

Intermediate Statistical Methods for Data Scientists

Anthony Almudevar, PhD

Department of Biostatistics and Computational Biology,
University of Rochester

Copyright 2019

Contents

1	Introduction	9
1.1	Course Material	10
1.2	A Reference Library of Statistical Methodology	12
I	Statistical Methodology	15
2	ANOVA	16
2.1	Methodology	17
2.2	ANOVA Table	19
2.3	Bonferroni Correction for Multiple Comparisons	21
2.4	<i>Post hoc</i> Analysis in ANOVA	22
2.5	Nonparametric ANOVA	23
2.6	Assumptions	24
2.7	ANOVA in R	24
2.8	Equality of Variances	27
2.9	The Kruskal-Wallis Test for Nonparametric ANOVA	28
2.10	Postscript	30
3	Linear Regression - Introduction	32
3.1	Residuals	35
3.2	ANOVA approach	37
3.3	The Relationship Between Linear Regression and Correlation	38
3.4	The Derivation of the Least Squares Coefficients	39
3.5	Assumptions	40
4	Linear Regression - Inference	41
4.1	Inference of Regression Parameters	41
4.1.1	Confidence intervals for simple linear regression	43
4.1.2	Hypothesis tests for simple linear regression	43
4.1.3	Prediction intervals for simple linear regression	43
4.1.4	Calculations based on sums of squares	44
4.2	Multiple Linear Regression	47
4.2.1	ANOVA tables for multiple linear regression	49
4.2.2	Full and reduced models	49

4.2.3 Example	50
5 Linear Regression - Modeling in R	53
5.1 Statistical Models	53
5.2 The Formula Object in R	53
5.3 Linear Regression in R	57
5.4 ANOVA and Linear Regression	60
5.5 Residuals and <code>lm()</code>	62
5.6 Interaction Terms	63
5.7 Polynomial Regression	68
6 Linear Regression - Formulation Using Matrix Algebra	71
6.1 Regression Coefficients β	71
6.2 Linear Combinations of β	72
6.3 Fitted Values \hat{y}	72
6.4 Residuals e	72
7 Least Squares Regression and Vector Spaces	74
7.1 Vector Spaces and Prediction Spaces	74
7.2 Linear Transformations of Prediction Spaces	78
7.2.1 Evaluation of transformation matrix A	79
7.3 General Transformation Equivalence	79
7.4 Orthogonalization of Predictor Matrices	81
7.4.1 Orthogonalization of simple linear regression	81
7.4.2 QR decomposition	82
7.4.3 Orthogonalization of multiple linear regression	82
8 Linear Regression Diagnostics - Outliers, Influential Observations and Collinearity	84
8.1 Leverage	84
8.2 Cook's Distance	85
8.3 Studentized Residuals	85
8.4 Influence Measures	86
8.5 Covariance Ratio	86
8.6 Collinearity	87
8.7 Postscript	88
9 Maximum Likelihood Estimation	89
9.1 Fisher Information	89
9.2 Inference Methods	91
9.3 The Likelihood Ratio Test and Deviance	92
9.4 Postscript	93

10 Logistic Regression	94
10.1 The Odds Ratio in Logistic Regression	95
10.2 Likelihood Method for Logistic Regression	95
10.3 Postscript	98
11 Survival Analysis	99
11.1 Memoryless Distributions	99
11.2 The Failure Rate	100
11.3 Estimation of the Survival Function	103
11.3.1 Censoring	103
11.3.2 Kaplan-Meier estimate of the survival function	104
11.3.3 Cox proportional hazards regression	107
11.4 Postscript	111
12 Bayesian Inference	113
12.1 The Bayes Estimator	113
12.2 Bayesian Inference for the Binomial Distribution	115
12.2.1 The gamma and beta functions	115
12.2.2 The beta distribution	115
12.2.3 Posterior distributions	116
12.3 Postscript	117
13 Simulation Methods	118
13.1 Permutation Test	118
13.2 The Bootstrap Procedure	121
13.3 General Principles of Computer Simulation	123
13.3.1 Pseudorandom number generation	123
13.3.2 Linear congruential generators	124
13.3.3 Uniform random number generation	126
13.3.4 The inverse transformation method	127
13.3.5 Simulation of discrete random variables	129
13.3.6 Computer simulation and reproducibility in R	130
13.3.7 Simulating dependent random variables in R	130
13.4 Postscript	130
14 Markov Chains, MCMC and Computational Bayesian Methods	133
14.1 Markov Chains	133
14.1.1 Maze example	136
14.1.2 Distributional properties of Markov chains	138
14.1.3 Balance equations and steady states	141
14.2 The Hastings-Metropolis algorithm	143
14.3 Simulated annealing	144
14.4 Postscript	145

II Supervised and Unsupervised Learning	147
15 Machine Learning and Statistical Learning - General Concepts	148
15.1 Some Notational Conventions	149
15.2 Structure of Data	149
15.2.1 Features	149
15.2.2 Response	150
15.3 Feature Distances	150
15.3.1 Metrics	150
15.3.2 L^p norms	151
15.3.3 Distance functions	152
15.4 Supervised and Unsupervised Learning	153
15.5 Loss and Risk	153
15.6 Cross-Validation	156
15.7 Bias and Variance	157
15.8 Model Selection for Classifiers	158
15.9 Postscript	158
16 Bayes Theorem and Classification	160
16.1 Odds	162
16.2 The Bayesian Model	163
16.3 The Fallacy of the Transposed Conditional	165
16.4 Diagnostic Testing - Basic Definitions	165
16.4.1 Diagnostic tests and contingency tables	166
16.4.2 The use of odds in the evaluation of diagnostic tests	167
16.5 The Odds Ratio	169
16.6 Bayes Classifiers	170
16.6.1 Prior probabilities	171
16.6.2 Naive Bayes classifiers	171
16.7 K Nearest Neighbor (KNN) Classifiers and Regression	172
16.8 Linear and Quadratic Discriminant Analysis	172
16.8.1 Estimation for LDA/QDA	174
16.9 Classification and the Receiver Operator Characteristic (ROC) Curve	174
16.9.1 Classifiers based on a numerical risk score	175
16.9.2 ROC curves	180
16.10 Artificial Neural Networks	183
16.11 Postscript	183
17 Unsupervised Learning	185
17.1 Hierarchical Clustering	186
17.2 K-Means Cluster Analysis	187
17.3 Principal Components Analysis	188
17.3.1 Calculation of principal components	189
17.3.2 Principal components and spectral decomposition	190
17.4 Postscript	194

18 Score Based Model Selection	195
18.1 AIC and BIC for Multiple Linear Regression	196
18.1.1 Model selection algorithms based on predictor subsets	196
18.2 Shrinkage Methods	198
18.3 Postscript	199
19 Basis Functions and Predictor Spaces	201
19.1 Transformation Equivalence of Basis Functions	202
19.2 Polynomial Regression	206
19.2.1 Polynomial regression in R	207
19.2.2 Demonstration of R function poly()	208
19.3 Splines	212
19.3.1 Degrees of freedom of a spline	215
19.3.2 Basis function representation of a spline	217
19.3.3 Splines of degree d	223
19.3.4 Natural cubic splines and smoothing splines	223
19.3.5 B-splines	225
19.3.6 Using B-splines in R	227
19.4 Postscript	229
20 Bayesian Networks	231
20.1 Fitting BNs	233
20.2 Equivalence Classes	233
20.3 Postscript	233
III Practice Problems	234
21 Practice Problems - ANOVA	235
21.1 Exercises	235
21.2 Data Analysis	238
22 Practice Problems - Linear Regression	243
22.1 Exercises	243
22.2 Data Analysis	247
22.3 Theoretical Complements	263
23 Practice Problems - Logistic Regression	271
23.1 Exercises	271
23.2 Data Analysis	279
24 Practice Problems - Survival Analysis	292
24.1 Exercises	292
24.2 Data Analysis	295
24.3 Theoretical Complements	300

25 Practice Problems - Bayesian Inference	305
25.1 Exercises	305
25.2 Data Analysis	307
25.3 Theoretical Complements	313
26 Practice Problems - Simulation Methods	320
26.1 Exercises	320
27 Practice Problems - Markov Chains, MCMC and Computational Bayesian Methods	328
27.1 Exercises	328
27.2 Simulation Projects	355
28 Practice Problems - Classification	364
28.1 Exercises	364
28.2 Data Analysis	371
28.3 Theoretical Complements	396
29 Practice Problems - Unsupervised Learning	407
29.1 Exercises	407
29.2 Data Analysis	415
29.3 Theoretical Complements	420
30 Practice Problems - Model Selection and Splines	430
30.1 Exercises	430
30.2 Data Analysis	445
30.3 Theoretical Complements	467
31 Practice Problems - Bayesian Networks	479
31.1 Exercises	479
31.2 Data Analysis	483
Appendices	488
A Linear Algebra	489
A.1 Numbers and Sets	489
A.2 Fields and Vector Spaces	490
A.3 Equivalence Relationships	490
A.4 Matrices	491
A.5 Eigenvalues and Spectral Decomposition	493
A.5.1 Right and left eigenvectors	494
A.6 Symmetric, Hermitian and Positive Definite Matrices	495
B Multivariate Distributions	497
B.1 Matrix Algebra and Multivariate Distributions	498
B.2 Multivariate Normal Distribution	499

C An R Tutorial	500
C.1 Mathematical Operations on Scalars and Vectors	500
C.1.1 Vectors in R	502
C.1.2 Global options	505
C.1.3 Modes (or types)	507
C.1.4 Index referencing	511
C.1.5 More vector operations	512
C.1.6 Pattern matching	513
C.1.7 Managing objects	514
C.2 Data Structures in R	515
C.2.1 Matrices	515
C.2.2 More on index subsets	519
C.2.3 Lists	521
C.2.4 Data frames	523
C.2.5 Factors	524
C.2.6 Arrays	524
C.3 Labels for Data Structures	525
C.3.1 Vector labels	526
C.3.2 Matrix and array labels	527
C.3.3 Labels for lists and data frames	528
C.4 Programming and Functions	530
C.4.1 Program control	530
C.4.2 User defined functions	532
C.4.3 Functions and environments	533
C.4.4 User defined binary operators	534
C.5 Vectorized Calculations	534
C.6 File Input and Output	535
C.7 Packages	535
C.8 Objects and Classes in R	538
C.8.1 Object modes	538
C.8.2 Object classes	539
C.8.3 Generic functions	540
C.8.4 User defined methods	542
C.8.5 S4 (formal) classes	543
C.8.6 Testing and coercion of object types	544
C.9 Random Variables in R	544
D Distribution Tables	546
Bibliography	562

Chapter 1

Introduction

Preamble

The material presented here is intended for a single term course in what might be called *Intermediate Statistics*, or a second course in statistical methodology. It is assumed that the reader is familiar with the definition of confidence intervals and hypothesis tests, basic probability and distribution theory, as well as a good sense of the role played in statistical inference by the t -, F - and χ^2 distributions. A course in calculus is also assumed as a prerequisite, and a good knowledge of linear algebra is highly recommended. Appendix A contains a review of basic linear algebra theory, and Appendix B contains a review of multivariate distributions. A tutorial in R programming is offered in Appendix C, while statistical probability tables are given in Appendix D. The course begins with a comprehensive introduction to ANOVA and linear regression. It is anticipated that a student will be familiar with these topics, but a thorough introduction is included.

Data science and statistical methodology

The course is intended for the training of *data science* professionals. It may be a good idea at this point to discuss a few general terms associated with this field. The term *machine learning* refers to the computerized automation of decision making, classification or quantitative prediction. The term “learning” signifies the reliance of the underlying algorithms on data, or the ability to use data (or prior experience) to optimize the performance of these tasks. Therefore, machine learning is the application of methods from various fields, including control theory, statistical inference and optimization, to the development of computer algorithms specialized to the exploitation of data. Then *statistical learning* refers to the application of statistical methodology to these tasks, especially as concerns classification or quantitative prediction.

Another term which enters this field is *big data*. This is either simple in form, but high-dimensional (bioinformatic data, image data), or complex, in the sense that a single data set contains many distinct forms of information (computerized marketing data, often). It is this aspect that usually drives the need to reduce computational burden.

There is no doubt that the advent of *big data* in many forms has significantly changed the emphasis placed on statistical methodology both in research and in college level instruction. But it has not significantly changed the statistical methodological canon itself, and for very good reasons.

Most students are introduced to statistical modeling via simple, then multiple, least squares

regression (LSR). Without reference to any particular method, the model looks something like this

$$y = g(x_1, \dots, x_p) + \epsilon. \quad (1.1)$$

Here, y is the *response variable*. The object is to build a predictor for y . This is to be done using p *predictor variables* x_1, \dots, x_p . Finally, ϵ represents the *error variable*. The meaning of this is that we do not expect that the target of prediction y can be predicted exactly, even if we have perfect knowledge of its behavior. But there are more than one possible reasons for this. The first is that we cannot capture in p predictors all information about y . The second is there is an inescapable random component of y which is impervious to prediction (as in a casino, for example). Probably, both factors are in play in any given application. Whatever the case, the essence of Equation (1.1) is that it decomposes response y into predictable and unpredictable components.

Clearly, any body of methodology which deals with a prediction problem generated by (1.1) has two objectives. The first is to build the best possible predictor, and the second is to estimate the limits on the accuracy of any predictor (in other words, to recognize the existence of ϵ , and to say something about its size). This, in fact, is one of the central problems of classical statistical methodology, and a true mastery of prediction methodology is not possible without an understanding of the underlying theory.

No reference has yet been made to LSR models. This is because that class of models is merely a special case of Equation (1.1), obtained by setting

$$\begin{aligned} g(x_1, \dots, x_p) &= \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p, \text{ and} \\ \epsilon &\sim N(0, \sigma^2). \end{aligned} \quad (1.2)$$

As we will see, there are other possible choices for both $g(x_1, \dots, x_p)$ and ϵ . The range of models expands further if we allow a certain flexibility regarding what type of object the response variable y is. The point to be made is that a theory of statistical learning would be quite limited if it confined itself to components of the type given in Equations (1.2).

Of course, this imperative has long been recognized in classical statistical methodology, and specialized models are available for a large variety of response types (discrete count data, binary data, survival times, and so on). So we need to ask at this point what the relationship is between classical statistical methodology and “big data”, “data science”, “machine learning” or “statistical learning”. The point of view of this author is that this relationship is essentially the same type of relationship that exists between the fields of statistics and statistical genetics. Or statistics and quality control. Or statistics and psychometrics. All of these disciplines place considerable emphasis on the transformation of data into information or knowledge. Statistical theory provides a language and framework, and a supporting probability theory, of sufficient generality to permit such analyses of data which are both correct and efficient (or, sometimes, provably optimal). Whether applied to big or small data, this relationship is essentially the same. The role of these notes, therefore, is to identify and explain those aspects of the statistical canon most relevant to the applications likely to be encountered by the data scientist.

1.1 Course Material

Given the purpose of these notes, the training of *data science* professionals, there is a certain balance which needs to be achieved, and to this end there are two objectives. The first is to provide

instruction in a sufficiently comprehensive range of methodologies, and the second is to train the reader in the application of these methods to data analysis applications likely to resemble those that will eventually be encountered.

Using R

All computing is done in the R statistical computing environment. This is available at www.r-project.org as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. Precompiled binary distributions are also available for Windows, MAC OS, and Linux OS (versions of archived precompiled distributions vary). An excellent introductory manual *An Introduction to R* is available at www.r-project.org (follow the Manuals link). These notes also contain a tutorial in R (Appendix C).

Course software

There is a series of R demonstration software files that accompany these notes, and which are intended to provide demonstrations and further instruction in the covered methodologies. They are thoroughly commented with this in mind, and may be used by an instructor during a lecture or tutorial. Another objective is to provide the data science professional with a software library for use in their practice. These files are listed by topic in Table 1. Two additional software files cover the creation of R packages, and parallel computing.

Practice problems

Part III of these notes (Chapters 21-31) contains an extensive list of practice problems created by the author. All have solutions, which are given in blue text immediately following the questions. Some include extensive R code.

Each of these chapters is devoted to a specific topic, or set of topics. However, there is not a one-to-one mapping of these chapters to those of the main body of the notes. This is partly due to dependence of some topics on previously covered topics. A good example of this is *principal components analysis* (Section 17.3), which can be a useful component of a larger analysis (see, for example, Problem 30.17). Also, Chapter 30 combines a number of separate topics which are perhaps more usefully examined together than separately.

Most practice problem chapters are divided into sections. Practice problems in an **Exercises** section emphasize mastery of the technical details of a method, and tend to be shorter in length. Most do not rely on data sets, but often begin with the computer output of various R modeling functions. Practice problems in a **Data Analysis** section are usually more extensive, and make use of a data set. Most make use of data from one of two R packages, MASS (which comes with the R distribution) and ISLR (which accompanies James *et al.* (2013), and is freely available (along with the textbook itself) at www-bcf.usc.edu/~gareth/ISL/). Problem 28.9 serves as an introduction to a quite rich source of bioinformatic data, the *Gene Expression Omnibus* at <http://www.ncbi.nlm.nih.gov/geo> maintained by the National Center for Biotechnology Information of the National Institutes of Health.

Finally, some chapters in Part III have a **Theoretical Complements** section. These are practice problems intended to address, or extend, the more theoretical aspects of the methodology.

Postscripts

Most chapters end with a **Postscript**. The postscripts have several purposes. Where appropriate, they point the reader to specific practice problems and demonstration software files, as well as additional topics and supporting literature. Specific R packages and functions are cited. They usually include a brief summary of the main topics covered in the chapter.

When needed, important subjects which cannot for practical reasons be covered in detail here are singled out, and the reader is directed to suitable sources.

Chapters 9 (Maximum Likelihood Estimation), 12 (Bayesian Inference) and 15 (Machine Learning and Statistical Learning - General Concepts) are more foundational in character. Further reading is more appropriately done in the context of a general theory of statistical inference (Section 1.2).

Linear regression is covered in Chapters 3 - 8, for which a single postscript is offered in the final chapter.

1.2 A Reference Library of Statistical Methodology

These notes aim to prepare the data scientist for actual data analysis in a professional setting, and the hope is that the background presented here will suffice for many, perhaps even most, applications. The methodology will be comprehensive enough for this, and the practice problems should prepare the analyst for the many subtle issues that will arise. To pick just one example, if the units of some variable are changed from pounds to grams, will the conclusion change? This issue is a surprisingly complex one, and is explored in Problem 28.14. Or another: suppose some learning algorithm makes use of the simple linear regression model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, \dots, n,$$

to estimate a crucial parameter, but that we may choose the predictor values x_1, \dots, x_n . How do we select these values to minimize the estimation error? This question has a quite definitive answer (Problem 22.12).

However, there is clearly a great deal of theory and method which cannot be compressed into a resource like this, so we will give a brief literature review to guide further reading.

There are a number of highly recommended textbooks on intermediate to advanced statistical theory. Two more accessible volumes would be Hogg *et al.* (2018) (undergraduate level) or Casella and Berger (2002) (graduate level). Bickel and Doksum (2015a) and Bickel and Doksum (2015b) are more advanced mathematically, but also offers much more comprehensive treatments of statistical computing methods, from a theoretical point of view.

For more advanced treatments of the theory of statistical inference see Cox and Hinkley (1979), Welsh (2011) or the volumes Lehmann and Casella (1998) and Lehmann and Romano (2006).

A number of reference books for mathematical background can be mentioned here. First, Horn and Johnson (1985) is an excellent reference for matrix analysis. Good references for analysis are Spivak (1967) (introductory) or Royden (1968) (intermediate). Ross (2014) and Ross (1996) provide excellent introductions to probability theory which do not rely on measure theory. The reader who wishes to study measure theory on the context of probability theory can consult Ash and Dolacutuseans-Dade (2000), Billingsley (1995), Durrett (2010), Feller (1968) and Feller (1971). An interesting approach to probability theory can be found in Whittle (2000).

Of textbooks familiar to this author, James *et al.* (2013) is perhaps closest in scope to these notes, and can be used in tandem. As it happens, James *et al.* (2013) serves as a more accessible version of Friedman *et al.* (2001). Both volumes can be highly recommended to the data scientist.

If there is one more text which a data scientist should own, it is Venables and Ripley (2013). The title of this book is “Modern applied statistics with S-PLUS”. It must be pointed out that S-PLUS is nothing more than the proprietary predecessor of R, so that the two statistical computing environments are interchangeable, for our purposes. It is a more advanced survey of statistical computing in R or (S-PLUS, if available) than the present lecture notes, but necessarily briefer in explanations, so some experience in R computing is highly recommended to make the most of this book. A considerable amount of supplementary material can be found at www.stats.ox.ac.uk/pub/MASS4/

We may also recommend Hastie (2017) as a text which concentrates on statistical modeling in R at a level closer to the design of software.

It is, of course, difficult to do justice to the literature on R computing. Springer Publishers has an extensive series of monographs on statistical computing in R, under the name “Use R!” www.springer.com/series/6991. Topics in this impressively extensive series include market analysis, psychometrics, numerical ecology or network models. They tend to be quite focused, and of course concentrate on the application of methodology in the R environment.

We finally note that machine learning algorithms are commonly used in computational biology, and many good overviews in this context have been published. Hahne *et al.* (2010) is just one, published in the Spring “Use R!” series. It gives excellent tutorials in advanced statistical methods, and is imbedded throughout by quite extensive R code, which can be downloaded from a dedicated website, and which would likely be quite useful for the practicing data analysts. Much of the science will be somewhat out of date by now, but this does not affect the volume’s utility to the analyst. See <https://master.bioconductor.org/help/publications/books/bioconductor-case-studies/>.

It is difficult to single out a subset of these texts. However, a good reference library for the data scientist (which is important to have) might include Devore (2011) for introductory statistics; Neter *et al.* (1996) for regression models and ANOVA; Ross (2014) for probability theory; and James *et al.* (2013), Venables and Ripley (2013) for intermediate statistical methodology in R. For more advanced statistical theory, Casella and Berger (2002) or possibly Hogg *et al.* (2018) could be recommended. For readers intending to study statistical theory at a graduate level the Feller and Lehmann volumes are especially recommended. Note that it need not be crucial to have the latest edition of a textbook intended for a reference library.

Table 1: Organization of demonstration software files by topic.

ANOVA	ANOVA.R
Linear Regression	REGRESSION-A.R REGRESSION-B.R REGRESSION-C.R REGRESSION-D.R
Bayesian Inference	BAYESIAN-INFERENCE.R
Survival Analysis	SURVIVAL.R
Simulation Methods	SIMULATIONS.R SIMULATED-ANNEALING.R
Computational Bayesian Methods	COMPUTATIONAL-BAYESIAN.R
Machine Learning Foundations	CROSS-VALIDATION.R
Classification, Logistic Regression	CLASSIFICATION-A.R CLASSIFICATION-B.R
Unsupervised Learning	UNSUPERVISED-LEARNING.R
Model Selection	MODEL-SELECTION-AIC.R MODEL-SELECTION-LASSO.R
Nonlinear Models	NONLINEAR-MODELS-POLYNOMIAL.R NONLINEAR-MODELS-SPLINES.R
Bayesian Networks	BAYESIAN-NETWORKS.R
Writing R Packages	PACKAGE-TUTORIAL.R
Parallel Computing in R	PARALLEL-COMPUTING.R

Part I

Statistical Methodology

Chapter 2

ANOVA

A good starting point to understanding the ANOVA model is to reconsider the standard two sample t -test. To briefly review, we are given two independent *iid* samples X_1, \dots, X_{n_1} and Y_1, \dots, Y_{n_2} from normal distributions $N(\mu_1, \sigma^2)$ and $N(\mu_2, \sigma^2)$, respectively. Note that sample sizes n_1, n_2 need not be equal, nor do we expect $\mu_1 = \mu_2$. On the other hand, we assume that both distributions have common variance σ^2 (other procedures are available when this cannot be assumed). Then consider the following hypotheses:

$$\begin{aligned} H_o &: \mu_1 = \mu_2 \\ H_a &: \mu_i \neq \mu_j. \end{aligned}$$

As is well known, the two-sampled t -test may be used to resolve these hypotheses. We let \bar{X}, \bar{Y} be the respective sample means, and let S_X^2, S_Y^2 be the respective sample variances. Since the two population variances are equal, it makes sense to combine the two sample variances into a single *pooled* estimate, in particular:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

We then use the t -statistic:

$$T = \frac{\bar{X} - \bar{Y}}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \quad (2.1)$$

which has a t -distribution with $n_1 + n_2 - 2$ degrees of freedom under the null hypothesis H_o , which can be used to develop a precise rejection region:

$$\text{Reject } H_o \text{ at level } \alpha \text{ if, } |T| > t_{\alpha/2},$$

for an appropriate critical value $t_{\alpha/2}$.

Testing for inequality among more than two means

What if we wish to compare more than two means? We sometimes have situations in which we have k random samples from $k \geq 2$ distinct populations with population means μ_1, \dots, μ_k . Interest

is then in testing the hypothesis

$$\begin{aligned} H_o : \mu_1 &= \mu_2 = \dots = \mu_k \\ H_a : \mu_i &\neq \mu_j \text{ for some } i, j. \end{aligned} \quad (2.2)$$

In other words, are there some differences between the means (H_a) or are they all the same (H_o). The standard statistical procedure for this is the ANOVA method. This covers a wide variety of models, and we discuss only the simplest here (see Section 2.10 for more on this). However, it is also useful to recognize the basic ANOVA model as a generalization of the pooled two-sample t-test for the purpose of comparing $k \geq 2$ samples.

2.1 Methodology

The most frequently used method for testing the hypotheses on Equation (2.2) is referred to as *analysis of variance*, or *ANOVA*. The data then has the following structure:

Pop'n	Pop'n Mean	Sample Size	Sample	Sample Mean	Sum of Squares
1	μ_1	n_1	$y_{11}, y_{12}, \dots, y_{1n_1}$	\bar{y}_1	$\sum_{i=1}^{n_1} (y_{1i} - \bar{y}_1)^2$
2	μ_2	n_2	$y_{21}, y_{22}, \dots, y_{2n_2}$	\bar{y}_2	$\sum_{i=1}^{n_2} (y_{2i} - \bar{y}_2)^2$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
k	μ_k	n_k	$y_{k1}, y_{k2}, \dots, y_{kn_k}$	\bar{y}_k	$\sum_{i=1}^{n_k} (y_{ki} - \bar{y}_k)^2$

The groups may be referred to as *treatments*. Sometimes it is convenient to refer to a treatment as a *factor* or *factor variable*. The observations y_{ij} are then *responses*, and μ_i is a *mean response*. Here, we only have one factor, so the procedure is referred to as *one-way ANOVA*. If the sample sizes n_i are equal, we refer to this as a *balanced design*.

We also have the *total mean*

$$\begin{aligned} \bar{y} &= \frac{\text{sum of all observations}}{n_1 + n_2 + \dots + n_k} \\ &= \frac{n_1\bar{y}_1 + n_2\bar{y}_2 + \dots + n_k\bar{y}_k}{n_1 + n_2 + \dots + n_k}. \end{aligned}$$

In order to develop a test statistic for the hypothesis, we define the *treatment sum of squares*

$$\text{SST} = \sum_{i=1}^k n_i(\bar{y}_i - \bar{y})^2$$

and the *error sum of squares*

$$\text{SSE} = \sum_{i=1}^k \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2.$$

It can be shown that if we define the *total sum of squares* to be

$$\text{SSTO} = \sum_{i=1}^k \sum_{j=1}^{n_i} (y_{ij} - \bar{y})^2,$$

then

$$\text{SSTO} = \text{SST} + \text{SSE}.$$

The test statistic we use is then

$$F_{obs} = \frac{\text{SST}/(k-1)}{\text{SSE}/(n-k)} \quad (2.3)$$

where

$$n = n_1 + n_2 + \dots + n_k.$$

Given the form of SST we can see that if there are large differences among the sample means, F_{obs} will tend to be larger. To reject the null hypothesis we use the observed significance level defined by

$$\alpha_{obs} = P(F_{k-1,n-k} > F_{obs})$$

where F_{ν_1, ν_2} is a random variable with an *F distribution* with ν_1 *numerator degrees of freedom* and ν_2 *denominator degrees of freedom*. Most statistical software packages will calculate this significance level.

Example 2.1. It is left as an exercise for the reader to show that for $k = 2$, the statistic T^2 , where T is used for a two-sampled pooled *t*-test (Equation (2.1)), is equivalent to F_{obs} as defined in Equation (2.3). Therefore, the two-sampled pooled *t*-test is simply a special case of the ANOVA model. \square

Example 2.2. This example is due to Johnson and Bhattacharya (*Statistics: Principles and Methods*, Wiley, 3rd edition).

In an effort to improve the quality of recording tapes, the effects of four kinds of coatings A, B, C and D on reproduction quality are assessed by applying each to a separate sample of tape and measuring the resulting distortion. The results are given in the following table.

Coating	Sample	Sample Mean	Sum of Squares
A	10, 15, 8, 12, 15	$\bar{y}_1 = 12$	$\sum_{i=1}^5 (y_{1i} - \bar{y}_1)^2 = 38$
B	14, 18, 21, 15	$\bar{y}_2 = 17$	$\sum_{i=1}^4 (y_{2i} - \bar{y}_2)^2 = 30$
C	17, 16, 14, 15, 17, 15, 18	$\bar{y}_3 = 16$	$\sum_{i=1}^7 (y_{3i} - \bar{y}_3)^2 = 12$
D	12, 15, 17, 15, 16, 15	$\bar{y}_4 = 15$	$\sum_{i=1}^6 (y_{4i} - \bar{y}_4)^2 = 14$

We therefore have

$$\begin{aligned}
 k &= 4 \\
 n &= n_1 + n_2 + n_3 + n_4 \\
 &= 5 + 4 + 7 + 6 \\
 &= 22 \\
 \hat{y} &= \frac{n_1\bar{y}_1 + n_2\bar{y}_2 + n_3\bar{y}_3 + n_4\bar{y}_4}{n_1 + n_2 + \dots + n_k} \\
 &= \frac{5 \times 12 + 4 \times 17 + 7 \times 16 + 6 \times 15}{22} \\
 &= 15.
 \end{aligned}$$

The sums of squares are given by

$$\begin{aligned}
 SSE &= \sum_{i=1}^5 (y_{1i} - \bar{y}_1)^2 + \sum_{i=1}^4 (y_{2i} - \bar{y}_2)^2 + \sum_{i=1}^7 (y_{3i} - \bar{y}_3)^2 + \sum_{i=1}^6 (y_{4i} - \bar{y}_4)^2 \\
 &= 38 + 30 + 12 + 14 \\
 &= 94
 \end{aligned}$$

and

$$\begin{aligned}
 SST &= n_1(\bar{y}_1 - \bar{y})^2 + n_2(\bar{y}_2 - \bar{y})^2 + n_3(\bar{y}_3 - \bar{y})^2 + n_4(\bar{y}_4 - \bar{y})^2 \\
 &= 5(-3)^2 + 4(2)^2 + 7(1)^2 + 6(0)^2 \\
 &= 68
 \end{aligned}$$

giving test statistic

$$\begin{aligned}
 F_{obs} &= \frac{SST/(k-1)}{SSE/(n-k)} \\
 &= \frac{68/3}{94/18} \\
 &= 4.34
 \end{aligned}$$

The observed significance level can be calculated using the appropriate table or with a computer program, and can be found to be

$$\alpha_{obs} = .018$$

meaning that there is evidence that the four means are not identical. \square

2.2 ANOVA Table

For our purposes, the *ANOVA* model serves two purposes. First, it is an important statistical procedure. But it also provides a canonical description of the essential structure of much of statistical inference. In particular, this is the decomposition of the total variation of a set of observations into that explainable by a model, and that which cannot be predicted (at least not with the data

currently available). This idea is summarized in the *ANOVA table*. It is important to note that, apart from its application to the one-way *ANOVA* model, this type of table is commonly used to summarize the decomposition of variation for a wide variety of models, which will be seen later in these notes. For the current application the table takes the following form:

Source	SS	df	MS	
Between Treatment (or Treatment)	SST	$k - 1$	$MST = \frac{SST}{k-1}$	$F = \frac{MST}{MSE}$
Within Treatment (or Error)	SSE	$n - k$	$MSE = \frac{SSE}{n-k}$	
Total	SSTO	$n - 1$		

Where

k	Number of groups		
n_i	Sample size of group i		
n	Total sample size	$n_1 + \dots + n_k$	
\bar{y}_i	Sample mean of group i		
\bar{y}	Total sample mean	$\frac{n_1\bar{y}_1 + \dots + n_k\bar{y}_k}{n_1 + \dots + n_k}$	
SSE	Error sum of squares or Within Treatment SS	$\sum_{i=1}^k (n_k - 1)s_k^2$	
SST	Treatment sum of Squares or Between Treatment SS	$\sum_{i=1}^k n_k(\bar{y}_k - \bar{y})^2$	
SSTO	Total sum of squares	$SST + SSE$	
MSE	Mean error sum of squares or Mean within Treatment SS	$\frac{SSE}{n-k}$	
MST	Mean treatment sum of squares or Mean between Treatment SS	$\frac{SST}{k-1}$	
F	F-ratio	$\frac{MST}{MSE}$	

Example 2.3. To construct an ANOVA table for the tape coating problem of Example 2.2 we need the following:

$$\begin{aligned}
 n &= 22 \\
 k &= 4 \\
 SSE &= 94 \\
 MSE &= 94/(n - k) \\
 &= 5.22 \\
 SST &= 68 \\
 MST &= 68/(k - 1) \\
 &= 22.67 \\
 F &= 4.34
 \end{aligned}$$

The ANOVA table is then

Source	SS	df	MS	
Between Treatment	68	3	22.67	4.34
Within Treatment	94	18	5.22	
Total	162	21		

□

Recall that it is assumed that each sample comes from a population with possibly differing means, but with one common variance σ^2 . It can be shown that MSE is an estimator of σ^2 . In fact, if there are $k = 2$ groups then the MSE is identical to the pooled sample variance S_p^2 and plays the same role when $k > 2$.

2.3 Bonferroni Correction for Multiple Comparisons

We often encounter a situation in which we wish to report several confidence intervals or hypothesis tests. If we use a confidence level $(1 - \alpha)$ for each confidence interval, or a significance level of α for each hypothesis test, we must consider the fact that the probability of at least one error among all inference statements will be greater than α .

A number of procedures exist with which to control *familywise error rate* (FWE), that is, the probability that among a set of m inference statements there is at least one error (the term *group* is sometimes used in place of ‘familywise’). The commonly used convention is that an error rate suitable for a single inference procedure should also be applied to multiple inferences, so that the FWE is commonly set to $\alpha_{FWE} = 0.05$. We can also refer to familywise (or group) confidence level $1 - \alpha_{FWE}$.

A large number of *multiple comparison* procedures exist, some specialized and others general. Probably the most commonly encountered method is known as the *Bonferroni correction procedure* (BCP), which is applicable, in principle, to any multiple comparison model. Recall Boole’s inequality:

$$P(\cup_{i=1}^m E_i) \leq \sum_{i=1}^m P(E_i).$$

Suppose we are given m level $(1 - \alpha)$ confidence intervals

$$E_i = \{ \text{the } i\text{th CI is incorrect} \}.$$

Then $P(E_i) = \alpha$ and

$$P(\cup_{i=1}^m E_i) \leq m\alpha. \quad (2.4)$$

This means that all m confidence intervals are correct with a probability of at least $1 - m\alpha$. Therefore, in order to achieve a FWE of α_{FWE} , we would need to use a confidence level of $(1 - \alpha_{FWE}/m)$.

Example 2.4. Normally, to achieve a confidence level of 95% for a confidence interval

$$\bar{X} \pm z_{\alpha/2} \sigma / \sqrt{n} \quad (2.5)$$

we would set $\alpha = 0.05$ and therefore use critical value $z_{0.025} \approx 1.96$. If we wanted to simultaneously report $m = 4$ confidence intervals with FWE $\alpha_{FWE} = 0.05$, we would build separate level $(1 - \alpha/m)$ confidence intervals. From (2.4), this would give

$$\alpha_{FWE} \leq m\alpha/m = \alpha,$$

so that it would be appropriate to set $\alpha = \alpha_{FWE}$. Therefore, in (2.5) we would use the critical value

$$z_{\alpha_{FWE}/(2m)} = z_{0.05/8} = z_{0.00625} \approx 2.5$$

for $\alpha_{FWE} = 0.05$. However, construction of the confidence intervals uses the same methodology once the Bonferroni correction has been applied.

The sample principle applies to hypothesis tests. If we want to report m hypothesis tests with a familywise Type I error of α_{FWE} (that is, at least one Type I error among the m tests), then each test must be carried out with a significance level of α_{FWE}/m . \square

2.4 Post hoc Analysis in ANOVA

If we conclude that there is some difference between means using the F -test, then we may wish to further explore how the means differ. A common way to achieve this is through the use of *pairwise multiple comparisons*.

Using the BCP we have

$$\bar{y}_i - \bar{y}_j \pm t_{\alpha/(m^2), n-k} \sqrt{MSE \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}$$

where m is the number of comparisons we wish to make (we need not be interested in all available comparisons).

If μ_i and μ_j are two group means, then a confidence interval for $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm q_\alpha \sqrt{\frac{MSE}{2} \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}$$

where q_α is the critical value from the *studentized range* distribution with $k - 1$ treatment degrees of freedom and $n - k$ error degrees of freedom (these have the same interpretation as the numerator and denominator degrees of freedom for the F -distribution). Tables for these critical values are available in most textbooks. This is known as *Tukey's pairwise procedure* or sometimes the *Tukey-Kramer pairwise procedure*. It should be noted that the procedure is approximate, unless the design is balanced.

Note that there will be $k(k - 1)/2$ comparisons. It needs to be stressed that the confidence level $1 - \alpha$ represents the probability that *all* $k(k - 1)/2$ confidence intervals are correct, and not just each one taken individually.

To continue with the tape coating problem, we have 18 error degrees of freedom and 3 treatment degrees of freedom so if we want a 95% confidence interval for all pairwise comparisons simultaneously we set

$$q_{0.05} = 4.00$$

and we also have

$$MSE = 5.22$$

with

Coating	n_i	Sample mean
A	$n_1 = 5$	$\bar{y}_1 = 12$
B	$n_2 = 4$	$\bar{y}_2 = 17$
C	$n_3 = 7$	$\bar{y}_3 = 16$
D	$n_4 = 6$	$\bar{y}_4 = 15$

giving pairwise confidence intervals

Pair	Confidence Interval
$\mu_1 - \mu_2:$	-5 ± 4.33
$\mu_1 - \mu_3:$	-4 ± 3.78
$\mu_1 - \mu_4:$	-3 ± 3.91
$\mu_2 - \mu_3:$	1 ± 4.06
$\mu_2 - \mu_4:$	2 ± 4.17
$\mu_3 - \mu_4:$	1 ± 3.59

We may conclude that μ_1 is significantly different from μ_2 and μ_3 but can make no other conclusions based on this procedure.

Note that there are many pairwise procedures, most notably the *Scheffe test* which is very conservative. This provides a FWE of α_{FWE} of confidence intervals for all *contrasts*

$$C = \sum_{i=1}^k c_i \mu_i, \text{ where } \sum_{i=1}^k c_i = 0.$$

A pairwise comparison of the form $\mu_i - \mu_j$ is a contrast of the form $c_i = 1$, $c_j = -1$, $c_{i'} = 0$ for $i' \neq i, j$.

2.5 Nonparametric ANOVA

Recall that ANOVA may be thought of as an extension of the two-sample t-test for differences in mean to a K -sample test for differences in means, under the assumptions that variances are equal. Similarly, the *Kruskal-Wallis test* is an extension of the Wilcoxon rank sum test to K samples, and may be considered a nonparametric alternative to ANOVA. Under the null hypothesis, K samples are taken from K identical distributions (not necessarily normally distributed). We won't discuss the details of this test, but will note that the Kruskal-Wallis test is implemented in most statistical software packages. It would be appropriate to use whenever ANOVA might be used, but does not assume that the data is normally distributed.

2.6 Assumptions

The essential assumptions made for ANOVA are that population i has a $N(\mu_i, \sigma^2)$ distribution. The means may differ between populations but variances do not. In addition, each sample is a true random sample, and the samples are independent of each other.

Of course, ANOVA is a technique which has received a great deal of attention by statistical practitioners, so that there are a wide variety of techniques which may be used when these assumptions do not hold.

2.7 ANOVA in R

To fit an ANOVA model in R, we express the data as such a model. The variable Y is a single vector which contains all the variable. X is a single vector of *factors* which define the treatments. For example, to set up an ANOVA model in R, we can use the commands:

```
> y1 = rnorm(5,mean=10,sd=2.4)
> y2 = rnorm(6,mean=20,sd=2.4)
> y3 = rnorm(4,mean=20,sd=2.4)
>
> y = c(y1, y2, y3)
> x = c(rep(1,5), rep(2,6), rep(3, 4))
> x = as.factor(x)
> cbind(x,y)
      x         y
[1,] 1  6.526123
[2,] 1 10.639951
[3,] 1  8.128591
[4,] 1 10.922928
[5,] 1 13.934701
[6,] 2 20.502000
[7,] 2 22.109955
[8,] 2 22.575551
[9,] 2 23.177065
[10,] 2 19.509436
[11,] 2 19.890914
[12,] 3 15.414790
[13,] 3 18.572681
[14,] 3 20.668094
[15,] 3 15.777068
>
```

This simulates an ANOVA model with $k = 3$ treatments, identified in the factor variable x . The sample sizes are $n_1 = 5$, $n_2 = 6$, $n_3 = 4$, with means $\mu_1 = 10$, $\mu_2 = \mu_3 = 20$. The common variance is $\sigma^2 = 2.4^2$.

It is usually a good idea to plot the data, and this can be done using boxplots. The R model notation can be used to separate the groups:

```
> boxplot(y ~ x)
```

The plot is shown in Figure 2.1.

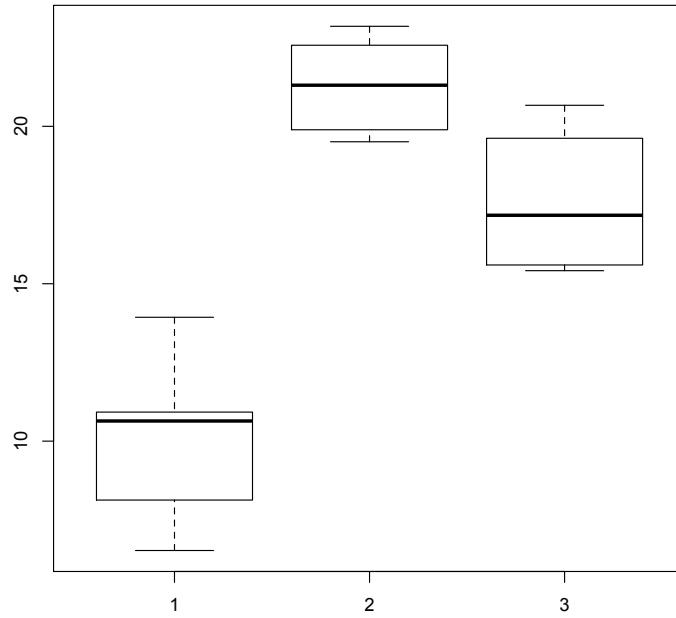


Figure 2.1: Multiple boxplots for ANOVA example

There are several ways to fit an ANOVA model in R. One dedicated function is `aov()` used as follows:

```
> fit = aov(y ~ x)
> summary(fit)
    Df Sum Sq Mean Sq F value    Pr(>F)
x        2   352.0   176.0   33.85 1.17e-05 ***
Residuals 12   62.4     5.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the fit itself can be stored as an object, which is generally good practice. Fit objects can then be used as input for generic functions, which provide summaries for the appropriate type of object. For example `summary()` gives for an `aov()` object the standard ANOVA table.

Tukey's pairwise procedure is also available for an ANOVA fit using the `TukeyHSD()` function (HSD refers to 'Honest Significant Difference'):

```
> fit.Tukey = TukeyHSD(fit)
> fit.Tukey
```

```

Tukey multiple comparisons of means
 95% family-wise confidence level

Fit: aov(formula = y ~ x)

$x
    diff      lwr      upr     p adj
2-1 11.263695  7.579838 14.9475516 0.0000085
3-1  7.577699  3.496636 11.6587623 0.0009009
3-2 -3.685995 -7.613000  0.2410094 0.0665423

>

```

Notice that a new R object was produced by the TukeyHSD() function. If we use the generic plot() function we get the following plot (Figure 2.2):

```
> plot(fit.Tukey)
```

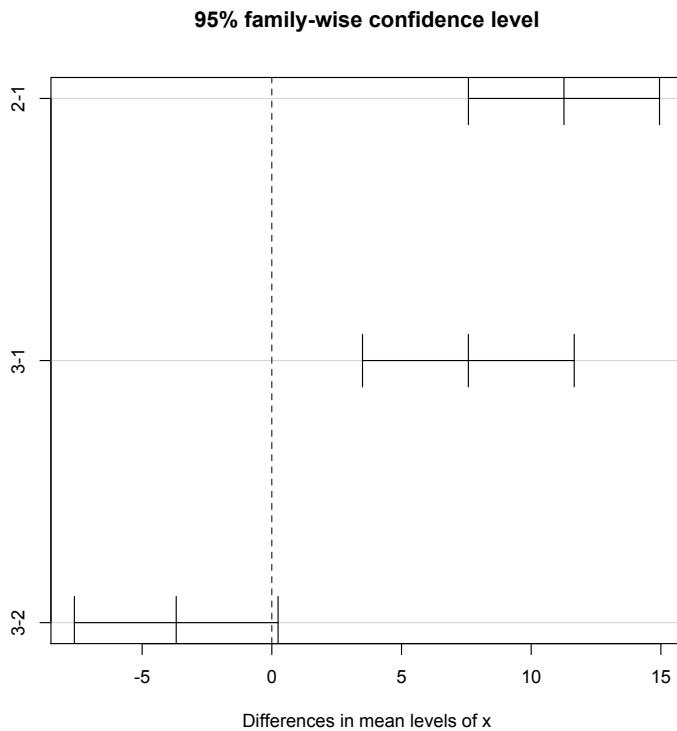


Figure 2.2: Graphical representation of Tukey's pairwise procedure.

2.8 Equality of Variances

We have seen how to test for the equality of two variances. *Bartlett's Test* is a generalization to k variances suitable for ANOVA. This is available using the `bartlett.test()`, using the same model notation. This may use the same model notation:

```
> bartlett.test(y ~ x)

Bartlett test of homogeneity of variances

data: y by x
Bartlett's K-squared = 1.5712, df = 2, p-value = 0.4558
```

The large p -value means that the hypothesis of equality of variances (also known as *homoscedasticity*, as opposed to *heteroscedasticity*) was not rejected.

It is also possible to use `bartlett.test()` by input a list of samples:

```
> y.list = list(y1, y2, y3)
> y.list
[[1]]
[1] 6.526123 10.639951 8.128591 10.922928 13.934701

[[2]]
[1] 20.50200 22.10996 22.57555 23.17706 19.50944 19.89091

[[3]]
[1] 15.41479 18.57268 20.66809 15.77707

> bartlett.test(y.list)
```

```
Bartlett test of homogeneity of variances

data: y.list
Bartlett's K-squared = 1.5712, df = 2, p-value = 0.4558
```

>

However, `aov()` cannot be used this way.

We finally note that a model can be converted to a list using the `split()` function:

```
> split(y,x)
$'1'
[1] 6.526123 10.639951 8.128591 10.922928 13.934701

$'2'
[1] 20.50200 22.10996 22.57555 23.17706 19.50944 19.89091
```

```
$‘3’
[1] 15.41479 18.57268 20.66809 15.77707

>
```

2.9 The Kruskal-Wallis Test for Nonparametric ANOVA

We have briefly introduced the Kruskal-Wallis test as an extention of the rank sum procedure to more than 2 samples. This is implemented in R using the `kruskal.test()` function, which is similar to `aov()`. For example, consider the simulated data:

```
> y1 = rnorm(5,mean=10,sd=2.4)
> y2 = rnorm(6,mean=20,sd=2.4)
> y3 = rnorm(4,mean=20,sd=2.4)
>
> y = c(y1, y2, y3)
> x = c(rep(1,5), rep(2,6), rep(3, 4))
> x = as.factor(x)
> cbind(x,y)
      x         y
[1,] 1 11.695035
[2,] 1 7.967687
[3,] 1 8.891760
[4,] 1 12.476237
[5,] 1 10.996793
[6,] 2 21.324256
[7,] 2 19.594350
[8,] 2 15.141688
[9,] 2 21.956811
[10,] 2 19.251928
[11,] 2 16.064112
[12,] 3 21.472492
[13,] 3 20.310484
[14,] 3 26.817237
[15,] 3 20.906802
>
> boxplot(y ~ x)
>
```

The boxplot is shown in Figure 2.3. The data may be input into the `kruskal.test()` function as a model:

```
> fit = kruskal.test(y ~ x)
> summary(fit)
    Length Class  Mode
statistic 1     -none- numeric
```

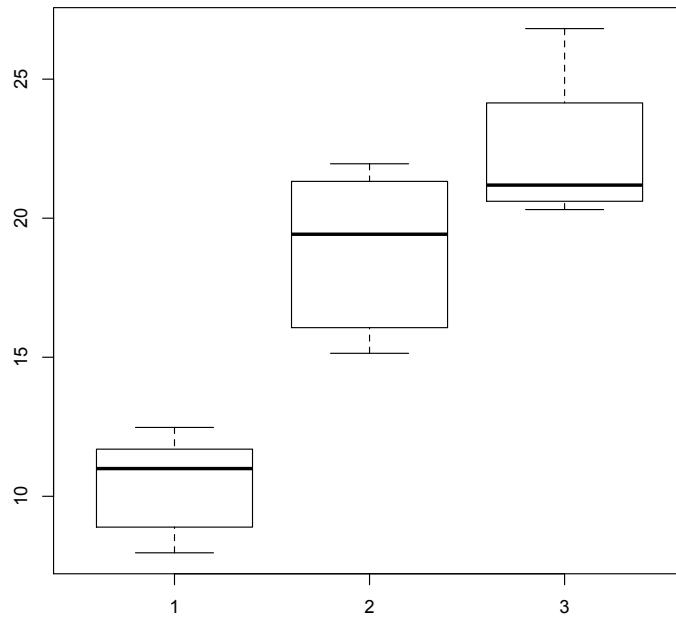


Figure 2.3: Multiple boxplots for Kruskal-Wallis test.

```

parameter 1      -none- numeric
p.value   1      -none- numeric
method    1      -none- character
data.name 1      -none- character
> fit

Kruskal-Wallis rank sum test

data:  y by x
Kruskal-Wallis chi-squared = 10.3958, df = 2, p-value = 0.005528

> stat = fit$statistic
> df = fit$parameter
> 1-pchisq(stat,df)
Kruskal-Wallis chi-squared
                  0.005528069
>

```

However, in this case the `summary()` function only lists the labels which define the list elements of the `fit` object. These labels provide access to the output quantities. For example, the significance level is calculated from a χ^2 statistic with 2 degrees of freedom. We can access the statistic and the

degrees of freedom by references to `fit$statistic` and `fit$parameter`. We have justly illustrated this by recalculating the *p*-value from the output.

The function `kruskal.test()` also accepts multiple samples in list form. In addition, it has a `g` option which defines groups, permitting the data to be entered as a single array”

```
> y.list = list(y1, y2, y3)
> y.list
[[1]]
[1] 11.695035 7.967687 8.891760 12.476237 10.996793

[[2]]
[1] 21.32426 19.59435 15.14169 21.95681 19.25193 16.06411

[[3]]
[1] 21.47249 20.31048 26.81724 20.90680

> kruskal.test(y.list)

Kruskal-Wallis rank sum test

data: y.list
Kruskal-Wallis chi-squared = 10.3958, df = 2, p-value = 0.005528

> kruskal.test(y, g = x)

Kruskal-Wallis rank sum test

data: y and x
Kruskal-Wallis chi-squared = 10.3958, df = 2, p-value = 0.005528

>
```

2.10 Postscript

ANOVA is an acronym for “(An)alysis (o)f (Va)riance”, which is the principle that the variation of a response can be decomposed quantitatively into two sources. The conventional terms used for these sources varies. If this author had to invent new names at a moments notice, they might be “seen” and “unseen”. The source of “seen” variation would be any observed predictor variable, or variables, capable of explaining a significant amount of the response variation. The “unseen” variation is the portion of response variation which cannot be explained by any predictor. If we define variation quantitatively by one possible method of evaluation, namely, *sum of squares* (Section 2.1), we have the remarkable identity

$$SS_{total} = SS_{seen} + SS_{unseen}.$$

An important part of the transition from introductory to intermediate statistics is an understanding of this identity, and the idea that variation is quantifiable and decomposable.

Of course, SS_{seen} is usually referred to as *treatment sum of squares* in ANOVA, or *model sum of squares* in linear regression analysis (Section 3.2). Furthermore, this source of variation can be further decomposed into sources attributable to multiple predictors when appropriate. Then SS_{unseen} is usually referred to as *error sum of squares* or *residual sum of squares*. The latter is more often associated with regression analysis, although, as we point out below, both are formally correct in either ANOVA or regression analysis. Like SS_{seen} , SS_{unseen} can sometimes be further decomposed, in a manner which can be measured. For example, the term *repeated measures* refers to multiple measurements observed from some set of subjects. In this case, SS_{unseen} can, in principle, be decomposed into *within subject* and *between subject* variation, the former attributable to innate differences between subjects, the latter attributable to “whatever variation remains” (as suggested by the term “residual”).

We must also note that the ANOVA model presented in this chapter is really *one-way ANOVA*, which is essentially an extension of the two-sample *t*-test with pooled variance to two or more samples (alternatively, the two-sample *t*-test is a special case of the one-way ANOVA model). The question is simply whether or not there is a difference in means among two or more samples (or *treatments*). It is possible to introduce multiple *factors* (*m*-way ANOVA) or *blocking*, resulting in further decomposition of SS_{seen} . For a good reference see Neter *et al.* (1996) or Box *et al.* (2018).

Apart from being a methodology of importance for its own sake, the idea of ANOVA is applicable, and quite important, to most models we consider. It therefore appears repeatedly in these notes (Sections 3.2, 9.3, 10.2, 17.2).

It should also be noted that the problem of multiple comparisons (Section 2.3) is an important one in its own right, and has assumed more prominence with the onset of high-throughput, or “big data”. The underlying theory is quite detailed, and a comprehensive treatment on the subject can be found in Dudoit and van der Laan (2008). This author has contributed a somewhat more compact overview of the subject (Almudevar, 2013).

Chapter 3

Linear Regression - Introduction

In statistical inference, the term *model* usually refers to an attempt to use any number of *independent variables* X_1, \dots, X_m to *predict* a response Y . By “predict”, we mean deduce a form of the distribution of Y that is allowed to depend on X_1, \dots, X_m . The roles played by the variables are not symmetric. In most cases, the independent variables are *not* considered random, and do not possess a distribution. If they do, a distinct class of statistical models is available, for example, *Deming regression* (Deming, 1943).

The ANOVA model of Chapter 2 is the simplest type of model. The predictor variable was not explicitly introduced, but can be taken to be the categorical variable which identifies the treatment label of each observation. We next introduce, by example, a model in which the predictor variable is quantitative.

Consider the scatter plot in Figure 3.1 representing 392 automobiles. The horizontal axis gives the engine displacement in cubic inches and the vertical axis gives horsepower.

There seems to be a definite increasing trend in horsepower as engine displacement increases. If we set

$$\begin{aligned} X &= \text{Engine Displacement} \\ Y &= \text{Horsepower} \end{aligned}$$

then we should have approximately a linear relationship

$$Y = \beta_0 + \beta_1 X$$

where β_0 and β_1 are *coefficients* to be determined from the data. Looking at the scatter plot we see that the relationship will not be exact, so we introduce a random term ϵ (*epsilon*) into the equation.

$$Y = \beta_0 + \beta_1 X + \epsilon.$$

The usual terminology is to refer to Y as the *dependent variable* and to X as the *independent variable* or *predictor*. Here, there is only one predictor X , so the model is termed *simple linear regression*. When more predictors are used, for example $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, the model is termed *multiple linear regression*.

If we have n pairs of dependent and independent observations

$$(X_1, Y_1), \dots, (X_n, Y_n)$$

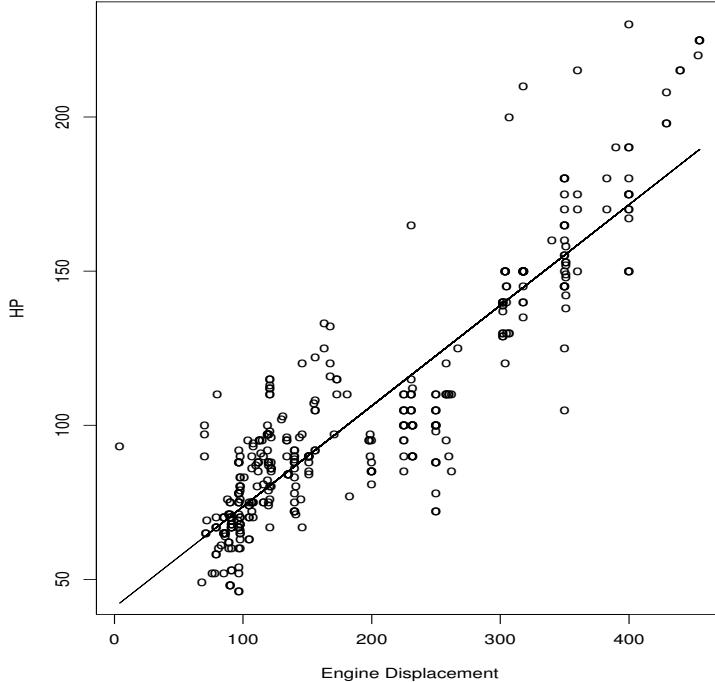


Figure 3.1: Scatter plot of automobile data

then the regression equation can be written in terms of the sample

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad i = 1, \dots, n. \quad (3.1)$$

Here, we assume that the *error terms* $\epsilon_1, \dots, \epsilon_n$ form a random sample from $N(0, \sigma^2)$. We do not observe the error terms (unlike X_i and Y_i), but we can estimate σ^2 .

The *linear least squares coefficients* are given by

$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}. \end{aligned} \quad (3.2)$$

The derivation of these coefficients will be discussed in Section 3.4, but we will simply note for now that they seem to achieve the purpose at hand. In particular, with a large enough sample size we have estimates

$$\hat{\beta}_0 \approx \beta_0 \text{ and } \hat{\beta}_1 \approx \beta_1,$$

giving the estimated relationship between X and Y

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

We also have the *predicted responses*

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

for each sample pair $i = 1, \dots, n$. Of course, we may construct a predicted response for a predictor value not represented in the sample, that is,

$$\hat{Y}_x = \hat{\beta}_0 + \hat{\beta}_1 x$$

is the predicted response for a predictor value $X = x$.

Example 3.1. For the data shown in Figure 3.1 we have least squares coefficients

$$\begin{aligned}\hat{\beta}_1 &= 0.327 \\ \hat{\beta}_0 &= 41.002,\end{aligned}$$

and this line is drawn in that plot. \square

A brief introduction to inference for least squares coefficients

Most statistical software implements linear regression, giving output in the following format (using the data of Figure 3.1):

Model	Unstandardized Coefficients				
	B	Std. Error	t	Sig.	
1	(Constant) 41.002	1.792	22.884	.000	
	Engine Displacement (cu. inches) .327	.008	40.500	.000	

Least squares coefficients can be taken directly from this table. If we wish to construct a level $(1 - \alpha)100\%$ confidence interval for β_0 and β_1 we may use

$$\begin{aligned}\hat{\beta}_0 &\pm t_{n-2,\alpha/2} \times \text{Std. Error for } \hat{\beta}_0 \\ \hat{\beta}_1 &\pm t_{n-2,\alpha/2} \times \text{Std. Error for } \hat{\beta}_1\end{aligned}$$

where the standard error may be taken from the table. Note that the appropriate degrees of freedom for the t -distribution critical values are $n - 2$. If n is very large, we may use the standard normal critical value $z_{\alpha/2}$ instead. For the above example we have 95% confidence intervals

$$\begin{aligned}CI_{.95} &= 41.002 \pm 1.96 \times 1.792 \\ &= 41.002 \pm 3.5\end{aligned}$$

for β_0 and

$$\begin{aligned}CI_{.95} &= 0.327 \pm 1.96 \times 0.008 \\ &= 0.327 \pm 0.0157\end{aligned}$$

for β_1 .

An important hypothesis test is

$$\begin{aligned} H_o &: \beta_1 = 0 \\ H_a &: \beta_1 \neq 0. \end{aligned}$$

This tells us whether or not there is any relationship between the dependent and independent variable. The observed significance level can be read directly from the table in the last column. Here, the P -value is given as 0, which we should report as a $P < 0.001$, meaning that there is strong evidence of a linear relationship between engine displacement and horsepower.

3.1 Residuals

An important assumption of the linear regression model is that the error terms $\epsilon_1, \epsilon_2, \dots, \epsilon_n$ defined by

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

are equivalent to a random sample from a normal distribution with mean 0 and variance σ^2 . Implicit in this formulation is the assumption that there is a linear relationship between X and Y .

Of course, the ϵ_i 's cannot be directly observed, but they can be estimated by the *residuals*, given by

$$e_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i = Y_i - \hat{Y}_i$$

once the regression has been calculated. There are several ways to use the residuals to check the assumptions.

- (1) Draw a scatter plot of the points (e_i, \hat{Y}_i) where \hat{Y}_i is the *predicted value*

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

If the assumptions are satisfied there should be no pattern.

- (a) Check to see if the variation of the residuals appears to increase or decrease systematically. If so, this means that the variance of the error terms is not constant.
 - (b) If large groups of residuals located next to each other appear to be all above or all below zero, then the assumption that the error terms are independent of each other may be incorrect. This is a frequent occurrence when the X_i 's represent sequential points in time.
 - (c) If the residuals appear to suggest some functional form, then the assumption of a linear relationship between X and Y may be incorrect.
- (2) To check for the assumption of normality of the error terms, construct a normal probability plot of the residuals. Departures from linearity indicate departures from normality of the error terms.

As a final remark linear regression, like ANOVA, is a widely used tool, and many techniques exist which may be used when some of these assumptions are not valid.

Example 3.2. To continue with the automobile section we present a residual plot and a normal probability plot (Figure 3.2).

The residual plot shows a somewhat different behavior below and above 100 horsepower, which might be investigated. Other than that, no systematic departure from the assumption of no pattern is indicated.

The normal probability plot is approximately linear, except for the two extreme regions. This indicates that the normal distribution might not be accurate for very small tail probabilities, but otherwise should suffice as an approximation.

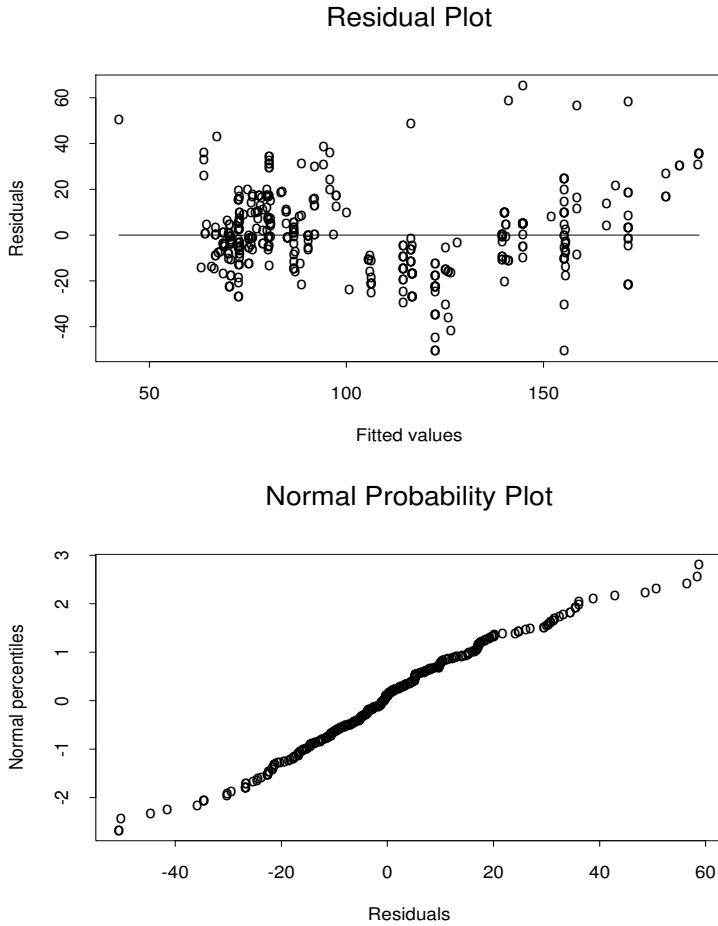


Figure 3.2: Residual plot and normal probability plot of residuals for Example 3.2

□

Example 3.3. As a second example, a scatter plot is presented with a linear regression fit in Figure 3.3. From the scatter plot, we can see that although there is a strong relationship between miles per gallon and horsepower, it is not a strictly linear one. Accordingly, the residual plot indicates a systematic functional form, suggesting that a linear fit is not the appropriate one.

□

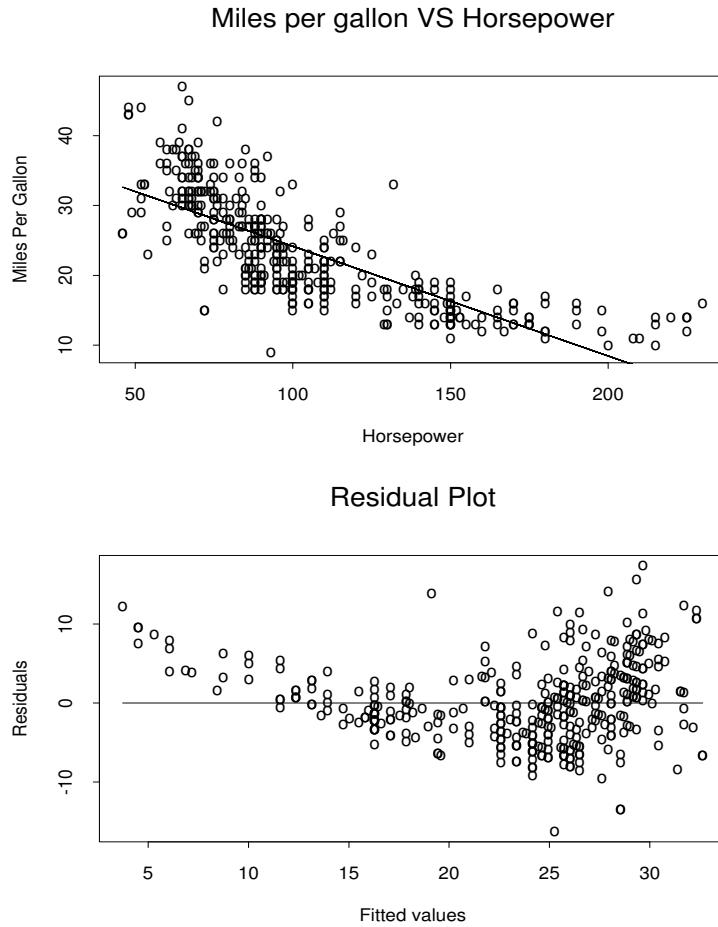


Figure 3.3: Scatter plot of MPG vs. Horsepower with linear regression fit, and residual plot for Example 3.3

3.2 ANOVA approach

In linear regression, variation may be decomposed in a manner similar to that of ANOVA. We start with the *error sum of squares*

$$SSE = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2,$$

and the *mean error sum of squares*

$$MSE = \frac{SSE}{n-2}.$$

The MSE is analogous to the MSE encountered in ANOVA, with $K = 2$ treatments corresponding to the 2 unknown parameters β_0 and β_1 . In fact, the MSE functions as an estimate of the variance σ^2 encountered in the distribution $\epsilon_i \sim N(0, \sigma^2)$:

$$\hat{\sigma}^2 = MSE \approx \sigma^2.$$

We then have, as for ANOVA, the total sum of squares

$$SSTO = \sum_{i=1}^n (Y_i - \bar{Y})^2.$$

By convention, instead of the *treatment sum of squares* SST we define the *regression sum of squares*

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

noting that the two quantities serve similar functions. It can be shown that

$$SSTO = SSR + SSE.$$

This means that, as in ANOVA, the total variation $SSTO$ can be decomposed into variation SSR explained by the model and variation SSE attributable to the error terms ϵ_i . The ANOVA table for simple linear regression therefore looks like:

Source	SS	df	MS	
Regression	SSR	1	$MSR = \frac{SSR}{1}$	$F = \frac{MSR}{MSE}$
Error	SSE	$n - 2$	$MSE = \frac{SSE}{n-2}$	
Total	SSTO	$n - 1$		

As in ANOVA, F has an F -distribution with 1 numerator and $n - 2$ denominator degrees of freedom under the hypothesis

$$H_0 : \beta_1 = 0.$$

A quantity of considerable importance is the *coefficient of determination*

$$R^2 = 1 - \frac{SSE}{SSTO} = \frac{SSR}{SSTO}$$

which can be interpreted as the proportion of the total variation explainable by the model. We always have $0 \leq R^2 \leq 1$, so that larger values (say, $R^2 \geq 0.25$) mean that the predictor X has significant explanatory power.

3.3 The Relationship Between Linear Regression and Correlation

It is important to note the similarity between the definition of r and the estimate of the slope β_1 for simple linear regression:

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

This means r and $\hat{\beta}_1$ have a close relationship:

$$\begin{aligned}
 r &= r \frac{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2}}{\sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}} \\
 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \times \frac{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2}}{\sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}} \\
 &= \hat{\beta}_1 \sqrt{\frac{S_X^2}{S_Y^2}}
 \end{aligned} \tag{3.3}$$

where S_X^2 and S_Y^2 are the samples variances of the X_i 's and Y_i 's.

When deducing the distribution properties of r , it is usually assumed that X and Y together possess a *bivariate normal distribution*. This means that X and Y are both normally distributed, and also possess a linear relationship of the form

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{3.4}$$

where β_0 and β_1 are constants and $\epsilon \sim N(0, \sigma^2)$ is independent of both X and Y . It can be shown that when (3.4) holds the correlation between X and Y is

$$\rho = \rho_{XY} = \beta_1 \frac{\sigma_X}{\sigma_Y},$$

which is directly comparable to (3.3) (when convenient, subscripts may be added to the symbols r or ρ to identify the relevant random variables). Of course, one important difference remains between (3.4) and the simple linear regression model, namely that for simple linear regression X is interpreted as a nonrandom predictor variable, whereas in (3.4) X is a random variable. Nonetheless, both models depend on the very specific notion of linear dependence between two variables.

It is important to note that assuming only that X and Y are normally distributed does not suffice to define the bivariate normal distribution. The assumption of a linear relationship is also needed.

3.4 The Derivation of the Least Squares Coefficients

The least squares coefficients $\hat{\beta}_0, \hat{\beta}_1$ are those which minimize the error sum of squares SSE . It is natural, therefore, to consider that quantity a function of two variables, say $SSE[\beta_0, \beta_1]$. The main step is to identify the stationary points. So, write

$$SSE[\beta_0, \beta_1] = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i)^2,$$

then take the partial derivatives

$$\begin{aligned}\frac{\partial SSE[\beta_0, \beta_1]}{\partial \beta_0} &= \sum_{i=1}^n -2(Y_i - \beta_0 - \beta_1 X_i), \\ \frac{\partial SSE[\beta_0, \beta_1]}{\partial \beta_1} &= \sum_{i=1}^n -2X_i(Y_i - \beta_0 - \beta_1 X_i).\end{aligned}$$

If each partial derivative is set to 0, the solution is obtained by substituting $\hat{\beta}_0, \hat{\beta}_1$ for β_0, β_1 . This can be expressed

$$\begin{aligned}\sum_{i=1}^n Y_i &= \sum_{i=1}^n (\hat{\beta}_0 + \hat{\beta}_1 X_i) \\ &= n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^n X_i\end{aligned}$$

for the partial derivative with respect to β_0 , and

$$\begin{aligned}\sum_{i=1}^n X_i Y_i &= \sum_{i=1}^n X_i (\hat{\beta}_0 + \hat{\beta}_1 X_i) \\ &= \hat{\beta}_0 \sum_{i=1}^n X_i + \hat{\beta}_1 \sum_{i=1}^n X_i^2,\end{aligned}$$

for the partial derivative with respect to β_1 . These are referred to as the *normal equations*, and form a system of two linear equations in two unknowns, with solution

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}.\end{aligned}$$

as given in Equation (3.2). What remains is the technical problem of verifying that the stationary points identify the minimum of $SSE[\beta_0, \beta_1]$, which follows from the strict convexity of this function.

3.5 Assumptions

The assumptions underlying simple linear regression are all implied in the model defined in (3.1):

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad i = 1, \dots, n. \quad (3.5)$$

Essentially, we assume $Y_i \sim N(\mu_i, \sigma^2)$ for some σ^2 which does not vary with the index i , where the means μ_i are given by

$$\mu_i = \beta_0 + \beta_1 X_i. \quad (3.6)$$

Finally, the responses Y_i are assumed to be independent. This is equivalent to assuming that $\epsilon_1, \dots, \epsilon_n$ is a random sample from distribution $N(0, \sigma^2)$, and the response Y_i is given by (3.1).

Chapter 4

Linear Regression - Inference

In this section we consider in more detail inference for linear regression. We will emphasize the simple linear regression model:

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

but the ideas can be generalized when additional predictors are added.

Usually, there is greater interest in β_1 , the *slope* of the regression line, than β_0 , the point on the vertical response axis intercepted by the regression line at predictor value $X = 0$ (hence β_0 is commonly known as the *intercept*). This is because the motivation for regression is usually to determine a relationship between the dependent and independent variables, and a relationship can be said to exist between them if and only if the slope β_1 is not zero.

We may consider two estimation problems. The mean response for predictor value x is

$$\mu_x = \beta_0 + \beta_1 x.$$

In principle, we may consider μ_x for any value x , even if x does not equal the value of any predictor in a given sample. However, it is usually not recommended that x be *extrapolated* beyond the range of the observed predictors. If we have some reason to set $x > \max_i X_i$ or $x < \min_i X_i$, it should be noted in any report that the resulting inference represents an extrapolation beyond the observed range of the predictor variables. Of course, the intercept β_0 is a special case of μ_x , in particular,

$$\beta_0 = \beta_0 + \beta_1 \times 0 = \mu_0,$$

however, β_1 cannot be expressed as μ_x for some x in this way.

4.1 Inference of Regression Parameters

We may define a general parameter β_i , and note that its inference assumes a general form (we have, so far, encountered β_0 and β_1 for simple linear regression). We have seen estimates

$$\hat{\beta}_i \approx \beta_i, \quad i = 0, 1$$

and we may add

$$\hat{\mu}_x \approx \hat{\beta}_0 + \hat{\beta}_1 x.$$

Note that the *predicted responses*

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i, \quad i = 1, \dots, n,$$

are special cases $\hat{Y}_i = \hat{\mu}_{X_i}$, and are commonly known as *fitted values*, since the estimated regression line passes through the points (X_i, \hat{Y}_i) .

Under the assumption that the error terms $\epsilon_1, \dots, \epsilon_n$ are an independent random sample from $N(0, \sigma^2)$ for some fixed variance σ^2 , we have

$$\hat{\beta}_i \sim N\left(\beta_i, \sigma_{\hat{\beta}_i}^2\right),$$

and

$$\hat{\mu}_x \sim N\left(\mu_x, \sigma_{\hat{\mu}_x}^2\right).$$

It is worth noting at this point that $\hat{\beta}_i$ and $\hat{\mu}_x$ are *unbiased* estimates of β_i and μ_x , since

$$E[\hat{\beta}_i] = \beta_i \text{ and } E[\hat{\mu}_x] = \mu_x,$$

(not all commonly used estimators are unbiased).

For simple linear regression, the values of $\sigma_{\hat{\beta}_i}^2$ and $\sigma_{\hat{\mu}_x}^2$ can be shown to be:

$$\sigma_{\hat{\beta}_1}^2 = \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \tag{4.1}$$

and

$$\sigma_{\hat{\mu}_x}^2 = \sigma^2 \left[\frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right] \tag{4.2}$$

where we have mean value of the predictor:

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n}.$$

Since $\beta_0 = \mu_0$ we can obtain directly from (4.2) the variance of $\hat{\beta}_0$ by substituting $x = 0$:

$$\sigma_{\hat{\beta}_0}^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right]. \tag{4.3}$$

As we might expect, the values of $\sigma_{\hat{\beta}_i}^2$ and $\sigma_{\hat{\mu}_x}^2$ directly depend on error variance σ^2 , which is usually unknown. Of course, we already have estimate

$$\hat{\sigma}^2 = MSE = \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n-2} \approx \sigma^2,$$

so we replace σ^2 in (4.1), (4.2) and (4.3) with $\hat{\sigma}^2$, to obtain the *standard errors*

$$S_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2}}, \tag{4.4}$$

$$S_{\hat{\mu}_x} = \hat{\sigma} \sqrt{\frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2}} \tag{4.5}$$

and

$$S_{\hat{\beta}_0} = \hat{\sigma} \sqrt{\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2}}. \tag{4.6}$$

4.1.1 Confidence intervals for simple linear regression

Given the standard errors for β_i or μ_x a level $(1 - \alpha)$ confidence interval for β_i is given by

$$\hat{\beta}_i \pm t_{n-2,\alpha/2} \times S_{\hat{\beta}_i},$$

or for μ_x by

$$\hat{\mu}_x \pm t_{n-2,\alpha/2} \times S_{\hat{\mu}_x},$$

where $t_{n-2,\alpha/2}$ is the $\alpha/2$ critical value for a t -distribution with $n - 2$ degrees of freedom.

4.1.2 Hypothesis tests for simple linear regression

If we wish to test against a hypothesis

$$H_o : \beta_i = \beta'_i \quad (4.7)$$

we use statistic

$$T = \frac{\hat{\beta}_i - \beta'}{S_{\hat{\beta}_i}}$$

which, under the hypothesis defined in Equation (4.7) has a t -distribution with $n - 2$ degrees of freedom.

The most common hypothesis test in the context of simple linear regression is obtained by setting hypothetical value $\beta'_1 = 0$, that is, the two-sided test:

$$H_o : \beta_1 = 0 \text{ against } H_a : \beta_1 \neq 0,$$

which gives observed significance level

$$\alpha_{obs} = 2P(T \leq -|T_{obs}|)$$

where

$$T_{obs} = \frac{\hat{\beta}_1}{S_{\hat{\beta}_1}}$$

and T has a t -distribution with $n - 2$ degrees of freedom. When suitable, one-sided hypothesis tests can be carried out.

4.1.3 Prediction intervals for simple linear regression

Let's define a random variable

$$Y_x \sim N(\mu_x, \sigma^2),$$

which can be interpreted as a *future response* from a linear model

$$Y_x = \beta_0 + \beta_1 x + \epsilon$$

for a given predictor value $X = x$, and $\epsilon \sim N(0, \sigma^2)$. We might wish to place *prediction bounds* on Y_x , that is, values Y_L, Y_U for which

$$P(Y_L \leq Y_x \leq Y_U) = 1 - \alpha.$$

We might set $1 - \alpha = 95\%$. If $\beta_0, \beta_1, \sigma^2$ are known, this is easy to do:

$$\begin{aligned} Y_L &= \mu_x - z_{\alpha/2}\sigma, \\ Y_U &= \mu_x + z_{\alpha/2}\sigma. \end{aligned}$$

Otherwise, we estimate μ_x and σ^2 , and the prediction interval can be based on the deviation

$$D = Y_x - \hat{\mu}_x,$$

that is, the deviation of a future response Y_x from its estimated mean $\hat{\mu}_x$. At this point we note that Y_x , being some future response, is independent of the data used to estimate $\hat{\mu}_x$. This means Y_x and $\hat{\mu}_x$ are independent, so that the variance of D is

$$\begin{aligned} \text{var}(D) &= \text{var}(Y_x) + \text{var}(\hat{\mu}_x) \\ &= \sigma^2 + \sigma_{\hat{\mu}_x}^2 \\ &= \sigma^2 \left[1 + \frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right] \end{aligned}$$

making use of Equation (4.2). This leads to level $(1 - \alpha)$ prediction interval

$$\hat{\mu}_x \pm t_{n-2,\alpha/2} \times \hat{\sigma} \left[1 + \frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right]^{1/2}.$$

4.1.4 Calculations based on sums of squares

Despite the apparent complexity of the computations associated with linear regression, they can be organized around the 5 quantities

$$\sum_{i=1}^n X_i, \quad \sum_{i=1}^n Y_i, \quad \sum_{i=1}^n X_i^2, \quad \sum_{i=1}^n Y_i^2, \quad \sum_{i=1}^n X_i Y_i$$

from which we derive quantities

$$\begin{aligned} \bar{X} &= \frac{\sum_{i=1}^n X_i}{n} \\ \bar{Y} &= \frac{\sum_{i=1}^n Y_i}{n} \\ SS_X &= \sum_{i=1}^n (X_i - \bar{X})^2 = \sum_{i=1}^n X_i^2 - n^{-1} \left(\sum_{i=1}^n X_i \right)^2 \\ SS_Y &= \sum_{i=1}^n (Y_i - \bar{Y})^2 = \sum_{i=1}^n Y_i^2 - n^{-1} \left(\sum_{i=1}^n Y_i \right)^2 \\ SS_{XY} &= \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}) = \sum_{i=1}^n X_i Y_i - n^{-1} \left(\sum_{i=1}^n X_i \right) \left(\sum_{i=1}^n Y_i \right). \end{aligned}$$

The relevant quantities then become

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} = \frac{SS_{XY}}{SS_X}, \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}, \\ \hat{\mu}_x &= \hat{\beta}_0 + \hat{\beta}_1 x.\end{aligned}$$

This means estimates are most conveniently calculated in the order $\hat{\beta}_1$, $\hat{\beta}_0$ and $\hat{\mu}_x$ as required.

We next calculate SSE and $SSTO$, following which we may calculate any required standard errors. First

$$SSTO = SS_Y.$$

Although the calculation of SSE is not, at first, as straightforward, it can be shown that

$$SSE = \sum_{i=1}^n Y_i^2 - \hat{\beta}_0 \sum_{i=1}^n Y_i - \hat{\beta}_1 \sum_{i=1}^n X_i Y_i.$$

giving

$$\hat{\sigma}^2 = MSE = \frac{SSE}{n-2}$$

and coefficient of determination

$$R^2 = 1 - \frac{SSE}{SSTO}.$$

At this point we may calculate standard errors:

$$S_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{SS_X}},$$

$$S_{\hat{\mu}_x} = \hat{\sigma} \sqrt{\frac{1}{n} + \frac{(x - \bar{X})^2}{SS_X}}$$

and

$$S_{\hat{\beta}_0} = \hat{\sigma} \sqrt{\frac{1}{n} + \frac{\bar{X}^2}{SS_X}},$$

with $(1 - \alpha)$ prediction interval for Y_x

$$\hat{\mu}_x \pm t_{n-2,\alpha/2} \times \hat{\sigma} \left[1 + \frac{1}{n} + \frac{(x - \bar{X})^2}{SS_X} \right]^{1/2}.$$

Example 4.1. The following example is due to Devore, Probability and Statistics for Engineering and the Sciences, 1995, (Example 12.10). Suppose we are given data ($n = 11$):

X = 16.1 31.5 21.5 22.4 20.5 28.4 30.3 25.6 32.7 29.2 34.7

Y = 4.41 6.81 5.26 5.99 5.92 6.14 6.84 5.87 7.03 6.89 7.87

We wish to construct a CI for β_1 . We have the summary

$$\begin{aligned}\sum_{i=1}^n X_i &= 292.90 \\ \sum_{i=1}^n Y_i &= 69.03 \\ \sum_{i=1}^n X_i^2 &= 8141.75 \\ \sum_{i=1}^n Y_i^2 &= 442.1903 \\ \sum_{i=1}^n X_i Y_i &= 1890.200.\end{aligned}$$

We then have

$$\begin{aligned}\bar{X} &= 292.9/11 = 26.627 \\ \bar{Y} &= 69.03/11 = 6.275 \\ SS_X &= 8141.75 - (292.9^2)/11 = 342.622 \\ SS_Y &= 442.1903 - (69.03^2)/11 = 8.996 \\ SS_{XY} &= 1890.20 - 292.9 * 69.03/11 = 52.119,\end{aligned}$$

Giving coefficient estimates:

$$\begin{aligned}\hat{\beta}_1 &= \frac{SS_{XY}}{SS_X} = \frac{52.119}{342.622} = 0.152 \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X} = 6.275 - 0.1520 \times 26.627 = 2.228.\end{aligned}$$

The next step is to calculate SSE :

$$\begin{aligned}SSE &= \sum_{i=1}^n Y_i^2 - \hat{\beta}_0 \sum_{i=1}^n Y_i - \hat{\beta}_1 \sum_{i=1}^n X_i Y_i \\ &= 442.1903 - 2.228 \times 69.03 - 0.152 \times 1890.200 \\ &= 1.08,\end{aligned}$$

then

$$\hat{\sigma}^2 = SSE/(n - 2) = 1.08/9 = 0.12.$$

Then

$$S_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{SS_X}} = \frac{\sqrt{0.12}}{\sqrt{342.622}} = 0.019.$$

A level 95% confidence interval is then

$$0.152 \pm t_{9,\alpha/2} S_{\hat{\beta}_1} = 0.152 \pm 2.262 \times 0.019 = 0.152 \pm 0.042.$$

To calculate the coefficient of determination we set

$$SSTO = SS_Y = 8.992$$

so that

$$R^2 = 1 - SSE/SSTO = 1 - 1.08/8.992 = 0.88.$$

The high value for R^2 is evident in Figure 4.1. \square

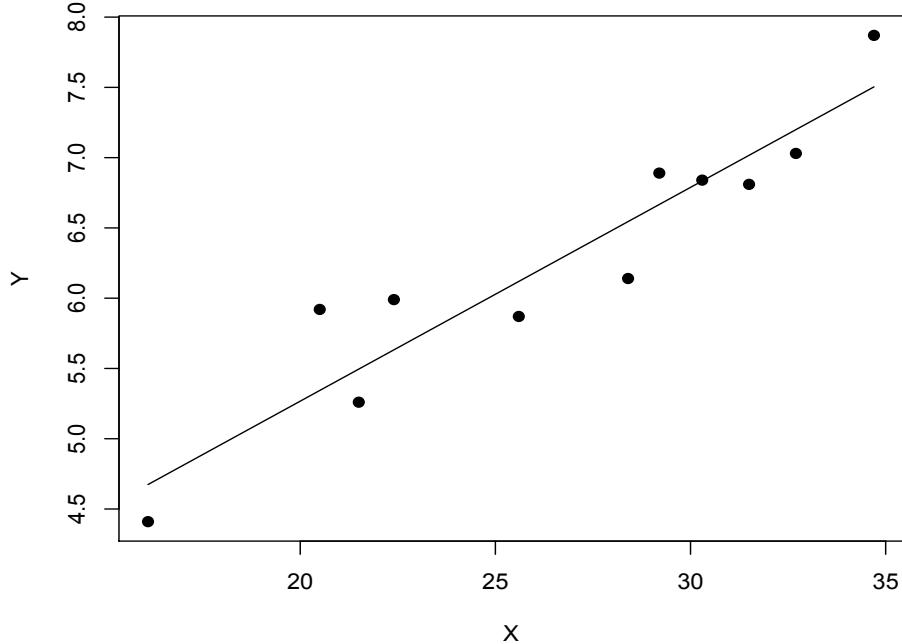


Figure 4.1: Scatter plot and regression fit for Example 4.1

4.2 Multiple Linear Regression

In contrast with simple regression, *multiple regression* permits $q \geq 1$ predictors:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \dots + \beta_q X_{qi} + \epsilon_i, \quad i = 1, \dots, n. \quad (4.8)$$

The predictors X_{ji} use a double subscript notation, where j refers to the predictor, and i refers to the sample. So, instead of observations in pairs (Y_i, X_i) for simple linear regression, observations come in the form $(Y_i, X_{1i}, X_{2i}, \dots, X_{qi})$ for $i = 1, \dots, n$. In the context of multiple regression, it is usually the practice to refer to the j th predictor as X_j , on the understanding that a second subscript is needed to refer to the actual data. As in simple linear regression, the error terms $\epsilon_1, \dots, \epsilon_n$ are a random sample from $N(0, \sigma^2)$, so that

$$Y_i \sim N(\mu_i, \sigma^2)$$

where

$$\mu_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \dots + \beta_q X_{qi}, \quad i = 1, \dots, n$$

defines the *linear regression function*.

For each coefficient β_i we may obtain a *least squares estimate* $\hat{\beta}_i$ and standard error $S_{\hat{\beta}_i}$. Their calculation requires techniques from matrix algebra which are beyond the scope of this course, so we rely on statistical computing. As in simple linear regression we have predicted, or fitted, values

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \hat{\beta}_2 X_{2i} + \dots + \hat{\beta}_q X_{qi}, \quad i = 1, \dots, n,$$

residuals

$$e_i = Y_i - \hat{Y}_i, \quad i = 1, \dots, n,$$

error sum of squares

$$SSE = \sum_{i=1}^n e_i^2,$$

and total sum of squares

$$SSTO = SS_Y = \sum_{i=1}^n (Y_i - \bar{Y})^2.$$

The regression sum of squares is similarly obtained from the equality

$$SSTO = SSR + SSE.$$

We also have the various mean sums of squares. The degrees of freedom associated with SSTO remains $n - 1$, but for SSE it is now $n - (q + 1)$, and for SSR it is q , giving

$$\begin{aligned} MSE &= \frac{SSE}{n - (q + 1)}, \\ MSR &= \frac{SSR}{q}. \end{aligned}$$

As in the simple linear regression case, an estimate of the error variance σ^2 is given by

$$\sigma^2 \approx \hat{\sigma}^2 = MSE.$$

Confidence intervals for each coefficient β_j are given by

$$\hat{\beta}_j \pm t_{n-(q+1), \alpha/2} \times S_{\hat{\beta}_j}.$$

A test against null hypothesis

$$H_o : \beta_j = \beta'_j$$

can be based on test statistic

$$T = \frac{\hat{\beta}_j - \beta'_j}{S_{\hat{\beta}_j}},$$

which under H_o has a *t*-distribution with $n - (q + 1)$ degrees of freedom. If the null hypothesis $H_o : \beta_j = 0$ can be rejected, we may conclude that the response depends on the j th predictor X_j (in addition, possibly, to other predictors). Otherwise, there is no relationship between X_j and the response, and this predictor need not be included in the model (we usually include β_0 in the model without the need of any formal inference).

4.2.1 ANOVA tables for multiple linear regression

The ANOVA table extends naturally to the multiple linear regression case:

Source	SS	df	MS	
Regression	SSR	q	$MSR = \frac{SSR}{q}$	$F = \frac{MSR}{MSE}$
Error	SSE	$n - (q + 1)$	$MSE = \frac{SSE}{n-(q+1)}$	
Total	SSTO	$n - 1$		

Here F has an F -distribution with q numerator and $n - (q + 1)$ denominator degrees of freedom under the hypothesis

$$H_0 : \beta_i = 0, i = 1, \dots, q.$$

Note that this hypothesis does not specify that the intercept β_0 is 0.

In the context of multiple regression the *coefficient of multiple determination* is

$$R^2 = 1 - \frac{SSE}{SSTO} = \frac{SSR}{SSTO}.$$

This definition is equivalent to the coefficient of determination defined for simple linear regression, but the alternative terminology emphasizes the influence on R^2 of the number of parameters in the model.

4.2.2 Full and reduced models

Before we consider an actual example, it is important to understand the concept of the *full and reduced models*. Suppose we begin with the model (4.8) with q predictors. If any coefficient β_i is actually 0, there is no need to include it in the model. Of course, we don't know the exact value of β_i , but we might conclude on a statistical basis that it is not significantly different from 0, and so on that basis we can decide which predictors to keep in the model. It might seem that all we need to do is test each coefficient separately, keeping only those for which the null hypothesis $H_0 : \beta_i = 0$ is rejected. There are two concerns with this approach. First, two predictors may be correlated with each other. When this happens, the respective coefficients may become difficult to interpret independently. In this case it is better to assess the predictive ability of the model as a whole. In addition, separate inferences formally require multiple testing procedures, the application of which can be cumbersome when used to select predictors for inclusion.

One basic tool for *model selection* (that is, the problem of deciding which predictors to retain in a model) is the F -test for groups of predictors. We'll refer to (4.8) as the *full model* (in a more compact form)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_q X_q + \epsilon \text{ Full Model.} \quad (4.9)$$

For some $p < q$ we have the *reduced model*

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon \text{ Reduced Model,} \quad (4.10)$$

that is, the reduced model is obtained from the full model by removing the final $q - p$ predictors X_{p+1}, \dots, X_q . We say such models are *nested models*. The motivation here is to determine whether or not the predictive ability of the reduced model can be improved by adding these final predictors

(there may be more than one of these). Of course, we have assumed that the predictors have been indexed appropriately.

Concluding that the full model is more predictive than the reduced model is equivalent to rejecting the hypothesis

$$H_o : \beta_{p+1} = \beta_{p+2} = \dots = \beta_q = 0$$

in favor of

$$H_a : \text{at least one of } \beta_{p+1}, \beta_{p+2}, \dots, \beta_q \text{ is not zero.}$$

The relevant F statistic is

$$F = \frac{(SSE_p - SSE_q)/(q - p)}{SSE_q/(n - (q + 1))}$$

Where SSE_q and SSE_p are the error sums of squares of the full and reduced model respectively. Under H_o F has an F -distribution with $q - p$ numerator degrees of freedom and $n - (q + 1)$ denominator degrees of freedom, and so can be rejected at significance level α if

$$F_{obs} \geq F_{q-p, n-(q+1), \alpha}.$$

It is important to note that we may set $p = 0$, in which case the reduce model is simply

$$Y = \beta_0 + \epsilon,$$

that is, responses are a random sample from $N(\beta_0, \sigma^2)$ and are not related to any of the predictors (this is why β_0 is usually retained in the model). In fact, the F -statistic $F = MSR/MSE$ given in the ANOVA table is the relevant test statistic, that is, it tests against the null hypothesis

$$H_o : \beta_1 = \beta_2 = \dots = \beta_q = 0,$$

as we have already seen.

The coefficient of multiple determination R^2 must always be interpreted carefully, since it may be shown that its value is always larger for a full model than for a (nested) reduced model, even when the relevant coefficients are truly zero. This gives the often false impression that appending a new predictor to a model improves its predictive ability. Whether or not an increase in R^2 is truly significant can be resolved by the appropriate F -test.

For this reason, we often use instead the *adjusted R*²:

$$R_{adj}^2 = 1 - \frac{SSE/(n - (q + 1))}{SSTO/(n - 1)}.$$

This value, in a sense, is adjusted for the number of parameters, and permits a more accurate comparison between models with differing numbers of parameters, which need not be nested.

4.2.3 Example

Consider the following output for two regression models involving independent variables

birthwt (weight of infant at birth)

headcirc (head circumference of infant at birth)

length (length of infant at birth)

toxemia (= 1 if toxins present in blood, = 0 otherwise)

The objective is to estimate an infants prenatal or neonatal weight based on various measurements, which would be observable with a sonogram. The full model would be

$$\text{birthwt} = \beta_0 + \beta_1 \times \text{headcirc} + \beta_2 \times \text{length} + \beta_3 \times \text{toxemia} + \epsilon,$$

which has $SSE(\text{full}) = 1647237.79$ with $q = 3$ predictors. There is also a reduced model

$$\text{birthwt} = \beta_0 + \beta_1 \times \text{headcirc} + \epsilon,$$

with $SSE(\text{reduced}) = 2611443.88$ with $p = 1$ predictors. The sample size was $n = 100$. Model summaries are given below.

To test hypothesis

$$H_o : \beta_2 = \beta_3 = 0$$

against

$$H_a : \text{at least one of } \beta_2, \beta_3 \text{ is not zero}$$

we use F -statistic

$$\begin{aligned} F &= \frac{(SSE(\text{reduced}) - SSE(\text{full}))/2}{SSE(\text{full})/(100 - 4)} \\ &= \frac{(2611443.88 - 1647237.79)/2}{1647237.79/(100 - 4)} \\ &= 28.097. \end{aligned}$$

Under the null distribution H_o , F has an F distribution with numerator and denominator degrees of freedom $\nu_{\text{num}} = q - p = 2$ and $\nu_{\text{den}} = n - (q + 1) = 96$. The p -value is very small, say $P < 0.001$, since we have critical value $F_{2,96,0.001} = 7.43$. So, the full model is more predictive than the reduced model.

Summary for reduced model:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
headcirc	1	4605298.87	4605298.87	172.82	0.0000
Residuals	98	2611443.88	26647.39		

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1154.1087	172.1523	-6.70	0.0000
headcirc	85.1780	6.4793	13.15	0.0000

Summary for full model:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
headcirc	1	4605298.87	4605298.87	268.39	0.0000
length	1	889039.76	889039.76	51.81	0.0000
toxemia	1	75166.33	75166.33	4.38	0.0390
Residuals	96	1647237.79	17158.73		

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1567.6048	148.7759	-10.54	0.0000
headcirc	48.3632	7.4349	6.50	0.0000
length	38.0639	5.2565	7.24	0.0000
toxemia	-67.9216	32.4518	-2.09	0.0390

Chapter 5

Linear Regression - Modeling in R

The chapter has two objectives. The first is to introduce R as a tool for statistical modeling. While this is carried out using linear regression, many of the methods are equally applicable to most other types of statistical models that one would encounter in an intermediate course on statistical methodology. For this reason, this chapter also introduces, mainly by example, some new modeling techniques which are of interest on their own.

5.1 Statistical Models

In a *statistical model* a random response Y is dependent on *predictors* X_1, X_2, \dots, X_m , in the sense that the distribution of Y depends on X_1, \dots, X_k . In many frequently used models, the relationship is given by

$$Y = \mu(X_1, X_2, \dots, X_m) + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \quad (5.1)$$

or equivalently,

$$Y \sim N(\mu(X_1, X_2, \dots, X_m), \sigma^2).$$

ANOVA is a simple example of this. There is a single predictor X , which is a *factor*, or categorical variable, which assumes levels $1, \dots, k$ (that is, there are k treatments, or groups). In this case, there are k means μ_1, \dots, μ_k , so that

$$\begin{aligned} Y &= \mu(X) + \epsilon \\ &= \mu_X + \epsilon \\ &= \mu_i + \epsilon \text{ if } X = i. \end{aligned}$$

Linear regression is a somewhat more complex example, but also conforms to Equation (5.1):

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_m X_m + \epsilon. \quad (5.2)$$

5.2 The Formula Object in R

R supports a specialized notation for statistical models, based on the `formula` class. We have already seen a number of examples. In the following script, a data set consisting of a numerical vector `color.value` and a character vector `color.type` is created. There are 26 records, with

equal numbers of “Red” and “Green” color types. The intention is that “Green” types tend to have higher values. The resulting plot is shown in Figure 5.1.

```
> par(mfrow=c(1,2),cex=1.0)
> color.value = rnorm(26,mean=rep(c(10,12),13))
> color.type = rep(c("Red","Green"),13)
> boxplot(color.value,ylab='Color Value')
> boxplot(color.value ~ color.type,ylab='Color Value')
```

The command `boxplot(color.value, ylab='Color Value')` creates a single boxplot of all the data, while the command `boxplot(color.value ~ color.type, ylab='Color Value')` creates side-by-side boxplots for each color type.

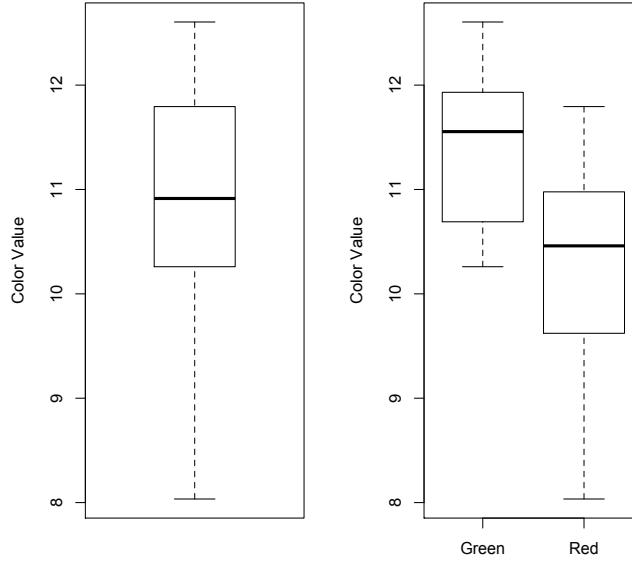


Figure 5.1: Example of the use of the `formula` class in the `boxplot`

The expression `color.value ~ color.type` within the final `boxplot` command is an example of a `formula`, which takes the general form

`response ~ predictor expression`

It’s exact effect depends on the context. In it’s simplest form, as in the boxplot example of Figure 5.1, it separates a set of measurements by a group variable. However, it can also describe an analytical relationship between the response and multiple predictors.

The following script demonstrates the creation of a `formula` object, with the symbol `~` separating the response from the predictors:

```
> f1 = y ~ x
> f1
y ~ x
```

```
> class(f1)
[1] "formula"
```

The multiple regression model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \quad (5.3)$$

would be represented

```
> f1 = y ~ x1 + x2
> f1
y ~ x1 + x2
```

We will make use later in the chapter of the *interaction term*, which is a predictor formed by taking the product of two or more other predictor terms. When interactions are present in a model, the predictors forming the interaction are referred to as *main effects*. In Equation (5.3) there is one possible interaction term $X_1 X_2$, leading to regression equation

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2.$$

To specify all possible interactions and main effects involving two predictors the “*” operator can be used:

```
> f2 = y ~ x1 * x2
> f2
y ~ x1 * x2
> terms(f2)
y ~ x1 * x2
attr(,"variables")
list(y, x1, x2)
attr(,"factors")
  x1 x2 x1:x2
y  0  0    0
x1 1  0    1
x2 0  1    1
attr(,"term.labels")
[1] "x1"    "x2"    "x1:x2"
attr(,"order")
[1] 1 1 2
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
<environment: R_GlobalEnv>
```

Note that the function `terms()` is used to extract details of a formula.

If we wanted to include only the main effect for X_1 and the interaction term, for example:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1 X_2$$

we could use the operator “:” to generate only the specified interactions.

```
> f3 = y ~ x1 + x1:x2
> f3
y ~ x1 + x1:x2
> terms(f3)
y ~ x1 + x1:x2
attr(,"variables")
list(y, x1, x2)
attr(,"factors")
  x1 x1:x2
y   0      0
x1  1      2
x2  0      1
attr(,"term.labels")
[1] "x1"     "x1:x2"
attr(,"order")
[1] 1 2
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
<environment: R_GlobalEnv>
```

The intercept term is implicitly included in a formula (but can be removed if needed). If we want to model to include only the intercept (for example, for model comparisons), we use the symbol 1:

```
> f4 = y ~ 1
> f4
y ~ 1
> terms(f4)
y ~ 1
attr(,"variables")
list(y)
attr(,"factors")
integer(0)
attr(,"term.labels")
character(0)
attr(,"order")
integer(0)
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
<environment: R_GlobalEnv>
```

This is a fairly in-depth topic, so we will discuss just the basics at first (see `help(formula)` for more detail).

5.3 Linear Regression in R

First, we note that R comes with a set of example datasets, in a package called 'MASS'

```
> library(MASS)
> help(package=MASS)
...
Functions and datasets to support Venables and Ripley,
'Modern Applied Statistics with S' (4th edition, 2002).
...
```

One of these datasets is called `nlschools`:

```
> help(nlschools)
Description
```

Snijders and Bosker (1999) use as a running example a study of 2287 eighth-grade pupils (aged about 11) in 132 classes in 131 schools in the Netherlands. Only the variables used in our examples are supplied.

Usage

```
nlschools{1.0\textwidth}
Format
```

This data frame contains 2287 rows and the following columns:

```
lang
language test score.
```

```
IQ
verbal IQ.
```

```
class
class ID.
```

```
GS
class size: number of eighth-grade pupils recorded in the class
(there may be others: see COMB, and some may have been omitted
with missing values).
```

SES

social-economic status of pupil's family.

COMB

were the pupils taught in a multi-grade class (0/1)? Classes which contained pupils from grades 7 and 8 are coded 1, but only eighth-graders were tested.

Source

Snijders, T. A. B. and Bosker, R. J. (1999) Multilevel Analysis. An Introduction to Basic and Advanced Multilevel Modelling. London: Sage.

References

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

The object `nlschools` is a data frame. You can verify that it is a list (a data frame is a list) using the `is.list()` function. The names of the variables is obtained using the `names()` command.

```
> is.list(nlschools)
[1] TRUE
> names(nlschools)
[1] "lang"   "IQ"     "class"  "GS"     "SES"    "COMB"
> nlschools[1:5,]
  lang   IQ class GS SES COMB
1  46 15.0  180 29  23    0
2  45 14.5  180 29  10    0
3  33  9.5  180 29  15    0
4  46 11.0  180 29  23    0
5  20  8.0  180 29  10    0
> dim(nlschools)
[1] 2287     6
>
```

There are 2287 rows and 6 columns.

Note that in the dataset `nlschols`, `COMB` is an *indicator variable*, that is, a variable that assumes only values 0, 1 (or, a factor with two levels). We can do a *t*-test to see if there is a significant difference in language test scores between students in multigrade classes, and those not in multigrade classes. We can use model notation, but we need to specify the data frame with the `data` option.

```
> t.test(lang ~ COMB, data=nlschools)
```

Welch Two Sample t-test

```

data: lang by COMB
t = 5.3849, df = 991.978, p-value = 9.052e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.540178 3.306355
sample estimates:
mean in group 0 mean in group 1
 41.60133      39.17806

```

So, for $\text{COMB} = 0$ (not in multigrade class) the estimated mean score is $\hat{\mu}_0 = 41.6 \approx \mu_0$ and for $\text{COMB} = 1$ (in multigrade class) the estimated mean score is $\hat{\mu}_1 = 39.2 \approx \mu_1$. There is a significant detrimental effect (about 2.4 points) on language test scores attributable to presence in multigrade class.

Next, suppose we consider regression model

$$\text{lang} = \beta_0 + \beta_1 \times \text{COMB} + \epsilon$$

Since COMB is an indicator variable, we can match the regression coefficients directly to the two group means:

$$\begin{aligned}\mu_0 &= \beta_0 \\ \mu_1 &= \beta_0 + \beta_1, \text{ with estimates} \\ \hat{\mu}_0 &= \hat{\beta}_0 \\ \hat{\mu}_1 &= \hat{\beta}_0 + \hat{\beta}_1.\end{aligned}$$

In R, regression fits can be calculated using the `lm()` function, using the model notation:

```
> fit = lm(lang ~ COMB, data=nlschools)
> summary(fit)
```

Call:

```
lm(formula = lang ~ COMB, data = nlschools)
```

Residuals:

Min	1Q	Median	3Q	Max
-30.1781	-6.1781	0.8219	7.3987	18.8219

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	41.6013	0.2196	189.472	< 2e-16 ***
COMB1	-2.4233	0.4187	-5.788	8.1e-09 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.94 on 2285 degrees of freedom

Multiple R-squared: 0.01445, Adjusted R-squared: 0.01402

F-statistic: 33.5 on 1 and 2285 DF, p-value: 8.1e-09

We have coefficient estimates $\hat{\beta}_0 = 41.6013$ and $\hat{\beta}_1 = -2.4233$, giving

$$\begin{aligned}\hat{\mu}_0 &= 41.6013 \\ \hat{\mu}_1 &= 41.6013 - 2.4233 = 39.178,\end{aligned}$$

which conform to the estimates we obtained above.

5.4 ANOVA and Linear Regression

This example is due to Johnson and Bhattacharya (*Statistics: Principles and Methods, 3rd Edition*). In an effort to improve the quality of recording tapes, the effects of four kinds of coatings A , B , C and D on reproduction quality are assessed by applying each to a separate sample of tape and measuring the resulting distortion. The results are given in the following table.

Coating	Sample	Sample Mean	Sum of Squares
A	10, 15, 8, 12, 15	$\bar{y}_1 = 12$	$\sum_{i=1}^5 (y_{1i} - \bar{y}_1)^2 = 38$
B	14, 18, 21, 15	$\bar{y}_2 = 17$	$\sum_{i=1}^4 (y_{2i} - \bar{y}_2)^2 = 30$
C	17, 16, 14, 15, 17, 15, 18	$\bar{y}_3 = 16$	$\sum_{i=1}^7 (y_{3i} - \bar{y}_3)^2 = 12$
D	12, 15, 17, 15, 16, 15	$\bar{y}_4 = 15$	$\sum_{i=1}^6 (y_{4i} - \bar{y}_4)^2 = 14$

We can create a data frame for the data in the following way:

```
> y1 = c(10, 15, 8, 12, 15)
> y2 = c(14, 18, 21, 15)
> y3 = c(17, 16, 14, 15, 17, 15, 18)
> y4 = c(12, 15, 17, 15, 16, 15)
> y = c(y1,y2,y3,y4)
> gr = c(rep("A",5), rep("B",4), rep("C",7), rep("D",6) )
>
> tapes.data = data.frame(y,gr)
> tapes.data
   y gr
1 10 A
2 15 A
3  8 A
4 12 A
5 15 A
6 14 B
7 18 B
8 21 B
9 15 B
10 17 C
11 16 C
12 14 C
13 15 C
```

```

14 17 C
15 15 C
16 18 C
17 12 D
18 15 D
19 17 D
20 15 D
21 16 D
22 15 D
>

```

The variable y contains the responses, with treatment groups indicators by the factor variable gr . As in the previous example, we can express the ANOVA model as a linear regression model using indicator variables:

$$Y = \beta_0 + \beta_1 \times I_B + \beta_2 \times I_C + \beta_3 \times I_D + \epsilon$$

where, for example, I_B is the indicator variable for treatment B . Note that we don't need (or want) an indicator variable for treatment A . The coefficients can be related to the treatment means in the following way:

$$\begin{aligned}\mu_A &= \beta_0 \\ \mu_B &= \beta_0 + \beta_1 \\ \mu_C &= \beta_0 + \beta_2 \\ \mu_D &= \beta_0 + \beta_3,\end{aligned}$$

with estimates:

$$\begin{aligned}\hat{\mu}_A &= \hat{\beta}_0 \\ \hat{\mu}_B &= \hat{\beta}_0 + \hat{\beta}_1 \\ \hat{\mu}_C &= \hat{\beta}_0 + \hat{\beta}_2 \\ \hat{\mu}_D &= \hat{\beta}_0 + \hat{\beta}_3.\end{aligned}$$

As can be seen, we only need indicators variables for 3 of the 4 treatments. To implement this using `lm()`, we could construct the indicator variables, but the same effect can be achieved using a factor variable.

```

> fit = lm(y ~ gr, data=tapes.data)
> summary(fit)

```

Call:

```
lm(formula = y ~ gr, data = tapes.data)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.00	-1.75	0.00	1.00	4.00

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	12.000	1.022	11.742	7.16e-10 ***
grB	5.000	1.533	3.262	0.00433 **
grC	4.000	1.338	2.989	0.00787 **
grD	3.000	1.384	2.168	0.04381 *

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

Residual standard error: 2.285 on 18 degrees of freedom

Multiple R-squared: 0.4198, Adjusted R-squared: 0.323

F-statistic: 4.34 on 3 and 18 DF, p-value: 0.01814

>

The coefficients match the model with

$$\begin{aligned}\hat{\beta}_0 &= 12.0 \\ \hat{\beta}_1 &= 5.0 \\ \hat{\beta}_2 &= 4.0 \\ \hat{\beta}_3 &= 3.0,\end{aligned}$$

and we can recreate the treatment mean estimates

$$\begin{aligned}\hat{\mu}_A &= \hat{\beta}_0 = 12.0 \\ \hat{\mu}_B &= \hat{\beta}_0 + \hat{\beta}_1 = 12.0 + 5.0 = 17.0 \\ \hat{\mu}_C &= \hat{\beta}_0 + \hat{\beta}_2 = 12.0 + 4.0 = 16.0 \\ \hat{\mu}_D &= \hat{\beta}_0 + \hat{\beta}_3 = 12.0 + 3.0 = 15.0.\end{aligned}$$

In addition, the F statistic $F = 4.34$ is equivalent to that obtained by the ANOVA procedure, as is the F test for difference in means itself.

5.5 Residuals and lm()

The output of `lm()` is a list:

```
> names(fit)
[1] "coefficients"   "residuals"      "effects"       "rank"
"fitted.values"    "assign"        "qr"
"df.residual"      "contrasts"     "xlevels"
[11] "call"          "terms"         "model"
```

One of the components of this list is `residuals`, which is a vector of the residuals $e_i = Y_i - \hat{Y}_i$. We can, for example, examine the normality of the residuals with a normal quantile plot (Figure 5.2):

```
> qqnorm(fit$residual)
> qqline(fit$residual)
```

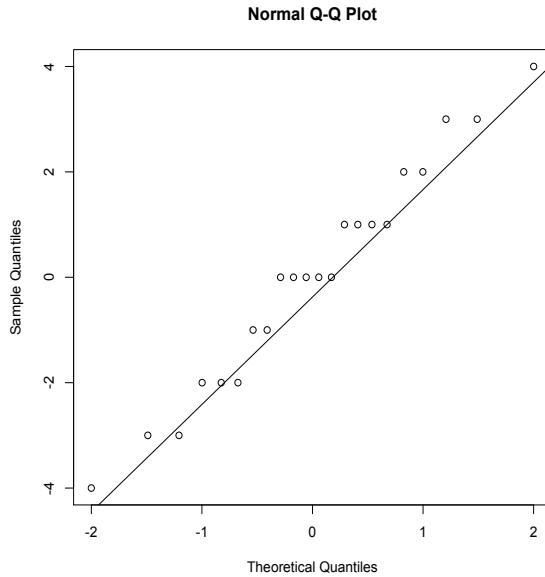


Figure 5.2: Normal quantile plot for Section 5.5.

The quantile plot suggests that the assumption of normality is reasonable.

5.6 Interaction Terms

We'll return to the `nlschools` data, and fit the model

$$\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \epsilon$$

The following script will fit the model, store the coefficients in a vector `cf`, do a scatter-plot of the independent against the dependent variable, then superimpose the actual regression line (Figure 5.3).

```
> fit = lm(lang ~ IQ, data = nlschools)
> summary(fit)
```

Call:

```
lm(formula = lang ~ IQ, data = nlschools)
```

Residuals:

Min	1Q	Median	3Q	Max
-28.7022	-4.3944	0.6056	5.2595	26.2212

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	9.52848	0.86682	10.99	<2e-16 ***
IQ	2.65390	0.07215	36.78	<2e-16 ***

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 7.137 on 2285 degrees of freedom
Multiple R-squared: 0.3719, Adjusted R-squared: 0.3716
F-statistic: 1353 on 1 and 2285 DF, p-value: < 2.2e-16
```

```
> cf = fit$coefficients
> cf
(Intercept)      IQ
  9.528484    2.653896
> range(nlschools$IQ)
[1] 4 18
> plot(nlschools$IQ, nlschools$lang, pch=20)
> lines(range(nlschools$IQ),
        cf[1] + cf[2]*range(nlschools$IQ), lwd=2)
>
```

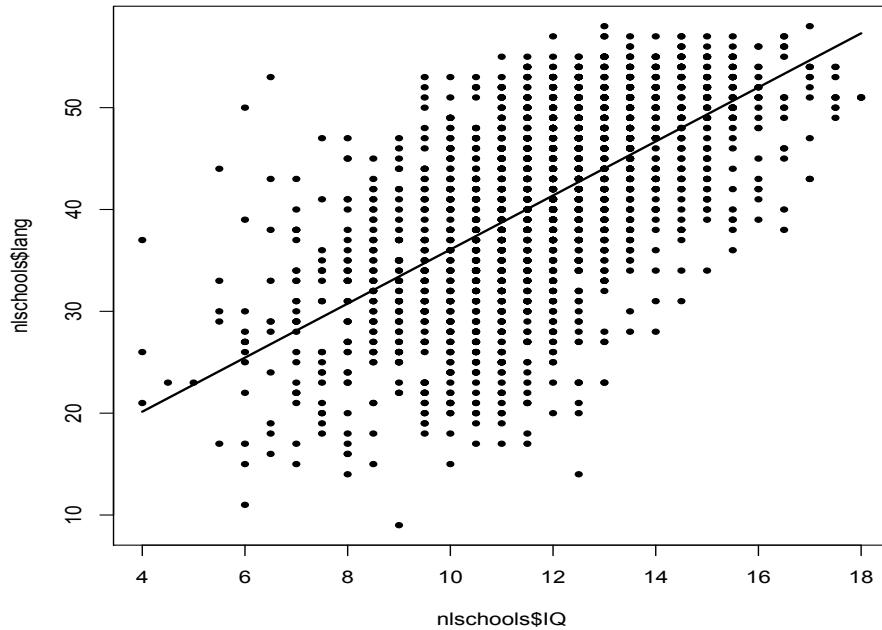


Figure 5.3: Regression fit for model $\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \epsilon$

The following script can be used to produce diagnostic plots. Note that `par(mfrow=c(1,2))` permits two plots to appear on one window. The resulting residual plot and residual normal quantile plot appear in Figure 5.4. Note that these plots may also be obtained using `plot(fit)`. This type of feature is generally available for models in R.

```
>
> par(mfrow=c(1,2))
```

```
> plot(fit$fitted.values, fit$residuals, pch=20)
> lines(c(-100,100), c(0,0))
> qqnorm(fit$residuals)
> qqline(fit$residuals)
>
```

Next, recall that the variable `COMB` had a significant effect on the language scores, so we may wish to introduce it into our model. First, remember to change the graphics window properties if needed (here, we only want one plot, so use `par(mfrow=c(1,1))`). We now have model

$$\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \beta_2 \times \text{COMB} + \epsilon$$

where `COMB` is an indicator variable. However, we can consider this as two linear regression models, one for multigrade classes, and one for single grade classes. We can then superimpose two fits on one plot, in particular $y = \beta_0 + \beta_1 x$ for single grade classes, and $y = (\beta_0 + \beta_2) + \beta_1 x$ for multigrade classes.

```
> par(mfrow=c(1,1))
> fit2 = lm(lang ~ IQ + COMB, data = nlschools)
> summary(fit2)

Call:
lm(formula = lang ~ IQ + COMB, data = nlschools)

Residuals:
    Min      1Q  Median      3Q     Max 
-27.3890 -4.4989  0.5011  5.1841 25.6214 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 10.25824   0.87212 11.76 < 2e-16 ***
IQ           2.63390   0.07181 36.68 < 2e-16 ***
COMB1       -1.79296   0.33265 -5.39 7.77e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.094 on 2284 degrees of freedom
Multiple R-squared:  0.3798, Adjusted R-squared:  0.3792 
F-statistic: 699.2 on 2 and 2284 DF,  p-value: < 2.2e-16
```

```
> cf = fit2$coefficients
> cf
(Intercept)          IQ          COMB1
  10.258240    2.633900   -1.792956
> plot(nlschools$IQ, nlschools$lang, pch=20)
> lines(range(nlschools$IQ),
```

```

cf[1] + cf[2]*range(nlschools$IQ), lwd=2, col=2)
> lines(range(nlschools$IQ),
      cf[1] + cf[3] + cf[2]*range(nlschools$IQ), lwd=2, col=3)
> legend(14,20,
      legend=c("Multigrade class", "Single grade class"),
      lty=c(1,1), col=c(3,2))
>

```

Note here the use of the `col` option in `plot()` to color lines, thus distinguishing the groups. Also, the `lwd` option controls the width of the line, and `pch` defines the plotting symbol type. The `legend()` function is then used to add a legend. Compare the magnitude of the `COMB` effect of -1.79296, to the `COMB` effect obtained by the two sample mean comparison (also obtained by the simple regression fit $\text{lang} = \beta_0 + \beta_1 \times \text{COMB}$) of -2.4233. The resulting plot is shown in Figure 5.5.

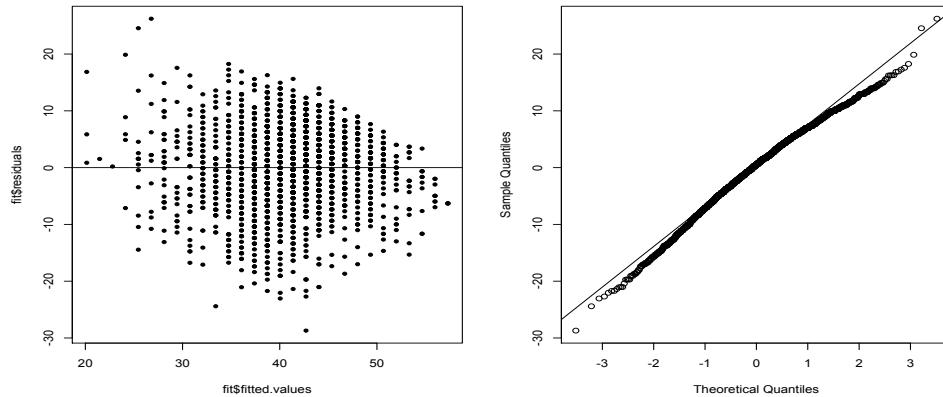


Figure 5.4: Residual plot and residual normal quantile plot for model $\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \epsilon$

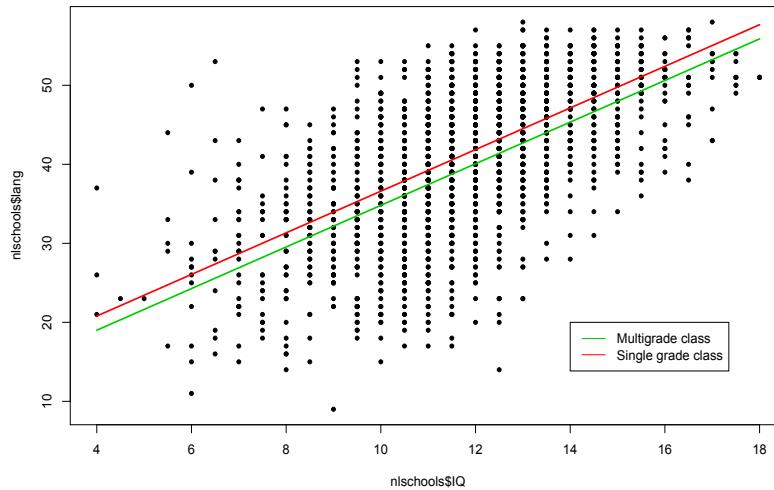


Figure 5.5: Regression fit for model $\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \beta_2 \times \text{COMB} + \epsilon$.

We may wonder, however, whether or not the slope of the regression line also differs by COMB group. To test for this, we can use *interaction terms*, which are simply products of other predictors. These are often very useful. When interactions are present, their components are referred to as *main effects*.

For example, we might fit model:

$$\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \beta_2 \times \text{COMB} + \beta_3 \times \text{IQ} \times \text{COMB} + \epsilon.$$

Here, both the intercept and slope can differ by COMB group:

$$\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \epsilon, \quad \text{for COMB} = 0$$

and

$$\text{lang} = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \text{IQ} + \epsilon, \quad \text{for COMB} = 1.$$

In other words, the intercepts and slopes differ between the two COMB groups by β_2 and β_3 respectively, therefore such differences can be tested based on null hypotheses $H_o : \beta_2 = 0$ and $H_o : \beta_3 = 0$.

We can fit this model using `lm()` (Figure 5.6). Interaction between two predictors are defined in model notion using the operator “:”. Alternatively, the operator “*” will introduce interactions and *main effects*.

```
> par(mfrow=c(1,1))
> fit2 = lm(lang ~ IQ + COMB + IQ:COMB, data = nlschools)
> summary(fit2)

Call:
lm(formula = lang ~ IQ + COMB + IQ:COMB, data = nlschools)

Residuals:
    Min      1Q  Median      3Q     Max 
-27.768 -4.484   0.473   5.153  24.646 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 12.4068    1.0228 12.131 < 2e-16 ***
IQ           2.4533    0.0847 28.966 < 2e-16 ***
COMB1       -9.2019    1.8875 -4.875 1.16e-06 ***
IQ:COMB1    0.6317    0.1584  3.987 6.90e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.071 on 2283 degrees of freedom
Multiple R-squared:  0.3841, Adjusted R-squared:  0.3832 
F-statistic: 474.5 on 3 and 2283 DF,  p-value: < 2.2e-16

> cf = fit2$coefficients
```

```

> cf
(Intercept)           IQ          COMB1      IQ:COMB1
  12.406772     2.453349    -9.201913     0.631680
> plot(nlschools$IQ, nlschools$lang, pch=20)
> lines(range(nlschools$IQ),
         cf[1] + cf[2]*range(nlschools$IQ), lwd=2, col=2)
> lines(range(nlschools$IQ),
         cf[1] + cf[3] + (cf[2]+cf[4])*range(nlschools$IQ), lwd=2, col=3)
> legend(14,20, legend=c("Multigrade class", "Single grade class"),
         lty=c(1,1), col=c(3,2))

```

Here we see that language scores differ by COMB group, but also that this difference is larger among students with lower IQs. We say that IQ *interacts* with the COMB factor. This suggests that the performance of students with higher IQs may not be as influenced by external factors as other students are.

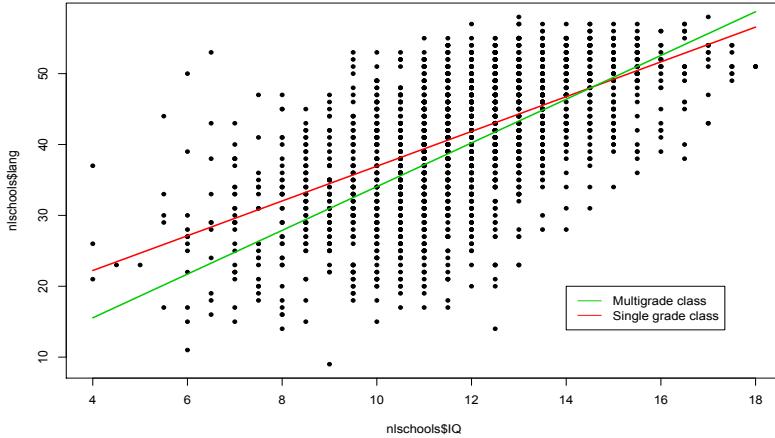


Figure 5.6: Regression fit for model $\text{lang} = \beta_0 + \beta_1 \times \text{IQ} + \beta_2 \times \text{COMB} + \beta_3 \times \text{IQ} \times \text{COMB} + \epsilon$.

5.7 Polynomial Regression

The terms *linear* in *linear regression* nominally refers to the relationship between the predictors and the response. However, this has as much to do with the form of the inference as with any functional relationship. Suppose we have a model of the form

$$Y = 10 + 2.3x - 0.2x^2 + \epsilon \quad (5.4)$$

where ϵ is the familiar error term. If we regard x as a single predictor, than Equation (5.4) conforms to (5.1) but not (5.2). On the other hand, we could also regard $X_1 = x$ and $X_2 = x^2$ as two distinct predictors, in which case (5.4) conforms to both (5.1) and (5.2), with $\beta_0 = 10$, $\beta_1 = 2.3$ and $\beta_2 = -0.2$.

Fitting this type of model using multiple linear regression is referred to as *polynomial regression*. The methodology and inference remain exactly the same, as long as the linear structure of the

inference is understood. In fact, introducing *quadratic terms* into a regression equation is a common method of both testing for nonlinear relationships between response and predictor, and for modeling such relationships when appropriate. We might first compare the full and reduced models

$$\begin{aligned} Y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon, \\ Y &= \beta_0 + \beta_1 x + \epsilon. \end{aligned}$$

In this case, if a hypothesis test is able to reject the hypothesis $H_0 : \beta_2 = 0$, then the full model could be accepted and summarized. For more complex problems of this type the methods of Section 4.2.2 are available. We should note that somewhat more mathematically sophisticated methods exist for polynomial regression, which are implemented in R. One common practice is to use $(x - \bar{x})^2$ instead of x^2 as the quadratic term, particularly when a large range in x leads to a much larger range in x^2 . Although the fitted values \hat{Y} will be identical using either form, the actual coefficient values will be different, and are usually more intuitively interpretable using the form $(x - \bar{x})^2$. The quadratic term can be introduced into an R `formula` object using the notation `I(x^2)`.

The following script simulates data from the model of Equation (5.4), using 19 equally spaced values for x ranging from 0 to 5.4. The error terms have standard deviation $\sigma = 0.5$ (how can you tell this?). The model formula is created in object `lrform`, and used directly in function `lm()`. In general, elements of a formula can refer to columns in a data frame, which would then be explicitly reference using the `data` option in the `lm()` function, as shown in the examples using the `nlschools` data frame earlier in this chapter.

```
> f0 = function(x) {10 + 2.3*x - 0.2*x^2}
>
> x = seq(0,5.5,0.3)
> xsq = x^2
>
> plot(x,f0(x))
>
> mux = f0(x)
> y = mux + rnorm(length(x))/2
> plot(x,y,pch=20,cex=1)
> lrform = y~x + I(x^2)
> lrform
y ~ x + I(x^2)
> fit = lm(lrform)
> summary(fit)
```

Call:

```
lm(formula = lrform)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.56569	-0.25812	-0.05227	0.24923	0.75776

Coefficients:

```

Estimate Std. Error t value Pr(>|t|)
(Intercept) 9.73262    0.24194 40.228 < 2e-16 ***
x           2.53668    0.20771 12.212 1.6e-09 ***
I(x^2)      -0.25088   0.03713 -6.758 4.6e-06 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3892 on 16 degrees of freedom
Multiple R-squared: 0.9701, Adjusted R-squared: 0.9663
F-statistic: 259.4 on 2 and 16 DF, p-value: 6.417e-13

> lines(x,mux,lty=2)
> lines(x,predict(fit),lty=1)
> legend('bottomright',legend=c('True Mean Response','Estimated Mean Response'),
  col=c(1,1),lty=c(2,1))
>

```

The resulting plot is shown in Figure 5.7. The fitted and true mean response curves are shown, along with the data points. The estimates $\hat{\beta}_0 = 9.73262$, $\hat{\beta}_1 = 2.53668$ and $\hat{\beta}_2 = -0.25088$ are quite close to the true values $\beta_0 = 10.0$, $\beta_1 = 2.3$ and $\beta_2 = -0.2$.

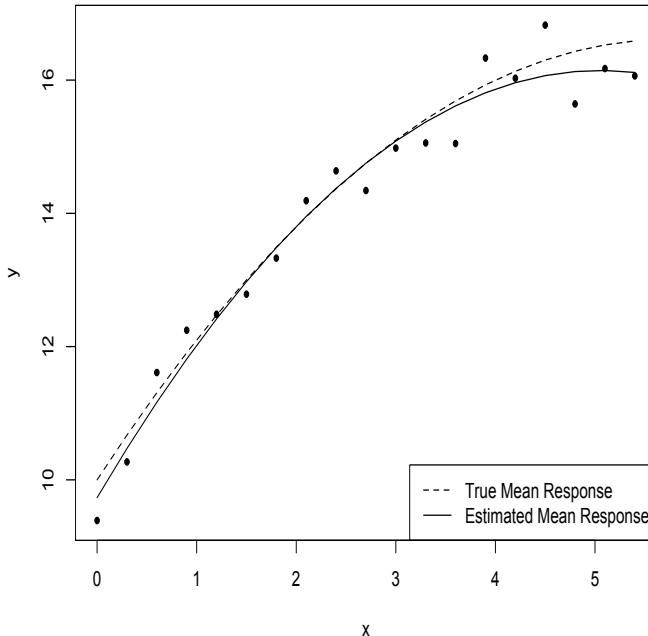


Figure 5.7: Data and linear model fit for polynomial regression example of Section 5.7.

There is, however, much more to be discussed regarding polynomial regression. We will do so in Chapter 19 after a suitable mathematical foundation has been laid.

Chapter 6

Linear Regression - Formulation Using Matrix Algebra

There will be considerable advantage to expressing linear regression in terms of linear algebra. At this point we adopt the notation of Chapter 15. The responses are \mathbf{y} and the predictors are $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_q]$. What was written before as model:

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \dots, n, \quad \epsilon_i \sim N(0, \sigma^2),$$

now becomes

$$\mathbf{y} = \beta_1 \mathbf{x}_1 + \dots + \beta_q \mathbf{x}_q + \boldsymbol{\epsilon} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ is the column vector of error terms ϵ_i . It is important to note that the intercept term has not been removed. It is now column vector $\vec{1} = [1 \cdots 1]^T$ in \mathbf{X} . This means if we have p predictors in the usual sense, then \mathbf{X} will have $q = p + 1$ column vectors, assuming the intercept is to be included.

We also note that the SSE can be expressed in matrix form as

$$SSE[\boldsymbol{\beta}] = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

6.1 Regression Coefficients $\boldsymbol{\beta}$

The least squares estimates of regression coefficients $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{6.1}$$

using standard matrix multiplication. The covariance matrix of $\hat{\boldsymbol{\beta}}$ is given by

$$\Sigma_{\hat{\boldsymbol{\beta}}} = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}. \tag{6.2}$$

Then the estimated covariance matrix is

$$S_{\hat{\boldsymbol{\beta}}}^2 = MSE \times (\mathbf{X}^T \mathbf{X})^{-1} \tag{6.3}$$

so that the standard errors referred to in Section 4.2 are given directly by

$$S_{\hat{\beta}_j} = \sqrt{MSE \times [(\mathbf{X}^T \mathbf{X})^{-1}]_{jj}}, \tag{6.4}$$

that is, the square root of the j th diagonal element of matrix (6.3).

6.2 Linear Combinations of β

One important problem in linear regression is the estimation of linear combinations of the regression coefficients. Let

$$\eta = a_1\beta_1 + \dots + a_q\beta_q = \mathbf{a}^T \boldsymbol{\beta},$$

where $\mathbf{a} = [a_1 \dots a_q]^T$ is the appropriate column vector. The obvious estimator for η is

$$\hat{\eta} = \mathbf{a}^T \hat{\boldsymbol{\beta}}. \quad (6.5)$$

It may also be shown that the standard error $S_{\hat{\eta}}$ of this estimate is given by

$$S_{\hat{\eta}}^2 = MSE \times \mathbf{a}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{a}, \quad (6.6)$$

which may be used in the inference procedures described in Section 4.2.

6.3 Fitted Values $\hat{\mathbf{y}}$

The column vector of fitted values is then expressed as, using (6.1),

$$\hat{\mathbf{y}} = \mathbf{X} \hat{\boldsymbol{\beta}} = \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{H} \mathbf{y}$$

where

$$\mathbf{H} = \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$$

is known as the ‘hat’ matrix. This matrix is symmetric. It is also *idempotent*, meaning that $\mathbf{H} = \mathbf{H}^2$. The covariance matrix of \mathbf{y} is $\Sigma_{\mathbf{y}} = \sigma^2 \mathbf{I}_n$, where \mathbf{I}_n is the $n \times n$ identity matrix, so the covariance matrix of $\hat{\mathbf{y}}$ is therefore

$$\Sigma_{\hat{\mathbf{y}}} = \mathbf{H} [\sigma^2 \mathbf{I}_n] \mathbf{H}^T = \sigma^2 \mathbf{H},$$

(see Appendix B). The variance of a single fitted values is therefore

$$\sigma_{\hat{y}_i}^2 = \sigma^2 \mathbf{H}_{ii}. \quad (6.7)$$

The standard errors are obtained by substituting MSE for σ^2 :

$$\begin{aligned} S_{\hat{y}}^2 &= MSE \times \mathbf{H}, \\ S_{\hat{y}_i}^2 &= MSE \times \mathbf{H}_{ii}. \end{aligned} \quad (6.8)$$

6.4 Residuals \mathbf{e}

It is important to realize that while $MSE^{1/2}$ is an estimate of σ , it is not an estimate of the standard deviation of $e_i = y_i - \hat{y}_i$, which is itself a linear combination of the responses \mathbf{y} . Viewed this way, the residual vector is

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = [\mathbf{I}_n - \mathbf{H}] \mathbf{y},$$

where \mathbf{I}_n is the $n \times n$ identity matrix. The covariance matrix of \mathbf{y} is $\Sigma_{\mathbf{y}} = \sigma^2 \mathbf{I}_n$, so the covariance matrix of \mathbf{e} is therefore

$$\Sigma_{\mathbf{e}} = \sigma^2 [\mathbf{I}_n - \mathbf{H}] [\mathbf{I}_n - \mathbf{H}]^T = \sigma^2 [\mathbf{I}_n - \mathbf{H}],$$

(see Appendix B) so that the variance of e_i is

$$\sigma_{e_i}^2 = \sigma^2 (1 - \mathbf{H}_{ii}), \quad (6.9)$$

with standard errors

$$\begin{aligned} S_{\mathbf{e}}^2 &= MSE \times (\mathbf{I}_n - \mathbf{H}), \\ S_{e_i}^2 &= MSE \times (1 - \mathbf{H}_{ii}). \end{aligned}$$

This is an important result, because it reveals that the variances of e_i are not equal, depending directly on the leverage \mathbf{H}_{ii} . This means a high leverage observation tends to disproportionately pull the regression line towards it. In fact, since \mathbf{H}_{ii} may be arbitrarily close to 1, from (6.9) we can see that $\sigma_{e_i}^2$ may be arbitrarily close to zero, so this effect may be quite dominant.

Chapter 7

Least Squares Regression and Vector Spaces

In the following discussion, it may be useful to keep in mind an important difference between pure and applied mathematics. Applied mathematics is usually concerned with a *program*. This may be the development of a fluid flow model, the application of a statistical estimation procedure, or the optimization of a function. The explanation of such programs follows a natural sequence of tasks. On the other hand, pure mathematics is often concerned with establishing the equivalence, or otherwise, of two mathematical objects, which may have quite different derivations. In the author's experience, this difference must be understood in order for applied mathematicians to make good use of pure mathematics. The end result should be that applied mathematics is made simpler, not more challenging.

7.1 Vector Spaces and Prediction Spaces

We first define a collection of p vectors, denoted $\tilde{\mathbf{v}} = (\mathbf{v}_1, \dots, \mathbf{v}_p)$, where $\mathbf{v}_j \in \mathbb{R}^n$. The *span* of a set of vectors $\tilde{\mathbf{v}}$, denoted $\text{span}(\tilde{\mathbf{v}})$, is the set of all linear combinations of vectors in $\tilde{\mathbf{v}}$. Next, assign a symbol to $\text{span}(\tilde{\mathbf{v}})$, say

$$\mathcal{V} = \text{span}(\tilde{\mathbf{v}}). \quad (7.1)$$

What kind of object is \mathcal{V} ? Any linear combination of vectors from $\tilde{\mathbf{v}}$ is itself a vector in \mathbb{R}^n , so \mathcal{V} is a special type of subset, in particular,

$$\mathcal{V} \subset \mathbb{R}^n.$$

Consider the following example of a vector span.

Example 7.1. We are given the set of vectors $\tilde{\mathbf{v}} = (\mathbf{v}_1, \mathbf{v}_2)$ in \mathbb{R}^3 :

$$\begin{aligned}\mathbf{v}_1 &= (1/2, 1/2, 5/8), \\ \mathbf{v}_2 &= (1/2, -1/2, -1/8).\end{aligned}$$

Then $\text{span}(\tilde{\mathbf{v}})$ is the set of all vectors

$$\mathcal{V} = \{\beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 : \beta_1, \beta_2\}. \quad (7.2)$$

Then suppose $\mathbf{v}' \in \mathcal{V}$. This means

$$\mathbf{v}' = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (\beta_1 + \beta_2)/2 \\ (\beta_1 - \beta_2)/2 \\ (5\beta_1 - \beta_2)/8 \end{bmatrix} \quad (7.3)$$

for some β_1, β_2 . After some algebra we have

$$\begin{aligned} \beta_1 &= x_1 + x_2 \\ \beta_2 &= x_1 - x_2, \end{aligned}$$

which upon substitution into the third element of the vector defined in (7.3) gives the constraint

$$x_3 = \frac{1}{2}x_1 + \frac{3}{4}x_2. \quad (7.4)$$

Thus, the span \mathcal{V} can be defined explicitly by construction, as in (7.2), or implicitly, as any vector (or point) in \mathbb{R}^3 which satisfies constraint (7.4):

$$\mathcal{V} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_3 = \frac{1}{2}x_1 + \frac{3}{4}x_2\}.$$

In addition, we maybe regard \mathcal{V} as the surface of the multivariate function

$$f(x_1, x_2) = \frac{1}{2}x_1 + \frac{3}{4}x_2,$$

so that

$$\mathcal{V} = \{(x_1, x_2, f(x_1, x_2)) : (x_1, x_2) \in \mathbb{R}^2\}.$$

However \mathcal{V} is conceived, it can be visualized as a 2-dimensional plane in \mathbb{R}^3 , which is illustrated graphically in Figure 7.1.

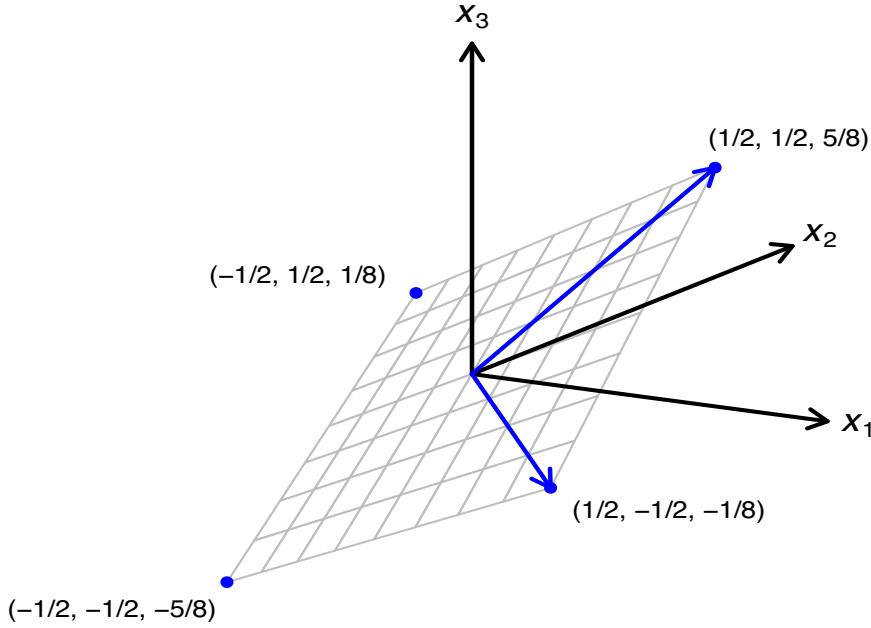


Figure 7.1: Plot shows rectangular section of $\text{span}(\tilde{\mathbf{v}})$, where $\tilde{\mathbf{v}} = (\mathbf{v}_1, \mathbf{v}_2)$, $\mathbf{v}_1 = (1/2, 1/2, 5/8)$, $\mathbf{v}_2 = (1/2, -1/2, -1/8)$.

□

It is important to observe that the vector span \mathcal{V} given in Example 7.1 exists in three dimensions, but is essentially a two-dimensional object. This is a crucial concept in studying linear models. That \mathcal{V} is two-dimensional would seem to follow from the fact that it is a span of two vectors. But the number of vectors defining the span does not necessarily determine its dimension. This issue is explored further in the next example.

Example 7.2. We continue Example 7.1. Suppose we add a new vector $\mathbf{v}_3 = (-1/2, -1/2, -5/8)$, and set $\tilde{\mathbf{v}}' = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. Clearly, the dimension of $\text{span}(\tilde{\mathbf{v}}')$ will be at least 2, because $\tilde{\mathbf{v}}'$ includes \mathbf{v}_1 and \mathbf{v}_2 . However, \mathbf{v}_3 is in the span of \mathbf{v}_1 and \mathbf{v}_2 (see Figure 7.1). More precisely, \mathbf{v}_3 is a linear combination of \mathbf{v}_1 and \mathbf{v}_2 :

$$\mathbf{v}_3 = -\mathbf{v}_1 + 0\mathbf{v}_2.$$

This means for any $\beta_1, \beta_2, \beta_3$

$$\begin{aligned}\mathbf{v}' &= \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3 \\ &= \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 (-\mathbf{v}_1 + 0\mathbf{v}_2) \\ &= (\beta_1 - \beta_3) \mathbf{v}_1 + \beta_2 \mathbf{v}_2.\end{aligned}$$

In other words, any linear combination of \mathbf{v}_1 , \mathbf{v}_2 and \mathbf{v}_3 is also a linear combination of \mathbf{v}_1 and \mathbf{v}_2 , so that $\text{span}(\tilde{\mathbf{v}}') = \text{span}(\tilde{\mathbf{v}})$. Therefore, adding \mathbf{v}_3 does not increase the dimension of the vector span.

Suppose for the third vector we use instead $\mathbf{v}_3 = (0, 0, 1)$, and redefine $\tilde{\mathbf{v}}'$ accordingly. Then all elementary vectors $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ are in $\text{span}(\tilde{\mathbf{v}}')$. This is verified by

observing

$$\begin{aligned} e_1 &= \mathbf{v}_1 + \mathbf{v}_2 - \frac{1}{2}\mathbf{v}_3 \\ e_2 &= \mathbf{v}_1 - \mathbf{v}_2 - \frac{3}{4}\mathbf{v}_3 \\ e_3 &= \mathbf{v}_3. \end{aligned}$$

This means that any linear combination of the elementary vectors e_1 , e_2 and e_3 is also a linear combination of \mathbf{v}_1 , \mathbf{v}_2 and \mathbf{v}_3 . But any vector in \mathbb{R}^3 is a linear combination of elementary vectors, so we may conclude $\text{span}(\tilde{\mathbf{v}}') = \mathbb{R}^3$, and the span is now three dimensional. \square

Example 7.2 makes clear that the dimension of a vector span need not be the same as the number of vectors defining the span (it can be less, but not greater). Consider the following definition:

Definition 7.1. The elements of vectors $\tilde{\mathbf{v}} = (\mathbf{v}_1, \dots, \mathbf{v}_m)$ are *linearly independent* if $\sum_{j=1}^m a_j \mathbf{v}_j = 0$ implies $a_j = 0$ for all j . Equivalently, the property holds if no \mathbf{v}_j is a linear combination of the remaining vectors.

Suppose the vectors in $\tilde{\mathbf{v}}$ are not linearly independent. This means that, say, \mathbf{v}_m is a linear combination of the remaining vectors, and so any linear combination in $\text{span}(\tilde{\mathbf{v}})$ including \mathbf{v}_m may be replaced with one including only the remaining vectors, so that $\text{span}(\tilde{\mathbf{v}}) = \text{span}(\mathbf{v}_1, \dots, \mathbf{v}_{m-1})$. This situation is illustrated in Example 7.2.

We then define the *vector space*:

Definition 7.2. A set of vectors \mathcal{V} is a *vector space* if it is closed under all linear combinations. In particular, if $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}$, then $a\mathbf{v}_1 + b\mathbf{v}_2 \in \mathcal{V}$, for any two scalars a, b . This means that a vector space must include the identity $\vec{0}$, and for every element $\mathbf{v} \in \mathcal{V}$ there is an inverse $\mathbf{v}^{-1} \in \mathcal{V}$ such that $\mathbf{v} + \mathbf{v}^{-1} = \vec{0}$.

Then a set of vectors $\tilde{\mathbf{v}}$ is a *basis* for vector space \mathcal{V} if

- (i) $\mathcal{V} = \text{span}(\tilde{\mathbf{v}})$, and
- (ii) The vectors in $\tilde{\mathbf{v}}$ are linearly independent (Definition 7.1).

Note that the definition of a vector space given in a formal treatment of abstract algebra or functional analysis is typically much more detailed. The purpose of much of this detail, however, is to ensure that linear combinations of vectors can be well defined. Once this is given, the important property of a vector space is closure under linear combinations, which in turn implies the existence of the identity and of inverses. Also, note that vector spaces can be defined for sets of general types of objects. Here, we take a “vector” to be an element of \mathbb{R}^n , but \mathcal{V} may also be some class of functions, as we will see below.

The *dimension* of a vector space \mathcal{V} is the minimum number of vectors whose span equals \mathcal{V} . Clearly, this equals the number in any set of linearly independent vectors which span \mathcal{V} . Equivalently, this is the number of vectors in any basis (Definition 7.2). Clearly, a vector space will not have a unique basis. However, the dimension is a characteristic of the vector space itself, so that any basis must have the same number of vectors.

7.2 Linear Transformations of Prediction Spaces

Consider the matrix representation of the multiple linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (7.5)$$

where \mathbf{y} is an $n \times 1$ response vector,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

is a $n \times p$ matrix, $\boldsymbol{\beta}$ is a $p \times 1$ vector of coefficients, and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of error terms. Then the least squares estimates $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ are obtained by minimizing

$$SSE = \sum_{i=1}^n (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$$

where $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ is the $n \times 1$ vector of fitted values.

Suppose we then consider a linear transformation of \mathbf{X}

$$\mathbf{X}' = \mathbf{XA} = \begin{bmatrix} x'_{11} & x'_{12} & \dots & x'_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ x'_{n1} & x'_{n2} & \dots & x'_{np} \end{bmatrix} \quad (7.6)$$

so that

$$x'_{ij} = \sum_{k=1}^p a_{kj} x_{ik}.$$

That is, each predictor column of \mathbf{X}' is a linear combination of the predictor columns of \mathbf{X} .

This generates a transformed model

$$\mathbf{y} = \mathbf{X}'\boldsymbol{\beta}' + \boldsymbol{\epsilon}. \quad (7.7)$$

Note that \mathbf{y} and $\boldsymbol{\epsilon}$ are identical in models (7.5) and (7.7), and \mathbf{X}' is derived from \mathbf{X} .

Clearly, we can obtain least squares estimates $\hat{\boldsymbol{\beta}}' = [\hat{\beta}'_1 \dots \hat{\beta}'_p]^T$ for model (7.7) in the same way least squares estimates $\hat{\boldsymbol{\beta}} = [\hat{\beta}_1 \dots \hat{\beta}_q]^T$ were obtained for model (7.5).

The crucial question is whether or not the least squares fitted model for (7.5) and (7.7) differ in any important way. Clearly, $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}}'$ will not in general be equal. To see this, consider the following example.

Example 7.3. Suppose the transformation matrix \mathbf{A} is purely diagonal:

$$\mathbf{A} = \begin{bmatrix} 1/2 & 0 & \dots & 0 & 0 \\ 0 & 1/2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1/2 & 0 \\ 0 & 0 & \dots & 0 & 1/2 \end{bmatrix}.$$

Then the transformation is simply

$$\mathbf{X}' = \mathbf{XA} = \frac{1}{2}\mathbf{X},$$

and model (7.7) becomes

$$\mathbf{y} = \mathbf{X}'\boldsymbol{\beta}' + \boldsymbol{\epsilon} = \frac{1}{2}\mathbf{X}\boldsymbol{\beta}' + \boldsymbol{\epsilon}. \quad (7.8)$$

So, if $\hat{\boldsymbol{\beta}}$ is the least squares estimate of $\boldsymbol{\beta}$ for model (7.5), then it seems reasonable to expect that $\hat{\boldsymbol{\beta}}' = 2\hat{\boldsymbol{\beta}}$ will be the least squares estimates of $\boldsymbol{\beta}'$ for model (7.8), and the two models will not differ in any important way. This is what we might expect of the units of a predictor were changed from, say, feet to inches. \square

7.2.1 Evaluation of transformation matrix \mathbf{A}

We are sometimes given both design matrices \mathbf{X} , \mathbf{X}' of Equation (7.6), but then are faced with the problem of deducing the transformation matrix \mathbf{A} by which they are related. In fact, R provides many powerful functions with which to construct design matrices \mathbf{X}' which are linear transformations of predictors given in some convenient form, in which case it may be useful to know \mathbf{A} .

In this case we may simply premultiply Equation (7.6) by $[\mathbf{X}']^T$, yielding

$$[\mathbf{X}']^T \mathbf{X}' = [\mathbf{X}']^T \mathbf{XA},$$

so that

$$\mathbf{A} = ([\mathbf{X}']^T \mathbf{X})^{-1} [\mathbf{X}']^T \mathbf{X}'. \quad (7.9)$$

We will see that \mathbf{A} is often chosen so that \mathbf{X}' has orthonormal column vectors, so that

$$\mathbf{A} = ([\mathbf{X}']^T \mathbf{X})^{-1} \mathbf{I} = ([\mathbf{X}']^T \mathbf{X})^{-1}. \quad (7.10)$$

We will see examples of this in later applications.

