

Generalized Additive Models: an introduction with R

Simon N. Wood



Contents

Preface	xi
1 Linear Models	1
1.1 A simple linear model	2
Simple least squares estimation	3
1.1.1 Sampling properties of $\hat{\beta}$	3
1.1.2 So how old is the universe?	5
1.1.3 Adding a distributional assumption	7
Testing hypotheses about β	7
Confidence intervals	9
1.2 Linear models in general	10
1.3 The theory of linear models	12
1.3.1 Least squares estimation of β	12
1.3.2 The distribution of $\hat{\beta}$	13
1.3.3 $(\hat{\beta}_i - \beta_i)/\hat{\sigma}_{\hat{\beta}_i} \sim t_{n-p}$	14
1.3.4 F-ratio results	15
1.3.5 The influence matrix	16
1.3.6 The residuals, $\hat{\epsilon}$, and fitted values, $\hat{\mu}$	16
1.3.7 Results in terms of \mathbf{X}	17
1.3.8 The Gauss Markov Theorem: what's special about least squares?	17
1.4 The geometry of linear modelling	18
1.4.1 Least squares	19
1.4.2 Fitting by orthogonal decompositions	20

1.4.3	Comparison of nested models	21
1.5	Practical linear models	22
1.5.1	Model fitting and model checking	23
1.5.2	Model summary	28
1.5.3	Model selection	30
1.5.4	Another model selection example	31
A follow up		35
1.5.5	Confidence intervals	36
1.5.6	Prediction	36
1.6	Practical modelling with factors	37
1.6.1	Identifiability	38
1.6.2	Multiple factors	39
1.6.3	‘Interactions’ of factors	40
1.6.4	Using factor variables in R	41
1.7	General linear model specification in R	44
1.8	Further linear modelling theory	45
1.8.1	Constraints I: general linear constraints	46
1.8.2	Constraints II: ‘contrasts’ and factor variables	46
1.8.3	Likelihood	48
1.8.4	Non-independent data with variable variance	49
1.8.5	AIC and Mallow’s statistic,	51
1.8.6	Non-linear least squares	53
1.8.7	Further reading	55
1.9	Exercises	55
2	Generalized Linear Models	59
2.1	The theory of GLMs	60
2.1.1	The exponential family of distributions	62
2.1.2	Fitting Generalized Linear Models	63
2.1.3	The IRLS objective is a quadratic approximation to the log-likelihood	66

CONTENTS	v
2.1.4 AIC for GLMs	67
2.1.5 Large sample distribution of $\hat{\beta}$	68
2.1.6 Comparing models by hypothesis testing	68
Deviance	69
Model comparison with unknown ϕ	70
2.1.7 $\hat{\phi}$ and Pearson's statistic	70
2.1.8 Canonical link functions	71
2.1.9 Residuals	72
Pearson Residuals	72
Deviance Residuals	73
2.1.10 Quasi-likelihood	73
2.2 Geometry of GLMs	75
2.2.1 The geometry of IRLS	76
2.2.2 Geometry and IRLS convergence	79
2.3 GLMs with R	80
2.3.1 Binomial models and heart disease	80
2.3.2 A Poisson regression epidemic model	87
2.3.3 Log-linear models for categorical data	92
2.3.4 Sole eggs in the Bristol channel	96
2.4 Likelihood	101
2.4.1 Invariance	102
2.4.2 Properties of the expected log-likelihood	102
2.4.3 Consistency	105
2.4.4 Large sample distribution of $\hat{\theta}$	107
2.4.5 The generalized likelihood ratio test (GLRT)	107
2.4.6 Derivation of $2\lambda \sim \chi_r^2$ under H_0	108
2.4.7 AIC in general	110
2.4.8 Quasi-likelihood results	112
2.5 Exercises	114

3 Introducing GAMs	119
3.1 Introduction	119
3.2 Univariate smooth functions	120
3.2.1 Representing a smooth function: regression splines	120
A very simple example: a polynomial basis	120
Another example: a cubic spline basis	122
Using the cubic spline basis	124
3.2.2 Controlling the degree of smoothing with penalized regression splines	126
3.2.3 Choosing the smoothing parameter, λ : cross validation	129
3.3 Additive Models	131
3.3.1 Penalized regression spline representation of an additive model	132
3.3.2 Fitting additive models by penalized least squares	133
3.4 Generalized Additive Models	135
3.5 Summary	137
3.6 Exercises	138
4 Some GAM theory	143
4.1 Smoothing bases	144
4.1.1 Why splines?	144
Natural cubic splines are smoothest interpolators	144
Cubic smoothing splines	146
4.1.2 Cubic regression splines	147
4.1.3 A cyclic cubic regression spline	149
4.1.4 P-splines	150
4.1.5 Thin plate regression splines	152
Thin plate splines	152
Thin plate regression splines	155
Properties of thin plate regression splines	156
Knot based approximation	158
4.1.6 Shrinkage smoothers	158

CONTENTS	vii
4.1.7 Choosing the basis dimension	159
4.1.8 Tensor product smooths	160
Tensor product bases	160
Tensor product penalties	163
4.2 Setting up GAMs as penalized GLMs	165
4.2.1 Variable coefficient models	166
4.3 Justifying P-IRLS	167
4.4 Degrees of freedom and residual variance estimation	168
4.4.1 Residual variance or scale parameter estimation	169
4.5 Smoothing Parameter Estimation Criteria	170
4.5.1 Known scale parameter: UBRE	170
4.5.2 Unknown scale parameter: Cross Validation	171
Problems with Ordinary Cross Validation	172
4.5.3 Generalized Cross Validation	173
4.5.4 GCV/UBRE/AIC in the Generalized case	175
Approaches to GAM GCV/UBRE minimization	177
4.6 Numerical GCV/UBRE: performance iteration	179
4.6.1 Minimizing the GCV or UBRE score	179
Stable and efficient evaluation of the scores and derivatives	180
The weighted constrained case	183
4.7 Numerical GCV/UBRE optimization by outer iteration	184
4.7.1 Differentiating the GCV/UBRE function	184
4.8 Distributional results	187
4.8.1 Bayesian model, and posterior distribution of the parameters, for an additive model	187
4.8.2 Structure of the prior	189
4.8.3 Posterior distribution for a GAM	189
4.8.4 Bayesian confidence intervals for non-linear functions of parameters	192
4.8.5 P-values	192
4.9 Confidence interval performance	194

4.9.1	Single smooths	194
4.9.2	GAMs and their components	197
4.9.3	Unconditional Bayesian confidence intervals	200
4.10	Further GAM theory	202
4.10.1	Comparing GAMs by hypothesis testing	202
4.10.2	ANOVA decompositions and Nesting	204
4.10.3	The geometry of penalized regression	206
4.10.4	The “natural” parameterization of a penalized smoother	207
4.11	Other approaches to GAMs	210
4.11.1	Backfitting GAMs	211
4.11.2	Generalized smoothing splines	213
4.12	Exercises	215
5	GAMs in practice: mgcv	219
5.1	Cherry trees again	219
5.1.1	Finer control of <code>gam</code>	221
5.1.2	Smooths of several variables	223
5.1.3	Parametric model terms	226
5.2	Brain Imaging Example	228
5.2.1	Preliminary Modelling	230
5.2.2	Would an additive structure be better?	234
5.2.3	Isotropic or tensor product smooths?	235
5.2.4	Detecting symmetry (with <code>by</code> variables)	237
5.2.5	Comparing two surfaces	239
5.2.6	Prediction with <code>predict.gam</code>	241
	Prediction with <code>lpmatrix</code>	243
5.2.7	Variances of non-linear functions of the fitted model	244
5.3	Air Pollution in Chicago Example	245
5.4	Mackerel egg survey example	251
5.4.1	Model development	252
5.4.2	Model predictions	257

CONTENTS	ix
5.5 Portuguese larks	259
5.6 Other packages	263
5.6.1 Package <code>gam</code>	263
5.6.2 Package <code>gss</code>	265
5.7 Exercises	267
6 Mixed models	275
6.1 Mixed models for balanced data	275
6.1.1 A motivating example	275
The wrong approach: a fixed effects linear model	276
The right approach: a mixed effects model	278
6.1.2 General principles	279
6.1.3 A single random factor	280
6.1.4 A model with two factors	283
6.1.5 Discussion	288
6.2 Linear mixed models in general	289
6.2.1 Estimation of linear mixed models	290
6.2.2 Directly maximizing a mixed model likelihood in R	291
6.2.3 Inference with linear mixed models	292
Fixed effects	292
Inference about the random effects	293
6.2.4 Predicting the random effects	294
6.2.5 REML	295
The explicit form of the REML criterion	297
6.2.6 A link with penalized regression	298
6.2.7 The EM algorithm	299
6.3 Linear mixed models in R	300
6.3.1 Tree Growth: an example using <code>lme</code>	302
6.3.2 Several levels of nesting	306
6.4 Generalized linear mixed models	307
6.5 GLMMs with R	309

6.6	Generalized Additive Mixed Models	312
6.6.1	Smooths as mixed model components	313
6.6.2	Inference with GAMMs	315
6.7	GAMMs with R	316
6.7.1	A GAMM for sole eggs	316
6.7.2	The Temperature in Cairo	319
6.8	Exercises	322
A	Some Matrix Algebra	329
A.1	Basic computational efficiency	329
A.2	Covariance matrices	330
A.3	Differentiating a matrix inverse	330
A.4	Kronecker product	331
A.5	Orthogonal matrices and Householder matrices	331
A.6	QR decomposition	332
A.7	Choleski decomposition	332
A.8	Eigen-decomposition	333
A.9	Singular value decomposition	334
A.10	Pivoting	335
A.11	Lanczos iteration	335
B	Solutions to exercises	339
B.1	Chapter 1	339
B.2	Chapter 2	344
B.3	Chapter 3	349
B.4	Chapter 4	351
B.5	Chapter 5	358
B.6	Chapter 6	367
Bibliography		377
Index		382

Preface

This book is designed for readers wanting a compact, but thorough, introduction to linear models, generalized linear models , generalized additive models, and the mixed model extension of these, with particular emphasis on generalized additive models. The aim is to provide a full, but concise, theoretical treatment, explaining how the models and methods work, in order to underpin quite extensive material on practical application of the models using R.

Linear models are statistical models in which a univariate response is modelled as the sum of a ‘linear predictor’ and a zero mean random error term. The linear predictor depends on some predictor variables, measured with the response variable, and some unknown parameters, which must be estimated. A key feature of linear models is that the linear predictor depends *linearly* on these parameters. Statistical inference with such models is usually based on the assumption that the response variable has a normal distribution. Linear models are used widely in most branches of science, both in the analysis of designed experiments, and for other modeling tasks, such as polynomial regression. The linearity of the models endows them with some rather elegant theory, which is explored in some depth in Chapter 1, alongside practical examples of their use.

Generalized linear models (GLMs) somewhat relax the strict linearity assumption of linear models, by allowing the expected value of the response to depend on a smooth monotonic function of the linear predictor. Similarly the assumption that the response is normally distributed is relaxed by allowing it to follow any distribution from the exponential family (for example, normal, Poisson, binomial, gamma etc.). Inference for GLMs is based on likelihood theory, as is explained, quite fully, in chapter 2, where the practical use of these models is also covered.

A Generalized Additive Model (GAM) is a GLM in which part of the linear predictor is specified in terms of a sum of smooth functions of predictor variables. The exact parametric form of these functions is unknown, as is the degree of smoothness appropriate for each of them. To use GAMs in practice requires some extensions to GLM methods:

1. The smooth functions must be represented somehow.
2. The degree of smoothness of the functions must be made controllable, so that models with varying degrees of smoothness can be explored.

3. Some means for estimating the most appropriate degree of smoothness from data is required, if the models are to be useful for more than purely exploratory work.

This book provides an introduction to the framework for Generalized Additive Modelling in which (i) is addressed using basis expansions of the smooth functions, (ii) is addressed by estimating models by penalized likelihood maximization, in which wiggly models are penalized more heavily than smooth models in a controllable manner, and (iii) is performed using methods based on cross validation or sometimes AIC or Mallow's statistic. Chapter 3 introduces this framework, chapter 4 provides details of the theory and methods for using it, and chapter 5 illustrated the practical use of GAMs using the R package `mgcv`.

The final chapter of the book looks at mixed model extensions of linear, generalized linear, and generalized additive models. In mixed models, some of the unknown coefficients (or functions) in the model linear predictor are now treated as random variables (or functions). These 'random effects' are treated as having a covariance structure that itself depends on some unknown fixed parameters. This approach enables the use of more complex models for the random component of data, thereby improving our ability to model correlated data. Again theory and practical application are presented side by side.

I assume that most people are interested in statistical models in order to use them, rather than to gaze upon the mathematical beauty of their structure, and for this reason I have tried to keep this book practically focused. However, I think that practical work tends to progress much more smoothly if it is based on solid understanding of how the models and methods used actually work. For this reason, the book includes fairly full explanations of the theory underlying the methods, including the underlying geometry, where this is helpful. Given that the applied modelling involves using computer programs, the book includes a good deal of material on statistical modelling in R. This approach is now fairly standard when writing about practical statistical analysis, but in addition Chapter 3 attempts to introduce GAMs by having the reader 'build their own' GAM using R: I hope that this provides a useful way of quickly gaining a rather solid familiarity with the fundamentals of the GAM framework presented in this book. Once the basic framework is mastered from chapter 3, the theory in chapter 4 is really filling in details, albeit practically important ones.

The book includes a moderately high proportion of practical examples which reflect the reality that statistical modelling problems are usually quite involved, and rarely require only straightforward brain-free application of some standard model. This means that some of the examples are fairly lengthy, but do provide illustration of the process of producing practical models of some scientific utility, and of checking those models. They also provide much greater scope for the reader to decide that what I've done is utter rubbish.

Working through this book from Linear Models, through GLMs to GAMs and eventually GAMMs, it is striking that as model flexibility increases, so that the models become better able to describe the reality that we believe generated a set of data, so the methods for inference become less well founded. The linear model class is quite

restricted, but within it, hypothesis testing and interval estimation are exact, while estimation is unbiased. For the larger class of GLMs this exactness is generally lost in favour of the large sample approximations of general likelihood theory, while estimators themselves are consistent, but not necessarily unbiased. Generalizing further to GAMs, penalization lowers the convergence rates of estimators, hypothesis testing is only approximate, and satisfactory interval estimation seems to require the adoption of a Bayesian approach. With time, improved theory will hopefully reduce these differences. In the meantime, this book is offered in the belief that it is usually better to be able to say something approximate about the right model, rather than something very precise about the wrong model.

Life is too short to spend too much of it reading statistics text books. This book is of course an exception to this rule and should be read from cover to cover. However, if you don't feel inclined to follow this suggestion, here are some alternatives.

- For those who are already very familiar with linear models and GLMs, but want to use GAMs with a reasonable degree of understanding: work through Chapter 3 and read chapter 5, trying some exercises from both, use chapter 4 for reference. Perhaps skim the other chapters.
- For those who want to focus only on practical modelling in R, rather than theory. Work through the following: 1.5, 1.6.4, 1.7, 2.3, Chapter 5, 6.3, 6.5 and 6.7.
- For those familiar with the basic idea of setting up a GAM using basis expansions and penalties, but wanting to know more about the underlying theory and practical application: work through Chapters 4 and 5, and probably 6.
- For those not interested in GAMs, but wanting to know about linear models, GLMs and mixed models. Work through Chapters 1 and 2, and Chapter 6 up to section 6.6.

The book is written to be accessible to numerate researchers and students from the last two years of an undergraduate programme upwards. It is designed to be used either for self study, or as the text for the ‘regression modelling’ strand of mathematics and/or statistics degree programme. Some familiarity with statistical concepts is assumed, particularly with notions such as random variable, expectation, variance and distribution. Some knowledge of matrix algebra is also needed, although Appendix A is intended to provide all that is needed beyond the fundamentals.

Finally, I'd like to thank the people who have in various ways helped me out in the writing of this book, or in the work that lead to writing it. Among these, are Lucy Augustin, Nicole Augustin, Miguel Bernal, Steve Blythe, David Borchers, Mark Bravington, Steve Buckland, Richard Cormack, José Pedro Granadeiro ,Chong Gu, Bill Gurney, John Halley, Joe Horwood, Sharon Hedley, Peter Jupp, Alain Le Tetre, Stephan Lang, Mike Lonergan, Henric Nilsson, Roger D. Peng, Charles Paxton, Björn Stollenwerk, Yorgos Stratoudaki, the R core team in particular Kurt Hornik and Brian Ripley, the Little Italians, and the RiederAlpinists. I am also very grateful to the people who have sent me bug reports and suggestions which have greatly improved the the mgcv package over the last few years: the list is rather too long to reproduce here, but thankyou.

CHAPTER 1

Linear Models

How old is the universe? The standard big-bang model of the origin of the universe says that it expands uniformly, and locally, according to Hubble's law:

$$y = \beta x$$

where y is the relative velocity of any two galaxies separated by distance x , and β is "Hubble's constant" (in standard astrophysical notation $y \equiv v$, $x \equiv d$ and $\beta \equiv H_0$). β^{-1} gives the approximate age of the universe, but β is unknown and must somehow be estimated from observations of y and x , made for a variety of galaxies at different distances from us.

Figure 1.1 plots velocity against distance for 24 galaxies, according to measurements made using the Hubble Space Telescope. Velocities are assessed by measuring the Doppler effect red shift in the spectrum of light observed from the Galaxies concerned, although some correction for 'local' velocity components is required. Dis-

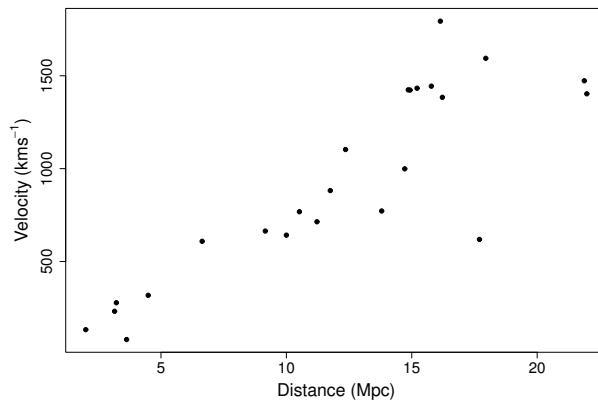


Figure 1.1 A Hubble diagram showing the relationship between distance, x , and velocity, y , for 24 Galaxies containing Cepheid stars. The data are from the Hubble Space Telescope key project to measure the Hubble constant as reported in Freedman et al. (2001).

tance measurement is much less direct, and is based on the 1912 discovery, by Henrietta Leavitt, of a relationship between the period of a certain class of variable stars, known as the Cepheids, and their luminosity. The intensity of Cepheids varies regularly with a period of between 1.5 and something over 50 days, and the mean intensity increases predictably with period. This means that, if you can find a Cepheid, you can tell how far away it is, by comparing its apparent brightness to its period predicted intensity.

It is clear, from the figure, that the observed data do not follow Hubble's law exactly, but given the measurement process, it would be surprising if they did. Given the apparent variability, what can be inferred from these data? In particular: (i) what value of β is most consistent with the data? (ii) what range of β values is consistent with the data and (iii) are some particular theoretically derived values of β consistent with the data? Statistics is about trying to answer these three sorts of question.

One way to proceed is to formulate a linear statistical model of the way that the data were generated, and to use this as the basis for inference. Specifically, suppose that, rather than being governed directly by Hubble's law, the observed velocity is given by Hubble's constant multiplied by the observed distance plus a 'random variability' term. That is

$$y_i = \beta x_i + \epsilon_i \quad i = 1 \dots 24 \quad (1.1)$$

where the ϵ_i terms are independent random variables such that $\mathbb{E}(\epsilon_i) = 0$ and $\mathbb{E}(\epsilon_i^2) = \sigma^2$. The random component of the model is intended to capture the fact that if we gathered a replicate set of data, for a new set of galaxies, Hubble's law would not change, but the apparent random variation from it would be different, as a result of different measurement errors. Notice that it is not implied that these errors are completely unpredictable: their mean and variance are assumed to be fixed, it is only their particular values, for any particular galaxy, that are not known.

1.1 A simple linear model

This section develops statistical methods for a simple linear model of the form (1.1). This allows the key concepts of linear modelling to be introduced without the distraction of any mathematical difficulty.

Formally, consider n observations, x_i, y_i , where y_i is an observation on random variable, Y_i , with expectation, $\mu_i \equiv \mathbb{E}(Y_i)$. Suppose that an appropriate model for the relationship between x and y is:

$$Y_i = \mu_i + \epsilon_i \text{ where } \mu_i = x_i \beta. \quad (1.2)$$

Here β is an unknown parameter and the ϵ_i are mutually independent zero mean random variables, each with the same variance σ^2 . So the model says that Y is given by x multiplied by a constant plus a random term. Y is an example of a *response variable*, while x is an example of a *predictor variable*. Figure 1.2 illustrates this model for a case where $n = 8$.

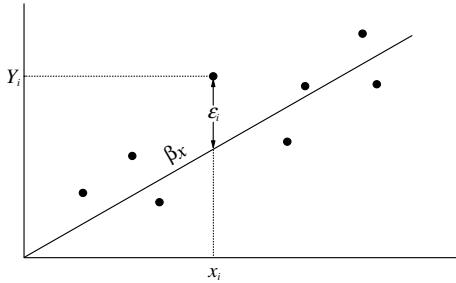


Figure 1.2 Schematic illustration of a simple linear model with one explanatory variable.

Simple least squares estimation

How can β , in model (1.2) be estimated from the x_i, y_i data? A sensible approach is to choose a value of β that makes the model fit closely to the data. To do this we need to define a measure of how well, or how badly, a model with a particular β fits the data. One possible measure is the residual sum of squares of the model:

$$\mathcal{S} = \sum_{i=1}^n (y_i - \mu_i)^2 = \sum_{i=1}^n (y_i - x_i \beta)^2$$

If we have chosen a good value of β , close to the true value, then the model predicted μ_i should be relatively close to the y_i , so that \mathcal{S} should be small, whereas poor choices will lead to μ_i far from their corresponding y_i , and high values of \mathcal{S} . Hence β can be estimated by minimizing \mathcal{S} w.r.t. β and this is known as the method of *least squares*.

To minimize \mathcal{S} , differentiate w.r.t. β :

$$\frac{\partial \mathcal{S}}{\partial \beta} = - \sum_{i=1}^n 2x_i(y_i - x_i \beta)$$

and set the result to zero to find $\hat{\beta}$, the least squares estimate of β :

$$-\sum_{i=1}^n 2x_i(y_i - x_i \hat{\beta}) = 0 \Rightarrow \sum_{i=1}^n x_i y_i - \hat{\beta} \sum_{i=1}^n x_i^2 = 0 \Rightarrow \hat{\beta} = \sum_{i=1}^n x_i y_i / \sum_{i=1}^n x_i^2. *$$

1.1.1 Sampling properties of $\hat{\beta}$

To evaluate the reliability of the least squares estimate, $\hat{\beta}$, it is useful to consider the sampling properties of $\hat{\beta}$. That is, we should consider some properties of the distribution of $\hat{\beta}$ values, which would be obtained from repeated independent replication

* $\partial^2 \mathcal{S} / \partial \beta^2 = 2 \sum x_i^2$ which is clearly positive, so a minimum of \mathcal{S} has been found.

of the x_i, y_i data used for estimation. To do this, it is helpful to introduce the concept of an *estimator*, which is obtained by replacing the observations, y_i , in the estimate of $\hat{\beta}$ by the random variables, Y_i , to obtain

$$\hat{\beta} = \sum_{i=1}^n x_i Y_i / \sum_{i=1}^n x_i^2.$$

Clearly the *estimator*, $\hat{\beta}$, is a random variable and we can therefore discuss its distribution. For now, consider only the first two moments of that distribution.

The expected value of $\hat{\beta}$ is obtained as follows:

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\left(\sum_{i=1}^n x_i Y_i / \sum_{i=1}^n x_i^2\right) = \sum_{i=1}^n x_i \mathbb{E}(Y_i) / \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i^2 \beta / \sum_{i=1}^n x_i^2 = \beta.$$

So $\hat{\beta}$ is an unbiased estimator — its expected value is equal to the true value of the parameter that it is supposed to estimate.

Unbiasedness is a reassuring property, but knowing that an estimator gets it right on average, does not tell us much about how good any one particular estimate is likely to be: for this we also need to know how much estimates would vary from one replicate data set to the next — we need to know the estimator variance.

