but with the input expression values replaced by random normal values.[6]

The function `simulateScores()` makes it easy, using different numbers of features, and different numbers of observations and groupings of those observations, to examine the use of the procedure just described with different configurations of random data. Readers are encouraged to experiment, using the code in footnote 6 as a model.

The selection of 15 features from a total of 7129, selected to have the largest F-statistics, makes it unwise to give much credence to the clear separation achieved with expression array data in Figure 12.6A. The extent of separation in Figure 12.6B from use of random normal data indicates the potential severity of the selection effect, for the data used for Figure 12.6A. The 15 most extreme F-statistics out of 7129, from a null distribution in which there is no separation between groups, will all individually show some separation. Choice of the "best" two discriminant axes that use these 15 features will achieve even clearer separation than is possible with any of the features individually. Clearer apparent separation, both for random data and for the Golub data, can be achieved by choosing more than 15 features.

Distributional extremes

There can be a small number of F-statistics that are so large that they are unlikely to be extremes from the null distribution. Plots such as Figure 12.6A can then be based on these features, with no concern about possible effects of selection bias. Correlations between features vitiates use of a theory that assumes that genes are independent. Additionally, the distributions for individual features may not be normal, in a context where the interest is in the distributional extremes of F-statistics and normality is likely to matter.

Permutation methods, implemented in the package *multtest* (Pollard, K. S., Ge, Y. and Dudoit, S., 2005), can however be used to determine a relevant reference distribution. The stand-alone version of this package, available from CRAN, is adequate for present purposes. (For installation of the BioConductor version, a minimal BioConductor installation must first be in place.)

The function `mt.maxT()` determines the needed empirical distribution, as part of its implementation of a multiple testing procedure – the "step-down" method. Our interest here is not in the multiple testing procedure, but in the empirical distribution of the F-statistics, which can be obtained as `qf(1-GolubB.maxT$rawp, 2, 28)`. The F-statistics for the assignment of labels as in the actual sample are stored in `GolubB.maxT$teststat`, i.e., the values are the same as those obtained from `aovFbyrow(GolubB, tissue.mfB)`.

The code used for the calculation is:

```
## The calculation may take tens of minutes, even with adequate
## memory and a fast processor.
## If necessary, use a smaller value of B.
```

[6]
```
## Panel B: Repeat plot, now with random normal data
simscores <- simulateScores(nrow=7129, cl=rep(1:3, c(19,10,2)),
                            cl.other=4, nfeatures=15, seed=41)
  # Returns list elements: scores, cl, scores.other & cl.other
scoreplot(simscores)
```

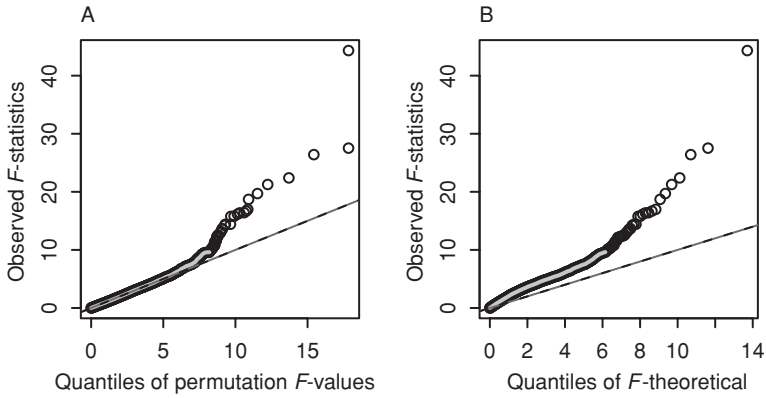


Figure 12.7 These QQ-plots are for the subset of ALL observations for which `Gender` was known, but excluding the one `PB:f` observation. The left plot compares the ordered F-statistics with the ordered statistics from the permutation distribution. The right plot compares the ordered F-statistics with F-distribution quantiles. Also shown, in both plots, is the line $y = x$.

```
library(multtest)
GolubB.maxT <- mt.maxT(GolubB, unclass(tissue.mfB)-1, test="f",
                       B=100000)
## Compare calculated F-statistics with permutation distribution
qqthin(qf(1-GolubB.maxT$rawp, 2, 28), GolubB.maxT$teststat)
## Compare calculated F-statistics with theoretical F-distribution
qqthin(qf(ppoints(7129), 2, 28), GolubB.maxT$teststat)
  # The theoretical F-distribution gives estimates of quantiles
  # that are too small
## NB also (not included in Figure 12.10) the comparison between
## the permutation distribution and the theoretical F:
qqthin(qf(ppoints(7129), 2, 28), qf(1-GolubB.maxT$rawp, 2, 28))
  # qqthin() is a version of qqplot() that thins out points where
  # overlap is substantial, thus giving smaller graphics files.
```

The argument `B` sets the number of permutations that will be taken. This needs to be substantially larger than the number of features in order to get estimates of the extreme upper quantiles of the distribution that are as accurate as possible, Figure 12.7 uses the function `qqthin()` (*hddplot*) to show QQ-plots that compare the relevant distributions.

Panel A, which uses an appropriate reference distribution, suggests that the largest two features, and others as well, show differential expression. The theoretical F-distribution, used for the horizontal axis in panel B, shows differences at almost all quantiles, and is not an appropriate reference distribution.

Use of the permutation distribution as reference (panel A), indicates that it is very unlikely that the null distribution would generate the largest of the F-statistics, and that these show genuine evidence of differential expression.

The features that show very clear evidence of differential expression are unlikely to be affected by selection bias. Selection bias is an issue when a choice is made among features whose F-statistics are relatively similar, and that collectively are inconsistent with the null.

Effective mechanisms for handling such issues are the subject of active research, and will not be pursued further in this section.

The subsequent discussion will demonstrate adaptations of the procedure used for Figure 12.6A, but which avoid its evident flaws and do not require a limiting of attention to a small number of features that show uniquely unequivocal evidence of differential expression.

Selection of features that discriminate

The function `orderFeatures()` will be used extensively in the sequel. It selects features that, as measured by an analysis of variance F-statistic, give, individually, the best separation between groups. The function takes as arguments

- `x`: the matrix of expression values, in the features by observations layout that is usual in work with expression arrays.
- `cl`: a factor that classifies the observations.
- `subset`: if changed from its default (`NULL`), this identifies a subset of observations (columns of `dset`) that will be used for calculating the statistics.
- `FUN`: currently the only available function (`aovFbyrow()`) uses the analysis of variance F-statistic as a measure of between-group separation.
- `values`: by default the function returns the order. If set to `TRUE` the function returns the ordered F-statistic values as well as the order.

Selection of features that discriminate best individually, in an analysis of variance F-statistic sense, is not guaranteed to select the features that perform best in combination. It is akin to using, in a multiple regression, the variables that perform well when used as the only predictors. It may however be a reasonable strategy for use in an initial exploratory analysis, in the absence of an obviously better alternative. The development of good variable selection methods, applicable to the data used in this section, is the subject of ongoing research.

12.3.3 Accuracies and scores for test data

Where there are adequate data, a cautious strategy is to split the data into two sets, here named I and II. Then set I can be used to train discriminant functions and to determine discriminant scores for the test observations in set II. The key requirement is that the scores must relate to observations that are distinct from those used to develop the discriminant functions, and are therefore free from the selection bias that affects the set I (training) scores. The set II test data has no role in either the selection of features, or the determination of the discriminant functions and associated scores.

The process can then be reversed, with set II used for training and scores calculated for set I. Two plots are then available, the first of which shows scores for set I, and the second scores for set I. The two plots, conveniently identified as I/II and II/I, will use different features and different discriminant functions and cannot be simply superposed.

Because there are only three observations in the `PB:m` category, the data used for Figure 12.6A cannot satisfactorily be split into training and test data. We can however use this approach to examine the classification of the bone marrow (`BM.PB=="BM"`) samples into ALL B-cell, ALL T-cell, and AML. This larger data set (62 observations), with larger numbers (8 or more) in each level of the classification, allows a split into a set I and a

set II such that in both cases each level of the classification has at least four observations.

Two approaches that might be used to determine the optimum number of features, when developing a discriminant rule from set I, are:

- Use the predicted accuracies for set II.
- Use cross-validation on set I.

Cross-validation will be demonstrated later in this section, albeit working with the total `allB` data.

The function `divideUp()` (*hddplot*) has a default that (with `nset=2`) is designed to make a training/test split, while ensuring similar relative numbers in the three levels of the classification.

```
Golub.BM <- with(golubInfo, Golub[, BM.PB=="BM"])
cancer.BM <- with(golubInfo, cancer[BM.PB=="BM"])
## Now split each of the cancer.BM categories between two subsets
##  Uses divideUp(), from hddplot
gp.id <- divideUp(cancer.BM, nset=2, seed=29)
  # Set seed to allow exact reproduction of the results below
```

Tabulating the division into two sets, we find:

```
> table(gp.id, cancer.BM)
     cancer.BM
gp.id allB allT aml
    1 17    4   10
    2 16    4   11
```

The maximum number of features for calculations using `lda()` with the set I data are 28 ($= 17 + 4 + 11 - 3 - 1$) for set I, and 26 for set II. Hence we will work with a maximum of 26 features, in each case. Steps in handling the calculations are:

1. Using set I as the training data, find the 26 features that, for a classification of the data into three groups according to levels of `cancer.BM`, have the largest F-statistics.
2. For each value of $n_f = 1, 2, \ldots, 26$ in turn
   - use the best n_f features to develop discriminant functions
   - apply this function to the data in set II, and calculate the accuracy.
3. The accuracies that result will be called the I/II accuracies.
4. Now make set II the training data and set I the test data, and repeat items 1 and 2. The accuracies that result will be called the II/I accuracies.

These calculations can be carried out using the function `accTrainTest()` (from *hddplot*).

```
> accboth <- accTrainTest(x = Golub.BM, cl = cancer.BM,
+                         traintest=gp.id)

Training/test split      Best accuracy, less 1SD   Best accuracy
I(training) / II(test)   0.89 (14 features)        0.94 (20 features)
II(training) / I(test)   0.92 (10 features)        0.97 (17 features)
```

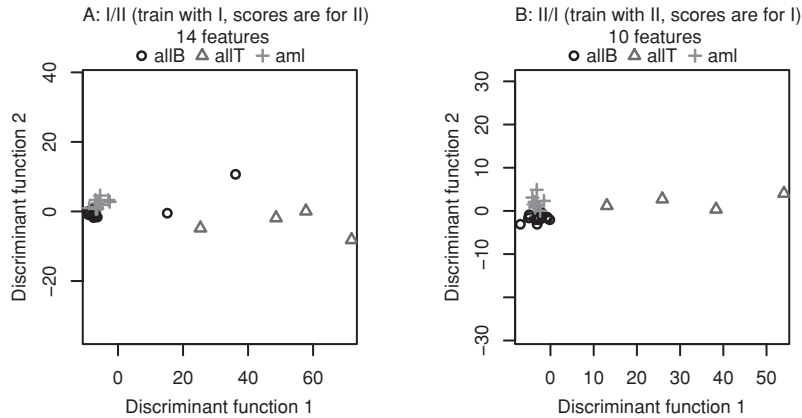


Figure 12.8 Panel A plots scores for the set II data, using set I for training (the I/II split), as described in the text. Panel B plots the scores for the set I data when the roles of the two sets were reversed, i.e., the split was II/I.

Notice that, as well as giving the number of features that gives the maximum accuracy, the output gives the number that achieves the maximum accuracy, less one standard deviation. This gives a more conservative estimate of the optimum number of features. (The standard deviation is estimated as $p(1-p)/n$, where p is the estimated maximum accuracy, and n is the number of observations used to estimate p.)

We now calculate both sets of test scores (I/II and II/I) for the more conservative choices of 14 and 10 features respectively, and use the function `plotTrainTest()` to plot the scores. Figure 12.8A shows the test scores for the I/II split, while Figure 12.8B shows the test scores for the II/I split.[7] Readers should construct the plots with other divisions of the data into training and test sets. To determine each new division, specify:

```
gp.id <- divideUp(cl=cancer.BM, nset=2, seed=NULL)
```

The graphs can vary greatly, depending on how the data have been split. The ALL T-cell points seem more dispersed than points for the other two categories.

It is instructive to compare the choices of features between I/II and II/I. The first 20 in the two cases are, by row number:

```
> rbind(accboth$sub1.2[1:20],accboth$sub2.1[1:20])
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
[1,] 6606 4342 6510 3594 4050 6236 1694 1207 1268  4847  5542
[2,] 4050 2794 6510 6696 4342 5542 4357 5543 1207  4584  6236
     [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
[1,]  2061  5543  4055  4375  1144   379  6696  4196   229
[2,]  1429  6575  2833  4750  2335  1704  4882  6225  3544
> match(accboth$sub1.2[1:20],accboth$sub2.1[1:20])
 [1] NA  5  3 NA  1 11 NA  9 NA NA  6 NA  8 NA NA NA NA  4 NA NA
```

[7] `## Use function plotTrainTest() from hddplot`
`plotTrainTest(x=Golub.BM, nfeatures=c(14,10), cl=cancer.BM, traintest=gp.id)`

Note that the first feature in the first list does not appear at all in the second list, and that the first feature in the second list is fifth in the first list.

Cross-validation to determine the optimum number of features

We will demonstrate the use of 10-fold cross-validation, repeated for each choice of number of features in the range that is pertinent, to determine how many features to choose.

Consider again the classification of a subset of the ALL B-cell Golub data for which gender is known into `BM:f`, `BM:m`, and `PB:m`, but omitting the one `PB:f` sample. There are 31 observations, divided into three groups, so that the maximum number of features that can be used for discrimination is 23. This is calculated as follows. At each fold, the training data consists of 9 out of 10 subsets in the 10-fold division of the data, i.e., at least 27 out of the 31 points. (Each subset has three or four observations.) One degree of freedom is lost for each of the three subgroups, and at least one degree of freedom should be left for estimating the variance. Thus at most 23 ($= 27 - 4$) degrees of freedom can be used for estimating linear discriminant parameters.

In order to choose the optimum number of features, the cross-validation must be repeated for each choice of g = number of features in the range $1, 2, \ldots, g_{\max} = 23$, calculating the cross-validation accuracy for each such choice. The number of features will be chosen to give an accuracy that is, or is close to, the maximum.

The full procedure is:

For $g = 1, 2, \ldots, g_{\max}$, do the following:

For each fold $i = 1, \ldots, k$ in turn (k = number of folds) do the following:

Split: Take the ith set as the test data, and use the remaining data (all except the ith set) for training.

Select: Choose the g features that have the largest anova between group F-statistics.

Classify: Determine discriminant functions, using the chosen features and the current training data.

Predict: Predict the groups to which observations in the current test set belong.

Record, against the number g of features used, the proportion of correct predictions. (This is calculated across all folds, and hence for the total data.)

Accuracies are now available for all choices of number of features. Choose the smallest number of features that gives close to the maximum accuracy.

The 10-fold cross-validation will be repeated for each of four different splits into 10 subsets. Especially in the present context, where at each fold of each repeat of the cross-validation there is a variable selection step, such use of repeats is desirable for adequate sampling of the variability.

Computations are greatly reduced by determining the ordering of features, for each fold of the data, in advance. These orderings are stored in a matrix of character values, with as many columns as there are folds, and with number of rows equal to the maximum number of features under consideration. A rigid upper limit is the number of features that can

be accommodated on the discriminant analysis, which as noted earlier is 23. When the preliminary calculations are finished, column i of the matrix will record the features that give the 23 highest F-statistics for the fold i training data. For selecting the "best" n_f features, one set for each different fold, the first n_f rows of this matrix ($f \leq 23$) will be taken.

Calculations will use the function `cvdisc()` (*hddplot*). For comparison, results are obtained both from the resubstitution measure of accuracy and from a defective use of cross-validation:

```
> ##  Cross-validation to determine the optimum number of features
> ## Accuracy measure will be: tissue.mfB.cv$acc.cv
> tissue.mfB.cv <- cvdisc(GolubB, cl=tissue.mfB, nfeatures=1:23,
+                         nfold=c(10,4))  # 10-fold CV (x4)

Accuracy           Best accuracy, less 1SD   Best accuracy
(Cross-validation) 0.85 (3 features)         0.9 (4 features)
> ## Defective measures will be in acc.resub (resubstitution)
> ## and acc.sel1 (select features prior to cross-validation)
> tissue.mfB.badcv <- defectiveCVdisc(GolubB, cl=tissue.mfB,
+                                     foldids=tissue.mfB.cv$folds,
+                                     nfeatures=1:23)
> ## NB: Warning messages have been omitted
> ##
> ## Calculations for random normal data:
> set.seed(43)
> rGolubB <- matrix(rnorm(prod(dim(GolubB))), nrow=dim(GolubB)[1])
> rtissue.mfB.cv <- cvdisc(rGolubB, cl=tissue.mfB, nfeatures=1:23,
+                          nfold=c(10,4))

Accuracy           Best accuracy, less 1SD   Best accuracy
(Cross-validation) 0.39 (1 features)         0.48 (9 features)
> rtissue.mfB.badcv <- defectiveCVdisc(rGolubB, cl=tissue.mfB,
+                                     nfeatures=1:23,
+                                     foldids=rtissue.mfB.cv$folds)
```

The resubstitution and "defective CV" points show biased and therefore inappropriate accuracy measures. The resubstitution points show the proportion of correct predictions when the discrimination rule is applied to the data used to develop the rule. The "defective CV" points show the proportion of correct predictions when the same features, selected using all the data, are used at each fold, and do not change from one fold to the next.

Figure 12.9B applies the same calculations to random data. The bias in the two incorrect "accuracies" is now obvious. The correct cross-validation estimates are now much worse than chance for more than three or four features, while the "defective CV" estimates continue to increase up to about 15 features. At each fold, the rule is tuned to be optimal for the quirks of the training data. It is therefore sub-optimal for the test data at each fold.

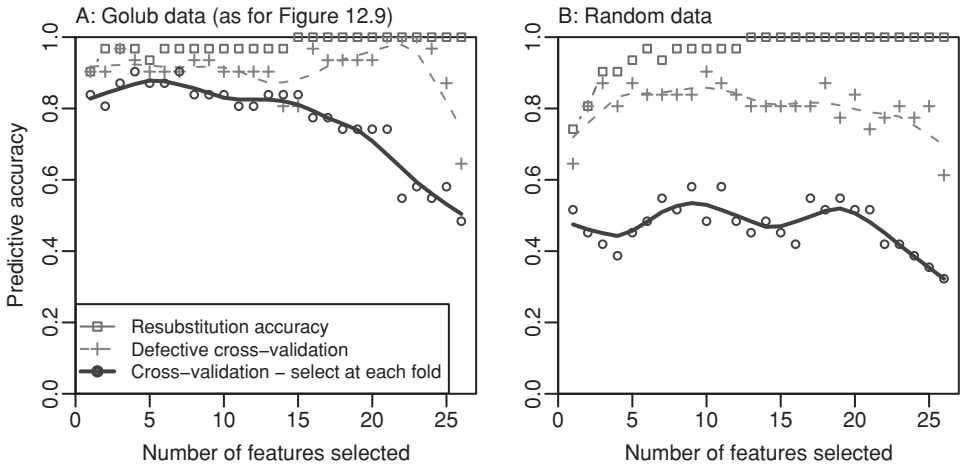


Figure 12.9 Comparison of different accuracy measures, in the development of a discriminant rule for the classification, into the categories `BM:f`, `BM:m`, and `PB:m`, of the B-cell ALL data for which gender is known. The resubstitution measure (□) is a severely biased measure. Cross-validation, but with features selected using all the data (+), is less severely biased. An acceptable measure of predictive accuracy (○) requires reselection of features at each fold of the cross-validation. The right panel shows the performance of each of these measures when the expression values were replaced by random data.

Which features?

It is of interest to see what features have been used at the different folds. This information is available from the list element `genelist`, in the object `tissue.mfB.cv` that the function `cvdisc()` returned. As the interest is in working with three features, it is the first three rows that are relevant. The following is a summary:

```
> genelist <- matrix(tissue.mfB.cv$genelist[1:3, ,], nrow=3)
> tab <- table(genelist, row(genelist))
> ord <- order(tab[,1], tab[,2], decreasing=TRUE)
> tab[ord,]

genelist           1  2  3
  M58459_at       32  4  0
  S74221_at        4  0  0
  U29195_at        4  0  0
  X54870_at        0 16  8
  U91327_at        0  8 16
  L08666_at        0  4  0
  U49395_at        0  4  0
  X00437_s_at      0  4  0
  X53416_at        0  0  4
  X62654_rna1_at   0  0  8
  X82494_at        0  0  4
```

Observe that M58459_at is almost always the first choice. There is much less consistency in the second and third choices.

Cross-validation: bone marrow (BM) samples only

It turns out to be sufficient to calculate accuracies for 1, 2, . . . , 25 features (the choice of 25 was a guess):

```
> BMonly.cv <- cvdisc(Golub.BM, cl=cancer.BM, nfeatures=1:25,
>                     nfold=c(10,4))

Accuracy           Best accuracy, less 1SD   Best accuracy
(Cross-validation) 0.9 (19 features)         0.94 (23 features)
```

The maximum is 94%, from use of 23 features. The more conservative assessment, based on the one-standard-deviation rule, suggests use of 19 features with an accuracy of 90%. If the interest is in using a small number of features to explain the evident group differences, this may seem unsatisfactory.

12.3.4 Graphs derived from the cross-validation process

With a methodology available for choosing the number of features, it is now possible to look for an alternative to Figure 12.6A that does not run the same risk of bias. Figure 12.8 demonstrated an approach that is sometimes available, but it gave two plots, each for half of the data. The function cvscores() (*hddplot*) makes it possible to give one plot for all the data. It can be used with any data where there are enough groups in each subset of the classification that cvdisc() can be used satisfactorily.

Consider first the creation of a plot for the subset of the allB data that formed classification 1. Figure 12.9A suggested that the optimum number of features is, conservatively, 3. The calculations that will be described here will use three features.

Test scores can be calculated for the test data at each of the folds. However the different pairs of scores (with three groups, there can be at most two sets of scores) relate to different discriminant functions and to different choices of features, and are thus appropriately called "local" test scores. Local fold i training scores are similarly available, again with one set for each value of i.

The local training scores are used to make the connection between the test scores at the different folds. A vignette that gives details will be included with the package *hddplot*. The methodology is a modification of that described in Maindonald and Burden (2005). Code for Figure 12.10A is:

```
## Panel A: Uses tissue.mfB.acc from above
tissue.mfB.scores <-
  cvscores(cvlist = tissue.mfB.cv, nfeatures = 3, cl.other = NULL)
scoreplot(scorelist = tissue.mfB.scores, cl.circle=NULL,
          prefix="B-cell subset -")
```

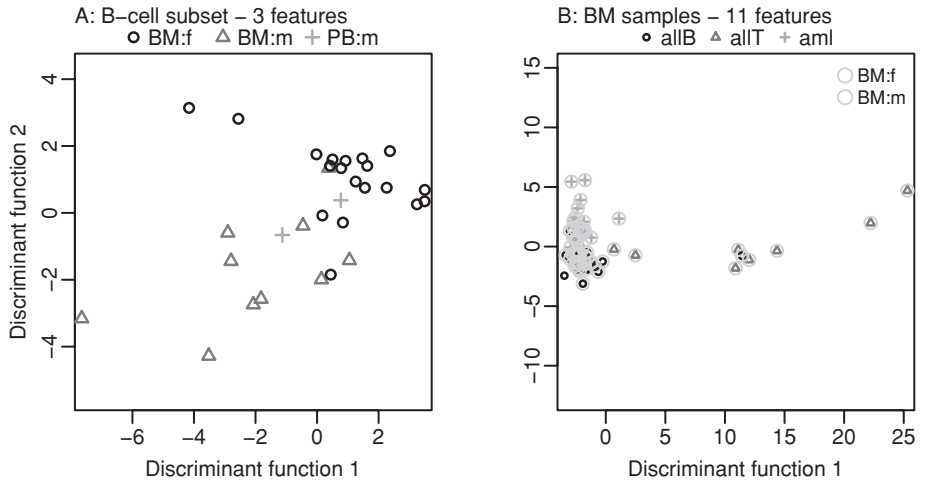


Figure 12.10 These plots of projections of linear discriminant analysis scores are designed to fairly reflect the performance of a linear discriminant in distinguishing between known groups in the data. The two panels relate to different subsets of the `Golub` data, with different groupings in the two cases. In panel B, for the classification of the 62 bone marrow (`BM`) samples into `allB`, `allT`, and `aml`, points where the sex is known are identified as male or female. See Plate 6 for a color version.

There are two clusters of points, with tissues from females mostly in one cluster and tissues from males in the other cluster.

Figure 12.10B has applied the same methodology to the classification of the bone marrow samples according to cancer type. Points where `Gender` is known are identified as male or female.[8]

Notice the clear clustering of points from females on the left of the graph. This complicates interpretation; is there a bias from the different gender balances in the three groups? This limited exploration indicates that heterogeneity of the samples is an important issue for the analysis of these data, and for the interpretation of any graphs.

The lines of investigation that have been pursued in this section should be taken as suggestions for an initial series of steps that might be followed. Extensive further investigation would be required for any analysis that claims to be moderately complete.

Further comments

The ideas that have been introduced in this section have far-reaching importance. The choice of variables, in Subsections 12.3.1 and 12.3.4, can be viewed as a form of model tuning. By tuning the fitted model to accidental characteristics of the training data, performance on the test data deteriorates, as seen in Figure 12.9B.

[8]
```
## Panel B; classify bone marrow samples a/c cancer type.
BMonly.scores <- cvscores(cvlist=BMonly.cv, nfeatures=19, cl.other=NULL)
scoreplot(scorelist=BMonly.scores, cl.circle=tissue.mfB,
          circle=tissue.mfB%in%c("BM:f","BM:m"),
          params=list(circle=list(col=c("cyan","gray"))),
          prefix="B: BM samples -")
```

With more complicated models (really, families of models) such as neural nets and Support Vector Machines (SVMs), there are many more tuning choices and tuning parameters. Thus, for example, see the details of tuning parameters that are given on the help page for the function `svm()` in the package *e1071* (Meyer, 2001). Such tuning can interact in complex ways with feature selection. For valid accuracy assessment, such tuning (in principle, at least) must be repeated at each cross-validation fold.

The *randomForest* package seems an attractive alternative, for working with expression array data, to the methods that have been discussed here. Its function `MDSplot()` can be used to obtain a low-dimensional representation of the data, based as above on known prior groupings.

Better understanding of gene interactions may suggest better alternatives to using large numbers of features as discriminant variables. Such understanding seems certain to lead also, in the course of time, to more targeted data collection. There will be a greater use of studies that gather data on a small number of features of known relevance to the phenomena under investigation.

12.4 Further reading

There is a large literature on the spectrum of methodologies discussed, and a large and growing range of methodologies that are available in R and in other software. See Venables and Ripley (2002) for a summary overview as of 2002, aimed at practical data analysts, and including the R code needed for examples in the text. Some idea of what is currently available in R can be gleaned by typing `RSiteSearch("multivariate")` and glancing through the long list of hits.

Manly (2005) is a useful brief introduction, though with no reference to methods that have been popular in the data mining literature, and with no mention of R or S-PLUS. Krzanowski (2000) is a comprehensive and accessible overview of the methodology of classical multivariate analysis.

Machine learning and data mining specialists have brought to these problems a research perspective that has been different from that of academic statisticians. Useful insights have been mixed with highly exaggerated claims. See the discussion in Maindonald (2006). Recently, there has been extensive interchange between the two streams of methodological development. Ripley (1996), Hastie *et al.* (2009), and Berk (2008) are important contributions to the ongoing dialogue.

Data visualization systems offer many different tools beyond those that have been described here. Note especially the R package *rggobi*, which provides an R interface to the GGobi system (Cook and Swayne, 2007). Note also the *rgl* package.

Data where there are many times more variables ("features") than observations are a huge challenge for data analysts. There are also new opportunities for gaining information that, in a classical regression or classification problem, is unlikely to be available. Variables that have little or no effect on the outcome variable may nevertheless give important clues about the dependence structure between observations, as described in Leek and storey (2007). Exaggerated predictive accuracy claims are common; see Ambroise and McLachlan (2002) for examples.

Gentleman *et al.* (2005) is a wide-ranging overview both of the computational challenges of processing and analyzing data from genomics and molecular biology, and of the abilities offered by the Bioconductor suite of packages. See also Ewens and Grant (2005).

See the CRAN Task Views for Machine Learning and Multivariate Analysis.

References for further reading

Ambroise, C. and McLachlan, G. J. 2002. Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS*, 99: 6262–6.

Berk, R.A. 2008. *Statistical Learning from a Regression Perspective.*

Cook, D. and Swayne, D. F. 2007. *Interactive and Dynamic Graphics for Data Analysis.*

Ewens, W. J. and Grant, G. R. 2005. *Statistical Methods in Bioinformatics: An Introduction*, 2nd edn.

Gentleman, R., Carey, V., Huber, W., Irizarry, R. and Dudoit, S. 2005. *Bioinformatics and Computational Biology Solutions using R and Bioconductor.*

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd edn.

Krzanowski, W. J. 2000. *Principles of Multivariate Analysis. A User's Perspective*, 2nd edn.

Leek, J. T. and Storey, J. D. 2007. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet* 3(9): e161.

Maindonald, J. H. 2006. Data mining methodological weaknesses and suggested fixes.

Manly, B. F. J. 2005. *Multivariate Statistical Methods. A Primer*, 3rd edn.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks.*

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn.

12.5 Exercises

1. Carry out the principal components analysis of Subsection 12.1.2, separately for males and females. For each of the first and second principal components, plot the loadings for females against the loadings for all data combined, and similarly for males. Are there any striking differences?

2. In the discriminant analysis for the `possum` data (Subsection 12.2.4), determine, for each site, the means of the scores on the first and second discriminant functions. Plot the means for the second discriminant function against the means for the first discriminant function. Identify the means with the names of the sites.

3. The data frame `possumsites` (*DAAG*) holds latitudes, longitudes, and altitudes, for the seven sites. The following code, which assumes that the *oz* package is installed, locates the sites on a map that shows the eastern Australian coastline and nearby state boundaries.

```
library(DAAG); library(oz)
oz(sections=c(3:5, 11:16))
with(possumsites, points(latitude, longitude))
posval <- c(2, 4, 2, 2, 4, 2, 2)
with(possumsites,
     text(latitude, longitude, row.names(possumsites), pos=posval))
```

Do the site means that were calculated in Exercise 2 relate in any obvious way to geographical position, or to altitude?

4. Determine two-dimensional representations of the data in the `painters` data frame (*MASS*) using (1) classical metric scaling; (2) Sammon scaling; (3) Kruskal's non-metric scaling. On each graph show the school to which the painter belonged.

5. Create a version of Figure 12.4B that shows the discriminant line. In the example of Subsection 12.2.1, investigate whether use of `logpet`, in addition to `logwid` and `loglen`, improves discrimination.

6.* The data set `leafshape` has three leaf measurements – `bladelen` (blade length), `bladewid` (blade width), and `petiole` (petiole length). These are available for each of two plant architectures, in each of six locations. (The data set `leafshape17` that we encountered in Subsection 12.2.1 is a subset of the data set `leafshape`.) Use logistic regression to develop an equation for predicting architecture, given leaf dimensions and location. Compare the alternatives: (i) different discriminant functions for different locations; (ii) the same coefficients for the leaf shape variables, but different intercepts for different locations; (iii) the same coefficients for the leaf shape variables, with an intercept that is a linear function of latitude; (iv) the same equation for all locations. Interpret the equation that is finally chosen as discriminant function.

7. The data frame `Vehicle` (*mlbench*) has values of 18 features that were extracted from the images of silhouettes of four different "Corgie" model vehicles.

   – Compare the performance of `lda()`, `qda()`, and `randomForest()` in classifying the vehicle types.

```
'confusion' <-
  function(actual, predicted, digits=4){
    tab <- table(actual, predicted)
    confuse <- apply(tab, 1, function(x)x/sum(x))
    print(round(confuse, digits))
    acc <- sum(tab[row(tab)==col(tab)])/sum(tab)
    invisible(print(c("Overall accuracy" = round(acc,digits))))
  }
library(MASS); library(DAAG); library(randomForest)
library(mlbench)
data(Vehicle)
lhat <- lda(Class ~ ., data=Vehicle, CV=TRUE)$class
qhat <- lda(Class ~ ., data=Vehicle, CV=TRUE)$class
confusion(Vehicle$Class, lhat)
confusion(Vehicle$Class, qhat)
randomForest(Class ~ ., data=Vehicle, CV=TRUE)
```

   – What proportion of the trace do the first two linear discriminants explain? (For this, refit with `CV=FALSE`.)

   – Plot the first discriminant function against the second discriminant function, adding also density contours. The function `quickplot()` from the *ggplot2* package is an easy way to do this:

```
Vehicle.lda <- lda(Class ~ ., data=Vehicle)
twoD <- predict(Vehicle.lda)$x
qplot(twoD[,1], twoD[,2], color=Vehicle$Class,
      geom=c("point","density2d"))
```

What hints does the plot give that might explain the difference between the `lda()` and `qda()` results?

8. Run the following code (it will require a live internet connection):

```
library(ape); library(MASS)
webpage <-
  "http://evolution.genetics.washington.edu/book/primates.dna"
primates.dna <- read.dna(webpage)
 # Alternatively, get primates.dna from the DAAGbio package
primates.dist <- dist.dna(primates.dna, model="F84")
primates.cmd <- cmdscale(primates.dist)
eqscplot(primates.cmd)
rtleft <- c(4,2,4,2)[unclass(cut(primates.cmd[,1], breaks=4))]
text(primates.cmd, labels=row.names(primates.cmd), pos=rtleft)
```

Do the following calculations and comment on the result:

```
d <- dist(primates.cmd)
sum((d-primates.dist)^2)/sum(primates.dist^2)
```

9. Run the following code:

```
library(DAAG)
pacific.dist <- dist(x = as.matrix(rockArt[-c(47, 54, 60, 63, 92),
                                           28:641]), method = "binary")
sum(pacific.dist==1)/length(pacific.dist)
plot(density(pacific.dist, to = 1))
## Check that all columns have at least one distance < 1
symmat <- as.matrix(pacific.dist)
table(apply(symmat, 2, function(x) sum(x<1)))
pacific.cmd <- cmdscale(pacific.dist)
pacific.sam <- sammon(pacific.dist)
```

Why were rows 47, 54, 60, 63 and 92 omitted? Compare the plot from `pacific.cmd` with that from `pacific.sam$points`. Why are they so different?

13

Regression on principal component or discriminant scores

Dimension reduction techniques reduce the number of candidate explanatory variables. Perhaps best known is the replacement of a large number of candidate explanatory variables by the first few principal components. The hope is that they will adequately summarize the information in the candidate explanatory variables. In favorable circumstances, simple modifications of the components will give new variables that are readily interpretable, but this is not always the case.

Propensity scores, often simply called propensities, may be helpful where a response is compared between two groups – a control and a treatment group – that have not been assigned randomly. The response may for example, in a medical context, be death rate in some interval of time. Variables that are not of direct interest, but which may in part explain any differences between the two groups, are commonly known as *explanatory variables*. Results from such analyses are likely to be suggestive rather than definitive, irrespective of the methodology used to account for explanatory variable effects.

Propensities aim to capture, in a single variable, the explanatory variable effects that are important in accounting for differences between two groups. The propensity score, commonly derived from a discriminant analysis, then becomes the only explanatory variable in the regression calculation.

Other types of ordination scores may be used in place of principal component scores. There are a variety of other possibilities.

13.1 Principal component scores in regression

The data set `socsupport` has the following columns:

1. `gender: male or female`
2. `age: 18-20, 21-24, 25-30, 31-40, 40+`
3. `country: Australia, other`
4. `marital: married, single, other`
5. `livewith: alone, friends, parents, partner, residences, other`
6. `employment: full-time, part-time, govt assistance, parental support, other`
7. `firstyr: first year, other`
8. `enrolment: full-time, part-time, blank`

9 10. `emotional, emotionalsat: availability of emotional support, and associated satisfaction (5 questions each)`

```
11  12. tangible, tangiblesat: availability of tangible support
    and associated satisfaction (4 questions each)
13  14. affect, affectsat: availability of affectionate support
    sources and associated satisfaction (3 questions each)
15  16. psi: psisat: availability of positive social interaction
    and associated satisfaction (3 questions each)
17. esupport: extent of emotional support sources (4 questions)
18. psupport: extent of practical support sources (4 questions)
19. socsupport: extent of social support sources  (4 questions)
20. BDI: Score on the Beck depression index (total over
    21 questions)
```

The Beck depression index (`BDI`) is a standard psychological measure of depression (see for example Streiner and Norman, 2003). The data are from individuals who were generally normal and healthy. One interest was in studying how the support measures (columns 9–19 in the data frame) may affect `BDI`, and in what bearing the information in columns 1–8 may have. Pairwise correlations between the 11 measures range from 0.28 to 0.85. In the regression of `BDI` on all of the variables 9–19, nothing appears significant, though the F-statistic makes it clear that, overall, there is a statistically detectable effect. It is not possible to disentangle the effects of these various explanatory variables. Attempts to take account of variables 1–8 will only make matters worse. Variable selection has the difficulties that we noted in Chapter 6. In addition, any attempt to interpret individual regression coefficients focuses attention on specific variables, where a careful account will acknowledge that we observe their combined effect.

Hence the attraction of a methodology that, prior to any use of regression methods, has the potential to reduce the 11 variables to some smaller number of variables that together account for the major part of the variation. Here, principal components methodology will be used. A complication is that the number of questions whose scores were added varied, ranging from 3 to 5. This makes it more than usually desirable to base the principal components calculation on the correlation matrix.

Here now is a summary of the steps that have been followed, to obtain the results that will be described:

1. Following a principal components calculation on variables 9–19, we obtained scores for the first six principal components.
2. The six sets of scores were then used as six explanatory variables, in a regression analysis that had BDI as the response variable. The first run of this regression calculation identified an outlier, which was then omitted and the regression calculation repeated.
3. The regression output suggested that only the first of the variables used for the regression, i.e., only the principal component scores for the first principal component, contributed to the explanation of BDI. Compare $p = 0.00007$ for scores on the first component with p-values for later sets of scores, all of which have $p > 0.05$.
4. It was then of interest to examine the coefficients or loadings of the first principal component, to see which of the initial social support variables was involved.

Code to do the initial analysis, then presenting a scatterplot matrix of the scores on the first three principal components, is:

```
## Principal components: data frame socsupport (DAAG)
ss.pr1 <- princomp(as.matrix(na.omit(socsupport[, 9:19])), cor=TRUE)
pairs(ss.pr1$scores[, 1:3])
sort(-ss.pr1$scores[,1], decr=TRUE)[1:10]   # Note the outlier
## Alternative to pairs(), using the lattice function splom()
splom(~ss.pr1$scores[, 1:3])
```

The name given with the point that we have identified as an outlier is "36", which is the row name in the initial file. We omit this point and repeat the calculation.

```
not.na <- complete.cases(socsupport[, 9:19])
not.na[36] <- FALSE
ss.pr <- princomp(as.matrix(socsupport[not.na, 9:19]), cor=TRUE)
```

The output from `summary()` is:

```
> summary(ss.pr)     # Examine the contributions of the components
Importance of components:
                        Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
Standard deviation      2.394  1.219  1.137 0.8448 0.7545  0.695
Proportion of Variance  0.521  0.135  0.117 0.0649 0.0517  0.044
Cumulative Proportion   0.521  0.656  0.773 0.8383 0.8901  0.934
                        Comp.7 Comp.8 Comp.9 Comp.10 Comp.11
Standard deviation     0.4973 0.4561 0.3595 0.29555 0.23189
Proportion of Variance 0.0225 0.0189 0.0118 0.00794 0.00489
Cumulative Proportion  0.9565 0.9754 0.9872 0.99511 1.00000
```

We now regress `BDI` on the first six principal components. Because the successive columns of scores are uncorrelated, the coefficients are independent. Extraneous terms that contribute little except noise will have little effect on residual mean square, and hence to the standard errors. Thus, there is no reason to restrict the number of terms that we choose for initial examination. The coefficients in the regression output are:

```
> ss.lm <- lm(BDI[not.na] ~ ss.pr$scores[, 1:6], data=socsupport)
> summary(ss.lm)$coef
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                 10.461      0.893  11.709 3.49e-19
ss.pr$scores[, 1:6]Comp.1    1.311      0.373   3.513 7.23e-04
ss.pr$scores[, 1:6]Comp.2   -0.396      0.733  -0.540 5.91e-01
ss.pr$scores[, 1:6]Comp.3    0.604      0.786   0.768 4.45e-01
ss.pr$scores[, 1:6]Comp.4    1.425      1.058   1.347 1.82e-01
ss.pr$scores[, 1:6]Comp.5    2.146      1.184   1.812 7.36e-02
ss.pr$scores[, 1:6]Comp.6    1.288      1.285   1.003 3.19e-01
```

Components other than the first do not make an evident contribution to prediction of BDI. We now examine the loadings for the first component:

```
> ss.pr$loadings[, 1]
   emotional emotionalsat     tangible  tangiblesat       affect
```

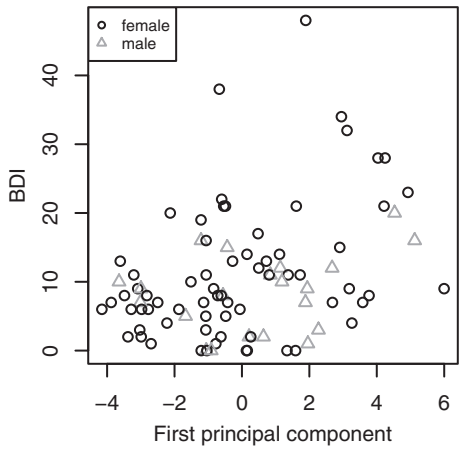


Figure 13.1 Plot of BDI against scores on the first principal component.

```
    -0.320       -0.298       -0.247       -0.289       -0.307
affectsat          psi       psisat     esupport     psupport
    -0.288       -0.363       -0.332       -0.289       -0.285
socsupport
    -0.285
```

The first component is like an average of the 11 measures.[1] A further step is then to plot `BDI` against the scores on the first principal component, using different colors and/or different symbols for females and males. This should be repeated for each of the other seven factors represented by columns 1–8 of the data frame `socsupport`. Figure 13.1 does this for the factor `gender`.[2]

Two observations seem anomalous, with BDI indices that are high given their scores on the first principal component. Both are females. We leave it as an exercise for the reader to recalculate the principal components with these points omitted, and repeat the regression.