7.3 General Transformation Equivalence

It turns out that this type of equivalence illustrated in Example 7.3 will hold for any invertible transformation matrix \mathbf{A} . Consider again model (7.5). Viewed geometrically, the vector of fitted values $\hat{\mathbf{y}}$ is a linear combination of the form

$$\hat{\mathbf{y}} = \hat{\beta}_1 \mathbf{x}_1 + \dots + \hat{\beta}_q \mathbf{x}_q,$$

where $\mathbf{x}_1, \dots, \mathbf{x}_q$ are the $n \times 1$ column vectors of \mathbf{X} . Let \mathcal{Y} be the set of all $n \times 1$ vectors which are linear combinations of the column vectors of \mathbf{X} (this set is strictly smaller than \mathbb{R}^n , provided $n > q$). If these column vectors are linearly independent, they form a basis for vector space \mathcal{Y} . Then $\hat{\mathbf{y}}$ is the unique vector in \mathcal{Y} which minimizes SSE (this is because SSE is interpretable as a strictly convex mapping of $\boldsymbol{\beta}$).

If we consider instead the matrix representation of the multiple linear regression model (7.7) we have

$$\mathbf{y} = \mathbf{X}'\boldsymbol{\beta}' + \boldsymbol{\epsilon},$$

with fitted values

$$\hat{\mathbf{y}}' = \hat{\beta}'_1 \mathbf{x}'_1 + \dots + \hat{\beta}'_q \mathbf{x}'_p.$$

Here, the least squares solution is obtained by minimizing SSE with respect to $\hat{\mathbf{y}}'$ over \mathcal{Y}' , the span of the column vectors of \mathbf{X}' . However, if A is invertible, it is easily shown that $\mathcal{Y} = \mathcal{Y}'$. In particular, if $\hat{\mathbf{y}} \in \mathcal{Y}$, then there exists $\hat{\boldsymbol{\beta}}$ such that

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{A}^{-1}\hat{\boldsymbol{\beta}} = \mathbf{X}'\hat{\boldsymbol{\beta}}',$$

where

$$\hat{\boldsymbol{\beta}}' = \mathbf{A}^{-1}\hat{\boldsymbol{\beta}}. \quad (7.11)$$

This establishes an important principle for transformed regression models.

Theorem 7.1. Suppose we have linear models

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \\ \mathbf{y}' &= \mathbf{X}'\boldsymbol{\beta}' + \boldsymbol{\epsilon}, \end{aligned}$$

where \mathbf{y} and $\boldsymbol{\epsilon}$ are common to both models, \mathbf{X} is an $n \times p$ matrix of linearly independent columns, and

$$\mathbf{X}' = \mathbf{XA}$$

for some invertible $p \times p$ matrix \mathbf{A} . Then suppose $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\beta}}'$ are the respective least squares estimates of the coefficients, and denote the fitted values $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{y}}' = \mathbf{X}'\hat{\boldsymbol{\beta}}'$. Then the two fitted models are equivalent in the sense that

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}'.$$

In addition, the following relation always holds

$$\hat{\boldsymbol{\beta}}' = \mathbf{A}^{-1}\hat{\boldsymbol{\beta}}.$$

Proof. The argument is essentially that leading to Equation (7.11). Alternatively, consider the SSE for model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$:

$$SSE[\boldsymbol{\beta}^*] = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*)^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*). \quad (7.12)$$

Note that the vector $\boldsymbol{\beta}^*$ may assume any value in \mathbb{R}^p . Then, it may be verified that $SSE[\boldsymbol{\beta}]$ is uniquely minimized by some $\hat{\boldsymbol{\beta}} \in \mathbb{R}^p$.

Then consider the SSE for model $\mathbf{y} = \mathbf{X}'\boldsymbol{\beta}' + \boldsymbol{\epsilon}$:

$$SSE'[\boldsymbol{\beta}^*] = (\mathbf{y} - \mathbf{X}'\boldsymbol{\beta}^*)^T(\mathbf{y} - \mathbf{X}'\boldsymbol{\beta}^*). \quad (7.13)$$

As in the previous SSE , $\boldsymbol{\beta}^*$ is simply a dummy variable over which SSE and SSE' are minimized. Then substitute the transformation $\mathbf{X}' = \mathbf{XA}$ into the preceding expression:

$$SSE'[\boldsymbol{\beta}^*] = (\mathbf{y} - \mathbf{XA}\boldsymbol{\beta}^*)^T(\mathbf{y} - \mathbf{XA}\boldsymbol{\beta}^*). \quad (7.14)$$

From Equation (7.12), $SSE[\boldsymbol{\beta}^*]$ is minimized by allowing $\boldsymbol{\beta}^*$ to vary over domain \mathbb{R}^p . However, the expression in (7.14) is equivalent to (7.12), except that $\mathbf{A}\boldsymbol{\beta}^*$ replaces $\boldsymbol{\beta}^*$. Since \mathbf{A} is invertible, as $\boldsymbol{\beta}^*$

varies over \mathbb{R}^p , so does $\mathbf{A}\beta^*$, so that the two minimization problems are equivalent. In particular, if $\beta^* = \hat{\beta}$ minimizes $SSE[\beta^*]$, and $\beta^* = \hat{\beta}'$ minimizes $SSE'[\beta^*]$, then

$$SSE[\hat{\beta}] = SSE'[\hat{\beta}'],$$

and

$$\hat{\beta} = \mathbf{A}\hat{\beta}'$$

□

7.4 Orthogonalization of Predictor Matrices

The *dot product* of two vectors $\mathbf{u} = (u_1, \dots, u_n)$ and $\mathbf{v} = (v_1, \dots, v_n)$ is the scalar

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^n u_i v_i.$$

Then \mathbf{u} is a *unit vector* if $\mathbf{u} \cdot \mathbf{u} = 1$. Vectors \mathbf{u} and \mathbf{v} are *orthogonal* if $\mathbf{u} \cdot \mathbf{v} = 0$, and are *orthonormal* if, in addition, they are both unit vectors.

A square $m \times m$ matrix \mathbf{M} is an *orthogonal matrix* if distinct column vectors are orthonormal. In that case, \mathbf{M} is invertible, with $\mathbf{M}^{-1} = \mathbf{M}^T$, so that $\mathbf{M}^T \mathbf{M} = \mathbf{I}$.

More generally, an $n \times p$ matrix \mathbf{X} may have orthogonal or orthonormal column vectors (this is only possible if $n \geq p$). In the latter case \mathbf{X} shares with an orthogonal matrix the property

$$\mathbf{X}^T \mathbf{X} = \mathbf{I}.$$

When we refer to the orthogonalization of a rectangular matrix, this is what is meant.

Then, linear transformations are often used to “orthogonalize” a linear regression model. If $\hat{\beta} = [\hat{\beta}_1 \dots \hat{\beta}_q]^T$ is the vector of least squares coefficient estimates, then the covariance matrix is given by

$$\Sigma_{\hat{\beta}} = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}$$

(Section 6.1). But an invertible square matrix is diagonal if and only if its inverse is diagonal. Therefore, the components of $\hat{\beta}$ are uncorrelated if and only if $\mathbf{X}^T \mathbf{X}$ is a diagonal matrix. This condition is equivalent to

$$\sum_{i=1}^n x_{ij} x_{ik} = 0 \tag{7.15}$$

for each pair $j \neq k$. If the columns of \mathbf{X} are *orthogonal*, $\mathbf{X}^T \mathbf{X}$ will be a diagonal matrix, and if the columns are *orthonormal*, $\mathbf{X}^T \mathbf{X}$ will be the identity matrix.

7.4.1 Orthogonalization of simple linear regression

Simple linear regression offers a ready demonstration of the orthogonalization process. Here we have

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

Select transformation matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -\bar{x} \\ 0 & 1 \end{bmatrix}$$

where $\bar{x} = n^{-1} \sum_i x_i$. Then

$$\mathbf{X}' = \begin{bmatrix} 1 & x_1 - \bar{x} \\ \vdots & \vdots \\ 1 & x_n - \bar{x} \end{bmatrix}.$$

We then have

$$[\mathbf{X}']^T \mathbf{X}' = \begin{bmatrix} n & 0 \\ 0 & \sum_{i=1}^n (x_i - \bar{x})^2 \end{bmatrix}$$

therefore condition (7.15) is satisfied, which implies that the least squares coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ are uncorrelated.

7.4.2 QR decomposition

Clearly, the basis for a vector space is not unique, and the formulation of a basis with certain advantageous properties can be an important problem. Suppose \mathbf{X} is a $n \times p$ design matrix, and let \mathcal{V} be the span of the column vectors. If the column vectors are linearly independent, then they will form a basis for \mathcal{V} .

Definition 7.3. A *QR* decomposition of an $n \times p$ matrix \mathbf{X} is any product

$$\mathbf{X} = \mathbf{Q}\mathbf{R} \tag{7.16}$$

where \mathbf{Q} is an $n \times r$ matrix with orthonormal column vectors and \mathbf{R} is a $r \times p$ matrix. The smallest value of r for which this decomposition is possible is the *matrix rank* of \mathbf{X} . Then \mathbf{X} is a *full rank matrix* if its rank is p , assuming $p \leq n$.

The column vectors of a full rank matrix are necessarily linearly independent, in which case \mathbf{R} is invertible.

Clearly, a *QR* decomposition will not be unique. They can be constructed using a number of well-known algorithms, including the *Gram-Schmidt process*, *Householder transformations*, or *Givens rotations* (Chambers, 1977). For a full rank matrix, \mathbf{R} can be readily constructed to be upper triangular. Note that an upper triangular matrix is invertible if and only if all diagonal elements are nonzero.

7.4.3 Orthogonalization of multiple linear regression

QR decomposition provides a general method of extending the idea discussed in Section 7.4.1 to multiple linear regression. First note that we may force the new predictor matrix \mathbf{X}' to have orthonormal in addition to orthogonal column vectors, in which case

$$\Sigma_{\hat{\beta}} = \sigma^2 ([\mathbf{X}']^T \mathbf{X}')^{-1} = \sigma^2 I_p. \tag{7.17}$$

Then given prediction matrix \mathbf{X} and *QR* decomposition (7.16), we set transformation matrix $\mathbf{A} = \mathbf{R}^{-1}$, giving transformed predictor matrix

$$\mathbf{X}' = \mathbf{X}\mathbf{R}^{-1} = \mathbf{Q},$$

possessing orthonormal columns. We will see this idea used later in Section 19.2. It is also a useful countermeasure in the presence of collinearity (Section 8.6).

Chapter 8

Linear Regression Diagnostics - Outliers, Influential Observations and Collinearity

An anomalous observation in a single sample is almost always an *outlier*, or a measurement that is a relatively large distance from almost all remaining measurements. In regression, there is more than one reason to consider an observation anomalous (an observation here is collectively the response with associated predictor values). Also complicating matters is the fact that the responses are not from a single well defined distribution. Anomalies in linear regression are usually (at least) one of the following three types:

- An **outlier** is an observation with a large residual.
- An observation with **high leverage** is an observation with one or more relatively extreme predictor values.
- An observation is **influential** if its removal changes the fitted model significantly. This applies either to any of the coefficients, or to a fitted values.

Since the motivation is to test whether or not removing an observation significantly changes the model fit (relative to the remaining observations), quantitative diagnostic measures often measure the effect of deleting an observation. This can be done by simply recalculating the fit after removing each observation in turn, but explicit formula usually exist, saving considerable computation time.

We use the notation $\hat{\mathbf{y}}^{-i}$, $\hat{\beta}_j^{-i}$, $[S_{\hat{\beta}}^2]^{-i}$, for example, to denote the various quantities associated with a regression model calculated after deleting the i th observation.

The reader should review Chapter 6, Appendix B, and Appendix A as needed.

8.1 Leverage

The i th diagonal element H_{ii} of the hat matrix H (Section 6.3) is referred to as the *leverage* for the i th observation. The motivation for this definition is as follows. Estimates should be reasonably stable, in the sense that a small change in a data set should not result in a large change in the

model estimate. However, if one observation has a relatively large value for H_{ii} (referred to as a *high leverage point*), this suggests that it has a disproportionately large effect on the fitted model, as can be seen by Equation (6.7). This may be problematic, since we would not like the fitted model to depend significantly on the presence or absence of one, or a few, high leverage points.

We next consider what constitutes high leverage, and there are a number of principles that may be used. The simplest approach is to examine all leverage values H_{ii} with a boxplot or histogram to detect outliers. However, there are several methods by which the magnitude of H_{ii} can be judged without having to examine all leverage values at once.

It may be shown that we always have

$$\text{trace}(H) = q,$$

where the *trace* of a square matrix is the sum of the diagonal elements. Also, it always holds that $n^{-1} \leq H_{ii} \leq 1$. If there are n observations then the average value for H_{ii} must be q/n . For this reason, high leverage points may be flagged with a simple rule such as

$$H_{ii} \geq 2q/n.$$

8.2 Cook's Distance

Another commonly used diagnostic is based on *Cook's distance*:

$$D_i = \frac{e_i^2}{q \times MSE} \left[\frac{H_{ii}}{(1 - H_{ii})^2} \right].$$

This is an interesting statistic for a number of reasons. First, it can be shown that an equivalent form for D_i is

$$D_i = \frac{\sum_{j \neq i} (\hat{y}_j - \hat{y}_j^{-i})^2}{q \times MSE},$$

where \hat{y}_j is the j th fitted value using all data, and \hat{y}_j^{-i} is the j th fitted value obtained after deleting observation i . The equivalence of these two forms for D_i show that a fitted model may change considerably following the addition or deletion of a high leverage point.

We also note that D_i may be compared to a $F_{q,n-q}$ distribution, and on this basis high leverage points may be flagged by comparison to an appropriate quantile. A number of rules are used, which are more or less conservative, for example:

$$D_i \geq 1.$$

8.3 Studentized Residuals

In statistics, the term *studentize* refers to the adjustment of a statistic by dividing it by its standard error, in the form of an estimate of the true standard deviation obtained from the data. The t -statistic is one example. The *studentized residuals* are therefore given by

$$e_i^* = \frac{e_i}{S'_{e_i}}, \quad i = 1, \dots, n. \tag{8.1}$$

Because e_i^* is usually used for diagnostic purposes, we do not estimate σ_{e_i} using all the data. For this reason, we use the notation $S'_{e_i} \approx \sigma_{e_i}$ in (8.1). We retain the quantity H_{ii} appearing in the expression for $\sigma_{e_i}^2$ in (6.9). However, because we are accepting the possibility that the i th observation is anomalous, it is appropriate to use MSE^{-i} instead of MSE to estimate σ^2 in (6.9), noting that both are unbiased estimates of σ^2 . This gives

$$S'_{e_i} = \sqrt{MSE^{-i}(1 - H_{ii})},$$

which, combined with (8.1) defined the studentized residual, sometimes denoted $RSTUDENT_i$.

8.4 Influence Measures

One method of determining the influence of an observation is to simply delete it, recalculate any of the various model quantities, and then note the change. Accordingly, given q predictors (including the intercept), define the quantities

$$DFBETA_{ij} = \hat{\beta}_j - \hat{\beta}_j^{-i} = \frac{[(\mathbf{X}^T \mathbf{X})^{-1} \dot{x}_i]_j e_i}{1 - H_{ii}}, \quad i = 1, \dots, n, \quad j = 1, \dots, q.$$

Similarly, define

$$DFFIT_i = \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_i^{-i} = \frac{H_{ii} e_i}{1 - H_{ii}},$$

where $\hat{\mathbf{y}}_i^{-i}$ is the fitted value of the model at predictor value \dot{x}_i recalculated after deleting the i th observation.

These quantities can be made more easily interpretable in their *standardized* form. Since $DFBETA_{ij}$ measures a change in $\hat{\beta}_j$ resulting from the deletion of the i th observation, it makes sense to standardize it by dividing by its standard error, the equation for which is given in (6.4). Since we are interested in the standard error of $\hat{\beta}_j$ and not $\hat{\beta}_j^{-i}$ (which are different) we do not delete the i th observation in \mathbf{X} . However, because we are accepting the possibility that the i th observation is anomalous, it is appropriate to substitute MSE^{-i} for MSE in (6.4), noting that both are unbiased estimates of σ^2 . This gives the standardized form

$$DFBETAS_{ij} = \frac{DFBETA_{ij}}{\sqrt{MSE^{-i} \times [(\mathbf{X}^T \mathbf{X})^{-1}]_{jj}}}.$$

Similarly, from the standard error for $\hat{\mathbf{y}}_i$ given in (6.8) we have the standardized version of $DFFIT_i$:

$$DFFITS_{ij} = \frac{DFFIT_i}{\sqrt{MSE^{-i} \times H_{ii}}}.$$

8.5 Covariance Ratio

The *covariance ratio* is defined as

$$cov.ratio = \frac{\det([S_{\hat{\beta}}^2]^{-i})}{\det(S_{\hat{\beta}}^2)}$$

This measures the aggregate effect of deleting an observation on the standard errors of the coefficient estimate $\hat{\beta}$.

8.6 Collinearity

Based on (6.3), the standard error for $\hat{\beta}_j$ can be shown to be obtained from

$$S_{\hat{\beta}_j}^2 = MSE \times [(\mathbf{X}^T \mathbf{X})^{-1}]_{jj} = \frac{MSE}{\sum_{i=1}^n (x_{ij} - \bar{x}_j)^2} \frac{1}{1 - R_{\mathbf{x}_j | \mathbf{X}^{-j}}^2}. \quad (8.2)$$

Here, \bar{x}_j is the mean of predictor \mathbf{x}_j , and $R_{\mathbf{x}_j | \mathbf{X}^{-j}}^2$ is the value of R^2 obtained by regressing \mathbf{x}_j onto the remaining predictors. It is interesting to compare this expression to the standard error for the slope coefficient in simple regression

$$S_{\hat{\beta}_1}^2 = \frac{MSE}{\sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}$$

(see Equation (4.4)). The form is the same for both, except for the *variance inflation factor* (VIF):

$$VIF_j = \frac{1}{1 - R_{\mathbf{x}_j | \mathbf{X}^{-j}}^2}$$

which appears in equation (8.2). A few things are notable about VIF_j . First, the quantity $R_{\mathbf{x}_j | \mathbf{X}^{-j}}^2$ is a direct measure of *collinearity*, or the degree to which \mathbf{x}_j is linearly correlated to other predictors. It can be seen that if \mathbf{x}_j is exactly equal to some linear combination of other predictors, then it is not needed in the model. For example, if $\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{x}_3$, then a fitted model

$$\hat{\mathbf{y}} = 2.5\mathbf{x}_1 - 4.7\mathbf{x}_2 + 10.4\mathbf{x}_3 \quad (8.3)$$

can always be replaced by

$$\begin{aligned} \hat{\mathbf{y}} &= 2.5\mathbf{x}_1 - 4.7\mathbf{x}_2 + 10.4\mathbf{x}_3 \\ &= 2.5(\mathbf{x}_2 + \mathbf{x}_3) - 4.7\mathbf{x}_2 + 10.4\mathbf{x}_3 \\ &= -2.2\mathbf{x}_2 + 12.9\mathbf{x}_3, \end{aligned} \quad (8.4)$$

and we can dispense with \mathbf{x}_1 entirely. Equations (8.3) and (8.4) are equivalent in the sense that they yield exactly the same fitted values. So they are the same model. However, this example does not convey the entire problem. We could easily construct a third equivalent model:

$$\hat{\mathbf{y}} = 1.5\mathbf{x}_1 - 3.7\mathbf{x}_2 + 11.4\mathbf{x}_3. \quad (8.5)$$

Although we generally expect the least squares estimates to uniquely minimize the *SSE*, when predictors are not linearly independent, there will exist an infinite number of least squares fits.

Next, suppose the predictors are linearly independent but that for some predictor \mathbf{x}_j , $R_{\mathbf{x}_j | \mathbf{X}^{-j}}^2$ is very close to 1. We will have a unique least squares fit, but something of the character of the preceding example remains. Models with widely varying fitted coefficients will have values of *SSE* close to the minimum attainable, and will yield very similar fitted values $\hat{\mathbf{y}}$. The consequence of

this can be seen directly, since we would then have a very large value of VIF_j and, by (8.2), a very large value for $S_{\hat{\beta}_j}^2$, meaning that the coefficient β_j cannot be reliably estimated. A generally used rule of thumb flags collinearity effects when

$$VIF_j \geq 10.$$

Collinearity can be avoided by orthogonalizing the prediction matrix \mathbf{X} using QR decomposition (Section 7.4.3) or principal components analysis (Section 17.3).

Demonstration software file **REGRESSION-D.R** gives an introduction to the use of R in regression diagnostics, while Problems 22.7 and 22.8 explore the topic further.

8.7 Postscript

Chapter 22 gives practice problems for linear regression, although this topic is explored in the context of other methodology introduced in these notes. There are also four demonstration software files **REGRESSION-A.R**, **REGRESSION-B.R**, **REGRESSION-C.R** and **REGRESSION-D.R**.

The literature on linear regression is, of course, quite extensive, and it would be difficult to do it justice.

First, as usual, James *et al.* (2013) provides an introduction to linear regression comparable to that provided here, in the context of R, and is an excellent source of practice problems. Neter *et al.* (1996) can be singled out as an excellent introductory text. Concepts are introduced throughout in the context of detailed examples. It is also quite comprehensive in scope, and covers at an introductory level a number of important topics not included in these notes (correlated errors in linear models, for example). One further advantage is that it includes an extended section on ANOVA, which by itself is comparable to an independent textbook. This can be highly recommended as a reference text for the practicing data scientist.

For the reader interested in a more detailed development of linear regression from the linear algebraic point of view, Sen and Srivastava (2012) can be highly recommended. More advanced topics are covered in Seber and Lee (2012) and Seber and Wild (1989).

When error terms are not independent, or identically distributed, as will often be the case, *generalized least squares* may be used (otherwise, inferences which make use of the *iid* assumption will be incorrect). Alternatively, *longitudinal models* anticipate correlation induced by *repeated measures*, or observations clustered by subject, typically observed over time. This subject is quite advanced, and requires a separate study. A good starting point would be McCulloch *et al.* (2008).

Chapter 9

Maximum Likelihood Estimation

Suppose we are given a joint density $f(\tilde{X}; \theta)$ of a random vector $\tilde{X} = (X_1, \dots, X_n)$, noting that the density depends on a parameter $\theta \in \Theta$, where Θ is known as the *parameter space*. If we think of $f(\tilde{X}; \theta)$ as a function on the n -dimensional sample space, fixing θ , it is a probability density function. However, we may also think of it as a function of θ over Θ , holding \tilde{X} fixed. In this case, it is referred to as the *likelihood function*

$$l(\theta) = l(\theta; \tilde{X}) = f(\tilde{X}; \theta),$$

or, equivalently, the *log-likelihood function*

$$L(\theta) = L(\theta; \tilde{X}) = \log f(\tilde{X}; \theta).$$

If we are given a sample \tilde{X} , and we know the density has form $f(\tilde{X}; \theta)$, but we don't know the value of θ , then $L(\theta; \tilde{X})$ becomes a type of index describing how well a particular parameter value $\theta' \in \Theta$ describes the data. This is because, intuitively, we would expect the likelihood $l(\theta'; \tilde{X})$ to be relatively large when θ' is close to the true value of θ . Thus, the *maximum likelihood estimate* (MLE) $\hat{\theta}_{MLE}$ is defined as

$$\hat{\theta}_{MLE} = \operatorname{argmax}_{\theta \in \Theta} L(\theta; \tilde{X}). \quad (9.1)$$

There are, of course, a few technical issues. First, there is no guarantee that a maximum is unique, or even exists, although for many well known cases regularity conditions under which this holds have been derived. Note also that we use the log-likelihood function. It would be equivalent to use the likelihood function, but in practice the computation tends to be simpler using (9.1).

9.1 Fisher Information

There exists a general theory that we outline here. Given a function $f(x_1, \dots, x_n)$ of n variables the *Hessian matrix* is the matrix of second order partial derivatives:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

We will use the shorthand

$$H = \frac{\partial^2 f}{\partial x^2}$$

where we use vector representation $x = (x_1, \dots, x_n)$. The i, j th element of H is written

$$H_{ij} = \frac{\partial^2 f}{\partial x_j \partial x_i}.$$

The *Fisher information matrix* is defined as

$$\mathcal{I}(\theta) = -E \left[\frac{\partial^2 L(\theta; \tilde{X})}{\partial \theta^2} \right]$$

and plays an important role in the theory of statistical inference, since it can be shown that the covariance matrix of $\hat{\theta}_{MLE}$, say $\Sigma_{\hat{\theta}_{MLE}}$, is approximately equal to $\mathcal{I}(\theta^*)^{-1}$, where θ^* is the true value. Of course, θ^* will not be known in the typical inference problem, but is estimated by $\hat{\theta}_{MLE}$. In turn, the covariance matrix of $\hat{\theta}_{MLE}$ can be estimated by substituting $\hat{\theta}_{MLE}$ for θ^* , that is,

$$\Sigma_{\hat{\theta}_{MLE}} \approx \mathcal{I}(\hat{\theta}_{MLE})^{-1}.$$

The *observed Fisher information matrix* is then

$$\mathcal{J}(\theta) = -\frac{\partial L(\theta^2; \tilde{X})}{\partial \theta^2}.$$

When this information matrix is used, it is sometimes the convention to refer to $\mathcal{I}(\theta)$ as the *expected Fisher information matrix*, so that $\mathcal{I}(\theta) = E[\mathcal{J}(\theta)]$. It is also possible to estimate $\Sigma_{\hat{\theta}_{MLE}}$ using the observed information, setting

$$\Sigma_{\hat{\theta}_{MLE}} \approx \mathcal{J}(\hat{\theta}_{MLE})^{-1}.$$

Using either form of information, we then have a general approach to formal inference in maximum likelihood estimation. Suppose $\theta \in \Theta \subset \mathbb{R}^p$. This means

$$\hat{\theta}_{MLE} = (\hat{\theta}_1, \dots, \hat{\theta}_p)$$

is a p -dimensional vector. Under general conditions the distribution of $\hat{\theta}_{MLE}$ is well approximated by a multivariate normal distribution

$$\hat{\theta}_{MLE} \sim N(\theta^*, \Sigma_{\hat{\theta}_{MLE}}).$$

Since $\Sigma_{\hat{\theta}_{MLE}} \approx \mathcal{I}(\theta^*)^{-1}$ we also have the approximate distribution

$$\hat{\theta}_{MLE} \sim N(\theta^*, \mathcal{I}(\theta^*)^{-1}).$$

The final step is to substitute $\hat{\theta}_{MLE}$ for θ^* , using either the expected or observed information:

$$\hat{\theta}_{MLE} \sim N(\theta^*, \mathcal{I}(\hat{\theta}_{MLE})^{-1}) \text{ or } \hat{\theta}_{MLE} \sim N(\theta^*, \mathcal{J}(\hat{\theta}_{MLE})^{-1}).$$

The choice between the expected and observed information has been widely discussed in the literature, largely due to the seminal paper Efron and Hinkley (1978).

9.2 Inference Methods

Suppose we accept the estimate

$$\Sigma_{\hat{\theta}_{MLE}} \approx \hat{I}^{-1},$$

where

$$\hat{I} = \mathcal{I}(\hat{\theta}_{MLE}) \text{ or } \hat{I} = \mathcal{J}(\hat{\theta}_{MLE}).$$

The standard error of component $\hat{\theta}_i$ of $\hat{\theta}_{MLE}$ is therefore

$$SE_{\hat{\theta}_i} = \sqrt{[\hat{I}^{-1}]_{ii}},$$

that is, the square root of the i th element on the diagonal of \hat{I}^{-1} . Then, level $1 - \alpha$ confidence intervals for the i th element of θ are given by

$$CI_{1-\alpha} = \hat{\theta}_i \pm t_{n-d;\alpha/2} SE_{\hat{\theta}_i},$$

where d is the degrees of freedom of the model, or for large sample sizes

$$CI_{1-\alpha} = \hat{\theta}_i \pm z_{\alpha/2} SE_{\hat{\theta}_i}.$$

Similarly, under a null hypothesis $H_0 : \theta_i = \theta_i^*$ we have null distribution

$$\frac{\hat{\theta}_i - \theta_i^*}{SE_{\hat{\theta}_i}} \sim T_{n-d}$$

or for large samples

$$\frac{\hat{\theta}_i - \theta_i^*}{SE_{\hat{\theta}_i}} \sim N(0, 1).$$

Example 9.1. It is worth examining what the likelihood function looks like for a model we have already seen. The model for simple linear regression is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2).$$

Usually, the error terms ϵ_i are assumed to be independent. Of course, this might not be the case, but we will assume that here. This means that the responses y_i are also independent.

How do we assign a density to this model, in order to construct a likelihood function? The first problem is to define the unknown parameters. In simple linear regression, this usually includes β_0, β_1 , although even here this ultimately depends on the application. The next problem is to decide whether or not σ^2 is a parameter. It is almost always ‘unknown’, but if it is not the object of the inference, we may regard it as fixed (if not ‘known’), taking only the regression coefficients as parameters. In this particular case, either choice will lead to the same estimated regression coefficients, so we will opt to regard σ^2 as fixed.

Note also that the predictors x_i are considered fixed. This means

$$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2),$$

and so the density of $\tilde{y} = (y_1, \dots, y_n)$ is, following (B.5), is

$$f(\tilde{y}) = \prod_{i=1}^n \phi(y_i; \beta_0 + \beta_1 x_i, \sigma^2).$$

After some algebra, the log-likelihood function becomes

$$L(\beta_0, \beta_1; \tilde{y}) = -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 + C \quad (9.2)$$

where C is a constant which does not depend on parameters β_0, β_1 , and so may be removed from the likelihood function.

Finally, examining (9.2) we can see that the estimates $\hat{\beta}_0, \hat{\beta}_1$ which maximum the log-likelihood function are exactly those that minimize the least squares criterion

$$SSE[\beta_0, \beta_1] = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2.$$

This means that the MLEs of β_0, β_1 are equal to the least squares estimates. \square

9.3 The Likelihood Ratio Test and Deviance

In linear regression the quantity SSE serves as a goodness of fit measure. It also serves as a means of comparing *nested models*. Suppose a full model has q predictors (in addition to the intercept). Then define a reduced model which contains only $p < q$ of these predictors. The reduced model can be considered to be a special case of the full model in which the coefficients β_i for the removed predictors are forced to zero. We then have $SSE_{red} \geq SSE_{full}$, since SSE_{full} is the minimum SSE over the full model space, which includes the reduced model. The F -statistic is then

$$F = \frac{(SSE_{red} - SSE_{full})/(q-p)}{SSE_{full}/(n-(q+1))} \quad (9.3)$$

which has an $F_{q-p, n-(q+1)}$ distribution under tha null hypothesis that the reduced model is correct.

A similar method of inference is available for maximum likelihood estimation. Suppose we are given data \tilde{X} and a likelihood function $l(\theta; \tilde{X})$ on parameter space $\Theta_f \in \mathbb{R}^q$. Then let $\Theta_r \in \mathbb{R}^p$. Assume the parameter spaces Θ_r and Θ_f are nested, in the sense that models in Θ_r are contained in Θ_f . This situation describes the comparison of full and reduced regression models just described. Similarly, we calculate the full and reduced MLEs

$$\begin{aligned} \hat{\theta}_f &= \operatorname{argmax}_{\theta \in \Theta_f} l(\theta; \tilde{X}), \\ \hat{\theta}_r &= \operatorname{argmax}_{\theta \in \Theta_r} l(\theta; \tilde{X}). \end{aligned}$$

The likelihood ratio statistic is then

$$\Lambda(\tilde{X}; \hat{\theta}_r, \hat{\theta}_f) = \frac{l(\hat{\theta}_r; \tilde{X})}{l(\hat{\theta}_f; \tilde{X})},$$

with small values of $\Lambda(\tilde{X})$ tending to support the full model. If we define null hypothesis $H_0 : \theta \in \Theta_r$ (that is, the reduced model is correct) then by *Wilks's theorem*

$$-2 \log(\Lambda(\tilde{X}; \hat{\theta}_r, \hat{\theta}_f)) \sim \chi^2_{q-p} \quad (9.4)$$

approximately for large enough sample size. This serves the same purpose as the F -test defined in (9.3).

An analog of SSE may then be developed by defining the *saturated model*, for which the number of parameters is the same as the number of observations, so that the data are fitted exactly. For example, if in a linear regression model there are n linearly independent predictors (including the intercept) then we can attain $SSE = 0$ (since the fitted values can be made to equal the responses exactly). Suppose the resulting MLE is $\hat{\theta}_s$. Then any model is nested within the saturation model. Suppose $\hat{\theta}_m$ is the MLE for our model of interest. The model *deviance* is then based on the likelihood ratio test for the model of interest compared to the saturated model:

$$D(\hat{\theta}_m) = -2 \log(\Lambda(\tilde{X}; \hat{\theta}_m, \hat{\theta}_s)).$$

This statistic serves much the same purpose as the SSE , and permits a systematic comparison of models, since, for example

$$D(\hat{\theta}_r) - D(\hat{\theta}_f) = -2 \log(\Lambda(\tilde{X}; \hat{\theta}_r, \hat{\theta}_f)), \quad (9.5)$$

which is just a reformulation of (9.4). In practice, to compare nested models, we can calculate the change in deviance (9.5), and compare this quantity to a χ^2_{q-p} distribution. If the change is large (ie. the p -value is small), then we conclude that the full model is an improvement over the reduced model.

9.4 Postscript

Maximum likelihood theory is foundational, and must be studied in the context of a more general theory of statistical inference (Section 1.2). For our purposes, it provides a means of extending the ANOVA concept discussed in Section 2.10 to more general models. This idea is developed in Chapter 10 on logistic regression.

Chapter 10

Logistic Regression

Consider a vector of responses \mathbf{y} and a single predictor \mathbf{x} . So far, our regression models have assumed that the response has been normally distributed. We also noted that this assumption can be relaxed somewhat, with the inference methods for $\hat{\beta}_0, \hat{\beta}_1$, based on the t-distribution, remaining a reasonably accurate approximate.

Suppose, however, that we are interested in predicting a binary outcome. For example, we might track whether or not a clinic patient has a certain infection. Then the response would be, say, $y_i = 1$ if the patient is infected, and $y_i = 0$ otherwise. Thus, y_i is a Bernoulli random variable, and the normal assumption would not be appropriate.

Recall that when we develop a regression function

$$\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

we can think of \hat{f} either as a prediction of a future response y given feature x , or as an estimate of the expected response $E[y]$. Similarly, for binary response, we may regard the problem as one of developing a function which estimates the expected value

$$P(A) = E[y_i] = g(x_i) \quad (10.1)$$

where we recognize y_i as the indicator function $I\{A\}$. In our example, $A = \{\text{patient is infected}\}$.

The most common approach to this problem is the *logistic regression model*. We retain much of the structure of linear regression. For example, we have the usual prediction matrix \mathbf{X} and linear coefficients

$$\eta = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p.$$

However, η is not related to the response via the linear regression model $y = \eta + \epsilon$. Noting that an estimate of $E[y_i]$ should sensibly be forced into the unit interval $[0, 1]$, we select g in (10.1) which does this, which then relates η to the response:

$$E[y_i] = g(\eta). \quad (10.2)$$

The final choice is of g . Several are proposed in the literature, but the most commonly used is the *logistic function*

$$g(\eta) = \frac{e^\eta}{1 + e^\eta} = \frac{1}{1 + e^{-\eta}} \in (0, 1) \quad (10.3)$$

This completely specifies the model:

$$y_i \sim \text{bern}(g(\eta_i)), \text{ where } \eta_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}.$$

10.1 The Odds Ratio in Logistic Regression

One advantage of the logistic function (10.3) is that it gives a very convenient method of calculating the odds ratio of the defining event $y_i = 1$ between two predictor values. Given response/predictor pair (y, \dot{x}) we have estimate

$$\begin{aligned} P(y = 1) &\approx \frac{1}{1 + e^{-\hat{\beta}^T \dot{x}}}, \text{ and} \\ Odds(y = 1) &\approx e^{\hat{\beta}^T \dot{x}}. \end{aligned}$$

Given two predictor observations \dot{x}, \dot{x}' the odds ratio between them is therefore estimated by

$$OR(y = 1; \dot{x}, \dot{x}') \approx e^{\hat{\beta}^T (\dot{x} - \dot{x}')}.$$

10.2 Likelihood Method for Logistic Regression

The density function for a Bernoulli random variable $Y \sim bern(\pi)$ can be written

$$f(y) = \pi^y (1 - \pi)^{1-y}.$$

Then set expected values of y_i to be

$$\pi_i = g(\eta_i).$$

Note that π_i is ultimately a function of the regression coefficients β_i . If we assume the responses are independent (as is commonly done), the likelihood function is, after some algebra,

$$l(\hat{\beta}; \tilde{y}) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}, \quad (10.4)$$

and the log-likelihood function is

$$L(\hat{\beta}; \tilde{y}) = \sum_{i=1}^n y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i).$$

We can use the methods outlined in Chapter 9 to obtain MLEs and standard errors for $\hat{\beta}$. This is what most software does. However, there is no closed form solution to the optimization problem, so numerical algorithms are used.

We would also like to develop a goodness of fit measure, since this is an important tool for model selection. Although the model can be considered a form of classification, it yields a quantitative estimate in the form a probability, rather than a predicted class, so classification error CE is not a natural choice.

We then note that in linear regression, the goodness of fit measure is based on the criterion which is optimized in order to fit the model, which is the MSE. In logistic regression, it is the likelihood which is optimized.

In logistic regression, the fitted values are $\hat{\pi}_i$, the estimates of $E[y_i]$. Next, recall from the theory of linear regression the notion of a full and reduced model.

The Null Model

Recall that the simplest regression model is the one for which all coefficients are zero except the intercept:

$$y_i = \beta_0 + \epsilon_i, \quad (10.5)$$

that is, the predictors play no role. In this case, the least squares (and the maximum likelihood) estimate is $\hat{\beta}_0 = \bar{y}$.

The same logic applies to logistic regression. If the linear prediction term is simple $\eta_i = \beta_0$, it is easily shown that the maximum likelihood estimate of the fitted values is simply

$$\hat{\pi}_i = \hat{\pi}_{null} = \frac{1}{n} \sum_{i=1}^n y_i, \text{ for all } i,$$

which is the observed probability that $y_i = 1$ (the estimate $\hat{\beta}_0$ is whatever value uniquely achieves this). The *null likelihood* is then

$$l_{null} = \hat{\pi}_{null}^{n_1} (1 - \hat{\pi}_{null})^{n-n_1}$$

where n_1 is the number of responses equal to one.

The Fitted Likelihood

Once the MLEs $\hat{\beta}_i$ are calculated, they can be substituted back into the linear predictor terms to yield

$$\hat{\eta}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \dots + \hat{\beta}_p x_{ip}$$

and

$$\hat{\pi}_i = g(\hat{\eta}_i).$$

Substituting into (10.4) yields the *fitted likelihood*

$$l_{fit} = l(\hat{\beta}; \tilde{y}).$$

The Saturated Likelihood

Recall the saturated model of Section 9.3. If we had a ‘perfect’ model, then we would have enough predictors to force $y_i = \hat{y}_i$ for all i . As can be seen from (10.4) the likelihood would be equal to one in this case, yielding saturated likelihood:

$$l_{sat} = 1.$$

At this point recall the ANOVA structure in linear regression (Section 3.2). The total sum of squares decomposes as

$$SSTO = SSR + SSE.$$

While SSE is a goodness of fit measure for the regression model, it is also important to remember that SSTO is also interpretable as a special case of SSE for the null model (10.5), and SSR

is reduction in SSE when predictors are added to the null model. We also have coefficient of determination

$$R^2 = \frac{SSTO - SSE}{SSTO} = \frac{SSR}{SSTO}. \quad (10.6)$$

Following Section 9.3 we may define *model deviance*,

$$D_{\text{model}} = -[2 \log(l_{\text{fit}}) - 2 \log(l_{\text{sat}})] = -2 \log(l_{\text{fit}}),$$

and *null deviance*

$$D_{\text{null}} = -[2 \log(l_{\text{null}}) - 2 \log(l_{\text{sat}})] = -2 \log(l_{\text{null}}).$$

These quantities serve as a type of ANOVA decomposition, with analogous relationships

$$\begin{aligned} D_{\text{model}} &\iff SSE, \\ D_{\text{null}} &\iff SSTO, \text{ and} \\ D_{\text{null}} - D_{\text{model}} &\iff SSR. \end{aligned}$$

By Wilk's theorem (9.4), under the null hypothesis

$$H_0 : E[y_i] = \pi, \quad i = 1, \dots, n$$

that the null model is correct we have

$$D_{\text{null}} - D_{\text{model}} \sim \chi_p^2$$

approximately, where p is the number of predictors (in addition to the intercept).

In addition, it is common to define a *pseudo-R²*, of which several forms exist. By direct analogy to linear regression we have

$$R_L^2 = \frac{D_{\text{null}} - D_{\text{model}}}{D_{\text{null}}},$$

known as the *likelihood ratio R²* (compare to (10.6)). It is known that R_L^2 is not in a monotonic relationship with the odds ratio (Section 10.1). The *Cox-Snell R²* is an alternative pseudo- R^2 defined by

$$R_{CS}^2 = 1 - \left(\frac{l_{\text{null}}}{l_{\text{model}}} \right)^{2/n}.$$

This is a more natural choice for logistic regression based on maximum likelihood estimation. However, the maximum value of R_{CS}^2 is

$$R_{CS}^2 \leq 1 - (l_{\text{null}})^{2/n} < 1,$$

since R_{CS}^2 would be maximized by the saturated model. For this reason, the *Nagelkerke pseudo-R²* is sometimes used, which is simply R_{CS}^2 normalized to attain a maximum of 1:

$$R_N^2 = \frac{R_{CS}^2}{1 - (l_{\text{null}})^{2/n}}.$$

10.3 Postscript

One of the obvious limitations of Gaussian linear regression models considered in Chapters 3-8 is the restrictive assumption that the error terms are *iid* normal random variables, meaning that the responses are independent and normally distributed with common variance. In this chapter we considered the logistic regression model, for which responses are independent binary random variables.

We should note that logistic regression is one example of a class of models known as *generalized linear models* (GLM), originally formulated in the seminal paper Nelder and Wedderburn (1972) as a unified theory of linear models admitting responses of varying distributions. If we set the linear prediction term

$$\eta = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p,$$

we then specify a *link function* g through which the mean of response y is related to the predictors as follows:

$$E[y] = g^{-1}(\eta),$$

We then have a variance function

$$v(\mu) = \text{variance}(\mu),$$

which, depending on the distribution, relates the mean to the variance. For logistic regression with binary responses of mean μ , we have

$$E[y] = \frac{e^\eta}{1 + e^\eta} \tag{10.7}$$

and

$$v(\mu) = \mu(1 - \mu).$$

These models are fitted using maximum likelihood estimation (Chapter 9). However, it's worth noting that least squares regression estimation with Gaussian responses is mathematically equivalent to maximum likelihood estimation (Example 9.1), so the class of GLMs includes this model, after setting $g(x) = x$ and $v(\mu) \equiv \sigma^2$. GLMs are commonly used to build linear models with responses possessing the Poisson, gamma, inverse gamma, bernoulli or binomial distributions, and can also be used to model categorical responses with m classes (logistic regression is an example of this model with $m = 2$). While the variance function v is determined by the response distribution, the link function is a matter of choice. Usually, the choice of link function has significant implications for the type of inferences available. The choice of the link function implied by Equation (10.7) permits inference to be expressed in terms of the odds ratio (Sections 10.1, 16.5).

For a primary text, see, for example, McCullagh and Nelder (1989). In R, GLMs are implemented in the `glm()` function. See Venables and Ripley (2013) for a good introduction to its use.

GLMs are commonly extended to longitudinal models (Section 8.7). See, for example, McCulloch *et al.* (2008).

Chapter 23 contains a variety of practice problems in logistic regression modeling. In addition, demonstration software `CLASSIFICATION-B.R` give a detailed demonstration of a logistic regression analysis.

Chapter 11

Survival Analysis

We may think of *survival analysis* as the extension of statistical methods to models for which the response variable is some measure of time. This is conventionally referred to as a *survival time*, but represents any *time to event* observations (time between cancer diagnosis and cancer mortality, time from cancer treatment to cancer recurrence, waiting time in a queue, and so on). A survey of prediction methods should anticipate that the object to be predicted may be a survival time.

However, the statistical methods used in survival analysis differ in some important ways from other types of linear models, and therefore need a separate development (an in depth discussion of these methods is not included in, for example, James *et al.* (2013) or Friedman *et al.* (2001)). An important consideration is the possibility of *censored survival times*. Suppose the primary outcome of a study is time to recurrence of a form of cancer, represented as variable T . Suppose when the study ends a patient has been observed for, say 7 months without recurrence. We do not observe T for this patient, since any recurrence occurring after the study ends will not be observed by the researcher. But neither can we simply discard this information, since this will lead to a bias (probably, overall recurrence times would then be underestimated). Instead, we recognize that we still have partial information, in that we at least know that $T \geq 7$. Such an observation is *censored*, and the ability of survival analysis to model this type of data gives it much of its unique character. Once these methods are understood, survival analysis can be readily incorporated into the broader concepts of statistical learning.

Much of the more advanced theory of survival analysis is available in Cox and Oakes (1984). We also point out the introduction of the *Kaplan-Meier estimator* of the survival function (Section 11.3.2 below) in the seminal paper Kaplan and Meier (1958).

Formally, a *survival time* T is a nonnegative random variable variously interpreted as a *survival time*, *lifetime*, *waiting time*, *time to event*, and so on. A survival time may be discrete (for example, number of integer days until event) or continuous.

11.1 Memoryless Distributions

Recall the memoryless property which characterizes the geometric and exponential distributions, that the average waiting time remaining, after having already waited a time t , is the same as the original average waiting time.

Example 11.1. The geometric random variable possesses the interesting *memoryless property*. Suppose you have been repeatedly playing a game of chance with a probability p of winning. After k games you have not yet won. We assume the outcomes are independent. Is the amount of time until you win, starting from that point, different from the time to win when you started to play? Many are tempted to believe that prior losses shorten the expected time to future wins, as though the number of losses is somehow fixed. The question is answered by the following conditional probability:

$$P(X > k + t \mid X > k) = \frac{P(X > k + t)}{P(X > k)} = \frac{(1-p)^{k+t}}{(1-p)^k} = (1-p)^t, \quad (11.1)$$

using the CDF of the geometric distribution. This is the probability that at least t further losses precede a win following k consecutive losses. It is exactly the probability that at least t losses precede a win at the beginning of play. In other words, the waiting time for a win following k losses does not depend on k in any way. Equation (11.1) is a statement of the memoryless property. \square

At this point, it is worth noting the resemblance of the exponential density to the geometric density. First, the CDF is naturally calculated following the tail probability, for $t > 0$

$$\bar{F}_X(t) = P(X > t) = \int_t^\infty \lambda e^{-\lambda x} = -e^{-\lambda x} \Big|_t^\infty = e^{-\lambda t},$$

so that the CDF is

$$F_X(t) = \begin{cases} 0 & ; t < 0 \\ 1 - e^{-\lambda t} & ; t \geq 0 \end{cases}. \quad (11.2)$$

Second, the exponential RV possesses the same type of memoryless property demonstrated in Example 11.1.

Theorem 11.1. An exponentially distributed RV $X \sim \exp(\lambda)$ possesses the memoryless property

$$P(X > t + s \mid X > s) = P(X > t). \quad (11.3)$$

\square

Proof. We have directly from (11.2)

$$P(X > t + s \mid X > s) = \frac{P(X > t + s, X > s)}{P(X > s)} = \frac{P(X > t + s)}{P(X > s)} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} = e^{-\lambda t}$$

from which (11.3) follows. \square

11.2 The Failure Rate

The key to understanding this idea is to consider a *failure rate*. Suppose a survival time T is discrete, representing the number of days until failure of a component (a light bulb, for example). Let q_i be the failure rate on day i . This means that if the component has survived until day i ($T \geq i$) we toss a coin (independently of all previous coin tosses). If we get a ‘head’ (for our particular coins, this has probability q_i) the component ‘fails’ and the survival time is $T = i$, terminating the

process. First, note that the failure rates q_i are not a probability mass function for T . They are actually the conditional probabilities

$$q_i = P(T = i \mid T \geq i), \quad i = 1, 2, \dots$$

Second, these rates may increase or decrease in time, and what defines a memoryless distribution is precisely the assumption that the failure rates remain constant. This defines the geometric distribution.

We do not, on the other hand, expect the lifetime of, say, a car to be memoryless. We expect that the probability that a 10-year old car survives one more year is smaller than that for a 5-year old car, and smaller still than that for a new car. In other words, the failure rates q_i increase in i . Such a survival time is called *new better than used (NBU)*.

A survival time may also be *new worse than used (NWU)*, in which case the failure rates are decreasing. The survival time for young members of a species in an environment with high infant mortality will typically be NWU. This is because the period immediately after birth is very high in mortality risk, meaning that the failure rate is correspondingly high. However, if the infant survives this high risk period, the failure rate will decrease, resulting in a NWU survival time. Of course, if the infant survives into adulthood, the failure rate will begin to increase. A natural source of NWU survival times would be survival in competitive environments.

Example 11.2. A random variable W has a *Weibull distribution* if there are two parameters $k > 0$ and $\lambda > 0$ such that

$$X = (\lambda W)^k \tag{11.4}$$

has an *exp(1)* distribution (exponential distribution with $\lambda = 1$). This will be denoted $W \sim \text{weibull}(k, \lambda)$. This distribution is commonly used to model survival times. By convention, k is the *shape parameter* and θ is the *rate parameter*. Note that in some conventions λ is replaced by, say, $1/\tau$, in which case τ is referred to as a *scale parameter* (be careful, since λ may be used as a scale parameter). Both definitions are equivalent, once the transformation is understood. We use the rate parameter in order to emphasize the relationship with the exponential distribution.

Suppose $W \sim \text{weibull}(k, \lambda)$. The CDF of $X \sim \text{exp}(1)$ is $F_X(x) = 1 - \exp(-x)$ for $x \geq 0$, so

$$F_W(w) = P(W \leq w) = P\left(\lambda^{-1}X^{1/k} \leq w\right) = P\left(X \leq (\lambda w)^k\right) = 1 - \exp\left(-(\lambda w)^k\right), \quad w \geq 0,$$

and $F_W(w) = 0$ for $w < 0$. To evaluate the density function, take the derivative of the cumulative distribution function (CDF), giving

$$f_W(w) = \frac{d}{dw} \left\{ 1 - \exp\left(-(\lambda w)^k\right) \right\} = k\lambda^k w^{k-1} \exp\left(-(\lambda w)^k\right), \quad w \geq 0,$$

and $f_W(w) = 0$ for $w < 0$.

For some positive d , define the function

$$h(x; d, k, \lambda) = P(W \geq x + d \mid W \geq x) = \frac{P(W \geq x + d)}{P(W \geq x)}.$$

This is interpretable as the probability that a system with a lifetime of W survives an additional d time units, given that it has survived x time units. We may write an R function that accepts input (x, d, k, λ) and returns $h(x; d, k, \lambda)$. Note that the R function `dweibull` uses the scale parameter, not the rate parameter, so we need to transform accordingly.

```
f0 = function(x,d,ishape,irate) {
  pweibull(x+d,shape=ishape,scale=1/irate,lower.tail=F)
  /pweibull(x,shape=ishape,scale=1/irate,lower.tail=F)
}
```

Consider 3 Weibull distributions:

$$\begin{aligned} W_1 &\sim \text{weibull}(k = 1/2, \lambda = 1/5), \\ W_2 &\sim \text{weibull}(k = 1, \lambda = 1/10), \\ W_3 &\sim \text{weibull}(k = 3/2, \lambda = \sqrt{\pi}/20). \end{aligned}$$

The following R code draws the required plot in Figure 11.1.

```
ex1 = expression(italic(x))
ex2 = expression(paste(italic(h), '(', italic(x), ', ', italic(d), ',',
  ', italic(k), ', ', lambda, ')', sep=''))
ex3 = expression(paste(italic(d), ' = 1, ', italic(k), ' = 1/2,
  ', lambda, ' = 1/5', sep=''))
ex4 = expression(paste(italic(d), ' = 1, ', italic(k), ' = 1,
  ', lambda, ' = 1/10', sep=''))
ex5 = expression(paste(italic(d), ' = 1, ', italic(k), ' = 3/2,
  ', lambda, ' = ', sqrt(pi), '/20', sep=''))

xgr = seq(0,100,by = 1)
y = cbind(f0(xgr,1,0.5,1/5), f0(xgr,1,1.0,1/10), f0(xgr,1,2.0,sqrt(pi)/20))
par(mar=c(5,5,5,5))
matplot(xgr,y, col=c(2,3,4),lty=1,type='l',xlab=ex1,ylab=ex2)
legend('bottomleft',legend = c(ex3,ex4,ex5),col=c(2,3,4),lty=1,bty='n')
```

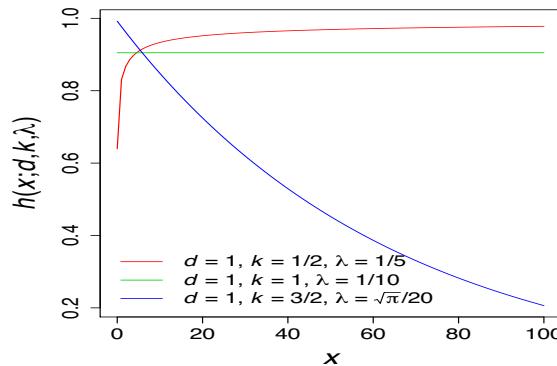


Figure 11.1: Plot for Example 11.2

The probability of surviving an additional time unit given survival up to time x increases for W_1 , remains constant for W_2 , and decreases for W_3 . This means W_1, W_2, W_3 are NWU, memoryless and NBU, respectively. See Figure 11.1. \square

The quantity $h(x; d, k, \lambda)$ of Example 11.2 is interpretable as a probability of surviving an addition d time units. If we divide by d , then the quantity $h(x; d, k, \lambda)/d$ can be interpreted as a *failure rate* or *hazard rate*. In fact, we can show that if we allow d to approach zero, the limit is precisely the *hazard function* of a survival time X :

$$h(x) = \frac{f(x)}{S(x)}, \quad (11.5)$$

where $S(x)$ is the *survival function*

$$S(x) = 1 - F(x) \quad (11.6)$$

and $f(x), F(x)$ are the density function and CDF of X . Then $h(x)$ is interpreted as the failure rate at time x . The *cumulative hazard function* is simply the integral

$$H(x) = \int_{y=0}^x h(y)dy. \quad (11.7)$$

and it can be shown that

$$H(x) = -\log(S(x)). \quad (11.8)$$

11.3 Estimation of the Survival Function

The survival function is of great importance in survival analysis. For example, a cancer prognosis is usually given in terms of $S(x)$. The statement ‘5 year survival is 30%’ means exactly $S(5) = 0.3$. Since $S(x)$ is simply the complement of the CDF $F(x)$, if we are given a sample of survival times x_1, \dots, x_n , we can first estimate F with the *empirical distribution function*

$$\hat{F}(x) = \frac{\text{number of } x_i \leq x}{n},$$

then the estimated survival function is

$$\hat{S}(x) = 1 - \hat{F}(x). \quad (11.9)$$

11.3.1 Censoring

Unfortunately, there is a feature common to samples of survival data that prohibits the use of (11.9). Suppose that we are studying the survival times of cancer patients (from time of diagnosis until death by cancer, for example). The survival time t_i recorded for patient i will, in practice, either be the time from diagnosis to death by cancer, or it will be the time that the patient was observed without having died from cancer. Presumably, a patient can be followed up only within the lifetime of the study, or it may be that the patient died of other causes, or left the study for any number of reasons. In this case we say that the observation t_i is *censored*. It is a partial observation of the survival time, in the sense that we can only say that the cancer survival time is $\geq t_i$. But, this is still useful information, and so should be incorporated into the analysis. Note there are other forms of censoring. The one we have described is known as *right censoring*.

The symbol ‘+’ is used to denote censoring. If we have data (in months)

$$10.3, 11.2+, 13.6, 15.2$$

this might mean, for example, that three patients died from cancer 10.3, 13.6 and 15.2 months after diagnosis, and one patient was observed for 11.2 months after diagnosis without having died from cancer.

11.3.2 Kaplan-Meier estimate of the survival function

Censored data cannot be used to construct the survival function estimate (11.9). Instead we may use the *Kaplan - Meier estimate*. Suppose we are given survival times

$$0 = t_0 < t_1 < t_2 < \dots < t_{m-1} < t_m. \quad (11.10)$$

Define intervals

$$I_i = [t_i, t_{i+1}).$$

Next, suppose p_i is the probability of surviving interval I_i , given survival up to time t_i . Then

$$S(t_i) \approx \prod_{j=0}^{i-1} p_j.$$

To estimate p_i , let $r(t_i)$ be the number at risk (still alive) just before time t_i , and let d_i be the number of deaths. Then we estimate

$$p_i \approx \hat{p}_i = \frac{r(t_i) - d_i}{r(t_i)}.$$

The Kaplan - Meier estimate of the survival function is then

$$\hat{S}(t) = \prod_{t_i < t} \hat{p}_i.$$

The estimator has a natural tabular representation. Suppose we are given a sample of survival times T_1, \dots, T_n . Values may be represented more than once, and some observations are censored. Suppose the represented values are sorted as in (11.10). Survival time 0 is included as t_0 whether or not it appears in the sample. Thus, m need not equal n . We can then construct table

i	t_i	d_i	$r(t_i)$	\hat{p}_i
0	$t_0 = 0$	d_0	$r(t_0) = n$	$(r(t_0) - d_0)/r(t_0)$
1	t_1	d_1	$r(t_1)$	$(r(t_1) - d_1)/r(t_1)$
\vdots	\vdots	\vdots	\vdots	\vdots
m	t_m	d_m	$r(t_m)$	$(r(t_m) - d_m)/r(t_m)$

Example 11.3. For example, if we have times 23.5, 34.0+, 34.0, 39.1+, 43.7, this yields Table 1

The following code calculates a Kaplan-Meier survival curve both directly from Table 1, and using the R function `survfit()`. See Figure 11.2. Note that censored observations are conventionally indicated in a plot by the symbol '+'.

Table 1: Survival table for example .

i	t_i	d_i	$r(t_i)$	\hat{p}_i
0	0	0	5	$(5-0)/5 = 1$
1	23.5	1	5	$(5-1)/5 = 4/5$
2	34.0	1	4	$(4-1)/4 = 3/4$
2	39.1	0	2	$(2-0)/2 = 1$
3	43.7	1	1	$(1-1)/1 = 0$

```
# Simple KM curve example. We'll need the 'survival' library.

library(survival)

# Note the option pty='s', to create a step plot.

par(mfrow=c(1,2),pty='s')

#
# Values from survival table.
#

p.hat = c(1,4/5,3/4,1,0)
km.time = c(0,23.5,34.0,39.1,43.7)

#
# Calculate cumulative properties with R function cumprod()
#

km.curve = cumprod(p.hat)
#
# Now draw plot
#

plot(km.time, km.curve, type='s', xlab='Time', ylab='Survival')
title("Cumulative probabilities")

#
# In R, censored observations are represented by the object Surv(time, event, type="right").
#
# time = observed time
# event = 1 if time is complete, = 0 if time is censored.
#     Intuitively, event = 1 if a "death" or "failure" has occurred.
# type = type of censoring, right censoring being the default.
#
```

```

#
# Data entered as follows: z = observed time, ev = 0 if observation is censored.
#
z = c(23.5, 34.0, 34.0, 39.1, 43.7)
ev = c(1,0,1,0,1)

#
# The R survfit function has several uses. When a formula is entered it
# creates a KM survival curve, but it also will produce a survival curve
# for a previously fitted Cox model. See help(survfit) for more.
#
# Note that in this example, there are no predictors. So the formula is
# Surv(z,ev)^1.
#
# Note also that with the mark.time = TRUE option a "+" symbol will
# mark each censored time.
#
plot(survfit(Surv(z,ev)^1), xlab='Time', ylab='Survival', mark.time = TRUE)
title("Using R survfit() function")

```

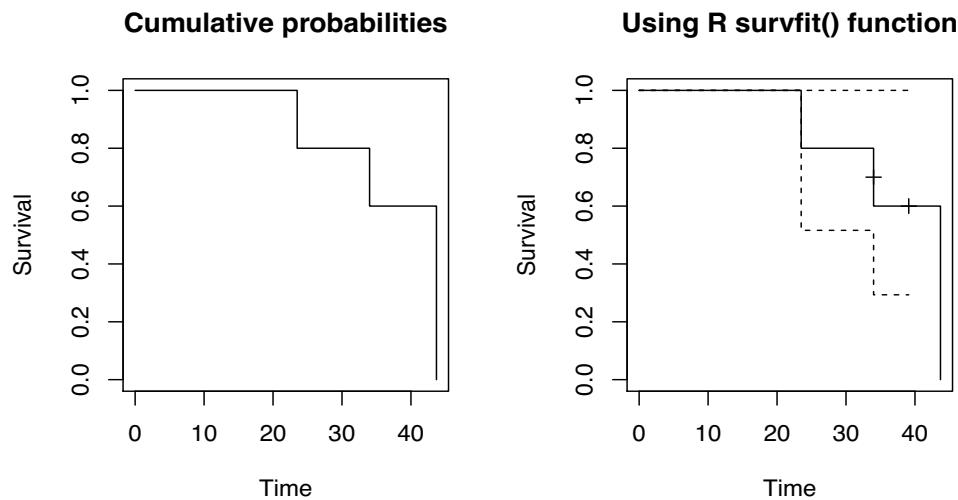


Figure 11.2: Plot for Example 1



11.3.3 Cox proportional hazards regression

There are various ways to incorporate linear models into survival analysis. One of the most widely used is the *Cox proportional hazards regression model* (CPH). We have the same linear prediction term used in Gaussian linear models and logistic regression:

$$\eta = \beta_1 x_1 + \dots + \beta_p x_p. \quad (11.11)$$

There is a *baseline hazard function* $h_0(x)$. The crucial assumption is that for any set of predictions, there is an associated survival time with a hazard function $h(x) \propto h_0(x)$, the exact relationship being

$$h(x) = h_0(x)e^\eta. \quad (11.12)$$

The data consists of observed survival times (possibly censored) associated with a set of predictors. Note that the intercept term β_0 is not needed. The important quantity is the *hazard ratio*

$$HR = e^{\eta - \eta'}, \quad (11.13)$$

where η and η' are the linear prediction terms for two sets of predictors. This plays a similar role to the odds ratio in logistic regression (Section 10.1).

Despite the apparent difference in structure between the CPH model and other linear models already considered, the actual process of model building may be quite similar, and the R function `coxph()` (from the `survival` package) outputs a coefficient table very similar to that produced by `lm()` or `glm()` (the function used for logistic regression).

We will give a brief introduction to using the CPH model in R. A more detailed tutorial can be found in the demonstration software file `SURVIVAL.R`. We will make use of the `Melanoma` data set from the `MASS` library. The `help(Melanoma)` documentation is printed below:

```
Melanoma {MASS} R Documentation
Survival from Malignant Melanoma
```

Description

The `Melanoma` data frame has data on 205 patients in Denmark with malignant melanoma.

Usage

```
Melanoma
Format
```

This data frame contains the following columns:

```
time
survival time in days, possibly censored.

status
1 died from melanoma, 2 alive, 3 dead from other causes.
```

sex
 1 = male, 0 = female.

age
 age in years.

year
 of operation.

thickness
 tumour thickness in mm.

ulcer
 1 = presence, 0 = absence.

Source

P. K. Andersen, O. Borgan, R. D. Gill and N. Keiding (1993) Statistical Models based on Counting Processes. Springer.

The survival times are contained in **time**, and the censoring status can be obtained from **status**. The object will then be to model the effect on survival time of melanoma patients of various predictors, which may include **sex**, **age**, **year**, **thickness** or **ulcer**. The descriptions are given above.

Here, observations with **status == 3** (dead from other causes) are removed for the analysis, although the associated survival times may conceivably be interpreted as censored. Then **status == 2** (alive) signifies a censored survival time. Once the event indicator is created (in **ev**, say), it may be used to create the survival time response object **Surv(time, ev)** and used in a formula.

```
> #
> # We will need the following libraries:
> #
>
> library(MASS)
> library(survival)
>
> #
> # Remove status = 3
> #
>
> Melanoma2 = subset(Melanoma, status!=3)
>
> #
> # Add Surv object event to the current dataframe.
> # We will then use Melanoma3
```

```

>
> head(Melanoma2)
  time status sex age year thickness ulcer
3    35      2   1  41 1977       1.34      0
5   185      1   1  52 1965      12.08      1
6   204      1   1  28 1971       4.84      1
7   210      1   1  77 1972       5.16      1
9   232      1   1  49 1968      12.88      1
10  279      1   0  68 1971       7.41      1
> Melanoma3 = data.frame(Melanoma2, ev = 1*(Melanoma2$status==1))
> head(Melanoma3)
  time status sex age year thickness ulcer ev
3    35      2   1  41 1977       1.34      0  0
5   185      1   1  52 1965      12.08      1  1
6   204      1   1  28 1971       4.84      1  1
7   210      1   1  77 1972       5.16      1  1
9   232      1   1  49 1968      12.88      1  1
10  279      1   0  68 1971       7.41      1  1
>
> #####
> ##### Use Cox PH to build multiple regression models
> #####
>
> ##### sex alone
>
> fit = coxph(Surv(time,ev) ~ sex, data=Melanoma3)
> summary(fit)$coef
      coef exp(coef)  se(coef)      z  Pr(>|z|)
sex 0.7007396  2.015243 0.2651396 2.642908 0.008219743
>
> ##### sex and thickness
>
> fit = coxph(Surv(time,ev) ~ sex+thickness, data=Melanoma3)
> summary(fit)$coef
      coef exp(coef)  se(coef)      z  Pr(>|z|)
sex      0.6279349  1.873737 0.26521309 2.367662 1.790090e-02
thickness 0.1602204  1.173769 0.03236341 4.950664 7.396087e-07
>
> ##### all main effects and 2nd order interactions
>
> fit = coxph(Surv(time,ev) ~ (sex+thickness+age+ulcer)^2, data=Melanoma3)
> summary(fit)$coef
      coef exp(coef)  se(coef)      z  Pr(>|z|)
sex        1.900499083 6.6892321 1.051709628 1.8070568 0.07075342
thickness -0.159887271 0.8522399 0.215200367 -0.7429693 0.45750025

```

```

age          0.026088647 1.0264319 0.019929315 1.3090589 0.19051440
ulcer        1.990570432 7.3197080 1.250533580 1.5917769 0.11143485
sex:thickness 0.114050523 1.1208088 0.092297031 1.2356900 0.21657380
sex:age       -0.025583430 0.9747411 0.016974041 -1.5072092 0.13175705
sex:ulcer     -0.641314413 0.5265998 0.636445158 -1.0076507 0.31362218
thickness:age 0.003040403 1.0030450 0.002664797 1.1409509 0.25389036
thickness:ulcer 0.050266714 1.0515515 0.098163899 0.5120692 0.60860255
age:ulcer    -0.012165871 0.9879078 0.021875595 -0.5561390 0.57811587
>

```

From the first model `Surv(time, ev) ~ sex`, the coefficient associated with the sole predictor `sex` is estimated to be $\beta_1 = 0.7007396$. Note that `sex` is an indicator variable, with `sex = 1/0` for males/females. So for this example, Equations (11.11), (11.12) and (11.13) can be readily interpreted. The linear term itself is simply

$$\eta = \beta_1 \times \text{sex}$$

The hazard function for females (`sex == 0`) is

$$h_{females}(x) \approx h_0(x)e^{\beta_1 \times 0} = h_0(x),$$

while for males (`sex == 1`) it is

$$h_{males}(x) \approx h_0(x)e^{\beta_1 \times 1} = h_0(x)e^{0.701}.$$

It is important to note that, unlike linear or logistic regression, the mean of the response plays no role in the actual fitting method. However, we can, in principle, obtain the mean survival time from the hazard function, since it is a complete representation of the survival density. However, in practice, the mean is often considered of secondary importance, since survival densities are often highly skewed, and quantiles or tail probabilities (ie survival probabilities) are usually more readily interpreted.

In an analysis like that illustrated in our example, interest is more likely to be in the effect of predictor variables on survival rates. Here, we set η_{males} and $\eta_{females}$ to be the predictor terms of the two classes of subjects. Then the hazard ratio between classes is, from Equation (11.13),

$$HR = e^{\eta_{males} - \eta_{females}} = e^{(\beta_1 \times 1 - \beta_1 \times 0)} = e^{\beta_1} \approx e^{\hat{\beta}_1} = e^{0.701} = 2.015.$$

Thus, the hazard rate of males is estimated to be about twice that of females. Also, note that from the coefficient table the P -value for the two-sided alternative against null hypothesis $H_0 : \beta_1 = 0$ is $P \approx 0.008$, so that the equivalent null hypothesis $H_0 : HR = 1$ is rejected at this significance level. A confidence interval for HR is also obtainable from the standard error $S_{\hat{\beta}_1} = 0.2651396$, so that an approximate 95% confidence interval would have bounds

$$CI_{HR} = e^{\hat{\beta}_1 \pm 2S_{\hat{\beta}_1}} = e^{0.701 \pm 0.532} = (1.19, 3.42).$$

11.4 Postscript

Among the canon of statistical methodology, survival analysis tends to be more of a specialty. It is not covered in many general textbooks recommended by this author, for example, James *et al.* (2013) or Neter *et al.* (1996). However, anyone interested in building predictive models or classifiers must anticipate that the target responses may be *time to event* observations (cancer prognosis, waiting time in a queue, etc). Cox and Oakes (1984) is a good primary text, and Van Belle *et al.* (2004) provides an excellent applied introduction, ideal for the student familiar with the principles of linear models, but unfamiliar with survival analysis. As for most other topics considered here, Venables and Ripley (2013) gives a good introduction to R survival analysis functions, as well as a compact but effective introduction to the mathematics underlying Kaplan-Meier estimation, and the Cox proportional hazards model. Like the theory itself, R functions supporting survival analysis tend to have their own structures and conventions, which must be taken into account when mastering this topic.

A knowledge of survival analysis is important to the data science analyst, in that it extends expertise to cover a class of applications which will almost certainly be encountered sooner or later. But it is also an important companion to the analysis of positive random variables, which arise not only as time to event data, but quite frequently in finance, econometrics, and engineering reliability.

What is often of interest here is not means and variances, but tails of density functions. Note that survival analysis centers around the failure, or hazard, rate, and little use is made of means and variances (Section 11.2). The crucial property becomes the rate at which the upper density tail approaches zero. For the normal density this is quite fast, in particular, $\propto e^{-x^2/\sigma^2}$ as $x \rightarrow \infty$. On the other hand, the *Pareto density* is proportional to $1/x^k$, $k > 1$, and so convergence to zero as $x \rightarrow \infty$ is much slower (the Pareto distribution is also referred to as a *power-law distribution*). The term *heavy-tailed distribution* for a positive random variable X has various conventions. The most precise definition requires that the distribution not possess a moment generating function (meaning that the expected value $E[e^{tX}] = \infty$ for all $t > 0$). It can be more informally used to describe distributions with “heavier tails” than the normal, meaning those that decrease more slowly to zero.

Heavy-tailed distributions typically do not possess all finite moments (although important exceptions exist, such as the log-normal distribution), meaning that $E[X^r] = \infty$ for all r above some positive integer. For the Pareto distribution, $E[X^r] < \infty$ if and only if $r < k - 1$, and so does not possess a mean if $k \leq 2$, or a variance if $k \leq 3$.

Monetary quantities tend to have heavy-tailed distributions. Part of the reason for this is that pools of money tend to grow exponentially (for example the effect of interest rates). Exceptionally large monetary values also appear when money beyond that needed for basic needs is reinvested. One characteristic of heavy-tailed distributions is that such “outliers” can be predicted as part of a distribution, rather than as an anomaly. See, for example, Ibragimov *et al.* (2015) for further reading.

This type of positive random variable also emerges in engineering system reliability for the purpose of modeling time to failure of a component or process. A very good introduction can be found in Ross (2014), with more advanced statistical methods described in Meeker and Escobar (2014).

Chapter 24 contains practice problems for survival analysis. Most provide practice in the application of the Kaplan-Meier survival curve estimate or the Cox proportional hazards model. Problem 24.5 emphasizes the crucial proportional hazards assumption. Problem 24.6 illustrates how

time to failure observations arise from engineering systems. In addition, Problem 27.2 describes an engineering application making use of the Weibull distribution discussed in Example 11.2.

Chapter 12

Bayesian Inference

Suppose random data X is observed, which possesses a density $f(x | \theta)$ from a family of models parametrized by $\theta \in \Theta \subset \mathbb{R}^p$, where Θ is known as the parameter space.

Most modeling techniques we have seen attempt to minimize prediction error. We have also seen the maximum likelihood principle. These methods can be modified to incorporate complexity penalties, but what they have in common is that the selected model is the one which optimizes some criterion.

Technically, the main difference between Bayesian inference and these other methods is that optimization is replaced by integration. In likelihood, we regard the quantity $l(\theta) = f(x | \theta)$ as a modeling criterion to be optimized *with respect to* θ . In Bayesian inference, θ itself is taken to be a random variable or vector. To formalize the idea, the following framework is adopted. We assume there is a *prior density* $\pi(\theta)$ for θ . This describes the range of possible values for θ , and an initial description of their relative plausibility, sometimes referred to as *belief* (or *prior belief*). This might be based on some model, or it may be entirely subjective.

We have seen exactly this form of inference before in the Bayes classifier (Chapter 16.6). Given classes $j = 1, \dots, m$ we have prior probabilities π_1, \dots, π_m . In a sense we can think of class j as the parameter θ within the set of all classes $\Theta = \{1, \dots, m\}$. We then have posterior probability, given the data

$$P(j | x) = \frac{f(x | j)\pi_j}{\int_{\Theta} f(x | \theta)\pi(\theta)d\theta} = \frac{f(x | j)\pi_j}{\sum_{j=1}^m f(x | j)\pi_j}.$$

In other words, we have a prior distribution on the space of all models, and this distribution is altered by conditioning on data (or evidence) to yield the posterior distribution.

In much the same way, if $\pi(\theta)$ is a continuous density on a parameter space Θ in \mathbb{R}^p , we would have posterior density

$$\pi(\theta | x) = \frac{f(x | \theta)\pi(\theta)}{\int_{\Theta} f(x | \theta)\pi(\theta)d\theta}. \quad (12.1)$$

12.1 The Bayes Estimator

In Chapter 16.6, we noted that the Bayes classifier minimized classification error. A similar result holds for Bayesian inference in general.

The posterior distribution is the basis for Bayesian inference, but it is usually more convenient to refer to a single point estimate. There is a well developed theory behind this problem, which

we briefly introduce. Recall from Section 15.5 the idea of *loss* $L(x, y)$ and *risk* $R = E[L(x, y)]$. That discussion was in the context of prediction, that is, the construction of a predictor y which is meant to be close to x , based on any available feature data. The ideas are much the same for estimation, in which $x = \theta$ is an unknown parameter to be estimated, and y is the estimator. The loss function serves the same purpose, and is usually taken to be squared error $L(x, y) = (x - y)^2$, although absolute deviation $L(x, y) = |x - y|$ is a commonly used alternative. In any case, we generally assume $L(x, x) = 0$. Next, suppose $\hat{\theta}$ is an estimator of θ . Risk is then

$$R(\theta, \hat{\theta}) = E_{\theta}[L(\theta, \hat{\theta})]$$

where the expectation is calculated assuming that θ is the correct parameter value. Any loss function may be used, but then, of course, the risk depends on that choice.

Risk is used to measure the accuracy of an estimator. We generally wish risk to be small, but we also want this to hold in some sense over the entire parameter space $\theta \in \Theta$. For example, suppose we wish to estimate θ from distribution $N(\theta, \sigma^2)$, based on observation $X \in N(\theta, \sigma^2)$. Clearly, estimator $\hat{\theta}$ cannot depend on θ , but it should depend on observation X . Suppose, ignoring this advice, we set $\hat{\theta} \equiv 10.51$ for any value of X . In fact, this would be an excellent estimator if, indeed, θ was equal to 10.5, since $R(10.51, \hat{\theta}) = 0$. But we can't expect this, and $R(\theta, \hat{\theta})$ would be very large for most θ (not near 10.51).

There are a number of ways to use risk to formulate coherent criterion for the selection of estimators. For example, suppose we have an *iid* sample from $N(\theta, \sigma^2)$. It can be shown that among unbiased estimates of θ , that is, estimators for which

$$E_{\theta}[\hat{\theta}] = \theta$$

for all $\theta \in \Theta$, the sample mean \bar{X} has uniformly minimum squared error risk over $\theta \in (-\infty, \infty)$ (the estimator $\hat{\theta} \equiv 10.51$ is not unbiased).

Bayesian inference provides a natural method of using risk to select estimators. We first integrate risk over the prior distribution

$$B(\pi, \hat{\theta}) = \int_{\theta \in \Theta} R(\theta, \hat{\theta}) \pi(\theta) d\theta,$$

a quantity known as *Bayes risk*. We interpret $\hat{\theta}$ as a *decision rule* which makes use of data to be collected. Bayes risk expresses the expected performance of the decision rule *before* the data is collected, and so is a property of an inference method, rather than a summary of a particular inference. It also depends on the prior distribution. The problem, then, is to determine the estimator $\hat{\theta}$ (or decision rule) which minimizes Bayes risk $B(\pi, \hat{\theta})$. This is known as the *Bayes estimator*. It turns out that a very elegant solution to this problem exists. If L is squared error loss then the Bayes estimator is the mean of the posterior distribution:

$$\hat{\theta}_{MSE} = \int_{\theta \in \Theta} \theta \pi(\theta | x) d\theta,$$

which minimizes mean squared error MSE in the sense that it minimizes the mean squared error loss over the prior distribution. Similarly, if L is absolute deviation, then the Bayes estimator is the median of the posterior distribution:

$$\hat{\theta}_{MAD} = median[\pi(\theta | x)]$$

which minimizes mean absolute deviation MAD in the sense that it minimizes the mean absolute error loss over the prior distribution.

12.2 Bayesian Inference for the Binomial Distribution

Suppose θ is a probability p in a binomial distribution $bin(n, p)$. If we have no reason to favor one choice of p over the other, we might set $\pi(p)$ to be the uniform distribution on $[0, 1]$. This would correspond to the uniform prior discussed in Section 16.6.1. However, a quite rich theory of Bayesian inference for this problem exists.

12.2.1 The gamma and beta functions

First, the *gamma* function is defined by the definite integral

$$\Gamma(t) = \int_{x=0}^{\infty} x^{t-1} e^{-x} dx, \quad t > 0.$$

It can be shown that

$$\Gamma(t+1) = t\Gamma(t),$$

and since $\Gamma(1) = 1$ we have

$$\Gamma(n) = (n-1)!$$

for integers $n = 1, 2, 3, \dots$. The gamma function can therefore be thought of as a generalization of the factorial. In addition, we have

$$\Gamma(1/2) = \sqrt{\pi}.$$

Similarly, the *beta* function is defined by the definite integral

$$B(\alpha, \beta) = \int_{u=0}^1 u^{\alpha-1} (1-u)^{\beta-1} du, \quad \alpha, \beta > 0.$$

It can be shown that we always have

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

12.2.2 The beta distribution

The beta function is used to normalize the beta distribution $Z \sim beta(\alpha, \beta)$, which has density function

$$f(z | \alpha, \beta) = \frac{1}{B(\alpha, \beta)} z^{\alpha-1} (1-z)^{\beta-1}, \quad z \in [0, 1].$$

The beta distribution has support on the unit interval $[0, 1]$, and so it is useful for modeling quantities that are interpretable as random probabilities. If $Z \sim beta(1, 1)$ then Z is uniformly distributed on $[0, 1]$, otherwise, the beta family admits a wide variety of shapes. The mean and variance are important to note. We have

$$\begin{aligned} E[Z] &= \frac{1}{B(\alpha, \beta)} \int_{z=0}^1 z \times z^{\alpha-1} (1-z)^{\beta-1} dz \\ &= \frac{1}{B(\alpha, \beta)} \int_{z=0}^1 z^{(\alpha+1)-1} (1-z)^{\beta-1} dz \\ &= \frac{B(\alpha+1, \beta)}{B(\alpha, \beta)} \\ &= \frac{\alpha}{\alpha+\beta}, \end{aligned}$$

making use of the equalities noted above. The variance is given by

$$\text{var}[Z] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}.$$

It is instructive to reparametrize the beta density, that is, to construct a one-to-one mapping between parameter pairs (α, β) and, say, (ρ, T) such as

$$\begin{aligned}\rho &= \frac{\alpha}{\alpha + \beta}, \\ T &= \alpha + \beta.\end{aligned}\tag{12.2}$$

Under the new parametrization we have

$$\begin{aligned}E[Z] &= \rho, \\ \text{var}[Z] &= \frac{\rho(1 - \rho)}{T + 1}.\end{aligned}$$

This can be compared to the estimator of a binomial proportion $\hat{p} = X/n$, where $X \sim \text{bin}(n, p)$, which has mean and variance p and $p(1 - p)/n$, respectively (see Chapter 15 of the CSC252 lecture notes). Clearly, \hat{p} resembles $Z \sim \text{beta}(\alpha, \beta)$ where, under the parametrization of (12.2), ρ can be equated with p and T can be equated with $n - 1$.

12.2.3 Posterior distributions

Now, suppose we use $\text{beta}(\alpha, \beta)$ as the prior density $\pi(p)$ for a binomial parameter p , to construct a posterior distribution for p conditional on observation $X \sim \text{bin}(n, p)$. If we wish to use an uninformative prior (Section 16.6.1), we have the uniform prior $p \sim \text{beta}(1, 1)$. On the other hand, if we believe that p is close to some value p_{prior} , we set $\rho = p_{\text{prior}}$ in (12.2). The remaining parameter T reflects the level of certainty we have in this prior assumption, with larger values of T representing greater certainty. In a sense, T can be calibrated by comparison with the sample size n used in the binomial parameter estimate \hat{p} (although \hat{p} is not actually used in this analysis).

Interpreting X as having a binomial distribution conditional on p , we write

$$P(X = x | p) = \binom{n}{x} p^x (1 - p)^{n-x},$$

leading to posterior distribution

$$\begin{aligned}\pi(p | x) &= \frac{P(X = x | p)\pi(p)}{\int_{p=0}^1 P(X = x | p)\pi(p)dp} \\ &= \frac{\binom{n}{x} p^x (1 - p)^{n-x} \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1 - p)^{\beta-1}}{\int_{p=0}^1 \binom{n}{x} p^x (1 - p)^{n-x} \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1 - p)^{\beta-1} dp}.\end{aligned}\tag{12.3}$$

Although this expression seems complicated, it is actually quite simple, as long as the objective is kept in mind, which is to derive a density of p . We can always express (12.3) in the form

$$\pi(p | x) = K g(p),$$

where K does not depend on p . This is sometimes written as a proportional relationship

$$\pi(p | x) \propto g(p).$$

We can then renormalize to get

$$\pi(p | x) = \frac{g(p)}{\int_{p=0}^1 g(p) dp},$$

and the normalization constant may have a convenient form. Clearly, from (12.3) we have

$$\pi(p | x) \propto p^{x+\alpha-1} (1-p)^{n-x+\beta-1}.$$

In other words, the posterior density of p given observation $X = x$ is $\text{beta}(x + \alpha, n - x + \beta)$. This is an example of a *conjugate prior*, for which the prior and posterior distributions are in the same parametric family. As a technical matter, note that the quantities $\binom{n}{x}$ and $1/B(\alpha, \beta)$ in (12.3) play no role, since they do not depend on p (in fact, they appear in both the numerator and denominator, and so cancel). In particular, we do not need to explicitly evaluate the integral in the denominator.

In the case of the binomial parameter p with $\text{beta}(\alpha, \beta)$ prior and observation $X \sim \text{bin}(n, p)$, since the posterior density of p given observation $X = x$ is $\text{beta}(x + \alpha, n - x + \beta)$, the Bayes estimator with respect to squared error loss is

$$\hat{p}_{MSE} = \frac{x + \alpha}{n + \alpha + \beta}.$$

12.3 Postscript

The subject of Bayesian inference is a foundational one, and is best studied in the context of a more general theory of statistical inference (Section 1.2). In these notes, Bayesian ideas will be revisited in Chapter 14 in the context of *computational Bayesian methods*. This field might be said to originate in the practical problem of evaluating the posterior density of Equation (12.1) when the denominator (or normalization constant) cannot in practice be calculated. In this case, samples from this density can be simulated without requiring that constant, which is the subject of Chapter 14. It can be said that a problem turns into an opportunity, since this method provides a convenient solution to a large number of nonstandard inference problems, especially those using data which is too sparse to admit more conventional methods (see, for example, Problem 27.2).

Bayesian ideas will also be applied to the problem of classification (Chapter 16), and we argue there that a complete understanding of classification is impossible without them.

Practice problems in Bayesian inference are available in Chapter 25, and further examples can be found in demonstration software file `BAYESIAN-INFERENCE.R`. Chapters 14 and 16 cite further resources.

Chapter 13

Simulation Methods

Simulation methods play an important role in modern statistical practice. They provide a ready means of obtaining P -values and confidence intervals when analytical methods would be impractical, or even unavailable. Or, to be perfectly frank, they can simply save the analyst valuable time. And it is a good habit to test one's algorithm using simulated data sets, since their output can be compared to what is known to be the correct answer (sometimes referred to as an *oracle*). Demonstration software file MODEL-SELECTION-LASSO.R gives an extended example of this.

Simulation methods can involve either to simulating from known distribution, or resampling from existing data. We discuss here primarily resampling methods, then review some of the basic principles of the computational simulation of random variables.

Much of the theory we have seen is based on approximations to the normal distribution. We either assume that the measurements we collect are normally distributed, or that the sample size is large enough for the Central Limit Theorem to apply to the test statistic. This applies also to test statistics with a χ^2 distribution used for categorical data, with the additional assumption that each category count is large enough.

There is very good reason to study normal-based theory, since much of it is provably optimal when the assumptions are satisfied, as they often are. However, these assumptions will often prove problematic, and even when they may hold, it might not be practical to verify them, for example, when a procedure is to be repeatedly applied to a large number of cases. This is generally the case in the analysis of, for example, gene expression data.

We then briefly consider two forms of simulation methods which do not rely on distributional assumptions, and which are generally applicable.

13.1 Permutation Test

A *permutation test* is a hypothesis test in which a null distribution is created by a random permutation of the data. Consider the following paired data,

```
X = 16.1 31.5 21.5 22.4 20.5 28.4 30.3 25.6 32.7 29.2 34.7  
Y = 4.41 6.81 5.26 5.99 5.92 6.14 6.84 5.87 7.03 6.89 7.87
```

for which the Pearson correlation coefficient is $r_{obs} = 0.939$. Suppose we randomly permute one of the variables (say, Y), then recalculate r . We can generate a random permutation with the `sample()` function:

```
> sample(11)
[1] 8 9 4 5 6 11 3 2 1 7 10
>
```

We can then permute Y , then recalculate r :

```
> Yrandom = Y[sample(11)]
> X
[1] 16.1 31.5 21.5 22.4 20.5 28.4 30.3 25.6 32.7 29.2 34.7
> Yrandom
[1] 4.41 6.81 6.84 7.03 6.14 6.89 5.26 7.87 5.87 5.99 5.92
> cor(X,Y)
[1] 0.9388037
> cor(X,Yrandom)
[1] 0.1196821
>
```

The correlation of the permuted data, $r^* = 0.1196821$, is much smaller than the original $r_{obs} = 0.939$. This number is quite relevant, however. Under the null hypothesis $H_0 : \rho = 0$, there is no association between the paired variables X and Y . Therefore, if H_0 is true, the observed sample correlation r_{obs} should be comparable to a correlation coefficient r^* produced by randomly permuting the data. This gives directly a test procedure that does not require any distribution assumptions. We can estimate the *null distribution* of r^* by repeatedly permuting the data, and then compared r_{obs} to this distribution, either by comparing it to a critical value of the null distribution, or by estimating the appropriate tail probability to obtain a p -value.

We first simulate r^* $N = 50,000$ times, and display the distribution in a histogram (Figure 13.1).

```
> r.perm = rep(NA,50000)
> for (i in 1:50000) {r.perm[i] = cor(X,Y[sample(11)])}
>
> hist(r.perm, nclass=25)
> lines(rep(0.735,2), c(0,5000), col=4)
> lines(rep(-0.735,2), c(0,5000), col=4)
> text(-0.735,5500,"r = -0.735")
> text(0.735,5500,"r = 0.735")
```

We can show that the critical value $r_{\alpha/2}$ for a two-sided test against $H_0 : \rho = 0$ ($\alpha = 0.01$, $n = 11$) was $r_{\alpha/2} = 0.735$, that is we reject H_0 if $|r_{obs}| \geq 0.735$. This critical value is shown in Figure 13.1. This means that under the null distribution, the correlation coefficient satisfies:

$$P(|r| \geq 0.735 \mid \rho = 0) = 0.01.$$

We can estimate the same probability for r^* from the simulated null distribution:

```
> mean(abs(r.perm) >= 0.735)
[1] 0.00862
```

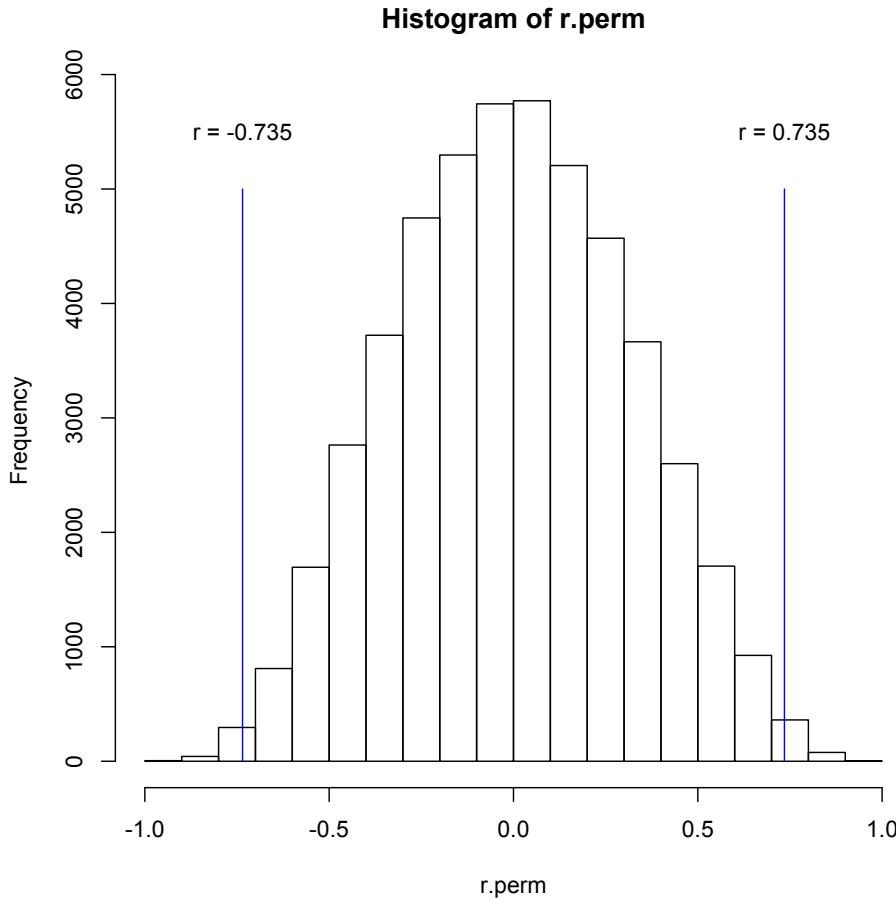


Figure 13.1: Histogram of 50,000 replications of r^* . The critical values $r_{\alpha/2}$, $-r_{\alpha/2}$ for a two-sided test against $H_0 : \rho = 0$ ($\alpha = 0.01$, $n = 11$) are superimposed.

so that

$$P(|r^*| \geq 0.735 \mid \rho = 0) \approx 0.0082,$$

which is close to $\alpha = 0.01$. In fact, the level 95% margin of error of an estimate of a proportion $p = 0.01$ with $n = 50,000$ is

$$ME = 1.96 \sqrt{\frac{0.01 \times 0.99}{50000}} = 0.00087.$$

Judging from the margin of error, the tail probabilities for 0.735 is close to, but slightly less than, $\alpha = 0.01$. We can obtain a critical value for r_{obs} based on the distribution of r^* using the `quantile()` function:

```
> quantile(abs(r.perm), 0.99)
  99%
0.7257387
```

```
> quantile(r.perm, 0.995)
  99.5%
0.734525
> quantile(r.perm, 0.005)
  0.5%
-0.7181912
>
```

Note that we can obtain the $\alpha/2 = 0.005$ critical value for r^* ($r_{0.005}^* = 0.734525$), the lower tail $1 - \alpha/2 = 0.995$ critical value ($r_{1-0.005}^* = -0.7181912$), or the $\alpha = 0.01$ critical value for $|r^*|$ ($|r^*|_{0.01} = 0.7257387$). If the null distribution is symmetric, which we would expect in this case, the critical value $|r^*|_{0.01}$ should be used, since we would expect

$$|r^*|_\alpha \approx r_{\alpha/2}^* \approx -r_{1-\alpha/2}^*.$$

The p -value can be obtained by estimating the relevant tail probability of r_{obs} , in this case

$$P(r^* \geq r_{obs}), \quad P(r^* \leq r_{obs}), \quad \text{or } P(|r^*| \geq |r_{obs}|)$$

for an upper-tailed, lower-tailed or two sided test, respectively. However, it is usually the practice to add the observed statistics r_{obs} with the simulated values of r^* for this purpose, giving, for example:

$$P(|r^*| \geq |r_{obs}|) \approx \frac{\#\{|r^*| \geq |r_{obs}|\} + 1}{N + 1}. \quad (13.1)$$

This avoids p -values equal to zero, making the procedure somewhat conservative, although less so with increasing N . To assign a p -value to $r_{obs} = 0.939$, we can determine the numerator of (13.1) with the following command:

```
> sum(abs(r.perm) >= 0.939)
[1] 0
>
```

that is, no simulated value of r^* exceeds 0.939 in magnitude. This gives p -value

$$P \approx 1/50001 = 1.99996 \times 10^{-5}.$$

13.2 The Bootstrap Procedure

Suppose we are given a sample of size $n = 10$:

```
X = 36.1 16.1 16.7 32.7 33.9 21.8 15.5 26.0 37.8 18.6
```

A 95% confidence interval for the mean is given by

$$\bar{X} \pm t_{9,0.025} S / \sqrt{n} = 25.2 \pm 6.37 = (19.15, 31.89).$$

Remember that a confidence interval is a statement about a statistical method as well as a specific data set. If we could observe repeated samples collected under identical conditions, obtaining

repeated observations of \bar{X} , we could observe the distribution of \bar{X} directly, and form inference statements accordingly, without the need to specify a distribution.

The *bootstrap procedure* is a method of simulating such samples, thus obtaining an estimated *sampling distribution* of, for example, \bar{X} , or any other statistic of interest. This is done by the simple device of sampling, *with replacement*, from the original sample (of size n), a new sample of the same size n .

This can be done in R by the `sample()` function in the following way:

```
> n = 10
> sample(1:n, n, replace = TRUE)
[1] 1 6 4 2 3 5 5 8 8 3
>
```

(the command `sample.int(n, n, replace = TRUE)` will do the same thing). A *bootstrap sample* is then obtained by replacing the indices in the original sample:

```
> Xboot = X[sample(1:n, n, replace = TRUE)]
> X
[1] 36.1 16.1 16.7 32.7 33.9 21.8 15.5 26.0 37.8 18.6
> Xboot
[1] 33.9 21.8 16.1 37.8 36.1 15.5 16.7 32.7 21.8 16.1
> mean(X)
[1] 25.52
> mean(Xboot)
[1] 24.85
>
```



The bootstrap sample contains repeats, but we can still calculate most statistics for it. In this case, we get a new sample mean $\bar{X}_{boot} = 24.85$ close to, but not exactly equal to, to original observed sample mean $\bar{X}_{obs} = 25.52$. As for the permutation procedure, we may then obtain a simulated sample, shown as a histogram in Figure 13.2.

```
> xbar.boot = rep(NA,50000)
> for (i in 1:50000)
    {xbar.boot[i] = mean(X[sample(1:n, n, replace = TRUE)])}
> hist(xbar.boot, nclass=25)
>
```

To obtain a 95% confidence interval, we need only obtain the 0.025 and 0.975 quantiles from the bootstrap sample,

```
> quantile(xbar.boot, c(0.025, 0.975))
 2.5% 97.5%
20.40 30.81
>
```

yielding an estimated 95% confidence interval of $(20.40, 30.81)$, which is quite close to the confidence interval $(19.15, 31.89)$ obtained using the t -distribution above.

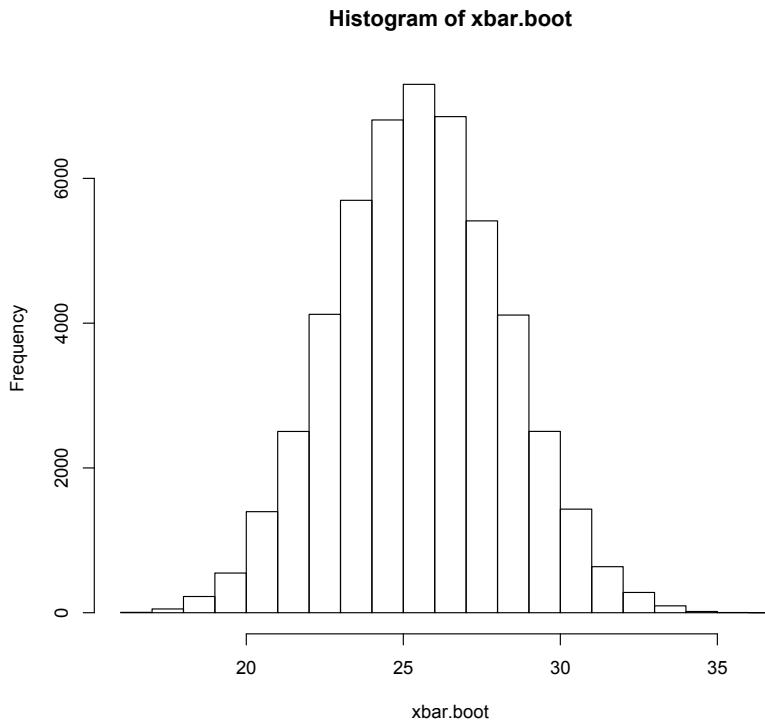


Figure 13.2: Histogram of 50,000 replications of \bar{X}_{boot} .

~~13.3~~ General Principles of Computer Simulation

R provides functions for simulating samples from a large class of distributions. For example, the function `rnorm()` simulates a (pseudo-) random sample of normal random variables of given mean and standard deviation:

```
> rnorm(5,mean=20,sd=0.1)
[1] 20.04352 20.13176 20.16265 20.23345 19.95649
```

See Section C.9 for a list of the relevant R functions. Although much of random number generation is transparent to the R user, it is still important to understand how this is done.

13.3.1 Pseudorandom number generation

Random number generation is usually performed by an iterative algorithm, which is, of course, deterministic, but also too chaotic to admit serial prediction. It can be thought of as a mapping T between two integers within the range $0, M - 1$, where M is very large, and usually a power of 2. If we start with x_0 , we produce the next pseudorandom variate $x_1 = T(x_0)$, then $x_2 = T(x_1)$, and so on. This iteration is repeated every time a new random number is requested. If we standardize the variates $u_i = x_i/M$, then the sequence u_1, u_2, \dots resembles for all practical purposes an *iid* sequence from a uniform distribution. In principle, any other random variable can be expressed as a transformation of a uniform random variable, so this forms the basis of a general approach

for the simulation of samples from any given distribution. Ross (2014) offers an excellent chapter on this topic. As for the design of random number generators, this is a highly technical topic. A good introduction can be found in Chambers (1977). Otherwise, a long but fascinating journey into this field can start with the command `help("RNG")`. In the meantime, it is worth exploring a simple but widely used method, which exemplifies the essential properties of pseudorandom number generation. We do this next.

13.3.2 Linear congruential generators

A *linear congruential generator* takes the form

$$x_{n+1} = (ax_n + b) \bmod P, \quad n = 0, 1, 2, \dots, \quad (13.2)$$

resulting in a sequence x_0, x_1, \dots . The numbers a , b and P are fixed integers, and their informed choice is crucial. The number P is the *period*, and the initial value x_0 is usually referred to as the *seed*. If a and b are both positive then the generator is *mixed congruential*, and if $b = 0$ the generator is *multiplicative congruential*. Keeping track of the seed is crucial for reproducibility, since two linear congruential generators with the same parameters a, b, P will generate the same sequence using the same seed. Most computing environments, including R, allow the user to specify the seed, and allowing this option is good practice when designing randomized algorithms. The period P determines the range of the linear congruential generator, since any evaluation $y = x \bmod P$ must be an integer between 0 and $P - 1$ inclusive.

Suppose we set $a = 2$, $b = 3$, $P = 20$, with seed $x_0 = 5$. Then (13.2) produces the sequence

$$\begin{aligned} x_0 &= 5 \\ x_1 &= (2 \times x_0 + 3) \bmod 20 = 13 \\ x_2 &= (2 \times x_1 + 3) \bmod 20 = 9 \\ x_3 &= (2 \times x_2 + 3) \bmod 20 = 1 \\ x_4 &= (2 \times x_3 + 3) \bmod 20 = 5 \\ x_5 &= (2 \times x_4 + 3) \bmod 20 = 13 \\ x_6 &= (2 \times x_5 + 3) \bmod 20 = 9 \\ &\vdots \end{aligned}$$

yielding sequence 5, 13, 9, 1, 5, 13, 9, Note that the sequence returns to the seed value $x_0 = x_4 = 5$ after 4 iterations of (13.2). As we would expect, the sequence begins to repeat itself, and will, in fact, repeat the sequence 5, 13, 9, 1 indefinitely. To what degree does the sequence depend on the seed? If we set $x_0 = 19$, we obtain sequence

$$19, 1, 5, 13, 9, 1, 5, 13, 9, \dots \quad (13.3)$$

so that the same four number sequence is repeated indefinitely. Interestingly, the sequence will never return to 19. On the other hand, if we set $x_0 = 12$ we obtain sequence

$$12, 7, 17, 17, 17, \dots \quad (13.4)$$

so that 17 will be repeated indefinitely, and the sequence will never return to 12 or 7.

Since the period is $P = 20$ we only need consider sequence elements $0, 1, \dots, 19$. If we set the seed in turn to each of these values, we find that when the seed is 2, 7, 12 or 17, the sequence eventually converges to 17, as in sequence (13.4). For all other seeds, the sequences will eventually cycle through 1, 5, 13, 9 as in sequence (13.3).

To understand this behavior, it is important to note that the sequence is deterministic, in the sense that each possible value possesses exactly one other value that can follow it. For 17, that number happens to be 17 itself, that is, it is a *fixed point*, which by definition satisfies

$$x = (2 \times x + 3) \bmod 20.$$

We can then see that whenever the sequence enters the cycle 1, 5, 13, 9 it will never leave, and whenever the sequence reaches 17, it will remain there indefinitely.

Clearly, the linear congruential generator possesses some interesting structure, but the type of behavior we have seen is clearly problematic, given its intended use. Ideally, we would like the sequence to avoid the type of behavior exemplified in (13.3) or (13.4), and consist of one single cycle of length P . This means that for any seed, the sequence will visit each possible value exactly once, and then return to the seed.

Fortunately, theory exists which informs the choice of parameters a, b, P . We have referred to P as the period. The attained period, as we seen, may be smaller. In the preceding example the attained period is either 4 or 1, depending on the seed. The largest possible attained period is called the *maximal period*, and we would like to have conditions under which the maximal period is attained. For linear congruential generators, this has been resolved with a great deal of precision and simplicity for the case $P = 2^i$. We present two results from Chambers (1977).

Theorem 13.1. Suppose in (13.2) $P = 2^i$.

For the mixed congruential generator ($a, b > 0$) the maximal period is P , and is attained if and only if

- (i) $a \bmod 4 = 1$,
- (ii) b is odd.

For the multiplicative congruential generator ($a > 0, b = 0$) the maximal period is 2^{i-2} , and is attained if and only if

- (i) $a \bmod 8 = 3$ or 5 ,
- (ii) the seed x_0 is odd.

The multiplicative congruential generator may be more advantageous when the computing platform must be economical.

Example 13.1. We examine two mixed congruential generators satisfying the conditions of Theorem 13.1. For both, we set $P = 2^{20}$, then consider separately $(a, b) = (936001, 102295)$ and $(5, 1)$ (we have $936001 \bmod 4 = 1$, for example).

The following R code produces plots with which to examine visually the degree of “randomness”. Figure 13.3 shows the first 100 elements of the sequence in various forms (top plots for $(a, b) = (936001, 102295)$, bottom plots for $(a, b) = (5, 1)$). Roughly, the first generator appears generally random throughout. The second generator appears similarly unordered, except for short sequences of similar magnitude. This is due to the relatively small values of a and b compared to P .

```

> x = 1:100
> y = sin(x)
> plot(x,y)
>
> par(mfrow=c(2,2))
>
> P = 2^20
> a = 936001
> b = 102295
>
> x = rep(NA, 100)
>
> x0 = 1
> for (i in 1:100) {
+   x0 = (x0*a + b) %% P
+   x[i] = x0
+ }
> plot(1:100, x)
> plot(1:100, x, type='l')
>
> a = 5
> b = 1
> x0 = 1
> for (i in 1:100) {
+   x0 = (x0*a + b) %% P
+   x[i] = x0
+ }
> plot(1:100, x)
> plot(1:100, x, type='b')

```

□

13.3.3 Uniform random number generation

If we compute the first 10,000 variates from the generator $(a, b, P) = (936001, 102295, 2^{20})$ of Example 13.1, the resulting histogram is shown in Figure 13.4. By Theorem 13.1 the maximal period $2^{20} = 1,048,576$ is attained, and so the histogram accordingly covers approximately the range $(0, 1048576)$. In fact, the largest value observed in the sequence was 1,048,383, just below P . Simulated random variables with a uniform distribution on $[0, 1]$ are given simply by dividing by P , that is

$$x_0/P, x_1/P, \dots, x_N/P$$

can be taken as a random sample from $\text{unif}[0, 1]$.

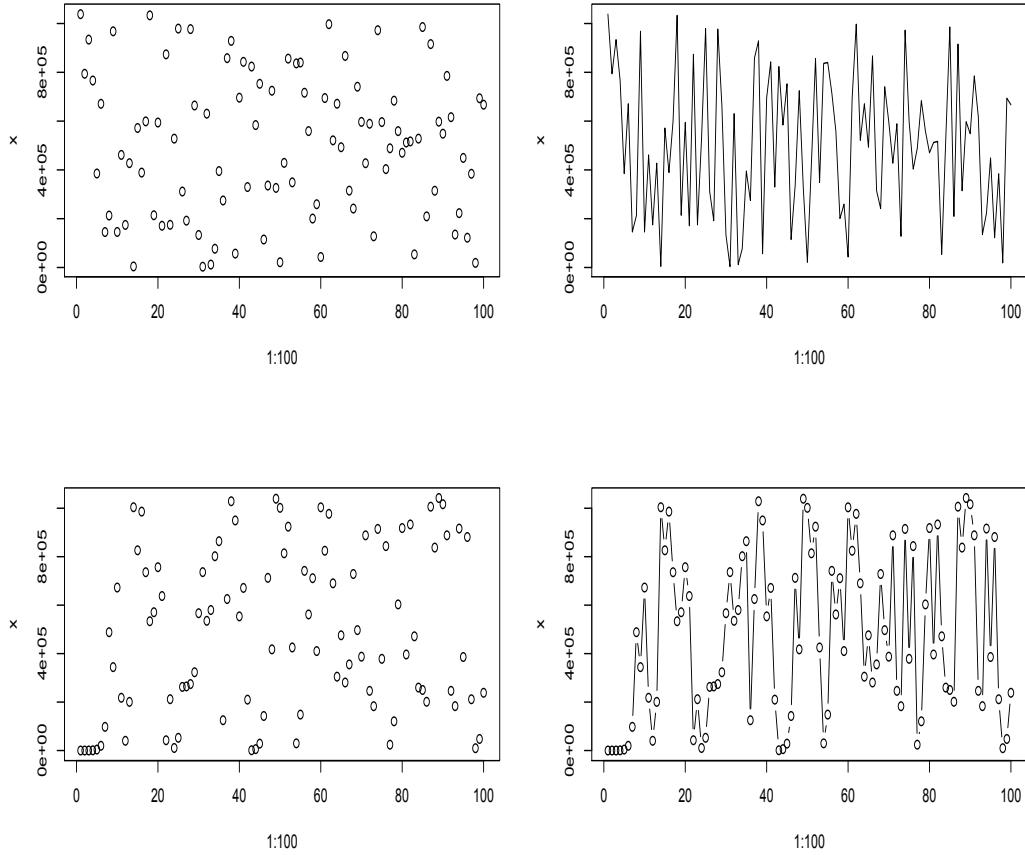


Figure 13.3: Sample sequences from linear congruential generators defined in Example 13.1. Top plots demonstrate $(a, b) = (936001, 102295)$, bottom plots demonstrate $(a, b) = (5, 1)$.

13.3.4 The inverse transformation method

Many simulation methods begin with uniform random variables. Suppose we are given a continuous CDF F and $U \sim \text{unif}[0, 1]$. Then F possesses inverse F^{-1} , since F is also increasing. This provides an elegant method of simulating a random variable with such a CDF.

Theorem 13.2. Suppose F is a continuous CDF, and $U \sim \text{unif}[0, 1]$. Then the CDF of RV

$$X = F^{-1}(U) \tag{13.5}$$

is exactly $F_X = F$. □

Proof. Under the hypothesis, the CDF of X is given by

$$X = F^{-1}(U). \tag{13.6}$$

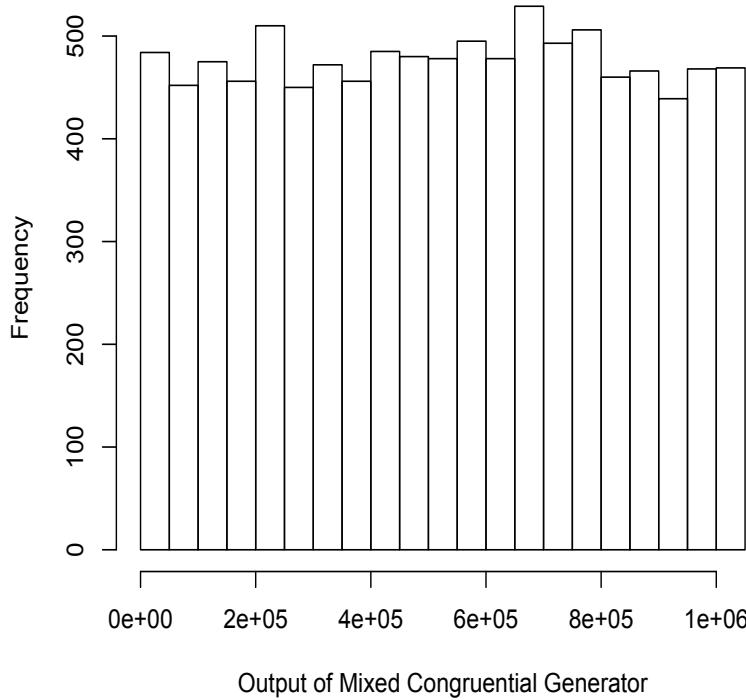


Figure 13.4: Histogram of the first 10,000 variates from the generator $(a, b, P) = (936001, 102295, 2^{20})$ of Example 13.1.

The CDF of X is then

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(F^{-1}(U) \leq x) \\ &= P(U \leq F(x)) \\ &= F_U(F(x)). \end{aligned}$$

The CDF of U is $F_U(u) = u$ for $u \in [0, 1]$, so that completing the analysis gives

$$F_X(x) = F(x),$$

which completes the proof. \square

A common application of this method is the simulation of exponentially distributed random variables.

Example 13.2. If $X \sim \exp(\lambda)$ then

$$F_X(x) = \begin{cases} 1 - \exp(-\lambda x) & ; \quad x \geq 0 \\ 0 & ; \quad x < 0 \end{cases}.$$

The inverse is obtained by solving

$$\begin{aligned} u &= 1 - \exp(-\lambda x), \\ \exp(-\lambda x) &= 1 - u, \\ x &= \frac{-\log(1 - u)}{\lambda}. \end{aligned}$$

So, given $U \sim \text{unif}[0, 1]$, if we set

$$X = F_X^{-1}(U) = \frac{-\log(1 - U)}{\lambda},$$

then $X \sim \exp(\lambda)$. Note that $1 - U$ may be replaced by U , since both have the same distribution. \square

13.3.5 Simulation of discrete random variables

The occurrence of an event with probability p is easily simulated with a random variable $U \sim \text{unif}[0, 1]$. For any interval $I \subset [0, 1]$, set

$$X = \begin{cases} 1 & ; \quad U \in I \\ 0 & ; \quad U \notin I \end{cases}$$

If I has length p , then $X \sim \text{Bern}(p)$.

The same idea may be used to simulate more complex discrete RVs. Suppose X has support $\mathcal{S}_X = \{0, 1, \dots, M\}$, and possesses PMF $P_X(i) = p_i$. Generate $U \sim \text{unif}[0, 1]$. If $U \in [0, p_0)$ set $X = 0$. If $U \in [p_0, p_0 + p_1)$ set $X = 1$. In general, set

$$X = \begin{cases} 0 & ; \quad U \in [0, p_0) \\ 1 & ; \quad U \in [p_0, p_0 + p_1) \\ \vdots & ; \quad \vdots \\ M & ; \quad U \in [p_0 + \dots + p_{M-1}, 1) \end{cases}.$$

This rule can be seen to be equivalent to

$$X = k \text{ if } U \in [F_X(k-1), F_X(k)) \tag{13.7}$$

and so is similar to the inverse transform method of the previous section.

Example 13.3. Suppose $X \sim \text{geom}(p)$. Then $F_X(k) = 1 - (1 - p)^k$. Then using the method of Equation (13.7) we have

$$\begin{aligned} X &= \min\{k : U > 1 - (1 - p)^k\} \\ &= \min\{k : k > \frac{\log(1 - U)}{\log(1 - p)}\} \\ &= 1 + \lfloor \frac{\log(1 - U)}{\log(1 - p)} \rfloor, \end{aligned}$$

where $\lfloor x \rfloor$ is the largest integer $i \leq x$, (referred to as the *floor function*). \square

13.3.6 Computer simulation and reproducibility in R

That pseudorandom variates are not actually random is something of an advantage. One drawback of simulation methods is that they are not completely reproducible, and two separate applications of the same simulation method on the same data will give answers that differ (slightly, hopefully). For this reason, a good habit to develop is to set the *random number seed* explicitly at the beginning of the algorithm, and to document its value. By *seed*, we simply mean the first variate x_0 of the sequence of pseudorandom variates, which in R can be done using the `set.seed()` function. Consider the following code:

```
>
> rnorm(5,mean=20,sd=0.1)
[1] 19.90912 20.01406 20.02100 19.97172 20.18680
> rnorm(5,mean=20,sd=0.1)
[1] 20.09122 20.05108 19.99625 20.04081 19.81416
> set.seed(2143)
> rnorm(5,mean=20,sd=0.1)
[1] 20.03002 20.01832 20.16643 19.89089 19.95767
> set.seed(2143)
> rnorm(5,mean=20,sd=0.1)
[1] 20.03002 20.01832 20.16643 19.89089 19.95767
>
```

The identical command `rnorm(5,mean=20,sd=0.1)` is applied four times. The first two applications of the command `rnorm(5,mean=20,sd=0.1)` produce distinct variates. However, before the third application, the seed is set explicitly with the `set.seed(2143)` command. If the seed is reset to the same value, as it is before the fourth application, the identical variates are produced.

13.3.7 Simulating dependent random variables in R

Sequences of pseudorandom variates can be regarded as independent. Certainly, this is a desirable property of a random number generator. Of course, sometimes we wish to generate random variables which possess some form of dependence. This can be a challenging topic, but we can review a number of available solutions. For example, Problem 22.11 provides a method of generating pairs of correlated normal random variables, assuming we have a method of generating independent variates (for example, function `rnorm()`). The R package `mvtnorm` can be used for simulating general multivariate normal and t -distributions, while the R package `bindata` simulates correlated vectors of binary random variables.

13.4 Postscript

This chapter provides only a brief introduction by example to a topic crucial to model statistical methodology. The situations for which analytical forms for P -values, standard errors or confidence intervals are convenient, practical, or even available, is quite limited. And even these are often highly dependent on distributional assumptions which may be difficult to validate. Simulation methods provide a very powerful answer to this issue.

Hopefully, the examples of this chapter demonstrate why the “learning by doing” approach works for this topic. The methods seem quite intuitive. The permutation test of Section 13.1 is a case in point. If paired observations are not correlated, then their pairing should be irrelevant. It is then a simple matter to randomly “shuffle” the pairings, to see if the data changes character in any important way.

We should, however, recognize what we are doing. A statistical hypothesis test is constructed from the distribution of the data under a null hypothesis. What the permutation test does is to define a null distribution, then estimate it by simulating a sample from it. Roughly, the null hypothesis is

H_0 : The distribution of the data does not depend on how certain observations are labeled.

and the distribution is simulated by randomly permuting those labels. But if this seems too easy, it often is. In particular, it must be carefully asked if a random permutation results in data which might plausibly be observed. See Good (2013) for a good source on this topic.

The other method considered in this chapter is the bootstrap. Again, the idea seems simple. If we wish to estimate the standard deviation of an estimator, we can always resample new data sets with properties similar to the dataset actually observed. We then recalculate the estimator for each one. The standard deviation of the resampled estimators should be approximately equal to the standard deviation we are looking for.

This is, however, a relatively recent idea (Efron, 1979), which at first struck some as counterintuitive (and was presented as such in a science article appearing in *The Economist* magazine). An excellent treatment of the bootstrap is given by its inventor in Efron and Tibshirani (1994). See also Efron and Gong (1983), Davison and Hinkley (1997) and Good (2005). It is important to note that the use of resampling methods in linear models requires special care.

Chapter 26 contains a number of practice problems on permutation and bootstrap procedures. The demonstration software file **SIMULATIONS.R** contains more examples of the permutation and bootstrap procedures. It also demonstrates R support for parallel computing, which may be particularly useful for these methods.

The implementation of resampling methods is often straightforward, given the emphasis of R on vectorized computation. However, we also note the R package **boot**, which provides a quite general method for implementing the bootstrap (Canty and Ripley, 2017). Its main function, also named **boot()**, has a variety of modes of use. We demonstrate here only a relatively simple application:

```
>
> x.obs = rnorm(100)
> boot(data = x.obs, statistic = function(x,d) {mean(x[d])}, R=1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

```
Call:
boot(data = x.obs, statistic = function(x, d) {
  mean(x[d])
}, R = 1000)
```

```
Bootstrap Statistics :
      original     bias   std. error
t1* 0.2161051 -1.483572e-05  0.09079445
>
```

A sample of $n = 100$ *iid* standard normal random variables is generated as data. The estimator in question is the sample mean. In the default mode, the `statistic` argument expects a function of two arguments. The first accepts the data, the second accepts the indices of a sample with replacement. There are then R replications. The correct standard error is $1/\sqrt{n} = 0.1$, for which the estimate given here is 0.09079445.

The practical value of such a method can perhaps be seen in the next example. If we want the standard error for an alternative estimator, the *IQR* (the difference between the 75th and 25th sample percentiles), we need only replace `mean` with `IQR` in the code:

```
>
> boot(data = x.obs, statistic = function(x,d) {IQR(x[d])}, R=1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

```
Call:
boot(data = x.obs, statistic = function(x, d) {
  IQR(x[d])
}, R = 1000)
```

```
Bootstrap Statistics :
      original     bias   std. error
t1* 1.534805 -0.04888736  0.1512865
>
```

Reasonable estimates of an essentially limitless variety of estimators are obtainable with little difficulty.

The demonstration software file `SIMULATIONS.R` contains further applications of the `boot` package.

Chapter 14

Markov Chains, MCMC and Computational Bayesian Methods

14.1 Markov Chains



can predict
average
interval
rate but
not precise

A *stochastic process* may be defined as a (possibly uncountable) indexed collection of random variables $\{X_t\}$, $t \in \mathcal{T}$. The index t usually represents time, although stochastic processes may also be used to describe random processes defined on some space.

Most stochastic processes are either *discrete time*, and take the form of a sequence X_1, X_2, \dots , or continuous time, and may be represented as a process $X[t]$ on a subset $t \in [0, \infty)$, with $X[t]$ being the value of the process at time t .

The *Markov chain* is a discrete time stochastic process. The defining property is the *memoryless property* or *Markovian property*, essentially, that the distribution of future states depends on the current state, but not on previous states.

Definition 14.1. Suppose we are given a discrete time stochastic process $X_n \in \mathcal{X}$, $i = 0, 1, 2, \dots$, which assumes values in a discrete *state space* \mathcal{X} . Without loss of generality we have either a finite state space $\mathcal{X} = \{0, 1, \dots, n\}$ or countable state space $\mathcal{X} = \{0, 1, \dots\}$. Then X_i is a *Markov chain* if the following *memoryless property* holds:

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \dots, X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j \mid X_n = i) = P_{ij}.$$

The quantity P_{ij} is called the *transition probability* from state i to state j . We also have *transition probability matrix* (or *transition matrix*)

$$P = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ \vdots & \vdots & \vdots & \\ P_{i0} & P_{i1} & P_{i2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

stochastic matrix

□

Row i of transition matrix P is equivalent to the conditional probability

$$P(X_{n+1} = j \mid X_n = i) = P_{ij}, \quad j \in \mathcal{X}.$$

Note also that P will be a matrix of infinite dimension when \mathcal{X} is countable. We also have no difficulty conceiving of P as ‘doubly infinite’ when the state space is the set of positive and negative integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$, which requires no important change of Definition 14.1.

Example 14.1. We start with an example of a two-state Markov chain, which, despite its simplicity, demonstrates a number of important features of Markov chains. Formally, we have state space $\mathcal{X} = \{0, 1\}$. However, we lose nothing by replacing the notation of Definition 14.1 with something more intuitive.

For example, the time index $i = 0, 1, 2, \dots$ may represent a sequence of days, and we may wish to define a simple infection model, in which state $i = 0$ represents a *healthy state H* and $i = 1$ represents a *sick state I* (due to, say, an infection). The transition matrix is therefore the 2×2 matrix

$$P = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}.$$

However, the true degrees of freedom of P is 2, since each row is constrained, as a probability distribution, to sum to 1 (such a matrix is known as a *stochastic matrix*). We can therefore write, without loss of generality,

$$P = \begin{bmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{bmatrix}. \quad (14.1)$$

for two numbers $\alpha, \beta \in [0, 1]$. This means that if a subject is healthy on day i , he/she is sick on day $i + 1$ with probability α , and if the subject is sick on day i , he/she is sick on healthy in day $i + 1$ with probability β . The state transition diagram for infection model is shown in Figure 14.1.

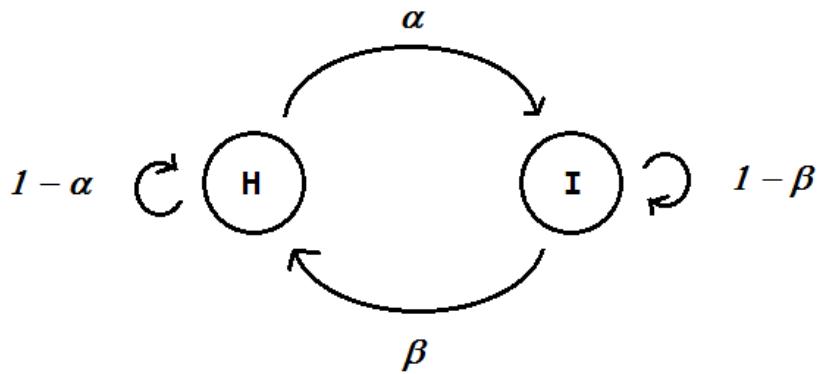


Figure 14.1: State transition diagram for infection model of Example 14.1.

Is this a reasonable model? First, we note that when the subjects enters state H , he/she remains there for a geometrically distributed waiting time, with mean α^{-1} . If we suppose that acquiring an

infection is a consequence of a chance exposure, which happens on any given day with probability α , then the memoryless ‘coin toss’ model for the waiting time until infection would be reasonable.

On the other hand, the infection lifetime also follows a geometric distribution, but with mean β^{-1} . Presumably, clinical experience would guide the choice of β , setting

$$\beta^{-1} = E[\text{infection lifetime}].$$

However, whether the geometric distribution adequately models an infection lifetime would be an important question to resolve. \square

The transition probability P_{ij} may be more formally referred to as the *one-step transition probability*, since it describes transition following a single time step. We may also describe the k -step transition probability

$$P(X_{n+k} = j \mid X_n = i) = P_{ij}^k, \quad (14.2)$$

noting that this probability does not depend on n . We will demonstrate this computation for $k = 2$. In Equation (14.2) set

$$E = \{X_{n+2} = j\}, \quad B = \{X_n = i\},$$

and we may form partition

$$A_k = \{X_n = k\} \text{ for all } k \in \mathcal{X}.$$

This gives

$$P_{ij}^2 = P(X_{n+2} = j \mid X_n = i) = \sum_{k \in \mathcal{X}} P(X_{n+2} = j \mid X_{n+1} = k, X_n = i) P(X_{n+1} = k \mid X_n = i). \quad (14.3)$$

Then consider each term in the summation. Recall by the Markovian property of Definition 14.1 that the distribution of X_{n+2} given history $X_{n+1}, X_n, \dots, X_1, X_0$ depends only on the most recent state X_{n+1} . We therefore have

$$P(X_{n+2} = j \mid X_{n+1} = k, X_n = i) = P(X_{n+2} = j \mid X_{n+1} = k) = P_{kj}. \quad (14.4)$$

The remaining quantity is simply the one step transition probability

$$P(X_{n+1} = k \mid X_n = i) = P_{ik}. \quad (14.5)$$

Substituting (14.4) and (14.5) into (14.3) yields

$$P_{ij}^2 = \sum_{k \in \mathcal{X}} P_{ik} P_{kj}. \quad (14.6)$$

This is a particularly important relationship, since we can recognize (14.6) as the result of matrix multiplication. We summarize this in the following definition.

Definition 14.2. The k -step transition probability from state i to j is the probability that a Markov chain in state i occupies state j after exactly k transitions. Formally,

$$P(X_{n+k} = j \mid X_n = i) = P_{ij}^k,$$

The k -step transition probability matrix P^k has value P_{ij}^k at (row,column) (i, j) , and can be calculated by iteratively multiplying P k times:

$$P^k = [P]^k$$

\square

14.1.1 Maze example

There has always been considerable interest in the skill with which rats navigate mazes. Markov chains provide an analytical tool with which the translate experimental observations into relevant conclusions.

For example, the term ‘memoryless’ implies an inability to learn. A Markov chain should be able to model navigational behavior which is more or less random, implying an inability to learn by trial and error. Therefore, we should be able to decide whether or not experimental data is explainable by truly memoryless behavior.

Consider the maze shown in Figure 14.2. A rat is introduced to the maze at the ‘Start’ label, and is provided some incentive to reach the ‘Finish’ label. There are 4 nodes of decision, labeled a, b, c, d , and a terminal node e . Node a is the first node encountered by the rat.

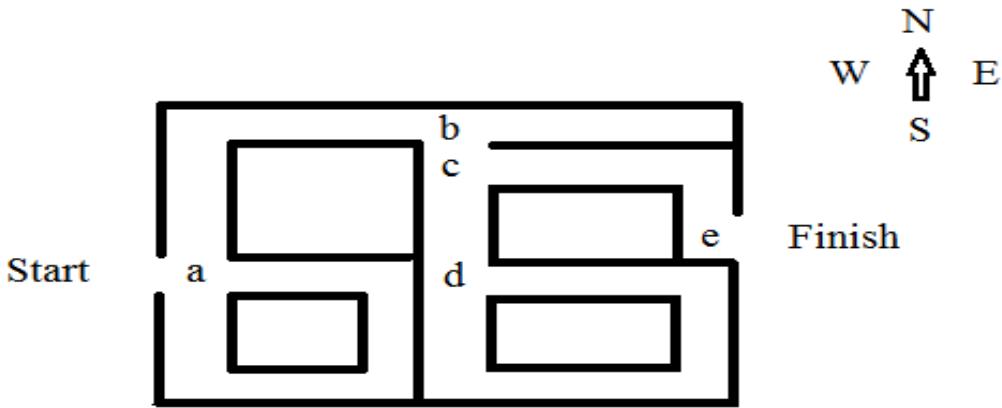


Figure 14.2: Maze diagram for Section 14.1.1.

How should we model the rat’s navigation? It seems reasonable to define the state space $\mathcal{X} = \{a, b, c, d, e\}$, modeling transitions between decision points. We might suppose that if the rat’s behavior is truly random, whenever it reaches a decision node, it simply chooses any available direction with equal probability, and otherwise it walks along its chosen path.

Assuming the entrance is closed after entry, from node a the rat can proceed N(orth), E or S. If it proceed N, it next reaches node b , otherwise it must return to a . This gives transition probabilities

$$P_{aa} = 2/3, \quad P_{ab} = 1/3, \quad P_{ak} = 0 \text{ for } k = c, d, e.$$

From node b the rat can proceed W, E or S. If it proceeds W it returns to a , if it proceeds E it reaches a dead end, and so must return to b , and if it proceeds S it reaches c . This gives transition probabilities

$$P_{ba} = 1/3, \quad P_{bb} = 1/3, \quad P_{bc} = 1/3, \quad P_{ak} = 0 \text{ for } k = d, e.$$

Continuing in this way, and ordering states a, b, c, d, e as $0, 1, 2, 3, 4$ we have transition matrix

$$P = \begin{bmatrix} & a & b & c & d & e \\ a & 2/3 & 1/3 & 0 & 0 & 0 \\ b & 1/3 & 1/3 & 1/3 & 0 & 0 \\ c & 0 & 1/3 & 0 & 1/3 & 1/3 \\ d & 0 & 0 & 1/3 & 1/3 & 1/3 \\ e & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (14.7)$$

Note that $P_{ee} = 1$, so that once the Markov chain reaches state e it remains there indefinitely, which is commonly interpreted as a termination of the process. Such a state is referred to as an *absorbing state*.

After some reflection, we might ask if a rat, when reaching a decision node, is likely to respond by reversing direction. We have assumed that the rat is as likely to do this as to choose any of the remaining directions.

Let's consider what happens if we try to rule out this behavior. If the rat reaches node a while traveling N, if we rule out direction reversals, the rat will either proceed N, reaching b , or proceed E, returning to a . The nonzero transition probabilities are now $P_{aa} = 1/2$, $P_{ab} = 1/2$, which differ from those given in (14.7). Unfortunately, we can see that these transition probabilities depend on the direction of travel. If the rat reaches node a by traveling S, barring direction reversals, the rat can only proceed S or E, forcing a return to a , suggesting that $P_{aa} = 1$. The problem is that the Markovian property is being violated. To say that the transition probabilities depend on the direction of travel is equivalent to saying that they depend not only on the current state (node), but also on the previous state (node). After all, if the rat reaches node a traveling S, it must previously have been at node b , whereas if it reaches node a traveling N, it must previously have been at node a . Definition 14.1 is therefore not satisfied.

Fortunately, it is possible to rule out direction reversals while maintaining the memoryless property. This can be done by expanding the state space (sometimes referred to as *Markovianizing* a process). In particular we include in the definition of the state both the node and the direction of approach. That is, the rat transitions to state $a - N$ by arriving at node a traveling N. At state $a - N$ the rat proceeds N or E with equal probability. By proceeding N the rat reaches state $b - E$, and by proceeding E the rat returns to state $a - N$. This gives nonzero transition probabilities

$$P_{a-N,a-N} = 1/2, \quad P_{a-N,b-E} = 1/2.$$

Node a must be expanded into states $a - E$, $a - N$, $a - S$ and $a - W$. Our protocol is to designate $X_0 = a - E$ as the initial state, although the definition of the Markov chain model remains the same for any initial state. Once the rat leaves state $a - E$ it does not return.

Similarly, state b is expanded into state $b - E$, $b - W$ and $b - N$. Since the rat cannot approach b traveling S, no state $b - S$ is needed. From state $b - E$ the rat can proceed E or S. If the rat proceeds E, it meets a dead end, and must return to node b traveling W, thus transitioning to state $b - W$. Otherwise, it arrives at node c traveling S, thus transitioning to node $c - S$. This gives nonzero transition probabilities

$$P_{b-E,b-W} = 1/2, \quad P_{b-E,c-S} = 1/2.$$

Continuing in this way, we may deduce the transition probabilities given in Table 1.

Table 1: Transition probabilities for Markov chain model of Section 14.1.1

$P_{ij} =$	a-E	a-N	a-S	a-W	b-N	b-E	b-W	c-N	c-S	d-N	d-S	d-W	e-S
a-E	0	1/3	0	1/3	0	1/3	0	0	0	0	0	0	0
a-N	0	1/2	0	0	0	1/2	0	0	0	0	0	0	0
a-S	0	1/2	0	1/2	0	0	0	0	0	0	0	0	0
a-W	0	0	0	1/2	0	1/2	0	0	0	0	0	0	0
b-N	0	0	1/2	0	0	0	1/2	0	0	0	0	0	0
b-E	0	0	0	0	0	0	1/2	0	1/2	0	0	0	0
b-W	0	0	1/2	0	0	0	0	0	1/2	0	0	0	0
c-N	0	0	0	0	1/2	0	0	0	0	0	0	0	1/2
c-S	0	0	0	0	0	0	0	0	0	0	1/2	0	1/2
d-N	0	0	0	0	0	0	0	1/2	0	1/2	0	0	0
d-S	0	0	0	0	0	0	0	0	0	1/2	0	1/2	0
d-W	0	0	0	0	0	0	0	1/2	0	0	0	1/2	0
e-S	0	0	0	0	0	0	0	0	0	0	0	0	1

A Markov chain, as a set of discrete probability distributions, is easily simulated (use, for example, the `rmultinom()` function, or consult Ross (2014)). An example of a single path is shown in Figure 14.3. In this example there are 26 transitions. A total of 27 states are visited, including initial and final states $a - E$ and $e - S$. We note some 'cycling' behavior around node a at transitions 1 and 13, and around node d at transitions 8 and 21. This is characteristic of memoryless behavior.

14.1.2 Distributional properties of Markov chains

All distributional properties of a Markov chain follow from transition matrix P . In principle, these properties may be studied by simulating *sample paths*, using some large number of replications N . For example, in the maze example suppose T is the number of transitions required to reach terminal state $e - S$ from initial state $a - E$. This is a random survival time, and examining Figure 14.3 we can see that the minimum possible value is $T = 3$, attained exclusively by path

$$a - E \rightarrow b - E \rightarrow c - S \rightarrow e - S. \quad (14.8)$$

Note that T is one less than the number of states in a path. Suppose we observe a rat solve the maze with $T = 3$. If the rat has already had some experience with the maze, we may suspect that it has 'learned' the shortest route. On the other hand, $T = 3$ is an outcome which may occur under the memoryless model defined in Table 1. We can form some judgement by calculating the probability $P(T = 3)$. If it is very small, we conclude that the rat is unlikely to have achieved the optimal path by chance.

Simulations may be used to estimate $P(T = 3)$. If out of N simulated paths, X equal the optimal path, then

$$P(T = 3) \approx X/N.$$

However, the Markov chain model proves to be a very powerful analytic tool, and many interesting properties may be calculated exactly. When this is possible, it is to be preferred to simulation.

For example, the probability of a path is easily calculated.

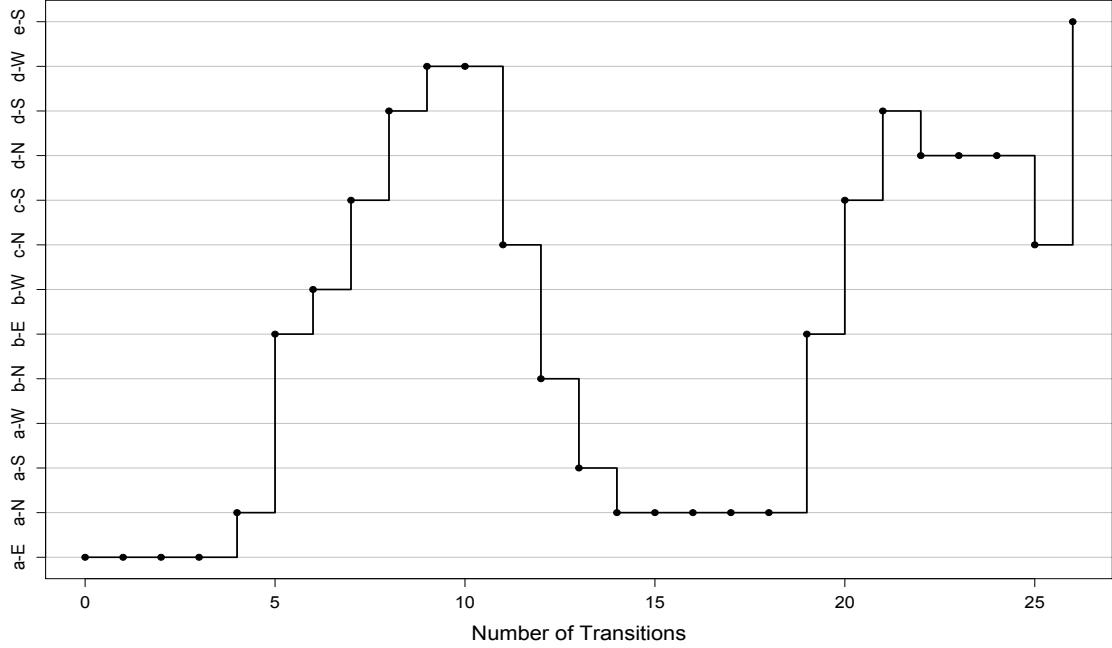


Figure 14.3: Simulated Markov chain navigation for maze example of Section 14.1.1, based on transition probabilities of Table 1.

Theorem 14.1. Let

$$i_0 \rightarrow i_1 \dots \rightarrow i_{m-1} \rightarrow i_m$$

be any path of m transitions. The probability of the path, given initial state $X_0 = i_0$ is

$$P(X_m = i_m, X_{m-1} = i_{m-1}, \dots, X_1 = i_1 | X_0 = i_0) = P_{i_0, i_1} \times P_{i_1, i_2} \times \dots \times P_{i_{m-2}, i_{m-1}} \times P_{i_{m-1}, i_m}. \quad (14.9)$$

Proof. We may write

$$\begin{aligned} P(X_m = i_m, X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) &= P(X_m = i_m | X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) \\ &\quad \times P(X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0). \end{aligned} \quad (14.10)$$

By the Markov property

$$P(X_m = i_m | X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) = P(X_m = i_m | X_{m-1} = i_{m-1}) = P_{i_{m-1}, i_m}. \quad (14.11)$$

Substituting (14.11) into (14.10) yields

$$\begin{aligned} P(X_m = i_m, X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) &= \\ P_{i_{m-1}, i_m} P(X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0). \end{aligned} \quad (14.12)$$

We may apply essentially the same argument to the quantity $P(X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0)$ in (14.12), giving

$$\begin{aligned} P(X_m = i_m, X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) &= \\ P_{i_{m-2}, i_{m-1}} P_{i_{m-1}, i_m} P(X_{m-2} = i_{m-2}, \dots, X_1 = i_1, X_0 = i_0). \end{aligned}$$

Repeating enough times, we have

$$\begin{aligned} P(X_m = i_m, X_{m-1} = i_{m-1}, \dots, X_1 = i_1, X_0 = i_0) &= \\ P_{i_0, i_1} \times P_{i_1, i_2} \times \dots \times P_{i_{m-2}, i_{m-1}} \times P_{i_{m-1}, i_m} P(X_1 = i_1, X_0 = i_0). \end{aligned} \quad (14.13)$$

We condition both sides of (14.13) on $\{X_0 = i_0\}$ by dividing by $P(X_0 = i_0)$, noting that

$$P(X_1 = i_1, X_0 = i_0) / P(X_0 = i_0) = P(X_1 = i_1 | X_0 = i_0) = P_{i_0, i_1},$$

which yields (14.9). \square

Note that the probability of a path starting at any time index may be calculated using the method of Theorem 14.1.

Example 14.2. For the Markov chain of Table 1, if T is the number of transitions required to reach terminal state $e - S$, then, as discussed earlier in this section, the minimum value $T = 3$ is reached exclusively by the path of (14.8). By Theorem 14.1, we have

$$\begin{aligned} P(T = 3) &= P(X_3 = e - S, X_2 = c - S, X_1 = b - E | X_0 = a - E) \\ &= P_{a-E, b-E} P_{b-E, c-S} P_{c-S, e-S} \\ &= 1/3 \times 1/2 \times 1/2 \\ &= 1/12. \end{aligned}$$

Thus, under the memoryless model, the optimal path is attained with probability 1/12, and would therefore be consistent with the memoryless model. \square

It turns out that the exact distribution of T is readily obtained. Recall the k -step transition probability

$$P_{a-E, e-S}^k = P(X_k = e - S | X_0 = a - E).$$

Clearly, if $X_k = e - S$, we must have $T \leq k$ (the path may have entered state $e - S$ before the k th transition). However, recall that $e - S$ is an absorbing state, so that if $X_j = e - S$ for some j , we must also have if $X_k = e - S$ for all $k \geq j$. This means that if $T \leq k$ we must also have $X_k = e - S$, so that

$$\{T \leq k\} = \{X_k = e - S\}$$

and so we have CDF

$$F_T(k) = P(T \leq k) = P(X_k = e - S) = P_{a-E, e-S}^k.$$

Note that we are implicitly conditioning the distribution of T on the event $\{X_0 = a - E\}$. Then, the k -step probabilities $P_{a-E, e-S}^k$ can be obtained by matrix multiplication of P . Interpreting Table 1 as a 13×13 matrix P , we have

$$F_T(k) = P^k[1, 13],$$

from which we get the PMF

$$p_T(k) = F_T(k) - F_T(k-1) = P^k[1, 13] - P^{k-1}[1, 13],$$

where the matrix P^0 is taken to be the identity matrix. Figure 14.4 shows the values of $P(T = k)$. We may compute the expected value

$$E[T] \approx 11.33.$$

On average, under the memoryless model, a rat requires 11.33 transitions to solve the maze. However, note that the tail of the distribution extends well above this value. For example, we have the probability $P(T \geq 25) \approx 0.083$ and $P(T \geq 35) = 0.027$, so observing a solution time well above the mean will not be a rare occurrence.

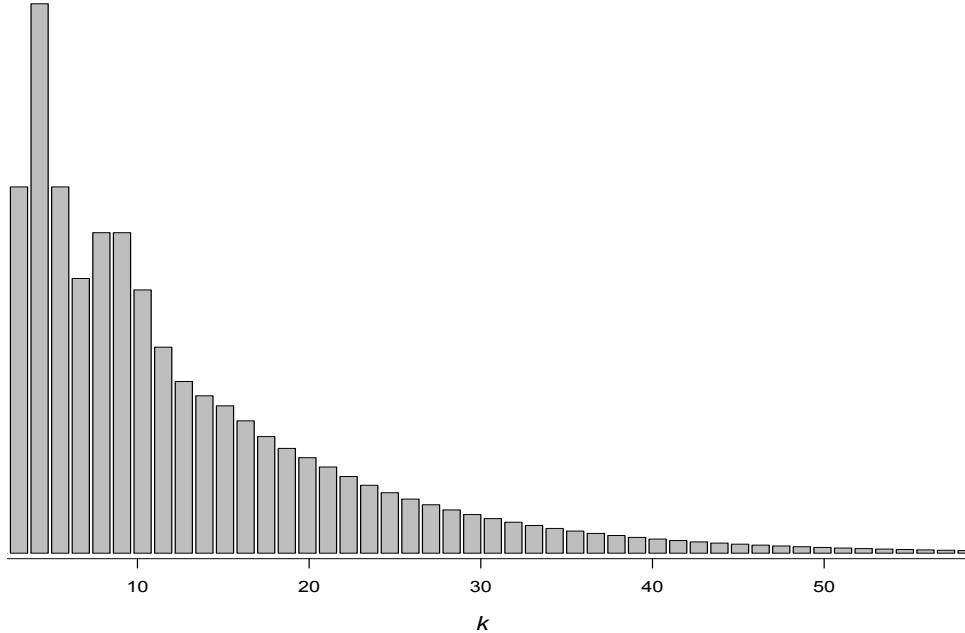


Figure 14.4: PMF $p_T(k) = P(T = k)$ of T , the number of transitions required to reach terminal state $e - S$, for maze example of Section 14.1.1, based on transition probabilities of Table 1.

14.1.3 Balance equations and steady states

There is often interest in the long run behavior of a Markov chain. In Example 14.1 the process fluctuates between *Healthy* and *Infected* states indefinitely, and we may be interested in knowing the long run proportion of time spent in each state.

Suppose we have counting process

$N_j(k) =$ The number of transitions into state j after the k th transition.

A long run frequency (formally, a *steady state frequency*) would then be defined by

$$\pi_j = \lim_{k \rightarrow \infty} \frac{N_j(k)}{k}. \quad (14.14)$$

A number of mathematical questions lurk here. Does the limit always exist? If so, under what conditions is π_j zero or positive? Last, and far from least, does this quantity depend on the initial state? To formally resolve these questions requires some amount of mathematical theory, even when dealing with relatively simple models. While this would be beyond the scope of this course, we can give some insight into the issues.

Deducing π_j for general models requires acknowledgement of an apparently obvious fact, that is,

$$\text{number of times process enters a state} = \text{number of times process exits a state}, \quad (14.15)$$

(within one). The value of this statement becomes more apparent if we think in terms of *rates*. Clearly, π_j can be interpreted as the *occupancy rate* of state j , but also as its *entrance rate* and the *exit rate*. Next, we may consider the rate at which the Markov chain transitions from states i to j . There are two components to this. First, to transition from i to j , the Markov chain must first enter i . This occurs at rate π_i . Second, given that the Markov chain is in i , it transitions from i to j with probability P_{ij} . The rate of transition from i to j is therefore $\pi_i P_{ij}$.

We are now in a position to use (14.15). We recognize the exit rate for state j as π_j . The entrance rate, on the other hand, can be given as the sum of all other transition rates *into* state j , that is, $\pi_i P_{ij}$ for $i \in \mathcal{X}$ (this includes $i = j$, when $P_{jj} > 0$). Equation (14.15) then yields the *balance equation*

$$\pi_j = \sum_{i \in \mathcal{X}} \pi_i P_{ij}. \quad (14.16)$$

Example 14.3. Consider the two-state Markov chain of Example 14.1. We write a balance equation for each state, yielding

$$\begin{aligned} \pi_0 &= \pi_0 P_{00} + \pi_1 P_{10} \\ \pi_1 &= \pi_0 P_{01} + \pi_1 P_{11}, \end{aligned}$$

which, after substituting transition probabilities (14.1) gives

$$\begin{aligned} \pi_0 &= \pi_0(1 - \alpha) + \pi_1\beta \\ \pi_1 &= \pi_0\alpha + \pi_1(1 - \beta). \end{aligned}$$

Rewriting the first equation yields

$$\frac{\pi_0}{\pi_1} = \frac{\beta}{\alpha}. \quad (14.17)$$

Since we must have $\sum_i \pi_i = 1$, only one balance equation is actually needed to solve for (π_0, π_1) , and we obtain

$$\pi_0 = \frac{\beta}{\alpha + \beta}, \quad \pi_1 = \frac{\alpha}{\alpha + \beta}.$$

That the frequencies π_0, π_1 should possess ratio β/α is to be expected. The time spent in each state prior to transition is geometrically distributed with means $1/\alpha$ and $1/\beta$ respectively. The ratio π_0/π_1 should then be the ratio of the means, which is confirmed by (14.17). \square

14.2 The Hastings-Metropolis algorithm

In Bayesian inference, we have a posterior density of the form

$$\pi(\theta | x) = \frac{f(x | \theta)\pi(\theta)}{\int_{\Theta} f(x | \theta)\pi(\theta)d\theta},$$

or, similarly when model space Θ is discrete,

$$\pi(\theta | x) = \frac{f(x | \theta)\pi(\theta)}{\sum_{\theta \in \Theta} f(x | \theta)\pi(\theta)}.$$

It is often not possible to evaluate the normalization constant (which is $f(x)$ in this example). In this case, we may use a Markov chain Monte Carlo method to simulate a sample from a density or probability mass function, say $p(y)$, on state space \mathcal{S}_y , which is known only up to a normalizing constant. This means we can write

$$p(y) = Kg(y),$$

where $g(y)$ is known, K does not depend on y but K is otherwise unknown. We can, for example, write a Bayesian posterior density

$$\pi(\theta | x) = Kg(\theta)$$

where, for fixed x , $g(\theta) = f(x | \theta)\pi(\theta)$ and $K = 1/f(x)$.

The *Hastings-Metropolis algorithm* (Hastings, 1970) is an MCMC method which simulates a Markov chain on a state space \mathcal{S}_y which has steady state distribution $p(y)$. It depends on $p(y)$ only through ratios of the form $p(y')/p(y)$. Therefore, if $p(y) = Kg(y)$, where K does not depend on y , the algorithm only needs ratios

$$\frac{p(y')}{p(y)} = \frac{Kg(y')}{Kg(y)} = \frac{g(y')}{g(y)},$$

that is, all we need is a function $g(y)$ which is *proportional* to $p(y)$.

The algorithm takes the following steps:

- First define $Q(y_2 | y_1)$, a *proposal* Markov chain on \mathcal{S}_y . This MC should be *irreducible*, which can in practice be difficult to prove (a MC is *irreducible* if any state may be reached from any other state).
- Construct a Markov chain y_1, y_2, \dots according to the following algorithm:
 - (1) Given current state y_n , select *proposal state* y' according to probability distribution $Q(y' | y_n)$.
 - (2) With probability α set $y_{n+1} = y'$, and with probability $1 - \alpha$ set $y_{n+1} = y_n$, where

$$\begin{aligned} r &= \frac{p(y')Q(y_n | y')}{p(y_n)Q(y' | y_n)} \\ \alpha &= \min(r, 1). \end{aligned}$$

- The Markov chain is irreducible, with steady-state distribution $p(y)$.

There are alternative formulations. See Hastings (1970) “Monte Carlo sampling methods using Markov chains and their applications”, *Biometrika*.

14.3 Simulated annealing

Suppose the objective is to maximize a function $f(y)$ on a state space \mathcal{S}_y (or minimize $-f(y)$). This can be done using *simulated annealing*. This is an MCMC algorithm similar to the Hastings-Metropolis algorithm, in that it simulates a stochastic process y_1, y_2, \dots using a similar proposal acceptance mechanism. The difference is that, under known conditions, the process converges to y^* , where $\max_y f(y) = f(y^*)$. In particular, if \mathcal{S}_y is discrete, then

$$\lim_{n \rightarrow \infty} P(y_n = y^*) = 1,$$

assuming the maximum is unique. Under quite general conditions, we have for any positive constant $\delta > 0$

$$\lim_{n \rightarrow \infty} P(f(y_n) > f(y^*) - \delta) = 1.$$

Many optimization algorithms guarantee convergence to a local maximum, but simulated annealing is one of the few optimization algorithms which is able to guarantee convergence to the global maximum.

The algorithm takes the following steps:

- Construct a proposal Markov chain $Q(y' | y_n)$ as in the Hastings-Metropolis algorithm.
- Define a decreasing *temperature* sequence t_1, t_2, \dots (the *cooling schedule*).
- Construct a process y_1, y_2, \dots according to the following algorithm:
 - (1) Given current state y_n , select *proposal state* y' according to probability distribution $Q(y' | y_n)$.
 - (2) With probability α set $y_{n+1} = y'$, and with probability $1 - \alpha$ set $y_{n+1} = y_n$, where

$$\alpha = \begin{cases} 1 & \text{if } f(y') \geq f(y_n) \\ \exp\left(\frac{f(y') - f(y_n)}{t_n}\right) & \text{if } f(y') < f(y_n) \end{cases}$$

- The process y_1, y_2, \dots converges to the maximum if $f(y)$ in the sense given above.