From general probability theory we know that if Y_1, Y_2, \dots, Y_n are *independent* random variables and a_1, a_2, \dots, a_n are real constants then

$$\text{var}\left(\sum_i a_i Y_i\right) = \sum_i a_i^2 \text{var}(Y_i).$$

But we can write

$$\hat{\beta} = \sum_i a_i Y_i \text{ where } a_i = x_i / \sum_i x_i^2,$$

and from the original model specification we have that $\text{var}(Y_i) = \sigma^2$ for all i . Hence,

$$\text{var}(\hat{\beta}) = \sum_i x_i^2 / \left(\sum_i x_i^2\right)^2 \sigma^2 = \left(\sum_i x_i^2\right)^{-1} \sigma^2. \quad (1.3)$$

In most circumstances σ^2 itself is an unknown parameter and must also be estimated. Since σ^2 is the variance of the ϵ_i , it makes sense to estimate it using the variance of the estimated ϵ_i , the model residuals, $\hat{\epsilon}_i = y_i - x_i \hat{\beta}$. An unbiased estimator of σ^2 is:

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_i (y_i - x_i \hat{\beta})^2$$

(proof of unbiasedness is given later for the general case). Plugging this into (1.3) obviously gives an unbiased estimate of the variance of $\hat{\beta}$.

1.1.2 So how old is the universe?

The least squares calculations derived above are available as part of the statistical package and environment R. The function `lm` fits linear models to data, including the simple example currently under consideration. The Cepheid distance — velocity data shown in figure 1.1 are stored in a data frame[†] `hubble`. The following R code fits the model and produces the (edited) output shown.

```
> data(hubble)
> hub.mod <- lm(y~x-1,data=hubble)
> summary(hub.mod)

Call:
lm(formula = y ~ x - 1, data = hubble)

Coefficients:
            Estimate Std. Error
x       76.581     3.965
```

The call to `lm` passed two arguments to the function. The first is a *model formula*, $y \sim x - 1$, specifying the model to be fitted: the name of the response variable is to the left of ‘ \sim ’ while the predictor variable is specified on the right; the ‘ -1 ’ term indicates that the model has no ‘intercept’ term, i.e. that the model is a straight line through the origin. The second (optional) argument gives the name of the data frame in which the variables are to be found. `lm` takes this information and uses it to fit the model by least squares: the results are returned in a ‘fitted model object’, which in this case has been assigned to an object called `hub.mod` for later examination. ‘ $<-$ ’ is the assignment operator, and `hub.mod` is created by this assignment (overwriting any previously existing object of this name).

The `summary` function is then used to examine the fitted model object. Only part of its output is shown here: $\hat{\beta}$ and the estimate of the standard error of $\hat{\beta}$ (the square root of the estimated variance of $\hat{\beta}$, derived above). Before using these quantities it is important to check the model assumptions. In particular we should check the plausibility of the assumptions that the ϵ_i are independent and all have the same variance. The way to do this is to examine residual plots.

The ‘fitted values’ of the model are defined as $\hat{\mu}_i = \hat{\beta}x_i$, while the residuals are simply $\hat{\epsilon}_i = y_i - \hat{\mu}_i$. A plot of residuals against fitted values is often useful and the following produces the plot in figure 1.3 (a).

```
plot(fitted(hub.mod),residuals(hub.mod),xlab="fitted values",
      ylab="residuals")
```

What we would like to see, in such a plot, is an apparently random scatter of residuals around zero, with no trend in either the mean of the residuals, or their variability,

[†] A data frame is just a two dimensional array of data in which the values of different variables are stored in different named columns.

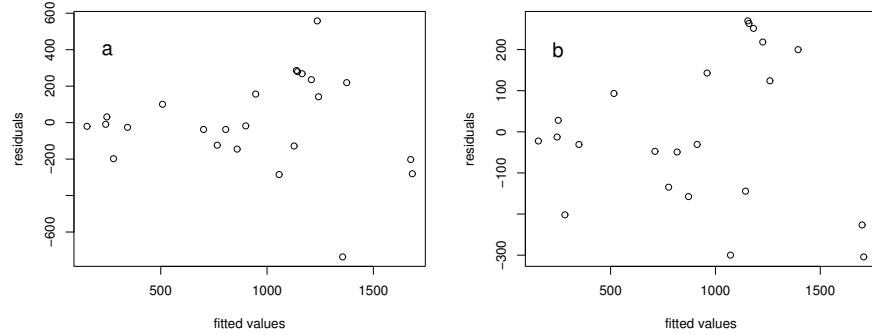


Figure 1.3 *Residuals against fitted values for (a) the model (1.1) fitted to all the data in figure 1.1 and (b) the same model fitted to data with two substantial outliers omitted.*

as the fitted values increase. A trend in the mean violates the independence assumption, and is usually indicative of something missing in the model structure, while a trend in the variability violates the constant variance assumption. The main problematic feature of figure 1.3(a) is the presence of two points with very large magnitude residuals, suggesting a problem with the constant variance assumption. It is probably prudent to repeat the model fit, with and without these points, to check that they are not having undue influence on our conclusions. The following code omits the offending points and produces a new residual plot shown in figure 1.3(b).

```
> hub.mod1 <- lm(y~x-1,data=hubble[-c(3,15),])
> summary(hub.mod1)

Call:
lm(formula = y ~ x - 1, data = hubble[-c(3, 15), ])

Coefficients:
            Estimate Std. Error
x       77.67      2.97

> plot(fitted(hub.mod1),residuals(hub.mod1),
+       xlab="fitted values",ylab="residuals")
```

The omission of the two large outliers has improved the residuals and changed $\hat{\beta}$ somewhat, but not drastically.

The Hubble constant estimates have units of $(\text{km})\text{s}^{-1} (\text{Mpc})^{-1}$. A Mega-parsec is $3.09 \times 10^{19} \text{ km}$, so we need to divide $\hat{\beta}$ by this amount, in order to obtain Hubble's constant with units of s^{-1} . The approximate age of the universe, in seconds, is then given by the reciprocal of $\hat{\beta}$. Here are the two possible estimates expressed in years:

```
> hubble.const <- c(coef(hub.mod), coef(hub.mod1))/3.09e19
> age <- 1/hubble.const
> age/(60^2*24*365)
12794692825 12614854757
```

Both fits give an age of around 13 billion years. So we now have an idea of the best estimate of the age of the universe, but what range of ages would be consistent with the data?

1.1.3 Adding a distributional assumption

So far everything done with the simple model has been based only on the model equations and the two assumptions of independence and equal variance, for the response variable. If we wish to go further, and find confidence intervals for β , or test hypotheses related to the model, then a further distributional assumption will be necessary.

Specifically, assume that $\epsilon_i \sim N(0, \sigma^2)$ for all i , which is equivalent to assuming $Y_i \sim N(x_i\beta, \sigma^2)$. Now we have already seen that $\hat{\beta}$ is just a weighted sum of Y_i , but the Y_i are now assumed to be normal random variables, and a weighted sum of normal random variables is itself a normal random variable. Hence the estimator, $\hat{\beta}$, must be a normal random variable. Since we have already established the mean and variance of $\hat{\beta}$, we have that

$$\hat{\beta} \sim N\left(\beta, \left(\sum x_i\right)^{-1} \sigma^2\right). \quad (1.4)$$

Testing hypotheses about β

One thing we might want to do is to try and evaluate the consistency of some hypothesized value of β with the data. For example some Creation Scientists estimate the age of the universe to be 6000 years, based on a reading of the Bible. This would imply that $\beta = 163 \times 10^6$ [‡]. The consistency with data of such a hypothesized value for β , can be based on the probability that we would have observed the $\hat{\beta}$ actually obtained, if the true value of β was the hypothetical one.

Specifically, we can test the null hypothesis, $H_0 : \beta = \beta_0$, versus the alternative hypothesis, $H_1 : \beta \neq \beta_0$, for some specified value β_0 , by examining the probability of getting the observed $\hat{\beta}$, or one further from β_0 , assuming H_0 to be true. If σ^2 were known then we could work directly from (1.4), as follows.

The probability required is known as the **p-value** of the test. It is the *probability of getting a value of $\hat{\beta}$ at least as favourable to H_1 as the one actually observed, if H_0 is*

[‡] This isn't really valid, of course, since the Creation Scientists are not postulating a big bang theory.

actually true[§]. In this case it helps to distinguish notationally between the estimate, $\hat{\beta}_{\text{obs}}$, and estimator $\hat{\beta}$. The p-value is then

$$\begin{aligned} p &= \Pr \left[|\hat{\beta} - \beta_0| \geq |\hat{\beta}_{\text{obs}} - \beta_0| \middle| H_0 \right] \\ &= \Pr \left[\frac{|\hat{\beta} - \beta_0|}{\sigma_{\hat{\beta}}} \geq \frac{|\hat{\beta}_{\text{obs}} - \beta_0|}{\sigma_{\hat{\beta}}} \middle| H_0 \right] \\ &= \Pr [|Z| > |z|] \end{aligned}$$

where $Z \sim N(0, 1)$, $z = (\hat{\beta}_{\text{obs}} - \beta_0)/\sigma_{\hat{\beta}}$ and $\sigma_{\hat{\beta}} = (\sum x_i^2)^{-1}\sigma^2$. Hence, having formed z , the p-value is easily worked out, using the cumulative distribution function for the standard normal, built into any statistics package. Small p-values suggest that the data are inconsistent with H_0 , while large values indicate consistency. 0.05 is often used as the boundary between ‘small’ and ‘large’ in this context.

In reality σ^2 is usually unknown. Broadly the same testing procedure can still be adopted, by replacing σ with $\hat{\sigma}$, but we need to somehow allow for the extra uncertainty that this introduces (unless the sample size is very large). It turns out that if $H_0 : \beta = \beta_0$ is true then

$$T \equiv \frac{\hat{\beta} - \beta_0}{\hat{\sigma}_{\hat{\beta}}} \sim t_{n-1}$$

where n is the sample size, $\hat{\sigma}_{\hat{\beta}} = (\sum x_i^2)^{-1}\hat{\sigma}$, and t_{n-1} is the t-distribution with $n - 1$ degrees of freedom. This result is proven in section 1.3. It is clear that large magnitude values of T favour H_1 , so using T as the test statistic, in place of $\hat{\beta}$ we can calculate a p-value by evaluating

$$p = \Pr [|T| > |t|]$$

where $T \sim t_{n-1}$ and $t = (\hat{\beta}_{\text{obs}} - \beta_0)/\hat{\sigma}_{\hat{\beta}}$. This is easily evaluated using the c.d.f. of the t distributions, built into any decent statistics package. Here is some code to evaluate the p-value for H_0 : the Hubble constant is 163000000.

```
> cs.hubble <- 163000000
> t.stat<- (coef(hub.mod1)-cs.hubble) /
+ summary(hub.mod1)$coefficients[2]
> pt(t.stat,df=21)*2 # 2 because of |T| in p-value defn.
3.906388e-150
```

i.e. as judged by the test statistic, t , the data would be hugely improbable if $\beta = 1.63 \times 10^8$. It would seem that the hypothesized value can be rejected rather firmly (in this case, using the data with the outliers increases the p-value by a factor of 1000 or so).

Hypothesis testing is particularly useful when there are good reasons to want to stick with some null hypothesis, until there is good reason to reject it. This is often the

[§] This definition holds for any hypothesis test, if the specific ‘ $\hat{\beta}$ ’ is replaced by the general ‘*a test statistic*’.

case when comparing models of differing complexity: it is often a good idea to retain the simpler model until there is quite firm evidence that it is inadequate. Note one interesting property of hypothesis testing. If we choose to reject a null hypothesis whenever the p-value is less than some fixed level, α (often termed the *significance level* of a test), then we will inevitably reject a proportion, α , of correct null hypotheses. We could try and reduce the probability of such mistakes by making α very small, but in that case we pay the price of reducing the probability of rejecting H_0 when it is false!

Confidence intervals

Having seen how to test whether a *particular* hypothesized value of β is consistent with the data, the question naturally arises of what *range* of values of β would be consistent with the data. To do this, we need to select a definition of ‘consistent’: a common choice is to say that any parameter value is consistent with the data if it results in a p-value of ≥ 0.05 , when used as the null value in a hypothesis test.

Sticking with the Hubble constant example, and working at a significance level of 0.05, we would have rejected any hypothesized value for the constant, that resulted in a t value outside the range $(-2.08, 2.08)$, since these values would result in p-values of less than 0.05. The R function `qt` can be used to find such ranges: e.g. `qt(c(0.025, 0.975), df=21)` returns the range of the middle 95% of t_{21} random variables. So we would accept any β_0 fulfilling:

$$-2.08 \leq \frac{\hat{\beta} - \beta_0}{\hat{\sigma}_{\hat{\beta}}} \leq 2.08$$

which re-arranges to give the interval

$$\hat{\beta} - 2.08\hat{\sigma}_{\hat{\beta}} \leq \beta_0 \leq \hat{\beta} + 2.08\hat{\sigma}_{\hat{\beta}}.$$

Such an interval is known as a ‘95% Confidence interval’ for β .

The defining property of a 95% confidence interval is this: if we were to gather an infinite sequence of independent replicate data sets, and calculate 95% confidence intervals for β from each, then 95% of these intervals would include the true β , and 5% would not. It is easy to see how this comes about. By construction, a hypothesis test with a significance level of 5%, rejects the correct null hypothesis for 5% of replicate data sets, and accepts it for the other 95% of replicates. Hence 5% of 95% confidence intervals must exclude the true parameter, while 95% include it.

For the Hubble example, a 95% CI for the constant (in the usual astro-physics units) is given by:

```
> sigb <- summary(hub.mod1)$coefficients[2]
> h.ci<-coef(hub.mod1)+qt(c(0.025, 0.975), df=21)*sigb
> h.ci
[1] 71.49588 83.84995
```

This can be converted to a confidence interval for the age of the universe, in years, as follows:

```
> h.ci<-h.ci*60^2*24*365.25/3.09e19 # convert to 1/years
> sort(1/h.ci)
[1] 11677548698 13695361072
```

i.e. the 95% CI is (11.7,13.7) billion years. Actually this ‘Hubble age’ is the age of the universe if it has been expanding freely, essentially unfettered by gravitation. If the universe is really ‘matter dominated’ then the galaxies should be slowed by gravity over time so that the universe would actually be younger than this, but it is time to get on with the subject of this book.

1.2 Linear models in general

The simple linear model, introduced above, can be generalized by allowing the response variable to depend on multiple predictor variables (plus an additive constant). These extra predictor variables can themselves be transformations of the original predictors. Here are some examples, for each of which a response variable datum, y_i , is treated as an observation on a random variable, Y_i , where $\mathbb{E}(Y_i) \equiv \mu_i$, the ϵ_i are zero mean random variables, and the β_j are model parameters, the values of which are unknown and will need to be estimated using data.

1. $\mu_i = \beta_0 + x_i\beta_1$, $Y_i = \mu_i + \epsilon_i$, is a straight line relationship between y and predictor variable, x .
2. $\mu_i = \beta_0 + x_i\beta_1 + x_i^2\beta_2 + x_i^3\beta_3$, $Y_i = \mu_i + \epsilon_i$, is a cubic model of the relationship between y and x .
3. $\mu_i = \beta_0 + x_i\beta_1 + z_i\beta_2 + \log(x_i z_i)\beta_3$, $Y_i = \mu_i + \epsilon_i$, is a model in which y depends on predictor variables x and z and on the log of their product.

Each of these is a linear model because the ϵ_i terms and the model parameters, β_j , enter the model in a linear way. Notice that the predictor variables can enter the model non-linearly. Exactly as for the simple model, the parameters of these models can be estimated by finding the β_j values which make the models best fit the observed data, in the sense of minimizing $\sum_i(y_i - \mu_i)^2$. The theory for doing this will be developed in section 1.3, and that development is based entirely on re-writing the linear model using using matrices and vectors.

To see how this re-writing is done, consider the straight line model given above. Writing out the μ_i equations for all n pairs, (x_i, y_i) , results in a large system of

linear equations:

$$\begin{aligned}\mu_1 &= \beta_0 + x_1\beta_1 \\ \mu_2 &= \beta_0 + x_2\beta_1 \\ \mu_3 &= \beta_0 + x_3\beta_1 \\ &\vdots \\ &\vdots \\ \mu_n &= \beta_0 + x_n\beta_1\end{aligned}$$

which can be re-written in matrix-vector form as

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \vdots \\ \mu_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

So the model has the general form $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, i.e. the expected value vector $\boldsymbol{\mu}$ is given by a **model matrix** (also known as a design matrix), \mathbf{X} , multiplied by a parameter vector, $\boldsymbol{\beta}$. All linear models can be written in this general form.

As a second illustration, the cubic example, given above, can be written in matrix vector form as

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \vdots \\ \mu_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}.$$

Models in which data are divided into different groups, each of which are assumed to have a different mean, are less obviously of the form $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$, but in fact they can be written in this way, by use of dummy indicator variables. Again, this is most easily seen by example. Consider the model

$$\mu_i = \beta_j \text{ if observation } i \text{ is in group } j,$$

and suppose that there are three groups, each with 2 data. Then the model can be re-written

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

Variables indicating the group to which a response observation belongs, are known as *factor* variables. Somewhat confusingly, the groups themselves are known as *levels*

of a factor. So the above model involves one factor, ‘group’, with three levels. Models of this type, involving factors, are commonly used for the analysis of designed experiments. In this case the model matrix depends on the design of the experiment (i.e on which units belong to which groups), and for this reason the terms ‘design matrix’ and ‘model matrix’ are often used interchangeably. Whatever it is called, \mathbf{X} is absolutely central to understanding the theory of linear models, generalized linear models and generalized additive models.

1.3 The theory of linear models

This section shows how the parameters, β , of the linear model

$$\mu = \mathbf{X}\beta, \quad \mathbf{y} \sim N(\mu, \mathbf{I}_n\sigma^2)$$

can be estimated by least squares. It is assumed that \mathbf{X} is a matrix, with n rows and p columns. It will be shown that the resulting estimator, $\hat{\beta}$, is unbiased, has the lowest variance of any possible linear estimator of β , and that, given the normality of the data, $\hat{\beta} \sim N(\beta, (\mathbf{X}^\top \mathbf{X})^{-1}\sigma^2)$. Results are also derived for setting confidence limits on parameters and for testing hypotheses about parameters — in particular the hypothesis that several elements of β are simultaneously zero.

In this section it is important not to confuse the *length* of a vector with its *dimension*. For example $(1, 1, 1)^\top$ has dimension 3 and length $\sqrt{3}$. Also note that no distinction has been made notationally between random variables and particular observations of those random variables: it is usually clear from the context which is meant.

1.3.1 Least squares estimation of β

Point estimates of the linear model parameters, β , can be obtained by the method of least squares, that is by minimizing

$$\mathcal{S} = \sum_{i=1}^n (y_i - \mu_i)^2,$$

with respect to β . To use least squares with a linear model, written in general matrix-vector form, first recall the link between the Euclidean length of a vector and the sum of squares of its elements. If \mathbf{v} is any vector of dimension, n , then

$$\|\mathbf{v}\|^2 \equiv \mathbf{v}^\top \mathbf{v} \equiv \sum_{i=1}^n v_i^2.$$

Hence

$$\mathcal{S} = \|\mathbf{y} - \mu\|^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2$$

Since this expression is simply the squared (Euclidian) length of the vector $\mathbf{y} - \mathbf{X}\beta$, its value will be unchanged if $\mathbf{y} - \mathbf{X}\beta$ is rotated. This observation is the basis for a

practical method for finding $\hat{\beta}$, and for developing the distributional results required to use linear models.

Specifically, like any real matrix, \mathbf{X} can always be decomposed

$$\mathbf{X} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_f \mathbf{R} \quad (1.5)$$

where \mathbf{R} is a $p \times p$ upper triangular matrix[†], and \mathbf{Q} is an $n \times n$ orthogonal matrix, the first p columns of which form \mathbf{Q}_f (see A.6). Recall that orthogonal matrices rotate vectors, but do not change their length. Orthogonality also means that $\mathbf{Q}\mathbf{Q}^T = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_n$. Applying \mathbf{Q}^T to $\mathbf{y} - \mathbf{X}\beta$ implies that

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 = \|\mathbf{Q}^T\mathbf{y} - \mathbf{Q}^T\mathbf{X}\beta\|^2 = \left\| \mathbf{Q}^T\mathbf{y} - \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \beta \right\|^2.$$

Writing $\mathbf{Q}^T\mathbf{y} = \begin{bmatrix} \mathbf{f} \\ \mathbf{r} \end{bmatrix}$, where \mathbf{f} is vector of dimension p , and hence \mathbf{r} is a vector of dimension $n - p$, yields

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 = \left\| \begin{bmatrix} \mathbf{f} \\ \mathbf{r} \end{bmatrix} - \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \beta \right\|^2 = \|\mathbf{f} - \mathbf{R}\beta\|^2 + \|\mathbf{r}\|^2. \dagger$$

The length of \mathbf{r} does not depend on β , while $\|\mathbf{f} - \mathbf{R}\beta\|^2$ can be reduced to zero by choosing β so that $\mathbf{R}\beta$ equals \mathbf{f} . Hence

$$\hat{\beta} = \mathbf{R}^{-1}\mathbf{f} \quad (1.6)$$

is the least squares estimator of β . Notice that $\|\mathbf{r}\|^2 = \|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2$, the residual sum of squares for the model fit.

1.3.2 The distribution of $\hat{\beta}$

The distribution of the estimator, $\hat{\beta}$, follows from that of $\mathbf{Q}^T\mathbf{y}$. Multivariate normality of $\mathbf{Q}^T\mathbf{y}$ follows from that of \mathbf{y} , and since the covariance matrix of \mathbf{y} is $\mathbf{I}_n\sigma^2$, the covariance matrix of $\mathbf{Q}^T\mathbf{y}$ is

$$\mathbf{V}_{\mathbf{Q}^T\mathbf{y}} = \mathbf{Q}^T \mathbf{I}_n \mathbf{Q} \sigma^2 = \mathbf{I}_n \sigma^2.$$

Furthermore

$$\begin{aligned} \mathbb{E} \begin{bmatrix} \mathbf{f} \\ \mathbf{r} \end{bmatrix} &= \mathbb{E}(\mathbf{Q}^T\mathbf{y}) = \mathbf{Q}^T\mathbf{X}\beta = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \beta \\ &\Rightarrow \mathbb{E}(\mathbf{f}) = \mathbf{R}\beta \text{ and } \mathbb{E}(\mathbf{r}) = \mathbf{0}. \end{aligned}$$

i.e. we have that

$$\mathbf{f} \sim N(\mathbf{R}\beta, \mathbf{I}_p\sigma^2) \text{ and } \mathbf{r} \sim N(\mathbf{0}, \mathbf{I}_{n-p}\sigma^2)$$

[†] i.e. $R_{i,j} = 0$ if $i > j$.

[‡] If the last equality isn't obvious recall that $\|\mathbf{x}\|^2 = \sum_i x_i^2$, so if $\mathbf{x} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}$, $\|\mathbf{x}\|^2 = \sum_i v_i^2 + \sum_i w_i^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2$.

with both vectors independent of each other.

Turning to the properties of $\hat{\beta}$ itself, unbiasedness follows immediately:

$$\mathbb{E}(\hat{\beta}) = \mathbf{R}^{-1}\mathbb{E}(\mathbf{f}) = \mathbf{R}^{-1}\mathbf{R}\beta = \beta.$$

Since the covariance matrix of \mathbf{f} is $\mathbf{I}_p\sigma^2$, it also follows that the covariance matrix of $\hat{\beta}$ is

$$\mathbf{V}_{\hat{\beta}} = \mathbf{R}^{-1}\mathbf{I}_p\mathbf{R}^{-T}\sigma^2 = \mathbf{R}^{-1}\mathbf{R}^{-T}\sigma^2. \quad (1.7)$$

Furthermore, since $\hat{\beta}$ is just a linear transformation of the normal random variables \mathbf{f} , it must follow a multivariate normal distribution,

$$\hat{\beta} \sim N(\beta, \mathbf{V}_{\hat{\beta}}).$$

The forgoing distributional result is not usually directly useful for making inferences about β , since σ^2 is generally unknown and must be estimated, thereby introducing an extra component of variability that should be accounted for.

1.3.3 $(\hat{\beta}_i - \beta_i)/\hat{\sigma}_{\hat{\beta}_i} \sim t_{n-p}$

Since the elements of \mathbf{r} are i.i.d. $N(0, \sigma^2)$ random variables,

$$\frac{1}{\sigma^2}\|\mathbf{r}\|^2 = \frac{1}{\sigma^2} \sum_{i=1}^{n-p} r_i^2 \sim \chi_{n-p}^2.$$

The mean of a χ_{n-p}^2 r.v. is $n - p$, so this result is sufficient (but not necessary: see exercise 7) to imply that

$$\hat{\sigma}^2 = \|\mathbf{r}\|^2/(n - p) \quad (1.8)$$

is an unbiased estimator of σ^2 [§]. The independence of the elements of \mathbf{r} and \mathbf{f} also implies that $\hat{\beta}$ and $\hat{\sigma}^2$ are independent.