Regression on principal component scores has made it possible to identify a clear effect from the social support variables. Because we have regressed on the principal components, it is not possible to ascribe these effects, with any confidence, to individual variables. The attempt to ascribe effects to individual social support variables, independently of other support variables, may anyway be misguided. It is unlikely to reflect the reality of the way that social support variables exercise their effects.

[1] The vector of loadings is unique up to multiplication by -1; the presence of negative signs here is due to the nature of the algorithm.

[2]
```
## Plot first principal componenet score against BDI
attach(socsupport)
plot(BDI[not.na] ~ ss.pr$scores[ ,1], col=as.numeric(gender[not.na]),
     pch=as.numeric(gender[not.na]), xlab ="1st principal component",
     ylab="BDI")
topleft <- par()$usr[c(1,4)]
legend(topleft[1], topleft[2], col=1:2, pch=1:2, legend=levels(gender))
detach(socsupport)
```

13.2* Propensity scores in regression comparisons – labor training data

A propensity is a measure, determined by explanatory variable values, of the probability that an observation will fall in the treatment rather than in the control group. Various forms of discriminant analysis may be used to determine scores. The propensity score is intended to account for between-group differences that are not due to the effect under investigation. If there is substantial overlap between propensity scores for the different groups, then comparison of observations within the approximate region of overlap may be reasonable, but using the propensity score to adjust for differences that remain. See Rosenbaum and Rubin (1983) for further comments on the methodology.

We will first describe the data, then investigate more conventional regression approaches to the analysis of these data, then investigate the use of propensity scores. The results highlight the difficulty in reaching secure conclusions from the use of observational data.

The labor training data

Data are from an experimental study, conducted under the aegis of the US National Supported Work (NSW) Demonstration Program, of individuals who had a history of employment and related difficulties. Over 1975–1977, an experiment randomly assigned individuals who met the eligibility criteria either to a treatment group that participated in a 6–18 months training program, or to a control group that did not participate.

The results for males, because they highlight methodological problems more sharply, have been studied more extensively than the corresponding results for females. Participation in the training gave an increase in male 1978 earnings, relative to those in the control group, by an average of $886 [SE $472].

Can the same results be obtained from data that matches the NSW training group with a non-experimental control group that received no such training? Lalonde (1986) and Dehejia and Wahba (1999) both investigated this question, using two different non-experimental control groups. These were:

1. The Panel Study of Income Dynamics (PSID: 2490 males, data in `psid1`, filtered data in `psid2` and `psid3`).
2. Westat's Matched Current Population Survey – Social Security Administration file (CPS: 16 289 males, data in `cps1`, filtered data in `cps2` and `cps3`).

Variables are:

```
trt (0 = control 1=treatment)
age (years)
educ (years of education)
black (0=white 1=black)
hisp (0=non-hispanic 1=hispanic)
marr (0 = not married 2=married)
nodeg (0=completed high-school 1=dropout); i.e. educ <= 11
re74 (real earnings in 1974; available for a subset of the
     experimental data only)
re75 (real earnings in 1975)
re78 (real earnings in 1978)
```

Table 13.1 *Proportion in the stated category, for each of the data sets indicated. Proportions for the experimental data are in the final two lines of the table.*

			Proportion			
	Black	Hispanic	Married	Dropout	re75 > 0	re78 > 0
psid1	0.25	0.03	0.87	0.31	0.90	0.89
psid2	0.39	0.07	0.74	0.49	0.66	0.66
psid3	0.45	0.12	0.70	0.51	0.39	0.49
cps1	0.07	0.07	0.71	0.30	0.89	0.86
cps2	0.11	0.08	0.46	0.45	0.82	0.83
cps3	0.20	0.14	0.51	0.60	0.69	0.77
nsw-ctl	0.80	0.11	0.16	0.81	0.58	0.70
nsw-trt	0.80	0.09	0.17	0.73	0.63	0.77

Observe that `trt`, `black`, `hisp`, `marr`, and `nodeg` are all binary variables. Here, they will be treated as dummy variables. In the language of Section 7.1, observations that have the value zero are the baseline, while the coefficient for observations that have the value 1 will give differences from this baseline. (For `marr`, where values are 0 or 2, the coefficient for observations that have the value 2 will be half the difference from the baseline.)

Note that `nodeg` is a categorical summary of the data in `educ`. It will not be used, additionally to `educ`, as an explanatory variable in the various analyses.

Summary information on the data

Table 13.1 has summary information on proportions on discrete categories that are of interest.[3] Information on `re74` is complete for the non-experimental sets of control data, but incomplete for the experimental data. We will examine the issue of how to handle `re74` below.

Notice the big differences, for `black`, `marr`, and `nodeg` (dropout), between the non-experimental controls (first six lines) and both sets of experimental data (final two lines). Even in the filtered data sets (`psid2`, `psid3`, `cps2`, and `cps3`), the differences are

[3]
```
showprop <-
  function(dframe=psid1, facCols=4:7, zeroCols=9:10){
    info <- numeric(length(facCols)+length(zeroCols))
    info[1:length(facCols)] <- sapply(dframe[,facCols], function(x){
      z <- table(x); z[2]/sum(z)})
    info[-(1:length(facCols))] <- sapply(dframe[,zeroCols], function(x)
      sum(x>0)/sum(!is.na(x)))
    info
  }
## Create matrix to hold resuilt
propmat <- matrix(0, ncol=6, nrow=8)
dimnames(propmat) <-
  list(c("psid1", "psid2", "psid3", "cps1", "cps2", "cps3",
         "nsw-ctl", "nsw-trt"), names(nswdemo)[c(4:7, 9:10)])
## Run function
for(k in 1:8){
  dframe <- switch(k, psid1, psid2, psid3, cps1, cps2, cps3,
                   subset(nswdemo, trt==0), subset(nswdemo, trt==1))
  propmat[k,] <- showprop(dframe)
}
```

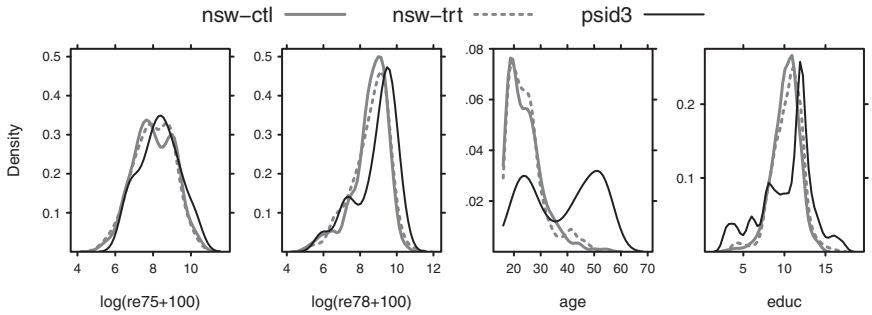


Figure 13.2 Overlaid density plots, comparing treatment groups with the experimental control data in `nswdemo` and with the non-experimental control data in `psid3`, for the variables `age` `educ`, `log(re75+30)`, and `log(re78+30)`.

substantial. The big changes that the filtering has made to the proportion with non-zero earnings is worrying. Notice particularly the huge differences between `psid3` and `psid1`, both for `re75` and `re78`.

For those who did earn an income, how do the distributions compare? The very heavy tails in the distributions of `re75` and `re78` make use of a logarithmic transformation desirable. Figure 13.2 compares the distributions of values, in the control and treatment groups, for the explanatory variables `age`, `educ`, `log(re75+30)`, and `log(re78+30)`. The offset of 30 is around half the minimum non-zero value for each of these variables, in the combined data. Plate 7 is an extended version of Figure 13.2 that has comparisons with all of the candidate sets of control data.

Examination of Figure 13.2, and of the additional comparisons in Plate 7, makes it clear that there are large differences between treatment and controls, whichever set of non-experimental controls is chosen.

The distributions of non-zero values of $\log(re78 + 30)$ are almost identical between experimental treated and control observations, just as similar as for $\log(re75 + 30)$. A more careful comparison will use QQ-plots. The comparison can be repeated with several bootstrap samples, as a check that such small differences as are apparent are not maintained under bootstrap sampling. This is pursued in the exercises at the end of the chapter. We will later check whether the differences that are apparent between non-experimental controls and treatment are maintained after a propensity score adjustment.

Plate 8 shows the scatterplot matrix, again for the data set that combines the `psid1` control data with the experimental treatment data, for the same variables as shown in Figure 13.2. Slightly simplified code is:

```
vnames <- c("trt","educ","age","re75","re78")
nsw <- rbind(psid1, subset(nswdemo, trt==1))
## Check minimum non-zero values of re75 and re78
round(sapply(nsw[,c("re75","re78")], function(x)unique(sort(x))[2]))
nsw[,c("re75","re78")] <- log(nsw[,c("re75","re78")] + 30)
lab <- c(vnames[2:3], paste("log\n", vnames[-(1:3)], "+", 30))
nsw$trt <- factor(nsw$trt, labels=c("Control (psid1)","Treatment"))
splom(~ nsw[,vnames[-1]], type=c("p","smooth"), groups=nsw$trt,
      varnames=lab, auto.key=list(columns=2))
```

13.2.1 Regression comparisons

One possibility is to use regression methods directly to compare the two groups, with variables other than `re78` used as explanatory variables. The nature of the data does however raise serious issues, for its use for this purpose.

Issues for the use of regression methods

The following points require consideration:

- Continuous variables almost certainly require some form of non-linear transformation. Regression splines may be a reasonable way to go.
- Should interaction terms be included?
- The large number of explanatory variables, and interactions if they are included, complicates the use of diagnostic checks.
- A substantial proportion of the values of `re78` are zero. The distribution of non-zero values of `re78` is highly skew, in both of the experimental groups (treatment and non-treatment), and in all of the non-experimental controls. A consequence is that the regression results will be strongly influenced by a small number of very large values. A $\log(re78 + 30)$ transformation (the choice of offset, in a range of perhaps 20–200, is not crucial) gives values that may more reasonably be used for regression, however. (In spite of the evident skewness, both Lalonde (1986) and Dehejia and Wahba (1999) used `re78` as the response variable in their analyses.)
- In the experimental data, almost 40% of values of the explanatory variable `re74` are missing. It is then necessary to ask whether these are "missing at random", or whether there is a pattern in the missingness. An indication that values of `re74` may not be missing at random is that its minimum value in the experimental data is 445 (dollars), which is close to 6 times the minimum of 74 for `re75` and almost 10 times the minimum of 45 for `re78`. Perhaps information on 1974 income was more readily available for participants who for most of 1974 held one steady job, or a small number of steady jobs. In the analysis below, a factor will be created from `re74` that has the levels: no income, some income, and income status unknown.
- Control and training groups can be made more comparable by some initial filtering of the data, on values of the explanatory variables. Inevitably, the choice of filtering mechanism and extent of filtering will be somewhat arbitrary, and filtering may introduce its own biases.
- Explanatory variables must both model within-group relationships acceptably well and model between-group differences acceptably well. These two demands can be in conflict.

Taken together, these points raise such serious issues that results from any use of regression methods have to be treated skeptically.

The complications of any use of regression analyses, and the uncertainties that remain after analysis, are in stark contrast to the relative simplicity of analysis for the experimental data. Experimental treatment and control distributions can be compared directly

and (assuming that the randomization was done properly) with confidence, without the complications that arise from the attempt to adjust for explanatory variable effects.

Regression calculations

As there may be information on whether or not `re74` is known, and on whether known values are non-zero, it seems useful to distinguish three categories – no income in 1974, some income in 1974, and details of income not known. Hence an argument for the use of a factor `fac74` that is derived thus:

```
nsw$fac74 <- with(nsw, factor(re74>0, exclude=NULL))
table(nsw$fac74)     # Check the order of the levels
levels(nsw$fac74) <- c("0","gt0","<NA>")
```

In the following analysis, two degrees of freedom have been allowed for a regression using a natural spline basis for each of $\log(re75 + 30)$, `age`, and `educ`. Here is a function that can be used for the calculations. The function has an argument that controls whether or not to apply a logarithmic transformation to `re78`.

```
nswlm <-
  function(control=psid1, df1=2, log78=TRUE, offset=30, printit=TRUE){
    nsw0 <- rbind(control, subset(nswdemo, trt==1))
    nsw0$fac74 <- factor(nsw0$re74>0, exclude=NULL)
    levels(nsw0$fac74) <- c("0","gt0","<NA>")
    if(log78) nsw.lm <- lm(log(re78+offset) ~ trt + ns(age,df1) +
                           ns(educ,df1) + black + hisp + fac74 +
                           ns(log(re75+offset),df1), data=nsw0) else
    nsw.lm <- lm(re78 ~ trt + ns(age,df1) + ns(educ,df1) + black +
                           hisp + fac74 + ns(log(re75+offset),df1),
                           data=nsw0)
    if(printit) print(summary(nsw.lm))
    trtvec <- unlist(summary(nsw.lm)$coef["trt", 1:2])
    trtEst <- c(trtvec[1], c(trtvec[1]+trtvec[2]*c(-1.96,1.96)))
    if(log78) {
      trtEst <- c(trtEst[1], exp(trtEst[1]), exp(trtEst[-1]))
      names(trtEst)=c("Est.","exp(Est.)","CIlower","CIupper")
    } else
    names(trtEst)=c("Est.","CIlower","CIupper")
    if(printit) print(trtEst)
    invisible(list(obj=nsw.lm, est=trtEst))
  }
## Try for example
library(splines)
nsw.lm1 <- nswlm(control=psid1)$nsw.lm
nswlm(control=subset(nswdemo, trt=0))
nswlm(control=psid1, log78=FALSE)
for (z in list(psid1,psid2,psid3,cps1,cps2,cps3))
  print(nswlm(control=z, printit=FALSE)$est)
```

Use of `termplot()` with the arguments `partial=TRUE` and `smooth=panel.smooth` suggests that the default numbers of degrees of freedom are adequate or more

than adequate. The coefficients of other terms in the equation are not highly sensitive to the number of degrees of freedom allowed.

The following table summarizes results, showing how they depend on the choice of control group:

Control used	Estimate of treatment effect	95% CI
`psid1`	exp(0.99) = 2.7	(1.9, 3.7)
`psid2`	exp(0.61) = 1.8	(1.0, 3.4)
`psid3`	exp(0.92) = 2.5	(1.2, 5.2)
`cps1`	exp(0.85) = 2.3	(1.7, 3.1)
`cps2`	exp(0.47) = 1.6	(1.1, 2.4)
`cps3`	exp(0.49) = 1.6	(0.96, 2.8)
`subset(nswdemo, trt=0)`	exp(0.35) = 1.4	(1.1, 1.9)

These results vary widely, but do all point in the same direction as the experimental comparison in the final row. It is instructive to rerun the above calculations with `log78=FALSE`. The different results do not now all point in the same direction. The likely reason is that a few very large values of `re78` now have high leverage and a large influence. (Exercise 5 at the end of the chapter is designed to check this out.)

13.2.2 A strategy that uses propensity scores

A propensity "score" is a single variable whose values characterize the difference between the control and treatment groups. Importantly, the score is designed to model only between-group differences; it does not model within-group differences. Use of a single propensity score in place of many explanatory variables facilitates the use of standard checks to investigate whether the propensity score effect is plausibly linear. There is just one explanatory variable to investigate, rather than the complicated and often unfruitful task of carrying out checks on several explanatory variables.

For the analyses described here, we will start by using control observations from the data set `psid1`. Analyses that start by using control observations from one of the other data sets are left as an exercise for the reader.

Propensity scores will be derived from a discriminant analysis that uses the `randomForest()` function, from the package of the same name. Advantages of this approach are that prior transformation of variables is unnecessary, assumptions about the form of model are minimal, and there is automatic allowance for interactions. The extent of prior filtering of observations should not unduly matter.

We will however check out, for comparison, scores that arise from use of the function `lda()` (*MASS* package). It will turn out that `lda()` is similarly effective, as measured by predictive accuracy, in distinguishing the control and treatment groups. The key point is that it does no better than `randomForest()`, and thus that there is no reason to prefer the `lda` scores.

Either method yields, for each observation, an estimated probability p that the observation is from the treatment group. A convenient choice of propensity score is then $\log(p/(1-p))$.

The analysis will then replace the explanatory variables by a single propensity score. This is justifiable on theoretical grounds if the distribution of the explanatory variables is, conditional on the propensity score, the same for treatment and control observations. Checks can be performed to determine whether this assumption is plausible. If these checks fail, the analysis might still give reasonable results, but the theory does not give good grounds for confidence.

Derivation and investigation of scores

We now derive propensity scores. We convert `re74` to a factor with three levels – `0` (no income in 1974), `gt0` (income in 1974), and `<NA>` (income status in 1974 not known). The observations for which 1974 income information is available may be a biased selection, and it seems safest to use information on `re74` as a coarse indicator only.

```
## Use the dataset nsw that combines psid1 with exptl trt obsns
nsw.rf <- randomForest(trt ~ ., data=nsw[, -c(7:8,10)],
                       sampsize=c(297,297))
## NB: Use of equal bootstrap sample sizes (= 297 = number of
## treatment observations) gives the two groups equal prior weight.
```

We can check model accuracy

```
> nsw.rf
. . .
        OOB estimate of  error rate: 4.52%
Confusion matrix:
    0     1 class.error
0 2381  109      0.0438
1   17  280      0.0572
```

The random forest calculation should be rerun several times. We have found error rates that vary, over four runs, between 4.38% and 4.56%. These are the error rates that would be expected from a separate random sample from the same population.

The following fits a logistic regression model:

```
> library(MASS)
> library(splines)
> nsw.lda <- lda(trt ~ ns(age,2) + ns(educ,2) + black + hisp +
+                fac74 +  ns(log(re75+30),3),
+                CV=TRUE, prior=c(.5,.5), data=nsw)
> tab <- table(nsw.lda$class, nsw$trt)
> 1 - sum(tab[row(tab)==col(tab)])/sum(tab)
[1] 0.042
```

The `lda()` cross-validation error rate is very similar to that for `randomForest()`. The simple `lda()` model that does not allow for interaction effects may be adequate. The regression spline terms in the `lda` model seem to account for most of the non-linearity in the explanatory variables.

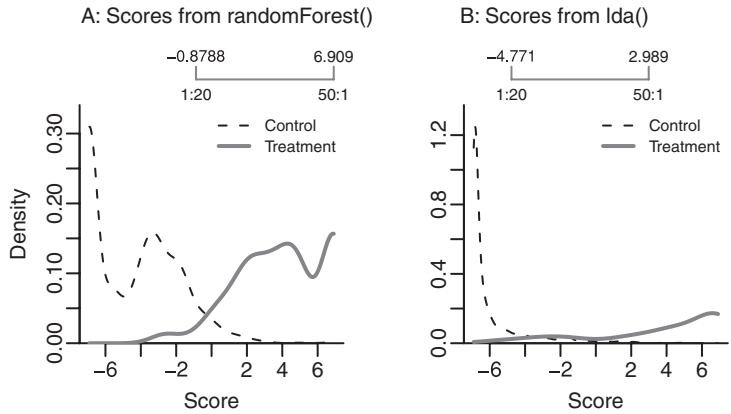


Figure 13.3 Panel A shows density plots of scores ($\log((p + 0.001)/(1 + 0.001 - p))$, where p is predicted value) from the object `sc.rf`, separately for control and treatment groups. Panel B is for scores, calculated similarly, from `sc.lda`. The ranges shown are ranges of relative numbers.

Here is code that calculates the scores and compares the densities between the control and treatment groups, as shown in Figure 13.3:

```
logit <- function(p, offset=0.001)log((p+offset)/(1+offset-p))
tnum <-  unclass(nsw$trt)
## NB: Derive scores by a logit transform of probabilities
sc.rf <- logit(predict(nsw.rf, type="prob")[,2])
overlapDensity(sc.rf[tnum==1], sc.rf[tnum==2], ratio=c(1/20, 50))
nsw.lda <- lda(trt ~ ns(age,2) + ns(educ,2) + black + hisp + fac74 +
               ns(log(re75+30),3), prior=c(.5,.5), data=nsw)
sc.lda <- logit(nsw.lda$posterior[,2])
overlapDensity(sc.lda[tnum==1], sc.lda[tnum==2], ratio=c(1/20, 50),
               compare.numbers=TRUE, plotval="Density")
```

The bulk of the control observations lie, in each instance, off to the left of the minimum score for which the ratio of treatment frequency to control frequency reached $\frac{1}{20} = 0.05$. For use of the `randomForest` scores, choosing observations with a score of more than -1.5 will retain approximately equal numbers (307/289, varying from run to run) of control and treatment scores. Without some such filtering, there may be undue leverage from the very large proportion of control observations that have large negative scores, where there are no treatment observations. Even modest filtering of observations with high scores (e.g., insist on a ratio of less than 50 treatment to one control observation) will filter out a large fraction of the treatment observations, and we keep such filtering to a minimum.

Now recalculate the propensity scores, at the same time calculating proximities between observations. The proximity between any pair of observations is the proportion of trees, out of the total number of trees (by default, 500), where the two observations appear together at the same terminal node.

```
nswa <- nsw[sc.rf>-1.5, ]
nswa.rf <- randomForest(trt ~ ., data=nswa[, -c(7:8,10)])
proba.rf <- predict(nswa.rf, type="prob")[,2]
sca.rf <- logit(proba.rf)
```

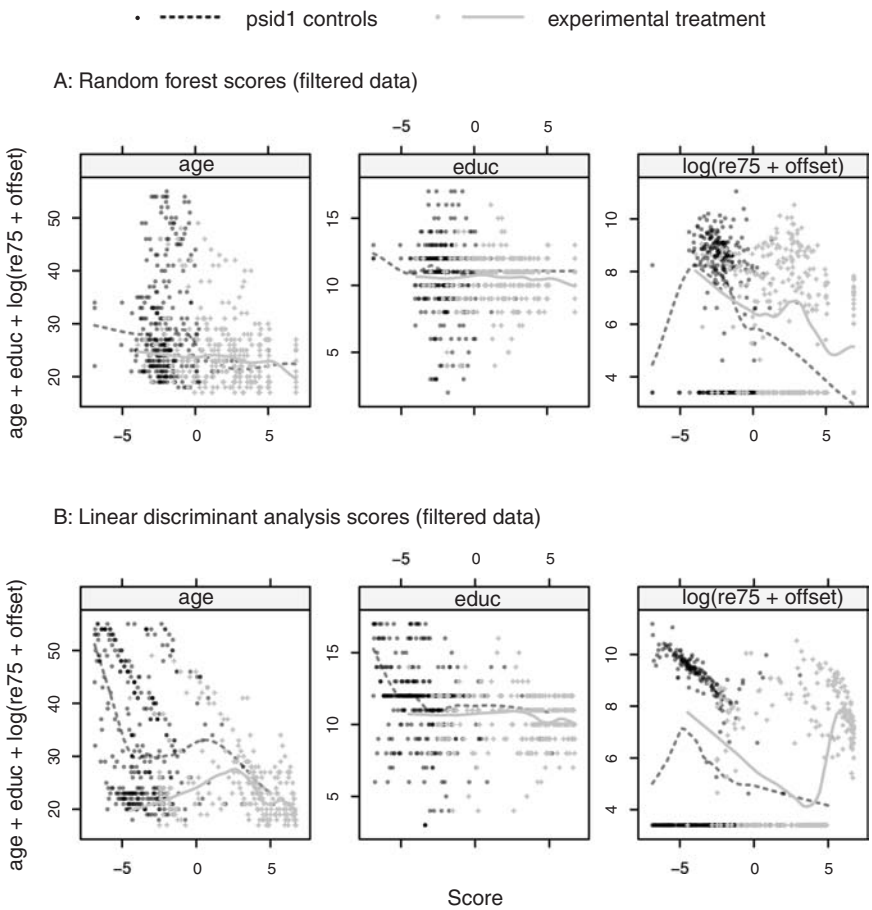


Figure 13.4 These plots are designed as a check whether, in each case, the distribution of the explanatory variable is, conditional on the score, similar for treated and controls. Panel A shows scores from `randomForest`, while panel B shows scores from `lda()`. Plate 9 is a color version that shows, also, the distribution of new `randomForest` scores obtained by refitting the model to data for which the scores shown in A were at least −1.5.

For `lda` scores, choosing observations with a score of more than −4 would retain somewhat more treatment than control scores (329/281).

Checks on the propensity scores

Is the distribution of the explanatory variables, conditional on the propensity score, the same for treatment and control? This can be checked for each individual explanatory variable. As interactions have seemed unimportant in determining the propensities, this may be enough. Figure 13.4 and and Plate 9 provide a visual check. Code that gives a close equivalent of Figure 13.4A is:

```
xyplot(age + educ + log(re75+30) ~ sca.rf, groups=trt, layout=c(3,1),
       data=nswa, type=c("p","smooth"), span=0.4, aspect=1,
```

```
par.settings=simpleTheme(lwd=c(2,1.5), col=c("gray", "black"),
  pch=c(20,3), cex=0.5, lty=c("solid","21")),
scales=list(y=list(relation="free"), tck=0.5),
auto.key=list(columns=2, points=TRUE, lines=TRUE,
  text=c("psid1 controls", "experimental treatment")),
xlab="Scores, derived using randomForest()")
```

For Figure 13.4B, replace `sca.rf` by `sca.lda`, obtained by linear discriminant calculations that use the subset of `nsw` for which `nsw.lda` is at least -4.

Conditional on the scores, both sets of panels show substantial differences for `age` and for `log(re75+30)`. The `randomForest` scores seem however preferable. In A (`randomForest()`), removal of points with very low scores (less than -1.5) has largely dealt with the most serious differences. In B (`lda()`) there is, for both of these variables, a large cluster of control points on the right of the plot. For `educ`, differences seem minor, for both sets of scores.

The graphs suggest that the formal requirements of the propensity score theory are in doubt. There are not good grounds for confidence that propensity scores will work well in making the necessary adjustment.

Use of proximities to give a two-dimensional representation

A more global graphical comparison is available by using the proximities from a `randomForest()` discriminant analysis as the basis for an ordination, i.e., for a two-dimensional representation as described in Subsection 12.1.3. Plots are shown for three ranges of scores – low, medium, and high. The text in the panel is labeled according to the equivalent range of probabilities.

Figure 13.5 shows the result. The code is:

```
## Repeat randomForest calculation, now with proximities
nswa.rf <- randomForest(trt ~ ., data=nswa[, -c(7:8,10)],
                        proximity=TRUE)
## Subtract proximities from 1.0, add 0.001, and use as "distances"
# NB: Use of isoMDS() will require all "distances" to be +ve
dmat <- 1-nswa.rf$proximity +0.001
proba.rf <- predict(nswa.rf, type="probability")[,2]
## From "distances", derive an ordination.
pts <- cmdscale(dmat)
ordScores <- isoMDS(dmat, pts)$points
cutpts <- c(0, round(quantile(proba.rf, c(1/3,2/3)), 2), 1).
cutp <- cut(proba.rf, breaks=cutpts, include.lowest=TRUE)
xyplot(ordScores[,2] ~ ordScores[,1]|cutp, groups=nswa$trt,
       xlab="Co-ordinate 1", ylab="Co-ordinate 2",
       auto.key=list(columns=2), aspect=1, layout=c(3,1),
       par.settings=simpleTheme(col=c("black","gray"), pch=c(1,3)))
```

In the sequel, the `randomForest` scores will be used. They do at least as well as the `lda` scores in accounting for differences between the two groups. Conditional on the propensity score, the distribution of the explanatory variables may be rather more similar

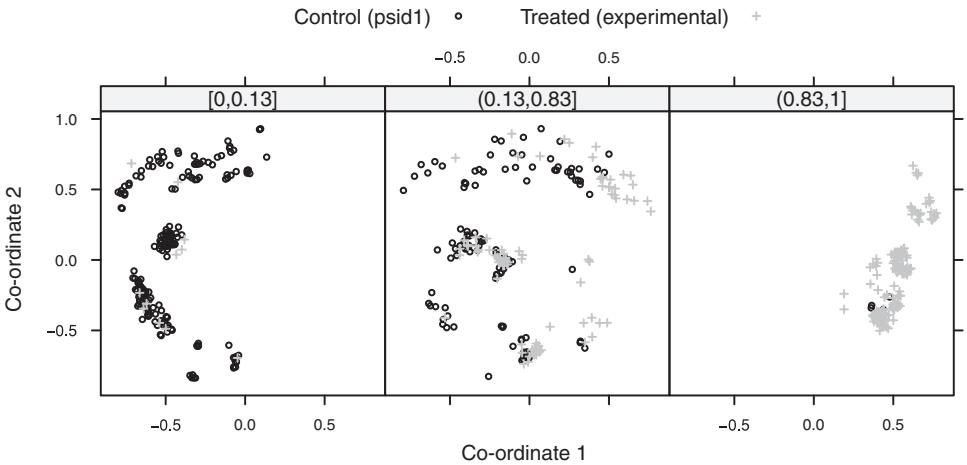


Figure 13.5 These plots are designed as a check whether, in each case, the distribution of the explanatory variables is, conditional on the score from `randomForest()`, similar for treated and controls. They examine a two-dimensional representation that is derived from the propensities. The ranges shown are for the probabilities before use of the logit transformation to give scores. Cutpoints have been chosen so that the three ranges contain an approximately equal number of observations. Note that results will differ somewhat from one run to the next.

between treatment and control than for the `lda` scores. They minimize opportunities for bias such as arise from the assumption, in the `lda` analysis, of a specific form of additive model.

Probability of non-zero earnings – analysis using the scores

The following checks whether there is a detectable training effect on the probability of non-zero earnings:

```
> sca.rf <- logit(sca.rf)
>  rf.glm <- glm(I(re78>0) ~ ns(sca.rf,2)+trt, data=nswa,
+                family=binomial)
>  summary(rf.glm)
. . . .
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.926  -1.363   0.705   0.831   1.313

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                   0.158      0.555    0.28    0.776
ns(sca.rf, 2)1                1.001      1.313    0.76    0.446
ns(sca.rf, 2)2               -1.026      0.492   -2.09    0.037
trttreated (experimental)     0.740      0.305    2.43    0.015
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 691.39  on 595  degrees of freedom
Residual deviance: 676.63  on 592  degrees of freedom
AIC: 684.6

Number of Fisher Scoring iterations: 4
```

The estimate is in line with that from comparing experimental treatment data with experimental controls. Use of the linear discriminant scores yields a result that is even more clearcut.

Distribution of non-zero earnings – analysis using the scores

```
> rf.lm <- lm(log(re78+30) ~ ns(sca.rf,2)+trt, data=nswa,
+               subset = re78>0)
> summary(rf.lm)
. . . .
Residuals:
   Min     1Q Median     3Q    Max
-3.639 -0.441  0.153  0.660  2.695

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                  9.699      0.329   29.47   <2e-16
ns(sca.rf, 2)1              -1.488      0.768   -1.94    0.053
ns(sca.rf, 2)2              -0.432      0.263   -1.64    0.101
trttreated (experimental)   -0.373      0.151   -2.47    0.014

Residual standard error: 0.987 on 433 degrees of freedom
Multiple R-squared: 0.0931,        Adjusted R-squared: 0.0868
F-statistic: 14.8 on 3 and 433 DF,  p-value: 3.37e-09
```

The negative (and statistically significant) treatment estimate contrasts with the result from the experimental data, where the estimated treatment effect is well below the threshold of statistical detectability.

```
> round(summary(lm(log(re78+30) ~ trt, data=nswdemo,
+       subset=re78>0))$coef, 4)
            Estimate Std. Error  t value Pr(>|t|)
(Intercept)   8.5601     0.0578 148.1486   0.0000
trt           0.0021     0.0874   0.0245   0.9804
```

In the absence of the check that the experimental data provides, it would be necessary to treat any of these results with extreme caution. Use of `psid2` or `psid3` (or `cps2` or `cps3`) is not an adequate answer. There are large elements of arbitrariness in the choice of observations to be removed, the filtering leaves data sets that still differ from the experimental treatment data in important respects, and results vary depending on which of these data sets is used as a control.

13.3 Further reading

Streiner and Norman (2003) discuss important issues that relate to the collection and analysis of multivariate data in medicine, in the health social sciences, and in psychology. On the use of propensity scores, see Rosenbaum and Rubin (1983), Rosenbaum (2002). On wider issues with respect to the analysis of observational data, see Rosenbaum (1999, 2002, 2005).

References for further reading

Rosenbaum, P. and Rubin, D. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55.

Rosenbaum, P. R. 1999. Choice as an alternative to control in observational studies. *Statistical Science* 14: 259–78. With following discussion, pp. 279–304.

Rosenbaum, P. R. 2002. *Observational Studies*, 2nd edn.

Rosenbaum, P. R. 2005. Reasons for effects. *Chance* 18: 5–10.

Streiner, D. L. and Norman, G. R. 2003. *Health Measurement Scales. A Practical Guide to their Development and Use*, 3rd edn.

13.4 Exercises

1. Repeat the principal components calculation omitting the points that appear as outliers in Figure 13.1, and redo the regression calculation. What differences are apparent, in loadings for the first two principal components and/or in the regression results?

2. Examine the implications that the use of the logarithms of the income variables in the analysis of the data set `nswpsid1` has for the interpretation of the results. Determine predicted values for each observation. Then exp(predicted values) gives predicted incomes in 1978. Take exp(estimated treatment effect) to get an estimate of the factor by which a predicted income for the control group must, after adding the offset, be multiplied to get a predicted (income+offset) for the treatment group, if explanatory variable values are the same.

3. Investigate the sensitivity of the regression results in Subsection 13.2.2 to the range of values of the scores that are used in filtering the data. Try the effect of including data where: (a) the ratio of treatment to control numbers, as estimated from the density curve, is at least 1:40; (b) the ratio lies between 1:40 and 40; (c) the ratio is at least 1:10.

4. Modify the function `nswlm()` so that use of `fac74` as an explanatory factor is optional. With the `psid3` controls, is use of `fac74` as an explanatory factor justified? What is the effect on the confidence interval for the treatment effect?

5. Subsection 13.2.1 defined a function `nswlm()`, then using it to create a table that gives treatment effect estimates. Rerun the calculations that generated the table entries, now supplying the argument `log 78=FALSE`. Comment on changes in the treatment effect estimates.

14

The R system – additional topics

14.1 Graphical user interfaces to R

The R Commander (*Rcmdr*) will be the main focus, with brief reference to other GUIs. Subsection 1.1.1 mentioned the usefulness of the R Commander for data input. Especially for novices or infrequent users of R, a GUI can be similarly helpful for creating simple graphs, for statistical testing, for simple tabulation and summarization, and for fitting standard models. The final subsection describes how the function `gui()` (*fgui* package) can be used to create simple GUIs.

Some tasks are best done from the command line, and some from a GUI, with the balance likely to change in favor of the command line as familiarity with R increases. The two modes of use can be mixed. All the GUIs discussed here make available the commands used by R, for inspection and/or modification and/or for audit trail purposes. The user can examine the help page for the relevant function(s), modify the code as required, and re-execute it.

In addition to the R Commander, note:

- *JGR* (Java Graphics for R) and the *Deducer* GUI that is designed to work with *JGR*.
- The *rattle* GUI gives access to a range of multivariate graphics, regression and classification routines.
- A more limited alternative to the R Commander is *pmg* (Poor Man's GUI).
- The function (*latticist()*) (*latticist* package) should be invoked with a data frame or table as argument. It then opens a GUI to the *lattice* and *vcd* packages, allowing rapid creation of plots that may be useful in their own right, or may be a first step in creating more carefully honed plots.
- The *playwith* package has extensive features for interacting with graphs, including the addition of annotation. It includes an interface to *latticist*. The *playwith* package is described further in Subsection 15.5.4.

For *rattle*, both the Gtk + system and Glade must be installed. Further brief details are in Subsection 14.1.2. The *playwith* package requires Gtk +.

Installing the R *Commander*

To install the R Commander from the command line, enter:

```
install.packages("Rcmdr", dependencies=TRUE)
```

Among the many dependencies are the graphics packages *rgl* (3D dynamic graphics), *vcd* (visualization of categorical data), and *colorspace* (for generation of color palettes, etc).

14.1.1 The R *Commander's interface – a guide to getting started*

To start the R Commander, start up R and enter:[1]

```
library(Rcmdr)
```

This opens an R Commander script window, with the output window underneath. This window can be closed by clicking on the × in the top left corner. If thus closed, enter `Commander()` to reopen it again later in the session.

From GUI to writing code. The R Commander displays the code that it generates. Users can take this code, modify it, and rerun it. The code can be run either from the R Commander script window or from the R console window (if open).

The active data set. There is at any time a single "active" data set. Start by clicking on the Data drop-down menu. Here are alternative ways to select or create the active data set:

- Click on Active data set, and pick from among data sets, if any, in the workspace.
- Click on Import data, and follow instructions, to read in data from a file. The data set is read into the workspace, at the same time becoming the active data set.
- Click on New data set . . . , then entering data via a spreadsheet-like interface.
- Click on Data in packages, then on Read data from package, then select an attached package and choose a data set from among those included with the package.
- A further possibility is to load data from an R image (`.RData`) file; click on Load data set . . . .

Creating graphs. To draw graphs, click on the Graphs drop-down menu. Then

- Click on Scatterplot . . . to obtain a scatterplot. This uses `scatterplot()` from the *car* package, which is an option-rich interface to functions that are in base graphics.
- Click on X Y conditioning plot . . . for lattice scatterplots and panels of scatterplots.
- Click on 3D graph to obtain a 3D scatterplot, using the R Commander function `scatter3d()` that is an interface to functions in the *rgl* package.

Statistics (& fitting models). Click on the Statistics drop-down menu to get submenus that give summary statistics and/or carry out various statistical tests. This includes (under Contingency tables) tables of counts and (under Means) One-way ANOVA. Also, click here to get access to the Fit models submenu.

***Models.** Click here to extract information from model objects once they have been fitted. (NB: To fit a model, go to the Statistics drop-down menu, and click on Fit models).

[1] At startup, the R Commander may check whether all the suggested packages, needed to use all its features, are available. If some are missing, then the R Commander offers to install them. To install such packages, there must be a live internet connection.

14.1.2 The rattle GUI

To get started, type:

```
library(rattle}
rattle()
```

First click on the radio button that identifies the source of the data – CSV or ARFF (used by some data mining packages); Library (from an R package); an RData file (a `.RData` image file); or ODBC. Click on the relevant button and click on Execute. Selecting CSV and clicking on the Execute tab leads to an offer to load the `audit` sample data set, which can be used for experimenting with *rattle*'s abilities.

Once loaded, information will be displayed on the data columns, with a tentative specification as Input or Target or Risk or Ident or Ignore. This can be changed as required.

Immediately below Execute is a row of tabs: Data (to load data), Explore (for data exploration), ... Model (fit a model), Evaluate (create a new column), and Log (examine code). Thus, to explore the data, click on Explore, click on the type of information required (start maybe with Summary), and click on Execute. To fit a model, click on Model, click maybe on the Forest button, and click on Execute.