Note that the process y_1, y_2, \dots is Markovian, or memoryless, but is not a time-homogenous Markov chain, since the transition probabilities depend on time index n through the cooling schedule.

We next consider the question of the the cooling schedule (a good review can be found in Nourani and Andresen (1998)). For a cooling schedule of the form

$$t_n = \frac{c}{\log(n + d)}, \quad (14.18)$$

the simulated annealing has been proven to converge to the optimal solution in the sense defined above, provided c is large enough. Unfortunately, the minimum value of c for which convergence can be guaranteed depends on the problem, and in general, the extremely slow convergence rate makes this cooling schedule impractical. Furthermore, because the algorithm is stochastic, it will not generally be possible to define a stopping rule, that is, a rule which can be used to decide when the optimal solution has been reached (assuming the largest value $f(y^*)$ is not known).

Despite these qualifications, simulated annealing is useful for optimization problems not possessing regularity conditions for which more specialized algorithms would be available. It is also widely applicable, and relatively easy to implement. In practice the cooling schedule (14.18) is not used. Commonly used choices include the exponential schedule

$$t_n = t_0 \rho^n$$

and the linear schedule

$$t_n = t_0 - \beta n,$$

although neither results in a provably convergent algorithm.

14.4 Postscript

The texts Ross (2014), then Ross (1996), provide an excellent introduction to the theory of Markov chains. The maze example of Section 14.1.1 demonstrates how the Markov chain model can be used to simulate and analyze seemingly complex systems. The **Simulation Projects** section of Chapter 27, as well as Problem 28.12, offer a few more examples of Markov chain modeling. Problem 27.1 is similar to the maze example.

MCMC sampling is quite old. Originally published in 1953 (Metropolis *et al.*, 1953), the ideas have probably been in practice several years earlier. It is commonly used today in a form closer to that proposed in Hastings (1970). An accessible introduction to this topic can be found in Shonkwiler and Mendivil (2009).

At one level, once the MCMC sampling method is understood, its application to Bayesian inference is clear enough. It is simply used to sample from a posterior density, the advantage being that the normalization constant is not needed. Chapter 27 contains a number of detailed applications. Problem 27.3 demonstrates the accuracy of the method by sampling from a known (binomial) distribution. Demonstration software file COMPUTATIONAL-BAYESIAN.R contains a detailed example based on the “archaeology” example in the Wikipedia entry for Bayesian inference (https://en.wikipedia.org/wiki/Bayesian_inference).

It might be going too far to say that computational Bayesian applications are easy to do, but difficult to do well. But we have only provided the briefest of introductions, and the analyst who makes consistent use of this method should read more widely on this topic. Albert (2009) and Marin and Robert (2014) provide good starting points, in an R context.

We have not covered another widely used MCMC sampler, the *Gibbs sampler* (Geman and Geman, 1984), which tends to be used to sample from densities on very high dimensional objects possessing some spatial structure (image processing, graphical models, etc). Bayesian inference based on the Gibbs sampler is supported by a widely used class of software applications, an initiative of the *BUGS Project* (Bayesian inference Using Gibbs Sampling) at www.mrc-bsu.cam.ac.uk/software/bugs/. See also Lunn *et al.* (2000).

Simulated annealing is a later adaption of the MCMC method, generally credited to Kirkpatrick *et al.* (1983). While MCMC sampling is used to approximate a density (by sampling from it), simulated annealing is used to optimize a function, which need not be a density. The text Shonkwiler and Mendivil (2009) recommended above also contains an introduction to simulated annealing. For a more advanced treatment, see, for example, van Laarhoven and Aarts (1987).

This technique is relatively slow, and is often thought of as a “method of last resort”, suitable for problems lacking regularity conditions required by more efficient optimization techniques (especially combinatorial, or discrete, optimization problems). However, Problem 27.4 offers a demonstration of simulated annealing which highlights one of its truly remarkable properties, in particular, its ability to distinguish between local and global minima/maxima. This ability is actually quite rare among more commonly used optimization algorithms, so simulated annealing is a method worth considering in general.

As an example demonstration software `SIMULATED-ANNEALING.R` contains an application to the well-known *travelling salesman problem*.

We finally note that the R function `optim()`, available for general multivariate optimization, implements simulated annealing as an option. The command `help(optim)` provides a good introduction to optimization in R. For univariate optimization the function `optimize()` is preferable.

Part II

Supervised and Unsupervised Learning

Chapter 15

Machine Learning and Statistical Learning - General Concepts

Machine learning describes computer applications in pattern recognition, computational learning or artificial intelligence. Examples include the detection of purposeful motion in a dynamic pixel field, or the development of software capable of identifying handwritten letters. Statistical learning is the application of statistical methodology to these problems, focusing on the exploitation of data sets.

Most statistical learning problems share a common structure. We are given, at the very least, a set of n observations $\dot{x}_1, \dots, \dot{x}_n$. For our purposes, we can take \dot{x}_i to be a vector of length p :

$$\dot{x}_i = (x_{i1}, \dots, x_{ip}).$$

If we combine the observations into rows, we get an $n \times p$ matrix

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ip} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_n \end{bmatrix}.$$

It is important to note that \mathbf{X} may also be decomposed by column. We may also define vector $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})$, so that

$$\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_p].$$

Formally, we are interpreting \dot{x}_i as a $1 \times p$ row vector, and \mathbf{x}_j as a $n \times 1$ column vector, that is

$$\dot{x}_i = [x_{i1} \cdots x_{ip}] \text{ and } \mathbf{x}_j = \begin{bmatrix} x_{1j} \\ \vdots \\ x_{nj} \end{bmatrix}.$$

The symbol \mathbf{X} represents a data set, with row/column i, j element $[\mathbf{X}]_{ij} = x_{ij}$. We refer to \mathbf{x}_j as a *feature*, *predictor* or *independent variable*. The latter two are more common in statistical literature, the first more common in computer science. Intuitively, a feature refers to a type of

information, of which there are p in this data set. Each element of feature \mathbf{x}_j is a member of an outcome set $x_{ij} \in \mathcal{E}_j$. These p data types may be of any form, quantitative (integer, real or complex), qualitative or categorical (nominal or ordinal) or logical (true or false). However, it is necessary that all outcomes in a feature be of the same type, and additionally of the same unit (inches, degrees celsius, etc) when they are quantitative. The feature space is then the product space $\dot{\mathbf{x}}_i \in \mathcal{E}_{\mathbf{x}} = \mathcal{E}_1 \times \cdots \times \mathcal{E}_p$.

15.1 Some Notational Conventions

The ‘tilde’ notation \tilde{x} or $\tilde{\beta}$ will be used to denote vectors in general, say, $\tilde{x} = (x_1, \dots, x_n)$ or $\tilde{\alpha} = (\alpha_1, \dots, \alpha_p)$. The symbols \dot{x}_i and \mathbf{x}_j denote specifically the row and column vectors of a feature matrix \mathbf{X} .

In the context of linear algebra, unless otherwise specified an n -dimensional vector is interpreted as an $n \times 1$ column vector. Therefore, given, for example, $\tilde{x} = (x_1, \dots, x_n)$ and $\tilde{y} = (y_1, \dots, y_n)$ we may write

$$\tilde{x}^T \tilde{y} = \sum_{i=1}^n x_i y_i.$$

As in the previous section, in some cases bold font will denote vectors, for example $\boldsymbol{\mu} = (\mu_1, \dots, \mu_p)$.

If \mathbf{A} is a matrix, then the i, j element may be written A_{ij} or $[A]_{ij}$, based on whichever seems clearer in the context.

15.2 Structure of Data

The observations are usually sampling units. That is, observation \dot{x}_i is a set of p features outcomes associated with a single sampling unit (person, city, computer image, etc). It is possible that a observation does not contain outcomes for all features. In this case, a special symbol, say $x_{ij} = \text{NA}$, is used when the j th feature of sampling unit i is not observed. This becomes a *missing value*, and considerable research has gone into the development of principled methods for dealing with this problem.

15.2.1 Features

As a practical matter, since \mathbf{X} will be subject to algebraic operations, we are forced to regard any element x_{ij} as a number. Usually, this is easily done. A logical feature can translate outcomes *true* or *false* to integers 1 or 0. An ordinal feature \mathbf{x}_j can be converted to a rating scale $1, \dots, N$, when there are N ordered outcomes in \mathcal{E}_j (eg, ‘nonsmoker’, ‘light smoker’, ‘heavy smoker’). We may refer to an indicator variable \mathbf{x}_j as one whose outcome set is $\mathcal{E}_j = \{0, 1\}$. This device is often used to indicate the presence or absence of a particular characteristic in the sampling unit (eg ‘possesses college degree’ = 1). Formally, this type of feature is nominal, but may be used within algebraic operations in a logical manner. While a single indicator variable can represent a nominal feature with only two outcomes, nominal features with $m > 2$ outcomes can be expanded into m (or $m - 1$)

indicator variables, by assigning one indicator variable to each outcome:

$$\begin{bmatrix} \text{Red} \\ \text{Red} \\ \text{Blue} \\ \vdots \\ \text{Green} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ \vdots \\ 0 & 0 \end{bmatrix}. \quad (15.1)$$

Here, a single feature consisting of outcomes $\mathcal{E}_j = \{\text{Red}, \text{Blue}, \text{Green}\}$ shown in (15.1) has been converted to 3, then 2 indicator variables. It is important to note that the 2 column representation is obtained simply by deleting the third column. No information has been lost, as long as we know that the outcome is Green, if it is not Red or Blue, as the definition of the outcome set \mathcal{E}_j tells us. In general, it is preferable from a mathematical point of view to represent an m outcome nominal feature with $m - 1$ indicator variables.

15.2.2 Response

Some datasets include a *response variable* $\mathbf{y} = (y_1, \dots, y_n)$. As for features, the elements are of a common type, with outcomes in outcome set $y_i \in \mathcal{E}_{\mathbf{y}}$. In fact, in all ways \mathbf{y} is a feature, except that it plays a special role in the machine learning application. Alternatively, we may have a data set \mathbf{X} in which one feature $\mathbf{y} = \mathbf{x}_j$ is selected as the response variable, the selection depending on the objective of the application. In this case we take the data set to be (\mathbf{y}, \mathbf{X}) .

15.3 Feature Distances

It is sometimes important to define a distance d between two vectors observations $\mathbf{u} = (u_1, \dots, u_m)$, $\mathbf{v} = (v_1, \dots, v_m)$. The most natural is Euclidean distance

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + \dots + (u_m - v_m)^2},$$

but there are often good reasons why alternative distance functions should be considered. A number of mathematical objects are important when considering this problem, particularly the *metric* and the *norm*.

15.3.1 Metrics

The *metric* can be thought of as a generalization of the notion of Euclidean distance, allowing more flexible notions of distance, while retaining the most important properties.

Definition 15.1. Suppose we have real-valued mapping $d : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ operating on two observations \mathbf{u}, \mathbf{v} . Then d is a *metric* if

- (i) $d(\mathbf{u}, \mathbf{v}) \geq 0$ (non-negativity);
- (ii) $d(\mathbf{u}, \mathbf{v}) = 0$ if and only if $\mathbf{u} = \mathbf{v}$ (identifiability);

- (iii) $d(\mathbf{u}, \mathbf{v}) = d(\mathbf{v}, \mathbf{u})$ (symmetry);
- (iv) $d(\mathbf{u}, \mathbf{v}) \leq d(\mathbf{u}, \mathbf{w}) + d(\mathbf{w}, \mathbf{v})$ for any $\mathbf{w} \in \mathbb{R}^m$ (triangle inequality).

The term *distance function* may be used for mappings satisfying some but not all of these axioms, which otherwise satisfy the intuitive notion of a distance. For example, if (ii) is replaced by (ii)' $d(\mathbf{u}, \mathbf{v}) = 0$ if $\mathbf{u} = \mathbf{v}$; then d is a *pseudometric*. If (iii) does not hold then d is a *quasimetric*. A non-symmetric mapping can always be *symmetrized* by taking

$$d^*(\mathbf{u}, \mathbf{v}) = d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{u}).$$

Note that a metric multiplied by a positive scalar remains a metric. A real-valued mapping $s : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$ operating on two observations \mathbf{u}, \mathbf{v} is a *similarity measure* if it is negatively associated with a distance. \square

The similarity measure of Definition 15.1 is not as precisely defined as a metric. Some conventions require that it be non-negative or symmetric. On the other hand, a similarity measure can be simply constructed as $s(\mathbf{u}, \mathbf{v}) = -d(\mathbf{u}, \mathbf{v})$, where d is a distance function, and later standardized to be non-negative if needed.

15.3.2 L^p norms

The magnitude of a vector $\mathbf{u} = (u_1, \dots, u_m)$ in Euclidean space is

$$|\mathbf{u}| = \sqrt{u_1^2 + \dots + u_m^2}.$$

Similarly, the Euclidean distance between vectors \mathbf{u} and \mathbf{v} is $|\mathbf{u} - \mathbf{v}|$. In much the same way that the metric generalizes Euclidean distance, the *norm* generalizes Euclidean magnitude.

Definition 15.2. Suppose we have real-valued mapping $\|\cdot\| : \mathbb{R}^m \mapsto \mathbb{R}$ is a *norm* if for any vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ and scalar $a \in \mathbb{R}$

- (i) $\|a\mathbf{u}\| = |a|\|\mathbf{u}\| \geq 0$ (absolute scalability);
- (ii) $\|\mathbf{u}\| = 0$ implies $\mathbf{u} = \vec{0}$ (identifiability);
- (iii) $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (triangle inequality).

Note that $\vec{0} = (0, \dots, 0)$ is the *zero vector*. It is not necessary to state that $\|\vec{0}\| = 0$ since this is implied by axiom (i). In addition, that $\|\mathbf{u}\| \geq 0$ follows from axiom (i) ($\|\mathbf{u}\| = \|\mathbf{-u}\|$) and axiom (iii) (set $\mathbf{v} = -\mathbf{u}$).

If axiom (ii) does not hold, then $\|\cdot\|$ is a *semimetric*, which shares all properties of a metric, except that $\|\mathbf{u}\| = 0$ does not imply that $\mathbf{u} = \vec{0}$. \square

The L^p norms are an important class of norms.

Definition 15.3. The L^p norm for $\mathbf{u} \in \mathbb{R}^m$ is defined as

$$\|\mathbf{u}\|_p = \left[\sum_{i=1}^m u_i^p \right]^{1/p},$$

for $p \in (0, \infty)$. In addition, the *supremum norm* (setting $p = \infty$) is defined as

$$\|\mathbf{u}\|_{\infty} = \max_{i=1,\dots,m} |u_i|.$$

It can be verified that L^p norms are true norms according to Definition 15.2.

Suppose $w_i > 0$, $i = 1, \dots, m$ are a set of *weights*. The *weighted L^p norm*, denoted L_w^p , is defined as

$$\|\mathbf{u}\|_{p,w} = \left[\sum_{i=1}^m (w_i u_i)^p \right]^{1/p},$$

for $p \in (0, \infty)$. In addition, the *weighted supremum norm* is defined as

$$\|\mathbf{u}\|_{\infty,w} = \max_{i=1,\dots,m} |w_i u_i|.$$

Weighted L_w^p norms are also true norms. \square

15.3.3 Distance functions

One property of norms proves to be useful in statistical learning. Given any norm $\|\cdot\|$ on \mathbb{R}^m (Definition 15.2), the distance function

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|$$

is a true metric (Definition 15.1). Many commonly used distance functions are based on norms, including *Euclidean distance*

$$d_{euc}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2,$$

Manhattan distance

$$d_{man}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_1,$$

and *supremum or maximum distance*

$$d_{sup}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_{\infty}.$$

More variety of distances follow by using weighted L_w^p norms in the same way. An $m \times m$ matrix Σ is *positive definite* if $\mathbf{u}^T \Sigma \mathbf{u} > 0$ for all nonzero column vectors \mathbf{u} . In this case Σ is invertible, and Σ^{-1} is also positive definite. Then *Mahalanobis distance* is defined by

$$d_{mah}(\mathbf{u}, \mathbf{v}) = [(\mathbf{u} - \mathbf{v})^T \Sigma^{-1} (\mathbf{u} - \mathbf{v})]^{1/2},$$

and is a true metric. When this metric is used, Σ is usually a covariance matrix. Note that some conventions refer to Mahalanobis distance as d_{mah}^2 . However, d_{mah}^2 is not a metric (it does not satisfy the axioms of Definition 15.2). This distinction should be kept in mind.

If \mathbf{u} and \mathbf{v} are binary vectors, assuming only values in $\{0, 1\}$, then *Hamming distance* is often used, which is equivalent to

$$d_{ham}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_1,$$

that is, Hamming distance is the L^1 metric applied to binary vectors. It is equivalent to the number of element pairs which differ between \mathbf{u} and \mathbf{v} .

It is also possible to define similarity or distance using a correlation coefficient. A correlation already serves as a similarity measure, and a distance function can be defined as

$$d_{cor}(\mathbf{u}, \mathbf{v}) = 1 - r(\mathbf{u}, \mathbf{v})$$

where $r(\mathbf{u}, \mathbf{v})$ can be any correlation coefficient, including the Pearson and Spearman correlation coefficients, and Kendall's τ .

15.4 Supervised and Unsupervised Learning

Given a set of features \mathbf{X} , and possibly a response \mathbf{y} , we may define the two main classes of problems in statistical learning. These are distinguished by the presence or absence of a response variable.

In *unsupervised learning*, there is no response variable \mathbf{y} . The goal is to uncover relationships or patterns within the observations or features. Perhaps the most common application is *cluster analysis*, in which observations, or possibly features, are divided into *clusters* of similar observations (or features).

In *supervised learning* the objective is to relate the features \mathbf{X} to the response variable \mathbf{y} . The formal object is to develop a mapping $\hat{f} : \mathcal{E}_x \mapsto \mathcal{E}_y$, with the property that $\hat{f}(\dot{x}_i) \approx y_i$, in some sense, whether \mathbf{y} is qualitative or quantitative.

The distinction can be seen in Figure 15.1. The MPG and Horsepower ratings of a sample of cars manufactured in 1973 and 1981 are shown as a scatterplot. The year is indicated by distinct symbols. If we ignore the year, we may note that the points seem to separate into two distinct clusters, and we can imagine a boundary A between them (Figure 15.1). Intuitively, we might conjecture that the two clusters are distinct car styles, perhaps large sedans on one side of the boundary, and smaller economy cars on the other. This is an example of unsupervised learning, since the boundary was constructed without any information beyond the two features MPG and Horsepower.

Next, suppose we wish to develop a rule with which to predict the manufacture year, which then becomes the response variable. We will study methods which, when applied to this data set, might yield a boundary similar to B (Figure 15.1). This yields a mapping \hat{f} which assigns category $\hat{f}(\dot{x}) = '1981'$ if \dot{x} is in the interior of B , and $\hat{f}(\dot{x}) = '1973'$ otherwise. We can assess the accuracy of \hat{f} by systematically comparing $\hat{f}(\dot{x}_i)$ to y_i . If we do, we find that there are two observations with response '1981' outside boundary B and two observations with response '1973' inside, for a total of 4 errors, but all other predictions are correct. Note that to construct this boundary we needed to know the response \mathbf{y} . This is a typical example of supervised learning.

15.5 Loss and Risk

The purpose of \hat{f} is to predict a response y given an observation \dot{x} , developed from data (\mathbf{y}, \mathbf{X}) using the types of methodologies we will discuss below.

Note that \mathbf{y} may be qualitative or quantitative. Up to a point, the principles of supervised learning do not depend on this distinction, but eventually, the difference becomes unavoidable. In either, case, it is assumed that a response y obeys a distribution $f(y | \dot{x})$ conditional on its associated feature. When a new feature \dot{x} is presented, we form a prediction $\hat{y} = \hat{f}(\dot{x})$ for y . Recognizing that a prediction $\hat{f}(\dot{x})$ is subject to some error, we define a *loss function* $L(y, \hat{y})$, which

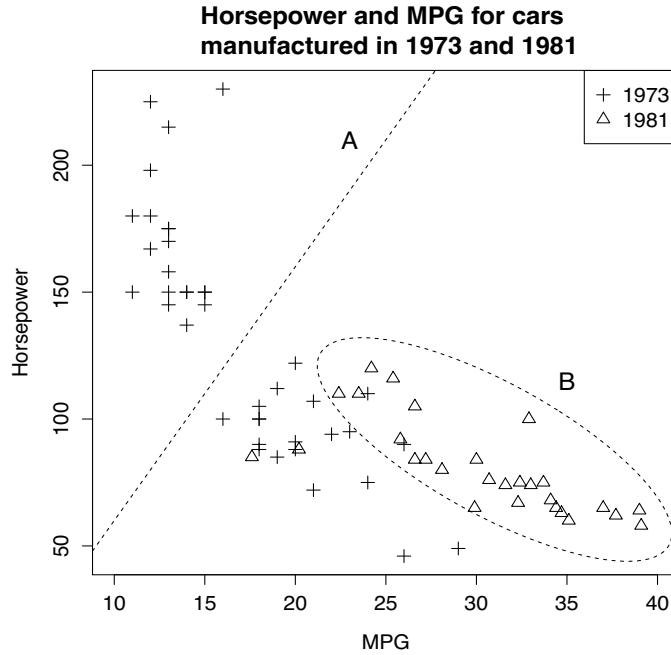


Figure 15.1: Sample of cars manufactured in 1973 and 1981 (Section 15.4).

is our assessment of the cost of prediction error. This can vary with the goal of the predictor, and a variety of loss functions might be considered. This is because we might view one type of error as more consequential than another (a false negative would be of greater consequence for a fire alarm than a false positive). Given a probability model, we define *risk*:

$$R(\dot{x}) = E[L(y, \hat{f}(\dot{x}))],$$

which is the expected loss for feature \dot{x} . The goal of a predictor is to minimize risk, the predictor which achieves this will depend on the loss function.

We first assume that \mathbf{y} is quantitative. In this case we define the model

$$y = f(\dot{x}) + \epsilon. \quad (15.2)$$

Here, ϵ is a random error, assumed to have a mean $E[\epsilon] = 0$. Furthermore, ϵ is often assumed to be normally distributed, that is, $\epsilon \sim N(0, \sigma^2)$. The conditional density of response y is then $y \sim N(f(\dot{x}), \sigma^2)$. There are good reasons for this assumption, but the basic ideas do not depend on this. Of course, it is possible that σ^2 depends on \dot{x} . The usual practice is to first develop a theory of statistical modeling based on the assumption of constant σ^2 , then to modify these models for more general cases. An example of a model with varying σ^2 (logistic regression) will be discussed in Chapter 10.

The mapping f is not known, so must be estimated by \hat{f} . We now formally state the problem. We are given data (\mathbf{y}, \mathbf{X}) . Depending on the methodology, \hat{f} is to be chosen from some class of functions $\hat{f} \in \mathcal{F}$. Clearly, \hat{f} should be close to the f given in (15.2). We can achieve this by systematically testing candidate functions $\hat{f} \in \mathcal{F}$ using the data. Assuming ϵ is not too large, if

$\hat{f} \approx f$, then we would expect

$$y_i \approx \hat{f}(\dot{x}_i), \quad 1, \dots, n.$$

All elements of this approximation are observable, so we develop an aggregate *goodness of fit* measure. The most commonly used is the *error sum of squares*:

$$SSE = \sum_{i=1}^n (y_i - \hat{f}(\dot{x}_i))^2 = \sum_{i=1}^n e_i^2,$$

where $e_i = y_i - \hat{f}(\dot{x}_i)$ are the *residuals*. Note that SSE is also known as the *residual sum of squares* RSS . This corresponds to a loss function $L(x, y) = (x - y)^2$.

The *least squares* fit \hat{f} is then

$$\hat{f} = \operatorname{argmin}_{f^* \in \mathcal{F}} SSE[f^*],$$

where we write $SSE[f^*]$ to emphasize the dependence on the sum of squares on f^* . By convention, when we write SSE alone, this refers to the minimum possible value over \mathcal{F} .

We now ask a crucial question. What will SSE be if we are correct, that is, if $\hat{f} = f$? In this case, $e_i = y - f(\dot{x}) = \epsilon_i$, where ϵ_i is the true error term given in (15.2). If we assume that $\operatorname{var}[\epsilon_i] = \sigma^2$, whether or not ϵ_i is normally distributed, then

$$SSE = \sum_{i=1}^n \epsilon_i^2,$$

and the *mean squared error* MSE will be

$$MSE = \frac{1}{n} \sum_{i=1}^n \epsilon_i^2 \approx \sigma^2. \quad (15.3)$$

This means the object is not to make the MSE as close to zero as possible, rather, it is to make it as close to σ^2 as possible.

To see the problem suggested here, suppose we attempt a simple regression model with only two observations. It will then be possible to find a line which passes exactly through these points, and we will have $MSE = 0$. This, of course, does not mean that the true model is error free, rather, it means that our modeling method is not well conceived. In particular, our model space \mathcal{F} is too large. In fact, there is a special term for this. We say that a model which is sufficiently flexible to force $\hat{f}(\dot{x}_i) = y_i$ for all i is a *saturated model* (this idea is discussed further in Chapters 9 and 10). To achieve this, we only need let \mathcal{F} be ‘the set of all functions conceivable’, and we will achieve $MSE = 0$. More realistically, it is always possible that the model space \mathcal{F} is large and rich enough that the process of minimizing MSE leads to an underestimate of σ^2 , a problem usually referred to as *overfitting*.

How do we overcome this problem, especially if we don’t know the value of σ^2 , which is usually the case? We first need to recognize that we really have two estimation problems. We need to estimate f using $\hat{f} \in \mathcal{F}$, but we also need to estimate the appropriate level of complexity for \mathcal{F} . That is, we are estimating both \mathcal{F} and \hat{f} . This process is referred to as *model selection*.

Of course, this problem is related to the estimation of σ^2 , and a general solution is expressible in those terms. Suppose we set our goal not as selecting \hat{f} which minimizes MSE , but as the function which minimizes

$$MSE_{test} = E \left[(y' - \hat{f}(\dot{x}'))^2 \right]$$

where (y', \dot{x}') is an response/observation pair not previously sampled (but sampled under identical conditions). Then, omitting some details, we have, given model (15.2),

$$\begin{aligned} MSE_{test} &= E[(y' - f(\dot{x}') + f(\dot{x}') - \hat{f}(\dot{x}'))^2] \\ &= E[\epsilon^2 + 2\epsilon(f(\dot{x}') - \hat{f}(\dot{x}')) + (f(\dot{x}') - \hat{f}(\dot{x}'))^2] \\ &= E[\epsilon^2] + E[(f(\dot{x}') - \hat{f}(\dot{x}'))^2] \\ &= \sigma^2 + E[(f(\dot{x}') - \hat{f}(\dot{x}'))^2]. \end{aligned} \quad (15.4)$$

The second term of (15.4) is positive, and also approaches 0 as \hat{f} becomes more accurate, in which case MSE_{test} yields an estimate of σ^2 . Therefore, finding \hat{f} which minimizes MSE_{test} is a better strategy than minimizing the MSE defined in (15.3). In fact, this is the approach which minimizes risk based on squared error loss.

15.6 Cross-Validation

The problem now is the estimation of MSE_{test} . We can think of this as a two stage process. First, we build a predictor \hat{f} using *training data* (\mathbf{y}, \mathbf{X}) . Then we estimate MSE_{test} distinct *test data* $(\mathbf{y}', \mathbf{X}')$. If \hat{f} is the predictor fit using the training data, then

$$MSE_{test} \approx \frac{1}{n} \sum_{i=1}^{n'} (y'_i - \hat{f}(\dot{x}'_i))^2,$$

where (y'_i, \dot{x}'_i) , $i = 1, \dots, n'$, are the response/observation pairs from the test data set. It is important to note that the data used to build the predictor \hat{f} is independent of the data used to test the predictor's accuracy. The degree to which a predictors' accuracy is overestimated by failing to do this can be surprisingly large, particularly with smaller data sets.

So, where does the test data come from? In medical studies, collecting new data to test a previously developed model is inevitable, for any model which shows promise. Absent this, we may take the point of view that we already have test data. All we need do is divide our current data into training and test data sets, preferably at random. At this point, we may then refer to the MSE calculated from training data as MSE_{train} .

We may then use the following approach.

Algorithm 15.1. For a given data set (\mathbf{y}, \mathbf{X}) take the following steps:

- (i) Define a sequence of model spaces $\mathcal{F}_1, \dots, \mathcal{F}_K$.
- (ii) Calculate \hat{f}_i minimizing MSE_{train} on model space \mathcal{F}_i , $i = 1, \dots, K$.
- (iii) Estimate MSE_{test} for each predictor \hat{f}_i , $i = 1, \dots, K$.
- (iv) Select the predictor with the smallest MSE_{test} .

This simple partition method is sound, but may be subject to considerable variability, and the predictor may depend considerably on the particular training/test partition. One alternative is *cross-validation* (CV). Suppose we delete the first response/observation pair from the data set, then fit a predictor $\hat{f}^{(-1)}$ using the remaining data. Then we may expect

$$E[(y_1 - \hat{f}^{(-1)}(\dot{x}_1))^2] \approx MSE_{test}.$$

Doing this for each observation yields the CV estimate

$$MSE_{CV} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}^{(-i)}(\dot{x}_i))^2, \quad (15.5)$$

and we expect $MSE_{CV} \approx MSE_{test}$.

More generally, we have k -fold cross validation. The sample is divided into k groups, or *folds*. The first group is used as test data for a model fit with the remaining (training) data, yielding $MSE_{test}(1)$. This is repeated for each group, and the cross-validated MSE is taken to be the average

$$MSE_{CV}[k] = \frac{1}{k} \sum_{i=1}^k MSE_{test}(i).$$

When $k = n$, we have (15.5), commonly referred to as *leave-one-out* CV (LOOCV).

In this way Algorithm 15.1 is altered accordingly:

Algorithm 15.2. For a given data set (\mathbf{y}, \mathbf{X}) and some fixed k take the following steps:

- (1) Define a sequence of model spaces $\mathcal{F}_1, \dots, \mathcal{F}_K$.
- (2) Calculate $MSE_{CV}[k]$ for each model space $i = 1, \dots, K$.
- (3) Select the model space yielding the smallest $MSE_{CV}[k]$. Use this to construct the predictor.

15.7 Bias and Variance

It is worth decomposing equation (15.4) a little further. First note that an estimator $\hat{\theta}$ of any parameter θ is *unbiased* if $E[\hat{\theta}] = \theta$ (otherwise they are *biased*). Despite what we might expect, not all estimators are unbiased, including many widely used ones. The *bias* of an estimate is defined as

$$Bias[\hat{\theta}] = E[\hat{\theta}] - \theta.$$

Then for squared error loss function $L(\theta, \hat{\theta})$, the risk is, after some algebra,

$$\begin{aligned} R(\theta) &= E[(\hat{\theta} - \theta)^2] \\ &= E\left[\left(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta\right)^2\right] \\ &= var[\hat{\theta}] + Bias[\hat{\theta}]^2. \end{aligned} \quad (15.6)$$

It might seem a simple matter to subtract the bias from $\hat{\theta}$ to yield an unbiased estimated, thereby reducing risk. Of course, we can't do this if the bias depends on θ , which is unknown, and this is often the case.

In statistical inference we are sometimes confronted with the bias/variance tradeoff suggested by (15.6). Suppose we implement Algorithm 15.1 or 15.2. We first note that MSE_{test} is an estimate of risk. Then, we will often find that the sequence of model spaces $\mathcal{F}_1, \dots, \mathcal{F}_K$ represents a gradual decrease in complexity. More complex models tend to have less bias, due to their greater flexibility, but also more variance, again due to their greater flexibility. As complexity decreases, the variance term in (15.6) decreases, while the bias term increases. Hopefully, at some point within the sequence we will find a ‘sweet spot’, where the optimal tradeoff between variance and bias yields the minimum MSE_{test} or, approximately, risk.

15.8 Model Selection for Classifiers

The approach of Section 15.6 can be used for classifiers. We are using a different loss function, usually $L(y, \hat{y}) = I\{y \neq \hat{y}\}$, so that risk is now

$$R(\hat{f}(\dot{x})) = P(y \neq \hat{f}(\dot{x}))$$

for a new response/observation pair (y, \dot{x}) . Given training data, the risk can be estimated as the observed error rate

$$CE_{train} = \frac{1}{n} \sum_{i=1}^n \{y_i \neq \hat{f}(\dot{x}_i)\}.$$

Of course, this measure suffers from the same defects as MSE_{train} , that is, it is subject to overfitting, and may overestimate the predictor’s accuracy. We may use CV and Algorithms 15.1-15.2 in much the same way as described in Section 15.6, except that observed classification error is used in place of mean squared error. Then the quantities CE_{test} , CE_{CV} and $CE_{CV}[k]$ follow in much the same way.

15.9 Postscript

The subjects of prediction, classification and model selection are foundational, and best studied in the context of a more general theory of statistical inference (Section 1.2). There is no single set of practice problems for this chapter. Of course, these ideas support most of the methods considered in these notes, so related practice problems can be found throughout Part III.

A number of problems in Chapters 28 and 30 make use of cross-validation for variable or parameter selection. We will see in later chapters that many R functions either support cross validation, or are accompanied in their packages by modified functions which do.

Problems 30.19, 30.20 and 30.22 involve more analytical explorations of the bias/variance trade-off central to the problem of prediction, and demonstrate that cross-validation is merely one method of resolving these problems.

There is also a demonstration software file `CROSS-VALIDATION.R` which contains examples illustrating some of the finer points of cross validation. These are generally introduced in the context of methods to be discussed in later chapters.

The importance of cross validation cannot be overstated. The difference between estimates of prediction accuracy obtained with and without cross validation can be large, even alarming. In other words, cross validation might prevent costly errors (a demonstration of this can be found in

CROSS-VALIDATION.R). Even when used, it is important to ensure that *all* steps involved in building a classifier or predictive model are included within the cross validation loop. A common error is to use all data for an initial feature selection step, before applying cross validation to the selection of the remaining model parameters (see Ambroise and McLachlan (2002) for an interesting discussion of this problem). Problem 28.13 illustrates this issue.

James *et al.* (2013) and Friedman *et al.* (2001) contain further discussion on the various topics considered in this chapter. See also Efron and Gong (1983) and Picard and Cook (1984).

Chapter 16

Bayes Theorem and Classification

If we are given a conditional probability $P(E | A)$ we often would like to “reverse the order” of the events to obtain $P(A | E)$. To do this we use *Baye’s theorem*

Theorem 16.1. For two events A and E , with $P(E) > 0$, we have

$$P(A | E) = P(E | A) \frac{P(A)}{P(E)}. \quad (16.1)$$

□

Proof. The Equation (16.1) is proven with the following argument:

$$\begin{aligned} P(A | E) &= \frac{P(AE)}{P(E)} \\ &= \frac{P(E | A)P(A)}{P(E)}. \end{aligned}$$

□

The following definition, though straightforward, is quite important to understanding the current chapter.

Definition 16.1. In the context of Baye’s theorem $P(A)$ is the *prior probability* of A , and $P(A | E)$ is the *posterior probability* of A given information E . □

The following is a useful variation of Baye’s theorem.

Theorem 16.2. Suppose events A_1, \dots, A_n is a partition of sample space S , that is, the events are mutually exclusive with

$$S = \bigcup_{i=1}^n A_i.$$

For any $1 \leq i \leq n$

$$\begin{aligned} P(A_i | E) &= \frac{P(E | A_i)P(A_i)}{P(E)} \\ &= \frac{P(E | A_i)P(A_i)}{P(E | A_1)P(A_1) + \dots + P(E | A_n)P(A_n)}. \end{aligned} \quad (16.2)$$

□

Proof. Equation 16.2 follows from the law of total probability. \square

Example 16.1. Suppose a test for a certain infection is evaluated by administering the test to 50 patients with the infection, and 100 patients known to be without the infection (control patients). The test was positive for 49 of the 50 infected patients and positive for 4 of the 100 control patients. Let

$$\begin{aligned} T &= \{ \text{Patient tests positive} \} \\ D &= \{ \text{Patient has infection} \} \end{aligned}$$

From the above data we can estimate directly

$$\begin{aligned} P(T | D) &= 49/50 \\ P(T^c | D) &= 1/50 \\ P(T | D^c) &= 4/100 \\ P(T^c | D^c) &= 96/100. \end{aligned}$$

Thus, from the data we get directly

$$P(T | D) = \text{Probability of testing positive when infected}$$

and

$$P(T | D^c) = \text{Probability of testing positive when not infected}$$

but what is of ultimate interest are the probabilities

$$P(D | T) = \text{Probability of being infected when testing positive}$$

and

$$P(D | T^c) = \text{Probability of being infected when not testing positive}$$

since this is the quantity which is clinically relevant. We use Baye's Theorem to calculate these probabilities, setting

$$\begin{aligned} A_1 &= D \\ A_2 &= D^c. \end{aligned}$$

Then

$$\begin{aligned} P(D | T) &= \frac{P(T | D)P(D)}{P(T | D)P(D) + P(T | D^c)P(D^c)} \\ &= \frac{(49/50)P(D)}{(49/50)P(D) + (4/100)P(D^c)} \end{aligned}$$

and

$$\begin{aligned} P(D | T^c) &= \frac{P(T^c | D)P(D)}{P(T^c | D)P(D) + P(T^c | D^c)P(D^c)} \\ &= \frac{(1/50)P(D)}{(1/50)P(D) + (96/100)P(D^c)}. \end{aligned}$$

Note that in order to evaluate these probabilities we need to know $P(D)$ which was not obtained from the original experiment. This should be the case, since if $P(D) = 0$ (*i.e.*, the infection is nonexistent) we would also expect both $P(D | T) = P(D | T^c) = 0$. \square

16.1 Odds

The term *odds* is synonymous with probability, and is formally defined as follows:

Definition 16.2. For a given event A we define the *odds* to be

$$Odds(A) = \frac{P(A)}{P(A^c)} = \frac{P(A)}{1 - P(A)}.$$

□

If I roll a die, the probability of getting a six is $1/6$, but the odds are $1/5$. Mathematically, the odds and the probability of an event A are equivalent, since we can calculate the odds from the probability, as well as the probability from the odds:

$$P(A) = \frac{Odds(A)}{1 + Odds(A)}.$$

In particular, if A is certain to occur then

$$\begin{aligned} P(A) &= 1 \\ Odds(A) &= \infty \end{aligned}$$

and if A is certain to not occur then

$$\begin{aligned} P(A) &= 0 \\ Odds(A) &= 0. \end{aligned}$$

We can also define the *conditional odds* of A given E .

Definition 16.3. The *conditional odds* of A given E is defined as

$$Odds(A | E) = \frac{P(A | E)}{P(A^c | E)} = \frac{P(A | E)}{1 - P(A | E)}.$$

□

The conditional odds leads to a particularly intuitive form of Baye's theorem.

Theorem 16.3. The conditional odds of A given E may be expressed

$$Odds(A | E) = \frac{P(E | A)}{P(E | A^c)} \times Odds(A). \quad (16.3)$$

□

Proof. Equation (16.3) is proven with the following argument:

$$\begin{aligned} Odds(A | E) &= \frac{P(A | E)}{P(A^c | E)} \\ &= \frac{P(E | A)P(A)}{P(E)} \times \frac{P(E)}{P(E | A^c)P(A^c)} \\ &= \frac{P(E | A)}{P(E | A^c)} \times \frac{P(A)}{P(A^c)} \\ &= \frac{P(E | A)}{P(E | A^c)} \times Odds(A). \end{aligned}$$

□

16.2 The Bayesian Model

Under the **Bayesian model** we are interested in the probability of a hypothesis A , or more specifically, the effect on this probability of the introduction of information or evidence E . There may be a well known prevalence of a certain condition (hypothesis A) among a population. For any given patient entering a clinic, this prevalence may be $P(A)$. A diagnostic test is then done. Let E be the event that this test is positive. We are now no longer interested in $P(A)$, but in $P(A | E)$ or $P(A | E^c)$, depending on the outcome of the test.

Based on an evaluation of the accuracy of the test, we may know $P(E | A)$ and $P(E | A^c)$. Examining Equation (16.3), we define the *likelihood ratio* as follows:

Definition 16.4. When considering the odds of an event A given evidence E , the *likelihood ratio* is given by

$$LR = \frac{P(E | A)}{P(E | A^c)},$$

from which we get, as a reexpression of Theorem 16.3,

$$Odds(A | E) = LR \times Odds(A). \quad (16.4)$$

We refer to $Odds(A)$ as the *prior odds* and to $Odds(A | E)$ as the *posterior odds*. \square

The relationship between the prior and posterior odds is the same as that between the prior and posterior probability. However, Equation (16.4) very neatly captures the ability of the evidence to alter our assessment of the probability of a hypothesis in a way which does not depend on the prior probability.

Example 16.2. We will express the previous Example 16.1 in terms of odds of having the infection. If a patient tests positive, the odds are adjusted by the formula

$$\begin{aligned} Odds(D | T) &= \frac{P(T | D)}{P(T | D^c)} \times Odds(D) \\ &= \frac{49/50}{4/100} \times Odds(D) \\ &= 24.5 \times Odds(D) \end{aligned}$$

so that testing positive *increases* the odds of having the infection by a factor of 24.5.

If the patient tests negative, the odds are adjusted by the formula

$$\begin{aligned} Odds(D | T^c) &= \frac{P(T^c | D)}{P(T^c | D^c)} \times Odds(D) \\ &= \frac{1/50}{96/100} \times Odds(D) \\ &= \frac{1}{48} \times Odds(D) \end{aligned}$$

so that testing negative *decreases* the odds of having the infection by a factor of 48.

We are therefore in a better position to evaluate the accuracy of the test when the problem is expressed in terms of odds. \square

Example 16.3. Suppose blood collected at a crime scene is typed for DNA. A genotype if found which is estimated to occur in the population with a frequency of p . A suspect is similarly typed and found to have the same genotype. Suppose

$$\begin{aligned} A &= \{ \text{Suspects blood is that found at the crime scene} \} \\ E &= \{ \text{Suspect has the same genotype as blood found at crime scene} \} \end{aligned}$$

Then the likelihood ratio is constructed by noting that

$$\begin{aligned} P(E | A) &= 1 \\ P(E | A^c) &= p \end{aligned}$$

giving

$$\begin{aligned} LR &= \frac{P(E | A)}{P(E | A^c)} \\ &= \frac{1}{p} \end{aligned}$$

so that the odds that the blood is the same is adjusted by

$$\begin{aligned} Odds(A | E) &= LR \times Odds(A) \\ &= \frac{1}{p} \times Odds(A) \end{aligned}$$

We usually have no way of directly evaluating $Odds(A)$. We can only describe how the evidence changes the odds. If it were established without doubt that the suspect was not at the crime scene by other evidence then we would have

$$Odds(A) = 0$$

and

$$Odds(A | E) = 0$$

for any value of LR . If guilt were established with absolute certainty then

$$Odds(A) = \infty$$

and

$$Odds(A | E) = \infty.$$

for any value of LR .

Now, suppose the genotype does not match. (That is, E^c occurs). The likelihood ratio is now calculated from

$$\begin{aligned} P(E^c | A) &= 0 \\ P(E^c | A^c) &= 1 - p \end{aligned}$$

giving $LR = 0$ so that

$$\begin{aligned} Odds(A | E) &= LR \times Odds(A) \\ &= 0 \end{aligned}$$

for any $Odds(A)$. □

16.3 The Fallacy of the Transposed Conditional

In the previous example suppose we set $p = 1/100$. We could then say

$$P(E | A^c) = 1/100$$

which is the probability of a genotype match if the suspect is not guilty. A common error is to *transpose the conditional* which yields (incorrectly)

$$P(A^c | E) = 1/100.$$

This statement says that the probability that the suspect is not guilty is 1/100 if a match occurs. After some algebra we then get

$$\begin{aligned} P(A | E) &= 1 - P(A^c | E) \\ &= 99/100 \end{aligned}$$

which says that, given a match, the probability that the suspect is guilty is 99/100, which cannot be concluded from the evidence. The odds of guilt given the evidence depends on the prior odds. This is often referred to as the *prosecutor's fallacy*.

16.4 Diagnostic Testing - Basic Definitions

A common problem in medical research is the evaluation of the accuracy of diagnostic tests. This can be framed in the context of probability theory. In the simplest case, a diagnostic test is either positive (in which case the patient is predicted to have the condition being tested) or negative (in which case the patient is predicted to not have the condition being tested). There are 4 events of interest:

$$\begin{aligned} O_- &= \{ \text{the patient does not have the condition} \} \\ O_+ &= \{ \text{the patient has the condition} \} \\ T_- &= \{ \text{the patient tests negative} \} \\ T_+ &= \{ \text{the patient tests positive} \} \end{aligned}$$

Clearly, $O_-^c = O_+$ and $T_-^c = T_+$, so that $P(O_-) + P(O_+) = 1$ and $P(T_-) + P(T_+) = 1$.

Conditional probabilities and Bayes theorem can be very useful in developing a probabilistic model for these outcomes, and a widely used terminology has been developed:

$$\begin{aligned} \text{sensitivity (sens)} &= P(T_+ | O_+) \\ \text{specificity (spec)} &= P(T_- | O_-) \\ \text{positive predictive value (PPV)} &= P(O_+ | T_+) \\ \text{negative predictive value (NPV)} &= P(O_- | T_-) \\ \text{prevalance (prev)} &= P(O_+). \end{aligned} \tag{16.5}$$

Two more related definitions are sometimes used:

$$\begin{aligned} \text{true discovery rate (TDR)} &= \text{sens} \\ \text{false discovery rate (FDR)} &= P(T_+ | O_-) = 1 - \text{spec}. \end{aligned}$$

In an evaluation study, a diagnostic test will typically be administered to subjects with known outcomes, which will allow sensitivity and specificity to be estimated. However, when used in a clinical setting, the outcomes will not be known. These are to be predicted based on the test result, so it will be PPV and NPV which are more relevant. These quantities can be related to sensitivity, specificity and prevalence using Baye's Theorem:

$$\begin{aligned} PPV &= P(O_+ | T_+) \\ &= \frac{P(T_+ | O_+)P(O_+)}{P(T_+ | O_+)P(O_+) + P(T_+ | O_-)P(O_-)} \\ &= \frac{sens \times prev}{sens \times prev + (1 - spec) \times (1 - prev)} \end{aligned} \quad (16.6)$$

and

$$\begin{aligned} NPV &= P(O_- | T_-) \\ &= \frac{P(T_- | O_-)P(O_-)}{P(T_- | O_-)P(O_-) + P(T_- | O_+)P(O_+)} \\ &= \frac{spec \times (1 - prev)}{spec \times (1 - prev) + (1 - sens) \times prev}. \end{aligned} \quad (16.7)$$

It is important to understand the degree to which PPV and NPV depend on prevalence. We have already seen in Example 16.1 that if, for example, $prev = 0$ we would necessarily have $PPV = 0$, no matter what sensitivity and specificity are. On the other hand, sensitivity and specificity do not depend on prevalence, and this distinction is an important one.

16.4.1 Diagnostic tests and contingency tables

The outcomes of a study used to evaluate a diagnostic test can be summarized in Table 1 below,

Table 1: Outcomes of diagnostic testing

	Condition		
	Positive	Negative	
Test	Positive	TP	FP
	Negative	FN	TN

where

$$\begin{aligned} TP &= T_+ \cap O_+ = \text{True Positive} \\ FP &= T_+ \cap O_- = \text{False Positive} \\ TN &= T_- \cap O_- = \text{True Negative} \\ FN &= T_- \cap O_+ = \text{False Negative.} \end{aligned} \quad (16.8)$$

Table 1 can be interpreted as a contingency table, with numerical entries TP, FP, TN, FN giving the counts of subjects in each category. These can be used to estimate all important quantities. If

we let $N = TP + FP + TN + FN$ (the total number of entries in Table 1) we can calculate the *marginal probabilities*:

$$\begin{aligned} P(O_-) &= \frac{FP + TN}{N} \\ P(O_+) &= \frac{TP + FN}{N} \\ P(T_-) &= \frac{FN + TN}{N} \\ P(T_+) &= \frac{TP + FP}{N}, \end{aligned} \quad (16.9)$$

and the important diagnostic quantities

$$\begin{aligned} prev &= \frac{TP + FN}{N} = P(O_+) \\ sens &= \frac{TP}{TP + FN} = P(T_+ | O_+) \\ spec &= \frac{TN}{TN + FP} = P(T_- | O_-) \\ PPV &= \frac{TP}{TP + FP} = P(O_+ | T_+) \\ NPV &= \frac{TN}{TN + FN} = P(O_- | T_-). \end{aligned} \quad (16.10)$$

However, the prevalence must be very carefully interpreted. If we calculate *prev* directly from Table 1, we obtain the prevalence of an outcome within the study population, which may have no relationship to the prevalence in any given clinical population. This would be especially true if the study was designed to ensure a large enough sample of disease positive subjects to accurately evaluate the test. In such cases, we would expect *prev* to be much higher than it would be in a clinical population.

Therefore, it is important to understand that it is always possible, and usually preferable, to calculate prevalence independently of sensitivity and specificity. In particular, if we are using a study such as that represented by Table 1 we would use equations (16.10) to estimate *sens* and *spec* but not *prev*, *PPV* or *NPV*. Instead, we would use an independent estimate of *prev* which more accurately estimates the prevalence within the clinical population of interest, and then use (16.6)-(16.7) with that value of *prev*.

In summary, the important question is whether or not the subjects used in Table 1 are representative of the population in which the test is to be applied, in terms of the relative frequencies of outcomes O_+ and O_- . The values of *prev*, *PPV* and *NPV* calculated by equations (16.10) would be interpretable only if this is the case.

16.4.2 The use of odds in the evaluation of diagnostic tests

In the absence of a reliable estimate of prevalence, the accuracy of a diagnostic test can be expressed using odds, as shown above. Using the previous terminology we have

$$LR = \frac{P(T_+ | O_+)}{P(T_+ | O_-)} = \frac{\text{sensitivity}}{1 - \text{specificity}}$$

so that

$$Odds(O_+ | T_+) = LR \times Odds(O_+).$$

Then $Odds(O_+)$ is the prevalence expressed as odds, and the predictive ability of the test can be expressed using only the sensitivity and specificity.

Note that we can also assess the accuracy of a negative test outcome. In this case we can distinguish between the LR for a positive test outcome LR_+ and the LR for a negative test outcome LR_- :

$$LR_+ = LR \text{ as defined above } LR_- = \frac{P(T_- | O_+)}{P(T_- | O_-)} = \frac{1 - \text{sensitivity}}{\text{specificity}}$$

so that

$$\begin{aligned} Odds(O_+ | T_+) &= LR_+ \times Odds(O_+) \\ Odds(O_+ | T_-) &= LR_- \times Odds(O_+). \end{aligned}$$

Example 16.4. Studies into the accuracy of a diagnostic test often proceed by pairing the test with a *gold standard* in a study group of size N , the latter assumed to be perfectly accurate. In this case, we can estimate sensitivity and specificity. After the study we would construct a contingency table like the following ($N = 1000$):

Table 2: Outcomes of diagnostic testing for Example 16.4

Test	Condition	
	Positive	Negative
Positive	30	110
Negative	10	850

We have, using equations (16.10), ($TP = 30$, for example):

$$\begin{aligned} sens &= 30/40 = 0.75 \\ spec &= 850/960 \approx 0.885 \\ LR_+ &= (30/40)/(1 - 850/960) \approx 6.545 \\ LR_- &= (1 - 30/40)/(850/960) \approx 0.282 \end{aligned}$$

and the Bayes model gives

$$\begin{aligned} Odds(O_+ | T_+) &\approx 6.545 \times Odds(O_+) \\ Odds(O_+ | T_-) &\approx 0.282 \times Odds(O_+). \end{aligned}$$

A positive test result increases the odds of a positive outcome by a factor of 6.545, while a negative test result decreases the odds of a positive outcome by a factor of 0.282.

Next, if we calculate $prev$, PPV and NPV directly from the contingency table, using equations (16.10). we would have

$$\begin{aligned} prev &= (10 + 30)/1000 = 0.04 \\ PPV &= 30/140 \approx 0.214 \\ NPV &= 850/860 \approx 0.988. \end{aligned}$$

The values of PPV and NPV assume a prevalence of 4%, estimated directly from the data. Suppose the true prevalence was 2%. We would then use (16.6)-(16.7) with $prev = 0.02$ and the estimates of $sens$ and $spec$ obtained from the data (remember that these quantities do not depend on the prevalence). This gives

$$\begin{aligned} PPV &= \frac{sens \times prev}{sens \times prev + (1 - spec) \times (1 - prev)} \\ &= \frac{0.75 \times 0.02}{0.75 \times 0.02 + (1 - 0.885) \times (1 - 0.02)} \\ &\approx 0.117 \end{aligned}$$

and

$$\begin{aligned} NPV &= \frac{spec \times (1 - prev)}{spec \times (1 - prev) + (1 - sens) \times prev} \\ &= \frac{0.885 \times (1 - 0.02)}{0.885 \times (1 - 0.02) + (1 - 0.75) \times 0.02} \\ &\approx 0.994. \end{aligned}$$

Reducing the prevalence by 1/2 results in a reduction in PPV of almost the same magnitude (verify that if we use $prev = 0.04$ in equations (16.6)-(16.7) we reproduce the values of PPV and NPV obtained using equations (16.10)).

Using either method, that PPV is much smaller than sensitivity is typical, and is due to the fact that PPV depends on the prevalence. Expecting the two to be equal is an example of the ‘prosecutor’s fallacy’, since one is obtained from the other by transposing the conditional.

Note also that NPV is quite large. This is a function both of the ability of the test to rule out a positive outcome (measured by specificity) and of the relatively small prevalence. This means NPV would be smaller when the test is confined to a higher risk population. \square

16.5 The Odds Ratio

Consider the following events

$$\begin{aligned} O_- &= \{ \text{the patient does not have the condition} \} \\ O_+ &= \{ \text{the patient has the condition} \} \\ G_1 &= \{ \text{the patient is in Group 1} \} \\ G_2 &= \{ \text{the patient is in Group 2} \}. \end{aligned}$$

Typically, we are interested in comparing

$$P(O_+ | G_1) \text{ and } P(O_+ | G_2).$$

Perhaps the obvious comparison method is to examine the difference:

$$\Delta = P(O_+ | G_1) - P(O_+ | G_2).$$

This will be, sometimes, a reasonable approach, but will not work well when the probabilities are small. Alternatively, we have the *relative risk*

$$RR = \frac{P(O_+ | G_1)}{P(O_+ | G_2)}$$

and the *odds ratio* (OR)

$$OR = \frac{Odds(O_+ | G_1)}{Odds(O_+ | G_2)} = \frac{P(O_+ | G_1)/(1 - P(O_+ | G_1))}{P(O_+ | G_2)/(1 - P(O_+ | G_2))}.$$

The OR has an interesting property, in that events defining it may be transposed, that is

$$OR = \frac{Odds(G_1 | O_+)}{Odds(G_1 | O_-)},$$

so that the OR does not depend on the marginal probabilities (that is, the prevalences). For some applications, this is a considerable advantage, for the reasons discussed earlier in this chapter.

16.6 Bayes Classifiers

We now consider the problem of classification. We have, as before, responses and features \mathbf{y} and \mathbf{X} . But now, \mathbf{Y} is qualitative, and the predictor function $\hat{f}(\dot{x})$ now assigns a predicted class from \mathcal{E}_y to each feature \dot{x} . The additive error model (15.2) is no longer applicable. Instead, either a predicted class is correct, or an incorrect class is predicted. Suppose there are m classes in \mathcal{E}_y , which we can always label $1, \dots, m$. If there is any information in the features \dot{x} which is able to distinguish between classes, then there must be conditional densities $f(\dot{x} | y = j)$, $j \in 1, \dots, m$ which differ noticeably from each other. We can then use Bayes theorem to get conditional distribution:

$$P(y = j | \dot{x}) = \frac{f(\dot{x} | y = j)\pi_j}{f(\dot{x})} \quad (16.11)$$

where π_j is the prior probability of class j (the prevalence of class j in the population of interest), and

$$f(\dot{x}) = \sum_{j=1}^m f(\dot{x} | y = j)\pi_j$$

is the total probability distribution of \dot{x} . Suppose we use loss function:

$$L(y, \hat{f}(\dot{x})) = I\{y \neq \hat{f}(\dot{x})\},$$

that is, the loss is one if and only if the classification is incorrect. Then given feature \dot{x} , the minimum risk classifier, referred to as the *Bayes classifier* can be shown to be

$$\hat{f}(\dot{x}) = \operatorname{argmax}_{j \in \mathcal{E}_y} P(y = j | \dot{x}) = \operatorname{argmax}_{j \in \mathcal{E}_y} f(\dot{x} | y = j)\pi_j, \quad (16.12)$$

noting that the denominator in (16.11) does not depend on class type j . Of course, we may choose to define a more complex loss function. For example, if the true class is 1, then it may be a less costly error to predict 2 than 3. We may set $L(1, 2) = 1/2$ and $L(1, 3) = 1$ accordingly. Then a distinct classifier will minimize risk.

16.6.1 Prior probabilities

It is important to understand the role of the prior probabilities $\tilde{\pi} = (\pi_1, \dots, \pi_m)$, since the optimal properties of the Bayes classifier depend on their correct identification. To see this, suppose we are developing a test for the presence of a type of infection. When we develop the test, we presumably have training data (\mathbf{y}, \mathbf{X}) with which to estimate conditional densities $f(\dot{x} | y = j)$. However, it would usually not be appropriate to use as estimates of $\tilde{\pi}$ the proportions of each class in the training data. We would hope that the prior probability of infection, say π_1 , would be much less than 1/2, and so for the purposes of efficient estimation, the proportion of the infected class in \mathbf{y} should be chosen to be much higher than π_1 .

For this reason, the choice of prior probabilities is often made independently of the training data. To take an extreme example, suppose the infection in question is nonexistent (small pox, for example). In this case, it would be appropriate to set $\pi_1 = 0$, in which case the Bayes classifier will predict noninfection for any observation \dot{x} .

When there is apparently no basis on which to choose prior probabilities a commonly used strategy is to select an *uninformative prior*, which weights each prediction equally. When the number of classes is finite, the logical choice would be the *uniform prior* $\pi_j = 1/m$. This is an example of the *principle of indifference*. However, in more complex example of Bayesian prediction, the question of what constitutes an uninformative prior can be a very deep one, and there may be a number of competing answers.

For our purposes, there are three choices:

- (1) We use prior knowledge to inform the choice of $\tilde{\pi}$.
- (2) We use a uniform prior as the indifferent choice.
- (3) We use estimates for $\tilde{\pi}$ the class frequencies observed in the training data

The third option is the easiest to take, since it is often the default choice of algorithms which build classifiers. Sometimes it will be the correct choice, but the fact that a choice is being made should always be kept in mind.

16.6.2 Naive Bayes classifiers

Typically, the technical issue for Bayes classifiers is the estimation of the conditional density $f(\dot{x} | y = j)$ in (16.12). In principle, this may be done directly from the training data. However, there is scalability issue with respect to the number of features p . If we consider, for example, the p -dimensional multivariate normal distribution we note that the mean vector contains p parameters, while the covariance matrix contains $p + p(p - 1)/2$ parameters (p variances and $p(p - 1)/2$ covariances). This means the number of parameters to be estimated is of order $O(p^2)$, and so does not scale well with increasing numbers of predictors. A simple (ie ‘naive’) solution is to assume that the features are independent (even when evidence to the contrary exists), so that the conditional density is

$$f(\dot{x} | y = j) = \prod_{i=1}^p f_i(x_i | y = j).$$

Instead of estimating one multivariate density $f(\dot{x} | y = j)$, p univariate densities $f_i(x_i | y = j)$ are estimated, so that the total number of parameters to estimate is of order $O(p)$. In the normal case, that number is exactly $2p$, absent any further constraints.

16.7 K Nearest Neighbor (KNN) Classifiers and Regression

Assume there is a distance function d defined on the feature space \mathcal{E}_x . We have data set (\mathbf{y}, \mathbf{X}) . For any feature \dot{x} and $K \geq 1$ define the neighborhood

$$\mathcal{N}_K(\dot{x}) = \{i : \text{rank of } d(\dot{x}_i, \dot{x}) \text{ no greater than } K\},$$

that is, $\mathcal{N}_K(\dot{x})$ consists of the indices of the K features nearest to \dot{x} (\dot{x} need not be in \mathbf{X}). Then for quantitative responses, the KNN predictor of $f(\dot{x})$ is

$$\hat{f}(\dot{x}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\dot{x})} y_i,$$

or, the mean response in the neighborhood.

For the classification problem,

$$\hat{f}(\dot{x}) = j \text{ if } j \text{ is the most frequent class in } \mathcal{N}_K(\dot{x}),$$

where ties are resolved randomly.

In a typical application, Algorithms 15.1-15.2 may be used with the model spaces $\mathcal{F}_1, \dots, \mathcal{F}_N$, where \mathcal{F}_i is the model space for the KNN classifier with neighborhood size parameter K_i .

16.8 Linear and Quadratic Discriminant Analysis

Linear and quadratic discriminant analysis (LDA and QDA) are special cases of Bayes classifiers based on the normal distribution. Suppose, we have p features and m classes, and we wish to build a Bayes classifier with conditional distributions

$$f(\dot{x} | y = j) = \phi(\dot{x}; \boldsymbol{\mu}_j, \Sigma_j), \quad j = 1, \dots, m.$$

This requires an estimate of mean vectors and covariance matrices $\boldsymbol{\mu}_j, \Sigma_j$ for each class j . As discussed in Section 16.6.2 without further constraint the number of parameters to estimate becomes very large with increasing p . One way to control this is to assume (if justified) that the class covariance matrices are equal, so that $\Sigma_j = \Sigma$ for any j . Another method is to use a naive Bayes classifier (Section 16.6.2). In this case, the feature independence assumption forces each Σ_j to be a diagonal matrix, but they may still differ by class.

Once the conditional distributions are given the Bayes classifier is given directly by (16.12). It is generally simpler to apply a log transformation, in which case we have

$$\log(\phi(\dot{x}; \boldsymbol{\mu}_j, \Sigma_j) \pi_j) = -\frac{1}{2} Q_j(\dot{x}) - \frac{1}{2} \log(\det(\Sigma_j)) + \log(\pi_j) - \frac{p}{2} \log(2\pi), \quad (16.13)$$

where

$$Q_j(\dot{x}) = (\dot{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\dot{x} - \boldsymbol{\mu}_j).$$

At one level, all that is needed at this point is to calculate (16.13) for each class for any given observation \dot{x} , then take as the prediction that class with the largest value. If needed, the parameters can be estimated, as discussed in Section 16.8.1 below.

However, some insight can be gained by trying to refine the approach. First, we note that the procedure can be represented as a collection of functions $h_j(\dot{x})$, $j = 1, \dots, m$, yielding prediction

$$\hat{y} = \operatorname{argmax}_j h_j(\dot{x}). \quad (16.14)$$

This means these functions can be subject to a common strictly increasing transformation while yielding exactly the same predictions. This includes addition of a constant (positive or negative), or multiplication by a positive scalar. If we initially set $h_j(\dot{x})$ equal to (16.13), we might then note that there is a common term $-\frac{p}{2} \log(2\pi)$ (that it, it does not depend on j). We can therefore remove this term to get

$$h_j(\dot{x}) = -\frac{1}{2}Q_j(\dot{x}) - \frac{1}{2} \log(\det(\Sigma_j)) + \log(\pi_j).$$

Next, suppose we adopt a uniform prior $\pi_j = 1/m$ (Section 16.6.1). The term $\log(\pi_j)$ is now constant across classes, so it can be removed, yielding classifiers:

$$h'_j(\dot{x}) = -\frac{1}{2}Q_j(\dot{x}) - \frac{1}{2} \log(\det(\Sigma_j)). \quad (16.15)$$

Finally, suppose the covariance matrices $\Sigma_j = \Sigma$ are constant. The term $-\frac{1}{2} \log(\det(\Sigma_j))$ may then be removed, but further simplification is possible. The quadratic term $Q_j(\dot{x})$ may also be decomposed:

$$\begin{aligned} Q_j(\dot{x}) &= (\dot{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\dot{x} - \boldsymbol{\mu}_j) \\ &= \dot{x}^T \Sigma_j^{-1} \dot{x} - 2(\dot{x})^T \Sigma_j^{-1} \boldsymbol{\mu}_j + \boldsymbol{\mu}_j^T \Sigma_j^{-1} \boldsymbol{\mu}_j. \end{aligned}$$

When $\Sigma_j = \Sigma$ the first term of this decomposition is constant across classes, so we have classifiers

$$h''_j(\dot{x}) = \dot{x}^T \Sigma^{-1} \boldsymbol{\mu}_j - \frac{1}{2} \boldsymbol{\mu}_j^T \Sigma^{-1} \boldsymbol{\mu}_j. \quad (16.16)$$

Note that the covariance matrix Σ still appears in (16.15), but only in terms which otherwise vary by class.

At this point we have the distinction between *linear* and *quadratic* discriminant analysis, specifically, whether or not the covariance matrices differ by class, which yield respective classifiers (in their most general form):

$$\begin{aligned} LDA \quad h_j(\dot{x}) &= \dot{x}^T \Sigma^{-1} \boldsymbol{\mu}_j - \frac{1}{2} \boldsymbol{\mu}_j^T \Sigma^{-1} \boldsymbol{\mu}_j + \log(\pi_j) \\ QDA \quad h_j(\dot{x}) &= -\frac{1}{2}(\dot{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\dot{x} - \boldsymbol{\mu}_j) - \frac{1}{2} \log(\det(\Sigma_j)) + \log(\pi_j). \end{aligned}$$

Recall the example in Figure 15.1. In general, classifiers can be defined geometrically by the boundaries they induce in the feature space. One advantage of discriminant analysis is that this boundary takes an analytical form. Suppose we have $m = 2$ classes. From (16.14) we can see that this boundary is given by the equation

$$h_1(\dot{x}) - h_2(\dot{x}) = 0.$$

By (16.17) it can be seen that for LDA this boundary is linear (boundary A , Figure 15.1) and for QDA it will be quadratic (boundary B , Figure 15.1).

Finally, the naive Bayes classifier (Section 16.6.2) is easily defined. For a multivariate normal distribution, independence is equivalent to zero covariance. Therefore a naive Bayes classifier is implemented simply by forcing each Σ_j to be a diagonal matrix. This holds for both LDA and QDA. This method is known as *diagonal discriminant analysis* (DDA).

16.8.1 Estimation for LDA/QDA

The estimation problem is straightforward. We are given training data (\mathbf{y}, \mathbf{X}) . The feature matrix is then partitioned by class, into $\mathbf{X}_1, \dots, \mathbf{X}_m$. The class mean vector $\boldsymbol{\mu}_j$ is estimated by the sample means $\bar{\boldsymbol{\mu}}_j = (\bar{x}_1, \dots, \bar{x}_p)$, where \bar{x}_i is the sample mean of feature i using class j feature data \mathbf{X}_j .

Similarly, Σ_j is estimated by the sample covariance matrix using feature data \mathbf{X}_j :

$$\hat{\Sigma}_j = \frac{1}{n_j - 1} \begin{bmatrix} \sum_{i=1}^{n_j} (x_{i1} - \bar{x}_1)^2 & \cdots & \sum_{i=1}^{n_j} (x_{i1} - \bar{x}_1)(x_{ip} - \bar{x}_p) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{n_j} (x_{ip} - \bar{x}_p)(x_{i1} - \bar{x}_1) & \cdots & \sum_{i=1}^{n_j} (x_{ip} - \bar{x}_p)^2 \end{bmatrix}.$$

where n_j is the number of observations of class j , and the indices are defined relative to \mathbf{X}_j .

If we assume identical covariance matrices $\Sigma = \Sigma_j$, then the single estimate of Σ may be obtained by *pooling* the separate class estimates:

$$\hat{\Sigma}_{pooled} = \frac{1}{n - p} \sum_{j=1}^p (n_j - 1) \hat{\Sigma}_j,$$

(compare this procedure to the pooled t -test).

For DDA, the nondiagonal terms of $\hat{\Sigma}_j$ are set to zero, and the diagonal estimated as already described.

16.9 Classification and the Receiver Operator Characteristic (ROC) Curve

We already introduced a probabilistic model for the evaluation of a classifier, in the context of diagnostic testing. This was based on an application of Baye's theorem to the following events on a probability space:

$$\begin{aligned} O_+ &= \{ \text{positive outcome} \} \\ O_- &= \{ \text{negative outcome} \} \\ T_+ &= \{ \text{positive test outcome} \} \\ T_- &= \{ \text{negative test outcome} \}. \end{aligned}$$

We defined sensitivity and specificity as the following quantities:

$$\begin{aligned} sens &= P(T_+ | O_+) \\ spec &= P(T_- | O_-), \end{aligned}$$

and we may also define the *false positive rate* and *false negative rate* as

$$\begin{aligned} fpr &= P(T_+ \mid O_-) = 1 - spec \\ fnr &= P(T_- \mid O_+) = 1 - sens. \end{aligned}$$

These quantities are relevant in the evaluation phase of the development of a classifier. The ultimate goal is to maximize the positive predictive value (PPV) and negative predictive value (NPV), defined as

$$\begin{aligned} PPV &= P(O_+ \mid T_+) \\ NPV &= P(O_- \mid T_-), \end{aligned}$$

but to do so we need to test the classifier using subjects with known outcomes O_+ and O_- , which gives *sens* and *spec*. We also need the prevalence of the outcome

$$prev = P(O_+),$$

with which Baye's theorem leads to

$$\begin{aligned} PPV &= P(O_+ \mid T_+) \\ &= \frac{P(T_+ \mid O_+)P(O_+)}{P(T_+ \mid O_+)P(O_+) + P(T_+ \mid O_-)P(O_-)} \\ &= \frac{sens \times prev}{sens \times prev + (1 - spec) \times (1 - prev)} \end{aligned}$$

and

$$\begin{aligned} NPV &= P(O_- \mid T_-) \\ &= \frac{P(T_- \mid O_-)P(O_-)}{P(T_- \mid O_-)P(O_-) + P(T_- \mid O_+)P(O_+)} \\ &= \frac{spec \times (1 - prev)}{spec \times (1 - prev) + (1 - sens) \times prev}. \end{aligned}$$

16.9.1 Classifiers based on a numerical risk score

The preceding section summarizes a probabilistic classification model for which the classifier can be reduced to two outcomes T_+ and T_- . Of course, classifiers are often based on a numerical score. We can adopt the convention that higher scores can be interpreted as evidence in favor of a positive outcome O_+ (if needed, reverse the score by multiplying by -1). In this way, the numerical classifier can be interpreted as a *risk score*, with high risk implying greater probability (risk) of a positive outcome O_+ .

To fix ideas, consider the `Melanoma` data set included in the `MASS` package:

```
> library(MASS)
> help(Melanoma)
```

Survival from Malignant Melanoma

Description

The Melanoma data frame has data on 205 patients in Denmark with malignant melanoma.

Usage

Melanoma

Format

This data frame contains the following columns:

time

survival time in days, possibly censored.

status

1 died from melanoma, 2 alive, 3 dead from other causes.

sex

1 = male, 0 = female.

age

age in years.

year

of operation.

thickness

tumour thickness in mm.

ulcer

1 = presence, 0 = absence.

Source

P. K. Andersen, O. Borgan, R. D. Gill and
N. Keiding (1993) Statistical Models based on Counting
Processes. Springer.

We will investigate the possibility of using **thickness** (tumor thickness in mm) to predict death from melanoma. We have outcome **status**, which classifies the patient as dead from melanoma (**status** = 1); alive (**status** = 2); or dead from other causes (**tttstatus** = 3). We may remove

from the analysis patients who died from other causes, leaving outcomes

$$\begin{aligned} O_+ &= \{\text{patient died from melanoma}\} \\ O_- &= \{\text{patient is still alive}\}. \end{aligned}$$

In practice, this type of analysis would take into account the observation times of the patients, which may vary considerably. For example, a patient with outcome O_- may have only been observed for a short period of time, so that that negative outcome would be more difficult to interpret than an negative outcome which follows a longer observation period. With that caveat, we will accept survival as the outcome.

We have a quick first look at the data:

```
> names(Melanoma)
[1] "time"      "status"     "sex"        "age"        "year"
"thickness"   "ulcer"
> Melanoma[1:3,]
  time status sex age year thickness ulcer
1  10     3    1  76 1972      6.76      1
2  30     3    1  56 1968      0.65      0
3  35     2    1  41 1977      1.34      0
> is.factor(Melanoma$status)
[1] FALSE
>
```

Note that `status` is not a factor variable. So, to subset the data we use the command:

```
> Melanoma2 = Melanoma[Melanoma$status < 3,]
> dim(Melanoma2)
[1] 191   7
>
```

and use data frame `Melanoma2`. There are $n = 191$ subjects remaining.

Next, look at boxplots of the variable `thickness` by outcome group (Figure 16.1):

```
> par(mfrow=c(1,1), mar=c(3,5,3,3), cex=1.1)
> boxplot(thickness ~ status, data = Melanoma2,
  names = c("Died", "Alive"), ylab="Tumor Thickness in mm.")
> for (i in 1:10) {lines(c(0,i), rep(i,2), col=4)}
```

We have superimposed lines (in blue) at `thickness` levels $1, 2, \dots, 10$. Clearly, death outcomes are associated with higher values of `thickness`, which can therefore be used as a risk score for melanoma mortality (higher values of `thickness` mean greater mortality risk). Suppose we select a *risk score threshold* T , possibly one of the blue lines. We may then define a positive test outcome as

$$T_+ = \{\text{thickness} \geq T\}. \quad (16.17)$$

This allows us to apply a classifier in an intuitive way, in the sense that if T_+ occurs we predict O_+ , and if $T_- = T_+^c$ occurs we predict O_- .

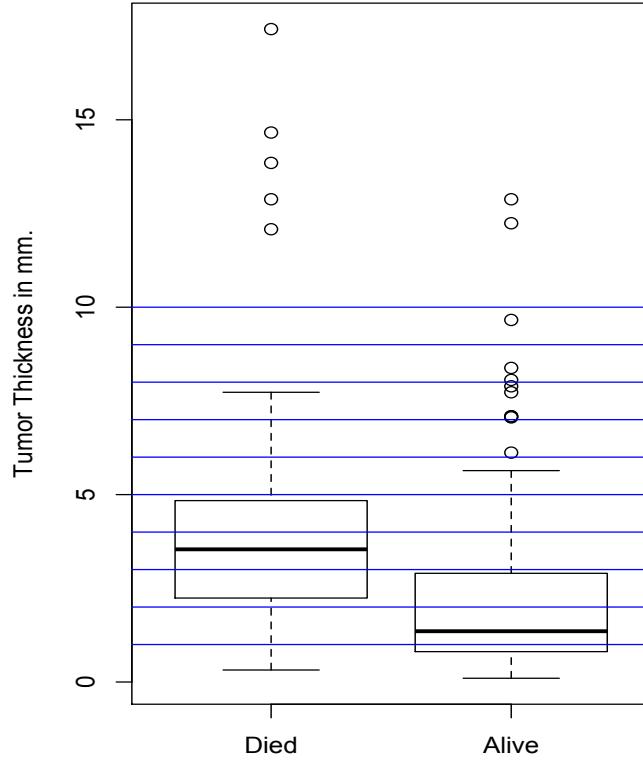


Figure 16.1: Boxplot of melanoma *thickness* variable (tumor thickness in mm) by survival outcome group.

Of course, this leaves open the problem of selecting T . If there were no overlap (that is, if the smallest risk score among the O_+ group was larger than the largest risk score among the O_- group) then the selection of T would be straightforward. If there was some T that is larger than all risk scores in the O_- group and smaller than all risk scores in the O_+ group, we would use that threshold to define a positive test according to (16.17), which would yield $sens = spec = 1$ that is, perfect classification (at least for this sample).

Of course, we don't usually expect this ideal. Suppose we consider the blue lines in Figure 16.1 as possible values for the threshold T used to define the positive test outcome T_+ in (16.17). Clearly, for each value of T we will have false positives and false negatives, as long as within each group there are risk scores on both sides of the threshold.

It is instructive, however, to consider the limiting case. If $T = 0$ (all risk scores are above 0), then the test outcome will be positive for *all* subjects in *both* groups. Since mortality is (correctly) predicted for all subjects in O_+ , we have $sens = 1$. At the same time, mortality is (incorrectly) predicted for all subjects in O_- , so $spec = 0$. This is clearly not a satisfactory predictor. If $T = 100$ (that is, a value larger than all observed risk scores) we (incorrectly) predict survival for all subjects in O_+ , so that $sens = 0$. We also (correctly) predict survival for all subjects in O_- , so that $spec = 1$.

Clearly, we must find a balance between *sens* and *spec*. As T increases, *sens* decreases and *spec* increases. At this point, we can write an R function that calculates *sens* and *spec* for a given threshold, for a given data set. The function will have to input three things, namely, the threshold T , risk score *score* and the outcome groups *gr*. The variable *gr* will be a 0-1 numerical vector, with 1 corresponding to high risk. We assume *score* and *gr* are paired. Subjects with $\text{score} \geq T$ are assigned positive test outcomes.

To estimate *sens* and *spec* from the data, we can use the following calculation:

$$\begin{aligned}sens &= \frac{P(O_+ \cap T_+)}{P(O_+)} = \frac{\text{Num subjects for which } \text{score} \geq T \text{ and } gr = 1}{\text{Num subjects for which } gr = 1} \\ spec &= \frac{P(O_- \cap T_-)}{P(O_-)} = \frac{\text{Num subjects for which } \text{score} < T \text{ and } gr = 0}{\text{Num subjects for which } gr = 0}.\end{aligned}$$

We therefore write the function:

```
> diag.thresh = function(thresh, score, gr) {
+   sens= sum( (score >= thresh) & (gr == 1) )/sum(gr == 1)
+   spec= sum( (score < thresh) & (gr == 0) )/sum(gr == 0)
+   ans = c(sens, spec)
+   names(ans) = c("Sensitivity", "Specificity")
+   return(ans)
+ }
```

We can create our data variables for input

```
> gr = 1*(Melanoma2$status == 1)
> score = Melanoma2$thickness
> gr = gr[sort.list(score)]
> score = score[sort.list(score)]
```

Note that we have sorted the paired data using the `sort.list()` function. We can, for example, get the sensitivity associated with a threshold of $T = 5$:

```
> diag.thresh(5, score, gr)
Sensitivity Specificity
 0.2456140  0.8955224
>
```

While the specificity is quite good ($spec = 0.8955224$) the sensitivity would be, by most standards, too low ($sens = 0.2456140$), and we would probably want to use a lower threshold for an actual application.

To see how the specificity and sensitivity vary with threshold T , we can create a loop to calculate a range of values for T , and create a simple table.

```
> diag.tab = NULL
> for (i in 1:10) {diag.tab =
```

```

            rbind(diag.tab,diag.thresh(i, score, gr))
+ }
> rownames(diag.tab) = paste("Threshold",1:10)
> colnames(diag.tab) = c("Sensitivity", "Specificity")
> diag.tab
      Sensitivity Specificity
Threshold 1    0.8947368   0.3507463
Threshold 2    0.7719298   0.6641791
Threshold 3    0.5964912   0.7611940
Threshold 4    0.3859649   0.8656716
Threshold 5    0.2456140   0.8955224
Threshold 6    0.1578947   0.9179104
Threshold 7    0.1403509   0.9253731
Threshold 8    0.0877193   0.9626866
Threshold 9    0.0877193   0.9776119
Threshold 10   0.0877193   0.9850746
>

```

A value of T in the range 2 to 3 would seem to offer a better balance of false positive and false negative rates.

16.9.2 ROC curves

Of course, this type of analysis can be much more refined. First of all, we can input as threshold T all observed value of the risk score, obtaining much greater resolution than the previous table. To do this, we could use a `for` loop, but it's good to remember at this point that R permits vectorized operation. This would seem to suggest that if in the function call `diag.thresh(i, score, gr)` we substitute `score` for i , we would get the *sens*, *spec* values for all observed values of `score` in a single object. However, if we try this we get:

```

> diag.thresh(score, score, gr)
Sensitivity Specificity
      1          0
>

```

which is not what we wish. The problem lies in the fact that the other inputs are also vectors, leading to ambiguity. The problem may be fixed by using the `Vectorize()` function, which modifies an existing function by designating one or more of its inputs as the *vectorized* input as an option. The original function is evaluated for each element of the vectorized input. A new function is created in this way:

```

> help(Vectorize)
> diag.thresh.vect = Vectorize(diag.thresh, "thresh")
> temp = diag.thresh.vect(score,score,gr)
> dim(temp)
[1] 2 191

```

```
> sens = temp[1,]
> spec = temp[2,]
```

A new function `diag.thresh.vect()` has been created, which evaluates `diag.thresh()` separately for each element of the vector used as the first argument. The results are stored as a 191×2 matrix, each column containing the values of `sens`, `spec` for each element of `score`.

At this point we are ready to plot an *ROC curve*, which is simply a plot of sensitivity (or true positive rate) against 1-specificity (or false positive rate) ('ROC' is an acronym for *receiver operating characteristic*). The script used to draw the plot is given below (Figure 16.2).

```
> auc = roc.area(class, gr)
> pv = wilcox.test(gr ~ class)$p.value
> par(mfrow=c(1,1), cex=1.1, oma = c(1,2,1,1))
> plot(1-spec, sens, xlab="false positive rate (1 - specificity)",
       ylab="true positive rate (sensitivity)", type = "s")
> title("ROC curve for prediction of melanoma
         survival \n based on tumor thickness")
> lines(c(0,0),c(0,1),col=3)
> lines(c(0,1),c(1,1), col=3)
> lines(c(0,1), c(0,1))
> text(.7,.1, paste("AUC = ",signif(auc,3),",
       P = ", signif(pv,3),sep=""))
```

First note the option `type = "s"` in the `plot()` function, which produces a step function type plot, which is appropriate for an ROC curve. Also, the control character “\n” may be used in the plot title to force a line break. In addition, an identity reference line has also been added to the plot, as well as green lines joining the points $(0,0)$, $(0,1)$ and $(1,1)$. The plot also gives two quantities, *AUC* as well as a *p*-value, which we now explain.

First, recall the “perfect” classifier discussed above, with sensitivity and specificity both equal to one. In this case, the ROC curve would coincide with the green lines of Figure 16.2. A highly accurate risk score would produce an ROC curve close to the green lines in some sense.

We next explain *AUC*. This is simply an acronym for *area under curve*. That is, *AUC* is defined as the area under the ROC curve. It may be shown that *AUC* is equal to the probability that a randomly chosen positive subject has a higher risk score than a randomly chosen negative subject. This can be given directly from the data:

$$AUC = \frac{\sum_{i \in \text{-ve}} \sum_{j \in \text{+ve}} I\{score_j > score_i\} + 0.5 \times I\{score_j = score_i\}}{n_- \times n_+} \quad (16.18)$$

where n_- , n_+ are the number of negative and positive outcome subjects. Note that ties are assumed to be resolved randomly, hence the presence of the 0.5 factor in the numerator of (16.18). A function which calculates *AUC* is given below, and was used to calculate the value of *AUC* shown in figure Figure 16.2. This function is not, but could be, vectorized, as for the `diag.thresh.vect()` function given above.

```
> roc.area<-function(x,y) {
+ }
```

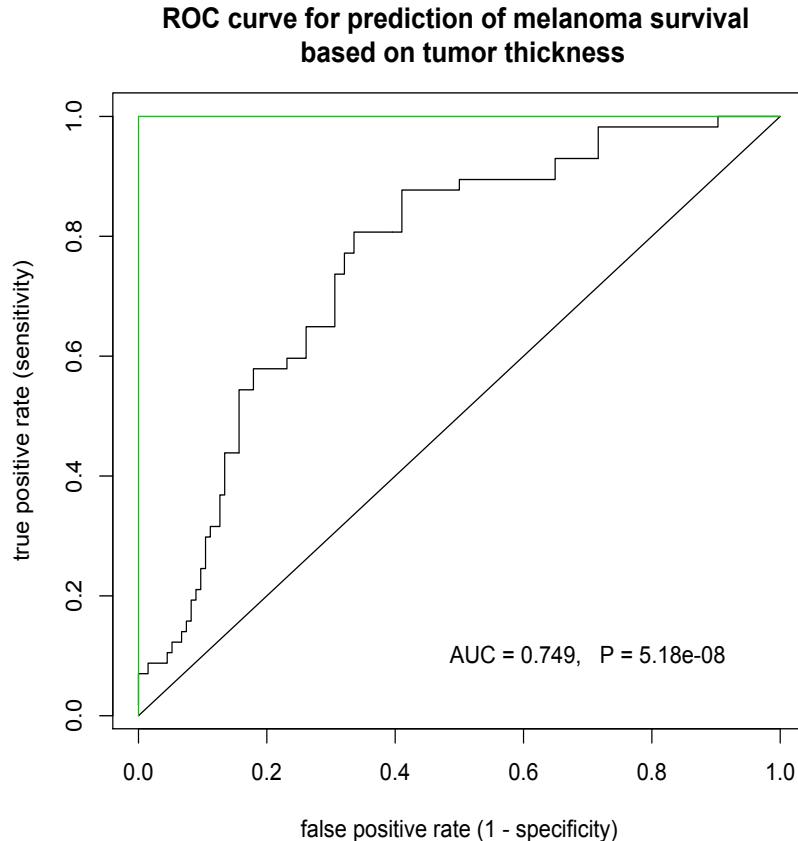


Figure 16.2: ROC curve for prediction of melanoma cancer survival based on melanoma *thickness* variable (tumor thickness in mm). The *AUC* is given, as well as the *p*-value for a Wilcoxon rank sum test for group homogeneity of risk score distributions. The green lines represent a “perfect” classifier, with sensitivity and specificity both equal to one. The diagonal identity line represents a noninformative risk score of *AUC* = 0.5.

```
+ y0 = y[x==0]
+ y1 = y[x==1]
+
+ count<-0
+ for (i in 1:length(y0))
+   {count = count+sum(y1 > y0[i]) + 0.5*sum(y1 == y0[i])}
+ ans = count/(length(y0)*length(y1))
+ return(ans)
+
>
```

Suppose, in contrast to the perfect classifier indicated by the green line in Figure 16.2, that the risk score actually contains no information about the outcome. In this case, a randomly selected positive subject is equally likely to have a higher or lower risk score than a randomly selected

negative subject. In this case we would expect $AUC = 0.5$, and the ROC curve would therefore lie on the identity. For this reason, the identity line is often included in an ROC curve graphic, and the degree to which the ROC curve lies above the identity gives a direct assessment of the predictive value of the risk score (the green lines are usually not given). What would you conclude if the ROC curve lay significantly *below* the identity?

Finally, we explain the *p*-value. It may be shown mathematically that the AUC is equivalent to the Wilcoxon rank sum statistic for a comparison of the risk score between the two outcome groups. This means the Wilcoxon rank sum test is interpretable as a test against the null hypothesis $H_0 : AUC = 0.5$. For this reason the *p*-value may be used to confirm that the risk score is significantly predictive of the outcome in a formal statistical sense.

16.10 Artificial Neural Networks

We can briefly discuss the use of the *artificial neural network* (ANN) in classification and prediction, a method which is currently at the center of much machine learning. What is referred to as a *deep learning network* is really an especially complex variant of the ANN. It is worth noting, then, that the ANN was originally proposed as a computational method in the 1940's (McCulloch and Pitts, 1943). It is, in fact, a function, which maps an input to an output in an entirely deterministic manner, and which depends on a usually large number of parameters (or *weights*). Of course, this is also a perfectly reasonable description of the object

$$g(x_1, \dots, x_p) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p,$$

which defines the linear model. So, from the point of view of our stated goals, the object $g(x_1, \dots, x_p)$ could be replaced with an ANN. Instead of estimating the β_j coefficients using paired observations of response and predictors, we would estimate the weights of the ANN using the same data, in much the same way. The potential for this type of procedure has been long recognized by statistical methodologists. An excellent reference would be Ripley (1994), and the R package **nnet** has been long available for this purpose.

To take just one example, we have seen how linear predictor terms can be the medium by which predictor variables are used to predict survival times, using the Cox proportional hazards model (Section 11.3.3). However, it is possible to replace the linear terms with an ANN, and has been used in exactly this way to predict breast cancer prognosis (Ripley *et al.*, 1998) and time loss due to injury in worker's compensation management (Almudevar, 2006).

16.11 Postscript

The range of classifiers considered in this chapter (with logistic regression in Chapter 10) is relatively limited. We do not discuss a number of widely used methods, such as *classification/regression trees*, *random forests* or *support vector machines*. There are also a number of associated techniques for constructing compound or aggregate classifiers, such as *bagging* or *boosting*. We otherwise offer a brief discussion of artificial neural networks in Section 16.10. Given the prominence of such methods in the field of machine learning, the reader will want to explore this subject further, the texts James *et al.* (2013), Friedman *et al.* (2001) offering ideal points of departure. See also Ripley and Hjort (1996) (www.stats.ox.ac.uk/~ripley/PRbook/).

This lack of comprehensiveness can, however, be justified by the purpose of these notes, which is to establish a statistical foundation for machine learning. More advanced classification techniques require a development which is, as a practical matter, beyond the scope of what is essentially a second course in statistical theory. For this reason, we develop classification in a Bayesian context, and so place more emphasis on foundational Bayesian ideas than on computational algorithms.

However, there is a larger issue. The Bayesian formulation permits a definitive resolution to what would seem to be a rather important question: which classifier minimizes classification error? The answer is the Bayes classifier, without qualification (see Equation (16.12) of Section 16.6). So, why don't we use only Bayes classifiers? In fact, as far as classical statistical theory is concerned, we almost always do. Referring again to Equation (16.12), we can see that the entire basis of the Bayes classifier is the conditional density $f(\dot{x} | y = j)$, which is simply the distribution of the available data on which classification is to be based, conditional on class $y = j$. The Bayes classifier also depends on the prior probabilities π_j , but the predictive ability of a Bayes classifier can be evaluated independently of their values (Section 16.2).

Of course, as a practical matter, the conditional densities $f(\dot{x} | y = j)$ are not known, but can be estimated from data. In fact, in many cases, “learning” is nothing more than this process. And the KNN, LDA and QDA classifiers are nothing more than approximate Bayes classifiers, which work well because they estimate $f(\dot{x} | y = j)$ (LDA and QDA classifiers make the simplifying assumption that the data is normally distributed, while the KNN classifier is essentially a nonparametric estimate of $P(y = j | \dot{x}) \propto f(\dot{x} | y = j)\pi_j$).

A number of the practice problems on classification (Chapter 28) involve comparison of various classification methods. It may seem to the reader who explores them that the competing methods tend not to differ too greatly in their accuracy. This should be expected when all classifiers are simply different forms of approximate Bayes classifiers, which may each be close to achieving the theoretical upper bound on accuracy which holds for *all* classifiers, Bayes or otherwise. For an interesting exception to this tendency, see Problem 28.12.

We note also that while logistic regression (Chapter 10) is often used as a classifier technique, its purpose is to estimate a probability that is allowed to depend on predictor variables. The response variable is modeled as a random outcome, not a class assignment, and this should be kept in mind.

Practice problems on classification can be found in Chapter 28 (problems on logistic regression are collected in Chapter 23). The demonstration software files **CLASSIFICATION-A.R** and **CLASSIFICATION-B.R** contain further extended examples.

Classification and prediction share much of the same theoretical basis. For example, while cross-validation was introduced in Chapter 15 in the context of prediction, it is easily extended to classification simply by selecting the appropriate goodness of fit score (Section 15.8). Support for cross-validation tends to be embedded within R classification functions. KNN classification is implemented in the `knn()` function of the library **class**, which also includes the function `knn.cv()` for cross-validation support. LDA and QDA classification is implemented in library **MASS**, and each includes a cross-validation option.

In Section 16.9.2 on ROC curves, supporting code was written from scratch. However, a good implementation of this and related methods can be found in the package **ROCR**, which is introduced in demonstration software file **CLASSIFICATION-B.R**.

Chapter 17

Unsupervised Learning

In *supervised learning*, data consists of responses are paired with predictor variables. The “learning” consists in discerning the mathematical relationship between response and predictor, which is possible because of the paired structure of the data. A successful application of supervised learning follows when this relationship can be used to predict unseen responses associated with newly sampled predictor variables within a controlled error. These notes are primarily concerned with supervised learning, which includes prediction and classification.

In *unsupervised learning* there is no response (with which to supervise the learning). The goal is to discern structure within the predictor variables. These predictor variables tend to have the same nature as those used in supervised learning, although they no longer function as predictor variables (in these notes, the term used in that case is *features*). The goal is usually to discern *clusters* within these variables, which can be taken to be subsets of the data containing relatively homogenous predictor variable values. A significant challenge in this type of problem is the determination of the number of clusters, which will usually be unknown.

The reader should review the distinction between supervised and unsupervised learning discussed in Section 15.4. As in that section we have an $n \times p$ matrix of data

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ip} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_n \end{bmatrix} = [\mathbf{x}_1 \cdots \mathbf{x}_p]. \quad (17.1)$$

The columns \mathbf{x}_j of \mathbf{X} represent p features, or types of information. The rows \dot{x}_i of \mathbf{X} represent n observations associated with, for example, individual subjects in a study. Then \dot{x}_i contains the specific value of each feature for subject i . Note that we do not have a response variable \mathbf{y} as would be needed for supervised learning. The object is to partition the subjects $\{1, \dots, n\}$ into clusters A_1, \dots, A_m , each subject belonging to exactly one of the m clusters. Sometimes, the features may be clustered, in which case the methodology is the same. It will be important to review Section 15.3 on distances, since many unsupervised learning algorithms are based on an $n \times n$ distance matrix D , in which element d_{ij} is a distance between observations i and j .

17.1 Hierarchical Clustering

We are given an $n \times n$ pairwise distance matrix D for n observations. We can use D to define a *cluster distance*, that is, a way of measuring the distance between two clusters of observations A, B , or alternatively, clusters of indices from $\{1, \dots, n\}$. Three commonly used methods are given below:

- **Single link (connected).** Distance between nearest observations.

$$D(A, B) = \min\{d_{ij} : i \in A, j \in B\}$$

- **Compact.** Distance between furthest objects.

$$D(A, B) = \max\{d_{ij} : i \in A, j \in B\}$$

- **Average.** Average distance.

$$D(A, B) = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} d_{ij}$$

Note that if A and B each consist of a single label i and j then $D(A, B) = d_{ij}$ for each of the above methods.

Hierarchical clustering proceeds using the following steps:

- (1) Define a cluster for each observation.
- (2) Form a cluster from the two observations with the smallest pairwise distance.
- (3) There are now $n - 1$ 'clusters', one with two observations, $n - 2$ with one observation each.
- (4) Successively join the two clusters with the shortest distance D between them.

The resulting cluster is usually represented by a **dendrogram**. This is a tree in which terminal nodes represent the observations and the remaining nodes represent the cluster consolidations. Usually, the vertical distance corresponds to the actual cluster distances. A horizontal cross-section of a dendrogram induces a partition.

Example 17.1. Single link clustering applied to the distance matrix below results in the dendrogram shown in Figure 17.1.

Distance matrix:

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0	1	10	10	10
[2,]	1	0	10	10	10
[3,]	10	10	0	5	5
[4,]	10	10	5	0	5
[5,]	10	10	5	5	0

□

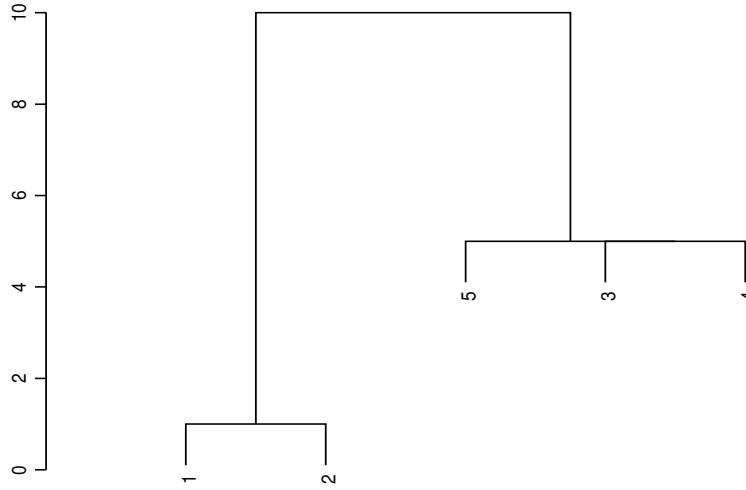


Figure 17.1: Dendogram for Example 17.1

17.2 K-Means Cluster Analysis

K-means clustering is a method in which a fixed number K of clusters is specified, and an attempt is made to find the partition of size K which minimizes some objective function. (We say 'attempt' because many of these algorithms are heuristic).

- We need to define a **centroid** $g(A) = (g_1(A), \dots, g_p(A))$ of a cluster A of features. This may be the component-wise average of the features, but other alternatives are sometimes used.
- In the **sum of squares method** the objective function for partition $\tilde{A} = (A_1, \dots, A_K)$ is the *within cluster sum of squares*

$$SS_{\text{within}} = \sum_{i=1}^K \sum_{j \in A_i} d(\dot{x}_j, g(A_i))^2$$

where d is a distance function on the space of features.

As in linear regression models and ANOVA we can define a *total sum of squares*

$$SS_{\text{total}} = \sum_{i=1}^n d(\dot{x}_i, g(A_0))^2$$

where $g(A_0)$ is the centroid of the entire data set. By analogy, we can define a quantity similar to the coefficient of determination:

$$R^2 = 1 - \frac{SS_{\text{within}}}{SS_{\text{total}}}.$$

Values of R^2 close to one imply that most of the total variation is explainable by the clustering.

17.3 Principal Components Analysis

Mathematically, a *principal components analysis* (PCA) is nothing more than a linear transformation of a feature matrix:

$$\mathbf{Y} = \mathbf{XA}. \quad (17.2)$$

The transformation matrix \mathbf{A} is of course constrained to have certain properties, which in turn force certain properties on \mathbf{Y} . If \mathbf{X} is an $n \times p$ matrix, \mathbf{A} is a $p \times p$ matrix, so that \mathbf{X} and \mathbf{Y} are of the same dimension. In many applications, the intention is that \mathbf{Y} replace \mathbf{X} as the feature matrix. If \mathbf{A} is invertible, which is usually the case, then \mathbf{Y} can be thought of as an alternative representation of \mathbf{X} containing all of the original information. In this case, the j th column of \mathbf{Y} is the j th *principal component*.

However, the column vectors of \mathbf{X} are usually assumed to be standardized by, at least, subtracting the means, and, often, by then dividing by the standard deviation (but see Problem 29.9). As we will see, the main R function for PCA `prcomp` *centers* (subtracts the mean) by default, but does not *scale* (divide by the standard deviation) unless the `by default` option is changed. Note also that R has a second PCA function `princomp` which uses slightly different conventions. The examples and demonstration software here use `prcomp`.

PCA is a quite general method which is widely used in diverse applications. Here, we classify it as an unsupervised learning method because it operates only on a feature matrix, without reference to any response variable, and because it is often used in exploratory analysis to detect clusters. It is worth considering carefully how this is done. PCA is often referred to as a *dimension reduction* method. The idea is that although the matrix \mathbf{X} nominally contains p features, it may be expected that several features may share the same information. This would be the case when there is significant collinearity (Section 8.6), usually, but not always, detectable by pairwise correlations. In such cases, it may be possible to reduce the dimension of the feature matrix, or, put another way, deduce how many dimensions are needed to express the information contained in \mathbf{X} .

This idea is expressed by the terminology of PCA. The *first principal component* is the first column of \mathbf{Y} , and, in a sense to be described below, is the most informative principal component. In fact, the amount of information in the successive principal components decreases by rank, and this information can be quantified. Therefore, it is possible to say, for example, that almost all information is contained in the first 3 principal components, that is, the feature matrix has, approximately, 3 dimensions of data, even if $p > 3$. This idea is illustrated in the next example.

Example 17.2. A pair of psychometric scales measuring *Anxiety* and *Stress* (with scores in the interval [0,1]) are applied to 20 subjects. A scatterplot of the scores is shown in Figure 17.2. As might be expected, the two scales are highly correlated, suggesting that for this sample *Anxiety* and *Stress* are similar constructs, and that the combined information contained in both scores can be expressed in a single dimension. Mathematically, this idea can be expressed by a change of coordinates. The two orthogonal axes in Figure 17.2 show the change in axes resulting from a PCA. These are labeled PC1 and PC2 (principal components 1 and 2). The scores have been centered but not scaled. The origin of the new axes is located at the score means. What is crucial here is that the new axes have been rotated so that most of the variation on the data coincides with the first principal component (PC1 in Figure 17.2). Then the second principal component is orthogonal to the first.

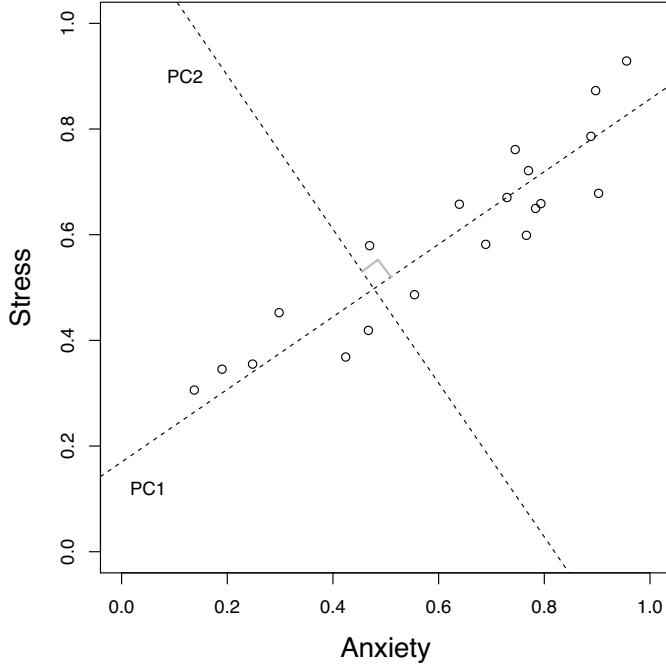


Figure 17.2: *Anxiety* and *Stress* scores, with first two principal components for Example 17.2.

To produce this change of coordinates, the feature matrix \mathbf{X} was first centered to produce $\bar{\mathbf{x}}$ (the column means being subtracted from each column of \mathbf{X}). Then the transformation

$$\mathbf{Y} = \bar{\mathbf{X}}\mathbf{A}$$

is calculated. The original coordinates were $(x_1, x_2) = (\text{Anxiety}, \text{Stress})$, while the transformed coordinates are $(y_1, y_2) = (\text{PC1}, \text{PC2})$. We consider below how the transformation matrix \mathbf{A} is derived.

□

17.3.1 Calculation of principal components

The method of calculating transformation \mathbf{A} has already been described informally, we next make this precise. First, \mathbf{A} is a $p \times p$ matrix. We may then transform \mathbf{X} through matrix multiplication

$$\mathbf{Y} = \mathbf{X}\mathbf{A},$$

so that \mathbf{Y} is another $n \times p$ matrix, and can be interpreted in much the same way. The column vectors of \mathbf{Y} , $\mathbf{y}_1, \dots, \mathbf{y}_p$ are linear combinations of the original feature vectors in \mathbf{X} , with coefficients

given by matrix A :

$$\begin{aligned}\mathbf{y}_1 &= a_{11}\mathbf{x}_1 + \dots + a_{p1}\mathbf{x}_p \\ &\vdots \\ \mathbf{y}_i &= a_{1i}\mathbf{x}_1 + \dots + a_{pi}\mathbf{x}_p \\ &\vdots \\ \mathbf{y}_p &= a_{1p}\mathbf{x}_1 + \dots + a_{pp}\mathbf{x}_p.\end{aligned}$$

The principal components $\mathbf{y}_1, \dots, \mathbf{y}_p$ are constructed using the following steps:

- (i) Normalize each column of \mathbf{X} to have zero mean and unit variance (this step might be omitted for specific reasons).
- (ii) To create the first principal component \mathbf{y}_1 , determine coefficients a_{11}, \dots, a_{p1} which maximize the variance of \mathbf{y}_1 subject to constraint

$$a_{11}^2 + \dots + a_{p1}^2 = 1.$$

- (iii) Successively create the remaining principal components in order. Create the i th principal component \mathbf{y}_i by determining the coefficients a_{1i}, \dots, a_{pi} which maximize the variance of \mathbf{y}_i subject to constraint

$$a_{1i}^2 + \dots + a_{pi}^2 = 1,$$

such that \mathbf{y}_i is orthogonal to previous principal components $\mathbf{y}_1, \dots, \mathbf{y}_{i-1}$.

If dimension reduction is feasible, then the first few principal components will have significantly greater variance than the remaining ones, so that the data set \mathbf{X} can be approximately represented by those.

Note that, assuming \mathbf{X} is linearly independent, then the maximum number of distinct principal components is $\min(n - 1, p)$.

17.3.2 Principal components and spectral decomposition

Recall from Section 15.3.3 that an $m \times m$ matrix Σ is *positive definite* if $\mathbf{u}^T \Sigma \mathbf{u} > 0$ for all nonzero column vectors \mathbf{u} . In this case Σ is invertible, and Σ^{-1} is also positive definite. If instead $\mathbf{u}^T \Sigma \mathbf{u} \geq 0$, then Σ is *positive semidefinite*, and is not invertible unless it is also positive definite.

We restate part of Theorem A.4 of Appendix A:

Theorem 17.1. A real valued square matrix Σ is symmetric if and only if there exists a real orthogonal matrix \mathbf{Q} and real diagonal matrix Λ for which $\Sigma = \mathbf{Q}\Lambda\mathbf{Q}^T$. The diagonal entries of Λ are the *eigenvalues* of Σ , and the columns of \mathbf{Q} are the *eigenvectors* of Σ . Then Σ is positive definite (semidefinite) if and only if the minimum eigenvalue is positive (nonnegative).

The method of *singular-value decomposition* (SVD) is a generalization of this idea to general matrices, in particular, if \mathbf{X} is any real $n \times p$ matrix, there exists $n \times n$ orthogonal matrix \mathbf{U} , $p \times p$ orthogonal matrix \mathbf{V} , and $n \times p$ rectangular diagonal matrix \mathbf{M} , such that

$$\mathbf{X} = \mathbf{U}\mathbf{M}\mathbf{V}^T. \tag{17.3}$$

All matrices in Equation (17.3) are real, however, the SVD can also be applied to complex matrices. Furthermore, the diagonal elements of \mathbf{M} can be chosen so that

$$m_{11} \geq m_{22} \geq \dots \geq m_{ss} \geq 0, \quad s = \min(n, p).$$

Note that a rectangular diagonal matrix \mathbf{M} is any matrix for which elements $m_{ij} = 0$ unless $i = j$.

Then it may be shown that the transformation matrix $\mathbf{A} = \mathbf{V}$ applied to Equation (17.2) yields the principal component matrix \mathbf{Y} , where \mathbf{V} is one of the components of the SVD of \mathbf{X} given in Equation 17.3. This will then be equivalent to the construction method given in Section 17.3.1. Since the inverse of an orthogonal matrix is equal to its transpose, the principal components can also be obtained directly from Equation (17.3)

$$\mathbf{Y} = \mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{M}\mathbf{V}^T\mathbf{V} = \mathbf{U}\mathbf{M}.$$

Thus, PCA is a direct consequence of SVD theory. However, a more direct route is possible. By Theorem 17.1 we may construct an *eigenvalue decomposition* (EVD) of the form

$$\mathbf{X}^T\mathbf{X} = \mathbf{Q}\Lambda\mathbf{Q}^T. \quad (17.4)$$

Then apply the transformation matrix $\mathbf{A} = \mathbf{Q}$, where \mathbf{Q} is a component of the EVD (17.4) of $\mathbf{X}^T\mathbf{X}$, giving

$$\mathbf{Y} = \mathbf{X}\mathbf{Q}.$$

In this case we have

$$\mathbf{Y}^T\mathbf{Y} = \mathbf{Q}^T\mathbf{Q}\Lambda\mathbf{Q}^T\mathbf{Q} = \Lambda. \quad (17.5)$$

Since Λ is diagonal, we may conclude that the column vectors of \mathbf{Y} are orthogonal, as constrained by the construction method of Section 17.3.1. However, we can now see that the eigenvalues of $\mathbf{X}^T\mathbf{X}$ play a very important role. Note that if the matrix \mathbf{X} is centered, then \mathbf{Y} will be as well. Therefore, the eigenvalues are directly proportional to the variances of their associated principal components (the empirical variance is obtained by dividing by n or $n - 1$. We have already interpreted this quantity as the amount of information contained in a principal component. Thus, when the minimum eigenvalue λ_{min} is significantly smaller than the maximum eigenvalue λ_{max} we can conclude that the information in \mathbf{X} can be expressed in strictly less than p dimensions.

Example 17.3. We will continue Example 17.2. The data used to plot Figure 17.2 is displayed below in the R console:

```
> ### Display data
>
> cbind(x.anxiety, x.stress)
      x.anxiety  x.stress
[1,] 0.4670723 0.4188556
[2,] 0.7661788 0.5988583
[3,] 0.6891377 0.5817171
[4,] 0.8971890 0.8727328
[5,] 0.7835571 0.6497151
[6,] 0.4694058 0.5792233
```

```
[7,] 0.5543739 0.4864679
[8,] 0.8882472 0.7862491
[9,] 0.9027775 0.6782207
[10,] 0.7449408 0.7611479
[11,] 0.1376258 0.3060455
[12,] 0.7935094 0.6587902
[13,] 0.4241280 0.3685453
[14,] 0.9557800 0.9287993
[15,] 0.7700636 0.7212453
[16,] 0.6392427 0.6576750
[17,] 0.2980684 0.4525163
[18,] 0.2478763 0.3551216
[19,] 0.7293248 0.6704141
[20,] 0.1901021 0.3455024
>
```

The function `prcomp()` calculates the PCA.

```
>
> ### Calculate PCA components, store in pr.fit object
>
> pr.fit = prcomp(cbind(x.anxiety, x.stress))
> class(pr.fit)
[1] "prcomp"
> is.list(pr.fit)
[1] TRUE
>
```

The object `pr.fit` is a class `prcomp` object, and functions as a list. The *loadings* (that is, the coefficients of transformation matrix \mathbf{A}) are displayed as a *rotation matrix*

```
>
> ### Loadings
>
> pr.fit$rotation
      PC1        PC2
x.anxiety 0.8244803 -0.5658906
x.stress   0.5658906  0.8244803
>
```

This means

$$\mathbf{A} = \begin{bmatrix} 0.8244803 & -0.5658906 \\ 0.5658906 & 0.8244803 \end{bmatrix},$$

equivalently

$$\begin{aligned} \text{PC1} &= 0.8244803 \times \bar{A} + 0.5658906 \times \bar{S} \\ \text{PC2} &= -0.5658906 \times \bar{A} + 0.8244803 \times \bar{S}, \end{aligned}$$

where \bar{A} and \bar{S} are the centered *Anxiety* and *Stress* scores. As suggested by Figure 17.2 the first principal component is a weighted average of the two scores, whereas the second principal component measures the difference between the two scores, representing information that is not common to both.

The standard deviations of the principal components are displayed as the `sdev` element of the `pr.fit` object:

```
>
> ### Standard deviations and variances of principal components
>
> pr.fit$sdev
[1] 0.30513090 0.05828952
> pr.fit$sdev^2
[1] 0.093104866 0.003397668
>
```

The principal components themselves (that is, the \mathbf{Y} matrix of Equation (17.2)) are contained in the `x` element of the `pr.fit` object. Here, we calculate directly the variance of each principal component:

```
>
> apply(pr.fit$x, 2, var)
      PC1          PC2
0.093104866 0.003397668
>
```

As would be expected, these values are equivalent to those obtained by the command `pr.fit$sdev^2` (but note that the alternative PCA function `princomp`) uses the denominator $1/n$ rather than $1/(n - 1)$ when calculating sample variances).

Next, we show how a PCA can be derived directly from the EVD of $\mathbf{X}^T \mathbf{X}$ using the `eigen()` function. We first center the scores:

```
>
> xy = cbind(x.anxiety - mean(x.anxiety), x.stress - mean(x.stress))
>
```

The EVD follows:

```
> ### Calculate EVD
>
> eigen.fit = eigen(t(xy) %*% xy)
> eigen.fit
eigen() decomposition
$values
[1] 1.76899245 0.06455568

$vectors
[,1]      [,2]
```

```
[1,] -0.8244803  0.5658906
[2,] -0.5658906 -0.8244803
>
```

Here, the eigenvalues are $(1.76899245, 0.06455568)$ and the eigenvector matrix is

$$\mathbf{Q} = \begin{bmatrix} -0.8244803 & 0.5658906 \\ -0.5658906 & -0.8244803 \end{bmatrix}.$$

Interestingly, \mathbf{Q} is equal to the rotation matrix produced by application of the `prcomp()` function to the data, multiplies by -1 . This is entirely consistent, and points to the fact that a PCA is not unique, since the properties defining the PCA given in Section 17.3.1 remain unchanged if any column of loadings are uniformly multiplied by -1 . In this sense, the PCA solution offered by function `eigen()` is equivalent to that offered by `prcomp()`.

Finally, as implied by Equation (17.5) we can relate the eigenvalues to the principal component variances after dividing by $n - 1$:

```
> ### Standardize
>
> eigen.fit$values/19
[1] 0.093104866 0.003397668
>
> apply(pr.fit$x,2,var)
      PC1          PC2
0.093104866 0.003397668
>
```

□

17.4 Postscript

One difficulty encountered in unsupervised learning is the lack of a probability model with which to develop formal inference methods. As a consequence, unsupervised learning tends to be most useful as an exploratory tool. A notable exception to this is the Gaussian mixture model (GMM) (Fraley and Raftery, 2002) implemented in the R package `mclust`. For an application of GMMs to the development of a Bayes classifier created by this author, see Pichichero *et al.* (2018).

The implementation of PCA in R has been introduced in Example 17.3 above, and further examples can be found in demonstration software `UNSUPERVISED-LEARNING.R`. Practice problems on PCA can be found in Chapter 29, and also in the context of model selection in Chapter 30.

Hierarchical clustering is implemented in the R function `hclust`. Practice problems can be found in Chapter 29, and a number of its features are introduced in demonstration software `UNSUPERVISED-LEARNING.R`.

Similarly, K -means clustering is implemented in R function `kmeans()` and demonstrated in Chapter 29 in demonstration software `UNSUPERVISED-LEARNING.R`.

A good overview of these methods can be found in James *et al.* (2013), Friedman *et al.* (2001) or Ripley and Hjort (1996) (www.stats.ox.ac.uk/~ripley/PRbook/).

Chapter 18

Score Based Model Selection

So far, model selection has been based on goodness of fit measures such as SSE for least squares regression or deviance for likelihood based inference. These quantities are used to compare full and reduced nested models, or to estimate MSE_{test} for alternative model families using cross-validation.

Such methods are necessary because goodness of fit scores tend to reward model complexity. When comparing full and reduced nested models, the SSE is always smaller, and the likelihood, and therefore the model deviance, is always larger, for the full model. Even when models are not nested the trend is the same.

To some extent these effects are controlled by alternative scores. We have, for example,

$$MSE = \frac{SSE}{n - q},$$

where q is the number of parameters. With addition of a new predictor, both the numerator and denominator decrease, so MSE does not necessarily reward increasing complexity. However, as discussed in Section 15.5 if in a linear regression model the number of predictors $q = n$, the sample size, then $SSE = 0$ and $MSE = 0$. Furthermore, this bias towards zero can be observed well before that point is reached. The same is true, for the same reason, for the adjusted R^2_{adj} (Section 4.2.2)

$$R^2_{adj} = 1 - \frac{SSE/(n - p - 1)}{SSTO/(n - 1)},$$

for a regression model with p predictors.

In this chapter we consider alternative scores. Suppose, for example, that instead of minimizing SSE or model deviance D_{model} , we minimize

$$\Lambda = SSE + \lambda(\theta) \text{ or } \Lambda = D_{model} + \lambda(\theta) \tag{18.1}$$

where $\lambda(\theta)$ is a *complexity penalty* that depends on a parameter θ which represents the model. Possibly, $\lambda(\theta)$ is an increasing function of the number of model parameters q , so that the tendency of model complexity to force SSE to zero will be balanced by the complexity penalty.

There is, in fact, a rich theory behind such score based model selection techniques, and a number of widely used alternatives exist. Scores similar to Λ defined in (18.1) have been derived based on specific mathematical principles. Perhaps the two most widely used are the *Akaike information criterion* (AIC) and the *Bayesian information criterion* (BIC) (the BIC is also known

as the *Schwarz information criterion* (SIC)). The most general definition is given in terms of the likelihood function $l(\hat{\theta}_{MLE})$:

$$\begin{aligned} AIC &= -2 \log(l(\hat{\theta}_{MLE})) + 2q, \\ BIC &= -2 \log(l(\hat{\theta}_{MLE})) + \log(n)q, \end{aligned} \quad (18.2)$$

where n is the number of observations, and q is the number of model parameters (for linear regression, q is the number of predictors plus one for the intercept).

18.1 AIC and BIC for Multiple Linear Regression

There is a technical issue regarding the application of maximum likelihood estimation to linear regression, which relates to the term $-2 \log(l(\hat{\theta}_{MLE}))$ appearing in (18.2). Following equation (9.1) of Example 9.1, for linear regression we have

$$-2 \log(l(\hat{\theta}_{MLE})) = \frac{1}{\sigma^2} SSE + C, \quad (18.3)$$

where C is a constant which does not depend on unknown parameters, and may be set to zero for convenience. However, this was obtained assuming that σ^2 was fixed, or equivalently, did not need to be estimated using the data. We could also include σ^2 in the maximum likelihood calculation, in which case we would have

$$-2 \log(l(\hat{\theta}_{MLE})) = n \log(SSE/n) + C, \quad (18.4)$$

where C is the same type of constant, and may be set to zero. The coefficient estimates $\hat{\beta}$ are the same in both cases (that is, the least squares estimates), and in both cases σ^2 can be estimated in whatever manner is most appropriate. The distinction lies in the manner in which different models are compared.

18.1.1 Model selection algorithms based on predictor subsets

Suppose we have a total of p predictors, which gives full (or *complete*) model

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}_1 + \dots + \beta_p \mathbf{x}_p + \boldsymbol{\epsilon}.$$

The problem is to decide which predictors to retain, assuming that all models contain the intercept. A model M can then be defined as a subset of indices $\{1, \dots, p\}$ indicating the retained predictors. Model $M = \{1, 3\}$ then denotes

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}_1 + \beta_3 \mathbf{x}_3 + \boldsymbol{\epsilon}.$$

In principle, we can calculate

$$AIC_M = \frac{1}{\hat{\sigma}^2} SSE_M + 2k_M \quad (18.5)$$

for any model M , where SSE_M is the SSE for that model, k_M is the number of parameters in the model. We also need an estimate $\hat{\sigma}^2$ of σ^2 which is model independent. This is usually estimated using the MSE of the full model.

For *all subsets* model selection, the value AIC_M is calculated for all models, or equivalently, for all subjects M of index set $\{1, \dots, p\}$. The model selected is the one that minimized AIC_M . Note that this includes the empty set, representing null model

$$\mathbf{y} = \beta_0 + \boldsymbol{\epsilon}.$$

If the null model is selected, the inference is that none of the predictors is significantly related to the response.

This seemingly straightforward method suffers from a serious drawback. Using the product rule of combinatorics, it can be seen that the total number of models is 2^p . We choose a model by deciding independently for each predictor to include or not include it. This represents p decisions of two outcomes. Note that $2^{10} = 1024$, $2^{15} = 32,768$, $2^{20} = 1,048,576$, and so on. Eventually, for large numbers of predictors, all subsets model selection will not be computationally feasible.

A reasonable approach is to order the predictors in decreasing order of importance. Assuming there is some basis on which to do this, we let M_0 be the null model, and let M_k be the model including the k most important predictors. The selected model is then simply

$$M^* = \operatorname{argmin}_{M_k: k=0, \dots, p} AIC_{M_k},$$

the model in the sequence with the minimum AIC .

Sometimes there is some basis for such an ordering. If not *stepwise regression* can be used to empirically generate an ordering. There are a variety of such methods, most of which are based on the following two algorithms.

Algorithm 18.1. Forward stepwise selection:

- (1) Let M_0 be the null model.
- (2) For $k = 1, \dots, p$, add to M_{k-1} the predictor whose addition yields the greatest reduction in SSE . This gives model M_k .
- (3) Select the model from M_0, M_1, \dots, M_p with the minimum AIC .

Algorithm 18.2. Backward stepwise selection:

- (1) Let M_p be the full model.
- (2) For $k = p - 1, \dots, 0$, delete from M_{k+1} the predictor with the largest p -value. This gives model M_{k-1} .
- (3) Select the model from M_0, M_1, \dots, M_p with the minimum AIC .

Algorithms 18.1 or 18.2 may also be used with any other model selection score, such as BIC or R_{adj}^2 .

A few technical notes are needed. Another version of the AIC , based on (18.4), is given by

$$AIC = n \log(SSE/n) + 2k,$$

which does not require an independent estimate of σ^2 .

It is also worth noting the similarity between the AIC and BIC scores. Both are of the form

$$IC = -2 \log(l(\hat{\theta}_{MLE})) + \lambda q$$

where $\lambda = 2$ for AIC and $\lambda = \log(n)$ for BIC . Thus, the preceding remarks concerning AIC also hold for BIC .

There is some flexibility in the definition of q . In regression applications, we get the same minimum AIC or BIC whether q is the number of predictors p or the number of parameters $p + 1$ or $p + 2$ (including the intercept term and the unknown variance σ^2 as appropriate).

A model selection criterion for linear regression known as *Mallow's C_p* has been proposed independently of AIC , but is equivalent to .

Finally, we note that a *finite sample* modification of the AIC has been proposed:

$$AIC_c = AIC + \frac{2k(k+1)}{n-k-1},$$

which may be used when the sample size n is relatively small. Here, it does matter whether k is the number of predictors or the number of parameters, in which case the latter should be used.

18.2 Shrinkage Methods

In the previous section, we have considered model selection methods based on subset selection. There exist a class of methods, known as *shrinkage* or *regularization* methods, which instead constrain the regression coefficients directly while fitting a full model. Suppose we define a model score for a p predictor linear regression model

$$\Lambda = SSE + \lambda \sum_{j=1}^p |\beta_j|^d, \quad (18.6)$$

where $\lambda \geq 0$ is a *tuning parameter* and d is a positive power. Note that the intercept β_0 is not included in the sum in the second term. We also note that such a methodology generally assumes that predictors have been standardized to zero mean and unit variance. Otherwise, summing coefficients in this manner would depend arbitrarily on the predictor's units.

This score resembles the AIC and BIC scores, in the sense that a complexity penalty is added to a standard goodness of fit measure such as SSE . However, instead of considering alternative predictor subsets, the complete model is fit for a fixed λ by minimizing Λ instead of SSE . Then λ is allowed to vary. If we set $\lambda = 0$, we have no complexity penalty, which yields the standard least squares estimates. However, as λ increases, the total magnitude of the coefficients is forced to *shrink* to 0, so that at $\lambda \rightarrow \infty$ we converge to the null model.

The usual approach is to select a relatively fine grid of λ values, $\lambda_1, \dots, \lambda_N$. Then model M_i minimizes Λ for $\lambda = \lambda_i$. Finally, the selected model M^* of the one from the sequence M_1, \dots, M_N with the minimum value of MSE_{test} . If a simpler model is needed, the model for which the error is within 1 standard error of the cross-validated estimate of the minimum may be used.

The choice of power d is of some consequence. When $d = 2$ the method is commonly known as *ridge regression*. When $d = 1$, the method is commonly known as LASSO (least absolute shrinkage and selection operator). One of the problems with ridge regression is that all predictors remain in the model. If a predictor does not add significantly to the model, its coefficient will be very small but not zero.

On the other hand, one of the attractive features of LASSO is that it forces some coefficients to 0, and these tend to be the ones with smaller magnitudes in a ridge regression model.

18.3 Postscript

In the basic theory of prediction and classification, much is made of the term “over-fitting”. Suppose in some model we had as many predictors x_j as we had observations y_i . As long as some fairly general conditions were met, we would be able to devise some function $g(x_1, \dots, x_p)$ so that the available data was fit perfectly, that is,

$$y = g(x_1, \dots, x_p).$$

But an error term ϵ either exists, or it does not. If it does (meaning that its variance is greater than zero) we cannot wish it away. The practical consequence of this is that if our predictor was applied to a new set of data (which is presumably the intention) it would no longer fit perfectly (there is no way it could).

Thus, the topic of model selection is a very important one, and possesses some very deep theoretical implications. Despite this, our treatment here is based on a relatively simple idea, expressed in Equation (18.1), that model fitting criterion, such as the *SSE* or the maximum likelihood score can be usefully supplemented with a model complexity term to control overfitting. The purpose of this is to allow comparisons of models with varying levels of complexity, in support of model selection.

Of course, the next problem is to decide which penalty term to use, and the fact that there is more than one in common use suggests that the solution to this problem is not straightforward. It helps, in this regard, to clarify the goal of model selection. And although it might not seem obvious at this point, we have encountered in these notes two very distinct (and possibly incompatible) goals. To take one example, the purpose of the *F*-test for comparing full versus reduced linear models introduced in Section 4.2.2 is *model identification*, simply, the identification of the correct model. However, the purpose of cross-validation is to reduce prediction error, which is a different goal. Obviously, model identification helps, but cross-validation also controls for the greater variation of more complex models, which contributes to prediction error. This is expressed in the bias-variance tradeoff (Section 15.7), in which an increase in bias represents a compromise of the goal of model identification for the sake of prediction accuracy. Problem 30.20 is intended to clarify this issue.

As it happens, the *AIC* score may be thought of as a theoretical cross-validation. The original theory is quite technical, but rewards patient reading (Akaike, 1973a,b, 1974). We will at least note that the theoretical model described in those foundational papers considers response variables separated into training and test data, the *AIC* being based on their predicted distributional properties (something similar is done in Problem 30.20). On the other hand, the *BIC* was developed as an approximate Bayesian model, and is intended for model identification (Schwarz *et al.*, 1978), as suggested in the title of the paper in which the method was originally proposed (“Estimating the dimension of a model”).

The use of *AIC* and *BIC* scores in model selection is introduced in the demonstration software file `MODEL-SELECTION-AIC.R`, while Chapter 30 offers additional exercises. We make use of the R library `MuMIn`, which offers general model selection utilities (which is introduced in `MODEL-SELECTION-AIC.R`).

We also discussed *shrinkage methods* in this chapter. Again, the method is easy to describe (Equation (18.6)), but their complete understanding would require considerable study. We consider specifically *ridge regression* and *LASSO*. The latter method has gained especial prominence

(Tibshirani, 1996; Tibshirani *et al.*, 2015) due to its utility in variable selection, particularly its remarkable ability to eliminate variables from linearly models by setting their coefficient estimates to zero (that is, $\hat{\beta}_j = 0$), using mathematically objective criterion. However, these methods differ from the *AIC* or *BIC* method in that the choice of model is ultimately to be decided by cross-validation, so that the penalty term is not ultimately based on some theoretical principle.

Shrinkage methods are implemented in R by the `glmnet` library. The main functions are `glmnet()` and `cv.glmnet()`, the latter supporting cross-validation. The demonstration software file `MODEL-SELECTION-LASSO.R` gives a comprehensive introduction to their use, while Chapter 30 contains a number of practice problems. It is interesting to note that a closed form solution for ridge regression coefficients can be obtained after some matrix algebra, which gives some insight into how these methods work (Problem 30.21).

Information theory has provided much of the theoretical foundation for model selection, a fact which is not always apparent. In fact, this is true of both the *AIC* and *BIC* scores, which is why a familiarity with the foundational papers might be recommended. The *minimum description length principle* (MDL) provides a theory of model selection more explicitly based on information theory, in the context of coding theory (the text Cover and Thomas (2012) can be highly recommended). The MDL method is also a score based method, in which the score is the estimated length of a code used to represent the data in compressed form (as in a `zip` file). The data compression makes use of a *model*, which allows more efficient compression by capturing any structure the data may contain (this is, after all, what we expect the models considered in these notes to do). The catch is that the model must also be coded, and the more complex the model, the greater its contribution to the code length. The MDL method then selects the model offering the optimal tradeoff between model complexity and data compression, based on the criterion of code length. The MDL method was first formulated by Jorma Rissanen (Rissanen, 1978). See also Rissanen (2007) and Grünwald (2007). In Almudevar (2016), this author contributes a discussion of information theoretic methods to Bayesian networks (Chapter 20).

Chapter 19

Basis Functions and Predictor Spaces

The term “linear” in “linear regression” is often understood to imply a linear functional relationship between the mean μ_y of a response variable y and one or more predictor variables x_1, \dots, x_p . Put another way, we assume the existence of some function

$$\mu_y = g(x_1, \dots, x_p) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p,$$

which then leads to the problem of estimating the unknown coefficients $\beta_0, \beta_1, \dots, \beta_p$. For generalized linear models, the considerable advantages of this formulation are preserved by allowing linearity to hold for a mapping of μ_y :

$$h(\mu_y) = g(x_1, \dots, x_p) = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p,$$

for example, $h(\mu_y) = \log(\mu_y/(1 - \mu_y))$ for logistic regression (Chapter 10).

However, we have seen in polynomial regression (Section 5.7) that a nonlinear relationship between μ_y and a predictor x can be modeled in the context of linear regression. In particular, the same inference methods are available, as long as we are willing to increase the number of coefficients β_i to be estimated.

In this chapter, we explore this technique further, using the concept of *basis functions*. Suppose we have, as for simple linear regression, a response y and single predictor x , and paired observations (x_i, y_i) , $i = 1, \dots, n$. We assume the predictor x is confined to an interval $I_x = [a, b]$. We then have model

$$y_i = g(x_i) + \epsilon_i,$$

where the values ϵ_i are an *iid* sample from $N(0, \sigma^2)$. For simple linear regression we have

$$\mu_y = g(x) = \beta_0 + \beta_1 x.$$

But we may not be able to assume a linear relationship between x and μ_y . On the other hand, we may wish to retain the quite powerful fitting and inference methodology afforded by the linear regression model.

Basis function methods provide a quite powerful approach in this situation. Suppose we define $p \geq 1$ basis functions $b_1(x), \dots, b_p(x)$, for $x \in I_x$, then set

$$g(x) = \beta_0 + \beta_1 b_1(x) + \dots + \beta_p b_p(x). \tag{19.1}$$

In a sense, we are converting a simple regression model to a multiple regression model by defining a new $n \times (p + 1)$ design matrix

$$\mathbf{X} = \begin{bmatrix} 1 & b_1(x_1) & \cdots & b_p(x_1) \\ 1 & b_1(x_2) & \cdots & b_p(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_1(x_n) & \cdots & b_p(x_n) \end{bmatrix} \quad (19.2)$$

which then defines the new regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

It is important to note that an intercept term β_0 was included in the response function (19.1), but was not created by the basis functions b_1, \dots, b_p . This could be regarded as a matter of convention. We could have included the intercept term as a basis function itself, say $b_0(x) \equiv 1$, then (19.1) could be equivalently written

$$g(x) = \sum_{j=0}^p \beta_j b_j(x). \quad (19.3)$$

In one sense, basis functions do not change the overall theory of linear models. However, basis function models vary considerably in complexity (or model degrees of freedom). For this reason, practice problems on model selection and basis function methods are collected in the single Chapter 30, with many problems involving both techniques.

19.1 Transformation Equivalence of Basis Functions

A review of Chapter 7 is highly recommended at this point. Recall that two $n \times p$ design matrices \mathbf{X}, \mathbf{X}' are equivalent from the point of view of linear least squares regression if there exists invertible $p \times p$ matrix \mathbf{A} such that $\mathbf{X}' = \mathbf{XA}$ (Theorem 7.1). Much of our discussion below relies on this idea.

Suppose we consider two possible sets of basis functions $\mathcal{B} = (b_1(x), \dots, b_p(x))$ and $\mathcal{B}' = (b'_1(x), \dots, b'_q(x))$ defined on I_x . For the moment, we do not assume p and q are equal. How can we characterize the notion of transformation equivalence for basis functions? A straightforward approach would be to assume we have predictor observations x_1, \dots, x_n , then construct a design matrix such as that given in Equation (19.2). Then the theory of Chapter 7 (Theorem 7.1 in particular) may be applied directly.

However, there is nothing to be gained by allowing the analysis to depend on hypothetical predictor observations, and the study of basis functions has greater clarity without reference to them. Recall that in Chapter 7, although the vector spaces discussed were subsets of some Euclidean space \mathbb{R}^m , this was not specifically stated in their formal definition (Definition 7.2). A vector space \mathcal{V} may also be a subset of functions $g : I_x \rightarrow \mathbb{R}$. In fact, it is easily verified that the set of all functions

$$g(x) = \sum_{j=1}^p \beta_j b_j(x), \quad \beta_j \in \mathbb{R}, \quad j = 1, \dots, p, \quad (19.4)$$

denoted \mathcal{V} , is a vector space. Note that \mathcal{V} contains the zero function $g(x) = \vec{0} \equiv 0$ for all $x \in I_x$, since we may set $\beta_j = 0$ for all j .

Note, at this point, that we have so far created vector spaces as the span of a set of basis functions, but we have not yet used the term *basis* as defined in Definition 7.2. Formally, a set of basis functions is a *basis* for the vector space spanned by those vectors only if those functions are linearly independent, that is,

$$0 = \sum_{j=1}^p \beta_j b_j(x), \quad x \in I_x \text{ implies } \beta_j = 0 \text{ for all } j.$$

Equivalently, no basis function is a linear combination of the remaining basis functions.

Thus, we can equate the vector space \mathcal{V} generated by (19.4) with the span of a set of vectors, given in Equation (7.1). In turn, this allows us to define the equivalence of two sets of basis functions, that is, they are equivalent if the vector spaces generated by the respective “spans” (via (19.4)) are equal.

As in the linear transformations of design matrices $\mathbf{X}' = \mathbf{XA}$ described in Chapter 7 equivalence of sets of basis functions can be characterized by essentially the same linear transformations, except that the basis functions play the role of the column vectors of the predictor matrix. Suppose the sets of basis functions are related by the linear combinations:

$$\begin{aligned} b'_1(x) &= \sum_{k=1}^p a_{k1} b_k(x) \\ b'_2(x) &= \sum_{k=1}^p a_{k2} b_k(x) \\ &\vdots \\ b'_q(x) &= \sum_{k=1}^p a_{kq} b_k(x). \end{aligned}$$

If the preceding coefficients a_{ij} are also the coefficients defining $p \times q$ matrix \mathbf{A} , the transformation can also be expressed in matrix notation as

$$[b'_1(x) \cdots b'_q(x)] = [b_1(x) \cdots b_p(x)] \mathbf{A}, \quad (19.5)$$

each side interpretable as a $1 \times q$ matrix, with equality holding for all $x \in I_x$.

We next present an equivalence theorem for basis functions analogous to Theorem 7.1.

Theorem 19.1. Suppose we consider two possible sets of basis functions $b_1(x), \dots, b_p(x)$ and $b'_1(x), \dots, b'_q(x)$ defined on I_x (q and p need not be equal).

- (i) Suppose each function $b_j(x)$ is a linear combination of the basis functions $b'_1(x), \dots, b'_q(x)$, and each function $b'_j(x)$ is a linear combination of the basis functions $b_1(x), \dots, b_p(x)$. Then the two sets of basis functions are equivalent if and only if this condition holds.
- (ii) Suppose $p = q$, and transformation (19.5) holds for some invertible transformation matrix \mathbf{A} . Then the two sets of basis functions are equivalent.

Proof. We consider the two claims in turn.

- (i) Let $\mathcal{V}, \mathcal{V}'$ be the vector spaces generated by the respective sets of basis functions. Suppose $g(x) \in \mathcal{V}$. Then $g(x)$ is a linear combination of basis functions $b_1(x), \dots, b_p(x)$. But each basis function $b_j(x)$ is, by hypothesis, a linear combination of basis functions $b'_1(x), \dots, b'_q(x)$. This in turn implies that $g(x)$ is also a linear combination of basis functions $b'_1(x), \dots, b'_q(x)$. Therefore $g(x) \in \mathcal{V}'$, and we conclude $\mathcal{V} \subset \mathcal{V}'$. By an identical argument $\mathcal{V}' \subset \mathcal{V}$, and therefore $\mathcal{V} = \mathcal{V}'$, so that the two sets of basis functions are equivalent.

Next, suppose the condition does not hold. Then there exists a basis function, say b_j , which is not a linear combination of basis functions $b'_1(x), \dots, b'_q(x)$. Then $b_j(x) \in \mathcal{V}$, but $b_j(x) \notin \mathcal{V}'$, therefore $\mathcal{V} \neq \mathcal{V}'$, so that the two sets of basis functions are not equivalent.

- (ii) If \mathbf{A} is invertible, then the hypothesis of Part (i) holds, therefore the two sets of basis functions are equivalent.

□

Theorem 19.1 does not require that the sets of basis functions satisfy the formal definition of a basis. However, for linear regression analysis, the column vectors of design matrix \mathbf{X} are usually assumed to be linearly independent. We will impose the same requirement on a set of basis functions, in which case they form a basis for the vector space which they span. This is necessary if the design matrix used in fitting the model is to possess linearly independent columns. The set of predictor observations must also be large enough ($n \geq p$), and must also possess sufficient variability. This means that no column of the design matrix \mathbf{X} generated by (19.2) can be entirely zero. However, if this occurs, it usually means that the basis being used is not appropriate for the particular data set, and can be modified accordingly.

The next theorem verifies that basis equivalence implies equivalence of the least squares linear regression model, in the same sense as Theorem 7.1.

Theorem 19.2. Suppose we have model

$$y_i = g(x_i) + \epsilon_i,$$

with paired observations (x_i, y_i) , $i = 1, \dots, n$, $x_i \in I_x$. Suppose we consider two possible bases $\mathcal{B} = (b_1(x), \dots, b_p(x))$ and $\mathcal{B}' = (b'_1(x), \dots, b'_p(x))$ defined on I_x , and transformation (19.5) holds for some invertible transformation matrix \mathbf{A} . Assume also that the respective design matrices \mathbf{X} , \mathbf{X}' defined by (19.2) are linearly independent. This leads to proposed models

$$\begin{aligned} y_i &= \sum_{j=1}^p \beta_j b_j(x_i) + \epsilon_i, \text{ and} \\ y_i &= \sum_{j=1}^p \beta'_j b'_j(x_i) + \epsilon_i. \end{aligned}$$

Then the respective fitted value vectors satisfy equality $\hat{\mathbf{y}} = \hat{\mathbf{y}}'$. Furthermore, the least squares estimates of the coefficients $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ satisfy

$$\hat{\boldsymbol{\beta}}' = \mathbf{A}^{-1} \hat{\boldsymbol{\beta}}.$$

Proof. The first step is to construct respective $n \times p$ design matrices \mathbf{X} , \mathbf{X}' as in Equation (19.2). Then the transformation $\mathbf{X}' = \mathbf{XA}$ follows from transformation (19.5). Then Theorem 7.1 can be applied, which completes the proof. \square

Example 19.1. Consider basis functions

$$\begin{aligned} b_1(x) &= 1, \\ b_2(x) &= x, \\ b_3(x) &= x^2 \end{aligned}$$

and a second set of basis functions from the centering transformation

$$\begin{aligned} b'_1(x) &= 1, \\ b'_2(x) &= x - \bar{x}, \\ b'_3(x) &= (x - \bar{x})^2, \end{aligned}$$

where \bar{x} is the sample mean of the available predictor variables. Clearly, the basis functions $\mathcal{B} = (b_1, b_2, b_3)$ are linearly independent, and span a vector space \mathcal{V} of dimension 3. Thus, if \mathcal{B} is equivalent to the set of basis functions $\mathcal{B}' = (b'_1, b'_2, b'_3)$, we can conclude that \mathcal{B}' is also a basis for \mathcal{V} , since it contains exactly 3 basis functions. Expanding the polynomials we have

$$\begin{aligned} b'_1(x) &= b_1(x), \\ b'_2(x) &= x - \bar{x} = b_2(x) - \bar{x}b_1(x), \\ b'_3(x) &= x^2 - 2\bar{x}x + \bar{x}^2 = b_3(x) - 2\bar{x}b_2(x) + \bar{x}^2b_1(x). \end{aligned}$$

We can see that the two sets of basis functions obey transformation (19.5), with

$$\mathbf{A} = \begin{bmatrix} 1 & -\bar{x} & \bar{x}^2 \\ 0 & 1 & -2\bar{x} \\ 0 & 0 & 1 \end{bmatrix}.$$

This matrix is invertible, which implies, by Theorem 19.1 (ii), that the two sets of basis functions are equivalent, and both form bases for a common vector space \mathcal{V} . \square

Example 19.2. [Triangular Transformations] One way to construct a transformation that preserves equivalence is to consider what we may refer to as a *triangular transformation*. These are transformations for which, in the notation of Theorem 19.1, $p = q$ and \mathbf{A} is upper or lower triangular, with nonzero diagonal entries. This guarantees that \mathbf{A} is invertible. This is easily done by (i) allowing b'_k to depend only on b_j for which $j \geq k$; and (ii) ensuring that b'_k does depend on b_k . We would therefore have

$$b'_k(x) = \sum_{j=k}^p a_{jk} b_j(x),$$

with $a_{jj} \neq 0$ (any other coefficient may be zero). Then \mathbf{A} will be lower triangular and invertible.

Alternatively, we may (i) allow b'_k to depend only on b_j for which $j \leq k$; and (ii) ensure that b'_k does depend on b_k . We would then have

$$b'_k(x) = \sum_{j=1}^k a_{jk} b_j(x),$$

and \mathbf{A} will be upper triangular and invertible. \square

19.2 Polynomial Regression

Polynomial regression (Section 5.7) is an example of the basis function technique. The simplest way of expressing polynomial regression is to simply set basis functions to be increasing powers of x :

$$b_j(x) = x^j, \quad j = 0, 1, \dots, p, \quad (19.6)$$

to yield an order p response function (again, whether or not to regard $b_0(x) \equiv 1$ as a basis function is a matter of convention). As polynomials of strictly increasing degree, they must be linearly independent, therefore $\mathcal{B} = (b_0, b_1, \dots, b_p)$ forms a basis for the vector space \mathcal{P}_p of all polynomials of degree no greater than p .

If we include the intercept we have design matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^p \end{bmatrix}$$

Then consider the linear transformation

$$\mathbf{X}' = \mathbf{XA} = \begin{bmatrix} z_{10} & z_{11} & z_{12} & \dots & z_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{n0} & z_{n1} & z_{n2} & \dots & z_{np} \end{bmatrix} \quad (19.7)$$

where \mathbf{A} is a $(p+1) \times (p+1)$ matrix of coefficients. Note that the first column of \mathbf{X}' has been relabelled $j=0$. The coefficient matrix \mathbf{A} is usually upper triangular, so the transformed predictor variables become

$$\begin{aligned} z_{i0} &= a_{00} \\ z_{i1} &= a_{01} + a_{11}x_i \\ z_{i2} &= a_{02} + a_{12}x_i + a_{22}x_i^2 \\ &\vdots \\ z_{ip} &= a_{0p} + a_{1p}x_i + a_{2p}x_i^2 + \dots + a_{pp}x_i^p \end{aligned} \quad (19.8)$$

noting that the first row and column of \mathbf{A} are here labeled $i=j=0$. The model is now

$$y_i = \beta'_0 z_{i0} + \beta'_1 z_{i1} + \beta'_2 z_{i2} + \dots + \beta'_{ip} z_{ip}. \quad (19.9)$$

Thus, if all diagonal elements of \mathbf{A} are nonzero, this defines a transformation either of the design matrix \mathbf{X} (Equation (19.7)) or of the basis \mathcal{B} to an alternative basis \mathcal{B}' for \mathcal{P}_p defined by Equation (19.5).

Thus, by Theorem 19.2, polynomial regression need not be based on the original basis \mathcal{B} . Any transformation based on an invertible transformation matrix \mathbf{A} will yield exactly the same fitted model. In practice, \mathbf{A} is often chosen so that the transformed design matrix \mathbf{X}' possesses orthonormal column vectors (see Section 7.4). It is worth spending some time understanding how this is done in R.

19.2.1 Polynomial regression in R

One of the strengths of R is a comprehensive but unified language for the definition of linear regression formula (Chapter 5). This is especially true for linear models defined by basis functions. Recall that if within the R environment there exists response vector y and predictor vectors x_1 and x_2 , the least squares estimate of the model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

is obtained by including the formula object

```
y ~ x1 + x2
```

as an argument in the `lm()` function:

```
lm(y ~ x1 + x2)
```

Note that the intercept term is included by default, and can be removed by including the term `-1` in the formula object.

Suppose instead we have a single predictor x , and which to fit a polynomial regression model of degree 3. The most direct approach would be to create the terms separately by applying the basis transformations directly to x :

```
x.squared = x^2
x.cubed = x^3
lm(y ~ x + x.squared + x.cubed)
```

A more compact alternative would be to use the `AsIs` function `I()` (which inhibits interpretation or conversion of objects):

```
lm(y ~ x + I(x^2) + I(x^3))
```

Both preceding methods will fit a basis function model using basis (19.6). In particular, the coefficient estimates $\hat{\beta}$ will be identical.

However, R also supports functions which construct design matrices from function bases of the form (19.2) without the need to construct each column vector explicitly. Furthermore, these design matrices may be placed directly in formula objects, which may also contain additional predictor terms.

For polynomial regression, this function is `poly()`. The arguments and options important to us at this point are `x` (the single predictor variable), `degree` (the degree of the polynomial regression) and `raw` (basis to be used). For option `raw = TRUE`, the basis functions used will be those given in (19.6). If `raw = FALSE` (the default option) a basis is selected which will yield orthonormal column vectors in the design matrix (19.2). However, it is important to understand that while `poly()`, under either option, does not include the intercept term in the design matrix, when the orthonormalization option `raw = FALSE` is selected, all column vectors will be orthogonal to the intercept term.

The three R commands:

```
> lm(y ~ x + x.squared + x.cubed)
> lm(y ~ x + I(x^2) + I(x^3))
> lm(y ~ poly(x, 3, raw = TRUE))
```

will produce the same regression fit, as well as the same coefficient estimates $\hat{\beta}_j$. The R command:

```
> lm(y ~ poly(x, 3, raw = FALSE))
```

will produce the same regression fit, but because the basis is a transformation of \mathcal{B}_{raw} the coefficient estimates $\hat{\beta}'_j$ will differ from $\hat{\beta}_j$. They can, however, be related by Equation (7.10) (see below).

In order to understand the orthonormalization process used by R, it is helpful to understand it as a linear transformation of the “raw” polynomial basis \mathcal{B}_{raw} of Equation (19.6), expressed either as transformation (19.5) to basis \mathcal{B}_{ortho} , or transformation $\mathbf{X}' = \mathbf{XA}$, where \mathbf{X} is given by (19.2), using basis \mathcal{B}_{raw} . It must be remembered that although the intercept column is not included in the design matrix output by `poly()`, it is an indispensable component of the orthonormalization operation, and so we must assume both bases \mathcal{B}_{raw} and \mathcal{B}_{ortho} include the constant intercept function (for example, the basis function $b_0(x) \equiv 1$ of the basis defined in (19.6)).

Finally, note that we have so far only considered univariate polynomial regression. If more than one predictor is to be transformed, x can be a matrix with one column for each predictor. In this case, the basis functions can be defined as the terms of a suitable multivariate polynomial.

19.2.2 Demonstration of R function `poly()`

Consider as an example the univariate predictor variable defined in the R environment by

```
x = seq(-1, 1, 0.1)
n = length(x)
```

We will produce design matrices for polynomial regression of degree 3 using both the raw basis \mathcal{B}_{raw} (`raw = TRUE`), and the orthogonal option (the default `raw = FALSE`). This may be done using the following code:

```
>
> ### Create mathematical notation for basis function plots
>
> ex0 = expression(italic(x))
> exb1 = expression(italic(b)[1])
> exb2 = expression(italic(b)[2])
> exb3 = expression(italic(b)[3])
> exbp1 = expression({italic(b)^{scriptscriptstyle("//")}}[1])
> exbp2 = expression({italic(b)^{scriptscriptstyle("//")}}[2])
> exbp3 = expression({italic(b)^{scriptscriptstyle("//")}}[3])
>
> ### Create the design matrices
>
> x.ortho = poly(x, degree = 3)
> x.raw = poly(x, degree = 3, raw = TRUE)
>
> ### Display some of the matrix
>
> head(x.ortho)
```

```

1           2           3
[1,] -0.3603750  0.42285541 -0.43329786
[2,] -0.3243375  0.29599879 -0.17331914
[3,] -0.2883000  0.18249549  0.01824412
[4,] -0.2522625  0.08234553  0.14899365
[5,] -0.2162250 -0.00445111  0.22653116
[6,] -0.1801875 -0.07789442  0.25845837
>
> ### Plot the basis functions
>
> par(mfrow=c(1,2),pty='s')
> matplot(x,x.raw,type='l',lwd=2,xlab=ex0)
> legend('bottomright',legend=c(exb1,exb2,exb3),col=1:3,lty=1:3,cex=1,lwd=2)
> abline(h=0,col='lightgray')
> matplot(x,x.ortho,type='l',lwd=2,xlab=ex0)
> legend('bottomright',legend=c(exbp1,exbp2,exbp3),col=1:3,lty=1:3,cex=1, bg=NA,lwd=2)
> abline(h=0,col='lightgray')

```

The design matrices (excluding intercept) are stored in objects `x.ortho` (orthogonal) and `x.raw` (raw). The command `head(x.ortho)` displays the first six rows of the design matrix `x.ortho`. The vector space \mathcal{P}_3 spanned by bases \mathcal{B}_{raw} and $\mathcal{B}_{\text{ortho}}$ has 4 dimensions, so, excluding the intercept column, `x.ortho` is interpretable as a matrix with 3 columns, as would be expected.

The code above also plots the basis functions of \mathcal{B}_{raw} and $\mathcal{B}_{\text{ortho}}$, shown in Figure 19.1. The “raw” basis functions $b_1(x) = x$, $b_2(x) = x^2$, $b_3(x) = x^3$ are easily recognized in the left plot. The basis functions of $\mathcal{B}_{\text{ortho}}$ appear to conform to the upper triangular transformation of \mathcal{B}_{raw} given in (19.8), so that $b'_j(x)$ is a polynomial of degree j .

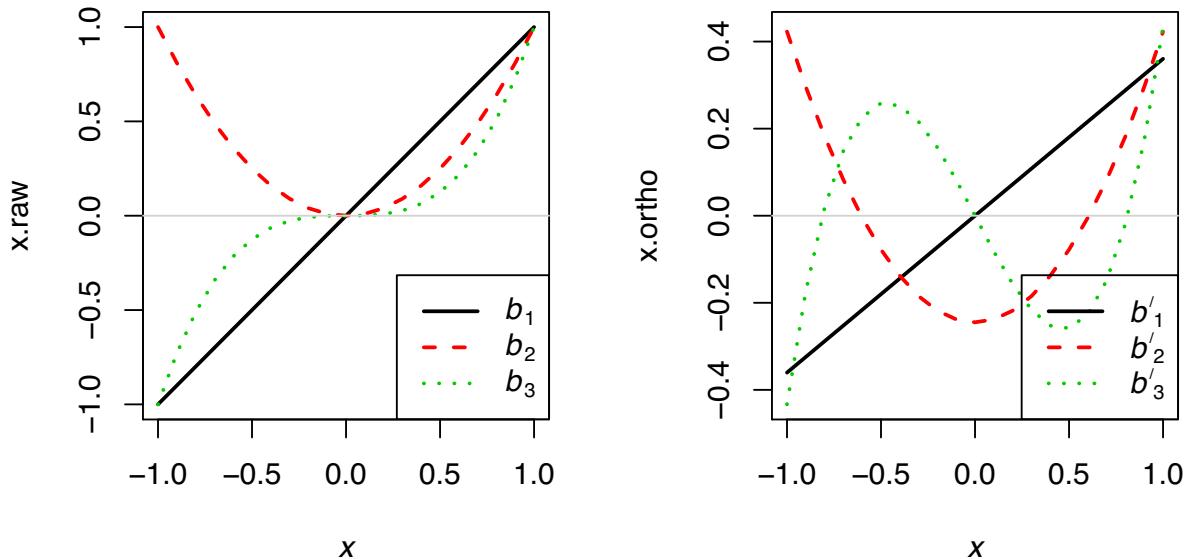


Figure 19.1: Plot of basis functions described in Section 19.2.2.