Now consider a single parameter estimator, $\hat{\beta}_i$, with standard deviation, $\sigma_{\hat{\beta}_i}$, given by the square root of element i, i of $\mathbf{V}_{\hat{\beta}}$. An unbiased estimator of $\mathbf{V}_{\hat{\beta}}$ is $\hat{\mathbf{V}}_{\hat{\beta}} = \mathbf{V}_{\hat{\beta}}\hat{\sigma}^2/\sigma^2 = \mathbf{R}^{-1}\mathbf{R}^{-T}\hat{\sigma}^2$, so an estimator, $\hat{\sigma}_{\hat{\beta}_i}$, is given by the square root of element i, i of $\hat{\mathbf{V}}_{\hat{\beta}}$, and it is clear that $\hat{\sigma}_{\hat{\beta}_i} = \sigma_{\hat{\beta}_i}\hat{\sigma}/\sigma$. Hence

$$\frac{\hat{\beta}_i - \beta_i}{\hat{\sigma}_{\hat{\beta}_i}} = \frac{\hat{\beta}_i - \beta_i}{\sigma_{\hat{\beta}_i}\hat{\sigma}/\sigma} = \frac{(\hat{\beta}_i - \beta_i)/\sigma_{\hat{\beta}_i}}{\sqrt{\frac{1}{\sigma^2}\|\mathbf{r}\|^2/(n - p)}} \sim \frac{N(0, 1)}{\sqrt{\chi_{n-p}^2/(n - p)}} \sim t_{n-p}$$

(where the independence of $\hat{\beta}_i$ and $\hat{\sigma}^2$ has been used). This result enables confidence intervals for β_i to be found, and is the basis for hypothesis tests about individual β_i 's (for example $H_0 : \beta_i = 0$).

[§] Don't forget that $\|\mathbf{r}\|^2 = \|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2$.

1.3.4 F-ratio results

It is also of interest to obtain distributional results for testing, for example, the simultaneous equality to zero of several model parameters. Such tests are particularly useful for making inferences about factor variables and their interactions, since each factor (or interaction) is typically represented by several elements of β .

First consider partitioning the model matrix into two parts so that $\mathbf{X} = [\mathbf{X}_0 : \mathbf{X}_1]$, where \mathbf{X}_0 and \mathbf{X}_1 have $p - q$ and q columns, respectively. Let β_0 and β_1 be the corresponding sub-vectors of β .

$$\mathbf{Q}^T \mathbf{X}_0 = \begin{bmatrix} \mathbf{R}_0 \\ \mathbf{0} \end{bmatrix}$$

where \mathbf{R}_0 is the first $p - q$ rows and columns of \mathbf{R} (\mathbf{Q} and \mathbf{R} are from (1.5)). So rotating $\mathbf{y} - \mathbf{X}_0\beta_0$ using \mathbf{Q}^T implies that

$$\|\mathbf{y} - \mathbf{X}_0\beta_0\|^2 = \|\mathbf{f}_0 - \mathbf{R}_0\beta_0\|^2 + \|\mathbf{f}_1\|^2 + \|\mathbf{r}\|^2$$

where \mathbf{f} has been partitioned so that $\mathbf{f} = \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \end{bmatrix}$ (\mathbf{f}_1 being of dimension q). Hence $\|\mathbf{f}_1\|^2$ is the increase in residual sum of squares that results from dropping \mathbf{X}_1 from the model (i.e. setting $\beta_1 = \mathbf{0}$).

Now, under

$$H_0 : \beta_1 = \mathbf{0},$$

$\mathbb{E}(\mathbf{f}_1) = \mathbf{0}$ (using the facts that $\mathbb{E}(\mathbf{f}) = \mathbf{R}\beta$ and \mathbf{R} is upper triangular), and we already know that the elements of \mathbf{f}_1 are i.i.d. normal r.v.s with variance σ^2 . Hence, if H_0 is true,

$$\frac{1}{\sigma^2} \|\mathbf{f}_1\|^2 \sim \chi_q^2.$$

So, forming an ‘F-ratio statistic’, F , assuming H_0 , and recalling the independence of \mathbf{f} and \mathbf{r} we have

$$F = \frac{\|\mathbf{f}_1\|^2/q}{\hat{\sigma}^2} = \frac{\frac{1}{\sigma^2} \|\mathbf{f}_1\|^2/q}{\frac{1}{\sigma^2} \|\mathbf{r}\|^2/(n-p)} \sim \frac{\chi_q^2/q}{\chi_{n-p}^2/(n-p)} \sim F_{q,n-p}$$

and this result can be used to test the significance of model terms.

In general if β is partitioned into sub-vectors $\beta_0, \beta_1, \dots, \beta_m$ (each usually relating to a different effect), of dimensions q_0, q_1, \dots, q_m , then \mathbf{f} can also be so partitioned, $\mathbf{f}^T = [\mathbf{f}_0^T, \mathbf{f}_1^T, \dots, \mathbf{f}_m^T]$, and tests of

$$H_0 : \beta_j = \mathbf{0} \text{ vs. } H_1 : \beta_j \neq \mathbf{0}$$

are conducted using the result that under H_0

$$F = \frac{\|\mathbf{f}_j\|^2/q_j}{\hat{\sigma}^2} \sim F_{q_j, n-p}$$

with F larger than this suggests, if the alternative is true. This is the result used to draw up sequential ANOVA tables for a fitted model, of the sort produced by a single

argument call to `anova` in R. Note, however, that the hypothesis test about β_j is only valid in general if $\beta_k = 0$ for all k such that $j < k \leq m$: this follows from the way that the test was derived, and is the reason that the ANOVA tables resulting from such procedures are referred to as ‘sequential’ tables. The practical upshot is that, if models are reduced in a different order, the p-values obtained will be different. The exception to this is if the $\hat{\beta}_j$ ’s are mutually independent, in which case the all tests are simultaneously valid, and the ANOVA table for a model is not dependent on the order of model terms: such independent $\hat{\beta}_j$ ’s usually arise only in the context of balanced data, from designed experiments.

Notice that sequential ANOVA tables are very easy to calculate: once a model has been fitted by the QR method, all the relevant ‘sums of squares’ are easily calculated directly from the elements of \mathbf{f} , with the elements of \mathbf{r} providing the residual sum of squares.

1.3.5 The influence matrix

One matrix which will feature extensively in the discussion of GAMs is the *influence matrix* (or *hat matrix*) of a linear model. This is the matrix which yields the fitted value vector, $\hat{\mu}$, when post-multiplied by the data vector, \mathbf{y} . Recalling the definition of \mathbf{Q}_f , as being the first p columns of \mathbf{Q} , $\mathbf{f} = \mathbf{Q}_f \mathbf{y}$ and so

$$\hat{\beta} = \mathbf{R}^{-1} \mathbf{Q}_f^\top \mathbf{y}.$$

Furthermore $\hat{\mu} = \mathbf{X}\hat{\beta}$ and $\mathbf{X} = \mathbf{Q}_f \mathbf{R}$ so

$$\hat{\mu} = \mathbf{Q}_f \mathbf{R} \mathbf{R}^{-1} \mathbf{Q}_f^\top \mathbf{y} = \mathbf{Q}_f \mathbf{Q}_f^\top \mathbf{y}$$

i.e. the matrix $\mathbf{A} \equiv \mathbf{Q}_f \mathbf{Q}_f^\top$ is the influence (hat) matrix such that $\hat{\mu} = \mathbf{A}\mathbf{y}$.

The influence matrix has a couple of interesting properties. Firstly, the trace of the influence matrix is the number of (identifiable) parameters in the model, since

$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{Q}_f \mathbf{Q}_f^\top) = \text{tr}(\mathbf{Q}_f^\top \mathbf{Q}_f) = \text{tr}(\mathbf{I}_p) = p.$$

Secondly, $\mathbf{AA} = \mathbf{A}$, a property known as *idempotency*. Again the proof is simple:

$$\mathbf{AA} = \mathbf{Q}_f \mathbf{Q}_f^\top \mathbf{Q}_f \mathbf{Q}_f^\top = \mathbf{Q}_f \mathbf{I}_p \mathbf{Q}_f^\top = \mathbf{Q}_f \mathbf{Q}_f^\top = \mathbf{A}.$$

1.3.6 The residuals, $\hat{\epsilon}$, and fitted values, $\hat{\mu}$

The influence matrix is helpful in deriving properties of the fitted values, $\hat{\mu}$, and residuals, $\hat{\epsilon}$. $\hat{\mu}$ is unbiased, since $\mathbb{E}(\hat{\mu}) = \mathbb{E}(\mathbf{X}\hat{\beta}) = \mathbf{X}\mathbb{E}(\hat{\beta}) = \mathbf{X}\beta = \mu$. The covariance matrix of the fitted values is obtained from the fact that $\hat{\mu}$ is a linear transformation of the random vector \mathbf{y} , which has covariance matrix $\mathbf{I}_n \sigma^2$, so that

$$\mathbf{V}_{\hat{\mu}} = \mathbf{A} \mathbf{I}_n \mathbf{A}^\top \sigma^2 = \mathbf{A} \sigma^2,$$

by the idempotence (and symmetry) of \mathbf{A} . The distribution of $\hat{\mu}$ is degenerate multivariate normal.

Similar arguments apply to the residuals.

$$\hat{\epsilon} = (\mathbf{I} - \mathbf{A})\mathbf{y},$$

so

$$\mathbb{E}(\hat{\epsilon}) = \mathbb{E}(\mathbf{y}) - \mathbb{E}(\hat{\mu}) = \boldsymbol{\mu} - \boldsymbol{\mu} = \mathbf{0}.$$

As in the fitted value case, we have

$$\mathbf{V}_{\hat{\epsilon}} = (\mathbf{I}_n - \mathbf{A})\mathbf{I}_n(\mathbf{I}_n - \mathbf{A})^T\sigma^2 = (\mathbf{I}_n - 2\mathbf{A} + \mathbf{A}\mathbf{A})\sigma^2 = (\mathbf{I}_n - \mathbf{A})\sigma^2$$

Again, the distribution of the residuals will be degenerate normal. The results for the residuals are useful for model checking, since they allow the residuals to be standardized, so that they should have constant variance, if the model is correct.

1.3.7 Results in terms of \mathbf{X}

The presentation so far has been in terms of the method actually used to fit linear models in practice (the QR decomposition approach[¶]), which also greatly facilitates the derivation of the distributional results required for practical modelling. However, for historical reasons, these results are more usually presented in terms of the model matrix, \mathbf{X} , rather than the components of its QR decomposition. For completeness some of the results are restated here, in terms of \mathbf{X} .

Firstly consider the covariance matrix of $\boldsymbol{\beta}$. This turns out to be $(\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$, which is easily seen to be equivalent to (1.7) as follows:

$$\mathbf{V}_{\hat{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2 = (\mathbf{R}^T\mathbf{Q}_f^T\mathbf{Q}_f\mathbf{R})^{-1}\sigma^2 = (\mathbf{R}^T\mathbf{R})^{-1}\sigma^2 = \mathbf{R}^{-1}\mathbf{R}^{-T}\sigma^2.$$

The expression for the least squares estimates is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, which is equivalent to (1.6):

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{R}^{-T}\mathbf{R}^T\mathbf{Q}_f^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{Q}_f^T\mathbf{y} = \mathbf{R}^{-1}\mathbf{f}.$$

Given this last result it is easy to see that the influence matrix can be written:

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

These results are of largely historical and theoretical interest: they should not be used for computational purposes, and derivation of the distributional results is much more difficult if one starts from these formulae.

1.3.8 The Gauss Markov Theorem: what's special about least squares?

How good are least squares estimators? In particular, might it be possible to find better estimators, in the sense of having lower variance while still being unbiased?

[¶] A few programs still fit models by solution of $\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y}$, but this is less computationally stable than the rotation method described here, although it is a bit faster.

The Gauss Markov theorem shows that least squares estimators have the lowest variance of all unbiased estimators that are linear (meaning that the data only enter the estimation process in a linear way).

Theorem: Suppose that $\mu \equiv \mathbb{E}(\mathbf{Y}) = \mathbf{X}\beta$ and $\mathbf{V}_y = \sigma^2\mathbf{I}$, and let $\tilde{\phi} = \mathbf{c}^\top \mathbf{Y}$ be any *unbiased* linear estimator of $\phi = \mathbf{t}^\top \beta$, where \mathbf{t} is an arbitrary vector. Then:

$$\text{Var}(\tilde{\phi}) \geq \text{Var}(\hat{\phi})$$

where $\hat{\phi} = \mathbf{t}^\top \hat{\beta}$, and $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$ is the least squares estimator of β . Notice that, since \mathbf{t} is arbitrary, this theorem implies that each element of $\hat{\beta}$ is a minimum variance unbiased estimator.

Proof: Since $\tilde{\phi}$ is a linear transformation of \mathbf{Y} , $\text{Var}(\tilde{\phi}) = \mathbf{c}^\top \mathbf{c} \sigma^2$. To compare variances of $\hat{\phi}$ and $\tilde{\phi}$ it is also useful to express $\text{Var}(\hat{\phi})$ in terms of \mathbf{c} . To do this, note that unbiasedness of $\hat{\phi}$ implies that

$$\mathbb{E}(\mathbf{c}^\top \mathbf{Y}) = \mathbf{t}^\top \beta \Rightarrow \mathbf{c}^\top \mathbb{E}(\mathbf{Y}) = \mathbf{t}^\top \beta \Rightarrow \mathbf{c}^\top \mathbf{X}\beta = \mathbf{t}^\top \beta \Rightarrow \mathbf{c}^\top \mathbf{X} = \mathbf{t}^\top.$$

So the variance of $\hat{\phi}$ can be written as

$$\text{Var}(\hat{\phi}) = \text{Var}(\mathbf{t}^\top \hat{\beta}) = \text{Var}(\mathbf{c}^\top \mathbf{X}\hat{\beta}).$$

This is the variance of a linear transformation of $\hat{\beta}$, and the covariance matrix of $\hat{\beta}$ is $(\mathbf{X}^\top \mathbf{X})^{-1} \sigma^2$, so

$$\text{Var}(\hat{\phi}) = \text{Var}(\mathbf{c}^\top \mathbf{X}\hat{\beta}) = \mathbf{c}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{c} \sigma^2 = \mathbf{c}^\top \mathbf{A} \mathbf{c} \sigma^2$$

(where \mathbf{A} is the influence or hat matrix). Now the variances of the two estimators can be directly compared, and it can be seen that

$$\text{Var}(\tilde{\phi}) \geq \text{Var}(\hat{\phi})$$

iff

$$\mathbf{c}^\top (\mathbf{I} - \mathbf{A}) \mathbf{c} \geq 0.$$

This condition will always be met, because it is equivalent to:

$$[(\mathbf{I} - \mathbf{A}) \mathbf{c}]^\top (\mathbf{I} - \mathbf{A}) \mathbf{c} \geq 0$$

by the idempotency of $(\mathbf{I} - \mathbf{A})$, but this last condition is saying that a sum of squares can not be less than 0, which is clearly true. \square

Notice that this theorem uses independence and equal variance assumptions, but does not assume normality. Of course there is a sense in which the theorem is intuitively rather unsurprising, since it says that the minimum variance estimators are those obtained by seeking to minimize the residual variance.

1.4 The geometry of linear modelling

A full understanding of what is happening when models are fitted by least squares is facilitated by taking a geometric view of the fitting process. Some of the results derived in the last few sections become rather obvious when viewed in this way.

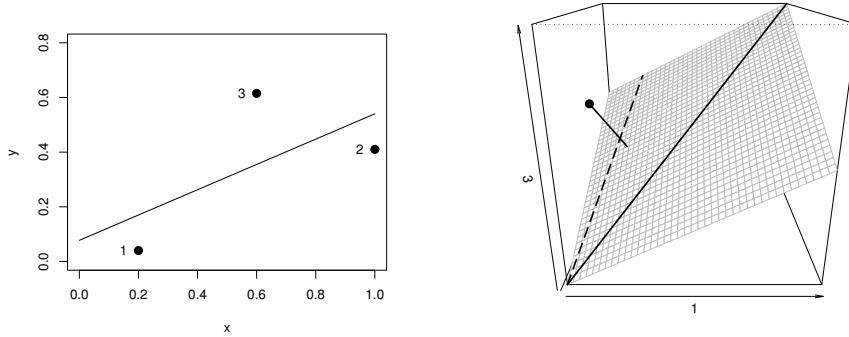


Figure 1.4 *The geometry of least squares.* The left panel shows a straight line model fitted to 3 data by least squares. The right panel gives a geometric interpretation of the fitting process. The 3 dimensional space shown is spanned by 3 orthogonal axes: one for each response variable. The observed response vector, \mathbf{y} , is shown as a point (\bullet) within this space. The columns of the model matrix define two directions within the space: the thick and dashed lines from the origin. The model states that $\mathbb{E}(\mathbf{y})$ could be any linear combination of these vectors: i.e. anywhere in the ‘model subspace’ indicated by the grey plane. Least squares fitting finds the closest point in the model sub space to the response data (\bullet): the ‘fitted values’. The short thick line joins the response data to the fitted values: it is the ‘residual vector’.

1.4.1 Least squares

Again consider the linear model,

$$\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}, \quad \mathbf{y} \sim N(\boldsymbol{\mu}, \mathbf{I}_n\sigma^2),$$

where \mathbf{X} is an $n \times p$ model matrix. But now consider an n dimensional Euclidean space, \mathbb{R}^n , in which \mathbf{y} defines the location of a single point. The space of all possible linear combinations of the columns of \mathbf{X} defines a subspace of \mathbb{R}^n , the elements of this space being given by $\mathbf{X}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ can take any value in \mathbb{R}^p : this space will be referred to as the *space of \mathbf{X}* (strictly the *column space*). So, a linear model states that, $\boldsymbol{\mu}$, the expected value of \mathbf{Y} , lies in the space of \mathbf{X} . Estimating a linear model, by least squares, amounts to finding the point, $\hat{\boldsymbol{\mu}} \equiv \mathbf{X}\hat{\boldsymbol{\beta}}$, in the space of \mathbf{X} , that is closest to the observed data \mathbf{y} . Equivalently, $\hat{\boldsymbol{\mu}}$ is the orthogonal projection of \mathbf{y} on to the space of \mathbf{X} . An obvious, but important, consequence of this is that the residual vector, $\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\boldsymbol{\mu}}$, is orthogonal to all vectors in the space of \mathbf{X} .

Figure 1.4 illustrates these ideas for the simple case of a straight line regression model for 3 data (shown conventionally in the left hand panel). The response data and model are

$$\mathbf{y} = \begin{bmatrix} .04 \\ .41 \\ .62 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\mu} = \begin{bmatrix} 1 & 0.2 \\ 1 & 1.0 \\ 1 & 0.6 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}.$$

Since $\boldsymbol{\beta}$ is unknown, the model simply says that $\boldsymbol{\mu}$ could be any linear combination of

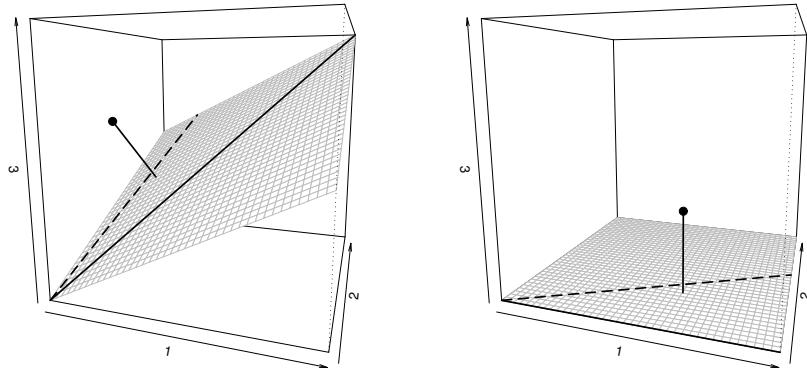


Figure 1.5 *The geometry of fitting via orthogonal decompositions.* The left panel illustrates the geometry of the simple straight line model of 3 data introduced in figure 1.4. The right hand panel shows how this original problem appears after rotation by \mathbf{Q}^T , the transpose of the orthogonal factor in a QR decomposition of \mathbf{X} . Notice that in the rotated problem the model subspace only has non-zero components relative to p axes (2 axes for this example), while the residual vector has only zero components relative to those same axes.

the vectors $[1, 1, 1]^T$ and $[.2, 1, .6]^T$. As the right hand panel of figure 1.4 illustrates, fitting the model by least squares amounts to finding the particular linear combination of the columns of these vectors, that is as close to \mathbf{y} as possible (in terms of Euclidean distance).

1.4.2 Fitting by orthogonal decompositions

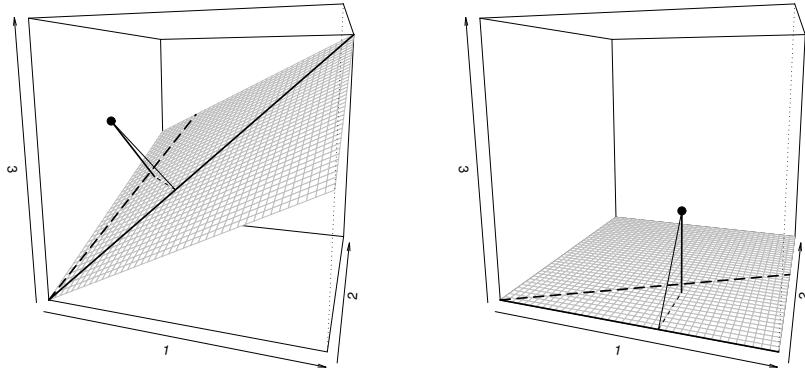
Recall that the actual calculation of least squares estimates involves first forming the QR decomposition of the model matrix, so that

$$\mathbf{X} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix},$$

where \mathbf{Q} is an $n \times n$ orthogonal matrix and \mathbf{R} is a $p \times p$ upper triangular matrix. Orthogonal matrices rotate vectors (without changing their length) and the first step in least squares estimation is to rotate both the response vector, \mathbf{y} , and the columns of the model matrix, \mathbf{X} , in exactly the same way, by pre-multiplication with \mathbf{Q}^T ^{||}.

Figure 1.5 illustrates this rotation for the example shown in figure 1.4. The left panel shows the response data and model space, for the original problem, while the right

^{||} In fact the QR decomposition is not uniquely defined, in that the sign of rows of \mathbf{Q} , and corresponding columns of \mathbf{R} , can be switched, without changing \mathbf{X} — these sign changes are equivalent to reflections of vectors, and the sign leading to maximum numerical stability is usually selected in practice. These reflections don't introduce any extra conceptual difficulty, but can make plots less easy to understand, so I have suppressed them in this example.

Figure 1.6 *The geometry of nested models.*

hand panel shows the data and space after rotation by \mathbf{Q}^T . Notice that, since the problem has simply been rotated, the relative position of the data and basis vectors (columns of \mathbf{X}) has not changed. What has changed is that the problem now has a particularly convenient orientation relative to the axes. The first two components of the fitted value vector can now be read directly from axes 1 and 2, while the third component is simply zero. By contrast, the residual vector has zero components relative to axes 1 and 2, and its non-zero component can be read directly from axis 3. In terms of section 1.3.1, these vectors are $[f^T, \mathbf{0}^T]^T$ and $[\mathbf{0}^T, r^T]^T$, respectively.

The $\hat{\beta}$ corresponding to the fitted values is now easily obtained. Of course we usually require fitted values and residuals to be expressed in terms of the un-rotated problem, but this is simply a matter of reversing the rotation using \mathbf{Q} . i.e.

$$\hat{\mu} = \mathbf{Q} \begin{bmatrix} f \\ \mathbf{0} \end{bmatrix}, \text{ and } \hat{\epsilon} = \mathbf{Q} \begin{bmatrix} \mathbf{0} \\ r \end{bmatrix}.$$

1.4.3 Comparison of nested models

A linear model with model matrix \mathbf{X}_0 is nested within a linear model with model matrix \mathbf{X}_1 if they are models for the same response data, and the columns of \mathbf{X}_0 span a subspace of the space spanned by the columns of \mathbf{X}_1 . Usually this simply means that \mathbf{X}_1 is \mathbf{X}_0 with some extra columns added.