14.1.3 The creation of simple GUIs – the fgui package

The *fgui* package allows the ready creation of a GUI to a user function. Here is a simple example, with explanation following, that uses the function `gui()` in this package:

```
library(fgui)
library(DAAG); library(lattice)
## Create function that does the smoothing
fitsmooth <-
  function(form=mdbRain^(1/3) ~ Year, df=bomregions, myspan=0.75)
    print(xyplot(form, data=df, type=c("p","smooth"), span=myspan))
## Create function that sets up a GUI interface to fitsmooth()
callsmooth <- function()gui(func=fitsmooth,
    argSlider=list(myspan=c(0.1,1.5,0.025)),  # start,stop,stepsize
    output=NULL, callback=guiExec)
callsmooth()
## Click on and/or move the slider to display the graph.
```

- The first argument to `gui()` is the function `fitsmooth` that is to be executed.
- The argument `argSlider` was used to specify values, controlled by a slider, for `myspan`. Text boxes are automatically created for any arguments to `func`, here `form` and `df`, that are not otherwise specified. If these text boxes are left blank, the defaults `form=mdbRain^(1/3)` and `df=bomregions` will be used.
- The argument `output=NULL` suppresses the text box that would otherwise be created to hold function output. (Here, there is no output to hold.)
- The argument `callback=guiExec` ensures that the function `fitsmooth()` is re-executed whenever one of its arguments is changed, e.g., by moving the slider.
- Click OK to terminate the interaction.

14.2 Working directories, workspaces, and the search list

14.2.1 The search path*

At any time in an R session, R has a set of names of repositories ("databases") where it looks, in order, for objects that are required for a command line evaluation. The workspace (`.Globalenv`) is first on this *search list*. Thus, `ls(pos=1)` is equivalent to `ls()`. Use `ls(pos=2)` to display names of objects in the database in position 2 on the search list, and so on. The following search list is from a version 2.8.0 installation immediately after startup:

```
> search()
[1] ".GlobalEnv"         "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"    "Autoloads"         "package:base"
```

This listing includes all packages automatically loaded at startup. For information on `Autoloads`, see the relevant help page.

The search list can be extended in three ways – by the use of `library()` to attach another package, by the use of `attach()` to attach a data frame or list object, and by the attachment of an image file, perhaps the `.RData` file from another directory. For packages, the effect of `library()` is to give access to the functions and data sets in the package. For data frames, the effect of `attach()` is to allow reference to data frame columns by name, without further mention of the name of the data frame. For image files, the effect of `attach()` is to give access to the objects that have been stored in the image file. Examples will be given below.

14.2.2 Workspace management

Failure to remove objects that are no longer needed leads to a cluttered workspace, which can be difficult to manage. Meaningful names should be used for objects that will be retained for future sessions. Acronyms can help in shortening names, but they should be memorable. For data sets of modest size, it may be reasonable to keep two versions: one with a name such as `BodyMassIndex` and another such as `BMI`; the latter helps to keep typing to a minimum. Short names with slight mnemonic significance can be reserved for use for objects that can be deleted almost immediately after use; such names are in the style of `a`, `b`, `x`, `tmp`, and `junk`. Such strategies make it easier to identify the objects that can be removed prior to quitting a session.

There are two complementary strategies for managing data and other objects.

- Objects that cannot easily be reconstructed or copied from elsewhere, but are not for the time being required, can be saved to an image file, using `save()`. For example:

  ```
  xy <- matrix(rnorm(60000), nrow=600)
  xy.rowrange <- apply(xy, 1, range)
  save(xy, xy.rowrange, file="xy.RData")
  rm(xy, xy.rowrange)
  ```

  Use of the command `attach("xy.RData")` then makes these objects available, as and when required. They are in memory (this may change), but would not be saved into

an image of the workspace. (If modified in some way, the modified version will appear in the workspace.)
- Use a separate working directory for each major project.

Such management can be more than otherwise important when working with data objects that occupy a substantial part of available memory.

Thus, for working through the code and exercises in this book, it may be helpful to reserve a different working directory for each different chapter. On a Windows system, suitable names for the respective working directories, each with its own default `.RData` image file, would be `c:\r\ch1`, `c:\r\ch2`, ... If while in the working directory `c:\r\ch2\`, access is required to objects from the image file in `c:\r\ch1\`, enter:

```
attach("c:/r/ch1/.RData")  # Microsoft Windows system
  # NB: Forward slashes have replaced Windows backslashes
ls(pos=2)                  # Check names of available objects
```

An alternative is to continue use of the same directory, but once finished with chapter 1 use, e.g., `save.image(file="ch1.RData")`, to save chapter 1 objects into the image file `ch1.RData`. The current workspace contents can then be removed (use the relevant menu option, or specify `rm(list=ls(all=TRUE))`), leaving an empty workspace for use with chapter 2. For later resoration of the chapter 1 workspace, type `load("ch1.RData")`. Alternatively, use `attach("ch1.RData")` to give access to its objects.

Workspaces typically contain a number of objects whose names have "`.`" as their first character, and that are ordinarily hidden from display. For removal of all objects from a workspace, be sure to specify:

```
rm(list = ls(all=TRUE))
```

The default action of `save.image()`, i.e., save all objects in the workspace to the default **`.RData`** file, is `save(list=ls(all=TRUE), file=".RData")`.

Changing the working directory and/or workspace

As a preliminary to loading a new workspace, it will usually be desirable to remove unwanted objects, save the workspace, clear it, and move to a new working directory. These operations may be performed from the menu, if any. Alternatively, use `save.image()` to save the current workspace, `rm(list=ls(all=TRUE))` to clear the workspace, `setwd()` to change the working directory, and `load()` to load a new workspace.

14.2.3 Utility functions

Useful functions are:

```
dir()                # List files in the working directory
file.choose()        # Choose a file interactively
sessionInfo()        # Print version numbers for R and for
                     # attached packages
```

```
system.file()        # Show path, by default to 'package="base"'
                     # Try, e.g.: system.file(package="DAAG")
R.home()             # Give the path to the R home directory
.Library             # Path to the default library.
Sys.getenv()         # Show settings of environment variables
object.size()        # Show size of R object
```

Here is an example of the use of `dir()`:

```
> dir()
[1] "chap12.Rnw"    "diary.R"       "figs12.R"      "oneBadRow.txt"
[5] "scan-demo.txt"
```

Optionally, a directory path and/or search pattern can be given as argument.

Type `names(Sys.getenv())` to get names of environment variables. Individual settings can then be obtained thus:

```
> Sys.getenv("R_HOME")
                     R_HOME
"C:\\PROGRA~1\\R\\R-29~1.0"
> normalizePath(Sys.getenv("R_HOME"))
[1] "C:\\Program Files\\R\\R-2.9.0"
```

The following can be used to get the path to files that come with an installed package:

```
> system.file("misc/ViewTemps.RData", package="DAAG")
[1] "C:/PROGRA~1/R/R-29~1.0/library/DAAG/misc/ViewTemps.RData"
```

14.3 R system configuration

14.3.1 The R *Windows installation directory tree*

The R system is installed into a *directory tree* that has a directory with the R version name (e.g., R-2.10.0) at its base. This can be placed anywhere that write permissions allow. The Windows R-2.10.0 default, for installers who have administrator privileges, is:

```
"C:\PROGRAM FILES\R\R-2.10.0\"
```

A likely alternative, in the absence of administrator privileges, is:

```
"C:\Documents and Settings\Owner\My Documents\R\R-2.10.0\"
```

The directory tree is relocatable. It can be copied to a flash drive or to a CD or DVD. An R session can then be run from the executable `bin\Rgui.exe` in, for example, the directory `D:\R\R-2.10.0\`. For running R from a CD or DVD, be sure to change the Start In directory to a directory where the user has write permission.

14.3.2 The library directories

Users can install packages into their own library directory. For users who do not have write permission for the library directory used for the initial installation, installation into a local directory is a necessity.

An R session that is running from one installation, perhaps the main installation on the hard drive, can in principle access an installed library directory from any compatible R installation. Thus, the following gives access, from an R session that is started (e.g.) from the hard drive, to a library tree that is under `D:\R-2.10.0`:

```
.libPaths("D:/R-2.10.0/library")
```

This can, for example, be used to make available the library tree from an R installation tree on a DVD that is in the `D:` drive.

In order to avoid the need to type this each time a new session is created in the session's working directory, create the following function **.First()**:

```
.First <- function().libPaths("D:/R-2.10.0/library")
```

This function will then be saved as part of the default workspace, and executed at the start of any new session in that directory.

14.3.3 The startup mechanism

Various system variables are set at startup, and a number of packages are attached. The details can be controlled at an installation level, at a user level, and at a startup directory level. See `help(Startup)` for details. If started in the standard manner, and with the "R_PROFILE_USER" environment variable unset, R searches at startup for a file called `.Rprofile` in the current directory or in the user's home directory (in that order). The `.Rprofile` file can, for example, define a `.First()` and/or a `.Last()` function.

14.4 Data input and output

Definitive information is in the help information for the relevant functions or packages, and in the R Data Import/Export manual that is part of the official R documentation. New features will appear from time to time.

The present discussion will use two files which, in order for the code given below to work, must reside in the working directory. These files have the names `oneBadRow.txt` and `scan-demo.txt`. The *DAAG* function `datafile()` offers an easy and convenient way to place these in the working directory, thus:

```
library(DAAG)
datafile("oneBadRow")
datafile("scan-demo")
```

As viewed from a text editor, or using the function `file.show()`, the entries of the first three lines of `oneBadRow.txt` should be as follows:

```
10 9 17  # First of 7 lines
11 13 1 6
9 14 16
```

The second line has four fields, while the other lines have three.

The entries in `scan-demo.txt` (again viewed with a text editor) are:

```
First of 4 lines
a 2 3
b 11 13
c 9 7
```

14.4.1 Input of data

Most common data input requirements, for data where records all have the same number of fields (usually numeric or character) can be handled using `read.table()`, or one of its variants that differs only in the choice of default arguments. Variants of `read.table()` include `read.delim()` for reading tab-delimited data files and `read.csv()` for reading comma-delimited data files. The function `scan()` allows greater user control, and extends somewhat the range of data formats that are readily handled.

The package *foreign* has functions that accept data in various proprietary and other formats. At the time of writing, this includes ARFF, DBF, and formats used by S-PLUS, Minitab, SPSS, SAS, Stata, and Systart. Files can be written in ARFF, Stata binary format, and in DBF database format. See the `help(package="foreign")`. Then examine the help page for individual functions.

Most of the available functionality for input of data from a rectangular file, or from a clipboard or spreadsheet equivalent, or from one of the common statistical package formats, can be accessed from the R Commander `(RCmdr)` GUI, or from the `rattle` GUI (see Section 14.1). Both of these GUIs can be convenient for input of data from Excel or similar spreadsheet format. See also the R Data Import/Export manual.

Issues for data input with `read.table()` *and variants*

Any column that does not consist entirely of numeric data (or entirely of logical or complex values) is taken to be of mode character. By default it is stored as a factor, with as many levels as there are unique text strings. Small mistakes in data entry, such as accidental use of the letter "O" in place of the number "0", will have the result that the columns are treated as text and stored as a factor.

The argument `stringsAsFactors=FALSE` prevents conversion to factors. The argument `colClasses="numeric"` forces all columns to be numeric, with an error message if an input field is not a legal numeric value. For finer control, either of `stringsAsFactors` and `colClasses` can be a vector; see the help page for `read.table()` for details.

Missing value symbols that differ from the default `NA` (or a space for logical, numeric, or complex fields) must be explicitly indicated. Thus, if the period (.) has been used as the missing value symbol, supply the argument `na.strings=c(".")`. Multiple missing

value symbols are allowed; thus the argument `na.strings=c("*", ".")` will cause both `"*"` and `"."` to be interpreted as `NA`.

The argument `comment.char` is by default set to `"#"`. Anything that follows # on an input field is treated as comment and ignored.

By default, both double and single quotes are treated as character string delimiters. Following the appearance of a `"` that initiates a text field, anything that appears up until the next `"`, including `'`, is taken as part of the text field. The same is true with the roles of `"` and `'` reversed.

For input of a file whose text fields may include the vertical single quote, set `quote="\""` (set string delimiter to `"`) or `quote=""`. Thus, suppose that the file `quote.txt` has the single line

```
'Quotes: "'  "Quotes: ' and '"
```

The following indicates the behavior of the `quote` argument:

```
> read.table("quote.txt")
         V1                               V2
1 Quotes: " Quotes: the 'Fortunes' package
> read.table("quote.txt", quote="'")
         V1       V2  V3       V4       V5
1 Quotes: " "Quotes: the Fortunes package"
```

Tracking errors in input data

An error message stating that different rows have different numbers of fields may be an indication that one or more arguments should be changed. The argument `fill=TRUE` allows data input to proceed regardless, filling out rows with blank fields as necessary. The data can then be checked carefully to identify the source of the problem.

The function `count.fields()` counts the number of fields that it finds in each row, making it easy to identify any differences between rows in the number of fields.

```
> nfields <- count.fields("oneBadRow.txt")
> nfields            # Number of fields, for each row
[1] 3 4 3 3 3 3 3
>
> ## Now identify rows where the number of fields seems anomalous
> (1:length(nfields))[nfields == 4]
[1] 2
```

One character string for each input row – `readLines()`

The function `readLines()` creates a character vector that holds one character string for each input row of the file. The argument `n` controls the number of lines that are read. By default `n = -1`, which inputs the entire file:

```
readLines("oneBadRow.txt", n=3)    # First 3 lines
readLines("oneBadRow.txt", n=-1)   # All lines
```

Input of fixed format data

Use `read.fwf()`. The argument `widths` is an integer vector that specifies the widths of the fixed-width fields. Alternatively, it may be a list of integer vectors, with one list element (an integer vector) for each line of multi-line records. See `help(read.fwf)`.

Input of large rectangular files

The function `scan()` is likely to be faster than `read.table()` (possibly, much faster) for reading a large data set in which data are all of the one type. By default, `scan()` stores all data in a single numeric vector; this is then readily dimensioned as a matrix.

```
> scan("oneBadRow.txt", skip = 1, quiet= TRUE)
 [1] 11 13  1  6  9 14 16 12 15 14  8 15 15  9 13 12  7 14 18
```

Alternatively, the argument `what=""` can be a vector or list. The output is then a list that has one vector for each vector or list element. For example, `what=list(ab="", col2=1, col3=1)` will generate a list consisting of one character vector, named `ab` and two numeric vectors, named `col2` and `col3`. The following turns the list into a data frame (with the code as it stands, the column `ab` becomes a factor):

```
> data.frame(scan("scan-demo.txt", skip = 1, quiet= TRUE,
+            what=list(ab="", col2=1, col3=1)))
  ab col2 col3
1  a    2    3
2  b   11   13
3  c    9    7
```

Example – input of the Boston housing data

The StatLib data sets archive at http://lib.stat.cmu.edu/datasets/ includes, among others, the file `boston_corrected.txt`. The command `datafile("bostonc")` places this file in the working directory, under the name `bostonc.txt`.

Careful examination of the file, and some preliminary experimentation that uses `readLines()` in the way that will now be described, indicates that the first nine lines are background documentation, line 10 is header information, and line 11 is the first record.

To check this, enter:

```
readLines("bostonc.txt", n=11)[10:11]
```

The output makes it clear that the tab symbol (`\t`) is the separator. More useful output is obtained from:

```
> strsplit(readLines("bostonc.txt", n=11)[10:11], split="\t")
[[1]]
 [1] "OBS."    "TOWN"    "TOWN#"   "TRACT"   "LON"     "LAT"
 [7] "MEDV"    "CMEDV"   "CRIM"    "ZN"      "INDUS"   "CHAS"
[13] "NOX"     "RM"      "AGE"     "DIS"     "RAD"     "TAX"
[19] "PTRATIO" "B"       "LSTAT"
```

```
[[2]]
 [1] "1"           "Nahant"      "0"           "2011"
 [5] "-70.955000" "42.255000"  "24.0"        "24.0"
 [9] "0.00632"     "18.0"        "2.31"        "0"
[13] "0.538"       "6.575"       "65.2"        "4.0900"
[17] "1"           "296"         "15.3"        "396.90"
[21] "4.98"
```

The following uses `scan()` to input this file:

```
boston <- scan("bostonc.txt", n=-1, sep="\t", skip=10,
               what=c(list(1,""), as.list(rep(1,19))))
colnams <- scan("bostonc.txt", skip=9, n=21, what="")
boston <- data.frame(boston)
names(boston) <- colnams
```

Compare the above with the use of `read.table()`:

```
boston <- read.table("bostonc.txt", sep="\t", skip=9,
                     comment.char="", header=TRUE)
```

The argument `comment.char=""` allows the input of fields, here the column name `TOWN#`, containing the # character. Text that follows `TOWN#` would otherwise be ignored.

14.4.2 Data output

Output of data frames

The function `write.table()` writes out data frames, thus:

```
write.table(fossilfuel, file="fuel.txt")
```

The file `fuel.txt` then contains the following:

```
"year" "carbon"
"1" 1800 8
"2" 1850 54
"3" 1900 534
"4" 1950 1630
"5" 2000 6611
```

Specify `row.names=FALSE` to suppress row names, and/or `col.names=FALSE` to suppress column names, and/or `quote=FALSE` to suppress quotes in the output.

For completeness, note also `write()`, used primarily for writing matrices or vectors. See `help(write)`.

Redirection of screen output to a file

The function `sink()` takes as argument the name of a file. Screen output is then directed to that file. To direct output back again to the screen, call `sink()` without specifying an argument. For example:

```
> fossilfuel          # Below, this will be written to the file
  year carbon
1 1800      8
2 1850     54
3 1900    534
4 1950   1630
5 2000   6611
> sink("fuel2.txt")
> fossilfuel          # NB: No output on screen
> sink()
```

Output to a file using `cat()`

The function `cat()` can be used to direct output either to the screen or to a file. Output objects must at present be scalars or vectors or matrices. Matrices are output as single vectors, with elements in columnwise order. The user can place limited format controls (spaces, tabs, and newlines) between the names of output objects.

14.4.3 Database connections

Note in particular the *RSQLite*, the *RMySQL*, and the *ROracle* packages. These all use the common database interface provided by the *DBI* package.

The *RSQLite* package makes it possible to create an SQLite database, or to add new rows to an existing table, or to add new table(s), within an R session. The SQL query language can then be used to access tables in the database. Here is an example:

```
library(DAAG)
library(RSQLite)
driveLite <- dbDriver("SQLite")
con <- dbConnect(driveLite, dbname="hillracesDB")
dbWriteTable(con, "hills2000", hills2000, overwrite=TRUE)
dbWriteTable(con, "nihills", nihills, overwrite=TRUE)
dbListTables(con)
## Obtain rows 11 to 20 from the newly created nihills table
dbGetQuery(con, "select * from nihills limit 10 offset 10")
dbDisconnect(con)
```

The database `hillracesDB`, if it does not already exist, is created in the working directory.

14.5 Functions and operators – some further details

Once code is in place for a computation that will be carried out repeatedly, it makes sense to place the code in a function. This should happen sooner rather than later. Such functions ease the burden of documenting the computations, and of ensuring an audit trail. What precise sequence of changes was applied to the initial data frame, created from data supplied by a client, to create the data frame used for the analysis?

Functions are usually the preferred way to parcel code that is to be taken over and used by another person. The R packages do this, for a collection of functions that are built around

a common theme, in the carefully structured and documented way that is desirable for code that will be offered to a wider community.

A further possibility is to extend or adapt built-in functions. One of R's major merits is that its resources are at the user's command, to use or modify or enhance at one's discretion. Take care to use that freedom well!

Issues for the writing and use of functions

The number of the user's own functions may soon become large. Choose names carefully, so that they are meaningful. Choose meaningful names for arguments, even if longer than is preferred. Remember that they can be abbreviated in actual use. Use lists, where this seems appropriate, to group together arguments that are conceptually related.

If at all possible, give arguments sensible defaults. Often a good strategy is to use as defaults arguments that are suitable for a demonstration run of the function. `NULL` is a useful default where the argument is mostly not required, but where if it appears may be any one of several types of data structure. Within the function body, an `if(!is.null(ArgumentOfInterest))` form of construction then determines whether it is necessary to investigate that argument further. Values of constants that may need to change in later use of the function should appear as argument defaults.

A demonstration mode that shows by example what the function does may be useful for debugging. Thus, in the function `mean.and.sd()` in Subsection 1.4.3, the argument `x` had the default value `x = rnorm(10)`. The default could equally well have been `x = rnorm(20)` or `x = runif(10)`, or even `x = 1:10`. This last has the advantage of giving predictable output, which can make debugging easier.

Structure code to avoid multiple entry of information. As far as possible, make code self-documenting. Use meaningful names for objects. The R system allows the use of names for elements of vectors and lists, and for rows and columns of arrays and data frames. Consider the use of names rather than numbers when extracting individual elements, columns, etc. Thus, `fossilfuel[, "carbon"]` is more meaningful and safer than `fossilfuel[, 2]`.

Break functions into a small number of subfunctions or "primitives". Reuse existing functions wherever possible. Write any new "primitives" so that they can be reused. This helps ensure that functions contain well-tested and well-understood components. Functions that are useful for one or other task are posted from time to time on the R-help electronic mail list, and on other such lists. There may be functions in our *DAAG* package that readers will wish to take over and use or adapt for their own purposes.

Where data and labeling must be pulled together from a number of different objects and files, and especially where it may be necessary to retrace steps some months later, take the same care over organizing and naming data as over structuring code.

14.5.1 Function arguments

The `args()` function

This function displays named function arguments, with any default settings. For example:

```
> args(write.table)  # Version 2.10.0 of R
function (x, file = "", append = FALSE, quote = TRUE, sep = " ",
    eol = "\n", na = "NA", dec = ".", row.names = TRUE,
    col.names = TRUE, qmethod = c("escape", "double"))
NULL
```

Use of `args()` may, when a check is needed on argument names, be an alternative to looking up the help page. Note however that in some very common functions, including `plot()`, many or most of the arguments are not named in the argument list, but are instead passed via the optional `...` argument.

The . . . argument

The optional. . . argument allows the passing of arguments that are additional to those in the named arguments list. Parameters that are passed in this way can be accessed by extracting the appropriate list element from `list(...)`. The list elements have the names, if any, that they are given when the function is called.

The function `rm()` uses this mechanism to target an arbitrary set of objects for removal, as in the following code fragment:

```
x <- c(1,5,7); z <- c(4,9,10,NA); u <- 1:10
# Now do calculations that use x, z and u
rm(x, z, u)
```

The . . . argument can be useful when the number of possible arguments is large, as for `plot()` and related functions. The onus is on the user to ensure that arguments have names that will be recognized within the function, or within functions to which the . . . argument list is passed.

14.5.2 Character string and vector functions

The function `nchar()` counts the number of characters in a string, `substring()` extracts a substring, and `paste()` pastes its arguments together into a single string. Type `help.search("string")` for a more complete list.

```
> substring("abracadabra",3, 8) # Extract characters 3 to 8
                                # inclusive
[1] "racada"
> nchar("abracadabra")          # Count the number of characters
[1] 11
> strsplit("abracadabra", "r")  # Split wherever "r" appears
[[1]]
[1] "ab"     "acadab" "a"
> strsplit("abcd", split="")    # Split into separate characters
[[1]]
[1] "a" "b" "c" "d"
> paste("ab","c","d", sep="")   # Join together
[1] "abcd"
> paste(c("a","b","c"), collapse="")  # Join vector elements
[1] "abc"
```

All these functions can be applied to a character vector. If `strsplit()` is supplied with a vector of strings as argument, the output has one list element for each element of the vector. By default, the first argument to `strsplit()` is interpreted as a "regular expression", in which certain characters have a special meaning. To avoid this, supply the argument `fixed=TRUE`.

Use `is.character()` to test whether an object is a character rather than perhaps a factor.

14.5.3 Anonymous functions

Anonymous functions are defined in the same way as for named functions, except that there is no assignment to a name. Thus the two lines

```
ssfun <- function(x)sum(x^2)
sapply(elastic1, ssfun)   # elastic1 is from the DAAG package
```

can be replaced by the single expression

```
sapply(elastic1, function(x)sum(x^2))
```

The calculation of Subsection 1.4.7 can be handled conveniently using an anonymous function. First, here is a version that uses an explicit named function:

```
growthfun <- function(x)(x[9] - x[1])/x[1]
sapply(austpop[, -c(1,10)], growthfun)
```

Using an anonymous function, this becomes:

```
sapply(austpop[, -c(1,10)], function(x)(x[9] - x[1])/x[1])
```

14.5.4 Functions for working with dates (and times)

The R *base* system has several functions for working with dates. See `help(Dates)` and `help(as.Date)` and `help(format.Date)` for detailed information.

Use `as.Date()` to convert character strings into dates. The default format has year, then month, then day of month, thus:

```
> # Electricity Billing Dates
> dd <- as.Date(c("2003-08-24","2003-11-23","2004-02-22","2004-05-03"))
> diff(dd)
Time differences of 91, 91, 91 days
```

Use `format()` to set or change the way that a date is formatted. The following is a selection of the available symbols:

%d: day, as number
%a: abbreviated weekday name (%A: unabbreviated)
%m: month (00–12)
%b: month abbreviated name (%B: unabbreviated)
%y: final two digits of year (%Y: all four digits)

The default format is `"%Y-%m-%d"`. The character `/` can be used in place of `-`. Other separators (e.g., a space) must be specified explicitly, using the `format` argument, as in the examples below.

The function `as.Date()` takes a vector of character strings that has an appropriate format, and converts it into a dates object. By default, dates are stored in integer numbers of days, using January 1 1970 as origin. Use `julian()` to convert a date into its integer value. Here are examples:

```
> as.Date("1/1/1960", format="%d/%m/%Y")
[1] "1960-01-01"
> as.Date("1:12:1960",format="%d:%m:%Y")
[1] "1960-12-01"
> as.Date("1960-12-1") - as.Date("1960-1-1")
as.Date("1960-12-1") - as.Date("1960-1-1")
> as.Date("31/12/1960","%d/%m/%Y")
[1] "1960-12-31"
> julian(as.Date("1/1/2000","%d/%m/%Y"))
[1] 10957
attr(,"origin")
[1] "1970-01-01"
```

Use `format()` to control the formatting of dates in printed output. See `help(format.Date)`.

```
> dec1 <- as.Date("2004-12-1")
> format(dec1, format="%b %d %Y")
[1] "Dec 01 2004"
> format(dec1, format="%a %b %d %Y")
[1] "Wed Dec 01 2004"
```

Such formatting may be used to give meaningful labels on graphs. The following uses the function `seq()` to assign a regular sequence of dates that become positions on the x-axis:

```
## Labeling of graph: data frame jobs (DAAG)
startofmonth <- seq(from=as.Date("1Jan1995", format="%d%b%Y"),
                    by="1 month", length=24)
atdates <- seq(from=as.Date("1Jan1995", format="%d%b%Y"),
               by="6 month", length=4)
datelabs <- format(atdates, "%b%y")
xyplot(BC+Alberta ~ startofmonth, data=jobs, outer=TRUE,
       scale=list(x=list(at=atdates, labels=datelabs)),
       auto.key=list(columns=2, between=1))
```

See `help(seq.Date)` for details of options that are available for generating regular sequences of dates.

Other useful functions include `weekdays()`, `months()`, `quarters()`. For example:

```
> dd <- as.Date(c("2003-08-24","2003-11-23","2004-02-22","2004-05-03"))
> weekdays(dd)
[1] "Sunday" "Sunday" "Sunday" "Monday"
> months(dd)
[1] "August" "November" "February" "May"
quarters(dd)
[1] "Q1" "Q3" "Q4" "Q2"
```

The function `date()` returns the current date and time, while `Sys.Date()` returns the date. For information on functions for working with times, see `help(ISOdatetime)`. The CRAN Task View for Time Series Analysis has notes on classes and methods for working with times and dates, and on packages that provide useful functionality.

14.5.5 Creating groups

The following uses `cut()` to collapse the column `bp` in the data frame `Pima.tr2` (*MASS*) into four categories, and to tabulate the frequencies. Notice the use of the argument `exclude=NULL` (cf. `useNA="ifany"`) to ensure that any `NA`s are included in the tabulation.

```
> library(MASS)
> catBP <- cut(Pima.tr2$bp, breaks=4)
> table(catBP, exclude=NULL)
catBP
(37.9,57]   (57,76]   (76,95]  (95,114]      <NA>
       22       170        87         8        13
```

By default (`exclude=NA`), `table()` ignores `NA`s. Section 14.7 has further discussion.

14.5.6 Logical operators

The logical operators `&` (**and**) and `|` (**or**) operate on vectors. They have counterparts `&&` and `||` that expect a single element; if either operand is a vector, the first element only is taken. Thus compare:

```
> c(TRUE, TRUE, FALSE) & c(FALSE, TRUE, FALSE)
[1] FALSE  TRUE FALSE
> c(TRUE, TRUE, FALSE) && c(FALSE, TRUE, FALSE)
[1] FALSE
```

A vector form of `if` statement

The function `ifelse()` may be regarded as a vector form of the `if` statement. The comparison is made, and the resultant action taken, for all elements of a vector. See `help(ifelse)`.

Operators are functions

Operators are a special type of function. Thus `2+3` is the result of applying the operator `+` with the arguments `2` and `3`. The following syntax makes the function evaluation explicit:

```
> "+"(2, 3)
[1] 5
```

14.6 Factors

Factors are an essential adjunct to the use of model formulae and graphics formulae. They find wide use in the statistical modeling functions in the various R packages. Recall (Subsection 1.2.7) that factors are stored as integer vectors, where the integer values range from one to the number of levels. A table of levels, stored as an attribute of the factor, associates a unique character string with each different integer value. The levels can be accessed using `attr()` with the argument `which="levels"`. It is however better to use the function `levels()`.

Factors have the potential to cause surprises, so be careful! Note that:

- When a vector of character strings is included as a column in a call to `data.frame()`, the default is that the character vector becomes a factor, where the distinct character strings are the level names. To avoid this, either enclose the character vector name in the wrapper function `I()`, or supply the argument `stringsAsFactors=FALSE`.
- When the index variable in a `for` loop takes factor values, the values are the integer codes. For example:
  ```
  > fac <- factor(c("c", "b", "a"))
  > for (i in fac) print(i)
  [1] 3
  [1] 2
  [1] 1
  ```
- Use, for example, `as.character(fac)` to obtain a character vector in which factor levels replace the factor elements.
- If arguments to `cbind()` are matrices and/or vectors, a matrix is returned, and elements are coerced to the same vector mode. Thus factor elements are replaced by their integer values, if necessary coerced to character. See Subsection 14.9.4 for details.
- To extract the numeric values 1, 2, . . . , specify `unclass(fac)`.
- In Subsection 8.1.4, use of the function `aggregate()` generated a factor `conc` whose levels were `"0.8"`, `"1"`, `"1.2"`, `"1.4"`, `"1.6"`, and `"2.5"`. Use of `as.numeric(as.character(conc))` extracted the numeric values.

Ordered factors

Actually, it is the levels that are ordered. To create an ordered factor, or to turn a factor into an ordered factor, use the function `ordered()`. The levels of an ordered factor specify positions on an ordinal scale. Try

```
> stress.level <- rep(c("low","medium","high"), 2)
> ordf.stress <- ordered(stress.level, levels=c("low", "medium",
+ "high"))
> ordf.stress
[1] low    medium high   low    medium high
Levels:  low < medium < high
> ordf.stress < "medium"
[1]  TRUE FALSE FALSE  TRUE FALSE FALSE
> ordf.stress >= "medium"
[1] FALSE  TRUE  TRUE FALSE  TRUE  TRUE
```

Ordered factors have (*inherit*) the attributes of factors, and have a further ordering attribute. All factors, ordered or not, have an order for their levels! The special feature of ordered factors is that there is an ordering relation (<) between factor levels.

Factor contrasts

When the `lm()` or another similar modeling function is called, the model formula is parsed, and the function `model.matrix()` is used to create a model matrix. Where factors appear in the model formula, the choice of factor contrasts determines how the model matrix will be constructed. The different model matrices give different mathematical descriptions for the same model, as described in Subsection 7.1.2. Note also *polynomial* contrasts, usually reserved for use with ordered factors.

In order to use sum contrasts in place of treatment contrasts, specify `options(contrasts=c("contr.sum", "contr.poly"))`. It is also possible to set contrasts separately for each factor. See `help(C)`.

The coding for any particular choice of contrasts can be inspected directly. Note the following:

```
> contr.treatment(3)
  2 3
1 0 0
2 1 0
3 0 1
> contr.sum(3)
  [,1] [,2]
1    1    0
2    0    1
3   -1   -1
> cities <- factor(c("Melbourne","Sydney","Adelaide"))
          Sydney Adelaide
Melbourne      0        0
Sydney         1        0
Adelaide       0        1
> contr.sum(cities)
          [,1] [,2]
Melbourne    1    0
Sydney       0    1
Adelaide    -1   -1
```

The default is that `Adelaide` is level 1, `Melbourne` is level 2, and `Sydney` is level 3. Level names are by default ordered alphabetically.

**Tests for main effects in the presence of interactions?*

With rare exceptions, tests for main effects in the presence of interactions are a bad idea. The function `anova.lme()` in the *nlme* package allows the argument `type = "m"`. This gives marginal tests of the effect of dropping each term in the model in turn, while retaining other terms. This includes tests on the effect of "dropping" each factor main effect, while retaining all interaction terms – tests that in most circumstances do not make much sense. See Venables (1998) for commentary.

The word "drop" is inappropriate. Constraints are implicitly applied, in the "omission" of main effects from the model, that depend on the choice of contrasts. Thus the Type 3 test will test a different null hypothesis, depending on the choice of contrasts. Users of `lme()` should not go down this road unless they are very sure that they know what they are doing. Most who know what they are doing avoid Type 3 tests!

14.7 Missing values

Any arithmetic or logical operation that has the missing value indicator `NA` as one of its arguments returns `NA`. This applies to the relations `<, <=, >, >=, ==, !=`. The first four compare magnitudes, `==` tests for equality, and `!=` tests for inequality.

Be sure to use `is.na(x)` to test which values of `x` are `NA`. The expression `any(is.na(x))` returns `TRUE` if the vector has any `NA`s, and is otherwise `FALSE`.

The construct `x==NA` gives a vector in which all elements are `NA`s, and thus gives no information about the elements of `x`. The logic is that, in for example `5==NA`, the missing value might or might not equal 5.

Users who do not give adequate consideration to the handling of `NA`s may be surprised by the results. The following compares use of `is.na(x)` with `x==NA`:

```
> x <- c(1, 6, 2, NA)
> is.na(x)           # TRUE for NAs, otherwise FALSE
[1] FALSE FALSE FALSE  TRUE
> x == NA            # All elements are set to NA
[1] NA NA NA NA
> NA == NA
[1] NA
```

Values cannot be assigned to expressions whose subscripts include missing values:

```
> x <- c(1, NA, 6, 2, 10)
> x > 4 # The second element will be NA
[1] FALSE NA TRUE FALSE TRUE
> x[x>4] # NB: This generates a vector of length 3
[1] NA 6 10
> x[x > 4] <- c(101, 102)
Error: NAs are not allowed in subscripted assignments
```

Use of `!is.na(x)` limits the elements that are identified by a subscript expression to those that are not NAs. The following replaces elements that are greater than 4 by elements from the vector `c(101, 102)`:

```
> x[!is.na(x) & x > 4] <- c(101, 102)
> x
[1] 1 NA 101 2 102
```

Counting and identifying NA*s – the use of* `table()`

The variable `re74` in the data frame `nswdemo` (*DAAG*) has a number of missing values. The following gives a tabulation that indicates, for each treatment group, whether `re74` is 0, greater than 0, or unknown:

```
library(DAAG)
> with(nswdemo, table(trt, re74>0, useNA="ifany"))

trt FALSE TRUE <NA>
  0   195   65  165
  1   131   54  112
```

Another possibility is to create a factor named perhaps `fac_re74` that has levels `0`, `gt0`, and `<NA>`, thus:

```
> fac_re74 <- with(nswdemo, factor(re74>0, exclude=NULL))
> levels(fac_re74)
[1] "FALSE" "TRUE"  NA
> levels(fac_re74) <- c("0", "gt0", "<NA>")
> with(nswdemo, table(trt, fac_re74))
   fac_re74
trt    0 gt0 <NA>
  0  195  65  165
  1  131  54  112
```

The removal of NA*s*

The function `complete.cases()` takes as arguments any sequence of vectors, data frames, and matrices that all have the same number of rows. It returns a vector that has the value `TRUE` whenever a row is complete, i.e., has no missing values across any of the objects, and otherwise has the value `FALSE`.

The function `na.omit()` omits NAs from vectors. For matrices and data frames, it omits any rows where one or more elements is `NA`.

NA*s in modeling functions*

Many of the modeling functions have an argument `na.action`. We can inspect the global setting, thus:

```
> options()$na.action # Version 2.10.0, following startup
[1] "na.omit"
```

The argument `na.action=na.omit` causes omission of rows that hold NAs, prior to the analysis. A useful alternative to `na.omit`, when it is available, can be `na.exclude`. This omits rows that have NAs from the analysis. However, it returns values for all rows when fitted values, residuals, etc. are calculated, placing NAs in those positions.

Individual functions may have defaults that are different from the global setting. See `help(na.action)`, and help pages for individual modeling functions, for further details.

Sorting and ordering, where there are NA*s*

By default, `sort()` omits any NAs. The function `order()` places NAs last. Hence:

```
> x <- c(1, 20,  2, NA, 22)
> order(x)            # By default, na.last=TRUE
[1] 1 3 2 5 4
> x[order(x)]
[1]  1  2 20 22 NA
> sort(x)             # By default na.last=NA
[1]  1  2 20 22
> sort(x, na.last=TRUE)
[1]  1  2 20 22 NA
```

14.8* Matrices and arrays

Conceptually, a matrix is a rectangular array in which all elements have the same mode. The common modes are numeric, character, logical, or complex. A matrix is in this sense a more restricted structure than a data frame. Numeric matrices allow a variety of mathematical operations, including matrix multiplication, that are not available for data frames.

A matrix is a special case of an array, which may have more than two dimensions. Names may be assigned to the rows and columns of a matrix, or more generally to the different dimensions of an array. Details are below.

Matrix elements are stored in column order in one long vector, i.e., columns are stacked one above the other, with the first column first. Consistent with this, a matrix is a vector (numeric or character or logical) whose dimension attribute has length 2. Thus consider

```
> xx <- matrix(1:6, ncol=3)  # Equivalently, enter
                             # matrix(1:6, nrow=2)
> xx
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Use the function `dim()` to determine the dimensions. Thus:

```
> dim(xx)
[1] 2 3
```

The following are alternative ways to turn the matrix xx back into the vector of elements 1, 2, . . . , 6:

```
## Use as.vector()
x <- as.vector(xx)
## Alternatively, directly remove the dimension attribute
x <- xx
dim(x) <- NULL
```

Both mechanisms have the effect of removing the dimension attribute.

The function t() takes a matrix as its argument, which it transposes. It may also be used for data frames, with the side-effect that all elements will be promoted to a common type when columns are of different types.

Use rownames() to extract or assign row names, and colnames() for column names. An alternative, which assigns or extracts row and column names at the same time, is the use of dimnames(). The dimnames() function gives a list, in which the first list element is the vector of row names, and the second list element is the vector of column names.

The extraction of submatrices

The syntax is the same as for data frames. The result is now, except when a single row or column is extracted, a matrix.

For use in demonstrating the extraction of submatrices, set

```
> x34 <- matrix(1:12, ncol=4)
> x34
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

The following should be self-explanatory:

```
x34[2:3, c(1,4)]      # Extract rows 2 & 3 and columns 1 & 4
x34[-2, ]             # Extract all rows except the second
x34[-2, -3]           # Omit row 2 and column 3
```

Extraction of a single row or column yields, by default, a vector. To extract the second row, now as a matrix, use the syntax x34[1, , drop=FALSE]. Thus compare the following:

```
> x34[2, ]                # The dimension attribute is dropped
[1]  2  5  8 11
> x34[2, , drop=FALSE]    # Retain the dimension attribute
     [,1] [,2] [,3] [,4]
[1,]    2    5    8   11
```

Conversion of data frames and tables into matrices

Use `as.matrix()` to convert a data frame into a matrix. The columns should all be of one of the modes numeric or character or logical. If this is not the case, type conversion will be necessary. Where there is a choice between matrix computations and equivalent computations that start from the data frame equivalent of the matrix, the matrix computations can be much more efficient.

In R version 2.10.0, use of `as.matrix()` with a two-way table leaves the table unchanged, i.e., its class is still returned as "table". The rationale may be that two-way tables are already in matrix form. If `tab` is a two-way table, use `as(tab, "matrix")` to convert `tab` to the class matrix.

14.8.1 Matrix arithmetic

Matrix arithmetic has many different applications, including the implementation of new regression and multivariate methods. The following are some of the more basic operations:

```
## Set up example matrices G, H and B
G <- matrix(1:12, nrow=4); H <- matrix(112:101, nrow=4); B <- 1:3
G + H             # Element-wise addition (X & Y both n by m)
G * H             # Element-wise multiplication
G %*% B           # Matrix multiplication (X is n by k; B is k by m)
## Set up example matrices X (square, full rank) and Y
X <- matrix(c(1,1,1, -1,0,1, 2,4,2), nrow=3)
Y <- matrix(1:6, nrow=3)
solve(X, Y)       # Solve XB = Y (X must be square)
```

Note also the more numerically stable alternative for matrix multiplication, `crossprod(G,H)`, and the linear system solver `qr.solve(X, Y)`.