We may then deduce the actual transformation matrix \mathbf{A} using the method of Section 7.2.1 (extracting \mathbf{A} directly from `poly()` is an interesting technical exercise, here we make the point that an understanding of the underlying mathematics also suffices for this task). This allows us to fit the model using the orthogonal basis \mathcal{B}_{ortho} , calculating least squares coefficients $\hat{\beta}'$ using that basis, then to express the response function in simple polynomial form

$$g(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \hat{\beta}_3 x^3$$

using transformation $\hat{\beta} = \mathbf{A}\hat{\beta}'$.

Of course, we must add the intercept term for each design matrix. For the design matrix \mathbf{X} associated with basis \mathcal{B}_{raw} , this is a column of 1's. However, for the design matrix \mathbf{X} associated with basis \mathcal{B}_{ortho} each column is a unit vector, so the intercept column contains the entry $1/\sqrt{n}$. Then, since \mathbf{X}' has orthonormal column vectors, we may use Equation (7.10) to obtain

$$\mathbf{A} = ([\mathbf{X}']^T \mathbf{X})^{-1}.$$

We next perform this calculation:

```
>
> ### Add the appropriate intercept column to each design matrix
>
> x.ortho.intcpt = cbind(1/sqrt(n),x.ortho)
> x.raw.intcpt = cbind(1, x.raw)
>
> ### calculate transformation matrix A
>
> solve(t(x.ortho.intcpt)%*%x.raw.intcpt)
      1           2           3
2.182179e-01 -4.141667e-17 -2.448110e-01  4.022340e-17
1 4.612709e-17  3.603750e-01  2.347001e-16 -8.336549e-01
2 -1.132291e-16  5.342622e-17  6.676664e-01 -7.663380e-17
3 -7.673633e-17 -5.069035e-17 -4.304390e-16  1.266953e+00
>
> ### Round off
>
> round(solve(t(x.ortho.intcpt)%*%x.raw.intcpt),4)
      1           2           3
0.2182  0.0000 -0.2448  0.0000
1 0.0000  0.3604  0.0000 -0.8337
2 0.0000  0.0000  0.6677  0.0000
3 0.0000  0.0000  0.0000  1.2670
```

Matrix inversion is often a challenging numerical computation, and values of a matrix inverse which are in reality equal to zero are often given as very small, but nonzero, values (for example, here we have $a_{0,3} = 4.022340e-17$). It may aid clarity to round off to, say, 4 significant digits. If we do so, it is clear that \mathbf{A} is essentially an upper triangular matrix.

Finally, we may perform the orthogonalization itself using QR decomposition. From Section 7.4.2, we may write

$$\mathbf{X} = \mathbf{Q}\mathbf{R}$$

where \mathbf{Q} is an $n \times p$ matrix of orthonormal columns, and \mathbf{R} is a $p \times p$ matrix, which can be constructed to be upper triangular. We then equate $\mathbf{X}' = \mathbf{Q}$ and $\mathbf{A} = \mathbf{R}^{-1}$. The following code performs the calculations. The R function `qr()` evaluates a QR decomposition. Here, it is applied to the design matrix `x.raw.intcpt` associated with the basis \mathcal{B}_{raw} . Then functions `qr.Q()` and `qr.R()` are used to extract the components of the decomposition from the class `qr` object output by function `qr()`.

```

>
> ### QR decomposition
>
> qr.raw = qr(x.raw.intcpt)
>
> ### extract Q
>
> head(qr.Q(qr.raw))
      [,1]      [,2]      [,3]      [,4]
[1,] -0.2182179 -0.3603750 -0.42285541  0.43329786
[2,] -0.2182179 -0.3243375 -0.29599879  0.17331914
[3,] -0.2182179 -0.2883000 -0.18249549 -0.01824412
[4,] -0.2182179 -0.2522625 -0.08234553 -0.14899365
[5,] -0.2182179 -0.2162250  0.00445111 -0.22653116
[6,] -0.2182179 -0.1801875  0.07789442 -0.25845837
>
> ### compare to x.ortho.intcpt
>
> head(x.ortho.intcpt)
      1         2         3
[1,] 0.2182179 -0.3603750 0.42285541 -0.43329786
[2,] 0.2182179 -0.3243375 0.29599879 -0.17331914
[3,] 0.2182179 -0.2883000 0.18249549  0.01824412
[4,] 0.2182179 -0.2522625 0.08234553  0.14899365
[5,] 0.2182179 -0.2162250 -0.00445111 0.22653116
[6,] 0.2182179 -0.1801875 -0.07789442 0.25845837
>
> ### extract R (then take inverse)
>
> round(solve(qr.R(qr.raw)),4)
      [,1]      [,2]      [,3]      [,4]
-0.2182  0.0000  0.2448  0.0000
1  0.0000  0.3604  0.0000  0.8337
2  0.0000  0.0000 -0.6677  0.0000
3  0.0000  0.0000  0.0000 -1.2670

```

```

>
> ### compare to A
>
> round(solve(t(x.ortho.intcpt)%*%x.raw.intcpt),4)
      1       2       3
0.2182 0.0000 -0.2448 0.0000
1 0.0000 0.3604 0.0000 -0.8337
2 0.0000 0.0000 0.6677 0.0000
3 0.0000 0.0000 0.0000 1.2670
>

```

As can be seen, \mathbf{Q} differs from \mathbf{X}' only by column-wise changes of sign, which does not change the transformation in any important way. Similarly, \mathbf{R}^{-1} differs from \mathbf{A} only by column-wise changes in sign, which yields essentially the same transformation.

19.3 Splines

The method of *splines* are a particularly powerful application of basis functions in linear regression. Formally, a spline is system of piecewise polynomials which may be constructed from basis functions, and therefore used to fit regression models defined by response functions of the form (19.1). The definition is straightforward. We will set $g(x)$ be the piecewise polynomial. Let I_x be the range of the predictor variable x . The first step is to define the number of “pieces”. This is done indirectly by defining K knots $\xi_1 < \xi_2 < \dots < \xi_{K-1} < \xi_K$, which are within the range I_x . Then $g(x)$ is taken to be a polynomial between each pair of consecutive knots, below ξ_1 , and above ξ_K :

$$g(x) = \begin{cases} a_{1,0} + a_{1,1}x + \dots + a_{1,d_1}x^{d_1} & ; \quad x < \xi_1 \\ a_{2,0} + a_{2,1}x + \dots + a_{2,d_2}x^{d_2} & ; \quad \xi_1 \leq x < \xi_2 \\ \vdots & ; \quad \vdots \\ a_{K,0} + a_{K,1}x + \dots + a_{K,d_K}x^{d_K} & ; \quad \xi_{K-1} \leq x < \xi_K \\ a_{K+1,0} + a_{K+1,1}x + \dots + a_{K+1,d_{K+1}}x^{d_{K+1}} & ; \quad \xi_K \leq x \end{cases}. \quad (19.10)$$

We have defined $K + 1$ polynomials on $K + 1$ contiguous left-closed intervals. The degree of the polynomials is allowed to vary, the degree of the k th polynomial being d_k . Note that Equation (19.10) yields a right-continuous piecewise polynomial. This is because polynomials themselves are continuous, and the segments are defined on left-closed intervals, for example $[\xi_k, \xi_{k+1})$. This is a convention which can be altered. Of course, if the piecewise polynomial is constrained to be at least continuous at the knots, the distinction has no effect.

There are three components required to define a family of spline functions intended to model a response function $g(x)$ defined on interval I_x :

- (i) K knots $\xi_1 < \xi_2 < \dots < \xi_{K-1} < \xi_K$, with $\xi_k \in I_x$.
- (ii) $K + 1$ polynomials of degrees d_1, \dots, d_{K+1} , which define the polynomial segments between the knots.
- (iii) Continuity conditions are optionally imposed at the knots. In particular, at knot ξ_k , the piecewise polynomial may be constrained to be continuous up to the s_k th derivative.

Note that the zeroth order derivative of a function $f(x)$ is conventionally defined as the function itself.

Properties (i) and (ii) are expressed directly in Equation (19.10), while property (iii) is an additional constraint. This is because there is nothing in the spline construction of (19.10) which forces any type of continuity. Of course, all order derivatives of a polynomial are continuous, so continuity constraints only need to be considered at the knots.

Usually, splines are constrained to be continuous everywhere. In addition, continuity may be imposed on higher order derivatives, which causes the spline to become smoother. This is a quite natural constraint. Recall, for example, that velocity is a first order derivative, and acceleration is a second order derivative, which we expect to change smoothly (a discrete change in velocity would require infinite acceleration, which is a physical impossibility).

In particular, when continuity of all derivatives up to the second order is imposed, the piecewise structure of a spline is usually difficult to detect visually, and this type of spline is widely used (in particular, various classes of *cubic splines*, which we will introduce below).

As a rule of thumb, continuity constraints can be relaxed for exploratory analyses, when the objective is to visualize the general shape of response function $g(x)$. However, it must also be considered when a response function can be expected to be smooth, and to understand how this constraint can be imposed.

Formally, in order for a function $f(x)$ to have an order $s > 0$ derivative at x_0 , it must possess continuous derivatives of all orders less than s , including order 0, that is, $f(x)$ must be continuous at x_0 . However, this does not mean that forcing order s continuity on a spline will, by itself, force continuity of all lower order derivatives. This is made clear in the next example.

Example 19.3. Suppose in the spline construction of (19.10) we set $K = 1$, $\xi_1 = 3$, and $d_1 = d_2 = 1$. This is a spline with a single knot at $x = \xi_1 = 3$, and linear polynomial segments. We may impose equality of any derivative evaluated at a knot of the segments joined there in a direct manner. For example, to impose continuity (the zeroth order derivative) we obtain with this method the linear constraint

$$a_{1,0} + a_{1,1}\xi_1 = a_{2,0} + a_{2,1}\xi_1,$$

equivalently

$$a_{1,0} + 3a_{1,1} = a_{2,0} + 3a_{2,1}. \quad (19.11)$$

The first order derivatives of each segment are the constants $a_{1,1}$, $a_{2,1}$ respectively, so equality of the first order derivatives at the knot is expressed as the linear constraint

$$a_{1,1} = a_{2,1}. \quad (19.12)$$

The spline contains 4 parameters $(a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1})$. However, if we were to impose only constraint (19.11) the number of parameters would effectively be reduced by 1. This could be expressed by simple substitution. Suppose these parameters are denoted α, β, γ . We might set

$$\alpha = a_{1,0}, \quad \beta = a_{1,1}, \quad \gamma = a_{2,1}.$$

Then constraint (19.11) forces the remaining substitution

$$a_{2,0} = a_{1,0} + 3a_{1,1} - 3a_{2,1} = \alpha + 3(\beta - \gamma).$$

This spline is now

$$g(x) = \begin{cases} \alpha + \beta x & ; \quad x < \xi_1 \\ \alpha + 3(\beta - \gamma) + \gamma x & ; \quad \xi_1 \leq x \end{cases} .$$

It is easily verified that this spline is continuous at the knot.

What happens if we impose only the first order constraint (19.12)? Mathematically, there is no reason we can't, and Figure 19.2 gives an example of what such a spline might look like. Clearly, both segments have the same first order derivative (equal to 1), but the spline itself is not continuous at the knot. Nor does it have a first order derivative at the knot by any evaluation method (the right-derivative is 1, while the left-derivative would be ∞ using the usual limit method). Thus, constraint (19.12) is not sufficient by itself to force first order continuity. In general, continuity of the order s derivative requires continuity of all lower order derivatives, and each of these is represented by a separate linear constraint.

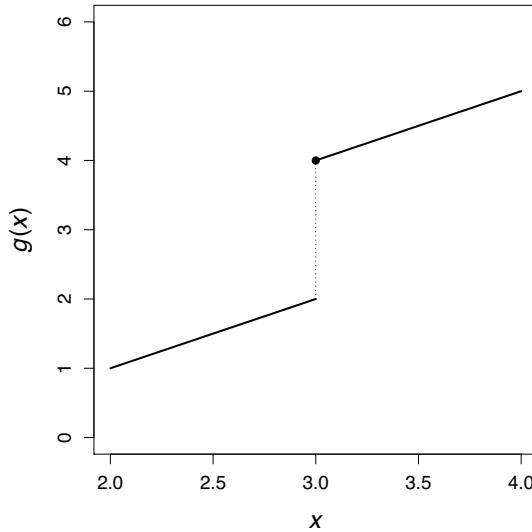


Figure 19.2: Spline used in Example 19.3.

□

Boundary knots

It will sometimes be convenient to explicitly define *boundary knots* ξ_0, ξ_{K+1} , where $\xi_0 < \xi_1$ and $\xi_{K+1} > \xi_K$. These do not appear in (19.10), so we must remember that $g(x)$ has domain I_x . In practice, all numerical computation is bounded, so the boundary knots are sooner or later required. In particular, the first and last polynomial segments are defined on respective intervals $[\xi_0, \xi_1]$ and $[\xi_K, \xi_{K+1}]$, so that $I_x = [\xi_0, \xi_{K+1}]$.

In much of the discussion of this section, the domain I_x plays no role, and we need only assume that it contains all knots. Furthermore, continuity constraints are not applied at the boundary knots, so there is little to be gained by introducing them into most of the analysis (although some applications, not considered here, do introduce some form of boundary constraint).

However, much of the study of splines concerns their representation by basis functions (which we will discuss below), and the design of such functions with desirable computational properties. Some such representations do make use of boundary knots. When these are required by R functions, the usual convention is to set the boundary knots to be the minimum and maximum observed predictor variables x_i , when available. It may be possible, however, for the user to specify alternative values. This is further discussed in Section 19.3.5.

19.3.1 Degrees of freedom of a spline

The moral of Example 19.3 is that equality at a knot of the order s derivatives of the adjoining spline segments is not sufficient to guarantee the existence of an order s derivative of the spline itself. Therefore, we can assume that when continuity of an order s derivative is imposed at a given knot, continuity of all lower order derivatives will also be imposed. Since each derivative constraint generates a distinct linear constraint of the type developed in Example 19.3, order s continuity will require $s + 1$ such constraints. Furthermore, each knot generates a separate set of constraints.

Suppose for a piecewise polynomial of the form (19.10) we set $K = 1$ knot ξ_1 , and $d_1 = 1$, $d_2 = 2$. The polynomial segments are linear and quadratic, respectively, so that

$$g(x) = \begin{cases} a_{1,0} + a_{1,1}x & ; \quad x < \xi_1 \\ a_{2,0} + a_{2,1}x + a_{2,2}x^2 & ; \quad \xi_1 \leq x \end{cases}.$$

When used in regression models, the knots of a spline are usually considered fixed, but the polynomial coefficients are to be estimated. In this example, there are five coefficients $a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, a_{2,2}$. Without any further constraint, this model therefore has five independent parameters, or, the spline has five degrees of freedom.

Next, suppose we impose continuity at the knot ξ_1 . This can be written explicitly as a linear constraint on the parameters:

$$a_{1,0} + a_{1,1}\xi_1 = a_{2,0} + a_{2,1}\xi_1 + a_{2,2}\xi_1^2.$$

Thus, (as in Example 19.3) we can eliminate one parameter by expressing it as a function of the remaining ones,

$$a_{1,0} = -a_{1,1}\xi_1 + a_{2,0} + a_{2,1}\xi_1 + a_{2,2}\xi_1^2$$

so that when the continuity constraint is imposed, the spline now has four degrees of freedom. We have already seen that an equality constraint imposed on any order derivative at a knot can be expressed as a linear constraint on the coefficients of the polynomial segments joined at that knot. Thus, we can see that the spline degrees of freedom is simply the total number of coefficients minus the total number of constraints.

Then let c_k be the number of constraints at knot ξ_k . This will be

$$c_k = \begin{cases} s_k + 1 & ; \quad s_k \text{ is defined} \\ 0 & ; \quad s_k \text{ is undefined} \end{cases}$$

where s_k is the order derivative constrained to be continuous at knot ξ_k . If no continuity constraints are imposed, s_k is undefined. If only continuity of the spline is imposed, we have $s_k = 0$.

We then have the formula for the model degrees of freedom of a spline:

$$\text{model DF} = \sum_{k=1}^{K+1} (d_k + 1) - \sum_{k=1}^K c_k. \quad (19.13)$$

If the spline is continuous s_k is always defined, and we may equivalently write

$$\text{model DF} = 1 + \sum_{k=1}^{K+1} d_k - \sum_{k=1}^K s_k. \quad (19.14)$$

Note that to apply (19.13) we must assume that $s_k \leq \max\{d_k, d_{k+1}\}$ at each knot. Otherwise, the order s_k derivative at that knot will already be constrained to be zero, and there is no need to impose any further continuity conditions. As a practical matter, we further expect $s_k < \max\{d_k, d_{k+1}\}$ (what is $g(x)$ if $d_k = 1$, $s_k = 1$ for all knots and segments?).

Example 19.4. We wish to fit the model

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n, \quad (19.15)$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable. The function $g(x)$ has the following properties:

- (i) There are three knots $\xi_1 < \xi_2 < \xi_3$.
- (ii) $g(x)$ is continuous at the knots.
- (iii) $g(x)$ possesses a continuous first derivative at the knots.
- (iv) $g(x)$ is linear $g(x) = a_0 + b_0x$ for $x < \xi_1$.
- (v) $g(x)$ is a second order polynomial $g(x) = a_1 + b_1x + c_1x^2$ for $x \in (\xi_1, \xi_2)$.
- (vi) $g(x)$ is a second order polynomial $g(x) = a_2 + b_2x + c_2x^2$ for $x \in (\xi_2, \xi_3)$.
- (vii) $g(x)$ is linear $g(x) = a_3 + b_3x$ for $x > \xi_3$.

Continuity of $g(x)$ and continuity of the first derivative each represent a single linear constraint at each knot, so there are 6 linear constraints. These are

$$\begin{aligned} a_0 + b_0\xi_1 &= a_1 + b_1\xi_1 + c_1\xi_1^2 \\ a_1 + b_1\xi_2 + c_1\xi_2^2 &= a_2 + b_2\xi_2 + c_2\xi_2^2 \\ a_2 + b_2\xi_3 + c_2\xi_3^2 &= a_3 + b_3\xi_3 \end{aligned}$$

for continuity, and

$$\begin{aligned} b_0 &= b_1 + 2c_1\xi_1 \\ b_1 + 2c_1\xi_2 &= b_2 + 2c_2\xi_2 \\ b_2 + 2c_2\xi_3 &= b_3 \end{aligned}$$

for continuity of the first derivative. The three polynomials together have 10 coefficients. With 6 constraints, this leaves $10 - 6 = 4$ degrees of freedom.

Applying (19.14) directly, we have $d_1 = d_4 = 1$, $d_2 = d_3 = 2$, $s_1 = s_2 = s_3 = 1$, so

$$\text{model DF} = 1 + (1 + 2 + 2 + 1) - (1 + 1 + 1) = 4.$$

□

19.3.2 Basis function representation of a spline

Without continuity constraints, the space of all piecewise polynomials of the form (19.10) forms a vector space \mathcal{V}_{pp} with dimension equal to the model degrees of freedom. Then note that continuity of any order is preserved under addition and scalar multiplication. This means the set of all piecewise polynomials with a fixed set of continuity constraints is also a vector space, say $\mathcal{V}_{pp}^* \subset \mathcal{V}_{pp}$, and will also have dimension equal to the model degrees of freedom. This leaves the problem of determining a basis for \mathcal{V}_{pp}^* . In fact, this field is a very rich component of many areas of applied mathematics, especially numerical analysis, computer graphics and physics. Apart from enabling many important computational methods, it is also yields important theory. We will get a glimpse of this later in this chapter.

We must also consider here the issue raised in Section 19.1 (Theorem 19.2 and the preceding discussion). Although some of the theory of basis functions does not refer to any particular observed predictor values x_1, \dots, x_n , when used in linear regression models to define a response function such as

$$g(x) = \sum_{k=1}^p \beta_k b_k(x),$$

it must be ensured that a design matrix of the form (19.2) have linearly independent columns. To ensure this, there must a some minimum number of observed predictors x_i between each pair of knots ξ_k, ξ_{k+1} , below ξ_0 , and above ξ_K . This minimum depends on the degrees of the segments, and the constraints at the knots. If the total number of observations n equals the model degrees of freedom, it will be possible under general conditions to determine parameters which force $g(x_i) = y_i$ for all i , yielding $SSE = 0$. Of course, it is good modeling practice to ensure that n is much larger than the model degrees of freedom, and in the case of splines, to ensure that the each of the segments partitioned by the knots contains a sufficient number of observations x_i . A good strategy is to select the knots based on sample quantiles of the observed predictors. For example, if $K = 1$, ξ_1 would be a median, or for $K = 4$, the knots would be quintiles. See, for example, James *et al.* (2013) for more on the selection of knots.

Step functions

First consider the *step function* (for example, Spline (a) of Figure 19.3). This is a particularly simple spline, with $d_k = 0$ for each segment in Equation (19.10), and no continuity constraints, so that $c_k = 0$ at each knot. This type of function is easily defined as any function that is constant between knots, and has model degrees of freedom

$$\text{model DF} = \sum_{k=1}^{K+1} (d_k + 1) - \sum_{k=1}^K c_k = K + 1.$$

Here, we may define basis functions

$$\begin{aligned} b_0(x) &= I\{x < \xi_1\} \\ b_1(x) &= I\{\xi_1 \leq x < \xi_2\} \\ &\vdots \\ b_{K-1}(x) &= I\{\xi_{K-1} \leq x < \xi_K\} \end{aligned} \tag{19.16}$$

$$b_K(x) = I\{\xi_K \leq x\}. \tag{19.17}$$

Then any left-continuous step function with knots $\xi_1 < \xi_2 < \dots < \xi_{K-1} < \xi_K$ can be represented as

$$g(x) = \sum_{k=0}^K \beta_k b_k(x),$$

and linear regression may be used to estimate the coefficients β_k . Of course, with this representation, an intercept term would not be included.

It is clear that the basis functions $\mathcal{B} = (b_0, b_1, \dots, b_K)$ are linearly independent, and therefore form the basis of a vector space \mathcal{V}_{step} consisting of all step functions with discontinuities at the given knots (recall Definition 7.2).

Then consider the following set of basis functions $\mathcal{B}' = (b'_0, b'_1, \dots, b'_K)$ defined by

$$\begin{aligned} b'_0(x) &= 1 \\ b'_1(x) &= I\{\xi_1 \leq x\} \\ &\vdots \end{aligned} \tag{19.18}$$

$$\begin{aligned} b'_{K-1}(x) &= I\{\xi_{K-1} \leq x\} \\ b'_K(x) &= I\{\xi_K \leq x\}, \end{aligned} \tag{19.19}$$

We can use Theorem 19.1 to verify that \mathcal{B} and \mathcal{B}' are equivalent. Then, because \mathcal{B} is a basis for \mathcal{V} , and \mathcal{B} and \mathcal{B}' have the same number of basis functions, \mathcal{B}' must also be a basis for \mathcal{V} , and can be used to fit an equivalent least squares regression model. It is easy to verify that the following equalities hold:

$$\begin{aligned} b'_0(x) &= \sum_{k=0}^K b_k(x), \\ b'_1(x) &= \sum_{k=1}^K b_k(x), \\ &\vdots \\ b'_K(x) &= b_K(x), \end{aligned}$$

in fact, we may more compactly write

$$b'_j(x) = \sum_{k=j}^K b_k(x),$$

for all indices $j = 0, 1, \dots, K$. It follows that the bases \mathcal{B} and \mathcal{B}' can be related by the linear transformation (19.5), with $(K+1) \times (K+1)$ transformation matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

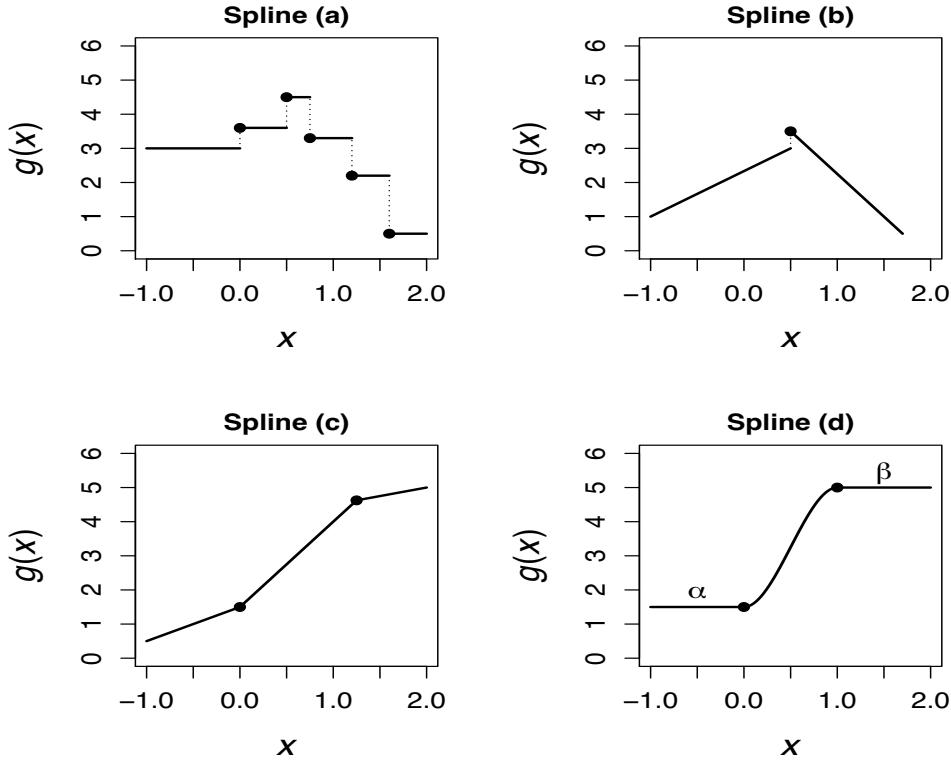


Figure 19.3: Four examples of splines.

This is a lower triangular matrix with nonzero diagonal entries, and is therefore invertible. We conclude from Theorem 19.1 (ii) that \mathcal{B} and \mathcal{B}' are equivalent, and therefore \mathcal{B}' is a basis for \mathcal{V} .

We next give an example.

Example 19.5. We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and $x_i \in [0, 10]$ is a predictor variable. We will assume that $g(x)$ is a step function with two discontinuities at $\xi = 2, 5$. We may use the basis

$$\begin{aligned} b_1(x) &= I\{x < 2\} \\ b_2(x) &= I\{x \in [2, 5)\} \\ b_3(x) &= I\{x \geq 5\}. \end{aligned}$$

Then set

$$g(x) = \beta_1 b_1(x) + \beta_2 b_2(x) + \beta_3 b_3(x)$$

and use least squares regression to estimate β_1, β_2 and β_3 (no intercept term will be added here).

Alternatively, we could use basis

$$\begin{aligned} b'_1(x) &= 1 \\ b'_2(x) &= I\{x \geq 2\} \\ b'_3(x) &= I\{x \geq 5\}, \end{aligned}$$

Then set

$$g(x) = \beta'_1 b'_1(x) + \beta'_2 b'_2(x) + \beta'_3 b'_3(x)$$

and use least squares regression to estimate β'_1 , β'_2 and β'_3 (again, no intercept term will be added here).

Then suppose, using basis $(b_1(x), b_2(x), b_3(x))$, least squares estimates $\hat{\beta}_1$, $\hat{\beta}_2$ and $\hat{\beta}_3$ of coefficients β_1 , β_2 and β_3 are obtained. We can derive the least squares estimates of β'_1 , β'_2 and β'_3 (using basis $(b'_1(x), b'_2(x), b'_3(x))$) as a function of $\hat{\beta}_1$, $\hat{\beta}_2$ and $\hat{\beta}_3$.

We relate the two sets of basis functions as follows:

$$\begin{aligned} b_1(x) &= b'_1(x) - b'_2(x) \\ b_2(x) &= b'_2(x) - b'_3(x) \\ b_3(x) &= b'_3(x). \end{aligned}$$

Then write explicitly the least squares fit, and substitute the basis functions:

$$\begin{aligned} g(x) &\approx \hat{\beta}_1 b_1(x) + \hat{\beta}_2 b_2(x) + \hat{\beta}_3 b_3(x) \\ &= \hat{\beta}_1 [b'_1(x) - b'_2(x)] + \hat{\beta}_2 [b'_2(x) - b'_3(x)] + \hat{\beta}_3 b'_3(x) \\ &= \hat{\beta}_1 b'_1(x) + [\hat{\beta}_2 - \hat{\beta}_1] b'_2(x) + [\hat{\beta}_3 - \hat{\beta}_2] b'_3(x) \\ &= \hat{\beta}'_1 b'_1(x) + \hat{\beta}'_2 b'_2(x) + \hat{\beta}'_3 b'_3(x). \end{aligned}$$

This gives

$$\begin{aligned} \hat{\beta}'_1 &= \hat{\beta}_1 \\ \hat{\beta}'_2 &= \hat{\beta}_2 - \hat{\beta}_1 \\ \hat{\beta}'_3 &= \hat{\beta}_3 - \hat{\beta}_2. \end{aligned}$$

□

Piecewise linear splines

A piecewise linear spline is usually taken to be a continuous piecewise polynomial with linear polynomial segments. Thus, given K knots we set $d_k = 1$, $k = 1, \dots, K+1$, $s_k = 0$, $k = 1, \dots, K$. The model degrees of freedom is therefore

$$\text{model DF} = 1 + \sum_{k=1}^{K+1} d_k - \sum_{k=1}^K s_k = K + 2. \quad (19.20)$$

If, for the moment, we disregard the continuity constraint, the class of piecewise linear polynomials constructed from (19.10) will be the vector space spanned by the $2K$ basis functions

$$\begin{aligned}
 b_1(x) &= I\{x < \xi_1\} \\
 b_2(x) &= xI\{x < \xi_1\} \\
 b_3(x) &= I\{\xi_1 \leq x < \xi_2\} \\
 b_4(x) &= xI\{\xi_1 \leq x < \xi_2\} \\
 &\vdots \\
 b_{2K+2}(x) &= I\{\xi_K \leq x\} \\
 b_{2K+3}(x) &= xI\{\xi_K \leq x\}.
 \end{aligned} \tag{19.21}$$

Furthermore, these basis functions are clearly linearly independent, and therefore form a basis for the vector space \mathcal{V}_{lin} of all piecewise linear polynomials for a given set of knots.

However, the continuity constraint complicates the problem of constructing a basis for the class of *continuous* piecewise linear splines. It may help at this point to compare an unconstrained linear spline and a continuous linear spline (Splines (b) and (c) in Figure 19.3). Spline (b) can be constructed directly from the basis (19.21), setting $K = 1$,

$$g(x) = \sum_{j=1}^4 \beta_j b_j(x)$$

so that $\beta_1 + \beta_2 x$ and $\beta_3 + \beta_4 x$ are the two polynomial segments.

As for the continuous piecewise linear spline, Spline (c), it turns out that a basis transformation similar to that between bases (19.17) and (19.19) will prove useful in deducing how to constrain the β_j coefficients to induce continuity. This yields equivalent basis (by an argument similar to that used above for the step function basis)

$$\begin{aligned}
 b'_1(x) &= 1 \\
 b'_2(x) &= x \\
 b'_3(x) &= I\{\xi_1 \leq x\} \\
 b'_4(x) &= xI\{\xi_1 \leq x\} \\
 &\vdots \\
 b'_{2K+1}(x) &= I\{\xi_K \leq x\} \\
 b'_{2K+2}(x) &= xI\{\xi_K \leq x\}.
 \end{aligned} \tag{19.22}$$

Now Spline (c) can be written

$$g(x) = \sum_{j=1}^6 \beta_j b'_j(x) \tag{19.23}$$

or, equivalently,

$$g(x) = \begin{cases} \beta_1 b'_1(x) + \beta_2 b'_2(x) &; x < \xi_1 \\ \beta_1 b'_1(x) + \beta_2 b'_2(x) + \beta_3 b'_3(x) + \beta_4 b'_4(x) &; \xi_1 \leq x < \xi_2 \\ \beta_1 b'_1(x) + \beta_2 b'_2(x) + \beta_3 b'_3(x) + \beta_4 b'_4(x) + \beta_5 b'_5(x) + \beta_6 b'_6(x) &; \xi_2 \leq x \end{cases} .$$

For basis (19.22) it is much easier to determine how to apply the continuity constraints to the β_j coefficients. First note that by Equation (19.20) this spline has 4 model degrees of freedom. Denote the associated independent parameters $\alpha_1, \dots, \alpha_4$. For x below the first knot, the only contribution to $g(x)$ are the terms $\beta_1 b'_1(x) + \beta_2 b'_2(x)$. But these terms contribute to $g(x)$ over the entire domain, so we will set the first two independent parameters:

$$\begin{aligned}\alpha_1 &= \beta_1 \\ \alpha_2 &= \beta_2.\end{aligned}$$

Note that when x reaches the first knot ξ_1 from below, the terms $\beta_3 b'_3(x) + \beta_4 b'_4(x)$ are added to $g(x)$. If $g(x)$ is continuous at ξ_1 we must therefore have

$$\beta_3 b'_3(\xi_1) + \beta_4 b'_4(\xi_1) = 0$$

or equivalently

$$\beta_3 + \beta_4 \xi_1 = 0.$$

This becomes the linear constraint induced by continuity at ξ_1 , and by a similar argument for knot ξ_2 we have additional constraint

$$\beta_5 + \beta_6 \xi_2 = 0.$$

then set

$$\begin{aligned}\alpha_3 &= \beta_4 \\ \alpha_4 &= \beta_6.\end{aligned}$$

After a bit of algebra spline (19.23) can now be written

$$g(x) = \alpha_1 + \alpha_2 x + \alpha_3(x - \xi_1)_+ + \alpha_4(x - \xi_2)_+,$$

using the notational convention

$$(x)_+ = I\{x \geq 0\}.$$

We can now identify a basis of $K + 2$ functions for the vector space of continuous piecewise linear polynomials with K knots:

$$\begin{aligned}h_1(x) &= 1 \\ h_2(x) &= x \\ h_3(x) &= (x - \xi_1)_+ \\ &\vdots \\ h_{K+2}(x) &= (x - \xi_K)_+. \tag{19.24}\end{aligned}$$

Essentially this method can be used to construct a basis for important classes of splines, which we demonstrate next.

19.3.3 Splines of degree d

The reader should be aware that the term “spline” is not used consistently in the literature. We have used the term here to describe any piecewise polynomial function. This might be seen as reasonable, up to a point. The discussion surrounding Example 19.3 suggests that the properties of such a “spline” do not follow directly from the polynomial segments, and can only be characterized when the segments are aggregated into a single function, now called a “spline”.

However, in practice the term “spline” often refers specifically to a piecewise polynomial with segments of degree d and continuity of all derivatives up to and including order $d - 1$. We refer to this as a *spline of degree d* . The step function (by default) and the continuous piecewise linear polynomial are thus splines of degree $d = 0$ and $d = 1$, respectively. Then (19.24) is a basis for the class of splines of degree $d = 1$, and is easily generalized to form bases for splines of any degree d :

$$\begin{aligned} h_1(x) &= 1 \\ h_2(x) &= x \\ &\vdots \\ h_d(x) &= x^{d-1} \\ h_{d+1}(x) &= x^d \\ h_{d+2}(x) &= (x - \xi_1)_+^d \\ &\vdots \\ h_{K+d+1}(x) &= (x - \xi_K)_+^d. \end{aligned} \tag{19.25}$$

The number of basis functions, $K + d + 1$ must equal the model degrees of freedom, which is easily verified for any $d \geq 1$, setting $d_k = d$, $s_k = d - 1$:

$$\text{model DF} = 1 + \sum_{k=1}^{K+1} d_k - \sum_{k=1}^K s_k = 1 + (K + 1)d - K(d - 1) = K + d + 1. \tag{19.26}$$

Lower order splines are useful for exploratory analysis, otherwise, the *cubic spline* ($d = 3$) is commonly used, in part because it possesses some remarkable properties (Section 19.3.4 below).

Of course, this type of uniformity of degree and continuity can be relaxed, and there are often good reasons to do so. Consider Spline (d) of Figure 19.3. The outer polynomial segments are of degree $d = 0$, the remaining segment of degree $d = 3$. There are 6 coefficients, which bounds the number of linear constraints by 6 (or, the total order continuity across all knots by 4). We may impose order 0 or 1 continuity at both knots, which yield distinct models. As an exercise, the reader might describe the spline when order $s_1 = 3$, $s_2 = 0$ continuity is imposed, or order $s_1 = s_2 = 2$ continuity (for the latter example, describe a vector space containing only one vector). This example is further considered in Practice Problem 30.6.

19.3.4 Natural cubic splines and smoothing splines

A *natural cubic spline* is simply a cubic spline for which the two outer segments (below the first knot and above the last knot) are constrained to be linear. This may be done by forcing the second and third degree coefficients of these segments to zero, reducing the total number of parameters by

4. However, order 2 continuity must still be imposed at all knots, since, even after the linearization of the outer segments, at least one adjoining segment of each knot is a polynomial of degree 3. So the model degrees of freedom of a natural spline is 4 less than that of a cubic spline, that is,

$$\text{model DF} = K.$$

There are a number of reasons to consider the natural cubic spline. The cubic spline itself offers continuity up to the second derivative, while only requiring $K + 4$ parameters (4 for a cubic polynomial, and 1 more for each knot). Thus, there is considerable flexibility in modeling the predictor function $g(x)$ for a modest price in increased model complexity.

However, as a practical matter, cubic splines tend to exhibit undesirable properties at the extreme regions of the predictor domain I_x , particularly outside the range of the knots. These include large standard errors of the fitted values (see Section 6.3), as well as counterintuitive behavior of the response function $g(x)$ itself. These problems can usually be controlled by using a natural cubic spline.

In addition, the natural cubic spline possesses a remarkable property. Consider the following optimization problem. Suppose we are given paired observations of predictor and response variables (x_i, y_i) , $i = 1, \dots, n$. Assume for the moment that the predictor observations are distinct. Then consider the following penalized least squares function

$$\Lambda(g) = \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int_{I_x} g''(u)^2 du, \quad (19.27)$$

for some fixed $\lambda \geq 0$. Here, g is *any* twice-differentiable function on I_x , and g'' is the second order derivative of g . It is interesting to note that since the purpose of the second term is to bound the curvature of g , we could also relax the twice-differentiable assumption on g . Then if we claim, reasonably, that the lack of a second derivative implies unbounded curvature, we could eliminate from consideration all such functions.

It may be proven that the function g which minimizes $\Lambda(g)$ for any $\lambda \geq 0$ is a natural cubic spline with knots located at the predictor observations x_1, \dots, x_n . However, this optimal spline depends on λ , which is therefore denoted g_λ . This is referred to as a *smoothing spline*. The natural cubic spline, outside of the context of this optimization problem, has K model degrees of freedom. However, the curvature penalty in Equation (19.27) has the effect of imposing further constraints on g_λ , so it is worth considering what happens as λ ranges from 0 to ∞ .

For $\lambda = 0$, we have the error sum of squares SSE that is minimized in least squares regression models. There is an important difference, however. We have assumed that the x_i values are unique. In addition, a natural spline with $K = n$ knots has n model degrees of freedom. It is therefore possible to introduce the n constraints

$$g(x_i) = y_i, \quad i = 1, \dots, n,$$

attaining the minimum possible value of $\Lambda(g)$:

$$\Lambda(g) = \sum_{i=1}^n (y_i - g(x_i))^2 + 0 \times \int_{I_x} g''(u)^2 du = \sum_{i=1}^n (y_i - g(x_i))^2 = 0. \quad (19.28)$$

Then, we note that as λ approaches ∞ , the curvature penalty in Equation (19.27) becomes increasingly dominant. Of course, if $g(x)$ is linear, $g''(x) = 0$, so the increasing weight placed on

the penalty can be compensated for by reducing the aggregate curvature of $g(x)$ by any required amount. This means that as $\lambda \rightarrow \infty$, $g_\lambda(x)$ approaches a linear function, specifically, the least squares fit for the simple linear regression model for the predictor/response pairs.

Finally, we note that the assumption that the observed predictors are unique can be relaxed, and the algorithm will remain well defined. This is implemented in R by the `smooth.spline()` function.

Effective degrees of freedom (EDF)

For $\lambda = 0$, $g_\lambda(x)$ is an unpenalized natural cubic splines, which has n model degrees of freedom, while for $\lambda = \infty$ $g_\lambda(x)$ is a linear function with 2 model degrees of freedom. However, since all g_λ are nominally natural cubic splines, an alternative to model degrees of freedom must be used to quantify model complexity.

To this end, suppose we have linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where \mathbf{X} is the $n \times p$ design matrix. Recall from Chapter 6 that the least squares fit may be expressed

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}, \quad (19.29)$$

where \mathbf{H} is the $n \times n$ “hat matrix”

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \quad (19.30)$$

It turns out that the model degrees of freedom may be calculated from \mathbf{H} as

$$p = \text{trace}(\mathbf{H}) \quad (19.31)$$

However, models which are not calculated by the least squares method may nonetheless be expressible in the form (19.29), even if \mathbf{H} has a calculation method other than (19.30). As it happens, the smoothing spline is such a model (with \mathbf{H} relying on both \mathbf{X} and λ). So, we may define the *effective degrees of freedom* (EDF) as

$$\text{EDF} = \text{trace}(\mathbf{H}).$$

The EDF conforms to the model degrees of freedom for linear least squares regression, but is also able to extend this definition of model complexity to a much larger class of modeling techniques. For smoothing splines, the EDF of $g_\lambda(x)$ decreases from n to 2, as λ ranges from 0 to ∞ . We therefore have an important example of the bias/variance tradeoff induced by varying model complexity (Section 15.7). Practice Problem 30.23 discusses the EDF in some depth, providing a proof of Equation (19.31), as well as examples of the evaluation of EDF for fitting methods that do not rely on the least squares method. See also James *et al.* (2013) or Friedman *et al.* (2001) for more on this topic.

19.3.5 B-splines

One of the points made throughout this section is that bases can be equivalent while having very different properties from a more practical point of view. For example, the two bases (19.17) and (19.19) for a vector space of step functions are equivalent. While (19.19) is analytically simpler, the basis functions of (19.17) have much smaller *support* (the subset of the domain on which the function is nonzero). This can be a considerable advantage, especially for large data sets. When

large portions of a matrix can be assumed to be zero, specialized algorithms can reduce computation time by an order of magnitude.

How then, can this idea be extended to splines of higher degree? Equivalently, what basis for a spline of degree d possesses minimum support? To develop a basis for (continuous) splines of degree 1 we first transformed (unconstrained) basis (19.21) to (19.22), then directly derived the continuity constraints from the latter basis, yielding (19.24). So, we may try to find a linear transformation of basis (19.24) which reduces basis function support.

First, note that basis function support can be expressed as the number of domain segments separated by the knots. Then, for example, basis function $h_2(x)$ of basis (19.24) covers all segments. However, suppose we take $\xi_1 = 2$, $\xi_2 = 4$. At this point we will introduce the boundary knot $\xi_0 = 0$ (Section 19.3). Then consider the linear combination

$$h'_2(x) = h_2(x) - 2h_3(x) + h_4(x). \quad (19.32)$$

The basis functions h_2 and h'_2 are shown in Figure 19.4, assuming knots $\xi_1 = 2$, $\xi_2 = 4$ (the reader may wish to verify this as an exercise). Clearly, $h'_2(x)$ is easy to describe. It is a “triangle function” with base $[\xi_0, \xi_2]$, and peak at knot ξ_1 . Crucially, it is zero outside the segments $[\xi_0, \xi_1]$ and $[\xi_1, \xi_2]$. In other words, its support consists of only two segments.

The total number of knots $K \geq 2$ is not important. If we take K to be arbitrarily large, we can continue this process, defining transformed basis functions

$$h'_k(x) = a_{k,k}h_k(x) + a_{k+1,k}h_{k+1}(x) + a_{k+2,k}h_{k+2}(x). \quad (19.33)$$

selecting coefficients $a_{k,k}, a_{k+1,k}, a_{k+2,k}$ so that $h'_k(x)$ has the same triangle structure, with base $[\xi_{k-1}, \xi_{k+1}]$ and peak at knot ξ_k . This can always be done (again, the reader may verify this).

Clearly, this generates a triangular transformation (Example 19.2), so that the basis functions $\{h_k, k \geq 2\}$ and $\{h'_k, k \geq 2\}$, are equivalent (K may be finite, but its value is still not important at this point).

We then note that the intercept term $h_1(x) \equiv 1$ is not included in the vector space spanned by $\{h_k, k \geq 2\}$, since (19.24) is a basis, and its basis functions are therefore linearly independent. Therefore, h_1 is not included in the vector space spanned by $\{h'_k, k \geq 2\}$, since it is equivalent to $\{h_k, k \geq 2\}$.

The class of degree 1 splines includes h_1 , so the vector space $\{h'_k, k \geq 2\}$ must be extended to include it. This can be done by simply adding $h_1(x)$ to the set $\{h'_k, k \geq 2\}$. However, the point of the exercise is to minimize the basis function support. To this end, note that we may always multiply basis functions by a nonzero scalar without changing the vector space they span. This means we may assume that the coefficients in (19.33) are selected so that the maximum of $h'_k(x)$ equals one. In this case, these basis functions will possess a remarkable property, in that their sum has a quite simple form:

$$\sum_{k \geq 2} h'_k(x) = \begin{cases} h'_2(x) & ; \quad x < \xi_1 \\ 1 & ; \quad x \geq \xi_1 \end{cases} \quad (19.34)$$

for all $x \in I_x$. Then suppose we define the remaining basis function

$$h'_1(x) = (1 - h'_2(x))I\{x < \xi_1\} \quad (19.35)$$

Then

$$\sum_{k \geq 1} h'_k(x) = 1, \quad (19.36)$$

$h_1(x)$ is in the vector space spanned by $\{h'_k, k \geq 1\}$, and the support of h'_1 consists only of the single segment $(-\infty, \xi_1)$.

We have, in fact, just constructed what is known as a *B-spline* (or the basis thereof) for the class of degree 1 splines. This can be done for any degree $d \geq 0$ and can be defined as a basis $\mathcal{B} = (b_1, \dots, n)$ possessing the following properties:

- (i) The basis spans the class of splines of degree d , and therefore has dimension $K + d + 1$.
- (ii) The support of any basis function never exceeds $d + 1$ segments.
- (iii) The basis functions satisfy the identity $\sum_{j=1}^n b_j(x) \equiv 1$.

With the exception of some number of basis functions with support near the boundary knots (this number depends on d) all basis functions of a B-spline have support of exactly $d + 1$ segments, and have the same shape. In this sense, they resemble the simplest spline basis considered in this section, that given in Equation (19.17) (which is a B-spline of degree 0). And it is this resemblance that yields their advantageous properties.

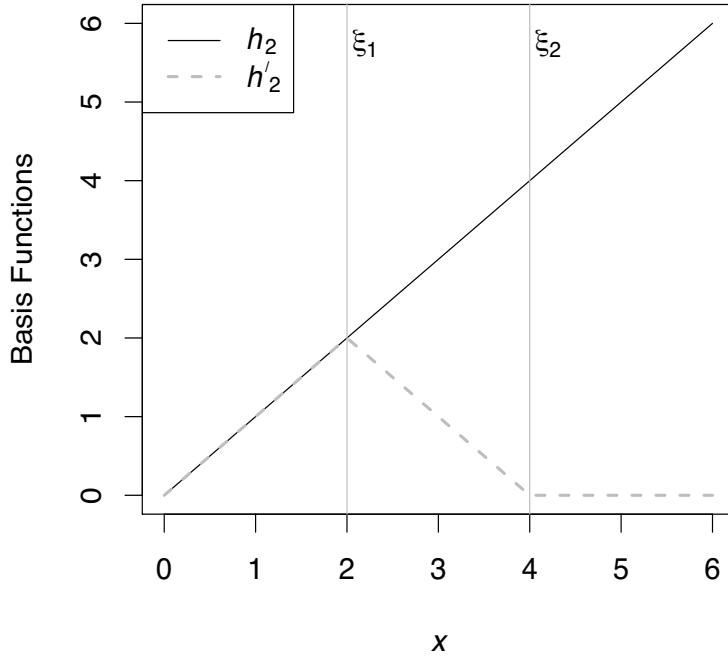


Figure 19.4: Transformed basis function h'_2 for degree $d = 1$ spline (Section 19.3.5).

19.3.6 Using B-splines in R

R provides a number of functions which support least squares spline regression using basis functions. The function `bs()` calculates design matrices of the form (19.2) based on B-spline basis functions. It is used much like the polynomial regression function `poly()` (Section 19.2.2), in particular, it can

be used directly in `formula` objects. The function accepts as an argument `x`, a vector containing the predictor observations; `degree`, the degree of the spline (with default `degree = 3`); and `knots`, a vector of spline knots.

The option `intercept` specifies whether or not the intercept is included in the span of the basis functions. The default is `intercept = FALSE`. If the option `intercept = TRUE` is used, the additional function will not be the constant $b_0(x) \equiv 1$, but a basis function more like $h'_1(x)$ of Equation (19.35).

Boundary knots can be specified as a vector of length 2 using the `Boundary.knots` argument. The default is `Boundary.knots = range(x)` (see Section 19.3).

It is also possible to specify `df`, the model degrees of freedom. Suppose a value of D is selected. Since a spline of degree d has $K+d+1$ model degrees of freedom, the `bs()` function will automatically select $K = D - d - 1$ knots with option `intercept = TRUE`, or $K = D - d$ knots with option `intercept = FALSE`. The knots will be selected as suitable quantiles. However, this option will be ignored if a vector of knots is provided. If neither `knots` or `df` are given, `bs()` assumed $K = 0$, which is equivalent to polynomial regression.

Natural cubic splines (Section 19.3.4) are implemented using the `ns()` function. The arguments are similar to those used by `bs()`, except that only splines of degree 3 are produced (with outer segments constrained to be linear).

The following code demonstrates the `bs()` function, producing Figure 19.5. Splines of degree $d = 1, 2, 3$ are shown, on interval $I_x = [0, 10]$, with $K = 9$ equally spaced knots.

```
> library(splines)
> par(mfrow=c(3,1),mar=c(4,6,5,3),cex.lab=1.5)
> x = seq(0,10,0.1)
> matplot(x,bs(x,degree=1,knots=1:9),type='l',ylab='Basis Functions')
> title("Degree 1 B-spline")
> abline(v=1:9,col='gray')
> matplot(x,bs(x,degree=2,knots=1:9),type='l',ylab='Basis Functions')
> title("Degree 2 B-spline")
> abline(v=1:9,col='gray')
> matplot(x,bs(x,degree=3,knots=1:9),type='l',ylab='Basis Functions')
> title("Degree 3 B-spline")
> abline(v=1:9,col='gray')
```

Finally, it should be noted that the B-spline basis is not orthonormal (except for $d = 0$). However, the resulting matrix $X^T X$ is still well behaved, and has zero entries at elements more than $d + 1$ rows or columns from the diagonal. The following code demonstrates this.

```
>
> ### (X^T)X matrix for degree 1 B-spline
>
> X = bs(x,degree=1,knots=1:4)
> t(X)%*%X
      1     2     3     4     5
1 6.70 1.65 0.00 0.00 0.00
2 1.65 6.70 1.65 0.00 0.00
```

```

3 0.00 1.65 6.70 1.65 0.00
4 0.00 0.00 1.65 6.70 1.65
5 0.00 0.00 0.00 1.65 3.85
>
> ### (X^T)X matrix for degree 2 B-spline
>
> X = bs(x,degree=2,knots=1:4)
> t(X)%*%X
      1         2         3         4         5         6
1 3.333250 2.083375 0.083325 0.000000 0.000000 0.00000
2 2.083375 5.499950 2.166700 0.083325 0.000000 0.00000
3 0.083325 2.166700 5.499950 2.166700 0.083325 0.00000
4 0.000000 0.083325 2.166700 5.499950 2.083375 0.16665
5 0.000000 0.000000 0.083325 2.083375 3.333250 1.15005
6 0.000000 0.000000 0.000000 0.166650 1.150050 2.53330
>
> ### (X^T)X matrix for degree s B-spline
>
> X = bs(x,degree=3,knots=1:4)
> t(X)%*%X
      1         2         3         4         5         6         7
1 2.214053125 1.562530833 0.345235417 0.002976875 0.000000000 0.000000000 0.0000000
2 1.562530833 3.267869931 2.246027917 0.237102986 0.001984583 0.000000000 0.0000000
3 0.345235417 2.246027917 4.793658333 2.363090972 0.237102986 0.002976875 0.0000000
4 0.002976875 0.237102986 2.363090972 4.793658333 2.246027917 0.345235417 0.0119075
5 0.000000000 0.001984583 0.237102986 2.246027917 3.267869931 1.562530833 0.1844837
6 0.000000000 0.000000000 0.002976875 0.345235417 1.562530833 2.214053125 0.8502038
7 0.000000000 0.000000000 0.000000000 0.011907500 0.184483750 0.850203750 1.9784050
>
```

19.4 Postscript

We have already discussed in some detail how basis function models are supported in R. Further examples are given in demonstration software files `NONLINEAR-MODELS-POLYNOMIAL.R` and `NONLINEAR-MODELS-SPLINES.R`, with more exercises in Chapter 30. An excellent specialized treatment of this topic is offered in Ramsay *et al.* (2009).

One limitation of the material in this chapter is that it only anticipates one predictor variable, which precludes the development of more complex predictive models. However, a class of models known as *general additive models* extends spline regression models to multivariate models, allowing splines built from distinct predictor variables to be used additively in a single linear model (Hastie and Tibshirani, 1990). This model is implemented in the R package `gam`, which is introduced in demonstration software file `NONLINEAR-MODELS-SPLINES.R`.

In general the topics in this chapter are discussed in James *et al.* (2013) or Friedman *et al.* (2001).

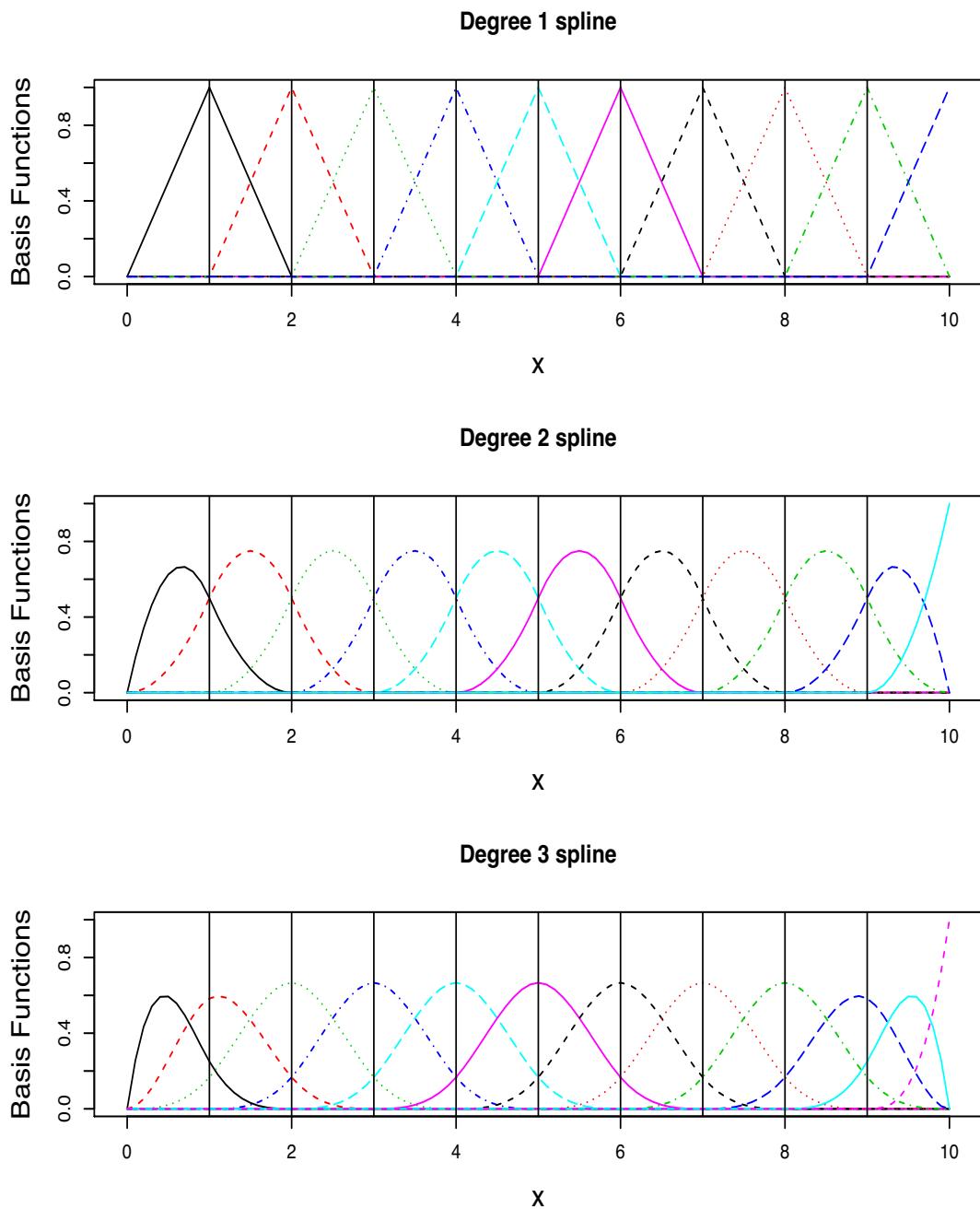


Figure 19.5: B-splines of degree 1,2 and 3 (Section 19.3.6).

Chapter 20

Bayesian Networks

The Bayesian network (BN) is an example of a *graphical model*. First, we define a *graph*. Let V be a discrete set of *nodes* (or *vertices*). These are usually labeled, and so without loss of generality we can set $V = \{1, \dots, n\}$, for a graph with n nodes. We also have a set of *edges*, which are pairs of nodes from V . An edge consisting of nodes i, j implies a connection between them. The edge may be defined by an unordered pair, in which case the edge is *undirected*. If the pair is ordered, then the edges defined by (i, j) and (j, i) are distinct (a graph may contain both of these edges). In this case the edge is *directed*, so that the edge (i, j) points from i to j (i is a parent, j is the child). The number of parents (children) of a node is commonly referred to as the *indegree* (*outdegree*).

A graph G , then, is simply a set of nodes V , and a set of edges from V . A *path* is a sequence of nodes a_1, \dots, a_m with edges between consecutive elements. If each edge (a_i, a_{i+1}) in the path is directed with parent a_i , then the path is directed, otherwise it is undirected.

A graph consisting of (un)directed edges is an (un)directed graph, although a graph may include both types of edges. In a directed graph, if a directed path exists from node i to node j , then i is an *ancestor* of j and j is a descendant of i . A directed path which starts and stops at the same node is a *cycle*. *Directed acyclic graphs* (DAG) are an important class of graphs, defined as any directed graph which does not contain any cycles. A family tree is an example of a DAG. Figure 20.1 shows examples of each type. Note that graph b) contains a cycle involving nodes b, c, d, g , but graph c) contains no cycles.

Formally, a BN is a random vector $\tilde{X} = (X_1, \dots, X_n)$, and a DAG G with n nodes. The i th node is associated with the component X_i of \tilde{X} . The role played by G is subtle but crucial. Recall the definition of a Markov chain (Chapter 7, CSC262 lecture notes). It defines a sequence of random variables Z_1, Z_2, Z_3, \dots possessing the *memoryless property*

$$P(Z_{i+1} = a_{i+1} \mid Z_i = a_i, Z_{i-1} = a_{i-1}, \dots, Z_1 = a_1) = P(Z_{i+1} = a_{i+1} \mid Z_i = a_i),$$

where a_1, \dots, a_{i+1} are any values that the Markov chain may assume. Intuitively, this means that the distribution of a future state Z_{i+1} given the current state Z_i and the history Z_{i-1}, \dots, Z_1 depends only on the current state Z_i . In a sense, the current state separates the future and past states, formally, the future is independent of the past conditional on the present.

In a sense, a BN is a generalization of a Markov chain (and so a Markov chain is a special case of a BN). In graph c) of Figure 20.1 the direction of the edges suggests a sequential structure, with nodes a, b in one ‘generation’, then c , then d , then e, f, g forming subsequent generations (it

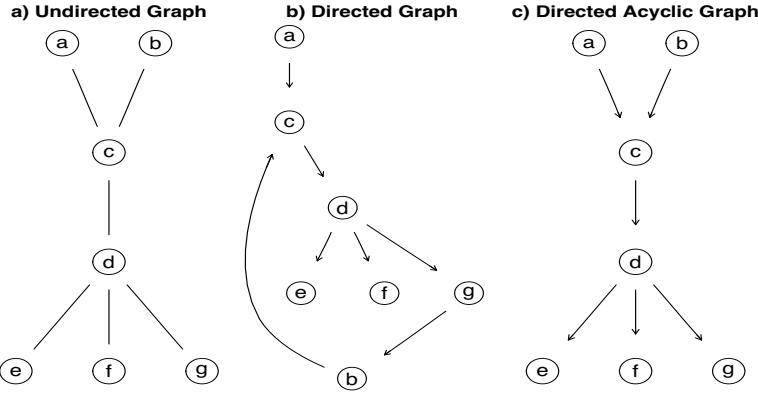


Figure 20.1: Types of graphs.

is helpful to think of a family tree in this context). There is, in fact, a version of the memoryless property for DAGs, expressed in terms of conditional independence.

Definition 20.1. A random vector $\tilde{X} = (X_1, \dots, X_n)$ satisfies the *local Markov property* with respect to a DAG G if each node X_i is independent of its non-descendants conditional on its parents (here X_i is used interchangeably with node i). In this case, (\tilde{X}, G) is a BN.

Compare Definition 20.1 to the phrase ‘*the future is independent of the past conditional on the present*’ used to define the memoryless property of a Markov chain. There are several ways to express the local Markov property. Consider the next definition:

Definition 20.2. Let G be a DAG, and let $\tilde{X} = (X_1, \dots, X_n)$ be a random vector associated with the nodes of G . The *Markov blanket* of node i is the set consisting of all parents of i ; all children of i ; and all parents of all children of i . If each node X_i is independent of all nodes outside its Markov blanket, conditional in its Markov blanket, then (\tilde{X}, G) is a BN (here X_i is used interchangeably with node i).

From a practical point of view, the advantage of the BN model is that it permits considerable simplification of the joint density $f(x_1, \dots, x_n)$. Suppose, given DAG G , we let S_i be the set of parents of node i . Then if f satisfied either definition of a BN with respect to G (the two are equivalent) then F may be factored by

$$f(x) = \prod_{i=1}^n f_i(x_i | x_j, j \in S_i), \quad (20.1)$$

where $f_i(x_i | x_j, j \in S_i)$ is the density of X_i conditional on $\{X_i : i \in S_i\}$. For example, a BN based on graph c) of Figure 20.1 can be factored as

$$f = f(e | d)f(f | d)f(g | d)f(d | c)f(c | a, b)f(a)f(b)$$

(simplifying notation somewhat). Note that the components $f(a), f(b)$ are the marginal densities, since they have no parents. We could also say that the parents set of a and b is \emptyset , and write $f(a | \emptyset) = f(a)$ and $f(b | \emptyset) = f(b)$.

20.1 Fitting BNs

The factorized form of (20.1) allows standard statistical methods to be used to fit BN models. The two most common choices are Gaussian (for continuous, normally distributed observations) or multinomial (for categorical data). For Gaussian models, $f_i(x_i | x_j, j \in S_i)$ can be obtained by linear regression, using the parental observations as dependent variables for each node. Otherwise, $f_i(x_i | x_j, j \in S_i)$ is estimated by tabulating conditional probabilities. A likelihood function can therefore be defined, and AIC or BIC scores used to select a model.

20.2 Equivalence Classes

BNs are associated with causality models, but it must be remembered that it is the conditional independence constraints, and not the direction of the edges, that imply causal hypotheses (although the two are clearly related). Consider the following definition.

Definition 20.3. In a DAG, a *v-structure* is a subgraph of the form $a \rightarrow b \leftarrow c$, that is, three nodes for which two are parents of the third, and the parents are not connected by an edge. A *topology* of a DAG is the underlying undirected graph, that is, the undirected graph obtained by converting all directed edges to undirected edges. Two DAGs are *equivalent*, or are members of the same *equivalence class* if they have the same v-structures and the same topology.

Two equivalent DAGs have identical conditional independence structure. In practice, this means that for a given data set, a likelihood, AIC or BIC score will be identical for BNs based on equivalent DAGs. In other words, observational data cannot be used to distinguish between equivalent DAGs.

20.3 Postscript

Much of the theoretical foundation of Bayesian networks is due to Judea Pearl (for example, Pearl (1988)). A quite comprehensive treatment of the subject can be found in Koller and Friedman (2009). In demonstration software file `BAYESIAN-NETWORKS.R` we introduce the R library `bnlearn()`, a remarkably comprehensive collection of functions and utilities supporting Bayesian network modeling. (Scutari, 2010). Additional exercises are given in Chapter 31. In addition, this author has published a discussion on Bayesian network modeling which makes uses of information theoretic methods (Almudevar, 2016). See discussion in Section 18.3).

Part III

Practice Problems

Chapter 21

Practice Problems - ANOVA

21.1 Exercises

Problem 21.1. Independent samples for $k = 3$ treatments are summarized in the table below. Assume sample j is from a normally distributed population with mean μ_j and fixed variance σ^2 .

	1	2	3	4	5	\bar{X}_i	S_i	n_i
Treatment 1	13.13	15.16	10.60	16.41	17.99	14.66	2.88	5
Treatment 2	9.04	6.94	9.30	9.00	10.69	8.99	1.34	5
Treatment 3	20.55	16.37	20.20	14.39	21.89	18.68	3.16	5

- (a) Construct an ANOVA table (fill in the 9 spaces in the ANOVA table below).

	SS	DF	MS	F
Treatment	_____	_____	_____	_____
Error	_____	_____	_____	
Total	_____	_____		

- (b) Use an F -test for null hypothesis $H_0 : \mu_i = \mu_j$ for all i, j . Use significance level $\alpha = 0.05$.

SOLUTION:

- (a) ANOVA table is

ANOVA Table				
	SS	DF	MS	F
Treatment	236.83	2.00	118.42	17.70
Error	80.27	12.00	6.69	
Total	317.11	14.00		

(b) $F = 17.702$. Reject H_0 if

$$F \geq F_{k-1, n-k, \alpha} = 3.885.$$

Therefore, reject the null hypothesis at a significance level $\alpha = 0.05$ (P -value = 0.0002631).

Problem 21.2. A new type of insecticide was tested against 3 standard alternatives. Each of the four insecticides was tested in 6 separate plots (requiring 24 separate plots). The percentage crop loss was recorded for each of the 24 plots at the end of the experiment. For each insecticide, the sample mean and sample standard deviation of the 6 outcomes is given in the following table (for example, the average percentage crop loss for the six plots using Standard Insecticide B was 9.91). The ANOVA table for the data is also given. Using a Bonferroni multiple comparison procedure, determine whether or not the new insecticide resulted in the lowest average percentage crop loss of all the insecticides tested. Use a family-wise error rate of $\alpha_{FWE} = 0.05$.

	\bar{X}_i	S_i	n_i
New Insecticide	5.17	3.34	6
Standard Insecticide A	10.65	3.04	6
Standard Insecticide B	9.91	3.20	6
Standard Insecticide C	9.88	2.51	6

	SS	DF	MS	F
Treatment	113.69	3	37.90	4.10
Error	184.65	20	9.23	
Total	298.35	23		

SOLUTION:

We need $m = 3$ comparisons, to compare $\mu_1 - \mu_i$, $i = 2, 3, 4$. Since $n_i = 6$ for $i = 1, 2, 3, 4$, the CIs take form

$$\begin{aligned} CI &= \bar{X}_i - \bar{X}_j \pm t_{n-k, \alpha_{FWE}/(m2)} \sqrt{MSE \left(\frac{1}{n_i} + \frac{1}{n_j} \right)} \\ &= \bar{X}_i - \bar{X}_j \pm t_{20, 0.05/6} \sqrt{9.23 \left(\frac{1}{6} + \frac{1}{6} \right)} \\ &= \bar{X}_i - \bar{X}_j \pm 2.613 \sqrt{9.23 \left(\frac{1}{6} + \frac{1}{6} \right)} \\ &= \bar{X}_i - \bar{X}_j \pm 4.58. \end{aligned}$$

The CIs are given in the following table. We can conclude with confidence $1 - \alpha_{FWE} = 0.95$ that μ_1 is the smallest mean, since $\mu_1 - \mu_i < 0$ for $i = 2, 3, 4$ within each comparison.

Multiple comparisons (Bonferroni procedure) for Problem 21.2.

	Treatment 1	Treatment 2	Difference	Margin of Error	LB	UB
Comp 1	1	2	-5.48	4.58	-10.06	-0.90
Comp 2	1	3	-4.73	4.58	-9.32	-0.15
Comp 3	1	4	-4.71	4.58	-9.30	-0.13

Problem 21.3. An ANOVA model is analyzed based on data for 4 treatments, of which each have n_j observations. An observation from treatment $j = 1, \dots, 4$ has distribution $N(\mu_j, \sigma^2)$, the variance being assumed constant. Observations are independent. The treatment means, standard deviations and sample sizes are given in a table below. The sum of squares (SS), mean sum of squares (MSS) and degrees of freedom for treatment and error sources of variation are given in the subsequent table.

Using a Bonferroni multiple comparison procedure, with a family-wise error rate of $\alpha_{FWE} = 0.15$, can we conclude that μ_4 is the minimum treatment mean?

Treatment j	\bar{X}_j	S_j	n_j
1	10.03	1.39	7
2	10.24	1.31	7
3	13.22	1.21	7
4	6.92	2.06	7

Source of Variation	DF	SS	MSS
Treatment	3	139.50	46.50
Error	24	56.15	2.34

SOLUTION:

We need $m = 3$ comparisons, to compare $\mu_4 - \mu_i$, $i = 1, 2, 3$. Since $n_i = 7$ for $i = 1, 2, 3, 4$ and $n = 28$, the CIs take form

$$\begin{aligned}
CI &= \bar{X}_i - \bar{X}_j \pm t_{n-k, \alpha_{FWE}/(m2)} \sqrt{MSE \left(\frac{1}{n_i} + \frac{1}{n_j} \right)} \\
&= \bar{X}_i - \bar{X}_j \pm t_{24, 0.15/6} \sqrt{2.34 \left(\frac{1}{7} + \frac{1}{7} \right)} \\
&= \bar{X}_i - \bar{X}_j \pm 2.064 \sqrt{2.34 \left(\frac{1}{7} + \frac{1}{7} \right)} \\
&= \bar{X}_i - \bar{X}_j \pm 1.684.
\end{aligned}$$

The CIs are given in the following table. We can conclude with confidence $1 - \alpha_{FWE} = 0.85$ that μ_4 is the smallest mean, since $\mu_4 - \mu_j < 0$ for $j = 1, 2, 3$ within each comparison.

Comparison	Estimate	ME	LB	UB
$\mu_4 - \mu_1$	-3.11	1.69	-4.80	-1.42
$\mu_4 - \mu_2$	-3.32	1.69	-5.01	-1.63
$\mu_4 - \mu_3$	-6.30	1.69	-7.99	-4.61

21.2 Data Analysis

Problem 21.4. For this question, use the `Carseats` data set from the `ISLR` package. This data set is simulated, but is intended to represent sales data from 400 different stores. We will make use of the variables:

- `Sales` = Unit sales (in thousands) at each location.
- `Population` = Population size in region (in thousands).
- `ShelveLoc` = A factor with levels `Bad`, `Good` and `Medium` indicating the quality of the shelving location for the car seats at each site.
- `Urban` = A factor with levels `No` and `Yes` to indicate whether the store is in an urban or rural location.

The objective is to determine how important shelf location is to sales volume, since securing advantageous shelf position requires considerable effort. Suppose μ_{bad} , μ_{med} , μ_{good} are the respective mean sales volumes for each shelf location category. Two hypotheses might be

$$H_1 : \mu_{good} > \mu_{med} > \mu_{bad}$$

or

$$H_2 : \mu_{good} > \mu_{med} = \mu_{bad}.$$

If H_1 were true, then it would be worth securing good shelf space, and if that were not possible, it would be worth securing medium shelf space over bad shelf space. On the other hand, if H_2 were true, then there would be nothing to gain by attempting to secure medium shelf space if good shelf space were not available.

Fit an ANOVA model then perform a post-hoc analysis using the following steps:

- The analysis will be done for rural stores only. Create a subset of the data including only records with factor level `Urban=='No'`. You can use the `subset()` function.
- An analysis should examine any association between the response and other variables. For example, it is possible that `Sales` is associated with `Population`, and should therefore be adjusted. Construct a scatterplot of the two variables and test for correlation (you can use the `cor.test()` function). What do you conclude?
- Construct side-by-side boxplots of `Sales` for the factor levels of `ShelveLoc`. Do a Bartlett's test for equality of variance of `Sales` across these factor levels (use `bartlett.test()`). Are the standard assumptions of normality and equality of variance reasonable in this case? You can draw your conclusions from the equality of variance test and the boxplots alone.

- (d) Fit an ANOVA model, and report a p -value for the rejection of the null hypothesis

$$H_0 : \mu_{good} = \mu_{med} = \mu_{bad}.$$

Describe precisely the test statistic, and its distribution under the null hypothesis.

- (e) Using Tukey's pairwise procedure (function `TukeyHSD()`) report confidence intervals for each pairwise difference in the means $\mu_{good}, \mu_{med}, \mu_{bad}$. Use a family-wise error rate of $\alpha_{FWE} = 0.01$. You will have to use the `conf.level` option. What can be said about the rankings of the means with confidence level 99%?
- (f) Suppose, anticipating that better shelf location will never result in *lower* sales volume, we attempt to resolve the problem by constructing confidence intervals for the differences $\mu_{good} - \mu_{med}$ and $\mu_{med} - \mu_{bad}$ only, using Bonferroni's procedure to attain a family-wise error rate of $\alpha_{FWE} = 0.01$. What can be said about the rankings of the means with confidence level 99%, using this procedure? Does this contradict the conclusion of Part (e)?

SOLUTION:

The analysis is given in the following code. Comments follow.

```
> par(mfrow=c(1,2),pty='s',oma=c(2,2,2,2),cex=1,cex.axis=0.85,cex.lab=0.85)
>
> library(ISLR)
>
> ### (a)
>
> Carseats2 = subset(Carseats, Urban=='No')
>
> ### (b)
>
> plot(Carseats2$Population,Carseats2$Sales,xlab='Sales',ylab='Population')
> cor.test(Carseats2$Population,Carseats2$Sales)
```

Pearson's product-moment correlation

```
data: Carseats2$Population and Carseats2$Sales
t = 0.80446, df = 116, p-value = 0.4228
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
-0.1077250  0.2518532
sample estimates:
      cor
0.07448475

>
> ###(c)
>
```

```

> boxplot(Sales~ ShelveLoc, data = Carseats2,xlab='Sales',ylab='Population')
> bartlett.test(Sales~ ShelveLoc, data = Carseats2)

Bartlett test of homogeneity of variances

data: Sales by ShelveLoc
Bartlett's K-squared = 0.53081, df = 2, p-value = 0.7669

>
> ### (d)
>
> fit = aov(Sales~ ShelveLoc, data = Carseats2)
> summary(fit)
      Df Sum Sq Mean Sq F value    Pr(>F)
ShelveLoc     2  253.5   126.75   21.83 9.16e-09 ***
Residuals   115  667.6     5.81
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> TukeyHSD(fit,conf.level=0.99)
  Tukey multiple comparisons of means
  99% family-wise confidence level

Fit: aov(formula = Sales ~ ShelveLoc, data = Carseats2)

$ShelveLoc
        diff      lwr      upr     p adj
Good-Bad  4.384156  2.34383768  6.424474 0.0000000
Medium-Bad 1.693610 -0.06293306  3.450152 0.0136136
Medium-Good -2.690546 -4.29861786 -1.082475 0.0000069

>
> ### (f)
>
> # Get the treatment sample size
>
> ni = table(Carseats2$ShelveLoc)
>
> # We need MSE, get this from the ANOVA table
>
> mse = summary(fit)[[1]][2,3]
>
> # We can get the estimated difference from the TukeyHSD object

```

```

>
> tr.diff = TukeyHSD(fit)$ShelveLoc[,1]
>
> # Assemble the margins of error
>
> t.crit = qt(1-0.01/4,df=sum(ni)-3)
> nh = c(1/ni[1]+1/ni[2],1/ni[1]+1/ni[3],1/ni[2]+1/ni[3])
> me = t.crit*sqrt(mse*nh)
>
> # We only need the last two comparisons
>
> cbind(tr.diff,tr.diff-me,tr.diff+me)[2:3,]
      tr.diff
Medium-Bad   1.693610  0.002130126  3.385089
Medium-Good -2.690546 -4.239054118 -1.142038
>
```

- (a) See code.
- (b) The correlation is $r = 0.07448475$. The p-value against the null hypothesis $H_0 : \rho = 0$ is $P = 0.4228$, so there is no significant correlation. The scatter plot otherwise shows no apparent association between Sales and Population (Figure 21.1).
- (c) Bartlett's test for equality of variance has p-value $P = 0.7669$ against the null hypothesis of equal variances. The boxplots suggest that the distributions for each treatment are symmetric, and of equal variances (Figure 21.1). Based on these diagnostics, the assumptions of normality and equal variance are reasonable.
- (d) See code. The test is based on the statistic $F = MST/MSE = 21.83$ (from the ANOVA table). Under H_0 , F has an F -distribution with 2 numerator and 115 denominator degrees of freedom. The p-value is $P = 9.16e-09$, so we have very strong evidence to reject H_0 .
- (e) See code. Assuming the confidence intervals are correct (ie contain the true difference in means), we can conclude that $\mu_{good} > \mu_{med}$ and $\mu_{good} > \mu_{bad}$. To summarize, with 99% confidence we can conclude that μ_{good} is the uniquely largest mean. Otherwise, we cannot claim anything regarding the ordering of μ_{med} and μ_{bad} .
- (f) See code. To the Bonferroni procedure, construct confidence intervals for mean differences $\mu_{good} - \mu_{med}$ and $\mu_{med} - \mu_{bad}$ of the form

$$\bar{y}_i - \bar{y}_j \pm t_{\alpha/(m2),n-k} \sqrt{MSE \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}$$

where $m = 2$, $\alpha = 0.01$, $k = 3$, $n_{good} = 28$, $n_{med} = 68$, $n_{bad} = 22$, $n = 28 + 68 + 22 = 118$. Assuming the confidence intervals are correct (ie contain the true difference in means), we can conclude that $\mu_{med} > \mu_{bad}$ and $\mu_{med} < \mu_{good}$. To summarize, with 99% confidence we can conclude that $\mu_{good} > \mu_{med} > \mu_{bad}$. This does not contradict the conclusion of Part (e). If a confidence interval for a difference in means contains zero, it means that the ordering cannot be resolved, and not that the means are equal. So, the conclusion of Part (f) is simply more precise.

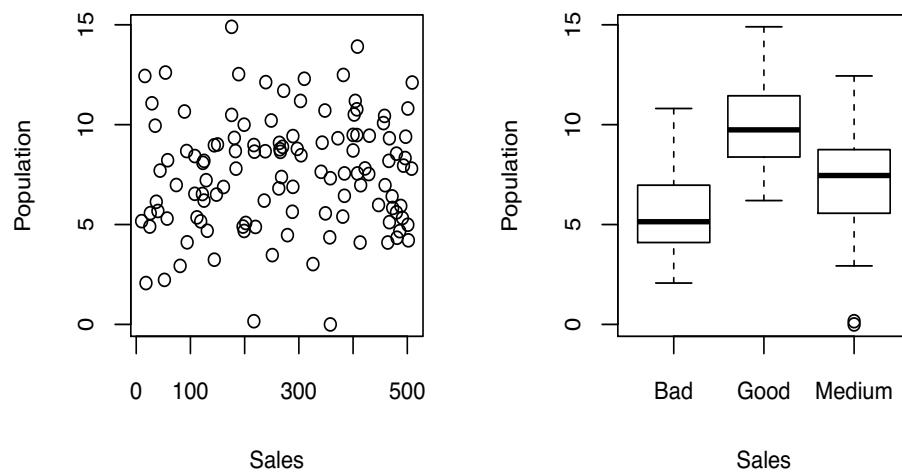


Figure 21.1: Plots for Problem 21.4.

Chapter 22

Practice Problems - Linear Regression

22.1 Exercises

Problem 22.1. Two variables Y and X are believed to have the following relationship:

$$Y = aX^b$$

for two constants a, b . According to a certain conjecture, Y is proportional to the square root of X . In order to resolve this question paired observations $(X_1, Y_1), \dots, (X_n, Y_n)$ are sampled, where $n = 51$. The simple linear regression model

$$\log(Y) = \beta_0 + \beta_1 \log(X) + \epsilon$$

is fit, with the following output:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.3137646	0.02512828	92.078108	1.447613e-56
log(x)	0.4985705	0.05774591	8.633867	2.084971e-11

Formulate appropriate null and alternative hypotheses for this question in terms of the regression coefficients β_0 and/or β_1 . Is there evidence at an $\alpha = 0.05$ significance level with which to reject the conjecture?

SOLUTION:

The hypotheses are

$$H_o : \beta_1 = 1/2 \text{ against } H_a : \beta_1 \neq 1/2.$$

The appropriate t -statistic is

$$T = \frac{\hat{\beta}_1 - 1/2}{SE_{\hat{\beta}_1}} = \frac{0.4985705 - 1/2}{0.05774591} \approx -0.0247.$$

Since $|T| < t_{49,0.025}$ we do not reject the conjecture at significance level $\alpha = 0.05$.

Problem 22.2. The following full linear regression model is considered:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

An all-subsets model selection procedure is to be used to determine which of the predictors and interactions to retain. The relevant SSE values are given in the following table. The sample size is $n = 25$. Which model possesses the largest coefficient of determination R^2 ? Which model possesses the largest adjusted coefficient of determination R_{adj}^2 ?

Model	SSE
1 y=1	10638.06
2 y=x1	7810.40
3 y=x2	2210.16
4 y=x1+x2	29.23
5 y=x1+x2+x1*x2	27.39

SOLUTION:

The model with the highest R^2 must be model 5, since all other models are reduced models.

The total sum of squares is given by the null model (model 1):

$$SSTO = 10638.06$$

The formula is

$$R_{adj}^2 = 1 - \frac{SSE/(n - (q + 1))}{SSTO/(n - 1)}.$$

where q is the number of predictors. We can construct table:

Model	SSE	q	R_{adj}^2
1 y=1	10638.06	0	0.00000
2 y=x1	7810.40	1	0.23388
3 y=x2	2210.16	1	0.78321
4 y=x1+x2	29.23	2	0.99700
5 y=x1+x2+x1*x2	27.39	3	0.99706

Model 5 has the highest R_{adj}^2 .

Problem 22.3. Two variables Y and X are believed to have the following relationship:

$$Y = aX^b$$

for two constants a, b . There is special interest in knowing whether or not this relationship is concave (equivalently, $b < 1$). In order to resolve this question paired observations $(X_1, Y_1), \dots, (X_n, Y_n)$ are sampled, where $n = 34$. The simple linear regression model

$$\log(Y) = \beta_0 + \beta_1 \log(X) + \epsilon$$

is fit, with the following output:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.7115358	0.09647197	7.37557	2.174990e-08
log(x)	0.8948970	0.03583328	24.97391	1.483693e-22

Formulate appropriate null and alternative hypotheses for this question in terms of the regression coefficients β_0 and/or β_1 . Is there evidence at an $\alpha = 0.05$ significance level that $b < 1$?

SOLUTION:

The appropriate hypotheses are $H_o : b \geq 1$ and $H_a : b < 1$, since we are looking for evidence that $b < 1$. In terms of the regression coefficients this is equivalent to

$$H_o : \beta_1 \geq 1 \text{ and } H_a : \beta_1 < 1.$$

The test statistic is

$$T = \frac{\hat{\beta}_1 - 1}{S_{\hat{\beta}_1}} = \frac{0.8948970 - 1}{0.03583328} = -2.933112,$$

which has a t -distribution with $n - 2$ degrees of freedom under H_o . Since $t_{32,0.05} = 1.69$, we reject H_o at a 0.05 significance level, and conclude that $b < 1$.

Problem 22.4. A client hires a consulting firm to conduct a study of two types of mutual funds (we'll call them simply Type A and Type B). It uses a simple regression model

$$Y = e^{\beta_0 + \beta_1 X + \epsilon}$$

where $X = 1$ for a Type A mutual fund, and $X = 0$ otherwise; $\epsilon \sim N(0, \sigma^2)$; and Y is the value of an original investment of \$1 after a year (that is, if $Y = 1.05$, the yearly rate of return is 5%). The model is first log-transformed, giving

$$\log(Y_i) = \beta_0 + \beta_1 X_i + \epsilon_i, \quad i = 1, \dots, n. \quad (22.1)$$

A random sample of $n = 62$ paired observations (Y_i, X_i) , $i = 1, \dots, 62$ is collected. A simple least squares regression model is used to fit the model (22.11), producing the following coefficient table:

Coefficient	Estimate	Standard Error	t-value	Pr(> t)
$\hat{\beta}_0$	0.0745	0.0040	18.8376	2.43×10^{-34}
$\hat{\beta}_1$	0.0333	0.0056	5.9514	4.13×10^{-8}

The consultant believes Type A mutual funds have a higher average yield, but the client currently purchases mutual funds of Type B, and there would be a significant cost to switching to Type A. Therefore, the consultant will only recommend switching to Type A if there is significant statistical evidence that $\beta_1 > 0.015$ (approximately, that the rate of return of Type A mutual finds exceeds Type B mutual funds by more than 1.5%). Using a significance level of $\alpha = 0.05$, can the consultant recommend switching?

SOLUTION:

The hypotheses are

$$H_0 : \beta_1 \leq 0.015 \text{ against } H_a : \beta_1 > 0.015.$$

The appropriate t -statistic is

$$T = \frac{\hat{\beta}_1 - 0.015}{SE_{\hat{\beta}_1}} = \frac{0.0333 - 0.015}{0.0056} \approx 3.268.$$

Since $T > t_{60,0.05} = 1.671$ we do not reject the conjecture at significance level $\alpha = 0.05$.

Problem 22.5. There is often interest in determining whether or not two quantities X and Y have a *power-law* relationship:

$$Y = aX^b \quad (22.2)$$

for two constants a, b . Suppose, given $n = 41$ independent paired observations (X_i, Y_i) of these quantities, we fit model

$$\log(Y_i) = \beta_0 + \beta_1 \log(X_i) + \beta_2 \log(X_i)^2 + \epsilon_i, \quad i = 1, \dots, n,$$

assuming any relevant distributional assumption holds, and get output:

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	38.33180	19.50917	1.965	0.0568 .
log.x	-5.93521	2.67072	-2.222	0.0323 *
log.x.squared	0.02227	0.08868	0.251	0.8031
<hr/>				
Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .
	0.1	'	'	1

- (a) How can the output be used to assess the validity of the power-law relationship of Equation (22.2)? What do you conclude?
- (b) Give level 95% confidence intervals for parameters a and b , based on this output.

SOLUTION:

- (a) If we take a log-transform of the model $\log(Y) = \log(a) + b \log(X)$. We can therefore equate $\beta_0 = \log(a)$ and $\beta_1 = b$ in the linear model. In addition, if the model holds we must have $\beta_2 = 0$. The 2-sided p -value for rejecting null hypothesis $H_0 : \beta_2 = 0$ is $p = 0.8031$ from the output. We do not reject the power-law relationship, therefore.
- (b) Since $\beta_0 = \log(a)$ and a 95% CI for β_0 is

$$\hat{\beta}_0 \pm 2 \times SE = 38.33180 \pm 2 \times 19.50917 = (-0.68654, 77.35014)$$

a 95% CI for a is

$$e^{\hat{\beta}_0 \pm 2 \times SE} = e^{38.33180 \pm 2 \times 19.50917} = (0.5033, 3.9 \times 10^{33}).$$

Since $\beta_1 = b$ a 95% CI for b is identical to the 95% CI for β_1 :

$$\hat{\beta}_0 \pm 2 \times SE = -5.93521 \pm 2 \times 2.67072 = (-11.27665, -0.59377).$$

22.2 Data Analysis

Problem 22.6. For this question, use the data set `UScereal` from the `MASS` package. This data contains ingredient quantities taken from the mandatory FDA label printed on 65 brands of cereal. The objective is to determine the ingredient that contributes most to calorie content.

- (a) Create side-by-side boxplots of `calories` by manufacturer (given by the categorical variable `mfr`). Identify two outliers (defined as `calories > 300`), and delete from the data all cereals made by the manufacturer responsible for those outliers.
- (b) Fit a linear regression model using `calories` as the dependent variable, and all remaining variables as independent variables. This is easily done using the formula `lm(calories ~ ., data = myData)`. Which variables have regression coefficients significantly different from 0 (at a $P < 0.05$ significance level)?
- (c) Refit the model, again with `calories` as dependent variable, but including as independent variables only those with significantly nonzero coefficients reported in Part (b). Include the intercept. Do these independent variables remain significant? Do the values of the coefficients change significantly?
- (d) Construct side-by-side boxplots for the independent variables of Part (c). In what units are these variables (use `help(UScereal)`)? Which regression coefficient is largest (other than the intercept). Looking at the boxplots, does the independent variable with the largest coefficient necessarily contribute most to calorie content?
- (e) Standardize each of the independent variables of the model fit in Part (c) to have zero mean and standard deviation one. Refit the model. Have the t -statistics reported for each independent variable changed? Which predictor contributes most to calorie content?

SOLUTION:

The code for the analysis is given below.

- (a) See Figure 22.1 for boxplots. The manufacturer responsible for those outliers is labeled P (for Post).
- (b) The significant independent variables are `protein`, `fat`, `carbo` and `sugars`.
- (c) The fit is shown below. The independent variables remain significant. The respective coefficient values in the original fit for these variables are 3.983205, 9.424864, 4.002802, 4.180946. The new values are 3.7978, 8.4661, 4.0402, 4.2139. They are not identical between fits, but are otherwise quite close.

- (d) The units used for the four independent variables are grams (in one portion). The largest regression coefficient is associated with `fat`. However, from the boxplots of Figure 22.1 (right side) the variation of `fat` is smaller than for the remaining independent variables. Therefore, the independent variable with the largest coefficient does not necessarily contribute most to calorie content.

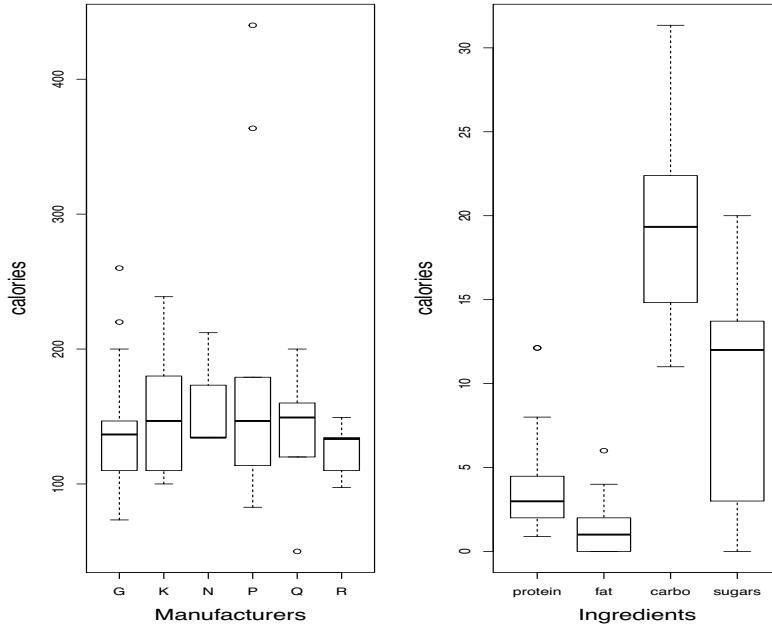


Figure 22.1: Plots for Problem 22.6 (d).

- (e) The t -statistics are identical. To compare calorie contributions, the coefficients should be calculated with respect to a change in unit standard deviation for each ingredient. Therefore, `sugars` contributes most to calorie content, with $\hat{\beta}_{\text{sugar}} = 24.8079$ change in calories per standard deviation change in `carbo`.

```
> library(MASS)
> par(mfrow=c(1,2))
>
>
> ### (a)
>
> boxplot(calories~mfr,data=UScereal,ylab='calories',xlab='Manufacturers',cex.lab=1.5)
> data2 = subset(UScereal, mfr!='P')
>
> ### (b)
>
> fit3 = lm(calories ~ ., data=data2)
> summary(fit3)
```

```

Call:
lm(formula = calories ~ ., data = data2)

Residuals:
    Min      1Q  Median      3Q     Max 
-14.8371 -3.1176  0.1532  3.2058 12.8722 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.667136  5.836452 -0.457   0.650    
mfrK          3.706381  2.368999  1.565   0.125    
mfrN          7.375987  7.187820  1.026   0.311    
mfrQ         -2.497673  3.474514 -0.719   0.476    
mfrR         -0.452942  3.567393 -0.127   0.900    
protein       3.983205  0.862820  4.616 3.82e-05 ***  
fat           9.424864  0.910656 10.350 5.30e-13 ***  
sodium        0.005558  0.010999  0.505   0.616    
fibre          0.668536  0.766720  0.872   0.388    
carbo          4.002802  0.222710 17.973 < 2e-16 ***  
sugars         4.180946  0.238287 17.546 < 2e-16 ***  
shelf          0.427604  1.463232  0.292   0.772    
potassium     -0.035009  0.030312 -1.155   0.255    
vitaminsenriched -0.274217  3.338727 -0.082   0.935    
vitaminsnone   -1.765723  7.271831 -0.243   0.809    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 5.992 on 41 degrees of freedom
 Multiple R-squared: 0.9862, Adjusted R-squared: 0.9815
 F-statistic: 209.9 on 14 and 41 DF, p-value: < 2.2e-16

```

>
> ### (c)
>
> data3 = data2[,c("calories","protein","fat","carbo","sugars")]
> fit3 = lm(calories ~ ., data=data3)
> summary(fit3)

```

```

Call:
lm(formula = calories ~ ., data = data3)

Residuals:
    Min      1Q  Median      3Q     Max 
-16.6047 -4.0734  0.1079  3.8670 13.4827 

```

```

Min      1Q  Median      3Q     Max 
-16.6047 -4.0734  0.1079  3.8670 13.4827 

```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.2281	3.3515	-0.366	0.716
protein	3.7978	0.3753	10.118	8.62e-14 ***
fat	8.4661	0.7449	11.365	1.37e-15 ***
carbo	4.0402	0.1548	26.105	< 2e-16 ***
sugars	4.2139	0.1639	25.713	< 2e-16 ***

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

Residual standard error: 5.924 on 51 degrees of freedom

Multiple R-squared: 0.9833, Adjusted R-squared: 0.982

F-statistic: 749.3 on 4 and 51 DF, p-value: < 2.2e-16

```

>
> ### (d)
>
> boxplot(data3[-1], ylab='calories', xlab='Ingredients', cex.lab=1.5)
>
> ### (e)
>
> data4 = data3
> for (i in 2:5) {data4[[i]] = (data4[[i]] - mean(data4[[i]]))/sd(data4[[i]])}
> fit4 = lm(calories ~ ., data=data4)
> summary(fit4)

```

Call:

```
lm(formula = calories ~ ., data = data4)
```

Residuals:

Min	1Q	Median	3Q	Max
-16.6047	-4.0734	0.1079	3.8670	13.4827

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	142.1199	0.7916	179.53	< 2e-16 ***
protein	9.2881	0.9179	10.12	8.62e-14 ***
fat	11.6437	1.0245	11.37	1.37e-15 ***
carbo	22.4673	0.8606	26.11	< 2e-16 ***
sugars	24.8079	0.9648	25.71	< 2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.924 on 51 degrees of freedom

```
Multiple R-squared:  0.9833, Adjusted R-squared:  0.982
F-statistic: 749.3 on 4 and 51 DF,  p-value: < 2.2e-16
```

>

Problem 22.7. For this question we will explore the `influence.measures()` function. The input is a fitted model object. In our example this will be a class `lm` object. It returns a `infl` class object, which has list elements `infmat`, `is.inf`, `call`. Here, `infmat` is a matrix with n rows, with columns containing a number of influence measures. They are as follows:

- (i) Given q predictors (including the intercept), the first q columns give the quantities $DFBETAS_{ij}$ in row i column j .
- (ii) The next column gives $DFFITS_i$.
- (iii) The remaining columns give the covariance ratio, Cook's distance D_i and leverage H_{ii} .

Then `is.inf` is a logical matrix of the same dimensions as `infmat`. The i, j th entry of `is.inf` is TRUE if the i, j th entry of `infmat` indicates that observation i has been flagged as an influential point according to the diagnostic measure of column j . Finally `call` gives the unevaluated expression which produced the fit (this is a type of object of mode `call`).

We will make use of the `birthwt` data set from the `MASS` package.

- (a) Do a simple linear regression fit of response birth weight (`bwt`) against predictor age (`age`).
- (b) Apply the `influence.measures()` function to the fit.
- (c) Create a plot with the following elements (use a comparable scheme if you would rather create a black and white graphic):
 - (i) The plot contains a scatter plot of `bwt` against `age`. Each point should be represented using a solid circle (`pch = 20`).
 - (ii) The fitted regression line should be superimposed on the scatter plot.
 - (iii) A point should be black, unless it is flagged as a high leverage point, in which case it should be red.
 - (iv) A point flagged by $DFFITS$ should have a triangle (pointing up) surrounding it (`pch = 2`). The triangle should be blue if $DFFITS < 0$ and green otherwise.
 - (v) A point flagged by the covariance ratio value should have a triangle (pointing down) surrounding it (`pch = 6`). The triangle should be blue if the covariance ratio is < 1 and green otherwise.
- (d) What distinguishes high (> 0) from low (< 0) $DFFITS_i$ values?
- (e) Is there any flag criterion which dominates, that is, one which is flagged when any of the others is flagged (if the answer is yes, this might, or might not, mean that we only need that criterion)?
- (f) What distinguishes high (> 1) from low (< 1) covariance ratio values? Refer to the plot.
- (g) If we can select the values of the predictor variable, we refer to this as a design, which is ideally guided by some objective criterion. Suppose we wish to estimate the relationship $Y = \beta_0 + \beta_1 X + \epsilon$ and, under the standard assumptions for linear regression, we are able to

observe $n = 10$ responses Y_1, \dots, Y_{10} for any 10 *design points* X_1, \dots, X_{10} we wish, provided $X_i \in [0, 1]$. We'll compare two designs:

$$\begin{aligned} X' &= (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0), \\ X'' &= (0, 0, 0, 0, 0, 1, 1, 1, 1, 1). \end{aligned}$$

The comparison can be based on the variance of the coefficient estimates $\hat{\beta}_0, \hat{\beta}_1$. Carry out this comparison. Does either design yield uniformly lower variances?

- (h) Examine the form of the variances $\sigma_{\hat{\beta}_1}^2$ and $\sigma_{\hat{\beta}_0}^2$. What effect does the sample variance of the predictor variable have on these quantities?
- (i) Would it be a good idea to rely exclusively on the covariance ratio to flag anomalies? Justify your answer.

SOLUTION:

- (a) The following code implements parts (a)-(c). See Figure 22.2 below.

(b) etc

(c) etc

```
# get fit and influence measures

n = dim(birthwt)[1]
fit0 = lm(bwt~age,data=birthwt)
mm = influence.measures(fit0)

# this tells us how many observations are flagged by each diagnostic
apply(mm$is.inf,2,sum)

# create a variable color vector

colv = rep(1,n)
colv[mm$is.inf[,6]] = 2

# we can draw the plot this way (note use of the with() function)

par(mfrow=c(1,1))

# main plots (note use of color vector)

with(birthwt,plot(age,bwt,col=colv,pch=20))

# DFFITS diagnostic
```

```

ind = mm$is.inf[,3]
with(birthwt,points(age[ind],bwt[ind],col= 3+(mm$infmat[ind,3] < 0),pch=2))

# cov.ratio diagnostic

ind = mm$is.inf[,4]
with(birthwt,points(age[ind],bwt[ind],col= 3+(mm$infmat[ind,4] < 1),pch=6))

# include legend

legend('bottomright',legend=c('High Leverage','Low DFFITS',
    'High DFFITS','Low cov.ratio','High cov.ratio'),
    pch=c(20,2,2,6,6),col=c(2,4,3,4,3))

# finally, include regression fit

summary(fit0)
coef = fit0$coefficients
abline(coef[1],coef[2],col='green')

```

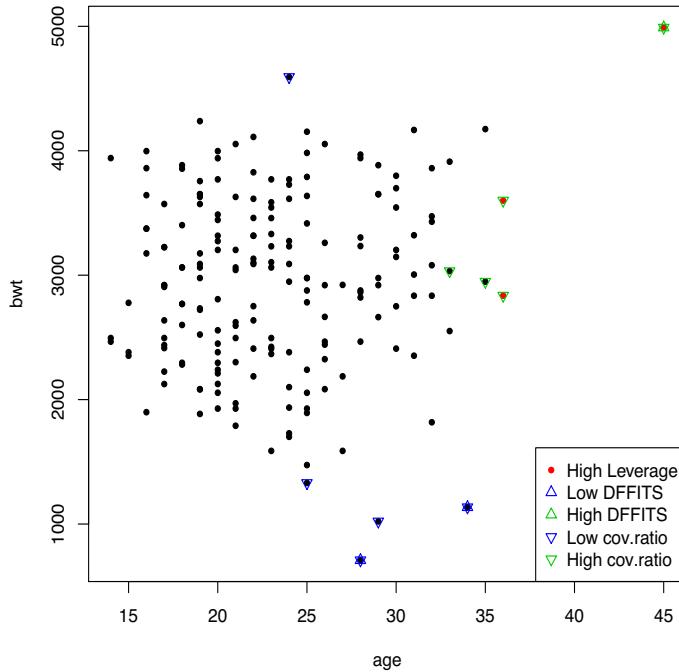


Figure 22.2: Figure for Problem 22.7 (c).

- (d) A high (low) $DFFITS_i$ value indicates that the fitted value at the predictor values of the i th observation is higher (lower) than would be the case if the i th observation were deleted.

- (e) All flagged observations are flagged at least by the covariance ratio (this can be seen in the plot).
- (f) High (low) covariance ratio values indicate that deleting an observation increases (decreases) the standard errors of the coefficient estimates $\hat{\beta}_i$.
- (g) The covariance matrix for $\hat{\beta}$ is

$$\Sigma_{\hat{\beta}} = \sigma^2(\mathbf{X}^T \mathbf{X})^{-1}. \quad (22.3)$$

Since σ^2 remains unchanged for the comparison, we evaluate $(\mathbf{X}^T \mathbf{X})^{-1}$ for X' and X'' . We can use the R code below. The diagonal elements of each matrix are proportional to $\sigma_{\hat{\beta}_i}^2$. They are uniformly smaller for X'' .

```
> ### This gives the inverse of t(x)%*%x for each design,
>           which suffices to resolve the question
>
> # X'
>
> x = cbind(rep(1,11),seq(0,1,0.1))
> solve(t(x)%*%x)
      [,1]      [,2]
[1,]  0.3181818 -0.4545455
[2,] -0.4545455  0.9090909
>
> # X"
>
> x = cbind(rep(1,10),rep(c(0,1),each=5))
> solve(t(x)%*%x)
      [,1]  [,2]
[1,]  0.2 -0.2
[2,] -0.2  0.4
```

- (h) The expressions are:

$$\sigma_{\hat{\beta}_0}^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right] \quad (22.4)$$

$$\sigma_{\hat{\beta}_1}^2 = \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}. \quad (22.5)$$

In each case the variances decrease with increasing values of $\sum_{i=1}^n (X_i - \bar{X})^2$, which is directly proportional to the sample variance of the predictor variable X .

- (i) The point of parts (g) and (h) is that the standard errors of the regression coefficients is directly affected by the distribution of the predictor variable. So, high values of the covariance ratio may be associated with observations at the outer limits of the predictor value range, but these need not be anomalous in any meaningful sense, and their deletion would simply result in a less accurate estimate.

Problem 22.8. For this question, use the data set `birthwt` data set from the MASS package used in 22.7.

- (a) Create a subset of this data by removing all observations flagged by the *DFFITTS* diagnostic calculated in 22.7. Use the `subset()` function.
- (b) We will examine 6 regression models, using `bwt` as a response and `age` and `smoke` as predictors. Note that `smoke` is an binary variable. Formally, it a numeric vector, but it should be interpreted as an indicator variable, with `smoke` = 1 if the subject smokes. The models are

$$\begin{aligned}
 (M1) \quad bwt &= \beta_0 + \beta_1 \times age \\
 (M2) \quad bwt &= \beta_0 + \beta_1 \times smoke \\
 (M3) \quad bwt &= \beta_0 + \beta_1 \times smoke + \beta_2 \times age \\
 (M4) \quad bwt &= \beta_0 + \beta_1 \times smoke + \beta_2 \times smoke \times age \\
 (M5) \quad bwt &= \beta_0 + \beta_1 \times smoke + \beta_2 \times (1 - smoke) \times age \\
 (M6) \quad bwt &= \beta_0 + \beta_1 \times smoke + \beta_2 \times age + \beta_3 \times smoke \times age
 \end{aligned}$$

Model (M1) does not distinguish by smoking group. Otherwise, the remaining models essentially create separate linear fits for the two groups, but with a pooled estimate for σ^2 . What distinguishes the models is the inclusion or exclusion of an `age` term for each group. Create a linear regression fit for each model. Construct a table with a row for each model, and columns as follows:

- (i) Columns 1-2 should contain the intercept and slope for the nonsmoking group (the slope may be 0).
- (ii) Columns 3-4 should contain the intercept and slope for the smoking group (the slope may be 0).
- (iii) Column 5 should contain R_{adj}^2 .
- (iv) Columns 6-9 should contain the F statistic, numerator d.f., denominator d.f., and p -value for a goodness-of-fit F -test comparing each model to the reduced model $Y = \beta_0 + \epsilon$.

It is recommended that the fitting of models is automated by constructing a suitable R function. Also, the quantities in columns 1-4 of the of the summary tables will be linear combinations of the coefficients estimated by the `lm()` function. For each model a single $4 \times q$ matrix can be constructed which will calculate all four quantities at once. A good strategy would therefore be to construct a list of 6 model formula, and a list of 6 coefficient transformation matrices. When creating formula, consider the following passage from the R documentation from `help(formula)`:

To avoid this confusion, the function `I()` can be used to bracket those portions of a model formula where the operators are used in their arithmetic sense. For example, in the formula `y ~ a + I(b+c)`, the term `b+c` is to be interpreted as the sum of `b` and `c`.

- (c) For each model construct a scatter-plot of `bwt` against `age`. Use something like `par(mfrow=c(3,2))` to display all plots on a single page. Superimpose the linear relationship separately for smokers and nonsmokers. Use appropriate coloring or symbols, with a legend, to distinguish between the two. Make sure the model represented by each plot is clearly indicated.

- (d) Which model has the highest R^2_{adj} ? Write explicitly the estimated linear relationship between `bwt` and `age` separately for smokers and nonsmokers.
- (e) Of the remaining models, which is the submodel of the one identified in the previous part with the largest number of model degrees of freedom (which equals the number of coefficients to be estimated)? Include the null model $Y = \beta_0 + \epsilon$ if necessary. Do a goodness-of-fit test F -test to compare the two models (you can use the `anova()` function for this). Does the conclusion yield the same conclusion as the R^2_{adj} ranking?

SOLUTION:

- (a) The following code implements parts (a)-(c). See Figure 22.3 below.
 (b) etc
 (c) etc

```
# function for F-test summary

f0 = function(lm.obj) {
  c(summary(lm.obj)$fstatistic,
    1-pf(summary(lm.obj)$fstatistic[1],summary(lm.obj)$fstatistic[2],
          summary(lm.obj)$fstatistic[3]))
}

# This will remove observations flagged by DFFITS

birthwt.sub = subset(birthwt,!mm[[2]][,3] )

# list of 5 models to be fit, in formula representation

model.list = c(
  bwt~age,
  bwt~smoke,
  bwt~smoke + age,
  bwt~ smoke + smoke:age,
  bwt~ smoke + I(1-smoke):age,
  bwt~smoke*age)

# list of linear transformations needed to calculate slopes and
#   intercepts for the smoking and nonsmoking groups

lc.list = list(
  matrix(c(1,0, 0,1, 1,0, 0,1),nrow=4,byrow=T),
  matrix(c(1,0, 0,0, 1,1, 0,0),nrow=4,byrow=T),
  matrix(c(1,0,0, 0,0,1, 1,1,0, 0,0,1),nrow=4,byrow=T),
```

```

matrix(c(1,0,0, 0,0,0, 1,1,0, 0,0,1),nrow=4,byrow=T),
matrix(c(1,0,0, 0,0,1, 1,1,0, 0,0,0),nrow=4,byrow=T),
matrix(c(1,0,0,0, 0,0,1,0, 1,1,0,0, 0,0,1,1),nrow=4,byrow=T)
)

# create summary table

beta.mat = matrix(NA,6,9)

# combine plots and calculations into one loop.

pdf('FIGA1.pdf')
par(mfrow=c(4,2))
for (i in 1:6) {

  with(birthwt.sub,plot(age,bwt,col=3-smoke,pch=20))
  fit0 = lm(model.list[[i]],data=birthwt.sub)
  coef = fit0$coefficients
  betav = lc.list[[i]]%*%coef
  if (i ==1) {
    abline(betav[1],betav[2],col='black',lwd=2)
  } else {
    abline(betav[1],betav[2],col='green',lwd=2)
    abline(betav[3],betav[4],col='red',lwd=2)
  }
  beta.mat[i,] = c(betav,summary(fit0)$adj.r.squared,f0(fit0))
  title(deparse(model.list[[i]]))
}

plot(c(0,1),c(0,1),type='n',axes=F,xlab=NA,ylab=NA)
legend('topleft',legend=c('Smoke -ve','Smoke +ve','All'),
       col=c(3,2,1),pch=c(20,20,NA),lty=c(1,1,1),cex=1)
dev.off()

# label the summary table

rownames(beta.mat) = paste('M',1:6,sep='')
colnames(beta.mat) = c('b0[smoke=0]', 'b1[smoke=0]', 'b0[smoke=1]',
                      'b1[smoke=1]', 'Rsq.adj', 'F', 'num.df', 'den.df', 'P')

```

(d) The table is given below

```

> round(beta.mat,3)
   b0[smoke=0] b1[smoke=0] b0[smoke=1] b1[smoke=1] Rsq.adj      F num.df den.df      P
M1     2700.092     11.080    2700.092     11.080    0.001 1.211      1     184 0.273
M2     3038.728      0.000    2823.306      0.000    0.018 4.410      1     184 0.037
M3     2805.090     10.055    2594.842     10.055    0.018 2.711      2     183 0.069

```

M4	3038.728	0.000	2898.994	-3.331	0.013	2.214	2	183	0.112
M5	2624.257	17.837	2823.306	0.000	0.024	3.230	2	183	0.042
M6	2624.257	17.837	2898.994	-3.331	0.018	2.155	3	182	0.095

Model (M5) has the largest R^2_{adj} . We have

$$bwt = \beta_0 + \beta_2 \times age = 2624.257 + 17.837 \times age, \text{ for nonsmokers} \quad (22.6)$$

$$bwt = \beta_0 + \beta_1 = 2823.306, \text{ for smokers.} \quad (22.7)$$

(e) We can use the R code

```
> anova(fit0,fit1)
Analysis of Variance Table

Model 1: bwt ~ smoke
Model 2: bwt ~ smoke + I(1 - smoke):age
  Res.Df   RSS Df Sum of Sq    F Pr(>F)
1     184 85444096
2     183 84509173  1    934922 2.0245 0.1565
>
```

The model (M2) is a submodel of (M5), and represents the null hypothesis for the F -test. Rejection is interpretable as evident that (M5) is more predictive than (M2). The P -value is 0.1565, so we don't have evidence that (M5) is the better model. This does not conform to the selection of (M5) using R^2_{adj} .

Problem 22.9. Relationships between the size of two physiological components Y , X of a species of animal often obey a power relationship

$$Y = KX^r,$$

where K and r are two fixed constants. Of course, the value r is not necessarily $r = 1$, but will depend on the size measure, and any number of scaling principles. Suppose we have paired observations (X_i, Y_i) , $i = 1, \dots, n$. Then K and r may be estimated using simple linear regression, after taking the double-log transformation:

$$\log Y_i = \log K + r \log X_i, \quad (22.8)$$

that is, we have intercept and slope $\beta_0 = \log K$, $\beta_1 = r$.

For this problem use data set **Animals** from the MASS package, which contains X = average body (kg) and Y = brain (g) weights for 28 species of land animals. We may expect Y to be positively associated with an animal's cognitive abilities. However, we also expect X and Y to be positively associated for reasons having nothing to do with cognitive abilities. Thus, the *encephalization quotient* (EQ) measures the relative brain size after controlling for body size.

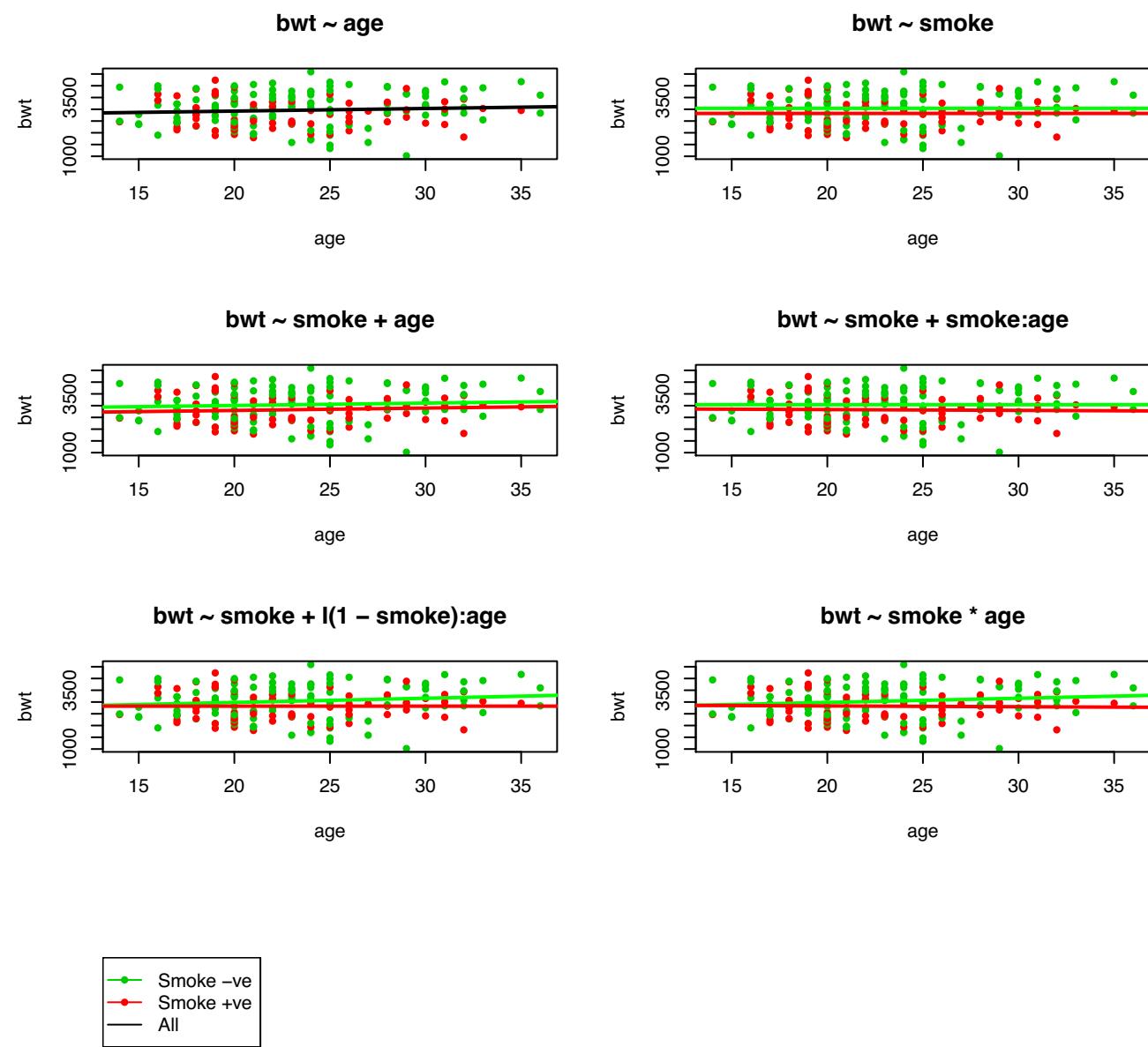


Figure 22.3: Figure for Problem 22.8 (c)

- (a) Construct a scatter-plot of $\log \text{Brain}$ against $\log \text{Body}$. Use $\log \text{Body}$ as the horizontal axis. Instead of plotting symbols, plot the actual name of the species (here, the `text` command may be used). Does there seem to be a linear trend on the double-log scale? Identify the three most obvious outliers. How do they differ from the remaining species?
- (b) Fit the model (22.8), with and without the outliers, and superimpose each fitted line on the scatter-plot. Give the estimates of K and r for each fit.
- (c) The *encephalization quotient* (EQ) can be formally defined as the ratio of the actual brain mass to the predicted brain mass based on the species size. If model (22.8) is used to predict brain mass, show that for species i

$$EQ \approx \exp(e_i)$$

where e_i is the residual from the regression fit for that species.

- (d) After removing the outliers, rank the species by their EQ. How would the EQ of the outlier species rank?

SOLUTION:

- (a) The following code will produce the required plot (Figure 22.4). There seems to be a clear linear trend, with the exception of three outliers, which are all dinosaurs.

```
par(mfrow=c(1,1),pty='m')
with(Animals,plot(log(body),log(brain),type='n',xlim=c(-4,13)))
with(Animals,text(log(body),log(brain),rownames(Animals),cex=0.9))
```

- (b) The following code will produce the required plot (Figure 22.5) and calculations. There seems to be a clear linear trend, with the exception of three outliers, which are all dinosaurs. From the fit summaries we have $\log \hat{K} = 2.55490$, $\hat{r} = 0.49599$ (with dinosaurs); $\log \hat{K} = 2.15041$, $\hat{r} = 0.75226$ (without dinosaurs)

```
> # remove dinosaurs
>
> Animals2 = subset(Animals, !(rownames(Animals)
+                               %in% c("Triceratops", "Diplodocus", "Brachiosaurus")))
>
> # fit with and without dinosaurs
>
> fit = lm(log(brain) ~ log(body), data=Animals)
> fit2 = lm(log(brain) ~ log(body), data=Animals2)
>
> # redraw plot, with fits
>
> par(mfrow=c(1,1),pty='m')
> with(Animals,plot(log(body),log(brain),type='n',xlim=c(-4,13)))
> with(Animals,text(log(body),log(brain),rownames(Animals),cex=0.9))
> abline(fit$coef,lty=2)
```

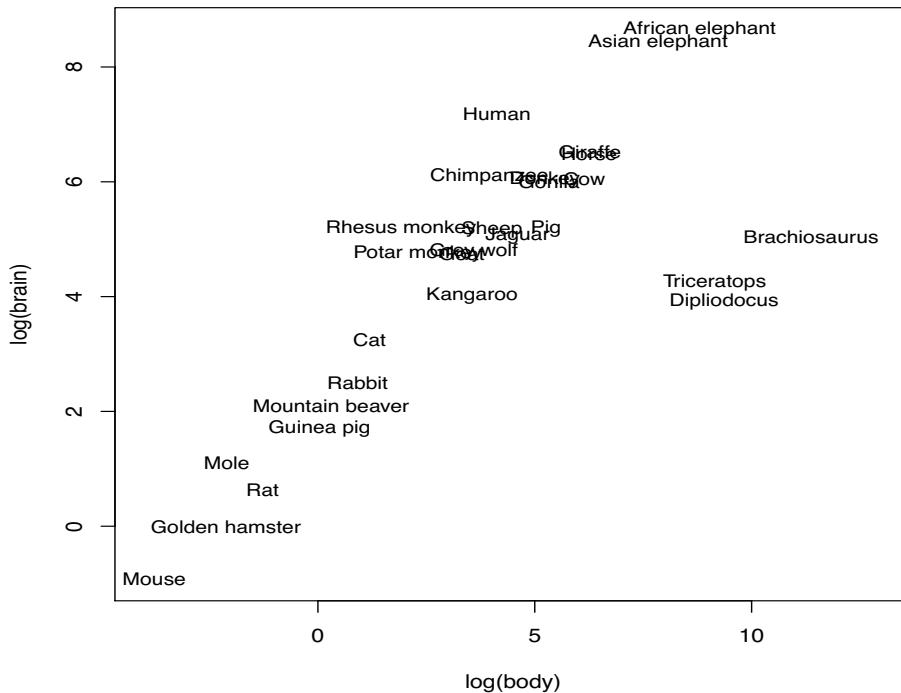


Figure 22.4: Plot for Problem 22.9 (a).

```

> abline(fit2$coef)
> legend('topleft',legend=c('With dinosaurs','Without dinosaurs'),lty=c(2,1))
>
> # give coefficient summary
>
> summary(fit)

Call:
lm(formula = log(brain) ~ log(body), data = Animals)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.2890 -0.6763  0.3316  0.8646  2.5835 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2.55490   0.41314   6.184 1.53e-06 ***
log(body)    0.49599   0.07817   6.345 1.02e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.532 on 26 degrees of freedom

```

```

Multiple R-squared:  0.6076, Adjusted R-squared:  0.5925
F-statistic: 40.26 on 1 and 26 DF,  p-value: 1.017e-06

> summary(fit2)

Call:
lm(formula = log(brain) ~ log(body), data = Animals2)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.9125 -0.4752 -0.1557  0.1940  1.9303 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2.15041   0.20060 10.72 2.03e-10 ***
log(body)    0.75226   0.04572 16.45 3.24e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```

Residual standard error: 0.7258 on 23 degrees of freedom
 Multiple R-squared: 0.9217, Adjusted R-squared: 0.9183
 F-statistic: 270.7 on 1 and 23 DF, p-value: 3.243e-14

- (c) The actual brain mass is Y_i , and the predicted brain mass is KX_i^r . Then

$$EQ_i = \frac{Y_i}{KX_i^r},$$

and

$$\log(EQ_i) = \log(Y_i) - \log(KX_i^r) = \log(Y_i) - [\log(K) + r \log(X_i)].$$

After we substitute estimates $\hat{\beta}_0 = \log(\hat{K})$ and $\hat{\beta}_1 = \hat{r}$ we have

$$\log(EQ_i) \approx \log(Y_i) - [\hat{\beta}_0 + \hat{\beta}_1 \log(X_i)] = e_i,$$

where e_i is the i th residual of the linear model (1). Then

$$EQ_i \approx \exp(e_i).$$

- (d) A sorting of the residuals ranks the EQ_i values. Pig has the smallest and Human has the highest. From Figure 22.5 it can be seen that the dinosaurs have observed brain mass Y well below predicted brain mass KX^r . They would have the smallest EQ_i .

```

> sort(fit2$residuals)
          Pig        Kangaroo         Cow        Jaguar
-0.912462456 -0.799609067 -0.723453348 -0.558454892
Golden hamster           Rat       Guinea pig        Horse
-0.555421133 -0.550956139 -0.475168201 -0.371731840

```

Rabbit	Giraffe	Mountain beaver	Mouse
-0.346496158	-0.345737520	-0.284304924	-0.228979028
Gorilla	African elephant	Grey wolf	Donkey
-0.155653772	-0.122218673	-0.069700540	-0.048100839
Sheep	Goat	Cat	Asian elephant
-0.006993271	0.097024005	0.194039293	0.384317819
Mole	Potar monkey	Chimpanzee	Rhesus monkey
0.530756811	0.862375734	0.961686125	1.594948144
Human			
1.930293870			

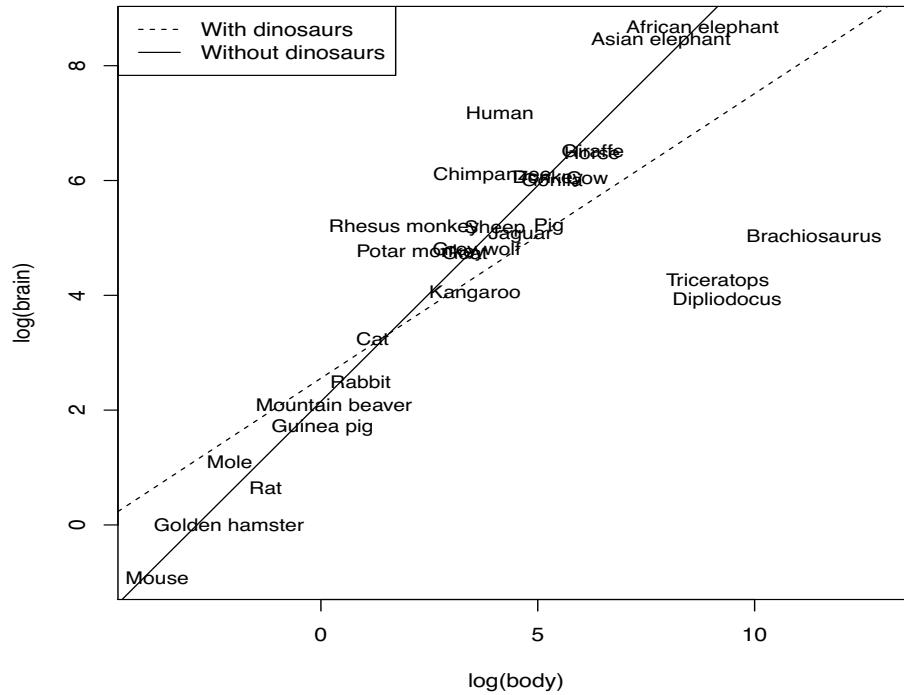


Figure 22.5: Plot for Problem 22.9 (b).

22.3 Theoretical Complements

Problem 22.10. Consider the case of linear regression through the origin:

$$Y_i = \beta X_i + \epsilon_i, \quad i = 1, \dots, n$$

where $\epsilon_i \sim N(0, \sigma^2)$ are *iid* error terms, and X_1, \dots, X_n are fixed predictor terms. Write explicitly the error sum of squares SSE for this model, where $\hat{\beta}$ is an estimate of β . After verifying that

SSE is a second order polynomial in $\hat{\beta}$, determine the least squares estimate of β directly in terms of the observation (X_i, Y_i) , $i = 1, \dots, n$.

SOLUTION:

We have

$$SSE = \sum_{i=1}^n (Y_i - \hat{\beta}X_i)^2 = \left[\sum_{i=1}^n Y_i^2 \right] - 2\hat{\beta} \left[\sum_{i=1}^n X_i Y_i \right] + \hat{\beta}^2 \left[\sum_{i=1}^n X_i^2 \right].$$

The minimum is directly given as

$$\hat{\beta} = \frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2}.$$

Problem 22.11. Suppose we are given the simple linear regression model

$$Y = \beta_0 + \beta_1 X + \epsilon, \quad (22.9)$$

where $\epsilon \sim N(0, \sigma_\epsilon^2)$. Suppose X is then interpreted itself as a random outcome with distribution $X \sim N(0, 1)$, which is independent of ϵ . Derive the correlation coefficient ρ_X of (X, Y) , where

$$\rho_{XY} = \frac{cov(X, Y)}{\sqrt{var(X)var(Y)}}.$$

This provides a method of simulating jointly distributed random variables.

SOLUTION:

The correlation is

$$\rho_{XY} = \frac{cov(X, Y)}{\sqrt{var(X)var(Y)}} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sqrt{var(X)var(Y)}}.$$

We have, given what we know of ϵ , X and Y ,

$$\begin{aligned} \mu_X &= 0 \\ \mu_Y &= \beta_0 \\ var(X) &= 1 \\ var(Y) &= var(\beta_1 X + \epsilon) = \beta_1^2 + \sigma_\epsilon^2 \\ E[(X - \mu_X)(Y - \mu_Y)] &= E[X(\beta_1 X + \epsilon)] = \beta_1 E[X^2] + E[X\epsilon] = \beta_1 + 0. \end{aligned}$$

So,

$$\rho_{XY} = \frac{\beta_1}{\sqrt{\beta_1^2 + \sigma_\epsilon^2}}.$$

Problem 22.12. We are given a simple linear regression model $Y = \beta_0 + \beta_1 X$, based on n pairs of independent and dependent variables (X_i, Y_i) . Let $\hat{\beta}_i$ be the least squares estimates of β_i , $i = 0, 1$.

- (a) Suppose we have some choice of the independent variables X_i , subject to the following constraints:

- (i) Sample size n is even.
- (ii) The mean $\bar{X} = n^{-1} \sum_{i=1}^n X_i$ is constrained to equal some fixed number x^* .
- (iii) We impose the bound $x^* + M \geq X_i \geq x^* - M$, $i = 1, \dots, n$, for some fixed number $M > 0$.

Show that the variance of $\hat{\beta}_1$ is minimized by setting $X_i = x^* - M$ for half the sample, and $X_i = x^* + M$ for the other half (the selection of independent variables for the purpose of optimizing the efficiency of an inference is referred to as the *design problem*).

- (b) Suppose we construct estimate

$$\hat{\mu}_x = \hat{\beta}_0 + \hat{\beta}_1 x,$$

for some fixed x . Does the variance of $\hat{\mu}_x$ depend on x ? If so, for what value of x is the variance minimized?

SOLUTION:

- (a) The variance of $\hat{\beta}_1$ is

$$\sigma_{\hat{\beta}_1}^2 = \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}.$$

Since σ^2 is fixed $\sigma_{\hat{\beta}_1}^2$ is minimized by maximizing $\sum_{i=1}^n (X_i - \bar{X})^2$. Under constraint $\bar{X} = x^*$ we must have

$$\sum_{i=1}^n (X_i - \bar{X})^2 \leq nM^2,$$

assuming the given bound holds. But, under the proposed design we have

$$\sum_{i=1}^n (X_i - \bar{X})^2 = nM^2.$$

This gives the smallest possible variance

$$\sigma_{\hat{\beta}_1}^2 = \frac{\sigma^2}{nM^2}.$$

- (b) The variance of $\hat{\mu}_x$ is

$$\sigma_{\hat{\mu}_x}^2 = \sigma^2 \left[\frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right].$$

This quantity depends on x , and is minimized by setting $x = \bar{X}$.

Problem 22.13. We are given a multiple linear regression model

$$y_i = \beta_1 x_{i1} + \dots + \beta_q x_{iq} + \epsilon_i, \quad i = 1, \dots, n,$$

where ϵ_i are *iid* error terms with $\epsilon_i \sim N(0, \sigma^2)$. Let $\hat{\beta}_i$ be the least squares estimate of β_i , $i = 1, \dots, q$. There is an advantage with respect to the interpretability of the regression coefficients if they are uncorrelated. In this case, the contribution of each predictor to the model can be assessed independently of the other predictors.

- (a) If $\hat{\beta} = [\hat{\beta}_1 \dots \hat{\beta}_q]^T$ is the vector of least squares coefficient estimates, then the covariance matrix is given by

$$\Sigma_{\hat{\beta}} = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}. \quad (22.10)$$

Using this expression, show that the regression coefficients are mutually uncorrelated if and only if

$$\sum_{i=1}^n x_{ij} x_{ik} = 0 \quad (22.11)$$

for each pair $j \neq k$.

- (b) Consider the simple regression model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, \dots, n. \quad (22.12)$$

Suppose the independent variable is transformed to $x'_i = x_i - \bar{x}$, where $\bar{x} = n^{-1} \sum_i x_i$. Then, using the transformed independent variable, consider the alternative model

$$y_i = \beta'_0 + \beta'_1 x'_i + \epsilon_i, \quad i = 1, \dots, n. \quad (22.13)$$

The values y_i and ϵ_i are otherwise the same for both models (22.12) and (22.13).

- (i) Show that the two models are equivalent in the sense that the fitted values \hat{y}_i must be the same. How are the least squares estimates of the coefficients for the respective models related?
- (ii) Show that for model (22.13) the estimates of the coefficients β'_0 and β'_1 are uncorrelated.

SOLUTION:

- (a) The components of $\hat{\beta}$ are uncorrelated if and only if $\Sigma_{\hat{\beta}}$ is a diagonal matrix. But an invertible square matrix is diagonal if and only if its inverse is diagonal. Therefore, the components of $\hat{\beta}$ are uncorrelated if and only if $\mathbf{X}^T \mathbf{X}$ is a diagonal matrix. This condition is equivalent to (22.11).
- (b) Consider the matrix representation of the multiple linear regression model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

where \mathbf{y} is an $n \times 1$ response vector, \mathbf{X} is a $n \times q$ matrix, β is a $q \times 1$ vector of coefficients, and ϵ is an $n \times 1$ vector of error terms. Then the least squares estimates $\hat{\beta}$ of β are obtained by minimizing

$$SSE = \sum_{i=1}^n (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$$

where $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ is the $n \times 1$ vector of fitted values.

Viewed geometrically, $\hat{\mathbf{y}}$ is a linear combination of the form

$$\hat{\mathbf{y}} = \hat{\beta}_1 \mathbf{x}_1 + \dots + \hat{\beta}_q \mathbf{x}_q,$$

where $\mathbf{x}_1, \dots, \mathbf{x}_q$ are the $n \times 1$ column vectors of \mathbf{X} . Let \mathcal{Y} be the set of all $n \times 1$ vectors which are linear combinations of the column vectors of \mathbf{X} (this set is strictly smaller than \mathbb{R}^n , provided $n > q$). Then $\hat{\mathbf{y}}$ is the unique vector in \mathcal{Y} which minimizes SSE .

In general, if we have p vectors $\mathbf{v}_i \in \mathbb{R}^n$, then the *span* of $(\mathbf{v}_1, \dots, \mathbf{v}_p)$ is the set of all linear combinations of those p vectors. Let $\mathcal{Y}_{\mathbf{X}}$ be the span of the column vectors of \mathbf{X} . Then $\hat{\mathbf{y}}$ is the element of $\mathcal{Y}_{\mathbf{X}}$ which minimizes SSE , and the least squares coefficient estimations are the values $\hat{\boldsymbol{\beta}}$ for which $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$.

- (i) We can answer Part (b)-(i) by showing that the span of the predictors \mathbf{X} is the same for the two models. For model (22.12) the predictor vectors, or column vectors of \mathbf{X} , are

$$\begin{aligned}\mathbf{x}_1 &= [1 \dots 1]^T \\ \mathbf{x}_2 &= [x_1 \dots x_n]^T.\end{aligned}$$

For model (22.13) the predictor vectors, or column vectors of \mathbf{X}' , are

$$\begin{aligned}\mathbf{x}'_1 &= [1 \dots 1]^T \\ \mathbf{x}'_2 &= [x_1 - \bar{x} \dots x_n - \bar{x}]^T \\ &= [x_1 \dots x_n]^T - \bar{x}[1 \dots 1]^T \\ &= \mathbf{x}_2 - \bar{x} \cdot \mathbf{x}_1.\end{aligned}$$

It can be seen that \hat{y} is a linear combination of $\mathbf{x}_1, \mathbf{x}_2$ if and only if it is a linear combination of $\mathbf{x}'_1, \mathbf{x}'_2$. In other words $\mathcal{Y}_{\mathbf{x}} = \mathcal{Y}_{\mathbf{x}'}$, so the fitted values $\hat{\mathbf{y}}$ must be the same, and the two models are equivalent.

Since the models are equivalent, we can write

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i = \hat{\beta}'_0 + \hat{\beta}'_1 (x_i - \bar{x}) = [\hat{\beta}'_0 - \hat{\beta}'_1 \bar{x}] + \hat{\beta}'_1 x_i,$$

for $i = 1, \dots, n$. We must therefore have

$$\begin{aligned}\hat{\beta}_0 &= \hat{\beta}'_0 - \hat{\beta}'_1 \bar{x} \\ \hat{\beta}_1 &= \hat{\beta}'_1.\end{aligned}$$

- (ii) For model (22.13) we have

$$\sum_{i=1}^n 1 \times x'_i = \sum_{i=1}^n 1 \times (x_i - \bar{x}) = 0,$$

therefore condition (22.11) is satisfied, which implies that $\hat{\beta}'_0$ and $\hat{\beta}'_1$ are uncorrelated.

Problem 22.14. We are given a simple linear regression model $Y = \beta_0 + \beta_1 X$.

1. Let $\hat{\beta}_i$ be the least squares estimates of β_i , $i = 0, 1$. Suppose a constant c is added to each response and the model refit. What will be the new least squares estimates of β_i , expressed in terms of the old estimates? Verify your answer analytically.
2. The coefficient β_0 is referred to as the *intercept term* (where the Y -axis is intercepted by the regression line). It can be interpreted as a summary of the vertical location of the response Y , since the effect of changing the constant c of part (a) is directly observable in β_0 . Of course, $\beta_0 = \mu_0$, where $\mu_x = \beta_0 + \beta_1 x$, so we may construct a new intercept μ_x at any vertical line $X = x$ for the same purpose. The least squares estimate will be

$$\hat{\mu}_x = \hat{\beta}_0 + \hat{\beta}_1 x.$$

What analytical criterion can be used to select x , and what would be the resulting optimal choice?

SOLUTION:

- (a) For simple linear regression the least squares coefficient estimates are:

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}.\end{aligned}$$

Denote the new estimates $\hat{\beta}'_i$. If we substitute $Y_i + c$ for Y_i into the expression for $\hat{\beta}_1$, it is clear that the c 's will cancel, so that $\hat{\beta}'_1 = \hat{\beta}_1$. On the other hand, in the expression for $\hat{\beta}_0$ we replace \bar{Y} with $\bar{Y} + c$, so that $\hat{\beta}'_0 = \hat{\beta}_0 + c$.

- (b) The variance of $\hat{\mu}_x$ is

$$\sigma_{\hat{\mu}_x}^2 = \sigma^2 \left[\frac{1}{n} + \frac{(x - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right].$$

This quantity is minimized by setting $x = \bar{X}$.

Problem 22.15. Consider the matrix representation of the multiple linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where \mathbf{y} is an $n \times 1$ response vector, \mathbf{X} is a $n \times q$ matrix, $\boldsymbol{\beta}$ is a $q \times 1$ vector of coefficients, and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of error terms. Let

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

be the least squares estimate of β (we will assume that $\mathbf{X}^T \mathbf{X}$ is invertible). The vector of residuals is given by

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}},$$

where $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$ is the vector of fitted values. Let y_i , \hat{y}_i and e_i denote the individual responses, fitted values and residuals, $i = 1, \dots, n$.

- (a) Suppose H is an $n \times n$ matrix for which the fitted values satisfy

$$\hat{\mathbf{y}} = H\mathbf{y}.$$

Give H explicitly in terms of \mathbf{X} .

- (b) Prove that H is symmetric and idempotent (that is, $H = HH$).
 (c) Prove that \mathbf{e} and $\hat{\mathbf{y}}$ are orthogonal, that is, $\hat{\mathbf{y}}^T \mathbf{e} = 0$.
 (d) Prove that

$$\mathbf{X}^T \mathbf{e} = \mathbf{0}$$

where $\mathbf{0}$ is a $q \times 1$ column vector of zeros.

- (e) Prove that if the model contains an intercept (that is, \mathbf{X} contains a column of 1's), then the sum of the residuals satisfies $\sum_{i=1}^n e_i = 0$
 (f) Prove that

$$\sum_{i=1}^n y_i^2 = \sum_{i=1}^n \hat{y}_i^2 + \sum_{i=1}^n e_i^2.$$

SOLUTION:

- (a) The fitted values may be written

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = H\mathbf{y},$$

where $H = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$.

- (b) First note that we always have $(ABC)^T = C^T B^T A^T$ where defined. This implies that $\mathbf{X}^T \mathbf{X}$ is symmetric, therefore so is the inverse $(\mathbf{X}^T \mathbf{X})^{-1}$. We then have

$$H^T = \mathbf{X} [(\mathbf{X}^T \mathbf{X})^{-1}]^T \mathbf{X}^T = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T = H,$$

therefore H is symmetric. Then H is idempotent, since

$$\begin{aligned} HH &= \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \\ &= \mathbf{X} [(\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X})] (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \\ &= \mathbf{X} I_n (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \\ &= \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \\ &= H, \end{aligned}$$

where I_n is the $n \times n$ identity matrix.

(c) The residual vector is given by

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = I_n \mathbf{y} - H \mathbf{y} = (I_n - H) \mathbf{y}.$$

We may then write

$$\begin{aligned}\hat{\mathbf{y}}^T \mathbf{e} &= \mathbf{y}^T H^T (I_n - H) \mathbf{y} \\ &= \mathbf{y}^T H (I_n - H) \mathbf{y} \\ &= \mathbf{y}^T (H - HH) \mathbf{y} \\ &= \mathbf{y}^T (H - H) \mathbf{y} \\ &= 0,\end{aligned}$$

since H is idempotent.

(d) We can write

$$\begin{aligned}\mathbf{X}^T \mathbf{e} &= \mathbf{X}^T (I_n - H) \mathbf{y} \\ &= (\mathbf{X}^T - \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T) \mathbf{y} \\ &= (\mathbf{X}^T - \mathbf{X}^T) \mathbf{y} \\ &= \mathbf{0}\end{aligned}$$

(e) If the model contains an intercept, then one of the columns of \mathbf{X} consists of 1's. By Part (d) $\mathbf{X}^T \mathbf{e} = \mathbf{0}$. This implies $\sum_{i=1}^n e_i = 0$.

(f) We can write

$$\begin{aligned}\sum_{i=1}^n y_i^2 &= \mathbf{y}^T \mathbf{y} \\ &= (\hat{\mathbf{y}} + \mathbf{e})^T (\hat{\mathbf{y}} + \mathbf{e}) \\ &= \hat{\mathbf{y}}^T \hat{\mathbf{y}} + 2\hat{\mathbf{y}}^T \mathbf{e} + \mathbf{e}^T \mathbf{e} \\ &= \hat{\mathbf{y}}^T \hat{\mathbf{y}} + \mathbf{e}^T \mathbf{e} \\ &= \sum_{i=1}^n \hat{y}_i^2 + \sum_{i=1}^n e_i^2,\end{aligned}$$

since by Part (c) $\hat{\mathbf{y}}^T \mathbf{e} = 0$.

Chapter 23

Practice Problems - Logistic Regression

23.1 Exercises

Problem 23.1. Suppose a logistic regression model is fit with response $Y = \text{infection}$ (*at clinic visit*) and two predictors $X_1 = \text{antibody level (log titre)}$ and $X_2 = \text{age (years)}$. The estimated regression coefficients are $\hat{\beta}_0 = -1.23$, $\hat{\beta}_1 = -0.73$, $\hat{\beta}_2 = -0.05$.

- (a) Write an explicit formula for estimating $P(\text{infection})$ in terms of an antibody level x_1 and age x_2 , making use of the estimated regression coefficients.
- (b) What are the estimated values of $P(\text{infection})$ for three subjects of age 4.5 years with antibody levels -0.5, 1.25 and 3.5?
- (c) Suppose the IQR of the antibody levels is estimated to be 2.78. What is the estimated odds ratio for infection between subjects at the 75th percentile and the 25th percentile of antibody levels? Does this depend on age?
- (d) Construct a plot showing estimated $P(\text{infection})$ as a function of antibody level for ages 1, 5, 9, superimposed on one plot. Use the `plot` option `lty` to distinguish the lines, and use the `legend` function to label the lines accordingly.
- (e) Suppose another model is fit, with an interaction term added, so that the linear prediction term is now:

$$\eta = \beta_0 + \beta_1 \times \text{antibody} + \beta_2 \times \text{age} + \beta_3 \times \text{antibody} \times \text{age},$$

with estimated coefficients $\hat{\beta}_0 = -1.13$, $\hat{\beta}_1 = -1.05$, $\hat{\beta}_2 = -0.049$, $\hat{\beta}_3 = 0.064$. Repeat part (d) and comment on the differences between the two models.

- (f) The odds ratio defined in part (c) now depends on age. Develop a formula for that odds ratio as a function of age, and plot the function over the range [1, 10].
- (g) We generally expect the probability of infection to *decrease* with increasing antibody levels. In the model of part (e) for what ages does this hold? What does this say about the suitability of this model?

SOLUTION:

Suppose a logistic regression model is fit with response $Y = \text{infection}$ (at clinic visit) and two predictors $X_1 = \text{antibody level (log titre)}$ and $X_2 = \text{age (years)}$. The estimated regression coefficients are $\hat{\beta}_0 = -1.23$, $\hat{\beta}_1 = -0.73$, $\hat{\beta}_2 = -0.05$.

$$(a) P(\text{infection}) = \frac{1}{1+\exp(1.23+0.73x_1+0.05x_2)}$$

(b) The probabilities are given directly by:

$$\begin{aligned} 0.2516 &= \frac{1}{1 + \exp(1.23 + 0.73 \times -0.5 + 0.05 \times 4.5)} \\ 0.0857 &= \frac{1}{1 + \exp(1.23 + 0.73 \times 1.25 + 0.05 \times 4.5)} \\ 0.0178 &= \frac{1}{1 + \exp(1.23 + 0.73 \times 3.5 + 0.05 \times 4.5)} \end{aligned}$$

(c) Odds ratios are given generally by

$$OR[\text{infection}] = e^{\hat{\beta}_1(x_1 - x'_1) + \hat{\beta}_2(x_2 - x'_2)}$$

If x_1 alone is varied, and we compare (Q_{75}, x_2) to (Q_{25}, x_2) where Q_{75} and Q_{25} are the percentiles, we get

$$\begin{aligned} OR[\text{infection}] &= e^{\hat{\beta}_1(Q_{75} - Q_{25}) + \hat{\beta}_2(x_2 - x_2)} \\ &= e^{\hat{\beta}_1 \times IQR} \\ &= e^{-0.73 \times 2.78} \\ &= 0.1314. \end{aligned}$$

(d) The following code creates the required plot (Figure 23.1):

```
beta0 = -1.23
beta1 = -0.73
beta2 = -0.05

ilogit = function(x) {1/(1+exp(-x))}

ab = seq(-1,5,by=0.25)

plot(ab, ilogit(beta0 + beta1*ab + beta2*1),type='l',lty=1,ylim=c(0,0.4),
      xlab='AB (log)',ylab='P(infection)')
lines(ab, ilogit(beta0 + beta1*ab + beta2*5),type='l',lty=2)
lines(ab, ilogit(beta0 + beta1*ab + beta2*9),type='l',lty=3)
legend('topright',legend=c('1 yr','5 yr','9 yr'),lty=c(1,2,3))
```

(e) The following code creates the required plot (Figure 23.2). The effect of the interaction is to reduce the dependence of $P(\text{infection})$ on AB levels for older subjects. Specifically, the rate of decrease of $P(\text{infection})$ with increasing AB is lower for older subjects.

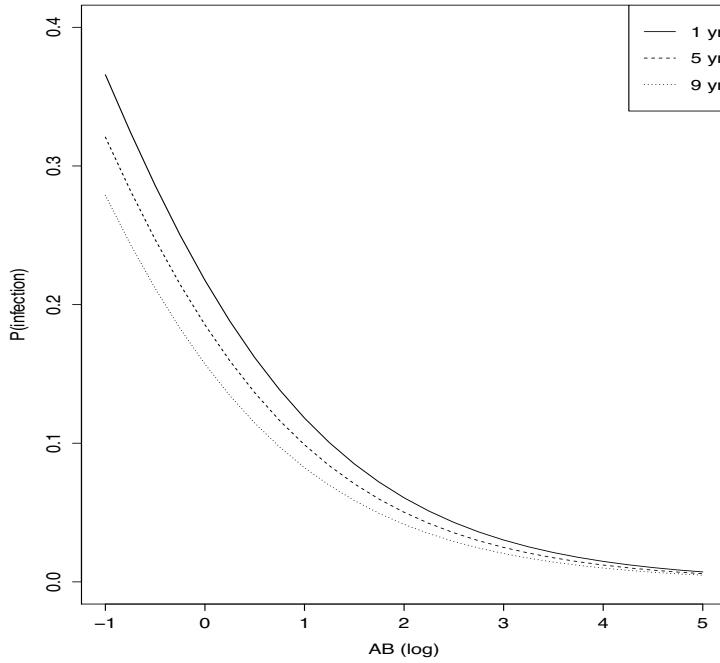


Figure 23.1: Plot for Problem 23.1 (d)

```

beta0 = -1.13
beta1 = -1.05
beta2 = -0.049
beta3 = 0.064

ilogit = function(x) {1/(1+exp(-x))}

ab = seq(-1,5,by=0.25)

plot(ab, ilogit(beta0 + beta1*ab+beta2*1 + beta3*1*ab),type='l',lty=1,ylim=c(0,0.5),
      xlab='AB (log)',ylab='P(infection)')
lines(ab, ilogit(beta0 + beta1*ab+beta2*5 + beta3*5*ab),type='l',lty=2)
lines(ab, ilogit(beta0 + beta1*ab+beta2*9 + beta3*9*ab),type='l',lty=3)
legend('topright',legend=c('1 yr','5 yr','9 yr'),lty=c(1,2,3))

```

(f) Following part (c) we have:

$$\begin{aligned}
 OR[infection] &= e^{\hat{\beta}_1(Q_{75}-Q_{25})+\hat{\beta}_2(x_2-x_1)+\hat{\beta}_3(Q_{75}x_2-Q_{25}x_1)} \\
 &= e^{\hat{\beta}_1 \times IQR + \hat{\beta}_3 \times IQR \times x_2}
 \end{aligned}$$

The following code creates the required plot (Figure 23.3).

```

ilogit = function(x) {1/(1+exp(-x))}
age = seq(1,10,by=0.25)
ex = expression(paste('OR for ', Q[75], ' vs ', Q[25], sep=''))
plot(age,exp((beta1+beta3*age)*2.78),type='l',xlab='Age (years)',ylab=ex)

```

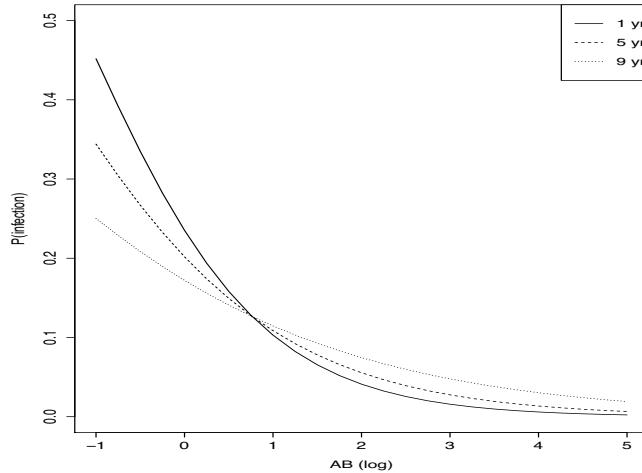


Figure 23.2: Plot for Problem 23.1 (e)

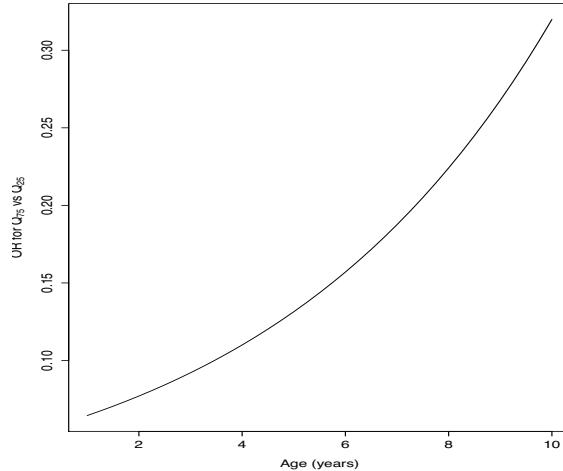


Figure 23.3: Plot for Problem 23.1 (f)

(g) The exponential rate of change of $P(\text{infection})$ with antibody for fixed age is

$$\Delta = \beta_1 + \beta_3 \times AGE.$$

We therefore have

$$\Delta < 0 \text{ if and only if } AGE < \frac{-\beta_1}{\beta_3}.$$

In this model

$$\Delta < 0 \text{ if and only if } AGE < \frac{1.05}{.064} = 16.40625.$$

This doesn't rule out using the model for subjects below a certain age. We may have data on subjects up to, say, age 10, and we can verify the suitability of the model for the observed age range. The model would be incorrect outside this range.