The vector of the difference between the fitted values of two nested linear models is entirely within the subspace of the larger model, and is therefore orthogonal to the residual vector for the larger model. This fact is geometrically obvious, as figure 1.6 illustrates, but it is a key reason why F ratio statistics have a relatively simple distribution (under the simpler model).

Figure 1.6 is again based on the same simple straight line model that forms the basis for figures 1.4 and 1.5, but this time also illustrates the least squares fit of the

simplified model

$$y_i = \beta_0 + \epsilon_i,$$

which is nested within the original straight line model. Again, both the original and rotated versions of the model and data are shown. This time the fine continuous line shows the projection of the response data onto the space of the simpler model, while the fine dashed line shows the vector of the difference in fitted values between the two models. Notice how this vector is orthogonal both to the reduced model subspace and the full model residual vector.

The right panel of figure 1.6 illustrates that the rotation, using the transpose of the orthogonal factor \mathbf{Q}_r of the full model matrix, has also lined up the problem very conveniently for estimation of the reduced model. The fitted value vector for the reduced model now has only one non-zero component, which is the component of the rotated response data (●) relative to axis 1. The residual vector has gained the component that the fitted value vector has lost, so it has zero component relative to axis one, while its other components are the positions of the rotated response data relative to axes 2 and 3.

So, much of the work required for estimating the simplified model has already been done, when estimating the full model. Note, however, that if our interest had been in comparing the full model to the model

$$y_i = \beta_1 x_i + \epsilon_i,$$

then it would have been necessary to reorder the columns of the full model matrix, in order to avoid extra work in this way.

1.5 Practical linear models

This section covers practical linear modelling, via an extended example: the analysis of data reported by Baker and Bellis (1993), which they used to support a theory of ‘sperm competition’ in humans. The basic idea is that it is evolutionarily advantageous for males to (sub-consciously) increase their sperm count in proportion to the opportunities that their mate may have had for infidelity. Such behaviour has been demonstrated in a wide variety of other animals, and using a sample of student and staff volunteers from Manchester University, Baker and Bellis set out to see if there is evidence for similar behaviour in humans. Two sets of data will be examined: `sperm.comp1` contains data on sperm count, time since last copulation and proportion of that time spent together, for single copulations, from 15 heterosexual couples; `sperm.comp2` contains data on median sperm count, over multiple copulations, for 24 heterosexual couples, along with the weight, height and age of the male and female of each couple, and the volume of one teste of the male. From these data, Baker and Bellis concluded that sperm count increases with the proportion of time, since last copulation, that a couple have spent apart, and that sperm count increases with female weight.

In general, practical linear modelling is concerned with finding an appropriate model

<code>lm</code>	Estimates a linear model by least squares. Returns a fitted model object of class <code>lm</code> containing parameter estimates plus other auxiliary results for use by other functions.
<code>plot</code>	Produces model checking plots from a fitted model object.
<code>summary</code>	Produces summary information about a fitted model, including parameter estimates, associated standard errors and p-values, r^2 etc.
<code>anova</code>	Used for model comparison based on F ratio testing.
<code>AIC</code>	Extract Akaike's information criterion for a model fit.**
<code>residuals</code>	Extract an array of model residuals from a fitted model.
<code>fitted</code>	Extract an array of fitted values from a fitted model object.
<code>predict</code>	Obtain predicted values from a fitted model, either for new values of the predictor variables, or for the original values. Standard errors of the predictions can also be returned.

Table 1.1 *Some standard linear modelling functions. Strictly all of these functions except `lm` itself end .`lm`, but when calling them with an object of class `lm` this may be omitted.*

to explain the relationship of a response (random) variable to some predictor variables. Typically, the first step is to decide on a linear model that can reasonably be supposed capable of describing the relationship, in terms of the predictors included and the functional form of their relationship to the response. In the interests of ensuring that the model is not too restrictive this ‘full’ model is often more complicated than is necessary, in that the most appropriate value for a number of its parameters may, in fact, be zero. Part of the modelling process is usually concerned with ‘model selection’: that is deciding which parameter values ought to be zero. At each stage of model selection it is necessary to estimate model parameters by least squares fitting, and it is equally important to check the model assumptions (particularly equal variance and independence) by examining diagnostic plots. Once a model has been selected and estimated, its parameter estimates can be interpreted, in part with the aid of confidence intervals for the parameters, and possibly with other follow up analyses. In R these practical modelling tasks are facilitated by a large number of functions for linear modelling, some of which are listed in table 1.1.

1.5.1 Model fitting and model checking

The first thing to do with the sperm competition data is to have a look at it.

```
pairs(sperm.compl[, -1])
```

produces the plot shown in figure 1.7. The columns of the data frame are plotted against each other pairwise (with each pairing transposed between lower left and upper right of the plot); the first column has been excluded from the plot as it simply contains subject identification labels. The clearest pattern seems to be of some decrease in sperm count as the proportion of time spent together increases.

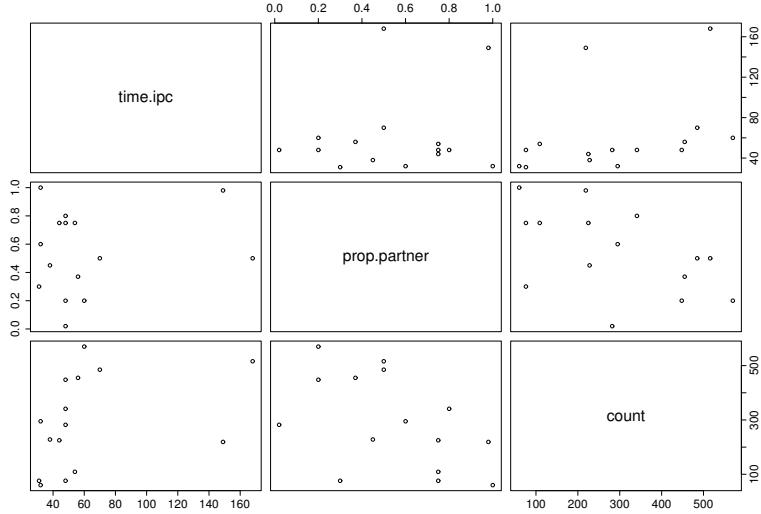


Figure 1.7 Pairs plot of the sperm competition data from Baker and Bellis 1993. ‘Count’ is sperm count (millions) from one copulation, ‘time.ipc’ is time (hours) since the previous copulation and ‘prop.partner’ is the proportion of the time since the previous copulation that the couple have spent together.

Following Baker and Bellis, a reasonable initial model might be,

$$y_i = \beta_0 + t_i\beta_1 + p_i\beta_2 + \epsilon_i, \quad (1.9)$$

where y_i is sperm count (count), t_i is the time since last inter pair copulation (time.ipc) and p_i is the proportion of time, since last copulation, that the pair have spent together (prop.partner). As usual, the β_j are unknown parameters and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. Really this model defines the *class* of models thought to be appropriate: it is not immediately clear whether either of β_1 and β_2 are non-zero.

The following fits the model (1.9) and stores the results in an object called sc.mod1.

```
sc.mod1 <- lm(count ~ time.ipc + prop.partner, sperm.compl)
```

The first argument to `lm` is a model formula, specifying the structure of the model to be fitted. In this case, the response (to the left of `~`) is `count`, and this is to depend on variables `time.ipc` and `prop.partner`. By default, the model will include an intercept term, unless it is suppressed by a ‘`-1`’ in the formula. The second argument to `lm` supplies a data frame, within which the variables in the formula can be found.

The terms on the right hand side of the model formula specify how the model matrix, X , is to be specified. In fact, in this example, the terms give the model matrix columns directly. It is possible to check the model matrix of a linear model:

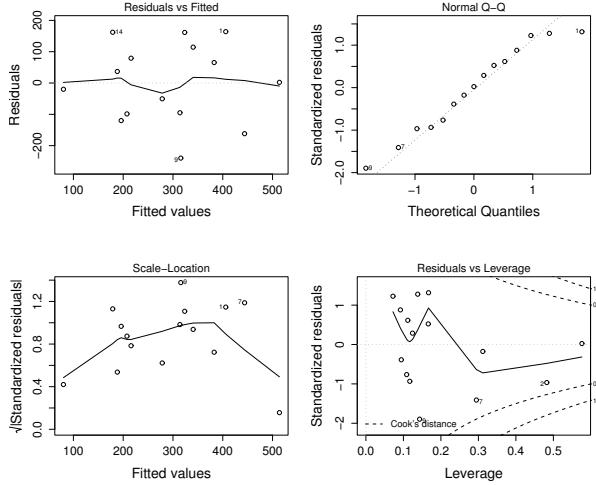


Figure 1.8 *Model checking plots for the linear model sc.mod1.*

```
> model.matrix(sc.mod1)
  (Intercept) time.ipc prop.partner
1             1      60       0.20
2             1     149       0.98
3             1      70       0.50
4             1     168       0.50
5             1      48       0.20
6             1      32       1.00
7             1      48       0.02
8             1      56       0.37
9             1      31       0.30
10            1      38       0.45
11            1      48       0.75
12            1      54       0.75
13            1      32       0.60
14            1      48       0.80
15            1      44       0.75
```

Having fitted the model, it is important to check the plausibility of the assumptions, graphically.

```
par(mfrow=c(2,2)) # split the graphics device into 4 panels
plot(sc.mod1)      # (uses plot.lm as cs.mod1 is class 'lm')
```

The resulting plots, shown in figure 1.8, require some explanation. Note that, in two of the plots, the residuals have been scaled, by dividing them by their estimated standard deviation (see section 1.3.6). If the model assumptions are met, then this standardization should result in residuals that look like $N(0, 1)$ random deviates.

- The upper left plot shows the model residuals, $\hat{\epsilon}_i$, against the model fitted values, $\hat{\mu}_i$, where $\hat{\mu} = \mathbf{X}\hat{\beta}$ and $\hat{\epsilon} = \mathbf{y} - \hat{\mu}$. The residuals should be evenly scattered above and below zero (the distribution of fitted values is not of interest). A trend in the mean of the residuals would violate the assumption of independent response variables, and usually results from an erroneous model structure: e.g. assuming a linear relationship with a predictor, when a quadratic is required, or omitting an important predictor variable. A trend in the variability of the residuals suggests that the variance of the response is related to its mean, violating the constant variance assumption: transformation of the response or use of a GLM may help, in such cases. The plot shown does not indicate any problem.
- The lower left plot is a scale-location plot. The raw residuals are standardized by dividing by their estimated standard deviation, $\hat{\sigma}\sqrt{1 - A_{ii}}$ (\mathbf{A} is the influence matrix). The square root of the absolute value of each standardized residual is then plotted against the equivalent fitted value. It can be easier to judge the constant variance assumption from such a plot, and the square root transformation reduces the skew in the distribution, which would otherwise be likely to occur. Again, the plot shown gives no reason to doubt the constant variance assumption.
- The upper right panel is a normal Q-Q (quantile-quantile) plot. The standardized residuals are sorted and then plotted against the quantiles of a standard normal distribution. If the residuals are normally distributed then the resulting plot should look like a straight line relationship, perturbed by some random scatter. The current plot fits this description, so the normality assumption seems plausible.
- The lower left panel plots the standardized residuals against the *leverage* of each datum. The leverage is simply A_{ii} , which measures the *potential* for particular datum to influence the overall model fit. The combination of a large residual and high leverage implies that the corresponding datum has a substantial influence on the overall fit. Large leverage with a small residual generally implies that, although the datum would exert undue influence on the fit, if out of line with the other data, it is actually consistent with the other data, and does not have too great an influence. Similarly a large residual may not have much influence on the fit if the corresponding datum has low leverage.

A good summary of how much influence each datum actually has is provided by its *Cook's distance*. Cook's distance is a measure of how much influence each observation has on the fitted model. If $\hat{\mu}_i^{[k]}$ is the i^{th} fitted value, when the k^{th} datum is omitted from the fit, then Cook's distance is

$$d_k = \frac{1}{(p+1)\hat{\sigma}^2} \sum_{i=1}^n (\hat{\mu}_i^{[k]} - \hat{\mu}_i)^2, \quad (1.10)$$

where p is the number of parameters and n the number of data. A very large value of d_k indicates a point that has a substantial influence on the model results. If the Cook's distance values indicate that model estimates may be very sensitive to just one or two data, then it is usually prudent to repeat any analysis without the offending points, in order to check the robustness of the modelling conclusions. Cook's distance can be shown to be a function of leverage and standardized residual, so

contours of Cook's distance are shown on the plot, from which the Cook's distance for any datum can be read. In this case none of the points look wildly out of line.

The QQ plot shows the reference line that a 'perfect' set of residuals would follow, which the other plots have a simple smooth overlaid, in order to guide the eye when looking for patterns in the residuals (these can be very useful for large datasets). By default the 'most extreme' three points in each plot are labelled with their row index in the original data frame, so that the corresponding data can be readily checked. The 9th datum is flagged in all 4 plots in figure 1.8. It should be checked:

```
> sperm.compl[9,]
  subject time.ipc prop.partner count
9          P        31         0.3      76
```

This subject has quite a low count, but not the lowest in the frame. Examination of the plot of `count` against `prop.partner` indicates that the point adds substantially to the uncertainty surrounding the relationship, but its hard to see a good reason to remove it, particularly since, if anything, it is obscuring the relationship, rather than exaggerating it.

Since the assumptions of model (1.9) appear reasonable, we can proceed to examine the fitted model object. Typing the name of an object in R causes the default print method for the object to be invoked (`print.lm` in this case).

```
> sc.mod1

Call:
lm(formula=count ~ time.ipc + prop.partner, data=sperm.compl)

Coefficients:
(Intercept)       time.ipc   prop.partner
            357.418           1.942        -339.560
```

The intercept parameter (β_0) is estimated to be 357.4. Notionally, this would be the count expected if `time.ipc` and `prop.partner` were zero, but the value is biologically implausible if interpreted in this way. Given that the smallest observed `time.ipc` was 31 hours we cannot really expect to predict the count at zero. The remaining two parameter estimates are $\hat{\beta}_1$ and $\hat{\beta}_2$, and are labelled by the name of the variable to which they relate. In both cases they give the expected increase in count for a unit increase in their respective predictor variable. Note the important point that the absolute values of the parameter estimates are only interpretable *relative* to the variable which they multiply. For example, we are not entitled to conclude that the effect of `prop.partner` is much greater than that of `time.ipc`, on the basis of the relative magnitudes of the respective parameters: they are measured in completely different units.

One point to consider, is whether `prop.partner` is the most appropriate predictor variable. Perhaps the total time spent together (in hours) would be a better predictor.

```
sc.mod2 <- lm(count~time.ipc+I(prop.partner*time.ipc),
               sperm.compl)
```

would fit such a model. The term `I(prop.partner*time.ipc)` indicates that, rather than use proportion of time together as a predictor, total time should be used. The `I()` function is used to ‘protect’ the product `prop.partner*time.ipc` within the model formula. This is necessary because symbols like `+` and `*` have special meanings within model formulae (see section 1.7): by protecting terms using `I()`, the usual arithmetic meanings are restored. Examination of diagnostic plots for `sc.mod2` shows that two points have much greater influence on the fit than the others, so for the purposes of this section it seems sensible to stick with the biologists’ preferred model structure and use `prop.partner`.

1.5.2 Model summary

The `summary`^{††} function provides a good deal more information about the fitted model.

```
> summary(sc.mod1)

Call:
lm(formula=count~time.ipc+prop.partner, data=sperm.compl)

Residuals:
    Min      1Q   Median      3Q     Max 
-239.740 -96.772   2.171  96.837 163.997 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 357.4184    88.0822   4.058  0.00159 ***
time.ipc     1.9416     0.9067   2.141  0.05346 .  
prop.partner -339.5602   126.2535  -2.690  0.01969 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 136.6 on 12 degrees of freedom
Multiple R-Squared: 0.4573,    Adjusted R-squared: 0.3669 
F-statistic: 5.056 on 2 and 12 DF,  p-value: 0.02554
```

The explanation of the parts of this output are as follows:

`Call` simply reminds you of the call that generated the object being summarized.

^{††}Note that calling the `summary` function with a fitted linear model object, `x`, actually results in the following: `summary` looks at the class of the `x`, finds that it is `lm` and passes it to `summary.lm`; `summary.lm` calculates a number of interesting quantities from `x` which it returns in a list, `y`, of class `lm.summary`; unless `y` is assigned to an object, `R` prints it, using the print method `print.lm.summary`.

Residuals gives a five figure summary of the residuals: this should indicate any gross departure from normality, for example a very skewed set of residuals might lead to very different magnitudes for $Q1$ and $Q2$, or to the median residual being a long way from 0 (the mean residual is always zero if the model includes an intercept: see exercise 6).

Coefficients gives a table relating to the estimated parameters of the model. The first two columns are the least squares estimates ($\hat{\beta}_j$'s) and the estimated standard errors associated with those estimates ($\hat{\sigma}_{\hat{\beta}_j}$), respectively. The standard error calculations follow sections 1.3.2 and 1.3.3. The third column gives the parameter estimates divided by their estimated standard errors:

$$T_j \equiv \frac{\hat{\beta}_j}{\hat{\sigma}_{\hat{\beta}_j}}.$$

T_j is a standardized measure of how far each parameter estimate is from zero. It was shown, in section 1.3.3, that under $H_0 : \beta_j = 0$,

$$T_j \sim t_{n-p}, \quad (1.11)$$

where n is the number of data and p the number of model parameters estimated. i.e. if the null hypothesis is true, then the observed T_j should be consistent with having been drawn from a t_{n-p} distribution. The final $\text{Pr}(>|t|)$ column provides the measure of that consistency, namely the probability that the magnitude of a t_{n-p} random variable would be at least as large as the observed T_j . This quantity is known as the **p-value** of the test of $H_0 : \beta_j = 0$. A large p-value indicates that the data are consistent with the hypothesis, in that the observed T_j is quite a probable value for a t_{n-p} deviate, so that there is no reason to doubt the hypothesis underpinning (1.11). Conversely a small p-value suggests that the hypothesis is wrong, since the observed T_j is a rather improbable observation from the t_{n-p} distribution implied by $\beta_j = 0$. Various arbitrary **significance levels** are often used as the boundary p-values for deciding whether to accept or reject hypotheses. Some common ones are listed at the foot of the table, and the p-values are flagged according to which, if any, they fall below.

Residual standard error gives $\hat{\sigma}$ where $\hat{\sigma}^2 = \sum(\hat{\epsilon}_i^2)/(n - p)$ (see section 1.3.3). $n - p$ is the ‘residual degrees of freedom’.

Multiple R-squared is an estimate of the proportion of the variance in the data explained by the regression:

$$r^2 = 1 - \frac{\sum \hat{\epsilon}_i^2/n}{\sum (y_i - \bar{y})^2/n}$$

where \bar{y} is the mean of the y_i . The fraction in this expression is basically an estimate of the proportion variance not explained by the regression.

Adjusted R-squared. The problem with r^2 is that it always increases when a new predictor variable is added to the model, no-matter how useless that variable

is for prediction. Part of the reason for this is that the variance estimates used to calculate r^2 are biased in a way that tends to inflate r^2 . If unbiased estimators are used we get the adjusted r^2

$$r_{\text{adj}}^2 = 1 - \frac{\sum \hat{\epsilon}_i^2 / (n - p)}{\sum (y_i - \bar{y})^2 / (n - 1)}.$$

A high value of r_{adj}^2 indicates that the model is doing well at explaining the variability in the response variable.

F-statistic. The final line, giving an F-statistic and p-value, is testing the null hypothesis that the data were generated from a model with only an intercept term, against the alternative that the fitted model generated the data. This line is really about asking if the whole model is of any use. The theory of such tests is covered in section 1.3.4.

So the summary of `sc.mod1` suggests that there is evidence that the model is better than one including just a constant (p-value = 0.02554). There is quite clear evidence that `prop.partner` is important in predicting sperm count (p-value = 0.019), but less evidence that `time.ipc` matters (p-value = 0.053). Indeed, using the conventional significance level of 0.05, we might be tempted to conclude that `time.ipc` does not affect count at all. Finally note that the model leaves most of the variability in count unexplained, since r_{adj}^2 is only 37%.

1.5.3 Model selection

From the model summary it appears that `time.ipc` may not be necessary: the associated p-value of 0.053 does not provide strong evidence that the true value of β_1 is non-zero. By the ‘true value’ is meant the value of the parameter in the model imagined to have actually generated the data; or equivalently, the value of the parameter applying to the whole population of couples from which, at least conceptually, our particular sample has been randomly drawn. The question then arises of whether a simpler model, without any dependence on `time.ipc`, might be appropriate. This is a question of *model selection*. Usually it is a good idea to avoid over-complicated models, dependent on irrelevant predictor variables, for reasons of interpretability and efficiency. Inferences about causality will be made more difficult if a model contains spurious predictors, but estimates using such a model will also be less precise, as more parameters than necessary have been estimated from the finite amount of uncertain data available.

Several approaches to model selection are based on hypothesis tests about model terms, and can be thought of as attempting to find the simplest model consistent with a set of data, where consistency is judged relative to some threshold p-value. For the sperm competition model the p-value for `time.ipc` is greater than 0.05, so this predictor might be a candidate for dropping.

```
> sc.mod3 <- lm(count ~ prop.partner, sperm.compl)
```

```

> summary(sc.mod3)
(edited)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 451.50     86.23   5.236 0.000161 ***
prop.partner -292.23    140.40  -2.081 0.057727 .
---
Residual standard error: 154.3 on 13 degrees of freedom
Multiple R-Squared:  0.25,      Adjusted R-squared: 0.1923
F-statistic: 4.332 on 1 and 13 DF,  p-value: 0.05773

```

These results provide a good example of why it is dangerous to apply automatic model selection procedures unthinkingly. In his case dropping `time.ipc` has made the estimate of the parameter multiplying `prop.partner` less precise: indeed this term also has a p-value greater than 0.05 according to this new fit. Furthermore, the new model has a much reduced r^2 , while the model's overall p-value does not give strong evidence that it is better than a model containing only an intercept. The only sensible choice here is to revert to `sc.mod1`. The statistical evidence indicates that it is better than the intercept only model, and dropping its possibly 'non-significant' term has lead to a much worse model.

Hypothesis testing is not the only approach to model selection. One alternative is to try and find the model that gets as close as possible to the true model, rather than to find the simplest model consistent with data. In this case we can attempt to find the model which does the best job of predicting the $\mathbb{E}(y_i)$. Selecting models in order to minimize Akaike's Information Criterion (AIC) is one way of trying to do this (see section 1.8.5). In R, the `AIC` function can be used to calculate the AIC statistic for different models.

```

> sc.mod4 <- lm(count~1,sperm.compl) # null model
> AIC(sc.mod1,sc.mod3,sc.mod4)
      df      AIC
sc.mod1 4 194.7346
sc.mod3 3 197.5889
sc.mod4 2 199.9031

```

This alternative model selection approach also suggests that the model with both `time.ipc` and `prop.partner` is best.

So, on the basis of `sperm.compl`, there seems to be reasonable evidence that sperm count increases with `time.ipc` but decreases with `prop.partner`: exactly as Baker and Bellis concluded.