Computational efficiency

As noted above, matrix computations can be much more efficient than the equivalent computations with data frames. The difference can be substantial. The following is on a 2.16GHz Intel Core 2 Duo Macbook Pro with 2GB of random access memory:

```
> xy <- matrix(rnorm(1500000),ncol=50)
> dim(xy)
[1] 30000    50
> system.time(xy+1)  # user and system are processor times
   user  system elapsed
  0.010   0.012   0.023
> xy.df <- data.frame(xy)
> system.time(xy.df+1)
   user  system elapsed
  0.345   0.233   0.580
```

14.8.2 Outer products

A common use of `outer()` is to generate a matrix of quantities of the form

$$x_{ij} = a_i b_j.$$

For example:

```
# Multiplication table
> outer(1:4, 1:10)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    2    3    4    5    6    7    8    9    10
[2,]    2    4    6    8   10   12   14   16   18    20
[3,]    3    6    9   12   15   18   21   24   27    30
[4,]    4    8   12   16   20   24   28   32   36    40
```

More generally, `outer()` may be used to generate an array whose (i, j)th element is a function of a_i and b_j. Often it is convenient to use an anonymous function.

The argument `col` in R plotting functions may be supplied either with a number, or with the name of a color. Many of these colors come in five different shades; thus in addition to "blue" there are "blue1", "blue2", "blue3", and "blue4". The following generates all five shades of the colors "red", "blue", and "green", storing the result in a matrix.

```
> rbgshades <- outer(c("red","blue","green"), c("",paste(1:4)),
+                    function(x,y)paste(x,y, sep=""))
> rbgshades          # Display the matrix
     [,1]    [,2]     [,3]     [,4]     [,5]
[1,] "red"   "red1"   "red2"   "red3"   "red4"
[2,] "blue"  "blue1"  "blue2"  "blue3"  "blue4"
[3,] "green" "green1" "green2" "green3" "green4"
> plot(rep(0:4, rep(3,5)), rep(1:3, 5), col=rbgshades, pch=15, cex=8)
```

14.8.3 Arrays

An array is a generalization of a matrix (2 dimensions), to allow > 2 dimensions. The dimensions are, in order, rows, columns, ... By way of example, we start with a numeric vector `x` of length 24. For ease of keeping track of the elements, they will be $1, 2, \ldots, 24$.

By giving the vector `x` suitable dimensions, it is readily changed into, e.g., a 3×8 matrix, or into a $3 \times 4 \times 2$ array:

```
> x <- 1:24
> dim(x) <- c(2, 12); print(x)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    3    5    7    9   11   13   15   17    19    21    23
[2,]    2    4    6    8   10   12   14   16   18    20    22    24
>
> dim(x) <- c(3, 4, 2); print(x)
```

```
, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2
     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

The function `aperm()` permutes the dimensions. Thus `aperm(x, c(3,2,1))` interchanges dimensions 1 and 3.

14.9 Manipulations with lists, data frames, matrices, and time series

It can be important to understand the points of connection between lists and data frames, and between data frames and matrices. Data frames are a specialized type of list, in which the list elements hold vectors (the columns) that all have the same length and are all indexed by the same row names.

Data frames and matrices use the same subscripting syntax, though the outcome can be subtly different. Certain functions that are primarily designed for use with matrices can also be applied to data frames. Data frames are however, unlike matrices, stored as lists, and functions that are designed for use with lists can also be used with data frames.

The functions `dim()` and `dimnames()` generalize in the obvious way for use with arrays.

14.9.1 Lists – an extension of the notion of "vector"

Vectors, in a general sense, are of two types. First, there are *atomic* vectors whose elements are logical, integer, numeric, complex, or character. These are atomic because their elements do not break down into anything more fundamental. In earlier chapters, the word "vector" was reserved for use with atomic vectors.

Second, there are lists (and data frames). These are *recursive* (or *generic*) vectors. Such vectors consist of a sequential set of elements, just as for atomic vectors. The difference is that each list element is itself a list, with implications that will be explained shortly.

As lists are vectors, elements can be referenced using the usual subscript notation. The first list element is `zz[1]`, the second is `zz[2]`, and so on. Those elements, themselves lists, are wrappers for objects of arbitrary class and type. The list elements can hold scalars, matrices or more general arrays, functions, other lists, atomic vectors, etc. The elements of lists can, and often do, hold a rag-tag collection of different objects. A good example is the list object that R creates as output from an `lm` calculation.

Lists do not store the contents of the list elements directly. Instead they store pointers that allow the extraction of those contents, when and if required. Use of square brackets to extract a subset of a list merely subsets the pointers.

Here is an example:

```
> zz <- list("Shireen", "Peter", c("Luke","Amelia","Ted"), c(8,4,0))
> zz[c(3,1)]
[[1]]
[1] "Luke"   "Amelia" "Ted"

[[2]]
[1] "Shireen"

> ## List whose only element is the vector c("Luke","Amelia","Ted")
> zz[3]
[[1]]
[1] "Luke"   "Amelia" "Ted"
```

A notation is then required for obtaining the contents of list elements, that is, the objects to which the pointers refer, without their bags. The syntax `zz[[1]], zz[[2]], ...` does this. Thus `zz[[1]]` extracts the object that is held in the first list element, and similarly for other elements in the list. For example:

```
> ## Return the vector c("Luke","Amelia")
> zz[[3]]
[1] "Luke"   "Amelia"
```

List elements can be named. Elements, or the contents of elements, can then be extracted by name, thus:

```
> duo <- list(family="Braun", names=c("Matthew","Phillip"),
+             ages=c(14,9))
> duo[["names"]]      # Alternatively, specify duo$names
[1] "Matthew" "Phillip"
```

Functions such as `c()`, `length()`, and `rev()` (take elements in the reverse order) can be applied to any vector, including a list. To interchange the first two elements of the list `zz`, write `zz[c(2, 1, 3:length(zz))]`.

The dual identity of data frames

Data frames have the form of rectangular arrays whose elements can be extracted using the same subscript notation as for matrices. More fundamentally, they are lists whose elements hold columns that are all of the same length. Thus, set:

```
xyz <- data.frame(x=1:4, y=11:14, z=I(letters[1:4]))
```

Then `rev(xyz)` gives a data frame whose columns are taken in the reverse order. The function `length()` returns the value 3; this is because `xyz` is a list that has three elements. The wrapper function `I()` was used to prevent `z` from becoming a factor.

Note the contrast between:

```
> xyz[1, ]            # Returns a data frame, i.e., a list)
  x  y z
1 1 11 a
> xyz[1, , drop=TRUE]
$x
[1] 1

$y
[1] 11

$z
[1] "a"
> unlist(xyz[1, ])    # Returns, here, a vector of atomic mode
   x    y    z
 "1" "11"  "a"
> unlist(xyz[1, , drop=TRUE])
   x    y    z
 "1" "11"  "a"
```

The elements of `xyz[1, ]` were of different modes. The inconsistency was resolved, in coercing `xyz[1, ]` to vector mode, by constraining all elements to character. (If `xyz[1, 3]` had been a factor, all elements would have become numeric. Why?)

14.9.2 Changing the shape of data frames (or matrices)

The manipulations that will be described will use the functions `melt()` and `cast()` in the *reshape* package. There are methods for both data frames and matrices.

Melting and casting

The `melt()` function is a counterpart of `stack()` (discussed in Subsection 1.3.2) that has the advantage of carrying along other columns of the data frame.

```
> library(reshape)
> Jobs <- melt(jobs, measure.vars=names(jobs)[1:6],
+              variable_name="Region", id.vars="Date")
> head(Jobs, 3)
      Date Region value
1 95.00000     BC  1752
2 95.08333     BC  1737
3 95.16667     BC  1765
```

The explicit use of the argument `id.vars="Date"` was unnecessary. If the argument `measure.vars` is given, the default is to carry along all other columns in the data frame.

The following returns the data frame `Jobs` that was created above, to the wide format:

```
> jobsBack <- cast(Jobs, Date ~ Region)
> head(jobsBack,3)
      Date   BC Alberta Prairies Ontario Quebec Atlantic
1 95.00000 1752    1366      982    5239   3196      947
2 95.08333 1737    1369      981    5233   3205      946
3 95.16667 1765    1380      984    5212   3191      954
```

In the casting formula `Date ~ Region`, `Date` and `Region` uniquely identified a row of the data frame `Jobs`. In the data frame (`jobsBack`) that results, values of `Date` identify rows, and elements of `Region` identify columns. When the casting formula does not uniquely specify rows, the argument `fun.aggregate` must be supplied.

Subsection 15.5.3 has an example of the use of `melt()`.

14.9.3 Merging data frames* – `merge()`

The *DAAG* package has the data frame `Cars93.summary`, which has as its row names the six different car types in the data frame `Cars93` from the *MASS* package. The column `abbrev` holds one or two character abbreviations for the car types. We show how to merge the information on abbreviations into the data frame `Cars93`, thus:

```
new.Cars93 <- merge(x=Cars93, y=Cars93.summary[, "abbrev", drop=F],
                    by.x="Type", by.y="row.names")
```

The arguments `by.x` and `by.y` specify the keys, the first from the data frame that is specified as the `x`- and the second from the data frame that is specified as the `y`-argument. The new column in the data frame `new.Cars93` has the name `abbrev`.

If there had been rows with missing values of `Type`, these would have been omitted from the new data frame. We can avoid this by ensuring that `Type` has `NA` as one of its levels, in both data frames.

14.9.4 Joining data frames, matrices, and vectors – `cbind()`

Use `cbind()` to join, side by side, two or more objects that may be any mix of data frames and vectors.

If arguments to `cbind()` are matrices and/or vectors, a matrix is returned, and elements will be coerced to the same vector mode – character if one or more columns is character, and otherwise numeric. If one or more arguments is a data frame, a data frame is returned, and factor columns remain factor.

Use `rbind()` to stack any combination of data frames, matrices, and (row) vectors. When data frames are stacked one above the other, the names and types (numeric, logical, character, factor, . . .) of the columns must agree.

Conversion of tables and arrays into data frames

Use `as.data.frame()` to handle the conversion, from a table, to a data frame that has one column of values of the classifying factor corresponding to each dimension

of the table. If the argument is an array, either first coerce it into a table with the same number of dimensions, or use an explicit call to the function `as.data.frame.table()`, with the array as argument.

The `adply()` function (*plyr* package) applies a given function across specified dimensions of an array, returning a data frame. Note also `aaply` (returns an array), and other such functions in the same package. See Exercise 12 at the end of the chapter for simple examples.

14.9.5 The `apply` *family of functions*

The functions that are in mind are `sapply()`, `lapply()`, `apply()`, and `tapply()`. The functions `sapply()` and `lapply()` operate on vectors, on lists, and on data frames. We encountered `sapply()` (`s` = simplify) in Chapter 1, and it has been used extensively in earlier chapters. The function `apply()` is designed for use with matrices and arrays. It can also be used with data frames, provided of course that the operations that the function performs are legal for elements in the data frame.

The `tapply()` *function*

The arguments are a vector, a list of factors, and a function that operates on a vector to return a single value. For each combination of factor levels, the function is applied to corresponding values of the variable. For simplicity, assume that the function returns a scalar. The default output (with `simplify=TRUE`) is then an array with as many dimensions as there are factors.

Compare the following two calculations, the first using `tapply()` and the second (from Subsection 2.2.2) using `aggregate()`:

```
## Compare tapply() with aggregate(): data frame kiwishade (DAAG)
with(kiwishade, tapply(yield, INDEX=list(block, shade), FUN=mean))
with(kiwishade, aggregate(yield, by=list(block, shade), FUN=mean))
```

The function `tapply()` yields an array, where `aggregate()` yields a data frame. Another difference is that the first argument to `tapply()` must be a vector, where `aggregate()` can be used with data frames or with time series objects and will then carry out the aggregation in parallel across all columns. Where there are no data values for a particular combination of factor levels `tapply()` returns `NA`, whereas `aggregate()` does not include that row in the data frame. Where the time taken for the calculation is an issue, `tapply()` typically completes the task in a fraction of the time taken by `aggregate()`.

The `apply()` *function*

The function `apply()` applies a function to rows or columns of a matrix, or to specified dimension(s) of an array, or (if necessary, after coercing all elements to the same vector mode) with a data frame.

The first argument to `apply()` is the array or data frame. The second argument specifies the dimension(s) – 1 for applying the function to each row in turn, and 2 when the function is to be applied to each column in turn (if the first argument is an array of more than two

dimensions, then a number greater than 2 is possible). Examples can be found in Exercise 11 in Chapter 1, Subsections 8.1.4 and 6.2.4, and elsewhere.

The functions `lapply()` *and* `sapply()`

The functions `lapply()` and `sapply()` apply a function to each element of a vector in turn. They are most commonly used with recursive vectors, i.e., with lists or data frames. For `lapply()`, the result is a list, with one element corresponding to each element of the list that was supplied as argument. The function `sapply()` differs only in simplifying the result as far as possible, to give a vector or matrix or list.

The arguments of `lapply()` or `sapply()` are the name of the data frame or list (or other vector), and the function that is to be applied. Additional arguments may be given that will be passed to the function that is the second argument:

```
> ## Uses data frame rainforest (DAAG)
> sapply(rainforest[, -7], range, na.rm=TRUE)
     dbh wood bark root rootsk branch
[1,]   4    3    8    2    0.3      4
[2,]  56 1530  105  135   24.0    120
```

Note the additional argument `na.rm=TRUE`, which `sapply()` passed to the function `range()`.

For use of `sapply()`, note that:

- If `sapply()` is used with a matrix as argument, the function is applied to all elements of the matrix – not usually what is wanted. Use `apply()` instead, or convert the matrix to a data frame.
- Any rectangular structure that results from the use of `sapply()` will be a matrix, not a data frame. For manipulations on the columns of such a matrix, use `apply()`, not `sapply()`.

14.9.6 Splitting vectors and data frames into lists – `split()`

As an example, we split the column `No.of.cars`, from the data frame `Cars93.summary` (*DAAG* library), according to distinct values of `Max.passengers`:

```
> with(Cars93.summary, split(No.of.cars, Max.passengers))
$'4'
[1] 14

$'5'
[1] 21

$'6'
[1] 16 11 22

$'8'
[1] 9
```

The argument to `split()` may alternatively be a data frame, which is split into a list of data frames. For example:

```
## Split dataframe by Max.passengers (2nd column)
split(Cars93.summary[, -2], Cars93.summary[, 2])
```

14.9.7 Multivariate time series

As demonstrated in Subsection 2.1.5, the function `ts()` can be used to create a multivariate time series from a matrix or data frame whose columns hold the separate series, thus:

```
> jobts <- ts(jobs[,1:6], start=1995, frequency=12)
> colnames(jobts)
[1] "BC"       "Alberta"  "Prairies" "Ontario"  "Quebec"   "Atlantic"
```

To extract the first column, specify `tsunits[, 2]` or `tsunits[, "Alberta"]`. The subscript notation can be used to extract rows, but returns a matrix rather than a time series. Use the function `window()` to extract, as a time series, a subseries. For example:

```
> ## Subseries through to the third month of 1995
> window(jobts, end=1995+2/12)
           BC Alberta Prairies Ontario Quebec Atlantic
Jan 1995 1752    1366      982    5239   3196      947
Feb 1995 1737    1369      981    5233   3205      946
Mar 1995 1765    1380      984    5212   3191      954
>  # Rows are 1995+0/12, 1990+1/12, 1990+2/12
```

There is a plot method for multivariate time series:

```
plot(jobts, plot.type="single")     # Use one panel for all
plot(jobts, plot.type="multiple")   # Separate panels.
```

Here is alternative code for Figure 2.10A that plots the data as time series:

```
## Alternative code for Figure 2.9A; plot as time series
plot(jobts, plot.type="single", xlim=c(1995,1997.2), lty=1:6, log="y",
     xaxt="n", xlab="", ylab="Number of Jobs")
## Move label positions so that labels do not overlap
ylast <- bounce(window(jobts, 1996+11/12), d=strheight("O"), log=TRUE)
 # bounce() is from DAAG
text(rep(1996+11/12,6), ylast, colnames(ylast), pos=4)
datlab <- format(seq(from=as.Date("1Jan1995", format="%d%b%Y"), by="3 month",
                 length=8), "%b%Y")
axis(1, at=seq(from=1995, by=0.25, length=8), datlab)
```

14.10 Classes and methods

Generic functions, whose action varies according to the *class* of the object that is given as the first argument, were mentioned briefly in Subsection 1.4.2. There are two implementations – the S3 implementation that is provided by *base* R, and the more recent S4

implementation of the *methods* package. Generic functions do not call the specific method, such as `print.factor()`, directly. Instead, they call a *dispatch* function, which in the case of `print()` calls the relevant print function.

S3 methods and classes

For S3 methods and classes, the dispatch function is `UseMethod()`. For example, here is the function `print()`:

```
> print
function (x,...)
UseMethod("print")
```

The function `UseMethod()` notes the class of the object, now identified as `x`, and calls the print function for that class. If the object is a factor, then `UseMethod()` will call `print.factor()`.

Use the function `class()` to determine the class of an object. Classes may be defined so that they inherit the properties of parent classes. Thus ordered factors inherit from factors, and inherit the print method for factors.

Use `methods(print)` to list all (S3) `print` methods. Use `methods(class="lm")` to list all methods for the linear model class `lm`. Use `getAnywhere()` to list functions whose names are shown with an asterisk that identifies them as non-visible.

14.10.1 Printing and summarizing model objects

Just as for any other R object, typing the name of a model object on the command line invokes the print function, if any, for that class of object. Thus typing `elastic.lm`, where `elastic.lm` is an `lm` object, has the same effect as `print.lm(elastic.lm)` or `print(elastic.lm)`.

Print functions for model objects, e.g., `print.lm()` for printing the model object `elastic.lm`, process output into a form that is, broadly, suitable for immediate inspection. Additional or different information may be available by directly accessing the list elements of the model object. For many classes of object there is, in addition, a `summary()` function that gives a different and often more detailed summary. The following stores summary information for `elastic.lm` in the `elastic.sum` summary object:

```
elastic.lm <- lm(distance ~ stretch, data=elasticband)
elastic.sum <- summary(elastic.lm)
```

Typing `elastic.sum` (or `summary(elastic.lm)`) on the command line invokes the function `print.summary.lm()`, thus printing the summary information.

Other generic functions that commonly find use with model objects are `coef()` (alias `coefficients()`), `residuals()` (alias `resid()`), `fitted()`, `predict()`, and `anova()`.

14.10.2 Extracting information from model objects

In S3, model objects are usually lists with named elements. To get the names of the elements of `elastic.lm`, type

```
> names(elastic.lm)
 [1] "coefficients"  "residuals"   "effects"   "rank"
 [5] "fitted.values" "assign"      "qr"        "df.residual"
 [9] "xlevels"       "call"        "terms"     "model"
```

Here are three different and equivalent ways to examine the contents of the first list element:

```
elastic.lm$coefficients
elastic.lm[["coefficients"]]
elastic.lm[[1]]
```

The preferred way to extract the coefficients is however to use the extractor function `coef()`, i.e.

```
> coef(elastic.lm)
(Intercept)     stretch
 -63.571429    4.553571
```

S3 classes are not formally defined. Classes can be assigned to objects in an arbitrary manner, whether or not the object has the structure (e.g., the expected list elements) for that class. For example:

```
> x <- 1:5
> class(x) <- "lm"   # Inappropriate assignment of class
> print(x)
Error in x$call : $ operator is invalid for atomic vectors
```

14.10.3 S4 classes and methods

S4 objects have formally defined *slots*; these have a similar role to the list elements in, e.g., `lm` objects. The names and classes of the slots are established at the time of the class definition. In computations with objects of an S4 class, the names and classes of the slots are validated against the definition. Methods must likewise be formally defined.

Use `showMethods()` to get information on S4 methods.

Users of packages (e.g., *lme4* or Bioconductor packages) may need to access the slots of S4 objects. Use `slotNames()` to obtain the names of the slots, and either `slot()` or the operator `@` to extract or replace a slot. For example, consider the `lmList` object that was created in Subsection 10.6.1:

```
## Use data frame humanpower1, from DAAG
> library(lme4)
> hp.lmList <- lmList(o2 ~ wattsPerKg | id, data=humanpower1)
> slotNames(hp.lmList)
[1] ".Data" "call"  "pool"
> slot(hp.lmList, "call")
lmList(formula = o2 ~ wattsPerKg | id, data = humanpower1)
```

```
> hp.lmList@call
lmList(formula = o2 ~ wattsPerKg | id, data = humanpower1)
```

For further information on S4 classes and methods, see `help(Methods)`, Chambers (2007), Gentleman (2008). Lumley (2004a) compares S3 and S4 approaches to the definition of a simple class. See also Bates and DebRoy (2003).

14.11 Manipulation of language constructs

As with any R object, formulae and expressions can be manipulated. This extends to constructing model or graphics formulae, or expressions from character strings. These and related abilities will be demonstrated.

14.11.1 Model and graphics formulae

The following function `plot.mtcars()` takes two named columns from the data frame `mtcars`, plotting them one against another:

```
> plot.mtcars <- function(xvar="disp", yvar="hp"){
+     mt.form <- paste(yvar, "~", xvar)
+     plot(formula(mt.form), data=mtcars)
+     }
> ## Plot using data frame mtcars (datasets)
> names(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
[10] "gear" "carb"
> plot.mtcars()
> plot.mtcars(xvar="disp", yvar="mpg")
```

Extraction of variable names from formula objects

The function `all.vars()` takes a formula as argument, and returns the names of the variables that appear in the formula. For example:

```
> all.vars(mpg ~ disp)
[1] "mpg" "disp"
```

As well as allowing the use of a formula to specify the graph, the following gives more informative x- and y-labels:

```
plot.mtcars <- function(form = mpg~disp){
   ## Include information that allows a meaningful label
   mtcars.info <- c(mpg= "Miles/(US) gallon",
                    cyl= "Number of cylinders",
                    disp= "Displacement (cu.in.)",
                    hp= "Gross horsepower",
                    drat= "Rear axle ratio",
                    wt= "Weight (lb/1000)",
                    qsec= "1/4 mile time",
                    vs= "V/S",
```

```
                    am= "Transmission (0 = automatic, 1 = manual)",
                    gear= "Number of forward gears",
                    carb= "Number of carburettors")
   xlab <- mtcars.info[all.vars(form)[1]]
   ylab <- mtcars.info[all.vars(form)[2]]
   plot(form, xlab=xlab, ylab=ylab, data=mtcars)
}
```

14.11.2 The use of a list to pass arguments

The following are equivalent:

```
mean(rnorm(20))
do.call("mean", args=list(x=rnorm(20)))
```

Use of `do.call()` allows the argument list to be set up in advance of the call, as in the following function:

```
simulate.distribution <-
  function(distn="weibull", params=list(n=10)){
    ## Simulates one of: weibull, gaussian, logistic
    if(distn=="weibull" & !("shape" %in% names(params)))
      params <- c(shape=1, params)
    ## weibull requires a default shape argument.
    ## ======================================================
    ## Choose the function that will generate the random sample
    rfun <- switch(distn,
                   weibull = rweibull,
                   normal = rnorm,
                   logistic= rlogis)
    ## Call rfun(). Use of do.call() makes it possible to give
    ## the argument list as a list of named values.
    ## Pass shape argument (or NULL), plus n = # of numbers
    do.call("rfun", args=params)
  }
```

Now try the following:

```
simulate.distribution()
plot(density(simulate.distribution("normal", params=list(n=100))))
plot(density(simulate.distribution("weibull",
             params=list(n=100, scale=0.5))))
```

The function `call()`, which has the same syntax as `do.call()`, sets up an unevaluated expression. The expression can be evaluated at some later time, using `eval()`:

```
> mean.call <- call("mean", x=rnorm(5))
> eval(mean.call)
[1] -0.6276536
> eval(mean.call)
[1] -0.6276536
```

Notice that the argument x was evaluated when call() was evoked. Hence the result is unchanged upon repeating the use of eval(). This can be verified by printing out the expression:

```
> mean.call
mean(x = c(-0.68467334794551, -0.376091734366091, -0.289459988631994,
-3.04694266628697, 1.25889972957396))
```

14.11.3 Expressions

An expression is anything that can be evaluated. Thus x^2 is an expression, y <- x^2 is an expression that returns the value that is assigned to y, and y == x^2 is an expression. Recall that y <- x^2 assigns the value of x^2 to y, while y == x^2 tests whether y equals x^2.

The following can be convenient for the repeated evaluation of a complicated expression, changing one or more of the arguments at each new evaluation:

```
> local(a+b*x+c*x^2, envir=list(x=1:4, a=3, b=5, c=1))
[1]  9 17 27 39
```

14.11.4 Environments

Every call to a function creates a frame that contains the local variables created in the function. This combines with the environment in which the function was defined to create a new environment.

The global environment, .Globalenv, is the workspace. This is frame number 0. Broadly, the frame number increases by one with each new function call. Additionally, frames may be referred to by name. Use

- sys.nframe() to get the number of the current evaluation frame.
- sys.frame(sys.nframe()) to identify the frame by name.
- sys.parent() to get the number of the parent frame.
- sys.call() to return, from within a function, the function call.

There are many other such functions, but these will do for present purposes!

Here is a function that determines, from within the function, its name:

```
test <- function()as.character(sys.call())
```

The result of executing the function is:

```
> test()
[1] "test"
> newtest <- test    # Create a copy with a different name
> newtest()
[1] "newtest"
```

Automatic naming of a file that holds function output

The following automatically matches the names of files that hold hard copies of graphs to the names of the functions that created them. Thus, functions for creating figures may be given names `fig1()`, `fig2()`, etc. These functions in turn call a function `hcopy()` that sets up the `pdf` graphics device, and directs output to a file with a name that is formed by appending `".pdf"` to the file name:

```
hcopy <-
    function(width=2.25, height=2.25, pointsize=8){
        funtxt <- sys.call(1)
        fnam <- paste(funtxt, ".pdf", sep="")
        print(paste("Output is to the file '", fnam, "'", sep=""))
        pdf(file=fnam, width=width, height=height, pointsize=
            pointsize)
    }
```

Now define `fig1()`, which is designed to plot the sine function over the range $(-\pi, \pi)$, so that it calls `hcopy()`:

```
fig1 <- function(){
    hcopy()             # Call with default arguments
    curve(sin, -pi, 2*pi)
    dev.off()
}
```

Output goes to the file `fig1.pdf`. For a function `fig2()` that calls `hcopy()`, the file name will be `fig2.pdf`. The function `hardcopy()` in the *DAAG* package is a more sophisticated version of `hcopy()`.

14.11.5 Function environments and lazy evaluation

When a function is defined, this sets up an evaluation environment for that function. Variables that are not passed as arguments are first searched for in the frame of the function, which in R is the environment in which the function was itself defined. If not found there, the parent frame (if any) is searched, then the parent frame of the parent frame, and so on. If they are not found in any of the frames, then they are sought in the search list. A complete description is beyond the scope of this book.

The consequences for lazy evaluation, which we now discuss, are mildly subtle.

Lazy evaluation

Expressions may be specified as arguments to R functions. The expression is evaluated only when it is encountered in the R function. For example:

```
> lazyfoo <- function(x=4, y = x^2)y
> lazyfoo()
[1] 16
```

```
> lazyfoo(x=3)
[1] 9
```

This is unsurprising. Expressions that appear in default arguments are evaluated within the function environment.

By contrast, however, expressions that are specified when the function is called are evaluated in the parent environment, i.e., in the environment from which the function was called. Hence the following behavior:

```
> lazyfoo(y=x^3)     # We specify a value for y in the function call
Error in lazyfoo(y = x^3) : Object "x" not found
> x <- 9             # Now, in the parent environment, x=9
> lazyfoo(y=x^3)
[1] 729
```

The function `new.env()` can be used to create new environments. Just as for lists, the operators `$` or `[[` can be used to access objects whose names are in an environment, or to add the names of new objects.

Example – a function that identifies objects added during a session

This illustrates points that were made in the discussion above.

At the beginning of a new session, we might store the names of the objects in the workspace in the vector `dsetnames`, thus:

```
dsetnames <- objects()
```

Now suppose that we have a function `additions()`, defined thus:

```
additions <- function(objnames = dsetnames){
    newnames <- objects(envir=.GlobalEnv)
    existing <- newnames %in% objnames
    newnames[!existing]
    }
```

The function call `sys.frame(0)` returns the name of the workspace. At some later point in the session, we can enter

```
additions(dsetnames)
```

to obtain the names of objects that have been added since the start of the session.

Use of `newnames <- objects()` in the above function, i.e., leaving arguments at their defaults, would have returned the names of objects in the function environment.

14.12* Creation of R packages

Much of the functionality of R, for many important tasks, comes from the packages that are built on top of base R. Users who make extensive use of R are likely to find a need to document and organize both their own functions and associated data. Packages are the preferred vehicle for making functions and/or data available to others.

Organization of data and functions into a package has the following benefits:

- The package format imposes minimum standards of documentation, and consistency between code and documentation.
- Attaching the package gives immediate access to package functions, data, and associated documentation.
- Submission to CRAN (Comprehensive R Archive Network), and use by others, extends opportunities for testing and/or getting contributions from other workers. CRAN enforces additional checks, beyond those required to get the package up and running.

Researchers may find it helpful to put together into a package all data and user functions that relate to any major project. This facilitates returning to the project at some later time, and/or passing the project across to other workers.

It is relatively straightforward to create packages that make no calls to externally compiled C or Fortran code. For Unix and Linux systems, the necessary tools should already be available as part of the operating system. For MacOS X, the Xcode Tools must be installed. For Windows, package construction tools must be downloaded and installed. Go to `http://www.murdoch-sutherland.com/Rtools`, and see the manual "Writing R Extensions".

A first step is to set up a workspace that has only the functions and data objects that are to be included in the package. The image file `viewtemps.RData` that is included in the R installation has data and several functions that can be used for experimentation. To find where this file is located on the system, type

```
system.file("misc/ViewTemps.RData", package="DAAG")
```

Then, to create a package from the functions and data in this image file, start with an empty workspace and load `viewtemps.RData`. There should then be four objects in the workspace – the functions `plotD`, `plotT` and `plotTofD`, and the data frame `housetemps`.

Starting with the workspace as just described, the following steps create a package with the name *ViewTemps*:

1. Type
   ```
   package.skeleton(name = "ViewTemps")
   ```
   This creates all the needed directories and files or file skeletons, in a subdirectory `ViewTemps` in the working directory. Instructions for proceeding further are placed in a file `Read-and-delete-me` in this directory; the following is a synopsis.
2. Edit the DESCRIPTION file in the directory `ViewTemps`, as required. (This is not absolutely necessary, if the file is for your own use.)
3. Edit the help file skeletons in `ViewTemps/man`. (Optionally, combine help files for two or more functions.)
4. Open a terminal window (on Windows, a DOS prompt), go to the parent directory of `ViewTemps`, and build the package tarball:

   ```
   R CMD build ViewTemps
   ```

5. Now check the package tarball:

```
R CMD check ViewTemps
```

6. If there are errors, make the necessary corrections, and repeat the build and check steps.

If new functions and/or data are added later, use `prompt()` to make skeletons that can be edited to create the new files.

To compile the package into a zip file on a Windows system, go to a DOS prompt in the parent directory and type

```
R CMD build --binary ViewTemps
```

Namespaces

Packages can have their own namespaces, with private functions and classes that are not ordinarily visible from the command line, or from other packages. For example, the function `intervals.lme()` that is part of the *nlme* package must be called via the generic function `intervals()`.

14.13 Document preparation – `Sweave()` and `xtable()`

Recall that R implements the S language. Hence the name `Sweave()` for the function that will now be described, and hence the reference to S code. According to the help page for `Sweave()`:

> 'Sweave' provides a flexible framework for mixing text and S code for automatic report generation. The basic idea is to replace the S code with its output, such that the final document only contains the text and the output of the statistical analysis.

The present description assumes use of the LaTeX markup language, i.e., R code is added into a LaTeX manuscript, creating an Sweave document. The Sweave file `sec1-1.Rnw`, available from the web page for this book, can be used to reproduce a close approximation to Section 1.1, including Figure 1.1 and Table 1.1. The file `sec1-1.Rnw` is in LaTeX markup format, supplemented with Sweave markup commands and associated R code that controls the inclusion of code and/or output, the figure and the table. To process this file, place it in the working directory, and type, from the R command line:

```
Sweave("sec1-1.Rnw", keep.source=TRUE)
  # actually, Sweave("sec1-1") is sufficient
  # The argument keep.source=TRUE preserves the code layout
  # and ensures that comments are retained.
```

This will write the LaTeX file `sec1-1.tex` to the working directory. Additionally, the graphics file that is needed for Figure 1.1 will be generated by the R code and placed in that directory. Providing LaTeX is installed, the file can then be processed to give PDF or postscript output that will pretty much reproduce Section 1.1.

Use of the Sweave document processing system makes it straightforward to regenerate a report if the input data and/or the code change. Gentleman and Lang (2004) argue for

making research results available in this form as a matter of standard practice, thus making research more genuinely reproducible.

The LaTeX system must be able to find the file `Sweave.sty`. If it is necessary to locate this file, type

```
R.home(component="share/texmf/Sweave.sty")
```

The Sweave syntax is based on the Noweb literate programming syntax. See `help(Sweave)` and Leisch (2002) for further details. For details of the version of LaTeX that is recommended for Microsoft Windows, go to `www.miktex.org/`

14.14 Further reading

The definitive document on R's implementation of the S language is the relevant current version of the R Language Definition (R Development Core Team). Recent texts that give comprehensive accounts of R syntax and semantics are Chambers (2007), Gentleman (2008). See also Braun and Murdoch (2007) and Muenchen (2008). Spector (2008) is a useful source of information on data manipulation. Murrell (2009) is a very useful reference on HTML, XML, data formats (plain text, binary, and other), databases, the SQL query language, and regular expressions. There is a chapter on "Data Processing Using R".

For citing R in a publication, use R Development Core Team (2009b). For historical background, see Ihaka and Gentleman (1996). The R function `citation()` can be used to obtain citation information, both for R itself and for any installed package, thus:

```
citation()
citation("DAAG")
```

Vignettes

Vignettes are (currently) pdf documents that describe the abilities in packages for R, and that can be accessed using the `vignette()` function. Many packages have no vignettes, while some have several. To get a list of vignettes in all installed packages, type

```
vignette()           # All vignettes in all installed packages
```

To get name(s) of vignette(s), if any, for specific packages, type, e.g.

```
vignette(package="grid")
```

The package *grid* had, at the time of writing, 13 vignettes.

Assuming that a pdf viewer (perhaps Acroread) is installed, the following displays the vignette "viewports":

```
vignette("viewports")   # Equivalent to vignette(topic="viewports")
```

References for further reading

Braun, W. J. and Murdoch, D. J. 2007. *A First Course in Statistical Programming with R.*
Chambers, J. M. 2007 *Software for Data Analysis: Programming with R.*

Gentleman, R. 2008. *R Programming for Bioinformatics*.
Ihaka, R. and Gentleman, R. 1996. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299–314.
Muenchen, R. A. 2008. *R for SAS and SPSS Users.*
Murrell, P. 2009. *Introduction to Data Technologies.*
R Development Core Team. R *Language Definition.* Available from CRAN sites.
R Development Core Team. 2004b. R: *a language and environment for statistical computing.* Spector, P. 2008. *Data Manipulation with* R.

14.15 Exercises

1. Compare the different outputs from `help.search("print")`, `apropos(print)`, and `methods(print)`. Look up the help for each of these three functions, and use what you find to explain the different outputs.

2. Identify as many R functions as possible that are specifically designed for manipulations with character strings.

3. Test whether `strsplit()` is vectorized, i.e., does it accept a vector of character strings as input, then operating in parallel on all elements of the vector?

4. For the data frame `Cars93`, get the information provided by `summary()` for each level of Type. [Use `split()`.]

5. Determine the number of cars, in the data frame `Cars93`, for each Origin and Type.

6. In the data frame `Insurance` (`MASS` package):

   (a) determine the number of rows of information for each age category (`Age`) and car type (`Group`);
   (b) determine the total number of claims for each age category and car type.

7. Enter the following, and explain the steps that are performed to obtain the result:

```
## Use of split() and sapply(): data frame science (DAAG)
with(science, sapply(split(school, PrivPub),
                     function(x)length(unique(x))))
```

8. Save the objects in your workspace, into an image (`.RData`) file, with the name `archive.RData`. Then remove all objects from the workspace. Demonstrate how, without loading the image file, it is possible to list the objects that were included in `archive.RData` and to recover a deleted object that is again required.

9. Determine the number of days, according to R, between the following dates:

   (a) January 1 in the year 1700, and January 1 in the year 1800;
   (b) January 1 in the year 1998, and January 1 in the year 2007.

10.* The following concatenates (x, y) data that are random noise to data pairs that contain a "signal", randomly permutes the pairs of data values, and finally attempts to reconstruct the signal:

```
### Thanks to Markus Hegland (ANU), who wrote the initial version
##1 Generate the data
```

```
  cat("generate data \n")
  n <- 800           # length of noise vector
  m <- 100           # length of signal vector
  xsignal <- runif(m)
  sig <- 0.01
  enoise <- rnorm(m)*sig
  ysignal <- xsignal**2+enoise
  maxys <- max(ysignal)
  minys <- min(ysignal)
  x <- c(runif(n), xsignal)
  y <- c(runif(n)*(maxys-minys)+minys, ysignal)
  # random permutation of the data vectors
  iperm <- sample(seq(x))
  x <- x[iperm]
  y <- y[iperm]
  # normalize the data, i.e., scale x & y values to
  # lie between 0 & 1
  xn <- (x - min(x))/(max(x) - min(x))
  yn <- (y - min(y))/(max(y) - min(y))
##1 End

##2 determine number of neighbors within
# a distance  <= h = 1/sqrt(length(xn))
  nx <- length(xn)
  # determine distance matrix
  d <- sqrt( (matrix(xn, nx, nx) - t(matrix(xn, nx, nx)) )**2 +
             (matrix(yn, nx, nx) - t(matrix(yn, nx, nx)) )**2 )
  h <- 1/sqrt(nx)
  nnear <- apply(d <= h, 1, sum)
##2 End

##3 Plot data, with reconstructed signal overlaid.
  cat("produce plots \n")
  # identify the points which have many such neighbors
  ns <- 8
  plot(x,y)
  points(x[nnear > ns], y[nnear > ns], col="red", pch=16)
##3 End
```

(a) Run the code and observe the graph that results.

(b) Work through the code, and write notes on what each line does.
[The key idea is that points that are part of the signal will, on average, have more near neighbors than points that are noise.]

(c) Split the code into three functions, bracketed respectively between lines that begin `##1`, lines that begin `##2`, and lines that begin `##3`. The first function should take arguments `m` and `n`, and return a list `xy` that holds data that will be used subsequently. The second function should take vectors `xn` and `yn` as arguments, and return values of `nnear`, i.e., for each point, it will give the number of other points that lie within a circle with the point as center and with radius `h`. The third function will take as arguments `x`, `y`, `nnear`, and

the constant `ns` such that points with more than `ns` near neighbors will be identified as part of the signal. Run the first function, and store the output list of data values in `xy`.

(d) Run the second and third functions with various different settings of `h` and `ns`. Comment on the effect of varying `h`. Comment on the effect of varying `ns`.

(e) Which part of the calculation is most computationally intensive? Which makes the heaviest demands on computer memory?

(f) Suggest ways to make the calculation more efficient.

11. Try the following, for a range of values of `n` between, e.g., 2×10^5 and 10^7. (On systems that are unable to cope with such large numbers of values, adjust the range of values of `n` as necessary.)