Problem 23.2. A logistic regression model is used to model $P(Y = 1)$ for some binary response variable Y . It depends on two predictors, a quantitative predictor x and the indicator variable `i.class`. The following logistic regression model is used:

$$P(Y = 1) = \frac{e^\eta}{1 + e^\eta}, \text{ where } \eta = \beta_0 + \beta_1 x + \beta_2 i.\text{class} + \beta_3 x \times i.\text{class}.$$

Using data with sample size $n = 94$, the following coefficient estimates were obtained. The estimated covariance matrix for the estimated coefficients in vector form $[\hat{\beta}_0, \dots, \hat{\beta}_3]^T$ is given immediately following.

```
>
> ### coefficient estimates
>
> summary(fit)$coef
      Estimate Std. Error   z value Pr(>|z|)
(Intercept) -1.0488571  0.7736844 -1.355665 0.175205679
x             0.7183279  0.2540876  2.827087 0.004697354
i.class       1.3787316  0.9861359  1.398115 0.162078447
x:i.class    -0.9788835  0.2823500 -3.466915 0.000526468
>
> ### estimated covariance matrix
>
> summary(fit)$cov.scaled
            (Intercept)      x     i.class   x:i.class
(Intercept)  0.5985876 -0.15894404 -0.5985876  0.15894404
x           -0.1589440  0.06456053  0.1589440 -0.06456053
i.class     -0.5985876  0.15894404  0.9724639 -0.22173994
x:i.class   0.1589440 -0.06456053 -0.2217399  0.07972153
```

(a) Carry out a hypothesis test for null hypothesis H_o and alternative hypothesis H_a given by:

$$\begin{aligned} H_o &: P(Y = 1) \text{ is not an increasing function of } x \text{ for fixed } i.\text{class} = 0, \text{ against} \\ H_a &: P(Y = 1) \text{ is an increasing function of } x \text{ for fixed } i.\text{class} = 0. \end{aligned}$$

Use a t -statistic based on the appropriate degrees of freedom. Use significance level $\alpha = 0.05$.

(b) Carry out a hypothesis test for null hypothesis H_o and alternative hypothesis H_a given by:

$$\begin{aligned} H_o &: P(Y = 1) \text{ is not a decreasing function of } x \text{ for fixed } i.\text{class} = 1, \text{ against} \\ H_a &: P(Y = 1) \text{ is a decreasing function of } x \text{ for fixed } i.\text{class} = 1. \end{aligned}$$

Use a t -statistic based on the appropriate degrees of freedom. Use significance level $\alpha = 0.05$.

SOLUTION:

- (a) The required hypothesis test is

$$\begin{aligned} H_o &: \beta_1 \leq 0, \text{ against} \\ H_a &: \beta_1 > 0. \end{aligned}$$

From the coefficient table we have estimate and standard deviation $\hat{\beta}_1 = 0.7183279$, $S = 0.2540876$, giving t -statistic

$$T = \frac{\hat{\beta}_1}{S} = \frac{0.7183279}{0.2540876} = 2.827088.$$

There are $p = 4$ coefficients, so the appropriate degrees of freedom is $n - 4 = 90$. We reject H_o if $T > t_{90,0.05} = 1.662$. Therefore, we reject H_o , and conclude that $P(Y = 1)$ is an increasing function of x for fixed `i.class = 0`.

- (b) When `i.class = 1`, the slope of η is $\beta_1 + \beta_3$. The required hypothesis test is therefore

$$\begin{aligned} H_o &: \beta_1 + \beta_3 \geq 0, \text{ against} \\ H_a &: \beta_1 + \beta_3 < 0. \end{aligned}$$

The estimate of $\beta_1 + \beta_3$ is

$$\hat{\beta}_1 + \hat{\beta}_3 = 0.7183279 - 0.9788835 = -0.2605556.$$

To calculate the standard error of $\hat{\beta}_1 + \hat{\beta}_3$ we need the standard errors S_1, S_3 of $\hat{\beta}_1$ and $\hat{\beta}_3$, and the estimated covariance S_{13} . From the estimated covariance matrix we have

$$\begin{aligned} S_1^2 &= 0.06456053 \\ S_3^2 &= 0.07972153 \\ S_{13} &= -0.06456053 \end{aligned}$$

The standard error S_+ of $\hat{\beta}_1 + \hat{\beta}_3$ is then given by

$$S_+^2 = S_1^2 + S_3^2 + 2S_{13} = 0.06456053 + 0.07972153 - 2 \times 0.06456053 = 0.015161.$$

The t -statistic is then

$$T = \frac{\hat{\beta}_1 + \hat{\beta}_3}{S_+} = \frac{-0.2605556}{0.015161^{1/2}} = \frac{-0.2605556}{0.12313} = -2.116102.$$

There are $p = 4$ coefficients, so the appropriate degrees of freedom is $n - 4 = 90$. We reject H_o if $T < t_{90,0.05} = 1.662$. Therefore, we reject H_o , and conclude that $P(Y = 1)$ is a decreasing function of x for fixed `i.class = 1`.

Problem 23.3. A model for predicting victory for a professional baseball team is developed. It depends on two predictors, T = temperature on game day, in degrees Fahrenheit, and the indicator variable $I_H = 1$ for home games. The following logistic regression model is used:

$$P(\text{Win Game}) = \frac{e^\eta}{1 + e^\eta}, \text{ where } \eta = \beta_0 + \beta_1 T + \beta_2 I_H + \beta_3 T \times I_H.$$

Suppose the parameter estimates are $\hat{\beta}_0 = -0.025$, $\hat{\beta}_1 = -0.027$, $\hat{\beta}_2 = -2.15$, $\hat{\beta}_3 = 0.076$.

- (a) When playing at home, does the team prefer higher or lower temperatures? What about when they play away?
- (b) Suppose the temperature is assumed to be within the range [65, 85]. What are the minimum and maximum values of $P(\text{Win Game})$ when the team plays at home, and when the team plays away?
- (c) In order to predict the number of wins for a season, a simple temperature model is developed. The temperature will be either 65° or 85° . We assume probabilities $P(T = 85^\circ | I_H = 1) = 0.45$, $P(T = 85^\circ | I_H = 0) = 0.62$. Assuming exact half of the games are home games, what is the overall predicted win rate for the season?

SOLUTION:

- (a) First, note that $P(\text{Win Game})$ is an increasing function of η . When playing at home, $I_H = 1$, so

$$\eta = \beta_0 + \beta_2 + (\beta_1 + \beta_3)T.$$

We have estimate $\beta_1 + \beta_3 \approx \hat{\beta}_1 + \hat{\beta}_3 = -0.027 + 0.076 = 0.049$. In this case, η , and therefore $P(\text{Win Game})$, increases with temperature T . On the other hand, for away games $I_H = 0$, so

$$\eta = \beta_0 + \beta_1 T.$$

Since $\beta_1 \approx \hat{\beta}_1 = -0.027$, we conclude that η , and therefore $P(\text{Win Game})$, decreases with temperature T .

- (b) In general, we have

$$\begin{aligned} P(\text{Win Game} | I_H = 0) &= (1 + \exp(0.025 + 0.027 \times T))^{-1} \\ P(\text{Win Game} | I_H = 1) &= (1 + \exp(2.175 - 0.049 \times T))^{-1} \end{aligned}$$

For both home and away games, the extreme points are calculated at $T = 65, 85$. We need the probabilities

$$\begin{aligned} P(\text{Win Game} | T = 65, I_H = 0) &= (1 + \exp(0.025 + 0.027 \times 65))^{-1} = 0.1443 \\ P(\text{Win Game} | T = 85, I_H = 0) &= (1 + \exp(0.025 + 0.027 \times 85))^{-1} = 0.0895 \\ P(\text{Win Game} | T = 65, I_H = 1) &= (1 + \exp(2.175 - 0.049 \times 65))^{-1} = 0.7330 \\ P(\text{Win Game} | T = 85, I_H = 1) &= (1 + \exp(2.175 - 0.049 \times 85))^{-1} = 0.8797. \end{aligned}$$

(c) The overall probability is found by the law of total probability:

$$\begin{aligned}
 p_W &= P(\text{Win Game} \mid T = 65, I_H = 0)P(T = 65, I_H = 0) \\
 &\quad + P(\text{Win Game} \mid T = 85, I_H = 0)P(T = 85, I_H = 0) \\
 &\quad + P(\text{Win Game} \mid T = 65, I_H = 1)P(T = 65, I_H = 1) \\
 &\quad + P(\text{Win Game} \mid T = 85, I_H = 1)P(T = 85, I_H = 1) \\
 &= [0.1443 \times (1 - 0.62) + 0.0895 \times 0.62 + 0.7330 \times (1 - 0.45) + 0.8797 \times 0.45] / 2 \\
 &\approx 0.454.
 \end{aligned}$$

Problem 23.4. We wish to develop a model which predicts the probability that a boxer wins a match based on weight differential. Suppose W is the amount in pounds by which the weight of boxer A exceeds the weight of boxer B (of course, W can be negative). Then

$$P(\text{Boxer } A \text{ wins} \mid W) = \frac{e^\eta}{1 + e^\eta}, \text{ where } \eta = \beta_0 + \beta_1 W.$$

(a) Why would we expect

$$P(\text{Boxer } A \text{ wins} \mid W) + P(\text{Boxer } A \text{ wins} \mid -W) = 1$$

as a general rule? Show that this happens when $\beta_0 = 0$.

(b) Suppose we constrain $\beta_0 = 0$, and we are given $P(\text{Boxer } A \text{ wins} \mid 15) = 0.61$. What is β_1 ?

SOLUTION:

(a) Suppose we have two boxers A , B . We expect $P(\text{Boxer } A \text{ wins}) + P(\text{Boxer } B \text{ wins}) = 1$. In addition, if $W = w$ for boxer A , then $W = -w$ for boxer B . Suppose $\beta_0 = 0$. Evaluate

$$\begin{aligned}
 \frac{e^{\beta_1 w}}{1 + e^{\beta_1 w}} + \frac{e^{-\beta_1 w}}{1 + e^{-\beta_1 w}} &= \frac{e^{\beta_1 w}}{1 + e^{\beta_1 w}} + \frac{e^{\beta_1 w}}{e^{\beta_1 w}} \times \frac{e^{-\beta_1 w}}{1 + e^{-\beta_1 w}} \\
 &= \frac{e^{\beta_1 w}}{1 + e^{\beta_1 w}} + \frac{1}{1 + e^{\beta_1 w}} \\
 &= 1.
 \end{aligned}$$

(b) We then have

$$P(\text{Boxer } A \text{ wins} \mid W) = \frac{1}{1 + e^{-\beta_1 W}}.$$

If

$$0.61 = \frac{1}{1 + e^{-\beta_1 15}},$$

then

$$\beta_1 = \frac{-1}{15} \log \left(\frac{1}{0.61} - 1 \right) = 0.02982.$$

23.2 Data Analysis

Problem 23.5. This problem will make use of the `birthwt` data set from the `MASS` library. This data set was collected from birth records at Baystate Medical Center, Springfield, Mass during 1986, for the purpose of analyzing risk factors associated with low infant birth weight. The response will be the indicator variable `low`, set to 1 for birth weight less than 2.5 kg.

- (a) First examine the integer variable `ptl`, the number of previous premature labors. Construct a frequency table for the outcomes. Also include in this table the sample mean of `low` for each `ptl` outcome.
- (b) Fit a logistic regression model with response `birthwt` and predictor `ptl`. Do this using two methods, first with `ptl` as a numerical variable, then as a factor with each integer outcome as a single level. Using the fitted coefficients calculate the expected response (here, the probability of low birthweight) for each `ptl` outcome, and append these to the table constructed for Part (a).
- (c) Comment on the results of Part (b). In particular, which fit most resembles the sample means, and why would you expect this?
- (d) Refit the model, replacing `ptl` with the indicator variable `I(ptl > 0)`, then add the predictor `age`. You can use the command

```
fit.mult = glm(low ~ I(ptl > 0)+age, data = birthwt, family='binomial')
```

Plot the estimated probability of low birthweight on a single plot. Use `age` as the horizontal axis, and $P(\text{low birth weight})$ as the vertical axis. Include on the same plot separate lines for $\text{ptl} = 0$ and $\text{ptl} > 0$, properly labeled. Make sure the range of `age` on the plot matches that of the data. In general, what group is most at risk of low birth weight?

- (e) Write explicitly $P(\text{low birth weight})$ as function of `ptl` and `age` as modeled in Part (d), and the appropriate regression coefficients β_i (you can leave these as symbols). Show that in this model the odds ratio

$$OR = \frac{Odds(\text{low birth weight} \mid \text{ptl} > 0)}{Odds(\text{low birth weight} \mid \text{ptl} = 0)}$$

does not depend on `age`. Estimate this odds ratio from the fitted model, and include a 95% confidence interval.

- (f) Divide the data into two groups defined by $\{\text{age} \leq 25\}$ and $\{\text{age} > 25\}$. For each group estimate directly $P(\text{low birth weight} \mid \text{ptl} > 0)$, $P(\text{low birth weight} \mid \text{ptl} = 0)$, and the odds ratio OR defined in Part (e).
- (g) For the groups defined by $\{\text{age} \leq 25\}$ in Part (f), we would expect the probabilities $P(\text{low birth weight} \mid \text{ptl} > 0)$ and $P(\text{low birth weight} \mid \text{ptl} = 0)$ to be strictly between the fitted probabilities for ages 14 and 25 given the respective values of $I\{\text{ptl} > 0\}$ (14 and 25 are the end points of the age range used). Test this rule. Do the same for the $\{\text{age} > 25\}$ group, using age endpoints 25 and 45.
- (h) Do the two odds ratios for the two groups defined by $\{\text{age} \leq 25\}$ and $\{\text{age} > 25\}$ calculated in Part (f) seem compatible with the assumption that the odds ratio does not depend on `age`?

SOLUTION:

- (a) The following code produces the required table

```
>
> ### (a)
>
> freq = table(birthwt$ptl)
> mean.ptl = tapply(birthwt$low,birthwt$ptl,mean)
> cbind(freq,mean.ptl)
  freq  mean.ptl
0 159 0.2578616
1 24 0.6666667
2 5 0.4000000
3 1 0.0000000
>
```

- (b) The following code produces the required fitted models

```
> ### (b)
>
> logistic = function(x) {1/(1+exp(-x))}
>
> # numerical predictor
> fit.num = glm(low ~ ptl, data = birthwt, family='binomial')
> cf.num = summary(fit.num)$coef
>
> # factor predictor
> fit.fact = glm(low ~ as.factor(ptl), data = birthwt, family='binomial')
> cf.fact = summary(fit.fact)$coef
>
> # print coefficient tables
>
> cf.num
      Estimate Std. Error   z value   Pr(>|z|)
(Intercept) -0.9641890  0.1749606 -5.510892 3.570202e-08
ptl          0.8018058  0.3171533  2.528133 1.146709e-02
> cf.fact
      Estimate Std. Error   z value   Pr(>|z|)
(Intercept) -1.0571126  0.1812866 -5.83116864 5.504052e-09
as.factor(ptl)1  1.7502597  0.4694303  3.72847616 1.926411e-04
as.factor(ptl)2  0.6516474  0.9306977  0.70017093 4.838206e-01
as.factor(ptl)3 -13.5089552 882.7433938 -0.01530338 9.877901e-01
>
> # obtain fitted values
>
```

```

> fitted.num = logistic(cf.num[1,1]+(0:3)*cf.num[2,1])
> fitted факт = logistic(cf факт[1,1]+c(0, cf факт[2:4,1]))
>
> # append to table
>
> cbind(freq,mean.ptl,fitted.num,fitted. факт)
   freq mean.ptl fitted.num fitted. факт
0  159  0.2578616  0.2760403 2.578616e-01
1   24  0.6666667  0.4594932 6.666667e-01
2     5  0.4000000  0.6546229 4.000000e-01
3    1  0.0000000  0.8086448 4.721032e-07
>

```

- (c) The fitted model which uses the predictor in factor form produces fitted values almost identical to the grouped sample proportions. The sample frequency of 0 for $\text{ptl} = 3$ is estimated by the logistic regression model as $4.721032\text{e-}07$. This can be expected, since the factor model has four degrees of freedom, and can therefore estimate the appropriate probabilities independently of each other.
- (d) The following code produces the required fitted model (Figure 23.4). Risk for low birth weight decreases with age of mother, and increases with a previous history of premature labors. The group at highest risk for low birth weight deliveries are younger mothers with a previous history of premature labors.

```

> ### (d)
>
> # Indicator predictor
> fit.mult = glm(low ~ I(ptl > 0)+age, data = birthwt, family='binomial')
> cf.mult = summary(fit.mult)$coef
> cf.mult
            Estimate Std. Error      z value Pr(>|z|)
(Intercept) 0.53224661 0.77459803  0.6871262 0.4920032031
I(ptl > 0)TRUE 1.60766387 0.42986071  3.7399647 0.0001840461
age         -0.07054101 0.03411005 -2.0680417 0.0386360997
>
>
> age.range = range(birthwt$age)
> age.grid = seq(age.range[1],age.range[2],0.1)
>
> pdf('A3-fig1.pdf')
>
> new.data.0 = data.frame(age=age.grid, ptl=0)
> new.data.1 = data.frame(age=age.grid, ptl=1)
>
> matplot(age.grid,cbind(logistic(predict(fit.mult, newdata=new.data.0)),
+                         logistic(predict(fit.mult, newdata=new.data.1))),
+          col=c('green','red'),type='l',lty=1,

```

```

+      xlab='Age of Mother (year)',ylab='P(low birth weight)', ylim=c(0,1))
>
> legend('topright',legend=c('0 premature labors', '>0 premature labor'),
+         col=c('green','red'),lty=1)
> dev.off()
RStudioGD
2
>
```

- (e) The functional form for the model is

$$\begin{aligned} P(\text{low birth weight}) &= \frac{e^{\beta_0 + \beta_1 I\{\text{ptl}>0\} + \beta_2 \text{age}}}{1 + e^{\beta_0 + \beta_1 I\{\text{ptl}>0\} + \beta_2 \text{age}}} \\ &= \frac{1}{1 + e^{-\beta_0 - \beta_1 I\{\text{ptl}>0\} - \beta_2 \text{age}}}. \end{aligned}$$

(Either form is fine). In general, of a probability is given in the form

$$P = \frac{e^\eta}{1 + e^\eta}$$

then the associated odds is

$$Odds = \frac{P}{1 - P} = \frac{\frac{e^\eta}{1 + e^\eta}}{1 - \frac{e^\eta}{1 + e^\eta}} = \frac{\frac{e^\eta}{1 + e^\eta}}{\frac{1}{1 + e^\eta}} = e^\eta.$$

Then let η_i , $i = 1, 2$ be the linear predictor term for two sets of predictor values. The odds ratio for low birth weight between predictor values η_1, η_2 is then

$$OR = \frac{Odds(\text{low birth weight} | \eta_1)}{Odds(\text{low birth weight} | \eta_2)} = \frac{e^{\eta_1}}{e^{\eta_2}} = e^{\eta_1 - \eta_2}.$$

Now, suppose two subjects $i = 1, 2$ have the same age A , but $\text{ptl} > 0$ for subject $i = 1$ and $\text{ptl} = 0$ for subject $i = 2$. Then

$$\eta_1 - \eta_2 = [\beta_0 + \beta_1 \times 1 + \beta_2 A] - [\beta_0 + \beta_1 \times 0 + \beta_2 A] = \beta_1,$$

so that the odds ratio

$$OR = e^{\beta_1} \approx e^{1.6077} = 4.991$$

does not depend on age. An approximate 95% confidence interval for β_1 is

$$CI_{\beta_1} = \hat{\beta}_1 \pm 2SE = 1.6077 \pm 2 \times 0.43 = 1.6077 \pm 0.86 = [0.7477, 2.4677].$$

The confidence interval for OR is therefore

$$CI_{OR} = e^{\hat{\beta}_1 \pm 2SE} = [e^{0.7477}, e^{2.4677}] = [2.11, 11.80].$$

- (f) The following code produces the required tables

```

> ### (f)
>
> bw2 = subset(birthwt, age <= 25)
> tab = table(bw2$low,bw2$ptl > 0)
> dimnames(tab) = list(c('low=0','low=1'),c('ptl=0','ptl>0'))
> tab
  ptl=0 ptl>0
low=0    82     7
low=1    33    13
>
> bw2 = subset(birthwt, age > 25)
> tab = table(bw2$low,bw2$ptl > 0)
> dimnames(tab) = list(c('low=0','low=1'),c('ptl=0','ptl>0'))
> tab
  ptl=0 ptl>0
low=0    36     5
low=1     8     5
>
```

From the table for $\text{age} \leq 25$ we have

$$\begin{aligned} P(\text{low birth weight} | \text{ptl} = 0) &= \frac{33}{33 + 82} \approx 0.287 \\ P(\text{low birth weight} | \text{ptl} > 0) &= \frac{13}{13 + 7} = 0.65 \\ OR &= \frac{82 \times 13}{33 \times 7} \approx 4.615. \end{aligned}$$

From the table for $\text{age} > 25$ we have

$$\begin{aligned} P(\text{low birth weight} | \text{ptl} = 0) &= \frac{8}{8 + 36} \approx 0.1818 \\ P(\text{low birth weight} | \text{ptl} > 0) &= \frac{5}{5 + 5} = 0.5 \\ OR &= \frac{82 \times 13}{33 \times 7} = 4.5. \end{aligned}$$

(g) The following code produces the required fitted values

```

> ### (g)
>
> # the required fitted values
>
> new.data.0 = data.frame(age=c(14,25,45), ptl=0)
> new.data.1 = data.frame(age=c(14,25,45), ptl=1)
> logistic(predict(fit.mult, newdata=new.data.0))
      1         2         3
0.38809484 0.22595771 0.06647766
```

```
> logistic(predict(fit.mult, newdata=new.data.1))
      1       2       3
0.7599374 0.5930010 0.2622252
>
```

The relevant probabilities are:

$$\begin{aligned}P(\text{low birth weight} \mid \text{ptl} = 0, \text{age} = 14) &= 0.3881 \\P(\text{low birth weight} \mid \text{ptl} = 0, \text{age} = 25) &= 0.2260 \\P(\text{low birth weight} \mid \text{ptl} = 0, \text{age} = 45) &= 0.0665 \\P(\text{low birth weight} \mid \text{ptl} > 0, \text{age} = 14) &= 0.7599 \\P(\text{low birth weight} \mid \text{ptl} > 0, \text{age} = 25) &= 0.5930 \\P(\text{low birth weight} \mid \text{ptl} > 0, \text{age} = 45) &= 0.2622\end{aligned}$$

The required ordering rule holds for $\text{age} \leq 25$

$$\begin{aligned}P(\text{low birth weight} \mid \text{ptl} = 0) &= \frac{33}{33 + 82} \approx 0.287 \in (0.2260, 0.3881) \\P(\text{low birth weight} \mid \text{ptl} > 0) &= \frac{13}{13 + 7} = 0.65 \in (0.5930, 0.7599)\end{aligned}$$

The required ordering rule holds for $\text{age} > 25$

$$\begin{aligned}P(\text{low birth weight} \mid \text{ptl} = 0) &= \frac{8}{8 + 36} \approx 0.1818 \in (0.0665, 0.2260) \\P(\text{low birth weight} \mid \text{ptl} > 0) &= \frac{5}{5 + 5} = 0.5 \in (0.2622, 0.5930)\end{aligned}$$

- (h) The *ORs* for $\text{age} \leq 25$ and $\text{age} > 25$ are 4.615 and 4.5 respectively. These are both close to the estimated value 4.991 obtained from the fitted model, and well within the confidence interval [2.11, 11.80]. Thus, that the OR does not depend on age is supported by this analysis.

Problem 23.6. For this problem use data set `Auto` from the `ISLR` package, which contains MPG; number of cylinders; model year; origin of car; and other information for 392 vehicles. Suppose a given application requires a prediction as to whether or not the model year of a vehicle is 1975 or later, bases on minimum technical specifications. The reasoning here is that a very low or very high MPG may be enough to give an accurate prediction (if not, more information would be collected).

- (a) Create a new data frame containing an indicator function `Y` which equals 1 for model year ≥ 1975 . Also retain only vehicles for which `origin == 1` (that is, we will only consider American cars).

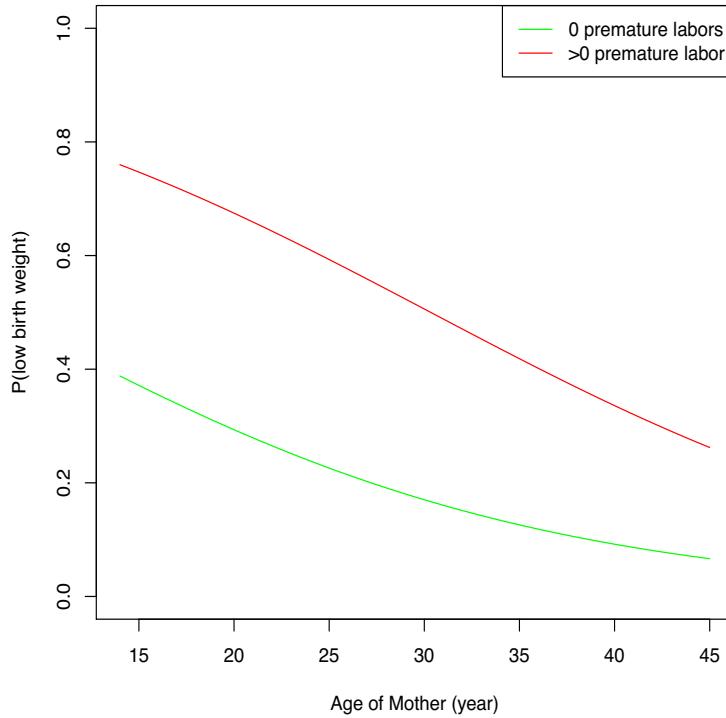


Figure 23.4: Figure for Problem 23.5 (d).

- (b) Fit a logistic regression model with binary response Y and predictor variable mpg . Report the standard coefficient table (ie. for each $\hat{\beta}_0, \hat{\beta}_1$ there are columns for Estimate, Standard Error, Z-score and P -value).
- (c) Create a graph with the following elements.
- (i) Each observed pair (mpg_i, Y_i) is plotted separately (the option `pch=3` works well for this).
 - (ii) The function $f(\text{mpg}) = E[Y | \text{mpg}] = P(1975+ | \text{mpg})$ is plotted. Use the `predict()` function. This has several advantages. First, we don't need to rely on the fitted values to construct the plot, since `predict()` will calculate fitted values for any predictor values, using the `newdata` option (use a list or data frame with consistent variable names). So, we can use an evenly spaced grid for the mpg axis. Second, we can get standard errors for the fits using the `se=T` option. This gives approximate 95% confidence bands after adding $\pm 2SE$ to each fitted value. Usually, the confidence bounds are drawn using dashed lines. Use something like the following commands (data in data frame `auto2`) to add the fitted curve to the plot:

```
logistic = function(x) {(1+exp(-x))^-1}
fit0 = glm(Y ~ mpg, family='binomial', data=auto2)
mpg.range = seq(min(auto2$mpg), max(auto2$mpg), 0.1)
```

```

pr = predict(fit0,newdata=list(mpg=mpg.range),se=T)
lines(mpg.range,logistic(pr$fit))
lines(mpg.range,logistic(pr$fit-2*pr$se.fit),lty=2)
lines(mpg.range,logistic(pr$fit+2*pr$se.fit),lty=2)

```

Note that `predict()` gives the linear predictor $\eta = X\beta$. You have to apply the logistic function yourself.

- (d) Construct side-by-side boxplots of `mpg` for each of the six combinations of `Y` and `cylinders`. You can use the formula `mpg ~ Y*cylinders` inside the `boxplot()` function, as long as you set the `data` option correctly. Examining the boxplot, would `mpg` = 20 be strong evidence that the model year was ≥ 1975 for a 4 cylinder vehicle? What about an 8 cylinder vehicle?
- (e) Expand your fit to include the new predictor `cylinder` as a factor. Note that `cylinder` has mode `numeric`, so it must be explicitly converted to a factor. This can be done directly to the data frame, or within the model formula, using `Y ~ mpg*as.factor(cylinders)`. This will essentially add two indicator functions to the model:

```

as.factor(cylinders)6 = I{cylinders == 6}
as.factor(cylinders)8 = I{cylinders == 8}

```

These indicator variables will appear as separate terms, and as interactions with `mpg`. These will appear in the resulting coefficient table.

- (f) Create a plot with the same axes as that of part (c):
 - (i) Include the observed pairs (mpg_i, Y_i) , but use separate colors or symbols for each cylinder level.
 - (ii) Plot the same function $f(mpg) = E[Y | mpg] = P(1975+ | mpg)$ as in part (c) (ie, the original fit with `mpg` only), but without the confidence bounds.
 - (iii) For each cylinder level identify the `mpg` range.
 - (iv) Plot the function $f(mpg, cylinders) = E[Y | mpg, cylinders] = P(1975+ | mpg, cylinders)$ separately for each cylinder level 4,6,8. Use distinct colors or line types. Make sure each of the 4 curves is properly labeled (best to use the `legend()` function for this). No confidence bounds should be drawn.
 - (v) The `predict()` function should be used as in Part (c). However, for each cylinder level, only use the `mpg` range observed for that level. Note that the `newdata` object will need to include that cylinder level. For example, to draw the fitted curve for level `cylinders == 4`:

```

fit1 = glm(Y ~ mpg*as.factor(cylinders), family='binomial',data=auto2)
range.by.cylinder = tapply(auto2$mpg,auto2$cylinders,function(x) range(x))
mpg.col=2
mpg.type=4
mpg.range = seq(range.by.cylinder['4'][1],range.by.cylinder['4'][2],0.1)
ngrid = length(mpg.range)
pr = predict(fit1,newdata=list(mpg=mpg.range,
                               cylinders = rep(mpg.type,ngrid)),se=T)
lines(mpg.range,logistic(pr$fit),col=mpg.col,lwd=2)

```

For `mpg` = 20, what is $P(1975+ | mpg)$ without knowing the cylinder level? What is $P(1975+ |$

mpg, cylinders) for $\text{mpg} = 20$, for each cylinder level. Does adding `cylinders` improve the prediction accuracy?

SOLUTION:

The required script follows. The plots are given in Figure 23.5.

```
> par(mfrow=c(2,2))
>
> ### (a) Use American cars (some cars from Europe and Japan have 3 or 5 cylinders)
>
> Auto.ex = subset(Auto, origin==1)
> Y = 1*(Auto.ex$year >= 75)
> auto2 = data.frame(Y,Auto.ex)
>
>
> ### (b) summary() gives the required table
>
> fit0 = glm(Y ~ mpg, family='binomial',data=auto2)
> summary(fit0)
```

Call:

```
glm(formula = Y ~ mpg, family = "binomial", data = auto2)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.2952	-0.8416	0.3046	0.9324	1.6657

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-4.27126	0.66623	-6.411	1.44e-10 ***
mpg	0.24395	0.03617	6.744	1.54e-11 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 332.75 on 244 degrees of freedom
Residual deviance: 256.79 on 243 degrees of freedom
AIC: 260.79
```

Number of Fisher Scoring iterations: 5

```
>
> ### (c) Create main plot
>
```

```

> # main plot
> with(auto2,plot(mpg,Y,pch=3,xlab='MPG',ylab='P(1975+)',cex=0.5,col=1))
>
> # create new data for plotting fitted curve
>
> mpg.range = seq(min(auto2$mpg),max(auto2$mpg),0.1)
> pr = predict(fit0,newdata=list(mpg=mpg.range),se=T)
>
> # Plot fitted values, and fitted values +/- 2*SE.
> # For plotting, fitted values need to be transformed using the logistic function.
> # This is done AFTER the +/- 2*SE operation.
>
> lines(mpg.range,logistic(pr$fit))
> lines(mpg.range,logistic(pr$fit-2*pr$se.fit),lty=2)
> lines(mpg.range,logistic(pr$fit+2*pr$se.fit),lty=2)
>
> ##### (d)
>
> boxplot(mpg~Y*cylinders,data=auto2,xlab="(1 = 1975+).(# cylinders)",ylab='MPG')
> abline(h=20,col='gray')
>
> ##### (e)
>
> fit1 = glm(Y ~ mpg*as.factor(cylinders), family='binomial',data=auto2)
>
> ##### (f)
>
> # Get cylinder-specific MPG ranges
>
> range.by.cylinder = tapply(auto2$mpg,auto2$cylinders,function(x) range(x))
>
> # Create a cylinder color scheme
>
> col.cyl = 2*(auto2$cylinders==4) + 3*(auto2$cylinders==6) + 4*(auto2$cylinders==8)
>
> # Create main plot
>
> with(auto2,plot(mpg,Y,pch=3,xlab='MPG',ylab='P(1975+)',cex=0.5,col=col.cyl))
>
> # Draw MPG only fit
>
> mpg.range = seq(min(auto2$mpg),max(auto2$mpg),0.1)
> pr = predict(fit0,newdata=list(mpg=mpg.range),se=T)
> lines(mpg.range,logistic(pr$fit),lwd=2)
>
```

```

> # Add cylinder-specific fits
>
> mpg.col=2
> mpg.type=4
> mpg.range = seq(range.by.cylinder$'4'[1],range.by.cylinder$'4'[2],0.1)
> ngrid = length(mpg.range)
> pr = predict(fit1,newdata=list(mpg=mpg.range,cylinders = rep(mpg.type,ngrid)),se=T)
> lines(mpg.range,logistic(pr$fit),col=mpg.col,lwd=2)
>
> mpg.col=3
> mpg.type=6
> mpg.range = seq(range.by.cylinder$'6'[1],range.by.cylinder$'6'[2],0.1)
> ngrid = length(mpg.range)
> pr = predict(fit1,newdata=list(mpg=mpg.range,cylinders = rep(mpg.type,ngrid)),se=T)
> lines(mpg.range,logistic(pr$fit),col=mpg.col,lwd=2)
>
> mpg.col=4
> mpg.type=8
> mpg.range = seq(range.by.cylinder$'8'[1],range.by.cylinder$'8'[2],0.1)
> ngrid = length(mpg.range)
> pr = predict(fit1,newdata=list(mpg=mpg.range,cylinders = rep(mpg.type,ngrid)),se=T)
> lines(mpg.range,logistic(pr$fit),col=mpg.col,lwd=2)
>
> legend('right',legend=c(paste(c(4,6,8),'cylinder'),'All Vehicles'),
>        col=c(2:4,1),lty=1,cex=0.75)
>
> # P-value for cylinder factor
>
> aov01 = anova(fit0,fit1)
> aov01
Analysis of Deviance Table

Model 1: Y ~ mpg
Model 2: Y ~ mpg * as.factor(cylinders)\end{color}
  Resid. Df Resid. Dev Df Deviance
1      243     256.79
2      239     229.88  4     26.91
> pchisq(aov01[2,4],df=aov01[2,3],lower.tail = FALSE)
[1] 2.073389e-05
>
> ### Give predicted values of P(1975+) for MPG = 20, for each cylinder level
>
> logistic(predict(fit1,newdata=list(mpg=rep(20,3),cylinders = c(4,6,8))))
      1          2          3
0.2125593 0.7002243 0.9647786

```

>

- (a) See above.
- (b) See above. The output table is

```
Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.27126   0.66623  -6.411 1.44e-10 ***
mpg          0.24395   0.03617   6.744 1.54e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- (c) See above, and Figure 23.5.
- (d) For a 4 cylinder vehicle of model year 1975+, mpg = 20 would be very low, while for an 8 cylinder vehicle of model year less than 1975, mpg = 20 would be very high (see boxplots of Figure 23.5). Therefore, mpg=20 is not evidence of model year 1975+ for a 4 cylinder vehicle, but is strong evidence of model year 1975+ for an 8 cylinder vehicle.
- (e) See above.
- (f) See Figure 23.5. The distribution of mpg strongly depends on cylinders, so that adjusting for cylinders clearly improves the prediction. Including cylinders as a factor into the model increases the number of parameters (and therefore the model degrees of freedom) by 4. The anova() function (see above) calculates the change in deviance:

$$X^2 = Dev[mpg] - Dev[mpg * cylinders] = 26.91.$$

Under the null hypothesis that cylinders does not improve model prediction $X^2 \sim \chi^2_4$, which gives $P\text{-value } 2.1 \times 10^{-5}$.

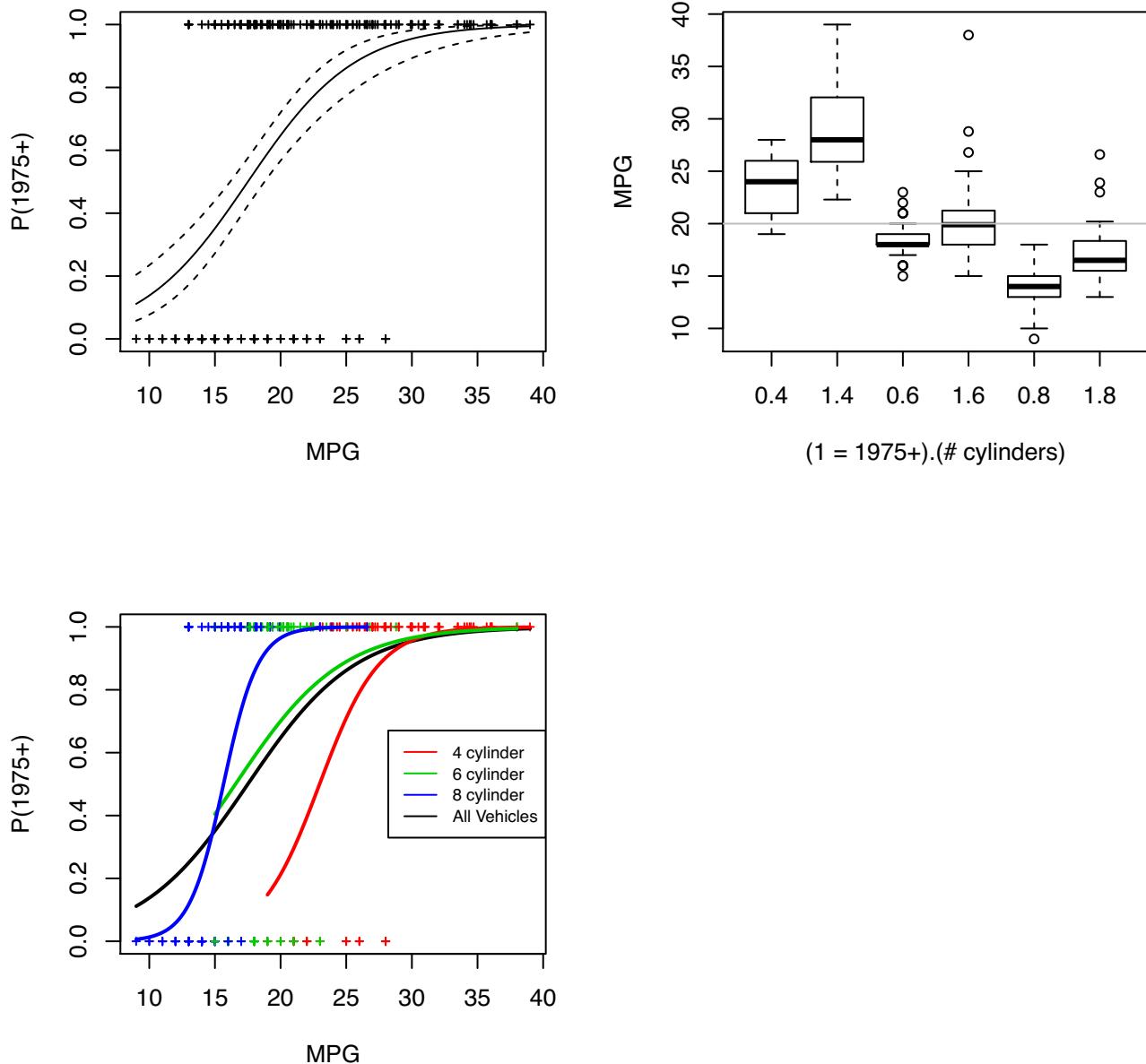


Figure 23.5: Plots for Problem 23.6.

Chapter 24

Practice Problems - Survival Analysis

24.1 Exercises

Problem 24.1. Suppose we observe survival times 21, 25+, 25, 27+, 34, 34, 35. Recall that the symbol ‘+’ denotes a right-censored observation. Construct and sketch a Kaplan-Meier estimate for the survival function.

SOLUTION:

For survival times 21, 25+, 25, 27+, 34, 34, 35 we have table:

i	t_i	d_i	$r(t_i)$	\hat{p}_i
0	0	0	7	$(7-0)/7 = 1$
1	21	1	7	$(7-1)/7 = 6/7$
2	25	1	6	$(6-1)/6 = 5/6$
3	27	0	4	$(4-0)/4 = 1$
4	34	2	3	$(3-2)/3 = 1/3$
5	35	1	1	$(1-1)/1 = 0$

Then plot the cumulative products

$$\hat{p}_0, \hat{p}_0\hat{p}_1, \hat{p}_0\hat{p}_1\hat{p}_2, \dots, \hat{p}_0\hat{p}_1 \times \dots \times \hat{p}_5 = 1, 6/7, 5/7, 5/7, 5/21, 0$$

at times

$$t_0, \dots, t_5 = 0, 21, 25, 27, 34, 35.$$

Note that ‘+’ indicates the position of a censored observation. See Figure 24.1.

Problem 24.2. Suppose we observe survival times 42, 51, 51, 51, 53+, 60, 60+, 64, where T_+ is a right-censored survival time.

- (a) Consider time points $(t_0, t_1, t_2, t_3, t_4, t_5) = (0, 42, 51, 53, 60, 64)$. For each of these times t' give the number *at risk* at time t' (the number of subjects with survival times $T \geq t'$ or $T_+ \geq t'$), and the number who die at time t' ($T = t'$). Summarize these quantities in a tabular form.

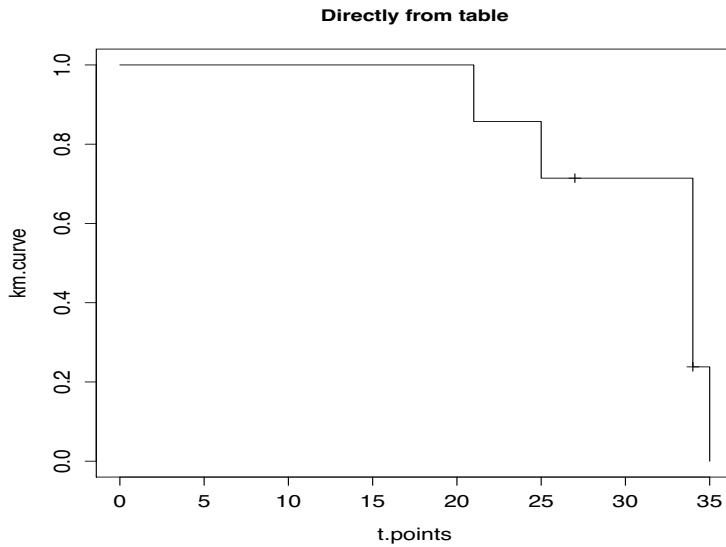


Figure 24.1: Kaplan-Meier estimate of survival function for Problem 24.1.

- (b) Plot a Kaplan-Meier estimate of the survival curve. Do this two ways:
- Plot the curve directly from the numbers in the table. Use the `pty='s'` option. Indicate all points on the curve at which an observation was censored. use a + symbol (`pch=3`).
 - Use the `survfit()` function. Set options `conf.int=FALSE` and `mark.time=TRUE`. Under what conditions are censored observations shown using the `mark.time=TRUE` option?

SOLUTION:

- (a) A completely observed survival time T is included in all *at risk totals* $t_i \leq T$, and the number who die at $t_i = T$. A censored observation $T+$ is included in all *at risk totals* $t_i \leq T$, but is not included in any death totals. This gives

i	t_i	d_i	$r(t_i)$	\hat{p}_i
0	0	0	8	$(8-0)/8 = 1$
1	42	1	8	$(8-1)/8 = 7/8$
2	51	3	7	$(7-3)/7 = 4/7$
3	53	0	4	$(4-0)/4 = 1$
4	60	1	3	$(3-1)/3 = 2/3$
5	64	1	1	$(1-1)/1 = 0$

- (b) To plot the Kaplan-Meier curve, plot cumulative products $\hat{p}_0, \hat{p}_0\hat{p}_1, \dots, \hat{p}_0\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_5$ against times t_0, \dots, t_5 . The plot should be a step function, using option `pty='s'`. Note that '+' indicates the position of a censored observation. The following code gives both plots. Note that if the option `mark.time=TRUE` is set, censored times $T+$ will be marked only if there is no other death which occurs at T . For this data, this means only the censored times $T = 53+$ is marked. See Figure 24.2.

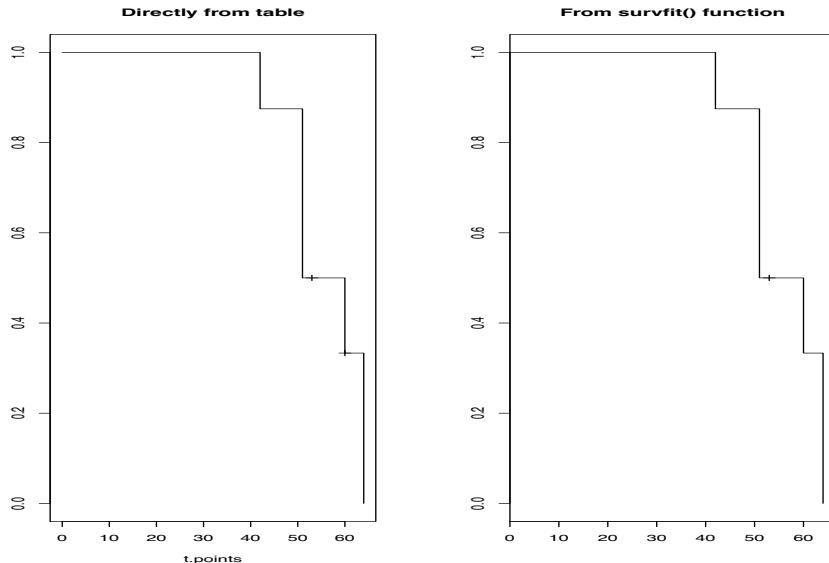


Figure 24.2: Kaplan-Meier estimate of survival function for Problem 24.2.

```

library(survival)
par(mfrow=c(1,2))

### Plot KM curve directly

t.points = c(0,42,51,53,60,64)
p.surv = c(1,7/8,4/7,1,2/3,0)
km.curve = cumprod(p.surv)
plot(t.points,km.curve,type='s')

# locate censored times:

points(t.points[c(4,5)],km.curve[c(4,5)],pch=3)

### Set up data, then plot KM curve directly

z = c(42, 51, 51, 51, 53, 60, 60,6/home/anthony/Desktop/STAT_LIB4)
ev = c(1,1,1,1,0,1,0,1)
plot(survfit(Surv(z,ev)^1),conf.int = F, mark.time=T)

```

Problem 24.3. The *Raleigh distribution* models positive random variables. It has one parameter $\sigma^2 > 0$ and a cumulative distribution function (CDF) given by

$$F(x) = 1 - e^{-x^2/(2\sigma^2)}, \quad x \geq 0.$$

Derive the survival function and the hazard function for the Raleigh distribution. Is a Raleigh survival time *new better than used* (NBU) or *new worse than used* (NWU)?

SOLUTION:

We have survival function

$$S(x) = 1 - F(X) = e^{-x^2/(2\sigma^2)}, \quad x \geq 0.$$

The density $f(x)$ is the derivative of F :

$$f(x) = \frac{d1 - e^{-x^2/(2\sigma^2)}}{dx} = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)},$$

so that the hazard function is

$$h(x) = \frac{f(x)}{S(x)} = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} \div e^{-x^2/(2\sigma^2)} = \frac{x}{\sigma^2}, \quad x \geq 0.$$

The hazard rate is increasing with x so the survival time is *new better than used* (NBU).

24.2 Data Analysis

Problem 24.4. Load the data frame `Aids2` from the library `MASS`. This data set contains survival data from Australian AIDS patients. The date of diagnosis (`diag`) and date of death or end of observation (`death`) are given separately, in Julian format with unspecified origin. The difference `death - diag` can be taken as the survival time. The factor `status` indicates whether the patient was alive or dead at the end of the observation period.

- (a) Use Cox proportional hazards regression to determine if survival time is related to age at diagnosis (variable `age`). Is higher age associated with higher or lower survival? What is the estimated hazards ratio between subjects aged 60 and 30 years at diagnosis?
- (b) Create a new variable indicating into which quartile a subject's age at diagnosis falls. The `cut` function can be used for this. It may be more convenient to first convert ages equal to 0 to, say, 0.1 years. Create Kaplan-Meier survival curves for each quartiles, displayed on a single plot (you can use the `survfit` function). Do the same for the cumulative hazard functions (use the `survfit` function with plot option `fun="cumhaz"`). Is the proportional hazards assumption used for Cox proportional hazards regression approximately valid?

SOLUTION:

The following R code can be used to set up the analysis:

```
library(survival)
library(MASS)
```

```

### set up new data frame

age2 = Aids2$age

### Create quartiles

age2[age2==0] = 0.1
br = c(0,quantile(age2,c(0.25,0.50,0.75,1)))
age.quartiles = cut(Aids2$age,breaks=br)

### Include time = death - diag

Aids3 = data.frame(Aids2,Aids2$death - Aids2$diag,as.integer(Aids2$status=="D"),
                    age.quartiles)
names(Aids3) = c(names(Aids2),'time','ev','ageQuartiles')

```

Run function `coxph()` to fit a Cox proportional hazards regression model.

```

> fit = coxph(Surv(time, ev) ~ age, data=Aids3)
> summary(fit)$coef
      coef  exp(coef)    se(coef)      z   Pr(>|z|)
age  0.01509529  1.01521  0.002449204 6.163344 7.122433e-10
>

```

From the output, the coefficient estimate for age is $\hat{\beta}_{age} = 0.015$, which corresponds to a hazards ratio of $\exp(\hat{\beta}_{age}) = 1.015$. The hazard rate therefore increases with age. The estimated hazard ratio between subjects aged 60 and 30 years at diagnosis is

$$\exp(\hat{\beta}_{age}(60 - 30)) = 1.573.$$

The following code can be used to create the plots. See Figure 24.3. If the hazard functions satisfy the proportionality assumption, the cumulative hazard functions should as well. From the plot, this assumption appears valid.

```

par(mfrow=c(1,2))
fit = survfit(Surv(time, ev)~ageQuartiles,data=Aids3)
plot(fit,col=c(1:4),xlab="Days",ylab='Survival')
legend('topright',legend=levels(Aids3$ageQuartiles),col=1:4,lty=1)
plot(fit,fun="cumhaz",col=c(1:4),xlab="Days",ylab='Cumulative Hazard')
legend('bottomright',legend=levels(Aids3$ageQuartiles),col=1:4,lty=1)

```

Problem 24.5. Load the data frame `leuk` from the library MASS. This data set contains `time` (survival time in weeks) and `wbc` (white blood counts) for $n = 33$ leukaemia patients. None of the survival times are censored (in this case `event` can be omitted from the `Surv` object). It

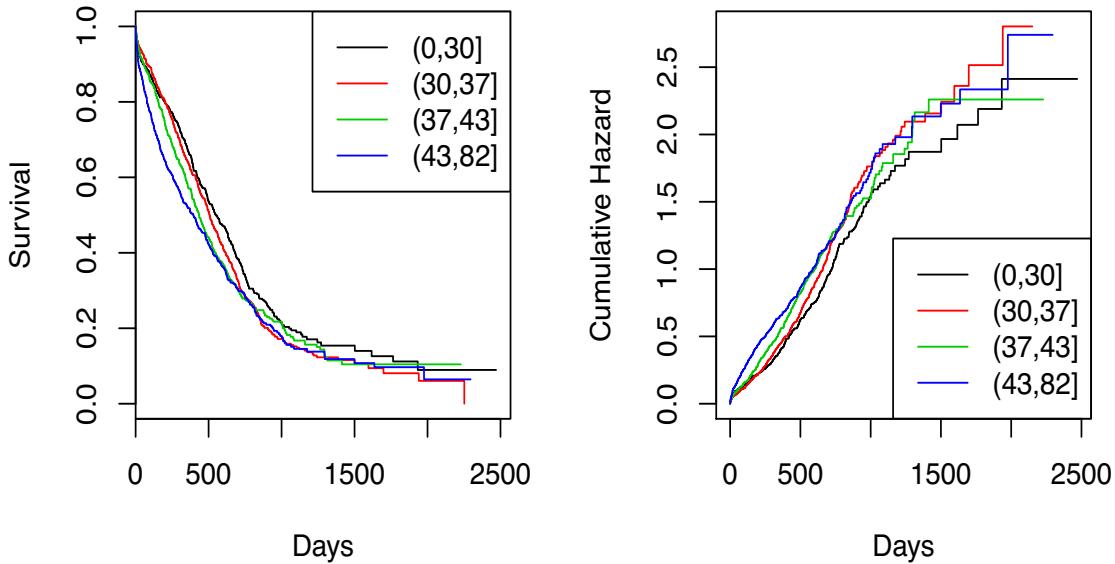


Figure 24.3: Plot for Problem 24.4.

also contains the variable `ag`. From the help file “[t]he patients were also factored into 2 groups according to the presence or absence of a morphologic characteristic of white blood cells. Patients termed AG positive were identified by the presence of Auer rods and/or significant granulation of the leukaemic cells in the bone marrow at the time of diagnosis”.

In the article Feigl, P. & Zelen, M. (1965) *Estimation of exponential survival probabilities with concomitant information*, Biometrics 21, 826–838, these lifetimes are assumed to be exponentially distributed with means λ^{-1} , where λ is allowed to depend on the predictors `wbc` and `ag`.

- (a) The Cox proportional hazards model predicts the hazard rate

$$h(x) = h_0(x)e^\eta, \quad x > 0,$$

where any predictor is incorporated into η , but not the baseline hazard rate $h_0(x)$. Therefore, a crucial assumption is that the hazard rate functions of all observations are proportional to h_0 , and therefore to each other. This can be checked when comparing a small number of groups, but is more difficult when using numerical predictors, since each survival time distribution is distinct, except for ties.

However, we can at least check the assumption approximately, by comparing the estimated hazard functions for the two levels of the factor `ag`, assuming provisionally that the survival times within each level are identically distributed.

- (i) Show that if the hazard rates are proportional, the cumulative hazard rates will be proportional as well.
- (ii) What is the cumulative hazard function $H(x)$ of an exponentially distributed lifetime of mean λ^{-1} ?
- (iii) If each survival time actually is exponentially distributed, with mean λ^{-1} dependent on `ag` and `wbc`, will the proportional hazard rate assumption be satisfied?

- (iv) Calculate Kaplan-Meier estimates of the survival curves separately for the two levels of factor `ag`. Plot the cumulative hazard functions. This can be done using essentially the same method used to plot the survival curves, except that the option `fun="cumhaz"` is used. Make sure the option `conf.int=TRUE` is used.
 - (v) Calculate the sample means of the survival times for each level of `ag`. Using these estimates, superimpose on the plot an estimate of the cumulative hazard function obtainable by assuming that survival times are exponentially distributed, and that the mean survival time is constant within each level of `ag`. Does the proportional hazards assumption seem reasonable?
- (b) Using function `cox.ph` fit the cox proportional hazards model with

$$\eta = \beta_1 ag + \beta_2 \log(wbc).$$

Assuming that survival times are exponentially distributed, use this model to estimate the proportion by which expected survival time is reduced by a 2-fold increase in white blood cell count.

SOLUTION:

- (a) (i) If $h(x)$ is the hazard rate, the cumulative hazard rate is the integral

$$H(x) = \int_{u=0}^x h(u)du.$$

If a second hazard rate $h^*(x)$ is proportion to $h(x)$, then we can write $h^*(x) = ch(x)$. The associated cumulative hazard rate is then

$$H^*(x) = \int_{u=0}^x h^*(u)du = \int_{u=0}^x ch(u)du = c \int_{u=0}^x h(u)du = cH(x),$$

so that proportionality is preserved.

- (ii) An exponentially distributed lifetime of mean λ^{-1} has a constant hazard rate λ , by the memoryless property. The cumulative hazard rate is therefore $H(x) = \lambda x$.
- (iii) If survival times are exponentially distributed, then all hazard rates are constants that depend on the predictors, so proportionality is satisfied.
- (iv) The following code can be used to construct the plots (Figure 24.4).

```
library(survival)
library(MASS)

par(mfrow=c(1,1))
plot(survfit(Surv(time) ~ ag, data = leuk), fun="cumhaz", lty = 1,
      col = 2:3, conf.int=T, xlab='Survival Time', ylab='Cumulative Hazard')
legend('bottomright', legend=c('absent', 'present'), lty=1, col=2:3)

### get means
```

```

means.ag = tapply(leuk$time,leuk$ag,mean)

### use the abline function to plot the cumulative hazard rates

abline(0,1/means.ag[1],col='red')
abline(0,1/means.ag[2],col='green')

```

- (v) The cumulative hazard rate estimates based on the exponential distribution assumption conform reasonably well to the Kaplan-Meier estimates, and fall within the confidence bands. The proportional hazards assumption is reasonable.
- (b) For the exponential distribution the hazard rate is constant, and equals the reciprocal of the mean. This means the mean survival time μ is given by

$$\begin{aligned}\mu^{-1} &= \mu_0^{-1} \times e^{\beta_1 ag + \beta_2 \log(wbc)} \\ &= \mu_0^{-1} \times e^{\beta_1 ag} \times (wbc)^{\beta_2},\end{aligned}$$

where μ_0 is the mean survival time associated with the baseline hazard rate. Then suppose two subjects have mean survival times μ_1, μ_2 , have the same factor level for `ag`, and have white blood cell counts $w_1 = w^*$ and $w_2 = 2w^*$ for some positive value w^* . Then

$$\frac{\mu_2}{\mu_1} = \frac{\mu_0^{-1} \times e^{\beta_1 ag} \times (w^*)^{\beta_2}}{\mu_0^{-1} \times e^{\beta_1 ag} \times (2w^*)^{\beta_2}} = \left(\frac{1}{2}\right)^{\beta_2}$$

where μ_0 is the mean survival under the baseline hazard rate. The coefficients can be estimated using the following code:

```

> leuk.cox = coxph(Surv(time) ~ ag + log(wbc), leuk)
> summary(leuk.cox)
Call:
coxph(formula = Surv(time) ~ ag + log(wbc), data = leuk)

n= 33, number of events= 33

            coef  exp(coef)  se(coef)      z Pr(>|z|)
agpresent -1.0691    0.3433   0.4293 -2.490  0.01276 *
log(wbc)    0.3677    1.4444   0.1360  2.703  0.00687 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

            exp(coef)  exp(-coef) lower .95 upper .95
agpresent    0.3433     2.9126    0.148    0.7964
log(wbc)     1.4444     0.6923    1.106    1.8857

Concordance= 0.726  (se = 0.065 )
Rsquare= 0.377  (max possible= 0.994 )
Likelihood ratio test= 15.64  on 2 df,  p=0.0004014

```

```

Wald test      = 15.06  on 2 df,   p=0.0005365
Score (logrank) test = 16.49  on 2 df,   p=0.0002629

```

From the coefficient table we have $\hat{\beta}_2 = 1.4444$. Therefore, the proportion by which expected survival time is reduced by a 2-fold increase in white blood cell count is approximately

$$\frac{\mu_2}{\mu_1} = \left(\frac{1}{2}\right)^{\beta_2} \approx \left(\frac{1}{2}\right)^{1.4444} = 0.367.$$

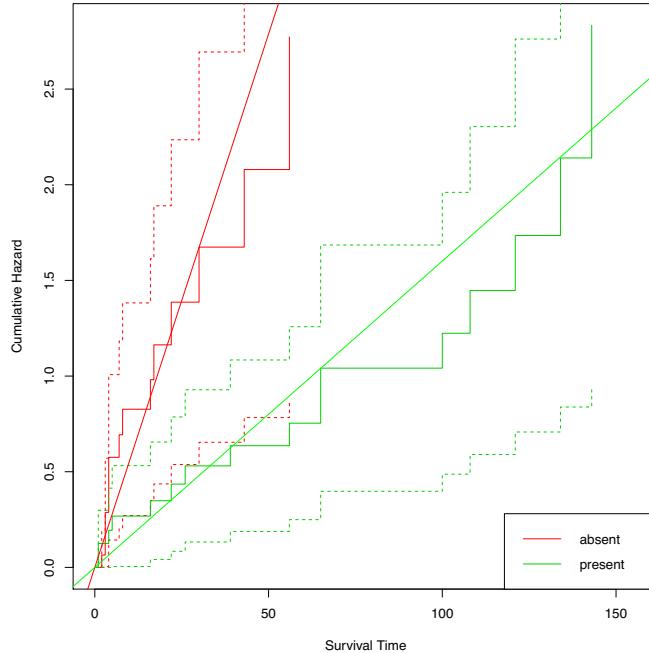


Figure 24.4: Plot for Problem 24.5 (a).

24.3 Theoretical Complements

Problem 24.6. Suppose $X \sim \exp(\lambda)$, that is, X is a random variable with density

$$f(x) = \lambda e^{-\lambda x} I\{x > 0\}.$$

- (a) A random survival time $T > 0$ is *memoryless* if

$$P(T > t + s \mid T > t) = P(T > s).$$

Essentially, the distribution of the remaining survival time at time t , given survival up to time t , is the same as the survival time from time 0. Prove that the exponentially distributed random variable is the unique memoryless continuous survival time with support $[0, \infty)$.

- (b) Prove that the exponentially distributed random variable is the unique continuous survival time with support $[0, \infty)$ that possesses a constant hazard rate.
- (c) *Reliability theory* is concerned with stochastic models which predict the survival time of a system. Suppose a system has m components, and component i has a random survival time of $X_i \sim \exp(\lambda_i)$, $i = 1, \dots, m$. A component or system *fails* immediately after its survival time expires. Assume the m component survival times are independent.
- We say the system is a *series* system if it fails as soon as any single component fails. Prove that a series system possesses a constant hazard rate of $h(t) = \sum_{i=1}^m \lambda_i$.
 - We say the system is a *parallel* system if it fails only after all m components fails. Prove that if $h(t)$ is the hazard rate of a parallel system, then $\lim_{t \rightarrow \infty} h(t) = \min_i \lambda_i$. In your proof it will be important to consider the following. Denote the minimum rate $\lambda^* = \min_i \lambda_i$, and let m^* denote the number of components with this rate. The proof will need to consider the possibility that $m^* > 1$, then show that the limit of $h(t)$ does not depend on this number. This implies that if the rates of the components are equal, then, rather counterintuitively, the limit of $h(t)$ does not depend on the number of components m . In addition, one approach is to use the cumulative hazard function, noting the identity $H(t) = -\log(S(t))$.
 - Suppose the components function as replacements. That is, they have a common rate $\lambda_i = \lambda$. Only one component functions at a time, and is replaced as soon as it fails. The system fails as soon as no functioning components remain. This means the system survival time is

$$T = X_1 + \dots + X_m.$$

Create plots of the hazard rates of T for $m = 2, 4, 6, 8$, setting $\lambda = 1$. Make use of the fact that the sum of *iid* exponential random variables has a gamma distribution. When you use the `pgamma()` function make sure you use the `lower.tail = FALSE`, that is, use

`pgamma(x, m, 1, lower.tail=FALSE)`

instead of

`1 - pgamma(x, m, 1, lower.tail=TRUE)`

to calculate the survival function $S(t) = 1 - F(t) = P(T > t)$. Add to your plots a horizontal line representing a hazard rate of 1. Use range $t \in (0, 200]$. What do the plots have in common?

- It can be shown that for any m , the limit of the hazard rate of T is $\lim_{t \rightarrow \infty} h(t) = 1$, the same hazard rate possessed by an individual component. Give an intuitive explanation for this.

SOLUTION:

- (a) A survival time X on $[0, \infty)$ is memoryless if and only if $P(X > t + s | X > t) = P(X > s)$, or $P(X > t + s) = P(X > s)P(X > t)$, for all $s, t \geq 0$. That this property is satisfied by the exponential distribution is easily verified by substituting $P(X > x) = 1 - F(x) = \bar{F}(x) = \exp(-\lambda x)$. To prove the converse, suppose a memoryless survival time on support $[0, \infty)$ has distribution function F . Letting $S(u) = \log(\bar{F}(u))$, the memoryless property implies

$S(t+s) = S(t) + S(s)$. Since S is monotone, a solution to this equation must be of the form $S(t) = ct$, which completes the proof.

- (b) Suppose $X \sim \exp(\lambda)$. The hazard rate is

$$h(t) = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} = \lambda.$$

Next, suppose X has constant hazard rate

$$h(t) = \frac{f(t)}{1 - F(t)} = \lambda,$$

where f and F are the density and CDF. Then

$$f(t) = \lambda(1 - F(t)) = \lambda S(t)$$

However, we have $dS(t)/dt = -f(t)$, giving differential equation

$$\frac{dS(t)}{dt} = -\lambda S(t)$$

Any solution to this differential equation is of the form $S(t) = ae^{-\lambda t}$. Since $S(t)$ is a survival function, we must have $a = 1$ and therefore $X \sim \exp(\lambda)$.

- (c) (i) For a series system, the survival time is $T = \min_i X_i$. Then

$$P(T > t) = P(\cap_i \{X_i > t\}) = \prod_i P(X_i > t) = \prod_i \exp(-\lambda_i t) = \exp\left(-t \sum_i \lambda_i\right).$$

Thus, T is exponentially distributed with rate $\sum_i \lambda_i$, and therefore possesses this constant hazard rate.

- (ii) For a parallel system, the survival time is $T = \max_i X_i$. Then

$$P(T \leq t) = P(\cap_i \{X_i \leq t\}) = \prod_i P(X_i \leq t) = \prod_i (1 - \exp(-\lambda_i t)).$$

Let $\lambda^* = \min_i \lambda_i$, and let m^* denote the number of components with this rate. Then

$$P(T > t) = 1 - P(T \leq t) = m^* e^{-\lambda^* t} + g(t)$$

where $g(t)$ is some linear combination of exponential functions with terms of the form ae^{-bt} , where $b > \lambda^*$. This means that for all large t , $g(t)$ becomes negligible compared to $m^* e^{-\lambda^* t}$ and therefore the cumulative hazard function is approximately

$$H(t) = -\log(S(t)) \approx \lambda^* t - \log m^*.$$

As t approaches ∞ the $-\log m^*$ term becomes negligible, and we have

$$H(t) \approx \lambda^* t.$$

We can conclude that as $t \rightarrow \infty$ the hazard rate approaches λ^* .

- (iii) The following code creates the plots (Figure 24.5). All hazard rates appear to approach 1 as t increases.

```
par(mfrow=c(2,2))
hf = function(x) {dgamma(x,m,1)/pgamma(x,m,1,lower.tail=FALSE)}
xgrid = seq(0,200,0.01)
for (m in c(2,4,6,8)) {
  plot(xgrid,hf(xgrid),type='l',ylim=c(0,1),main=paste('m = ',m),
    xlab=expression(italic(t)),ylab='hazard rate')
  abline(h=1,col='green')
}
```

- (iv) Suppose at time t the system has not yet failed. As $t \rightarrow \infty$ the probability that it is the last component m that is functioning approaches 1. Therefore, at this point the system hazard rate is dominated by the hazard rate of a single component, which is, by the memoryless property, equal to 1.

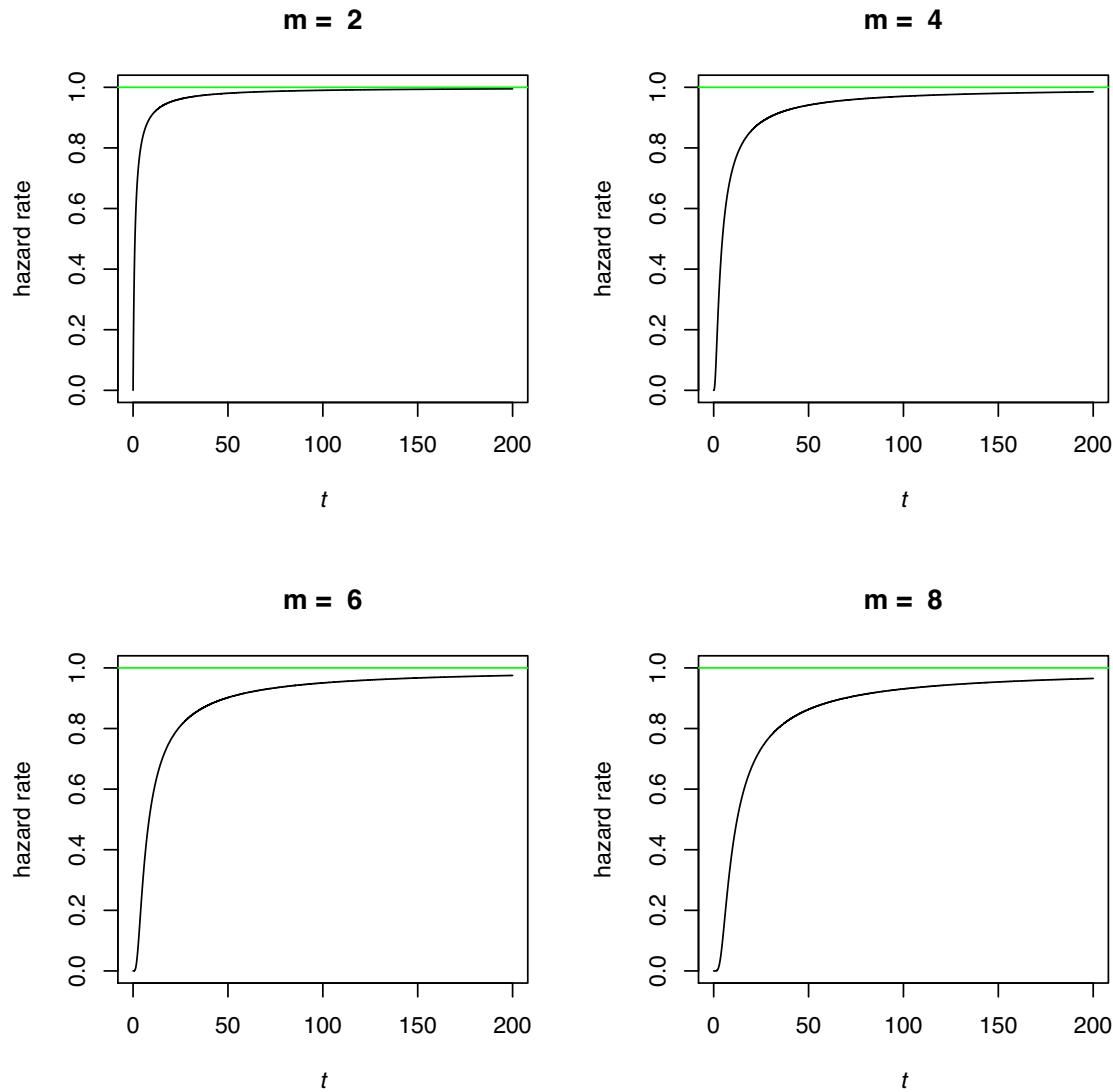


Figure 24.5: Plots for Problem 24.6.

Chapter 25

Practice Problems - Bayesian Inference

25.1 Exercises

Problem 25.1. The wheel of a car in a certain parking spot is marked with chalk. Suppose that the position of the chalk mark is recorded as a number from 1 to 12, corresponding to the face of a clock. It is noted that 3 hours later (in a parking zone with a 2 hour limit) the same car occupies the parking spot. Let

$$\begin{aligned} A &= \{ \text{The car has remained stationary for at least 3 hours} \} \\ E &= \{ \text{Chalk mark position has remained the same} \} \end{aligned}$$

Give $Odds(A | E)$ and $Odds(A | E^c)$. Assume that if the car has been driven, the chalk mark is equally likely to be in any of the 12 positions when it returns to the parking space.

SOLUTION:

Suppose the chalk mark position has remained the same. By Bayes Theorem, we have

$$Odds(A | E) = LR \times Odds(A),$$

where LR is the likelihood ratio

$$LR = \frac{P(E | A)}{P(E | A^c)} \approx \frac{1}{1/12} = 12.$$

This gives

$$Odds(A | E) = 12 \times Odds(A).$$

In contrast, if the chalk mark position has not remained the same, we have

$$Odds(A | E^c) = LR \times Odds(A),$$

where LR is the likelihood ratio

$$LR = \frac{P(E^c | A)}{P(E^c | A^c)} \approx \frac{0}{11/12} = 0.$$

This gives

$$Odds(A | E^c) = 0 \times Odds(A) = 0,$$

assuming $Odds(A) < \infty$.

Problem 25.2. A genotype found in a blood sample has a population frequency of 1/1000. A genotype from a suspect is extracted. Suppose that due to imperfect typing procedures if the suspect's blood is the same as that of the sample then the probability that the genotype is incorrectly typed is 0.01. Assume that this will be the only type of error, that is, the probability of a false match is negligible. Let A be the event that the suspect's blood is the same as that of the sample, and let E be the event that the suspect's observed genotype matches that of the sample. Give the posterior odds of A given evidence of the form E in terms of the prior odds of A . Also give the posterior odds of A^c given evidence of the form E^c in terms of the prior odds of A^c .

SOLUTION:

By Bayes Theorem, we have

$$Odds(A | E) = LR \times Odds(A),$$

where LR is the likelihood ratio

$$LR = \frac{P(E | A)}{P(E | A^c)} \approx \frac{1 - 0.01}{1/1000} = 990.$$

This gives

$$Odds(A | E) = 990 \times Odds(A).$$

Similarly,

$$Odds(A^c | E^c) = LR \times Odds(A^c),$$

where LR is the likelihood ratio

$$LR = \frac{P(E^c | A^c)}{P(E^c | A)} \approx \frac{1 - 1/1000}{0.01} = 99.9.$$

This gives

$$Odds(A^c | E^c) = 990 \times Odds(A^c).$$

Problem 25.3. The odds of an event A is denoted $Odds(A)$. Suppose the distribution of a random variable X depends on whether or not event A occurs. In particular, conditional on A , $X \sim bin(4, 0.5)$. Conditional on A^c , $X \sim bin(2, 0.9)$.

Determine the relationship between $Odds(A | X = x)$ and $Odds(A)$ for $x = 0, 1, 2, 3, 4$. For which values of x does evidence of the form $\{X = x\}$ increase the odds that A does not occur.

SOLUTION:

By Bayes Theorem, we have

$$Odds(A | X = x) = LR \times Odds(A),$$

where LR is the likelihood ratio

$$LR = \frac{P(X = x | A)}{P(X = x | A^c)}, \quad x = 0, 1, \dots, 4.$$

The PMF of $X \sim bin(n, p)$ is $p(x) = \binom{n}{x} p^x (1-p)^{n-x}$, $x = 0, 1, \dots, n$. However, where needed, we can set $p(x) = 0$ for any x not in the set $\{0, 1, \dots, n\}$. This gives

	$x =$	0	1	2	3	4
A	$p(x) =$	$\binom{4}{0} 0.5^4$	$\binom{4}{1} 0.5^4$	$\binom{4}{2} 0.5^4$	$\binom{4}{3} 0.5^4$	$\binom{4}{4} 0.5^4$
A^c	$p(x) =$	$\binom{2}{0} 0.9^0 0.1^2$	$\binom{2}{1} 0.9^1 0.1^1$	$\binom{2}{2} 0.9^2 0.1^0$	0	0
	LR	$(0.5^4)/(0.1^2)$ $= 5.25$	$(4 \times 0.5^4)/(2 \times 0.9 \times 0.1)$ ≈ 1.39	$(6 \times 0.5^4)/(0.9^2)$ ≈ 0.463	∞	∞

If $LR > 1$, the evidence increases the odds that A occurs, if $LR < 1$ the evidence increases the odds that A does not occur. The only value for which $LR < 1$ is $x = 2$.

25.2 Data Analysis

Problem 25.4. A test for a certain infection was evaluated experimentally. When administered to a test group of 285 individuals known to have the infection, the test was positive in 256 cases. The test was also administered to a control group of 220 subjects known to be free of the infection. The test was positive in 12 cases.

- (a) Estimate the sensitivity and specificity of the test directly from the data.
- (b) This test is intended to be used in clinical populations of varying infection prevalence. Use R to construct plots of PPV and NPV for values of prevalence ranging from 0 to 10%. Use the `type = 'l'` option of the `plot()` function.
- (c) Calculate prevalence, NPV and PPV directly from the data. How do these values compare to those shown in the plots of part (b)?

SOLUTION:

We can summarize the study with the following contingency table:

Table 1: Outcomes of test for Problem 25.4.

Diagnostic Test	Infection			Total
	Positive	Negative		
Positive	256	12		268
Negative	29	208		237
Total	285	220		505

(a) We have

$$\begin{aligned} sens &= \frac{TP}{TP + FN} = \frac{256}{285} \approx 0.898 \\ spec &= \frac{TN}{TN + FP} = \frac{208}{220} \approx 0.945. \end{aligned}$$

(b) Given our calculated values of *spec* and *sens* we can given *PPV* and *NPV* as functions of *prev*:

$$\begin{aligned} PPV &= \frac{sens \times prev}{sens \times prev + (1 - spec) \times (1 - prev)} \\ &= \frac{0.898 \times prev}{0.898 \times prev + (1 - 0.945) \times (1 - prev)} \end{aligned}$$

and

$$\begin{aligned} NPV &= \frac{spec \times (1 - prev)}{spec \times (1 - prev) + (1 - sens) \times prev} \\ &= \frac{0.945 \times (1 - prev)}{0.945 \times (1 - prev) + (1 - 0.898) \times prev} \end{aligned}$$

The plot itself can be constructed using the following code (see Figure 25.1 below):

```
### Calculate sens, spec as global variables

sens = 256/285
spec = 208/220

### Create functions for PPV and NPV

ppv0 = function(prev,sens,spec) { sens*prev/(sens*prev + (1-spec)*(1-prev)) }
npv0 = function(prev,sens,spec) { spec*(1-prev)/(spec*(1-prev) + (1-sens)*prev) }

### Create range of prev values
```

```

prev = seq(0,0.1,by = 0.001)

### Draw plots

par(mfrow=c(1,2),cex=1.0,cex.lab=1.0,cex.axis=1.0,mar=c(6,6,2,2),pty='s')
plot(prev, ppv0(prev,sens,spec),type='l', xlab='Prevalence',ylab='PPV')
plot(prev, npv0(prev,sens,spec),type='l', xlab='Prevalence',ylab='NPV')

```

(c) Directly from the table we have

$$\begin{aligned}
prev &= \frac{TP + FN}{N} = \frac{285}{505} \approx 0.564 \\
PPV &= \frac{TP}{TP + FP} = \frac{256}{268} \approx 0.955 \\
NPV &= \frac{TN}{TN + FN} = \frac{208}{237} \approx 0.878.
\end{aligned}$$

Directly from the data, the prevalence is $prev = 0.564$, which is much higher than the values used for the plot. Then $PPV = 0.955$ is much higher than the plotted values, and $NPV = 0.878$ is much lower than the plotted values. This is due to the dependence of PPV and NPV on the prevalence.

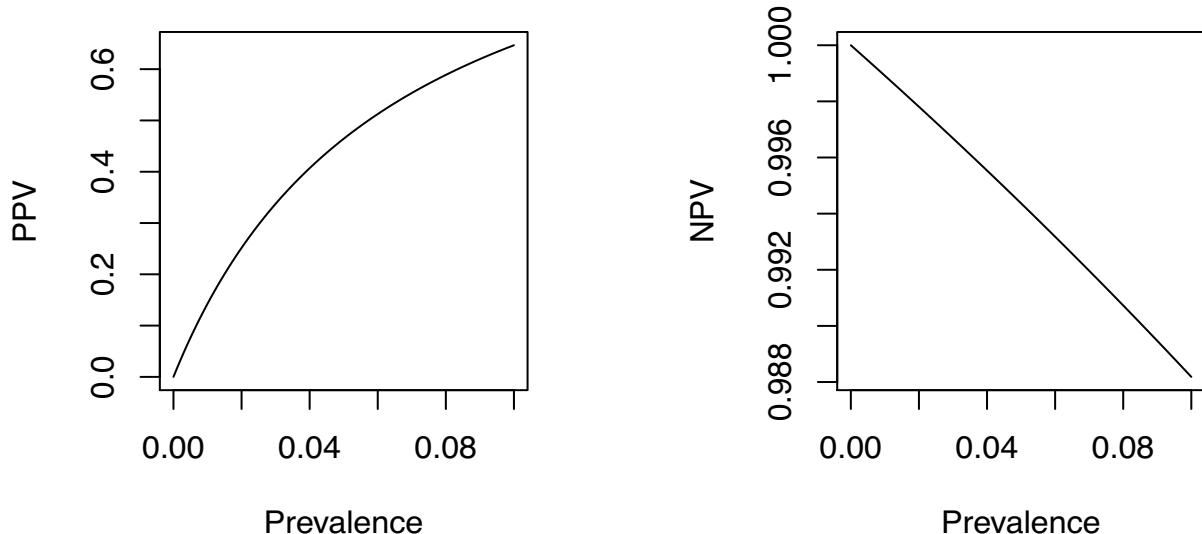


Figure 25.1: Plots for Problem 25.4.

Problem 25.5. A test for the presence of an infection is developed. The test is administered to a test group of 159 individuals known to have the infection. Of this group 154 test positive. The test is also administered to a control group of 325 subjects known to be free of the infection. Of these, 6 test positive.

- Estimate the sensitivity and specificity of the test directly from the data.
- Calculate prevalence, NPV and PPV directly from the data, then recalculate assuming a prevalence of 2%.
- Give the relationship between the prior and posterior odds of infection for both a positive and negative test result.

SOLUTION:

We can summarize the study with the following contingency table:

Table 2: Outcomes of test for Problem 25.5.

		Infection		Total
		Positive	Negative	
Diagnostic Test	Positive	154	6	160
	Negative	5	319	324
	Total	159	325	484

- (a) We have

$$\begin{aligned} sens &= \frac{TP}{TP + FN} = \frac{154}{159} \approx 0.969 \\ spec &= \frac{TN}{TN + FP} = \frac{319}{325} \approx 0.982. \end{aligned}$$

- (b) Directly from the table we have

$$\begin{aligned} prev &= \frac{TP + FN}{N} = \frac{159}{484} \approx 0.329 \\ PPV &= \frac{TP}{TP + FP} = \frac{154}{160} \approx 0.963 \\ NPV &= \frac{TN}{TN + FN} = \frac{319}{324} \approx 0.985. \end{aligned}$$

There is a prevalence of 32.9%, which is (hopefully) much higher than any prevalence we would expect to see in any population. If $prev = 0.02$, then

$$\begin{aligned} PPV &= \frac{sens \times prev}{sens \times prev + (1 - spec) \times (1 - prev)} \\ &= \frac{0.969 \times 0.02}{0.969 \times 0.02 + (1 - 0.982) \times (1 - 0.02)} \\ &\approx 0.517 \end{aligned}$$

and

$$\begin{aligned} NPV &= \frac{spec \times (1 - prev)}{spec \times (1 - prev) + (1 - sens) \times prev} \\ &= \frac{0.982 \times (1 - 0.02)}{0.982 \times (1 - 0.02) + (1 - 0.969) \times 0.02} \\ &\approx 0.999. \end{aligned}$$

(c) The positive and negative likelihood ratios are

$$\begin{aligned} LR_+ &= \frac{sens}{1 - spec} = \frac{154/159}{6/325} \approx 52.5, \\ LR_- &= \frac{1 - sens}{spec} = \frac{5/159}{319/325} \approx 0.032. \end{aligned}$$

Note that it is better to use the exact values of *sens* and *spec* in this calculation to avoid rounding error. Bayes' theorem for odds is then

$$\begin{aligned} Odds(O_+ | T_+) &= LR_+ \times Odds(O_+) = 52.5 \times Odds(O_+) \\ Odds(O_+ | T_-) &= LR_- \times Odds(O_+) = 0.032 \times Odds(O_+). \end{aligned}$$

Problem 25.6. A test for Hepatitis-B is developed. The test is administered to a test group of 147 individuals known to have Hepatitis-B. Of this group 123 test positive. The test is also administered to a control group of 220 subjects known to be free of Hepatitis-B. Of these, 15 test positive.

- (a) Estimate the sensitivity and specificity of the test directly from the data.
- (b) This test is intended to be used in clinical populations of varying infection prevalence. Use R to construct plots of *PPV* and *NPV* for values of prevalence ranging from 0 to 5%. Use the *type = 'l'* option of the *plot()* function.
- (c) Calculate prevalence, *NPV* and *PPV* directly from the data. How do these values compare to those shown in the plots of part (b)?

SOLUTION:

We can summarize the study with the following contingency table:

- (a) We have

$$\begin{aligned} sens &= \frac{TP}{TP + FN} = \frac{123}{147} \approx 0.837 \\ spec &= \frac{TN}{TN + FP} = \frac{205}{220} \approx 0.932. \end{aligned}$$

Table 3: Outcomes of hepatitis-B diagnostic test for Problem 25.6.

		Hepatitis-B		
		Positive	Negative	Total
Diagnostic Test	Positive	123	15	138
	Negative	24	205	229
	Total	147	220	367

- (b) The script shown below produces the plot in Figure 25.2.
(c) Directly from the table we have

$$\begin{aligned} prev &= \frac{TP + FN}{N} = \frac{147}{367} \approx 0.401 \\ PPV &= \frac{TP}{TP + FP} = \frac{123}{138} \approx 0.891 \\ NPV &= \frac{TN}{TN + FN} = \frac{205}{229} \approx 0.895. \end{aligned}$$

There is a prevalence of 40.1%, which is (hopefully) much higher than any prevalence we would expect to see in any population. The PPV estimated directly from the study is much higher than a PPV we would expect to encounter in a population, while the NPV is lower.

The following code produced Figure 25.2.

```
> sens = 123/147
> spec = (220-15)/220
>
> prev = seq(0,0.05,by = 0.001)
>
> par(mfrow=c(2,1),cex=1.0)
>
> f0 = function(prev,sens,spec) { sens*prev/(sens*prev + (1-spec)*(1-prev)) }
> plot(prev, f0(prev,sens,spec),type='l', xlab='Prevalence',ylab='PPV')
> title('Hepatitis-B Test')
>
>
> f0 = function(prev,sens,spec) { spec*(1-prev)/(spec*(1-prev) + (1-sens)*prev) }
> plot(prev, f0(prev,sens,spec),type='l', xlab='Prevalence',ylab='NPV')
> title('Hepatitis-B Test')
```

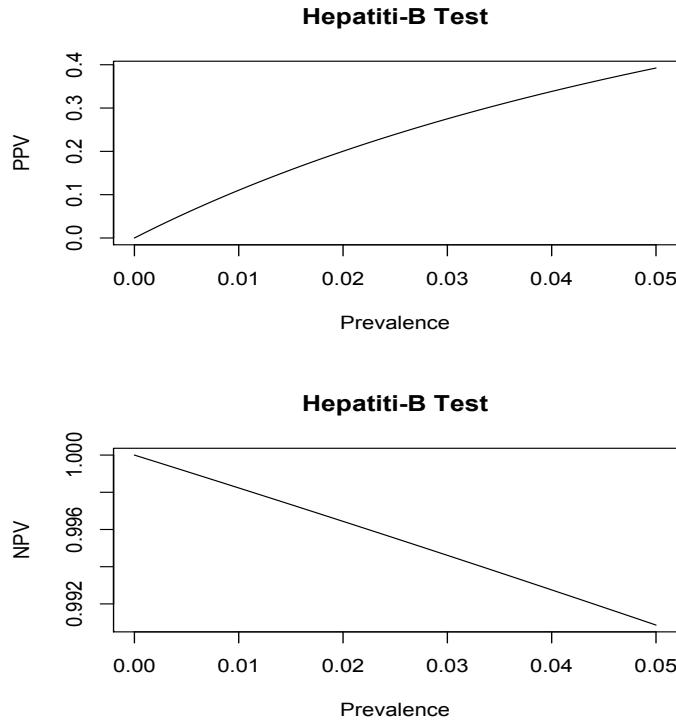


Figure 25.2: Plot for Problem 25.6.

25.3 Theoretical Complements

Problem 25.7. Suppose we observe a normally distributed random variable $X \sim N(\mu, \sigma)$. Assume σ is known, and that μ has a prior density $\pi(\mu)$:

$$\mu \sim N(\mu_0, \sigma_0),$$

for some fixed μ_0, σ_0 . Show that the normal prior density is a conjugate density for μ , that is, that the posterior density for μ given X is also normal. Give this density precisely.

SOLUTION:

Recall that to evaluate a posterior density of a parameter θ given data x it is often easiest to first express it as

$$\pi(\theta | x) = K g(\theta)$$

where K is a constant that does not depend on θ , and then normalize $g(\theta)$. This means we don't need to actually evaluate K . In this case we have

$$\pi(\mu | x) \propto f(x | \mu) \pi(\mu)$$

where $x \mid \mu \sim N(\mu, \sigma)$ and $\mu \sim N(\mu_0, \sigma_0)$. This means

$$\pi(\mu \mid x) = K e^{-\frac{1}{2}Q_1} e^{-\frac{1}{2}Q_0}$$

where

$$Q_1 = \frac{(x - \mu)^2}{\sigma^2}, \quad Q_0 = \frac{(\mu - \mu_0)^2}{\sigma_0^2}.$$

We then have

$$Q_1 + Q_0 = \mu^2 \left[\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2} \right] - 2\mu \left[\frac{x}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \right] + \left[\frac{x^2}{\sigma^2} + \frac{\mu_0^2}{\sigma_0^2} \right].$$

This means

$$\pi(\mu \mid x) = K e^{-\frac{1}{2} \left\{ \mu^2 \left[\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2} \right] - 2\mu \left[\frac{x}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \right] \right\}}$$

where K does not depend on μ , and that $\pi(\theta \mid x) \sim N(\mu_{post}, \sigma_{post}^2)$ is a normal density function with mean and variance

$$\begin{aligned} \mu_{post} &= \frac{\frac{x}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2}}, \\ \sigma_{post}^2 &= \frac{1}{\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2}}. \end{aligned}$$

Problem 25.8. The *gamma density*, denoted $Y \sim \text{gamma}(\alpha, \beta)$ is defined as

$$f(y \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} e^{-\beta y}$$

where $y, \alpha, \beta > 0$. Suppose we observe a Poisson random variable $X \sim \text{pois}(\lambda)$, so that

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda}, \quad x = 0, 1, 2, \dots$$

and that λ has a prior density $\pi(\lambda)$:

$$\lambda \sim \text{gamma}(\alpha_0, \beta_0)$$

for some fixed α_0, β_0 .

(a) Recall the definition of the gamma function

$$\Gamma(t) = \int_{u=0}^{\infty} u^{t-1} e^{-u} du, \quad t > 0.$$

Show that if $Y \sim \text{gamma}(\alpha, \beta)$ then for any $k > 0$

$$E[Y^k] = \beta^{-k} \frac{\Gamma(k + \alpha)}{\Gamma(\alpha)}.$$

Use this to derive the mean and variance of Y (recall that $\Gamma(t+1) = t\Gamma(t)$).

- (b) Show that the gamma prior density is a conjugate density for λ , that is, that the posterior density for λ given X is also gamma. Give this density precisely.
- (c) Suppose we observe a random sample X_1, \dots, X_n from a Poisson distribution with mean λ . Denote the sum $S = \sum_{i=1}^n X_i$. Define prior density $\lambda \sim \text{gamma}(\alpha_0, \beta_0)$. Show that the Bayes estimator under squared error loss for λ (that is, the mean of the posterior density) can be given by

$$\hat{\lambda}_{\text{Bayes}} = q\bar{X} + (1 - q)\frac{\alpha_0}{\beta_0}$$

where $\bar{X} = S/n$ and $q = n/(n + \beta_0)$, assuming that the posterior density is conditioned directly on S (recall that the sum of independent Poisson random variables is also a Poisson random variable).

- (d) A study is to use this model, and a sample size n is planned. Suppose prior knowledge suggests that $\lambda = \lambda^*$ for some fixed value λ^* . Since sample size can be taken as a measure of precision or certainty, the confidence in the prior belief can be expressed as a fraction of the proposed sample size, say $n^* = n/10$. What values α_0, β_0 would be appropriate for the prior density of λ in this situation?

SOLUTION:

- (a) Making use of variable substitution $u = \beta x$ we have

$$\begin{aligned} E[X^k] &= \int_{x=0}^{\infty} x^k f(x | \alpha, \beta) dx \\ &= \int_{x=0}^{\infty} x^k \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} dx \\ &= \int_{x=0}^{\infty} \frac{\beta^\alpha}{\Gamma(\alpha)} x^{k+\alpha-1} e^{-\beta x} dx \\ &= \int_{u=0}^{\infty} \frac{\beta^\alpha}{\Gamma(\alpha)} (u/\beta)^{k+\alpha-1} e^{-u} \beta^{-1} du \\ &= \frac{\beta^{-k}}{\Gamma(\alpha)} \int_{u=0}^{\infty} u^{k+\alpha-1} e^{-u} du \\ &= \beta^{-k} \frac{\Gamma(k+\alpha)}{\Gamma(\alpha)}. \end{aligned}$$

Making use of the identity $\Gamma(t+1) = t\Gamma(t)$ we have special cases

$$E[X] = \beta^{-1} \frac{\Gamma(1+\alpha)}{\Gamma(\alpha)} = \beta^{-1} \frac{\alpha\Gamma(\alpha)}{\Gamma(\alpha)} = \frac{\alpha}{\beta},$$

and

$$E[X^2] = \beta^{-2} \frac{\Gamma(2+\alpha)}{\Gamma(\alpha)} = \beta^{-2} \frac{(\alpha+1)\Gamma(\alpha+1)}{\Gamma(\alpha)} = \beta^{-2} \frac{(\alpha+1)\alpha\Gamma(\alpha)}{\Gamma(\alpha)} = \frac{(\alpha+1)\alpha}{\beta^2},$$

which gives variance

$$\sigma^2 = E[X^2] - E[X]^2 = \frac{(\alpha+1)\alpha}{\beta^2} - \left[\frac{\alpha}{\beta} \right]^2 = \frac{\alpha}{\beta^2}.$$

- (b) Recall that to evaluate a posterior density of a parameter θ given data x it is often easiest to first express it as

$$\pi(\theta | x) = K g(\theta)$$

where K is a constant that does not depend on θ , and then normalize $g(\theta)$. This means we don't need to actually evaluate K . In this case we have

$$\pi(\lambda | x) \propto f(x | \lambda) \pi(\lambda)$$

where $x | \lambda \sim \text{pois}(\lambda)$ and $\lambda \sim \text{gamma}(\alpha_0, \beta_0)$. This means

$$\begin{aligned}\pi(\lambda | x) &= K \lambda^x e^{-\lambda} \lambda^{\alpha_0-1} e^{-\beta_0 \lambda} \\ &= K \lambda^{x+\alpha_0-1} e^{-(\beta_0+1)\lambda},\end{aligned}$$

where K does not depend on λ . We conclude that the posterior density of λ given x is

$$\lambda | x \sim \text{gamma}(x + \alpha_0, \beta_0 + 1).$$

- (c) The posterior density $\pi(\lambda | S)$ is similar to that of Part (b), except that $S \sim \text{poisson}(n\lambda)$. This gives

$$\begin{aligned}\pi(\lambda | s) &= K(n\lambda)^s e^{-n\lambda} \lambda^{\alpha_0-1} e^{-\beta_0 \lambda} \\ &= K' \lambda^{s+\alpha_0-1} e^{-(\beta_0+n)\lambda},\end{aligned}$$

after incorporating into K' all factors that do not depend on λ . The posterior density is therefore $\text{gamma}(s + \alpha_0, \beta_0 + n)$. Then the Bayes estimator under squared error loss is the mean of the posterior density, which is, from Part (a):

$$\begin{aligned}\hat{\lambda}_{Bayes} &= \frac{S + \alpha_0}{\beta_0 + n} \\ &= \frac{n\bar{X} + \beta_0 \frac{\alpha_0}{\beta_0}}{\beta_0 + n} \\ &= q\bar{X} + (1-q)\frac{\alpha_0}{\beta_0}\end{aligned}$$

where $q = n/(n + \beta_0)$.

- (d) The mean of the prior density is α_0/β_0 , so set

$$\lambda^* = \frac{\alpha_0}{\beta_0}.$$

From Part (c), β_0 can be seen to function as a sample size, which accordingly weights the contribution of the prior density to the Bayes estimate. So set

$$\beta_0 = n^* = n/10,$$

so that $(\alpha_0, \beta_0) = (n^* \lambda^*, n^*) = (n\lambda^*/10, n/10)$.

Problem 25.9. Suppose we observe a binomial random variable $X \sim \text{bin}(n, p)$.

- (a) Show that the maximum likelihood estimate of p is $\hat{p}_{MLE} = X/n$.
- (b) Suppose in the context of Bayesian inference, we assign a $\text{beta}(\alpha, \beta)$ prior distribution to p . What is the expected value of p under the prior distribution (say \hat{p}_{prior}) and under the posterior distribution given observation $X = x$ (say \hat{p}_{post})?
- (c) Show that we can write

$$\hat{p}_{post} = q\hat{p}_{MLE} + (1 - q)\hat{p}_{prior}$$

where q depends only on α, β and n .

SOLUTION:

- (a) The density of the binomial is

$$P(X = x | p) = \binom{n}{x} p^x (1 - p)^{n-x}.$$

The log-likelihood is

$$L(p; x) = x \log(p) + (n - x) \log(1 - p) + C,$$

where C does not depend on p . Taking the derivative gives stationary conditions

$$\frac{dL(p; x)}{dp} = \frac{x}{p} - \frac{n - x}{1 - p} = 0.$$

It may then be verified that

$$\hat{p}_{MLE} = x/n$$

is a unique stationary point, and the unique global maximum of $L(p; x)$.

- (b) The beta density is given by

$$f(z | \alpha, \beta) = \frac{1}{B(\alpha, \beta)} z^{\alpha-1} (1 - z)^{\beta-1}, \quad z \in [0, 1],$$

and has expected value

$$\mu_{\alpha, \beta} = \frac{\alpha}{\alpha + \beta}.$$

For a beta prior, the prior mean of p is therefore

$$\hat{p}_{prior} = \frac{\alpha}{\alpha + \beta}.$$

The posterior density of p is then

$$\begin{aligned} \pi(p | x) &= \frac{P(X = x | p)\pi(p)}{\int_{p=0}^1 P(X = x | p)\pi(p)dp} \\ &= \frac{\binom{n}{x} p^x (1 - p)^{n-x} \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1 - p)^{\beta-1}}{\int_{p=0}^1 \binom{n}{x} p^x (1 - p)^{n-x} \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1 - p)^{\beta-1} dp}. \end{aligned}$$

Careful examination reveals that $\pi(p | x)$ is equivalent to a $\text{beta}(x + \alpha, n - x + \beta)$ density, therefore the posterior mean of p is

$$\hat{p}_{\text{post}} = \frac{x + \alpha}{n + \alpha + \beta}.$$

(c) We can write

$$\begin{aligned}\hat{p}_{\text{post}} &= \frac{x + \alpha}{n + \alpha + \beta} \\ &= \frac{x}{n + \alpha + \beta} + \frac{\alpha}{n + \alpha + \beta} \\ &= \frac{x}{n + \alpha + \beta} \left[\frac{n}{n} \right] + \frac{\alpha}{n + \alpha + \beta} \left[\frac{\alpha + \beta}{\alpha + \beta} \right] \\ &= \frac{x}{n} \left[\frac{n}{n + \alpha + \beta} \right] + \frac{\alpha}{\alpha + \beta} \left[\frac{\alpha + \beta}{n + \alpha + \beta} \right] \\ &= q\hat{p}_{\text{MLE}} + (1 - q)\hat{p}_{\text{prior}},\end{aligned}$$

where

$$q = \frac{n}{n + \alpha + \beta}.$$

Problem 25.10. Let Y be a random variable, and let X be any random observation $X = (X_1, \dots, X_n)$. Suppose we have the density of Y conditional on X , $f_{Y|X}(y | x)$, and the density of X , $f_X(x)$. The joint density of (X, Y) is then

$$f_{XY}(x, y) = f_{Y|X}(y | x)f_X(x).$$

Then for any function $h(x, y)$ denote the conditional expectation

$$E[h(X, Y) | X = x] = E[h(x, Y) | X = x] = \int_y h(x, y) f_{Y|X}(y | x) dy.$$

The *law of total expectation* states

$$\begin{aligned}E[h(X, Y)] &= \int_{x_1} \cdots \int_{x_n} \int_y h(x, y) f_{XY}(x, y) dy dx_1 \cdots dx_n \\ &= \int_{x_1} \cdots \int_{x_n} \int_y h(x, y) f_{Y|X}(y | x) f_X(x) dy dx_1 \cdots dx_n \\ &= \int_{x_1} \cdots \int_{x_n} f_X(x) \left[\int_y h(x, y) f_{Y|X}(y | x) dy \right] dx_1 \cdots dx_n \\ &= \int_{x_1} \cdots \int_{x_n} E[h(X, Y) | X = x] f_X(x) dx_1 \cdots dx_n,\end{aligned}$$

and can be informally summarized as $E[h(X, Y)] = E[E[h(X, Y) | X]]$.

The problem of prediction is to determine a function $\delta(X)$ which is closed to Y in some sense. We can defined *mean squared error* MSE_δ of δ as

$$MSE_\delta = E[(Y - \delta(X))^2],$$

and the *minimum mean squared error* (MMSE) predictor is the one which minimizes MSE_δ . Then recall that the Bayes (minimum expected risk) estimator for parameter θ under squared error loss is the expected value of θ under the posterior density. We can verify this by understanding how to derive an MMSE predictor. In the following development we will assume that all means and variances are finite, and in general this would need to be verified.

- (a) Prove that the constant c which minimizes $E[(Y - c)^2]$ is, uniquely, $c = E[Y]$.
- (b) Define the function

$$\mu_Y(x) = E[Y | X = x],$$

which gives the expected value of Y conditional on $\{X = x\}$. Using the result of Part (a), and the law of total expectation, show that the MMSE predictor of Y is $\delta(X) = \mu_Y(X)$.

- (c) Then derive the Bayes estimator for parameter θ under squared error loss.

SOLUTION:

- (a) We can write

$$E[(Y - c)^2] = E[Y^2 - 2cY + c^2] = E[Y^2] - 2cE[Y] + c^2.$$

This is a second order polynomial in c . Taking the derivative gives

$$\frac{dE[(Y - c)^2]}{dc} = -2E[Y] + 2c.$$

The second derivative is positive (equal to 2), so that $E[(Y - c)^2]$ is uniquely minimized by setting the first derivative equal to 0, giving $c = E[Y]$.

- (b) By the *law of total expectation* we have

$$\begin{aligned} MSE_\delta &= E[(Y - \delta(X))^2] \\ &= \int_{x_1} \cdots \int_{x_n} E[(Y - \delta(x))^2 | X = x] f_X(x) dx_1 \cdots dx_n. \end{aligned}$$

Consider the quantity $E[(Y - \delta(x))^2 | X = x]$ in the preceding integral. Since, conditional on $\{X = x\}$, $\delta(x)$ is a constant within the expectation, Part (a) applies. This means $E[(Y - \delta(x))^2 | X = x]$ is uniquely minimized by $E[Y | X = x] = \mu_Y(x)$. This means $MSE_\delta = E[(Y - \delta(X))^2]$ is uniquely minimized by setting $\delta(x) = \mu_Y(x)$ for all x .

- (c) In a Bayesian model, we have joint density

$$f_{\theta, X}(\theta, x) = f_{X|\theta}(x | \theta) \pi(\theta),$$

where $f_{X|\theta}(x | \theta)$ and $\pi(\theta)$ are the conditional and prior densities, respectively. By Part (b), the MMSE predictor of θ based on X , say $\delta(X)$, is the expected value of θ under the conditional density $\pi(\theta | x)$. But this is the posterior density, and MSE_δ is Bayes risk under squared error loss.

Chapter 26

Practice Problems - Simulation Methods

26.1 Exercises

Problem 26.1. The distributions of ratios of random variables are often difficult to evaluate, making formal inference methods for parameter ratios difficult to develop. The bootstrap procedure can be useful in this case.

Suppose we observe a sample of n paired observations (Y_1, Y_2) . The sample is independent, but the components within each pair might not be. The respective means are μ_1, μ_2 . Suppose we are interested in the solution to the equation:

$$\mu_1 + \mu_2 x = 165,$$

which is

$$x_0 = \frac{165 - \mu_1}{\mu_2}.$$

We estimate x_0 using the sample means

$$\hat{x}_0 = \frac{165 - \bar{Y}_1}{\bar{Y}_2}.$$

The problem is then to estimate the standard error $S_{\hat{x}_0}$. We will use a simulation study to investigate the accuracy of a bootstrap estimate of $S_{\hat{x}_0}$.

- (a) Write a function that returns a simulated independent sample of $Y = (Y_1, Y_2)$. We first assume that Y is bivariate normal with $\mu_1 = 90$, $\mu_2 = 120$, $\sigma_1^2 = \sigma_2^2 = 10$, and $cov(Y_1, Y_2) = 2.5$. The sample size is $n = 250$. Use function `rmvnorm()` from the `mvtnorm` library.
- (b) Write a function that returns the estimate \hat{x}_0 , given an $n \times 2$ matrix of data.
- (c) The first step is to estimate the true standard error by simulating the actual model (of course, in an actual inference problem, this step is not available). Simulate the model data $N_s = 10000$ times, in each case computing and storing the estimate \hat{x}_0 . The standard deviation of these estimates is approximately $S_{\hat{x}_0}$.

(d) Next, write the bootstrap function, which accepts as input the data matrix, and N_b , the number of bootstrap replicates. The function should first compute the sample size n . Then for each replicate take the following steps:

- (i) A new data set Y^* is created by sampling n observation pairs with replacement from the input data.
- (ii) A new estimate \hat{x}_0^* is calculated using Y^* and stored in an array.

The bootstrap estimate of $S_{\hat{x}_0}$ is the standard deviation of the N_b replicates \hat{x}_0^* .

(e) Finally, for a total of N_s replications, do the following steps:

- (i) Simulate a sample Y using the function of part (a).
- (ii) Use the function of part (d) to estimate $S_{\hat{x}_0}$.

Set $N_s = 1000$, and report the approximate true value of $S_{\hat{x}_0}$ calculated in part (c) and the median, 5th and 95th percentiles of the bootstrap estimates of $S_{\hat{x}_0}$ from part (e). Comment on the accuracy of the bootstrap procedure.

(f) Repeat part (e), this time setting (Y_1, Y_2) to be independent Poisson random variables with the same means $\mu_1 = 90$, $\mu_2 = 120$.

SOLUTION:

(a) We use `rmvnorm()` from the `mvtnorm` library.

```
give.data = function() {

  # set up mean vector and covariance matrix

  mu = c(90,120)
  Sigma = matrix(c(10,2.5,2.5,10),nrow=2)

  # sample size n = 250

  n = 250
  y = rmvnorm(n,mu,Sigma)
  return(y)

}
```

(b) The following function returns the required estimator:

```
give.est = function(y) {
  yest = apply(y,2,mean)
  est = (165-yest[1])/yest[2]
  return(est)
}
```

(c) The following function restimates $S_{\hat{x}_0}$:

```

n.true = 10000
est = numeric(n.true)
for (i in 1:n.true) {
  y = give.data()
  est[i] = give.est(y)
}
true.se = sd(est)

```

The estimated value of $S_{\hat{x}_0}$ is:

```

> true.se
[1] 0.002190398

```

- (d) The following function gives a bootstrap estimate of $S_{\hat{x}_0}$:

```

bs.fun = function(y,nb) {

  # determine sample size

  n = dim(y)[1]
  estb = numeric(nb)

  # generate nb bootstrap replicates, using function sample()
  # to resample the data with replacement

  for (i in 1:nb) {
    ind = sample(n,n,replace=T)
    yb = y[ind,]
    estb[i] = give.est(yb)
  }
  return(sd(estb))
}

```

- (e) The following code implements the simulation study:

```

# use nb = 2000 bootstrap replicates for each simulated data set

nb = 2000

# Do the simulation 1000 times

n.bs = 1000

# main loop

bs.se = numeric(n.bs)
for (i in 1:n.bs) {
  y = give.data()

```

```

    bs.se[i] = bs.fun(y,nb)
}

# Summarize the bootstrap estimates

se.quant = quantile(bs.se,c(0.05,0.95))
sm = c(true.se,se.quant)

```

The (approximately) true estimate of $S_{\hat{x}_0}$ is:

```
> true.se
[1] 0.002190398
```

and the bootstrap estimates are summarized by

```
> sm
      5%         95%
0.002161288 0.002003698 0.002342916
```

The bootstrap estimates are close to $S_{\hat{x}_0} \approx 0.00219$, usually within about 8%.

- (f) The function `give.data()` in part (a) can be replace by the following:

```

give.data = function() {

  # means and sample sizes

  mu = c(90,120)
  n = 250

  # generate the Poisson RVs

  y = cbind(rpois(n,lambda=mu[1]), rpois(n,lambda=mu[2]))
  return(y)
}

```

We can run the remaining code unaltered, giving summaries:

```
> true.se
[1] 0.006141413
> sm
      5%         95%
0.006141413 0.005673434 0.006650371
```

The bootstrap estimates are close to $S_{\hat{x}_0} \approx 0.00614$, again usually within about 8%.

Problem 26.2. Correlation between random variables can be modeled in the following way. Let X and ϵ be random variables with mean 0 and variance 1, and assume X and ϵ are independent. Then set, for constants $\beta_0, \beta_1, \beta_2$, a new random variable Y as follows:

$$Y = \beta_0 + \beta_1 X + \beta_2 \epsilon. \quad (26.1)$$

- (a) Derive an expression for the correlation between X and Y as a function of β_1 and β_2 (verify that this expression will not depend on β_0).
- (b) Suppose we wish to simulate pairs of random variables (X, Y) such that $\mu_X = \mu_Y = 0$, $\text{var}(X) = \text{var}(Y) = 1$, and the correlation between X and Y is fixed at $\rho \in (-1, 1)$. We can do this by simulating X and ϵ , then using Equation (26.1) to generate Y . To achieve this, what values must be used for $\beta_0, \beta_1, \beta_2$? Using R, generate four scatter plots from 1000 pairs of normally distributed random variables (X, Y) using this scheme, for $\rho = -0.5, 0, 0.5, 0.9$. Do an independent simulation for each of the four plots. Place all four scatter plots in one graphics window using the `par()` function. Indicate the relevant title for each plot, using the Greek font for ρ (consult `help(plotmath)` and the function `bquote()`). In addition, for each simulation, summarize in a table the sample variances and correlation of X and Y . Compare these sample values to the theoretical values.

SOLUTION:

- (a) The correlation is

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sqrt{\text{var}(X)\text{var}(Y)}} = \frac{E[XY] - \mu_X \mu_Y}{\sqrt{\text{var}(X)\text{var}(Y)}}.$$

We have, given what we know of ϵ , X and Y ,

$$\begin{aligned} \mu_X &= 0 \\ \mu_Y &= E[\beta_0 + \beta_1 X + \beta_2 \epsilon] = E[\beta_0] + \beta_1 E[X] + \beta_2 E[\epsilon] = \beta_0 + \beta_1 \times 0 + \beta_2 \times 0 = \beta_0 \\ \text{var}(X) &= 1 \\ \text{var}(Y) &= \text{var}(\beta_0) + \text{var}(\beta_1 X) + \text{var}(\beta_2 \epsilon) = 0 + \beta_1^2 + \beta_2^2 = \beta_1^2 + \beta_2^2 \\ E[XY] &= E[\beta_0 X + \beta_1 X^2 + \beta_2 \epsilon X] = \beta_0 E[X] + \beta_1 E[X^2] + \beta_2 E[\epsilon] E[X] = 0 + \beta_1 + 0 = \beta_1. \end{aligned}$$

So,

$$\rho_{XY} = \frac{\beta_1 - 0 \times \beta_0}{\sqrt{\beta_1^2 + \beta_2^2}} = \frac{\beta_1}{\sqrt{\beta_1^2 + \beta_2^2}}.$$

- (b) We have

$$\begin{aligned} \text{var}(Y) &= \beta_1^2 + \beta_2^2, \text{ and} \\ \rho &= \frac{\beta_1}{\sqrt{\beta_1^2 + \beta_2^2}} = \frac{\beta_1}{\sqrt{\text{var}(Y)}}. \end{aligned}$$

If $\text{var}(Y) = 1$, this gives (we can assume without loss of generality that $\beta_2 > 0$),

$$\begin{aligned} \beta_1 &= \text{var}(Y)^{1/2} \rho = \rho, \text{ and} \\ \beta_2 &= \sqrt{\text{var}(Y) - \beta_1^2} = \sqrt{1 - \rho^2} \end{aligned}$$

The following R script produces the required plot (Figure 26.1):

```

rho.list = c(-0.5, 0, 0.5, 0.9)
sumtab = matrix(NA, 4, 3)
par(mfrow=c(2,2))
for (i in 1:4) {
  rho = rho.list[i]
  eps = rnorm(1000)
  x = rnorm(1000)
  y = rho*x + sqrt(1 - rho^2)*eps
  plot(x,y, main = bquote(rho == .(rho)))
  sumtab[i,] = c(var(x), var(y), cor(x,y))
}

```

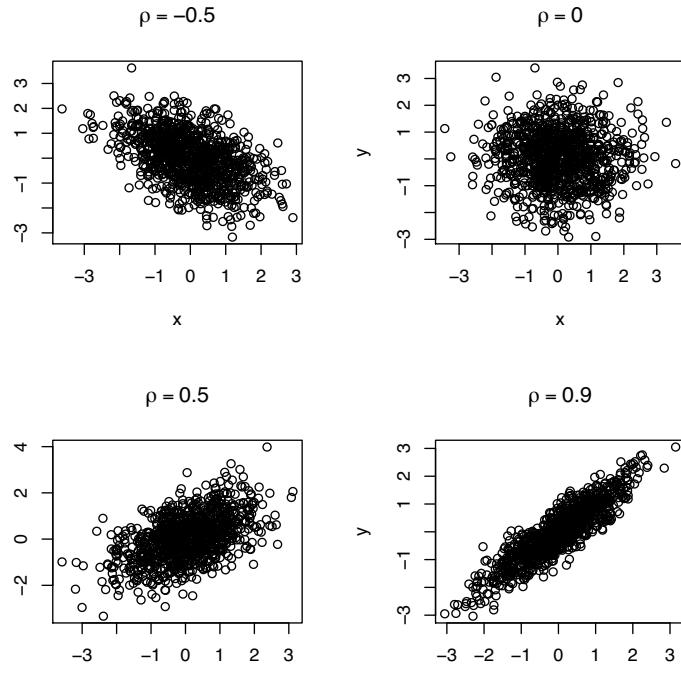


Figure 26.1: Plot for Problem 26.2.

The table is

```

> sumtab
      [,1]      [,2]      [,3]
[1,] 1.0364338 1.0496114 -0.5117990
[2,] 0.9644771 0.9503838 -0.0389420
[3,] 1.0288339 1.0237045  0.5053358
[4,] 0.9148030 0.9224029  0.8937180

```

The sample variances and correlations are all close to the theoretical values.

Problem 26.3. If r is a Pearson correlation coefficient for a bivariate normal random sample of size n , then if the true correlation is $\rho = 0$, then the transformed statistic

$$T = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}} \quad (26.2)$$

has an approximate t -distribution with $n-2$ degrees of freedom. We wish to investigate the accuracy of this approximation when applied to the Spearman rank correlation coefficient.

- (a) The R function `sample(n)` outputs a random permutation of the numbers $1, 2, \dots, n$. How can this be used to simulate the Spearman rank correlation coefficient when the true correlation is $\rho = 0$?
- (b) Simulate a random sample (of size 10,000) of Spearman rank correlation coefficients for $n = 25$ under the null hypothesis $H_0 : \rho = 0$.
- (c) Calculate the t -distribution transformation of (26.2) for each simulated value.
- (d) Plot a histogram of the transformed sample. Superimpose a t -density on the same plot, using the appropriate degrees of freedom. Is the t -distribution an accurate approximation? [Make sure you use the `probability = T` option when you plot the histogram].

SOLUTION:

- (a) Suppose we have output `x = sample(25)`. Then set `y = 1:25`. Calculate the correlation of `x` and `y` (you can use either the `method = 'spearman'` or the default `method = 'pearson'` method).
- (b) The following code creates the sample:

```
f0 = function(i) {
  x = sample(25)
  y = 1:25
  return(cor(x,y,method='spearman'))
}
cor.sample = sapply(1:10000,f0)
```

- (c) The following function calculates the transformation:

```
f.transform = function(x) {x/sqrt((1-x^2)/23)}
```

- (d) The following code creates the plot (Figure 26.2):

```
t.sample = f.transform(cor.sample)
tgrid = seq(-3,3,0.01)
hist(f.transform(cor.sample),probability=T,nclass=15)
lines(tgrid,dt(tgrid,df=23))
```

The t -distribution (with 23 degrees of freedom) accurately models the distribution.

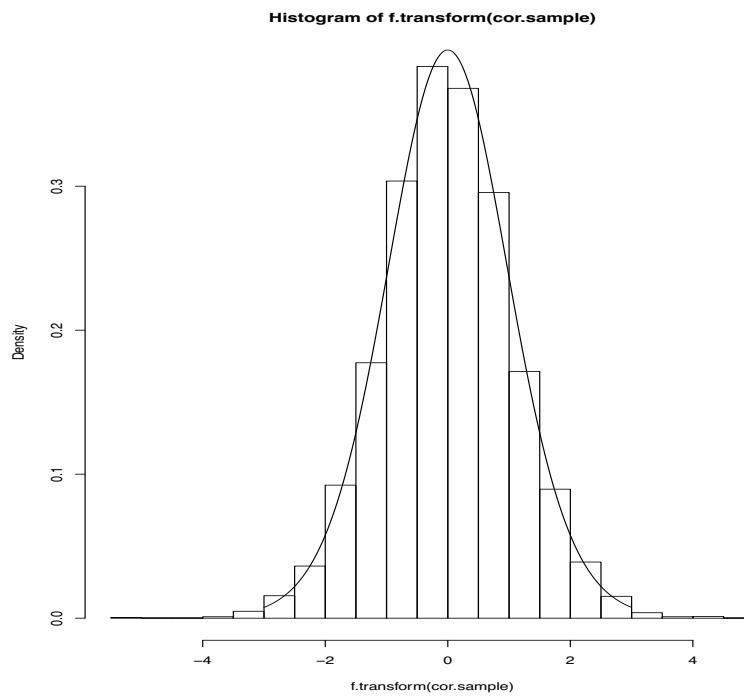


Figure 26.2: Plot for Problem 26.3.

Chapter 27

Practice Problems - Markov Chains, MCMC and Computational Bayesian Methods

27.1 Exercises

Problem 27.1. Consider the network shown in Figure 27.1. There are 5 nodes, N_i , $i = 1, \dots, 5$. Two nodes are considered connected if there is an edge between them. A traveller starts at node N_1 and makes transitions between nodes. The path is denoted $x_1, x_2, \dots, x_t, \dots$ with time represented by index $t = 1, 2, \dots$. The traveller is attempting to reach node N_5 , using the following rules:

Rule 1: The initial node $x_1 = N_1$ is fixed. Then x_2 is selected from all nodes connected to N_1 with equal probability.

Rule 2: Following the initial transition, at any $t > 1$, when the traveller is at any node x_t other than N_5 the subsequent node is selected from all nodes connected to x_t *except* for the previously visited node x_{t-1} , with equal probability assigned to each available node. For example, if the traveller transitions from nodes N_3 to N_2 , then the next node visited is selected from N_1 and N_5 , with probability 1/2 assigned to each.

Rule 3: The traveller remains in node N_5 once it is visited.

We can describe the independence structure this way: when necessary, the traveller can choose the subsequent node by flipping a 2- or a 3- sided coin. The outcome of the toss is independent of all previous coin tosses and on the current choice of coin.

- (a) Is x_1, x_2, \dots a Markov process? Why or why not?
- (b) Assuming you answered ‘no’ to part (a), show how the process can be *Markovianized*. That is, construct a Markov process z_1, z_2, \dots by defining as a state the transition between pairs of successively visited nodes, rather than the nodes themselves. For example, a transition from node N_2 to node N_1 defines state $N_2 \rightarrow N_1$. It may clarify things to introduce a dummy node

N_0 . The initial state of the Markovianized process would then be the transition $N_0 \rightarrow N_1$, which represents the entry of the traveller into the system. The traveller never visits N_0 again, so this transition occurs only once. Enumerate explicitly the state space, and derive the probability transition matrix P . Why is the transition $N_5 \rightarrow N_5$ an *absorbing state* in the Markovianized process?

- (c) Let T be the time at which the traveller visits node N_5 for the first time. For example, given a path $x_1 = N_1, x_2 = N_3, x_3 = N_2, x_4 = N_5$, we have $T = 4$. Let $P^{(k)}$ be the k -step probability transition matrix of the Markovianized process z_1, z_2, \dots . Explain how the probability $P(T < k)$ can be obtained from $P^{(k)}$.
- (d) Using R, or another suitable computing environment, calculate the probability mass function $p_T(x) = P(T = x)$ for T , and compute its mean. This can be done by successively calculating $P^{(k)}$ for increasing k . Construct a suitable plot of p_T . Note that the support of T is unbounded, so you can confine calculation of $p_T(x)$ to all $x \leq M$, selecting large enough M for which $p_T(M)$ is close to zero.

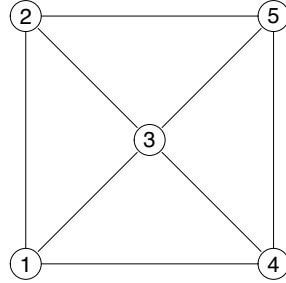
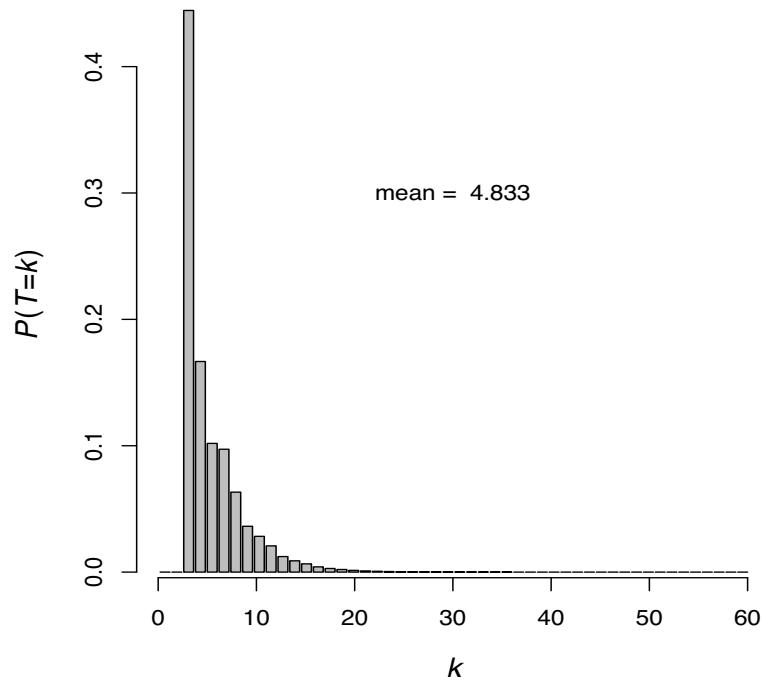


Figure 27.1: Network diagram for Problem 27.1.

SOLUTION:

- (a) No, x_1, x_2, \dots is not a Markov process. The probability $P(x_n = x' \mid x_{n-1}, x_{n-2}, \dots, x_1)$ depends on both x_{n-1} and x_{n-2} .
 - (b) The state space consists of transitions $N_j \rightarrow N_k$, or $j - k$ for short. For each edge in the network, there are two transitions, except for transitions starting from N_5 . In addition, we have the initial transition $0 - 1$ and the final transition $5 - 5$. That makes $2 \times 8 - 3 + 1 + 1 = 15$ transitions, enumerated in the table below. The transition probabilities can be deduced from the rules, yielding the transition matrix given in the table below. The transition $5 - 5$ is an absorbing state because $P(5 - 5 \mid 5 - 5) = 1$.
 - (c) We have
- $$P_{0-1,5-5}^{(k)} = P(z_k = 5 - 5 \mid z_1 = 0 - 1) = P(T < k).$$
- (d) We calculate $P^{(k)} = P^k$, where P is the (one-step) transition matrix given above. The PMF is then $p_k = P(T < k + 1) - P(T < k)$. The following code completes the problem. See Figure 27.2.

	0-1	1-2	1-3	1-4	2-1	2-3	2-5	3-1	3-2	3-4	3-5	4-1	4-3	4-5	5-5
0-1	0.00	0.33	0.33	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1-2	0.00	0.00	0.00	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33	0.33	0.00	0.00	0.00	0.00
1-4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	0.00
2-1	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33	0.33	0.00	0.00	0.00	0.00
2-5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
3-1	0.00	0.50	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3-2	0.00	0.00	0.00	0.00	0.50	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3-4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.50	0.00
3-5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
4-1	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.33	0.00	0.00	0.00	0.00
4-5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
5-5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 27.2: Distribution of T for Problem 27.1.

```
#### Construct Transition Matrix
```

```
x = c(
  0,1/3,1/3,1/3,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,1/2,1/2,1/2,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,1/3,0,1/3,1/3,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,1/2,1/2,0,
  0,0,1/2,1/2,0,0,0,0,0,0,0,0,0,0,0,0,
```

```

0,0,0,0,0,0,0,1/3,0,1/3,1/3,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,1/2,0,1/2,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,1/2,0,1/2,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,1/2,0,1/2,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,1/2,1/2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1/3,1/3,0,1/3,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)

pp = matrix(x, 15,15,byrow=T)

##### Iterate matrix multiplication

st = pp[1,15]
ppp = pp
for (i in 1:101) {
  ppp = ppp%*%pp
  st = c(st, ppp[1,15])
}
sum(1 - st) + 1

##### calculate mean and PMF

pa = diff(c(0,st))
mu = sum(pa*(1:length(pa)))

##### output results

ex1 = expression(paste(italic(k)))
ex2 = expression(paste(italic(P), '(', italic(T), '= ', italic(k), ')', sep=''))
par(mfrow=c(1,1),cex=1,oma=c(2,2,2,2))
barplot(pa[1:50])
axis(1)
mtext(ex1,1,line=3,cex=1.25)
mtext(ex2,2,line=3,cex=1.25)
text(30,.3,paste('mean = ',round(mu,3)))

```

Problem 27.2. One important application in remote sensor systems is the localization of a system node based on *received signal strength indication* (RSSI) measurements. Suppose m stationary

nodes equipped with radio signal receivers are deployed in an environment, which is mathematically a region within \mathbb{R}^2 . Suppose an additional mobile node to be localized is equipped with a radio signal transmitter. The strength of the signal (RSSI) received at each stationary node is inversely proportional to the distance between that node and the mobile node (that is, the transmission distance).

The *Weibull density* is often used to model survival times, and other positive random variables. Parameterizations vary, but it commonly incorporates a *shape parameter* $\kappa > 0$ and *scale parameter* $\mu > 0$ as follows:

$$f(z; \kappa, \mu) = \frac{\kappa}{\mu} \left(\frac{z}{\mu} \right)^{\kappa-1} e^{-(z/\mu)^\kappa}, \quad z \geq 0.$$

Then suppose Y is an RSSI measurement which is calibrated so that $Z = 1/Y$ has a Weibull distribution where $\kappa = 8.25$ and μ is the transmission distance. There are $m = 3$ stationary nodes, labeled $i = 1, 2, 3$, located on the two-dimensional deployment region at coordinates (in miles):

$$\begin{aligned} (x_1, y_1) &= (-0.50, 0.00) \\ (x_2, y_2) &= (0.42, 2.00) \\ (x_3, y_3) &= (1.50, 1.27). \end{aligned}$$

Suppose $\theta = (\theta_x, \theta_y)$ is the current location of the mobile node. After transmitting a radio signal, three RSSI measurements $(Y_1, Y_2, Y_3) = (0.926, 0.943, 0.787)$ are collected from the respective stationary nodes. Assume the RSSI measurements are independent.

- (a) The object is to construct a Bayesian model for the inference of $\theta = (\theta_x, \theta_y)$. For the prior density $\pi(\theta)$, use a uniform distribution over some large enough region containing all nodes. Then accept as the data the reciprocals $Z = (Z_1, Z_2, Z_3) = (1/Y_1, 1/Y_2, 1/Y_3)$. Write explicitly the conditional density of Z given θ , say $f(z_1, z_2, z_3 | \theta)$, and then give the posterior density of θ given $Z = z$, say $\pi(\theta | z_1, z_2, z_3)$. Note that the posterior density need not be normalized.
- (b) Evaluate the posterior density on a two-dimensional grid, with x and y coordinates defined as follows

```
xgrid = seq(-0.7, 1.0, 0.01)
ygrid = seq(-0.1, 1.8, 0.01)
```

That is, $\pi(\theta | z_1, z_2, z_3)$ is evaluated in a rectangle $\theta_x \in [-0.7, 1.0]$ and $\theta_y \in [-0.1, 1.8]$, with spacing $\delta = 0.01$ between grid points in each axis direction. Use the data given here for the values (z_1, z_2, z_3) . Use the `persp` function to create a 3-dimensional image of this density. Use options `theta=0`, `phi=30`. **HINT:** In R, densities and distribution functions can usually be evaluated on a logarithmic scale, using the `log = TRUE` or `log.p = TRUE` option, as appropriate. This is far preferable for an application like this. A good strategy would be to write a function which evaluates and returns the density on a logarithmic scale, which can later be exponentiated if needed. Again, note that the posterior density need not be normalized.

- (c) In this example, the maximum likelihood estimate $\hat{\theta}_{MLE}$ will be the value of θ which maximizes $\pi(\theta | z_1, z_2, z_3)$ (assuming there is only one global maximum). Why is this the case? Use the grid evaluation of Part (b) to obtain an approximation of $\hat{\theta}_{MLE}$. **HINT:** This can be done with the `which` function, using the `arr.ind = TRUE` option.

- (d) Create a Hastings-Metropolis algorithm to simulate a sample from $\pi(\theta | z_1, z_2, z_3)$. Implement the following features:
- Use as a proposal rule something like `theta.new = theta.old + runif(2,-1/10,1/10)`. This means the resulting state space is not discrete, but the algorithm will work in much the same way. Under this proposal rule we can take $1 = Q(\theta_2 | \theta_1)/Q(\theta_1 | \theta_2)$ when calculating the acceptance probability.
 - Allow $N = 100,000$ transitions. Capture in a single object all sampled values of θ . You can use $\theta = (0, 0)$ as the initial state. **HINT:** When constructing an MCMC algorithm, rather than calculate a ratio of densities, it is better to calculate a difference Δ in log-densities, and then calculate the exponential function of the difference, that is, e^Δ . For this reason, the function constructed in Part (b) should return log-densities.
- (e) The posterior density is defined on a two-dimensional plane. There are a number of R functions that can be used to visualize functions or densities defined on \mathbb{R}^2 .
- Use the `smoothScatter` function to plot the sampled θ values (this function is part of the `graphics` library). Rather than draw a simple scatter plot, this function draws a heat map representation of the sample density. Essentially, a deeper color shade indicates a higher density of sampled points. Use options `xlim=c(-1,1.5), ylim=c(-0.5,2.1)`. Include horizontal and vertical axes lines which pass through the origin $(0,0)$ (you can use the `abline` function for this). Also indicate the locations of the three stationary nodes (you can use the `points` function for this). Then indicate the location of $\hat{\theta}_{MLE}$, using a distinct symbol for clarity.
 - Superimpose on your plot a contour plot of the posterior density evaluated in Part (b). Use the `contour` function with the `add = TRUE` option.
- (f) One advantage of the use of MCMC sampling to estimate posterior densities is that various forms of inference become easy to implement. For example, suppose there is interest in the distance of the mobile node from the origin $(0, 0)$. Use the MCMC to estimate the posterior density of this distance. This can be done by first transforming the sampled values of θ , then constructing a histogram of these transformed values.

SOLUTION:

The following code can be used to complete the problem.

```
> ### (b)
>
> ### Reciever locations
>
> rec.loc = matrix(c(-0.5,0,0.42,2.0,1.5,1.27),3,2,byrow=T)
>
> ### RSSI data (use the reciprocal)
>
> xrfid = c(0.926, 0.943, 0.787)^(-1)
>
> ### Posterior density (assuming uniform prior)
```

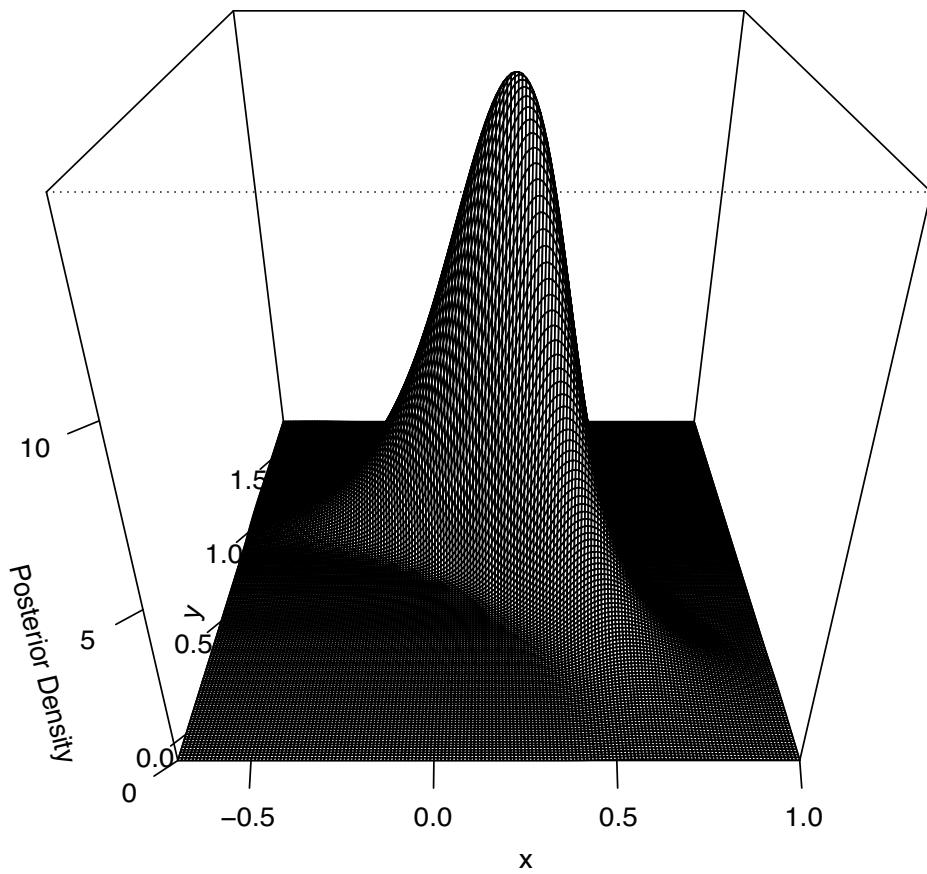


Figure 27.3: Plot for Problem 27.2 (b).

```

> ### Note log scale
>
> f.posterior = function(xy) {
+   rr = apply(rec.loc,1,function(rlxy) {sqrt(sum((xy-rlxy)^2))})
+   ans = sum(dweibull(xrfid,shape=8.25,scale=rr,log=TRUE))
+   return(ans)
+ }
>
> ### Evaluate posterior density on grid
>
> xgrid = seq(-0.7,1.0,0.01)
> ygrid = seq(-0.1,1.8,0.01)
> nx = length(xgrid)
> ny = length(ygrid)
> post.density = matrix(NA, nx, ny)

```

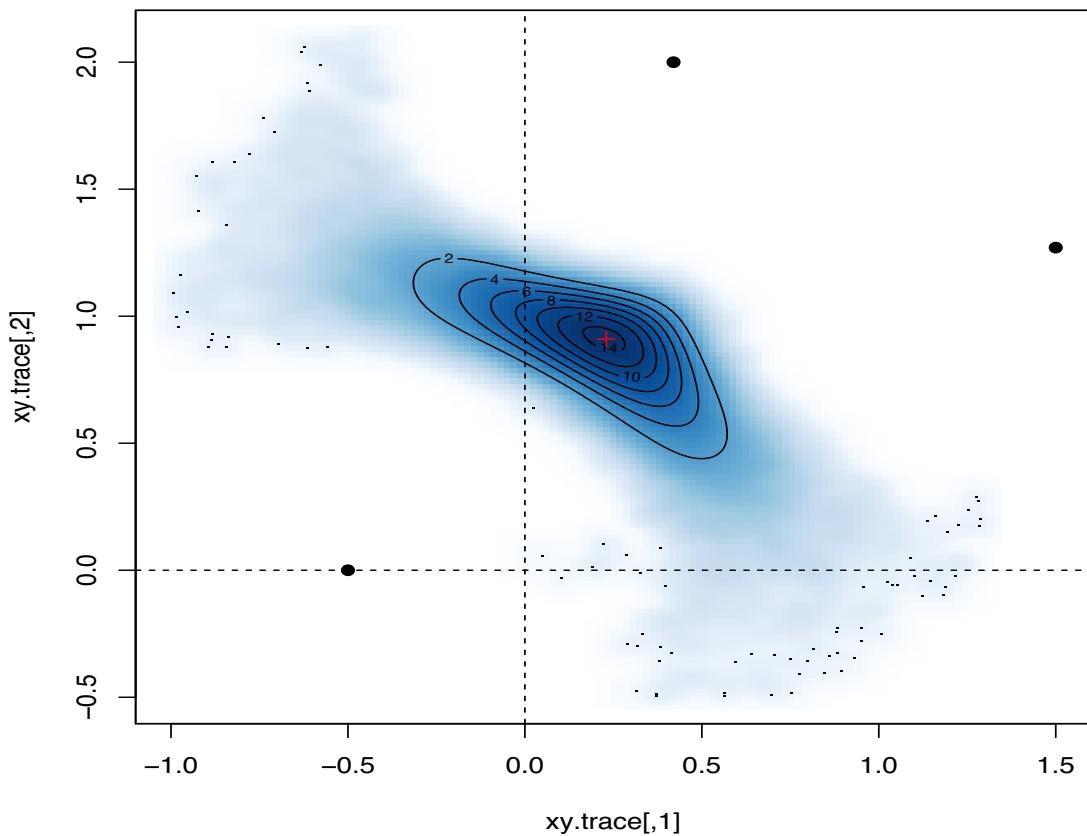


Figure 27.4: Plot for Problem 27.2 (e).

```

> for (i in 1:nx) {
+   for (j in 1:ny) {
+     post.density[i,j] = exp(f$posterior(c(xgrid[i],ygrid[j])))
+   }
+ }
>
> ### Draw perspective plot of posterior denisty
>
> pdf('figa2q4a-2019.pdf')
> par(mfrow=c(1,1))
> persp(xgrid,ygrid,post.density,theta=0,phi=30,xlab='x',ylab='y',
+         zlab='Posterior Density',ticktype='detailed')
> dev.off()
RStudioGD
2

```

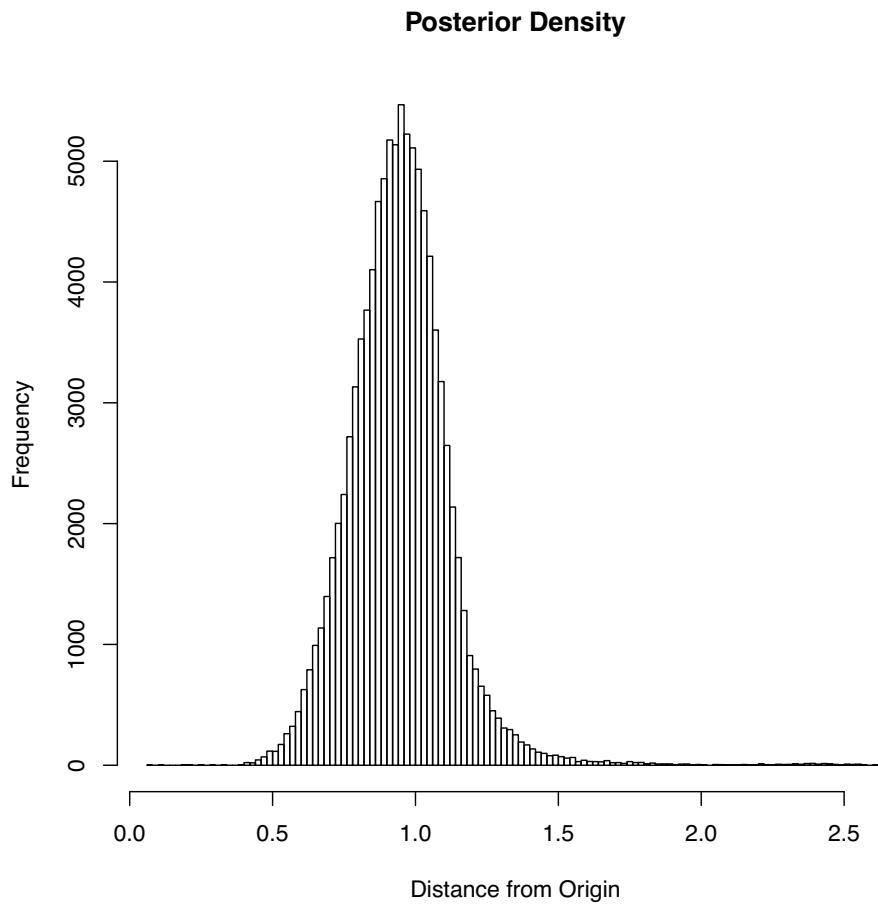


Figure 27.5: Plot for Problem 27.2 (f).

```

>
> ### (c)
>
> ### Get MLE of mobile node location
>
> max.ind = which(post.density==max(post.density),arr.ind=T)
> max.ind
  row col
[1,] 94 102
> xy.max = c(xgrid[max.ind[1]],ygrid[max.ind[2]])
> xy.max
[1] 0.23 0.91
>
> ### (d)
>

```

```
> ### Setup MCMC
>
> ntrace = 100000
> xy.trace = matrix(NA,ntrace,2)
> xy.old = c(0,0)
> set.seed(234)
>
> ### Loop through transitions
>
> for (i in 1:ntrace) {
+
+   # simulate transition from Q
+
+   xy.new = xy.old + runif(2,-1/10,1/10)
+
+   # determine acceptance probability (the q.ratio is always 1)
+   # recall that f.posterior() returns density on a log scale
+
+   alpha = exp(f.posterior(xy.new)-f.posterior(xy.old))
+   if (runif(1) <= alpha) {xy.old = xy.new}
+
+   # capture samples
+
+   xy.trace[i,]=xy.old
+
+ }
>
> ### (e)
>
>
> pdf('figa2q4b-2019.pdf')
> par(mfrow=c(1,1))
>
> # One way to visualize empirical densities in two dimensions:
>
> smoothScatter(xy.trace,xlim=c(-1,1.5),ylim=c(-0.5,2.1))
>
> # Add location of stationary nodes
>
> points(rec.loc,pch=19)
>
> # Add axes for clarity
>
> abline(h=0,lty=2)
> abline(v=0,lty=2)
```

```

>
> # add countour plot of posterior evaluated on grid
>
> contour(xgrid,ygrid,post.density,add=T)
>
> # add MLE of mobile node location
>
> points(xy.max[1],xy.max[2], pch=3,col='red')
> dev.off()
RStudioGD
      2
>
> pdf('figa2q4c-2019.pdf')
> par(mfrow=c(1,1))
> rr.post = apply(xy.trace,1,function(xy) sqrt(sum((xy-c(0,0))^2)))
> hist(rr.post,nclass=100,main='Posterior Density',xlab='Distance from Origin')
> dev.off()

```

- (a) First note that Z_i has a Weibull density with shape parameter $\kappa = 8.25$ and scale parameter

$$\mu_i = \|\theta - (x_i, y_i)\| = \sqrt{(\theta_x - x_i)^2 + (\theta_y - y_i)^2},$$

which is equal to the Euclidean distance between θ and (x_i, y_i) (ie. the radio signal transmission distance). Then, since the three RSSI measurements are assumed to be independent, the posterior density is

$$\pi(\theta | z_1, z_2, z_3) = K \prod_{i=1}^3 f(z_i; \kappa, \mu_i) \times \pi(\theta)$$

where K is a normalization constant. However, since the prior is uniform, $\pi(\theta)$ does not depend on θ , and can be incorporated into the normalization constant. We then have

$$\pi(\theta | z_1, z_2, z_3) = K' \prod_{i=1}^3 f(z_i; \kappa, \mu_i),$$

where K' does not depend on θ .

- (b) We can set $K' = 1$. Then see Figure 27.3 produced by the code above.
- (c) Since the prior density is uniform (and therefore constant) it follows that $\pi(\theta | z_1, z_2, z_3)$ is proportional to the likelihood function, that is, the likelihood is given by

$$l(\theta; z_1, z_2, z_3) = \prod_{i=1}^3 f(z_i; \kappa, \mu_i) \propto \pi(\theta | z_1, z_2, z_3).$$

Then using the code above, we obtain, approximately, $\hat{\theta}_{MLE} \approx (0.23, 0.91)$.

- (d) See code above.

- (e) See Figure 27.4 produced by the code above.
- (f) For each value of $\theta = (\theta_x, \theta_y)$ in the MCMC sample calculate $d = \sqrt{\theta_x^2 + \theta_y^2}$, which is the distance from the origin. Then the posterior density for d is estimated by, for example, a histogram constructed from the sampled values of d . See Figure 27.5 produced by the code above.

Problem 27.3. Create a Hastings-Metropolis algorithm to simulate a sample from a $bin(25, 0.75)$ distribution using the following steps.

- The state space is $\mathcal{S} = \{0, \dots, 25\}$.
- Define a proposal Markov chain in the following way. If the algorithm is in state $x \in \{0, \dots, 25\}$, the proposed state is x' with probability $Q(x' | x)$. Suppose that if $0 < x < 25$, then the proposal state is $x' = x + 1$ and $x' = x - 1$ with equal probability. If $x = 0$ then $x' = 1$ with probability 1. If $x = 25$ then $x' = 24$ with probability 1. Give $Q(x' | x)$ precisely.
- Program the Hastings-Metropolis algorithm, and allow it to run for 100,000 transitions. You can start from any initial state.
- Plot a histogram of the MCMC sample. Use one frequency bar for each state. Superimpose on this the true binomial density.

SOLUTION:

The proposal Markov chain is given by

$$Q(x' | x) = \begin{cases} 1/2 & ; |x' - x| = 1, 0 < x < 25 \\ 1 & ; x' = 1, x = 0 \\ 1 & ; x' = 24, x = 25 \\ 0 & ; \text{otherwise} \end{cases}$$

The algorithm can be run using the following code. The plot is given in Figure 27.6, which shows a close agreement between the true density and the MCMC estimate.

```
nstate=25
f0 = function(x) {dbinom(x,size=nstate,prob=0.75)}

# do ntrace transitions

ntrace = 100000
x = numeric(ntrace)
x0 = 1
set.seed(12345)
for (i in 1:ntrace) {

  # simulate transition from Q
```

```

if (x0 == 0) {
  x1 = 1
} else {
  if (x0 == nstate) {
    x1 = nstate-1
  } else {
    x1 = x0 + sample(c(-1,1),1)
  }
}

# determine Q[j,i]/Q[i,j]

if (x0 == 0 | x0 == nstate) {
  q.ratio = 1/2
} else {
  if ((x0==1 & x1==0) | (x0==nstate-1 & x1==nstate)) {
    q.ratio = 2
  } else {q.ratio=1}
}

# determine acceptance probability

a1 = f0(x1)/f0(x0)
alpha = min( (f0(x1)/f0(x0)) *q.ratio,1)
if (runif(1) <= alpha) {x[i] = x1} else {x[i] = x0}
x0 = x[i]

}

ex1 = expression(italic(X))
hist(x,probability=T,breaks=(c(0:(nstate+1))-0.5),col='gray',main='',cex.lab=1.25)
lines(0:nstate,dbinom(0:nstate,nstate,0.75),type='b',pch=19,cex=1.0)
legend('topleft',legend=c('True density','MCMC sample'),lty=c(1,NA),
      pch=c(19,15),col=c('black','darkgray'))

```

Problem 27.4. Consider the 4th order polynomial

$$f(x) = x \cdot (x + 1/2) \cdot (x - 2) \cdot (x - 4).$$

- (a) Plot $f(x)$ on a range [-2,5]. Identify all local minima, and the unique global minimum. This can be done either analytically, or numerically with R code such as

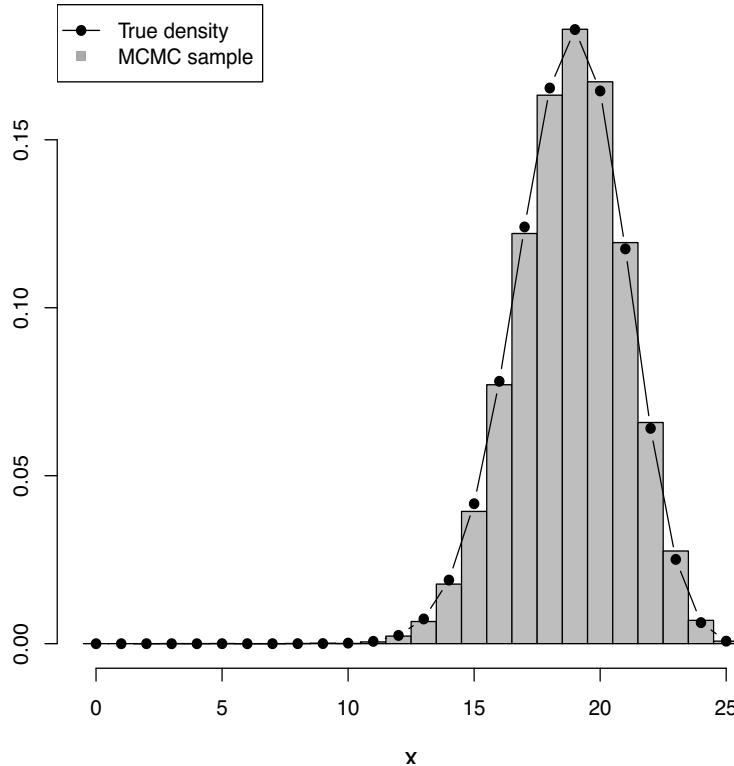


Figure 27.6: Plot for Problem 27.3.

```
f0 = function(x) {x*(x+0.5)*(x-2)*(x-4)}
optimize(f0,interval=c(-10,0))
optimize(f0,interval=c(0,10))
```

- (b) Create a simulated annealing algorithm to find the value of $x_{min} = x$ which gives the global minimum $f(x_{min})$ of $f(x)$. Suppose the algorithm is allowed to run for N transitions. Use the following components.
- We can specify the initial and final temperatures of the cooling schedule to be $t_0 > t_N$. This can be attained by setting
- $$t_n = t_0 \left(\frac{t_N}{t_0} \right)^{n/N}.$$
- Given current state x , the proposed state is given by $x' = x + U$, where U is uniformly distributed on the interval $(-1/4, 1/4)$.
- In your algorithm use parameters $N = 50,000$, $t_0 = 500$, $t_N = 0.01$. Remember that in simulated annealing, the proposal distribution is not used in the calculation of the acceptance probability.
- (c) When you run the algorithm, capture the sequence x_1, \dots, x_N . Plot x_n and $f(x_n)$ as a function of iteration index $n = 1, \dots, 50000$. Superimpose on your plots the local and global minima, either x^* or $f(x^*)$ as appropriate. Comment on the convergence properties.

SOLUTION:

The required code is given below. In particular, using the `optimize` function the minima are $x^* = -0.271, 3.266$, with objective values $f(x^*) = -0.602, -11.429$.

```
> f0 = function(x) {x*(x+0.5)*(x-2)*(x-4)}
> optimize(f0,interval=c(-10,0))
$minimum
[1] -0.2709331

$objective
[1] -0.6019377

> optimize(f0,interval=c(0,10))
$minimum
[1] 3.265703

$objective
[1] -11.42948
```

The algorithm and plots are given by the code given below. See plots in Figure 27.7. From the top right plot it can be seen that initially, the algorithm fluctuates between the two local minima, but finally converges to the global minimum. The bottom row gives plots of $f(x_n)$ against n , the first plot showing the entire range, the second showing only the final 15000 iterations. Both show that the value of $f(x_n)$ is subject to considerable variation near any n , but that this variation eventually approaches zero as the minimum is approached. In effect, convergence of simulated annealing can be understood as convergence of a *distribution*.