1.5.4 Another model selection example

The second dataset from Baker and Bellis (1993) is `sperm.comp2`. This gives median sperm count for 24 couples, along with ages (years), heights (cm) and weights

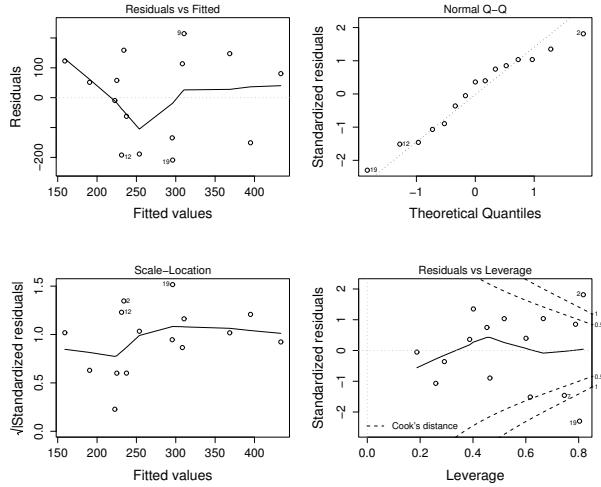


Figure 1.9 *Model checking plots for the sc2.mod1 model.*

(kg) for the male and female of each couple and volume (cm^3) of one teste for the male of the couple (m.vol). There are quite a number of missing values for the predictors, particularly for m.vol , but, for the 15 couples for which there is an m.vol measurement, the other predictors are also available. The number of copulations over which the median count has been taken varies widely from couple to couple. Ideally one should probably allow within couple and between couple components to the random variability component of the data, to allow for this, but this will not be done here. Following Baker and Bellis it seems reasonable to start from a model including linear effects of all predictors. i.e.

$$\begin{aligned} \text{count}_i = & \beta_0 + \beta_1 \times \text{f.age}_i + \beta_2 \times \text{f.weight}_i + \beta_3 \times \text{f.height}_i + \beta_4 \times \text{m.age}_i \\ & + \beta_5 \times \text{m.weight}_i + \beta_6 \times \text{m.height}_i + \beta_7 \times \text{m.vol} + \epsilon_i \end{aligned}$$

The following estimates and summarizes the model, and plots diagnostics.

```
> sc2.mod1<-lm(count ~ f.age+f.height+f.weight+m.age+m.height+
+ m.weight+m.vol, sperm.comp2)
> plot(sc2.mod1)
> summary(sc2.mod1)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1098.518    1997.984 -0.550   0.600
f.age        10.798     22.755   0.475   0.650
f.height     -4.639     10.910  -0.425   0.683
f.weight      19.716    35.709   0.552   0.598
m.age       -1.722     10.219  -0.168   0.871
```

m.height	6.009	10.378	0.579	0.581
m.weight	-4.619	12.655	-0.365	0.726
m.vol	5.035	17.652	0.285	0.784

Residual standard error: 205.1 on 7 degrees of freedom
 Multiple R-Squared: 0.2192, Adjusted R-squared: -0.5616
 F-statistic: 0.2807 on 7 and 7 DF, p-value: 0.9422

The resulting figure 1.9, looks reasonable, but datum 19 appears to produce the most extreme point on all 4 plots. Checking row 19 of the data frame, shows that the male of this couple is rather heavy (particularly for his height), and has a large m.vol measurement, but a count right near the bottom of the distribution (actually down at the level that might be expected to cause fertility problems, if this is typical). Clearly, whatever we conclude from these data will need to be double checked without this observation. Notice, from the summary, how poorly this model does at explaining the count variability: the adjusted r^2 is actually negative, an indication that we have a large number of irrelevant predictors in the model.

There are only 15 data from which to estimate the 8 parameters of the full model: it would be better to come up with something more parsimonious. In this case using AIC suggests a rather complicated model with only m.weight dropped. It seems quite sensible to switch to hypothesis testing based model selection, and ask whether there is really good evidence that all these terms are necessary? One approach is to perform ‘backwards model selection’, by repeatedly removing the *single* term with highest p-value, above some threshold (e.g. 0.05), and then refitting the resulting reduced model, until all terms have significant p-values. For example the first step in this process would remove m.age:

```
> sc2.mod2<-lm(count~f.age+f.height+f.weight+m.height+
+ m.weight+m.vol,sperm.comp2)
> summary(sc2.mod2)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1054.770   1856.843  -0.568   0.586    
f.age        8.847     18.359   0.482   0.643    
f.height     -5.119     9.871   -0.519   0.618    
f.weight      20.259    33.334   0.608   0.560    
m.height      6.033     9.727   0.620   0.552    
m.weight     -4.473    11.834  -0.378   0.715    
m.vol         4.506    16.281   0.277   0.789    

Residual standard error: 192.3 on 8 degrees of freedom
Multiple R-Squared: 0.216, Adjusted R-squared: -0.372 
F-statistic: 0.3674 on 6 and 8 DF, p-value: 0.8805
```

Notice how, relative to sc2.mod1, the reduced model has different estimates for each of the remaining parameter, as well as smaller standard error estimates for each

parameter, and (consequently) different p-values. This is part of the reason for only dropping one term at a time: when we drop one term from a model, it is quite possible for some remaining terms to have their p-values massively reduced. For example, if two terms are highly correlated it is quite possible for both to be highly significant individually, but both to have very high p-values if present together. This occurs because the terms are to some extent interchangeable predictors: the information provided by one is much the same as the information provided by the other, so that one must be present in the model but both are not needed. If we were to drop several terms from a model at once we might miss such effects.

Proceeding with backwards selection, we would drop `m.vol` next. This allows rather more of the couples to be used in the analysis. Continuing in the same way leads to the dropping of `m.weight`, `f.height`, `m.height` and finally `f.age` before arriving at a final model which includes only `f.weight`.

```
> sc2.mod7<-lm(count~f.weight,sperm.comp2)
> summary(sc2.mod7)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1002.281    489.352 -2.048   0.0539 .
f.weight      22.397     8.629   2.595   0.0173 *
Residual standard error: 147.3 on 20 degrees of freedom
Multiple R-Squared: 0.252,          Adjusted R-squared: 0.2146
F-statistic: 6.736 on 1 and 20 DF,  p-value: 0.01730
```

This model does appear to be better than a model containing only a constant, according both to a hypothesis test at the 5% level and AIC.

Apparently then, only female weight influences sperm count. This concurs with the conclusion of Baker and Bellis (1993), who interpreted the findings to suggest that males might ‘invest’ more in females with a higher reproductive potential. However, in the light of the residual plots we need to re-analyze the data without observation 19, before having too much confidence in the conclusions. This is easily done, for example ...

```
> sc <- sperm.comp2[-19,]
> sc3.mod1<-lm(count~f.age+f.height+f.weight+m.age+m.height+
+ m.weight+m.vol,sc)
> summary(sc3.mod1)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1687.406   1251.338   1.348   0.2262
f.age        55.248    15.991   3.455   0.0136 *
f.height     21.381     8.419   2.540   0.0441 *
f.weight     -88.992    31.737  -2.804   0.0310 *
m.age       -17.210     6.555  -2.626   0.0393 *
m.height    -11.321     6.869  -1.648   0.1504
```

```
m.weight      6.885      7.287    0.945    0.3812
m.vol        48.996     13.938    3.515    0.0126 *
--- [edited]
```

Notice how `m.vol` now has the lowest p-value. Repeating the whole backwards selection process, every term now drops out except for `m.vol`, leading to the much less interesting conclusion that the data only really supply evidence that size of testes influences sperm count. Given the rather tedious plausibility of this conclusion it probably makes sense to prefer it to the conclusion based on the full data set.

A follow up

Given the biological conclusions from the analysis of `sperm.comp2`, it would make sense to revisit the analysis of `sperm.comp1`. Baker and Bellis do not report `m.vol` values for these data, but the same couples feature in both datasets and are identified by label, so the required values can be obtained:

```
sperm.comp1$m.vol <-
sperm.comp2$m.vol[sperm.comp2$pair%in%sperm.comp1$subject]
```

Repeating the same sort of backwards selection we end up selecting a 1 term model:

```
> scl.mod1<-lm(count~m.vol,sperm.compl)
> summary(scl.mod1)

Call:
lm(formula = count ~ m.vol, data = sperm.compl)

Residuals:
    Min      1Q  Median      3Q      Max 
-187.236 -55.028 - 8.606  75.928 156.257 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -58.694    121.619 -0.483   0.6465    
m.vol        23.247     7.117   3.266   0.0171 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 120.8 on 6 degrees of freedom
Multiple R-Squared:  0.64,      Adjusted R-squared:  0.58 
F-statistic: 10.67 on 1 and 6 DF,  p-value: 0.01711
```

Although based on only 8 couples, this must call into question the original analysis, which concluded that time since last copulation and proportion of time spent together controlled sperm count. There is at least a suggestion that the explanation for sperm count variability may be rather more prosaic than is predicted by sperm competition theory.

1.5.5 Confidence intervals

Exactly as in section 1.1.3 the results from section 1.3.3 can be used to obtain confidence intervals for the parameters. In general, for a p parameter model of n data, a $(1 - 2\alpha)100\%$ confidence interval for the j^{th} parameter is

$$\hat{\beta}_j \pm t_{n-p}(\alpha)\hat{\sigma}_{\hat{\beta}_j},$$

where $t_{n-p}(\alpha)$ is the value below which a t_{n-p} random variable would lie with probability α .

As an example of its use, the following calculates a 95% confidence interval for the mean increase in count per cm^3 increase in `m.vol`.

```
> sc.c <- summary(sc1.mod1)$coefficients
> sc.c # check info extracted from summary
      Estimate Std. Error   t value Pr(>|t|)
(Intercept) -58.69444 121.619433 -0.4826075 0.64647664
m.vol        23.24653   7.117239  3.2662284 0.01711481
> sc.c[2,1]+qt(c(.025,.975),6)*sc.c[2,2]
[1]  5.831271 40.661784  # 95% CI
```

1.5.6 Prediction

It is possible to predict the expected value of the response at new values of the predictor variables using the `predict` function. For example: what are the model predicted counts for `m.vol` values of 10, 15, 20 and 25? The following obtains the answer along with associated standard errors (see section 1.3.6).

```
> df <- data.frame(m.vol=c(10,15,20,25))
> predict(sc1.mod1,df,se=TRUE)
$fit
    1         2         3         4
173.7708 290.0035 406.2361 522.4688

$se.fit
    1         2         3         4
60.39178 43.29247 51.32314 76.98471
```

The first line creates a data frame containing the predictor variable values at which predictions are required. The second line calls `predict` with the fitted model object and new data from which to predict. `se=TRUE` tells the function to return standard errors along with the predictions.

1.6 Practical modelling with factors

Most of the models covered so far have been for situations in which the predictor variables are continuous variables, but there are many situations in which the predictor variables are more qualitative in nature, and serve to divide the responses into groups. Examples might be eye- colour of subjects in a psychology experiment, which of three alternative hospitals were attended by patients in a drug trial, manufacturers of cars used in crash tests etc. Variables like these, which serve to classify the units on which the response variable has been measured into distinct categories, are known as *factor variables*, or simply *factors*. The different categories of the factor are known as *levels* of the factor. For example levels of the factor ‘eye colour’ might be ‘blue’, ‘brown’, ‘grey’ and ‘green’, so that we would refer to eye colour as a factor with four levels. Note that ‘levels’ is quite confusing terminology: there is not necessarily any natural ordering of the levels of a factor. Hence ‘levels’ of a factor might best be thought of as ‘categories’ of a factor, or ‘groups’ of a factor.

Factor variables are handled using dummy variables. Each factor variable can be replaced by as many dummy variables as there are levels of the factor — one for each level of the factor. For each response datum, only one of these dummy variables will be non- zero: the dummy variable for the single level that applies to that response. Consider an example to see how this works in practice: 9 laboratory rats are fed too much, so that they divide into 3 groups of 3: ‘fat’, ‘very fat’ and ‘enormous’. Their blood insulin levels are then measured 10 minutes after being fed a standard amount of sugar. The investigators are interested in the relationship between insulin levels and the factor ‘rat size’. Hence a model could be set up in which the predictor variable is the factor ‘rat size’, with the three levels ‘fat’, ‘very fat’ and ‘enormous’. Writing y_i for the i^{th} insulin level measurement, a suitable model might be:

$$\mathbb{E}(Y_i) \equiv \mu_i = \begin{cases} \beta_0 & \text{if rat is fat} \\ \beta_1 & \text{if rat is very fat} \\ \beta_2 & \text{if rat is enormous} \end{cases}$$

and this is easily written in linear model form, using a dummy predictor variable for each level of the factor:

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

A key difference between dummy variables and directly measured predictor variables, is that the dummy variables, and parameters associated with a factor, are almost always treated as a group during model selection — it does not usually make

sense for a subset of the dummy variables associated with a factor to be dropped or included on their own: either all are included or none. The F ratio tests derived in section 1.3.4 are designed for hypothesis testing in this situation.

1.6.1 Identifiability

When modelling with factor variables, model ‘identifiability’ becomes an important issue. It is quite easy to set up models, involving factors, in which it is impossible to estimate the parameters uniquely, because an infinite set of alternative parameter vectors would give rise to exactly the same expected value vector. A simple example illustrates the problem. Consider again the fat rat example, but suppose that we wanted to formulate the model in terms of an overall mean insulin level, α , and deviations from that level, β_j , associated with each level of the factor. The model would be something like:

$$\mu_i = \alpha + \beta_j \text{ if rat } i \text{ is rat size level } j$$

(where j is 0, 1 or 2, corresponding to ‘fat’, ‘very fat’ or ‘enormous’). The problem with this model is that there is not a one-to-one correspondence between the parameters and the fitted values, so that the parameters can not be uniquely estimated from the data. This is easy to see. Consider any particular set of α and β values, giving rise to a particular μ value: any constant c could be added to α and simultaneously subtracted from each element of β without changing the value of μ . Hence there is an infinite set of parameters giving rise to each μ value, and therefore the parameters can not be estimated uniquely. The model is not ‘identifiable’.

This situation can be diagnosed directly from the model matrix. Written out in full the example model is

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

But the columns of the model matrix are not independent, and this lack of full column rank means that the formulae for finding the least squares parameter estimates break down^{††}. Identifiable models have model matrices of full column rank, unidentifiable ones are column rank deficient.

The solution to the identifiability problem is to impose just enough linear constraints

^{††} In terms of section 1.3.1, \mathbf{R} will not be full rank, and will hence not be invertible; in terms of section 1.3.7, $\mathbf{X}^T \mathbf{X}$ will not be invertible.

on the model parameters, that the model becomes identifiable. For example, in the fat rat model, we could impose the constraint that

$$\sum_{j=0}^2 \beta_j = 0.$$

This would immediately remove the identifiability problem, but does require use of a linearly constrained least squares method (see sections 1.8.1 and 1.8.2). A simpler constraint is to set one of the unidentifiable parameters to zero, which requires only that the model is re-written, without the zeroed parameter, rather than a modified fitting method. For example, in the fat rat case, we could set α to zero, and recover the original identifiable model. This is perfectly legitimate, since the reduced model is capable of reproducing any expected values that the original model could produce.

In the one factor case this discussion of identifiability may seem to be un-necessarily complicated, since it is so easy to write down the model directly, in an identifiable form. However, when models involve more than one factor variable, the issue can not be avoided. For a more general treatment of identifiability constraints see sections 1.8.1 and 1.8.2.

1.6.2 Multiple factors

It is frequently the case that more than one factor variable should be included in a model, and this is straightforward to do. Continuing the fat rat example, it might be that the sex of the rats is also a factor in insulin production, and that the appropriate model is:

$$\mu_i = \alpha + \beta_j + \gamma_k \text{ if rat } i \text{ is rat size level } j \text{ and sex } k$$

where k is 0 or 1 for male or female. Written out in full (assuming the rats are MMFFFMFMM) the model is

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \\ \gamma_0 \\ \gamma_1 \end{bmatrix}.$$

It is immediately obvious that the model matrix is of column rank 4, implying that two constraints are required to make the model identifiable. You can see the lack of column independence by noting that column 5 is column 1 minus column 6, while column 2 is column 1 minus columns 3 and 4. An obvious pair of constraints would

be to set $\beta_0 = \gamma_0 = 0$, so that the full model is

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \gamma_1 \end{bmatrix}.$$

When you specify models involving factors in R, it will automatically impose identifiability constraints for you, and by default these constraints will be that the parameter for the ‘first’ level of each factor is zero (‘first’ is essentially arbitrary here — the order of levels of a factor is not important).

1.6.3 ‘Interactions’ of factors

In the examples considered so far, the effect of factor variables has been purely additive, but it is possible that a response variable may react differently to the combination of two factors, than would be predicted purely by the effect of the two factors separately. For example, if examining patient blood cholesterol levels, we might consider the factors ‘hypertensive’ (yes/no) and ‘diabetic’ (yes/no). Being hypertensive or diabetic would be expected to raise cholesterol levels, but being both is likely to raise cholesterol levels much more than would be predicted from just adding up the apparent effects when only one condition is present. When the effect of two factor variables together differs from the sum of their separate effects, then they are said to *interact*, and an adequate model in such situations requires *interaction terms*. Put another way, if the effects of one factor change in response to another factor, then the factors are said to interact.

Let us continue with the fat rat example, but now suppose that how insulin level depends on size varies with sex. An appropriate model is then

$$\mu_i = \alpha + \beta_j + \gamma_k + \delta_{jk} \text{ if rat } i \text{ is rat size level } j \text{ and sex } k,$$

where the δ_{jk} terms are the parameters for the interaction of rat size and sex. Writing

this model out in full it is clear that it is spectacularly unidentifiable:

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_0 \\ \beta_1 \\ \beta_2 \\ \gamma_0 \\ \gamma_1 \\ \delta_{00} \\ \delta_{01} \\ \delta_{10} \\ \delta_{11} \\ \delta_{20} \\ \delta_{21} \end{bmatrix}.$$

In fact, for this simple example, with rather few rats, we now have more parameters than data. There are of course many ways of constraining this model to achieve identifiability. One possibility (the default in R) is to set $\beta_0 = \gamma_0 = \delta_{00} = \delta_{01} = \delta_{10} = \delta_{20} = 0$. The resulting model can still produce any fitted value vector that the full model can produce, but all the columns of its model matrix are independent, so that the model is identifiable:

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \\ \mu_8 \\ \mu_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \gamma_1 \\ \delta_{11} \\ \delta_{21} \end{bmatrix}.$$

Of course, the more factor variables are present, the more interactions are possible, and the higher the order of the possible interactions: for example if three factors are present then each factor could interact with each factor, giving three possible ‘two-way’ interactions, while all the factors could also interact together, giving a three-way interaction (e.g. the way in which insulin levels dependence on rat size is influenced by sex is itself influenced by blood group — perhaps with interactions beyond two-way, equations are clearer than words.)

1.6.4 Using factor variables in R

It is very easy to work with factor variables in R. All that is required is that you let R know that a particular variable is a factor variable. For example, suppose z is a variable declared as follows:

```
> z<-c(1,1,1,2,2,1,3,3,3,4)
```

```
> z
[1] 1 1 1 2 2 1 3 3 3 3 4
```

and it is to be treated as a factor with 4 levels. The function `as.factor()` will ensure that `z` is treated as a factor:

```
> z<-as.factor(z)
> z
[1] 1 1 1 2 2 1 3 3 3 3 4
Levels: 1 2 3 4
```

Notice that, when a factor variable is printed, a list of its levels is also printed — this provides an easy way to tell if a variable is a factor variable. Note also that the digits of `z` are treated purely as labels: the numbers 1 to 4 could have been any labels. For example, `x` could be a factor with 3 levels, declared as follows:

```
> x<-c("A", "A", "C", "C", "C", "er", "er")
> x
[1] "A"   "A"   "C"   "C"   "C"   "er"  "er"
> x<-as.factor(x)
> x
[1] A   A   C   C   C   er  er
Levels: A C er
```

Once a variable is declared as a factor variable, then R can process it automatically within model formulae, by replacing it with the appropriate number of binary dummy variables (and imposing any necessary identifiability constraints on the specified model).

As an example of the use of factor variables consider the `PlantGrowth` data frame supplied with R. These are data on the growth of plants under control conditions and two different treatment conditions. The factor `group` has three levels `ctrl`, `trt1` and `trt2`, and it is believed that the growth of the plants depends on this factor. First check that `group` is already a factor variable:

```
> PlantGrowth$group
[1] ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl trt1
[12] trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt2 trt2
[23] trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2
Levels: ctrl trt1 trt2
```

... since a list of levels is reported, it must be. If it had not been then

```
PlantGrowth$group <- as.factor(PlantGrowth$group)
```

would have converted it. The response variable for these data is `weight` of the plants at some set time after planting, and the aim is to investigate whether the `group` factor controls this, and if so, to what extent.

```
> pgm.1 <- lm(weight ~ group, data=PlantGrowth)
> plot(pgm.1)
```

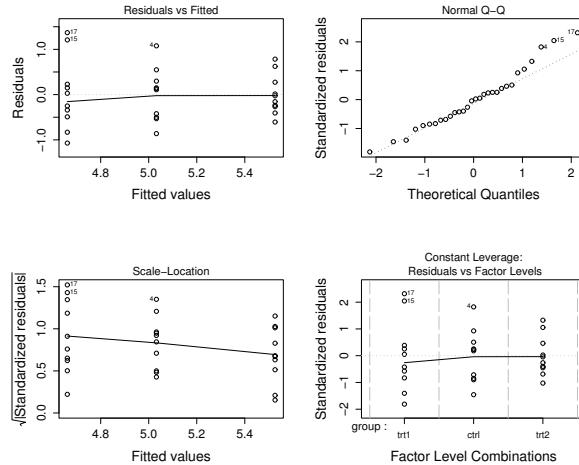


Figure 1.10 *Model checking plots for the plant growth example. Note that, since the leverages are all the same, in this case, the lower right plot is now simplified.*

As usual, the first thing to do, after fitting a model, is to check the residual plots shown in figure 1.10. In this case there is some suggestion of decreasing variance with increasing mean, but the effect does not look very pronounced, so it is probably safe to proceed.

```
> summary(pgm.1)
[edited]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.0320    0.1971 25.527 <2e-16 ***
grouptrt1   -0.3710    0.2788 -1.331  0.1944
grouptrt2    0.4940    0.2788  1.772  0.0877 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6234 on 27 degrees of freedom
Multiple R-Squared:  0.2641,    Adjusted R-squared:  0.2096
F-statistic: 4.846 on 2 and 27 DF,  p-value: 0.01591
```

Notice how R reports an intercept parameter and parameters for the two treatment levels, but, in order to obtain an identifiable model, it has not included a parameter for the control level of the group factor. So the estimated overall mean weight (in the population that these plants represent, given control conditions) is 5.032, while treatment 1 is estimated to lower this weight by 0.37, and treatment 2 to increase it by 0.49. However, neither parameter individually appears to be significantly different

from zero. (Don't forget that `model.matrix(pgm.1)` can be used to check up on the form of the model matrix used in the fit.)