```
n <- 10000; system.time(sd(rnorm(n)))
```

The function `system.time()` returns the user and system cpu times, and the elapsed time. Plot each of these numbers, separately, against `n`. Comment on the graphs. Is the elapsed time roughly linear with `n`? Try the computations both for an otherwise empty workspace, and with large data objects (e.g., with 10^7 or more elements) in the workspace.

12. The *plyr* package has functions that provide powerful and flexible data manipulation abilities. Functions have names in the style of `aaply()`, `adply()`, and so on, where the first letter of the name indicates the class of object that will be used as input (for `adply()` it is `a` = array), and the second letter denotes the output class (for `adply()` it is `d` = data frame). In addition to array and data frame, note `l` = list.

Try the following, and in each case explain the result:

```
library(plyr)
aaply(.data=UCBAdmissions, .margins=1:2, .fun=sum)
adply(.data=UCBAdmissions, .margins=1:3, .fun=identity)
daply(.data=tinting, .variables=c("sex","agegp"), .fun=length)
```

15

Graphs in R

This chapter starts with comments on the graphical devices that are available in R, on font families and fonts, on plotting symbols, and on R's color choices. It discusses the abilities in the *lattice* package in modest detail, with a more cursory discussion of the *ggplot2* package.

15.1 Hardcopy graphics devices

The following writes the graph to the pdf file `fossilfuel.pdf`:

```
pdf(file="fossilfuel.pdf", width=6.5, height=6.5, pointsize=18)
  # For pdf() and postscript(), heights and widths are in inches
plot(carbon ~ year, data=fossilfuel, pch=16)
dev.off()
```

Other functions that open hardcopy graphics devices include `png()`, `jpeg()`, `bmp()`, and `tiff()`. Unless the default `units="px"` is changed, height and width are in pixels. Use `dev.off()` to close the device, thus making the file available for display, or for printing, or for incorporation into a document. For a complete list of devices, and further details of specific devices, see `help(Devices)`.

Subsection 1.5.4 described commands used to open graphics windows on commonly used implementations of R. Use `dev.copy()` to copy a graph from the display that is currently active to a hardcopy graphics device. For example:

```
plot(carbon ~ year, data=fossilfuel, pch=16)  # Display graph
## Now open pdf device and copy graph to it
dev.copy(pdf, file="fossilcopy.pdf")
dev.off()             # Close pdf device
```

15.2 Plotting characters, symbols, line types, and colors

Setting `pch` to one of the numbers 0, 1, . . . , 25 gives one of 26 different plotting symbols. In addition, assuming that a single-byte character set is in use, `pch` may be set to any value in the range 32–255. (Multi-byte character sets are needed, e.g., for Asian languages.) Figure 15.1 gives the full range of characters for `font=1`. An alternative is to use a quoted string, e.g., `pch="a"`. For plotting, characters are centered vertically about their mid-position.

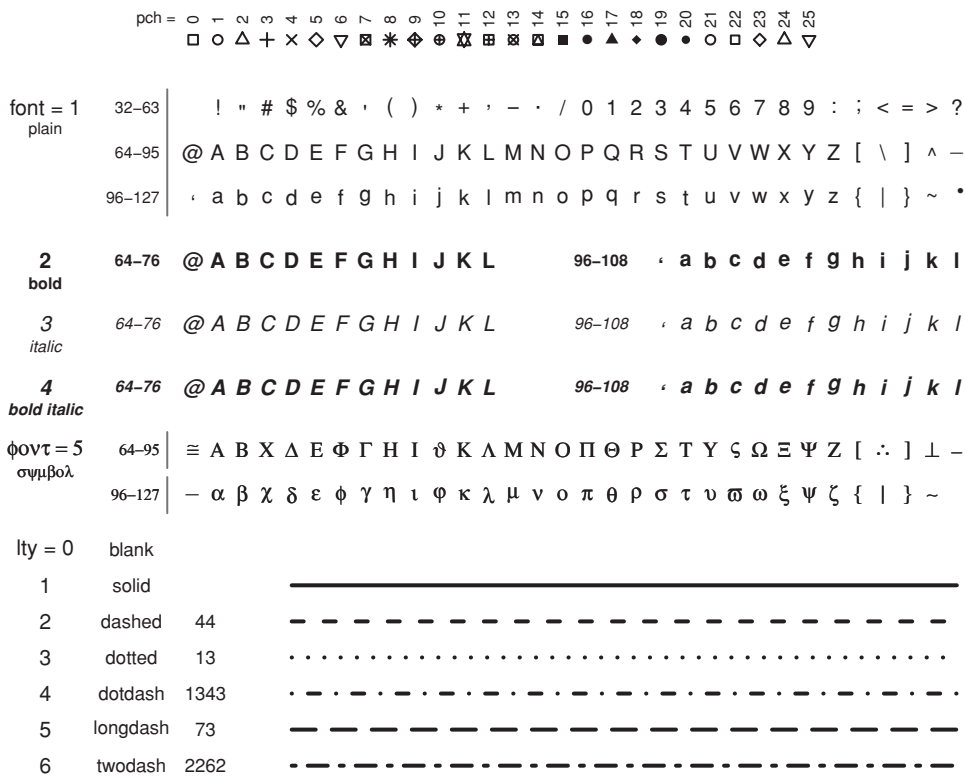


Figure 15.1 Fonts, symbols, line types, and colors, created using the default postscript/pdf encoding (see `help(postscript)` for details). For characters in the range 32–127, symbols may vary depending on the family and on the device. Here, the font family is `Helvetica`, which is the default sans serif font (`par(family="")` or `par(family="sans")`) for a postscript or pdf device. Observe that the "y" in "symbol" has translated, in the symbol font, to ψ. Line types may be specified by number, by name (`lty="dashed"`, ...), or by code (`lty="44"`, ...).

There are six line types that can be specified by number (0, 1, ..., 6) or by name (`"blank"`, `"solid"`, ...). Use of a number sequence that is enclosed in quotes allows a more flexible specification of line types. Thus `"33"` describes a length of 3 units that is drawn, followed by 3 units that are skipped. Up to 4 (draw, skip) pairs are allowed, i.e., a total of 8 numbers. Units are, on most devices, line widths.

Footnote 4 in Subsection 3.2.2, and Plate 1, show the use of `polygon()`. The argument `col` specifies the fill color (if any). Use `border` to change the border color from the default.

The function `symbols()` offers a choice of circles, squares, rectangles, stars, thermometers, and boxplots. Supply `fg` (foreground) to specify the color of the symbol outline, and `bg` (background) for the fill color. See `help(symbols)` for details.

Font families

A font family is a collection of device-specific font faces. Within any font family, the numbers 1–4 identify, wherever possible, a specific face. Font 1 is plain text, font 2 is bold

face, font 3 is italic, and font 4 is bold italic. Font 5 is a special setting that specifies the symbol font (in Adobe symbol encoding).

Generic family names that can be specified using the `family` argument to the `par()` function are: `"sans"` (on a vanilla setup, the default), `"serif"`, `"mono"`, and `"symbol"`. On devices that honor this setting, these are in each case mapped to device-specific families. Thus `Helvetica` is the default device-specific `"sans"` family for postscript or pdf devices, where for Windows (`win.graph()`) it is `Arial`.

Colors

The default color palette, shown in Plate 10, attaches the numbers $1, \ldots, 8$ to the eight colors in the default palette. The palette can be changed as required, perhaps to one of the other built-in palettes.

The `colors()` function returns the names of 657 named colors. Many are different shades of the same basic color. Thus `"red"`, `"red1"`, `"red2"`, `"red3"`, and `"red4"` are different shades of red. The function `show.colors()` (*DAAG*) can be used to show various selections of colors.

Plate 10 shows sets of graduated colors from `heat.colors()`, `topo.colors()`, `cm.colors()`, `rainbow()` (not usually a good choice), and `hcl()`. The hue, chroma, and lightness representation of `hcl()` may be a better starting point for generation of graduated colors than the red, green, and blue representation of `rgb()`. There are also various palettes from the *RColorBrewer* package, intended primarily for use in coloring maps.

Plate 11 shows several palettes, created using `dichromat()` (*dichromat* package). These are designed so that individuals with one of two common types of red–green color blindness can distinguish the colors. The final several rows simulate the effects of such red–green color blindness, first for the default palette plus `"green"`, and then for the *dichromat* `Categorical.12` palette. The color `"green3"`, which has replaced `"green"` in the default palette, does appear different from `"red"`.

Contour and filled contour plots

The functions `contour()`, `filled.contour()`, and `image()` all take arguments `x` and `y` that are coordinates, and `z` that is a matrix of contour level values. The matrix `z` has one row for each element of `x`, and one column for each element of `y`.

Alternatively, the argument `x` may be a list, with elements the vector `x$x`, the vector `x$y`, and the matrix `x$z`. Values of `x` and `y` must be in increasing order of `x` and of `y` within `x`.

Both `filled.contour()` and `image()` use a color scale to show the levels. Type `demo(image)` to see some of the visually appealing possibilities.

15.3 Formatting and plotting of text and equations

Subsection 1.5.5 demonstrated the formatting and plotting of mathematical symbols and formulae, either on their own or as part of character strings. The relevant expressions can be plotted using `text()`, `mtext()`, `title()`, `legend()`, or another such command.

Figures in earlier chapters that used these abilities are: 5.3D ($\sqrt{\text{Residual}}$ as the y-axis label), 10.7 (the y-axis label mixes text and mathematical symbols), and Plate 1B (mixes text and mathematical symbols). Below, various refinements will be described.

15.3.1 Symbolic substitution of symbols in an expression

Use of `substitute()`, in place of `expression()`, allows the character string to be given in symbolic form:

```
library(DAAG)
## Replace the symbol tx by its value
specnam <- "Acmena smithii"
plot(wood ~ dbh, data=rainforest, subset=species==specnam)
title(main=substitute(italic(tx) * ": " * "wood vs dbh",
                      list(tx=specnam)))
```

15.3.2 Plotting expressions in parallel

The function `expression()` calculates one expression for each argument that is supplied to it, thus:

```
## Sample from bivariate normal (x, y) with correlation rho
rho <- 0.75; x <- rnorm(40); y <- rnorm(40) + rho/sqrt(1-rho^2)*x
## Now map all values of x onto 3 integer values
x0 <- cut(x, breaks=c(-Inf,-1,1,Inf))
plot(unclass(x0), y, xaxt="n")
axis(3, at=1:3, labels=expression("(-Inf,-1]", "(-1,1]", "(1, Inf]"))
```

The following, more automated, alternative uses symbolic substitution in parallel to replace "Inf" by the symbol ∞:

```
tiklab <- lapply(strsplit(levels(x0), "Inf"),
              function(x)if(length(x)>1)substitute(x1*infinity*x2,
                                  list(x1=x[1],x2=x[2])) else x)
axis(1, at=1:3, labels=as.expression(tiklab))
```

Symbolic substitution in parallel

When a logarithmic scale is requested, and the axis labels are not otherwise specified, lattice functions label the scale in powers of the logarithmic base, by default 10.

Figure 15.2 demonstrates the use of a powers-of-ten style of labeling for a plot that uses base graphics. The function `lapply()` is used, applying an anonymous function that does the substitution in parallel to each element of the vector `log10ticks`, to yield the list object `ticklabs`.

```
seps <- c(-12.5, -10.523, -8.523, -6.85, -6.6, -3.523, -1, 1, 2.5)
plot(range(10^seps), c(0, 1), axes=FALSE, xlab="Wavelength (m)",
     ylab="", type="n", xaxs="i", yaxs="i", log="x")
log10ticks <- pretty(seps,6)
log10ticks <- log10ticks[log10ticks > seps[1]]
```

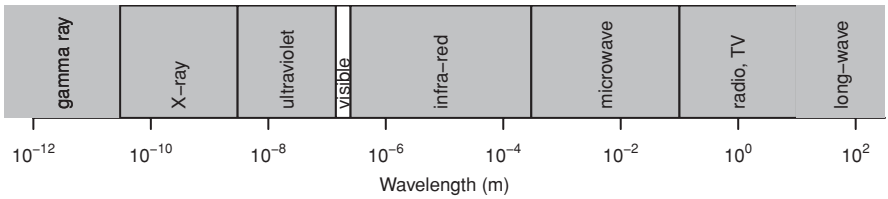


Figure 15.2 The code for this graph uses symbolic substitution to label the tick marks, on a logarithmic scale, in powers of 10.

```
ticklabs <- lapply(round(log10ticks,2),
                   function(x)substitute(10^a, list(a=x)))
axis(1, at=10^log10ticks, labels=as.expression(ticklabs))
## Use rect() to divide the axis a/c to types of radiation
len <- length(seps)
rect(xleft=10^seps[-len], ybottom=rep(0, len-1),
     xright=10^seps[-1], ytop=rep(1, len-1),
     border=c(NA, rep(1,len-3), NA),
     col=c(rep("gray70",3), "white", rep("gray70",4)))
radtypes <- c("gamma ray","X-ray","ultraviolet", "visible",
              "infra-red", "microwave", "radio, TV", "long-wave")
mid <- 0.5*(seps[-1]+seps[-len])
text(10^mid, rep(0.08, len), radtypes, adj=c(0,0.5), srt=90)
```

A further use for `substitute()`

The function `substitute()` may, additionally, be used to extract the names of the actual function arguments when they are passed at run time. This can be useful for graphical annotation. The following demonstrates the syntax:

```
> testfun <- function(x, y)deparse(substitute(x))
> testfun(x = AnyValidName)
[1] "AnyValidName"
```

For further details of the formatting and plotting of symbols and expressions, see the help pages `plotmath`, `expression`, `substitute`, and `bquote`.

15.4 Multiple graphs on a single graphics page

Note first, for comparison, the use of the base graphics parameter `fig` to mark out a rectangular region where the graph will appear. Thus, Figure 2.4A in Subsection 2.1.2 used:

```
par(fig = c(0, 1, 0.38, 1))  # xleft, xright, ybottom, ytop
## Plot graph A
par(fig = c(0, 1, 0, 0.38), new=TRUE)
## Plot graph B
par(fig = c(0, 1, 0, 1))     # Reset to default
```

The initial `par(fig = c(0, 1, 0.38, 1))` marked out a plot region that occupied the total width of the graphics page, started 38% of the way up, and extended to the top of the page. The subsequent `par(fig=c(0, 1, 0, 0.38), new=TRUE)` marked out the lower 38% of the page. The effect of `new=TRUE` is, counter-intuitively, "assume a new page is already open; do not open a new page".

For lattice graphs, the location of the graph can be determined by the argument `position`, when `print()` is called. The following simplified version of the code for Figure 2.8A in Subsection 2.1.4 demonstrates its use:

```
library(lattice)
cuckoos.strip <- stripplot(species ~ length, xlab="", data=cuckoos)
print(cuckoos.strip, position=c(0,0.5,1,1))
                        # x, y, x, y; left, bottom, right, top
cuckoos.bw <- bwplot(species ~ length, xlab="", data=cuckoos)
print(cuckoos.bw, position=c(0,0,1,0.5), newpage=FALSE)
  # NB: Use newpage=FALSE in order to retain the first graph
```

Base and trellis plots on the same graphics page

The following uses the base graphics command `mtext()` to label a lattice plot:

```
plot(0:1, 0:1, type="n", bty="n", axes=FALSE, xlab="", ylab="")
mtext(side=3, line=3, "Lattice bwplot (i.e., boxplot)")
cuckoos.bw <- bwplot(species~length, data=cuckoos)
print(cuckoos.bw, newpage=FALSE)
```

Use of `layout()` *– base graphics only*

The function `layout()` sets up a set of rectangular regions on the graphics page, in advance of any plotting. Use `layout.show(n)` to show where the first `n` regions are located on the page. Be sure to save the graphics parameters at the start, for later resetting.

15.5 Lattice graphics and the *grid* package

Subsection 1.5.8 gave an introductory account of lattice plots. Further details that will now be discussed include: customization of arguments, creation of keys or legends, interaction with plots, the use of panel functions to supplement plots, and the "printing" of multiple lattice plots on the one graphics page.

Lattice graphics versus base graphics

Lattice functions create trellis objects. They do not themselves plot (or "print") the graph. Objects can be created even if no device is open. Such objects can be updated. Objects are plotted (by this time, a device must be open), either when the object is returned to the command line (this implicitly invokes `print()`), or by the explicit use of `print()`.

By successively updating a trellis graphics object, it can be built up and/or modified in steps. Additionally, it is possible to make additions to a "printed" or displayed graphics page. Subsection 15.5.5 will show how to do this.



Figure 15.3 Australian and New Zealand apparent per person annual consumption (in liters) of the pure alcohol content of liquor products, for 1998 to 2006.

15.5.1 Groups within data, and/or columns in parallel

Here are selected lines from the data set `grog` (*DAAG* package):

	Beer	Wine	Spirit	Country	Year
1	5.24	2.86	1.81	Australia	1998
2	5.15	2.87	1.77	Australia	1999
....					
9	4.57	3.11	2.15	Australia	2006
10	4.50	2.59	1.77	NewZealand	1998
11	4.28	2.65	1.64	NewZealand	1999
....					
18	3.96	3.09	2.20	NewZealand	2006

There are three liquor products (drinks), in different columns, and two countries, occupying different rows that are indexed by the factor `Country`. The function `xyplot()` can accommodate any of the following:

- Different symbols and/or colors, in the one panel, distinguish drinks. Different panels distinguish countries, as in Figure 15.3.[1] If different countries are in the same panel, then drinks must be separated across different panels.
- Use a 3 drinks $\times$ 2 countries, or 2 countries $\times$ 3 drinks layout of panels.

Where plots are superposed in the one panel and, e.g., regression lines or smooth curves are fitted, this is done separately for each different set of points. Different colors, and/or different symbols and/or line styles, can be used to make the necessary distinctions.

Code for Figure 15.3 is:

```
## Simple version of plot
grogplot <- xyplot(Beer+Spirit+Wine ~ Year | Country, data=grog,
```

[1] The data (data set `grog`, from *DAAG*) are 1998–2006 Australian and New Zealand apparent per person annual consumption (in liters) of the pure alcohol content. Data, based on Australian Bureau of Statistics and Statistics New Zealand figures, are obtained by dividing estimates of total available alcohol by number of persons aged 15 or more.

```
                    outer=FALSE, auto.key=list(columns=3))
## Enhance, and print enhanced code
update(grogplot, ylim=c(0,5.5),
       xlab="", ylab="Amount consumed (per person)",
       par.settings=simpleTheme(pch=c(1,3,4)))
```

The footnote has alternative code that updates the object, then uses an explicit `print()`.[2] Observe that:

- Use of `Beer+Spirit+Wine` gives plots for each of `Beer`, `Spirit`, and `Wine`. The effect of `outer=FALSE` is that these appear in the same panel.
- Conditioning by country (`| Country`) gives separate panels for separate countries.
- The call to `simpleTheme()` sets up a "theme" that supplies point and line settings.

For separate panels for the three liquor products (different levels of `Country` can now use the same panel), specify `outer=TRUE`:

```
xyplot(Beer+Spirit+Wine ~ Year, groups=Country, outer=TRUE,
       data=grog, auto.key=list(columns=2) )
```

Here is a summary:

Overplot (a single panel)	Separate panels
Break data down a/c to levels of the factor `Country`:	
`Beer ~ Year, groups=Country`	`Beer ~ Year | Country`
Plot columns in parallel, as in `Beer+Wine+Spirit ~ Year`:	
`outer=FALSE`	`outer=TRUE`

Keys – `auto.key`, `key`, *and* `legend`

The argument `auto.key=TRUE` gives a basic key that identifies colors, plotting symbols, and names for the groups. For greater flexibility, `auto.key` can be a list. Settings that are often useful are:

- `points`, `lines`: in each case set to `TRUE` or `FALSE`.
- `columns`: number of columns of keys.
- `x` and `y`, which are co-ordinates with respect to the whole display area. Use these with `corner`, which is one of `c(0,0)` (bottom left corner of legend), `c(1,0)`, `c(1,1)`, and `c(0,1)`.
- `space`: one of `"top"`, `"bottom"`, `"left"`, `"right"`.

The argument `auto.key` sets up a call `key=simpleKey()`. If not otherwise specified, colors, plotting symbols, and line type use the current settings for the device. The argument `text` has `levels(groups)` as its default. If necessary, use `legend=NULL` when updating, to remove an existing key and allow the addition of a new key.

[2]
```
## Update trellis object, then print
frillyplot <- update(grogplot, xlab="", ylab="Amount consumed (per person)",
                     ylim=c(0,5.5), par.settings=simpleTheme(pch=c(1,3,4)))
print(frillyplot)
```

15.5.2 Lattice parameter settings

In general, use `themes` to make point, line, and fill color settings. Use the `scales` argument, in the call to the lattice function, for axes, tick marks, and tick labels.

Point, line, and fill color settings, using `simpleTheme()`

The function `simpleTheme()` creates a "theme", i.e., a list of parameter settings, in a form that can be supplied: (i) in the argument `par.settings` in the graphics function call; or (ii) in the argument `theme` in a call to `trellis.par.set()`, prior to calling the graphics function.

A further possibility is to include an argument `theme` when using `trellis.device()` to start a new device. The function `trellis.device()` has the default argument `retain=FALSE`. Parameters that are not specified as part of the `theme` are reset to their defaults for the relevant device.

The following creates two "themes":

```
settings1 <- simpleTheme(pch = c(1,3,4), cex=1.5)
settings2 <- simpleTheme(pch = c(1,3,4), alpha=0.75)
```

The `settings1` theme may be appropriate when the number of points is small, while `settings2` may be appropriate when there are many points and there is extensive overplotting. Here, `alpha` controls the background transparency (cf., also, `alpha.points` and `alpha.line`). Use of a value less than 1 helps in showing the density of points in regions where there is extensive overlapping.

The following gives the symbols and size of symbol used in Figure 15.3:

```
grogplot0 <- xyplot(Beer+Spirit+Wine ~ Year | Country, outer=FALSE,
                    data=grog, ylim=c(0,5.5))
grogplot1 <- update(grogplot0, par.settings=settings1)
  # settings1 are then stored with the object grogplot1
print(grogplot1)
```

In the following, `trellis.par.set()` changes the settings globally, so that they remain in place until there is a further change or a new device is opened:

```
trellis.par.set(settings2)
```

Then, `print(grogplot1)` will use the `settings1` parameters that are stored as part of the object, while `print(grogplot0)` will use the global `settings2`.

Settings that are not available using `simpleTheme()`

For changes that go beyond what `simpleTheme()` allows, it is necessary to know the names under which settings are stored. To inspect these, type

```
> names(trellis.par.get())
 [1] "fontsize"          "background"        "clip"
. . .
[28] "par.sub.text"
```

For a visual display that shows default settings for points, lines, and fill color, enter

```
trellis.device(color=FALSE)
show.settings()
trellis.device(color=TRUE)
show.settings()
```

The following sets the font size. Notice the separate settings for `text` and `points`.

```
trellis.par.set(list(fontsize = list(text = 7, points = 4)))
```

Parameters that affect axes, tick marks, and axis labels

These are readily manipulated by use of the `scales` argument to the lattice function. Figure 2.10B in Subsection 2.1.5 plotted quarterly labor force numbers, in six regions of Canada, over 1995–1996. The code was:

```
## Create a simplified version of the graphics object
jobsB.xyplot <-
  xyplot(Ontario+Quebec+BC+Alberta+Prairies+Atlantic ~ Date,
         data=jobs, type="b", layout=c(3,2), ylab="Number of jobs",
         scales=list(y=list(relation="sliced", log=TRUE)),
         outer=TRUE)
## Update jobsB.xyplot, with various enhancements
ylabpos <- exp(pretty(log(unlist(jobs[,-7])), 100))
ylabels <- paste(round(ylabpos),"\n(", log(ylabpos), ")", sep="")
## Create a date object 'startofmonth'; use this instead of 'Date'
atdates <- seq(from=95, by=0.5, length=5)
datelabs <- format(seq(from=as.Date("1Jan1995", format="%d%b%Y"),
                   by="6 month", length=5), "%b%y")
update(jobsB.xyplot, xlab="", between=list(x=0.5, y=0.5),
       scales=list(x=list(at=atdates, labels=datelabs),
                   y=list(at=ylabpos, labels=ylabels), tck=0.6) )
```

The enhancements are:

- The *y*-axis labels show number of jobs, with log(number) in parentheses underneath.
- Dates of the form `Jan95` label the *x*-axis. See further Subsection 14.5.4.
- Tick marks are reduced in length (`tck=0.6`, i.e., 60% of the default).

Notice also the use of `between=list(x=0.5, y=0.5)` to add horizontal and vertical space between the panels, ensuring that the tick labels do not overlap.

An example – the `ais` *data set*

Data in the data set `ais` (*DAAG*) were shown earlier, in Figure 1.3 in Subsection 1.5.8. Data were collected with a view to studying possible differences in blood characteristics, between athletes in endurance-related events and those in power-related events. See the help page for `ais` for details of the measurements, including a variety of blood cell counts. Here is a breakdown, by sex and sport, of numbers:

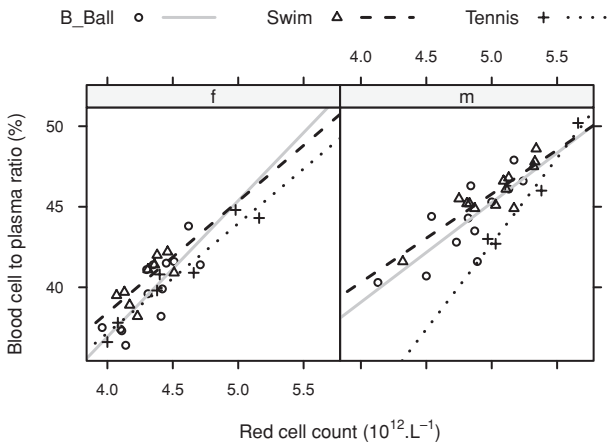


Figure 15.4 Blood cell to plasma ratio (`hc`) versus red cell count (`rcc`), by `sex` (different panels) and `sport` (distinguished within each panel), for a subset of the `ais` data. The argument `type=c("p", "r")` displays both points (`"p"`) and regression lines (`"r"`).

```
> with(ais, table(sex,sport))
   sport
sex B_Ball Field Gym Netball Row Swim T_400m T_Sprnt Tennis W_Polo
  f     13     7   4      23  22    9     11       4      7      0
  m     12    12   0       0  15   13     18      11      4     17
```

There are 202 athletes in total, all from the Australian Institute of Sport.

Figure 15.4 plots blood cell to plasma ratio (%) against red cell count, for three sports only. The three sports are in the one panel, distinguished by different symbols and/or colors. Females and males are in separate panels. Regression lines have been added. Again, an initial basic plot is updated to give the desired result. Code is:

```
basic1 <-  xyplot(hc ~ rcc | sex, groups=sport[drop=TRUE],
                  data=subset(ais, sport %in% c("B_Ball", "Swim",
                                                "Tennis")),
                  ylab="Blood cell to plasma ratio (%)")
parSet <- simpleTheme(pch = c(1:3), lty=1:3, lwd=1.5,
                      col.line=c("gray40","black","black"))
xaxlab <- expression("Red cell count (10"^{12}*"."*L^{-1}*")")
basic2 <- update(basic1, par.settings=parSet,
                 auto.key=list(columns=3, lines=TRUE),
                 scales=list(tck=0.5), xlab=xaxlab)
print(update(basic2, type=c("p", "r")))
```

Again, there are details that require explanation:

- The `drop=TRUE` in `groups=sport[drop=TRUE]` ensures that levels that are no longer present in the data (here, sports other than `B_Ball` and `Swim`) are omitted when the key is drawn. (Subsetting a factor leaves the levels attribute unchanged. Redundant levels must be explicitly removed.)

- As in base graphics, graphical annotation (tick labels, axis labels, labels on points, etc.) can be given using the function `expression()`.
- The argument `type=c("p","r")` gives both points and fitted regression lines.

15.5.3 Panel functions, strip functions, strip labels, and other annotation

Each lattice command that creates a graph has its own panel function. Thus `xyplot()` has the panel function `panel.xyplot()`. The following are equivalent:

```
xyplot(species ~ length, xlab="", data=cuckoos)
xyplot(species ~ length, xlab="", data=cuckoos,
          panel=panel.xyplot)
```

The user's own function can be substituted for `panel.xyplot()`. Panel functions that may be used, either in combination with functions such as `panel.xyplot()` or separately, include:

- `panel.points()`, `panel.lines()`, and a number of other such functions that are documented on the same help page as `panel.points()`;
- `panel.abline()`, `panel.curve()`, `panel.rug()`, `panel.average()`, and a number of other functions that are documented on the same help page as `panel.abline()`.

A panel function that fits and plots parallel lines

The following updates `basic2`, used for Figure 15.4 in Subsection 15.5.2 above, so that the lines for the two sports are parallel. Plate 13 shows the result.

```
update(basic2,
  strip=strip.custom(factor.levels=c("Female","Male")),
  # In place of level names c("f", "m"), use c("Female", "Male")
  panel=function(x, y, groups, subscripts, ...){
     panel.superpose(x,y, groups=groups,
                     subscripts=subscripts, ...)
     ## Obtain fitted values for parallel line model
     parallel.fitted <- fitted(lm(y ~ groups[subscripts] + x))
     panel.superpose(x, parallel.fitted, groups=groups, type="l",
                     subscripts=subscripts, ...)
     })
```

When a `groups` argument is supplied, `panel.xyplot()` calls `panel.superpose()`. The customized panel function calls `panel.superpose()` twice, once to plot the paints, and a second time to join the fitted values and thus generate the lines.

Here is a further example:

```
library(grid)
stripplot(species ~ length, xlab="", data=cuckoos,
          legend=list(top=list(fun=textGrob,
```

```
                        args=list(label="Stripplot", x=0, just="left"))))
  # Here, x=0 is equivalent to x=unit(0,"npc"); the range is (0,1)
```

Modification of the strip labels

The following has modified strip labels. It is in the style of Figure 6.18, but with more elaborate strip labels:

```
tau <- (0:5)/2.5; m <- length(tau); n <- 200; SD <- 2
x0 <- rnorm(n, mean=12.5, sd=SD)  # Generate x-values
df <- data.frame(sapply(tau, function(s)x0+rnorm(n, sd=2*s)))
  # Columns after the first are x-values with addedd error
df$y = 15+2.5*x0
names(df) <- c(paste("X", tau, sep=""), "y")
lab <- c(list("0"),
         lapply(tau[-1], function(x)substitute(A*s[z], list(A=x))))
form <- formula(paste("y ~ ", paste(paste("X", tau, sep=""),
                                    collapse="+")))
xyplot(form, data=df, outer=TRUE,
  strip=strip.custom(strip.names=TRUE, var.name="SD(added err)",
          sep=expression(" = "), factor.levels=as.expression(lab)))
```

The argument `var.name` has text that will appear in all strip labels. See also Subsection 3.3.3, where there was another example of this form of strip labeling.

In the internal code the variables that are printed in parallel become levels of a single factor, as in the following alternative that achieves the same result:

```
library(reshape)
longdf <- melt(df, measure.vars=1:6, id.vars="y",
               variable_name="tau")
  # Columns of "measure" variables are stacked in a column that has
  # the name "value": column 1, then 2, then 3, ...
xyplot(y ~ value | tau, data=longdf,
       strip=strip.custom(strip.names=TRUE,
                var.name="SD(added err)", sep=expression(" = "),
                factor.levels=as.expression(lab)))
```

Annotation using `textGrob()`

The following uses the function `textGrob()` (*grid*) to create a text object which is then supplied to the lattice function:

```
library(grid)
stripplot(species ~ length, xlab="", data=cuckoos,
          legend=list(top=list(fun=textGrob,
               args=list(label="Stripplot of cuckoo data", x=0))))
  # Here, x=0 is equivalent to x=unit(0,"npc"); the range is (0,1)
```

Multiple legends, for example a list element `bottom` as well as a list element `top`, can be supplied by this means.

15.5.4 Interaction with lattice (and other) plots – the playwith package

For using *playwith*, the GTK+ toolkit must be installed. For details, go to the web site http://playwith.googlecode.com/

For installing the *playwith* package type, from the command line:

```
install.packages("playwith", dependencies=TRUE)
```

Now type, for example

```
library(DAAG)
library(playwith)
playwith(xyplot(age ~ distance, data=hotspots),
         labels=hotspots$name)
```

Plate 12 was then obtained as described in the figure caption.

An alternative is

```
gph <- xyplot(age ~ distance, data=hotspots)
playwith(update(gph), labels=hotspots$name)
```

The menu that appears to the left of the graph can be used to initiate single-click identification, to add annotation or arrows, or to mark out a rectangle on the graph for zooming in or out. If labels are not specified, row names are used.

Note also the function `latticist()` in the *latticist* package. When called with a data frame as argument, this opens a window that has graphical summary information on the columns of the data frame. Additionally, the window gives a graphical user interface to the creation of further lattice plots from the data frame. As when `playwith()` is used as a wrapper for a call to a lattice function, various annotation features are available.

Note that `playwith()` can be used, also, for more limited interaction with plots that have been created using basic graphics, or using *ggplot2*.

15.5.5 Interaction with lattice plots – focus, interact, unfocus

As described here, interaction starts with the use of `trellis.focus()` to focus down to the relevant "viewport", by default a panel. It may be called without arguments. If there is only one panel, it is then selected immediately. If there is more than one panel, the user chooses a panel by clicking on it.

Other choices of `name` include `"panel"`, `"strip"`, `name="legend"`, and `"toplevel"`. For `name="legend"`, `side` should be indicated.

Use of `panel.text()` *to label points or add annotation*

Following a call to `trellis.focus()`, panel functions can be used to supplement plots. Use `trellis.panelArgs()` to extract arguments that are available to panel functions following such a call. The following adds text labels:

```
xyplot(Brainwt ~ Bodywt, data=primates)
trellis.focus("panel", row=1, column=1, clip.off=TRUE,
              highlight=FALSE)  # Non-interactive use
```

```
xyetc <- trellis.panelArgs()
panel.text(x=xyetc$x, y=xyetc$y, labels=row.names(primates), pos=3)
trellis.unfocus()
```

For non-interactive use, be sure to call `trellis.focus()` with the argument `highlight=FALSE`.

The following uses `textGrob()` (*grid*) to add labeling to a lattice plot:

```
library(grid)
stripplot(species ~ length, xlab="", data=cuckoos)
trellis.focus(name="toplevel", highlight=FALSE)
panel.text("Stripplot of cuckoo data", x=0.5, y=0.98)
  # x=0.05 translates to x=unit(0.05,"npc"); the range is (0,1)
trellis.unfocus()
```

Use of `panel.identify()` *to label points interactively*

Here is an example of interactive labeling:

```
## Use of xyplot(): data frame tinting (DAAG)
library(lattice)
xyplot(it ~ csoa | sex, data=tinting)
trellis.focus("panel", column=1, row=1)
panel.identify(labels=as.character(tinting$target))
## Now click near points as required.
## Terminate by right clicking inside the panel.
## Now interact with panel 2
trellis.focus("panel", row=1, column=2)
panel.identify(labels=as.character(tinting$target))
## Click, ..., and right click
```

By default, the `x` and `y` arguments to the function `panel.identify()` are taken to be those that were supplied to the lattice function, here `xyplot()`.

Functions that may be called include `panel.lines()` and related functions, and `panel.abline()` and related functions, as described earlier.

15.5.6 Overlaid plots with different scales

The data set `edcT` holds estimates of temperature anomalies `dT` (i.e., differences from the average of approximately −54.5 °C over the past 1000 years), from the EPICA (European Project for Ice Coring in Antarctica) Dome C ice cores that cover 0 to 800 000 years before the present. The data set `edcCO2` (*DAAG*) holds estimates of carbon dioxide levels, over the same time period. Plate 14 overlays mildly smoothed versions of the two data series.

The plot uses the function `doubleYScale()` from the *latticeExtra* package, thus:

```
library(latticeExtra)
library(DAAG)
CO2smu <- with(edcCO2, supsmu(age, co2, span=.005))
tempsmu <- with(edcT, supsmu(Age, dT, span=.001))
```

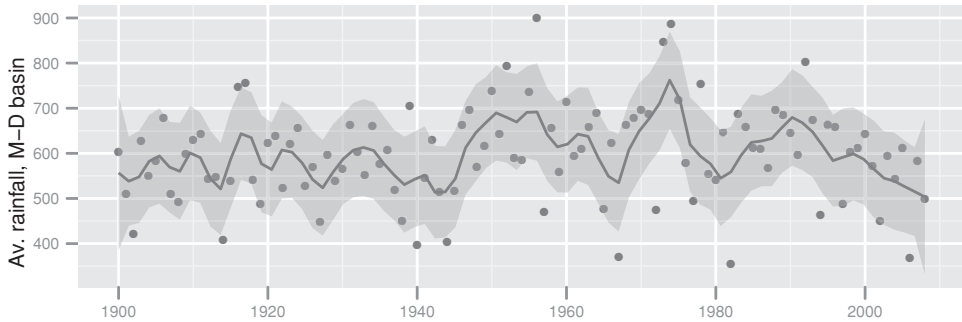


Figure 15.5 Annual rainfall (mm), from 1901 to 2008, for the Murray–Darling basin region of Australia. The curve is fitted using the default loess smoother. The pointwise standard error bands assume that errors about the curve are independent; this is unlikely to be strictly true. To suppress the bands, specify `se=FALSE`.