```
### Proposal function

proposal = function(x) {x + runif(1,-0.25,0.25)}

### Set up algorithm

x.trace = NULL
scr.trace = NULL
x.old = 0
scr.old = f0(x.old)
t0 = 500
tn = 0.01
set.seed(12345)
ntrace = 50000
for (i in 1:ntrace) {

  # get proposal and route distance

  x.new = proposal(x.old)
```

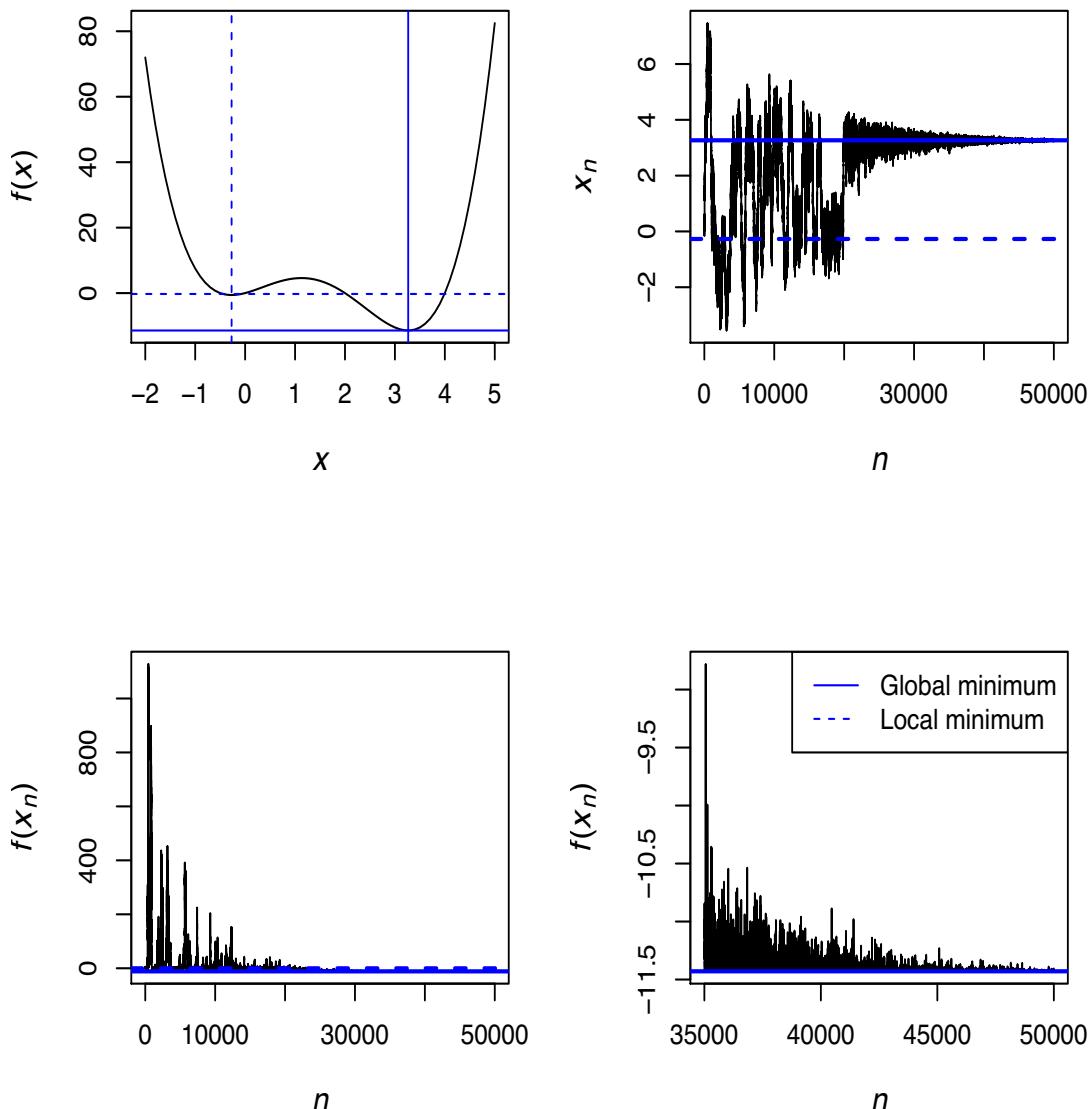


Figure 27.7: Plots for Problem 27.4.

```

scr.new = f0(x.new)

# update temperature

temp = t0*(tn/t0)^(i/ntrace)

# calculate acceptance probability

alpha = exp((scr.old-scr.new)/temp)

```

```

# accept or reject proposal

if (runif(1) <= alpha)
{
  # accept proposal

  x.old = x.new
  scr.old = scr.new
}

x.trace[i] = x.old
scr.trace[i] = scr.old
}

### Set up graphics detail

ex1 = expression(paste(italic(f), '(', italic(x), ')', sep=''))
ex2 = expression(italic(x))
ex3 = expression(italic(x)[italic(n)])
ex4 = expression(italic(n))
ex5 = expression(paste(italic(f), '(', italic(x)[italic(n)], ')', sep=''))
par(mfrow=c(2,2),cex.lab=1.25,oma=c(2,2,2,2))

### Draw plots

xgrid = seq(-2,5,0.1)
plot(xgrid,f0(xgrid),type='l',xlab=ex2,ylab=ex1)
abline(v=-0.271,lty=2,lwd=1,col='blue')
abline(v=3.266,lty=1,lwd=1,col='blue')
abline(h=-0.271,lty=2,lwd=1,col='blue')
abline(h=-11.429,lty=1,lwd=1,col='blue')

plot(x.trace,type='l',xlab=ex4,ylab=ex3)
abline(h=-0.271,lty=2,lwd=2,col='blue')
abline(h=3.266,lty=1,lwd=2,col='blue')

del = 15000
plot(scr.trace,type='l',xlab=ex4,ylab=ex5)
abline(h=-0.271,lty=2,lwd=2,col='blue')
abline(h=-11.429,lty=1,lwd=2,col='blue')
plot((ntrace-del):ntrace, scr.trace[(ntrace-del):ntrace],type='l',
      xlab=ex4,ylab=ex5)
abline(h=-0.271,lty=2,lwd=2,col='blue')
abline(h=-11.429,lty=1,lwd=2,col='blue')
legend('topright',legend=c('Global minimum','Local minimum'),

```

```
lty=c(1,2),col='blue',lwd=1)
```

Problem 27.5. Suppose in an apple orchard a thin cross-section of land of length 1000 feet is selected. We define a starting point at $x = 0$, so that the cross-section ends at $x = 1000$ feet. It is assumed that the number of apples on a tree at position x has a Poisson distribution with mean $\lambda = a + bx$, for two constants a, b . We count the number of apples on 4 trees at positions $x = (25, 135, 643, 719)$, observing counts $N = (590, 627, 736, 737)$ respectively. We assume the counts are statistically independent.

We will use a Bayesian analysis to estimate a, b . We assume these parameters are in the rectangle

$$(a, b) \in [500, 700] \times [0, 0.4] = \Theta.$$

The prior distribution $\pi(a, b)$ of (a, b) is taken to be uniform on Θ . The conditional distribution $P(N | a, b)$ of the counts given (a, b) is obtainable directly from the Poisson distribution.

To estimate the posterior distribution $\pi(a, b | N)$ given the observed counts, we discretize Θ by selecting integers N_a, N_b , and defining a discrete distribution on points

$$(a_i, b_j) = (500 + 200 \times i/N_a, 0 + 0.4 \times j/N_b) \in \Theta_{grid}, \quad i = 0, 1, \dots, N_a, \quad j = 0, 1, \dots, N_b.$$

- (a) Identify precisely the components $P(N | a, b)$ and $\pi(a, b)$ of the posterior density

$$\pi(a, b | N) = \frac{P(N | a, b)\pi(a, b)}{\int_{a,b \in \Theta} P(N | a, b)\pi(a, b) da db}.$$

- (b) First approximate the posterior distribution by evaluating it on all grid points in Θ_{grid} . Use $N_a = N_b = 40$. Store the values in a 41×41 matrix, where the value for (a_i, b_j) occupies matrix position $(i+1, j+1)$. The posterior density can be normalized by summing $P(N | a, b)\pi(a, b)$ over all grid points in Θ_{grid} . Use R functions `contour` and `persp` to plot the distribution.
- (c) Next, approximate the posterior distribution by sampling, using a Metropolis-Hastings algorithm, from the discrete density proportional to $P(N | a, b)\pi(a, b)$ with support on Θ_{grid} . This will require defining a proposal transition matrix on the two dimensional grid Θ_{grid} . This can be done in the following way. The first step of the proposal is to choose with equal probability to move either in the a or the b dimension (but not both). Then select a direction in that dimension as described in the lecture notes. Start the sampler in the middle of the grid. Record the sample both by capturing the sampled values of (a_n, b_n) , and by recording the occupancy frequency of the Markov chain in a suitably defined 41×41 matrix. Allow the sampler to run for $M = 500,000$ iterations.
- (d) The two approximations can be compared directly using the `contour` function. By setting the option `add = T` a contour plot is superimposed onto the current plot. In this way, draw contours for both approximations on the same plot (use different colors). It will be helpful to specify the same contour levels for each distribution. This can be done using, for example, option `levels = seq(.002,.01,by=.002)`. Is the sampler able to approximate the distribution?

- (e) One of the advantages of simulating a sample from a posterior distribution is that it becomes very easy to answer specific inference questions. For example, if the trees are evenly spaced along the cross-section, the average number of apples per tree is

$$\mu = a + b \times 500.$$

We can estimate the marginal posterior density $\pi(\mu | N)$ of μ by examining the sampled values $\mu_i = a_i + b_i \times 500$, $i = 1, \dots, M$. Using this method:

- (i) Estimate graphically the marginal posterior density of μ by plotting a histogram of the samples μ_i .
- (ii) Report the estimated mean, and the 5th and 95th percentiles of $\pi(\mu | N)$.

SOLUTION:

We have data $x = (25, 135, 643, 719) = (x_1, \dots, x_4)$, $N = (590, 627, 736, 737) = (n_1, \dots, n_4)$. If $N_i \sim \text{pois}(a + x_i b)$ then

$$P(N_i = n | a, b) = \frac{(a + x_i b)^n}{n!} e^{-(a + x_i b)}.$$

Then, Θ is a rectangle of area $(700 - 500) \times (0.4 - 0) = 80$. Therefore, the prior density of (a, b) is

$$\pi(a, b) = \frac{1}{80} I\{(a, b) \in \Theta\}.$$

Assuming conditional independence, the posterior density of (a, b) is proportional to

$$\begin{aligned} \pi(a, b | N) &\propto \left(\prod_{i=1}^4 P(N_i = n_i | a, b) \right) \times \pi(a, b) \\ &\propto \left(\prod_{i=1}^4 (a + x_i b)^{n_i} e^{-(a + x_i b)} \right) \times I\{(a, b) \in \Theta\}. \end{aligned}$$

Note that we omit any factors which do not depend on (a, b) .

The code needed to answer the remainder of the question is given below, with plots given in Figure 27.8.

- (i) The bivariate posterior density is represented in the top row of Figure 27.8.
- (ii) Regarding Part (d) (lower left plot of Figure 27.8), we would expect the estimate of a density to be less accurate at smaller values, where the sampling rate would, necessarily, be much smaller. This can be seen very clearly in the contours. The green contour represents the MCMC sample, and contains considerable variation at the 10^{-5} level contour. However, the MCMC estimate is quite accurate for the level 10^{-2} and 10^{-3} contours.
- (iii) Regarding Part (e) the estimated mean, and the 5th and 95th percentiles of $\pi(\mu | N)$ are, respectively, 698.4, 675 and 720. This can be obtained directly from the data represented in the lower right plot of Figure 27.8.

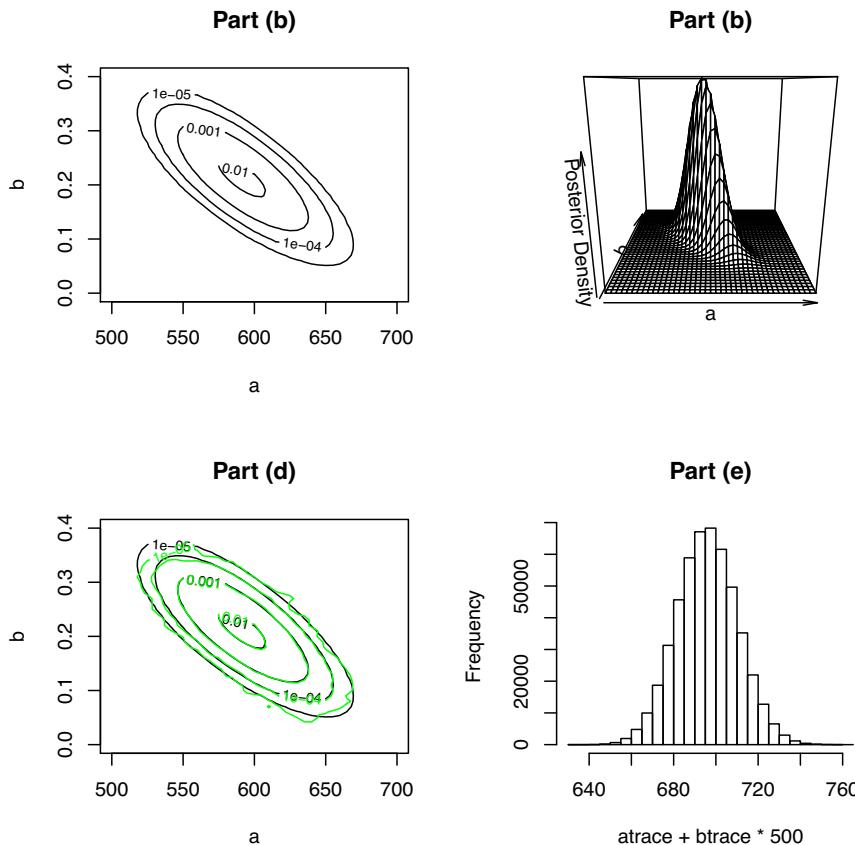


Figure 27.8: Plots for Problem 27.5.

```
prop.grid = function(x0,n) {
  if (x0 == 1) {
    x1 = 2
    qratio = 1/2
  } else {
    if (x0 == n) {
      x1 = n-1
      qratio = 1/2
    } else {
      x1 = x0 + sample(c(-1,1),1)
      if (x1 %in% c(1,n)) {qratio = 2} else {qratio = 1}
    }
  }
  return(c(x1,qratio))
}
```

```
}

### set up data

xpos = c(25, 135, 643, 719)
xobs = c(590, 627, 736, 737)

### calculate 'exact' posterior density

agrid = seq(500,700,by = 5)
bgrid = seq(0,0.4,by = .01)

na = length(agrid)
nb = length(bgrid)
zmat = matrix(0,na,nb)

for (i in 1:na) {
  for (j in 1:nb) {
    zmat[i,j] = prod(dpois(xobs,agrid[i]+bgrid[j]*xpos))
  }
}
zmat = zmat/sum(zmat)

##### MCMC samplers

fmat = matrix(0,na,nb)

x = floor(na/2)
y = floor(nb/2)
fmat[x,y]=1

ntrace = 500000

atrace = rep(0,ntrace)
btrace = rep(0,ntrace)

for (i in 1:ntrace) {

  if (runif(1) < 0.5) {
    propx = prop.grid(x,na)
    xnew = propx[1]
    ynew = y
    qratio = propx[2]
  } else {
    propy = prop.grid(y,nb)
```

```
ynew = propy[1]
xnew = x
qratio = propy[2]
}

pratio = prod(dpois(xobs,agrid[xnew]+bgrid[ynew]*xpos))
           /prod(dpois(xobs,agrid[x]+bgrid[y]*xpos))
alpha = min(pratio*qratio,1)

if (runif(1) <= alpha) {
  x = xnew
  y = ynew
}

atrace[i] = agrid[x]
btrace[i] = bgrid[y]

fmat[x,y] = fmat[x,y] + 1
}

### Create plots

par(mfrow=c(2,2),oma=c(2,2,2,2))

lv = c(10^seq(-5,0,1))
contour(agrid,bgrid,zmat,levels=lv,xlab='a',ylab='b')
title('Part (b)')
persp(agrid,bgrid,zmat,xlab='a',ylab='b',zlab='Posterior Density')
title('Part (b)')
lv = c(10^seq(-5,0,1))
contour(agrid,bgrid,zmat/sum(zmat),levels=lv,xlab='a',ylab='b')
contour(agrid,bgrid,fmat/sum(fmat),levels=lv,add=T,col='green')
title('Part (d)')

### marginal mean

hist(atrace+btrace*500,main=' ')
title('Part (e)')
mean(atrace+btrace*500)
quantile(atrace+btrace*500,c(0.05,0.95))
mean(xobs)
```

Problem 27.6. This problem will make use of the `cabbages` data set from the MASS library. This contains data from a cabbage field trial. We will be interested in `HeadWt` (weight of the cabbage head, presumably in kg) and `VitC` ascorbic acid content, in undefined units.

- (a) We are interested in the simple linear regression model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma_\epsilon^2)$, $Y_i = \text{VitC}$ and $X_i = \log(\text{HeadWt})$. Plot Y against X . Using the function `lm()`, calculate the least squares estimates of β_0 , β_1 , and superimpose the estimated regression line on your plot. Is a linear relationship between X and Y plausible?

- (b) We will construct a Bayesian model for the inference of (β_0, β_1) . The marginal prior densities will be

$$\begin{aligned}\beta_0 &\sim N(\mu_Y, \sigma_0^2) \\ \beta_1 &\sim N(0, \sigma_1^2),\end{aligned}$$

and under the prior assumption, β_0 and β_1 are independent. The conditional densities of Y_i given (β_0, β_1) are $Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma_\epsilon^2)$. Given (β_0, β_1) the responses Y_i can be assumed to be independent. Let $\phi(x; \mu, \sigma^2)$ denote the density function for a $N(\mu, \sigma^2)$ distribution. The joint posterior density of (β_0, β_1) will therefore be proportional to

$$\pi(\beta_0, \beta_1 | Y_1, \dots, Y_n) \propto \left[\prod_{i=1}^n \phi(Y_i; \beta_0 + \beta_1 X_i, \sigma_\epsilon^2) \right] \pi(\beta_0) \pi(\beta_1),$$

where $\pi(\beta_0) = \phi(\beta_0; \mu_Y, \sigma_0^2)$ and $\pi(\beta_1) = \phi(\beta_1; 0, \sigma_1^2)$. Create a Hastings-Metropolis algorithm to simulate a sample from $\pi(\beta_0, \beta_1 | Y_1, \dots, Y_n)$. Implement the following features.

- (i) Estimate μ_Y using the sample mean of the responses Y_i , and estimate σ_ϵ^2 using the *MSE* from the regression model in Part (a) (using data to estimate parameters in a Bayesian model can be referred to as the *empirical Bayesian method*).
- (ii) Use as a proposal rule something like `beta.new = beta.old + runif(2, -1, 1)`. This means the resulting state space is not discrete, but the algorithm will work in much the same way. Under this proposal rule we can take $1 = Q(y_2 | y_1)/Q(y_1 | y_2)$ when calculating the acceptance probability.
- (iii) Allow $N = 100,000$ transitions. Capture in a single object all sampled values of (β_0, β_1)
- (iv) Run the algorithm twice, first setting prior variance $\sigma_0^2 = \sigma_1^2 = \sigma_{prior}^2 = 100$, then $\sigma_{prior}^2 = 1000$. A parameter defining a prior density is referred to as a *hyperparameter*. Sometimes, this is used to represent prior information (for example μ_Y in this model). Otherwise, the hyperparameters are often set so as to make the prior density uniform, close to uniform, or otherwise highly variable, to reflect uncertainty regarding the parameter. This is known as a *diffuse prior*.
- (v) When constructing an MCMC algorithm, rather than calculate a ratio of densities, it is better to calculate a difference Δ in log-densities, and then calculate the exponential function of the difference, that is, e^Δ . This can be done using the `log = TRUE` option of the `dnorm()` function. This option is generally available for density functions in R.

- (c) Construct separate histograms for β_0 and β_1 for each hyperparameter choice $\sigma_{prior}^2 = 100, 1000$. Also, superimpose on each histogram the least squares estimate of β_0, β_1 from Part (a), as well as the confidence interval bounds $\hat{\beta}_i \pm t_{crit}SE_i$ (the `abline()` function can be used). In general, is the Bayesian inference for β_0 and β_1 consistent with the confidence intervals?
- (d) When a prior is intended to be diffuse, the usual practice is to investigate the sensitivity of the posterior density to the choice of prior. Ideally, in this case, the posterior density does not depend significantly on the prior. The simplest way to do this is to use a range of priors, then compare the resulting posterior densities. With this in mind, does the posterior density appear to be sensitive to the choice of σ_{prior}^2 ?

SOLUTION:

- (a) The following code may be used to calculate the fit and create the plot. A linear relationship is plausible. See Figure 27.9.

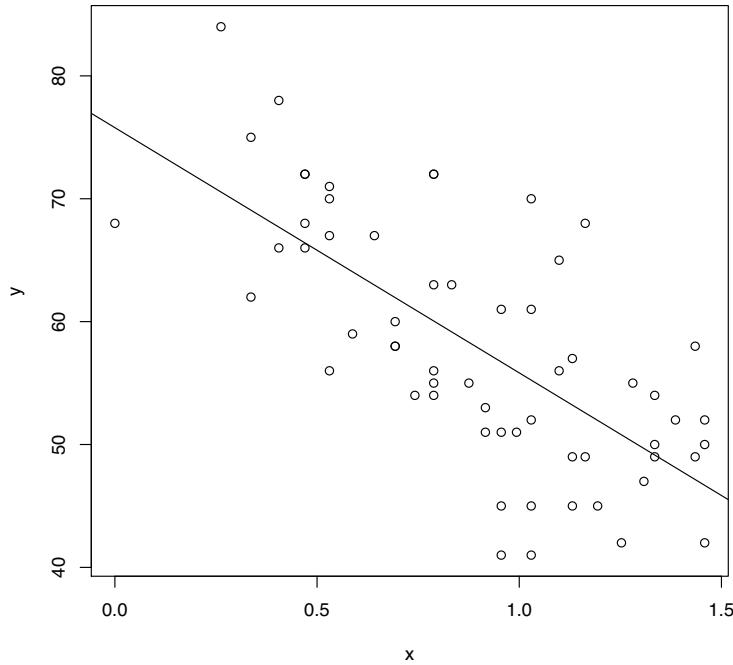


Figure 27.9: Plot for Problem 27.6 (a).

```
library(MASS)

### Part (a) Plot variables, calculate least squares fit
```

```

par(mfrow=c(1,1))
y = cabbages$VitC
x = log(cabbages$HeadWt)
plot(x,y)
fit = lm(y~x)
fit.coef = summary(fit)$coefficients
abline(fit.coef[,1])

(b) The following code may be used to run the MCMC algorithm.

### Part (b)

### get mean response

mean.y = mean(y)

### get MSE

mse = anova(fit)[2,3]

### get coefficient estimates, and confidence intervals

fit.coef = summary(fit)$coefficients
beta0.hat = fit.coef[1,1]
beta0.hat.se = fit.coef[1,2]
beta1.hat = fit.coef[2,1]
beta1.hat.se = fit.coef[2,2]
t.crit = qt(0.975,length(y)-2)

### construct conditional, prior, and posterior densities

f.cond = function(beta0,beta1) {sum(dnorm(y,mean=beta0+beta1*x,sd=sqrt(mse),log=T))}
f.prior = function(beta0,beta1) {sum(dnorm(c(beta0,beta1),mean=c(mean.y,0),
sd=sqrt(var.prior),log=T))}}
f.post = function(betav) {f.cond(betav[1],betav[2])+f.prior(betav[1],betav[2])}

### create graphics window for 4 plots in a 2x2 arrangement

par(mfrow=c(2,2))

### loop through 2 values of the prior variance = 1,100

for (var.prior in c(100,1000)) {

  ### set up MCMC algorithm
}

```

```

ntrace = 100000
beta.trace = matrix(NA,ntrace,2)
beta.old = c(mean.y,0)
set.seed(234)

### loop through iterations

for (i in 1:ntrace) {

  # simulate transition from Q

  beta.new = beta.old + runif(2,-1,1)

  # determine acceptance probability (the q.ratio is always 1)

  alpha = exp(f.post(beta.new)-f.post(beta.old))
  if (runif(1) <= alpha) {beta.old = beta.new}

  # capture samples

  beta.trace[i,]=beta.old

}

### Part (c) create plots

ex1 = expression(beta[0])
ex2 = expression(beta[1])

hist(beta.trace[,1],xlab=ex1,ylab='posterior density',
      main=bquote(var.prior == .(var.prior)),
      nclass=25)
abline(v = beta0.hat+t.crit*beta0.hat.se*c(-1,0,1),lty=c(2,1,2),
       col='green',lwd=2)
hist(beta.trace[,2],xlab=ex2,ylab='posterior density',
      main=bquote(var.prior == .(var.prior)),
      nclass=25)
abline(v = beta1.hat+t.crit*beta1.hat.se*c(-1,0,1),lty=c(2,1,2),
       col='green',lwd=2)

}

```

- (i) See code.
- (ii) See code.
- (iii) See code.
- (iv) See code.

(v) See code.

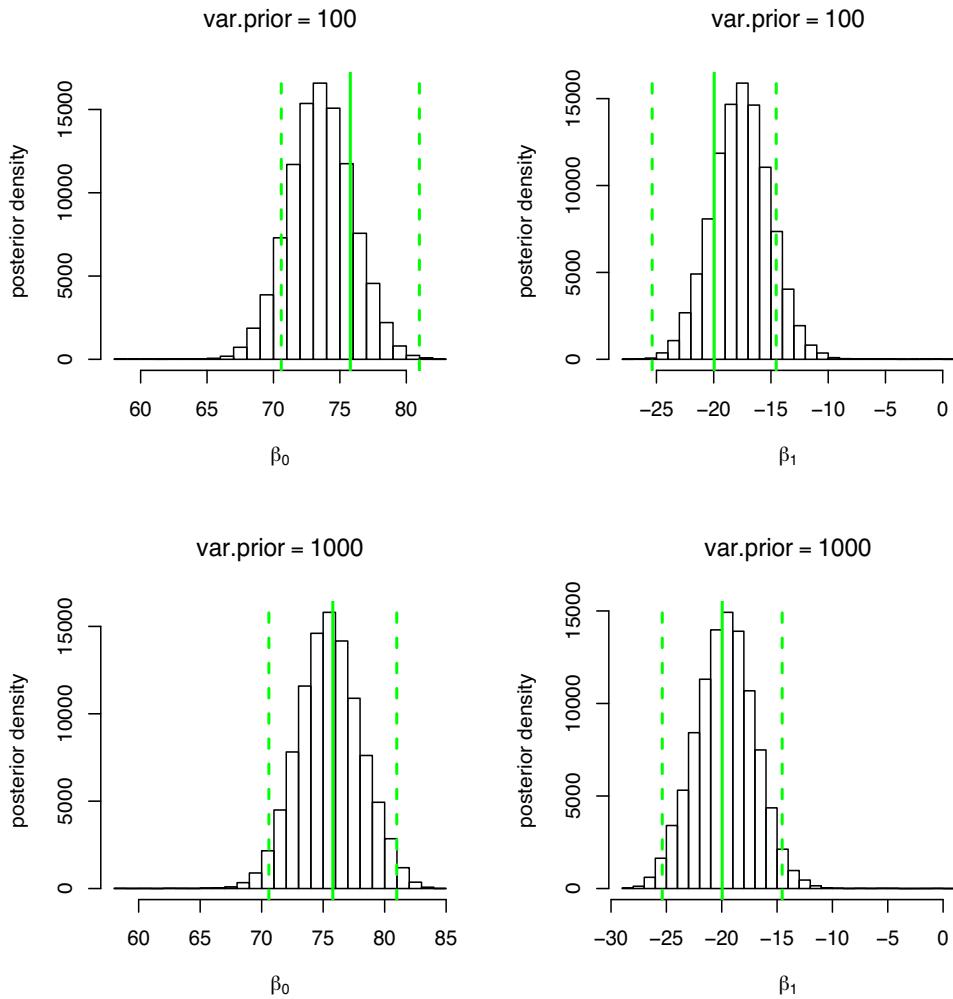


Figure 27.10: Plot for Problem 27.6 (c).

- (c) See Figure 27.10. The inference is consistent with the confidence intervals for $\sigma_{prior}^2 = 1000$, in that the posterior density is centered at the estimates $\hat{\beta}_0 = 75.79$ and $\hat{\beta}_1 = -19.96$. Also, most of the densities are contained within the respective confidence intervals. However, for $\sigma_{prior}^2 = 100$, the centers of the posterior densities are shifted significantly in the direction of the prior means of β_0, β_1 , which are $(\mu_Y, 0)$, where $\mu_Y = 57.95$.
- (d) The posterior density is quite sensitive to the prior density of β_0, β_1 . The prior variance σ_{prior}^2 determines the amount of weight (or certainty) to be assigned to the prior estimates of β_0, β_1 (here, these are $(\mu_Y, 0)$). When $\sigma_{prior}^2 = 100$ this weight is still relatively large, so that the posterior densities are shifted significantly towards these values. On the other hand, when $\sigma_{prior}^2 = 1000$, the influence of the prior density is small, so that the Bayesian inference will be quite consistent with likelihood methods (recall that for Gaussian models, the least squares estimates are equivalent to the maximum likelihood estimates).

27.2 Simulation Projects

Problem 27.7. Suppose a casino has a game in which a player bets x dollars, then with probability p wins back $2x$ dollars (for a net gain of x) and loses the original x dollars with probability $1 - p$ (for a net loss of x). Usually, $p < 1/2$. If $p = 1/2$ then the game is *fair*. Probability theory states that in such a fair game, there can be no strategy that results in a positive expected gain.

A commonly claimed counter-example to this is the following strategy. Enter the casino, then play the game, betting $x = 1$ each time, until you have a total gain of 1. For example, the following Win/Loss sequence will accomplish this: *LWLLWWW*, which has gain sequence $-1, 0, -1, -2, -1, 0, 1$, taking 7 games to reach a gain of 1. The Win/Loss sequence *W* also achieves a gain of 1 after a single game. Probability theory also states that the probability that a gain of 1 is reached after a finite number of games is 1 (although this *doesn't* hold if $p < 1/2$).

This seems to lead to a contradiction, since if we use this strategy, we can play once a day, and guarantee ourselves a regular income, noting that we can use any value of x we wish. Note that the case of the fair game, $p = 1/2$, is the important one, since if no winning strategy exists for this case, no winning strategy can exist when $p < 1/2$, which settles the matter.

- (a) Write an R program which simulates this process. Assume $p = 1/2$. For a single simulation, start at $gain = 0$, then increase or decrease $gain$ after each game by 1. This type of process is referred to as a *random walk*. The process stops when $gain = 1$. Store the number of games T needed to reach $gain = 1$. You may use the `rbinom()` function. Truncate the process at 1000 games. If $gain = 1$ has not been reached, indicate this by setting the number of games at, say, $T = 1001$.
- (b) Repeat the simulation to get 1000 replicates of T . Estimate the PMF $p_T(k) = P(T = k)$ directly from the data. Construct a *log-log* plot of $\log(p_T(k))$ against $\log(k)$. Note that the frequencies are not sorted in this case. How many times did T exceed 1000? How many times was T within 10 games, inclusive?
- (c) Using the `lines()` function, superimpose on this plot the lines

$$f(k) = \log(p_T(1)) - \alpha \log(k),$$

for $\alpha = 1.0, 1.25, 2.0$. Label your plot with the `legend()` function as in Question 4. We say that T has a *power law* distribution if

$$p_T(k) = \frac{c}{k^\alpha}$$

for positive constants α, c . If you can conclude that T conforms to a power law distribution, what can be said about $E[T]$? The rate at which a strategy earns money is

$$\text{gain rate} = \frac{\text{gain}}{\text{number of games played}}.$$

At what rate does this strategy earn money?

SOLUTION:

The following R script produces the plot in Figure 27.11.

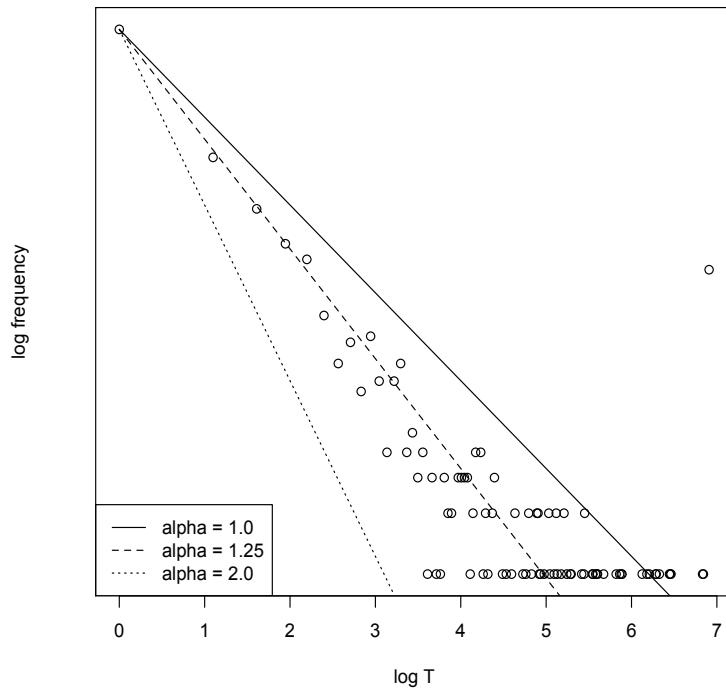


Figure 27.11: Plot for Problem 27.7

```
> nsim = 1000
> tt = rep(NA, nsim)
> bank = rep(NA, nsim)
>
> for (i in 1:nsim) {
+
+   x = rbinom(1000, size=1, prob=1/2)
+   z = cumsum(2*x-1)
+
+   if (sum(z==1) > 0) {
+     tt[i] = min(which(z==1))
+     bank[i] = min(z[1:tt[i]])
+   }
+   else
+   {
+     tt[i] = 1001
+     bank[i] = min(z)
+   }
}
```

```

+ }
>
> sum(tt <= 10)
[1] 752
> sum(tt == 1001)
[1] 32
>
> xx = as.integer(names(table(tt)))
> par(mfrow=c(1,1),cex=1)
> plot(log(xx), log(table(tt)/1000),xlab = 'log T', ylab = 'log frequency')
> lines(log(xx), max(log(table(tt)/1000)) - 1*log(xx),lty=1)
> lines(log(xx), max(log(table(tt)/1000)) - 1.25*log(xx),lty=2)
> lines(log(xx), max(log(table(tt)/1000)) - 2*log(xx),lty=3)
> legend('bottomleft',legend=paste('alpha = ',c('1.0','1.25','2.0'),sep=''),lty=1:3)
>

```

In this simulation there were 752/1000 simulated values of T within 10, and 32/1000 greater than 1000. Results will vary. From Figure 27.11 the power law $p_X(k) \propto 1/k^\alpha$ holds approximately, with $\alpha \approx 1.25$, and more generally with $\alpha < 2$. We then have, for some constant c ,

$$E[T] = \sum_{k=1}^{\infty} k \frac{c}{k^\alpha}$$

noting that the support of T is unbounded. However, $E[T] < \infty$ only if $\alpha > 2$ (compare the summation to the integral $\int_1^\infty x^{-\alpha} dx$). If $\alpha < 2$ then in our case $E[T] = \infty$. This means that although the gambler can win a gain of 1 with probability 1 in a finite amount of time, the *gain rate* is 0, since $E[T] = \infty$. If we let G_n be the total gain after the n th game (not day), we would find

$$\lim_{n \rightarrow \infty} \frac{G_n}{n} = 0.$$

As a practical matter, using this strategy, we would often find that we cannot play enough games in a single day to achieve the daily gain of 1.

Problem 27.8. Whether or not a Markov chain is an adequate model for a given application is an important question. We'll use a Markov chain model to design a simple tic-tac-toe player. It will play both sides.

Create in R the following objects:

- (a) The board will consist of a vector of length 9. An unoccupied position is set to 0, otherwise the position is occupied by player 1 or 2.
- (b) A tic-tac-toe board has 8 ‘rows’. The three horizontal rows are (1,2,3), (4,5,6) and (7,8,9), the three vertical rows are (1,4,7), (2,5,8) and (3,6,9). The diagonal rows are (1,5,9) and (3,5,7). Create an 8×3 table which stores these rows.

- (c) Each position on the board belongs to certain rows. For example, position 6 belongs to rows (3,6,9) and (4,5,6), and so on. Create a list of length 9, in which the i th element is a vector of indices referencing the rows to which position i belongs.
- (d) To choose a move, player 1 examines each position, and assigns each a score. If position i is occupied it is assigned score 0. Otherwise, each row containing i is looked up. The number of positions in that row occupied by 1 and 2 are stored in `n.us` and `n.them` respectively. The row is scored according to the following table:

		<code>n.them</code> =		
		0	1	2
<code>n.us</code> =	0	10	100	10,000
	1	1,000	1	0
	2	100,000	0	0

Then, the score for position i is the sum of the scores of the rows containing i . For example, for the following board it is player 1's turn to move. Positions 5 and 7 are scored 0, since they are occupied. To score position 1, note that it is contained in 3 rows, (1,2,3), (1,4,7) and (1,5,9). For row (1,2,3), `n.us` = 0, `n.them` = 0, so this row contributes 10 to the score. For row (1,4,7) `n.us` = 0, `n.them` = 1, and for row (1,5,9) `n.us` = 1, `n.them` = 0, so these rows contribute 100 and 1,000, respectively. The total score for position 1 is then $1,000 + 100 + 10 = 1,110$.

0	0	0
0	1	0
2	0	0

- (e) After each position's score is calculated the position with the highest score is selected. If more than one position has the maximum score, one of these is chosen at random.
- (f) Player 2 uses the same strategy, calculating `n.us` and `n.them` accordingly.
- (g) Write an R program to simulate a tic-tac-toe game with alternating players using the same strategy. The game ends after one player completely occupies any row (and therefore wins), or the board is full. Run the simulation 1,000 times, and store the frequency of outcomes (player 1 wins, player 2 wins or the games ends in a draw). What are the frequencies of each outcome?
- (h) Which of the three outcomes can occur? Justify your answer. Symmetry plays a role here.

SOLUTION:

The following R program plays the game as defined in the question. All games in the 1,000 simulations end in draws.

```
#### create row.table: 8x3 matrix of row definitions

row.table = matrix( c(1,2,3,4,5,6,7,8,9,1,4,7,2,5,8,3,6,9,1,5,9,3,5,7),
```

```

ncol=3, byrow=T)

### create row.list. The ith element is a vector of indices to all
### rows in row.table in which position i is located

row.list = vector('list',9)
row.list[[1]] = c(1,4,7)
row.list[[2]] = c(1,5)
row.list[[3]] = c(1,6,8)
row.list[[4]] = c(2,4)
row.list[[5]] = c(2,5,7,8)
row.list[[6]] = c(2,6)
row.list[[7]] = c(3,4,8)
row.list[[8]] = c(3,5)
row.list[[9]] = c(3,6,7)

### score.matrix is a 3 x 3 matrix. Element score.matrix[n.us+1, n.them+1]
### gives the score for (n.us, n.them)

score.matrix = matrix(0, nrow=3, ncol=3)
score.matrix[3,1] = 100000
score.matrix[1,3] = 10000
score.matrix[2,1] = 1000
score.matrix[1,2] = 100
score.matrix[1,1] = 10
score.matrix[2,2] = 1

### choose.move is a function which accepts the current board,
### the values us.them = c(n.us, n.them), and the objects
### row.table, row.list, score.matrix created above.
### The score is calculated for each position. The highest score is identified.
### If more than one position has the highest score, one of them is chosen at random.
### The output is a list with elements names move (the selected position),
### max.score (the maximum score),
### score.temp (the vector of scores for each position)

choose.move = function(board, us.them, row.table, row.list, score.matrix) {

  score.temp = rep(0,9)

  # calculate score for each position

  for (i in 1:9) {
    if (board[i] == 0) {
      for (j in 1:length(row.list[[i]])) {
        score.temp[i] = score.temp[i] + score.matrix[j, row.list[[i]][j]]
      }
    }
  }

  max.score = max(score.temp)
  move = which(score.temp == max.score)
  move = move[1]
}

```

```

        row.temp = board[row.table[row.list[[i]][j], ]]
        n.us = sum(row.temp==us.them[1])
        n.them = sum(row.temp==us.them[2])
        score.temp[i] = score.temp[i] + score.matrix[n.us+1,n.them+1]
    }
}

# determine maximum score

max.score = max(score.temp)

# identify positions with maximum score

move.list = which(score.temp==max.score)

if (length(move.list)==1) {

    # if highest scoring position is unique, copy onto move

    move = move.list[1]
} else
{

    # otherwise, select position at random from highest scoring ones

    move = sample(which(score.temp==max.score),1)
}

# return selected position and score

return(list(move=move, max.score=max.score, score.temp=score.temp))
}

### simulate nsim games

nsim = 1000

### store result in sv 0=draw, 1=Player 1 wins, 2 = Player 2 wins

sv = rep(0, nsim)

for (iii in 1:nsim) {

```

```
# create playing board as vector of length 9
board = rep(0,9)

# flag==1 is used to indicate end of game

flag = 0
while (flag == 0) {

  # Player 1 plays

  junk = choose.move(board, c(1,2), row.table, row.list, score.matrix)

  # update board

  board[junk$move] = 1

  # Player 1 wins if score is 100000. Game ends if there are no
  #   empty positions on the board

  if ( (junk$max.score >= 100000) | (sum(board==0)==0) ) {
    flag=1
    if (junk$max.score >= 100000) {sv[iii] = 1}
  }

  # Player 2 plays if flag==0

  if (flag == 0) {

    # Player 2 plays

    junk = choose.move(board, c(2,1), row.table, row.list, score.matrix)

    # update board

    board[junk$move] = 2

    # Player 2 wins if score is 100000. Game ends if there are no
    #   empty positions on the board
    if ( (junk$max.score >= 100000) | (sum(board==0)==0) ) {
      flag=1
      if (junk$max.score >= 100000) {sv[iii] = 2}
    }
  }
}
```

```

}

# Examine results

table(sv)

```

After the program is run we should see something like this:

```

> table(sv)
sv
 0
1000

```

The table below shows the sequence of one game, including scores for each position, and the subsequent move. For the first move, note that when the board is empty, the middle position (5) is scored highest, so Player 1 always starts there.

For Player 2's first move, all four diagonal positions (1,3,7,9) are scored highest. Player 2 selects one of these at random. However, note that by symmetry there is no important difference between these moves.

For Player 1's second move, the two remaining diagonal positions which share a row with Player 2's first position are scored highest. Player 1 chooses one of these at random. Both are essentially identical, after symmetry is accounted for.

If we continue in this way for the remaining moves, we see that after symmetry is accounted for, there is really only one available move at each turn. All games must be essentially identical, and will therefore end in draws.

	Score			Board		
1	30	20	30	0	0	0
	20	40	20	0	1	0
	30	20	30	0	0	0
2	120	110	120	0	0	0
	110	0	110	0	1	0
	120	110	120	0	0	2
3	21	1010	1110	0	0	0
	1010	0	1100	0	1	0
	1110	1100	0	1	0	2
4	111	110	11010	0	0	2
	200	0	1100	0	1	0
	0	101	0	1	0	2
5	1101	1100	0	0	0	2
	2000	0	11000	0	1	1
	0	1001	0	1	0	2
6	1101	1100	0	0	0	2
	10100	0	0	2	1	1
	0	101	0	1	0	2
7	102	1100	0	0	1	2
	0	0	0	2	1	1
	0	1001	0	1	0	2
8	3	0	0	0	1	2
	0	0	0	2	1	1
	0	10001	0	1	2	2
9	3	0	0	1	1	2
	0	0	0	2	1	1
	0	0	0	1	2	2

Chapter 28

Practice Problems - Classification

28.1 Exercises

Problem 28.1. Under given conditions two sharp shooters $i = 1, 2$ are able to hit a target within a distance X , where X has an exponential density with mean μ_i (ie with density function $f(x) = \mu_i^{-1} \exp(-x/\mu_i)$, $x \geq 0$). Suppose we have independent observations of target accuracy X_1, \dots, X_n from one of the sharp shooters, and we wish to build a Bayesian classifier to predict that identity. Assume $\mu_1 < \mu_2$ are known, and that the prior probabilities are π_1, π_2 .

Show that the classifier requires only the sum $S = \sum_{i=1}^n X_i$, and give precise conditions under which identity $i = 1$ would be predicted.

SOLUTION:

For each class $i = 1, 2$ the joint distribution of $X = (X_1, \dots, X_n)$ is

$$f(x_1, \dots, x_n) = \prod_{k=1}^n \mu_i^{-1} e^{-x_k/\mu_i}.$$

The classifier function for class i may be given by

$$\begin{aligned} h_i(x_1, \dots, x_n) &= \log(f_i(x_1, \dots, x_n)\pi_i) \\ &= \log(\pi_i) - n \log(\mu_i) - \sum_{k=1}^n x_k/\mu_i. \end{aligned}$$

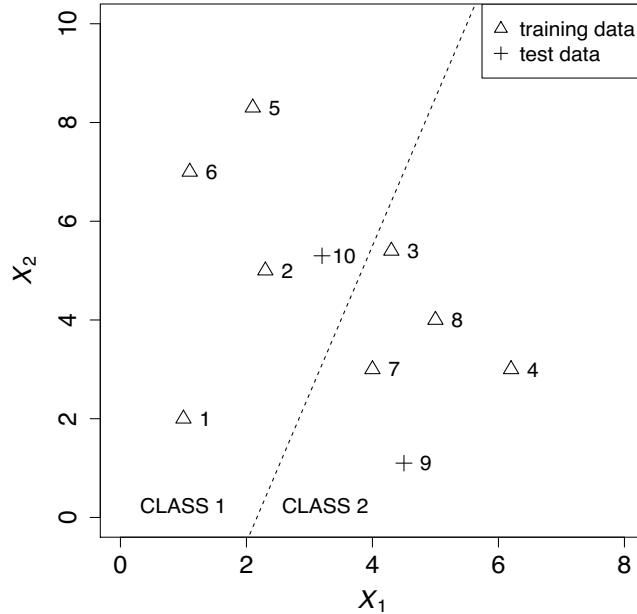
Class $i = 1$ is predicted if $h_1(x_1, \dots, x_n) - h_2(x_1, \dots, x_n) > 0$, which is equivalent to

$$\left[\log(\pi_1) - n \log(\mu_1) - \sum_{k=1}^n x_k/\mu_1 \right] - \left[\log(\pi_2) - n \log(\mu_2) - \sum_{k=1}^n x_k/\mu_2 \right] > 0,$$

or, more concisely,

$$\sum_{k=1}^n x_k < \frac{\log(\pi_1/\pi_2) - n \log(\mu_1/\mu_2)}{\mu_1^{-1} - \mu_2^{-1}}$$

Problem 28.2. To build a KNN classifier, the data in the following plot is used, partitioned into training and test data. That is, the test data is used to evaluate the accuracy of the KNN classifier built using the training data. As it happens, there are two classes, indicated in the plot by a class boundary. The pairwise distances are also given. Give the predicted class for each test observation (labelled 9 and 10), using neighborhood sizes $K = 1$ and $K = 3$. Show clearly how these were obtained.



Pairwise Distance =										
	1	2	3	4	5	6	7	8	9	10
1	0.00	3.27	4.74	5.30	6.40	5.00	3.16	4.47	3.61	3.97
2	3.27	0.00	2.04	4.38	3.31	2.33	2.62	2.88	4.48	0.95
3	4.74	2.04	0.00	3.06	3.64	3.58	2.42	1.57	4.30	1.10
4	5.30	4.38	3.06	0.00	6.70	6.48	2.20	1.56	2.55	3.78
5	6.40	3.31	3.64	6.70	0.00	1.64	5.63	5.19	7.59	3.20
6	5.00	2.33	3.58	6.48	1.64	0.00	4.94	4.92	6.81	2.70
7	3.16	2.62	2.42	2.20	5.63	4.94	0.00	1.41	1.96	2.44
8	4.47	2.88	1.57	1.56	5.19	4.92	1.41	0.00	2.94	2.22
9	3.61	4.48	4.30	2.55	7.59	6.81	1.96	2.94	0.00	4.40
10	3.97	0.95	1.10	3.78	3.20	2.70	2.44	2.22	4.40	0.00

SOLUTION:

The correct classes for test observations $i = 9, 10$ are $y_i = 2, 1$, respectively.

For $K = 1$, observation $i = 9$, the neighborhood is $N = \{7\}$, so $\hat{y}_9 = 2$ [correct], since observation 7 is class 2. For $i = 10$, $N = \{2\}$, $\hat{y}_{10} = 1$ [correct], since observation 2 is class 1.

For $K = 3$, observation $i = 9$, the neighborhood is $N = \{4, 7, 8\}$, so $\hat{y}_9 = 2$ [correct], since all

observations in N are class 2. For $i = 10$, $N = \{2, 3, 8\}$, $\hat{y}_{10} = 2$ [incorrect], since 2/3 in N are class 2.

Problem 28.3. We are given 2 classes, $j = 1, 2$. The distribution of a single dimensional observation is given by $X \sim N(\mu_j, \sigma_j^2)$, given classes $j = 1, 2$. Available estimates of μ_j are given by $\bar{X}_1 = 44.5$, $\bar{X}_2 = 20.7$. We assume $\sigma_1^2 = \sigma_2^2$, and a pooled estimate of the common variance is given by $s_{pooled}^2 = 3.82$. We accept as prior class probabilities $\pi_1 = 0.4$, $\pi_2 = 0.6$. Suppose an LDA classifier is constructed. Determine in which regions for which X predicts each class.

SOLUTION:

For LDA, the classifier is given by

$$\hat{y} = \operatorname{argmax}_j h_j(x)$$

where

$$h_j(x) = x\mu_j/\sigma^2 - \frac{1}{2}\mu_j^2/\sigma^2 + \log(\pi_j).$$

The classification boundary x_b is the solution to $h_1(x_b) = h_2(x_b)$. There is only one, since the $h_j(x)$ are linear. This gives, after substituting the estimates,

$$x_b \times (44.5/3.82) - \frac{1}{2} \times 44.5^2/3.82 + \log(0.4) = x_b \times (20.7/3.82) - \frac{1}{2} \times 20.7^2/3.82 + \log(0.6)$$

or,

$$\begin{aligned} x_b \times \frac{44.5 - 20.7}{3.82} &= -\frac{1}{2} \times \frac{44.5^2 - 20.7^2}{3.82} + \log(0.6/0.4), \\ x_b &= 32.665, \end{aligned}$$

so that class $y = 1$ is predicted when $X > x_b = 32.665$.

Problem 28.4. A certain classification problem involves 2 classes $j = 1, 2$, and a random observation of the form $X \in \{1, 2, 3, 4\}$. Suppose the prior probabilities π_j of class j are given by $\pi_1 = 1 - \pi_2 = 3/4$. The following table gives the conditional distribution $f(x | j)$ of X :

$x =$	1	2	3	4
$f(x j = 1)$	1/2	1/4	1/4	0
$f(x j = 2)$	0	1/3	1/3	1/3

- (a) What is the posterior probability of class $j = 1$ given $X = 2$?
- (b) Give the prediction made by a Bayes classifier for each outcome $X = 1, 2, 3, 4$. Justify your answers numerically.

SOLUTION:

(a) We have

$$\begin{aligned}
 P(j = 1 \mid X = 2) &= \frac{P(X = 2 \mid j = 1)P(j = 1)}{P(X = 2)} \\
 &= \frac{f(2 \mid j = 1)\pi_1}{f(2 \mid j = 1)\pi_1 + f(2 \mid j = 2)\pi_2} \\
 &= \frac{(1/4) \times (3/4)}{(1/4) \times (3/4) + (1/3) \times (1/4)} \\
 &= \frac{(3/4)}{(3/4) + (1/3)} \\
 &= \frac{9}{13}.
 \end{aligned}$$

(b) The Bayes classifier is given by

$$\hat{j} = \operatorname{argmax}_j h_j(x)$$

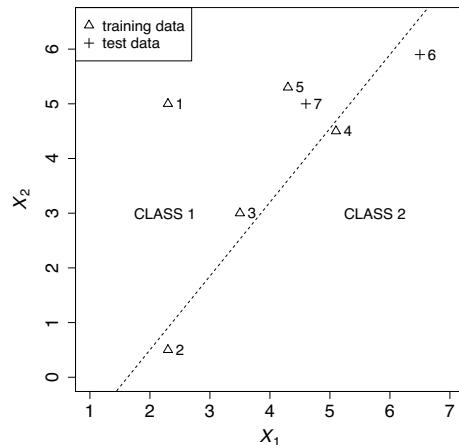
where

$$h_j(x) = f(x \mid j)\pi_j.$$

These values, along with \hat{j} , are given in the following table:

$x =$	1	2	3	4
$h_1(x)$	3/8	3/16	3/16	0
$h_2(x)$	0	1/12	1/12	1/12
\hat{j}	1	1	1	2

Problem 28.5. To build a KNN classifier, the data in the following plot is used, partitioned into training and test data (see the appropriate symbols in the plot legend). As it happens, there are two classes, indicated in the plot by a class boundary (the dashed line). The pairwise distances are also given. By evaluating the classifier with the test data, estimate the classification errors for neighborhood sizes $K = 1$ and $K = 3$. When evaluating a prediction, specify the neighborhood exactly. Note that the KNN classifier itself is built using only the training data.



	Pairwise Distance =					
	1	2	3	4	5	6
2	4.500					
3	2.332	2.773				
4	2.844	4.883	2.193			
5	2.022	5.200	2.435	1.131		
6	4.295	6.841	4.173	1.980	2.280	
7	2.300	5.054	2.283	0.707	0.424	2.102

SOLUTION:

The correct classes for test observations $i = 6, 7$ are $y_i = 2, 1$.

For $K = 1$, observation $i = 6$, the neighborhood is $N = \{4\}$, so $\hat{y}_6 = 2$. For $i = 7$, $N = \{5\}$, $\hat{y}_7 = 1$. This means $CE = 0.0$.

For $K = 3$, observation $i = 6$, the neighborhood is $N = \{3, 4, 5\}$, so $\hat{y}_6 = 1$ (2/3 in N are class 1). For $i = 7$, $N = \{3, 4, 5\}$, $\hat{y}_7 = 1$ (2/3 in N are class 2). This means $CE = 1/2$.

Problem 28.6. We are given 2 classes, $j = 1, 2$. The distribution of a single dimensional observation is given by $X \sim N(\mu_j, \sigma_j^2)$, given classes $j = 1, 2$. Available estimates of μ_j are given by $\bar{X}_1 = 102.5$, $\bar{X}_2 = 143.8$. We assume $\sigma_1^2 = \sigma_2^2$, and a pooled estimate of the common variance is given by $s_{pooled}^2 = 5.03$. We accept as prior class probabilities $\pi_1 = 0.7$, $\pi_2 = 0.3$. Suppose an LDA classifier is constructed. Determine the region for X which predicts class $j = 1$.

SOLUTION:

For LDA, the classifier is given by

$$\hat{y} = \operatorname{argmax}_j h_j(x)$$

where

$$h_j(x) = x\mu_j/\sigma^2 - \frac{1}{2}\mu_j^2/\sigma^2 + \log(\pi_j).$$

The classification boundary x_b is the solution to $h_1(x_b) = h_2(x_b)$. There is only one, since the $h_j(x)$ are linear. This gives, after substituting the estimates,

$$x_b \times (102.5/5.03) - \frac{1}{2} \times 102.5^2/5.03 + \log(0.7) = x_b \times (143.8/5.03) - \frac{1}{2} \times 143.8^2/5.03 + \log(0.3)$$

or,

$$\begin{aligned} x_b \times \frac{102.5 - 143.8}{5.03} &= -\frac{1}{2} \times \frac{102.5^2 - 143.8^2}{5.03} + \log(0.7/0.3), \\ x_b &= 123.2532, \end{aligned}$$

so that class $y = 1$ is predicted when $X < x_b = 123.2532$.

Problem 28.7. A total of n ancient Roman coins are discovered scattered at an archaeological site. Each coin has a distinctive mark identifying the mint at which the coin was produced. Suppose at the time the coins were produced there existed m mints in operation, and that the exact number m is of interest to historians. We may therefore calculate frequencies N_j , $j = 1, \dots, m$, equalling the number of coins in the collection from the j th observed mint label, so that $N_1 + \dots + N_m = n$. If N_+ is the number of distinct mints observed in the sample, we at least know that $m \geq N_+$. Assume, for convenience, that a coin is equally likely to come from any mint, and that the mint assignments are independent. We then interpret (N_1, \dots, N_m) as a multinomial vector (but for a more subtle interpretation of these frequencies see Nayak TK (1992) “On statistical analysis of a sample from a population of unknown species”, *Journal of Statistical Planning and Inference*).

- (a) Suppose we wish to develop a Bayes classifier to predict m . Assume we are given a prior distribution $\pi_j = P(m = j)$. Write explicitly the posterior distribution, and show how this can be interpreted as a Bayes classifier.
- (b) Suppose prior belief favors $m = 15$, and it is known that at least $m = 10$ mints existed. It is then assumed that $m \in [10, 20]$, so $\pi_j = 0$ if $j \notin [10, 20]$, and otherwise

$$(\pi_{10}, \pi_{11}, \dots, \pi_{19}, \pi_{20}) = K \times (1^2, 2^2, \dots, 5^2, 6^2, 5^2, \dots, 2^2, 1^2),$$

for some normalization constant K . Then, suppose there are $n = 8$ coins, with 2 mints represented by 2 coins and 4 mints are represented by 1 coin. Plot the prior and posterior densities for m . What value of m does the Bayes classifier predict?

SOLUTION:

- (a) Suppose we are given a probability distribution $P = (p_1, \dots, p_m)$ on $\mathcal{S} = \{1, \dots, m\}$. If we are given an *iid* sample of size n from P , and we let $\tilde{N} = (N_1, \dots, N_m)$ be the vector of sample frequencies for each outcome, then \tilde{N} has a *multinomial distribution* with density given by

$$f_{\tilde{N}}(n_1, \dots, n_m) = P(N_1 = n_1, \dots, N_m = n_m) = \frac{n!}{\prod_{i=1}^m n_i!} \prod_{i=1}^m p_i^{n_i}, \quad \min_i n_i \geq 0, \quad n_1 + \dots + n_m = n.$$

For the coin frequencies, given m mints we have distribution $P = (1/m, \dots, 1/m)$, which gives conditional density

$$\begin{aligned} P(n_1, \dots, n_m | m) &= \frac{n!}{\prod_{i=1}^m n_i!} \prod_{i=1}^m p_i^{n_i} \\ &= \frac{n!}{\prod_{i=1}^m n_i!} \prod_{i=1}^m (1/m)^{n_i} \\ &= \frac{n!}{\prod_{i=1}^m n_i!} (1/m)^n, \end{aligned}$$

if $m \geq N_+$, and $P(n_1, \dots, n_m | m) = 0$ otherwise. Since n is fixed, and $0! = 1$, the multinomial coefficient in the conditional probability does not depend on m , so we may write

$$P(n_1, \dots, n_m | m) \propto (1/m)^n.$$

Thus, we have

$$P(n_1, \dots, n_m | m) = \begin{cases} K/m^n & ; \quad N_+ \leq m \\ 0 & ; \quad N_+ > m \end{cases}$$

where K is a normalization constant which does not depend on m . This gives posterior density

$$\pi(m | n_1, \dots, n_m) = \begin{cases} K' \pi_m / m^n & ; \quad N_+ \leq m \\ 0 & ; \quad N_+ > m \end{cases},$$

where K' is a normalization constant which does not depend on m , and Bayes classifier

$$\hat{m} = \operatorname{argmax}_m \pi(m | n_1, \dots, n_m).$$

- (b) The following code calculates and plots the densities (Figure 28.1). From the plot of the posterior density we have directly $\hat{m} = 12$.

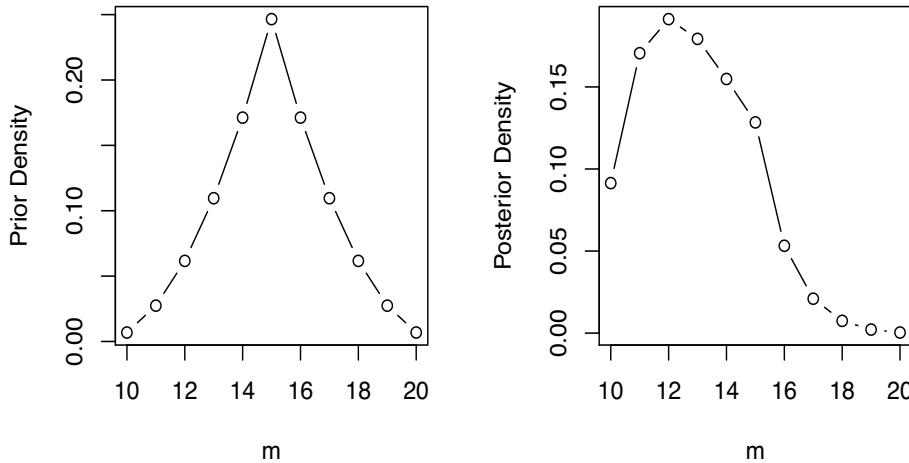


Figure 28.1: Figure for Problem 28.7 (b).

```
### data (we only need the sum and the number of nonzero frequencies)

x0 = c(2,2,1,1,1,0,0,0,0)
n = sum(x0)
n.plus = sum(x0 > 0)
c(n,n.plus)

### conditional density (up to normalization constant)

p.cond = (10:20)^(-n)

### prior density (we'll normalize this density, although its not strictly needed)
```

```

p.prior = c(1:5,6,5:1)^2
p.prior = p.prior/sum(p.prior)

### posterior density (we'll normalize this density, although
### its not strictly needed)

p.post = p.cond*p.prior
p.post = p.post/sum(p.post)

### plot densities

pdf('figq2b.pdf')
par(mfrow=c(1,2),pty='s')
plot(10:20,p.prior,type='b',xlab='m',ylab='Prior Density')
plot(10:20,p.post,type='b',xlab='m',ylab='Posterior Density')
dev.off()

```

28.2 Data Analysis

Problem 28.8. Suppose we may observe a vector of random counts $X = (X_1, \dots, X_m)$ which are statistically independent, with $X_i \sim Poisson(\lambda_i)$. The mean vector is then $\Lambda = (\lambda_1, \dots, \lambda_m)$. Next, suppose we have a classification problem in which the vector of Poisson counts X comes from class A or B , defined by respective mean vectors $\Lambda_A = (\lambda_1^A, \dots, \lambda_m^A)$ or $\Lambda_B = (\lambda_1^B, \dots, \lambda_m^B)$.

- (a) Suppose Λ_A, Λ_B are known. Suppose that the respective classes have prior probabilities π_A, π_B . Show that the Bayes classifier can be constructed, given observation $X = (X_1, \dots, X_m)$, from two functions of X of the form:

$$\begin{aligned} h_A(X) &= a_0 + \sum_{i=1}^m a_i X_i, \\ h_B(X) &= b_0 + \sum_{i=1}^m b_i X_i, \end{aligned}$$

with the prediction being A if $h_A(X) > h_B(X)$ and B if $h_B(X) > h_A(X)$ (with the prediction made randomly when $h_B(X) = h_A(X)$).

- (b) Write an R function which uses training data to develop the Bayes classifier of part (a), then applies the classifier to test data. Note that the Poisson parameter λ may be estimated by a sample mean. Apply this function to files `A2train.csv` and `A2test.csv` (posted on Blackboard). The files can be imported using the commands:

```
xmat.train = read.table(file='A2train.csv', header=T, sep=',')
xmat.test = read.table(file='A2test.csv', header=T, sep=',')
```

Apply also an LDA classifier and a KNN classifier ($K = 1$) to the same training and test data. Report confusion matrices and error rates for each. How do the respective classifiers differ in performance?

SOLUTION:

The density of the Poisson random variable with mean λ is

$$f(i) = \frac{\lambda^i}{i!} e^{-\lambda}, \quad i = 0, 1, \dots$$

(a) By independence, for class $y \in \{A, B\}$,

$$\begin{aligned} f(x_1, \dots, x_m \mid y = A) &= \prod_{i=1}^m \frac{(\lambda_i^A)^{x_i}}{x_i!} e^{-\lambda_i^A} \\ f(x_1, \dots, x_m \mid y = B) &= \prod_{i=1}^m \frac{(\lambda_i^B)^{x_i}}{x_i!} e^{-\lambda_i^B} \end{aligned}$$

The Bayes classifier is

$$\hat{y} = \operatorname{argmax}_{y \in \{A, B\}} f(x_1, \dots, x_m \mid y) \pi_y.$$

If we take a log transformation we have, for $y = A$

$$\begin{aligned} \log(f(x_1, \dots, x_m \mid A) \pi_A) &= \log \left(\prod_{i=1}^m \frac{(\lambda_i^A)^{x_i}}{x_i!} e^{-\lambda_i^A} \right) \\ &= \log(\pi_A) + \sum_{i=1}^m x_i \log(\lambda_i^A) - \log(x_i!) - \lambda_i^A, \end{aligned}$$

and similarly

$$\log(f(x_1, \dots, x_m \mid B) \pi_B) = \log(\pi_B) + \sum_{i=1}^m x_i \log(\lambda_i^B) - \log(x_i!) - \lambda_i^B.$$

If we subtract all terms of the form $-\log(x_i!)$ from the preceding expressions (since they do not depend on class) we have functions

$$\begin{aligned} h_A(x_1, \dots, x_m) &= a_0 + \sum_{i=1}^m a_i x_i, \\ h_B(x_1, \dots, x_m) &= b_0 + \sum_{i=1}^m b_i x_i, \end{aligned}$$

where

$$\begin{aligned} a_i &= \log(\lambda_i^A) \\ a_0 &= \log(\pi_A) - \sum_{i=1}^m \lambda_i^A \\ b_i &= \log(\lambda_i^B) \\ b_0 &= \log(\pi_B) - \sum_{i=1}^m \lambda_i^B \end{aligned}$$

so the Bayes classifier is equivalently given as

$$\hat{y} = \operatorname{argmax}_{y \in \{A, B\}} h_y(X),$$

with ties resolved randomly.

- (b) The following code implements the classifiers:

```
### read data

xmat.train = read.table(file='A2train.csv', header=T, sep=',')
xmat.test = read.table(file='A2test.csv', header=T, sep=',')

### extract feature data
### group vector gr is same for training and test data

n = 100
gr = rep(c(1,2), each=n)
xtrain = xmat.train[, 2:4]
xtest = xmat.test[, 2:4]

### subroutine for Bayes classifier based on Poisson distribution
### note that mest1 and mest2 are global objects

mest1 = apply(xtrain[gr==1,], 2, mean)
mest2 = apply(xtrain[gr==2,], 2, mean)
BayesPoiss = function(x) {
  b1 = sum(x*log(mest1)) - sum(mest1)
  b2 = sum(x*log(mest2)) - sum(mest2)
  ans = 1*(b1 > b2) + 2*(b2 > b1) + sample(c(1,2), 1)*(b1==b2)
  return(ans)
}

### apply subroutine

pr3 = apply(xtest, 1, BayesPoiss)
confusion.matrix.bayes = table(pr3, gr)
```

```

accuracy.bayes = mean(pr3==gr)

### apply LDA

fit1 = lda(xtrain,gr)
pr1 = predict(fit1,xtest)$class
confusion.matrix.lda = table(pr1,gr)
accuracy.lda = mean(pr1==gr)

### apply KNN (K=1)

fit2= knn(xtrain,xtest,gr, k = 1)
pr2 = fit2
confusion.matrix.knn1 = table(pr2,gr)
accuracy.knn1 = mean(pr2==gr)

```

The output is shown below. We have a correct classification rate of 0.875, 0.87, 0.85 for the Bayesian classifier, LDA, and KNN respectively. Although the Bayes classifier has (slightly) higher accuracy, the others are close, and can both be considered approximations of the Bayesian classifier.

```

> ### print results
>
> # Bayes classifier
>
> confusion.matrix.bayes
      gr
pr3  1  2
  1 81  6
  2 19 94
> accuracy.bayes
[1] 0.875
>
> # LDA
>
> confusion.matrix.lda
      gr
pr1  1  2
  1 81  7
  2 19 93
> accuracy.lda
[1] 0.87
>
> # KNN (K = 1)
>
> confusion.matrix.knn1

```

```

gr
pr2 1 2
 1 80 10
 2 20 90
> accuracy.knn1
[1] 0.85
>

```

Problem 28.9. This question will make use of data downloaded from GEO (Gene Expression Omnibus) with series accession number GSE10245. This data set contains gene expression profiles from non-small cell lung cancer tumor tissue. There are $n = 58$ gene expression profiles (from distinct subjects), collected from two cancer subtypes, adenocarcinoma (AC) ($n = 40$) and squamous cell carcinoma (SCC) ($n = 18$). The data to be used in this problem can be obtained from Blackboard, but you may also consult the link:

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10245>

The data set is in file `GSE10245.csv`. It has 58 rows (one for each gene expression profile) and 51 columns. Column 1 identifies the group (1 = AC; 2 = SCC). The remaining columns contain expressions from 50 genes. The gene symbols are given in the header. The file can be imported by the following command:

```
GSE10245.data = read.table(file='GSE10245.csv', header=T, sep=',')
```

The object of this question is to evaluate distinct classifiers on multiple, but similar, data sets.

- (a) Divide the data into 10 separate data sets, each consisting of 5 genes. Data sets should use genes from columns 2 – 6, 7 – 11, ..., 46 – 51.
- (b) Write a main function which accepts a class vector and a data set of features, and which applies 5 classification methods to the data, giving an estimated classification error CE for each:
 - (a) LDA, CE estimated using training data only;
 - (b) LDA, CE estimated using LOO cross-validation;
 - (c) QDA, CE estimated using training data only;
 - (d) QDA, CE estimated using LOO cross-validation;
 - (e) KNN, allowing K to vary over 1, 3, 5, ..., 23, 25. For each model, estimate CE using LOO cross-validation. Select K yielding the minimum CE . In the case of ties, select the smallest K yielding the minimum CE .

Why would we only consider odd numbers for K in the KNN classifier?

- (c) Apply the main function to each of the ten data sets. Construct a table with a row for each data set and a column for each of the 5 classifier methods. The table entry will be CE .
- (d) Compute the mean error for each method across the 10 data sets.

- (a) Which methods report the lowest and highest CE ?
- (b) How do error rates differ between using training data only and using cross-validation within the same method?
- (c) Which method would you recommend?
- (e) Instead of averaging error rates, we can average *blocked ranks*. That is, within each row, replace CE with their (within row) ranks. Each row then contains the ranks 1-5. Repeat part (d) using the average ranks. Is the conclusion the same?
- (f) For the first 2 data sets, plot CE against $K = 3, 5, \dots, 23, 25$. For each data set, what are the minimum and maximum values of K yielding the minimum CE ?

SOLUTION:

Take the following steps

- (a) The following code creates class vector `gr` and feature matrix `gem1`.

```
GSE10245.data = read.table(file='GSE10245.csv',header=T,sep=',')
gem1 = GSE10245.data[,2:51]
gr = GSE10245.data[,1]
```

- (b) The following code implements the required algorithms. For the KNN classifier, if K is even, the prediction will be randomized in case of ties.

```
main.function = function(gem1, gr) {

  ### vector to contain CE

  crate = rep(0,5)

  ### LDA

  fit.lda = lda(gem1,gr)
  pr.lda = predict(fit.lda)$class
  crate[1] = 1 - mean(pr.lda==gr)

  ### LDA-CV

  fit.ldacv = lda(gem1,gr,CV=T)
  pr.ldacv = fit.ldacv$class
  crate[2] = 1 - mean(pr.ldacv==gr)

  ### QDA

  fit.qda = qda(gem1,gr)
  pr.qda = predict(fit.qda)$class
  crate[3] = 1 - mean(pr.qda==gr)
```

```

#### QDA-CV

fit.qdacv = qda(gem1,gr,CV=T)
pr.qdacv = fit.qdacv$class
crate[4] = 1 - mean(pr.qdacv==gr)

#### KNN

# list of K values
k.list = seq(1,25,by=2)
nk = length(k.list)

# error vector
knn.err = integer(nk)

# loop through K values
for (i in 1:nk) {
  fit.knn = knn.cv(gem1,gr,k=k.list[i],prob=T)
  pr.knn = fit.knn
  knn.err[i] = 1-mean(pr.knn==gr)
}

# capture optimal K
k.best = min(k.list[knn.err==min(knn.err)]) 

# refit with opimal K
fit.knn = knn.cv(gem1,gr,k=k.best)
pr.knn = fit.knn
crate[5] = 1 - mean(pr.knn==gr)

# return error rates, and data from KNN algorithm
return(list(crate=crate, k.list=k.list, knn.err=knn.err, k.best=k.best))
}

```

(c) The following code may be used:

```

#### Read data

GSE10245.data = read.table(file='GSE10245.csv',header=T,sep=',')
gem1 = GSE10245.data[,2:51]
gr = GSE10245.data[,1]

### Apply main.function() to data sets extracted by column subsetting of gem1

tab1 = NULL

```

```
for (i in 1:10) {tab1 = rbind(tab1, main.function(gem1[,c(1:5)+(i-1)*5], gr)$crate) }
```

- (d) The following code may be used:

```
### Append column averages, then label tab1

tab1 = rbind(tab1,apply(tab1,2,mean))
rownames(tab1) = c(paste('Data Set',1:10),"Average")
colnames(tab1) = c('LDA','LDA-CV','QDA','QDA-CV','KNN')
```

We get the following table:

```
> tab1
      LDA    LDA-CV     QDA    QDA-CV     KNN
Data Set 1 0.08620690 0.08620690 0.05172414 0.1034483 0.08620690
Data Set 2 0.10344828 0.15517241 0.05172414 0.1379310 0.08620690
Data Set 3 0.05172414 0.05172414 0.08620690 0.1206897 0.05172414
Data Set 4 0.10344828 0.10344828 0.05172414 0.1379310 0.10344828
Data Set 5 0.08620690 0.10344828 0.12068966 0.1551724 0.08620690
Data Set 6 0.06896552 0.10344828 0.05172414 0.1034483 0.12068966
Data Set 7 0.10344828 0.10344828 0.06896552 0.0862069 0.10344828
Data Set 8 0.10344828 0.13793103 0.10344828 0.1206897 0.08620690
Data Set 9 0.08620690 0.08620690 0.08620690 0.1206897 0.08620690
Data Set 10 0.03448276 0.10344828 0.03448276 0.1379310 0.08620690
Average     0.08275862 0.10344828 0.07068966 0.1224138 0.08965517
```

- (a) The lowest error is reported by QDA, the highest error is reported by QDA-CV.
 - (b) The reported error is always higher for the CV method.
 - (c) The choice should be based on cross-validated error, so the KNN predictor is the best choice in this sense.
- (e) The following code may be used:

```
tab2 = NULL
for (i in 1:10) {tab2 = rbind(tab2, rank(tab1[i,]))}
tab2 = rbind(tab2,apply(tab2,2,mean))
rownames(tab2) = c(paste('Data Set',1:10),"Average")
colnames(tab2) = c('LDA','LDA-CV','QDA','QDA-CV','KNN')
```

The following table is produced. The ordering of the methods, and therefore the conclusion, is the same as for part (d).

```
> tab2
      LDA LDA-CV   QDA QDA-CV KNN
Data Set 1 3.0    3.0 1.00    5.00 3.0
Data Set 2 3.0    5.0 1.00    4.00 2.0
Data Set 3 2.0    2.0 4.00    5.00 2.0
Data Set 4 3.0    3.0 1.00    5.00 3.0
```

Data Set 5	1.5	3.0	4.00	5.00	1.5
Data Set 6	2.0	3.5	1.00	3.50	5.0
Data Set 7	4.0	4.0	1.00	2.00	4.0
Data Set 8	2.5	5.0	2.50	4.00	1.0
Data Set 9	2.5	2.5	2.50	5.00	2.5
Data Set 10	1.5	4.0	1.50	5.00	3.0
Average	2.5	3.5	1.95	4.35	2.7

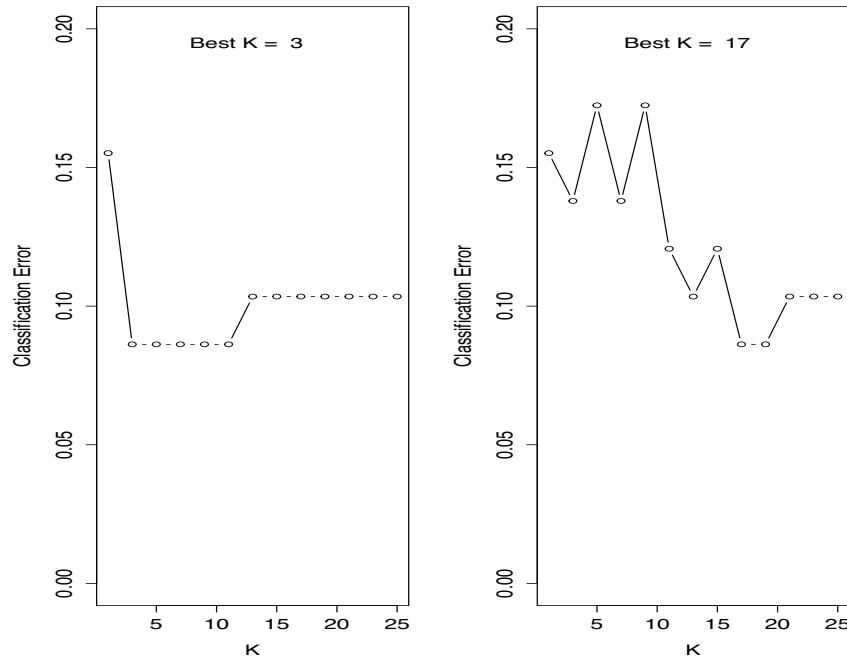


Figure 28.2: Plot for Problem 28.9 (f).

- (f) The following code draws the required plots (Figure 28.2). The best K is, respectively, 3 and 17.

```

### Refit data sets 1,2

i = 1
mf1 = main.function(gem1[,c(1:5)+(i-1)*5], gr)
i = 2
mf2 = main.function(gem1[,c(1:5)+(i-1)*5], gr)

### Draw plot

par(mfrow=c(1,2))
plot(mf1$k.list,mf1$knn.err,type='b',ylim=c(0,0.20), xlab='K',
      ylab='Classification Error')
title("Data Set 1")

```

```

text(12.5,.195,paste('Best K = ',mf1$k.best))
plot(mf2$k.list,mf2$knn.err,type='b',ylim=c(0,0.20), xlab='K',
      ylab='Classification Error')
title("Data Set 2")
text(12.5,.195,paste('Best K = ',mf2$k.best))

```

Problem 28.10. This problem will make use of the `biopsy` data set from the `MASS` library. From the help file:

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. He assessed biopsies of breast tumours for 699 patients up to 15 July 1992; each of nine attributes has been scored on a scale of 1 to 10, and the outcome is also known. There are 699 rows and 11 columns.

The data contains features labeled `V1, ..., V9` and a response labeled `class` with binary tumor outcomes `benign` and `malignant`. The object is to build a predictor which uses the nine features to predict tumor class. A *confusion table* is a contingency table of the form:

	true benign	true malignant
predicted benign	n_{11}	n_{12}
predicted malignant	n_{21}	n_{22}

Any record used for testing the predictor is placed in exactly one of the four cells.

- (a) Prepare the data by first removing the `ID` column, then removing records with missing values using the `na.omit()` function. The new data set should have $n = 683$ records.
- (b) Recall the odds representation of Baye's Rule, in this application:

$$Odds(\text{true malignant} \mid \text{predicted malignant}) = LR_+ \times Odds(\text{true malignant})$$

$$Odds(\text{true malignant} \mid \text{predicted benign}) = LR_- \times Odds(\text{true malignant})$$

Express LR_+ and LR_- in terms of the elements $(n_{11}, n_{12}, n_{21}, n_{22})$ of the confusion table. Create an R function that inputs the confusion table, and outputs a single vector with elements (CE, LR_+, LR_-) , where CE is classification error.

- (c) Using the function `lda()` fit a classifier using linear discriminant analysis (LDA). Use the function of Part (b) to record (CE, LR_+, LR_-) . Do not use cross-validation to fit this classifier.
- (d) Now, we will evaluate the LDA by splitting the data into training and test data. Do this four ways, constructing the training data by indices

$$\begin{aligned} T_1 &= (1, 2, \dots, 340, 341) \\ T_2 &= (342, 343, \dots, 682, 683) \\ T_3 &= (1, 3, \dots, 681, 683) \\ T_4 &= (2, 4, \dots, 680, 682) \end{aligned}$$

In each case fit an LDA classifier with the training data, then construct a confusion table by applying the predictor to the test data, combining the summary statistics (CE, LR_+, LR_-) to those of Part (c) into a single table.

- (e) Finally, create a confusion table using predictions obtained from leave-one-out cross-validation (LOOCV). This may be done using the `CV=TRUE` option for the function `lda()`. Add summary statistics (CE, LR_+, LR_-) to the table contructed in Part (d).
- (f) Examine the summary statistics. Comment on the relative merits of LOOCV versus data splitting.

SOLUTION:

Code for Parts (a)-(e) is given below. Comments folow.

(a)

```
library(MASS)
biopsy2 = biopsy[, -1]
biopsy2 = na.omit(biopsy2)
dim(biopsy2)
```

(b)

```
lrfp = function(m) { (m[2,2]/(m[1,2]+m[2,2]))/(m[2,1]/(m[1,1]+m[2,1])) }
lrfn = function(m) { (m[1,2]/(m[1,2]+m[2,2]))/(m[1,1]/(m[1,1]+m[2,1])) }
f0 = function(m) {c(1-sum(diag(m))/sum(m),lrfp(m),lrfn(m))}
```

(c)

```
fit.lda= lda(class~, data=biopsy2)
confusion.matrix.lda = table(predict(fit.lda)$class,biopsy2$class)
```

we'll need 6 rows.

```
lr.tab = matrix(NA,6,3)
lr.tab[1,] = f0(confusion.matrix.lda)
```

(d)

```
# define training indices
```

```
train.ind.list = list(c(1:341), c(342:683), seq(1,683,2), seq(2,683,2))
```

```
# loop through each training set
```

```
for (iii in 1:4) {
```

```

train.ind = train.ind.list[[iii]]
biopsy2.train = biopsy2[train.ind,]
biopsy2.test = biopsy2[-train.ind,]

fit.lda = lda(class~. , data=biopsy2.train)
confusion.matrix.lda = table(predict(fit.lda,biopsy2.test)$class,
                           biopsy2.test$class)
lr.tab[iii+1,] = f0(confusion.matrix.lda)
}

### (e)

# LOOCV with option CV=TRUE

fit.lda= lda(class~. , data=biopsy2, CV=TRUE)
confusion.matrix.lda = table(fit.lda$class,biopsy2$class)
confusion.matrix.lda
lr.tab[6,] = f0(confusion.matrix.lda)

```

- (a) See above.
 (b) The general for Baye's Rule using odds is:

$$Odds(A | E) = \frac{P(E | A)}{P(E | A^c)} \times Odds(A).$$

Therefore, if

$$Odds(\text{true malignant} | \text{predicted malignant}) = LR_+ \times Odds(\text{true malignant})$$

we have

$$LR_+ = \frac{P(\text{predicted malignant} | \text{true malignant})}{P(\text{predicted malignant} | \text{true benign})} = \frac{n_{22}/(n_{12} + n_{22})}{n_{21}/(n_{11} + n_{21})}.$$

Similarly, if

$$Odds(\text{true malignant} | \text{predicted benign}) = LR_- \times Odds(\text{true malignant})$$

we have

$$LR_- = \frac{P(\text{predicted benign} | \text{true malignant})}{P(\text{predicted benign} | \text{true benign})} = \frac{n_{12}/(n_{12} + n_{22})}{n_{11}/(n_{11} + n_{21})}.$$

- (c) See above.
 (d) See above.
 (e) See above.
 (f) The table is printed below. Splitting the data in various ways results in considerable variation in the estimated properties of the predictor (rows 2-5). On the other hand, LOOCV estimates accuracy to be essentially the same as the original fit using the entire data set (rows 1 and 6).

```
> ### (f)
>
> lr.tab
      [,1]      [,2]      [,3]
[1,] 0.03953148 51.08787 0.08095659
[2,] 0.02046784 122.44444 0.06220506
[3,] 0.07917889 26.44620 0.13741686
[4,] 0.04985337 40.06441 0.10402737
[5,] 0.02923977 69.40496 0.05864736
[6,] 0.03953148 51.08787 0.08095659
```

Problem 28.11. This problem will make use of the `biopsy` data set from the `MASS` library used in Problem 28.10.

- (a) Repeat Parts (a) and (b) of Problem 28.10. Then normalize the 9 features V_1, \dots, V_9 by subtracting the mean and dividing by the standard deviation within each feature. The features now have mean 0 and standard deviation 1.
- (b) We will build a KNN classifier for `class` based on the remaining 9 normalized features. Create an R function which accepts a vector `k.list` of values of K (the neighborhood size of the classifier), a training set of features, and a paired training set of classes. For each K in `k.list` a KNN fit will be evaluated using LOOCV. Use the `knn.cv()` function from the library `class`. The function should output a table with a row for each K in `k.list` consisting of the summary statistics (CE, LR_+, LR_-) for that fit.
- (c) Apply the function of Part (b) to the data set constructed in Part (a). Use `k.list = seq(1,15,2)`. What advantage is there to using only odd numbers for K ?
- (d) Given the output of Part (c), for each of the three summary statistics construct a plot of that statistic against K . Also determine for each the optimal values of K . Do they conform to each other?

SOLUTION:

- (a) Code for Part (a) follows.

```
library(class)
library(MASS)

### (a)

lrfp = function(m) { (m[2,2]/(m[1,2]+m[2,2]))/(m[2,1]/(m[1,1]+m[2,1])) }
lrfn = function(m) { (m[1,2]/(m[1,2]+m[2,2]))/(m[1,1]/(m[1,1]+m[2,1])) }
f0 = function(m) {c(1-sum(diag(m))/sum(m),lrfp(m),lrfn(m))}
```

```

library(class)

biopsy2 = biopsy[,-1]
biopsy2 = na.omit(biopsy2)
biopsy3 = biopsy2

for (i in 1:9) {
    biopsy3[,i] = (biopsy2[,i] - mean(biopsy2[,i]))/sd(biopsy2[,i])
}

```

- (b) The function can be written as follows.

```

### (b)

knn.function = function(k.list, xtrain, gr) {

pr.tab = matrix(NA,length(k.list),3)
for (i in 1:length(k.list)) {
    knn.fit = knn.cv(xtrain,gr,k=k.list[i],use.all=T)
    cm = table(knn.fit,gr)
    pr.tab[i,] = f0(cm)
}
return(pr.tab)
}

```

- (c) The following commands apply the function. Using only odd numbers for K avoids the need to break ties within the classification algorithm.

```

### (c)

k.list = seq(1,15,2)
cv.tab = knn.function(k.list,biopsy3[,1:9],biopsy3[,10])

```

- (d) The following commands create the required plots and summaries (Figure 28.3). Note that smaller values of CE , LR_- are preferred, but larger values of LR_+ are preferred. For each statistic, the optimal choices of K are 7 and 13.

```

>
> ### (d)
>
> pdf('figq4.pdf')
> par(mfrow=c(2,2))
> lbs = c('CE','LR+','LR-')
> for (i in 1:3) {plot(k.list, cv.tab[,i], type='b', xlab='K', ylab=lbs[i])}
> dev.off()
null device
1

```

```

>
> k.list[cv.tab[,1]==min(cv.tab[,1])]
[1] 7 13
> k.list[cv.tab[,2]==max(cv.tab[,2])]
[1] 7 13
> k.list[cv.tab[,3]==min(cv.tab[,3])]
[1] 7 13
>

```

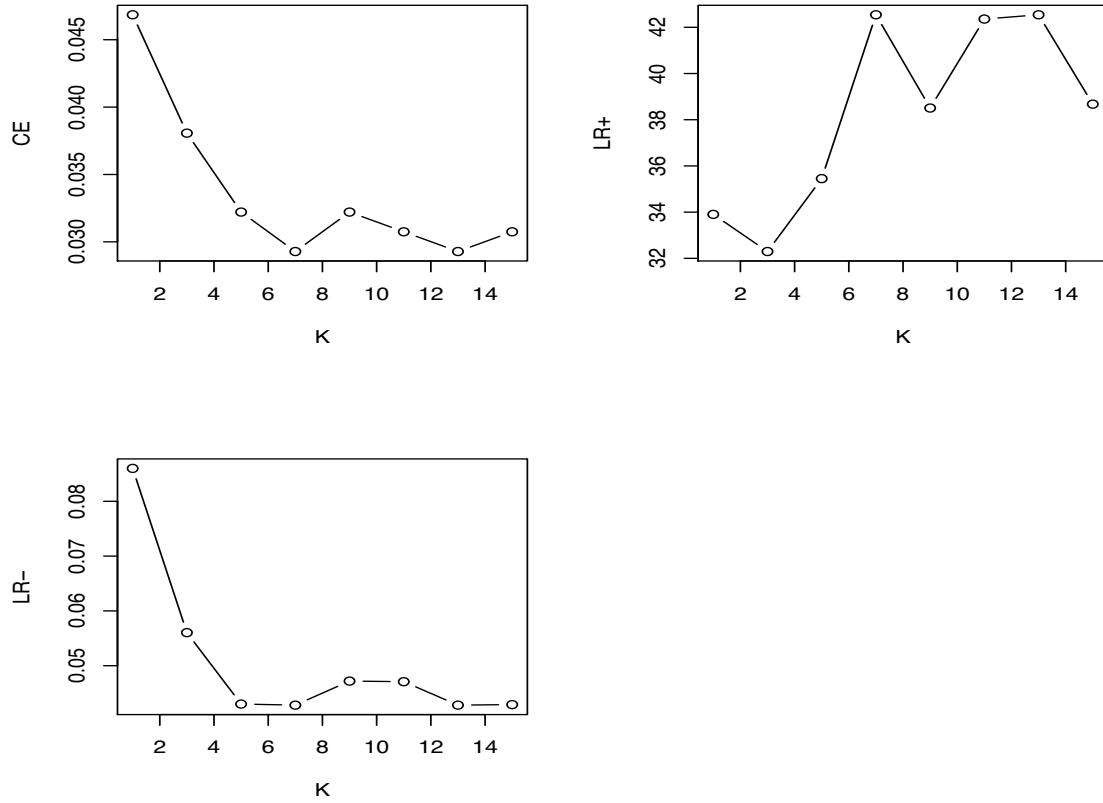


Figure 28.3: Figure for Problem 28.11 (d).

Problem 28.12. A Gaussian process is a stochastic process X_t in real time $t \geq 0$ for which any finite dimensional vector $(X_{t_1}, X_{t_2}, \dots, X_{t_m})$, $t_1 < t_2 < \dots, t_m$, possesses a multivariate normal distribution. The following two functions simulate two distinct Gaussian processes on time points $t = 1, 2, \dots, 999, 1000$.

```

### Brownian motion

f0 = function() {
  xv = rep(NA,1000)
  xv[1] = 0
  for (i in 2:1000) {
    xv[i] = xv[i-1]+rnorm(1)*0.1
  }
  return(xv)
}

### Brownian motion with negative feedback

f1 = function() {
  xv = rep(NA,1000)
  xv[1] = 0
  for (i in 2:1000) {
    xv[i] = xv[i-1]+rnorm(1)*0.1 - 0.01*xv[i-1]
  }
  return(xv)
}

```

Function `f0()` simulates the process

$$\begin{aligned} X_i &= X_{i-1} + \epsilon_i, \quad i = 2, \dots, 1000 \\ X_1 &= 0, \end{aligned}$$

while function `f1()` simulates the process

$$\begin{aligned} X_i &= X_{i-1} + \epsilon_i - 0.01X_{i-1} = 0.99X_{i-1} + \epsilon_i, \quad i = 2, \dots, 1000 \\ X_1 &= 0, \end{aligned}$$

where the ϵ_i 's are an *iid* sample from $N(0, 0.1^2)$.

- (a) Using the following code, generate a 200×1000 matrix, such that rows 1-100 and 101-200 each consist of 100 simulations of the respective Gaussian processes. Construct plots of rows 1,2,3 and 101,102,103 against indices 1, ..., 1000. Use something like `par(mfrow=c(2,3))` to draw all plots in a single window. Include the origin $X_t = 0$, and make sure the plots are labeled by series number or type of process.

```

### Simulate 100 trajectories of each type on time points 1,2,...,1000

xmat = matrix(NA,200,1000)
set.seed(12345)
for (i in 1:100) {xmat[i,] = f0()}
for (i in 101:200) {xmat[i,] = f1()}

```

- (b) We will attempt to construct a classifier able to distinguish between the two types of process. First, construct a feature matrix, using as features the observed process at a subset of time points $t = 200, 400, 600, 800, 1000$. Then create a data frame with a factor identifying the class, or process type, and the 5 extracted features $(X_{200}, \dots, X_{1000})$.
- (c) For each process derive $E[X_t]$ as a function of t .
- (d) Estimate separately for each process the 5×5 covariance and correlation matrices of the extracted features.
- (e) Use LDA to construct a classifier based on the extracted features. Use LOOCV to report a confusion matrix. Do the same with QDA.
- (f) Which classifier is more accurate? How are the answers to Part (c)-(d) related to this result?

SOLUTION:

- (a) The following code creates the required plots (Figure 28.4):

```
par(mfrow=c(2,3))
for (i in c(1:3,101:103)) {
  plot(xmat[i,],main=paste('Series ',i),xlab='i',ylab='x[i]')
  abline(h=0,col='red')
}
```

- (b) The following code creates the required data frame

```
### (b)

xmat2 = xmat[,seq(200,1000,200)]
x.data = data.frame(xmat2, gr = as.factor(rep(0:1,each=100)))
```

- (c) Function `f0()` simulates the process

$$\begin{aligned} X_i &= X_{i-1} + \epsilon_i, \quad i = 2, \dots, 1000 \\ X_1 &= 0. \end{aligned}$$

Taking the expectation of each side gives

$$E[X_i] = E[X_{i-1}] + E[\epsilon_i] = E[X_{i-1}]$$

since $E[\epsilon_i] = 0$. But $E[X_1] = 0$, so by iteration we have $E[X_t] = 0$ for all $t > 1$. The argument for the second process is identical:

$$E[X_i] = 0.99E[X_{i-1}] + E[\epsilon_i] = 0.99E[X_{i-1}]$$

so we similarly have $E[X_t] = 0$ for all $t > 1$ since $E[X_1] = 0$.

Thus for both processes, $E[X_t] = 0$ for all t .

- (d) The covariance and correlation matrices can be calculated as follows

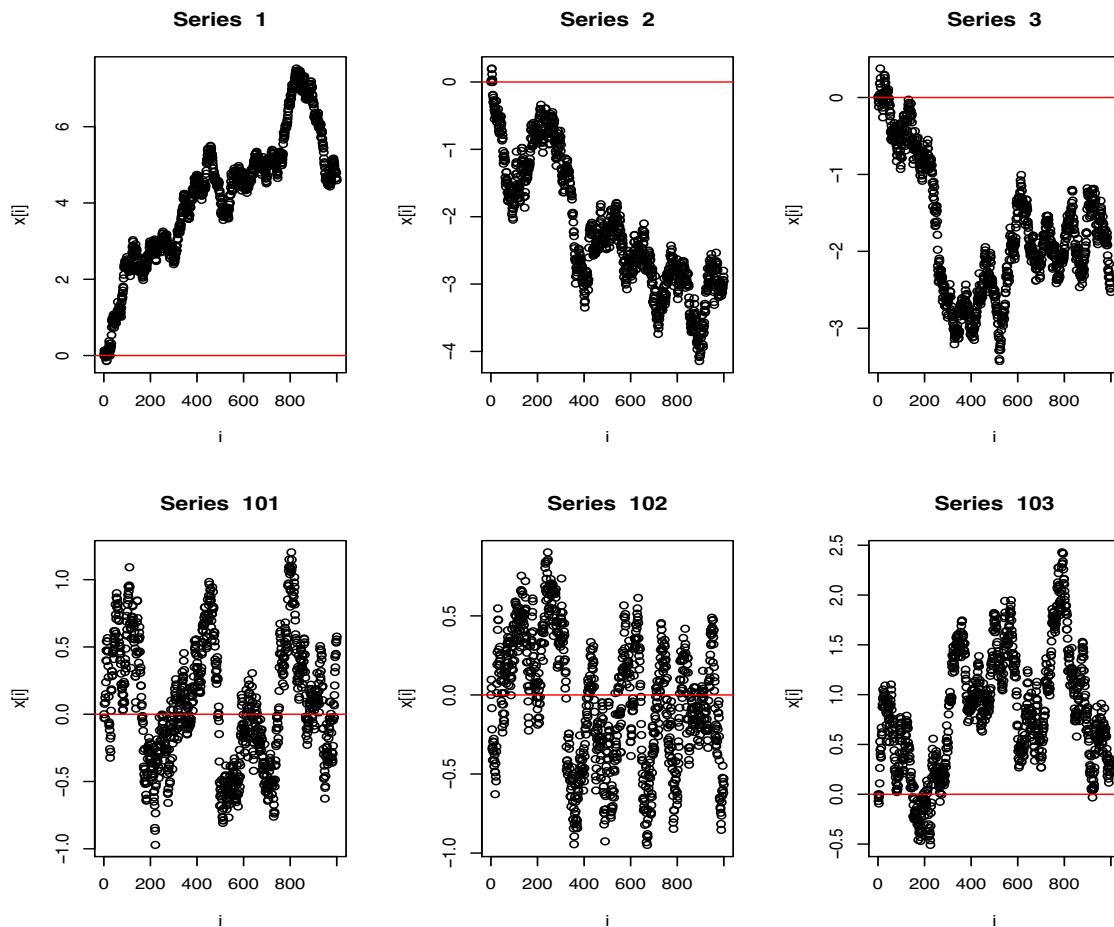


Figure 28.4: Figure for Problem 28.12 (a).

```

> ### (d)
>
> ### covariance and correlation matrix of process 1
>
> cov(xmat2[1:100,])
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 2.050972 2.541355 2.337354 2.370614 2.524570
[2,] 2.541355 5.059651 4.713319 4.832294 4.675420
[3,] 2.337354 4.713319 6.052238 5.798473 5.459291
[4,] 2.370614 4.832294 5.798473 7.291808 6.816338
[5,] 2.524570 4.675420 5.459291 6.816338 7.784094
> cor(xmat2[1:100,])
      [,1]      [,2]      [,3]      [,4]      [,5]

```

```
[1,] 1.0000000 0.7889059 0.6634170 0.6130043 0.6318348
[2,] 0.7889059 1.0000000 0.8517430 0.7955647 0.7450000
[3,] 0.6634170 0.8517430 1.0000000 0.8728467 0.7953792
[4,] 0.6130043 0.7955647 0.8728467 1.0000000 0.9047518
[5,] 0.6318348 0.7450000 0.7953792 0.9047518 1.0000000
>
>
> ### covariance and correlation matrix of process 2
>
> cov(xmat2[101:200,])
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.42521705 0.11694985 0.03711661 -0.03884944 -0.06101522
[2,] 0.11694985 0.49399796 0.17225558 0.01587851 -0.04457461
[3,] 0.03711661 0.17225558 0.40604060 0.11259178 -0.04279566
[4,] -0.03884944 0.01587851 0.11259178 0.68059012 0.05875723
[5,] -0.06101522 -0.04457461 -0.04279566 0.05875723 0.42133232
> cor(xmat2[101:200,])
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.00000000 0.25517127 0.08932607 -0.07221652 -0.14415192
[2,] 0.25517127 1.00000000 0.38461480 0.02738446 -0.09770406
[3,] 0.08932607 0.38461480 1.00000000 0.21418012 -0.10346718
[4,] -0.07221652 0.02738446 0.21418012 1.00000000 0.10972513
[5,] -0.14415192 -0.09770406 -0.10346718 0.10972513 1.00000000
>
```

(e) The LDA and QDA fits, with the respective confusion matrices can be calculated as follows

```
> ### (e)
>
> ### confusion matrix for LDA
>
> fit = lda(gr~. , data=x.data, CV=TRUE)
> table(fit$class,gr)
  gr
  0  1
 0 49 50
 1 51 50
>
> ### confusion matrix for QDA
>
> fit = qda(gr~. , data=x.data, CV=TRUE)
> table(fit$class,gr)
  gr
  0  1
 0 78  7
 1 22 93
```

>

- (f) Clearly, QDA is more accurate. In fact, LDA does not do better than chance. Recall that what distinguishes LDA and QDA is that in LDA the covariance matrix is assumed to be the same for each class. From Part (d) we can see that this is clearly not the case. The variances are noticeably larger for the first process. In addition, the features of the first process have very high positive correlation, in comparison to the second. Clearly, the assumption of equal covariance matrices made by the LDA classifier does not hold.

Because of the constant variance assumption, LDA relies on differences in means to distinguish between classes. However, in Part (c) it was shown that the feature means are the same for both classes. Thus, the unsuitability of LDA for this application goes beyond assumptions concerning variances.

Problem 28.13. This question will make use of data downloaded from GEO (Gene Expression Omnibus) identified by series accession number GSE364. From an abstract describing the study:

We analyzed the expression profiles of HCC [hepatocellular carcinoma] samples without or with intra-hepatic metastases. Using a supervised machine-learning algorithm, we generated for the first time a molecular signature that can classify metastatic HCC patients and identified genes that were relevant to metastasis and patient survival [Ye et al (2003) "Predicting hepatitis B virus-positive metastatic hepatocellular carcinomas using gene expression profiling and supervised machine learning." *Nature Medicine* (4):416–423].

Original data can be found at <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE364>

The file `GSE364n50.csv` contains 50 gene expression profiles from the GSE364 data set. Use the following command

```
gem = read.table(file='GSE364n50.csv', sep=',', row.names=1, header=F)
```

Then `gem` will be a data frame with 50 rows, each representing the gene expression profile of a single sample. There are 5913 columns, each representing the expression of a particular gene. The gene names themselves are not included here, and can simply be labelled 1 - 5913. The R expression `row.names(gem)` gives labels for the samples themselves. Otherwise, every element in `gem` is a numerical gene expression level.

Finally, two classes are represented among the 50 gene expression profiles. *Metastasis* occurs when, for example, a cancerous tumor spreads to a new site. Samples represented in rows 1-30 are from metastatic tissue, while rows 31-50 are from non-metastatic tissue. The problem considered here is to build a classifier with which tissue can be classified as metastatic or non-metastatic based on gene expression profiles (which therefore constitute the feature set). Therefore, you will have to create a factor variable which classifies each row accordingly.

- (a) Use the `prcomp()` function (from library `stats`) to create a matrix of principal components, using the gene expressions as a feature set. Use centering, but not scaling. Then construct a QDA classifier for metastasis class, using the first four principal components as features. Use the `CV=TRUE` option. Also, specify a uniform prior distribution for the classes (to do this, use option `prior=c(0.5,0.5)`).
- If do not specify the prior probabilities, what values will be used?
 - The object output from the function `qda` contains a matrix giving the posterior probability of each class (the predicted class is, of course, the one with the highest posterior probability). Create a single vector giving the maximum posterior probability p_{max} for each observation. What is the correct classification rate for observations with $p_{max} \geq 0.75$, and for observations with $p_{max} < 0.75$? Use a Wilcoxon rank-sum test to determine whether or not there is a difference in the distribution of p_{max} between observations that were correctly classified and those that weren't. Report a P -value.
 - Create a grid of pairwise scatterplots for the first four principal components, using the `pairs` function. Use colors black and red for metastasis -ve and +ve observations, respectively. In addition, use plotting character “+” (`pch=3`) for correctly classified observations, and “o” (`pch=1`) for incorrectly classified observations. In general terms, how do the correctly and incorrectly classified observations differ graphically?
- (b) We will next use cross-validation to determine the number of principal components to include in the classifier. We take the model parameter to be K if the first K principal components are used. We use two methods:
- The PCA is done first, using the entire data set. The principal components are accepted as the new feature set. Then K is varied using values $1, 2, \dots, 17, 18$. For each K , LOOCV is used to evaluate a QDA classifier accepting the first K principal components as the feature set.
 - Again, K is varied using values $1, 2, \dots, 17, 18$. For each K , LOOCV is applied. The PCA is recalculated for each new training data set considered. Whenever principal components are calculated for test data, the loadings used are those calculated using the training data only.

Obtain classification error rates for each K using both CV methods. Use the `matplot` function to plot classification error against K , placing both methods on the same graph. How do the methods compare? What would be the recommended number of principal components K ?

SOLUTION:

The following code may be used for this problem. Comments follow.

```
>
> #####
> ##### Q1
> #####
>
> library(class)
```

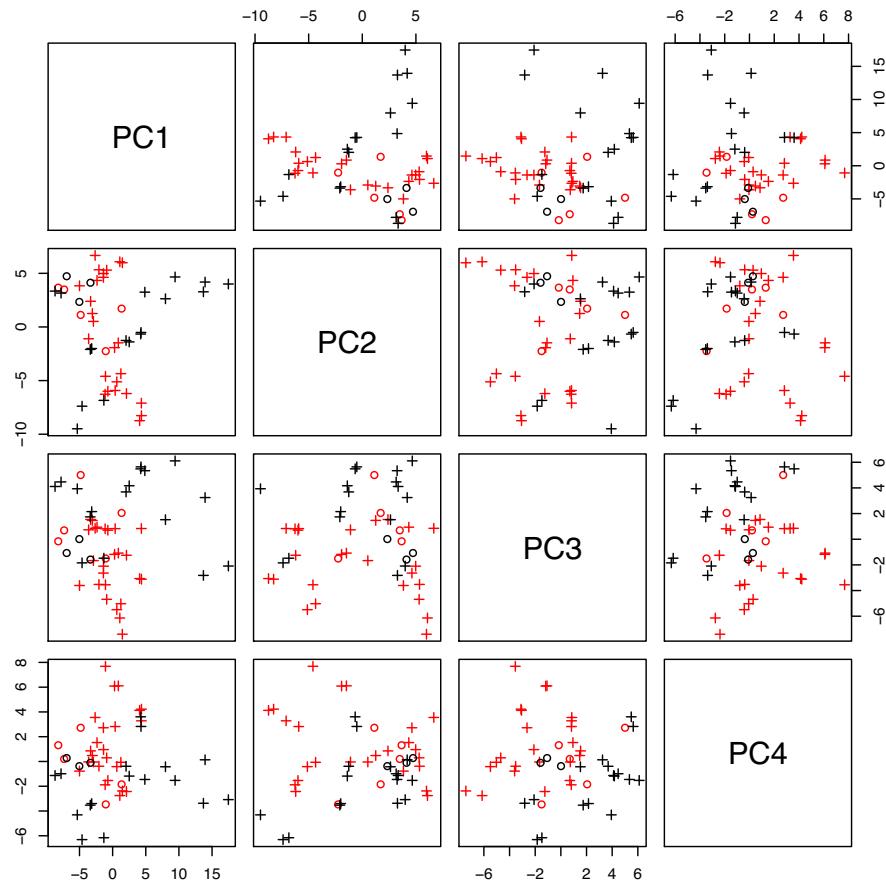


Figure 28.5: Plot for Problem 28.13 (a).

```

> library(MASS)
> library(stats)
>
> gem = read.table(file='GSE364n50.csv',sep=',',row.names=1,header=F)
> cl = c(rep('Yes',30), rep('No',20))
> cl = as.factor(cl)
>
> #### (a)
>
> # Do principal components analysis
>
> prc<-prcomp(gem, scale.=F)
>
> # Build QDA with LOOCV and uniform prior distribution
>

```

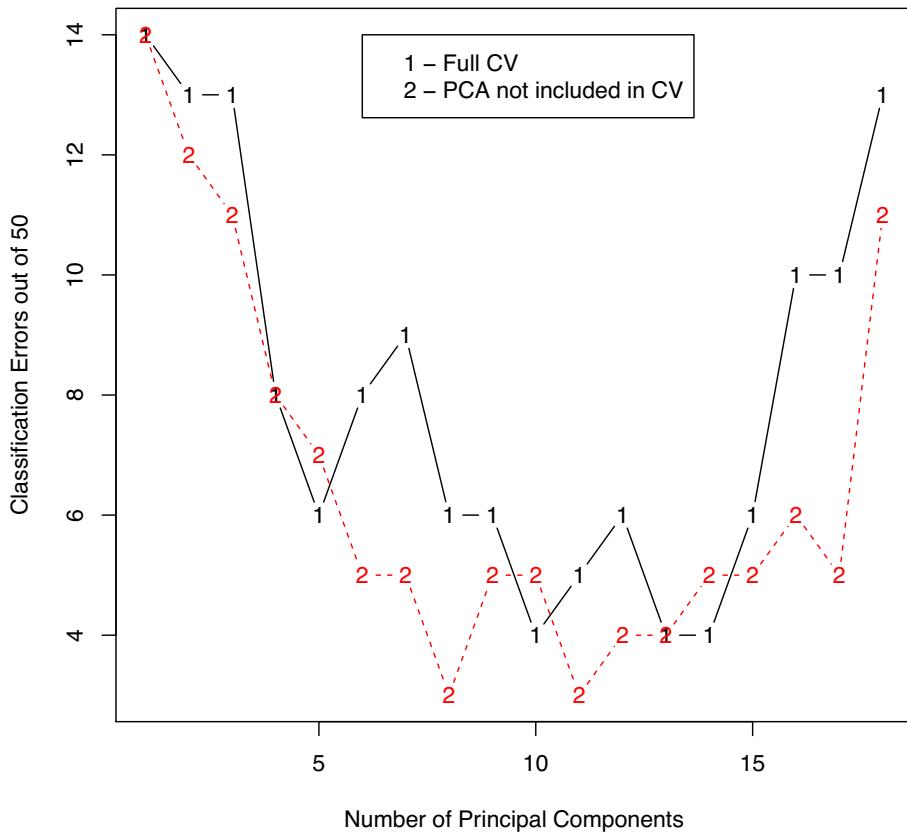


Figure 28.6: Plot for Problem 28.13 (b).

```

> nf = 4
> fit.qda = qda(x = prc$x[,1:nf], grouping = cl, CV=T, prior=c(0.5,0.5))
>
> ### (a)-(i)
>
> # Prior probabilities will be the proportions represented in the sample.
>
> table(cl)/length(cl)
cl
  No Yes
0.4 0.6
>
> ### (a)-(ii)
>
> # Posterior probabilities will be in the rows of the fit.qda$posterior matrix.

```

```

>
> post.qda = apply(fit.qda$posterior,1,max)
>
> # Obtain correct classificaiton rates for (post.qda >= 0.75)
> # and (post.qda < 0.75) groups
>
> thr = 0.75
> table(post.qda >= thr, (cl==fit.qda$class))

    FALSE  TRUE
FALSE      4     7
TRUE       4    35
>
> # Rank-sum test
>
> wilcox.test(post.qda ~ (cl==fit.qda$class))

Wilcoxon rank sum test

data: post.qda by cl == fit.qda$class
W = 76, p-value = 0.01336
alternative hypothesis: true location shift is not equal to 0

>
>
> ### (a)-(iii)
>
> # The col and pch options are used vector-wise to create points of distinct
> # shape and color
>
> pdf("figa4q1aiii.pdf")
> pairs(prc$x[,1:4],col=1+(cl=='Yes'), pch=1+2*(cl==fit.qda$class))
> dev.off()
null device
                  1
>
> ### (b)
>
> # Function first uses training data to calculate principal
> # components and build QDA classifier.
> #
> # Then the principal components for the test data are calculated. Note that the
> # PC loadings depend only on the training data.
> #
> # Finally, the QDA classifier is applied to the PCs calculated for the test data.

```

```
> #
>
> #
> # Main classifier function. Accepts training and test data
> # (other than test responses)
>
> class1 = function(x.train, y.train, x.test) {
+
+   prc = prcomp(x.train,scale.=F)
+   prc.new = predict(prc, newdata=x.test)
+   prcx = prc$x[,1:nf]
+   if (!is.matrix(prcx)) {prcx = matrix(prcx,ncol=1)}
+   fit.qda = qda(x = prcx, grouping = y.train, prior=c(0.5,0.5))
+   return(predict(fit.qda,newdata=prc.new[1:nf])$class)
+
+ }
>
> #
> # Set up main CV loop
> #
>
> x = gem
> y = cl
> prc.all<-prcomp(x, scale.=F)
> ce1 = NULL
> ce2 = NULL
>
> #
> # Evaluate classifier based on 1 through 18 principal components
> #
>
> for (nf in c(1:18)) {
+
+   # PCA included in CV loop
+
+   pred.cv1 = rep(NA,50)
+   for (i in 1:50) {pred.cv1[i] = class1(x[-i,],y[-i],x[i,])}
+   mm = table(pred.cv1,y)
+   ce1 = c(ce1, 50 - sum(diag(mm)))
+
+   # PCA not included in CV loop. CV implemented via CV=T option in qda()
+
+   prcx = prc.all$x[,1:nf]
+   if (!is.matrix(prcx)) {prcx = matrix(prcx,ncol=1)}
+   fit.qda = qda(x = prcx, grouping = cl, CV=T, prior=c(0.5,0.5))
```

```

+
+   mm = table(cl,fit.qda$class)
+   ce2 = c(ce2, 50 - sum(diag(mm)))
+ }
>
> # Draw plots
>
> pdf("figa4q1b.pdf")
> matplot(1:18, cbind(ce1,ce2),type='b',
+   xlab="Number of Principal Components",ylab="Classification Errors out of 50")
> legend(6,14,legend=c("1 - Full CV", "2 - PCA not included in CV"))
> dev.off()

```

(a) See code above.

- (i) By default, qda uses as the prior the class proportions found in the data. Here, from the output above, that would give prior probabilities $\pi_{Yes} = 0.6 = 1 - \pi_{No}$.
- (ii) The relevant output from above is

```

> thr = 0.75
> table(post.qda >= thr, (cl==fit.qda$class))

      FALSE  TRUE
FALSE      4     7
TRUE       4    35
>

```

For $p_{max} \geq 0.75$ the correct classification rate is $35/39 \approx 0.897$. For $p_{max} < 0.75$ the correct classification rate is $7/11 \approx 0.636$. For the rank-sum test we have $p\text{-value} = 0.01336$, which suggests that the distributions of p_{max} differ by classification correctness.

- (iii) See Figure 28.5. The incorrectly classified observations tend to be concentrated in the interior of clusters. In contrast, observations located near the minima and or maxima of one or more principal components tends to be correctly classified.
- (b) See Figure 28.6. Both methods give similar estimates of classification error as a function of K , but the error estimates are generally larger when the PCA is included in the cross-validation procedure. Since the PCA is formally part of the classifier, this is appropriate. Using the full CV curve, the minimum estimated error is attained for $K = 10, 13, 14$. We can accept the smallest value, $K = 10$.

28.3 Theoretical Complements

Problem 28.14. One question to consider when constructing classifiers is whether or not the method is *location-scale invariant*. This property holds if subjecting any subset of features in the training data to a linear transformation cannot change the prediction. For example, a classifier

is location-scale invariant if changing the units of a feature from feet to inches, or from degrees fahrenheit to celsius, does not change the predictions (as long as the transformation is consistently applied to all classes and test data).

This is a crucial question, since if a classifier is not location-scale invariant, we need to decide how to make the units of each feature comparable. One way of doing this is to standardize quantitative features, typically by subtracting the mean then dividing by the standard deviation, so that each feature has mean zero and standard deviation 1. Doing this makes them essentially unitless. However, for some applications this may not be a suitable approach to this problem.

For this problem use data set `fgl` from the `MASS` package. This is a forensic application. The observations consist of fragments of broken glass. The `type` column gives the type of glass. The `RI` column gives refractive index (but see description from `help(fgl)`). The remaining 8 columns are percentages by weight of oxides. The row totals of these 8 columns are approximately 100%.

The object is to build a classifier which predicts glass type from the measured `RI` and chemical composition of a glass fragment. We will consider three questions: (i) What form of classifier should be used? (ii) Which features should be included? (iii) How should the features be standardized (if this is needed)?

- (a) For this exercise we will only consider 4 glass types: window float glass (`WinF`), window non-float glass (`WinNF`), vehicle window glass (`Veh`), and vehicle headlamps (`Head`). Create a subset of the data accordingly.
- (b) We will consider three types of classifiers: KNN, LDA and QDA. Prove that QDA (and therefore LDA) is location-scale invariant. To do this, suppose a p -dimensional feature vector \dot{x} is transformed to

$$\dot{y} = A\dot{x} + b$$

where \dot{x} and \dot{y} are interpreted as $p \times 1$ column vectors, A is an invertible $p \times p$ matrix, and b is a $p \times 1$ column vector. Then each class-dependent mean vector and covariance matrix must be transformed accordingly. An important observation is that neither A nor b depend on the class.

- (c) Is the KNN classifier, assuming Euclidean distance is used, *location-scale invariant*? Why or why not?
- (d) For any classifier that is not location-scale invariant, we will use the following strategy. We are given an n -dimensional class vector \mathbf{Y} and an $n \times 9$ feature matrix \mathbf{X} . First, standardize only the `RI` column to zero mean and unit variance. Then multiply the standardized `RI` column by a selected scale value α . Use the resulting feature matrix for the classification, and capture the classification error CE . Vary α , selecting the value yielding the minimum CE . Then, write a main function which accepts a class vector and a data set of features, and which applies 3 classification methods to the data, giving an estimated CE for each:
 - (i) LDA, CE estimated using LOO cross-validation;
 - (ii) QDA, CE estimated using LOO cross-validation;
 - (iii) KNN, allowing K to vary over $1, 2, \dots, 25$. For each model, estimate CE using LOO cross-validation. Select K yielding the minimum CE . In the case of ties, select the smallest K yielding the minimum CE .

Make sure the scale procedure just described is used for any classifier which is not location-scale invariant. Use α values generated by `RI.scale = c(1:150)/10`. The function should

return enough information to identify the classifier with the smallest CE . The minimum output would be the minimum CE and the associated classifier, including the parameter K if that classifier is KNN, and any relevant scale factor α . Of course, more information can be returned.

- (e) The main function will be used within the following heuristic, which sequentially removes features from the initial feature set $B = (1, 2, \dots, 9)$ (ie the full feature set).

```

Blist is a list of Index Subsets
Blist = NULL

CElist is a list of Classification Errors
CElist = NULL

CEarray is a numerical array with elements addressed by index

# Initial feature set

B = (1,...,9)
CE = minimum possible CE using B

# Store classification results

Blist = append(Blist,B)
CElist = append(CElist,CE)

while length(B) > 1 {

    For each index I in B {

        # Remove index I from B, copy into B'

        B' = B - {I}
        CEarray[I] = minimum possible CE using B'
    }

    I* = Index from B which minimizes CEarray[I]

    # Update and store current feature set

    B = B - {I*}
    Blist = append(Blist,B)
    CElist = append(CElist,CEarray[I*])

}

```

Then select the feature set from `Blist` with the smallest CE (stored in `CElist`).

- (f) Identify the exact classifier selected by the algorithm, including K and α , if needed. Report CE , and construct a confusion matrix.
- (g) Use the `pairs` function to construct all pairwise scatterplots among the selected features (RI need not be scaled, if selected). Use symbols and/or colors to distinguish the four classes. Also, indicate which predictions are correct (for example, use colors to distinguish the classes, and symbols to indicate the prediction success). What does this plot suggest regarding why the classification method selected by the heuristic might be preferred?

SOLUTION:

The required script follows. The plot is given in Figure 28.7.

```
> library(MASS)
> library(class)
>
> #### (a)
>
> table(gr)
gr
  Head   Veh  WinF WinNF
  29     17    70    76
> fglex = subset(fgl, type %in% c('WinF', 'WinNF', 'Head', 'Veh'))
> gem1 = fglex[,1:9]
> gr = as.character(fglex$type)
>
> #### (d)
>
> main.function = function(gem1, gr, alpha.list=NULL) {
+
+   #### vector to contain CE
+
+   crate = rep(0,3)
+
+   #### LDA-CV
+
+   fit.ldacv = lda(gem1, gr, CV=T)
+   pr.ldacv = fit.ldacv$class
+   crate[1] = 1 - mean(pr.ldacv==gr)
+
+   #### QDA-CV
+
+   fit.qdacv = qda(gem1, gr, CV=T)
+   pr.qdacv = fit.qdacv$class
+   crate[2] = 1 - mean(pr.qdacv==gr)
+
+   #### KNN
```

```
+  
+   k.list = (1:25)  
+  
+   if (is.null(alpha.list)) {  
+  
+     # list of K values  
+  
+     nk = length(k.list)  
+  
+     # error vector  
+     knn.err = integer(nk)  
+  
+     # loop through K values  
+     for (i in 1:nk) {  
+       fit.knn = knn.cv(gem1,gr,k=k.list[i],prob=T)  
+       pr.knn = fit.knn  
+       knn.err[i] = 1-mean(pr.knn==gr)  
+     }  
+  
+     # capture optimal K  
+     k.best = c(min(k.list[knn.err==min(knn.err)]),NA)  
+  
+     # refit with opimal K  
+     fit.knn = knn.cv(gem1,gr,k=k.best[1],prob=T)  
+     pr.knn = fit.knn  
+     crate[3] = 1 - mean(pr.knn==gr)  
+  
+   } else {  
+  
+     # list of K values  
+  
+     nk = length(k.list)  
+     nalpha = length(alpha.list)  
+  
+     # error vector  
+     knn.err = matrix(NA, nalpha, nk)  
+  
+     # loop through K values  
+  
+     for (i in 1:nalpha) {  
+       gem2 = gem1  
+       gem2[,1] = alpha.list[i]*(gem1[,1] - mean(gem1[,1]))/sd(gem1[,1])  
+       for (j in 1:nk) {  
+         fit.knn = knn.cv(gem2,gr,k=k.list[j],prob=T)  
+         pr.knn = fit.knn
```

```
+      knn.err[i,j] = 1-mean(pr.knn==gr)
+    }
+
+  # capture optimal K
+
+  ki = which(knn.err==min(knn.err),arr.ind=T)
+  if (is.matrix(ki)) {ki = ki[1,]}
+  k.best = c(k.list[ki[2]], alpha.list[ki[1]])
+
+  # refit with optimal parameters
+
+  gem2 = gem1
+  gem2[,1] = k.best[2]*(gem1[,1] - mean(gem1[,1]))/sd(gem1[,1])
+
+  fit.knn = knn.cv(gem2,gr,k=k.best[1],prob=T)
+  pr.knn = fit.knn
+  crate[3] = 1 - mean(pr.knn==gr)
+
+ }
+
+
+ # return error rates, and data from KNN algorithm
+ return(list(crate=crate, knn.err=knn.err, k.best=k.best))
+
+ }
```

>

>

> ##### (e)

>

```
> RI.scale = c(1:150)/10
>
>
> crate.table = NULL
> vsel.list = list()
>
> vsel = 1:9
> cerr = main.function(gem1,gr,RI.scale)$crate
> crate.table=rbind(crate.table,cerr)
> vsel.list[[1]] = vsel
>
> for (iii in 1:8) {
+
+   cerr.temp = NULL
+   for (jjj in 1:length(vsel)) {
```

```

+     if ( (1 %in% vsel[-jjj]) & (length(vsel[-jjj])>1) )   {
+         alpha.list = RI.scale
+     } else {
+         alpha.list = NULL
+     }
+     cerr = main.function(as.matrix(gem1[,vsel[-jjj]]),gr,alpha.list)$crate
+     cerr.temp = rbind(cerr.temp,cerr)
+ }
+ cerr.min = apply(cerr.temp,1,min)
+ jjj.min = which.min(cerr.min)
+ vsel = vsel[-jjj.min]
+ crate.table = rbind(crate.table,cerr.temp[jjj.min,])
+ vsel.list[[iii+1]] = vsel
+ }
> crate.table
      [,1]      [,2]      [,3]
cerr 0.3385417 0.4062500 0.1562500
      0.3958333 0.4166667 0.1510417
      0.3854167 0.4218750 0.1354167
      0.3854167 0.3593750 0.1354167
      0.3854167 0.3802083 0.1354167
      0.3489583 0.4062500 0.1614583
      0.3645833 0.4479167 0.1822917
      0.3697917 0.4479167 0.2083333
      0.6458333 0.6093750 0.3697917
>
> ### (f)
>
> # Get best CE for each feature set
>
> ce.row.best = apply(crate.table,1,min)
>
> # Which row contains the minimum CE (pick the smallest feature set
> #      to break ties)?
>
> best.row = max(which(ce.row.best==min(ce.row.best)))
>
> # KNN is the best classifier for the best feature set
>
> crate.table[best.row,]
[1] 0.3854167 0.3802083 0.1354167
>
> # Identify the feature set
>
> best.feature.set = vsel.list[[best.row]]

```

```

> colnames(gem1)[best.feature.set]
[1] "RI"  "Mg"  "K"   "Ca"  "Fe"
>
> # Redo classifier selection
>
> class.obj = main.function(gem1[,best.feature.set],gr,RI.scale)
>
> ### For best classifier, K = 1, alpha = 1.3
>
> k.best = class.obj$k.best
> k.best
[1] 1.0 1.3
>
> ### Redo classification
>
> gem2 = gem1
> gem2[,1] = k.best[2]*(gem1[,1] - mean(gem1[,1]))/sd(gem1[,1])
> fit.knn = knn.cv(gem2[,best.feature.set],gr,k=k.best[1],prob=T)
>
> ### CE matches best CE reported
>
> 1-mean(fit.knn==gr)
[1] 0.1354167
>
> ### Confusion matrix
>
> table(gr,fit.knn)
      fit.knn
gr      Head Veh WinF WinNF
  Head     24    0     0     5
  Veh      0   13     4     0
  WinF     0     3    65     2
  WinNF    1     3     8    64
>
> par(mfrow=c(1,1))
> pairs(gem1[,best.feature.set], panel = function(x,y) {
+   points(x,y,pch=1+2*(fit.knn==gr),
+          col=c("red","green","blue","orange")[unclass(as.factor(gr))],cex=0.5)
+   legend('topright',legend=c(attr(as.factor(gr),"levels"),"Correct",'Incorrect'),
+          col=c("red","green","blue","orange","gray","gray"),
+          pch=c(3,3,3,3,3,1),cex=0.35,bty='n')
+ })

```

(a) See above.

(b) For QDA the class j discriminating function for $\dot{x} \sim N(\boldsymbol{\mu}_j, \Sigma_j)$ is

$$h_j(\dot{x}) = -\frac{1}{2}Q_j(\dot{x}) - \frac{1}{2}\log(\det(\Sigma_j)) + \log(\pi_j)$$

where

$$Q_j(\dot{x}) = (\dot{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\dot{x} - \boldsymbol{\mu}_j).$$

Then, if

$$\dot{y} = A\dot{x} + b$$

we have $\dot{y} \sim N(\boldsymbol{\mu}'_j, \Sigma'_j)$ where

$$\boldsymbol{\mu}'_j = A\boldsymbol{\mu}_j + b, \quad \Sigma'_j = A\Sigma_j A^T.$$

Then the discriminating function for \dot{y} is

$$\begin{aligned} h'_j(\dot{y}) &= -\frac{1}{2}Q'_j(\dot{y}) - \frac{1}{2}\log(\det(\Sigma'_j)) + \log(\pi_j) \\ &= -\frac{1}{2}Q'_j(\dot{y}) - \frac{1}{2}\log(\det(A\Sigma_j A^T)) + \log(\pi_j) \\ &= -\frac{1}{2}Q'_j(\dot{y}) - \frac{1}{2}\log(\det(\Sigma_j)) - \log(\det(A)) + \log(\pi_j) \end{aligned}$$

where

$$\begin{aligned} Q'_j(\dot{y}) &= (\dot{y} - \boldsymbol{\mu}'_j)^T [\Sigma'_j]^{-1} (\dot{y} - \boldsymbol{\mu}'_j) \\ &= (A\dot{x} + b - A\boldsymbol{\mu}_j - b)^T (A\Sigma_j A^T)^{-1} (A\dot{x} + b - A\boldsymbol{\mu}_j - b) \\ &= (\dot{x} - \boldsymbol{\mu}_j)^T A^T [A^T]^{-1} \Sigma_j^{-1} A (\dot{x} - \boldsymbol{\mu}_j) \\ &= (\dot{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\dot{x} - \boldsymbol{\mu}_j) \\ &= Q_j(\dot{x}), \end{aligned}$$

where we rely on identities $(AB)^T = B^T A^T$, $\det(AB) = \det(A)\det(B)$, $\det(A) = \det(A^T)$, $(AB)^{-1} = B^{-1}A^{-1}$ for two square invertible matrices A, B . Assembling everything gives

$$h'_j(\dot{y}) = h_j(\dot{x}) - \log(\det(A)).$$

The result follows after noting that the difference between $h'_j(\dot{y})$ and $h_j(\dot{x})$ does not depend on j .

- (c) The KNN algorithm depends on a distance matrix between observations. Suppose one feature is multiplied by a factor α , which is allowed to increase indefinitely. Then the classification will eventually be dominated by that feature. Applying this transformation to a different feature may result in a distinct classification. The classifier is therefore not location-scale invariant.
- (d) See above.
- (e) See above.
- (f) From above, the feature set selected is $(\text{RI}, \text{Mg}, \text{K}, \text{Ca}, \text{Fe})$. The best classifier is KNN, $K = 1$, $\alpha = 1.3$, with $CE = 0.1354167$ (the smallest feature set is selected in case of ties). The confusion matrix, from above, is:

```
fit.knn
gr      Head Veh WinF WinNF
Head     24   0    0    5
Veh      0   13    4    0
WinF     0    3   65    2
WinNF    1    3    8   64
```

- (g) LDA or QDA assumes that the features possess a multivariate normal density. This is appropriate when features within a class cluster about a single point in a stable way. Examining Figure 28.7, this clearly does not occur. KNN, on the other hand, makes no such assumption. For example, we can see that some glass fragments of class `type == WinNF` have 0% Mg, while others have > 0% Mg. KNN is able to incorporate this structure into the prediction.

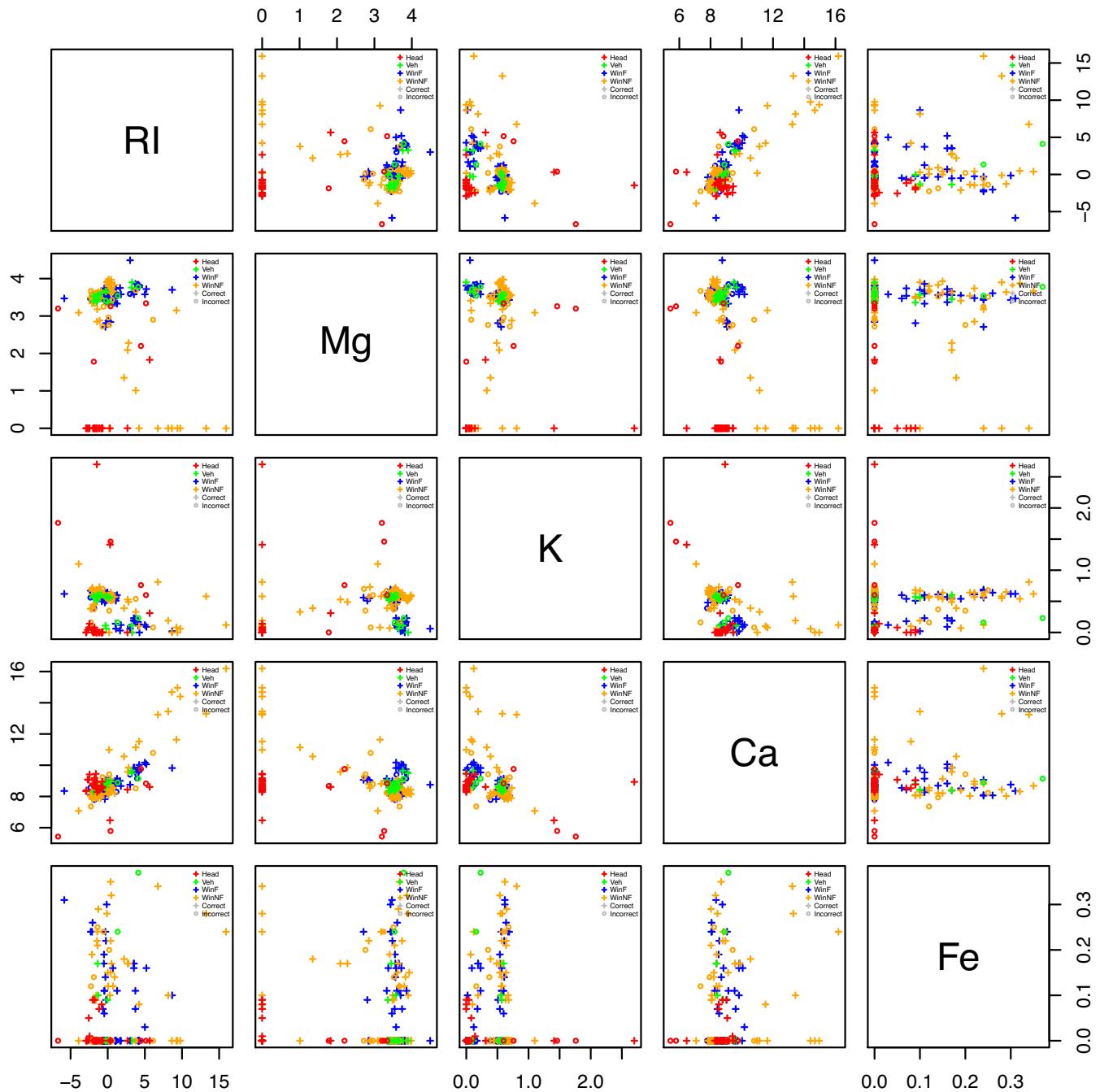


Figure 28.7: Plots for Problem 28.14.

Chapter 29

Practice Problems - Unsupervised Learning

29.1 Exercises

Problem 29.1. Suppose we have $n = 5$ observations of a feature vector. The distances between observations i and j , denoted d_{ij} , are given in the following distance matrix:

	1	2	3	4	5
1	0.000	8.853	9.022	9.540	10.982
2	8.853	0.000	7.803	9.537	10.753
3	9.022	7.803	0.000	9.377	10.562
4	9.540	9.537	9.377	0.000	9.957
5	10.982	10.753	10.562	9.957	0.000

Using the compact agglomeration method, for which cluster distance is defined by

$$D(A, B) = \max_{i \in A, j \in B} d_{ij}$$

for any two clusters A, B , construct a hierarchical cluster for this data. Justify each step precisely. Sketch a dendrogram, indicating precisely the height of each node.

SOLUTION:

The compact distance between two clusters A and B is

$$D(A, B) = \max_{i \in A, j \in B} d_{ij}.$$

To construct the clustering, we use the following steps:

1. Start with clusters $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$.

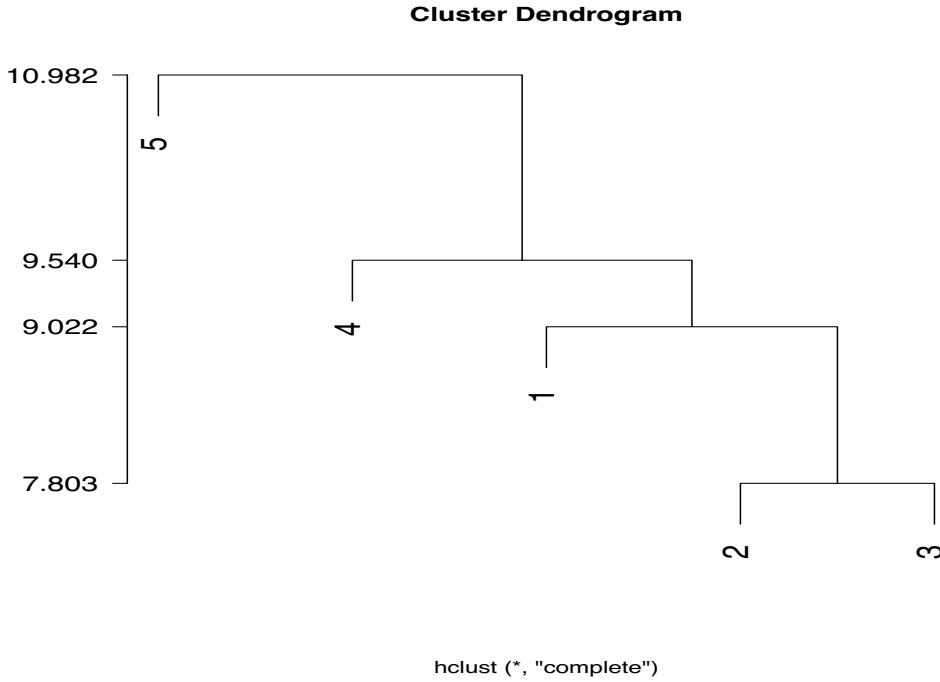


Figure 29.1: Dendrogram for Problem 29.1.

2. First join the two nearest observations, which are 2 and 3 ($d_{2,3} = 7.803$). This gives clusters $\{1\}$, $\{4\}$, $\{5\}$ and $\{2, 3\}$ joined at distance 7.80.
3. The cluster distances are now

$$\begin{aligned}
 D(\{1\}, \{4\}) &= d_{1,4} = 9.540, \\
 D(\{1\}, \{5\}) &= d_{1,5} = 10.982, \\
 D(\{4\}, \{5\}) &= d_{4,5} = 9.957, \\
 D(\{1\}, \{2, 3\}) &= \max\{d_{1,2}, d_{1,3}\} = \max\{8.853, 9.022\} = 9.022, \\
 D(\{4\}, \{2, 3\}) &= \max\{d_{4,2}, d_{4,3}\} = \max\{9.540, 9.377\} = 9.540, \\
 D(\{5\}, \{2, 3\}) &= \max\{d_{5,2}, d_{5,3}\} = \max\{10.753, 10.562\} = 10.753.
 \end{aligned}$$

The smallest cluster distance is $D(\{1\}, \{2, 3\}) = 9.022$, so combine clusters $\{1\}$ and $\{2, 3\}$. This gives clusters $\{1, 2, 3\}$, $\{4\}$ and $\{5\}$, joined at distance 9.022.

4. The cluster distances are now

$$\begin{aligned}
 D(\{1, 2, 3\}, \{4\}) &= \max\{d_{1,4}, d_{2,4}, d_{3,4}\} = \max\{9.540, 9.537, 9.377\} = 9.540, \\
 D(\{1, 2, 3\}, \{5\}) &= \max\{d_{1,5}, d_{2,5}, d_{3,5}\} = \max\{10.982, 10.753, 10.562\} = 10.982, \\
 D(\{4\}, \{5\}) &= \max\{d_{4,5}\} = \max\{9.96\} = 9.96.
 \end{aligned}$$

The smallest cluster distance is $D(\{1, 2, 3\}, \{4\}) = 9.540$, so combine clusters $\{1, 2, 3\}$ and $\{4\}$. This gives clusters $\{1, 2, 3, 4\}$ and $\{5\}$, joined at distance 9.540.

5. Join the remaining two clusters, at cluster distance

$$D(\{1, 2, 3, 4\}, \{5\}) = \max\{d_{1,5}, d_{2,5}, d_{3,5}, d_{4,5}\} = \max\{10.982, 10.753, 10.562, 9.957\} = 10.982.$$

This gives the dendrogram shown in Figure 29.1.

Problem 29.2. Suppose in an unsupervised learning application we are given observations $\dot{x}_1, \dots, \dot{x}_n$. Recall the *within cluster sum of squares*, for K clusters A_1, \dots, A_K where d is a distance function and $g(A_i)$ is a cluster centroid:

$$SS_{within} = \sum_{i=1}^K \sum_{j \in A_i} d(\dot{x}_j, g(A_i))^2.$$

A K -means clustering algorithm was applied to the data, allowing the number of clusters K to vary from 1 to 6. The following table gives the separate sum of squares within each cluster:

	1	2	3	4	5	6
1	38608.0	-	-	-	-	-
2	258.3	4911.1	-	-	-	-
3	501.9	258.3	218.6	-	-	-
4	191.6	112.6	94.8	258.3	-	-
5	53.5	42.1	94.8	191.6	112.6	-
6	42.1	77.3	40.9	53.5	112.6	38.8

Let R^2 be the proportion of total variation explained by the clustering. If we accept as the number of clusters the smallest value of K for which $R^2 \geq 95\%$, what is this number?

SOLUTION:

The total sum of squares SS_{total} is simply the SS for the $K = 1$ model, so

$$SS_{total} = 38608.0.$$

Otherwise, SS_{within} is the sum of the individual cluster sums of squares. Then

$$R^2 = 1 - \frac{SS_{within}}{SS_{total}}.$$

This gives, for $K = 1, 2, 3$:

$$\begin{aligned} R^2[1] &= 1 - \frac{SS_{total}}{SS_{total}} = 0, \\ R^2[2] &= 1 - \frac{258.3 + 4911.1}{38608.0} = 0.866, \\ R^2[3] &= 1 - \frac{501.9 + 258.3 + 218.6}{38608.0} = 0.975. \end{aligned}$$

The smallest number of clusters that yield at least 95% variation explained is $K = 3$.

Problem 29.3. A principal components analysis was performed on 4 measured psychometric scales: *Happiness*, *Joy*, *Mirth* and *Contentment*. The loadings on the PCs are given in the following table:

	PC1	PC2	PC3	PC4
<i>Happiness</i>	-0.58	0.07	-0.56	-0.58
<i>Joy</i>	-0.57	0.01	-0.23	0.79
<i>Mirth</i>	-0.58	0.01	0.79	-0.19
<i>Contentment</i>	0.05	1.00	0.03	0.03

The scree plot for the principal components is shown in Figure 29.2. What conclusion can be reached from the loadings and the scree plot regarding the issue of dimension reduction?

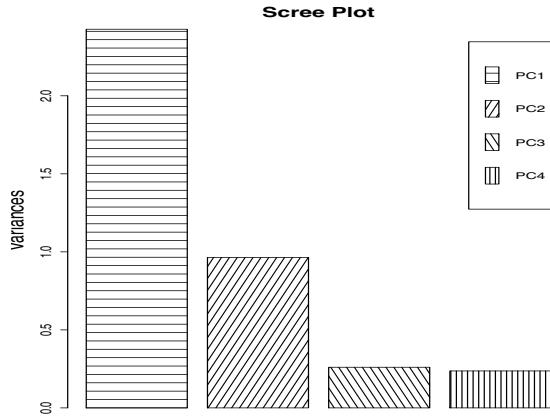


Figure 29.2: Scree plot for Problem 29.3.

SOLUTION:

From the scree plot, most of the variance is explained by the first two principal components. The loadings on the first PC are nearly equal for *Happiness*, *Joy* and *Mirth*, and relatively smaller for *Contentment*. The loadings on the second PC are concentrated on *Contentment*. We conclude that *Happiness*, *Joy* and *Mirth* are highly correlated, and together form a single dimension, while *Contentment* forms a second independent dimension.

Problem 29.4. Suppose we are given an $n \times 3$ matrix \mathbf{X} , with columns defining 3 standardized predictors x_1, x_2, x_3 . The three principal components are then calculated, and given in form

$$PC_j = a_{1j}x_1 + a_{2j}x_2 + a_{3j}x_3, \quad j = 1, 2, 3.$$

Suppose the matrix of variable loadings a_{ij} is given, in part, by

$$A = \begin{bmatrix} 1/2 & a_{12} & a_{13} \\ 1/2 & a_{22} & a_{23} \\ a_{31} & 1/\sqrt{2} & a_{33} \end{bmatrix}$$

Determine all values of the variable loadings a_{ij} left unspecified. For convenience, you may assume $a_{31} > 0$ and $a_{13} > 0$.

SOLUTION:

1st PC: The sum of squares of each column of A equals 1. Therefore

$$a_{31}^2 = 1 - (1/2)^2 - (1/2)^2 = 1/2.$$

Since $a_{31} > 0$ we must have $a_{31} = 1/\sqrt{2}$.

2nd PC: The columns of A are orthogonal. This means

$$a_{12}/2 + a_{22}/2 + 1/2 = 0.$$

In addition,

$$a_{12}^2 + a_{22}^2 + 1/2 = 1.$$

Substitution gives

$$(1 + a_{22})^2 + a_{22}^2 + 1/2 = 1,$$

or equivalently,

$$2a_{22}^2 + 2a_{22} + 1/2 = 2(a_{22} + 1/2)^2 = 0.$$

The unique solution is $a_{22} = -1/2$. Then substituting gives $a_{12} = -1/2$.

3rd PC: The columns of A are mutually orthogonal. This means

$$\begin{aligned} a_{13}/2 + a_{23}/2 + a_{33}/\sqrt{2} &= 0 \\ -a_{13}/2 + -a_{23}/2 + a_{33}/\sqrt{2} &= 0. \end{aligned}$$

Adding the equations gives $2a_{33}/\sqrt{2} = 0$, or $a_{33} = 0$. This then implies $a_{13} = -a_{23}$. If $a_{13} > 0$, and the sum of squares of each column equals, we must then have $a_{13} = -a_{23} = 1/\sqrt{2}$.

To summarize, we then have

$$A = \begin{bmatrix} 1/2 & -1/2 & 1/\sqrt{2} \\ 1/2 & -1/2 & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix}$$

Problem 29.5. Suppose we have $n = 5$ observations of features, for which the following distance matrix is calculated:

Using the single link agglomeration method, construct a hierarchical cluster for this data. Sketch a dendrogram, indicating precisely the height of each node.

	1	2	3	4	5
1	0.00	14.29	15.43	17.25	16.45
2	14.29	0.00	12.36	17.50	15.52
3	15.43	12.36	0.00	18.13	16.22
4	17.25	17.50	18.13	0.00	13.84
5	16.45	15.52	16.22	13.84	0.00

SOLUTION:

The single link distance between two clusters A and B is

$$D(A, B) = \min_{i \in A, j \in B} d_{ij}.$$

To construct the clustering, we use the following steps:

- (i) Start with clusters $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$.
- (ii) First join the two nearest observations, which are 2 and 3 ($d_{2,3} = 12.36$). This gives clusters $\{1\}, \{4\}, \{5\}$ and $\{2, 3\}$ joined at distance 12.36.
- (iii) The cluster distances are now

$$\begin{aligned} D(\{1\}, \{4\}) &= d_{1,4} = 17.25, \\ D(\{1\}, \{5\}) &= d_{1,5} = 16.45, \\ D(\{4\}, \{5\}) &= d_{4,5} = 13.84, \\ D(\{1\}, \{2, 3\}) &= \min\{d_{1,2}, d_{1,3}\} = \min\{14.29, 15.43\} = 14.29, \\ D(\{4\}, \{2, 3\}) &= \min\{d_{4,2}, d_{4,3}\} = \min\{17.50, 18.13\} = 17.50, \\ D(\{5\}, \{2, 3\}) &= \min\{d_{5,2}, d_{5,3}\} = \min\{15.52, 16.22\} = 15.52. \end{aligned}$$

The smallest cluster distance is $D(\{4\}, \{5\}) = 13.84$, so combine clusters $\{4\}$ and $\{5\}$. This gives clusters $\{1\}, \{2, 3\}$ and $\{4, 5\}$, joined at distance 13.84.

- (iv) The cluster distances are now

$$\begin{aligned} D(\{1\}, \{2, 3\}) &= \min\{d_{1,2}, d_{1,3}\} = \min\{14.29, 15.43\} = 14.29, \\ D(\{1\}, \{4, 5\}) &= \min\{d_{1,4}, d_{1,5}\} = \min\{17.25, 16.45\} = 16.45, \\ D(\{2, 3\}, \{4, 5\}) &= \min\{d_{2,4}, d_{2,5}, d_{3,4}, d_{3,5}\} = \min\{17.50, 15.52, 18.13, 16.22\} = 15.52. \end{aligned}$$

The smallest cluster distance is $D(\{1\}, \{2, 3\}) = 14.29$, so combine clusters $\{1\}$ and $\{2, 3\}$. This gives clusters $\{1, 2, 3\}$ and $\{4, 5\}$, joined at distance 14.29.

- (v) Join the remaining two clusters, at cluster distance

$$\begin{aligned} D(\{1, 2, 3\}, \{4, 5\}) &= \min\{d_{1,4}, d_{1,5}, d_{2,4}, d_{2,5}, d_{3,4}, d_{3,5}\} \\ &= \{17.25, 16.45, 17.50, 15.52, 18.13, 16.22\} \\ &= 15.52. \end{aligned}$$

This gives the dendrogram shown in Figure 29.3.

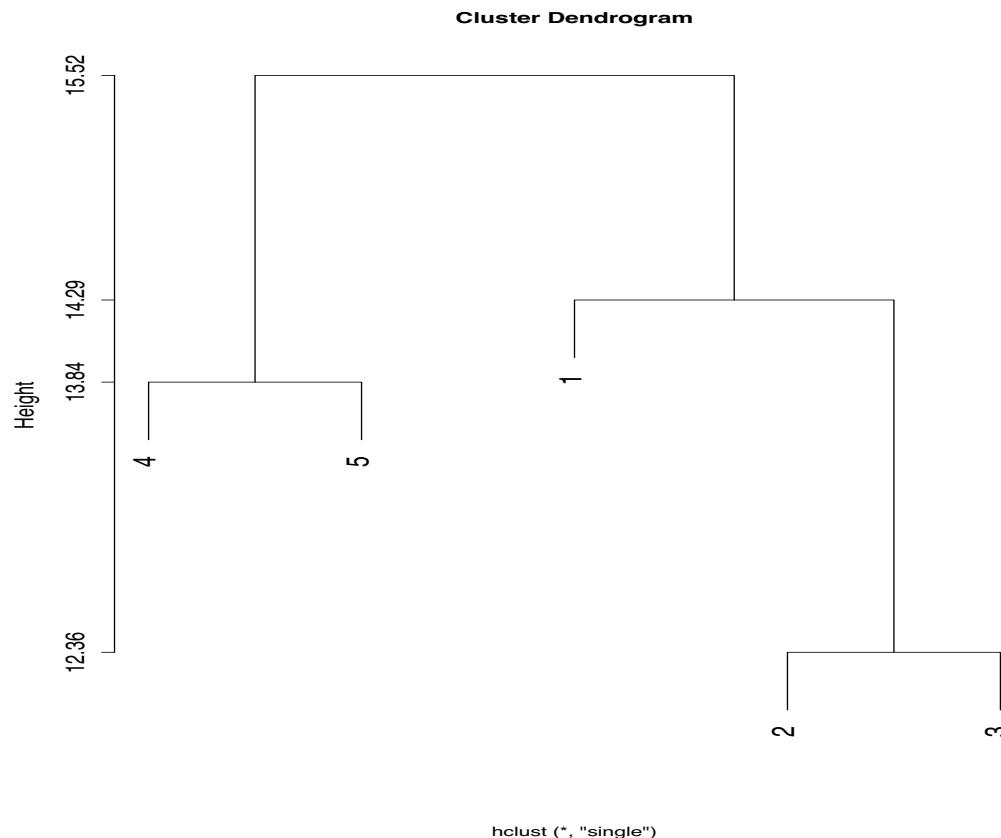


Figure 29.3: Dendrogram for Problem 29.5.

Problem 29.6. Suppose in an unsupervised learning application we are given observations $\dot{x}_1, \dots, \dot{x}_n$. Recall the *within cluster sum of squares*, for K clusters A_1, \dots, A_K where d is a distance function and $g(A_i)$ is a cluster centroid:

$$SS_{\text{within}} = \sum_{i=1}^K \sum_{j \in A_i} d(\dot{x}_j, g(A_i))^2.$$

A K -means clustering algorithm was applied to the data, allowing the number of clusters K to vary from 1 to 6. The following table gives the separate sum of squares within each cluster:

	1	2	3	4	5	6
1	12679.5	-	-	-	-	-
2	1741.7	1503.0	-	-	-	-
3	606.8	399.8	446.0	-	-	-
4	197.8	161.4	188.6	177.3	-	-
5	57.4	182.6	151.9	89.6	64.8	-
6	93.6	49.0	86.6	50.6	34.0	79.0

Let R^2 be the proportion of total variation explained by the clustering. If we accept as the number of clusters the smallest value of K for which $R^2 \geq 90\%$, what is this number?

SOLUTION:

The total sum of squares SS_{total} is simply the SS for the $K = 1$ model, so

$$SS_{total} = 12679.5.$$

Otherwise, SS_{within} is the sum of the individual cluster sums of squares. Then

$$R^2 = 1 - \frac{SS_{within}}{SS_{total}}.$$

This gives, for $K = 1, 2, 3, 4$:

$$\begin{aligned} R^2[1] &= 1 - \frac{SS_{total}}{SS_{total}} = 0, \\ R^2[2] &= 1 - \frac{1741.7 + 1503.0}{12679.5} = 0.7441, \\ R^2[3] &= 1 - \frac{606.8 + 399.8 + 446.0}{12679.5} = 0.8854, \\ R^2[4] &= 1 - \frac{197.8 + 161.4 + 188.6 + 177.3}{12679.5} = 0.9428. \\ R^2[5] &= 1 - \frac{57.4 + 182.6 + 151.9 + 89.6 + 64.8}{12679.5} = 0.9569. \\ R^2[6] &= +1 - \frac{93.6 + 49.0 + 86.6 + 50.6 + 34.0 + 79.0}{12679.5} = 0.9690. \end{aligned}$$

The smallest number of clusters that yield at least 90% variation explained is $K = 4$.

Problem 29.7. We observe n observations of the random vector $\tilde{X} = (X_1, X_2, X_3)$. Each component has zero mean and unit variance. The correlation matrix is

$$\Lambda = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{21} & 1 & \rho_{23} \\ \rho_{31} & \rho_{32} & 1 \end{bmatrix}.$$

A scaled principal components analysis is performed on the data, yielding the following principal component loadings:

	PC1	PC2	PC3
X_1	0.73	-0.02	-0.68
X_2	0.68	-0.02	0.73
X_3	-0.03	-1.00	0.00

and the scree plot shown in Figure 29.4. Which of the correlations in Λ could plausibly be 0, and which are likely to be positive? Explain your answer.

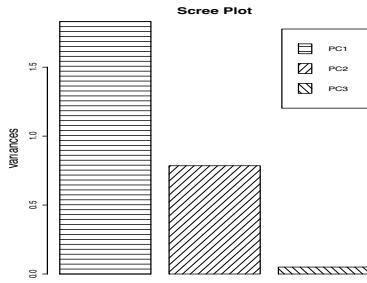


Figure 29.4: Scree plot for Problem 29.7.

SOLUTION:

By the scree plot, most variation is explained by the first 2 PCs. The first PC has loadings primarily on X_1, X_2 , of the same sign. We therefore expect $\rho_{12} = \rho_{21} > 0$. Most of the loadings on the second PC is on X_3 , so we expect X_3 to be approximately independent of X_1 and X_2 , this means we could have $\rho_{13} = \rho_{31} = \rho_{23} = \rho_{32} = 0$.

29.2 Data Analysis

Problem 29.8. For this problem use data set `fg1` from the MASS package. This is a forensic application. The observations consist of fragments of broken glass. The `type` column gives the type of glass. The `RI` column gives refractive index. See description from `help(fg1)`. The remaining 8 columns are percentages by weight of various oxides. The row totals of these 8 columns are approximately 100%.

In this problem the ability of hierarchical clustering to distinguish between types of glass based on forensic samples of chemical composition and refractive index will be examined.

- (a) For this exercise we will only consider 3 glass types: window float glass (`WinF`), window non-float glass (`WinNF`) and vehicle headlamps (`Head`). Create a subset of the data from only these glass types. Then standardize each column to zero mean and unit variance.
- (b) Using the function `hclust` plot dendograms for hierarchical clusterings using agglomeration methods `single`, `complete` and `average`. Generally, do the observations appear to cluster by class `gr` in any of the dendograms? Substitute single character labels when plotting the histograms, since `WinF` and `WinNF` will be difficult to distinguish visually.
- (c) There are various ways to quantify the ability of a hierachal clustering to accurately distinguish classes. Suppose we create a single clustering of size $k = \text{c.size}$, using `cutree(hfit, k=c.size)`. Suppose one sample from each of the 3 types of glass is chosen at random. Let α_k be the probability that the 3 observations are in different clusters. Suppose p_{js} is the proportion of samples of glass type $j \in \{1, 2, 3\}$ in cluster $s \in \{1, \dots, k\}$. Give an expression for α_k in terms of the proportions p_{js} .
- (d) The proportions p_{js} can be easily estimated by cross-tabulating glass type and cluster membership. Write an R program that estimates and plots α_k for each of the hierarchical cluster-

ings created in Part (b). Superimpose the three plots on a single graph, and use the range $k = 1, \dots, 10$. In general, how do the agglomeration methods compare in terms of accuracy?

- (e) One way to assess whether or not α_k is significantly large is to use a permutation procedure. Suppose the original types are contained in the vector `gr`. Then, create a new class vector `gr.perm` by randomly permuting the original class vector `gr` (you can use function `sample()`). Create a new sequence α'_k , $k = 1, \dots, 10$ with the same procedure used in Part (d), except that `gr` is replaced by `gr.perm`. Do the permutation 25 times, superimposing all α_k and α'_k sequences on the same plot. Make sure the sequence types are easily distinguishable (say, use green for α_k and gray for each α'_k). Do this for each of the hierarchical clusterings created in Part (b). Use separate plots for each, but use the `ylim=c(0,1)` option when plotting so that the scales will be comparable. Which clusterings are compatible with the actual glass types?

SOLUTION:

The following code may be used for the analysis. Comments follow.

```
library(MASS)

# (a)

fgl2 = subset(fgl, type %in% c("WinF", "WinNF", "Head"))
gr = as.factor(as.character(fgl2$type))
gri = as.integer(gr)
xf = fgl2[,1:9]
xf = apply(xf, 2, function(x) {(x - mean(x))/sd(x)} )

# (b)

par(mfrow=c(3,1))
hfit1 = hclust(dist(xf), method='single')
plot(hfit1, labels=gri)
hfit2 = hclust(dist(xf), method='complete')
plot(hfit2, labels=gri)
hfit3 = hclust(dist(xf), method='average')
plot(hfit3, labels=gri)

# (d)

# this function is used to calculate alpha[k]

f0 = function(x) {
  x[1]*x[2]*(1-x[3]) + x[1]*x[3]*(1-x[2]) + x[2]*x[3]*(1-x[1]) + x[1]*x[2]*x[3]
}

n = dim(fgl2)[1]
```

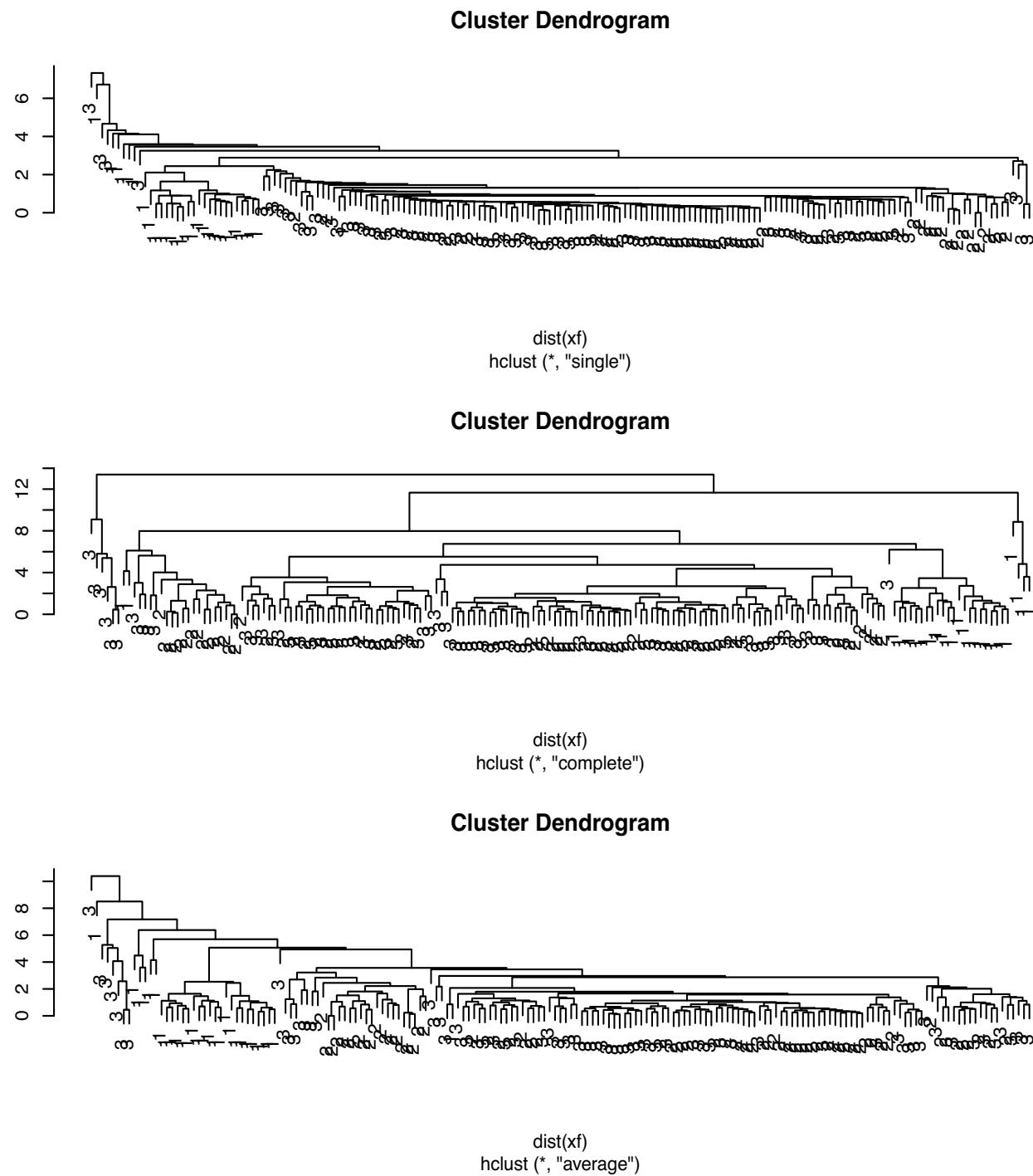


Figure 29.5: Plot for Problem 29.8 (b).

```
par(mfrow=c(1,1))
```

```

l2 = NULL
for (i in 1:10) {
  mm = table(gr,cutree(hfit1,k=i))
  mmp = apply(mm,1,function(x) {x/sum(x)})
  mmp = matrix(mmp,ncol=3)
  mmp = t(mmp)
  pp1 = 1-sum(apply(mmp,2,f0))

  mm = table(gr,cutree(hfit2,k=i))
  mmp = apply(mm,1,function(x) {x/sum(x)})
  mmp = matrix(mmp,ncol=3)
  mmp = t(mmp)
  pp2 = 1-sum(apply(mmp,2,f0))

  mm = table(gr,cutree(hfit3,k=i))
  mmp = apply(mm,1,function(x) {x/sum(x)})
  mmp = matrix(mmp,ncol=3)
  mmp = t(mmp)
  pp3 = 1-sum(apply(mmp,2,f0))

  l2 = rbind(l2, c(pp1,pp2,pp3))
}
matplot(1:10, l2,type='b',xlab='cluster size k', ylab='alpha[k]',lwd=2,ylim=c(0,1))
legend('bottomright',legend=c('single','complete','average'),pch=NA,lty=1:3,col=1:3)

# (e)

# permuted class vector

# create list of permuted alpha[k] values

l2.perm.list = list()
method.names = c('single','complete','average')
hfit.list = list(hfit1,hfit2,hfit3)

# do each cluster method

set.seed(12345)
nperm = 25
for (iii in 1:3) {
  l2.perm = matrix(NA, 10, nperm)
  for (jjj in 1:nperm) {
    grp = sample(gr)
    l2v = NULL
    for (i in 1:10) {
      l2v = rbind(l2v, c(pp1,pp2,pp3))
    }
    l2.perm[,jjj] = l2v
  }
}

```

```

mm = table(grp,cutree(hfit.list[[iii]],k=i))
mmp = apply(mm,1,function(x) {x/sum(x)})
mmp = matrix(mmp,ncol=3)
mmp = t(mmp)
pp = 1-sum(apply(mmp,2,f0))
l2v = c(l2v, pp)
}
l2.perm[,jjj] = l2v
}
l2.perm.list[[iii]] = l2.perm
}

# plot results

par(mfrow=c(2,2))
for (iii in 1:3) {
  matplot(1:10,l2.perm.list[[iii]], col='gray',lty=1,pch=NULL,type='l',
    xlab='cluster size k', ylab='alpha[k]',ylim=c(0,1),main=method.names[iii])
  lines(1:10,l2[,iii],col='green',type='b')
  legend('topleft',legend=c('Original','Permuted'),lty=1,col=c('green','grey'))
}

```

- (a) See code.
- (b) To some degree, glass types appear to cluster for each agglomeration method (Figure 29.5).
- (c) Let $A = \{\text{at least 2 two samples in the same cluster}\}$, $A_s = \{\text{at least 2 two samples in cluster } s\}$. Then $A = \bigcup_{s=1}^k A_s$. Furthermore, because at most one cluster can contain at least 2 samples, the sets A_s are mutually exclusive. This means

$$\alpha_k = 1 - P(A) = 1 - \sum_{s=1}^k P(A_s).$$

Then we have

$$\begin{aligned}
 P(A_s) &= P(\{\text{Type 1 and 2 only in cluster } s\}) + P(\{\text{Type 1 and 3 only in cluster } s\}) \\
 &\quad + P(\{\text{Type 2 and 3 only in cluster } s\}) + P(\{\text{Type 1, 2 and 3 in cluster } s\}) \\
 &= p_{1s}p_{2s}(1-p_{3s}) + p_{1s}p_{3s}(1-p_{2s}) + p_{2s}p_{3s}(1-p_{1s}) + p_{1s}p_{2s}p_{3s}.
 \end{aligned}$$

This gives

$$\alpha_k = 1 - P(A) = 1 - \sum_{s=1}^k [p_{1s}p_{2s}(1-p_{3s}) + p_{1s}p_{3s}(1-p_{2s}) + p_{2s}p_{3s}(1-p_{1s}) + p_{1s}p_{2s}p_{3s}].$$

- (d) Higher values of α_k signify more accurate clustering. By this criterion, the complete method is the most successful, while the single method is the least successful. (Figure 29.6).

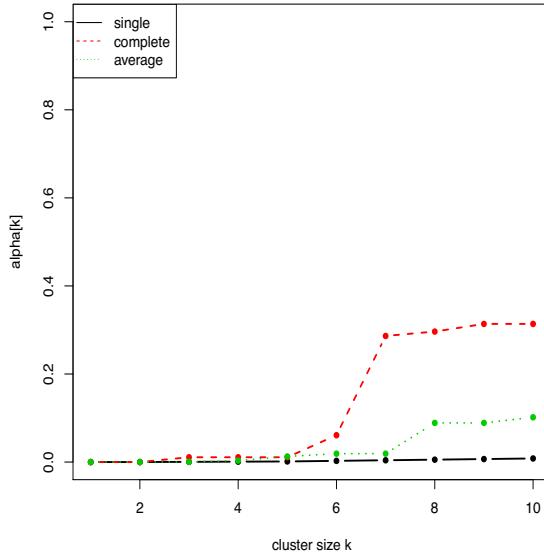


Figure 29.6: Plot for Problem 29.8 (d).

- (e) The observed values of α_k are compatible with the permutation method values for the single and average method, so we cannot conclude that these clusterings are compatible with the true glass types, using this criterion. For the complete method, the observed values of α_k are notably larger than the permutation values, for $k \geq 7$. These clusterings are to some degree compatible with the true glass types. (Figure 29.7).

29.3 Theoretical Complements

Problem 29.9. A classifier requires variation within a set of features, which can be quantified and analyzed in various ways. The purpose of a classifier can be thought of as to determine what portion of the variation can be explained by class. Sometimes, it is possible to identify, then remove, feature variation which we know will not be explained by class.

For this problem use data set `crabs` from the `MASS` package. This contains data on 200 crabs. There are 5 morphological measurements (columns 4-8). The variable `sp` identifies the crab by species (B for blue, O for orange). The variable `sex` identifies the sex (M or F).

- (a) By combining `sp` and `sex` we can identify 4 classes of crab in total. Create a class variable `gr` which does this.
- (b) Create a pairwise plot using all 5 morphological features (leave them all in their original units of millimeters `mm`). Color each class separately (they need not be labeled). How would you characterize feature variation attributable to class? What other form of variation is there which is not explainable by class?

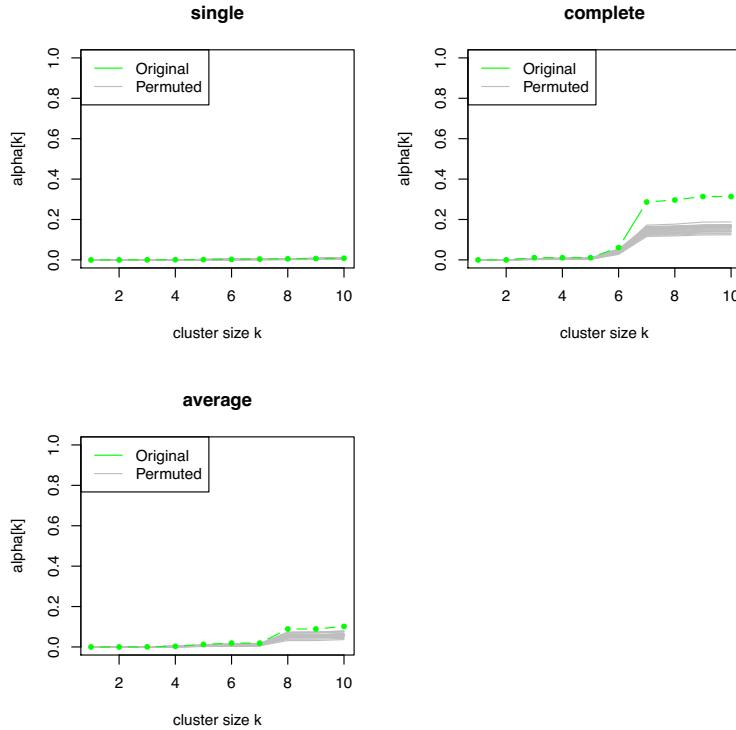


Figure 29.7: Plot for Problem 29.8 (e).

- (c) Calculate the principal components for the 5 morphological features. Using centering but not scaling. Create a pairwise plot using all 5 principal components, using separate coloring for each class (again, classes need not be labeled). What form of variation does the first principal component capture. What subset of principal components appears to best capture variation due to class?
- (d) Create a function that inputs a feature matrix X , number of classes K and class vector gr , and which performs the following steps:
 - (i) Calculates a K -means cluster solution based on the input X and K .
 - (ii) Draws pairwise plots using all features, and superimposes the centers output with clustering solution (see lecture code). Classes need not be visually distinguished, but this won't be discouraged, as long as the centers are clearly distinguishable.
 - (iii) Calculates $R^2 = 1 - SS_{\text{within}}/SS_{\text{total}}$.
 - (iv) Calculates the classification error rate. Because K -means clustering is an unsupervised learning algorithm, we need to define 'error' carefully. Assign to each true class the highest frequency cluster among observations of that class. Take that cluster to be a correct prediction, then calculate classification error accordingly.
- (e) Apply the function of part (d) to the original feature matrix X , the (centered but unscaled) principal components P , and to the feature matrix $P^{(-1)}$ defined by principal components 2,3,4 and 5 .
- (f) It can be shown that the principal components are an orthonormal transformation R of the

original data, which is *isometric*, or distance-preserving:

$$\|x - y\| = \|xR - yR\|.$$

What role does this fact play in the results of part (e)?

- (g) Suppose we wish to construct a classifier for $(species, sex)$, and we can choose between any of the three feature matrices used in part (e). Which have the highest R^2 and which have the lowest classification error? What form of variation is the K -means solution for feature matrix X attempting to explain? What form of variation is the K -means solution for feature matrix $P^{(-1)}$ attempting to explain?

SOLUTION:

- (a) Use the following code:

```
> xf = crabs[,4:8]
> gr = as.factor(paste(crabs$sp,crabs$sex,sep=' '))
> gri = as.integer(gr)
> nf = dim(xf)[1]
> table(gr)
gr
BF BM OF OM
50 50 50 50
```

- (b) Use the following code:

```
my.pch = gri
nf = dim(xf)[1]
pairs(xf,col=my.pch)
```

There appear to be distinct classes in the form of approximate linear relationships between the features. The class groups can be distinguished, by varying degrees, by the slope of these lines. Of course, within each class there is significant variation along the line, clearly driven by the overall size of an individual crab. This is the same within each class. See Figure 29.8.

- (c) Use the following code:

```
> crab.pc = prcomp(xf,center=T,scale.=F)
> xfp = crab.pc$x
> pairs(xfp,col=my.pch)
> crab.pc$rotation
      PC1        PC2        PC3        PC4        PC5
FL 0.2889810  0.3232500 -0.5071698  0.7342907  0.1248816
RW 0.1972824  0.8647159  0.4141356 -0.1483092 -0.1408623
CL 0.5993986 -0.1982263 -0.1753299 -0.1435941 -0.7416656
CW 0.6616550 -0.2879790  0.4913755  0.1256282  0.4712202
BD 0.2837317  0.1598447 -0.5468821 -0.6343657  0.4386868
>
```

PC1 has little information about class. By the loadings, we can see that PC1 is related to size. On the other hand PC2 and PC3 cluster very clearly by class. See Figure 29.9.

- (d) The following function performs the required tasks:

```
my.kmeans = function(x,k,gr) {
  # calculate fit
  nf = dim(x)[1]
  fit = kmeans(x,centers=k,nstart=100)

  # construct plot
  x = rbind(x,fit$centers)
  colv=c(rep(1,nf),rep(2,k))
  pchv=c(rep(3,nf),rep(19,k))
  pairs(x,col=colv,pch=pchv)

  # calculate metrics
  conf.mat = table(gri,fit$cluster)
  err = 1 - sum(apply(conf.mat,1,max))/nf
  r2 = fit$betweenss/fit$totss
  ans = c(err,r2)
  names(ans) = c('err','r2')
  return(ans)
}
```

- (e) Use the following code:

```
> my.kmeans(xf,4,gr)
      err          r2
0.6450000 0.8932867
>
> my.kmeans(crab.pc$x,4,gr)
      err          r2
0.6450000 0.8932867
>
> my.kmeans(crab.pc$x[,2:5],4,gr)
      err          r2
0.0650000 0.7444877
>
my.kmeans(xf,4,gr)
my.kmeans(crab.pc$x,4,gr)
my.kmeans(crab.pc$x[,2:5],4,gr)
```

See Figures 29.10-29.12.

- (f) K -means clustering is based on Euclidean distances. Therefore, a K -means solution for feature matrix X will be equivalent to that for feature matrix XR for any distance-preserving transformation R . For this reason, the cluster solutions for X and P of part (e) are identical.
- (g) R^2 is maximized for X , P , while classification error is minimized for $P^{(-1)}$. The feature matrix $P^{(-1)}$ is more suitable to this particular classification problem. Although X, P has a higher R^2 , it is clear from the plots that the K -means cluster for X is attempting to explain variation by size, not by class.

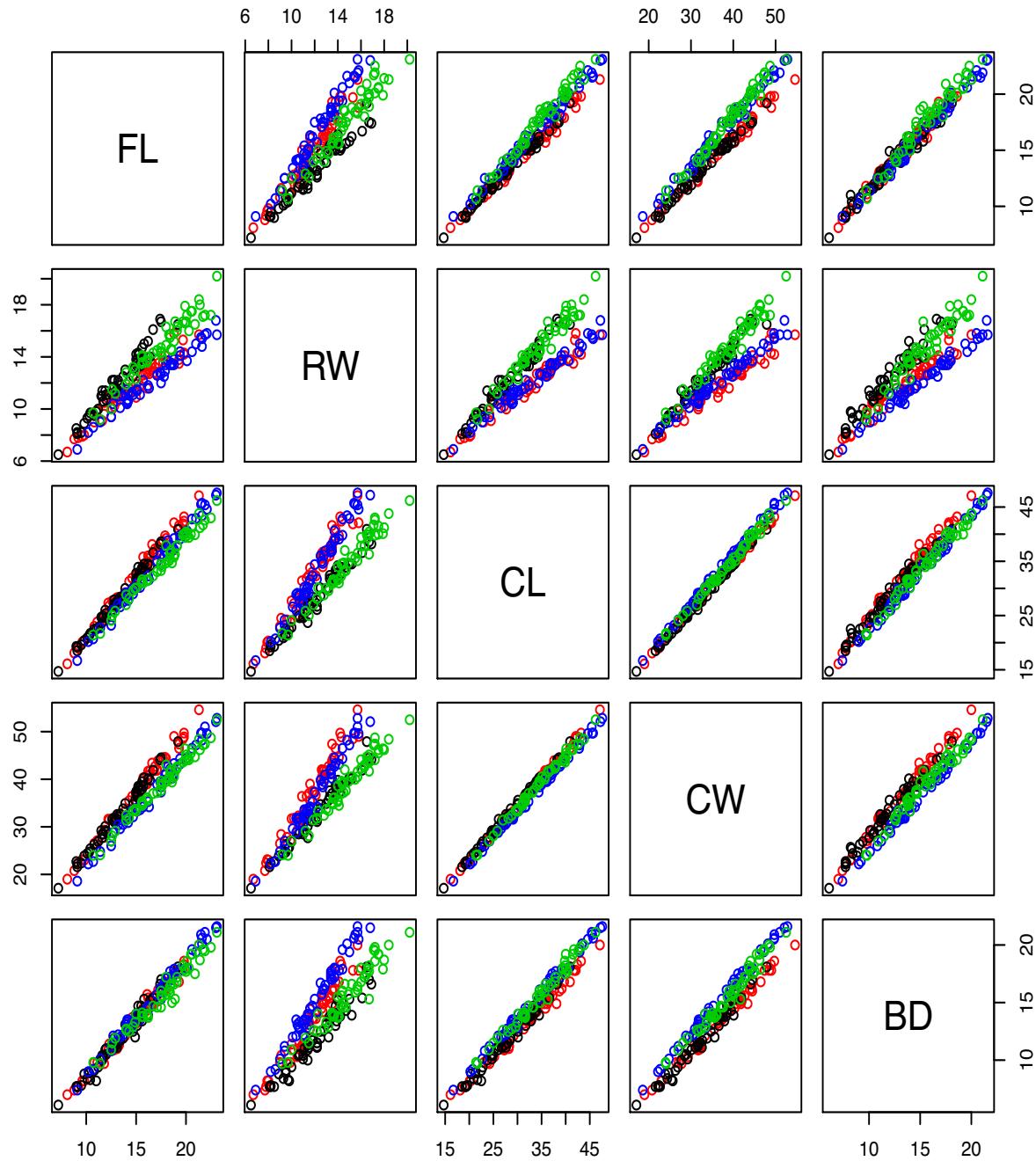


Figure 29.8: Pairwise plots for X Problem 29.9 (b).

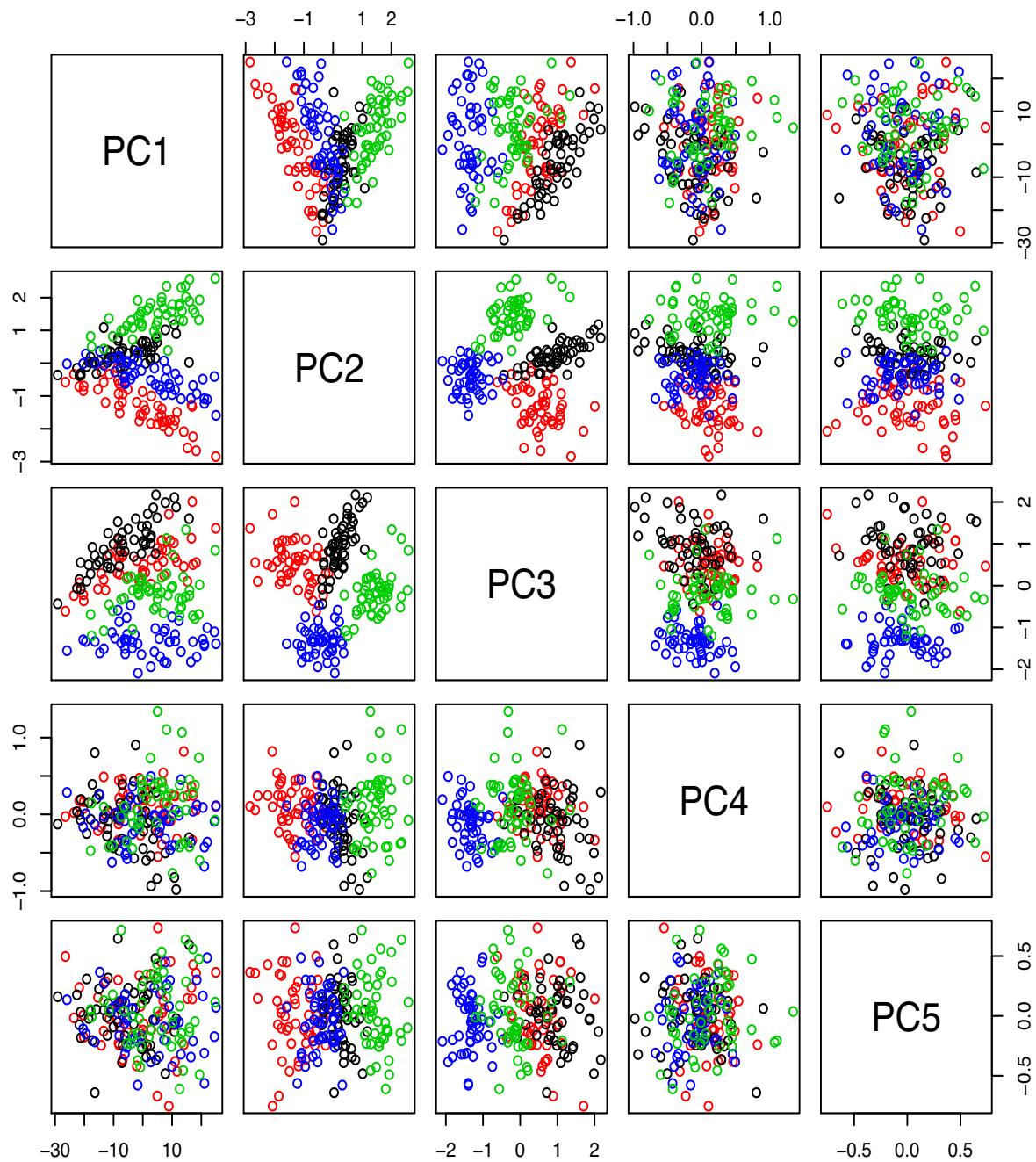


Figure 29.9: Pairwise plots for principal components P Problem 29.9 (c).

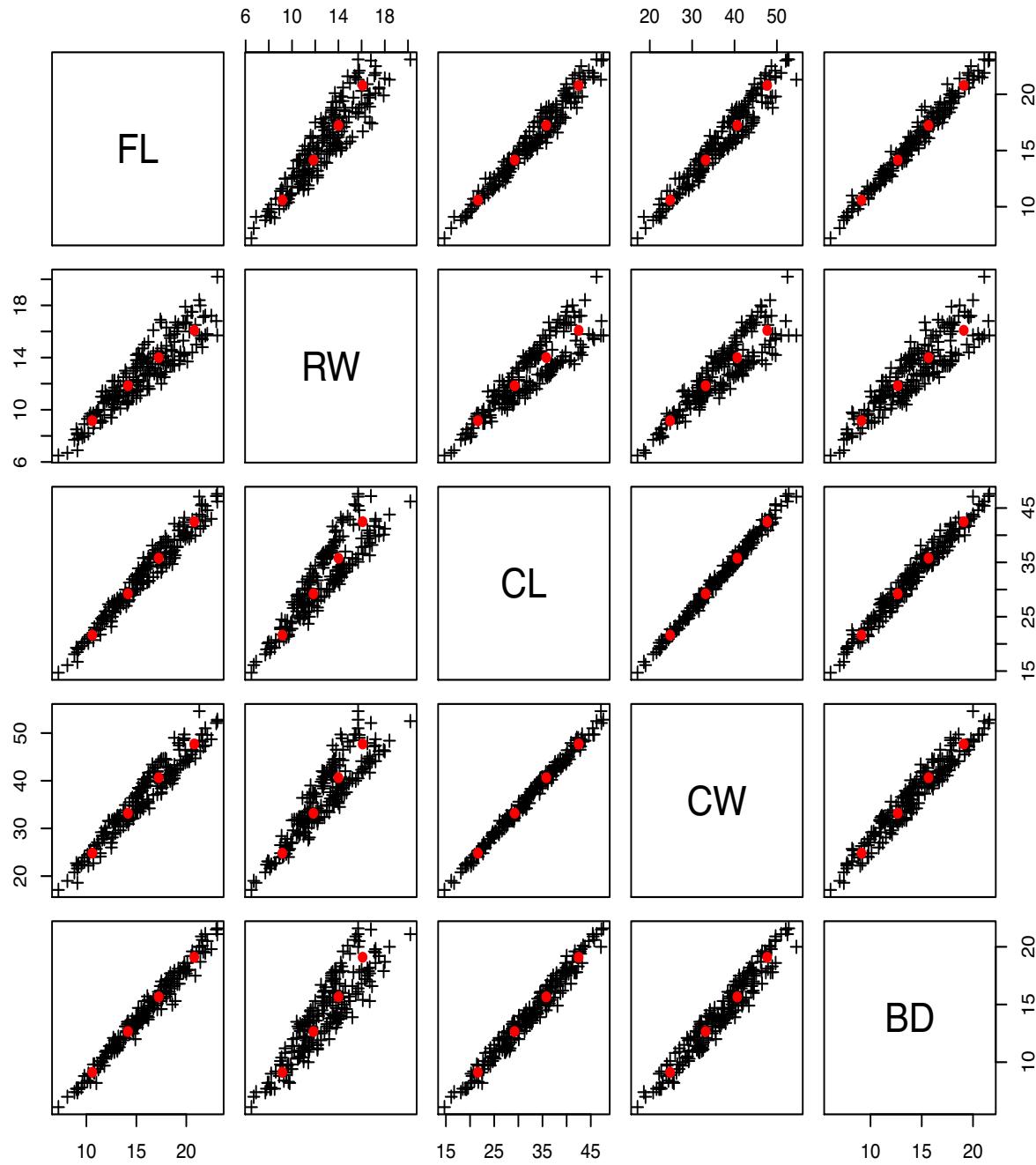
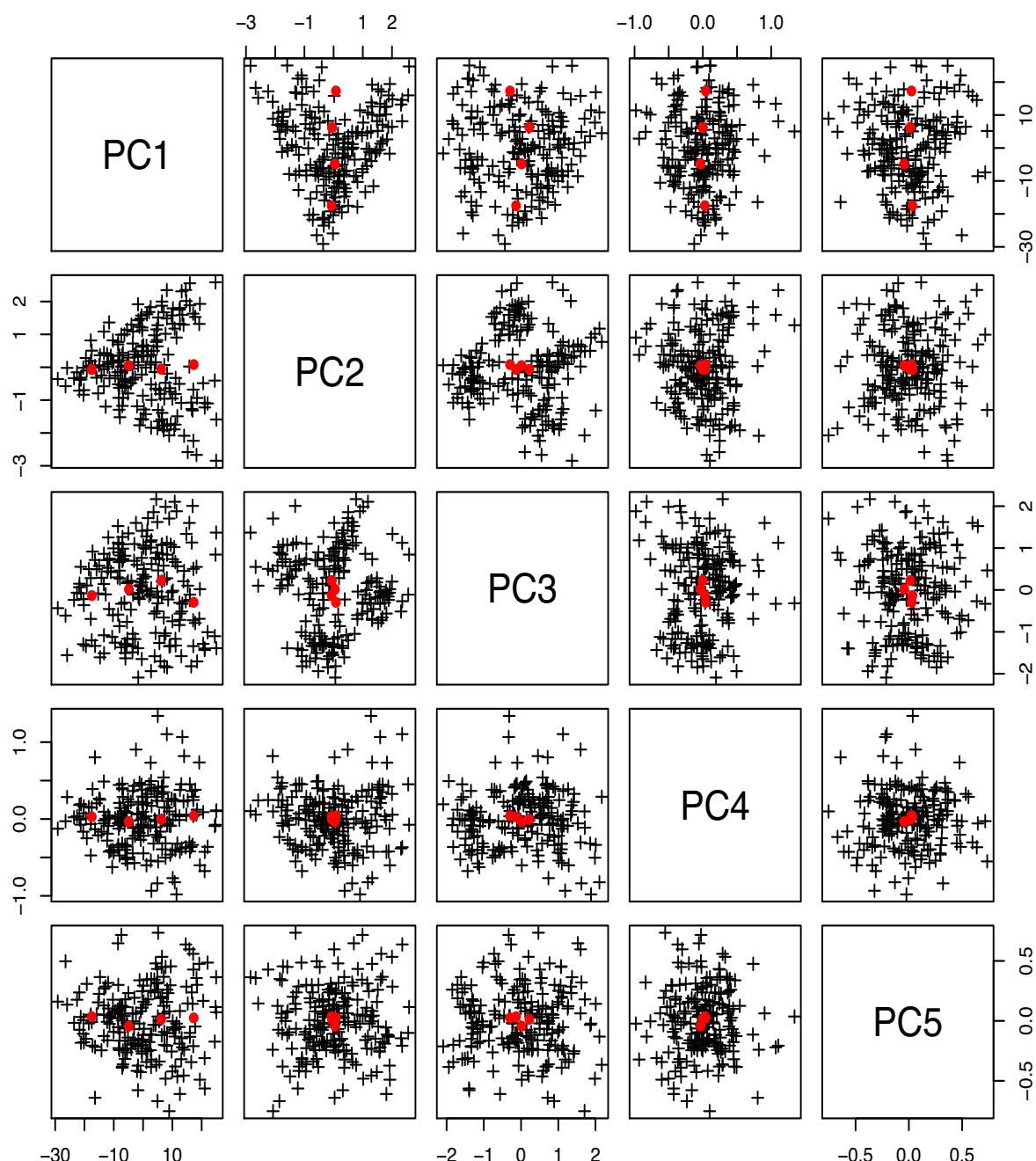
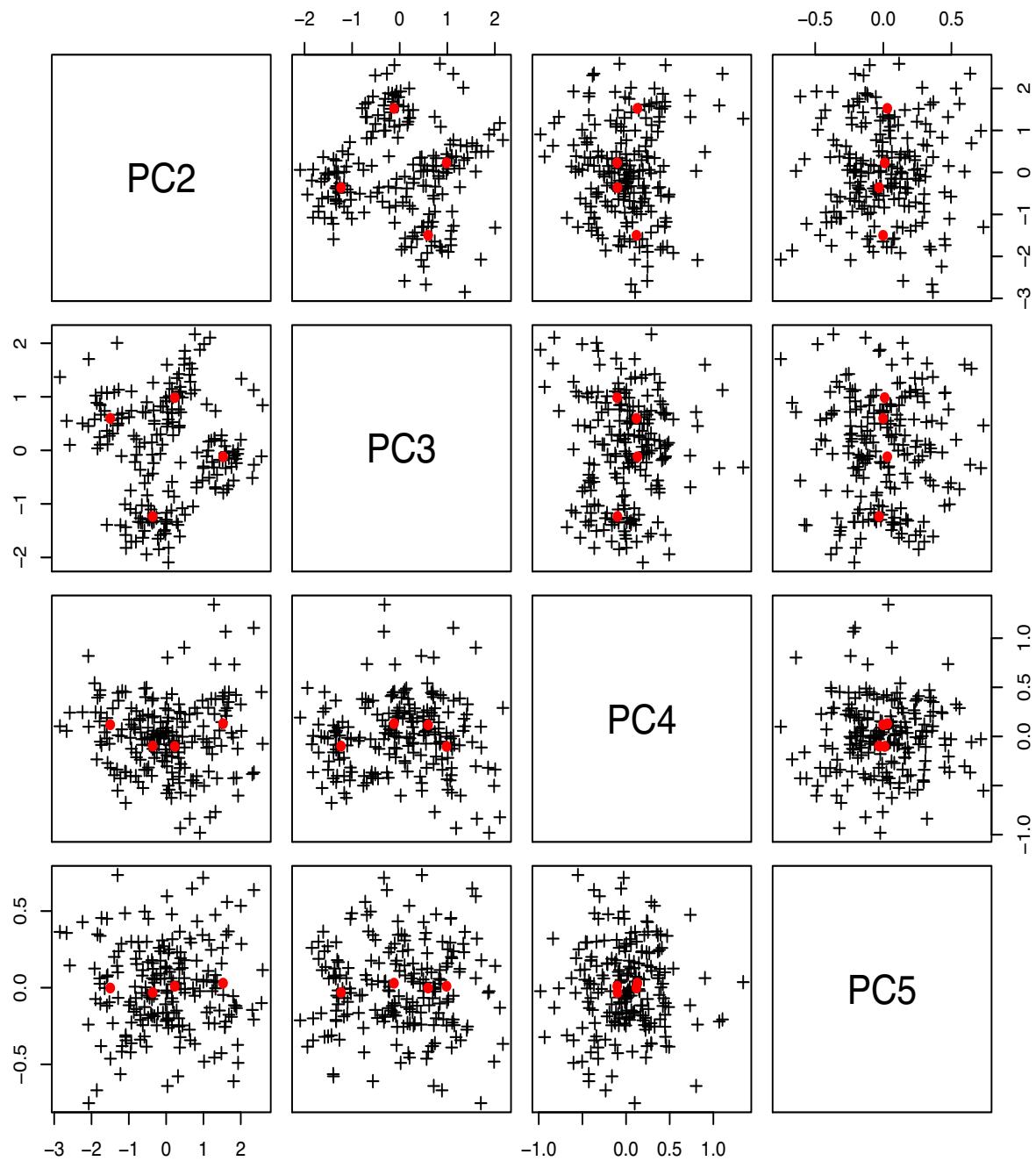


Figure 29.10: Clustering solution for X Problem 29.9 (e).

Figure 29.11: Clustering solution for P Problem 29.9 (e).

Figure 29.12: Clustering solution for $P^{(-1)}$ Problem 29.9 (e).

Chapter 30

Practice Problems - Model Selection and Splines

30.1 Exercises

Problem 30.1. Suppose we fit a simple linear regression model:

$$y = \beta_0 + \beta_1 x + \epsilon, \quad (30.1)$$

and after examining the residuals, suspect that the relationship between x and y is not linear.

- (a) Assume that the range of x is $[10, 45]$. We wish to consider a model of the form

$$y = f(x) + \epsilon,$$

where $f(x)$ is a continuous piecewise linear function with knots at 20 and 35. Find a way of constructing this model using the form

$$y = \beta_0 + \beta_1 x + \beta_2 b_1(x) + \beta_3 b_2(x) + \epsilon, \quad (30.2)$$

where $b_1(x)$ and $b_2(x)$ are two basis functions. Identify the basis functions precisely.

- (b) Suppose models (30.1) and (30.2) are fit using the same data with sample size $n = 54$, and the resulting error sums of squares are $SSE_1 = 2398.45$ and $SSE_2 = 2103.23$. Does the model (30.2) significantly reduce the SSE compared to (30.1)?

SOLUTION:

- (a) The required basis functions are

$$\begin{aligned} b_1(x) &= (x - 20)_+ \\ b_2(x) &= (x - 35)_+ \end{aligned}$$

where $(y)_+ = y$ if $y \geq 0$ and $(y)_+ = 0$ if $y < 0$.

- (b) The models (1) and (2) have 2 and 4 degrees of freedom, respectively. We may take (2) to be a full model and (1) to be a reduced model, giving F -statistics

$$F = \frac{(SSE_R - SSE_F)/2}{SSE_F/(n-4)} = \frac{(2398.45 - 2103.23)/2}{2103.23/(54-4)} = 3.509126$$

We can use R to get the P-value:

```
> 1 - pf(3.509126, 2, 54-4)
[1] 0.03748889
```

so that model (2) is a significant improvement over model (1) at a $\alpha = 0.05$ significance level.

Problem 30.2. The following full linear regression model is considered:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon.$$

An all subsets model selection procedure is to be used to determine which of the predictors X_1, X_2, X_3 to retain. The sample size is $n = 150$. The AIC score will be used, in the form $AIC = n \log(SSE/n) + C$. The error sum of squares for each tested model is given in the following table. What will be the selected model?

	Model	SSE
1	$Y = 1$	1940.373
2	$Y = X_1$	1399.735
3	$Y = X_2$	485.242
4	$Y = X_3$	1940.367
5	$Y = X_1 + X_2$	0.268
6	$Y = X_1 + X_3$	1399.713
7	$Y = X_2 + X_3$	485.239
8	$Y = X_1 + X_2 + X_3$	0.267

SOLUTION:

Formula is

$$AIC = n \log(SSE/n) + 2k,$$

where k is the number of parameters. We can construct table:

Model	k	SSE	AIC
1 $Y = 1$	1	1940.37	386.00
2 $Y = X_1$	2	1399.74	339.01
3 $Y = X_2$	2	485.24	180.10
4 $Y = X_3$	2	1940.37	388.00
5 $Y = X_1 + X_2$	3	0.27	-943.35
6 $Y = X_1 + X_3$	3	1399.71	341.01
7 $Y = X_2 + X_3$	3	485.24	182.10
8 $Y = X_1 + X_2 + X_3$	4	0.27	-941.41

The model $Y = X_1 + X_2$ has the lowest AIC (= -943.35).

Problem 30.3. A regression model is to be developed for a response Y and single predictor X in range $X \in [100, 200]$, based on $n = 11$ paired observations. The following two models were considered, and the resulting error sum of squares is reported:

- (i) Simple linear regression model $Y = \beta_0 + \beta_1 X + \epsilon$ [SSE = 67813.35].
- (ii) Cubic spline with single knot at $X = 150$ [SSE = 53599.91].

Using the BIC score, which is the best model? Use the form $BIC = n \log(SSE/n) + C$.

SOLUTION:

The formula is

$$BIC = n \log(SSE/n) + \log(n)k,$$

where k is the number of parameters. We can construct table:

Model	k	SSE	BIC
linear	2.00	67813.35	100.79
cubic spline	5.00	53599.91	105.39

The linear model has the lowest BIC (= 100.79).

Problem 30.4. Given a single predictor x and response y , a polynomial regression model is considered:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon.$$

Suppose we consider four models (full model and 3 reduced models). Suppose further that the sample size is $n = 75$, and that the four models are fit, yielding the following error sums of squares SSE:

MODEL	SSE
M_0	239.2
M_1	82.8
M_2	87.9
M_{12}	79.8

Determine the model selected by the AIC and the BIC criterion. Use the form $n \log(SSE/n) + C$.

SOLUTION:

We have scores:

$$\begin{aligned} AIC &= n \log(SSE/n) + 2q, \\ BIC &= n \log(SSE/n) + \log(n)q, \end{aligned}$$

where q is the number of parameters, and $n = 75$. This leads to table:

Model	q	AIC	BIC
M_0	1	88.99	91.30
M_1	2	11.42	16.06
M_2	2	15.90	20.54
M_{12}	3	10.65	17.61

The AIC selection is M_3 , while the BIC selection is M_1 .

Problem 30.5. A regression model is to be developed for a response Y and single predictor X in range $X \in [10, 24]$, based on $n = 29$ paired observations. The following four models were considered, and the resulting error sum of squares is reported:

- (i) Constant model $Y = \beta_0 + \epsilon$ [$SSE = 576983.05$].
- (ii) Simple linear regression model $Y = \beta_0 + \beta_1 X + \epsilon$ [$SSE = 16546.78$].
- (iii) Linear spline with knots at $X = 15, 20$ [$SSE = 13819.92$].
- (iv) Linear spline with knots at $X = 12, 17, 19, 22$ [$SSE = 12169.24$].

Using the AIC score, which is the best model?

SOLUTION:

We have two methods for calculating AIC:

$$\begin{aligned} AIC_1 &= \frac{1}{\hat{\sigma}^2} SSE + 2k \\ AIC_2 &= n \log(SSE/n) + 2k, \end{aligned}$$

where k is the number of parameters, and $\hat{\sigma}^2$ is an estimate of the error variance (here, use the estimate obtained from model 4, $\hat{\sigma}^2 = SSE/(n-6)$). For a linear spline with N knots and an intercept, we have $k = 2 + N$. This means the number of parameters for the four models is $k = 1, 2, 4, 6$, respectively. The AIC values are in the following table:

Model	k	AIC_1	AIC_2
1	1	1092.50	289.05
2	2	35.27	188.05
3	4	34.12	186.83
4	6	35.00	187.14

For both versions of the AIC, model (iii) is the selected model.

Problem 30.6. We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable. Suppose $g(x)$ is a piecewise polynomial with knots at $\xi = 0, 1$ of the form:

$$g(x) = \begin{cases} \alpha & ; \quad x \leq 0 \\ ax^3 + bx^2 + cx + d & ; \quad x \in (0, 1] \\ \beta & ; \quad x > 1 \end{cases} .$$

It is further required that $g(x)$ is continuous at both knots, and that the first derivative of $g(x)$ is continuous and equal to 0 at both knots. See Figure 30.1 for an example of $g(x)$.

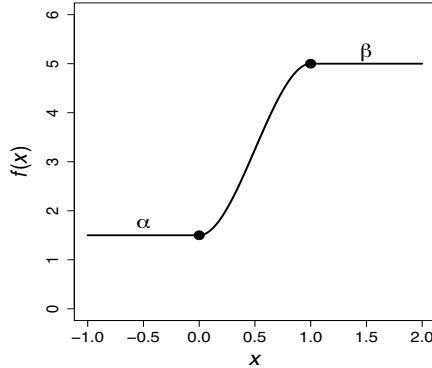


Figure 30.1: Example of $g(x)$ for Problem 30.6.

- (a) Express the coefficients a, b, c, d as functions of α and β .
- (b) How many degrees of freedom does the model have?

SOLUTION:

(a) Note that

$$\frac{d}{dx}(ax^3 + bx^2 + cx + d) = 3ax^2 + 2bx + c.$$

So, we have constraints:

$$\begin{aligned}\alpha &= d \\ 0 &= c \\ \beta &= a + b + c + d \\ 0 &= 3a + 2b + c.\end{aligned}$$

Directly, we have $c = 0$, $d = \alpha$. Substitution gives

$$\begin{aligned}a + b &= \beta - \alpha \\ 3a + 2b &= 0.\end{aligned}$$

To summarize:

$$\begin{aligned}a &= 2(\alpha - \beta) \\ b &= -3(\alpha - \beta) \\ c &= 0 \\ d &= \alpha.\end{aligned}$$

(b) 6 parameters with 4 constraints yields $2 = 6 - 4$ model degrees of freedom.**Problem 30.7.** We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and $x_i \in [10, 20]$ is a predictor variable. We consider the following six models**M1** $g(x) = \beta_1 x$, where β_1 is to be estimated.**M2** $g(x) = \beta_0 + \beta_1 x$, where β_0, β_1 are to be estimated.**M3** $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2$, where $\beta_0, \beta_1, \beta_2$ are to be estimated.**M4** $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$, where $\beta_0, \beta_1, \beta_2, \beta_3$ are to be estimated.**M5** $g(x)$ is a continuous piecewise linear spline with 1 knot at $\xi = 13$.**M6** $g(x)$ is a natural cubic spline with 2 knots at $\xi = 15, 17$ (note that $g(x)$ is continuous, and possesses continuous derivatives, at each knot).

Model	<i>SSE</i>
M1	607.807
M2	32.163
M3	14.116
M4	8.707
M5	6.263
M6	9.523

The relevant *SSE* values are given in the following table. The sample size is $n = 181$. Which model is preferred based on the BIC score (use form $BIC = n \log(SSE/n) + C$)?

SOLUTION:

The equation is

$$BIC = n \log(SSE/n) + \log(n)k,$$

where k is the number of parameters. Other than σ^2 , the number of parameters is

M1 β_1 , $k = 1$.

M2 β_0, β_1 , $k = 2$.

M3 $\beta_0, \beta_1, \beta_2$, $k = 3$.

M4 $\beta_0, \beta_1, \beta_2, \beta_3$, $k = 4$.

M5 4 parameters with one constraint, so $k = 4 - 1 = 3$.

M6 2+4+2 parameters with 4 constraints, so $k = 8 - 4 = 4$.

The number of parameters does not include σ^2 , but if this was included the model selection procedure would be unchanged, since we would simply add 1 to each k .

We can construct table:

Model	<i>SSE</i>	Without σ^2		With σ^2	
		<i>k</i>	<i>BIC</i>	<i>k</i>	<i>BIC</i>
M1	607.807	1	224.455	2	229.653
M2	32.163	2	-302.313	3	-297.115
M3	14.116	3	-446.171	4	-440.973
M4	8.707	4	-528.429	5	-523.231
M5	6.263	3	-593.260	4	-588.062
M6	9.523	4	-512.217	5	-507.018

So, model **M5** has the lowest BIC, and is therefore the preferred model.

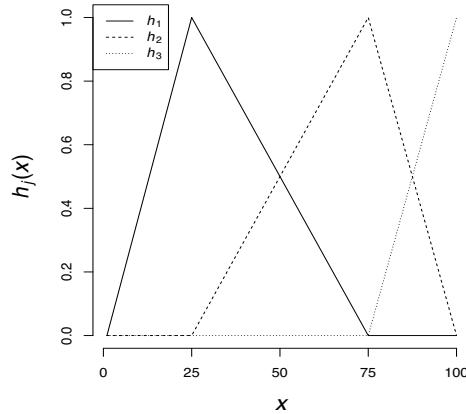
Problem 30.8. We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and $x_i \in [0, 100]$ is a predictor variable. We will assume that $g(x)$ is a continuous linear spline with two knots at $\xi = 25, 75$. One way to do this is to use the basis functions

$$b_1(x) = x; \quad b_2(x) = (x - 25)I\{x > 25\}; \quad b_3(x) = (x - 75)I\{x > 75\},$$

then set $g(x) = \beta_0 + \sum_{j=1}^3 \beta_j b_j(x)$. Suppose we then consider alternative basis functions $h_j(x)$, $j = 1, 2, 3$ shown in the following graph:



Each $h_j(x)$ is a continuous piecewise linear spline with $h_j(0) = 0$. The maximum of each $h_j(x)$ on the range $x \in [0, 100]$ is 1, and the discontinuities in slope occur at the knots $\xi = 25, 75$. Note that in the plot the functions overlap at various places on the horizontal axis. We then set $g(x) = \beta_0^* + \sum_{j=1}^3 \beta_j^* h_j(x)$.

- (a) Write explicitly each basis function $h_j(x)$, $j = 1, 2, 3$ as a linear combination of the functions $b_1(x), b_2(x), b_3(x)$.
- (b) Suppose we use multiple linear regression to estimate the coefficients β_j using basis functions b_1, b_2, b_3 . Suppose then that we use multiple linear regression to estimate the coefficients β_j^* using basis functions h_1, h_2, h_3 . Show that the fitted values will be identical.

SOLUTION:

- (a) If we write

$$h(x) = \alpha_1 b_1(x) + \alpha_2 b_2(x) + \alpha_3 b_3(x)$$

then $h(0) = 0$, since $b_j(0) = 0$ for $j = 1, 2, 3$. Clearly, $h(x)$ is also a linear spline with knots $\xi = 25, 75$. In addition, the slope of $h(x)$ is α_1 for $x < 25$, $\alpha_1 + \alpha_2$ for $x \in (25, 75)$, and $\alpha_1 + \alpha_2 + \alpha_3$ for $x > 75$.

Then note that the slope of $h_1(x)$ is $1/25$ for $x < 25$, $-1/50$ for $x \in (25, 75)$, and 0 for $x > 75$. Therefore, if

$$h_1(x) = \alpha_1 b_1(x) + \alpha_2 b_2(x) + \alpha_3 b_3(x)$$

then we must have $\alpha_1 = 1/25$, $\alpha_2 = -1/50 - \alpha_1 = -3/50$, $\alpha_3 = 0 - \alpha_1 - \alpha_2 = 1/50$.

Next, the slope of $h_2(x)$ is 0 for $x < 25$, $1/50$ for $x \in (25, 75)$, and $-1/25$ for $x > 75$. Therefore, $\alpha_1 = 0$, $\alpha_2 = 1/50$, $\alpha_3 = -1/25 - \alpha_2 = -3/50$.

Finally, $h_3(x) = (1/25) \cdot b_3(x)$. To summarize:

$$\begin{aligned} h_1(x) &= (1/25) \cdot b_1(x) - (3/50) \cdot b_2(x) + (1/50) \cdot b_3(x) \\ h_2(x) &= 0 \cdot b_1(x) + (1/50) \cdot b_2(x) - (3/50) \cdot b_3(x) \\ h_3(x) &= 0 \cdot b_1(x) + 0 \cdot b_2(x) + (1/25) \cdot b_3(x). \end{aligned}$$

- (b) The easiest approach is to note that the two sets of basis functions are related by a linear transformation:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 1/25 & -3/50 & 1/50 \\ 0 & 1/50 & -3/50 \\ 0 & 0 & 1/25 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

From Part (a) we have shown that any function $h_j(x)$ is a linear combination of the basis functions b_1, b_2, b_3 . Since the linear transformation is clearly invertible, any function $b_j(x)$ is a linear combination of the basis functions h_1, h_2, h_3 . Therefore, each set of basis functions span the same function space. This in turn implies that the least squares estimate of $g(x)$ will be the same using either set of basis functions.

Problem 30.9. We wish to fit the model

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n, \tag{30.3}$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable. The function $g(x)$ has the following properties:

- (i) There are two knots $\xi_1 < \xi_2$.
 - (ii) $g(x)$ is continuous at the knots.
 - (iii) $g(x)$ possesses a continuous first derivative at the knots.
 - (iv) $g(x)$ is a first order polynomial $g(x) = a_0 + b_0 x$ for $x < \xi_1$.
 - (v) $g(x)$ is a second order polynomial $g(x) = a_1 + b_1 x + c_1 x^2$ for $x \in (\xi_1, \xi_2)$.
 - (vi) $g(x)$ is a first order polynomial $g(x) = a_2 + b_2 x$ for $x > \xi_2$.
- (a) How many linear constraints are imposed on the parameters $(a_0, b_0, a_1, b_1, c_1, a_2, b_2)$ by properties (i)-(vi)? Write these explicitly.
- (b) Assume the knots ξ_1, ξ_2 are known, but the parameters $(a_0, b_0, a_1, b_1, c_1, a_2, b_2)$ are to be estimated. How many degrees of freedom does this estimation problem possess (that is, how many free parameters are required to completely define $g(x)$)?

SOLUTION:

- (a) Continuity of $g(x)$ and continuity of the first derivative each represent a single linear constraint at each knot, so there are 4 linear constraints. These are

$$\begin{aligned} a_0 + b_0\eta_1 &= a_1 + b_1\eta_1 + c_1\eta_1^2 \\ a_1 + b_1\eta_2 + c_1\eta_2^2 &= a_2 + b_2\eta_2 \end{aligned}$$

for continuity, and

$$\begin{aligned} b_0 &= b_1 + 2c_1\eta_1 \\ b_1 + 2c_1\eta_2 &= b_2, \end{aligned}$$

for continuity of the first derivative.

- (b) The three polynomials together have 7 coefficients. With 4 constraints, this leaves $7 - 4 = 3$ degrees of freedom.

Problem 30.10. We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable in the range $[1, 10]$. We consider the following six models

M1 $g(x) = \beta_1 x$, where β_1 is to be estimated.

M2 $g(x) = \beta_0 + \beta_1 x$, where β_0, β_1 are to be estimated.

M3 $g(x) = \beta_1 \sqrt{x}$, where β_1 is to be estimated.

M4 $g(x) = \beta_0 + \beta_1 \sqrt{x}$, where β_0, β_1 are to be estimated.

M5 $g(x)$ is a continuous piecewise linear spline with 1 knot at $\xi = 4$.

M6 $g(x)$ is a cubic spline with 2 knots at $\xi = 3, 6$.

The relevant SSE values are given in the following table. The sample size is $n = 91$. Which model is preferred based on the AIC score and on the BIC score (use form $n \log(SSE/n) + C$ for each)? Does this model minimize SSE among those considered?

SOLUTION:

The equations are

$$\begin{aligned} AIC &= n \log(SSE/n) + 2k, \\ BIC &= n \log(SSE/n) + \log(n)k, \end{aligned}$$

where k is the number of parameters. Other than σ^2 , the number of parameters is

Model	<i>SSE</i>
M1	74.007
M2	3.441
M3	9.258
M4	2.811
M5	2.935
M6	2.744

M1 β_1 , $k = 1$.

M2 β_0, β_1 , $k = 2$.

M3 β_1 , $k = 1$.

M4 β_0, β_1 , $k = 2$.

M4 $2 + N_{knots}$, where $N_{knots} = 1$ is the number of knots, so $k = 3$.

M5 $4 + N_{knots}$, where $N_{knots} = 2$ is the number of knots, so $k = 6$.

The number of parameters does not include σ^2 , but if this was included the model selection procedure would be unchanged, since we would simply add 1 to each k .

We can construct table:

Model	<i>SSE</i>	Without σ^2			With σ^2		
		<i>k</i>	<i>AIC</i>	<i>BIC</i>	<i>k</i>	<i>AIC</i>	<i>BIC</i>
M1	74.007	1	-16.809	-14.299	2	-14.809	-9.788
M2	3.441	2	-294.036	-289.014	3	-292.036	-284.503
M3	9.258	1	-205.973	-203.463	2	-203.973	-198.952
M4	2.811	2	-312.430	-307.408	3	-310.430	-302.897
M5	2.935	3	-306.497	-298.964	4	-304.497	-294.453
M6	2.744	6	-306.622	-291.557	7	-304.622	-287.046

So, model **M4** has the lowest AIC and BIC (**M4** happens to be the model used to simulate the data).

Problem 30.11. We wish to fit a model of the form

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable. We consider the following five models

M1 $g(x) = \beta_0 + \beta_1 x$, where β_0, β_1 are to be estimated.

M2 $g(x) = \beta_1/\sqrt{x}$, where β_1 is to be estimated.

M3 $g(x) = \beta_0 + \beta_1/\sqrt{x}$, where β_0, β_1 are to be estimated.

M4 $g(x)$ is a continuous piecewise linear spline with 2 knots at $\xi = 3, 8$.

M5 $g(x)$ is a cubic spline with 1 knot at $\xi = 5$.

The relevant SSE values are given in the following table. The sample size is $n = 91$. Which model is preferred based on the AIC score (use form $AIC = n \log(SSE/n) + C$).

Model	SSE
M1	3.171
M2	2.814
M3	2.812
M4	2.801
M5	2.780

SOLUTION:

The formula is

$$AIC = n \log(SSE/n) + 2k,$$

where k is the number of parameters. Other than σ^2 , the number of parameters is

M1 β_0, β_1 , $k = 2$.

M2 β_1 , $k = 1$.

M3 β_0, β_1 , $k = 2$.

M4 $2 + N_{knots}$, where $N_{knots} = 2$ is the number of knots, so $k = 4$.

M5 $4 + N_{knots}$, where $N_{knots} = 1$ is the number of knots, so $k = 5$.

We can construct table:

	Model	SSE	k	AIC	$k + 1$	AIC (with σ^2)
1	M1	3.171	2	-301.481	3	-299.481
2	M2	2.814	1	-314.339	2	-312.339
3	M3	2.812	2	-312.400	3	-310.400
4	M4	2.801	4	-308.749	5	-306.749
5	M5	2.780	5	-307.452	6	-305.452

So, model **M2** has the lowest AIC (including σ^2 in the parameter count necessarily yields the same conclusion).

Problem 30.12. Data for a linear predictive model consists of responses Y , and five feature variables X_1, \dots, X_5 . There are $n = 100$ observations. Each feature is standardized to a mean of 0 and a standard deviation of 1. The principal components PC_1, \dots, PC_5 are calculated. The coefficients (loadings) are given below:

	PC1	PC2	PC3	PC4	PC5
X1	0.4846410	0.02535008	-0.64416209	0.4756905738	-0.35107564
X2	0.4967931	0.04064938	-0.05731951	-0.8099920823	-0.30359769
X3	0.4659406	0.24895000	0.74092022	0.3366118674	-0.24218360
X4	0.5001321	0.13595192	-0.08208782	0.0003119937	0.85126166
X5	0.2260240	-0.95772959	0.16145740	0.0657545280	0.03570757

The following linear regression models (along with error sum of squares SSE) are fit:

Model	SSE
$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$	2287.042
$Y = \beta_0 + \beta_1 PC_1 + \epsilon$	10264.414
$Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \epsilon$	2150.351
$Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \beta_3 PC_3 + \epsilon$	2150.351

- (a) Which model is preferred based on the BIC score (use form $BIC = n \log(SSE/n) + C$)?
- (b) Suppose the predictor is to take the form of a linear combination of the five feature variables, plus an intercept term. One of the three strategies is to be used:
 - (i) Use whichever linear combination minimizes the SSE .
 - (ii) Use only the unweighted mean $\bar{X} = (X_1 + \dots + X_5)/5$ of the feature variables, that is $Y \approx \beta_0 + \beta_1 \bar{X}$.
 - (iii) Use the predictor of part (ii) but add one more feature, say, $m \in \{1, 2, 3, 4, 5\}$, that is $Y \approx \beta_0 + \beta_1 \bar{X} + \beta_2 X_m$.

Based on your BIC analysis, and the principal component loadings, which of the three strategies seems preferable?

SOLUTION:

- (a) The formula is

$$BIC = n \log(SSE/n) + \log(n)k,$$

where k is the number of parameters. We can construct table:

Model	SSE	k	BIC
1 $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$	2287.042	6	340.615
2 $Y = \beta_0 + \beta_1 PC_1 + \epsilon$	10264.414	2	472.337
3 $Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \epsilon$	2150.351	3	320.637
4 $Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \beta_3 PC_3 + \epsilon$	2150.351	4	325.242

Model 3 $Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \epsilon$ is selected by the BIC score.

- (b) For PC_1 the loadings are nearly equal for X_1, X_2, X_3, X_4 . On the other hand, PC_2 is dominated by X_5 . The best BIC model is based on PC_1, PC_2 , and most closely resembles strategy (iii) with $X_m = X_5$.

Problem 30.13. We are given a multiple regression model, with sample size $n = 17$:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon.$$

Suppose we are given the following error sums of squares SSE for the model, and all reduced models:

MODEL		SSE
M_0	$y = \beta_0 + \epsilon$	1,152,144.09
M_1	$y = \beta_0 + \beta_1 x_1 + \epsilon$	19,874.28
M_2	$y = \beta_0 + \beta_2 x_2 + \epsilon$	14,783.91
M_{1+2}	$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$	12,762.61
$M_{1\times 2}$	$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon$	12,291.50

Which model should be selected using the Bayesian information criterion (BIC)? Use the form $BIC = -2LL + C$, where LL is the log-likelihood function.

SOLUTION:

The BIC score is

$$BIC = -2LL + \log(n)d$$

where LL is the log-likelihood

$$LL = -\frac{n}{2} \log(SSE/n)$$

and d is the number of unknown parameters. If p is the number of model predictors (here, ranging from 0 to 3) it would be acceptable to set $d = p$, $p + 1$ or $p + 2$, the largest number incorporating parameters β_0, σ^2 . However, any choice will yield exactly the same rankings. Here, we'll take $d = p$. This gives

$$BIC = n \log(SSE/n) + \log(n)p.$$

Using

$$R_{adj}^2 = 1 - \frac{SSE/(n-p-1)}{SSTO/(n-1)}$$

will also be accepted. Here, $SSTO$ equals SSE for the null model M_0 . This gives calculations:

	Model	SSE	p	BIC	R_{adj}^2
1	M_0	1,152,144.09	1	189.1067	0.0000
2	M_1	19,874.28	2	122.9207	0.9816
3	M_2	14,783.91	2	117.8906	0.9863
4	M_{1+2}	12,762.61	3	118.2245	0.9873
5	$M_{1\times 2}$	12,291.50	4	120.4183	0.9869

Model M_2 is selected by the BIC score, while model M_{1+2} is selected by R^2_{adj} .

Problem 30.14. We wish to fit the model

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n, \quad (30.4)$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent error terms, and x_i is a predictor variable. We set

$$g(x) = \begin{cases} a_1x^3 + b_1x^2 + c_1x + d_1 & ; \quad x < \xi \\ a_2x^3 + b_2x^2 + c_2x + d_2 & ; \quad x \geq \xi \end{cases}, \quad (30.5)$$

where ξ is fixed, and the polynomial coefficients $a_j, b_j, c_j, d_j, j = 1, 2$ are to be estimated. However, the coefficients must be constrained so that $g(x)$ is continuous, and possesses continuous first and second derivatives, at ξ .

- (a) Give precisely the linear constraints on the coefficients required for the given continuity conditions. How can these constraints be used to determine the model degrees of freedom?
- (b) Show that the model

$$y_i = \beta_0 + \beta_1x_i + \beta_2x_i^2 + \beta_3x_i^3 + \beta_4(x_i - \xi)_+^3 + \epsilon_i, \quad i = 1, \dots, n, \quad (30.6)$$

is equivalent to (30.4)-(30.5). HINT: First show this is true for $\xi = 0$.

SOLUTION:

- (a) Equate the polynomial sections, and their first and second derivatives, at $x = \xi$:

$$\begin{aligned} a_1\xi^3 + b_1\xi^2 + c_1\xi + d_1 &= a_2\xi^3 + b_2\xi^2 + c_2\xi + d_2, \\ 3a_1\xi^2 + 2b_1\xi + c_1 &= 3a_2\xi^2 + 2b_2\xi + c_2, \\ 6a_1\xi + 2b_1 &= 6a_2\xi + 2b_2. \end{aligned}$$

There are 8 parameters, with 3 constraints. This leaves $8 - 3 = 5$ model degrees of freedom.

- (b) First, set $\xi = 0$. The constraints then become equivalent to:

$$\begin{aligned} d_1 &= d_2, \\ c_1 &= c_2, \\ b_1 &= b_2. \end{aligned}$$

Therefore, the coefficients associated with the 1, x and x^2 are the same on both sides of the knot, so we get the equivalent model

$$y_i = \beta_0 + \beta_1x_i + \beta_2x_i^2 + \beta_3x_i^3 + \beta_4(x_i)_+^3 + \epsilon_i, \quad i = 1, \dots, n.$$

Then consider the transformation $u_i = x_i + \xi$. Substituting $x_i = u_i - \xi$ gives the more general model.

30.2 Data Analysis

Problem 30.15. For this problem use data set `Cars93` from the `MASS` package. There is data on 93 makes of automobile, including `Manufacturer`, `Model`, `Type`, `Origin`, and miscellaneous technical data (`Wheelbase`, `RPM`, etc). The purpose of this analysis is to show how hierarchical clustering may be useful in exploratory analysis.

- (a) Select the features for this analysis using the following column indices:

```
xf = Cars93[,c(5,7,8,12,13,14,15,17,19,20,21,22,25)]
```

Standardize each column to zero mean and unit variance. Then, create a class vector `gr` from variable `Man.trans.avail`. This identifies whether or not manual transmission is available.

- (b) Using the function `hclust` plot dendograms for hierarchical clusterings using agglomeration methods `single`, `complete` and `average`. Label the observations by `gr`. Do the observations appear to cluster by class `gr` in any of the dendograms?
- (c) There are a number of quantities which may be used to determine whether or not a clustering conforms well to a known class variable. Suppose we create a single clustering of size `c.size` (using `cutree(hfit,k=c.size)`). Suppose `gr` contains exactly two classes. Let α_k be the probability that two observations, one randomly chosen from each class, are in the same cluster, given k clusters. This can be easily estimated by cross-tabulating class and cluster frequencies. Smaller values of α_k suggest that the cluster conforms to the class.
 - (i) Show that $\alpha_{k+1} \leq \alpha_k$ for $k \geq 1$.
 - (ii) Show that α_k approaches 0 as k approaches sample size n .
- (d) For each of the hierarchical clusterings, plot α_k for $k = 1, \dots, 93$ (plot all three on the same graph). Does one agglomeration method have smaller α_k for most cluster sizes k ?
- (e) One way to assess whether or not α_k is significantly small is to use a permutation procedure. Suppose the class vector is randomly permuted. This should eliminate any association between class and cluster. To see this, using the agglomeration method selected in part (d), plot again α_k for $k = 1, \dots, 25$. Then, create a new class vector `gr.perm` by randomly permuting the original class vector `gr` (you can use function `sample()`). Create a new sequence α'_k , $k = 1, \dots, 25$, using the same procedure, except that `gr` is replaced by `gr.perm`. Do the permutation 10 times, superimposing all α_k and α'_k sequences on the same plot. Make sure the sequence types are easily distinguishable (say, use black for α_k and gray for each α'_k). Does the plot suggest that there is a statistically significant association between cluster and class?
- (f) Finally, calculated a LASSO fit using `gr` as response and the feature matrix `xf` (use the `binomial` model). Examine the coefficients for the `fit$lambda.min` solution. Do the selected variables seem related to class? In other words, what type of cars tend to have manual transmission?

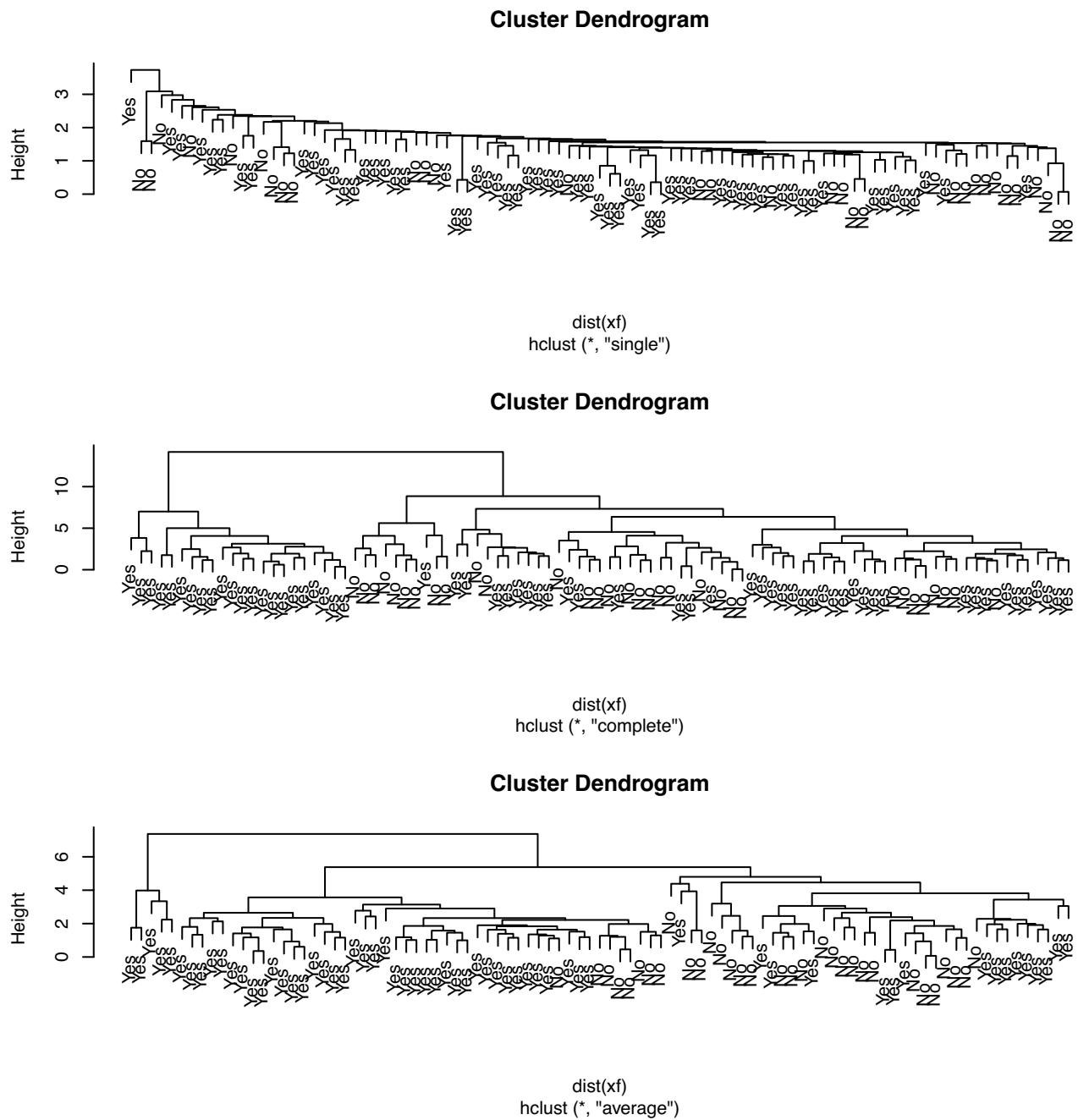


Figure 30.2: Dendograms for Problem 30.15 (b).

SOLUTION:

(a) Use code:

```

library(MASS)
xf = Cars93[,c(5,7,8,12,13,14,15,17,19,20,21,22,25)]
xf = apply(xf,2,function(x) {(x - mean(x))/sd(x)} )
gr = Cars93$Man.trans.avail
gri = as.integer(gr)

```

(b) Use code:

```

par(mfrow=c(3,1))
hfit1 = hclust(dist(xf),method='single')
plot(hfit1,labels=gr)
hfit2 = hclust(dist(xf),method='complete')
plot(hfit2,labels=gr)
hfit3 = hclust(dist(xf),method='average')
plot(hfit3,labels=gr)

```

Clustering by class is evident in all dendograms. See Figure 30.2.

(c) Suppose p_{ij} is the probability that an observation of class i is in cluster j . Then

$$\alpha_k = \sum_{s=1}^k p_{1s} p_{2s}.$$

Let n_1, n_2 be the number of observations of each class. Let n_{ij} be the number of observations of class i in cluster j . Then substitute estimate $p_{ij} \approx n_{ij}/n_i$.

(i) A new cluster is created by splitting one previous cluster. Suppose (without loss of generality) that cluster k is the one split. Then probabilities p_{1k}, p_{2k} are replaced by probabilities $p'_{1,k}, p'_{1,k+1}$ and $p'_{2,k}, p'_{2,k+1}$. Then

$$\alpha_{k+1} = \sum_{s=1}^{k-1} p_{1s} p_{2s} + p'_{1k} p'_{2k} + p'_{1,k+1} p'_{2,k+1}.$$

However, we must also have

$$\begin{aligned} p_{1k} &= p'_{1,k} + p'_{1,k+1} \\ p_{2k} &= p'_{2,k} + p'_{2,k+1}. \end{aligned}$$

Then

$$\begin{aligned} \alpha_k - \alpha_{k+1} &= p_{1k} p_{2k} - p'_{1k} p'_{2k} - p'_{1,k+1} p'_{2,k+1} \\ &= (p'_{1,k} + p'_{1,k+1})(p'_{2,k} + p'_{2,k+1}) - p'_{1k} p'_{2k} - p'_{1,k+1} p'_{2,k+1} \\ &= p'_{1,k} p'_{2,k+1} + p'_{1,k+1} p'_{2,k} \\ &\geq 0, \end{aligned}$$

which completes the proof.

(ii) Eventually, for $k = n$, all clusters contain only one observation. This means for any cluster j either $n_{1j} = 0$ or $n_{2j} = 0$. Using the above method, this forces $\alpha_n = 0$.

(d) Use code:

```
par(mfrow=c(1,1))
l2 = NULL
for (i in 1:25) {
  mm1 = table(gr,cutree(hfit1,k=i))
  pp1 = sum(mm1[1,]*mm1[2,])/prod(table(gr))
  mm2 = table(gr,cutree(hfit2,k=i))
  pp2 = sum(mm2[1,]*mm2[2,])/prod(table(gr))
  mm3 = table(gr,cutree(hfit3,k=i))
  pp3 = sum(mm3[1,]*mm3[2,])/prod(table(gr))
  l2 = rbind(l2, c(pp1,pp2,pp3))
}
matplot(l2,type='l',xlab='cluster size k', ylab='alpha[k]',lwd=2)
legend('topright',legend=c('single','complete','average'),pch=NA,lty=1:3,col=1:3)
```

Method 2 (complete agglomeration) possesses smallest α_k over the largest range of k . See Figure 30.3.

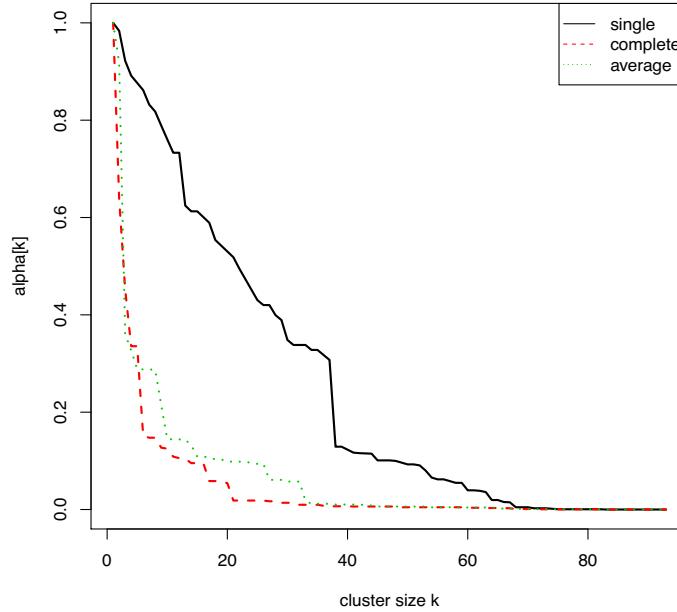


Figure 30.3: Plots of α_k for Problem 30.15 (d).

(e) The following code can be used:

```
# original class vector
l2 = NULL
l2v = NULL
```

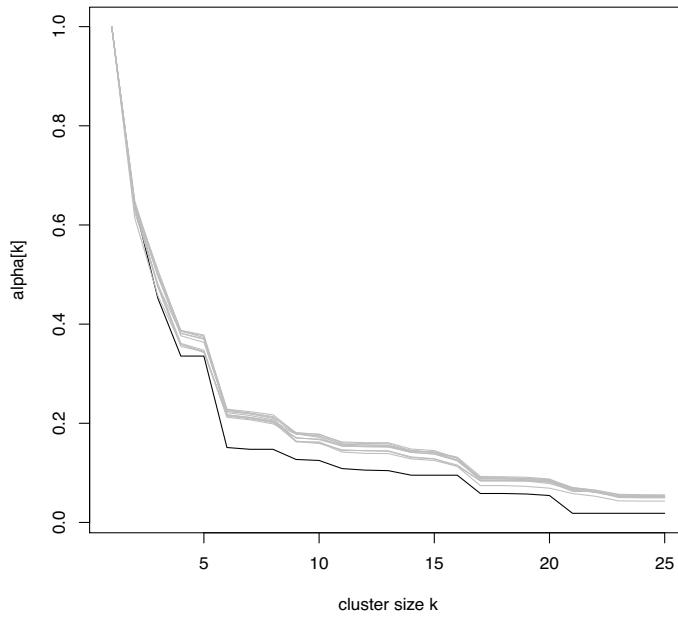


Figure 30.4: Plots of α_k (black), α'_k (gray) for Problem 30.15 (e).

```

for (i in 1:25) {
  mm2 = table(gr,cutree(hfit2,k=i))
  l2v = c(l2v, sum(mm2[1,]*mm2[2,])/prod(table(gr)))
}

# permuted class vector

nperm = 10
l2 = l2v
for (iii in 1:nperm) {
  grp = sample(gr)
  l2v = NULL
  for (i in 1:25) {
    mm2 = table(grp,cutree(hfit2,k=i))
    l2v = c(l2v, sum(mm2[1,]*mm2[2,])/prod(table(grp)))
  }
  l2 = cbind(l2,l2v)
}
matplot(l2, col=c('black',rep('gray',nperm)),lty=1,pch=NULL,type='l')

```

The values α'_k from the permuted class vector fall mostly above the original α_k values. This may be taken as statistical evidence of an association between class and cluster. See Figure 30.4.

(f)

The following code can be used:

```
> y = 1*(gr=='Yes')
> fit = cv.glmnet(xf,y, family='binomial', alpha=1)
> coef = coef(fit, s = fit$lambda.min)
> coef
14 x 1 sparse Matrix of class "dgCMatrix"
           1
(Intercept) 1.3151061
Price         .
MPG.city      .
MPG.highway   .
EngineSize    .
Horsepower    .
RPM          .
Rev.per.mile  0.6072918
Fuel.tank.capacity .
Length        -1.0850563
Wheelbase     -0.7461718
Width         .
Turn.circle   -0.5325136
Weight        .
>
```

There are three selected features `Length`, `Wheelbase` and `Turn.circle` which are all directly related to car size. Also, the coefficients are negative. This means smaller cars are more likely to offer manual transmission.

Problem 30.16. Consider the `crabs` data set from library `MASS`, and in particular the three variables:

```
crab$sp  = species - "B" or "O" for blue or orange
crab$RW  = rear width (mm)
crab$BD  = body depth (mm).
```

We are interested in determining the functional relationship between response $Y = RW$ and predictor $X = BD$.

- (a) Create a data set with columns RW , BD using only species ‘B’.
- (b) Fit a simple linear regression model

$$Y = \beta_0 + \beta_1 X + \epsilon. \quad (30.7)$$

Examine a residual vs fit plot (you can use the generic function `plot` with option `which = 1`). Does the linear model seem appropriate? What is the predicted value of Y for $X = 0$? Does this make sense?

- (c) Suppose we force the estimated function to pass through the origin, by fitting model

$$Y = \beta_1 X + \epsilon. \quad (30.8)$$

Examine a residual vs fit plot. Does this model seem appropriate?

- (d) Suppose we consider model

$$Y = \alpha X^\tau + \epsilon. \quad (30.9)$$

Use the estimated coefficients from model

$$\log(Y) = \beta_0 + \beta_1 \log(X) + \epsilon' \quad (30.10)$$

to estimate α and τ , where ϵ' is a suitably transformed error term. Perform a formal hypothesis test for which the null hypothesis is linearity, that is, that model (30.8) is correct, against the alternative model (30.9).

- (e) Plot the original untransformed data, and superimpose the fitted models (30.7), (30.8), (30.9). In addition, fit a smoothing spline with 4 degrees of freedom, and superimpose this fit as well. Comment on the ability of each model to accurately predict Y at the model extremes, around $X < 8$ and $X > 18$. Which model would be most useful for predicting W outside the range $X \in [6, 20]$?

SOLUTION:

- (a) Use index subsetting:

```
x = crabs[crabs$sp=="B",]$BD
y = crabs[crabs$sp=="B",]$RW
```

- (b) Use the following commands. The residual plot (Figure 30.5) suggests significant curvature, so that modeling a linear relationship between y and x is not appropriate. From the output below, the estimated coefficients are $\hat{\beta}_0 = 3.47492$ and $\hat{\beta}_1 = 0.67179$. So the fitted value at $X = 0$ is $\hat{\beta}_0 + \hat{\beta}_1 X = \hat{\beta}_0 = 3.47492$, with a standard error of $S_{\hat{\beta}_0} = 0.41501$, so that the intercept is significantly greater than 0, which is not what we would expect.

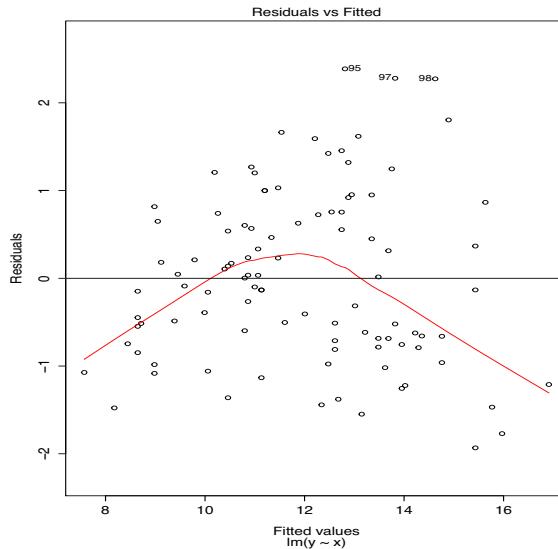


Figure 30.5: Plot for Problem 30.16 (b).

```

> fit1 = lm(y~x)
> summary(fit1)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.93271 -0.71994 -0.09354  0.72853  2.38726 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.47492   0.41501  8.373 4.03e-13 ***
x           0.67179   0.03205 20.959 < 2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9784 on 98 degrees of freedom
Multiple R-squared:  0.8176, Adjusted R-squared:  0.8157 
F-statistic: 439.3 on 1 and 98 DF,  p-value: < 2.2e-16

```

```
> plot(fit1,which=1)
```

- (c) The intercept is removed from the model by including -1 in the formula. We have estimate $\hat{\beta}_1 = 0.932597$. Even more than for model (30.7), the model defined in (30.8) is clearly inappropriate, since functional structural remains in the residual plot (Figure 30.6).

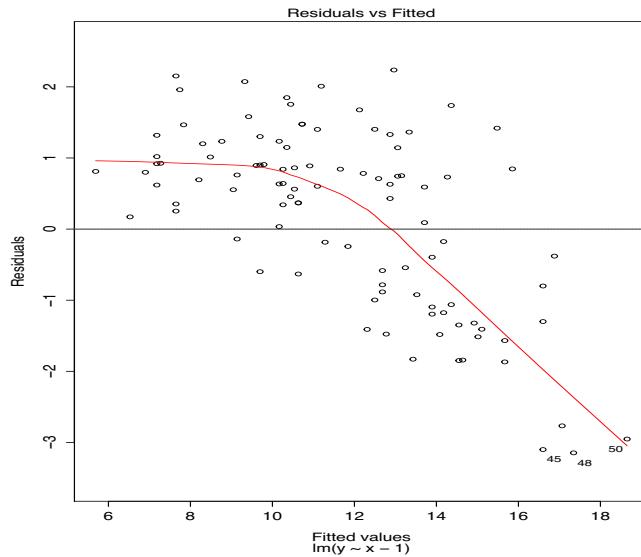


Figure 30.6: Plot for Problem 30.16 (c).

```

> fit2 = lm(y~x-1)
> summary(fit2)

Call:
lm(formula = y ~ x - 1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.1463 -0.7875  0.6106  1.0502  2.2369 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
x  0.932597   0.009847   94.71   <2e-16 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.275 on 99 degrees of freedom
Multiple R-squared:  0.9891, Adjusted R-squared:  0.989 
F-statistic: 8970 on 1 and 99 DF,  p-value: < 2.2e-16

> plot(fit2,which=1)
> abline(h=0)

```

- (d) The transformation can be directly incorporated into the `lm()` function as shown below. The hypothesis of linearity is equivalent to $H_0 : \tau = 1$, where after the transformation $\beta_1 = \tau$. Normally, we test against null hypothesis $H_0 : \beta_1 = 0$, but there is no reason we can't test

against null hypothesis $H_0 : \beta_1 = 1$. To do so, we use t -statistic

$$T = \frac{\hat{\beta}_1 - 1}{S_{\hat{\beta}_1}} = \frac{0.73151 - 1}{0.03005} = -8.934775.$$

The sample size is n , so the degrees of freedom for the t -statistics is $n - 2 = 98$. P-value is

```
> 2*pt(-8.934775,df=98)
[1] 2.476081e-14
```

so we reject $H_0 : \tau = 1$, and linearity, at a significance level of $\alpha = 0.05$ (and much lower significance levels).

```
> fit3 = lm(log(y)~log(x))
> summary(fit3)

Call:
lm(formula = log(y) ~ log(x))

Residuals:
    Min      1Q  Median      3Q     Max 
-0.151254 -0.057241 -0.000531  0.054963  0.166132 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.62990   0.07554   8.339 4.77e-13 ***
log(x)      0.73151   0.03005  24.346 < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07664 on 98 degrees of freedom
Multiple R-squared:  0.8581, Adjusted R-squared:  0.8567 
F-statistic: 592.7 on 1 and 98 DF,  p-value: < 2.2e-16

>
```

- (e) The following code draws the required plot (Figure 30.7). Model (30.7) overestimates $E[Y]$ for $X < 8$, but seems to work well for $X > 18$. Model (30.8) underestimates $E[Y]$ for $X < 8$ and overestimates $E[Y]$ for $X > 18$. The remaining models work well at both extremes. The spline, however, becomes more variable at high values $X > 18$. Because model (30.9) has a single analytical form, it is suitable for making predictions outside the observed range, especially since it is constrained to pass through the origin. The spline is defined piecewise, so it would be difficult to extrapolate outside the observed range.

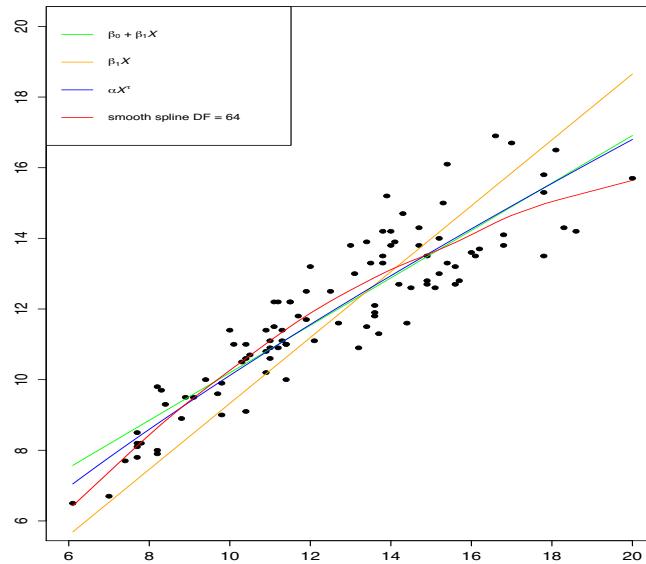


Figure 30.7: Plot for Problem 30.16 (e).

```

par(mfrow=c(1,1))
xgrid = seq(min(x),max(x),by=0.1)
plot(x,y,xlim=c(6,20),ylim=c(6,20),pch=19)

# model (3)
lines(xgrid,predict(fit1,newdata=list(x=xgrid)),col='green')

# model (4)
lines(xgrid,predict(fit2,newdata=list(x=xgrid)),col='orange')

# model (5)
lines(xgrid,exp(predict(fit3,newdata=list(x=xgrid))),col='blue')

# smooth spline
fit4 = smooth.spline(x,y,df=6)
lines(fit4,col='red')

ex1 = expression(paste(beta[0]," + ", beta[1], italic(X),sep=''))
ex2 = expression(paste(beta[1], italic(X),sep=''))
ex3 = expression(paste(alpha, italic(X)^tau ,sep=''))

legend('topleft',legend=c(ex1,ex2,ex3,'smooth spline DF = 6'),

```

```
col=c('green','orange','blue','red'),lty=1,cex=0.8)
```

Problem 30.17. This problem will make use of the `biopsy` data set from the `MASS` library. See `help(biopsy)` for details.

- (a) Prepare the data by first removing the `ID` column, then removing records with missing values using the `na.omit()` function. The new data set should have $n = 683$ records. Column 10 is now the `class` variable, containing the tumor class (`benign` or `malignant`). Columns 1-9 now contain quantitative tumor features with which to discriminate between tumor types. In this analysis, instead of normalizing the features to zero mean and unit variance, subject each feature to a log transformation (use the natural logarithm).
- (b) Calculate K -means cluster solutions based on the 9-log transformed features. Use $K = 1, \dots, 25$. For each K calculate $R^2 = 1 - SS_{within}/SS_{total}$, then plot R^2 against K . Between which two values of K does the greatest increase in R^2 occur? How does this relate to the true number of clusters?
- (c) Calculate the principal components of the 9 log transformed features using the `prcomp()` function. Use centering but not scaling, that is, use options `center=T` and `scale.=F`. Create a pairwise plot (using function `pairs()`) for the first 4 principal components, using separate coloring for each class (the classes need not be labeled). Then create a scree plot. This can be done using the command `plot(pr.fit)`, where `pr.fit` is the principal components object created by the `prcomp()` function. How do the various plots suggest that most of the discriminating information regarding tumor type is contained by the first principal component?
- (d) Finally, calculate a LASSO fit using response $Y_i = 1$ if `class` is `malignant` and $Y_i = 0$ otherwise (use the `binomial` model). Use cross-validation with function `cv.glmnet()` and options `family='binomial'` and `alpha=1`. Do this using the original log-transformed features as predictors, then using the 9 principal components calculated in Part (c) as predictors. Examine the coefficients for the `fit$lambda.1se` solution for each set of predictors. Do these conform to what you see in Part (c)? (Note that since cross-validation is random, repeated fits will yield different coefficients. However, the overall conclusion should be the same).

SOLUTION:

The following code may be used for the analysis of Parts (a)-(c). Comments follow.

```
library(MASS)
library(glmnet)

### (a)

biopsy2 = biopsy[,-1]
biopsy2 = na.omit(biopsy2)
xf = biopsy2[,1:9]
xf = apply(xf,2,function(x) {log(x)})
```

```

#### (b)

set.seed(13579)
r2 = rep(NA, 10)
for (i in 1:20) {
  fit = kmeans(xf, centers=i, nstart=100)
  r2[i] = fit$betweenss/fit$totss
}

pdf('A4fig4.pdf')
plot(r2, ylim=c(0,1), xlab='K', ylab='R-squared', type='b', pch=20)
dev.off()

#### (c)

pr.fit = prcomp(xf, center=T, scale.=F)
pdf('A4fig5.pdf')
pairs(pr.fit$x[,1:4], col=1+(biopsy2$class=='malignant'), pch=3)
dev.off()

pdf('A4fig6.pdf')
plot(pr.fit)
dev.off()

```

- (a) See code.
 (b) The greatest increase in K clearly occurs between $K = 1$ and $K = 2$ (Figure 30.8). This is what we would expect to see if the true number of clusters was $K = 2$.

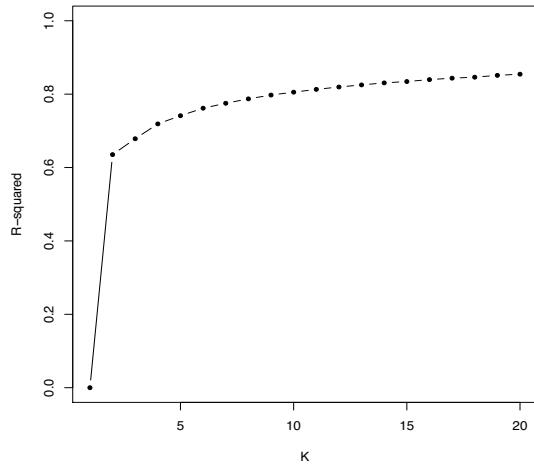


Figure 30.8: Plot for Problem 30.17 (b)

- (c) In any plot including the first principal component (PC1) the two tumor classes clearly separate with little overlap, with a boundary definable as a fixed value of PC1 (Figure 30.9). In addition, the scree plot indicates that most variation is contained in the first principal component (Figure 30.10). This indicates that most of the discriminating information regarding tumor type is contained in the first principal component.

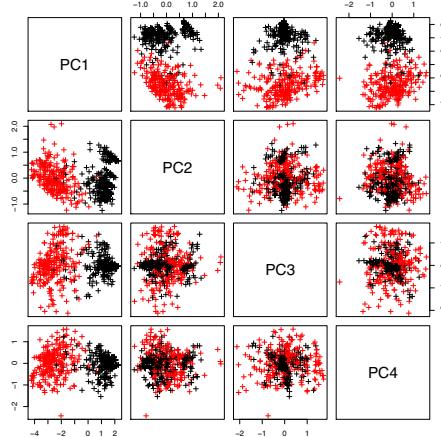


Figure 30.9: Plot for Problem 30.17 (c) (pairwise principal component plots)

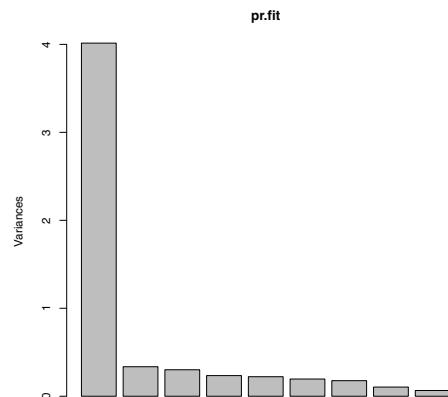


Figure 30.10: Plot for Problem 30.17 (c) (scree plot)

- (d) See the following code. When using the original log-transformed features, the coefficient values range from 0.1048410 to 1.0555343. All features have been selected, in the sense that no estimated coefficient is zero. On the other hand, when using principal components, only the first is selected. This reinforces the conclusion of Part (c), that most of the discriminating information is contained in the first principal component. (Note that since cross-validation is random, repeated fits will yield different coefficients. However, the overall conclusion should

be the same).

```
>
> ### (d)
>
> y = 1*(biopsy2$class=='malignant')
> fit = cv.glmnet(xf,y, family='binomial', alpha=1)
> coef = coef(fit, s = fit$lambda.1se)
> coef
10 x 1 sparse Matrix of class "dgCMatrix"
   1
(Intercept) -5.1130697
V1           0.5533361
V2           0.7405122
V3           0.8847965
V4           0.1544626
V5           0.1048410
V6           1.0555343
V7           0.6348426
V8           0.2876825
V9           0.1596633
>
>
> y = 1*(biopsy2$class=='malignant')
> fit = cv.glmnet(pr.fit$x,y, family='binomial', alpha=1)
> coef = coef(fit, s = fit$lambda.1se)
> coef
10 x 1 sparse Matrix of class "dgCMatrix"
   1
(Intercept) -1.342705
PC1         -1.767733
PC2          .
PC3          .
PC4          .
PC5          .
PC6          .
PC7          .
PC8          .
PC9          .
```

Problem 30.18. For this problem use data set `biopsy` from the `MASS` package. Biopsies of 699 breast tumors were examined and classified as `benign` or `malignant`. Nine attributes (features)

were scored on a scale of 1 to 10, higher scores signifying evidence that the tumour is malignant (use `help(biopsy)` for details).

We consider how to consolidate the features into a single quantitative predictor for malignancy. The simplest method is to sum the features to yield a single score S on a scale of 9 to 90, which the use of a 10 point scale suggests. This would be equivalent to a linear model in which all coefficients β_i are equal, other than the intercept. On the other hand, it is always possible that a combination of model selection and unequal coefficients will yield a strictly better predictor.

- (a) Remove observations with missing values using the `na.omit` function. There should be $n = 683$ observations remaining. The 9 biopsy features can then be used to create 683×9 feature matrix X . Label the i th feature (column of X) F_i .
- (b) Calculate the principal components of the 9 features. Set option `center = TRUE` and `scale. = FALSE`. That is, we don't rescale the original 10 point scale. The principal components are given by

$$P = \bar{X}A$$

where \bar{X} is the $n \times 9$ feature matrix after centering by column (ie. by subtracting the column means), P is the $n \times 9$ matrix of principal components, and A is the 9×9 matrix of loadings. So, for example, the first principal component is a linear combination of centered features, with coefficients given by the first column of A . Label the i th centered feature (column of \bar{X}) \bar{F}_i . Then $\bar{F}_i = F_i - m_i$, where m_i is an $n \times 1$ constant column vector, with all elements equal to the mean of F_i .

- (c) Create pairwise plots using all principal components. Use separate colors for the `benign` and `malignant` classes. Create a *scree* plot. Comment on what you see.
- (d) Examine the loadings matrix A .
 - (i) How does the 1st principal component resemble sum S ?
 - (ii) What feature has the largest (in magnitude) loading for the 2nd principal component?
 - (iii) What feature has the largest (in magnitude) loading for the 3rd principal component?
- (e) A principal components analysis can be insightful even if the principal components are not explicitly used in an analysis. For example, let B be an 9×9 matrix which has all diagonal entries equal to 1, and all other entries equal to 0, except for column 9, in which all entries equal $1/9$. Create new $n \times 9$ feature matrix W :

$$W = XB$$

Based on your answers to part (d), which columns of W correspond approximately to the first 3 principal components?

- (f) Using function `cv.glmnet()` create a LASSO fit using W as feature matrix and `class` as a binary response. This means setting option `family='binomial'` and `alpha=1` (why?). Other options can use default settings. Give the coefficients for the model corresponding to the smoothing parameter λ set to `lambda.1se` (what does this mean?). How is this solution related, approximately, to the principal components?
- (g) Create the same type of LASSO fit as for part (f), using the original feature matrix X (this need not be centered). Which features have nonzero coefficients? Calculate the predicted values for the fits using W and X as the feature matrix (these need not be transformed by the logistic function). Create a scatter plot of the predicted values (each point represents the

pair of predicted values for each observation). Superimpose the identity using `abline(0,1)`. Also, for each fit, use the response/predicted value pair to calculate the AUC statistic for the ROC curve. How similar are the two predictive models?

- (h) If two predictive models are similar, we might prefer the simpler or more interpretable model. Consider the model of part (f), based on feature matrix W . We can set $\bar{S} = S/9$, the average of the features on the original 10 point scale.
- Interpret the coefficients of the model of part (f). What role does \bar{S} play?
 - Identify all observations for which at least one feature F_i has score 10. How many of each class are among these?
 - Create a list of 9 vectors, where the i th vector is the subset of all values \bar{S} for observations with $F_i = 10$. Construct a single side-by-side boxplot using these vectors. Superimpose horizontal lines at the 25th, 50th, 75th percentiles of \bar{S} separately for each class (clearly indicate the classes). How does this plot explain the feature selection of the model of part (f)?

SOLUTION:

- (a) Use code

```
> library(MASS)
> library ( glmnet )
> bio2 = na.omit(biopsy)
> gr = 1*(bio2$class=='malignant')
> xdata = as.matrix(bio2[,2:10])
> dim(biopsy)
[1] 699 11
> dim(bio2)
[1] 683 11
```

- (b) Use command

```
pc1 = prcomp(xdata, center = TRUE, scale. = F)
```

- (c) Label the i th principal component PC_i . Use the following commands. From Figure 30.11, only PC_1 appears able to separate the two classes. From the scree plot of Figure 30.12, the variation of the PC_1 is considerably larger than for the remaining principal components. This suggests that PC_1 is by far the most important.

```
pairs(pc1$x,col=1+gr)
plot(pc1)
```

- (d) The matrix A is the rotation matrix, given by the command below.

```
> round(pc1$rotation,4)
    PC1      PC2      PC3      PC4      PC5      PC6      PC7      PC8      PC9
V1 -0.2967 -0.0735  0.8520 -0.0471  0.3998 -0.0975 -0.0361 -0.0880  0.0064
```

V2	-0.4040	0.2299	0.0263	-0.2946	-0.3380	-0.0924	0.1750	0.1305	0.7253
V3	-0.3928	0.1647	0.0745	-0.2065	-0.3702	-0.0873	0.3469	0.2423	-0.6684
V4	-0.3312	-0.0982	-0.4739	-0.3942	0.6425	-0.1898	0.1838	-0.1277	-0.0425
V5	-0.2497	0.2002	-0.0317	-0.1888	-0.1951	0.4014	-0.2835	-0.7539	-0.1139
V6	-0.4426	-0.7806	-0.0934	0.2944	-0.1687	0.2530	0.0537	0.0184	0.0613
V7	-0.2921	0.0085	-0.1224	0.0412	-0.0926	-0.4631	-0.7960	0.1837	-0.0874
V8	-0.3545	0.4692	-0.1337	0.7539	0.1969	-0.0042	0.1578	-0.0580	0.0285
V9	-0.1246	0.1881	-0.0266	-0.1431	0.2496	0.7062	-0.2689	0.5425	-0.0126

- (i) The loadings for PC_1 range from -0.4426 to -0.1246. This gives

$$\begin{aligned} PC_1 &= -0.2967 \times \bar{F}_1 + \dots + -0.1246 \times \bar{F}_9 \\ &= -0.2967 \times F_1 + \dots + -0.1246 \times F_9 + C \end{aligned}$$

where C is a $n \times 1$ column vector of constant elements. The coefficients for PC_1 are all of the same sign, and vary much less than for the remaining principal components. Then, very roughly, we have

$$PC_1 \approx aS + b$$

for two constants a, b .

- (ii) For PC_2 the largest coefficient in magnitude is -0.7806 for \bar{F}_6 .
- (iii) For PC_3 the largest coefficient in magnitude is 0.8520 for \bar{F}_1 .
- (e) Two predictors can be regarded as equivalent if one can be obtained as a linear combination of the other. The 9th column of W is $\bar{S} = S/9$, and so roughly corresponds to PC_1 . The 6th column is F_6 , and so roughly corresponds to PC_2 . The 1st column is F_1 , and so roughly corresponds to PC_3 .
- (f) The following code can be used.

```
> B = diag(9)
> B[,9]=1/9
> wdata = xdata %*% B
>
> set.seed(789)
> fit = cv.glmnet(wdata,gr, family='binomial', alpha=1)
> coefw = coef(fit, s = fit$lambda.1se)
> predw = predict(fit, s = fit$lambda.1se, newx=wdata)
> coefw
10 x 1 sparse Matrix of class "dgCMatrix"
   1
(Intercept) -6.0048115
V1           0.1071590
V2           .
V3           .
V4           .
V5           .
V6           0.1210325
```

```
V7      .
V8      .
V9      1.3164962
>
```

Because the response is binary, we use what is essentially a logistic regression model. That is

$$P(Y = 1) = (1 + \exp(-\eta))^{-1}$$

where η is a linear combination of predictors with coefficients β_i . This model required option `family='binomial'`.

A class of *shrinkage methods* for linear models is based on minimizing:

$$SSE + \lambda \sum_{j=1}^p |\beta_j|^d$$

for some tuning parameter λ , typically selected by cross-validation. For logistic regression, or other linear models, SSE can be replaced by deviance, or other goodness-of-fit measure. Ridge regression uses $d = 2$, and LASSO uses $d = 1$. The option `alpha` is essentially a mixture parameter, with `alpha = 1` defining LASSO and `alpha = 0` defining ridge regression. Values between 0 and 1 give a linear combination of the respective penalty terms (see `help(glmnet)`). The value `lambda.1se` gives the largest value of λ such that prediction error is within 1 standard error of the minimum.

Examining the coefficients, we can see that columns 1,6,9 of feature matrix W were selected. As discussed, these correspond approximately to PC_3, PC_2, PC_1 respectively, through F_1, F_6, S .

- (g) The following subroutine can be used to calculate the AUC statistics (or use, for example, library `ROCR`).

```
roc.area<-function(y,gr) {
  y0 = y[gr==0]
  y1 = y[gr==1]
  count<-0
  for (i in 1:length(y0)) {count = count+sum(y1 > y0[i]) + 0.5*sum(y1 == y0[i])}
  ans = count/(length(y0)*length(y1))
  return(ans)
}
```

The following code completes the calculations

```
> # calculate fit for feature matrix X
>
> fit = cv.glmnet(xdata,gr, family='binomial', alpha=1)
> predx = predict(fit, s = fit$lambda.1se, newx=xdata)
>
> # compare predicted values
>
> plot(predw,predx,xlab='feature matrix = W',ylab = 'feature matrix = X')
```

```

> abline(0,1)
>
> # calculate AUC statistic
>
> apply(cbind(predw,predx),2,function(x) {roc.area(x,gr)})
      1         1
0.9959808 0.9959384
>

```

From Figure 30.13 it can be seen that the predicted values of each model are highly correlated. In addition, $AUC = 0.9959808, 0.9959384$ for feature matrices W, X respectively. Overall, the two predictive models are very similar.

- (h) (i) Other than the intercept term, the predictor is dominated by \bar{S} , which is column 9 of W . But there are smaller contributions from features F_1 and F_6 (recall that the range of each column of W is $[1,10]$).
- (ii) The following code may be used.

```

> max.score = apply(xdata,1,max)
> table(gr[max.score==10])

 0   1
 5 204
>

```

Of all observations with $F_i = 10$ for at least one feature F_i , 204/209 have malignant tumors.

- (iii) Figure 30.14 is created by the following code:

```

ex1 = expression(bar(italic(S)))
mx = NULL
for (i in 1:9) {mx = append(mx, list(wdata[,9][xdata[,i]==10 & gr==1]))}
boxplot(mx,ylim=c(1,10),names = paste('F',1:9,sep=''),ylab=ex1)
abline(h=quantile(wdata[gr==1,9],c(0.25,0.5,0.75)),col='grey')
abline(h=quantile(wdata[gr==0,9],c(0.25,0.5,0.75)),col='grey',lty=2)

```

First note that $F_i = 10$ for any feature is strong evidence for malignant tumors. From Figure 30.14, it can be seen that the values of \bar{S} for observations with $F_1 = 10$ or $F_6 = 10$ tend to be smaller than for the other features, and are therefore less likely to be classified as malignant based on \bar{S} alone. The predictor therefore includes a small upward adjustment of malignancy risk based on F_1 and F_6 .

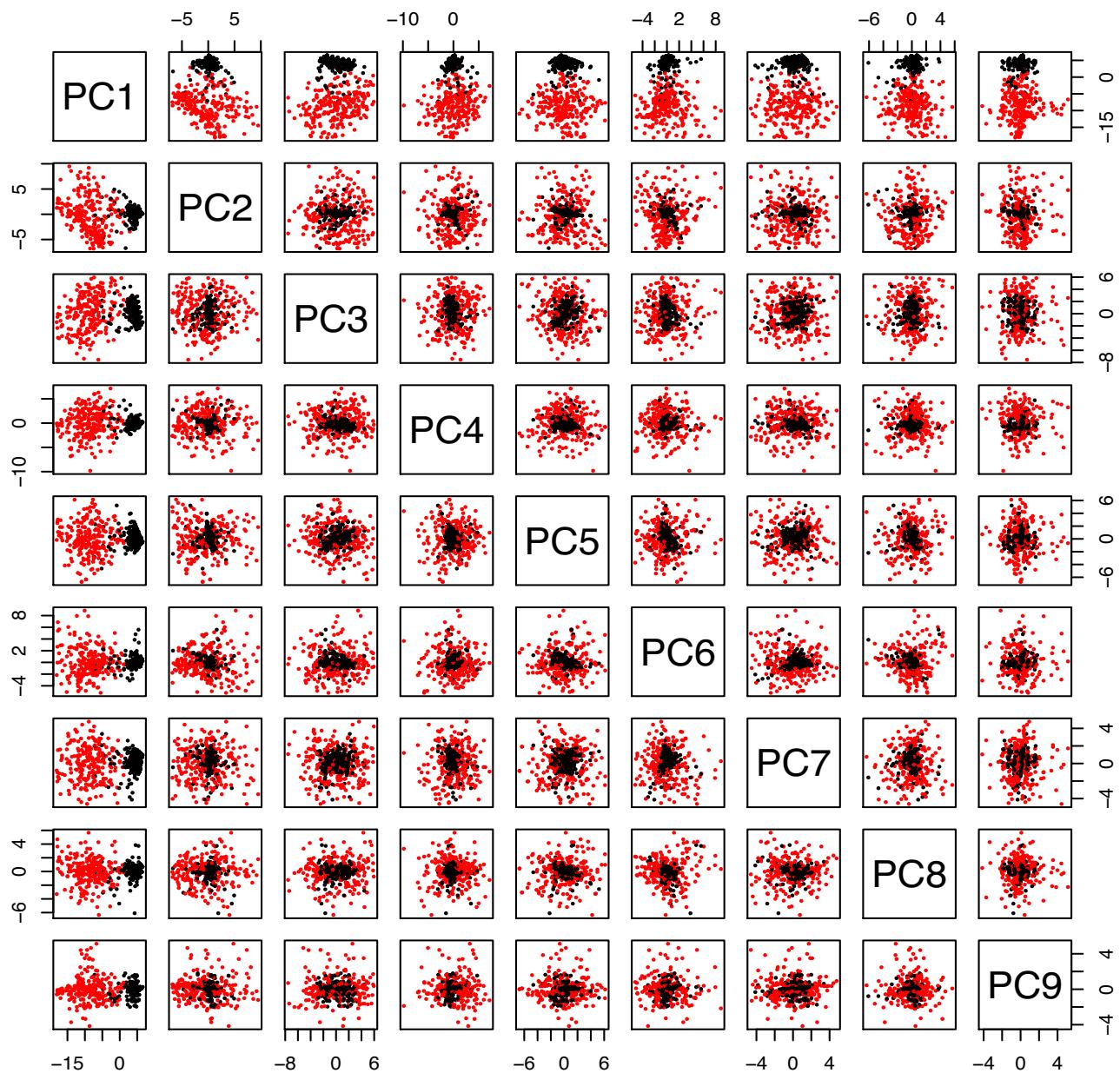


Figure 30.11: Pairwise plots of principal components for Problem 30.18 (c). Red points are malignant tumors, black points are benign tumors.

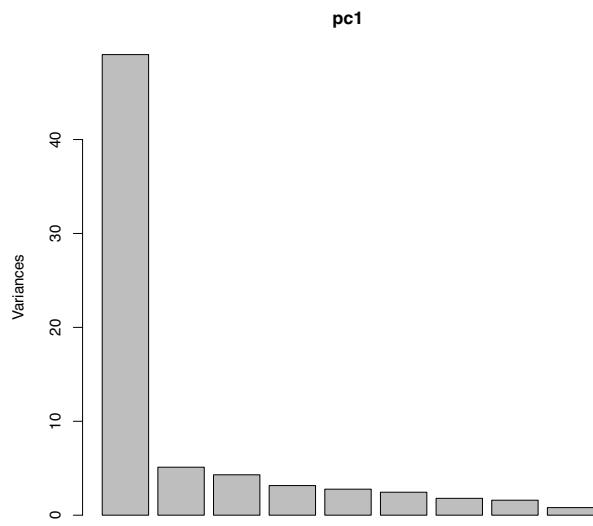


Figure 30.12: Scree plot of principal components for Problem 30.18 (c).

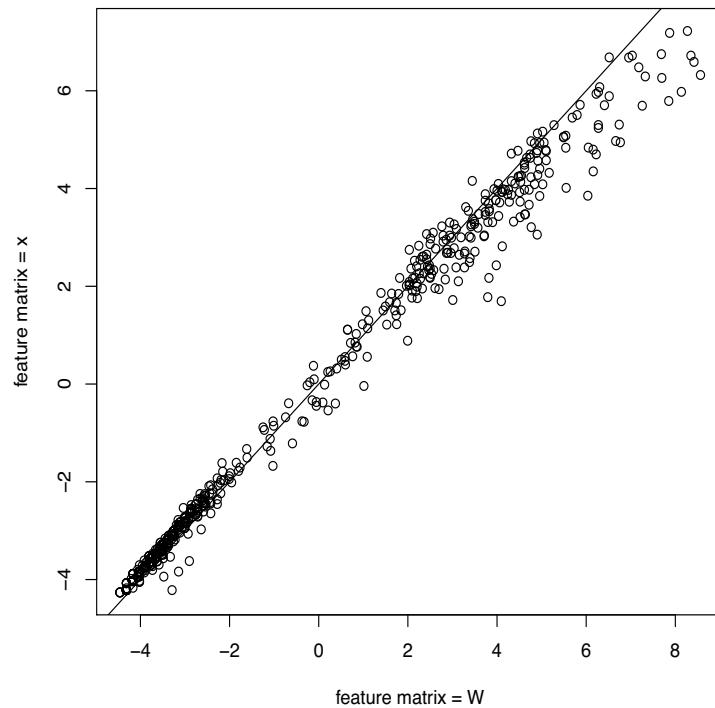


Figure 30.13: Scatter plot of predicted values for Problem 30.18 (g).

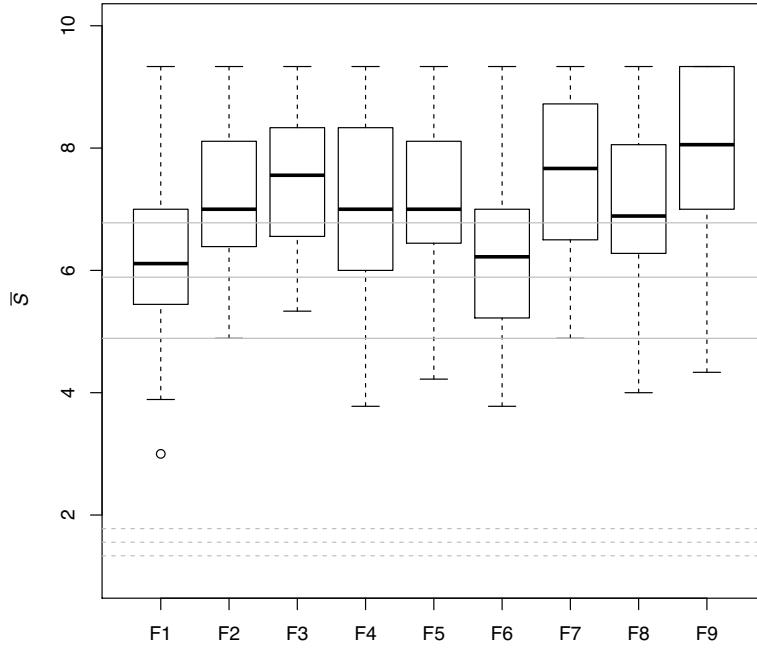


Figure 30.14: Boxplots, for Problem 30.18 (h)(iii), are constructed from values of \bar{S} for which the indicated feature equals 10. Solid (dashed) lines give 25th, 50th, 75th percentiles of \bar{S} for malignant (benign) tumors.

30.3 Theoretical Complements

Problem 30.19. Suppose ϕ is a true density function on \mathbb{R} .

- (a) For $a \in \mathbb{R}$, $b > 0$, show that $\phi_{a,b}(x) = b^{-1}\phi((x - a)/b)$ is also a density function on \mathbb{R} .
- (b) Let x_1, \dots, x_n be any numbers, and let $h > 0$. Verify that f_h defined as

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^n \phi\left(\frac{x - x_i}{h}\right)$$

is a density function. Then f_h is an approximation of the density from which x_1, \dots, x_n were sampled.

- (c) Let (x_i, y_i) be paired observations from model

$$y_i = g(x_i) + \epsilon_i, \quad i = 1, \dots, n, \tag{30.11}$$

where the ϵ_i are any zero mean error terms. The Nadaraya - Watson kernel regression estimate $\hat{y}_h(x) \approx g(x)$ is given by

$$\hat{y}_h(x) = \frac{\sum_{i=1}^n y_i K_h(x - x_i)}{\sum_{i=1}^n K_h(x - x_i)}, \quad (30.12)$$

wherever the expression is defined. Usually, $K_h(z) \propto \phi(z/h)$ where ϕ is a zero mean density, and $h > 0$ is the *bandwidth*.

- (i) Suppose $\phi(z) = I\{|z| \leq 1/2\}$, that is, the uniform density on interval $[-1/2, 1/2]$. Set $K_h(z) = \phi(z/h)$ in estimator (30.12). Let $N_h(x)$ be the set of indices from $\{1, \dots, n\}$ defined by

$$N_h(x) = \{i : |x - x_i| \leq h/2\},$$

and denote cardinality $n_h(x) = |N_h(x)|$. Express the $\hat{y}_h(x)$ explicitly in terms of (x_i, y_i) , $i = 1, \dots, n$, making use of $N_h(x)$ and $n_h(x)$.

- (ii) Suppose the error terms ϵ_i in (30.11) are an *iid* sample from $N(0, \sigma^2)$. Give an explicit expression for the variance $V_h(x)$ and bias $B_h(x)$ of $\hat{y}_h(x)$.
- (iii) We next consider a specific model. In (30.11) let

$$g(x) = \beta_0 + \beta_1 x$$

on some interval $x \in [-M, M]$, $M > 0$, for two constants $\beta_0, \beta_1 \neq 0$. A *Poisson process* of rate λ on any interval $\mathcal{I} \subset \mathbb{R}$ is a random set of points $\mathcal{X} \subset \mathcal{I}$ which possesses the following properties (in addition to others):

- (P1) The number of points N' from \mathcal{X} in interval $[a, a+h] \subset \mathcal{I}$ has a Poisson distribution with mean λh .
- (P2) Given that there are N' points from \mathcal{X} in interval $[a, a+h]$, these points form an *iid* sample from a uniform distribution on $[a, a+h]$.

Then assume the predictor values $\mathcal{X} = (x_1, \dots, x_N)$ are generated by a Poisson process on $[-M, M]$ with rate λ (this means that N is a Poisson random variable with mean $2M\lambda$). Then, once \mathcal{X} is generated, responses y_i are given by

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, \dots, N,$$

where the error terms ϵ_i are an *iid* sample from $N(0, \sigma^2)$.

In general, the conditional expected value of X given event A , denoted $E[X | A]$, is the expected value of X under the distribution $P(X \in E | A)$. Accordingly, the quantity $E[B_h(x)^2 | n_h(x) = n]$ is the expected value of $B_h(x)^2$ calculated after assuming that $n_h(x) = n$, so that the property (ii) of a Poisson process given above may be applied. Then calculate

$$E[MSE_h(x) | n_h(x) = n] = E[V_h(x) | n_h(x) = n] + E[B_h(x)^2 | n_h(x) = n].$$

You can assume that M is large enough so that $[x - h/2, x + h/2] \subset [-M, M]$.

- (iv) Write an R program to calculate $E[MSE_h(x) | n_h(x) \geq 1]$ (note that $MSE_h(x)$ is not defined unless $n_h(x) \geq 1$). Set $\beta_1 = 1$, $\lambda = 10$. Fix $\sigma = 0.25$, and calculate $E[MSE_h(x) | n_h(x) \geq 1]$ for h on a grid defined by `seq(0.001, 10, 0.001)`. Plot

$E[MSE_h(x) \mid n_h(x) \geq 1]$ against h , and identify h which minimizes $E[MSE_h(x) \mid n_h(x) \geq 1]$. Repeat for $\sigma = 1.0$.

HINT: Recall the law of total probability. This implies that the expected value of any random variable X may be calculated by:

$$E[X] = E[X \mid A_1]P(A_1) + \dots + E[X \mid A_m]P(A_m),$$

where A_1, \dots, A_m are mutually exclusive events which partition the sample space, so that $P(A_1) + \dots + P(A_m) = 1$. Then

$$E[MSE_h(x) \mid n_h(h) \geq 1] = \sum_{i=1}^{\infty} E[MSE_h(x) \mid n_h(x) = n]P(n_h(x) = n \mid n_h(x) \geq 1). \quad (30.13)$$

If $p(n)$, $n = 0, 1, 2, \dots$ is the PMF of the appropriate Poisson distribution, we have

$$P(n_h(x) = n \mid n_h(x) \geq 1) = p(n)/(1 - p(0)), \quad n = 1, 2, \dots$$

Then $E[MSE_h(x) \mid n_h(h) \geq 1]$ can be evaluated numerically using (30.13), after combining the answer to part (c) with the R `dpois` function. Make sure that enough terms are included in any approximate summation of (30.13) to avoid truncation error.

SOLUTION:

- (a) Clearly, $\phi_{a,b}(x) \geq 0$. Then, apply transformation $z = (x - a)/b$ to the integral

$$\int_{-\infty}^{\infty} \phi_{a,b}(x)dx = \int_{-\infty}^{\infty} b^{-1}\phi(z)bdz = \int_{-\infty}^{\infty} \phi(z)dz = 1.$$

Therefore $\phi_{a,b}(x) \geq 0$ is a density function.

- (b) Applying part (a) directly, it follows that $f_h(x) \geq 0$. Then

$$\int_{-\infty}^{\infty} f_h(x)dx = \frac{1}{n} \sum_{i=1}^n \int_{-\infty}^{\infty} h^{-1}\phi\left(\frac{x - x_i}{h}\right)dx = \frac{1}{n} \sum_{i=1}^n 1 = 1.$$

Therefore $f_h(x)$ is a density function.

- (c) (i) Note that $\hat{y}_h(x)$ is simply a sample mean of responses with indices in $N_h(x)$:

$$\hat{y}_h(x) = \frac{1}{n_h(x)} \sum_{i \in N_h(x)} y_i$$

provided $n_h(x) \geq 1$.

- (ii) We have $E[y_i] = g(x_i)$. Also, $\hat{y}_h(x)$ is an estimator of $g(x)$. The bias is therefore

$$B_h(x) = \left[\frac{1}{n_h(x)} \sum_{i \in N_h(x)} g(x_i) \right] - g(x) = \frac{1}{n_h(x)} \sum_{i \in N_h(x)} [g(x_i) - g(x)]$$

provided $n_h(x) \geq 1$. Since $\hat{y}_h(x)$ is a sample mean of $n_h(x)$ independent random variables with variance σ^2 we have

$$V_h(x) = \frac{1}{n_h(x)} \sigma^2$$

provided $n_h(x) \geq 1$.

- (iii) First consider $B_h(x)$. Note that

$$g(x_i) - g(x) = \beta_1(x_i - x).$$

If $n_h(x) = n$ then the random variables $(x_i - x)$, $i \in N_h(x)$ form an *iid* sample of n uniform random variables on $[-h/2, h/2]$. It follows that

$$E[B_h(x)^2 | n_h(x) = n] = \frac{1}{n} \frac{(\beta_1 h)^2}{12}, \quad n \geq 1.$$

We then have directly,

$$E[V_h(x) | n_h(x) = n] = \frac{1}{n} \sigma^2, \quad n \geq 1.$$

Then

$$E[MSE_h(x) | n_h(x) = n] = \frac{1}{n} \frac{(\beta_1 h)^2}{12} + \frac{1}{n} \sigma^2.$$

- (iv) Note that $n_h(x)$ has a Poisson distribution of mean λh . If its PMF is $p(n)$, then

$$E[MSE_h(x) | n_h(h) \geq 1] = \sum_{n=1}^{\infty} \left[\frac{1}{n} \frac{(\beta_1 h)^2}{12} + \frac{1}{n} \sigma^2 \right] p(n) / (1 - p(0)).$$

The following code performs the required calculates. The optimal values are $h = 1.008, 3.572$, $\sigma = 0.25, 1.0$. See Figure 30.15.