```
co2_layer <- xyplot(y ~ I(-x), data=CO2smu, type="l",
                    xlab="Years Before Present",
                    ylab = expression(CO[2]*" (ppm)"))
temp_layer <- xyplot(I(y+54.5) ~ I(-x), data=tempsmu, type="l",
                     ylab=expression("Temperature ("*degree*C*")"))
doubleYScale(co2_layer, temp_layer, add.ylab2 = TRUE)
```

Two separate graphics objects, `co2_layer` and `temp_layer`, are created. The function `doubleYScale()` superposes the two graphs. The argument `add.ylab2=TRUE` ensures that the y-label for the second object will appear in the right margin.

15.6 An implementation of Wilkinson's *Grammar of Graphics*

The *ggplot2* syntax implements a *Grammar of Graphics*, as expounded in Wilkinson (2005). It is consistent, but less stylized than *lattice*. As with *lattice*, a graphics object is created, which can then be saved or printed directly on to the graphics page.

Each different type of plot – scatterplot, histogram, density plot, etc. – has a different plot `geom`, or "geometry". Different geometries can be overlaid. This ability to build plots layer upon layer is a powerful feature of *ggplot2*. Examples will be given that demonstrate the syntax.

Australian rain data

Figure 15.5 plots annual rainfall for Australia's Murray–Darling basin region. The following code uses the function `quickplot()`:

```
library(DAAG)
library(ggplot2)
## Default loess smooth, with SE bands added.
quickplot(Year, mbdRain, data=bomregions, geom=c("point","smooth"),
          span=0.5, xlab="", se= TRUE, ylab="Av. rainfall, M-D basin")
```

Arguments such as `size` (e.g., `size=I(2.5)`) and `color` (e.g., `color=I("red")` can be supplied, but note that they affect both points and the added smooth curve. The slightly different result of `size=I(2.5)`, as opposed to `size=2.5`, will be explained below.

The function `quickplot()` (or `qplot()`) generates a sequence of function calls that create the different components of the plot. In order to allow better control of the details of the plot, those calls can be made separately, thus:

```
ggplot(bomregions, aes(x=Year, y=mdbRain)) +
  geom_point() +                        # Specify a scatterplot
  geom_smooth(span=0.5, se=TRUE) +      # Add a smooth curve
  xlab("") +                            # Blank out x-axis label
  ylab("Av. rainfall, M-D basin")
## NB: Note use of aes() to supply x- and y-axis variables
```

The successive "+" operators combine the separate graphical components into a single graphics object. The graph is built up in layers. In Figure 15.5, the first layer has the points, while the second has the smooth curve. Further modifications change the *x*- and *y*-axis labels from the defaults.

In the call to `ggplot()`, the `data` argument is the only mandatory argument. It can be repeated in the call(s) to one or more of the later `geom` functions. Different `geom`s can thus, if required, take their data from different data frames.

Changes to `color` or `size` or `shape` settings can be made separately for each different `geom`. Thus, changing `geom_point()` to `geom_point(size=2.5)` affects only the points. (As the default `size` for points is 2, this increases the size by 25%.)

The function `aes()` maps variables in the data to visual properties ("aesthetics") of `geom`s. In the code just given, the mappings are to the *x*- and *y*-axes of the plot. Other possible mappings are to `color` (use color to distinguish groups within the data), `shape` (distinguish by shape), `size` and `fill`. The choice of `pch` in base graphics becomes, in *ggplot2*, a choice of `shape`.

A further possibility is to use `quickplot()` for a substantial part of the task, then adding graphical components to the object that `quickplot()` has created. Here is a third way to create Figure 15.5:

```
quickplot(Year, mdbRain, data=bomregions, geom="point",
          xlab="", ylab="Av. rainfall, M-D basin") +
  geom_smooth(span=0.5, se=TRUE)
```

However created, the end result is a `ggplot` object. This can be `print`ed immediately, or it can be saved as a named object. The graph is created using the `print` method for a `ggplot` object.

Try also the following:

```
library(splines)
library(quantreg)
quickplot(Year, mdbRain, data=bomregions, geom=c("point", "quantile"),
          formula = y ~ ns(x,5), quantiles=c(0.2,0.5,0.8) )
```

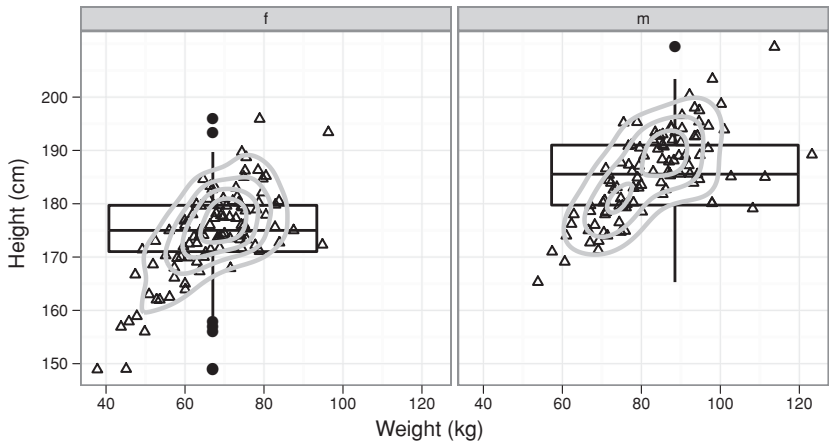


Figure 15.6 Height versus weight, by sex, for Australian athletes in the `ais` data set. Boxplots that show the distributions of heights, and two-dimensional density contours, have been added.

The natural spline basis `ns(x,5)` is supplied to the function that estimates the quantile curves, so that 5 d.f. spline curves are fitted at the 20%, 50%, and 80% quantiles.

Physical measurements of Australian athletes

Figure 15.6 plots height against weight, by sex, for the `ais` data. Additionally, boxplots show the distributions of heights, and there are two-dimensional density contours estimates. The graph is a tad crowded. The following gives a first draft version of the plot:

```
## Overlay scatterplots with boxplots and with density contours
quickplot(wt, ht, data=ais, geom=c("boxplot", "point", "density2d"),
          facets = . ~ sex)
```

To set axis labels, show the boxplot outline in gray, show contour lines in gray (the default is blue), and make various other changes as in Figure 15.6, specify:

```
quickplot(wt, ht, xlab="Weight (kg)", ylab="Height (cm)", data=ais,
          facets = . ~ sex) +
  geom_boxplot(outlier.size=1.75, outlier.colour="gray",
               color="gray") +
  geom_point(shape=2, size=1) +
  geom_density2d(color="gray")
```

The `facets` argument takes the form `row.var ~ col.var`, where `row.var` indexes the rows of panels, `col.var` indexes columns, and "`.`" is used as a placeholder when there is one row or one column only.

Code for further plots will work with a subset of the `ais` data, limiting attention to rowers and swimmers:

```
## Extract from ais the data for rowers and swimmers
aisBS <- subset(ais, sport %in% c("Row","Swim"))
aisBS$sport <- factor(aisBS$sport)
```

The following are equivalent:

```
## Single panel: distinguish sexes by color; sports by shape
## Use quickplot()
quickplot(wt, ht, data=aisBS, geom="point",
          color=sex, shape=sport)
## Follow a call to ggplot() with a call to geom_point()
ggplot(aisBS) + geom_point(aes(wt, ht, color=sex, shape=sport))

## Two panels (sports); sexes have different colors and shapes
quickplot(wt, ht, data=aisBS, geom="point", size=I(2.5),
          color=sex, shape=sex, facets = . ~ sport)
## Single panel: distinguish sexes by color; sports by shape
quickplot(wt, ht, data=aisBS, geom="point",
          color=sex, shape=sport, size=I(2.5))
## Single panel: distinguish sexes by color; sports by size
quickplot(wt, ht, data=aisBS, geom="point", color=sex, size=sport)
```

Possible choices of `geom`, additional to those already demonstrated, are `"path"` (join points), `"line"` (join points), `"histogram"`, and `"density"`.

Aesthetic mappings vs settings

Note the distinction between settings and aesthetic mappings:

	Calls to `quickplot()`	Plots based on `ggplot()`
Settings	`size=I(3)` or `cex=3`	`size=3`
Aesthetic mappings	`size=sport`	`aes(size=sport)`

Use of the argument `size=3` in a call to `quickplot()` does change the point size, but it adds an extraneous key. The same happens if the argument `mapping=aes(size=3)` is supplied to `ggplot()` or to `geom_point()` or to another such function.

In calls to `quickplot()`, `cex` and `size` are synonyms, as are `color` and `colour`. Also `type` is a synonym for `geom`.

Available geometries and settings

Table 15.1 has details of a number of geometries that are available for `ggplot` objects. Table 15.2 lists some of the settings, in addition to those already noted, that are available.

Themes and updates

Note also the following:

```
## Set theme_bw() defaults: black gridlines & white background
## Also set base text size to 8pt
old <- theme_set(theme_bw(base_size=8))
```

Table 15.1 *Some available* `geom`*s.*

`quickplot()`		Available arguments to the `geom` function
`geom=`	`ggplot()`	(`data`, `mapping`, `color`, `fill`, `alpha`, plus ...)
`"point"`	`+ geom_point()`	`size`, `shape`, etc.
`"line"`	`+ geom_line()`	`size`, `linetype`
`"path"`	`+ geom_path()`[a]	`size`, `linetype`
`"smooth"`	`+ geom_smooth()`	`linetype`, `weight`, `se` (`TRUE` or `FALSE`).
`"histogram"`	`+ geom_histogram()`	`linetype`, `weight`
`"density"`	`+ geom_density()`	`weight`, `linetype`, `size`
`"density2d"`	`+ geom_density2d()`	`weight`, `linetype`, `size`

[a] Use `geom_path()` to connect observations, in the original order.

Table 15.2 *Additional settings for* `ggplot` *objects.*

	`quickplot()`	`ggplot(),...`	Notes
		[eg, `+ xlab("myxlab")`]	
Title	`main="mytitle"`	`opts(title="mytitle")`	
Axis labels	`xlab="myxlab"`, etc.	`xlab("myxlab")`, etc [a]	
Facets	`facets=sex~sport`	`facet_grid(sex~sport)`	c.f., "conditioning"
Flip axes	see note[b]	`axis_flip()`	
log axes	see note[b]	`scale_x_log()`	`log` or `log10` or `log2`
Aspect ratio	see note[b]	see note[c]	

[a] Alternatively, use `scale_x_continuous()` or `scale_x_discrete()`, etc. as appropriate.
[b] The function `quickplot()` returns `ggplot` objects. Add plot components just as for any other `ggplot` object.
[c] Use `coord_equal()`. By default (`ratio=1`), 1cm represents the same range along both x- and y-axes.

```
## Reduce default size for geom_point(), from 2 to 1.5
update_geom_defaults("point", aes(size=1.5))
theme_set(old)        # Restore the earlier settings
```

The default theme is `theme_gray()`, called `theme_gray()` because its white gridlines overlay a very light gray background.

Figure 12.3 in Subsection 12.1.2 is a further example of the use of *ggplot2* abilities.

15.7 Dynamic graphics – the *rgl* and `rggobi` packages

Both these packages provide three-dimensional dynamic graphics. The left panel in Figure 6.9 in Subsection 6.3.1 used the abilities of the *rgl* package. Rather than using these directly, novices may find it more convenient to use functions in the *Rcmdr* package – `scatter3d()` and `identify3d()`, which call functions from the *rgl* package.

Here is code that gives an initial plot. By holding down the left mouse button and moving the mouse, this can be rotated to give the projection shown in the left panel of Figure 6.9:

```
## The Rcmdr and rgl packages must be installed
library(Rcmdr)        # This makes scatter3d() available
```

```
with(nihills, scatter3d(x=log(dist), y=log(climb), z=log(time),
                        grid=FALSE, surface.col="gray",
                        point.col="black", axis.scales=FALSE))
with(nihills, identify3d(x=log(dist), y=log(climb), z=log(time),
                         labels=row.names(nihills), col="gray40"))
## NB: Use the middle or right mouse button to drag a rectangle
## around any point that is to be labeled.
```

Following the call to `identify3d()`, use the middle (or maybe right) mouse button to drag a rectangle around any point that is to be labeled. To cease identifying points, make a middle (or right) click on an empty region of the plot. Use `rgl.snapshot()` to save the current plot into a file.

To go immediately to the projection shown in Figure 6.9, precede the call to `scatter3d()` with:

```
open3d(userMatrix= matrix(c(
       1, 0.000,  0.000,    0,
       0, 0.966, -0.259,    0,
       0, 0.259,  0.966,    0,
       0, 0.000,  0.000,    1), ncol=4, byrow=TRUE))
par3d(cex=0.6)       # Optional; see help(par3d) for other parameters
```

The call to `par3d()` is needed only if some change is required to graphics parameters.

The *rggobi* package offers a wider range of features, via an interface to the GGobi system (Cook and Swayne, 2007). For installation details go to `http://www.ggobi.org/`.

15.8 Further reading

Murrell (2005) shows R's graphical abilities off to impressive effect. The code used for the graphs is available from the web page that is given with the reference. For *lattice* graphics see Sarkar (2007). For *ggplot2* see Wickham (2009).

References for further reading

Murrell, P. 2005. *R Graphics*.
`http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html`

Sarkar, D. 2007. *Lattice: Multivariate Data Visualization with R*.
`http://lmdvr.r-forge.r-project.org/figures/figures.html`

Wickham, H. 2009. *ggplot2: Elegant Graphics for Data Analysis*.
`http://had.co.nz/ggplot2/`

Epilogue

The first several chapters of this book were introductory in style, though with more attention to the subtleties of practical application than is common in elementary treatments. The simple models considered here are somewhat restricted in their range of application, because they are based on assumptions of independence and stationarity which will not nearly always be true. The last several chapters treated topics that have traditionally been thought of as more advanced. We hope this has given readers a taste of models for data which can be applied in a wide variety of situations.

Models that are not strictly correct, or even perhaps badly broken, may nevertheless be useful, giving acceptably accurate predictions. The validity of model assumptions remains an important issue. We need to know the limits of the usefulness of our models. Comparing results from a simple model with results from a model that takes better account of the data can help in developing intuition. Somewhat ironically, Chapters 7–13 could be viewed as essential background for those who hope to do a good job of the analyses described in Chapters 3–6! An understanding of multi-level and repeated measures models seems particularly important, since these models bring attention to structure in the noise component which must be accounted for in a proper analysis.

Whether or not faulty assumptions matter will depend on the circumstances. For simpler models, the assumption of independently and identically distributed errors typically makes little difference to estimates of model parameters and to fitted values, but can have a large effect on standard errors. For example, a better model for the frogs data in Chapter 8 would account for spatial correlations. Because we did not take account of spatial autocorrelation in our more standard logistic regression analysis, the standard errors are not very reliable; the best we could do was to make a tentative distinction between coefficients that seemed clearly statistically significant, and those that were not. Modeling the correlation structure would have given us a description that should generalize better to sites in the vicinity of those that were studied, with more believable indications of the accuracy of such a description. For generalizing in time, e.g., to a subsequent year, the benefits are more doubtful. If data from multiple years were available, then for predictive purposes the modeling of the temporal structure should be the priority.

The emphasis in this text has been on careful modeling of the data, using both fixed and random effects as appropriate. This allows maximum flexibility in the subsequent use of the fitted model, whether the aim is scientific understanding or prediction. Predictive accuracy measurement makes its own modeling demands. In general, there may be two models. First, there is a model for the population from which the data have been drawn and

the sampling mechanism. Second, there must be a model for the population and associated sampling mechanism when predictions are made. AIC, BIC, and cross-validation error rates that relate to the data used to develop the model all assume that the two populations and associated sampling mechanisms are the same.

The following situations all occur in practice:

1. The data used to develop the model are, to a close approximation, a random sample from the population to which predictions will be applied. If this can be assumed, a simple use of a resampling method will give an estimate of the score function that is unbiased with respect to the population that is the target for predictions.
2. Test data are available from the target population, with a sampling mechanism that reflects the intended use of the model. The test data can then be used to derive a realistic estimate of predictive accuracy.
3. The sampling mechanism for the target data differs from the mechanism that yielded the data in 1, or yielded the test data in 2. However, there is a model that predicts how predictive accuracy will change with the change in sampling mechanism. Thus, in the attitudes to science data considered in Chapter 10, the predictive accuracy for the mean of a new class depends on the number in the class.
4. The connection between the population from which the data have been sampled and the target population may be weak or tenuous. It may be so tenuous that a confident prediction of the score function for the target population is impossible. In other words, a realistic test set and associated sampling mechanism may not be available. An informed guess may be the best that is available.

These four possibilities are not completely distinct; they overlap at the boundaries. A modeling approach offers a framework of understanding from which to make an informed judgment in all these situations.

References

Methodological References

Agresti, A. 2002. *Categorical Data Analysis*, 2nd edn. John Wiley.

Aitchison, J. 2003. *The Statistical Analysis of Compositional Data*. Blackburn Press.

Aldrich, J. 1995. Correlations genuine and spurious in Pearson and Yule. *Statistical Science* 10: 364–76.

Ambroise, C. and McLachlan, G. J. 2002. Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS* 99: 6262–6.

Andersen, B. 1990. *Methodological Errors in Medical Research: An Incomplete Catalogue*. Blackwell Scientific.

Barnett, V. 2002. *Sample Survey: Principles & Methods*, 2nd edn. Arnold Publishers.

Bartholemew, D. 2004. *Measuring Intelligence. Facts and Fallacies*. Cambridge University Press.

Bates, D. 2005. Fitting linear mixed models in R. *R News* 5(1): 27–30.

Bates, D. M. and DebRoy, S. 2003. Converting a large R package to S4 classes and methods. In K. Hornik, F. Leisch and A. Zeileis (eds), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/

Bates, D. M. and Watts, D. G. 1988. *Nonlinear Regression Analysis and Its Applications*. John Wiley.

Belson, W. A. 1959. Matching and prediction on the principle of biological classification. *Applied Statistics* 8: 65–75.

Berk, R. A. 2008. *Statistical Learning from a Regression Perspective*. Springer.

Bickel, P. J., Hammel, E. A. and O'Connell, J. W. 1975. Sex bias in graduate admissions: data from Berkeley. *Science* 187: 398–403.

Bland, M. and Altman, D. 2005. Do the left-handed die young? *Significance* 2: 166–70.

Bolker, B. M. 2008. *Ecological Models and Data in R*. Princeton University Press.

Box, G. E. P. and Cox, D. R. 1964. An analysis of transformations (with discussion). *Journal of the Royal Statistical Society* B 26: 211–52.

Braun, W. J. and Murdoch, D. J. 2007. *A First Course in Statistical Programming with R*. Cambridge University Press.

Breiman, L. 2001. Statistical modeling: the two cultures. *Statistical Science* 16: 199–215.

Brockwell, P. and Davis, R. A. 2002. *Time Series: Theory and Methods*, 2nd edn. Springer.

Canty, A. J. 2002. Resampling methods in R: the boot package. *R News* 2/3: 2–7.

Carroll, R. 2004. Measuring diet. Texas A & M Distinguished Lecturer Series. http://stat.tamu.edu/~carroll/talks.php

Carroll, R. J., Ruppert, D. and Stefanski, L. A. 2006. *Measurement Error in Nonlinear Models: A Modern Perspective*, 2nd edn. Chapman and Hall/CRC.

Chalmers, I. and Altman, D. G. 1995. *Systematic Reviews*. BMJ Publishing Group.

Chambers, J. M. 2007. *Software for Data Analysis: Programming with R*. Springer.

Chanter, D. O. 1981. The use and misuse of regression methods in crop modelling. In D. A. Rose and D. A. Charles-Edwards (eds), *Mathematics and Plant Physiology*. Academic Press.

Chatfield, C. 2002. Confessions of a statistician. *The Statistician* 51: 1–20.

Chatfield, C. 2003a. *The Analysis of Time Series: An Introduction*, 6th edn. Chapman and Hall.

Chatfield, C. 2003b. *Problem Solving. A Statistician's Guide*, 2nd edn. Chapman and Hall/CRC.

Clarke, D. 1968. *Analytical Archaeology*. Methuen.

Cleveland, W. S. 1981. Lowess: a program for smoothing scatterplots by robust locally weighted regression. *The American Statistician* 35: 54.

Cleveland, W. S. 1993. *Visualizing Data*. Hobart Press.

Cleveland, W. S. 1994. *The Elements of Graphing Data*, revised edn. Hobart Press.

Cochran, W. G. and Cox, G. M. 1957. *Experimental Designs*, 2nd edn. John Wiley.

Collett, D. 2003. *Modelling Survival Data in Medical Research*, 2nd edn. Chapman and Hall.

Cook, D. and Swayne, D. F. 2007. *Interactive and Dynamic Graphics for Data Analysis*. Springer.

Cook, R. D. and Weisberg, S. 1999. *Applied Regression Including Computing and Graphics*. John Wiley.

Cox, D. R. 1958. *Planning of Experiments*. John Wiley.

Cox, D. R. and Reid, N. 2000. *Theory of the Design of Experiments*. Chapman and Hall.

Cox, D. R. and Wermuth, N. 1996. *Multivariate Dependencies: Models, Analysis and Interpretation*. Chapman and Hall.

Cox, T. F. and Cox, M. A. A. 2001. *Multidimensional Scaling*, 2nd edn. Chapman and Hall.

Dalgaard, P. 2008. *Introductory Statistics with* R, 2nd edn. Springer.

Davison, A. C. and Hinkley, D. V. 1997. *Bootstrap Methods and Their Application*. Cambridge University Press.

Dehejia, R. H. and Wahba, S. 1999. Causal effects in non-experimental studies: re-evaluating the evaluation of training programs. *Journal of the American Statistical Association* 94: 1053–62.

Diggle, P. 1990. *Time Series: A Biostatistical Introduction*. Clarendon Press.

Diggle, P. J., Heagerty, P. J., Liang, K.-Y. and Zeger, S. L. 2002. *Analysis of Longitudinal Data*, 2nd edn. Clarendon Press.

Dobson, A. J. 2001. *An Introduction to Generalized Linear Models*, 2nd edn. Chapman and Hall.

Donner, A. and Klar, N. 2000. *Design and Analysis of Cluster Randomization Trials in Health Research.* Edward Arnold.

Durbin, R. S., Eddy, A., Krogh, A. and Mitchison, G. 1998. *Biological Sequence Analysis*. Cambridge University Press.

Edwards, D. 2000. *Introduction to Graphical Modelling*, 2nd edn. Springer.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. 2003. Least angle regression. `http://www-stat.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf`

Efron, B. and Tibshirani, R. 1993. *An Introduction to the Bootstrap*. Chapman and Hall.

Eubank, R. L. 1999. *Nonparametric Regression and Spline Smoothing*, 2nd edn. Marcel Dekker.

Ewens, W. J. and Grant, G. R. 2005. *Statistical Methods in Bioinformatics: an Introduction*, 2nd edn. Springer.

Fan, J. and Gijbels, I. 1996. *Local Polynomial Modelling and Its Applications*. Chapman and Hall.

Faraway, J. J. 2004. *Linear Models with R.* Chapman and Hall/CRC.

Faraway, J. J. 2006. *Extending the Linear Model with R. Generalized Linear, Mixed Effects and Nonparametric Regression Models.* Chapman and Hall/CRC.

Finney, D. J. 1978. *Statistical Methods in Bioassay*, 3rd edn. Macmillan.

Fisher, R. A. 1935. *The Design of Experiments*. Oliver and Boyd (7th edn, 1960).

Fox, J. 2002. *An* R *and* S-PLUS *Companion to Applied Regression*. Sage Books.

Gardner, M. J., Altman, D. G., Jones, D. R. and Machin, D. 1983. Is the statistical assessment of papers submitted to the *British Medical Journal* effective? *British Medical Journal* 286: 1485–8.

Gaver, D. P., Draper, D. P., Goel, K. P., Greenhouse, J. B., Hedges, L. V., Morris, C. N. and Waternaux, C. 1992. *Combining Information: Statistical Issues and Opportunities for Research*. National Research Council, National Academy Press.

Gelman, A. and Hill, J. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press.

Gelman, A. B., Carlin, J. S., Stern, H. S. and Rubin, D. B. 2003. *Bayesian Data Analysis*, 2nd edn. Chapman and Hall/CRC.

Gentleman, R. 2008. *R Programming for Bioinformatics*. Chapman and Hall/CRC.

Gentleman, R. and Lang, D. 2004. Statistical analyses and reproducible research. Bioconductor Project Working Papers. Working Paper 2. http://www.bepress. com/bioconductor/paper2

Gentleman, R., Carey, V., Huber, W., Irizarry, R. and Dudoit, S. 2005. *Bioinformatics and Computational Biology Solutions using R and Bioconductor*. Springer.

Gigerenzer, G. 1998. We need statistical thinking, not statistical rituals. *Behavioural and Brain Sciences* 21: 199–200.

Gigerenzer, G. 2002. *Reckoning with Risk: Learning to Live with Uncertainty*. Penguin Books.

Gigerenzer, G., Swijtink, Z., Porter, T., Daston, L., Beatty, J. and Krüger, L. 1989. *The Empire of Chance*. Cambridge University Press.

Gill, J. 2008. *Bayesian Methods: A Social and Behavioral Sciences Approach*, 2nd edn. Chapman and Hall/CRC.

Goldstein, H. 1995. *Multilevel Statistical Models*. Arnold. http://www.arnoldpublishers.com/support/goldstein.htm

Gordon, A. D. 1999. *Classification*, 2nd edn. Chapman and Hall/CRC.

Gourieroux, C. 1997. *ARCH Models and Financial Applications*. Springer.

Hall, P. 2001. Biometrika centenary: nonparametrics. *Biometrika* 88: 143–65.

Harlow, L. L., Mulaik, S. A. and Steiger, J. H. (eds) 1997. *What If There Were No Significance Tests?* Lawrence Erlbaum Associates.

Harrell, F. E. 2001. *Regression Modelling Strategies, with Applications to Linear Models, Logistic Regression and Survival Analysis*. Springer.

Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd edn. Springer.

Hauck, W. W. J. and Donner, A. 1977. Wald's test as applied to hypotheses in logit analysis. *Journal of the American Statistical Association* 72: 851–3.

Hoaglin, D. C. 2003. John W. Tukey and data analysis. *Statistical Science* 18: 311–18.

Hyndman, R. J. and Khandakar, Y. 2008. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* 27(3): 1–22. http://www.jstatsoft.org/v27/i03

Hyndman, R. J., Koehler, A. B., Snyder, R. D. and Grose, S. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* 18: 439–54.

Hyndman, R. J., Koehler, A. B., Ord, J. K. and Snyder, R. D. 2008. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer.

Ihaka, R. and Gentleman, R. 1996. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299–314.

Izenman, A. J. 2008. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer.

Jackson, R., Broad, J., Connor, J. and Wells, S. 2005. Alcohol and ischaemic heart disease: probably no free lunch. *The Lancet* 366: 1911–12.

Johnson, D. H. 1995. Statistical sirens: the allure of nonparametrics. *Ecology* 76: 1998–2000.

Krantz, D. H. 1999. The null hypothesis testing controversy in psychology. *Journal of the American Statistical Association* 44: 1372–81.

Krzanowski, W. J. 2000. *Principles of Multivariate Analysis. A User's Perspective*, revised edn. Clarendon Press.

Leavitt, S. D. and Dubner, S. J. 2005. *Freakonomics. A Rogue Economist Explores the Hidden Side of Everything*. William Morrow.

Leek, J. T. and Storey, J. D. 2007. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet* 3(9): e161. doi: 10.1371/journal.pgen.0030161.

Leisch, F. 2002. *Sweave User Manual*. http://www.ci.tuwien.ac.at/~leisch/Sweave

Liaw, A. and Wiener, M. 2002. Classification and regression by randomforest. *R News* 2(3): 18–22.

Lim, T.-S. and Loh, W.-Y. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning* 40: 203–28.

Lumley, T. 2004a. Programmers' niche: a simple class, in S3 and S4. *R News* 4(1): 33–6.

Lumley, T. 2004b. The survival package. *R News* 4(1): 26–8.

Maindonald, J. H. 1984. *Statistical Computation*. John Wiley.

Maindonald, J. H. 1992. Statistical design, analysis and presentation issues. *New Zealand Journal of Agricultural Research* 35: 121–41.

Maindonald, J. H. 2003. The role of models in predictive validation. Invited Paper.

Maindonald, J. 2006. Data mining methodological weaknesses and suggested fixes. In P. Christen, P. J. Kennedy, L. Jiuyong, S. J. Simoff and G. J. Williams (eds), *Fifth Australasian Data Mining Conference (AusDM2006)*, volume 61 of *CRPIT*, pp. 9–16. ACS, Sydney, Australia. http://crpit.com/abstracts/CRPITV61Maindonald.html

Maindonald, J. H. 2008. Using R for data analysis and graphics. http://wwwmaths.anu.edu.au/~johnm/r/usingR.pdf

Maindonald, J. H. and Burden, C. J. 2005. Selection bias in plots of microarray or other data that have been sampled from a high-dimensional space. In *Proceedings of 12th Computational Techniques and Applications Conference CTAC-2004*, volume 46, pp. C59–C74. http://www.maths.anu.edu.au/~johnm/dm/ausdm06/ausdm06-jm.pdf

Maindonald, J. H. and Cox, N. R. 1984. Use of statistical evidence in some recent issues of DSIR agricultural journals. *New Zealand Journal of Agricultural Research* 27: 597– 610.

Maindonald, J. H., Waddell, B. C. and Petry, R. J. 2001. Apple cultivar effects on codling moth (Lepidoptera: Tortricidae) egg mortality following fumigation with methyl bromide. *Postharvest Biology and Technology* 22: 99–110.

Manly, B. F. J. 2005. *Multivariate Statistical Methods. A Primer*, 3rd edn. Chapman & Hall/CRC.

McCullagh, P. and Nelder, J. A. 1989. *Generalized Linear Models*, 2nd edn. Chapman and Hall.

Meyer, D. 2001. Support vector machines. *R News* 1(3): 23–6.

Miller, R. G. 1986. *Beyond ANOVA, Basics of Applied Statistics*. John Wiley.

Muenchen, R. A. 2008. *R for SAS and SPSS Users*. Springer.

Murrell, P. 2005. *R Graphics*. Chapman and Hall/CRC. http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html

Myers, R. H. 1990. *Classical and Modern Regression with Applications*, 2nd edn. Brooks Cole.

Nelder, J. A. 1999. From statistics to statistical science. *Journal of the Royal Statistical Society, Series D* 48: 257–67.

Nicholls, N. 2000. The insignificance of significance testing. *Bulletin of the American Meteorological Society* 81: 981–6.

Ord, J. K., Koehler, A. B. and Snyder, R. D. 1997. Estimation and prediction for a class of dynamic nonlinear statistical models. *Journal of the American Statistical Association* 92: 1621–9.

Paradis, E. 2006. *Analysis of Phylogenetics and Evolution with R*. Springer.

Payne, R. W., Lane, P. W., Digby, P. G. N., Harding, S. A., Leech, P. K., Morgan, G. W., Todd, A. D., Thompson, R., Tunnicliffe Wilson, G., Welham, S. J. and White, R. P. 1997. *Genstat 5 Release 3 Reference Manual*. Oxford University Press.

Pinheiro, J. C. and Bates, D. M. 2000. *Mixed Effects Models in* S *and* S-PLUS. Springer.

R Development Core Team. 2009a. An introduction to R. The most recent version is available from CRAN sites. http://cran.r-project.org

R Development Core Team. 2009b. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org

R Development Core Team. 2009c. R *Language Definition*. Available from CRAN sites.

Ramsay, J. and Silverman, B. 2002. *Applied Functional Data Analysis*. Springer.

Rao, C. and Wu, Y. 2001. On model selection (with discussion). In P. Lahiri (ed.), *Model Selection* volume 38 of *IMS Lecture Notes – Monograph Series*, pp. 1–64. Institute of Mathematical Statistics.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rosenbaum, P. R. 1999. Choice as an alternative to control in observational studies. *Statistical Science* 14: 259–78. With following discussion, pp. 279–304.

Rosenbaum, P. R. 2002. *Observational Studies*, 2nd edn. Springer.

Rosenbaum, P. R. 2005. Reasons for effects. *Chance* 18: 5–10.

Rosenbaum, P. and Rubin, D. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55.

Sammon, J. W. 1969. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers* C-18: 401–9.

Sarkar, D. 2002. Lattice. *R News* 2(2): 19–23.

Sarkar, D. 2007. *Lattice: Multivariate Data Visualization with R*. Springer. http://lmdvr.r-forge.r-project.org/figures/figures.html

Schatzkin, A., Kipnis, V., Carroll, R., Midthune, D., Subar, A., Bingham, S., Schoeller, D., Troiano, R. and Freedman, L. 2003. A comparison of a food frequency questionnaire with a 24-hour recall for use in an epidemiological cohort study: results from the biomarker-based observing protein and energy nutrition (open) study. *International Journal of Epidemiology* 32: 1054–62.

Schmidt-Nielsen, K. 1984. *Scaling. Why Is Animal Size So Important?* Cambridge University Press.

Senn, S. 2003. *Dicing with Death: Chance, Risk and Health*. Cambridge University Press.

Sharp, S. J., Thompson, S. G. and Altman, D. G. 1996. The relation between treatment benefit and underlying risk in meta-analysis. *British Medical Journal* 313: 735–8.

Shumway, R. and Stoffer, D. 2006. *Time Series Analysis and Its Applications: With R Examples.* Springer.

Simpson, E. H. 1951. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B* 13: 238–41.

Smyth, G. K. 2004. Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology* 3 (1), Article 3.

Snijders, T. A. B. and Bosker, R. J. 1999. *Multilevel Analysis. An Introduction to Basic and Advanced Multilevel Modelling*. Sage Books.

Spector, P. 2008. *Data Manipulation with R*. Springer.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P. and van der Linde, A. 2002. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B* 64: 583–616. With following discussion, pp. 616–39.

Sprent, P. 1966. A generalized least squares approach to linear functional relationships. *Journal of the Royal Statistical Society, Series B* 28: 278–88. With following discussion, pp. 288–97.

Steel, R. G. D., Torrie, J. H. and Dickie, D. A. 1993. *Principles and Procedures of Statistics, A Biometrical Approach*, 3rd edn. McGraw-Hill.

Stidd, C. K. 1953. Cube-root-normal precipitation distributions. *Transactions of the American Geophysical Union* 34: 31–5.

Streiner, D. L. and Norman, G. R. 2003. *Health Measurement Scales. A Practical Guide to their Development and Use*, 3rd edn. Oxford University Press.

Talbot, M. 1984. Yield variability of crop varieties in the U.K. *Journal of the Agricultural Society of Cambridge* 102: 315–21.

Therneau, T. M. and Atkinson, E. J. 1997. An introduction to recursive partitioning using the rpart routines. Technical Report 61, Department of Health Science Research, Mayo Clinic, Rochester, MN. http://mayoresearch.mayo.edu/mayo/research/biostat/techreports.cfm

Therneau, T. M. and Grambsch, P. M. 2001. *Modeling Survival Data: Extending the Cox Model*. Springer.

Tufte, E. R. 1997. *Visual Explanations*. Graphics Press.

Tukey, J. W. 1991. The philosophy of multiple comparisons. *Statistical Science* 6: 100–16.

Turner, R. M., Spiegelhalter, D. J., Smith, G. C. S. and Thompson, S. G. 2009. Bias modelling in evidence synthesis. *Journal of the Royal Statistical Society, Series A* 172(1): 21–47. http://ideas.repec.org/a/bla/jorssa/v172y2009i1p21-47.html

Vaida, F. and Blanchard, S. 2005. Conditional Akaike information for mixed-effects models. *Biometrika* 92: 351–70.

Venables, W. N. 1998. Exegeses on linear models. In *Proceedings of the 1998 International S-PLUS User Conference*. http://www.stats.ox.ac.uk/pub/MASS3/Compl.html

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with* S, 4th edn. Springer. See also R Complements to Modern Applied Statistics with S. http://www.stats.ox.ac.uk/pub/MASS4/

Wainer, H. 1997. *Visual Revelations*. Springer.

Weisberg, S. 1985. *Applied Linear Regression*, 2nd edn. John Wiley.

Welch, B. L. 1949. Further note on Mrs. Aspin's tables and on certain approximations to the tabled function. *Biometrika* 36: 293–6.

Wickham, H. 2009. *ggplot2:Elegant Graphics for Data Analysis*. Springer. http://had.co.nz/ggplot2/

Wilkinson, L. 2005. *The Grammar of Graphics*. Springer.

Wilkinson, L. and Task Force on Statistical Inference. 1999. Statistical methods in psychology journals: guidelines and explanation. *American Psychologist* 54: 594–604.

Williams, E. R., Matheson, A. C. and Harwood, C. E. 2002. *Experimental Design and Analysis for Use in Tree Improvement*. CSIRO Information Services. Revised.

Williams, G. P. 1983. Improper use of regression equations in the earth sciences. *Geology* 11: 195–7.

Wonnacott, T. H. and Wonnacott, R. 1990. *Introductory Statistics*, 5th edn. John Wiley.

Wood, S. N. 2001. mgcv: GAMs and generalized ridge regression for R. *R News* 1(2): 20–25.

Wood, S. N. 2006. *Generalized Additive Models. An Introduction with R*. Chapman and Hall/CRC.

Würtz, D. 2004. *Rmetrics: An Environment for Teaching Financial Engineering and Computational Finance with R*. Rmetrics, ITP, ETH Zürich, Switzerland. http://www.rmetrics.org

Xie, Y. and Cheng, X. 2008. animation: A package for statistical animations. *R News* 8(2): 23–7.

Young, G. and Smith, R. L. 2005. *Essentials of Statistical Inference*. Cambridge University Press.

Zeger, S. L., Thomas, D., Dominici, F., Samet, J. M., Schwartz, J., Dockery, D. and Cohen, A. 2000. Exposure measurement error in time-series studies of air pollution: concepts and consequences. *Environmental Health Perspectives* 108: 419–26. See also vol. 109, p. A517.

Zhang, H. and Singer, B. 1999. *Recursive Partitioning in the Health Sciences*. Springer.

Zhu, X., Ambroise, C. and McLachlan, G. J. 2006. Selection bias in working with the top genes in supervised classification of tissue samples. *Statistical Methodology* 3: 29–41.

References for Data Sets

Andrews, D. F. and Herzberg, A. M. 1985. *Data. A Collection of Problems from Many Fields for the Student and Research Worker*. Springer.

Ash, J. and Helman, C. 1990. Floristics and vegetation biomass of a forest catchment, Kioloa, south coastal N.S.W. *Cunninghamia* 2: 167–82.

Blake, C. and Merz, C. 1998. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Boot, H. M. and Maindonald, J. 2008. New estimates of age- and sex-specific earnings and the male–female earnings gap in the British cotton industry, 1833–1906. *Economic History Review* 61: 380–408.

Burns, N. R., Nettlebeck, T., White, M. and Willson, J. 1999. Effects of car window tinting on visual performance: a comparison of elderly and young drivers. *Ergonomics* 42: 428–43.

Bussolari, S. 1987. Human factors of long-distance human-powered aircraft flights. *Human Power* 5: 8–12.

Charig, C. R. 1986. Comparison of treatment of renal calculi by operative surgery, percutaneous nephrolithotomy, and extracorporeal shock wave lithotripsy. *British Medical Journal* 292: 879–82.

Christie, M. 2000. *The Ozone Layer: A Philosophy of Science Perspective*. Cambridge University Press.

Chu, I., Secours, V., Villeneuve, D. C., Valli, V. E., Nakamura, A., Colin, D., Clegg, D. J. and Arnold, E. P. 1988. Reproduction study of toxaphene in the rat. *Journal of Environmental Science and Health Part B. Pesticides and Food Contamination* 23: 101–26.

Clutton-Brock, T. H., O'Riain, M. J., Brotherton, P. N. M., Gaynor, D., Kansky, R., Griffin, A. S. and Manser, M. 1999. Selfish sentinels in cooperative mammals. *Science*, pp. 1640–44.

Cohen, P. 1996. Pain discriminates between the sexes. *New Scientist* 2054: 16.

Daniels, M., Devlin, B. and Roeder, K. 1997. Of genes and IQ. In B. Devlin, S. Fienberg and K. Roeder (eds), *Intelligence, Genes and Success*, Chapter 3. Springer.

Darwin, C. 1877. *The Effects of Cross and Self Fertilisation in the Vegetable Kingdom*. Appleton and Company.

Dehejia, R. H. and Wahba, S. 1999. Causal effects in non-experimental studies: re-evaluating the evaluation of training programs. *Journal of the American Statistical Association* 94: 1053–62.

Ezzet, F. and Whitehead, J. 1991. A random effects model for ordinal responses from a crossover trial. *Statistics in Medicine* 10: 901–7.

Farmer, C. 2005. Another look at Meyer and Finney's 'who wants airbags?' *Chance* 19: 15–22.

Gihr, M. and Pilleri, G. 1969. Anatomy and biometry of Stenella and Delphinus. In G. Pilleri (ed.), *Investigations on Cetacea*. Hirnanatomisches Institute der Universität Bern.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S. 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286: 531–7.

Gordon, N. C., Gear, R. W., Heller, P. H. and Levine, J. D. 1995. Enhancement of morphine analgesia by the GABA[subscript B] agonist baclofen. *Neuroscience* 69: 345–9.

Gordon, W. 1894. *Our Country's Birds and How to Know Them*. Day and Son.

Grasso, L. C., Maindonald, J., Rudd, S., Hayward, D. C., Saint, R., Miller, D. J. and Ball, E. E. 2008. Microarray analysis identifies candidate genes for key roles in coral development. *BMC Genomics* 9: 540.

Guy, W. A. 1882. Two hundred and fifty years of small pox in London, together with a supplement relating to England and Wales. *Journal of the Royal Statistical Society* 45: 399–443.

Hales, S., de Wet, N., Maindonald, J. and Woodward, A. 2002. Potential effect of population and climate change global, distribution of dengue fever: an empirical model. *The Lancet* 360: 830–34.

Hall, P. 2003. A possum's tale – how statistics revealed a new mammal species. *Chance* 16: 8–13.

Harker, F. R. and Maindonald, J. H. 1994. Ripening of nectarine fruit. *Plant Physiology* 106: 165–71.

Hobson, J. A. 1988. *The Dreaming Brain*. Basic Books.

Hunter, D. 2000. *The conservation and demography of the southern corroboree frog (Pseudophryne corroboree)*. MSc, University of Canberra.

Jouzel, J. *et al*. 2007. EPICA Dome C ice core 800kyr deuterium data and temperature estimates. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series # 2007-091. NOAA/NCDC Paleoclimatology Program, Boulder, CO, USA.

King, D. A. 1998. Relationship between crown architecture and branch orientation in rain forest trees. *Annals of Botany* 82: 1–7.

King, D. A. and Maindonald, J. H. 1999. Tree architecture in relation to leaf dimensions and tree stature in temperate and tropical rain forests. *Journal of Ecology* 87: 1012–24.

Lalonde, R. 1986. Evaluating the economic evaluations of training programs. *American Economic Review* 76: 604–20.

Latter, O. H. 1902. The egg of *cuculus canorus*. An inquiry into the dimensions of the cuckoo's egg and the relation of the variations to the size of the eggs of the foster-parent, with notes on coloration, &c. *Biometrika* 1: 164–76.

Linacre, E. 1992. *Climate Data and Resources. A Reference and Guide*. Routledge.

Linacre, E. and Geerts, B. 1997. *Climates and Weather Explained*. Routledge.

Linde, K., Streng, A., Jurgens, S., Hoppe, A., Brinkhaus, B., Witt, C., Wagenpfeil, S., Pfaffenrath, V., Hammes, M., Weidenhammer, W., Willich, S. and Melchart, D. 2005. Acupuncture for patients with migraine. A randomized controlled trial. *Journal of the American Medical Association* 293: 2118–25.

Lindenmayer, D. B., Viggers, K. L., Cunningham, R. B. and Donnelly, C. F. 1995. Morphological variation among columns of the mountain brushtail possum, *Trichosurus caninus* Ogilby (Phalangeridae: Marsupiala). *Australian Journal of Zoology* 43: 449–58.

Lüthi, D., Floch, M. L., Bereiter, B., Blunier, T., Barnola, J.-M., Siegenthaler, U., Raynaud, D., Jouzel, J., Fischer, H., Kawamura, K. and Stocker, T. 2008. High-resolution carbon dioxide concentration record 650,000–800,000 years before present. *Nature* 453: 379–82.

Marland, G., Boden, T. A. and Andres, R. J. 2003. Global, regional, and national CO2 emissions. In *Trends: A Compendium of Data on Global Change*. Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, U.S. Department of Energy, Oak Ridge, TN, USA. http://cdiac.esd.ornl.gov/

Matthews, D. E. and Farewell, V. T. 1996. *Using and Understanding Medical Statistics*. Karger.

McLellan, E. A., Medline, A. and Bird, R. P. 1991. Dose response and proliferative characteristics of aberrant crypt foci: putative preneoplastic lesions in rat colon. *Carcinogenesis* 12: 2093–8.

McLeod, C. C. 1982. Effect of rates of seeding on barley grown for grain. *New Zealand Journal of Agriculture* 10: 133–6.

Meyer, M. 2006. Commentary on "another look at Meyer and Finney's 'who wants airbags?"' *Chance* 19: 23–2.

Meyer, M. and Finney, T. 2005. Who wants airbags? *Chance* 18: 3–16.

Mitchell, B. R. 1988. *British Historical Statistics*. Cambridge University Press.

Nadel, E. and Bussolari, S. 1988. The Daedalus project: physiological problems and solutions. *American Scientist* 76: 351–60.

Newton, A. 1893–1896. Cuckoos. In A. Newton and H. Gadow (eds), *Dictionary of Birds*. A.& C. Black.

Nicholls, N., Lavery, B., Frederiksen, C. and Drosdowsky, W. 1996. Recent apparent changes in relationships between the El Niño, southern oscillation and Australian rainfall and temperature. *Geophysical Research Letters* 23: 3357–60.

Nightingale, F. 1871. *Notes on Lying-in Institutions*. Longmans, Green and Co.

Perrine, F. M., Prayitno, J., Weinman, J. J., Dazzo, F. B. and Rolfe, B. 2001. Rhizobium plasmids are involved in the inhibition or stimulation of rice growth and development. *Australian Journal of Plant Physiology* 28: 923–7.

Roberts, H. V. 1974. *Conversational Statistics*. Hewlett-Packard University Business Series. The Scientific Press.

Shanklin, J. 2001. Ozone at Halley, Rothera and Vernadsky/Faraday. www.antarctica.ac.uk/met/jds/ozone/

Sharp, W. D. and Clague, D. A. 2006. 50-ma initiation of Hawaiian–Emperor bend records major change in Pacific Plate motion. *Science* 313: 1281–4.

Snelgar, W. P., Manson, P. J. and Martin, P. J. 1992. Influence of time of shading on flowering and yield of kiwifruit vines. *Journal of Horticultural Science* 67: 481–7.

Stewardson, C. L., Hemsley, S., Meyer, M. A., Canfield, P. J. and Maindonald, J. H. 1999. Gross and microscopic visceral anatomy of the male Cape fur seal, *Arctocephalus pusillus pusillus* (Pinnipedia: Otariidae), with reference to organ size and growth. *Journal of Anatomy (Cambridge)* 195: 235–55. (WWF project ZA-348.)

Stewart, K. M., Van Toor, R. F. and Crosbie, S. F. 1988. Control of grass grub (Coleoptera: Scarabaeidae) with rollers of different design. *New Zealand Journal of Experimental Agriculture* 16: 141–50.

Stiell, I. G., Wells, G. A., Vandemheen, K., Clement, C., Lesiuk, H., Laupacis, A., McKnight, R. D., Verbeek, R., Brison, R., Cass, D., Eisenhauer, M. A., Greenberg, G. H. and Worthington, J. F. 2001. The Canadian CT head rule for patients with minor head injury. *The Lancet* 357: 1391–6.

Stocks, P. 1942. Measles and whooping cough during the dispersal of 1939–1940. *Journal of the Royal Statistical Society* 105: 259–91.

Telford, R. D. and Cunningham, R. B. 1991. Sex, sport and body-size dependency of hematology in highly trained athletes. *Medicine and Science in Sports and Exercise* 23: 788–794.

Thall, P. F. and Vail, S. C. 1990. Some covariance models for longitudinal count data. *Biometrics* 46: 657–71.

Tippett, L. H. C. 1931. *The Methods of Statistics*. Williams and Norgate.

Wainright, P., Pelkman, C. and Wahlsten, D. 1989. The quantitative relationship between nutritional effects on preweaning growth and behavioral development in mice. *Developmental Psychobiology* 22: 183–93.

References for Packages

base: base, datasets, grDevices, graphics, grid, methods, splines, stats, stats4, tcltk, tools, utils, R Development Core Team, 2008. R Foundation for Statistical Computing. http://www.r-project.org

boot: Bootstrap R (S-Plus) Functions (Canty), Canty, A. and Ripley, B. 2008. (Version 1.2-34.) See further Canty (2002).

car: Companion to Applied Regression, Fox, J. 2008. (V. 1.2-8.) http://socserv.socsci.mcmaster.ca/jfox/

cluster: Cluster Analysis Basics and Extensions, Maechler, M., Rousseeuw, P., Struyf, A. and Hubert, M. 2008. (V. 1.11.11.) Based on S original by P. Rousseeuw, A. Struyf and M. Hubert; initial R port by K. Hornik.

colorRamps: Builds color tables, Keitt, T. 2007. (V. 2.2.)

compositions: Compositional Data Analysis, van den Boogaart, K. G., 2009. (V. 0.91-6.)

DAAG: Data Analysis And Graphics, Maindonald, J. and Braun, W. J. 2009. (V. 0.98.) http://www.stats.uwo.ca/DAAG

DAAGxtras: Data sets and Functions, supplementary to DAAG, Maindonald, J. 2009. (R package version 0.7-7.) http://www.maths.anu.edu.au/~johnm

dichromat: Color schemes for dichromats, Lumley, T. 2007. (V. 1.2-3.) *fgui: Automatic creation of Graphical User Interfaces for command-line R packages*, Hoffmann, T. J. 2009.

Forecasting (functions and datasets for forecasting): bundle of forecast, fma, Mcomp, expsmooth, Hyndman, R. J. 2009. (R package version 1.24.) http://www.robhyndman. info/Rlibrary/forecast/

foreign: Read Data from Minitab, S, SAS, SPSS, Stata, Systat, dBase, . . . , R-core, DebRoy, S., Bivand, R. and others. 2008. (V. 0.8-33.) See COPYRIGHTS file in the sources for acknowledgments.

ggplot2: An implementation of the Grammar of Graphics, Wickham, H. 2009. (V. 0.8.3.) http://had.co.nz/ggplot2/

golubEsets: exprSets for golub leukemia data, Golub, T. 2002. (V. 1.0.1.) Maintained by V. Carey.

hddplot: Use known groups in high-dimensional data to derive scores for plots, Maindonald, J. H. 2006. (V. 0.5-0.) See further Maindonald and Burden (2005).

lattice: Lattice Graphics, Sarkar, D. 2009. (V. 0.17-20.) See further Sarkar (2002).

latticeExtra: Extra Graphical Utilities Based on Lattice, Sarkar, D. and Andrews, F. 2008. (V. 0.5-4.) See further Sarkar (2002).

latticist: A Lattice-based tool for exploratory visualisation, Andrews, F. 2008. (R package version 0.9-40.) http://latticist.googlecode.com/

leaps: regression subset selection, Lumley, T. and Miller, A. 2009. (V. 2.8.) Compiled by T. Lumley; Fortran code by A. Miller. See further Miller (1986).

lme4: Linear mixed-effects models using S4 classes, Bates, D., Maechler, M. and Dai, B. 2008. (V. 0.999375-28.) See further Bates (2005).

MCMCpack: Markov chain Monte Carlo (MCMC) Package, Martin, A. D., Quinn, K. M. and Park, J. H. 2009. (V. 0.9-6.)

MEMSS: Data sets from Mixed-effects Models in S, Bates, D. 2008. (V. 0.3-4.)

mgcv: GAMs and Generalized Ridge Regression for R, Wood, S. N. 2008. (V. 1.4-1.) See further Wood (2001).

monoProc: strictly monotone smoothing procedure, Scheder, R. 2005. (V. 1.0-4.)

multtest: Resampling-based multiple hypothesis testing, Pollard, K. S., Ge, Y. and Dudoit, S. 2005. (V. 1.8.0.)

nlme: Linear and nonlinear mixed effects models, Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D. and R core team. 2008. (V. 3.1-89.) See further Pinheiro and Bates (2000).

oz: Plot the Australian coastline and states, Venables, W. N. and Hornik, K. 2007. (V. 1.0-16.) S original by W. Venables; R port by K. Hornik.

playwith: A GUI for interactive plots using GTK+, Andrews, F. 2008. (R package version 0.9-43; requires Gtk+.) http://playwith.googlecode.com/

randomForest: Classification and Regression, Liaw, A. and Wiener, M. 2008. (V. 4.5-30.) See further Liaw and Wiener (2002).

rattle: A graphical user interface for data mining in R using GTK, Williams, G. 2009. (R package version 2.4.54.) http://rattle.togaware.com/

RColorBrewer: ColorBrewer palettes, Neuwirth, E. 2007. (V. 1.0-2.)

reshape: Flexibly reshape data, Wickham, H. 2008. (V. 0.8.2.) http://had.co.nz/reshape

rpart: Recursive Partitioning, Therneau, T. M. and Atkinson, B. 2008. (V. 3.1-42.) R port by B. Ripley.

survival: Survival analysis, including penalised likelihood., Therneau, T. and Lumley, T. 2008. (V. 2.34-1.) S original by T. Therneau; R port by T. Lumley. See further Therneau and Grambsch (2001) and Lumley (2004b).

tseries: Time series analysis and computational finance, Trapletti, A. and Hornik, K. 2008. (V. 0.10-16.) Compiled by A. Trapletti.

VR packages: bundle of MASS, class, nnet, spatial, Venables, W. N. and Ripley, B. D. 2008. (V. 7.2-44.) R port by B. Ripley, following earlier work by K. Hornik and A. Gebhardt. See further Venables and Ripley (2002). http://www.stats.ox.ac.uk/pub/MASS4/

xtable: Export tables to LaTeX or HTML, Dahl, D. and others. 2008. (V. 1.5-4.)

Other packages and package bundles mentioned in the text are: *ape* (Paradis, E., Claude, J. and Strimmer, K.), *Deducer* (GUI; Fellows, I.), *Devore6* (Devore, J. L. and Bates, D.), *dr* (Weisberg, S.), *dse* (package bundle; Gilbert, P.), *fseries* (Wuertz, D.), *gam* (Hastie, T.), *JGR* (Helbig, H. and Urbanek, S.), *lars* (Hastie, T. and Efron, B.), *KernSmooth* (Wand, M. and Ripley, B.), *locfit* (Loader, C.), *MPV* (Braun, W. J.; data sets from Montgomery, Peck and Vining), *muhaz* (Hess, K. and Gentleman, R.), *plyr* (Wickham, H.), *pmg* (Verzani, J.), *RODBC* (Lapsley, M. and Ripley, B. D.) and *strucchange* (Zeileis *et al.*). For more complete acknowledgments, see the help pages for the individual packages. See also the information given by calling `citation()` with the package name as character string argument.

References for Web Pages

Comprehensive R Archive Network.
http://cran.r-project.org/

CRAN Task Views.
http://cran.r-project.org/web/views/

ESS – Emacs Speaks Statistics.
http://ess.r-project.org/

f*gui: Rapidly create a gui interface*, Hoffmann, T. J. and Laird, N. M. 2009 (V.1.0-0)
http://www.people.fas.harvard.edu/~tjhoffm/fgui.html

Gtk+. The development version is required for playwith; suggested for rattle.
http://downloads.sourceforge.net/gladewin32/

Machine Learning Repository.
http://www.ics.uci.edu/~mlearn/MLRepository.html

R News.
http://cran.r-project.org/doc/Rnews/

S+, Tibco Spotfire S+ (formerly S-PLUS).
http://spotfire.tibco.com/

Tinn-R.
http://sourceforge.net/projects/tinn-r

Acknowledgments for use of data

We thank the following for permission to reproduce graphs or tables that have appeared in published material: *Journal of Ecology*, for permission to reproduce Figure 12.5B, which is a redrawn version of the fourth panel in Figure 3 in King and Maindonald (1999); *Plant Physiology*, in relation to the right panel of Figure 2.6, which is a redrawn version of Figure 3 in Harker and Maindonald (1994), copyrighted by the American Society of Plant Biologists; SIR Publishing (The Royal Society of New Zealand), for permission to reproduce data in Subsection 3.1.1 from Stewart *et al.* (1988), Table 10.2 from Maindonald (1992), and Figure 10.4 which is similar to Figure 1 in Maindonald (1992); CSIRO Publishing, for permission to reproduce (in Section 4.4) data that appear in Table 4 in Perrine *et al.* (2001); *Australian Journal of Zoology*, for permission to reproduce a graph that is similar to a part of Figure 2 in Lindenmayer *et al.* (1995); *American Medical Association* (©2005, All rights reserved), for permission to use the data in Exercise 4.11 in Section 9, from Linde *et al.* (2005).

We acknowledge the help of the following individuals and organizations in making data available. Darren Kriticos (CSIRO Division of Entomology): data frame `houseprices`. J. D. Shanklin (British Antarctic Survey): `ozone` (Shanklin, 2001). W. S. Snelgar (HortResearch NZ): `kiwishade` (Snelgar *et al.*, 1992). Francine Adams and supervisors Rosemary Martin and Murali Nayadu (all from ANU): `science`. Jasmyn Lynch (ANU), for the rare and endangered plant species data in Subsection 4.3.1. D. B. Lindenmayer and coworkers (ANU): `possum` (Lindenmayer *et al.*, 1995). D. A. King (formerly ANU): `leafshape` and `leafshape17` (King and Maindonald, 1999). R. F. Harker (HortResearch NZ): `fruitohms` (Harker and Maindonald, 1994). E. Linacre (ANU): `dewpoint` (Linacre, 1992, Linacre and Geerts, 1997). N. R. Burns (University of Adelaide): `tinting` (Burns *et al.*, 1999). M. Boot (ANU): `wages1833` (Boot and Maindonald, 2008). J. Erickson (University of Chicago) and A. H. Welsh (University of Southampton): `anesthetic`. D. Hunter (University of Canberra): `frogs`. Sharyn Wragg (formerly ANU): `moths`. Claudia Haarman (ANU), for the flow inhibition data of Exercise 1 in Section 8.9. Melissa Manning (ANU): `socsupport`. J. Ash (ANU): `rainforest` (Ash and Helman, 1990). Katharina Siebke and Susan von Cammerer (ANU): `leaftemp`. Ranjana Bird (University of Manitoba), for the aberrant crypt foci data in Section 3.7. Australian Bureau of Meteorology: `bomregions`. Note the abbreviation ANU for Australian National University.

Acknowledgments for data from web pages

(Help pages are for the *DAAG* package.)

`bomregions`: See `help(bomregions)` for details.

`edcT` and `edcCO2`: See `help(edcT)` and `help(edcCO2)` for details. See Jouzel *et al.* (2007), Lüthi *et al.* (2008). The help pages for `edcCO2` and `edcT` give additional references and links.

`hills2000`: `http://www.hillrunning.co.uk`

`hotspots`, `hotspots2006`: See `help(hotspots)`; Sharp and Clague (2006).

`monica`: `http://www.ktl.fi/monica`

`nihills`: `http://www.nimra.org.uk/calendar.asp`

`ozone`: See `help(ozone)`; Shanklin (2001).

`nswdemo`, `cps1`, `psid1`, `nswpsid1`: See `help(nswdemo)`; Dehejia and Wahba (1999), Lalonde (1986).

Index of R symbols and functions

R symbols

!, 447
+ − * /, 2, 444
. . ., 439–40
< <= == != > >= >= 11, 14
<−, 2
[, 11, 16, 453, 460
[[, 453, 460
#, 2, 435
$, 4, 460
%%, 153
%in%, 23
&, &&, 443
|, ||, 443
^, 2

NA, 12–13, 24, 446–7
NaN, 13
Inf, 13
NULL, 13, 19, 439, 446, 460
TRUE, FALSE, 8, 10, 11
break, 24
for, 24–5
if, 23
in, 25
repeat, 24
while, 24

Functions

Functions that are not otherwise identified are defined in the text.

.First, 433
.Last, 433
.M1, 373
.libPaths, 433

aaply (*plyr*)
adply (*plyr*)
abbreviate (*base*), 40, 41, 75
abline (*graphics*), 50, 78, 80, 110, 129, 145, 150, 157, 247, 248, 337, 370
abs (*base*), 129, 145, 168, 192, 194
accTrainTest (*hddplot*), 399
acf (*stats*), 138, 284, 289, 291, 293, 344
add1 (*stats*), 194, 195
additions, 465
addmargins (*stats*), 61–3
aes (*ggplot2*), 488, 490, 491
aggregate (*stats*), 17, 18, 63, 64, 247, 313, 444, 456
AIC (*stats*), 202
all (*base*), 19
all.vars (*base*), 230, 461, 462
anova (*lme4*), 121, 147, 172, 214, 226, 240, 241, 257, 265, 322, 329, 330, 342, 459
anova.lme (*nlme*), 446
any (*base*), 19, 446
aov (*stats*), 120, 158, 218, 221–3, 305, 308, 319, 321, 350
aovFbyrow (*hddplot*), 396
aperm (*base*), 61, 452
apply (*base*), 89, 139, 141, 157, 182, 247, 301, 323, 328, 344, 408, 409, 430, 456, 457, 470
apropos (*utils*), 7, 469
ar (*stats*), 286, 287
arcsin, 279
args (*base*), 19, 439, 440
ARIMA, 289, 290, 294, 296–8
arima (*stats*), 288, 289, 297, 302
arima.sim (*stats*), 243, 291
array (*base*), 60, 139
arrows (*graphics*), 29
as (*methods*), 450
asin, 279
as.character (*base*), 247, 444, 463, 486
as.data.frame (*lme4*), 199, 455
as.data.frame.table (*base*), 257, 456
as.Date (*base*), 54, 441–3, 458, 481
as.expression (*base*), 90, 475, 476, 484
as.factor (*base*), 376
as.integer (*base*), 101, 214, 274, 275, 442
as.list (*mulltest*), 437
as.matrix (*Matrix*), 409, 412, 450
as.numeric (*base*), 247, 317, 413, 444
as.vector (*Matrix*), 157, 184, 208, 317, 349, 449

attach (*base*), 17, 36, 39, 45, 75, 313, 393, 413, 430, 431
attr (*base*), 96, 312, 315, 443, 444
attributes (*base*), 349
auto.arima (*forecast*), 288–91, 293, 294, 296–8
axis (*graphics*), 25, 26, 48, 117, 293, 296, 317, 458, 475, 476
axis_flip (*ggplot2*), 491

barchart (*lattice*), 32
bestsetNoise (*DAAG*), 197
binomial (*stats*), 247, 280
bmp (*grDevices*), 472
boot (*boot*), 130–2, 140, 156, 157
boot.ci (*boot*), 131, 132, 140, 156
bounce (*DAAG*), 458
box (*graphics*), 117
Box.test (*stats*), 297, 298, 301
boxcox (*MASS*), 161, 168
boxplot (*graphics*), 33, 47, 137, 313, 353, 394, 489
bs (*splines*), 242
bwplot (*lattice*), 32, 47, 52, 74, 76, 477

c (*base*), 2, 5, 6, 10–19, *passim*
C (*stats*), 295
call (*base*), 462, 463, 488
callsmooth, 429
cast (*reshape*), 454, 455
cat (*base*), 19, 256, 438, 470
cbind (*base*), 19, 169, 177, 251, 296, 297, 323, 388, 390, 444, 455
chisq.test (*stats*), 114, 116, 118
CIcurves, 230, 232
citation (*utils*), 468
class (*base*), 15, 22, 459, 460
cloglog, 279
cloud (*lattice*), 32, 379
cm.colors (*grDevices*), 474
cmdscale (*stats*), 384, 385, 409, 423
coef (*lme4*), 81, 150, 156, 169, 182, 183, 208, 215, 337, 340, 341, 459, 460
coefficients (*stats*), 459
col (*base*), 388, 390, 408, 420
colnames (*base*), 18, 19, 316, 370, 449, 458
colors (*grDevices*), 27, 28, 474
Commander (*Rcmdr*), 428
compareTreecalcs (*DAAG*), 367, 370
complete.cases (*stats*), 24, 38, 381, 383, 412, 447
confusion, 408
contour (*graphics*), 474
contr.sum (*stats*), 445
contr.treatment (*stats*), 445
coord_equal (*ggplot2*), 491
cor (*stats*), 67, 68, 76, 132, 140, 172, 177, 201, 250, 251
cor.test (*stats*), 68, 113
cos (*base*), 27, 28
count.fields (*utils*), 435
cox.zph (*survival*), 278
coxph (*survival*), 277
crossprod (*Matrix*), 450
CrossTable (*gmodels*), 63
cumprod (*base*), 19
cumsum (*base*), 19
curve (*graphics*), 141, 464
cut (*base*), 101, 409, 423, 443, 475
CVbinary (*DAAG*), 255, 256, 388
cvdisc (*hddplot*), 402–4
CVlm (*DAAG*), 154
cvscores (*hddplot*), 404, 405

daisy (*cluster*), 384
data (*utils*), 9, 199, 302, 393, 408
data.frame (*base*), 3, 7, 88, 89, 101, 117, 119, 150, 156, 181, 228, 230, 232, 243, 270, 292, 328, 356, 360, 370, 383, 394, 395, 436, 437, 444, 450, 453, 484
datafile (*DAAG*), 433, 436
date (*base*), 443
dbConnect (*RSQLite*), 438
dbDisconnect (*RSQLite*), 438
dbDriver (*RSQLite*), 438
dbGetQuery (*RSQLite*), 438
dbinom (*stats*), 82
dbListTables (*RSQLite*), 438
dbWriteTable (*RSQLite*), 438
defectiveCVdisc (*hddplot*), 402
demo (*utils*), 25, 29, 474
density (*stats*), 45, 129, 250, 409, 462, 491
densityplot (*lattice*), 32, 53, 89, 94
deparse (*base*), 23, 476
detach (*base*), 17, 23, 39, 45, 75, 313, 393, 413
dev.copy (*grDevices*), 472
dev.off (*grDevices*), 31, 464, 472
dfbetas (*stats*), 185, 186, 191
dichromat (*dichromat*), 474
diff (*base*), 19, 288, 302, 441
dim (*base*), 24, 166, 375, 376, 402, 448–52
dimnames (*base*), 60, 197, 257, 291, 415, 449, 452
dir (*base*), 21, 431, 432
dist (*stats*), 384, 409
dist.dna (*ape*), 384, 409
divideUp (*hddplot*), 399, 400
dnorm (*stats*), 85, 87, 133
do.call (*base*), 462
dotchart (*graphics*), 39
dotplot (*lattice*), 32, 40, 64, 75, 123, 261
doubleYScale (*latticeExtra*), 486, 487
dpois (*stats*), 83

edit (*utils*), 15
eqscplot (*MASS*), 409
equal.count (*lattice*), 239
errorsINseveral (*DAAG*), 206–7

errorsINx (*DAAG*), 205, 206, 216
eval (*base*), 462, 463
example (*utils*), 7, 272
exp (*base*), 54, 89, 160, 177, 188, 195, 254, 259, 260, 263, 264, 271, 277, 278, 316, 375, 418, 419, 426, 481
expand.grid (*base*), 378, 381
expression (*base*), 29, 90, 137, 145, 336, 339, 475, 482–4, 487

factor (*base*), 13, 39, 40, 75, 144, 157, 214, 215, 218, 260, 265, 268, 344, 350, 356, 383, 416, 418, 444, 445, 447, 489
fig1, 464
fig2, 464
file.choose (*base*), 431
file.show (*base*), 433
filled.contour (*graphics*), 474
fisher.test (*stats*), 116
fitsmooth, 429
fitted (*stats*), 81, 145, 168, 192, 219, 254, 260, 311, 327, 333, 336, 338, 459, 483
fixef (*nlme*), 311
fligner.test (*stats*), 260
for (*base*), 42, 75, 87, 129, 250, 291, 367, 370, 375, 415, 493
forecast (*forecast*), 289, 290
format (*base*), 54, 441, 442, 458, 481
formula (*lme4*), 461, 484
ftable (*stats*), 61–3
function (*base*), 22, 23, 61, 62, 64, 89, 101, 123, 130–3, 139–41, 156, 169, 230, 261, 292, 328, 336, 338, 344, 374, 375, 380, 408, 409, 415, 416, 418, 421, 427, 429, 433, 441, 451, 461–5, 469, 475, 476, 483, 484
funlik, 133
funRel, 169

gam (*mgcv*), 235, 236, 240, 243, 282
garch (*tseries*), 299, 300
geom_point, etc. ((*ggplot2*), 488–91
getAnywhere (*utils*), 459
getwd (*base*), 4
ggplot (*ggplot2*), 488, 490, 491
glm (*stats*), 247, 252, 253, 255, 257–60, 262, 263, 265, 266, 268, 281, 333, 386–8, 424
gpar (*grid*), 61
gray (*grDevices*), 491
grep (*base*), 21
grid (*graphics*), 491
gui (*fgui*), 427, 429

hardcopy (*DAAG*), 464
hatvalues (*stats*), 184, 185
hcl (*grDevices*), 474
hcopy, 464
head (*Matrix*), 15, 18, 41, 64, 263, 454, 455
heat.colors (*grDevices*), 474
help (*utils*), 7, 8, 11, 16, 21, 26–9, 34–6, 48, 57, 140, 166, 221, 232, 235, 242, 288, 294, 312, 364, 384, 433, 434, 437, 441–3, 445, 448, 461, 468, 472, 492
help.search (*utils*), 7, 440, 469
help.start (*utils*), 8
hist (*graphics*), 7, 33, 45, 87
histogram (*lattice*), 32, 491
history (*utils*), 19
HoltWinters (*stats*), 300
houseprices.fn, 156
HPDinterval (*lme4*), 312, 316, 343

I (*base*), 40, 194, 215, 228–30, 240, 242, 243, 252, 253, 259, 342–4, 399, 424, 444, 453, 487, 488, 490
identify (*graphics*), 29, 41
identify3d (*Rcmdr*), 491, 492
if (*base*), 418, 439, 462, 475
ifelse (*base*), 443
image (*graphics*), 7, 474
importance (*randomForest*), 370
install.packages (*utils*), 5, 427, 485
interaction.plot (*stats*), 123, 331
intervals, intervals.lme (*nlme*), 467
invisible (*base*), 30, 31, 408, 418
is.character (*base*), 441
is.factor (*base*), 19
is.logical (*base*), 19
is.matrix (*base*), 19
is.na (*base*), 12, 19, 24, 36, 38, 374, 375, 415, 446, 447
is.null (*base*), 439
isoMDS (*MASS*), 384, 385, 423
italic (*grDevices*), 29, 475

jpeg (*grDevices*), 472
julian (*base*), 442, 443

lag.plot (*stats*), 284
lapply (*base*), 456, 457, 475, 476, 484
larrows (*lattice*), 32
lars, 199
latticist (*latticist*), 485
layout (*graphics*), 166, 477
layout.show (*graphics*), 477
lazyfoo, 464, 465
lda (*MASS*), 385, 387–90, 392, 394, 395, 399, 408, 409, 419–22
lda.formula, 389
legend (*graphics*), 413, 474
length (*base*), 19, 41, 101, 103, 104, 107, 111, 129, 144, 157, 344, 383, 409, 415, 453, 469, 470, 475, 476
levels (*base*), 13, 52, 218, 327, 330, 413, 418, 444, 447, 475, 479
leverage.plots (*car*), 186

library (*base*), 9, 12, 14, 18, 21, *passim*
lines (*monoProc*), 25, 26, 32, 45, 50, 75, 87, 145, 150, 192, 228, 230, 234, 237, 279, 293, 295, 296, 317, 360
list (*base*), 16, 18, 23, 31, 436, 437, 440, 442, 453, 456, 462, 463, 475, 476, 479, 481–4, *passim*
llines (*lattice*), 32
lm (*stats*), 80, 88, 127, 143, *passim*
lm.influence (*stats*), 185
lm.ridge (*MASS*), 203, 215
lme (*nlme*), 127, 303, 312, 446
lmer (*lme4*), 127, 303, 305, 310–12, 314–16, 318, 319, 326, 327, 329, 330, 338, 342, 349, 350
lmList (*nlme*), 337, 340, 341, 460, 461
load (*base*), 431
local (*base*), 463
locator (*graphics*), 29
locpoly (*monoProc*), 237
loess (*stats*), 168, 236, 237, 359, 360
log (*base*), 40, 42, 51, 54, *passim*
log10 (*base*), 48, 49, 51, 386
log2 (*base*), 39
logisticsim (*DAAG*), 281
logit (*car*), 245, 248, 279, 281, 421, 424
loglm (*MASS*), 272
lowess (*stats*), 50, 79, 145, 168, 192, 236, 237, 242, 279, 292, 293, 295, 296, 317
lpoints (*lattice*), 32
lqs (*MASS*), 149, 192, 194, 216
ls (*base*), 4, 19, 21, 430, 431
lsegments (*lattice*), 32
ltext (*lattice*), 32

mad (*stats*), 66
mantelhaen.test (*stats*), 139
margin.table (*base*), 61, 96
Markov, 100, 101
match (*base*), 24, 316, 332, 400
matplot (*graphics*), 88
matrix (*base*), 18, 89, 116, 117, 166, 197, 291, 301, 367, 370, 383, 402, 403, 415, 430, 448–50, 470, 492
max (*base*), 78, 380, 470
MCMCregress (*MCMCpack*), 165, 166
mcmcsamp (*lme4*), 312, 316, 343
MDSplot (*randomForest*), 406
mean (*Matrix*), 12, 13, 17, 20, 22, 23, 75, 100, 103, 107, 110, 111, 129, 144, 292, 293, 311, 323, 340, 375, 462, 463
mean.and.sd, 22, 140, 439
median (*stats*), 12, 17, 20, 23, 74, 104, 130, 131
median.fun, 130
melt (*reshape*), 454, 455, 484
merge (*base*), 455
methods (*utils*), 459, 469
min (*base*), 380, 470
model.matrix (*lme4*), 163, 172, 185, 218, 234, 243, 445
model.tables (*stats*), 223, 224, 322
monoproc (*monoProc*), 237
months (*base*), 442
mosaic (*vcd*), 60, 61
mosaicplot (*graphics*), 60, 61, 139
mt.maxT (*multtest*), 396, 397
mtext (*graphics*), 25, 26, 29, 49, 291, 474, 477
muhaz (*muhaz*), 277
multinom (*nnet*), 390

na.omit (*stats*), 24, 374, 375, 380, 381, 390, 412, 447
names (*base*), 15, 18, 19, 25, (*passim*)
nchar (*base*), 6, 440
new.env (*base*), 465
newtest, 463
nls (*stats*), 210, 211
normalizePath (*utils*), 432
nrow (*base*), 100
ns (*splines*), 231–4, 418, 420, 421, 424, 425, 488, 489
nswlm, 418, 426
numeric (*base*), 25, 100, 129, 299, 383, 415

object.size (*utils*), 432
objects (*base*), 465
onetPermutation (*DAAG*), 129
oneway.plot (*DAAG*), 120
onewayPlot (*DAAG*), 120
open3d (*rgl*), 492
options (*base*), 25, 34, 87, 88, 143, 218, 221, 387, 390, 445, 448
opts (*ggplot2*), 491
order (*base*), 20, 75, 194, 323, 403, 448
ordered (*base*), 14, 270, 444, 445
orderFeatures (*hddplot*), 395, 398
outer (*base*), 141, 451
overlapDensity (*DAAG*), 139, 421
oz (*oz*), 407

pacf (*stats*), 284, 293
package.skeleton (*utils*), 466
pairs (*graphics*), 30, 179, 199, 249, 251, 412
palette (*grDevices*), 27
panel.abline (*lattice*), 483, 486
panel.average (*lattice*), 64, 123, 261, 483
panel.curve (*lattice*), 483
panel.densityplot (*lattice*), 90
panel.dotplot (*lattice*), 64, 123, 261
panel.identify (*lattice*), 486
panel.lines (*lattice*), 483, 486
panel.points (*lattice*), 483
panel.rug (*lattice*), 483
panel.smooth (*graphics*), 145, 292
panel.superpose (*lattice*), 336, 339, 483
panel.text (*lattice*), 485, 486
panel.xyplot (*lattice*), 483

par (*graphics*), 7, 26–8, 40, 42, 44, 45, 49, 51, 87, 110, 137, 144, 148, 173, 233, 238, 239, 250, 255, 291, 293, 295, 317, 333, 365, 413, 473, 474, 476, 477, 493
par3d (*rgl*), 492
parallel (*lattice*), 32
paste (*base*), 40, 41, 48, 54, 144, 170, 177, 179, 188, 193, 197, 247, 248, 291, 293, 317, 323, 356, 375, 378, 416, 440, 451, 461, 464, 481, 484
pbinom (*stats*), 82, 83
pdf (*grDevices*), 464, 472
persp (*graphics*), 141
pexp (*stats*), 99
phantom (*grDevices*), 29
playwith (*playwith*), 485
plot (*graphics*), 2–4, 7, 14, 25–30, *passim*
plot.mcmc (*MCMCpack*), 166
plot.mtcars, 461
plot.ts (*stats*), 48
plotcp (*rpart*), 364, 366
plotTrainTest (*hddplot*), 400
png (*grDevices*), 472
pnorm (*stats*), 84, 85, 106, 193
points (*graphics*), 25–8, 32, 48, 407, 470
poissonsim (*DAAG*), 282
polr (*MASS*), 270, 271
poly (*stats*), 187, 188, 194, 195, 229, 232, 233, 242, 243, 298
polygon (*graphics*), 85, 473
postscript (*grDevices*), 472
ppoints (*stats*), 137, 397
ppois (*stats*), 84
prcomp (*stats*), 169
predict (*stats*), 149, 150, 156, 157, 178, 188, 200, 211, 219, 228, 230, 254, 263, 360, 389, 395, 408, 421, 459
predict.lm (*stats*), 157, 215
pretty (*base*), 54, 85, 87, 150, 232, 293, 296, 317, 475, 481
princomp (*stats*), 380, 381, 383, 412
print (*base*), 7, 20, 21, 23, 31, 34, 36, 52, 62, 101, 111, 197, 291, 349, 366, 369, 375, 383, 408, 418, 429, 451, 459, 460, 464, 477, 479, 480, 482
print.data.frame (*base*), 21
print.default (*base*), 22
print.factor (*base*), 21, 459
print.lm (*stats*), 459
print.summary.lm (*stats*), 459
printcp (*rpart*), 355, 364–6
prod (*base*), 133, 402
profile (*stats*), 312
prompt (*utils*), 467
prop.table (*base*), 96, 97
prune (*rpart*), 364, 367
prune.rpart (*rpart*), 367
pt (*stats*), 106, 107
q (*base*), 3, 5, 35
qbinom (*stats*), 83
qda (*MASS*), 389, 394, 408, 409
qf (*stats*), 396, 397
qnorm (*stats*), 85, 106
qplot (*ggplot2*), 408
qqmath (*lme4*), 32, 93, 94, 144, 327, 328
qqnorm (*stats*), 93, 132, 298, 317, 341
qqplot (*stats*), 137, 397
qqthin (*hddplot*), 397
qr.solve (*base*), 450
qreference (*DAAG*), 93, 94, 144
qt (*stats*), 106, 107, 140, 150, 171
qtukey (*stats*), 220
quantile (*stats*), 46, 423
quarters (*base*), 442
quartz (*grDevices*), 28
quickplot (*ggplot2*), 383, 408, 487–91
qunif (*stats*), 137

R.home (*base*), 432, 468
rainbow (*grDevices*), 474
randomForest (*randomForest*), 351, 369–72, 374–6, 408, 419–21, 423, 424
ranef (*nlme*), 316, 327, 328
range (*base*), 6, 12, 20, 45, 49, 234, 242, 263, 341, 370, 457, 475
rattle (*rattle*), 429
rbind (*base*), 114, 261, 400, 416, 418, 455
rbinom (*stats*), 86, 87, 215, 216
rchisq (*stats*), 99
read.csv (*utils*), 434
read.delim (*utils*), 434
read.dna (*ape*), 409
read.fwf (*utils*), 436
read.table (*utils*), 8, 34, 35, 434–7
readLines (*base*), 435, 436
rect (*graphics*), 476
refit (*lme4*), 328
regsubsets (*leaps*), 197
relevel (*stats*), 120, 257, 262, 263, 332
reorder (*stats*), 304
rep (*base*), 12, 13, 23, 39, 75, 90, 93, 117, 119, 137, 144, 157, 166, 169, 268, 270, 323, 349, 383, 396, 437, 445, 451, 458, 476
resid (*stats*), 81, 168, 260, 279, 289, 293, 296–8, 301, 333, 339, 344
residuals (*stats*), 118, 144, 145, 185, 192, 194, 211, 239, 260, 266, 317, 327, 459
residuals.glm (*stats*), 266
rev (*base*), 20, 453
rexp (*stats*), 88, 99, 140
rfun, 462
rgb (*grDevices*), 474
rlm (*MASS*), 149, 167, 241
rm (*base*), 4, 21, 33, 40, 430, 431, 440

rnorm (*stats*), 22, 23, 87–9, 93, 99, 137, 138, 141, 144, 169, 197, 215, 299, 402, 430, 439, 450, 462, 470, 471, 475, 484
rollmean (*zoo*), 101
round (*base*), 54, 61, 62, 81, 96, 97, 202, 248, 256, 375, 388, 408, 416, 423, 425, 476, 481
row (*base*), 291, 388, 390, 403, 408, 420
row.names (*base*), 26, 29, 38, 39, 407, 409, 486, 492
rownames (*base*), 15, 19, 75, 192, 340, 341, 449
rpart (*rpart*), 354–8, 360, 364–7, 369–72
rpois (*stats*), 87, 100
RSiteSearch (*utils*), 8, 406
rt (*stats*), 99, 137
rug (*graphics*), 49, 50, 313
runif (*stats*), 88, 99, 215, 439, 470

s (*mgcv*), 236, 238, 240, 242, 243
sammon (*MASS*), 384, 385, 409
sampdist, 90, 94
sample (*base*), 90, 99, 100, 129, 153, 256, 349, 353, 375, 376, 383, 394, 470
samplingDist, 89
sampvals, 89, 90
sapply (*base*), 20, 21, 35, 42, 53, 66, 223, 260, 261, 311, 322, 344, 374, 375, 380, 415, 416, 441, 456, 457, 469, 484, 493
save (*base*), 430, 431
save.image (*base*), 33, 431
scale (*base*), 328
scale_x_continuous, etc. (*ggplot2*), 328
scan (*base*), 434, 436, 437
scatter3d (*Rcmdr*), 428, 491, 492
scatterplot (*car*), 428
scoreplot (*hddplot*), 395, 396, 404, 405
sd (*stats*), 22, 23, 66, 103, 104, 107, 144, 339, 471
search (*base*), 430
seq (*base*), 12, 41, 54, 88, 90, 101, 117, 230, 323, 353, 360, 442, 458, 470, 481
sessionInfo (*utils*), 9, 431
set.seed (*base*), 86, 87, 90, 141, 156, 291, 312, 402
setwd (*base*), 431
show.colors (*DAAG*), 474
show.settings (*lattice*), 481
showMethods (*methods*), 460
showprop, 415
signif (*base*), 129, 316
simpleKey (*lattice*), 479
simpleTheme (*lattice*), 57, 58, 378, 379, 423, 479, 480, 482
simulate (*lme4*), 88, 328
simulate.distribution, 462
simulateLinear (*DAAG*), 159
simulateScores (*hddplot*), 396
sin (*base*), 27, 28
sink (*base*), 437, 438
slot (*methods*), 460
slotNames (*methods*), 460
smooth (*stats*), 488, 491
smooth.spline (*stats*), 235
solve (*Matrix*), 450
sort (*base*), 6, 20, 23, 192, 412, 416, 448
source (*base*), 125
spline (*stats*), 230
split (*base*), 66, 90, 104, 130, 137, 260, 261, 311, 313, 322, 344, 353, 457, 458, 469
splom (*lattice*), 32, 175, 179, 181, 191, 378, 412, 416
sqrt (*base*), 2, 34, 40, *passim*
stack (*utils*), 18, 41, 117, 454
stem (*graphics*), 46
str (*utils*), 15, 20, 96, 274
strheight (*graphics*), 29, 458
strip.custom (*lattice*), 90, 483, 484
stripplot (*lattice*), 32, 52, 120, 217, 304, 477, 483, 484, 486
strsplit (*base*), 436, 440, 441, 469, 475
StructTS (*stats*), 300
sub (*base*), 52
subset (*base*), 12, 16, 44, 114, 263, 274, 275, 415, 416, 418, 419, 482, 489
substitute (*base*), 23, 475, 476, 484
substring (*base*), 340, 341, 440
sum (*base*), 12, 39, 61, 62, 247, 260, 344, 374, 375, 388, 390, 408, 409, 415, 420, 441
summary (*multtest*), 6, 15, 34, 36, *passim*
summary.aov (*stats*), 218, 330
summary.lm (*stats*), 186, 218, 220–3
summary.rpart (*rpart*), 366
supsmu (*stats*), 486
Surv (*survival*), 274, 276, 277
survfit (*survival*), 274, 276
svm (*e1071*), 406
Sweave (*utils*), 467
switch (*base*), 415, 462
symbols (*graphics*), 29, 473
sys.call (*base*), 463, 464
Sys.Date (*base*), 443
sys.frame (*base*), 463, 465
Sys.getenv (*base*), 432
sys.nframe (*base*), 463
sys.parent (*base*), 463
system.file (*base*), 432, 466
system.time (*base*), 450, 471

t (*Matrix*), 88, 117, 312, 316, 344, 395, 449, 470
t.test (*stats*), 107, 109, 111, 137, 341
table (*base*), 20, 21, 39–41, (*passim*)
tail (*utils*), 15
tapply (*base*), 317, 456
termplot (*stats*), 179, 193, 194, 212, 224, 255, 419
test, 463
testfun, 476
text (*graphics*), 25, 26, 29, 32, 39, 41, 75, 117, 170, 192, 247, 248, 341, 354, 356, 360, 365, 367, 407, 409, 458, 474, 476

text.rpart (*rpart*), 356
textGrob (*grid*), 484, 486
theme_bw (*ggplot2*), 490
theme_gray (*ggplot2*), 491
theme_set (*ggplot2*), 491
tiff (*grDevices*), 472
title (*graphics*), 39, 474, 475
topo.colors (*grDevices*), 474
toupper.initial, 330
trellis.device (*lattice*), 31, 480, 481
trellis.focus (*lattice*), 485, 486
trellis.panelArgs (*lattice*), 485, 486
trellis.par.get (*lattice*), 480
trellis.par.set (*lattice*), 57, 480, 481
trellis.unfocus (*lattice*), 486
ts (*stats*), 14, 48, 292, 458
tuneRF (*randomForest*), 369, 370
twot.permutation (*DAAG*), 129

unclass (*base*), 40, 170, 297, 317, 397, 409, 444, 475
unique (*base*), 20, 144, 260, 311, 332, 416, 469
unit (*grid*), 484, 486
unlist (*base*), 16, 35, 54, 418, 454, 481
unstack (*utils*), 18, 41, 323
update (*stats*), 52, 54, 57, 58, 263, 331, 343, 479–83, 485
update_geom_defaults (*ggplot2*), 491
UseMethod (*base*), 459

var (*stats*), 66, 100, 104, 111, 121, 309
VarCorr (*lme4*), 315, 318, 349
vif (*DAAG*), 201–3
vignette (*utils*), 468
vis.gam (*mgcv*), 240

weekdays (*base*), 442, 443
which (*base*), 20, 192
which.max (*base*), 20
which.min (*base*), 20
white.test (*tseries*), 299
window (*stats*), 14, 48, 49, 458
windows (*grDevices*), 28
wireframe (*lattice*), 32
with (*base*), 17, 18, 20, 25, 26, 29, 36, *passim*
woolf, 139
write (*base*), 437
write.table (*utils*), 437

x11 (*grDevices*), 28
xlab (*ggplot2*), 383
xtable (*xtable*), 467
xtabs (*xtabs*), 20, 21, 61–3
xyplot (*lattice*), 30–2, 36, 53, 55–8, 101, 239, 327, 336, 339, 340, 381, 422, 423, 429, 442, 478–87

ylab (*ggplot2*), 383

z.inverse, 140
z.transform, 140

Index of terms

Akaike information criterion (AIC), *see* information criteria
analysis issues
  analysis of summary data, 20, 59, 61, 63, 65, 67
  analysis styles & traditions, xix, 43, 44, **71–3**, 77, 136
  assumptions, 43–5, 58–9, **91–8**, 115, 117–18, 221
  changes of plan, 73
  data snooping, 43
  deficiencies, 72
  independence, 73, 91, 92, 115, 133, 283, 302
  non-parametric methods, **95–6**, 98, 240, 352, 373
  planning, 72
  presentation issues, 111
  prior information, 135
  random sampling assumptions, 91
  robust & resistant methods, 91, 147, 149, 167, 183, 184, 191–4, 212, 236, 241, 295, 296
  significance tests, 171
  source/target differences, 153, 158, 189, 303, 307, 328, 361, 385, 387–8, 414, **493–4**
  strategies, 71, 72, 214
analysis of variance
  decomposition (anova), 158, 159, 321
  degrees of freedom, 121, 305–6
analysis of variance model (aov), 120–1, 158–9, 218–23, 305, 308, 319, 321–2, 330, 350
  as linear model (lm), 217–21
  categorized data *vs* linear fit, 125–6, 158–60
  multi-way, 127, 222–305, 319–32
  multiple comparisons, *see* inference
  one-way layout, 120–2, 127, 218, 305, 307, 309, 313
argument, *see* function
array, 60, 139, 256, 448, **451–2**, 455–6
  dimensions, 139, 256, 448, 451, 456
  permutation of dimensions, 61, 452
assignment, 2, 20, 35, 36, 446
  subscripted, 446

Bayesian Information Criterion (BIC), *see* information criteria
Bayesian methods, 8, 98, **132–6**, 165, 166, 312, 388
  posterior density or probability, 133, 134
  prior density or probability, 133–5, 141, 312, 386–7
bootstrap, *see* resampling methods

C_p statistic (Mallows), *see* information criteria
censoring, *see* survival analysis
classes & methods, 30, 443, 458–61
  S3, 458–61
  S4, 460, 461
coefficients
  GLM
    standard error, 258, 263–4, 267, 281
  linear model
    confidence intervals, 171–2
    correlation between estimates, 172–3, 181, 229
    interpretation, 174–83, 208–9
    regression spline coefficients, 234–5
    standard error, 220, 222, 226, 228, 231
commands
  comment character, 2
  continuation character, 2
concatenation, 3, 10
confidence interval (CI), 68, 107–9, 111–13, 124–5, 149–50, 156, 171, 278, 426
  1 & 2-sample means, 107–9, 111–12, 140, 301–2, 341
  autocorrelation, 289
  correlation, 68, 113, 131, 132, 140
  median, 130–2
  predicted values, 149–50, 171–2, 187–8, 215, 228–30, 232–3, 238, 275, 290, 426
  proportions, 107, 112, 113
  regression coefficients, 149–50, 171, 220, 312, 316, 426
confounding, 119
contrasts, *see* factor
correlation, 67–9, 76, 110, 147, 163, 172, 177, 310, 316, 334–6
  Kendall, 68
  linear (Pearson), 67, 68, 76, 113, 140, 163, 201, 207–8, 284–5

confidence interval, 68, 113, 140
confidence interval (bootstrap) 131–2
rank (Spearman), 67, 68
R^2, *see* regression, linear model
cross-validation
see resampling methods

data
database connection, 438
input, 4, 427, 433–5, 437
comment character, 2, 435, 437
tracking errors, 435
management strategy, 33, 35, 430, 431
manipulation
apply family of functions, 456–8
convert to/from tables or matrices, 257
combine data objects (cbind, rbind), 19, 169, 177, 251, 296–7, 323, 388, 390, **444–5**
computational efficiency, 450
count & identify NAs, 24, 38, 381, 383, 412, **447**
merge data frames, 455
omit NAs, 24, 316, 374–5, 380–1, 390, 412, 447–8
plyr package: aaply, daply & related functions, 450
reshape data frames, 454, 484
selection & matching, 23–4
split by levels, 64, 66, **457–8**, *passim*
subset, 11–12, 15–16, 23, 31, 44, *passim*
transpose matrix, data frame, 395, 449
measurement issues, 79, 81, 84, 169, 213, 334, 335, 426
output
database connections, 438
decimal places, 30
write to file, 437–8
patterned, 11, 12
summary, 20, 59, 61, 63, 65, 67
data analysis & commentary
(Data sets are in *DAAG*, unless otherwise indicated)
Antarctica 800KYr ice core (`edcCO2`, `edcT`), 486
aberrant crypt foci (`ACF1`), rat colons, 258
acupuncture, real *vs* sham, 138, 139
admission rates, contrived data, 140
AIDS (`Aids2`, *MASS*), survival, 274, 275, 277
alcohol consumed, by year, country (`grog`), 478
anesthetic, effect on pain (`anesthetic`), 246–7
animal body, brain weight (`Animals`, *MASS*), 51, 67
Antiguan corn yields (`ant111b`), 304, 310
apple taste (`appletaste`), 223, 224
Australian athletes, morphology & blood (`ais`), 31, 482, 489
blood pressure, rabbits (`Rabbit`, *MASS*), 41
book weight, dimensions (`allbacks`, `softbacks`), 170, 173, 185, 186, 208, 214
biased sample (`oddbooks`), 168, 181, 182
brain, body weight, litter size (`litters`), 179, 180, 215
Canadian city populations (`cities`), 214, 445
cancer, gene expression data (`Golub`, `golubInfo`, *hddplot*), 392–405
car data
1993 Consumer reports (`Cars93`, *MASS*; `Cars93.summary`), 14, 455, 469
fuel consumption, etc, 1974 data (`mtcar`, *datasets*), 461
mileage *vs* weight (`car.test.frame`, *rpart*), 360
mileage, etc (`table.b3`, *MPV*), 215
speed *vs* distance to stop (`cars`, *datasets*), 167
car window tinting (`tinting`), 56, 329
carbon emissions *vs* year (`fossilfuel`), 3–5
CO_2 level & leaf temp (`leaftemp`), 225
comparison of firing methods (`Gun`, *MEMSS*), 349
cricketer life spans (`cricketer`), 282
cuckoo egg sizes (`cuckoos`), 52, 70
dengue projections (`dengue`), 65
depression, lawn roller weight (`roller`), 78–80, 88, 143, 147, 150, 151, 163–5
dewpoint, min & max temp (`dewpoint`), 238
distance *vs* ramp angle, starting point (`toycars`), 241
drawings of dreams, 116
effort *vs* stool type (`ergoStool`, *MEMSS*), 350
egg dimensions
host *vs* cuckoo eggs (`cuckoohosts`), 70
elastic bands, distance *vs* stretch –stretc+ heated *vs* unheated (`elastic1`, `elastic2`, `elasticband`)
paired (`pair65`), 94, 103, 105, 107, 110, 141
unpaired (`two65`), 139
electrical resistance, kiwifruit (`fruitohms`), 50, 69
email spam (`spam7`), 352–6, 366, 367, 374
Fisher's iris data (`iris`, *MASS*), 41
Food Frequency Questionnaire (FFQ), 204
frogs, spatial distribution (`frogs`), 256
gastric cancer screening, 119
geological substratum thickness *vs* distance (`geophones`), 241, 242
grain/head *vs* seed rate (`seedrates`), 228
head injury, simulated (`head.injury`), 374
height & weight of women (`women`, *datasets*), 204, 209, 214
hill race times (`nihills`, `hills2000`), 174, 175, 188, 189, 191, 214, 215, 438
house prices (`houseprices`), 157, 158, 312
human power, O_2 intake (`humanpower1`, `humanpower2`), 336, 338, 460
insurance claims (`Insurance`, *MASS*), 469

data analysis & commentary (*cont.*)
iron slag, magnetic *vs* chemical values (`ironslag`), 145
jobs, by Canadian region (`jobs`), 14, 18, 21, 32, 53–5, 442, 454, 455, 458, 481
kidney stone operations, 60, 61
kiwifruit shading (`kiwishade`), 69, 319, 320, 324, 325, 334, 345
labor training program (`nswdemo` & related datasets), 59–60, 71, 114–16, 136–8, 168, 374, 414–27
lake area, elevation (`Manitoba.lakes`), 38, 39
levels of Lake Huron (`LakeHuron`, *datasets*), 283–7, 289, 290
maths achievement (`MathAchieve`, *MEMSS*), 350
measles, deaths in London (`measles`), 47, 48
milk sweetness by additive (`milk`), 49
model cars (`modelcars`), 125
mortality, female heart attacks (`mifem`), 281, 364, 374
moths, occurrence (`moths`), 261–4, 266, 281, 332, 333
oil platform escape times (`ex01.36`, *Devore6*), 40
ozone levels (Antarctic), 1956–2000 (`ozone`), 242
plant architecture, leafshape (`leafshape`), 408
population growth
Australian states (`austpop`), 25
possums, morphometric data (`possum`), 44, 378–80
pressure of mercury vapour, *vs* temp (`pressure`, *datasets*), 168
primate body, brain weights (`primates`), 25, 26, 30
rare plants, habitat type (`rareplants`), 117, 118
record times, track, road (`worldRecords`), 241
rice variety 2-factor expt (`rice`), 123, 124
road accident mortality –mortalit+ US (`nassCDS`), 62, 63
science, school survey (`science`), 72, 313, 315
seal morphometrics (cross-sectional data) (`cfseal`), 17, 162
selfed *vs* crossed plants (`mignonette`, Darwin's data), 110, 140
skull growth in children (`Orthodont`, *MEMSS*), 340
social support survey (`socsupport`), 40, 71, 72, 410–13
Southern Oscillation Index, rainfall (`bomregions`), 291–2, 295–7, 302, 429, 487–8
space shuttle damage (`orings`), 38
sugar weight, wild type *vs* GM plants (`sugar`), 218
tomato yield, salinity (`ex10.22`, *Devore6*), 75
tree dimensions, biomass (`rainforest`), 12, 38, 457
UCB admissions, by sex & dept (`UCBAdmissions`, datasets), 96–7, 139, 210, 256–7, 272
wages, UK 19th C cotton workers (`wages1833`), 242
census data *vs* informal survey (`cottonworker`), 40
data frame, 3–4, 8, **14–19**, 20–1, 35, **452–8**, *passim*
see also data, manipulation
as database, 371, 430, 438
as list, 16–17, 453
attach & detach, 17, 35–6
names & numbers of rows & columns, 60, 257, 291
writing, 437, 440,
data mining, 158, 351, 406, 407, 429
dates, 441–3
degrees of freedom, **66–7**, 76, 105, 109–12, 115, 127, 147, 149–50, 229–30, 232, 246, 267, 305–7, 315, 319, 325, 330, 401
density
estimate, **44–6**, 76, 102, 426
plot, *see* plot
deviance, *see* model
discriminant analysis
linear, 387, 389, 390, 395, 408, 419, 420, 422–4
distance measure, 384, 385
binary data, 385
diagnostics & diagnostic plots
GLM, 265–6
linear models, 144, 148–9, 170, 173–6, 183, 185–7, 190–1, 194, 195, 214–15, 227, 232–3, 241–3
distribution
t-distribution, 105–8, 137, 264
degrees of freedom, 99, 105–8, 137, 264
Bernoulli, 86
binomial, 82, 83, 86, 245, 266, 267, 279, 425
chi-squared, 85, 99, 114–18, 260, 267, 271, 298
cumulative probability, 82–5, 107, 266
density, *see* density estimate
exponential, 85, 88, 99
heavy-tailed, 86, 105
normal, 45–7, 58, 68, **86–9**, 92–5, 99, **102–8**, *passim*
Poisson, 82–4, 245, 262, 264, 267, 279
quantile or percentile, 26, 46, 83, 85, 131, 132, 266
sampling distribution, 88, 89, 94, 95, 102, 103, 105, 155, 328
t-statistic, 105, 106
bootstrap estimate, 155
median, 130
simulation, 86–9, 99, 101, 134, 141, 165
skew, 45–6, **58**, 71, 89, 92, 94, 95, 113, 190, 417
uniform, 85, 88, 99, 137, 165

document preparation
Sweave, 467–8

experimental design, 123, 222, 319, 325, 334
exploratory data analysis (EDA), 43, 44, 72–4, 77, 378, 379, 381, 383
expression, 1–2, 12, 39, 441, 446, **461–5**, 474–6, 482–4
print on graph, 29, 474–6, 482–4, 487

factor, 10–12, **13–14**, 16–17, 434, 441, **441–6**, 447, 453–6, 459, 479, 482–4, 489, *passim*
coding & contrasts, 195, 218, 220–1, 445–6
columns in model matrix, **218–20**, 225
in model formula, 372, 445
levels, **13–14**, 17–18, 20, 31, 35, 39, 56, **444–5**, 447, 455–6, 475, 479, 482–4, *passim*
order, 13–14, 444–5
ordered, 14, 444–5
reorder levels, 13, 120, 144, 257, 262–3, 332, 418, 484
split by levels, *see* data, manipulation
file names, 4, 9, 431, 432, 433
fitted values, *see* prediction
function, 3, **19–25**, 429, 431–3, **438–44**, *passim*
anonymous, 441, 451, 475
argument, 4, **23**, 168, 439, 476
abbreviated, 439
lazy evaluation, 464–5
the . . . argument, 439–40
use list to pass arguments, 462–3
common useful functions, 19–20, 431–2
environment, 22, 36, 432–3, **463–5**, 469
evaluation frame, 463
generic, 21, 30, 36, **458–9**, 467, 474
issues for writing & use, 439
return value, 23, 31, 41, 254
utility functions, 21, 431–2, 463, 466, 468

generalized linear model, *see* regression, generalized linear model (GLM)
generalized linear mixed model, *see* regression, generalized linear mixed model
ggplot2 graphics, 33, 383, 408, 472, 487, 488, 491, 492
automatic generation of keys, 383
functions return ggplot objects, 487–8
layers, 488
use of quickplot(), 487–90
graph
see plot
graphics
see also ggplot2; lattice; plot
devices, 27, 30, 31, 57, 464, **472–4**, 477, 479, 480–1
good practice, 32
links with analysis, 44

image file, *see* R session
inference
1 & 2-sample means, 103–14
Bayesian, *see* Bayesian methods
bootstrap methods, *see* resampling methods
confidence interval, *see* confidence interval
confidence interval *vs* hypothesis test, 113–14
hypothesis test, 107–8, 114
likelihood ratio test, 133, 265, 277, 278
maximum likelihood, 133, 141, 286–7, 299, 329, 331, 343
multiple comparison, 73, 119–21, **122–3**, 209, 220, 319, 329, 352, 373–4, 476
Tukey's HSD, 120–2, 220
information criteria
Akaike Information Criterion (AIC), **186–7**, 194–5, 202, 248, 253, 259, 262–4, 270, 271, 287–9, 294, 297, 298, 310, 314, **330**, 332, 338, 342, 343, 425
Bayesian Information Criterion (BIC), 187, 289, 294, 297–8, 310, 314, 330, 332, 338, 342, 343
C_p (Mallows), 187
interaction plot, 124

lattice graphics, 25, **30–3**, 36, 52, 56, 74, 93, 152, 412, 427, 428, 472, 475, **477–87**, 492
add smooth curve, 239, 429, 486
adding to plots, 52, 54, 57–8, 479–85
box plot (bwplot), 32, 47, 52, 74, 76, 477
built on grid package, 33, 477
conditioning factor or variable, 31–2, 428, 479
dotplot, 40, 64, 75, 123, 261
functions return trellis objects, 30–1
interaction with plots, 485–6
keys & legends, 31, 53, 56, 58, 64, 261, 327, 378–9, 381, 416, 423, 442, **479**, 482
layout of panels, 55, 64, 76, 89, 101, 144, 239, 327, 340, 422, 481
panel function, 64, 89, 123, 261, 336, 483
par.settings, 57–8, 378–9, 423, 479–80, 482
point & text size, 412, 479–80, 492
print from user functions, 31, 472, 477
scaling of axes, 33, 475
strip plot, 32, 52, 477, 483, 484, 486
library, 9, 30, 179, 310, 483, 485
see also package
leverage
GLM, 266, 268
linear model, 149, 173, 176, 183–6, 195, 227, 233, 266, 419, 421
plot residuals *vs* leverage, 173, **184–5**, 227, 233
list, 16–18, 23, 35, 81, 374, 440, **452–4**, 457–62, 463, *passim* in code
concatenation, 36
data frame as list, 16, 453

Markov chain, 100, 101
Markov Chain Monte Carlo (MCMC) estimation, 134, 165, 312
matrix, 18–20, 36, 89, 100, 225, 436, 444–5, **448–51**, 455–8, 470, 474, 492, *passim*
  see also data, manipulation
  arithmetic, 450
  combine (cbind, rbind), *see* data, manipulation
  convert to/from data frame, 450, 455–6
  convert to/from vector, 19, 448–9
  extract sub-matrix, 19, 449
  form as outer product, 451
  names & numbers of rows & columns, **18–19**, 449
  storage in memory, 35, **430–1**, 450, 471
  subscripts, 19, 449
  transpose, 395
mean
  trimmed, 75, 76
methods & classes
  see classes & methods
missing values, **12–13**, 20, **24**, 38, 242, 373, 381, 383, 390, **446–7**, 455 *see also* data, manipulation
model
  allometric growth, 161–2, 167
  classification, *see* discriminant
  discriminant, *see* discriminant
  GAM, *see* regression, generalized additive model (GAM)
  GLM, *see* regression, generalized linear model (GLM)
  linear model, *see* regression, linear model
  model formula, **80**, 143, 305, 321, 360, 371–2, 445
  model object, 81, 459
    extractor function, 81, 143, 460
  multinomial (multinom), 390
  parameters, 79, 133, 134, 152, *passim*
  signal & noise, 78, 97, 98, 469, 470
  statistical *vs* deterministic, 77–9, 97
  survival analysis, *see* survival analysis
  time series, *see* time series
multi-dimensional scaling (MDS), *see* ordination
multi-level model, **302–50**
  fitted values, 311–12, 327, 331
  fit by restricted maximum likelihood (REML), 339, 331, 343
  random coefficients, 336–44
  repeated measures, 334–44
  residuals, 306–7, 311, 317, 325, 327–8
  variance components, 303, 308–9, 315–16, 318–19, 322, 323, 325, 326, 336, 344, 345, 349
multivariate analysis, 377–409

nonlinear model (nls), 210–11

object
  save, *see* R session
operator, 11, 14, 23, 444, 460, 465, 488
  arithmetic, 2, 444, 450
  assignment, *see* assignment
  logical, 443
  relational, 11, 14, 23
ordinal logistic model, *see* regression, ordinal logistic
ordination
  distance measure, *see* distance measure
  multi-dimensional scaling (MDS), 383
  principal components analysis (PCA), 169, 196, 198, 203, 347, **377–84**, 386, 407, 410–13, 426
    loadings, 382, 407, 411–13, 426
    regression on PCA scores, 410–13, 426
outliers, 132, 147–9, 169, 170, 183–6, 190, 192, 193, 211, 212, 215, 237, 242, 295, 426

package
  base, 9, 432, 465
  boot, 130, 156
  car, 186
  cluster, 384
  DAAG, 1, 5, 9, *passim*
  datasets, 6, 9, 30, 41, *passim*
  Deducer, 429
  Devore6, Devore7, 40, 75
  dichromat, 474
  dr, 191
  fgui, 427, 429
  forecast, 283, 288–90, 300–1
  foreign, 434
  fseries, 300
  gam, 236
  ggplot2, 33, 383, 408, 472, 491, 492
  golubEsets, 392
  grid, 33, 468, 477, 483, 485
  hddplot, 392, 395, 404
  KernSmooth, 237
  lars, 199
  latticeExtra, 486
  latticist, 427, 485
  lazy data mechanism, 9
  leaps, 197
  lme4, 127, 303, 305, 312, 337, 346, 460
  locfit, 240
  lqs, 192
  MASS, 9, 14, 40–1, *passim*
  MEMSS, 304, 340, 349–50
  mgcv, 235
  monoProc, 237
  muhaz, 277
  multtest, 396
  nlme, 303, 304, 312, 348, 446, 467
  playwith, 427, 485
  plyr, 450

pmg, 427
randomForest, 351, 352, 369, 371, 372, 374, 406, 408
Rcmdr, 427, 491
RColorBrewer, 474
reshape, 454, 484
RMySQL, ROracle, RSQLite, 438
rpart, 351, 352, 357, 371–4
stats, 9, 283, 287, 300
strucchange, 300
survival, 235, 256, 274, 280
tseries, 283, 299
xtable, 467
plot
see also ggplot2; graphics; lattice
aspect ratio, **28**, 31, 64, 93, 120, 123, 144, 217, 239, 328, 422–3, 491
axes, 33, 51, 78, 85, 117, 158, 475, 480–1, 488, 491
box-&-whisker (boxplot), 41, 47, 52, 58, 73, 75, 86, 122, 137, 313, 353, 394, 473, 477, 489
outlier criterion, 47
contour, 240, 408, 479, 489
dates as axis labels, 481
density plot, 32, 44–6, 52–3, 84–5, 89–90, 94, 99, 105–6, 122, 129–30, 133, 251, 408–9, 416, 421, 462, 480, 489–91
dynamic graphics, 484
expression, 29, 476
font family, 473
face, 473
histogram, 38, 44–6, 75, 487, 491
identify points, 29, 41, 486, 491–2
interaction plot, 124
legend, 58, 61, 413, 474, 479
mosaic (multi-way tables), 60, 61, 139
normal probability, 58, 88, **92–5**, 99, 144, 145, 227, 317, 327, 328
panel function, 64, 483
quantile-quantile (QQ), 397
shaded regions, 84, 85, 239
simulated data, 88, 93, 94, 159, 215, 281, 327, 328
size of points & text, 27, 28, 33, 451
stem-&-leaf, 46
strip plot, 52, 124, 483
transformation of scales, 40, 51, 58, 67, 160–1, 168, 175, 188, 214, 245, 250–1, 279, 292, 353
posterior density or probability, *see* Bayesian methods
prediction
predicted values
GLM, scale of linear predictor, 254, 263, 266
GLM, scale of response, 254, 333
linear model (lm), 79–80, 142, 145–6, 149, 150–1, 156, 157, 167, 172, 177, 182, 187–8, 192–3, 208, 214, 217
prediction interval (new y-value), 150
predictive accuracy, 152–3, 493–4
GLM, 255
linear model (lm), 153–8, 186–7, 189, 230
multi-level model, 303, 309, 319, 328
see also discriminant; survival analysis (survival estimate); time series; tree-based
principal components analysis (PCA), *see* ordination
printing, 20, 21, 366, 459, 463, 472, 477
digits, 36, 316, 408
prior density or probability, *see* Bayesian methods
propensity score, 410, 414–17, 419–23, 425, 426
use as regression covariate, 419–25

questionnaires & surveys, 71
quit session, *see* R session

R session
image file, 5, 430, 431, 466, 469
attachment of image files, 430–1
quit, 5, 430
search list, 17, 430, 464
database, 430
working directory, 4, 8, 33, 431, 433, 436, 466, 467
change, 431
workspace, 3–5, 8–9, 17, 21, 33, 35, 86, 428, 430–1, 433, 463, 465–6, 469, 471
image file, 3–5, 33, 428, 430, 431, 466, 469
management, 33, 35, 430, 431
random
coefficients, *see* multi-level model
numbers, 22, **86–9**, 93, 99, 155
seed, **86**, 93
sample, 71, 87, **90–1**, 99, **102**, 117, 137, 153, 186, 307, 353, 369, 420, 462, *see also* sampling
of permutations, 90, 128, 129, 199, 396, 470
regression
(Note the separate entries: regression, generalized additive model; regression, generalized linear model; regression, linear model)
see also discriminant; multi-level model; survival analysis; time series; tree-based
AIC, BIC, *see* information criteria
bootstrap, *see* resampling methods
C_p, *see* information criteria
cross-validation
see resampling methods
extrapolation, 230
fitted values, *see* prediction
functional relationship, 169
loglinear, 210, 257, 272
non-linear model, 210–13, 215, 217
observational data, 69–70, 183, 199, 212–13, 346, 385, 414, 426
normal probability plots, *see* plot, normal probability
ordinal logistic model, 268–72

regression (*cont.*)
resistant, 149, 183, 184, 191–4, 212, 236, 295, 296
robust, 147, 149, 167, 212, 241
spline smoothers
see regression, generalized additive model
strategies for fitting models, 189–91
regression, generalized additive model (GAM), 235, 240, 280
smooth terms, 232, 235, 236, 417
regression, generalized linear model (GLM), 210, 240, 244–68, 280–2, 332, 333
see also regression, linear model
coefficients
see also coefficients
Hauck-Donner effect, 254, 263
SEs & Wald ($z-$) statistics, 265, 267, 277–8
design, 268
deviance, 246, 248, 253, 254, 257, 259, 260, 262, 264–7, 270, 332, 424
dispersion, 260, 262, 264–7
family, 248, 253, 255, 258–60, 262, 264–6, 280
binomial, 248, 253, 255, 258–60, 262, 264–6, 280
poisson, 248, 253, 255, 258–60, 262, 264–6, 280
quasibinomial, 248, 253, 255, 258–60, 262, 264–6, 280
quasipoisson, 248, 253, 255, 258–60, 262, 264–6, 280
leverage, *see* leverage
link function, 245, 254, 255, 257, 258, 262, 266–8, 280
logistic regression, 244–6, 248, 253–7, 269, 270, 272, 279, 280, 387
predicted values, *see* prediction
residuals, *see* residuals
regression, linear model (lm), 59, 142–235, 246, 258, 263, 268, 280, 318
anova table, 226
assumptions, 183–4
bootstrap, *see* resampling methods
check for linearity, 59, 93, 152, 179, 183, 190, 193, 228
coefficients, *see* coefficients
Cook's distance, 148–9
cross-validation, *see* regression
design, 151–2
diagnostics & diagnostic plots, *see* diagnostics & diagnostic plots
errors in x, 195, 203, 208, 212
attenuation of coefficient, 195, 203, 204, 205
simulation, 204, 206
transfer of effect between variables, 206
fitted values, *see* prediction
hat matrix, 184
heterogeneity of variance, 146, 189, 210
influence, 147–9, 173, 183, 185, 419
leverage, *see* leverage
linearity between explanatory variables, 189–90
model formula, 80, 143
model matrix, 163–4, 167, 172, 184–5, 218, 201, 217–18, 220, 225, 228, 231, 234, 242, 243, 445
one term; several columns, 224, 226–7, 228–9, 231–2, 235–6, 238
model non-linear response, 228–35
multicollinearity, 199, 203, 251
remedies, 203
variance inflation factor (VIF), 201–3, 251
omission of intercept, 172
on principal component scores, *see* ordination, regression on PCA scores
on propensity scores, *see* propensity scores, use as regression covariates
outliers, *see* outliers
polynomial, 167, 194, 195, 228, 229, 231, 232, 235, 241–3, 298
predicted values, *see* prediction
R^2 & adjusted R^2, 147, 186
regression splines, 231–5
residual degrees of freedom, 121–2, 147, 150, 171, 223, 226
residuals, *see* residuals
robust & resistant methods, *see* regression
shrinkage methods, 198
straight line model, 228, 231
strategies for fitting models, *see* regression
term plots, 224, 226, 228, 230, 231, 236, 240
terms, 59, 193, 224, 226, 228, 231, 232, 235, 238, 240
variable selection, *see* variable selection
variance inflation factor (VIF), *see* multicollinearity
x on y *vs* y on x, 217, 224, 228, 230, 231, 234
replication, 140, 267
resampling methods
bootstrap, 128, 130–2, 140, 155–8, 189, 215, 351, 362, 369, 373, 375, 376, 382, 383, 416, 420
cross-validation, 153–5, 157–8, 187, 189, 198, 215, 236, 255, 256, 356, 361–3, 367, 374, 385, 387–9, 399, 401–4, 406, 420
validity of error estimate, 158
permutation, 128, 129, 396, 397
simulation, *see* simulation
residuals,
GAM, 239, 243
GLM, 254, 260, 266, 267, 278, 279
deviance, 266
Pearson, 266
working, 266
linear model, **80–1**, 143–6, 147, 148–51, 155, 164, 167–8, 173–4, 176, 178, 179, 183–5, 188, 191–5, 211–12, 215, 217–19, 227, 232, 233, 236, 242

loess smooths, 237
multi-level models, *see* multi-level model
time series, *see* time series
rounding, 34, 92, 344

sampling
see also random sample
cluster sampling, 91
with replacement, 90, 155, 382, *see also* resampling methods, bootstrap
search list, *see* R session
selection bias, 396–8, 407
session, *see* R session, 5, 430
simulation, 86–9, 91, 99–101, 134, 141, 158, 165, 197, 206, 207, 290, 291
smoother
see also regression, generalized additive model, spline
loess, 44, 80, 146, 168, 236, 237, 278, 359–61, 487
lowess, 50, 79, 96, 145–6, 168, 192, 236–7, 242, 279, 292–3, 295–6, 317
spline, 231–7, 239–42, 347, 418, 420, 489
knots, 231–3, 235, 242
regression spline, *see* regression, linear model
standard deviation, 22, 47, 65–7, *passim*
vs inter-quartile range, 65, 66
degrees of freedom, 66, 67, 76, 111, 133, 137, 143, 171
maximum likelihood estimate, 133, 141
pooled, 67, 76, 104, 110, 111, 141
standard error (SE), 102, 105, 109–11, 113, 130, 156, 185, 263
binomial distribution
proportions, 112, 267, 281, 425
difference of means (SED), 66, 103, 110, 111, 124, 133, 137, 141, 143
of mean (SEM), 102, 103, 105, 111, 130
of median, 102, 104, 130, 131, 143, 219, 425
survival analysis, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272–8, 280, 282, 352
censoring, 273, 275, 276, 279
non-informative, 275, 276, 278
right, 282
Cox proportional hazard, 277, 278, 280, 282
data collection, 273
frailty model (multiple levels of variation), 280
hazard rate, 275–8, 282
confidence interval, 278
residuals
martingale, 278, 279
Schoenfeld, 278
survival estimate, 276–7, 372
tree-based, 352, 372, 374

table
margins, 96, 97
of frequencies, 20, 21, 61–3
adding across tables (Simpson's paradox), 62, 96–7, 140
table formula, 62, 63
test
2-way table
chi-square statistic, 114, 116–18
Fisher exact, 116
residuals from row/column independence model, 118
3-way table
Mantel-Haenzel, 139
Woolf, 139
correlation, 113
homogeneity of variance (Fligner-Killeen), 260
likelihood ratio, 265, 277, 278
nonparametric, 999
permutation
one- & two-sample, 128–9
proportion(s), 112–13
sequential correlation, 172, 298
t-statistic
one- & two- sample, 109, 110
two-sample, unequal variances, 110–11
type III, 446
Wald, 265, 267, 277–8
white noise (Ljung-Box), 297–8, 301
time series, 10, 14, 47, 48, 53, 55, 158, 165, 283–90, 292–4, 296, 298–303
ARCH or GARCH model, 298–300
ARMA or ARIMA model, 287–98
automated selection, 289–91, 293, 294, 296–8
autocorrelation, 165, 284–90, 293–5, 298, 302
autoregressive (AR) model, 165, 283, 285–7, 298–300, 301
Ljung-Box test (white noise), 297–8, 301
moving average (MA) model, 283, 287–90
partial autocorrelation function, 284–9, 293, 294
predicted values, 290
residuals, 289, 293–8, 301
transformation, 40, 51, 58, 59, 67, 73, **160–1**, 168, 175, 189–90, 196, 210, 279, 286, 292, 354, 416–19, 424
Box-Cox, 161
count data
angular, 279
complementary log-log, 279
logit, 245, 254, 279, 424
probit, 279
square root, 160, 250, 279
cube root, 160, 292–3, 296
logarithmic, 51, 55, 58, 59, 160, 161, 175, 188, 190, 210, 214, 249, 250, 286, 354, 416, 418
logit, 245, 254, 279, 424
power, 160, 161, 196
tree-based, random forests (randomForest), 369, 371, 375, 420, 422

tree-based, decision trees (rpart)
  classification, 351, 352, 354, 358–60, 362, 364, 366, 368, 369, 372–4
  cost-complexity (CP), 361–2
  error or impurity measures, 359
  information on each split, 354, 357, 359, 366, 374
  output, 354, 357, 359, 366, 374
  predicted values, 360, 366
  predictive accuracy, 355, 356, 361
  pruning, 362, 372
  splitting criterion, 356, 359, 362, 372
    1 SE rule, 365–6
  table of accuracies, 355, 364, 366
  tree diagram, 354, 356, 360, 362, 363, 370
  regression with a continuous outcome, 348, 357, 359–61

variability
  heterogeneity, *see* model, homogeneity of variance
variable selection
  discriminant analysis, 398, 401
    unbiased accuracy estimates, 398–404
    simulation, 398–404
  linear models, 196–9, 212, 411
    realistic SEs, 197–9
    simulation, 197–9
  pruning decision trees as variable selection, 372
  survival analysis, 273
vector
  atomic, 452, 454
  character, 6, 13, 16, 19, 21, 36, 434, 435, 440–2, 444, 448, 452, 454, 455, 469, 474, 475
    number of characters, 6, 440
    splitting, 436, 440
  complex, 10, 434, 448, 452
  concatenation, 10
  logical, 10, 11, 19, 20, 24, 36, 434, 443, 446, 448, 452, 455
  numeric, 10, 13, 15, 16, 19, 30, 36, 217, 434, 436, 444, 448, 451, 452, 454, 455
  patterned, 11, 12
  recursive (list), 457
  subset, 11–12, 14, 23, 371, 393, 452

working directory, *see* R session
workspace, *see* R session

Index of authors

Agresti, 139
Aitchison, 199
Aldrich, 97
Ambroise and McLachlan, 392, 406
Andersen, 74

Barnett, 91
Bartholemew, 345
Bates, 305
Bates and DebRoy, 461
Bates and Watts, 213
Belson, 351
Berk, 203, 373, 406
Bickel *et al.*, 96
Blake and Merz, 353
Bland and Altman, 209, 280
Bolker, 136, 212
Boot and Maindonald, 242
Box and Cox, 161
Braun and Murdoch, 468
Breiman, 98
Brockwell and Davis, 300
Burns *et al.*, 56
Bussolari, 336

Canty, 130
Carroll, 204
Carroll *et al.*, 206
Chalmers and Altman, 348
Chambers, 37, 461, 468
Chanter, 74
Charig, 60
Chatfield, 74, 300
Christie, 242
Chu *et al.*, 141
Clarke, 98
Cleveland, 33 37, 74, 237
Clutton-Brock *et al.*, 242
Cochran and Cox, 346
Cohen, 209
Collett, 280
Cook and Swayne, 406, 492
Cook and Weisberg, 184, 190, 212
Cox, 136, 348
Cox and Cox, 385
Cox and Reid, 136, 348
Cox and Wermuth, 68, 213

Dalgaard, 36
Daniels *et al.*, 345
Davison and Hinkley, 132
Dehejia and Wahba, 414, 417
Diggle, 300
Diggle *et al.*, 331, 335, 348
Dobson, 280
Donner and Klar, 91

Edwards, 213
Efron and Tibshirani, 131, 132
Efron *et al.*, 199
Eubank, 240
Ezzet and Whitehead, 269

Fan and Gijbels, 237
Faraway, 212, 240, 280
Farmer, 62, 63
Finney, 280
Fisher, 348
Fox, 36, 186, 212

Gardner *et al.*, 74
Gaver *et al.*, 347, 348
Gelman and Hill, 348
Gelman *et al.*, 136
Gentleman, 37, 461, 468
Gentleman and Lang, 467
Gentleman *et al.*, 407
Gigerenzer, 135, 136
Gigerenzer *et al.*, xix, 136
Gihr and Pilleri, 162
Goldstein, 348
Golub *et al.*, 392
Gordon, 70, 385
Gordon *et al.*, 208
Gourieroux, 300

Grasso *et al.*, 123
Guy, 47

Hales *et al.*, 65
Hall, 98, 237, 390
Harker and Maindonald, 232
Harlow *et al.*, 136
Harrell, 24, 196, 197, 203, 212, 280
Hastie *et al.*, 196, 198, 203, 212, 330, 373, 406
Hauck and Donner, 268
Hoaglin, 43
Hobson, 116
Hunter, 249
Hyndman and Khandakar, 300
Hyndman *et al.*, 300

Ihaka and Gentleman, 468
Izenman, 385

Johnson, 96, 98

King, 386
King and Maindonald, 386
Krantz, 136
Krzanowski, 406

Lalonde, 59, 115, 414, 417
Latter, 52, 70
Leavitt and Dubner, 212, 344
Leisch, 468
Liaw and Wiener, 352
Lim and Loh, 373
Linacre, 238
Linacre and Geerts, 238
Linde *et al.*, 138
Lindenmayer *et al.*, 44, 390
Lumley, 461

Maindonald, 37, 74, 136, 158, 213, 319, 392, 404
Maindonald and Burden, 392, 404
Maindonald and Cox, 74
Maindonald *et al.*, 266, 280
Manly, 406
Marland *et al.*, 3
McCullagh and Nelder, 266, 267, 280
McLellan *et al.*, 258
McLeod, 228
Meyer, 406
Meyer and Finney, 62
Miller, 111, 116, 136, 285
Mitchell, 47
Muenchen, 468
Murrell, 33, 37, 492
Myers, 203, 213

Nadel and Bussolari, 336
Nelder, 74
Newton, 70
Nicholls, 136
Nicholls *et al.*, 291

Ord *et al.*, 300

Payne *et al.*, 346, 348
Pinheiro and Bates, 335, 348
Pollard *et al.*, 396

R Development Core Team, 468
Ramsay and Silverman, 347
Rao and Wu, 196
Ripley, 362, 373, 406
Rosenbaum, 212, 426
Rosenbaum and Rubin, 414, 426

Sammon, 384
Sarkar, 33, 492
Schatzkin *et al.*, 204, 205
Schmidt-Nielsen, 162
Senn, 47, 136
Shanklin, 242
Sharp *et al.*, 208
Shumway and Stoffer, 300
Simpson, 97
Smyth, 123
Snelgar *et al.*, 319
Snijders and Bosker, 348
Spiegelhalter *et al.*, 330
Sprent, 163
Steel *et al.*, 126
Stewart *et al.*, 78
Stidd, 292
Stiell *et al.*, 281
Stocks, 47
Streiner and Norman, 72, 411, 426

Talbot, 308, 348
Therneau and Atkinson, 351, 372, 373
Therneau and Grambsch, 280
Tippett, 52
Trapletti and Hornik, 299
Tufte, 37
Tukey, 113
Turner *et al.*, 347, 348

Vaida and Blanchard, 330
Venables, 212, 446
Venables and Ripley, 9, 36, 196, 212, 237, 240, 280, 300, 348, 351, 372, 373, 385, 406

Würtz, 300
Wainer, 37
Weisberg, 212
Welch, 111
Wickham, 492

Wilkinson, 487
Wilkinson and Task Force on Statistical Inference, 37, 74, 136
Williams, 208
Williams *et al.*, 348
Wonnacott and Wonnacott, 136
Wood, 240, 280
Xie and Cheng, 91
Young and Smith, 136
Zeger *et al.*, 207, 208
Zhang and Singer, 374
Zhu *et al.*, 392