```
### Parameters (note that we don't need beta0)

lam = 10
beta1 = 1

### bias

bb = function(h) {
  ((beta1*h)^2/12)*sum( (1:1:1000)*
    dpois(1:1000,lambda=h*lam)/(1-dpois(0,lambda=h*lam)) )
}

### variance

vv = function(h) {
  (sigma^2)*sum( (1:1:1000)*
    dpois(1:1000,lambda=h*lam)/(1-dpois(0,lambda=h*lam)) )
```

```

}

### MSE

mm = function(h) {bb(h)+vv(h)}

### grid for parameter h

x = seq(0.001,10,0.001)

### do calculations

par(mfrow=c(1,2),pty='s')

for (sigma in c(0.25,1)) {
  y = sapply(x,mm)
  plot(x,y,type='l',xlim='h',ylim='MSE')
  hopt = x[which.min(y)]
  ex1 = bquote(paste(sigma == .(sigma), ', Optimal ',h ==.(hopt),sep=''))
  title(ex1)
}

```

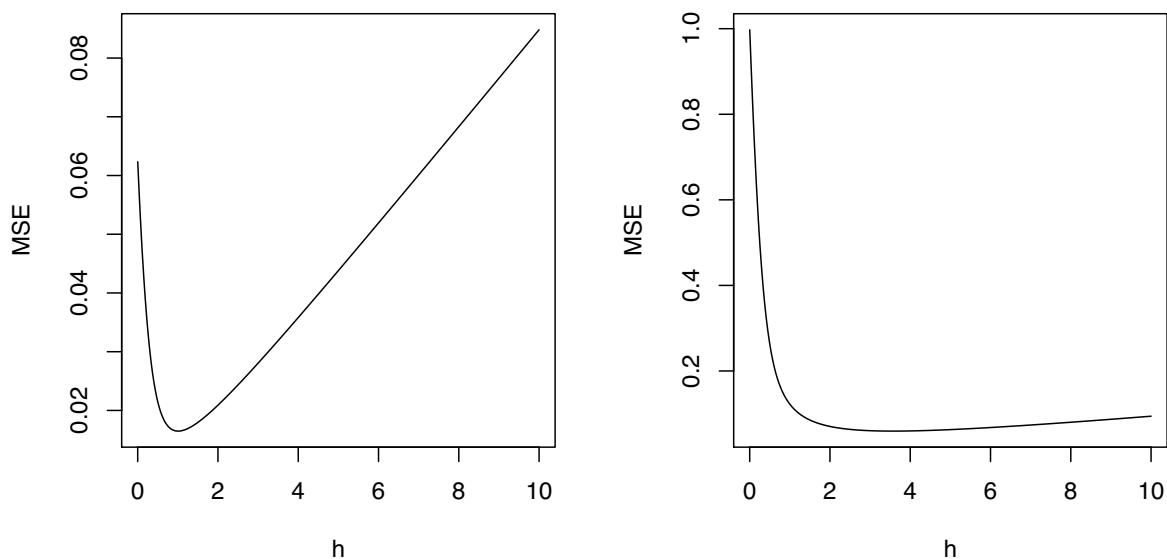
 $\sigma = 0.25, \text{Optimal } h = 1.008$ $\sigma = 1, \text{Optimal } h = 3.572$ 

Figure 30.15: Plots for Problem 30.19.

Problem 30.20. The purpose of the AIC score is to minimize prediction error, rather than to identify the correct model. The two goals are obviously related, but they are not identical. This can be seen using simple linear regression.

Suppose we are given model

$$y = \beta_0 + \beta_1 x + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

Suppose, given paired observations (y_i, x_i) , $i = 1, \dots, n$, we can calculate least-squares coefficient estimates $\hat{\beta}_0, \hat{\beta}_1$. We want to predict, for some fixed predictor value x a new observation $Y_x \sim N(\beta_0 + \beta_1 x, \sigma^2)$ from the model which is independent of the observations used in the estimates. We consider two predictors:

$$\bar{y} = n^{-1} \sum_{i=1}^n y_i \approx Y_x,$$

or

$$\hat{y}_x = \hat{\beta}_0 + \hat{\beta}_1 x \approx Y_x.$$

In one sense, \hat{y}_x is the correct choice, unless $\beta_1 = 0$ (which we don't rule out). One approach is to test against null hypothesis $H_0 : \beta_1 = 0$, then choose \hat{y}_x if we reject H_0 . The other approach is to try to minimize prediction error directly.

Here, the square-error risk is:

$$MSE(y') = E[(Y_x - y')^2],$$

where y' is whatever predictor (that is, \bar{y} or \hat{y}_x) we are considering.

- (a) Express $MSE(y')$ in terms of σ^2 , and the bias and variance of y' . Assume Y_x and y' are independent. Note that in this case $bias(y') = E[y'] - E[Y_x]$.
- (b) Derive $MSE(y')$ for $y' = \bar{y}$ and $y' = \hat{y}_x$.
- (c) Give conditions on $\beta_0, \beta_1, \sigma^2, SS_X = \sum_{i=1}^n (x_i - \bar{x})^2$ under which $MSE(\bar{y}) < MSE(\hat{y}_x)$. Is it possible that \bar{y} could have smaller prediction error even if $\beta_1 \neq 0$?

SOLUTION:

Let $\theta = E[Y_x] = \beta_0 + \beta_1 x$ and $\mu' = E[y']$.

- (a) Then

$$MSE = E[(Y_x - y')^2] = E[(U_1 - U_2 - U_3)^2]$$

where $U_1 = (Y_x - \theta)$, $U_2 = (\mu' - \theta)$, $U_3 = (y' - \mu')$. This can be rewritten

$$E[(U_1 - U_2 - U_3)^2] = E[U_1^2] + E[(U_2 + U_3)^2] - 2E[U_1(U_2 + U_3)].$$

However, U_1 is independent of both U_2, U_3 (note that U_2 is a constant), and $E[U_1] = 0$, so the final term is

$$-2E[U_1(U_2 + U_3)] = -2E[U_1]E[(U_2 + U_3)] = 0.$$

Similarly,

$$\begin{aligned} E[(U_2 + U_3)^2] &= E[U_2^2] + E[U_3^2] + 2E[U_2U_3] \\ &= E[U_2^2] + E[U_3^2], \end{aligned}$$

since $2E[U_2U_3] = 0$. This means

$$E[(U_1 - U_2 - U_3)^2] = E[U_1^2] + E[U_2^2] + E[U_3^2].$$

The first and third terms are the variances of Y_x and y' respectively, while U_2 is the bias of y' so that

$$MSE = \sigma^2 + bias^2(y') + var(y').$$

(b) For $y' = \bar{y}$,

$$E[\bar{y}] = n^{-1} \sum_{i=1}^n E[y_i] = n^{-1} \sum_{i=1}^n \beta_0 + \beta_1 x_i = \beta_0 + \beta_1 \bar{x}$$

where \bar{x} is the sample mean of the predictor observations x_1, \dots, x_n . We then have

$$\begin{aligned} bias(\bar{y}) &= E[\bar{y}] - E[Y_x] = (\beta_0 + \beta_1 \bar{x}) - (\beta_0 + \beta_1 x) = \beta_1(\bar{x} - x), \\ var(\bar{y}) &= \sigma^2/n. \end{aligned}$$

This gives

$$MSE[\bar{y}] = \sigma^2 + \sigma^2/n + \beta_1^2(\bar{x} - x)^2.$$

The MSE for \hat{y}_x is equivalent to the variance of D associated with the prediction interval:

$$MSE[\hat{y}_x] = \sigma^2 + \sigma^2/n + \sigma^2 \frac{(\bar{x} - x)^2}{SS_X},$$

where $SS_X = \sum_{i=1}^n (x_i - \bar{x})^2$.

(c) The two expressions for $MSE[y']$ can be directly compared:

$$MSE[\hat{y}_x] - MSE[\bar{y}] = (\bar{x} - x)^2 \left[\frac{\sigma^2}{SS_X} - \beta_1^2 \right].$$

If $x = \bar{x}$ then $MSE[\hat{y}_x] = MSE[\bar{y}]$. Otherwise, we have $MSE[\bar{y}] < MSE[\hat{y}_x]$ if and only if

$$\beta_1^2 < \frac{\sigma^2}{SS_X}.$$

In particular, \bar{y} may have smaller prediction error even if $\beta_1 \neq 0$. Note that this condition does not depend on β_0 .

Problem 30.21. Consider the matrix representation of the multiple linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where \mathbf{y} is an $n \times 1$ response vector, \mathbf{X} is a $n \times q$ matrix, $\boldsymbol{\beta}$ is a $q \times 1$ vector of coefficients, and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of error terms. The least squares solution is expressed using the coefficient vector $\boldsymbol{\beta}$ which minimizes the error sum of squares

$$SSE[\boldsymbol{\beta}] = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

- (a) By setting each partial derivative $\partial SSE[\beta]/\partial \beta_j$ to zero, $j = 1, \dots, q$, verify that the least squares solution is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

- (b) Next, recall that the ridge regression coefficients are obtained by minimizing

$$\Lambda = SSE[\beta] + \lambda \sum_{j=1}^q \beta_j^2$$

for a fixed constant $\lambda \geq 0$. By setting each partial derivative $\partial \Lambda / \partial \beta_j$ to zero, $j = 1, \dots, q$, show that the ridge regression solution is

$$\hat{\beta}_{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda I_q)^{-1} \mathbf{X}^T \mathbf{y},$$

where I_q is the $q \times q$ identity matrix.

SOLUTION:

- (a) Suppose we write $SSE[\beta]$ in the following way:

$$SSE[\beta] = \sum_{i=1}^n (y_i - \beta_1 x_{i1} - \dots - \beta_q x_{iq})^2,$$

where x_{ij} is the element of \mathbf{X} in row j and column j . Then take the partial derivative

$$\frac{\partial SSE[\beta]}{\partial \beta_j} = \sum_{i=1}^n -2x_{ij}(y_i - \beta_1 x_{i1} - \dots - \beta_q x_{iq}).$$

If each partial derivative is set to 0, the solution is obtained by substituting $\hat{\beta}$ for β . This can be expressed

$$\sum_{i=1}^n x_{ij} y_i = \sum_{i=1}^n x_{ij} (\hat{\beta}_1 x_{i1} + \dots + \hat{\beta}_q x_{iq}), \quad j = 1, \dots, q.$$

This is a system of q linear equations, which can be expressed in matrix notation as

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \hat{\beta}.$$

Premultiplying each side by $(\mathbf{X}^T \mathbf{X})^{-1}$ gives

$$(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \hat{\beta}.$$

It is also possible to use matrix calculus directly. This would give

$$\frac{\partial SSE[\beta]}{\partial \beta} = -2 \mathbf{X}^T (\mathbf{y} - \mathbf{X} \beta).$$

Setting this vector equal to 0 gives the same equation $\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \hat{\beta}$.

(b) The approach for ridge regression is very similar. The partial derivative is now

$$\frac{\partial SSE[\beta]}{\partial \beta_j} = \left\{ \sum_{i=1}^n -2x_{ij}(y_i - \beta_1 x_{i1} - \dots - \beta_q x_{iq}) \right\} + 2\lambda\beta_j.$$

In matrix notation, setting the partial derivatives equal to zero, then substituting $\hat{\beta}$ for β gives

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \hat{\beta} + \lambda \hat{\beta} = (\mathbf{X}^T \mathbf{X} + \lambda I_q) \hat{\beta}.$$

Premultiplying each side by $(\mathbf{X}^T \mathbf{X} + \lambda I_q)^{-1}$ gives

$$(\mathbf{X}^T \mathbf{X} + \lambda I_q)^{-1} \mathbf{X}^T \mathbf{y} = \hat{\beta}.$$

Matrix calculus would similarly give

$$\frac{\partial SSE[\beta]}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) + 2\lambda\beta,$$

leading to the same result.

Problem 30.22. Suppose for a certain application, data takes the form (X_1, X_2) , where X_i are independent Bernoulli random variables (ie. assume value 0 or 1) of mean p . The usual unbiased estimator of p is

$$\hat{p}_1 = \frac{X_1 + X_2}{2},$$

which has variance $\sigma_p^2 = p(1-p)/2$. Suppose we also consider as an alternative estimator of p :

$$\hat{p}_2 = \hat{p}_1/2 + 1/4.$$

- (a) Give an expression for $MSE = variance + bias^2$ as a function of p for each estimator.
- (b) Which estimator has smallest MSE when $p = 1/2$?
- (c) Which estimator has smallest MSE when $p = 1/8$?

SOLUTION:

- (a) For \hat{p}_1 we have:

$$\begin{aligned} var(\hat{p}_1) &= \frac{1}{2} \cdot p(1-p) \\ bias(\hat{p}_1) &= E[\hat{p}_1] - p = p - p = 0 \\ MSE(\hat{p}_1) &= var(\hat{p}_1) + bias(\hat{p}_1)^2 = \frac{1}{2} \cdot p(1-p). \end{aligned}$$

For \hat{p}_2 we have:

$$\begin{aligned} var(\hat{p}_2) &= var(\hat{p}_1/2 + 1/4) = \frac{1}{2^2} \cdot var(\hat{p}_1) = \frac{1}{8} \cdot p(1-p) \\ bias(\hat{p}_2) &= E[\hat{p}_2] - p = p/2 + 1/4 - p = -p/2 + 1/4 \\ MSE(\hat{p}_1) &= var(\hat{p}_2) + bias(\hat{p}_2)^2 = \frac{1}{8} \cdot p(1-p) + (p/2 - 1/4)^2 = \frac{1}{8} \cdot (p^2 - p + 1/2). \end{aligned}$$

(b) For $p = 1/2$,

$$\begin{aligned} MSE(\hat{p}_1) &= \frac{1}{2} \cdot 1/2(1 - 1/2) = 1/8, \\ MSE(\hat{p}_2) &= \frac{1}{8} \cdot ((1/2)^2 - 1/2 + 1/2) = 1/32. \end{aligned}$$

so $MSE(\hat{p}_2) < MSE(\hat{p}_1)$.

(c) For $p = 1/8$,

$$\begin{aligned} MSE(\hat{p}_1) &= \frac{1}{2} \cdot 1/8(1 - 1/8) = 7/128 = 7/2^7, \\ MSE(\hat{p}_2) &= \frac{1}{8} \cdot ((1/8)^2 - 1/8 + 1/2) = 25/512 = 25/2^9. \end{aligned}$$

so $0.0488 = MSE(\hat{p}_2) < MSE(\hat{p}_1) = 0.0547$.

Problem 30.23. In this question, we will explore various aspects of the ‘hat matrix’

$$H = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \quad (30.14)$$

associated with multiple linear regression with q predictors (including the intercept term). We can think of H as a linear transformation of an n -dimensional response vector \mathbf{y} to the n -dimensional fitted vector $\hat{\mathbf{y}} = H\mathbf{y}$. Furthermore, since $\hat{\mathbf{y}}$ is obtained by minimizing the SSE , the action taken by H on \mathbf{y} is to *project* it onto the q -dimensional subspace $\mathcal{S}_q \subset \mathbb{R}^n$ spanned by the q predictors (equivalently, \mathcal{S}_q is the set of all linear combinations of the predictors, see Appendix A.2). This means $\hat{\mathbf{y}}$ is the point in \mathcal{S}_q closest to \mathbf{y} .

(a) Given Equation (30.14) prove that

$$trace(H) = q,$$

assuming matrices are invertible where indicated. [HINT: First prove the following identity. If A, B are $n \times m$ matrices, then $trace(AB^T) = trace(B^TA)$].

(b) A square matrix A is *idempotent* if and only if $A = AA$. Show that H is idempotent, using two arguments:

- (i) Analytically, using matrix algebra following Equation (30.14). Assume matrices are invertible where indicated.
- (ii) Logically, using the fact that, for any $\mathbf{y} \in \mathbb{R}^n$, $H\mathbf{y}$ is the point in \mathcal{S}_q closest to \mathbf{y} .

(c) The ‘hat matrix’ structure appears in many modeling techniques, both parametric and non-parametric. Denote the least squares projection matrix defined in (30.14) H_{LS} . The following three modeling techniques yield various forms of fitted vectors $\hat{\mathbf{y}} = H'\mathbf{y}$ as linear transformations of response vector \mathbf{y} . In each case, describe precisely H' , and give its trace.

- (i) We have $\hat{\mathbf{y}} = (\bar{\mathbf{y}}, \dots, \bar{\mathbf{y}}) \in \mathbb{R}^n$, where $\bar{\mathbf{y}}$ is the sample mean of the elements of \mathbf{y} .
- (ii) Here, the elements of \mathbf{y} are assumed sorted in some sequence, either by a time index or some other predictor. We take *moving average*

$$\begin{aligned}\hat{\mathbf{y}}_1 &= \frac{\mathbf{y}_1 + \mathbf{y}_2}{2} \\ \hat{\mathbf{y}}_i &= \frac{\mathbf{y}_{i-1} + \mathbf{y}_i + \mathbf{y}_{i+1}}{3}, \quad i = 2, 3, \dots, n-1, \\ \hat{\mathbf{y}}_n &= \frac{\mathbf{y}_{n-1} + \mathbf{y}_n}{2}\end{aligned}$$

- (iii) We can define a *saturated model* as one which fits the original response vector exactly, that is, $\hat{\mathbf{y}} = \mathbf{y}$.
- (d) The quantity $\text{trace}(H)$ is sometimes referred to as the *effective degrees of freedom*. In linear regression it equals the model degrees of freedom (which is equal to the number of regression coefficients). But it has a similar interpretation for other forms of models, and can be taken as an index of model complexity. Order the four ‘hat matrices’ considered in part (c) by effective degrees of freedom, and describe briefly how this ordering relates to the complexity of the models considered. Note that we must assume $n \geq 2$, otherwise H_{LS} is not defined. Furthermore, for the ordering to hold, we may need to assume $n \geq n_{lower}$, where n_{lower} might be larger than 2.

SOLUTION:

- (a) Note that A and B have the same dimension. Denote the elements a_{ij}, b_{ij} . Then AB^T is an $n \times n$ matrix with i th diagonal element

$$[AB^T]_{ii} = \sum_{j=1}^m a_{ij}b_{ij}, \quad i = 1, \dots, n,$$

so that

$$\text{trace}(AB^T) = \sum_{i=1}^n \sum_{j=1}^m a_{ij}b_{ij}.$$

Applying essentially the same calculation method to $\text{trace}(B^TA)$ yields exactly the same value. This proves the identity. It follows that

$$\text{trace}(\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) = \text{trace}(\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}) = \text{trace}(I_q) = q,$$

where I_q is the $q \times q$ identity matrix.

(b) (i) We have

$$HH = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \{ \mathbf{X}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \} \mathbf{X}^T = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} I_q \mathbf{X}^T = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T = H,$$

where I_q is the $q \times q$ identity matrix.

(ii) If for any $\mathbf{y} \in \mathbb{R}^n$, $H\mathbf{y}$ is the point in \mathcal{S}_q closest to \mathbf{y} , then if $\mathbf{y} \in \mathcal{S}_q$ we must have $H\mathbf{y} = \mathbf{y}$, since \mathbf{y} would then be the point in \mathcal{S}_q closest to itself.

Then, for any $\mathbf{y} \in \mathbb{R}^n$, $H\mathbf{y}$ is in \mathcal{S}_q , so that $H(H\mathbf{y}) = H\mathbf{y}$. This is true for any $\mathbf{y} \in \mathbb{R}^n$, so we must have $H = HH$.

(c) (i) We have $\bar{\mathbf{y}}_i = \bar{\mathbf{y}} = \sum_{i=1}^n n^{-1} y_i$, so $H'_{ij} = 1/n$ for each matrix element. We therefore have

$$\text{trace}(H') = 1.$$

(ii) The coefficients defining the moving average model are placed directly into H' ,

$$H' = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & & & & & \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

For this matrix, assuming $n \geq 2$,

$$\text{trace}(H') = 1/2 + 1/2 + (n-2)1/3 = (n+1)/3.$$

(iii) In this case, we have $H' = I_n$, so

$$\text{trace}(H') = n.$$

(d) Label the ‘hat matrices’ of parts (i)-(iii) $H_{(i)}$, $H_{(ii)}$, $H_{(iii)}$. Then

$$\text{trace}(H_{(i)}) < \text{trace}(H_{LS}) < \text{trace}(H_{(ii)}) < \text{trace}(H_{(iii)}),$$

provided $n \geq n_{lower} = 6$ (the lower bound is needed for $\text{trace}(H_{LS}) < \text{trace}(H_{(ii)})$). That $\text{trace}(H_{(i)}) < \text{trace}(H_{LS})$ is clear since $H_{(i)}$ implements the linear model $Y = \beta_0 + \epsilon$, which is a submodel of $Y = \beta_0 + \beta_1 X + \epsilon$. Then $H_{(ii)}$ is a *smoother*, which tends to force the fitted values $\hat{\mathbf{y}}$ into a smooth functional form, while allowing more curvature than a strict linear fit. In this case, $\hat{\mathbf{y}}$ would be closer to the data than would be $H_{LS}\mathbf{y}$. So, for any large enough n we would expect $\text{trace}(H_{LS}) < \text{trace}(H_{(ii)})$. Finally, $H_{(iii)}$ simply reproduces the data as it is, and is in this sense the most complex model possible. We would therefore expect $\text{trace}(H_{(iii)})$ to be largest.

Chapter 31

Practice Problems - Bayesian Networks

31.1 Exercises

Problem 31.1. The following 6 variables are defined:

$$\begin{aligned} X_1 &= \text{Visits Hospital} \\ X_2 &= \text{Exposure to Bacteria} \\ X_3 &= \text{Immunity to Bacteria} \\ X_4 &= \text{Acquires Infection} \\ X_5 &= \text{Resistance to Antibiotics} \\ X_6 &= \text{Misses Work} \end{aligned}$$

and are used in the following model:

$$\begin{aligned} X_2 &= X_1 + \epsilon_1 \\ X_4 &= X_2 + X_3 + \epsilon_2 \\ X_6 &= X_4 + X_5 + \epsilon_3. \end{aligned}$$

The terms $\epsilon_1, \epsilon_2, \epsilon_3$ are independently distributed errors.

- Suppose we want to model $\tilde{X} = (X_1, X_2, X_3, X_4, X_5, X_6)$ as a Bayesian network. Sketch all DAGs which imply conditional independencies consistent with \tilde{X} .
- Suppose we wish to develop a predictive model for response X_6 using the remaining elements of \tilde{X} as predictors (we assume they are all observable). Which elements should we use? Answer the same question for response X_4 .

SOLUTION:

- (a) The equations imply 5 edges: $1 \rightarrow 2$, $2 \rightarrow 4$, $3 \rightarrow 4$, $4 \rightarrow 6$ and $5 \rightarrow 6$. This graph is shown Figure 31.1 (left). Any equivalent graph must be obtained by reversing the direction of some combination of edges without adding or removing a v-structure. There are 2 v-structures: $2 \rightarrow 4$, $3 \rightarrow 4$ and $4 \rightarrow 6$, $5 \rightarrow 6$. None of those edges can be changed. However, the direction of $1 \rightarrow 2$ can be changed, yielding the 2 equivalent DAGs in Figure 31.1.
- (b) The *Markov blanket* of a node is that node's children, parents and children's other parents. A node is independent of the remaining nodes conditional on its Markov blanket. The Markov blanket of X_6 is $\{X_4, X_5\}$, so we would only need to use X_4 and X_5 in a predictive model with response X_6 . The Markov blanket of X_4 is $\{X_2, X_3, X_5, X_6\}$, so we would only need to use those variables in a predictive model with response X_4 .

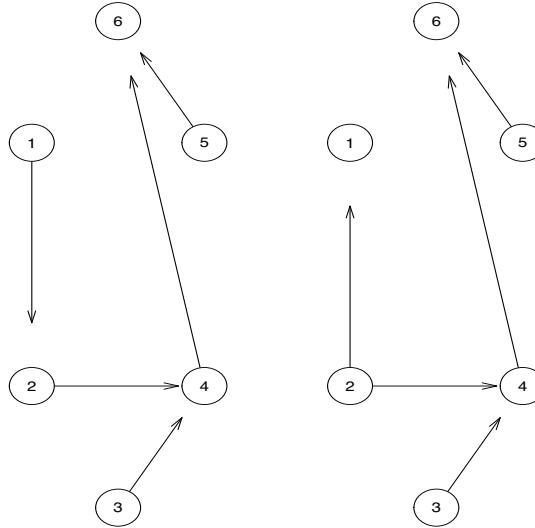


Figure 31.1: DAGs for Problem 31.1.

Problem 31.2. A Bayesian network model is fit for 8 genes labeled $a - h$, resulting in the following DAG. We say a gene y is downstream from gene x if there is a directed path from x to y .

- (a) List all v-structures of the DAG.
- (b) State precise conditions under which two DAGs are equivalent (that is, are members of the same equivalence class).
- (c) Suppose the DAG is accepted as a true model of regulatory control. Then, in reference to the equivalence classes of the DAG, a statement about regulatory order may be one of three types:
 - A. Implied by the Bayesian network model (true of all equivalent DAGs).
 - B. Compatible with the Bayesian network model (true of some but not all equivalent DAGs).
 - C. Not compatible with the Bayesian network model (not true of any equivalent DAG).

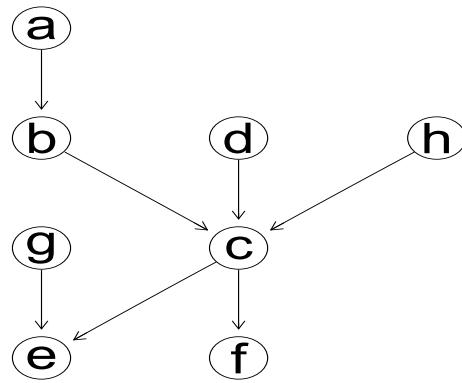


Figure 31.2: Sample DAG for Problem 31.2.

Note that a DAG is equivalent to itself. Of which type (A, B or C) is each of the following statements? Briefly justify your answer.

- (i) c is downstream from h .
- (ii) g is downstream from c .
- (iii) h has no parents.
- (iv) b has no parents.
- (v) c has exactly three parents.
- (vi) f is downstream from a .

SOLUTION:

- (a) There 4 v-structures:
- (a) $b \rightarrow c, d \rightarrow c$
 - (b) $b \rightarrow c, h \rightarrow c$
 - (c) $d \rightarrow c, h \rightarrow c$
 - (d) $g \rightarrow e, c \rightarrow e$
- (b) v-structures and topologies are identical.
- (c)
- (i) A. $h \rightarrow c$ is part of (at least) one v-structure which cannot be removed.
 - (ii) C. $g \rightarrow e$ and $c \rightarrow e$ form a v-structure, which cannot be removed.
 - (iii) A. h shares an edge with c only, which is part of a v-structure and cannot be changed.
 - (iv) B. The direction of the edge $a \rightarrow b$ can be changed to yield an equivalent graph in which b has no parents.
 - (v) A. All parents of c are part of v-structures pointing to c . Deletion or addition of any other parent would add or delete a v-structure.
 - (vi) B. The direction of the edge $a \rightarrow b$ can be changed to yield an equivalent graph in which f is not downstream of a .

Problem 31.3. Suppose the density of a random vector $\tilde{X} = (X_1, X_2, X_3, X_4, X_5)$ can be factorized in the following way:

$$f(x_1, \dots, x_5) = f(x_5 | x_4)f(x_4 | x_2, x_3)f(x_2 | x_1)f(x_1)f(x_3)$$

- (a) Suppose we want to model \tilde{X} as a Bayesian network. Sketch all DAGs which imply conditional independencies consistent with \tilde{X} .
- (b) Suppose we wish to develop a predictive model for response X_5 using the remaining elements of \tilde{X} as predictors. Which elements should we use? Answer the same question for response X_4 .

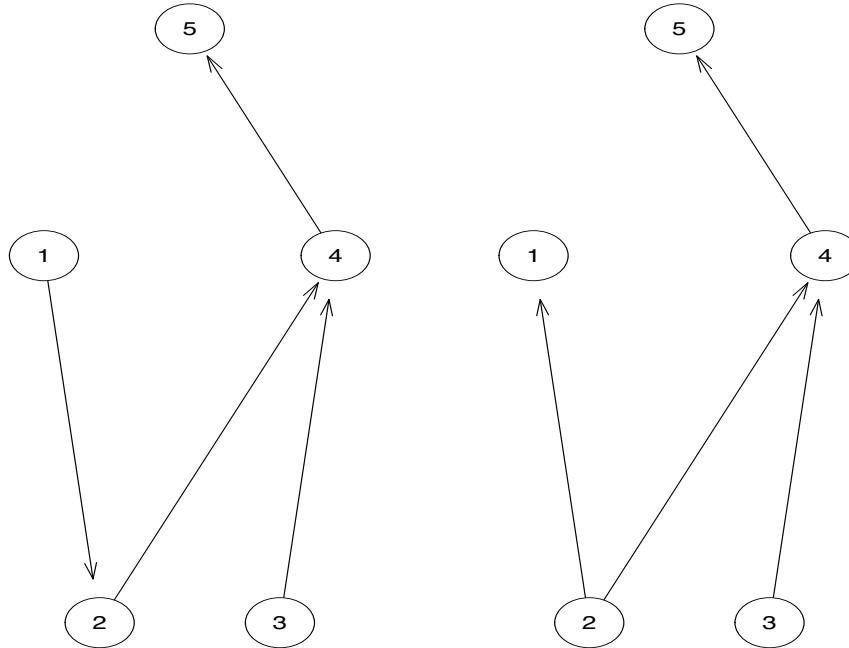


Figure 31.3: DAGs for Problem 31.3 (a).

SOLUTION:

- (a) The factorization implies 4 edges: $4 \rightarrow 5$, $2 \rightarrow 4$, $3 \rightarrow 4$ and $1 \rightarrow 2$. This graph is shown Figure 31.3 (left). Any equivalent graph must be obtained by reversing the direction of some combination of edges without adding or removing a v-structure. Note that edges $2 \rightarrow 4$ and $3 \rightarrow 4$ form a v-structure, and so cannot be changed. If edge $4 \rightarrow 5$ is changed, then a new v-structure will be added. However, the direction of $1 \rightarrow 2$ can be changed, yielding the 2 equivalent DAGs in Figure 31.3.
- (b) The *Markov blanket* of a node is that node's children, parents and children's other parents. A node is independent of the remaining nodes conditional on its Markov blanket. The Markov blanket of X_5 is X_4 , so we would only need to use X_4 in a predictive model with response

X_5 . The Markov blanket of X_4 is $\{X_2, X_3, X_5\}$, so we would only need to use X_2, X_3, X_5 in a predictive model with response X_4 .

31.2 Data Analysis

Problem 31.4. Load the data frame `Hitters` from the library `ISLR`. Create a new data frame from variables

`Hits, HmRun, Runs, RBI, Walks.`

Then divide each column by `AtBat`, so that the quantities are in units of ‘per at bat’. Then add to the data frame log-transforms of `Salary` and `PutOuts` (use base 10). Remove any observations for which `PutOuts == 0`. Use the `na.omit` function to delete observations with missing values. Finally, create all pairwise scatterplots for the data frame, then remove any obvious outliers.

- Fit a Bayesian network to the data using the `hc()` function from the library `bnlearn`. Use default options. Plot the model’s directed acyclic graph (DAG).
- Examine a correlation matrix of the data. Is it possible to say that each parent is positively associated with each child?
- Use the function `cpdag` to represent the equivalence class of the DAG. Plot this representation. Can the direction of any of the edges be changed without changing the model score?
- What is the Markov blanket for node `logSalary`? You can use function `mb` to verify your answer. How is the Markov blanket precisely interpreted?
- Suppose we wish to develop a model which predicts a player’s salary. Fit a regression model with `logSalary` as response and the remaining variables as predictor. Use all-subsets regression to select the variables using the *AIC* score (you can use the `dredge` and `get.models` functions from the `MuMIn` library). Does the model conform to the Bayesian network model?

SOLUTION:

We may use the following code:

```
> library(bnlearn)
> library(ISLR)
> library(MuMIn)
>
> ### create data frame
>
> ### hitting metrics
>
> var.list = c(2:6)
> x = Hitters[,var.list]
> n = as.numeric(Hitters[,1])
>
> nc = dim(x)[2]
```

```

> for (i in 1:5) {x[,i] = as.numeric(x[,i]/n)}
>
> ### add log Salary
>
> x1 = data.frame(x,log10(Hitters$Salary), log10(Hitters$PutOuts))
> x1 = na.omit(x1)
> names(x1)[6:7] = c('logSalary','logPutOuts')
>
>
> ### examine data
>
> pairs(x1)
> x1 = subset(x1, RBI > 0 & logPutOuts > -Inf)
>
>
> ### fit Bayesian network
>
> par(mfrow=c(1,2))
> bn = hc(x1)
> plot(bn,main='DAG')
> cor(x1)

      Hits      HmRun      Runs      RBI      Walks logSalary logPutOuts
Hits 1.00000000 0.06972857 0.43753009 0.23404450 -0.03377821 0.3376368 0.02643426
HmRun 0.06972857 1.00000000 0.35301351 0.77988977 0.14719857 0.1854447 0.12790676
Runs 0.43753009 0.35301351 1.00000000 0.25844140 0.33937901 0.2405413 0.02421579
RBI 0.23404450 0.77988977 0.25844140 1.00000000 0.16430481 0.2508894 0.09405556
Walks -0.03377821 0.14719857 0.33937901 0.16430481 1.00000000 0.1755441 0.09376591
logSalary 0.33763675 0.18544467 0.24054129 0.25088936 0.17554410 1.0000000 0.25156172
logPutOuts 0.02643426 0.12790676 0.02421579 0.09405556 0.09376591 0.2515617 1.00000000

> plot(cpdag(bn),main='EC')
>
> mb(bn,'logSalary')
[1] "Hits"       "RBI"        "Walks"       "logPutOuts"
>
> fit = lm(logSalary ~ ., data = x1, na.action=na.fail)
>
> junk = dredge(fit, rank = "AIC")
Fixed term is "(Intercept)"
> fit.aic = get.models(junk, 1)[[1]]
> fit.aic

Call:
lm(formula = logSalary ~ Hits + logPutOuts + RBI + Walks + 1,
    data = x1, na.action = na.fail)

```

Coefficients:

(Intercept)	Hits	logPutOuts	RBI	Walks
0.6075	4.0604	0.2430	1.4211	1.4122

>

Outliers were removed by the `x1 = subset(x1, RBI > 0 & logPutOuts > -Inf)` command. See Figure 31.4.

- (a) See Figure 31.5.
- (b) The correlation matrix conforms to the DAG in this sense. For example, `Hits` and `Runs` have sample correlation 0.43753009.
- (c) The equivalence class representation of the DAG is exactly the same as the DAG itself, so no directions can be changed without changing the model score. See Figure 31.5.
- (d) A Markov blanket of a node includes the parents of the node, the children of the node, and any other parents of a child of the node. In the case of our DAG, the Markov blanket for the `logSalary` node is `Hits`, `RBI`, `Walks` and `logPutOuts`. A node is independent of all other nodes conditional on the Markov blanket.
- (e) The variables selected in the model are exactly those included in the Markov blanket of node `logSalary`. According to the Bayesian network model, the Markov blanket of a node contains all available predictive information for that node. In this sense, the *AIC* model selection conforms to the Bayesian network.

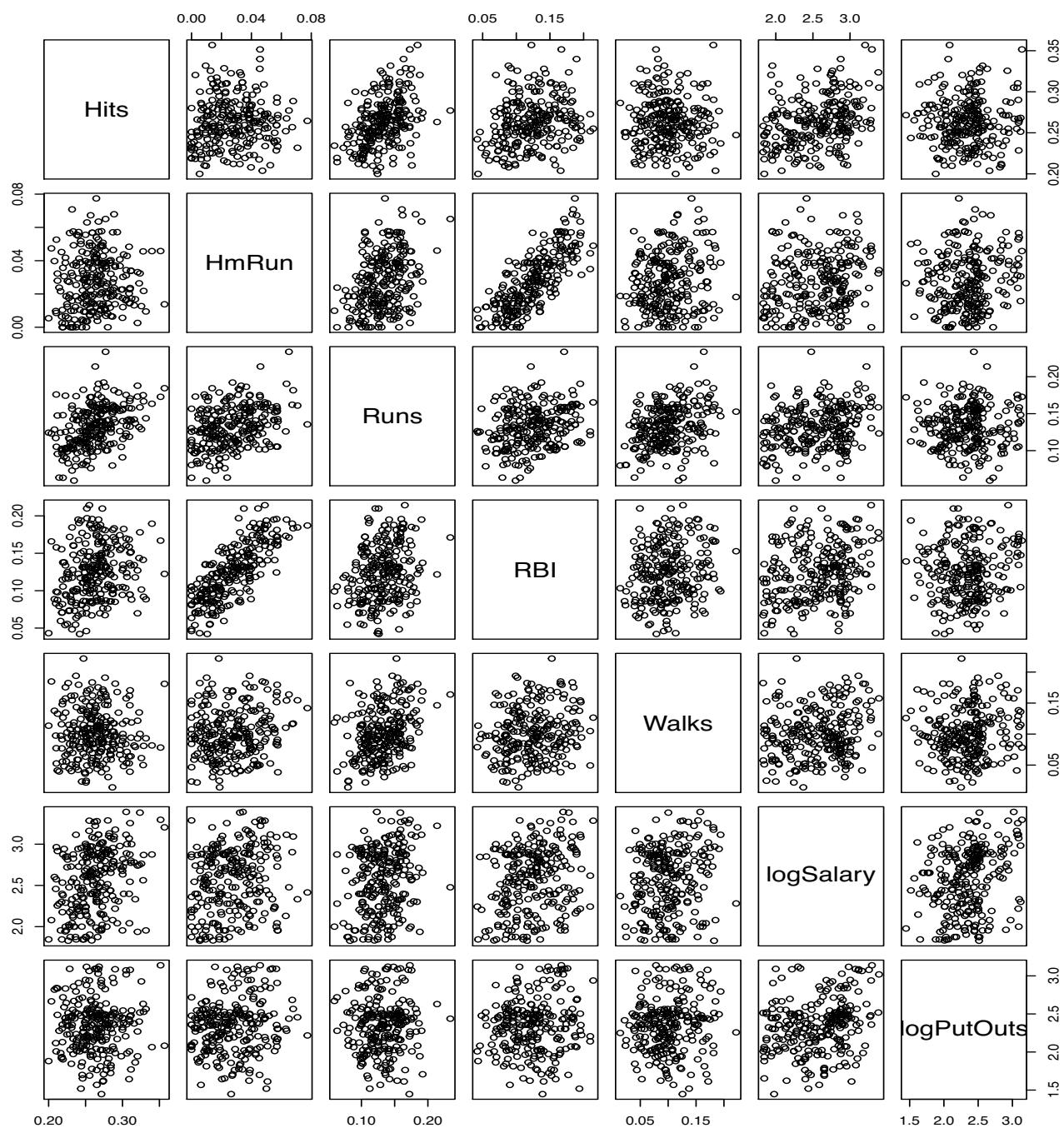


Figure 31.4: Plot for Problem 31.4.

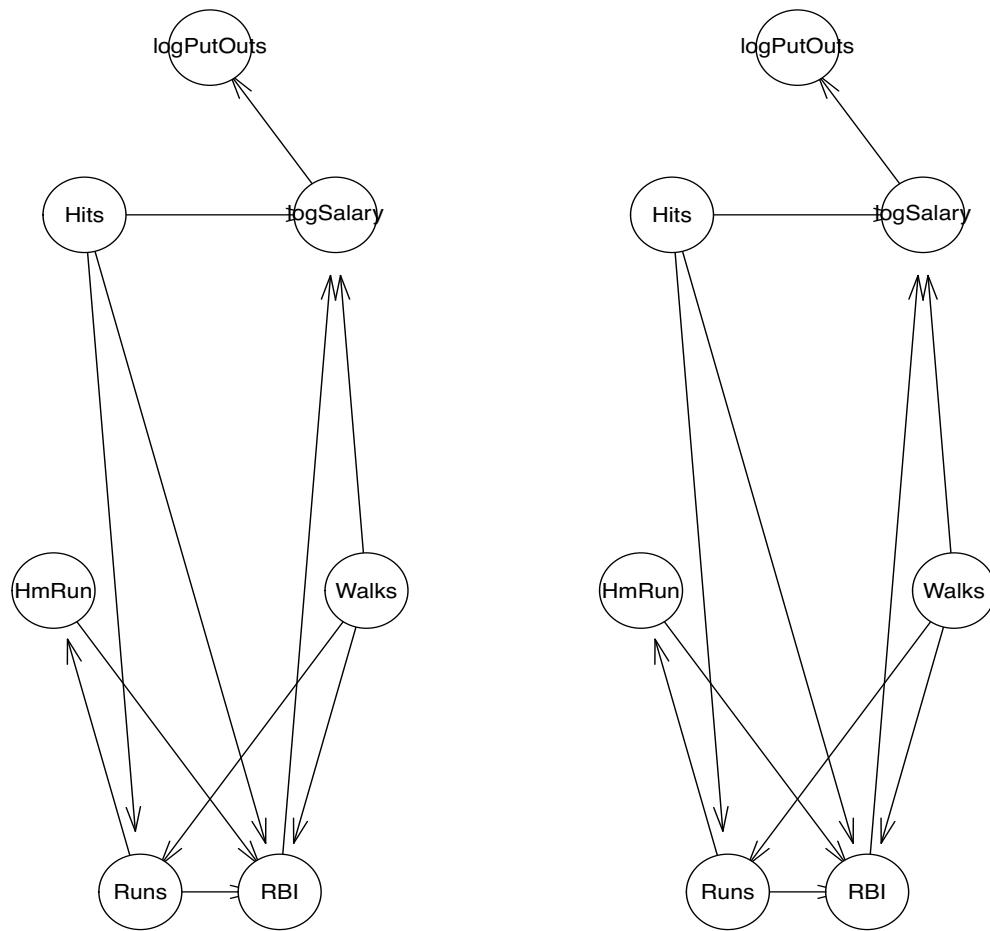


Figure 31.5: DAG and equivalence class for Problem 31.4 (a), (c).

Appendices

Appendix A

Linear Algebra

A.1 Numbers and Sets

A *set* is a collection of distinct objects of any kind. Each member of a set is referred to as an *element*, and is represented once. A set E may be *indexed*. That is, given an index set \mathcal{T} , each element may be assigned a unique index $t \in \mathcal{T}$, and all indices in \mathcal{T} are assigned to exactly one element of E , denoted x_t . We may then write $E = \{x_t; t \in \mathcal{T}\}$.

The set of (finite) real numbers is denoted \mathbb{R} , and the set of extended real numbers is denoted $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$. The restriction to nonnegative real numbers is written $\mathbb{R}_+ = [0, \infty)$ and $\bar{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{\infty\}$. We use standard notation for open, closed, left closed and right closed intervals (a, b) , $[a, b]$, $[a, b)$, $(a, b]$. A reference to a interval I on $\bar{\mathbb{R}}$ may be any of these types.

The set of (finite) integers will be denoted \mathbb{I} , while the extended integers will be $\mathbb{I}_\infty = \mathbb{I} \cup \{-\infty, \infty\}$. The set of natural numbers \mathcal{N} is taken to be the set of positive integers, which \mathcal{N}_0 is the set of nonnegative integers. A rational number is any real number expressible as a ratio of integers.

Then \mathcal{C} denotes the complex numbers $z = a + bi \in \mathcal{C}$, where $i = \sqrt{-1}$, $a, b \in \mathbb{R}$ is the imaginary number. Note that i is added and multiplied as though it were a real number, in particular $i^2 = -1$. Multiplication is defined by $z_1 z_2 = (a_1 + b_1 i)(a_2 + b_2 i) = a_1 a_2 - b_1 b_2 + (a_1 b_2 + a_2 b_1)i$. The *conjugate* of $z = a + bi \in \mathcal{C}$ is written $\bar{z} = a - bi$, so that $z\bar{z} = a^2 + b^2 \in \mathbb{R}$. Together, z and \bar{z} , without reference to their order, form a *conjugate pair*.

The absolute value of $a \in \mathbb{R}$ is denoted $|a| = \sqrt{a^2}$, while $|z| = (z\bar{z})^{1/2} = (a^2 + b^2)^{1/2} \in \mathbb{R}$ is also known as the magnitude or modulus of $z \in \mathcal{C}$.

If \mathcal{S} is a set of any type of number, \mathcal{S}^d , $d \in \mathcal{N}$, denotes the set of d -dimensional vectors $\tilde{s} = (s_1, \dots, s_d)$, which are ordered collections of numbers $s_i \in \mathcal{S}$. In particular, the set of d -dimensional real vectors is written \mathbb{R}^d . When $0, 1 \in \mathcal{S}$, we may write the zero or one vector $\vec{0} = (0, \dots, 0)$, $\vec{1} = (1, \dots, 1)$, so that $c\vec{1} = (c, \dots, c)$.

A collection of d numbers from \mathcal{S} is *unordered* if no reference is made to the order (they are unlabelled). Otherwise the collection is *ordered*, that is, it is a vector. An unordered collection from \mathcal{S} differs from a set in that a number $s \in \mathcal{S}$ may be represented more than once. Braces $\{\dots\}$ enclose a set while parentheses (\dots) enclose a vector (braces will also be used to denote indexed sequences, when the context is clear).

The *cardinality* of a set E is the number of elements it contains, and is denoted $|E|$. If $|E| < \infty$ then E is a finite set. We have $|\emptyset| = 0$. If $|E| = \infty$, this statement does not suffice to characterize the cardinality of E . Two sets A, B are in a *1-1 correspondence* if a collection of pairs (a, b) , $a \in A$,

$b \in B$ can be constructed such that each element of A and of B is in exactly one pair. In this case, A and B are of equal cardinality. The pairing is known as a *bijection*.

If the elements of A can be placed in a 1-1 correspondence with \mathcal{N} we say A is *countable* (is *denumerable*). We also adopt the convention of referring to any subset of a countable set as countable. This means all finite sets are countable. If for countable A we have $|A| = \infty$ then A is *infinitely countable*. Note that by some conventions, the term countable is reserved for infinitely countable sets. For our purposes, it is more natural to consider the finite sets as countable.

All infinitely countable sets are of equal cardinality with \mathcal{N} , and so are mutually of equal cardinality. informally, a set is countable if it can be written as a list, finite or infinte. The set \mathcal{N}^d is countable since, for example, $\mathcal{N}^2 = \{(1, 1), (1, 2), (2, 1), (1, 3), (2, 2), (3, 1), \dots\}$. The set of rational numbers is countable, since the pairing of numerator and denominator, in any canonical representation, is a subset of \mathcal{N}^2 .

A set A is *uncountable* (is *nondenumerable*) if $|A| = \infty$ but A is not countable. The set of real numbers, or any nonempty interval of real numbers, is uncountable.

If A_1, \dots, A_d are d sets, then $A_1 \times A_2 \times \dots \times A_d = \times_{i=1}^d A_i$ is a product set, consisting of the set of all ordered selections of one element from each set $a_i \in A_i$. A vector is an element of a product set, but a product set is more general, since the sets A_i need not be equal, or even contain the same type of element. The definition may be extended to arbitrary forms of index sets.

A.2 Fields and Vector Spaces

The notion of real numbers can be generalized to that of a *field* \mathcal{K} , which is a set of *scalars* that is closed under the rules of addition and multiplication comparable to those available for real numbers \mathbb{R} . Both \mathbb{R} and complex numbers \mathcal{C} are fields.

A *vector space* $\mathcal{V} \subset \mathcal{K}^n$ is any set of vectors $x \in \mathcal{K}^n$ which is closed under linear and scalar composition, that is, if $x, y \in \mathcal{V}$ then $ax + by \in \mathcal{V}$ for all scalars a, b . This means the zero vector $\vec{0}$ must be in \mathcal{V} , and that $x \in \mathcal{V}$ implies $-x \in \mathcal{V}$.

Elements x_1, \dots, x_m of \mathcal{K}^n are *linearly independent* if $\sum_{i=1}^m a_i x_i = 0$ implies $a_i = 0$ for all i . Equivalently, no x_i is a linear combination of the remaining vectors. The *span* of a set of vectors $\tilde{x} = (x_1, \dots, x_n)$, denoted $\text{span}(\tilde{x})$, is the set of all linear combinations of vectors in \tilde{x} , which must be a vector space. Suppose the vectors in \tilde{x} are not linearly independent. This means that, say, x_m is a linear combination of the remaining vectors, and so any linear combination in $\text{span}(\tilde{x})$ including x_m may be replaced with one including only the remaining vectors, so that $\text{span}(\tilde{x}) = \text{span}(x_1, \dots, x_{m-1})$. The *dimension* of a vector space \mathcal{V} is the minimum number of vectors whose span equals \mathcal{V} . Clearly, this equals the number in any set of linearly independent vectors which span \mathcal{V} . Any such set of vectors forms a *basis* for \mathcal{V} . Any vector space has a basis.

A.3 Equivalence Relationships

Suppose \mathcal{X} is a set of objects, and \sim defines a *binary relation* between two objects $x, y \in \mathcal{X}$.

Definition A.1. A binary relation \sim on a set \mathcal{X} is an *equivalence relation* if it satisfies the following three properties for any $x, y, z \in \mathcal{X}$:

Reflexivity $x \sim x$.

Symmetry If $x \sim y$ then $y \sim x$.

Transitivity If $x \sim y$ and $y \sim z$ then $x \sim z$.

Given an equivalence relation, an *equivalence class* is any set of the form $E_x = \{y \in \mathcal{X} \mid y \sim x\}$. If $y \in E_x$ then $E_y = E_x$. Each element $x \in \mathcal{X}$ is in exactly one equivalence class, so \sim induces a partition of \mathcal{X} into equivalence classes.

In Euclidean space, ‘is parallel to’ is an equivalence relation, while ‘is perpendicular to’ is not.

For finite sets, cardinality is a property of a specific set, while for infinite sets, cardinality must be understood as an equivalence relation.

A.4 Matrices

Let $M_{m,n}(\mathcal{K})$ be the set of $m \times n$ matrices A , for which $A_{i,j} \in \mathcal{K}$ (or, when required for clarity, $[A]_{i,j} \in \mathcal{K}$) is the element of the i th row and j th column. When the field need not be given, we will write $M_{m,n} = M_{m,n}(\mathcal{K})$. We will generally be interested in $M_{m,n}(\mathcal{C})$, noting that the real matrices $M_{m,n}(\mathbb{R}) \subset M_{m,n}(\mathcal{C})$ can be considered a special case of complex matrices, so that any resulting theory holds for both types. This is important to note, since even when interest is confined to real valued matrices, complex numbers enter the analysis in a natural way, so it is ultimately necessary to consider complex vectors and matrices. Definitions associated with real matrices (transpose, symmetric, and so on) have analogous definitions for complex matrices, which reduce to the more familiar definitions when the matrix is real.

The *square matrices* are denoted as $M_m = M_{m,m}$. Elements of $M_{m,1}$ are *column vectors* and elements of $M_{1,m}$ are *row vectors*. A matrix in $M_{m,n}$ is equivalently an ordered set of m row vectors or n column vectors. The transpose $A^T \in M_{n,m}$ of a matrix $A \in M_{m,n}$ has elements $A'_{j,i} = A_{i,j}$. For $A \in M_{n,k}$, $B \in M_{k,m}$ we always understand matrix multiplication to mean that $C = AB$ possesses elements $C_{i,j} = \sum_{k'=1}^k A_{i,k'} B_{k',j}$, so that matrix multiplication is generally not commutative (a binary operation \circ is *commutative* if $a \circ b = b \circ a$ for all pairs (a,b) for which the operation is defined). Then $(A^T)^T = A$ and $(AB)^T = B^T A^T$ where the product is permitted.

In the context of matrix algebra, a vector $x \in \mathcal{K}^n$ is usually assumed to be a column vector in $M_{n,1}$. Therefore, if $A \in M_{m,n}$ then the expression Ax is understood to be evaluated by matrix multiplication. Similarly, if $x \in \mathcal{K}^m$ we may use the expression $x^T A$, understanding that $x \in M_{m,1}$.

When $A \in M_{m,n}(\mathcal{C})$, the *conjugate matrix* is written \bar{A} , and is the component-wise conjugate of A . The identity $\bar{AB} = \bar{A}\bar{B}$ holds. The *conjugate transpose* (or *Hermitian adjoint*) of A is $A^* = \bar{A}^T$. As with the transpose operation, $(A^*)^* = A$ and $(AB)^* = B^* A^*$ where the product is permitted. This generally holds for arbitrary products, that is $(ABC)^* = (BC)^* A^* = C^* B^* A^*$, and so on. For $A \in M_{m,n}(\mathbb{R})$, we have $A = \bar{A}$ and $A^* = A^T$, so the conjugate transpose may be used in place of the transpose operation when matrices are real valued. We always may write $(A+B)^* = A^* + B^*$ and $(A+B)^T = A^T + B^T$ where dimensions permit.

A matrix $A \in M_n(\mathcal{C})$ is *diagonal* if the only nonzero elements are on the diagonal, and can therefore be referred to by the diagonal elements $\text{diag}(a_1, \dots, a_n) = \text{diag}(A_{1,1}, \dots, A_{n,n})$. A diagonal matrix is *positive diagonal* or *nonnegative diagonal* if all diagonal elements are positive or nonnegative.

The identity matrix $I \in M_m$ is the matrix uniquely possessing the property that $A = IA = AI$ for all $A \in M_m$. For $M_m(\mathcal{C})$, I is diagonal, with diagonal entries equal to 1. For any matrix $A \in M_m$

there exists at most one matrix $A^{-1} \in M_m$ for which $AA^{-1} = I$, referred to as the *inverse* of A . An inverse need not exist (for example, if the elements of A are constant).

The *inner product* (or *scalar product*) of two vectors $x, y \in \mathcal{C}^n$ is defined as $\langle x, y \rangle = y^*x$. For any $x \in \mathcal{C}^n$ we have $\langle x, x \rangle = \sum_i \bar{x}_i x_i = \sum_i |x_i|^2$, so that $\langle x, x \rangle$ is a nonnegative real number, and $\langle x, x \rangle = 0$ if and only if $x = \vec{0}$. The magnitude, or *norm*, of a vector may be taken as $\|x\| = (\langle x, x \rangle)^{1/2}$.

Two vectors $x, y \in \mathcal{C}^n$ are *orthogonal* if $\langle x, y \rangle = 0$. A set of vectors x_1, \dots, x_m is orthogonal if $\langle x_i, x_j \rangle = 0$ when $i \neq j$. A set of m orthogonal vectors are linearly independent, and so form the basis for an m dimensional vector space. If in addition $\|x_i\| = 1$ for all i , the vectors are *orthonormal*.

A matrix $Q \in M_n(\mathcal{C})$ is *unitary* if $Q^*Q = QQ^* = I$. Equivalently, Q is unitary if and only (i) it's column vectors are orthonormal; (ii) its row vectors are orthonormal; (iii) it possesses inverse $Q^{-1} = Q^*$. The more familiar term *orthogonal matrix* is usually reserved for a real valued unitary matrix (otherwise the definition need not be changed).

A unitary matrix preserves magnitude, since $\langle Qx, Qx \rangle = (Qx)^*(Qx) = x^*Q^*Qx = x^*Ix = x^*x = \|x\|^2$.

A matrix $Q \in M_n(\mathcal{C})$ is a *permutation* matrix if each row and column contains exactly one 1 entry, with all other elements equal to 0. Then $y = Qx$ is a permutation of the elements of $x \in \mathcal{C}^n$. A permutation matrix is always orthogonal.

Suppose $A \in M_{m,n}$ and let $\alpha \subset \{1, \dots, m\}$, $\beta \subset \{1, \dots, n\}$ be any two nonempty subsets of indices. Then $A[\alpha, \beta] \in M_{|\alpha|, |\beta|}$ is the *submatrix* of A obtained by deleting all elements except for $A_{i,j}$, $i \in \alpha$, $j \in \beta$. If $A \in M_n$, and $\alpha = \beta$, then $A[\alpha, \alpha]$ is a *principal submatrix*.

The determinant associates a scalar with $A \in M_m(\mathcal{C})$ through the recursive formula

$$\det(A) = \sum_{i=1} (-1)^{i+j} A_{i,j} \det(A^{i,j}) = \sum_{j=1} (-1)^{i+j} A_{i,j} \det(A^{i,j})$$

where $A^{i,j} \in M_{m-1}(\mathcal{C})$ is the matrix obtained by deleting the i th row and j th column of A . Note that in the respective expressions any j or i may be chosen, yielding the same number, although the choice may have implications for computational efficiency. As is well known, for $A \in M_1(\mathcal{C})$ we have $\det(A) = A_{1,1}$ and for $A \in M_2$ we have $\det(A) = A_{1,1}A_{2,2} - A_{1,2}A_{2,1}$. In general, $\det(A^T) = \det(A)$, $\det(A^*) = \overline{\det(A)}$, $\det(AB) = \det(A)\det(B)$, $\det(I) = 1$ which implies $\det(A^{-1}) = \det(A)^{-1}$ when the inverse exists.

A large class of algorithms is associated with the problem of determining a solution $x \in \mathcal{K}^m$ to the *linear systems of equations* $Ax = b$ for some fixed $A \in M_m$ and $b \in \mathcal{K}^m$.

Theorem A.1. The following statements are equivalent for $A \in M_m(\mathcal{C})$, and a matrix satisfying any one is referred to as *nonsingular*, any other matrix in $M_m(\mathcal{C})$ *singular*:

- (i) The columns vectors of A are linearly independent.
- (ii) The row vectors of A are linearly independent.
- (iii) $\det(A) \neq 0$.

(iv) $Ax = b$ possesses a unique solution for any $b \in \mathcal{K}^m$.

(v) $x = \vec{0}$ is the only solution of $Ax = \vec{0}$.

Matrices $A, B \in M_n$ are *similar*, if there exists a nonsingular matrix S for which $B = S^{-1}AS$. Similarity is an equivalence relation (Section A.3). A matrix is *diagonalizable* if it is similar to a diagonal matrix. Diagonalization offers a number of advantages. We always have $B^k = S^{-1}A^kS$, so that if A is diagonal, this expression is particularly easy to evaluate. More generally, diagonalization can make apparent the behavior of a matrix interpreted as a transformation. Suppose in the diagonalization $B = S^{-1}AS$ we know that S is orthogonal, and that A is real. Then the action of B on a vector is decomposed into S (a change in coordinates), A (elementwise scalar multiplication) and S^{-1} (the inverse change in coordinates).

A.5 Eigenvalues and Spectral Decomposition

For $A \in M_n(\mathcal{C})$, $x \in \mathcal{C}^n$, and $\lambda \in \mathcal{C}$ we may define the *eigenvalue equation*

$$Ax = \lambda x, \quad (\text{A.1})$$

and if the pair (λ, x) is a solution to this equation for which $x \neq \vec{0}$, then λ is an *eigenvalue* of A and x is an associated *eigenvector* of λ . Any such solution (λ, x) may be called an *eigenpair*. Clearly, if x is an eigenvector, so is any nonzero scalar multiple. Let R_λ be the set of all eigenvectors x associated with λ . If $x, y \in R_\lambda$ then $ax+by \in R_\lambda$, so that R_λ is a vector space. The dimension of R_λ is known as the *geometric multiplicity* of λ . We may refer to R_λ as an *eigenspace* (or *eigenmanifold*). In general, the *spectral properties* of a matrix are those pertaining to the set of eigenvalues and eigenvectors.

If $A \in M_n(\mathbb{R})$, and λ is an eigenvalue, then so is $\bar{\lambda}$, with associated eigenvectors $R_{\bar{\lambda}} = \bar{R}_\lambda$. Thus, in this case eigenvalues and eigenvectors occur in conjugate pairs. Similarly, if λ is real there exists a real associated eigenvector.

The eigenvalue equation may be written $(A - \lambda I)x = 0$. However, by Theorem A.1 this has a nonzero solution if and only if $A - \lambda I$ is singular, which occurs if and only if $p_A(\lambda) = \det(A - \lambda I) = 0$. By construction of a determinant, $p_A(\lambda)$ is an order n polynomial in λ , known as the *characteristic polynomial* of A . The set of all eigenvalues of A is equivalent to the set of solutions to the *characteristic equation* $p_A(\lambda) = 0$ (including complex roots). The multiplicity of an eigenvalue λ as a root of $p_A(\lambda)$ is referred to as its *algebraic multiplicity*. A *simple eigenvalue* has algebraic multiplicity 1. The geometric multiplicity of an eigenvalue can be less, but never more, than the algebraic multiplicity. A matrix with equal algebraic and geometric multiplicities for each eigenvalue is a *nondefective matrix*, and is otherwise a *defective matrix*.

We therefore denote the set of all eigenvalues as $\sigma(A)$. An important fact is that $\sigma(A^k)$ consists exactly of the eigenvalues $\sigma(A)$ raised to the k th power, since if (λ, x) solves $Ax = \lambda x$, then $A^2x = A\lambda x = \lambda Ax = \lambda^2 x$, and so on. A quantity of particular importance is the *spectral radius* $\rho(A) = \max\{|\lambda| \mid \lambda \in \sigma(A)\}$. There is sometimes interest in ordering the eigenvalues by magnitude. If there exists an eigenvalue $\lambda_1 = \rho(A)$, this is sometimes referred to as the *principal eigenvalue*, and any associated eigenvector is a *principal eigenvector*.

Suppose we may construct n eigenvalues $\lambda_1, \dots, \lambda_n$, with associated eigenvectors v_1, \dots, v_n . Then let $\Lambda \in M_n$ be the diagonal matrix with i th diagonal element λ_i , and let $V \in M_n$ be the

matrix with i th column vector ν_i . By virtue of (A.1) we can write

$$AV = V\Lambda. \quad (\text{A.2})$$

If V is invertable (equivalently, there exist n linearly independent eigenvectors, by Theorem A.1), then

$$A = V\Lambda V^{-1}, \quad (\text{A.3})$$

so that A is diagonalizable. Alternatively, if A is diagonalizable, then (A.2) can be obtained from (A.3) and, since V is invertable, there must be n independent eigenvectors. The following theorem expresses the essential relationship between diagonalization and spectral properties.

Theorem A.2. For square matrix $A \in M_n(\mathcal{C})$:

- (i) Any set of $k \leq n$ eigenvectors ν_1, \dots, ν_k associated with distinct eigenvalues $\lambda_1, \dots, \lambda_k$ are linearly independent,
- (ii) A is diagonalizable if and only if there exist n linearly independent eigenvectors,
- (iii) If A has n distinct eigenvalues, it is diagonalizable (this follows from (i) and (ii)),
- (iv) A is diagonalizable if and only if it is nondefective.

A.5.1 Right and left eigenvectors

The eigenvectors defined by (A.1) may be referred to as *right eigenvectors*, while *left eigenvectors* are nonzero solutions to

$$x^*A = \lambda x^*, \quad (\text{A.4})$$

(note that some conventions do not explicitly refer to complex conjugates x^* in (A.4)). This similarly leads to the equation $x^*(A - \lambda I) = 0$, which by an argument identical to that used for right eigenvectors, has nonzero solutions if and only if $p_A(\lambda) = 0$, giving the same set of eigenvalues as those defined by (A.1). There is therefore no need to distinguish between ‘right’ and ‘left’ eigenvalues. Then, fixing eigenvalue λ we may refer to the *left eigenspace* L_λ as the set of solution x to (A.4) (in which case, R_λ now becomes the *right eigenspace* of λ).

The essential relationship between the eigenspaces is summarized in the following theorem:

Theorem A.3. Suppose $A \in M_n(\mathcal{C})$.

- (i) For any $\lambda \in \sigma(A)$ L_λ and R_λ have the same dimension.
- (ii) For any distinct eigenvalues $\lambda_1, \dots, \lambda_m$ from $\sigma(A)$, any selection of vectors $x_i \in R_{\lambda_i}$ for $i = 1, \dots, m$ are linearly independent. The same holds for selections from distinct L_λ .
- (iii) Right and left eigenvectors associated with distinct eigenvalues are orthogonal.

Proof. Proofs may be found in, for example, Chapter 1 of *Matrix Analysis*, Horn and Johnson, 1985. \square

Next, if V is invertible, multiply both sides of (A.3) by V^{-1} yielding

$$V^{-1}A = \Lambda V^{-1}.$$

Just as the column vectors of V are right eigenvectors, we can set $U^* = V^{-1}$, in which case the i th column vector v_i of U is a solution x to the left eigenvector equation (A.4) corresponding to eigenvalue λ_i (the i th element on the diagonal of Λ). This gives the diagonalization

$$A = V\Lambda U^*.$$

Since $U^*V = I$, indefinite multiplication of A yields the *spectral decomposition*:

$$A^m = V\Lambda^m U^* = \sum_{i=1}^n \lambda_i^m v_i v_i^*. \quad (\text{A.5})$$

The apparent recipe for a spectral decomposition is to first determine the roots of the characteristic polynomial, and then to solve each resulting eigenvalue equation (A.1) after substituting an eigenvalue. This seemingly straightforward procedure proves to be of little practical use in all but the simplest cases, and spectral decompositions are often difficult to construct using any method. However, a complete spectral decomposition need not be the objective. First, it may not even exist for many otherwise interesting models. Second, there are many important problems related to A that can be solved using spectral theory, but without the need for a complete spectral decomposition. For example:

- (i) Determining bounds $\|Ax\| \leq a \|x\|$ or $\|Ax\| \geq b \|x\|$,
- (ii) Determining the convergence rate of the limit $\lim_{k \rightarrow \infty} A^k = A^\infty$,
- (iii) Verifying the existence of a scalar λ and vector ν for which $A\nu = \lambda\nu$, and guaranteeing that (for example) λ and ν are both real and positive.

Basic spectral theory relies on the identification of special matrix forms which impose specific properties on a the spectrum. We next discuss two cases.

A.6 Symmetric, Hermitian and Positive Definite Matrices

A matrix $A \in M_n(\mathcal{C})$ is *Hermitian* if $A = A^*$. A Hermitian real valued matrix is *symmetric*, that is, $A = A^T$. The spectral properties of Hermitian matrices are quite definitive (see, for example, Chapter 4, *Matrix Analysis*, Horn and Johnson, 1985).

Theorem A.4. A matrix $A \in M_n(\mathcal{C})$ is Hermitian if and only if there exists a unitary matrix U and real diagonal matrix Λ for which $A = U\Lambda U^*$.

A matrix $A \in M_n(\mathbb{R})$ is symmetric if and only if there exists a real orthogonal Q and real diagonal matrix Λ for which $A = Q\Lambda Q^T$.

Clearly, the matrices Λ and U may be identified with the eigenvalues and eigenvectors of A , with n eigenvalue equation solutions given by the respect columns of $AU = U\Lambda U^*U = U\Lambda$. An important implication of this is that all eigenvalues of a Hermitian matrix are real, and eigenvectors may be selected to be orthonormal.

If we interpret $x \in \mathcal{C}^n$ as a column vector $x \in M_{n,1}$ we have *quadratic form* x^*Ax , which is interpretable either as a 1×1 complex matrix, or as a scalar in \mathcal{C} , as is convenient.

If A is Hermitian, then $(x^*Ax)^* = x^*A^*x = x^*Ax$. This means if $z = x^*Ax \in \mathcal{C}$, then $z = \bar{z}$, equivalently $x^*Ax \in \mathbb{R}$. A Hermitian matrix A is *positive definite* if and only if $x^*Ax > 0$ for all $x \neq \vec{0}$. If instead $x^*Ax \geq 0$ then A is *positive semidefinite*. A nonsymmetric matrix satisfying $x^T Ax > 0$ can be replaced by $A' = (A + A^T)/2$, which is symmetric, and also satisfies $x^T A' x > 0$.

Theorem A.5. If $A \in M_n(\mathcal{C})$ is Hermitian then x^*Ax is real. If, in addition, A is positive definite then all of its eigenvalues are positive. If it is positive semidefinite then all of its eigenvalues are nonnegative.

If A is positive semidefinite, and we let λ_{\min} and λ_{\max} be the smallest and largest eigenvalues in $\sigma(A)$ (all of which are nonnegative real numbers) then it can be shown that

$$\lambda_{\min} = \min_{\|x\|=1} x^*Ax \text{ and } \lambda_{\max} = \max_{\|x\|=1} x^*Ax.$$

If A is positive definite then $\lambda_{\min} > 0$. In addition, since the eigenvalues of A^2 are the squares of the eigenvalues of A , and since for a Hermitian matrix $A^* = A$, we may also conclude

$$\lambda_{\min} = \min_{\|x\|=1} \|Ax\| \text{ and } \lambda_{\max} = \max_{\|x\|=1} \|Ax\|,$$

for any positive semidefinite matrix A .

Any diagonalizable matrix A possesses a k th root, $A^{1/k}$, meaning $A = (A^{1/k})^k$. Given diagonalization $A = Q^{-1}\Lambda Q$, this is easily seen to be $A^{1/k} = Q^{-1}\Lambda^{1/k}Q$, where $[\Lambda^{1/k}]_{i,j} = (\Lambda_{i,j})^{1/k}$. If A is a real symmetric positive definite matrix then $A^{1/2}$ is real, symmetric and nonsingular.

Appendix B

Multivariate Distributions

We will need to characterize the distribution of random vectors $\tilde{X} = (X_1, \dots, X_n)$, described in terms of densities, PMFs and CDFs. A random vector can have components of different types (say, X_1 is discrete and X_2 is continuous). However, the usual approach is to assume that all components are of the same type. Once the theory is understood under this restriction, extension to the more general case will be quite natural.

A *discrete* random vector $\tilde{X} = (X_1, \dots, X_n)$ possesses a PMF which assigns a probability $p_{\tilde{X}}(x_1, \dots, x_n) \in [0, 1]$ to each element $\tilde{x} = (x_1, \dots, x_n)$ of a support set \mathcal{S} such that

$$\sum_{\tilde{x} \in \mathcal{S}} p_{\tilde{X}}(\tilde{x}) = 1. \quad (\text{B.1})$$

Then the probability of $E \subset \mathbb{R}^n$ is

$$P(E) = \sum_{\tilde{x} \in E \cap \mathcal{S}} p_{\tilde{X}}(\tilde{x}). \quad (\text{B.2})$$

A *continuous* random vector $\tilde{X} = (X_1, \dots, X_n)$ possess a density function $f_{\tilde{X}}(x_1, \dots, x_n) \leq 0$ which satisfies the condition

$$\int_{\tilde{x} \in \mathbb{R}^n} f_{\tilde{X}}(\tilde{x}) d\tilde{x} = 1. \quad (\text{B.3})$$

Then the probability of $E \subset \mathbb{R}^n$ is

$$P(E) = \int_{\tilde{x} \in E} f_{\tilde{X}}(\tilde{x}) d\tilde{x}. \quad (\text{B.4})$$

The support of \tilde{X} consists of all points \tilde{x} for which $f_{\tilde{X}}(\tilde{x}) > 0$.

Suppose the components of \tilde{X} are independent, and X_i has density f_i . Then the joint density of \tilde{X} is

$$f_{\tilde{X}}(x_1, \dots, x_n) = \prod_{i=1}^n f_i(x_i). \quad (\text{B.5})$$

B.1 Matrix Algebra and Multivariate Distributions

Suppose we have random vector $\tilde{X} = (X_1, \dots, X_m)$. The *mean vector* can be written $\mu_{\tilde{X}} = E[\tilde{X}] = (E[X_1], \dots, E[X_m])$ in the appropriate context. In matrix algebra $\mu_{\tilde{X}}$ is usually interpreted as a column vector.

The $m \times m$ *variance matrix* (also referred to as the *covariance matrix*) of \tilde{X} is defined elementwise as

$$[\Sigma_{\tilde{X}}]_{i,j} = cov [X_i, X_j],$$

where we denote covariance $cov [X, Y] = E[(X - E[X])(Y - E[Y])]$ and consequently $var [X] = cov [X, X]$. Two RVs may be referred to as *linearly independent* if their covariance is zero, although this does not by itself imply independence under the formal definition.

When the context permits we may write $var [\tilde{X}] = \Sigma_{\tilde{X}}$. Since $cov [X, Y] = cov [Y, X]$, $\Sigma_{\tilde{X}}$ is always symmetric. For any linear combination $\tilde{Y} = a_1 X_1 + \dots + a_m X_m$ based on constant coefficients a_i it may be shown that

$$var [\tilde{Y}] = \tilde{a}^T \Sigma_{\tilde{X}} \tilde{a}, \quad (\text{B.6})$$

where $\tilde{a} = (a_1, \dots, a_m)$ is taken to be a column vector. Since a variance is always nonnegative this must mean $\Sigma_{\tilde{X}}$ is positive semidefinite, and is positive definite unless a subset of the elements of \tilde{X} are linearly dependent with probability 1.

Next, suppose b is a $k \times 1$ constant column vector, A is a $k \times m$ constant matrix, and \tilde{X} is a $m \times 1$ random vector. Then

$$\tilde{Y} = b + A\tilde{X}$$

is a linear transformation yielding a $k \times 1$ random vector, consisting of k linear combinations of \tilde{X} . The mean and variance matrices of \tilde{X} and \tilde{Y} are always related by

$$E[\tilde{Y}] = b + AE[\tilde{X}] \text{ and } var [\tilde{Y}] = A (var [\tilde{X}]) A^T.$$

Suppose $var [\tilde{X}]$ is positive definite. Then there exists an invertible symmetric square root matrix $var [\tilde{X}]^{1/2}$ (Section A.6). If $\tilde{Y} = var [\tilde{X}]^{-1/2} \tilde{X}$ then

$$\begin{aligned} var [\tilde{Y}] &= var [\tilde{X}]^{-1/2} var [\tilde{X}] var [\tilde{X}]^{-1/2} \\ &= var [\tilde{X}]^{-1/2} var [\tilde{X}]^{1/2} var [\tilde{X}]^{1/2} var [\tilde{X}]^{-1/2} \\ &= I. \end{aligned}$$

Thus, any random vector with a positive definite variance matrix $var [\tilde{X}]$ possesses a linear transformation yielding linearly independent coordinates of unit variance.

B.2 Multivariate Normal Distribution

Suppose $\tilde{\mu}$ is a $m \times 1$ column vector and Σ is a positive definite $m \times m$ matrix. The *multivariate normal density* function is defined as

$$\begin{aligned} f(x | \tilde{\mu}, \Sigma) &= (2\pi)^{-m/2} \det(\Sigma)^{-1/2} \exp(-Q/2), \quad x \in \mathbb{R}^m, \text{ where} \\ Q &= (x - \tilde{\mu})^T \Sigma^{-1} (x - \tilde{\mu}). \end{aligned} \quad (\text{B.7})$$

Then $\tilde{X} = (X_1, \dots, X_m)$ is a *multivariate normal random vector* if it possesses this density, in which case it may be shown that $E[\tilde{X}] = \tilde{\mu}$, $\text{var}[\tilde{X}] = \Sigma$. In addition, the marginal distributions are $X_i \sim N(\tilde{\mu}_i, \Sigma_{i,i})$. The $m = 2$ case is often referred to as the *bivariate normal* distribution.

It is important to note that a random vector with marginal normal densities is not necessarily multivariate normal. For example, if $X \sim N(0, 1)$ and $Y = SX$ where S is an independent random sign, then $Y \sim N(0, 1)$, $\text{cov}[X, Y] = 0$, but (X, Y) does not possess a multivariate normal density.

The definition of a multivariate normal random vector can be generalized to include any random vector of the form $\tilde{X} = \tilde{\mu} + A\tilde{Z}$, where $\tilde{\mu}$ is an $k \times 1$ column vector, A is an $k \times m$ matrix, and \tilde{Z} is a $m \times 1$ column vector of independent unit normal random variables. In this case $\text{var}[\tilde{X}]$ need not be positive definite, so (B.7) cannot be used directly.

Appendix C

An R Tutorial

R is an interpretive programming environment designed primarily for statistical computations, but which extends into more general applications of optimization and numerical analysis. An extensive library of statistical algorithms is accessible from within R and through curated software repositories such as cran.r-project.org, bioconductor.org and omegahat.org. To a large degree, the operation of R is independent of the operating system, but sometimes OS specific issues arise.

R is based on a command line interpreter. Its commands can also be used to construct high level programs permitting standard looping and conditional execution statements. It has especially powerful graphics abilities.

C.1 Mathematical Operations on Scalars and Vectors

A good first command is

```
> help(Arithmetric)
```

This produces documentation on the use of arithmetic binary operators. The format of the documentation will depend on the operating system.

Description

These binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage

```
x + y  
x - y  
x * y  
x / y  
x ^ y  
x %% y  
x %/% y
```

Arguments

x, y
numeric or complex vectors or objects which can be coerced to such,
or other objects for which methods have been written.

<remaining documentation omitted>

To make use of these operators we can assign numbers to variables x and y:

```
> x = 13
> y = 4
> x + y
[1] 17
> x - y
[1] 9
> x * y
[1] 52
> x / y
[1] 3.25
> x ^ y
[1] 28561
> x %% y
[1] 1
> x %/% y
[1] 3
```

Comments are defined by the # symbol

```
> # this is a comment
> x = 3
> # x = 2
> x
[1] 3
>
```

Assignment can also be done with the syntax

```
> x <- 13
> y <- 4
> x
[1] 13
> y
[1] 4
>
```

Note that R is case sensitive:

```

> a = 4
> A = 3
> a - A
[1] 1
> help(arithmetic)
No documentation for arithmetic in specified packages and libraries:
you could try ??arithmetic
>

```

C.1.1 Vectors in R

In the above examples, an index reference [1] is given whenever a variable is displayed (simply by typing the name of the variable). This is because R considers a single number as a special case of a vector (or array). One of the advantages of R for statistical computation is the manner in which vectors can be manipulated almost as easily as variables. Vectors are easily constructed using the `c()` function (you can always enter `help(c)`):

```

> x = c(3.4, -1, 99, 1/2)
> x
[1] 3.4 -1.0 99.0 0.5

```

An element of a vector is referenced using square brackets:

```

> x = c(3.4, -1, 99, 1/2)
> x[2]
[1] -1
> i = 3
> x[i]
[1] 99

```

Note that the `c()` function accepts mathematical expressions as argument (such as $1/2$ or $\sin(0.075)$), which are evaluated when the vector object is created.

Scalar operations on vectors follow intuitive rules:

```

> x = c(1, 3, 10, 99.3)
> x
[1] 1.0 3.0 10.0 99.3
> x+1
[1] 2.0 4.0 11.0 100.3
> x*10
[1] 10 30 100 993

```

Addition and multiplication of two vectors of equal length is done by element:

```

> x = c(1, 3, 10, 99.3)
> y = c(1, 1, 2, 2)
> x*y

```

```
[1] 1.0 3.0 20.0 198.6
> x + y
[1] 2.0 4.0 12.0 101.3
> x + 2*y
[1] 3.0 5.0 14.0 103.3
> x + 2*y*x*x
[1] 3.00 21.00 410.00 39541.26
```

It is important to note that R will *attempt* to add, subtract, multiply or divide vectors of unequal length. The decision made by R on your behalf is to recycle the shorter length vector enough times to permit the operation. If the length of the longer vector is not a multiple of the length of the shorter vector, R will give a warning:

```
> c(1,2,3) + c(10,20)
[1] 11 22 13
Warning message:
In c(1, 2, 3) + c(10, 20) :
  longer object length is not a multiple of shorter object length
```

In the above example, the shorter vector `c(10,20)` was extended to `c(10,20,10)` for the purposes of evaluation. In the following example, the shorter vector `c(10,20)` is extended to `c(10,20,10,20)`. In this case R will give no warning:

```
> c(1,2,3,4) + c(10,20)
[1] 11 22 13 24
```

The following examples should give an idea how this works. Note that only the last expression produces a warning.

```
> c(1,2,1,2,1,2,1,2) + c(10,20)
[1] 11 22 11 22 11 22 11 22
> c(100,200) + c(1,2,1,2,1,2,1,2)
[1] 101 202 101 202 101 202 101 202
> c(100,200)*c(1,2,1,2,1,2,1,2)
[1] 100 400 100 400 100 400 100 400
> c(50,75)/c(1,2,1,2,1,2,1,2)
[1] 50.0 37.5 50.0 37.5 50.0 37.5 50.0 37.5
> c(100,200) - c(1,2,1,2,1,2,1,2)
[1] 99 198 99 198 99 198 99 198
> c(100,200) + c(1,2,1,2,1,2,1,2,1)
[1] 101 202 101 202 101 202 101 202 101
Warning message:
In c(100, 200) + c(1, 2, 1, 2, 1, 2, 1, 2, 1) :
  longer object length is not a multiple of shorter object length
```

Special vectors can be created in a number of ways:

```
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> 1:5 + 5:1
[1] 6 6 6 6 6
> rep(3,10)
[1] 3 3 3 3 3 3 3 3 3 3
> rep(3:1,4)
[1] 3 2 1 3 2 1 3 2 1 3 2 1
> seq(0, 1, 0.25)
[1] 0.00 0.25 0.50 0.75 1.00
> seq(0, 1, 0.27)
[1] 0.00 0.27 0.54 0.81
```

The length of a vector is given by the function `length()`:

```
> x = rep(3:1,4)
> length(x)
[1] 12
```

Vectors of length 0 are defined in R and are assigned the symbol `NULL`

```
> x = NULL
> length(x)
[1] 0
> x = null
Error: object 'null' not found
```

R also has a missing value indicator `NA` which may be used in place of a number (formally, an element of a vector):

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

Conventions for dealing with missing values will depend on the particular function. For example, we can input a vector into the function `sum()`:

```
> x = c(2,4,6)
> sum(x)
[1] 12
```

If the vector has a missing value we get the following result:

```
> x = c(2,NA,6)
> sum(x)
[1] NA
```

unless we specify that missing values are to be removed by setting the remove missing value option `na.rm` to TRUE:

```
> x = c(2,NA,6)
> sum(x, na.rm=TRUE)
[1] 8
> sum(x, na.rm=T)
[1] 8
> sum(x, na.rm=TR)
Error: object 'TR' not found
```

If we consult the `sum()` documentation by using `help(sum)` we find that `na.rm = FALSE` is the default option.

R has some flexibility regarding undefined numbers, for example:

```
> 1/0
[1] Inf
> 1/c(0,0,0)
[1] Inf Inf Inf
> x = 1/c(0,0,0)
> x
[1] Inf Inf Inf
```

Note that R offers considerable flexibility when appending elements to vectors. A reference to an element beyond the length of the vector yields `NA`. However, an assignment to an element beyond the length of the vector forces an extension of the length of the vector.

```
> x = c(5,4,3,2,1)
> x
[1] 5 4 3 2 1
> x[10]
[1] NA
> x[10] = 999
> x
[1] 5 4 3 2 1 NA NA NA NA 999
```

C.1.2 Global options

Many system options can be changed. The function `options()` can be used to display the value of an option by inputting a character string holding the option name. For example, the character used as the R command line is an option with the name ‘prompt’:

```
> options("prompt")
$prompt
[1] "> "
```

We can also change options within the `option()` function argument. For example, we can include an inspirational message in the prompt:

```
> options(prompt = "Do It Right the First Time> ")
Do It Right the First Time> options('prompt')
$prompt
[1] "Do It Right the First Time> "
```

If we eventually change our mind, we can always return to the original prompt:

```
Do It Right the First Time> options(prompt = "> ")
> options('prompt')
$prompt
[1] "> "

> "That's better ..."
[1] "That's better ..."
```

This should be done sparingly, but, occasionally, a contributed function may change an option, so it's good to be aware of this possibility.

To see all options enter the `option()` command without argument:

```
> options()
$add.smooth
[1] TRUE

$bitmapType
[1] "quartz"

$browser
[1] "/usr/bin/open"

$browserNLdisabled
[1] FALSE

$CBoundsCheck
[1] FALSE

$check.bounds
[1] FALSE

$citation.bibtex.max
[1] 1

$continue
[1] "+ "

<remaining output omitted>
```

C.1.3 Modes (or types)

An element of a vector is stored as one of several types (in R the term *mode* is also used). Numerical data can be stored as an *integer* or a *double* type (that is, integer or real). Usually, operations in R are not highly dependent on the difference between integer and real.

```
> x = 4:6
> x[2]
[1] 5
> x[2.2]
[1] 5
> x[2.5]
[1] 5
> x[2.9]
[1] 5
> x[2.99999]
[1] 5
> x[3]
[1] 6
```

The distinction, however, is important when R passes data to other programs written in, for example, C++.

It is important to realize that the number stored is not necessarily the number displayed:

```
> x = 1.2344556543
> x
[1] 1.234456
> y = x - 1.234456
> y == 0
[1] FALSE
> y
[1] -3.457e-07
```

There is a default rounding of 7 significant digits for R displays.

```
> options('digits')
$digits
[1] 7

> 2.3433345994
[1] 2.343335
> options(digits=3)
> 2.3433345994
[1] 2.34
> options(digits=7)
> 2.3433345994
[1] 2.343335
> options('digits')
```

```
$digits
[1] 7
```

Types can be *coerced* into other types. Real numbers are rounded down when this happens.

```
> 3 == 2.9999999
[1] FALSE
> as.integer(2.9999999)
[1] 2
> as.integer(3.0000001)
[1] 3
```

Real numbers can be rounded off (`round()`, `signif()`), or converted to integers with the `floor()` or `ceiling()` functions, or with the `round()` function with option `digits=0`:

```
> x = 34.59996
> round(x,3)
[1] 34.6
> round(x,5)
[1] 34.59996
> round(x,0)
[1] 35
> ceiling(x)
[1] 35
> floor(x)
[1] 34
> round(0.003344,3)
[1] 0.003
> signif(0.003344,3)
[1] 0.00334
```

Complex numbers are also supported (see `help(complex)`).

Character strings can be stored in variables or vectors:

```
> days = c("Monday", "Tuesday", "Wednesday", "Thursday",
  "Friday", "Saturday", "Sunday")
> days
[1] "Monday"      "Tuesday"      "Wednesday"   "Thursday"
[5] "Friday"       "Saturday"     "Sunday"
```

Vectors can also be of *logical* type, taking values TRUE and FALSE (or T and F). These can be entered directly, or be produced as the value of a *conditional expression*:

```
> 4 > 1
[1] TRUE
> 4 < 1
[1] FALSE
> 4 == 6
```

```
[1] FALSE
> 4 > 4
[1] FALSE
> 4 >= 4
[1] TRUE
> 4 <= 4
[1] TRUE
> 4 != 6
[1] TRUE
```

Note that `==` is the logical relationship ‘is equal to’, while `=` is the assign operator. Confusing the two can lead to bugs which might be hard to detect. Also, `!=` means ‘is not equal to’. As expected `<,>,<=,>=` mean ‘less than’, ‘greater than’, ‘less than or equal to’, ‘greater than or equal to’.

```
> x = c(2<2, 2>2, 2<=2, 2>=2, 2==2, 2!=2)
> x
[1] FALSE FALSE TRUE TRUE TRUE FALSE
```

The logical (or Boolean) operators are `!` (negation); `&` or `&&` (logical AND); `|` or `||` (logical OR), `xor()` (exclusive OR). The difference between `&` and `&&` is that `&` yields elementwise evaluation of vectors, while `&&` is intended for single Boolean values. It should be noted that `&&` can be applied to vectors, but will examine elements left to right, using only the first. Confusing the two forms can lead to unexpected results. The same comment applies to `|` and `||`. The function `xor()` accepts two logical values as arguments, returning `TRUE` if and only one of the arguments equals `TRUE`. This function can do elementwise evaluation of vectors:

```
> x = T
> y = F
> x & y
[1] FALSE
> x && y
[1] FALSE
> x | y
[1] TRUE
> x || y
[1] TRUE
> x == y
[1] FALSE
> x != y
[1] TRUE
> xor(x,y)
[1] TRUE
>
> # try vectors
> x = c(T,F)
> y = c(F,F)
> x & y
```

```
[1] FALSE FALSE
> x && y
[1] FALSE
> x | y
[1] TRUE FALSE
> x || y
[1] TRUE
> xor(x,y)
[1] TRUE FALSE
```

Note that logical vectors can be constructed by applying conditional statements to other vectors:

```
> x = seq(0, 50, 7)
> x
[1] 0 7 14 21 28 35 42 49
> z = (x<25)
> z
[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
> z = (x %% 3 == 0)
> z
[1] TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
```

In the preceding example, we identify all elements of a vector `x` which are less than 25, and which are divisible by 3.

Modes have specific functions associated with them to create objects and test for object type:

```
# 
# Make integer object
#
x = integer()
x
length(x)
#
# Give it length 1
#
x = integer(1)
x
length(x)
#
# Create an integer vector of length 5
#
x = integer(5)
x
length(x)
#
# Verify that it is an integer object
#
```

```
is.integer(x)
#
# It's not a double precision object
#
is.double(x)
#
# Make a double precision, logical, complex and character object
#
double(5)
logical(5)
complex(5)
character(5)
#
# Or, we could just make a numeric vector
#
numeric(5)
#
# What is the mode?
#
x = double(5)
mode(x)
x = logical(5)
mode(x)
x = complex(5)
mode(x)
x = character(5)
mode(x)
x = numeric(5)
mode(x)
```

C.1.4 Index referencing

R has a flexible system of index references:

```
x = 2*(1:10)
> x
[1]  2  4  6  8 10 12 14 16 18 20
> x[c(3,6,9)]
[1]  6 12 18
> y = x[c(3,6,9)]
> length(y)
[1] 3
> y
[1]  6 12 18
```

This makes it straightforward to create subvectors.

C.1.5 More vector operations

The function `sort()` returns a sorted vector, while `sort.list()` returns a vector of indices which generates a sorted list. The function `rev()` returns a vector in reverse order, while `rank()` returns the ranks of the elements of a vector. By default, ties are represented by average ranks, although other options are available (see `help(rank)`).

```
> sort(c(2,4,5,3,2))
[1] 2 2 3 4 5
> x = c(2,4,5,3,2)
> sort(x)
[1] 2 2 3 4 5
> ind = sort.list(x)
> x[ind]
[1] 2 2 3 4 5
> rev(x)
[1] 2 3 5 4 2
> rank(x)
[1] 1.5 4.0 5.0 3.0 1.5
```

One useful function is `unique()` which returns the unique values of a vector:

```
> unique(c(1,2,2,3))
[1] 1 2 3
> unique(c(6,7,6,5,5,3,2,2,3))
[1] 6 7 5 3 2
> unique(c("Bob","bob","Mike", "Bob", "Bob "))
[1] "Bob"  "bob"  "Mike" "Bob "
```

Set operations can be performed on vectors. The `%in%` operator can be used to test for the presence of an element in a vector. It returns a logical value:

```
> 3 %in% c(1,2,3,4)
[1] TRUE
> 3 %in% c(1,2,4)
[1] FALSE
> "WNT3" %in% c("CCR5","SFRP","WNT3")
[1] TRUE
> "WNT3" %in% c("Ccr5","Sfrp","Wnt3")
[1] FALSE
```

The first argument may be a vector. In this case the test is applied to each element of the first vector, the second vector remaining fixed for each test. The result is therefore a logical vector equal in length to the first vector:

```
> c(2,3) %in% c(1,2,3,4)
[1] TRUE TRUE
> c(2,3,3,5,4,5,3) %in% c(1,2,3,4)
[1] TRUE TRUE TRUE FALSE TRUE FALSE TRUE
```

Formal set algebra is defined in R, in particular, `union()` ($A \cup B$), `intersect()` ($A \cap B$), `setdiff()` ($A \cap B^c$). Note that `setdiff()` is the *asymmetric* set difference, not the *symmetric* set difference ($(A \cap B^c) \cup (B \cap A^c)$).

```
> x = c(1,2,2,3,3,4)
> y = c(3,4,4,4,5,6,7,8,9)
> union(x,y)
[1] 1 2 3 4 5 6 7 8 9
> intersect(x,y)
[1] 3 4
> setdiff(x,y)
[1] 1 2
> setdiff(y,x)
[1] 5 6 7 8 9
```

The function `setequal()` ($A = B$) tests for equality of vectors *regarded as sets*. It evaluates to a logical value, which is TRUE if and only if every element in one vector may be found in the other vector. Note that `unique()` returns a *vector* of the unique values in a vector, and so is ordered, that is, it is not a set.

```
> setequal(c(1,2),c(1,2))
[1] TRUE
> setequal(c(1,2),c(2,1))
[1] TRUE
> setequal(c(1,2),c(2,1,2,1,2,2))
[1] TRUE
```

Finally, `is.element(x,y)` is identical to `x %in% y`. See discussion below on *binary operators*.

Set operations may be performed on general types:

```
> gene.list.1 = c('ALDH4', 'A1AP2B1', 'BBC3', 'BCL2', 'CDC42BPA',
+                 'CDC42', 'CDCA7', 'CENPA', 'CDC42', 'CDCA7', 'CENPA',
+                 'CMC2', 'DHX58', 'DEXH', 'DIAPH3', 'DTL')
> gene.list.2 = c('A1AP2B1', 'BBC3', 'CDC42', 'CDCA7', 'CENPA',
+                 'CMC2', 'COL4A2', 'DCK', 'DHX58', 'DEXH', 'DIAPH3',
+                 'DTL')
> #
> # Which genes are in both lists?
> #
> intersect(gene.list.1, gene.list.2)
[1] "A1AP2B1" "BBC3"    "CDC42"   "CDCA7"   "CENPA"   "CMC2"
      "DHX58"   "DEXH"    "DIAPH3"  "DTL"
```

C.1.6 Pattern matching

R has quite extensive pattern matching capabilities. For example, using `grep()` we can specify a character pattern, then determine indices of a character vector containing that pattern:

```
> gene.list.1 = c('ALDH4', 'A1AP2B1', 'BBC3', 'BCL2', 'CDC42BPA',
+                 'CDC42', 'CDCA7', 'CENPA', 'CDC42', 'CDCA7', 'CENPA',
+                 'CMC2', 'DHX58', 'DEXH', 'DIAPH3', 'DTL')
> ind = grep('CDC', gene.list.1)
> ind
[1] 5 6 7 9 10
> gene.list.1[ind]
[1] "CDC42BPA" "CDC42"     "CDCA7"      "CDC42"      "CDCA7"
```

The function `grepl()` performs the same function but returns a logical vector:

```
ind = grepl('CDC', gene.list.1)
> ind
[1] FALSE FALSE FALSE FALSE TRUE  TRUE  TRUE FALSE
      TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
> gene.list.1[ind]
[1] "CDC42BPA" "CDC42"     "CDCA7"      "CDC42"      "CDCA7"
```

See also function `regexp()` and `gregexpr()`. Pattern replacement is done with functions `sub()` and `gsub()`. Most of these function support Perl-style regular expressions, by setting option `perl=TRUE`.

The functions `substr()` and `strsplit()` can be used to extract or replace patterns within a single character string.

C.1.7 Managing objects

There are a number of functions useful for managing R objects. The function `ls()` can be used to list currently available objects. It can be used without argument to list all objects:

```
> ls()
[1] "f"      "junk"   "junka"  "junkb"  "x"      "x1"    "x2"    "y"
```

Conditional lists can be made (see `help(ls)`).

Objects can be removed using the `rm()` command.

```
> ls()
[1] "f"      "junk"   "junka"  "junkb"  "x"      "x1"    "x2"    "y"
> rm(y)
> ls()
[1] "f"      "junk"   "junka"  "junkb"  "x"      "x1"    "x2"
```

Be aware that `ls()` will return a list of all objects, which, combined with `rm()` can be used to remove all objects:

```
> rm(list = ls())
> ls()
character(0)
```

If the previous example is examined carefully, it will be noticed that no warning prompt was given before R removed all objects.

C.2 Data Structures in R

Besides vectors, data can be stored in matrices, as well as objects referred to as *arrays*, *lists* and *data frames*.

C.2.1 Matrices

A matrix is essentially a vector with multiple indices. A two dimensional matrix can be created with the function:

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
      dimnames = NULL)
```

The number of rows and columns are given explicitly by the `nrow` and `ncol` parameters. If a single value is specified for the data, all matrix elements will equal that value (with `NA` as default). It is also possible to input as data a vector of length `nrow × ncol` to be copied sequentially into the matrix either by row or by column, as specified by the `byrow` option. Several examples follow:

```
> matrix(data=c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> matrix(data=c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=F)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(data=c(1,2,3,4,5,6), nrow=2, ncol=3)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(data=99, nrow=2, ncol=3)
     [,1] [,2] [,3]
[1,]   99   99   99
[2,]   99   99   99
> matrix(nrow=2, ncol=3)
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

The function `dim()` is used to either set or retrieve the dimensions of a matrix (row then column). For example:

```
> x = matrix(data=99, nrow=2, ncol=3)
> dim(x)
[1] 2 3
```

Dimensions can also be set for a vector of data:

```
> x = 1:12
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12
> dim(x) = c(4,3)
> x
[,1] [,2] [,3]
[1,] 1 5 9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
```

At this point, we can introduce the matrix transpose function `t()`:

```
> x = 1:12
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12
> dim(x) = c(3,4)
> x
[,1] [,2] [,3] [,4]
[1,] 1 4 7 10
[2,] 2 5 8 11
[3,] 3 6 9 12
> y = t(x)
> y
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
[4,] 10 11 12
```

This allows us to impose 4×3 dimensions onto a vector using by row sequence.

A diagonal matrix can be constructed from a vector with the `diag()` function:

```
> diag(rep(1,10))
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 1 0 0 0 0 0 0 0 0 0
[2,] 0 1 0 0 0 0 0 0 0 0
[3,] 0 0 1 0 0 0 0 0 0 0
[4,] 0 0 0 1 0 0 0 0 0 0
[5,] 0 0 0 0 1 0 0 0 0 0
[6,] 0 0 0 0 0 1 0 0 0 0
[7,] 0 0 0 0 0 0 1 0 0 0
[8,] 0 0 0 0 0 0 0 1 0 0
[9,] 0 0 0 0 0 0 0 0 1 0
[10,] 0 0 0 0 0 0 0 0 0 1
```

When the input to `diag()` is a matrix, the diagonal elements are returned:

```
> matrix(c(1,2,3,4),2,2)
 [,1] [,2]
[1,]    1    3
[2,]    2    4
> diag(matrix(c(1,2,3,4),2,2))
[1] 1 4
```

Matrix multiplication is carried out by the operator `%*%`:

```
> A = matrix(1:12,3,4)
> B = diag(c(1,2,2))
> A
 [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> B
 [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    2
> C = B%*%A
> C
 [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    4   10   16   22
[3,]    6   12   18   24
```

A matrix may be constructed by appending rows or columns using the `rbind()` or `cbind()` function:

```
> A = rbind(c(1,0,0,0), c(1,1,0,0), c(1,1,1,0), c(1,1,1,1))
> A
 [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
> B = cbind(c(1,0,0,0), c(1,1,0,0), c(1,1,1,0), c(1,1,1,1))
> B
 [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    0    1    1    1
[3,]    0    0    1    1
[4,]    0    0    0    1
```

The linear system of equations

$$b = Ax$$

for unknown vector x can be solved by the function `solve()`:

```
> A = rbind(c(1,0,0,0), c(1,1,0,0), c(1,1,1,0), c(1,1,1,1))
> A
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
> b = 1:4
> b
[1] 1 2 3 4
> x = solve(A,b)
> x
[1] 1 1 1 1
> A%*%x
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
> dim(x)
NULL
```

Note that in the operator `A%*%x` of the previous example, a matrix dimension 4×1 was assumed for x , even though it is not a matrix. In this case, the dimension value of x is interpreted as an empty vector.

If we want just the inverse of a matrix we include only that matrix in `solve()` function:

```
> A = rbind(c(1,0,0,0), c(1,1,0,0), c(1,1,1,0), c(1,1,1,1))
> A
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
> solve(A)
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]   -1    1    0    0
[3,]    0   -1    1    0
[4,]    0    0   -1    1
> solve(A)%*%A
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
```

```
[3,] 0 0 1 0
[4,] 0 0 0 1
```

C.2.2 More on index subsets

The indexing of matrices is flexible. Recall that subvectors can be created from vectors using vectors of indices within the square brackets:

```
> x = seq(1,100,7)
> x
[1] 1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
> length(x)
[1] 15
> x[c(3,6:9)]
[1] 15 36 43 50 57
> x[]
[1] 1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
> x[NULL]
numeric(0)
```

In the preceding example, a subvector consisting of the 3rd, 6th, 7th, 8th and 9th element of a vector of length 15 is displayed. Leaving the square brackets empty yields the entire vector, while using `NULL` as the index vector yields a zero length vector.

The same principle applies to matrices and matrix like objects, except that each dimension is subsetted. If the index set is empty all elements of that dimension are included. This allows row and column vectors to be extracted from a matrix.

```
> A = cbind(c(1,26,1,1), c(7,76,7,7), c(4,43,4,4), c(3,39,3,3))
> A
[,1] [,2] [,3] [,4]
[1,] 1 7 4 3
[2,] 26 76 43 39
[3,] 1 7 4 3
[4,] 1 7 4 3
> A[,2:3]
[,1] [,2]
[1,] 7 4
[2,] 76 43
[3,] 7 4
[4,] 7 4
> A[1:3,]
[,1] [,2] [,3] [,4]
[1,] 1 7 4 3
[2,] 26 76 43 39
[3,] 1 7 4 3
> A[1,]
```

```
[1] 1 7 4 3
> A[,4]
[1] 3 39 3 3
> A[2:3,3:4]
 [,1] [,2]
[1,] 43 39
[2,] 4 3
> A[2:3,3]
[1] 43 4
```

Note that a column or row vector (or subvector) is not a matrix. That is `A[2:3,3]` yields a vector of length 2, and not a 2×1 matrix.

Negative indices can be used to *remove* elements from a vector or matrix:

```
> x = c(3:10)
> x
[1] 3 4 5 6 7 8 9 10
> x[-1]
[1] 4 5 6 7 8 9 10
> x[-(4:7)]
[1] 3 4 5 10
```

Also, logical vectors may be used to subset indices.

```
> x = 1:4
> z = c(T,T,F,T)
> x[z]
[1] 1 2 4
> x[c(FALSE,TRUE)]
[1] 2 4
> x[c(FALSE,TRUE,TRUE)]
[1] 2 3
> x = 1:50
> x[x%%7==0]
[1] 7 14 21 28 35 42 49
```

In the last command, all positive integers not greater than 50 which are divisible by 7 have been enumerated.

Normally, logical vectors used in the way are of the same length as the vector. If this is not the case, R will recycle the logical vector up to the same length.

The function `which()` can be used to identify the indices of TRUE elements of a logical vector:

```
> which(c(T,T,F,T))
[1] 1 2 4
```

This function is very useful when vector Boolean expressions are used. Repeated the previous example:

```
> x = seq(0, 50, 7)
> x
[1] 0 7 14 21 28 35 42 49
> ind = which(x%%3 == 0)
> ind
[1] 1 4 7
> x[ind]
[1] 0 21 42
```

C.2.3 Lists

A list is a labeled or ordered collection of any type of object. It has a number of uses, including the organization of function input and output, or the storage or irregular forms of data.

```
> input.obj = list(init.values = c(0.2, 0.1, 10), tolerance = 0.001,
maxIter = 100, memo.label = "July 3, 2013")
> input.obj
$init.values
[1] 0.2 0.1 10.0

$tolerance
[1] 0.001

$maxIter
[1] 100

$memo.label
[1] "July 3, 2013"

> input.obj[[1]]
[1] 0.2 0.1 10.0
> input.obj[[2]]
[1] 0.001
> input.obj[[3]]
[1] 100
> input.obj[[4]]
[1] "July 3, 2013"
>
```

Elements of a list can be addressed either by their label or by index (using double square brackets).

Lists of a specified length may be created in the following way

```
> xlist = vector('list', 4)
> xlist
[[1]]
NULL
```

```

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

> xlist[[3]] = "3rd entry in list"
> names(xlist) = paste('memo',1:4)
> xlist
$`memo 1`
NULL

$`memo 2`
NULL

$`memo 3`
[1] "3rd entry in list"

$`memo 4`
NULL

```

As is the case for vectors, R offers considerable flexibility in constructing lists. The function `list()` with no argument creates a list of length 0. The length can be extended to `k` by making an assignment to the `k`th element.

```

> x = list()
> x
list()
> length(x)
[1] 0
> xlist = list()
> xlist
list()
> length(xlist)
[1] 0
> xlist[[4]] = '4th element'
> xlist
[[1]]
NULL

[[2]]
NULL

```

```

[[3]]
NULL

[[4]]
[1] "4th element"

>

```

The function `split()` creates a list by separating elements of a vector `x` by groups defined in `y`:

```

> x = c(1,2,3,4,5,6,7,8)
> y = rep(1:2,4)
> x
[1] 1 2 3 4 5 6 7 8
> y
[1] 1 2 1 2 1 2 1 2
> xy = split(x,y)
> xy
$'1'
[1] 1 3 5 7

$'2'
[1] 2 4 6 8

```

C.2.4 Data frames

The data frame is an important type of object in R. In statistical applications data often consists of samples of *records*, or collections of various forms of data in a fixed structure. This has a tabular form, but it need not be a matrix. In R is it taken to be a list of vectors of common length, in which the vectors may be of various types. These vectors can be assembled using the `data.frame()` command:

```

> patient.id = c(101,102,103)
> gender = c('M', 'M', 'F')
> bmi = c(103, NA, 131)
> bmi.data = data.frame(patient.id, gender, bmi)
> bmi.data
  patient.id gender bmi
1         101      M 103
2         102      M   NA
3         103      F 131
> bmi.data
  patient.id gender bmi
1         101      M 103
2         102      M   NA

```

```

3      103      F 131
> bmi.data[[1]]
[1] 101 102 103
> bmi.data$gender
[1] M M F
Levels: F M
>

```

Note that as a list the columns of a data frame can be accessed by label or by index.

C.2.5 Factors

Note that in the previous example of Section C.2.4, although the vector `gender` was created as vector of character strings, within the data frame it is interpreted as a vector of `factors`, which is nonnumerical data assuming one of a finite number of well defined `levels`. This is the standard way of representing *categorical data* in R, so it is important to understand the various conventions.

A character vector can (sometimes) be interpreted as a factor but, for example, an integer vector must be coerced.

```

> x = c(2,3,2,2,1,3,2)
> x = as.factor(x)
> x
[1] 2 3 2 2 1 3 2
Levels: 1 2 3
> attributes(x)
$levels
[1] "1" "2" "3"

$class
[1] "factor"

```

A vector of factors includes, as an object, the `levels` giving the possible values (categories). This means a level need not be represented in the vector:

```

> y = x[1:4]
> y
[1] 2 3 2 2
Levels: 1 2 3

```

Often, when a factor is an appropriate input, a character vector, or even a numerical vector can be used instead. However, to avoid ambiguity, it is best to create factors when appropriate.

C.2.6 Arrays

An array object is a multidimensional array of elements of common type. A matrix is a two-dimensional array. However, a vector is not a one-dimensional array, because it has no dimension structure. It can be created with the `array()` function, in a manner similar to, but more general than, the `matrix()` function:

```

> array(data = 1:12, dim = c(2,6))
[,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    7    9   11
[2,]    2    4    6    8   10   12
> array(data = 1:12, dim = c(12))
[1] 1 2 3 4 5 6 7 8 9 10 11 12
> array(data = 1:12, dim = c(2,2,3))
, , 1

[,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

[,1] [,2]
[1,]    5    7
[2,]    6    8

, , 3

[,1] [,2]
[1,]    9   11
[2,]   10   12

> array(data = 1:12, dim = c(2,2,3))[1,2,1]
[1] 3
> array(data = 1:12, dim = c(2,2,3))[, ,2]
[,1] [,2]
[1,]    5    7
[2,]    6    8

```

Arrays with 3 or more dimensions can be created, and managed using the same type of indexing used by matrices, but with 3 or more indices.

C.3 Labels for Data Structures

One very important feature of R is the assignment of labels to objects (vectors, matrices, lists, data frames). We will look at functions `names()`, `rownames()`, `colnames()`, `row.names()`, `dimnames()`. Note that `names()` is a generic function, meaning that it's exact function depends on the type of object to which it is applied.

This is a good point at which to introduce the `paste()` function, which concatenates character strings

```

> paste("Elvis","Presley")
[1] "Elvis Presley"

```

```
> paste("Elvis","Presley",sep="-")
[1] "Elvis-Presley"
> paste("Elvis","Presley",sep="")
[1] "ElvisPresley"
```

The `sep` option defines the separator. The default is a single blank.

Numbers are coerced to character strings, and vectors remain vectors:

```
> paste(1:6)
[1] "1" "2" "3" "4" "5" "6"
> paste(1:6)[2]
[1] "2"
> paste(1:6)[2:5]
[1] "2" "3" "4" "5"
```

Vector concatenation can be useful:

```
> paste('gene', 1:12, sep='')
[1] "gene1"   "gene2"   "gene3"   "gene4"   "gene5"   "gene6"   "gene7"
"gene8"   "gene9"   "gene10"  "gene11"  "gene12"
> paste('gene', 1:12, sep='-')
[1] "gene-1"  "gene-2"  "gene-3"  "gene-4"  "gene-5"  "gene-6"
"gene-7"  "gene-8"  "gene-9"  "gene-10" "gene-11"
[12] "gene-12"
```

C.3.1 Vector labels

The `names()` function can be used to set and access labels assigned to the elements of a vector. For example, a function might return an estimate, a standard error and a p-value for an associated hypothesis test. These three values can be placed in a single vector, and labels assigned to identify the values:

```
> x = c(20.3, 1.02, 0.023)
> names(x) = c("Est", "SE", "P-value")
> x
      Est       SE P-value
20.300  1.020  0.023
> names(x)
[1] "Est"      "SE"       "P-value"
> names(x)[2]
[1] "SE"
> names(x)[2] = "S.E."
> x
      Est     S.E. P-value
20.300  1.020  0.023
```

C.3.2 Matrix and array labels

Matrix labels can be created, accessed changed in the much the same way using the `rownames()` and `colnames()` function:

```
> m = matrix(1:24,4,6)
> m
 [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
> rownames(m) = paste('gene',1:4,sep='')
> colnames(m) = paste('subject',1:6,sep='')
> m
      subject1 subject2 subject3 subject4 subject5 subject6
gene1        1        5        9       13       17       21
gene2        2        6       10       14       18       22
gene3        3        7       11       15       19       23
gene4        4        8       12       16       20       24
> rownames(m) = paste('protein',1:4,sep='')
> m
      subject1 subject2 subject3 subject4 subject5 subject6
protein1       1        5        9       13       17       21
protein2       2        6       10       14       18       22
protein3       3        7       11       15       19       23
protein4       4        8       12       16       20       24
```

The combined row and column labels also exist as a single object, namely, a list of two vectors (consisting of the row and column labels), which can be managed with the `dimnames()` function:

```
> dimnames(m)
[[1]]
[1] "protein1" "protein2" "protein3" "protein4"

[[2]]
[1] "subject1" "subject2" "subject3" "subject4" "subject5" "subject6"

> dimnames(m)[[2]]
[1] "subject1" "subject2" "subject3" "subject4" "subject5" "subject6"
> dimnames(m)[[2]] = paste('patient',1:6,sep='')
> m
      patient1 patient2 patient3 patient4 patient5 patient6
protein1       1        5        9       13       17       21
protein2       2        6       10       14       18       22
protein3       3        7       11       15       19       23
protein4       4        8       12       16       20       24
```

For arrays of general dimension the `dimnames()` function can be used in much the same way it is used for matrices

```
> m = array(1:8, c(2,2,2))
> m
, , 1

[,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

[,1] [,2]
[1,]    5    7
[2,]    6    8

> dimnames(m) = list(c('black','white'),c('up','down'),c('left','right'))
+
+
> dimnames(m) = list(c('black','white'),c('up','down'),c('left','right'))
> m
, , left

      up down
black  1    3
white  2    4

, , right

      up down
black  5    7
white  6    8

> m[1,2,]
left right
      3    7
> m[1,,]
      left right
up      1    5
down    3    7
```

C.3.3 Labels for lists and data frames

Labels for lists can be managed with the `name()` function:

```

> input.obj = list(init.values = c(0.2, 0.1, 10), tolerance = 0.001,
+                   maxIter = 100, memo.label = "July 3, 2013")
> names(input.obj)
[1] "init.values" "tolerance"    "maxIter"      "memo.label"
> names(input.obj)[2]
[1] "tolerance"
> names(input.obj)[2] = "maxTolerance"
> input.obj
$init.values
[1] 0.2 0.1 10.0

$maxTolerance
[1] 0.001

$maxIter
[1] 100

$memo.label
[1] "July 3, 2013"

```

Note that the name originally used for the second element of the preceding list has been changed from "tolerance" to "maxTolerance".

A data frame is a ‘matrix-like object’, and so the functions `rownames()`, `colnames()` and `dimnames()` can be used:

```

> patient.id = c(101,102,103)
> gender = c('M', 'M', 'F')
> bmi = c(103, NA, 131)
> bmi.data = data.frame(patient.id, gender, bmi)
>
> rownames(bmi.data)
[1] "1" "2" "3"
> colnames(bmi.data)
[1] "patient.id" "gender"      "bmi"
> dimnames(bmi.data)
[[1]]
[1] "1" "2" "3"

[[2]]
[1] "patient.id" "gender"      "bmi"

> rownames(bmi.data) = paste('subject', 1:3)
> bmi.data
      patient.id gender bmi
subject 1        101     M 103
subject 2        102     M  NA

```

```
subject 3      103      F 131
>
```

A data frame is also a list, so `names()` may be used. It has the same effect as `colnames()`. The function `row.names()` is intended for data frames, but has for most purposes the same functionality as `rownames()`.

```
> names(bmi.data)
[1] "patient.id" "gender"      "bmi"
> row.names(bmi.data)
[1] "subject 1" "subject 2" "subject 3"
```

C.4 Programming and Functions

R contains high-level programming capabilities. Commands entered on the command line can be included as lines in a program. The program can be invoked from an interactive editor (this is OS dependent) or run from a file using the `source()` function.

C.4.1 Program control

Program control relies on the reserved words `for`, `if`, `else`, `while`, `repeat`, `break`, `next`.

The syntax of `for` loops is somewhat different for R, consisting of a sequential assignment of elements of a vector to a variable. Formally it is given by

$$\text{for } (\text{var} \text{ in } \text{vector}) \text{ expr}$$

for iterative evaluation of a single command, or

$$\text{for } (\text{var} \text{ in } \text{vector}) \{ \text{code block} \}$$

for iterative evaluation of a block of code. Reserved words `break` or `next` can be used to terminate the loop, or to proceed immediately to the next iteration, respectively. A single expression example follows:

```
> x = c('A','B','C')
> y = rep(NA,3)
> for (i in 1:3) y[i] = paste(x[i],i,sep=' ')
> y
[1] "A1" "B2" "C3"
```

The iterated vector can be any type:

```
> x = c('A','B','C')
> y = NULL
> for (i in x) y = c(y,paste('[',i,']',sep=' '))
> y
[1] "[A]" "[B]" "[C]"
```

The following example iterates a block of code enclosed in braces { }:

```
> x = 0
> for (i in 1:10) {
+   y = i^2+1
+   x = x + y
+ }
> x
[1] 395
> sum( (1:10)^2 + 1)
[1] 395
```

The loop is intended to evaluate $\sum_{i=1}^{10} (i^2 + 1)$, and succeeds in doing so.

The **if** reserved word allows condition executions of expressions or code blocks. The parenthesis contains a conditional expression. If true, the expression or block is evaluated.

```
> x = 3
> a1 = 0
> a2 = 0
> if (x == 3) {
+   a1 = 1
+   a2 = 1
+ }
> a1
[1] 1
> a2
[1] 1
```

The **else** reserved word specifies an expression or code block to evaluate if the condition is false

```
> x = 3
> a1 = 0
> a2 = 0
> if (x > 3) {
+   a1 = 1
+   a2 = 1
+ } else
+ {
+   a1 = 99
+   a2 = 99
+ }
> a1
[1] 99
> a2
[1] 99
```

C.4.2 User defined functions

User defined functions are easy to make in R. The expression `function(...){...}` is assigned to an object, which becomes a *function object*. The parentheses includes all arguments, and the brackets include the block of code. The `return()` function is used to define the output:

```
> mean.and.variance = function(x) {
+   n = length(x)
+   meanx = sum(x)/n
+   varx = (sum(x^2) - (sum(x)^2)/n)/(n-1)
+   ans = c(meanx, varx)
+   names(ans) = c("mean", "variance")
+   return(ans)
+ }
>
> x = c(3,4,2,3,4,5,1,5)
> mean.and.variance(x)
  mean variance
3.375000 1.982143
> y = mean.and.variance(x)
> y
  mean variance
3.375000 1.982143
>
> mean(x)
[1] 3.375
> var(x)
[1] 1.982143
```

Needless to say, R already has functions which calculate means and variances.

One of the previous examples can be redesigned as a function:

```
> strange.sum = function(n) {
+   x = 0
+   for (i in 1:n) {
+     y = i^2+1
+     x = x + y
+   }
+   return(x)
+ }
> strange.sum(10)
[1] 395
```

Functions consisting of only a single expression do not require `return()` commands or braces (although braces can be used)

```
> mySum = function(x,y) {x + 2*y}
> mySum(2,4)
[1] 10
> mySum = function(x,y) x + 2*y
> mySum(2,4)
[1] 10
```

There is, of course, much more to this topic. Consult the R manual, or use `help('function')` (`function` is considered a reserved word).

C.4.3 Functions and environments

An *environment* is a *frame*, or collection of named R objects, and a pointer to an *enclosing environment*. It defines which R objects can be recognized for referencing. If an object is referenced, the current environment is searched. If it is not found, the enclosing environment is searched, and so on. The *global environment* `.GlobalEnv` (the user's workspace) is the first item on the search path, and is the current environment when R is started.

When a function is invoked, a new environment is created, and the calling environment becomes the enclosing environment. The function `environment()` displays the current environment.

```
> f = function(x) {environment()}
> environment()
<environment: R_GlobalEnv>
> f(0)
<environment: 0x109871e30>
```

An object created within a function is not the same object as an object in the enclosing environment.

```
> y = 5
> f = function(x) {
+   y = 99
+   return(y)
+ }
> f(0)
[1] 99
```

However, if an object is referenced but not found within an environment created by a function, it will search the enclosing environment.

```
> y = 5
> f = function(x) {
+   return(y)
+ }
> f(0)
[1] 5
> y = 6
> f(0)
[1] 6
```

Being aware of the *scope* of an object (in which environment an object is recognized) can be important when using a repository such as **bioconductor.org**. See <http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-scope.pdf> for more detail.

C.4.4 User defined binary operators

A function can be expressed as a *binary operator* using the assignment

```
> "%myBinaryOperator%" = function(x, y) { ... }
> x %myBinaryOperator% y
```

Binary operations can then be placed directly into algebraic expressions:

```
> '%max%' = function(x,y) {max(x,y)}
> 5 %max% 4
[1] 5
```

C.5 Vectorized Calculations

It is generally recommended in R that loops be avoided, and vectorized calculations used instead. For example, to evaluate $\sum_{i=1}^{10}(x^2 + 1)$ we used a loop, but we also used a type of vectorized calculation to achieve the same effect:

```
> sum((1:10)^2 + 1)
[1] 395
```

Suppose we are given a numeric vector **x** and a second vector **y** of the same length which identifies groups:

```
> x = c(1,2,3,2,3,4)
> y = c(1,1,1,1,2,2)
```

We wish to calculate a mean for each group defined by **y**. We could write a general function to do this:

```
> group.means = function(x,y) {
+ 
+   group.names = unique(y)
+   n.group = length(group.names)
+   gmeans = rep(NA, n.group)
+   for (i in 1:n.group) gmeans[i] = mean(x[y==group.names[i]])
+   ans = list(group.names, gmeans)
+   names(ans) = c('group.names', 'group.means')
+   return(ans)
+ 
+ }
```

- >
- > **group.means(x,y)**

```
$group.names
[1] 1 2

$group.means
[1] 2.0 3.5
```

A much simpler method is to apply the function `tapply()`:

```
> tapply(x,y,mean)
 1   2
2.0 3.5
```

In this function, the first and second arguments are the data and the group labels (these can be of any type). The third argument is any function which can be applied to a vector of the same type as `x`. There are several types of similar vectorized calculation functions. For example, `apply()` will apply a specified function to either the rows or columns of a matrix. See also `lapply()`, `sapply()`, `mapply()` and so on.

C.6 File Input and Output

There are a number of methods for reading and writing data. R objects can be conveniently saved and restored using the `save()` and `load()` commands (see `help`). Tabular data is typically input using the `read.table()` command. A command such as

```
> new.data = read.table("myData.txt")
```

is usually intended to store a data frame in the variable `new.data` using data from the file `myData.txt`. There are many options, so documentation should be consulted. A data frame is saved in a file using the `write.table()` command.

C.7 Packages

All R functions are part of `packages`. The R distribution comes with the `base` package of standard R functions. All other packages are add-on packages and become a permanent part of the local installation (a number of add-on packages are already installed in the R distribution). An installed add-on package must be loaded into the R environment to be used during a session (even if it has already been installed). It is possible to define default packages, which can be listed with the following command:

```
> options("defaultPackages")
$defaultPackages
[1] "datasets"    "utils"        "grDevices"    "graphics"
      "stats"       "methods"
```

This can be changed, but to do so will require a more in depth understanding of the R startup procedure. See *An Introduction to R*.

To see packages which are installed use the command:

```
> library()
```

to obtain a display such as

```
Packages in library '/Library/Frameworks/R.framework/Versions
/3.0/Resources/library':
```

AnnotationDbi	Annotation Database Interface
base	The R Base Package
Biobase	Biobase: Base functions for Bioconductor
BiocGenerics	Generic functions for Bioconductor
BiocInstaller	Install/Update Bioconductor and CRAN Packages
bnlearn	Bayesian network structure learning, parameter learning and inference
boot	Bootstrap Functions (originally by Angelo Canty for S)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
.	
.	
.	

To see packages which are loaded use the command

```
> search()
[1] ".GlobalEnv"      "package:linprog"    "package:lpSolve"
     "package:Matrix"    "tools:RGUI"       "package:stats"
     "package:graphics"
[8] "package:grDevices" "package:utils"     "package:datasets"
     "package:methods"   "Autoloads"        "package:base"
```

A default package is loaded automatically during the R startup. Otherwise a package may be loaded from the command line using the `library()` function. Sometimes a function will make use of another function belonging to a package which is generally not a default package. In this case a `require()` function may be included which will load the required function if it is installed, or otherwise issue a warning. Because R relies heavily on contributed packages, this feature is commonly encountered.

It is worth looking at the `help()` command in some more detail. A string argument not enclosed with quotes is assumed to be the name of an object (usually a function), while a string enclosed in quotes is a reserved word, such as `if`, `for` or `TRUE`. For example, the following commands:

```
> help(solve)
> help("for")
```

will provide detailed documentation on the function `solve()` and on the reserved word `for`, used in constructing `for` loops. If we want documentation on a specific package, we can use the option `package`:

```
> help(package = stats)
```

A list of packages available for installation is given using the command

```
> install.packages()
```

If the package to be installed is known, it becomes the argument to the function, enclosed in quotes:

```
> install.packages("bnlearn")
```

There are other ways of finding and installing packages, so see the documentation for more detail.

To take an example, suppose we are interesting in linear programming. This is not a standard R function, but is available in an R repository (as are many numerical algorithms in fields other than statistics). If we don't know where to find an appropriate library we can use a command suitable for vague searches of documentation:

```
> help.search("linear programming")
```

This will yield documentation on packages including "linear programming" in specific features of the documentation (this can be specified by the user; see `help(help.search)` for more detail). In this case the packages `boot`, `linprog` and `lpSolve` are listed. We may then install, say, `linprog` using the command:

```
> install.packages("linprog")
trying URL 'http://lib.stat.cmu.edu/R/CRAN/bin/macosx/contrib
 /3.0/linprog_0.9-2.tgz'
Content type 'application/x-gzip' length 33655 bytes (32 Kb)
opened URL
=====
downloaded 32 Kb
```

```
The downloaded binary packages are in
/var/folders/vs/v6dm307j09jfynfmgrpfplr0000gn/T
    //RtmpWJwRiz/downloaded_packages
>
```

We can obtain documentation for the package, then load it, using command:

```
> library(help = linprog)
> library(linprog)
```

Because R has extensive repositories of contributed packages there will often be several alternatives, so it is usually advisable to do some comparisons.

Note that '?' and '??' are shortcuts for `help()` and `help.search()`. For example:

```
> ?quantile
> ???'linear programming'
> ??eigenvalues
```

Packages are sometimes updated. The function `update.packages()` will compare the local library with the appropriate repositories to determine packages for which updates are available. By default, the user is asked if the library should be updated. Many options are available, so consult `help(update.packages)`:

```
> update.packages()
cluster :
  Version 1.14.4 installed in /Library/Frameworks/R.framework
/Versions/3.0/Resources/library
  Version 1.15.1 available at http://lib.stat.cmu.edu/R/CRAN
Update (y/N/c)? n
lattice :
  Version 0.20-24 installed in /Library/Frameworks/R.framework
/Versions/3.0/Resources/library
  Version 0.20-27 available at http://lib.stat.cmu.edu/R/CRAN
Update (y/N/c)? n
lme4 :
  Version 1.0-6 installed in /Library/Frameworks/R.framework
/Versions/3.0/Resources/library
  Version 1.1-5 available at http://lib.stat.cmu.edu/R/CRAN
Update (y/N/c)? c
cancelled by user
>
```

C.8 Objects and Classes in R

We've so far used the term *object* to informally describe different forms of data structure in R (vectors, lists and so on). We note that R is a type of *object oriented* programming language, meaning that R objects possess specific forms of data structure, and are associated with various *methods*, or procedures which act on some or all of the the objects' data.

C.8.1 Object modes

An object can be characterized by their *mode*. An object is of *atomic* structure if all data elements are of the same type. The mode (identified by function `mode()`) is then that type of data:

```
> mode(c(1,2,3))
[1] "numeric"
> mode(c(T,F,F))
[1] "logical"
> mode(c('a','b','c'))
[1] "character"
```

A list is of mode *list*:

```
> list('a',c(2,3,2),TRUE)
[[1]]
```

```
[1] "a"
[[2]]
[1] 2 3 2
[[3]]
[1] TRUE

> mode(list('a',c(2,3,2),TRUE))
[1] "list"
```

and a function is of mode `function`

```
> mode(sum)
[1] "function"
```

C.8.2 Object classes

It is important to understand that there are several generations of object types in R. What they have in common is that their properties are defined by *attributes*, which are represented by a list, which can be accessed by the `attributes()` function:

```
> m = array(1:12,12)
> dimnames(m) = list(paste('c',1:12,sep=''))
> m
   c1  c2  c3  c4  c5  c6  c7  c8  c9  c10  c11  c12
  1   2   3   4   5   6   7   8   9   10  11  12
> attributes(m)
$dim
[1] 12

$dimnames
$dimnames[[1]]
[1] "c1"  "c2"  "c3"  "c4"  "c5"  "c6"  "c7"  "c8"
         "c9"  "c10" "c11" "c12"
```

Of most concern here are the S3 and S4 objects. The notion of R objects most familiar today are the S3 objects introduced around 1988. An S3 object has a *class attribute*, consisting (at least) of a character string which identifies a *class*. In effect, an object is an *instance* of a class (in the sense that `x = 2.55` is an instance of a real number). The object of the preceding example is of class `array`. Objects need only one class, but it may have several, in which case the class attribute would be a vector of class identifiers. When multiple classes are present, the order they are listed in the vector is important. The object can be considered a type of the first class listed, but it *inherits* from the subsequent classes in the order given. The `class()` function identifies the class of an object:

```
> class(m)
[1] "array"
```

C.8.3 Generic functions

Identifying classes permits the use of *generic functions*. These are functions which accept objects of varying classes, and whose functionality depends on that class. General purpose functions such as `print()`, `plot()` or `summary()` are typical generic functions. This is quite important in statistical modeling. For example, a fitted model may be a linear model, a logistic model or a Cox proportional hazards model, but each of these types share a common output and summary structure (ANOVA table, residual plot, and so on). In addition, model selection procedures (stepwise regression, cross validation) may be applied in much the same way to each. The functions which fit these models (in this case `lm()`, `glm()`, `coxph()`) typically output an object of a specific class (in this case `lm`, `glm`, `coxph`) but these may be input to one of several generic functions. Consider a simple example of a linear linear model, which regresses vector `y` onto vector `x`:

```
> x = c(1:10)
> y = c(2,4,3,4,5,6,1,2,3,5)
> fit = lm(y ~ x)
> class(fit)
[1] "lm"
> mode(fit)
[1] "list"
> fit
```

Call:
`lm(formula = y ~ x)`

Coefficients:
`(Intercept)` `x`
 3.26667 0.04242

the function `lm()` writes an `lm` class object into `fit`. To examine `fit` we can simply enter it into the command line, as though it were a variable. We get the formula used in the fit, as well as the regression coefficients. However, an `lm` object contains much more information than this. We can input `fit` into the generic function `summary()`:

```
> summary(fit)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.56364 -1.14394  0.08485  1.14394  2.47879 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.26667   1.14186   2.861   0.0211 *  
x           0.04242   0.18403   0.231   0.8235    
                               * indicates significance at the 0.05 level.
```

```
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.672 on 8 degrees of freedom
Multiple R-squared: 0.006599, Adjusted R-squared: -0.1176
F-statistic: 0.05315 on 1 and 8 DF, p-value: 0.8235
```

We now get a much more complete summary of the model fit. In fact, we can get more information. The object attributes can be displayed using either the `names()` or the `attributes()` function (the `attributes()` function gives the class):

```
> attributes(fit)
$names
[1] "coefficients"   "residuals"      "effects"       "rank"
                 "fitted.values" "assign"
[7] "qr"             "df.residual"    "xlevels"       "call"
                 "terms"          "model"

$class
[1] "lm"

> names(fit)
[1] "coefficients"   "residuals"      "effects"       "rank"
                 "fitted.values" "assign"
[7] "qr"             "df.residual"    "xlevels"       "call"
                 "terms"          "model"
```

For example, the model residuals can be accessed using the `residuals` attribute

```
fit$residuals
     1         2         3         4         5
       6         7         8         9        10
-1.3090909  0.6484848 -0.3939394  0.5636364  1.5212121
      2.4787879 -2.5636364 -1.6060606 -0.6484848  1.3090909
>
```

For more description of the attributes use `help(lm)`

There exists a non-generic function `summary.lm()` which produces the same summary for an `lm` class. This is the standard nomenclature. If `myFunction()` is a generic function, then `myFunction.myClass()` is equivalent to `myFunction()` applied to an object of class `myClass`. Suppose an object `obj` has multiple classes `class1`, `class2`, `class3`, listed in that order. If `obj` in input to a generic function `fun()`, R will first search for function `fun.class1()`, then, if this is not found, `fun.class2()`, and so on. A generic function may have a default `fun.default()` which is used if no function for any of the objects classes is found.

```
> #
> # simulate a logistic regression model
```

```
> #
> x = c(1:10)
> y = rbinom(10, size = 1, prob = 0.5)
> fit.glm = glm(y ~ x, family = 'binomial')
> class(fit.glm)
[1] "glm" "lm"
```

The function `methods()` can be used either to display all generic functions available to a class, or all classes which may be input to a generic function:

```
> methods(summary)
[1] summary.aareg*           summary.aov          summary.aovlist
[4] summary.aspell*          summary.cch*        summary.connection
[7] summary.coxph*           summary.coxph.penal* summary.data.frame
[10] summary.Date             summary.default      summary.ecdf*
[13] summary.factor           summary.glm          summary.infl
[16] summary.lm               summary.loess*       summary.loglm*
[19] summary.manova           summary.matrix      summary.mlm
[22] summary.negbin*          summary.nls*        summary.packageStatus*
[25] summary.PDF_Dictionary* summary.PDF_Stream* summary.polr*
[28] summary.POSIXct          summary.POSIXlt      summary.ppr*
[31] summary.prcomp*          summary.princomp*   summary.proc_time
[34] summary.pyears*          summary.ratetable*  summary.rlm*
[37] summary.srcfile           summary.srcref       summary.stepfun
[40] summary.stl*              summary.survexp*     summary.survfit*
[43] summary.survfitms*       summary.survreg*     summary.table
[46] summary.tukeysmooth*     summary.XMLInternalDocument* summary.XMLInternalDocument*
```

Non-visible functions are asterisked

```
> methods(class='lm')
[1] add1.lm*           alias.lm*          anova.lm          case.names.lm*    confint.lm*
[6] cooks.distance.lm* deviance.lm*       dfbeta.lm*        dfbetas.lm*      drop1.lm*
[11] dummy.coef.lm*    effects.lm*        extractAIC.lm*   family.lm*       formula.lm*
[16] hatvalues.lm      influence.lm*     kappa.lm          labels.lm*       logLik.lm*
[21] model.frame.lm    model.matrix.lm  nobs.lm*         plot.lm          predict.lm
[26] print.lm          proj.lm*          qr.lm*          residuals.lm    rstandard.lm
[31] rstudent.lm       simulate.lm*     summary.lm        variable.names.lm* vcov.lm*
```

Non-visible functions are asterisked

A non-visible function produces output, but this will not be displayed (it can be copied into an object).

C.8.4 User defined methods

A simple example of the construction of a method for a class based on an existing generic function follows:

```
> x = 2
> class(x) = "myClass"
> class(x)
[1] "myClass"
```

```

>
> # For some reason, we would like objects of this class
> # to be displayed 7 times.
>
> print.myClass = function(x) {rep(x,7)}
> print(x)
[1] 2 2 2 2 2 2 2
> y = unclass(x)
> print(y)
[1] 2
> methods(class='myClass')
[1] print.myClass

```

New generic functions can be constructed using the `UseMethod()` function.

C.8.5 S4 (formal) classes

One problem with S3 classes is the rather informal nature of the identification of a class, which only requires setting a class attribute. In other words, classes are never really formally defined, which creates a problem when the same class of object is to be used by extensive collections of contributed packages. To address this problem S4 classes were introduced later in the 1990s. These require formal class definitions, which standardizes the attributes of an objects. This allows contributed applications to accept well defined standardized input. For this reason S4 classes are also referred to as *formal classes*.

The `setClass()` function is used to define a class (S3 classes have no corresponding functions):

```

> xyClass = setClass("xyClass",   slots = c(x="numeric", y="numeric"))
>
> xy.obj1 = xyClass(x = c(1,2,3,4), y = c(8,7,6,5))
> xy.obj2 = new("xyClass", x = c(1,2,3,4), y = c(8,7,6,5))
>
> class(xy.obj1)
[1] "xyClass"
attr(package)
[1] ".GlobalEnv"
> class(xy.obj2)
[1] "xyClass"
attr(package)
[1] ".GlobalEnv"

```

The command `new()` can be used to create an instance of an S4 class. Also, because the identifier `xyClass` was used in the assignment `xyClass = setClass("xyClass", slots = c(x="numeric", y="numeric"))` a function `xyClass()` is created which can be used to create a class instance as shown above (the name need not be the same as the class).

The function `setMethod()` is used to create formal (S4) methods, however, the procedure for S3 classes may also be used:

```
> print.xyClass = function(obj) {rep(c(obj@x,obj@y),3)}
```

See <http://cran.r-project.org/doc/contrib/Genolini-S4tutorialV0-5en.pdf> or <http://www.bioconductor.org/help/course-materials/2013/CSAMA2013/friday/afternoon/S4-tutorial.pdf> for more detail.

C.8.6 Testing and coercion of object types

It is important in an object oriented environment to be able to determine the type of an object, and to change it if necessary (this is generally referred to as coercion). These types of functions generally have names resembling `is.thing` or `as.thing`. For example:

```
> is.vector(c(2,3,3,4))
[1] TRUE
> is.matrix(c(2,3,3,4))
[1] FALSE
> as.matrix(c(2,3,3,4))
     [,1]
[1,]    2
[2,]    3
[3,]    3
[4,]    4
> is.na(NA)
[1] TRUE
> is.na(3)
[1] FALSE
> is.null(34)
[1] FALSE
> is.null(NULL)
[1] TRUE
> #
> # Is it an S4 class object?
> #
> xyClass = setClass("xyClass", slots = c(x="numeric", y="numeric"))
> xy.obj1 = xyClass(x = c(1,2,3,4), y = c(8,7,6,5))
> isS4(xy.obj1)
[1] TRUE
> isS4(c(2,3,3,4))
[1] FALSE
```

C.9 Random Variables in R

R contains functions associated with a wide class of random variables. There exists a standard naming convention. For example, the normal distribution has R name `norm`, and the binomial distribution has R name `binom`. For each such name there are four functions with prefix ‘d’ for

density, ‘p’ for CDF, ‘q’ for quantile and ‘r’ for simulated random variate. As needed, each R name has additional arguments needed to completely specify the distribution. For the binomial those are `size` and `prob` corresponding to `n,p` in the convention $X \sim \text{bin}(n, p)$. For example, for $X \sim \text{bin}(10, 0.25)$ the commands

```
> pbinom(3, size=10, prob=0.25)
[1] 0.7758751
> dbinom(3, size=10, prob=0.25)
[1] 0.2502823
> qbinom(0.5, size=10, prob=0.25)
[1] 2
>
```

tell us that $P(X \leq 3) = 0.7758751$, $P(X = 3) = 0.2502823$ and that 2 is (approximately) the median. We can simulate a random sample of size `n` in the following way:

```
> n = 5
> x = rbinom(n, size=10, prob=0.25)
> x
[1] 2 4 0 2 5
```

The following list is part of the R documentation.

- For the beta distribution see `dbeta`.
- For the binomial (including Bernoulli) distribution see `dbinom`.
- For the Cauchy distribution see `dcauchy`.
- For the chi-squared distribution see `dchisq`.
- For the exponential distribution see `dexp`.
- For the F distribution see `df`.
- For the gamma distribution see `dgamma`.
- For the geometric distribution see `dgeom`. (This is also a special case of the negative binomial.)
- For the hypergeometric distribution see `dhyper`.
- For the log-normal distribution see `dlnorm`.
- For the multinomial distribution see `dmultinom`.
- For the negative binomial distribution see `dnbinom`.
- For the normal distribution see `dnorm`.
- For the Poisson distribution see `dpois`.
- For the Student’s t distribution see `dt`.
- For the uniform distribution see `dunif`.
- For the Weibull distribution see `dweibull`.

For less common distributions of test statistics see `pbirthday`, `dsignrank`, `ptukey` and `dwilcox`.

See Also

RNG about random number generation in R.

Appendix D

Distribution Tables

Table 1: Standard Normal Curve Areas I. The table entry is the probability that a standard normal random variable is less than or equal to z .

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-3.4	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0002
-3.3	0.0005	0.0005	0.0005	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0003
-3.2	0.0007	0.0007	0.0006	0.0006	0.0006	0.0006	0.0006	0.0005	0.0005	0.0005
-3.1	0.001	0.0009	0.0009	0.0009	0.0008	0.0008	0.0008	0.0008	0.0007	0.0007
-3	0.0013	0.0013	0.0013	0.0012	0.0012	0.0011	0.0011	0.0011	0.001	0.001
-2.9	0.0019	0.0018	0.0018	0.0017	0.0016	0.0016	0.0015	0.0015	0.0014	0.0014
-2.8	0.0026	0.0025	0.0024	0.0023	0.0023	0.0022	0.0021	0.0021	0.002	0.0019
-2.7	0.0035	0.0034	0.0033	0.0032	0.0031	0.003	0.0029	0.0028	0.0027	0.0026
-2.6	0.0047	0.0045	0.0044	0.0043	0.0041	0.004	0.0039	0.0038	0.0037	0.0036
-2.5	0.0062	0.006	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.008	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.011
-2.1	0.0179	0.0174	0.017	0.0166	0.0162	0.0158	0.0154	0.015	0.0146	0.0143
-2	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.025	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.063	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.102	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.123	0.121	0.119	0.117
-1	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.166	0.1635	0.1611
-0.8	0.2119	0.209	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.242	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.305	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.281	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.33	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.352	0.3483
-0.2	0.4207	0.4168	0.4129	0.409	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247
0	0.5	0.496	0.492	0.488	0.484	0.4801	0.4761	0.4721	0.4681	0.4641

Table 2: **Standard Normal Curve Areas II.** The table entry is the probability that a standard normal random variable is less than or equal to z .

Table 3: Critical values for t -distribution. Table entry is the α critical value $t_{df,\alpha}$ for a t -distribution with df degrees of freedom.

df	α								
	0.1	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005	0.00025
1	3.078	6.314	12.706	31.821	63.657	127.321	318.309	636.619	1273.239
2	1.886	2.92	4.303	6.965	9.925	14.089	22.327	31.599	44.705
3	1.638	2.353	3.182	4.541	5.841	7.453	10.215	12.924	16.326
4	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.61	10.306
5	1.476	2.015	2.571	3.365	4.032	4.773	5.893	6.869	7.976
6	1.44	1.943	2.447	3.143	3.707	4.317	5.208	5.959	6.788
7	1.415	1.895	2.365	2.998	3.499	4.029	4.785	5.408	6.082
8	1.397	1.86	2.306	2.896	3.355	3.833	4.501	5.041	5.617
9	1.383	1.833	2.262	2.821	3.25	3.69	4.297	4.781	5.291
10	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587	5.049
11	1.363	1.796	2.201	2.718	3.106	3.497	4.025	4.437	4.863
12	1.356	1.782	2.179	2.681	3.055	3.428	3.93	4.318	4.716
13	1.35	1.771	2.16	2.65	3.012	3.372	3.852	4.221	4.597
14	1.345	1.761	2.145	2.624	2.977	3.326	3.787	4.14	4.499
15	1.341	1.753	2.131	2.602	2.947	3.286	3.733	4.073	4.417
16	1.337	1.746	2.12	2.583	2.921	3.252	3.686	4.015	4.346
17	1.333	1.74	2.11	2.567	2.898	3.222	3.646	3.965	4.286
18	1.33	1.734	2.101	2.552	2.878	3.197	3.61	3.922	4.233
19	1.328	1.729	2.093	2.539	2.861	3.174	3.579	3.883	4.187
20	1.325	1.725	2.086	2.528	2.845	3.153	3.552	3.85	4.146
21	1.323	1.721	2.08	2.518	2.831	3.135	3.527	3.819	4.11
22	1.321	1.717	2.074	2.508	2.819	3.119	3.505	3.792	4.077
23	1.319	1.714	2.069	2.5	2.807	3.104	3.485	3.768	4.047
24	1.318	1.711	2.064	2.492	2.797	3.091	3.467	3.745	4.021
25	1.316	1.708	2.06	2.485	2.787	3.078	3.45	3.725	3.996
26	1.315	1.706	2.056	2.479	2.779	3.067	3.435	3.707	3.974
27	1.314	1.703	2.052	2.473	2.771	3.057	3.421	3.69	3.954
28	1.313	1.701	2.048	2.467	2.763	3.047	3.408	3.674	3.935
29	1.311	1.699	2.045	2.462	2.756	3.038	3.396	3.659	3.918
30	1.31	1.697	2.042	2.457	2.75	3.03	3.385	3.646	3.902
40	1.303	1.684	2.021	2.423	2.704	2.971	3.307	3.551	3.788
50	1.299	1.676	2.009	2.403	2.678	2.937	3.261	3.496	3.723
60	1.296	1.671	2	2.39	2.66	2.915	3.232	3.46	3.681
70	1.294	1.667	1.994	2.381	2.648	2.899	3.211	3.435	3.651
80	1.292	1.664	1.99	2.374	2.639	2.887	3.195	3.416	3.629
90	1.291	1.662	1.987	2.368	2.632	2.878	3.183	3.402	3.612
100	1.29	1.66	1.984	2.364	2.626	2.871	3.174	3.39	3.598
110	1.289	1.659	1.982	2.361	2.621	2.865	3.166	3.381	3.587
120	1.289	1.658	1.98	2.358	2.617	2.86	3.16	3.373	3.578
Inf	1.282	1.645	1.96	2.326	2.576	2.807	3.09	3.291	3.481

Table 4: **Critical values for χ^2 -distribution.** Table entry is the α critical value $\chi_{df,\alpha}^2$ for a χ^2 -distribution with df degrees of freedom.

df	0.995	0.99	0.975	0.95	0.9	0.1	0.05	0.025	0.01	0.005
1	0	0	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.01	0.02	0.051	0.103	0.211	4.605	5.991	7.378	9.21	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.86
5	0.412	0.554	0.831	1.145	1.61	9.236	11.07	12.833	15.086	16.75
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.69	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.18	2.733	3.49	13.362	15.507	17.535	20.09	21.955
9	1.735	2.088	2.7	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.94	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.92	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.3
13	3.565	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688	29.819
14	4.075	4.66	5.629	6.571	7.79	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.39	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.26	9.591	10.851	12.443	28.412	31.41	34.17	37.566	39.997
21	8.034	8.897	10.283	11.591	13.24	29.615	32.671	35.479	38.932	41.401
22	8.643	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289	42.796
23	9.26	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638	44.181
24	9.886	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.98	45.559
25	10.52	11.524	13.12	14.611	16.473	34.382	37.652	40.646	44.314	46.928
26	11.16	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642	48.29
27	11.808	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963	49.645
28	12.461	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278	50.993
29	13.121	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588	52.336
30	13.787	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892	53.672
31	14.458	15.655	17.539	19.281	21.434	41.422	44.985	48.232	52.191	55.003
32	15.134	16.362	18.291	20.072	22.271	42.585	46.194	49.48	53.486	56.328
33	15.815	17.074	19.047	20.867	23.11	43.745	47.4	50.725	54.776	57.648
34	16.501	17.789	19.806	21.664	23.952	44.903	48.602	51.966	56.061	58.964
35	17.192	18.509	20.569	22.465	24.797	46.059	49.802	53.203	57.342	60.275
36	17.887	19.233	21.336	23.269	25.643	47.212	50.998	54.437	58.619	61.581
37	18.586	19.96	22.106	24.075	26.492	48.363	52.192	55.668	59.893	62.883
38	19.289	20.691	22.878	24.884	27.343	49.513	53.384	56.896	61.162	64.181
39	19.996	21.426	23.654	25.695	28.196	50.66	54.572	58.12	62.428	65.476
40	20.707	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691	66.766
41	21.421	22.906	25.215	27.326	29.907	52.949	56.942	60.561	64.95	68.053
42	22.138	23.65	25.999	28.144	30.765	54.09	58.124	61.777	66.206	69.336
43	22.859	24.398	26.785	28.965	31.625	55.23	59.304	62.99	67.459	70.616
44	23.584	25.148	27.575	29.787	32.487	56.369	60.481	64.201	68.71	71.893
45	24.311	25.901	28.366	30.612	33.35	57.505	61.656	65.41	69.957	73.166
46	25.041	26.657	29.16	31.439	34.215	58.641	62.83	66.617	71.201	74.437
47	25.775	27.416	29.956	32.268	35.081	59.774	64.001	67.821	72.443	75.704
48	26.511	28.177	30.755	33.098	35.949	60.907	65.171	69.023	73.683	76.969
49	27.249	28.941	31.555	33.93	36.818	62.038	66.339	70.222	74.919	78.231
50	27.991	29.707	32.357	34.764	37.689	63.167	67.505	71.42	76.154	79.49
60	35.534	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379	91.952
70	43.275	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425	104.215
80	51.172	53.54	57.153	60.391	64.278	96.578	101.879	106.629	112.329	116.321
90	59.196	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116	128.299
100	67.328	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807	140.169
150	109.142	112.668	117.985	122.692	128.275	172.581	179.581	185.8	193.208	198.36
200	152.241	156.432	162.728	168.279	174.835	226.021	233.994	241.058	249.445	255.264
250	196.161	200.939	208.098	214.392	221.806	279.05	287.882	295.689	304.94	311.346
300	240.663	245.972	253.912	260.878	269.068	331.789	341.395	349.874	359.906	366.844
350	285.608	291.406	300.064	307.648	316.55	384.306	394.626	403.723	414.474	421.9
400	330.903	337.155	346.482	354.641	364.207	436.649	447.632	457.305	468.724	476.606
450	376.483	383.163	393.118	401.817	412.007	488.849	500.456	510.67	522.717	531.026
500	422.303	429.388	439.936	449.147	459.926	540.93	553.127	563.852	576.493	585.207

Table 5: **Critical values for F -distribution I.** Table entry is the α critical value $\chi_{df_1, df_2, \alpha}^2$ for an F -distribution with df_1 numerator degrees of freedom and df_2 denominator degrees of freedom.

df_2	α	df_1								
		1	2	3	4	5	6	7	8	9
1	0.1	39.86	49.5	53.59	55.83	57.24	58.2	58.91	59.44	59.86
1	0.05	161.45	199.5	215.71	224.58	230.16	233.99	236.77	238.88	240.54
1	0.025	647.79	799.5	864.16	899.58	921.85	937.11	948.22	956.66	963.28
1	0.01	4052.18	4999.5	5403.35	5624.58	5763.65	5858.99	5928.36	5981.07	6022.47
1	0.005	16210.72	19999.5	21614.74	22499.58	23055.8	23437.11	23714.57	23925.41	24091
2	0.1	8.53	9	9.16	9.24	9.29	9.33	9.35	9.37	9.38
2	0.05	18.51	19	19.16	19.25	19.3	19.33	19.35	19.37	19.38
2	0.025	38.51	39	39.17	39.25	39.3	39.33	39.36	39.37	39.39
2	0.01	98.5	99	99.17	99.25	99.3	99.33	99.36	99.37	99.39
2	0.005	198.5	199	199.17	199.25	199.3	199.33	199.36	199.37	199.39
3	0.1	5.54	5.46	5.39	5.34	5.31	5.28	5.27	5.25	5.24
3	0.05	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81
3	0.025	17.44	16.04	15.44	15.1	14.88	14.73	14.62	14.54	14.47
3	0.01	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49	27.35
3	0.005	55.55	49.8	47.47	46.19	45.39	44.84	44.43	44.13	43.88
4	0.1	4.54	4.32	4.19	4.11	4.05	4.01	3.98	3.95	3.94
4	0.05	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6
4	0.025	12.22	10.65	9.98	9.6	9.36	9.2	9.07	8.98	8.9
4	0.01	21.2	18	16.69	15.98	15.52	15.21	14.98	14.8	14.66
4	0.005	31.33	26.28	24.26	23.15	22.46	21.97	21.62	21.35	21.14
5	0.1	4.06	3.78	3.62	3.52	3.45	3.4	3.37	3.34	3.32
5	0.05	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77
5	0.025	10.01	8.43	7.76	7.39	7.15	6.98	6.85	6.76	6.68
5	0.01	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29	10.16
5	0.005	22.78	18.31	16.53	15.56	14.94	14.51	14.2	13.96	13.77
6	0.1	3.78	3.46	3.29	3.18	3.11	3.05	3.01	2.98	2.96
6	0.05	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.1
6	0.025	8.81	7.26	6.6	6.23	5.99	5.82	5.7	5.6	5.52
6	0.01	13.75	10.92	9.78	9.15	8.75	8.47	8.26	8.1	7.98
6	0.005	18.63	14.54	12.92	12.03	11.46	11.07	10.79	10.57	10.39
7	0.1	3.59	3.26	3.07	2.96	2.88	2.83	2.78	2.75	2.72
7	0.05	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68
7	0.025	8.07	6.54	5.89	5.52	5.29	5.12	4.99	4.9	4.82
7	0.01	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84	6.72
7	0.005	16.24	12.4	10.88	10.05	9.52	9.16	8.89	8.68	8.51
8	0.1	3.46	3.11	2.92	2.81	2.73	2.67	2.62	2.59	2.56
8	0.05	5.32	4.46	4.07	3.84	3.69	3.58	3.5	3.44	3.39
8	0.025	7.57	6.06	5.42	5.05	4.82	4.65	4.53	4.43	4.36
8	0.01	11.26	8.65	7.59	7.01	6.63	6.37	6.18	6.03	5.91
8	0.005	14.69	11.04	9.6	8.81	8.3	7.95	7.69	7.5	7.34
9	0.1	3.36	3.01	2.81	2.69	2.61	2.55	2.51	2.47	2.44
9	0.05	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18
9	0.025	7.21	5.71	5.08	4.72	4.48	4.32	4.2	4.1	4.03
9	0.01	10.56	8.02	6.99	6.42	6.06	5.8	5.61	5.47	5.35
9	0.005	13.61	10.11	8.72	7.96	7.47	7.13	6.88	6.69	6.54

Table 6: **Critical values for F -distribution II.** Table entry is the α critical value $\chi_{df_1, df_2, \alpha}^2$ for an F -distribution with df_1 numerator degrees of freedom and df_2 denominator degrees of freedom.

df_2	α	10	12	15	20	24	30	60	120	Inf
1	0.1	60.19	60.71	61.22	61.74	62	62.26	62.79	63.06	63.33
1	0.05	241.88	243.91	245.95	248.01	249.05	250.1	252.2	253.25	254.31
1	0.025	968.63	976.71	984.87	993.1	997.25	1001.41	1009.8	1014.02	1018.26
1	0.01	6055.85	6106.32	6157.28	6208.73	6234.63	6260.65	6313.03	6339.39	6365.86
1	0.005	24224.49	24426.37	24630.21	24835.97	24939.57	25043.63	25253.14	25358.57	25464.46
2	0.1	9.39	9.41	9.42	9.44	9.45	9.46	9.47	9.48	9.49
2	0.05	19.4	19.41	19.43	19.45	19.45	19.46	19.48	19.49	19.5
2	0.025	39.4	39.41	39.43	39.45	39.46	39.46	39.48	39.49	39.5
2	0.01	99.4	99.42	99.43	99.45	99.46	99.47	99.48	99.49	99.5
2	0.005	199.4	199.42	199.43	199.45	199.46	199.47	199.48	199.49	199.5
3	0.1	5.23	5.22	5.2	5.18	5.18	5.17	5.15	5.14	5.13
3	0.05	8.79	8.74	8.7	8.66	8.64	8.62	8.57	8.55	8.53
3	0.025	14.42	14.34	14.25	14.17	14.12	14.08	13.99	13.95	13.9
3	0.01	27.23	27.05	26.87	26.69	26.6	26.5	26.32	26.22	26.13
3	0.005	43.69	43.39	43.08	42.78	42.62	42.47	42.15	41.99	41.83
4	0.1	3.92	3.9	3.87	3.84	3.83	3.82	3.79	3.78	3.76
4	0.05	5.96	5.91	5.86	5.8	5.77	5.75	5.69	5.66	5.63
4	0.025	8.84	8.75	8.66	8.56	8.51	8.46	8.36	8.31	8.26
4	0.01	14.55	14.37	14.2	14.02	13.93	13.84	13.65	13.56	13.46
4	0.005	20.97	20.7	20.44	20.17	20.03	19.89	19.61	19.47	19.32
5	0.1	3.3	3.27	3.24	3.21	3.19	3.17	3.14	3.12	3.1
5	0.05	4.74	4.68	4.62	4.56	4.53	4.5	4.43	4.4	4.36
5	0.025	6.62	6.52	6.43	6.33	6.28	6.23	6.12	6.07	6.02
5	0.01	10.05	9.89	9.72	9.55	9.47	9.38	9.2	9.11	9.02
5	0.005	13.62	13.38	13.15	12.9	12.78	12.66	12.4	12.27	12.14
6	0.1	2.94	2.9	2.87	2.84	2.82	2.8	2.76	2.74	2.72
6	0.05	4.06	4	3.94	3.87	3.84	3.81	3.74	3.7	3.67
6	0.025	5.46	5.37	5.27	5.17	5.12	5.07	4.96	4.9	4.85
6	0.01	7.87	7.72	7.56	7.4	7.31	7.23	7.06	6.97	6.88
6	0.005	10.25	10.03	9.81	9.59	9.47	9.36	9.12	9	8.88
7	0.1	2.7	2.67	2.63	2.59	2.58	2.56	2.51	2.49	2.47
7	0.05	3.64	3.57	3.51	3.44	3.41	3.38	3.3	3.27	3.23
7	0.025	4.76	4.67	4.57	4.47	4.41	4.36	4.25	4.2	4.14
7	0.01	6.62	6.47	6.31	6.16	6.07	5.99	5.82	5.74	5.65
7	0.005	8.38	8.18	7.97	7.75	7.64	7.53	7.31	7.19	7.08
8	0.1	2.54	2.5	2.46	2.42	2.4	2.38	2.34	2.32	2.29
8	0.05	3.35	3.28	3.22	3.15	3.12	3.08	3.01	2.97	2.93
8	0.025	4.3	4.2	4.1	4	3.95	3.89	3.78	3.73	3.67
8	0.01	5.81	5.67	5.52	5.36	5.28	5.2	5.03	4.95	4.86
8	0.005	7.21	7.01	6.81	6.61	6.5	6.4	6.18	6.06	5.95
9	0.1	2.42	2.38	2.34	2.3	2.28	2.25	2.21	2.18	2.16
9	0.05	3.14	3.07	3.01	2.94	2.9	2.86	2.79	2.75	2.71
9	0.025	3.96	3.87	3.77	3.67	3.61	3.56	3.45	3.39	3.33
9	0.01	5.26	5.11	4.96	4.81	4.73	4.65	4.48	4.4	4.31
9	0.005	6.42	6.23	6.03	5.83	5.73	5.62	5.41	5.3	5.19

Table 7: **Critical values for F -distribution III.** Table entry is the α critical value $\chi^2_{df_1, df_2, \alpha}$ for an F -distribution with df_1 numerator degrees of freedom and df_2 denominator degrees of freedom.

df_2	α	1	2	3	4	5	6	7	8	9
10	0.1	3.29	2.92	2.73	2.61	2.52	2.46	2.41	2.38	2.35
10	0.05	4.96	4.1	3.71	3.48	3.33	3.22	3.14	3.07	3.02
10	0.025	6.94	5.46	4.83	4.47	4.24	4.07	3.95	3.85	3.78
10	0.01	10.04	7.56	6.55	5.99	5.64	5.39	5.2	5.06	4.94
10	0.005	12.83	9.43	8.08	7.34	6.87	6.54	6.3	6.12	5.97
12	0.1	3.18	2.81	2.61	2.48	2.39	2.33	2.28	2.24	2.21
12	0.05	4.75	3.89	3.49	3.26	3.11	3	2.91	2.85	2.8
12	0.025	6.55	5.1	4.47	4.12	3.89	3.73	3.61	3.51	3.44
12	0.01	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.5	4.39
12	0.005	11.75	8.51	7.23	6.52	6.07	5.76	5.52	5.35	5.2
15	0.1	3.07	2.7	2.49	2.36	2.27	2.21	2.16	2.12	2.09
15	0.05	4.54	3.68	3.29	3.06	2.9	2.79	2.71	2.64	2.59
15	0.025	6.2	4.77	4.15	3.8	3.58	3.41	3.29	3.2	3.12
15	0.01	8.68	6.36	5.42	4.89	4.56	4.32	4.14	4	3.89
15	0.005	10.8	7.7	6.48	5.8	5.37	5.07	4.85	4.67	4.54
20	0.1	2.97	2.59	2.38	2.25	2.16	2.09	2.04	2	1.96
20	0.05	4.35	3.49	3.1	2.87	2.71	2.6	2.51	2.45	2.39
20	0.025	5.87	4.46	3.86	3.51	3.29	3.13	3.01	2.91	2.84
20	0.01	8.1	5.85	4.94	4.43	4.1	3.87	3.7	3.56	3.46
20	0.005	9.94	6.99	5.82	5.17	4.76	4.47	4.26	4.09	3.96
24	0.1	2.93	2.54	2.33	2.19	2.1	2.04	1.98	1.94	1.91
24	0.05	4.26	3.4	3.01	2.78	2.62	2.51	2.42	2.36	2.3
24	0.025	5.72	4.32	3.72	3.38	3.15	2.99	2.87	2.78	2.7
24	0.01	7.82	5.61	4.72	4.22	3.9	3.67	3.5	3.36	3.26
24	0.005	9.55	6.66	5.52	4.89	4.49	4.2	3.99	3.83	3.69
30	0.1	2.88	2.49	2.28	2.14	2.05	1.98	1.93	1.88	1.85
30	0.05	4.17	3.32	2.92	2.69	2.53	2.42	2.33	2.27	2.21
30	0.025	5.57	4.18	3.59	3.25	3.03	2.87	2.75	2.65	2.57
30	0.01	7.56	5.39	4.51	4.02	3.7	3.47	3.3	3.17	3.07
30	0.005	9.18	6.35	5.24	4.62	4.23	3.95	3.74	3.58	3.45
60	0.1	2.79	2.39	2.18	2.04	1.95	1.87	1.82	1.77	1.74
60	0.05	4	3.15	2.76	2.53	2.37	2.25	2.17	2.1	2.04
60	0.025	5.29	3.93	3.34	3.01	2.79	2.63	2.51	2.41	2.33
60	0.01	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82	2.72
60	0.005	8.49	5.79	4.73	4.14	3.76	3.49	3.29	3.13	3.01
120	0.1	2.75	2.35	2.13	1.99	1.9	1.82	1.77	1.72	1.68
120	0.05	3.92	3.07	2.68	2.45	2.29	2.18	2.09	2.02	1.96
120	0.025	5.15	3.8	3.23	2.89	2.67	2.52	2.39	2.3	2.22
120	0.01	6.85	4.79	3.95	3.48	3.17	2.96	2.79	2.66	2.56
120	0.005	8.18	5.54	4.5	3.92	3.55	3.28	3.09	2.93	2.81
Inf	0.1	2.71	2.3	2.08	1.94	1.85	1.77	1.72	1.67	1.63
Inf	0.05	3.84	3	2.6	2.37	2.21	2.1	2.01	1.94	1.88
Inf	0.025	5.02	3.69	3.12	2.79	2.57	2.41	2.29	2.19	2.11
Inf	0.01	6.63	4.61	3.78	3.32	3.02	2.8	2.64	2.51	2.41
Inf	0.005	7.88	5.3	4.28	3.72	3.35	3.09	2.9	2.74	2.62

Table 8: **Critical values for F -distribution IV.** Table entry is the α critical value $\chi^2_{df_1, df_2, \alpha}$ for an F -distribution with df_1 numerator degrees of freedom and df_2 denominator degrees of freedom.

df_2	α	df_1								
		10	12	15	20	24	30	60	120	Inf
10	0.1	2.32	2.28	2.24	2.2	2.18	2.16	2.11	2.08	2.06
10	0.05	2.98	2.91	2.85	2.77	2.74	2.7	2.62	2.58	2.54
10	0.025	3.72	3.62	3.52	3.42	3.37	3.31	3.2	3.14	3.08
10	0.01	4.85	4.71	4.56	4.41	4.33	4.25	4.08	4	3.91
10	0.005	5.85	5.66	5.47	5.27	5.17	5.07	4.86	4.75	4.64
12	0.1	2.19	2.15	2.1	2.06	2.04	2.01	1.96	1.93	1.9
12	0.05	2.75	2.69	2.62	2.54	2.51	2.47	2.38	2.34	2.3
12	0.025	3.37	3.28	3.18	3.07	3.02	2.96	2.85	2.79	2.72
12	0.01	4.3	4.16	4.01	3.86	3.78	3.7	3.54	3.45	3.36
12	0.005	5.09	4.91	4.72	4.53	4.43	4.33	4.12	4.01	3.9
15	0.1	2.06	2.02	1.97	1.92	1.9	1.87	1.82	1.79	1.76
15	0.05	2.54	2.48	2.4	2.33	2.29	2.25	2.16	2.11	2.07
15	0.025	3.06	2.96	2.86	2.76	2.7	2.64	2.52	2.46	2.4
15	0.01	3.8	3.67	3.52	3.37	3.29	3.21	3.05	2.96	2.87
15	0.005	4.42	4.25	4.07	3.88	3.79	3.69	3.48	3.37	3.26
20	0.1	1.94	1.89	1.84	1.79	1.77	1.74	1.68	1.64	1.61
20	0.05	2.35	2.28	2.2	2.12	2.08	2.04	1.95	1.9	1.84
20	0.025	2.77	2.68	2.57	2.46	2.41	2.35	2.22	2.16	2.09
20	0.01	3.37	3.23	3.09	2.94	2.86	2.78	2.61	2.52	2.42
20	0.005	3.85	3.68	3.5	3.32	3.22	3.12	2.92	2.81	2.69
24	0.1	1.88	1.83	1.78	1.73	1.7	1.67	1.61	1.57	1.53
24	0.05	2.25	2.18	2.11	2.03	1.98	1.94	1.84	1.79	1.73
24	0.025	2.64	2.54	2.44	2.33	2.27	2.21	2.08	2.01	1.94
24	0.01	3.17	3.03	2.89	2.74	2.66	2.58	2.4	2.31	2.21
24	0.005	3.59	3.42	3.25	3.06	2.97	2.87	2.66	2.55	2.43
30	0.1	1.82	1.77	1.72	1.67	1.64	1.61	1.54	1.5	1.46
30	0.05	2.16	2.09	2.01	1.93	1.89	1.84	1.74	1.68	1.62
30	0.025	2.51	2.41	2.31	2.2	2.14	2.07	1.94	1.87	1.79
30	0.01	2.98	2.84	2.7	2.55	2.47	2.39	2.21	2.11	2.01
30	0.005	3.34	3.18	3.01	2.82	2.73	2.63	2.42	2.3	2.18
60	0.1	1.71	1.66	1.6	1.54	1.51	1.48	1.4	1.35	1.29
60	0.05	1.99	1.92	1.84	1.75	1.7	1.65	1.53	1.47	1.39
60	0.025	2.27	2.17	2.06	1.94	1.88	1.82	1.67	1.58	1.48
60	0.01	2.63	2.5	2.35	2.2	2.12	2.03	1.84	1.73	1.6
60	0.005	2.9	2.74	2.57	2.39	2.29	2.19	1.96	1.83	1.69
120	0.1	1.65	1.6	1.55	1.48	1.45	1.41	1.32	1.26	1.19
120	0.05	1.91	1.83	1.75	1.66	1.61	1.55	1.43	1.35	1.25
120	0.025	2.16	2.05	1.94	1.82	1.76	1.69	1.53	1.43	1.31
120	0.01	2.47	2.34	2.19	2.03	1.95	1.86	1.66	1.53	1.38
120	0.005	2.71	2.54	2.37	2.19	2.09	1.98	1.75	1.61	1.43
Inf	0.1	1.6	1.55	1.49	1.42	1.38	1.34	1.24	1.17	1
Inf	0.05	1.83	1.75	1.67	1.57	1.52	1.46	1.32	1.22	1
Inf	0.025	2.05	1.94	1.83	1.71	1.64	1.57	1.39	1.27	1
Inf	0.01	2.32	2.18	2.04	1.88	1.79	1.7	1.47	1.32	1
Inf	0.005	2.52	2.36	2.19	2	1.9	1.79	1.53	1.36	1

Table 9: Cumulative binomial probabilities I. Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 10: **Cumulative binomial probabilities II.** Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 11: Cumulative binomial probabilities III. Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 12: **Cumulative binomial probabilities IV.** Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 13: **Cumulative binomial probabilities V.** Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 14: Cumulative binomial probabilities VI. Table entry is $P(X \leq x)$ where $X \sim \text{bin}(n, p)$.

Table 15: Cumulative Poisson probabilities I. Table entry is $P(X \leq x)$ where $X \sim \text{pois}(\lambda)$.

Table 16: **Cumulative Poisson probabilities II.** Table entry is $P(X \leq x)$ where $X \sim \text{pois}(\lambda)$.

Bibliography

- Akaike, H. (1973a). Information theory and an extension of the maximum likelihood principle. In B.N.Petrov and F.Cz'aki, editors, *2nd International Symposium in Information Theory*. Akademiai Ki'ado, Budapest.
- Akaike, H. (1973b). Information theory and the maximum likelihood principle. In B.N.Petrov and F.Cz'aki, editors, *2nd International Symposium in Information Theory*.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**(6), 716–723.
- Albert, J. (2009). *Bayesian computation with R*. Springer Science & Business Media.
- Almudevar, A. (2006). Using artificial neural networks to predict claim duration in a work injury compensation environment. In *Proceedings 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 378–384. IEEE.
- Almudevar, A. (2013). Multiple hypothesis testing: a methodological overview. In *Statistical Methods for Microarray Data Analysis*, pages 37–55. Springer.
- Almudevar, A. (2016). An information theoretic approach to pedigree reconstruction. *Theoretical population biology*, **107**, 52–64.
- Ambroise, C. and McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, **99**(10), 6562–6566.
- Ash, R. B. and Dolacutuseans-Dade, C. A. (2000). *Real Analysis and Probability*. Academic Press, San Diego, second edition.
- Bickel, P. J. and Doksum, K. A. (2015a). *Mathematical Statistics: Basic Ideas and Selected Topics, Volume I*, volume 117. CRC Press.
- Bickel, P. J. and Doksum, K. A. (2015b). *Mathematical Statistics: Basic Ideas and Selected Topics, Volume II*, volume 117. CRC Press.
- Billingsley, P. (1995). *Probability and Measure*. John Wiley and Sons, New York, NY, third edition.
- Box, G. E., Hunter, W. G., and Hunter, J. S. (2018). *Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building*. John Wiley & Sons.

- Canty, A. and Ripley, B. (2017). Package ‘boot’. *Bootstrap functions. Ver*, pages 1–3.
- Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury, Pacific Grove, CA, 2nd edition.
- Chambers, J. (1977). *Computational Methods for Data Analysis*. John Wiley & Sons.
- Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.
- Cox, D. and Hinkley, D. (1979). *Theoretical Statistics*. Taylor & Francis.
- Cox, D. R. and Oakes, D. (1984). *Analysis of Survival Data*. Chapman & Hall.
- Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*, volume 1. Cambridge University Press.
- Deming, W. E. (1943). *Statistical Adjustment of Data*. Wiley.
- Devore, J. (2011). *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 8th edition.
- Dudoit, S. and van der Laan, M. J. (2008). *Multiple Testing Procedures with Applications to Genomics*. Springer, New York.
- Durrett, R. (2010). *Probability: Theory and Examples*. Cambridge University Press, New York, NY, fourth edition.
- Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, **7**(1), 1–26.
- Efron, B. and Hinkley, D. V. (1978). Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information. *Biometrika*, **65**(3), 457–483.
- Efron, B. and Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. CRC Press.
- Efron, B. E. and Gong, G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician*, **37**(1), 36–48.
- Feller, W. (1968). *Probability Theory and Its Applications, Volume 1*. John Wiley and Sons, New York, NY, third edition.
- Feller, W. (1971). *Probability Theory and Its Applications, Volume 2*. John Wiley and Sons, New York, NY, second edition.
- Fraley, C. and Raftery, A. (2002). Model-based clustering, discriminant analysis, and density estimation. *JASA*, **97**, 611–631.
- Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics, New York.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**(6), 721–741.

- Good, P. (2005). *Permutation, Parametric, and Bootstrap Tests of Hypotheses*. Springer, 3rd edition.
- Good, P. (2013). *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer Science & Business Media.
- Grünwald, P. D. (2007). *The Minimum Description Length Principle*. MIT Press.
- Hahne, F., Huber, W., Gentleman, R., and Falcon, S. (2010). *Bioconductor Case Studies*. Springer Science & Business Media.
- Hastie, T. and Tibshirani, R. J. (1990). *Generalized Additive Models*. CRC press.
- Hastie, T. J. (2017). *Statistical Models in S*. Routledge.
- Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, **57**(1), 97–109.
- Hogg, R. V., McKean, J. M., and Craig, A. T. (2018). *Introduction to Mathematical Statistics (What's New in Statistics)*. Pearson, 8th edition.
- Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press, Cambridge, UK.
- Ibragimov, M., Ibragimov, R., and Walden, J. (2015). *Heavy-tailed Distributions and Robustness in Economics and Finance*, volume 214. Springer.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.
- Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, **53**(282), 457–481.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**(4598), 671–680.
- Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- Lehmann, E. and Casella, G. (1998). *Theory of Point Estimation*. Springer, New York, NY, 2nd edition.
- Lehmann, E. L. and Romano, J. P. (2006). *Testing Statistical Hypotheses*. Springer Science & Business Media.
- Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000). Winbugs - a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, **10**(4), 325–337.
- Marin, J.-M. and Robert, C. P. (2014). *Bayesian Essentials with R*, volume 48. Springer.
- McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman & Hall.

- McCulloch, C., Searle, S., and Neuhaus, J. (2008). *Generalized, Linear and Mixed Models*. Wiley-Interscience.
- McCulloch, W. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**(4), 115–133.
- Meeker, W. Q. and Escobar, L. A. (2014). *Statistical Methods for Reliability Data*. John Wiley & Sons.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**(6), 1087–1092.
- Nelder, J. and Wedderburn, R. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, **135**(3), 370–384.
- Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W. (1996). *Applied Linear Statistical Models*. Irwin, Chicago, 4th edition.
- Nourani, Y. and Andresen, B. (1998). A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, **31**(41), 8373.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California.
- Picard, R. R. and Cook, R. D. (1984). Cross-validation of regression models. *Journal of the American Statistical Association*, **79**(387), 575–583.
- Pichichero, M., Morris, M., and Almudevar, A. (2018). Three innate cytokine biomarkers predict presence of acute otitis media and relevant otopathogens. *Biomarkers and Applications*, pages BMAP–118.
- Ramsay, J., Hooker, G., and Graves, S. (2009). *Functional Data Analysis with R and MATLAB*. Springer.
- Ripley, B. (1994). Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society Series B*, **56**, 409–456.
- Ripley, B. and Hjort, N. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.
- Ripley, R., Harris, A., and Tarassenko, L. (1998). Neural network models for breast cancer prognosis. *Neural Computing & Applications*, **7**, 367–375.
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, **14**(5), 465–471.
- Rissanen, J. (2007). *Information and Complexity in Statistical Modeling*. Springer Science & Business Media.
- Ross, S. (2014). *Introduction to Probability Models*. Elsevier Science.

- Ross, S. M. (1996). *Stochastic Processes*. John Wiley and Sons, New York, NY, 2nd edition.
- Royden, H. L. (1968). *Real Analysis*. MacMillan Publishing, New York, NY, 2nd edition.
- Schwarz, G. *et al.* (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**(2), 461–464.
- Scutari, M. (2010). Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, **35**(i03).
- Seber, G. and Wild, C. (1989). *Nonlinear Regression*. Wiley, New York.
- Seber, G. A. and Lee, A. J. (2012). *Linear Regression Analysis*, volume 329. John Wiley & Sons.
- Sen, A. and Srivastava, M. (2012). *Regression Analysis: Theory, Methods, and Applications*. Springer Science & Business Media.
- Shonkwiler, R. W. and Mendivil, F. (2009). *Explorations in Monte Carlo Methods*. Springer Science & Business Media.
- Spivak, M. (1967). *Calculus*. Publish or Perish, 4th edition.
- Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288.
- Tibshirani, R., Wainwright, M., and Hastie, T. (2015). *Statistical Learning with Sparsity: The LASSO and Generalizations*. Chapman and Hall/CRC.
- Van Belle, G., Fisher, L. D., Heagerty, P. J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences*, volume 519. John Wiley & Sons.
- van Laarhoven, P. and Aarts, E. (1987). *Simulated Annealing: Theory and Application*. Springer.
- Venables, W. N. and Ripley, B. D. (2013). *Modern Applied Statistics with S-PLUS*. Springer Science & Business Media.
- Welsh, A. H. (2011). *Aspects of Statistical Inference*. John Wiley & Sons.
- Whittle, P. (2000). *Probability via Expectation*. Springer-Verlag, New York, NY, 4th edition.