Model selection based on the summary output is very difficult for models containing factors. It makes little sense to drop the dummy variable for just one level of a factor from a model, and if we did, what would we then do about the model identifiability constraints? Usually, it is only of interest to test whether the whole factor variable should be in the model or not, and this amounts to testing whether all its associated parameters are simultaneously zero or not. The F-ratio tests derived in section 1.3.4 are designed for just this purpose, and in R, such tests can be invoked using the `anova` function. For example, we would compare `pgm.1` to a model in which the expected response is given by a single parameter that does not depend on `group`:

```
> pgm.0<-lm(weight~1,data=PlantGrowth)
> anova(pgm.0,pgm.1)
Analysis of Variance Table

Model 1: weight ~ 1
Model 2: weight ~ group
  Res.Df   RSS Df Sum of Sq    F   Pr(>F)
1     29 14.2584
2     27 10.4921  2     3.7663 4.8461 0.01591 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output here gives the F-ratio statistic used to test the null hypothesis that the simpler model is correct, against the alternative that `pgm.1` is correct. If the null is true then the probability of obtaining so large an F value is only 0.016, suggesting that the null hypothesis is not very plausible.

So we see that the data provide evidence for an effect of the `group` factor variable on `weight`, which appeared marginal or absent when we examined p-values for the individual model parameters. This comes about because we have, in effect, considered all the parameters associated with the factor simultaneously, thereby obtaining a more powerful test than any of the single parameter tests could be. In the light of this analysis, the most promising treatment to look at is clearly treatment 2, since this gives the largest and 'most significant' effect and it is a positive effect.

1.7 General linear model specification in R

Having seen several examples of the use of `lm`, it is worth briefly reviewing the way in which models are specified using model formulae in R. The main components of a formula are all present in the following example

`y ~ a*b + x:z -1`

Note the following:

- \sim separates the response variable, on the left, from the ‘linear predictor’, on the right. So in the example y is the response and a, b, x and z are the predictors.
- $+$ indicates that the response depends on what is to the left of $+$ *and* what is to the right of it. Hence within formulae ‘ $+$ ’ should be thought of as ‘*and*’ rather than ‘the sum of’.
- $:$ indicates that the response depends on the *interaction* of the variables to the left and right of $:$. Interactions are obtained by forming the element-wise products of all model matrix columns corresponding to the two terms that are interacting and appending the resulting columns to the model matrix (although, of course, some identifiability constraints may be required).
- $*$ means that the response depends on whatever is to the left of $*$ and whatever is to the right of it *and* their interaction. i.e. $a*b$ is just a shorter way of writing $a + b + a:b$.
- -1 means that the default intercept term should not be included in the model. Note that, for models involving factor variables, this often has no real impact on the model structure, but simply reduces the number of identifiability constraints by one, while changing the interpretation of some parameters.

Because of the way that some symbols change their usual meaning in model formulae, it is necessary to take special measures if the usual meaning is to be restored to arithmetic operations within a formula. This is accomplished by using the identity function $I()$ which simply evaluates its argument and returns it. For example, if we wanted to fit the model:

$$y_i = \beta_0 + \beta_1(x_i + z_i) + \beta_2 v_i + \epsilon_i$$

then we could do so using the model formula

$$y \sim I(x+z) + v$$

Note that there is no need to ‘protect’ arithmetic operations within arguments to other functions in this way. For example

$$y_i = \beta_0 + \beta_1 \log(x_i + z_i) + \beta_2 v_i + \epsilon_i$$

would be fitted correctly by

$$y \sim \log(x+z) + v$$

1.8 Further linear modelling theory

This section covers linear models with constraints on the parameters (including a discussion of contrasts), the connection with maximum likelihood estimation, AIC and Mallows C_p , linear models for non-independent data with non-constant variance and non-linear models.

1.8.1 Constraints I: general linear constraints

It is often necessary, particularly when working with factor variables, to impose constraints on the linear model parameters, of the general form

$$\mathbf{C}\boldsymbol{\beta} = \mathbf{0},$$

where \mathbf{C} is an $m \times p$ matrix of known coefficients. The general approach to imposing such constraints is to rewrite the model in terms of $p - m$ unconstrained parameters. There are a number of ways of doing this, but a simple general approach uses the QR decomposition of \mathbf{C}^\top . Let

$$\mathbf{C}^\top = \mathbf{U} \begin{bmatrix} \mathbf{P} \\ \mathbf{0} \end{bmatrix}$$

where \mathbf{U} is a $p \times p$ orthogonal matrix and \mathbf{P} is an $m \times m$ upper triangular matrix. \mathbf{U} can be partitioned $\mathbf{U} \equiv (\mathbf{D} : \mathbf{Z})$ where \mathbf{Z} is a $p \times (p - m)$ matrix.

It turns out that

$$\boldsymbol{\beta} = \mathbf{Z}\boldsymbol{\beta}_z$$

will meet the constraints for any value of the $p - m$ dimensional vector $\boldsymbol{\beta}_z$. This is easy to see:

$$\mathbf{C}\boldsymbol{\beta} = [\mathbf{P}^\top \quad \mathbf{0}] \begin{bmatrix} \mathbf{D}^\top \\ \mathbf{Z}^\top \end{bmatrix} \mathbf{Z}\boldsymbol{\beta}_z = [\mathbf{P}^\top \quad \mathbf{0}] \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{p-m} \end{bmatrix} \boldsymbol{\beta}_z = \mathbf{0}.$$

Hence to minimize $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$ w.r.t. $\boldsymbol{\beta}$, subject to $\mathbf{C}\boldsymbol{\beta} = \mathbf{0}$, the following algorithm can be used.

1. Find the QR decomposition of \mathbf{C}^\top : the final $p - m$ columns of the orthogonal factor define \mathbf{Z} .
2. Minimize the unconstrained sum of squares $\|\mathbf{y} - \mathbf{X}\mathbf{Z}\boldsymbol{\beta}_z\|^2$ w.r.t. $\boldsymbol{\beta}_z$ to obtain $\hat{\boldsymbol{\beta}}_z$.
3. $\hat{\boldsymbol{\beta}} = \mathbf{Z}\hat{\boldsymbol{\beta}}_z$.

Note that, in practice, it is computationally inefficient to form \mathbf{Z} explicitly, when we only need to be able to post-multiply \mathbf{X} by it, and pre-multiply $\boldsymbol{\beta}_z$ by it. The reason for this is that \mathbf{Z} is completely defined as the product of m ‘Householder rotations’: simple matrix operations that can be applied very rapidly to any vector or matrix. R includes routines for multiplication by the orthogonal factor of a QR decomposition which makes use of these efficiencies. See A.5 and A.6 for further details on QR decomposition.

1.8.2 Constraints II: ‘contrasts’ and factor variables

There is another approach to imposing identifiability constraints on models involving factor variables. To explain it, it is worth revisiting the basic identifiability problem with a simple example. Consider the model

$$y_i = \mu + \alpha_j + \epsilon_i \text{ if } y_i \text{ is from group } j$$

and suppose that there are three groups with two observations in each. The model matrix in this case is

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix},$$

but this is not of full column rank: any of its columns could be made up from a linear combination of the other 3. In geometric terms the model space is of dimension 3 and not 4, and this means that we can not estimate all 4 model parameters, but at most 3 parameters. Numerically this problem would manifest itself in the rank deficiency of \mathbf{R} in equation (1.6), which implies that \mathbf{R}^{-1} does not exist.

One approach to this issue is to remove one of the model matrix columns, implicitly treating the corresponding parameter as zero. This gives the model matrix full column rank, so that the remaining parameters are estimable, but since the model space is unaltered, we have not fundamentally changed the model. It can still predict every set of fitted values that the original model could predict. By default \mathbf{R} drops the model matrix column corresponding to the first level of each factor, in order to impose identifiability on models with factors. For the simple example this results in

$$\mathbf{X}' = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

To generalize this approach, it helps to write out this deletion in a rather general way. For example, if we re-write the original model matrix in partitioned form, $\mathbf{X} = [\mathbf{1} : \mathbf{X}_1]$, where $\mathbf{1}$ is a column of 1s, then

$$\mathbf{X}' = [\mathbf{1} : \mathbf{X}_1 \mathbf{C}_1] \text{ where } \mathbf{C}_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Now all that \mathbf{C}_1 has done is to replace the 3 columns of \mathbf{X}_1 by a 2 column linear combination of them, which cannot be combined to give $\mathbf{1}$. On reflection, *any* matrix which did these two things would have served as well as the particular \mathbf{C}_1 actually given, in terms of making the model identifiable. This observation leads to the idea of choosing alternative matrix elements for \mathbf{C}_1 , in order to enhance the interpretability of the parameters actually estimated, for some models.

Matrices like \mathbf{C}_1 are known as *contrast matrices*[†] and several different types are available in \mathbf{R} . The degree of interpretability of some of the contrasts is open to

[†] Actually, in much of the literature on linear models the given \mathbf{C}_1 would not be called a ‘contrast’, as its columns are not orthogonal to $\mathbf{1}$, but \mathbf{R} makes no terminological distinction.

debate. For the \mathbf{C}_1 given, suppose that parameters μ , α'_2 and α'_3 are estimated: μ would now be interpretable as the mean for group 1, while α'_2 and α'_3 would be the differences between the means for groups 2 and 3, and the mean for group 1.

With all contrast matrices, the contrast matrix itself can be used to transform back from the parameters actually estimated, to estimates in the original redundant parameterization. e.g.

$$\hat{\boldsymbol{\alpha}} = \mathbf{C}_1 \hat{\boldsymbol{\alpha}}'.$$

The contrast approach generalizes easily to models with multiple factors and their interactions. Again a simple example makes things clear. Consider the model:

$$y_i = \mu + \alpha_j + \beta_k + \gamma_{jk} + \epsilon_i \text{ if } y_i \text{ from groups } j \text{ and } k$$

The unconstrained model matrix for this might have the form $\mathbf{X} = [\mathbf{1} : \mathbf{X}_\alpha : \mathbf{X}_\beta : \mathbf{X}_\alpha \cdot \mathbf{X}_\beta]$, where \mathbf{X}_α and \mathbf{X}_β are the columns generated by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and $\mathbf{X}_\alpha \cdot \mathbf{X}_\beta$ are the columns corresponding to $\boldsymbol{\gamma}$, which are in fact generated by element wise multiplication of all possible pairings of the column from \mathbf{X}_α and \mathbf{X}_β .

To make this model identifiable we would choose contrast matrices \mathbf{C}_α and \mathbf{C}_β for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively, and then form the following identifiable model matrix:

$$\mathbf{X}' = [\mathbf{1} : \mathbf{X}_\alpha \mathbf{C}_\alpha : \mathbf{X}_\beta \mathbf{C}_\beta : (\mathbf{X}_\alpha \mathbf{C}_\alpha) \cdot (\mathbf{X}_\beta \mathbf{C}_\beta)]$$

(in this case $\boldsymbol{\gamma} = \mathbf{C}_\alpha \otimes \mathbf{C}_\beta \boldsymbol{\gamma}'$, where \otimes is the Kronecker product). Some further information can be found in Venables and Ripley (2004).

1.8.3 Likelihood

The theory developed in section 1.3 is quite sufficient to justify the approach of estimating linear models by least squares, but although it doesn't directly strengthen the case, it is worth understanding the link between the method of least squares and the method of maximum likelihood, for normally distributed data.

The basic idea of likelihood is that, given some parameter values, a statistical model allows us to write down the probability, or probability density, of any set of data, and in particular of the set actually observed. In some sense, parameter values which cause the model to suggest that the observed data are probable, are more ‘likely’ than parameter values that suggest that what was observed was improbable. In fact it seems reasonable to use as estimates of the parameters, those values which maximize the probability of the data according to the model: these are the ‘maximum likelihood estimates’ of the parameters.

In the next chapter the properties of maximum likelihood estimation will be covered in greater detail. For the moment consider the likelihood for the parameters of a linear model. According to the model, the joint p.d.f. of the response data is

$$f_{\boldsymbol{\beta}, \sigma^2}(\mathbf{y}) = (2\pi\sigma^2)^{-n/2} e^{-\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 / (2\sigma^2)}.$$

Now suppose that the observed data are plugged into this expression and it is treated as a function of its parameters β and σ^2 . This is known as the likelihood function

$$L(\beta, \sigma^2) = (2\pi\sigma^2)^{-n/2} e^{-\|\mathbf{y} - \mathbf{X}\beta\|^2/(2\sigma^2)},$$

and it is important to note that \mathbf{y} is now representing the actual observed data, rather than arguments of a p.d.f. To estimate the parameters, L should be maximized w.r.t. them, and it is immediately apparent that the value of β maximizing L will be the value minimizing

$$\mathcal{S} = \|\mathbf{y} - \mathbf{X}\beta\|^2$$

(irrespective of the value of σ^2 or its MLE).

In itself this connection is of little interest, but it suggests how to estimate linear models when data do not meet the constant variance assumption, and may not even be independent. To this end consider the linear model

$$\mu = \mathbf{X}\beta, \quad \mathbf{y} \sim N(\mu, \mathbf{V}\sigma^2)$$

where \mathbf{V} is any positive definite[†] matrix. In this case the likelihood for β is

$$L(\beta) = \frac{1}{\sqrt{(2\pi\sigma^2)^n |\mathbf{V}|}} e^{-(\mathbf{y} - \mathbf{X}\beta)^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\beta)/(2\sigma^2)}$$

and if \mathbf{V} is known then maximum likelihood estimation of the β is achieved by minimizing

$$\mathcal{S}_v = (\mathbf{y} - \mathbf{X}\beta)^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\beta).$$

In fact the likelihood approach can be taken further, since if \mathbf{V} depends on unknown parameters then these too can be estimated by maximum likelihood estimation: this is what is done in linear mixed modelling, which is discussed in Chapter 6.

1.8.4 Non-independent data with variable variance

In the previous section a modified least squares criterion was developed for linear model parameter estimation, when data follow a general multivariate normal distribution, with unknown mean and covariance matrix known to within a constant of proportionality. It turns out to be possible to transform this fitting problem so that it has exactly the form of the fitting problem for independent data with constant variance. Having done this, all inference about the model parameters can proceed using the methods of section 1.3.

First let \mathbf{L} be any matrix such that $\mathbf{L}^T \mathbf{L} = \mathbf{V}$: a Choleski decomposition is usually

[†] A matrix \mathbf{A} is positive definite if $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$ for any non zero vector \mathbf{x} . Equivalent to this condition is the condition that all the eigenvalues of \mathbf{A} must be strictly positive. Practical tests for positive definiteness are examination of the eigenvalues of the matrix, or (more efficiently) seeing if a Choleski decomposition of the matrix is possible (this must be performed without pivoting, otherwise only positive semi-definiteness is tested): see A.7.

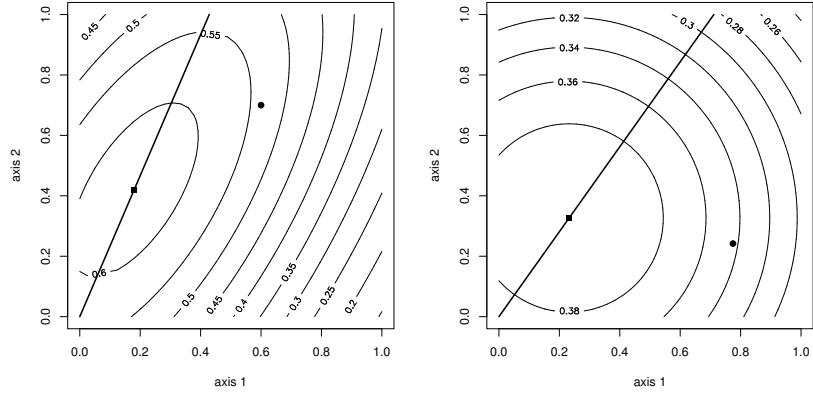


Figure 1.11 *Fitting a linear model to data that are not independent/constant-variance. The example is a straight line through the origin fit to two data. In both panels the response data values give the co-ordinates of the point • and the straight line is the vector defining the ‘model subspace’ (the line along which the fitted values could lie). It is assumed that the data arise from the multivariate normal distribution contoured in the left panel. The right panel shows the fitting problem after transformation in the manner described in section 1.8.4: the response data and model subspace have been transformed and this implies a transformation of the distribution of the response. The transformed data are an observation from a radially symmetric multivariate normal density.*

the easiest way to obtain this (see section A.7). Then

$$\begin{aligned}\mathcal{S}_v &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{L}^{-1} \mathbf{L}^{-\top} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \|\mathbf{L}^{-\top} \mathbf{y} - \mathbf{L}^{-\top} \mathbf{X}\boldsymbol{\beta}\|^2,\end{aligned}$$

and this least squares objective can be minimized by the methods already met (i.e. form a QR decomposition of $\mathbf{L}^{-\top} \mathbf{X}$ etc.)

It is not just the model fitting that carries over from the theory of section 1.3. Since $\mathbf{L}^{-\top} \mathbf{y}$ is a linear transformation of a normal random vector, it must have a multivariate normal distribution, and it is easily seen that $\mathbb{E}(\mathbf{L}^{-\top} \mathbf{y}) = \mathbf{L}^{-\top} \mathbf{X}\boldsymbol{\beta}$ while the covariance matrix of $\mathbf{L}^{-\top} \mathbf{y}$ is

$$\mathbf{V}_{\mathbf{L}^{-\top} \mathbf{y}} = \mathbf{L}^{-\top} \mathbf{V} \mathbf{L}^{-1} \sigma^2 = \mathbf{L}^{-\top} \mathbf{L}^\top \mathbf{L} \mathbf{L}^{-1} = \mathbf{I} \sigma^2$$

i.e. $\mathbf{y} \sim N(\mathbf{L}^{-\top} \mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma^2)$. In other words, the transformation has resulted in a new linear modelling problem, in which the response data are independent normal random variables with constant variance: exactly the situation which allows all the results from section 1.3 to be used for inference about $\boldsymbol{\beta}$.

Figure 1.11 illustrates the geometry of the transformation for a simple linear model,

$$y_i = \beta x_i + \epsilon_i, \quad \epsilon_i \sim N(\mathbf{0}, \mathbf{V}),$$

where $\mathbf{y}^T = (.6, .7)$, $\mathbf{x}^T = (.3, .7)$, $\beta = .6$ and $\mathbf{V} = \begin{bmatrix} .6 & .5 \\ .5 & 1.1 \end{bmatrix}$. The left panel shows the geometry of the original fitting problem, while the right panel shows the geometry of the transformed fitting problem.

1.8.5 AIC and Mallow's statistic,

Consider again the problem of selecting between nested models, which is usually equivalent to deciding whether some terms from the model should simply be set to zero. The most natural way to do this would be to select the model with the smallest residual sum of squares or largest likelihood, but this is always the largest model under consideration. Model selection methods based on F-ratio or t-tests address this problem by selecting the simplest model consistent with the data, where consistency is judged using some significance level (threshold p-value), that has to be chosen more or less arbitrarily.

In this section an alternative approach is developed, based on the idea of trying to select the model that should do the best job of predicting $\mu \equiv \mathbb{E}(\mathbf{y})$, rather than the model that gets as close as possible to \mathbf{y} . This can be approached from a least squares or likelihood perspective, with the latter being slightly more satisfactory in the handling of the parameter σ^2 .

As discussed in section 1.8.3, a linear model is estimated by finding the β_i values maximizing

$$f(\boldsymbol{\beta}, \sigma^2) = \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp \left[-\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 / (2\sigma^2) \right] \quad (1.12)$$

and generally, the more parameters we allow the model, the more closely it will fit \mathbf{y} , even if that means fitting the noise component of \mathbf{y} . This over-fitting problem would be solved if we could find the model maximizing

$$K(\hat{\boldsymbol{\beta}}, \sigma^2) = \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp \left[-\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 / (2\sigma^2) \right]. \quad (1.13)$$

Consider adding terms to the model, so that at some point in the sequence of models the ‘true’ model appears. If we really knew $\boldsymbol{\mu}$ then (1.13) would reach a maximum once the model was just large enough to correctly represent $\boldsymbol{\mu}$, and would then stay at that maximum if further (redundant) terms were added. If the model had in fact been estimated by maximizing (1.12) then we would still expect (1.13) to be maximized once the true model is reached, but to decline thereafter, as the model predictions move away from $\boldsymbol{\mu}$, in order to fit the noise component of \mathbf{y} increasingly well.

Clearly (1.13) can not be used directly since $\boldsymbol{\mu}$ is unknown, but it can be estimated.

First consider the norm in (1.13):

$$\begin{aligned}\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 &= \|\boldsymbol{\mu} - \mathbf{A}\mathbf{y}\|^2 = \|\mathbf{y} - \mathbf{A}\mathbf{y} - \boldsymbol{\epsilon}\|^2 \\ &= \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2 + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} - 2\boldsymbol{\epsilon}^T(\mathbf{y} - \mathbf{A}\mathbf{y}) \\ &= \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2 + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} - 2\boldsymbol{\epsilon}^T(\boldsymbol{\mu} + \boldsymbol{\epsilon}) + 2\boldsymbol{\epsilon}^T \mathbf{A}(\boldsymbol{\mu} + \boldsymbol{\epsilon}) \\ &= \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2 - \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} - 2\boldsymbol{\epsilon}^T \boldsymbol{\mu} + 2\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\mu} + 2\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\epsilon}.\end{aligned}$$

Now, $\mathbb{E}(\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}) = \mathbb{E}(\sum_i \epsilon_i^2) = n\sigma^2$, $\mathbb{E}(\boldsymbol{\epsilon}^T \boldsymbol{\mu}) = \mathbb{E}(\boldsymbol{\epsilon}^T) \boldsymbol{\mu} = 0$ and $\mathbb{E}(\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\mu}) = \mathbb{E}(\boldsymbol{\epsilon}^T) \mathbf{A} \boldsymbol{\mu} = 0$. The final term is slightly trickier, but using the fact that a scalar is its own trace:

$$\mathbb{E}[\text{tr}(\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\epsilon})] = \mathbb{E}[\text{tr}(\mathbf{A} \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T)] = \text{tr}(\mathbf{A} \mathbb{E}[\boldsymbol{\epsilon} \boldsymbol{\epsilon}^T]) = \text{tr}(\mathbf{A} \mathbf{I}) \sigma^2 = \text{tr}(\mathbf{A}) \sigma^2.$$

Hence

$$\mathbb{E}(\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2) = \mathbb{E}(\|\boldsymbol{\mu} - \mathbf{A}\mathbf{y}\|^2) = \mathbb{E}(\|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2) - n\sigma^2 + 2\text{tr}(\mathbf{A}) \sigma^2, \quad (1.14)$$

which can be estimated by

$$\widehat{\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2} = \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2 - n\sigma^2 + 2\text{tr}(\mathbf{A}) \sigma^2. \quad (1.15)$$

Hence an appropriate estimate of $\kappa = \log K$ is

$$\tilde{\kappa} = -n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2 + n/2 - \text{tr}(\mathbf{A}).$$

i.e. if l is the log likelihood of the model then

$$\tilde{\kappa} = l(\hat{\boldsymbol{\beta}}, \sigma^2) - p + n/2$$

where $p = \text{tr}(\mathbf{A})$ is the number of identifiable model parameters. This will be maximized by whatever model minimizes

$$\kappa = 2(-l(\hat{\boldsymbol{\beta}}, \sigma^2) + p)$$

which is known as Akaike's information criteria (Akaike, 1973) or AIC (the factor of -2 is conventional, for reasons that will become apparent in the next chapter).

The above derivation assumes that σ^2 is known, whereas, in fact, it is usually estimated. If we use the MLE, $\hat{\sigma}^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2/n$, then the AIC criteria becomes

$$\kappa = 2(-l(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2) + p) + 2$$

where the extra 2 serves to penalize the extra free parameter in the model: justification of this is postponed until section 2.4.7, where AIC is derived for any model estimated by likelihood maximization.

Notice how the above derivation does not involve any assumption that directly implies that the models to be compared must be nested: however a more detailed examination of the comparison of models by AIC would suggest that the comparison will be more reliable for nested models, since in that case some of the neglected terms in the approximation of κ cancel. Notice also that if the true model is not in the set of models under consideration then the properties of (1.13) will be less ideal. Indeed

in this case (1.13) would itself tend to favour more complex models, since it would be impossible to match μ exactly: as sample size increases and estimates become more precise, this tendency starts to overcome the negative effects of overfitting and leads to more complex models being selected. This tendency for AIC to favour more complex models with increasing sample size is often seen in practice: presumably because the true model is rarely in the set of candidate models.

If we were to start from a least squares perspective then we could simply try to estimate $\|\mu - X\hat{\beta}\|^2/\sigma^2$. Re-using the derivations given above, this results in an estimate known as Mallow's C_p (Mallows, 1973)

$$C_p = \|y - X\hat{\beta}\|^2/\sigma^2 + 2p - n.$$

Model selection by C_p minimization, works well if σ^2 is known, but for most models σ^2 must be estimated, and using the estimate derived from the model fit has the unfortunate consequence that C_p ceases to depend on which model has been fitted, in any meaningful way. To avoid this, σ^2 is usually fixed at the estimate given by the fit of the largest candidate model (unless it really is known, of course).

1.8.6 Non-linear least squares

Some non-linear models can be estimated by iterative approximation by a linear model. At each iterate the fitted approximating linear model suggests improved parameter estimates, and at convergence the parameter estimates are least squares estimates. In addition, the approximating linear model at convergence can be used as the basis for approximate inference about the parameters.

Formally, consider fitting the model:

$$\mathbb{E}(y) \equiv \mu = f(\beta)$$

to response data y , when f is a non-linear vector valued function of β . An obvious fitting objective is:

$$\mathcal{S} = \sum_{i=1}^n \{y_i - f_i(\beta)\}^2 = \|y - f(\beta)\|^2$$

and if the functions, f_i , are sufficiently well behaved then this non-linear least squares problem can be solved by iterative linear least squares. To do this, we start from a guess at the best fit parameters, $\hat{\beta}^{[k]}$, and then take a first order Taylor expansion of f_i around $\hat{\beta}^{[k]}$ so that the fitting objective becomes

$$\mathcal{S} \approx \mathcal{S}^{[k]} = \|y - f(\hat{\beta}^{[k]}) + J^{[k]}\hat{\beta}^{[k]} - J^{[k]}\beta\|^2$$

where $J^{[k]}$ is the ‘Jacobian’ matrix such that $J_{ij}^{[k]} = \partial f_i / \partial \beta_j$ (derivatives evaluated at $\hat{\beta}^{[k]}$, of course). Defining a vector of *pseudodata*,

$$z^{[k]} = y - f(\hat{\beta}^{[k]}) + J^{[k]}\hat{\beta}^{[k]},$$

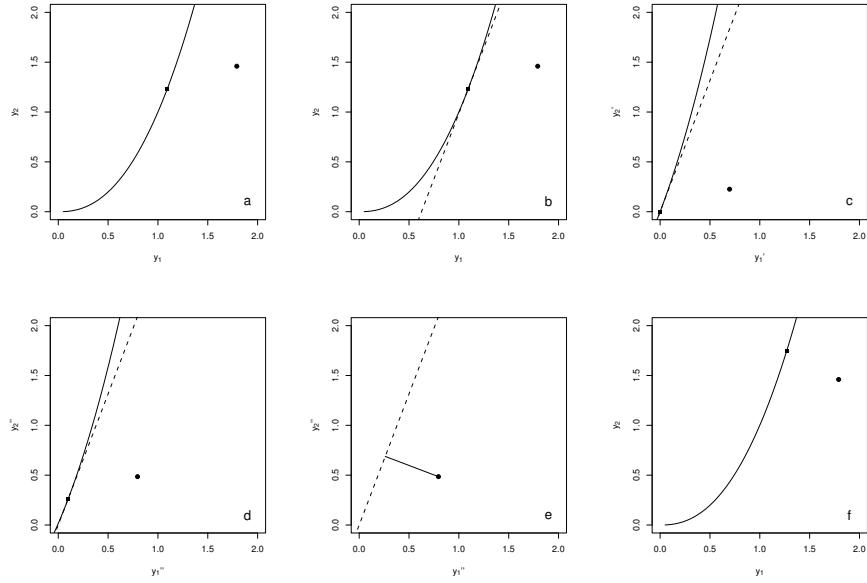


Figure 1.12 *Geometry of a single iteration of the Gauss Newton approach to non-linear model fitting. The example is fitting the model $\mathbb{E}(y_i) = \exp(\beta x_i)$ to x_i, y_i data ($i = 1, 2$). (a) plots y_2 against y_1 (\bullet) with the curve illustrating the possible values for the expected values of the response variable: as the value of β changes $\mathbb{E}(y)$ follows this curve - the ‘model manifold’. An initial guess at the parameter gives the fitted values plotted as ■. (b) The tangent space to the model manifold is found and illustrated by the dashed line. (c) The model, tangent and data are linearly translated so that the current estimate of the fitted values is at the origin. (d) The model, tangent and data are linearly translated so that $\mathbf{J}\hat{\beta}^{[k]}$ gives the location of the current estimate of the fitted values. (e) $\hat{\beta}^{[k+1]}$ is estimated by finding the closest point in the tangent space to the response data, by least squares. (f) shows the original model and data, with the next estimate of the fitted values, obtained using $\hat{\beta}^{[k+1]}$. The steps can now be repeated until the estimates converge.*

the objective can be re-written

$$S^{[k]} = \|\mathbf{z}^{[k]} - \mathbf{J}^{[k]}\beta\|^2,$$

and, since this is a linear least squares problem, it can be minimized with respect to β to obtain an improved estimated parameter vector $\hat{\beta}^{[k+1]}$. If $f(\beta)$ is not too non-linear then this process can be iterated until the $\hat{\beta}^{[k]}$ sequence converges, to the final least squares estimate $\hat{\beta}$.

Figure 1.12 illustrates the geometrical interpretation of this method. Because the non-linear model is being approximated by a linear model, large parts of the theory of linear models carry over as approximations in the current context. For example, under

the assumption of equal variance, σ^2 , and independence of the response variable, the covariance matrix of the parameters is simply: $(\mathbf{J}^\top \mathbf{J})^{-1}\sigma^2$, where \mathbf{J} is evaluated at convergence.

If \mathbf{f} is sufficiently non linear that convergence does not occur, then a simple ‘step reduction’ approach will stabilize the fitting. The vector $\Delta = \hat{\beta}^{[k+1]} - \hat{\beta}^{[k]}$ is treated as a trial step. If $\hat{\beta}^{[k+1]}$, does not decrease \mathcal{S} , then trial steps $\hat{\beta}^{[k]} + \alpha\Delta$ are taken, with ever decreasing α , until \mathcal{S} does decrease (of course, $0 < \alpha < 1$). Geometrically, this is equivalent to performing an updating step by fitting to the average $\mathbf{y}\alpha + (1 - \alpha)\mathbf{f}(\beta^{[k]})$, rather than original data \mathbf{y} : viewed in this way it is clear that for small enough α , each iteration must decrease \mathcal{S} , until a minimum is reached. It is usual to halve α each time that a step is unsuccessful, and to start each iteration with twice the α value which finally succeeded at the previous step (or $\alpha = 1$ if this is less). If it is necessary to set $\alpha < 1$ at the final step of the iteration then it is likely that inference based on the final approximating linear model will be somewhat unreliable.

1.8.7 Further reading

The literature on linear models is rather large, and there are many book length treatments. For a good introduction to linear models with R, see Faraway (2004). Other sources on the linear modelling functionality in R are Chambers (1993) and Venables and Ripley (2003). Numerical estimation of linear models is covered in Golub and van Loan (1996, chapter 5). Dobson (2001), McCullagh and Nelder (1989) and Davison (2003) also consider linear models as part of broader treatments. For R itself see R Development Core Team (2005).

1.9 Exercises

1. 4 car journeys in London of length 1, 3, 4 and 5 kilometres took 0.1, 0.4, 0.5 and 0.6 hours respectively. Find a least squares estimate of the mean journey speed.
2. Given n observations x_i, y_i , find the least squares estimate of β in the linear model: $y_i = \mu_i + \epsilon_i$, $\mu_i = \beta$.
3. Which, if any, of the following common linear model assumptions are required for $\hat{\beta}$ to be unbiased: (i) The Y_i are independent, (ii) the Y_i all have the same variance, (iii) the Y_i are normally distributed?
4. Write out the following three models in the form $\mathbf{y} = \mathbf{X}\beta + \epsilon$, ensuring that all the parameters left in the written out model are identifiable. In all cases y is the response variable, ϵ the residual “error” and other greek letters indicate model parameters.
 - (a) The ‘balanced one-way ANOVA model’:

$$y_{ij} = \alpha + \beta_i + \epsilon_{ij}$$

where $i = 1 \dots 3$ and $j = 1 \dots 2$.

- (b) A model with two explanatory factor variables and only 1 observation per combination of factor variables:

$$y_{ij} = \alpha + \beta_i + \gamma_j + \epsilon_{ij}$$

The first factor (β) has 3 levels and the second factor has 4 levels.

- (c) A model with two explanatory variables: a factor variable and a continuous variable, x .

$$y_i = \alpha + \beta_j + \gamma x_i + \epsilon_i \quad \text{if obs. } i \text{ is from factor level } j$$

Assume that $i = 1 \dots 6$, that the first two observations are for factor level 1 and the remaining 4 for factor level 2 and that the x_i 's are 0.1, 0.4, 0.5, 0.3, 0.4 and 0.7.

5. Consider some data for deformation (in mm), y_i of 3 different types of alloy, under different loads (in kg), x_i . When there is no load, there is no deformation, and the deformation is expected to vary linearly with load, in exactly the same way for all three alloys. However, as the load increases the three alloys deviate from this ideal linear behaviour in slightly different ways, with the relationship becoming slightly curved (possibly suggesting quadratic terms). The loads are known very precisely, so errors in x_i 's can be ignored, whereas the deformations, y_i are subject to larger measurement errors, that do need to be taken into account. Define a linear model suitable for describing these data, assuming that the same 6 loads are applied to each alloy, and write it out in the form $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$.
6. Show that for a linear model with model matrix \mathbf{X} , fitted to data \mathbf{y} , with fitted values $\hat{\boldsymbol{\mu}}$,

$$\mathbf{X}^T \hat{\boldsymbol{\mu}} = \mathbf{X}^T \mathbf{y}.$$

What implication does this have for the residuals of a model which includes an intercept term?

7. Equation (1.8) in section 1.3.3 gives an unbiased estimator of σ^2 , but in the text unbiasedness was only demonstrated assuming that the response data were normally distributed. By considering $\mathbb{E}(r_i^2)$ and the independence of the elements of \mathbf{r} , show that the estimator (1.8) is unbiased whatever the distribution of the response, provided that the response data are independent with constant variance.
8. The MASS library contains a data frame Rubber on wear of tyre rubber. The response loss measures rubber loss in grammes per hour. The predictors are tens, a measure of the tensile strength of the rubber with units of kgm^{-2} , and hard, the hardness of the rubber in Shore[§] units. Modelling interest focuses on predicting wear from hardness and tensile strength.
- (a) Starting with a model in which loss is a polynomial function of tens and hard, with all terms up to 3rd order present, perform backwards model selection, based on hypothesis testing, to select an appropriate model for these data.

[§] measures hardness as the extent of the rebound of a diamond tipped hammer, dropped on the test object.

- (b) Note the AIC scores of the various models that you have considered. It would also be possible to do model selection based on AIC, and the `step` function in R provides a convenient way of doing this. After reading the `step` help file, use it to select an appropriate model for the `loss` data, by AIC.
- (c) Use the `contour` and `predict` functions in R to produce a contour plot of model predicted `loss`, against `tens` and `hard`. You may find functions `seq` and `rep` helpful, as well.
9. The R data frame `warpbreaks` gives the number of `breaks` per fixed length of wool during weaving, for two different `wool` types, and 3 different weaving `tensions`. Using a linear model, establish whether there is evidence that the effect of tension on break rate is dependent on the type of wool. If there is, use `interaction.plot` to examine the nature of the dependence.
10. This question is about modelling the relationship between stopping distance of a car and its speed at the moment that the driver is signalled to stop. Data on this are provided in R data frame `cars`. It takes a more or less fixed ‘reaction time’ for a driver to apply the car’s brakes, so that the car will travel a distance directly proportional to its speed before beginning to slow. A car’s kinetic energy is proportional to the square of its speed, but the brakes can only dissipate that energy, and slow the car, at a roughly constant rate per unit distance travelled: so we expect that once braking starts, the car will travel a distance proportional to the square of its initial speed, before stopping.
- (a) Given the information provided above, fit a model of the form
- $$\text{dist}_i = \beta_0 + \beta_1 \text{speed}_i + \beta_2 \text{speed}_i^2 + \epsilon_i$$
- to the data in `cars`, and from this starting model, select the most appropriate model for the data using both AIC, and hypothesis testing methods.
- (b) From your selected model, estimate the average time that it takes a driver to apply the brakes (there are 5280 feet in a mile).
- (c) When selecting between different polynomial models for a set of data, it is often claimed that one should not leave in higher powers of some continuous predictor, while removing lower powers of that predictor. Is this sensible?
11. This question relates to the material in sections 1.3.1 to 1.3.4, and it may be useful to review sections A.5 and A.6 of Appendix A. R function `qr` computed the QR decomposition of a matrix, while function `qr.qry` provides a means of efficiently pre-multiplying a vector by the `Q` matrix of the decomposition and `qr.R` extracts the `R` matrix. See `?qr` for details.
- The question concerns calculation of estimates and associated quantities for the linear model, $y_i = \mathbf{X}_i \boldsymbol{\beta} + \epsilon_i$, where the ϵ_i are i.i.d. $N(0, \sigma^2)$.
- (a) Write an R function which will take a vector of response variables, `y`, and a model matrix, `X`, as arguments, and compute the least squares estimates of associated parameters, $\boldsymbol{\beta}$, based on QR decomposition of `X`.
- (b) Test your function by using it to estimate the parameters of the model

$$\text{dist}_i = \beta_0 + \beta_1 \text{speed}_i + \beta_2 \text{speed}_i^2 + \epsilon_i$$

for the data found in R data frame `cars`. Note that:

```
X <- model.matrix(dist ~ speed + I(speed^2), cars)
```

will produce a suitable model matrix. Check your answers against those produced by the `lm` function.

- (c) Extend your function to also return the estimated standard errors of the parameter estimators, and the estimated residual variance. Again, check your answers against what `lm` produces, using the `cars` model. Note that `solve(R)` or more efficiently `backsolve(R, diag(ncol(R)))` will produce the inverse of an upper triangular matrix `R`.
 - (d) Use R function `pt` to produce p-values for testing the null hypothesis that each β_i is zero (against a two sided alternative). Again check your answers against a summary of an equivalent `lm` fit.
 - (e) Extend your fitting function again, to produce the p-values associated with a sequential ANOVA table for your model (see section 1.3.4). Again test your results by comparison with the results of applying the `anova` function to an equivalent `lm` fit. Note that `pf` is the function giving the c.d.f. of F-distributions.
12. R data frame `InsectSprays` contains counts of insects in each of several plots. The plots had each been sprayed with one of 6 insecticides. A model for these data might be

$$y_i = \mu + \beta_j \text{ if } i^{\text{th}} \text{ observation is for spray } j.$$

A possible identifiability constraint for this model is that $\sum_j \beta_j = 0$. In R, construct the rank deficient model matrix, for this model, and the coefficient matrix for the sum to zero constraint on the parameters. Using the methods of section 1.8.1, impose the constraint via QR decomposition of the constraint matrix, fit the model (using `lm` with your constrained model matrix), and then obtain the estimates of the original parameters. You will need to use R functions `qr`, `qr.qty` and `qr.qty.y` as part of this. Confirm that your estimated parameters meet the constraint.

13. The R data frame `trees` contains data on Height, Girth and Volume of 31 felled cherry trees. A possible model for these data is

$$\text{Volume}_i = \beta_1 \text{Girth}_i^{\beta_2} \text{Height}_i^{\beta_3} + \epsilon_i,$$

which can be fitted by non-linear least squares using the method of section 1.8.6.

- (a) Write an R function to evaluate (i) the vector of $\mathbb{E}(\text{Volume})$ estimates given a vector of β_j values and vectors of `Girth` and `Height` measurements and (ii) the 31×3 ‘Jacobian’ matrix with $(i, j)^{\text{th}}$ element $\partial\mathbb{E}(\text{Volume}_i)/\partial\beta_j$, returning these in a two item list. (Recall that $\partial x^y / \partial y = x^y \log(x)$.)
- (b) Write R code to fit the model, to the `trees` data, using the method of section 1.8.6. Starting values of .002, 2 and 1 are reasonable.
- (c) Evaluate approximate standard error estimates for your estimated model parameters.

CHAPTER 2

Generalized Linear Models

Generalized linear models* (Nelder and Wedderburn, 1972) allow for response distributions other than normal, and for a degree of non-linearity in the model structure. A GLM has the basic structure

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta},$$

where $\mu_i \equiv \mathbb{E}(Y_i)$, g is a smooth monotonic ‘link function’, \mathbf{X}_i is the i^{th} row of a model matrix, \mathbf{X} , and $\boldsymbol{\beta}$ is a vector of unknown parameters. In addition, a GLM usually makes the distributional assumptions that the Y_i are independent and

$$Y_i \sim \text{some exponential family distribution.}$$

The *exponential family* of distributions includes many distributions that are useful for practical modelling, such as the Poisson, Binomial, Gamma and Normal distributions. The comprehensive reference for GLMs is McCullagh and Nelder (1989), while Dobson (2001) provides a thorough introduction.

Because generalized linear models are specified in terms of the ‘linear predictor’, $\mathbf{X}\boldsymbol{\beta}$, many of the general ideas and concepts of linear modelling carry over, with a little modification, to generalized linear modelling. Basic model formulation is much the same as for linear models, except that a link function and distribution must be chosen. Of course, if the identity function is chosen as the link, along with the normal distribution, then ordinary linear models are recovered as a special case.

The generalization comes at some cost however: model fitting now has to be done iteratively, and distributional results, used for inference, are now approximate and justified by large sample limiting results, rather than being exact. But before going further into these issues, consider a couple of simple examples.

Example 1: In the early stages of a disease epidemic, the rate at which new cases occur can often increase exponentially through time. Hence, if μ_i is the expected number of new cases on day t_i , a model of the form

$$\mu_i = c \exp(bt_i),$$

* Note that there is a distinction between ‘generalized’ and ‘general’ linear models - the latter term being sometimes used to refer to all linear models other than simple straight lines.

might be appropriate, where c and b are unknown parameters. Such a model can be turned into GLM form, by using a log link so that

$$\log(\mu_i) = \log(c) + bt_i = \beta_0 + t_i\beta_1$$

(by definition of $\beta_0 = \log c$ and $\beta_1 = b$). Note that the right hand side of the model is now linear in the parameters. The response variable is the number of new cases per day and, since this is a count, the Poisson distribution is probably a reasonable distribution to try. So the GLM for this situation uses a Poisson response distribution, log link, and linear predictor $\beta_0 + t_i\beta_1$.

Example 2: The rate of capture of prey items, y_i , by a hunting animal, tends to increase with increasing density of prey, x_i , but to eventually level off, when the predator is catching as much as it can cope with. A suitable model for this situation might be

$$\mu_i = \frac{ax_i}{h + x_i},$$

where a is an unknown parameter, representing the maximum capture rate, and h is an unknown parameter, representing the prey density at which the capture rate is half the maximum rate. Obviously this model is non-linear in its parameters, but, by using a reciprocal link, the right hand side can be made linear in the parameters:

$$\frac{1}{\mu_i} = \frac{1}{a} + \frac{h}{a} \frac{1}{x_i} = \beta_0 + \frac{1}{x_i}\beta_1$$

(here $\beta_0 \equiv 1/a$ and $\beta_1 \equiv h/a$). In this case the standard deviation of prey capture rate might be approximately proportional to the mean rate, suggesting the use of a Gamma distribution for the response, and completing the model specification.

Of course we are not restricted to the simple straight line forms of the examples, but can have any structure for the linear predictor that was possible for linear models.

2.1 The theory of GLMs

Estimation and inference with GLMs is based on the theory of maximum likelihood estimation, although the maximization of the likelihood turns out to require an iterative least squares approach, related to the method of section 1.8.6. This section begins by introducing the exponential family of distributions, which allows a general method to be developed for maximizing the likelihood of a GLM. Inference for GLMs is then discussed, based on general results of likelihood theory (which are derived at the end of the chapter). In this section it is sometimes useful to distinguish between the response data, y , and the random variable that it is an observation of, Y , so they are distinguished notationally: this has not been done for estimates and estimators.

THE THEORY OF GLMS

	Normal	Poisson	Binomial	Gamma	Inverse Gaussian
$f(y)$	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(y-\mu)^2}{2\sigma^2}\right)$	$\frac{\mu^y \exp(-\mu)}{y!}$	$\binom{n}{y} \left(\frac{\mu}{n}\right)^y \left(1 - \frac{\mu}{n}\right)^{n-y}$	$\frac{1}{\Gamma(\nu)} \left(\frac{\nu}{\mu}\right)^\nu y^{\nu-1} \exp\left(-\frac{\nu y}{\mu}\right)$	$\sqrt{\frac{\gamma}{2\pi y^3}} \exp\left[\frac{-\gamma(y-\mu)^2}{2\mu^2 y}\right]$
Range	$-\infty < y < \infty$	$y = 0, 1, 2, \dots$	$y = 0, 1, \dots, n$	$y > 0$	$y > 0$
θ	μ	$\log(\mu)$	$\log\left(\frac{\mu}{n-\mu}\right)$	$-\frac{1}{\mu}$	$-\frac{1}{2\mu^2}$
ϕ	σ^2	1	1	$\frac{1}{\nu}$	$\frac{1}{\gamma}$
$a(\phi)$	$\phi (= \sigma^2)$	$\phi (= 1)$	$\phi (= \frac{1}{\nu})$	$\phi (= \frac{1}{\gamma})$	
$b(\theta)$	$\frac{\theta^2}{2}$	$\exp(\theta)$	$n \log(1 + e^\theta)$	$-\log(-\theta)$	$-\sqrt{-2\theta}$
$c(y, \phi)$	$-\frac{1}{2} \left[\frac{y^2}{\phi} + \log(2\pi\phi) \right]$	$-\log(y!)$	$\log\binom{n}{y}$	$\nu \log(\nu y) - \log(y\Gamma(\nu))$	$-\frac{1}{2} \left[\log(2\pi y^3 \phi) + \frac{1}{\phi y} \right]$
$V(\mu)$	1	μ	$\mu(1 - \mu/n)$	μ^2	μ^3
$g_c(\mu)$	μ	$\log(\mu)$	$\frac{\mu}{n-\mu}$	$\frac{1}{\mu}$	$\frac{1}{\mu^2}$
$D(y, \hat{\mu})$	$(y - \hat{\mu})^2$	$2y \log\left(\frac{y}{\hat{\mu}}\right) - 2 \left[y \log\left(\frac{y}{\hat{\mu}}\right) + (n-y) \log\left(\frac{n-y}{n-\hat{\mu}}\right) \right]$	$2 \left[\frac{y-\hat{\mu}}{\hat{\mu}} - \log\left(\frac{y}{\hat{\mu}}\right) \right]$	$\frac{(y-\hat{\mu})^2}{\hat{\mu}^2 y}$	

Table 2.1 Some Exponential Family Distributions. Note that when $y = 0$, $y \log(y/\hat{\mu})$ is replaced by zero (its limit as $y \rightarrow 0$).

2.1.1 The exponential family of distributions

The response variable in a GLM can have any distribution from the *exponential family*. A distribution belongs to the exponential family of distributions if its probability density function, or probability mass function, can be written as

$$f_\theta(y) = \exp \{[y\theta - b(\theta)]/a(\phi) + c(y, \phi)\},$$

where b , a and c are arbitrary functions, ϕ an arbitrary ‘scale’ parameter, and θ is known as the ‘canonical parameter’ of the distribution (in the GLM context, θ will completely depend on the model parameters β , but it is not necessary to make this explicit yet).

For example, it is easy to see that the normal distribution is a member of the exponential family since

$$\begin{aligned} f_\mu(y) &= \frac{1}{\sigma\sqrt{2\pi}} \exp \left[-\frac{(y-\mu)^2}{2\sigma^2} \right] \\ &= \exp \left[\frac{-y^2 + 2y\mu - \mu^2}{2\sigma^2} - \log(\sigma\sqrt{2\pi}) \right] \\ &= \exp \left[\frac{y\mu - \mu^2/2}{\sigma^2} - \frac{y^2}{2\sigma^2} - \log(\sigma\sqrt{2\pi}) \right], \end{aligned}$$

which is of exponential form, with $\theta = \mu$, $b(\theta) = \theta^2/2 \equiv \mu^2/2$, $a(\phi) = \phi = \sigma^2$ and $c(\phi, y) = -y^2/(2\phi) - \log(\sqrt{\phi 2\pi}) \equiv -y^2/(2\sigma^2) - \log(\sigma\sqrt{2\pi})$. Table 2.1 gives a similar breakdown for the members of the exponential family implemented for GLMs in R.

It is possible to obtain general expressions for the mean and variance of exponential family distributions, in terms of a , b and ϕ . The log likelihood of θ , given a particular y , is simply $\log[f_\theta(y)]$ considered as a function of θ . That is

$$l(\theta) = [y\theta - b(\theta)]/a(\phi) + c(y, \phi)$$

and so

$$\frac{\partial l}{\partial \theta} = [y - b'(\theta)]/a(\phi).$$

Treating l as a random variable, by replacing the particular observation y by the random variable Y , enables the expected value of $\partial l/\partial \theta$ to be evaluated:

$$\mathbb{E}\left(\frac{\partial l}{\partial \theta}\right) = [\mathbb{E}(Y) - b'(\theta)]/a(\phi).$$

Using the general result that $\mathbb{E}(\partial l/\partial \theta) = 0$ (at the true value of θ , see (2.14) in section 2.4) and re-arranging implies that

$$\mathbb{E}(Y) = b'(\theta). \tag{2.1}$$

i.e. the mean, of any exponential family random variable, is given by the first derivative of b w.r.t. θ , where the form of b depends on the particular distribution. This equation is the key to linking the model parameters, β , of a GLM to the canonical

parameters of the exponential family. In a GLM, the parameters β determine the mean of the response variable, and, via (2.1), they thereby determine the canonical parameter for each response observation.

Differentiating the likelihood once more yields

$$\frac{\partial^2 l}{\partial \theta^2} = -b''(\theta)/a(\phi),$$

and plugging this into the general result, $\mathbb{E}[\partial^2 l / \partial \theta^2] = -\mathbb{E}[(\partial l / \partial \theta)^2]$ (the derivatives are evaluated at the true θ value, see result (2.16), section 2.4), gives

$$b''(\theta)/a(\phi) = \mathbb{E}[(Y - b'(\theta))^2]/a(\phi)^2,$$

which re-arranges to the second useful general result:

$$\text{var}(Y) = b''(\theta)a(\phi).$$

a could in principle be any function of ϕ , and when working with GLMs there is no difficulty in handling any form of a , if ϕ is known. However, when ϕ is unknown matters become awkward, unless we can write $a(\phi) = \phi/\omega$, where ω is a known constant. This restricted form in fact covers all the cases of practical interest here (see e.g. table 2.1). $a(\phi) = \phi/\omega$ allows the possibility of, for example, unequal variances in models based on the normal distribution, but in most cases ω is simply 1. Hence we now have

$$\text{var}(Y) = b''(\theta)\phi/\omega. \quad (2.2)$$

In subsequent sections it will often be convenient to consider $\text{var}(Y)$ as a function of $\mu \equiv \mathbb{E}(Y)$, and, since μ and θ are linked via (2.1), we can always define a function $V(\mu) = b''(\theta)/\omega$, such that $\text{var}(Y) = V(\mu)\phi$. Several such functions are listed in table 2.1.

2.1.2 Fitting Generalized Linear Models

Recall that a GLM models an n -vector of independent response variables, \mathbf{Y} , where $\boldsymbol{\mu} \equiv \mathbb{E}(\mathbf{Y})$, via

$$g(\mu_i) = \mathbf{X}_i \boldsymbol{\beta}$$

and

$$Y_i \sim f_{\theta_i}(y_i),$$

where $f_{\theta_i}(y_i)$ indicates an exponential family distribution, with canonical parameter θ_i , which is determined by μ_i (via equation 2.1) and hence ultimately by $\boldsymbol{\beta}$. Given vector \mathbf{y} , an observation of \mathbf{Y} , maximum likelihood estimation of $\boldsymbol{\beta}$ is possible. Since the Y_i are mutually independent, the likelihood of $\boldsymbol{\beta}$ is

$$L(\boldsymbol{\beta}) = \prod_{i=1}^n f_{\theta_i}(y_i),$$

and hence the log-likelihood of β is

$$\begin{aligned} l(\beta) &= \sum_{i=1}^n \log[f_{\theta_i}(y_i)] \\ &= \sum_{i=1}^n [y_i \theta_i - b_i(\theta_i)]/a_i(\phi) + c_i(\phi, y_i), \end{aligned}$$

where the dependence of the right hand side on β is through the dependence of the θ_i on β . Notice that the functions a , b and c may vary with i — this allows different binomial denominators, n_i , for each observation of a binomial response, or different (but known to within a constant) variances for normal responses, for example. ϕ , on the other hand, is assumed to be the same for all i . As discussed in the previous section, for practical work it suffices to consider only cases where we can write $a_i(\phi) = \phi/\omega_i$, where ω_i is a known constant (usually 1), in which case

$$l(\beta) = \sum_{i=1}^n \omega_i [y_i \theta_i - b_i(\theta_i)]/\phi + c_i(\phi, y_i).$$

Maximization proceeds by partially differentiating l w.r.t. each element of β , setting the resulting expressions to zero and solving for β .

$$\frac{\partial l}{\partial \beta_j} = \frac{1}{\phi} \sum_{i=1}^n \omega_i \left(y_i \frac{\partial \theta_i}{\partial \beta_j} - b'_i(\theta_i) \frac{\partial \theta_i}{\partial \beta_j} \right),$$

and by the chain rule

$$\frac{\partial \theta_i}{\partial \beta_j} = \frac{\partial \theta_i}{\partial \mu_i} \frac{\partial \mu_i}{\partial \beta_j},$$

so that differentiating (2.1), we get

$$\frac{\partial \mu_i}{\partial \theta_i} = b''_i(\theta_i) \Rightarrow \frac{\partial \theta_i}{\partial \mu_i} = \frac{1}{b''_i(\theta_i)},$$

which then implies that

$$\frac{\partial l}{\partial \beta_j} = \frac{1}{\phi} \sum_{i=1}^n \frac{[y_i - b'_i(\theta_i)]}{b''_i(\theta_i)/\omega_i} \frac{\partial \mu_i}{\partial \beta_j}.$$

Substituting from (2.1) and (2.2), into this last equation, implies that the equations to solve for β are

$$\sum_{i=1}^n \frac{(y_i - \mu_i)}{V(\mu_i)} \frac{\partial \mu_i}{\partial \beta_j} = 0 \quad \forall j. \quad (2.3)$$

However, these equations are exactly the equations that would have to be solved in order to find β by non-linear weighted least squares, if the weights $V(\mu_i)$ were known in advance and were independent of β . In this case the least squares objective would be

$$S = \sum_{i=1}^n \frac{(y_i - \mu_i)^2}{V(\mu_i)}, \quad (2.4)$$

where μ_i depends non-linearly on β , but the weights $V(\mu_i)$ are treated as fixed. To find the least squares estimates involves solving $\partial S/\partial \beta_j = 0 \forall j$, but this system of equations is easily seen to be (2.3), when the $V(\mu_i)$ terms are treated as fixed.

This correspondence immediately suggests an iterative method for solving (2.3). Let $\hat{\beta}^{[k]}$ denote the estimated parameter vector at the k^{th} iterate and let $\eta^{[k]}$ and $\mu^{[k]}$ be the vectors with elements $\eta_i^{[k]} = \mathbf{X}_i \hat{\beta}^{[k]}$ and $\mu_i^{[k]} = g^{-1}(\eta_i^{[k]})$, respectively, where $g^{-1}(\cdot)$ is the inverse function of the link. Starting with a parameter guesstimate, $\hat{\beta}^{[0]}$, the following steps are iterated until the sequence of $\hat{\beta}^{[k]}$'s converges:

1. Calculate the $V(\mu_i^{[k]})$ terms implied by the current $\hat{\beta}^{[k]}$.
2. Given these estimates use the method of section 1.8.6 to minimize (2.4) with respect to β , in order to obtain $\hat{\beta}^{[k+1]}$ (the $V(\mu_i^{[k]})$ being treated as fixed and not as functions of β).
3. Set k to $k + 1$.

In fact this method is slower than it need be. Step 2 itself involves iteration, but there is little point in actually iterating the non-linear least squares method to convergence before the $V(\mu_i^{[k]})$ have converged. Hence step 2 is usually replaced by:

2. Using $\hat{\beta}^{[k]}$ as the starting values, perform one iteration only, of the iterative method of solving (2.4) given in section 1.8.6, to obtain $\hat{\beta}^{[k+1]}$.

Applying this approach results in a rather compact and neat scheme. To see this let us write the non-linear least squares problem in matrix form. Defining diagonal matrix $\mathbf{V}_{[k]}$ where $V_{[k]ii} = V(\mu_i^{[k]})$, (2.4) becomes

$$S = \left\| \sqrt{\mathbf{V}_{[k]}^{-1}} [\mathbf{y} - \boldsymbol{\mu}(\beta)] \right\|^2$$

and, following the method of section 1.8.6, $\boldsymbol{\mu}$ is replaced by its first order Taylor expansion around $\hat{\beta}^{[k]}$ so that

$$S \approx \left\| \sqrt{\mathbf{V}_{[k]}^{-1}} \left[\mathbf{y} - \boldsymbol{\mu}^{[k]} - \mathbf{J} (\beta - \hat{\beta}^{[k]}) \right] \right\|^2$$

where \mathbf{J} is the ‘Jacobian’ matrix, with elements $J_{ij} = \partial \mu_i / \partial \beta_j |_{\hat{\beta}^{[k]}}$. Now

$$g(\mu_i) = \mathbf{X}_i \beta \Rightarrow g'(\mu_i) \frac{\partial \mu_i}{\partial \beta_j} = X_{ij}$$

and hence

$$J_{ij} = \left. \frac{\partial \mu_i}{\partial \beta_j} \right|_{\hat{\beta}^{[k]}} = X_{ij} / g'(\mu_i^{[k]}).$$

So defining \mathbf{G} as the diagonal matrix with elements $G_{ii} = g'(\mu_i^{[k]})$, $\mathbf{J} = \mathbf{G}^{-1} \mathbf{X}$. Hence, without further approximation,

$$\begin{aligned} S &\approx \left\| \sqrt{\mathbf{V}_{[k]}^{-1}} \mathbf{G}^{-1} \left[\mathbf{G}(\mathbf{y} - \boldsymbol{\mu}^{[k]}) + \boldsymbol{\eta}^{[k]} - \mathbf{X}\beta \right] \right\|^2 \\ &= \left\| \sqrt{\mathbf{W}^{[k]}} (\mathbf{z}^{[k]} - \mathbf{X}\beta) \right\|^2 \end{aligned}$$

by definition of ‘pseudodata’

$$z_i^{[k]} = g'(\mu^{[k]}) (y_i - \mu_i^{[k]}) + \eta_i^{[k]}$$

and diagonal weight matrix, $\mathbf{W}^{[k]}$, with elements

$$W_{ii}^{[k]} = \frac{1}{V(\mu_i^{[k]})g'(\mu_i^{[k]})^2}.$$

The following steps are therefore iterated to convergence

1. Using the current $\boldsymbol{\mu}^{[k]}$ and $\boldsymbol{\eta}^{[k]}$ calculate pseudodata $\mathbf{z}^{[k]}$ and iterative weights $\mathbf{W}^{[k]}$.
2. Minimize the sum of squares $\left\| \sqrt{\mathbf{W}^{[k]}} (\mathbf{z}^{[k]} - \mathbf{X}\boldsymbol{\beta}) \right\|^2$ with respect to $\boldsymbol{\beta}$, in order to obtain $\hat{\boldsymbol{\beta}}^{[k+1]}$, and hence $\boldsymbol{\eta}^{[k+1]} = \mathbf{X}\hat{\boldsymbol{\beta}}^{[k+1]}$ and $\boldsymbol{\mu}^{[k+1]}$. Increment k by one.

The converged $\hat{\boldsymbol{\beta}}$ solves (2.3), and is hence the maximum likelihood estimate of $\boldsymbol{\beta}^\dagger$. The algorithm converges in most practical circumstances, although there are exceptions (for example poor or overly flexible models of binomial data).

Notice that to start the iteration we only need $\boldsymbol{\mu}^{[0]}$ and $\boldsymbol{\eta}^{[0]}$ values, but not $\hat{\boldsymbol{\beta}}^{[0]}$. Hence the iteration is usually started by setting $\mu_i^{[0]} = y_i$ and $\eta_i^{[0]} = g(\mu_i^{[0]})$, with slight adjustment of $\mu_i^{[0]}$, as required, to avoid infinite $\eta_i^{[0]}$ ’s (e.g. if $y_i = 0$ with a log link). The method is known as *Iteratively Re-weighted Least Squares* (IRLS) for obvious reasons, and is due, in this context, to Nelder and Wedderburn (1972).

2.1.3 The IRLS objective is a quadratic approximation to the log-likelihood

The working linear model in the IRLS iteration is not simply a means of finding the maximum likelihood estimates of the parameters. To within an additive constant

$$\mathcal{S} = -\frac{1}{2\phi} \left\| \sqrt{\mathbf{W}} (\mathbf{z} - \mathbf{X}\boldsymbol{\beta}) \right\|^2$$

(at convergence) is also a quadratic approximation to the log likelihood of the model in the vicinity of $\hat{\boldsymbol{\beta}}$. Clearly the first derivatives w.r.t. the β_j match between the log-likelihood and \mathcal{S} : in fact they are all zero. The second derivative matrix of \mathcal{S} is $-\mathbf{X}\mathbf{W}\mathbf{X}/\phi$ and this turns out to match the expected second derivative matrix of the log likelihood, and hence, by the law of large numbers, the second derivative matrix itself, in the large sample limit.

To prove this, first define \mathbf{u} as the vector of derivatives of the log likelihood w.r.t. the model parameters, so that $u_i = \partial l / \partial \beta_i$, and then re-write the derivatives in (2.3) in matrix vector form as

$$\mathbf{u} = \mathbf{X}^T \mathbf{G}^{-1} \mathbf{V}^{-1} (\mathbf{y} - \boldsymbol{\mu}) / \phi.$$

[†] Note that the algorithm would not minimize (2.4) if $V(\mu_i)$ was treated as a function of $\boldsymbol{\beta}$, since in that case equating the derivatives to zero would not yield (2.3). i.e. the likelihood maximization is fundamentally different to least squares with a mean variance relationship.

Then

$$\begin{aligned}\mathbb{E}(\mathbf{u}\mathbf{u}^T) &= \mathbf{X}^T \mathbf{G}^{-1} \mathbf{V}^{-1} \mathbb{E}[(\mathbf{Y} - \boldsymbol{\mu})(\mathbf{Y} - \boldsymbol{\mu})^T] \mathbf{V}^{-1} \mathbf{G}^{-1} \mathbf{X} / \phi^2 \\ &= \mathbf{X}^T \mathbf{G}^{-1} \mathbf{V} \mathbf{V}^{-1} \mathbf{G}^{-1} \mathbf{X} / \phi \\ &= \mathbf{X}^T \mathbf{W} \mathbf{X} / \phi\end{aligned}$$

since $\mathbb{E}[(\mathbf{Y} - \boldsymbol{\mu})(\mathbf{Y} - \boldsymbol{\mu})^T] = \mathbf{V}\phi$. By the general likelihood result (2.19), in section 2.4.2, $-\mathbb{E}(\mathbf{u}\mathbf{u}^T)$ is also the expected second derivative matrix of the log likelihood.

This correspondence of derivatives is sufficient to demonstrate that \mathcal{S} is a quadratic approximation to the log-likelihood in the vicinity of the $\hat{\boldsymbol{\beta}}$, and, by consistency of MLEs, in the vicinity of the true parameter values.

2.1.4 AIC for GLMs

Model selection by direct comparison of likelihoods suffers from the problem that, if redundant parameters are added to a correct model, the likelihood almost always increases (and never decreases), because the extra parameters let the model get closer to the data, even though that only means ‘modelling the noise’ component of the data. As in the linear model case, this problem would be alleviated if we were somehow able to choose models on the basis of their ability to fit the mean of the data, $\boldsymbol{\mu}$, rather than the data, \mathbf{y} . In a GLM context, a reasonable approach would be to choose between models on the basis of their ability to maximize $l(\boldsymbol{\beta}; \boldsymbol{\mu})$, rather than $l(\boldsymbol{\beta}; \mathbf{y})$, but to do so we have to be able to estimate $l(\boldsymbol{\beta}; \boldsymbol{\mu})$.

Actually this estimation is straightforward. From section 2.1.3 we have that

$$l(\hat{\boldsymbol{\beta}}; \mathbf{y}) \simeq k - \frac{1}{2\phi} \left\| \sqrt{\mathbf{W}} (\mathbf{z} - \mathbf{X}\hat{\boldsymbol{\beta}}) \right\|^2,$$

and since this must also hold true if $\mathbf{y} = \boldsymbol{\mu}$

$$l(\hat{\boldsymbol{\beta}}; \boldsymbol{\mu}) \simeq k - \frac{1}{2\phi} \left\| \sqrt{\mathbf{W}} (\boldsymbol{\eta} - \mathbf{X}\hat{\boldsymbol{\beta}}) \right\|^2.$$

Then the argument leading to (1.15) in section 1.8.5 (modified only to include weights) yields the estimator

$$\begin{aligned}\widehat{l}(\hat{\boldsymbol{\beta}}; \boldsymbol{\mu}) &= k - \frac{1}{2\phi} \left\| \sqrt{\mathbf{W}} (\mathbf{z} - \mathbf{X}\hat{\boldsymbol{\beta}}) \right\|^2 + n/2 - \text{tr}(\mathbf{A}) \\ &\simeq l(\hat{\boldsymbol{\beta}}; \mathbf{y}) - \text{tr}(\mathbf{A}) + n/2\end{aligned}$$

where $\mathbf{A} = \mathbf{X}(\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}$ and hence $\text{tr}(\mathbf{A}) = p$, the number of (identifiable) model parameters.

Hence, when choosing between models, we would choose whichever model had the highest value of $l(\hat{\boldsymbol{\beta}}) - p$, which is equivalent to choosing the model with the lowest value of Akaike’s Information Criterion (Akaike, 1973),

$$AIC = 2[-l(\hat{\boldsymbol{\beta}}) + p].$$

The forgoing argument assumes that ϕ is known. If it is not then an estimate[†], $\hat{\phi}$, will be needed in order to evaluate the AIC, and as a result the penalty term p in the AIC will become $p + 1$. This generalization is justified in section 2.4.7.

2.1.5 Large sample distribution of $\hat{\beta}$

Distributional results for GLMs are not exact, but are based instead on large sample approximations, making use of general properties of maximum likelihood estimators including consistency (see section 2.4). From the general properties of maximum likelihood estimators, we have that, in the large sample limit,

$$\hat{\beta} \sim N(\beta, \mathcal{I}^{-1}),$$

where $\mathcal{I} = \mathbb{E}(\mathbf{u}\mathbf{u}^T)$ is the *information* matrix of the model parameters, and \mathbf{u} is the vector of derivatives of the log-likelihood w.r.t. the model parameters, so that $u_i = \partial l / \partial \beta_i$ (see (2.20) and (2.19) in section 2.4). In section 2.1.3 it was shown that $\mathbb{E}(\mathbf{u}\mathbf{u}^T) = \mathbf{X}^T \mathbf{W} \mathbf{X} / \phi$ and hence in the large sample limit

$$\hat{\beta} \sim N(\beta, (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \phi).$$

For distributions with known scale parameter, ϕ , this result can be used directly to find confidence intervals for the parameters, but if the scale parameter is unknown (e.g. for the normal distribution), then it must be estimated, and intervals must be based on an appropriate t distribution. Scale parameter estimation is covered in section 2.1.7.

2.1.6 Comparing models by hypothesis testing

Consider testing

$$H_0 : g(\mu) = \mathbf{X}_0 \beta_0$$

against

$$H_1 : g(\mu) = \mathbf{X}_1 \beta_1,$$

where μ is the expectation of a response vector, \mathbf{Y} , whose elements are independent random variables from the same member of the exponential family of distributions, and where $\mathbf{X}_0 \subset \mathbf{X}_1$. If we have an observation, \mathbf{y} , of the response vector, then a generalized likelihood ratio test can be performed. Let $l(\hat{\beta}_0)$ and $l(\hat{\beta}_1)$ be the maximized likelihoods of the two models. If H_0 is true then in the large sample limit,

$$2[l(\hat{\beta}_1) - l(\hat{\beta}_0)] \sim \chi^2_{p_1 - p_0}, \quad (2.5)$$

where p_i is the number of (identifiable) parameters (β_i) in model i (see sections 2.4.5 and 2.4.6 for the derivation of this result). If the null hypothesis is false, then model 1 will tend to have a substantially higher likelihood than model 0, so that twice the

[†] which should strictly be a maximum likelihood estimate, or an estimator tending to the MLE in large sample limit.

difference in log likelihoods would be too large for consistency with the relevant χ^2 distribution.

The approximate result (2.5) is only directly useful if the log likelihoods of the models concerned can be calculated. In the case of GLMs estimated by IRLS, this is only the case if the scale parameter, ϕ , is known. Hence the result can be used directly with Poisson and binomial models, but not with the normal[§], gamma or inverse Gaussian distributions, where the scale parameter is not known. What to do in these latter cases will be discussed shortly.

Deviance

When working with GLMs in practice, it is useful to have a quantity that can be interpreted in a similar way to the residual sum of squares, in ordinary linear modelling. This quantity is the *deviance* of the model and is defined as

$$D = 2[l(\hat{\beta}_{\max}) - l(\hat{\beta})]\phi \quad (2.6)$$

$$= \sum_{i=1}^n 2\omega_i [y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)], \quad (2.7)$$

where $l(\hat{\beta}_{\max})$ indicates the maximized likelihood of the saturated model: the model with one parameter per data point. $l(\hat{\beta}_{\max})$ is the highest value that the likelihood could possibly have, given the data, and is evaluated by simply setting $\hat{\mu} = \mathbf{y}$ and evaluating the likelihood. $\tilde{\theta}$ and $\hat{\theta}$ denote the maximum likelihood estimates of canonical parameters, for the saturated model and model of interest, respectively. Notice how the deviance is defined to be independent of ϕ . Table 2.1 lists the contributions of a single datum to the deviance, for several distributions — these are the terms inside the summation in the definition of the deviance.

Related to the deviance is the *scaled deviance*,

$$D^* = D/\phi,$$

which does depend on the scale parameter. For the Binomial and Poisson distributions, where $\phi = 1$, the deviance and scaled deviance are the same, but this is not the case more generally.

By the generalized likelihood ratio test result (2.5), we might expect that, if the model is correct, then approximately

$$D^* \sim \chi^2_{n-p}, \quad (2.8)$$

in the large sample limit. Actually such an argument is bogus, since the limiting argument justifying (2.5) relies on the number of parameters in the model staying fixed, while the sample size tends to infinity, but the saturated model has as many parameters as data. Asymptotic results *are* available for some of the distributions in table 2.1, to justify (2.8) as a large sample approximation under many circumstances (see

[§] Of course for normal distribution and identity link we use the results of chapter 1.

McCullagh and Nelder, 1989), and it is exact for the Normal case. Note, however, that it breaks down entirely for the binomial with binary data.

Given the definition of deviance, it is easy to see that the likelihood ratio test, with which this section started, can be performed by re-expressing the twice log-likelihood ratio statistic as $D_0^* - D_1^*$. Then under H_0

$$D_0^* - D_1^* \sim \chi_{p_1-p_0}^2 \quad (2.9)$$

(in the large sample limit), where D_i^* is the deviance of model i which has p_i identifiable parameters. But again, this is only useful if the scale parameter is known so that D^* can be calculated.

Model comparison with unknown ϕ

Under H_0 we have the approximate results

$$D_0^* - D_1^* \sim \chi_{p_1-p_0}^2 \text{ and } D_1^* \sim \chi_{n-p}^2,$$

and, if $D_0^* - D_1^*$ and D_1^* are treated as asymptotically independent, this implies that

$$F = \frac{(D_0^* - D_1^*)/(p_1 - p_0)}{D_1^*/(n - p_1)} \sim F_{p_1-p_0, n-p_1},$$

in the large sample limit (a result which is exactly true in the ordinary linear model special case, of course). The useful property of F is that it can be calculated without knowing ϕ , which can be cancelled from top and bottom of the ratio yielding, under H_0 , the approximate result that

$$F = \frac{(D_0 - D_1)/(p_1 - p_0)}{D_1/(n - p_1)} \sim F_{p_1-p_0, n-p_1}. \quad (2.10)$$

The advantage of this result is that it can be used for hypothesis testing based model comparison, when ϕ is unknown. The disadvantages are the dubious distributional assumption for D_1^* , and the independence approximation, on which it is based.

Of course an obvious alternative approach would be to use an estimate, $\hat{\phi}$, to obtain an estimate, $\hat{D}_i^* = D_i\hat{\phi}$, for each model, and then to use (2.9) for hypothesis testing. However if we use the estimate (2.11) for this purpose then it is readily seen that $\hat{D}_0^* - \hat{D}_1^*$ is simply $(n - p_1) \times F$, so our test would be exactly equivalent to using the F ratio result (2.10), but with $F_{p_1-p_0, \infty}$ as the reference distribution. Clearly direct use of (2.10) is a more conservative approach, and hence usually to be preferred: it at least makes some allowance for the uncertainty in estimating the scale parameter.

2.1.7 $\hat{\phi}$ and Pearson's statistic

As we have seen, the MLEs of the parameters, β , can be obtained without knowing the scale parameter, ϕ , but, in those cases in which this parameter is unknown, it must usually be estimated. Approximate result (2.8) provides one obvious estimator.

The expected value of a χ^2_{n-p} random variable is $n - p$, so equating the observed $D^* = D/\phi$ to its approximate expected value we have

$$\hat{\phi}_D = \hat{D}/(n - p). \quad (2.11)$$

A second estimator is based on the *Pearson statistic*, which is defined as

$$X^2 = \sum_{i=1}^n \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

Clearly X^2/ϕ would be the sum of squares of a set of zero mean, unit variance, random variables, having $n - p$ degrees of freedom, suggesting ¶ that if the model is adequate then approximately $X^2/\phi \sim \chi^2_{n-p}$: this approximation turns out to be well founded. Setting the observed Pearson statistic to its expected value we get

$$\hat{\phi} = \hat{X}^2/(n - p).$$

Note, that it is straightforward to show that

$$X^2 = \|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\hat{\beta})\|^2,$$

where \mathbf{W} and \mathbf{z} are the IRLS weights and pseudodata, evaluated at convergence.

2.1.8 Canonical link functions

The canonical link, g_c , for a distribution, is the link function such that $g_c(\mu_i) = \theta_i$, where θ_i is the canonical parameter of the distribution. For example, for the Poisson distribution the canonical link is the log function (see table 2.1 for other examples). Use of the canonical link means that $\theta_i = \mathbf{X}_i\beta$ (where \mathbf{X}_i is the i th row of \mathbf{X}).

Canonical links tend to have some nice properties, such as ensuring that μ stays within the range of the response variable, but they also have more subtle advantages, one of which is derived here. Recall that likelihood maximization involves differentiating the log likelihood with respect to each β_j , and setting the results to zero, to obtain the system of equations

$$\frac{\partial l}{\partial \beta_j} = \sum_{i=1}^n \omega_i \left(y_i \frac{\partial \theta_i}{\partial \beta_j} - \mu_i \frac{\partial \theta_i}{\partial \beta_j} \right) = 0 \quad \forall j.$$

But if the canonical link is being used then $\partial \theta_i / \partial \beta_j = X_{ij}$, and if, as is often the case, $w_i = 1 \forall i$ this system of equations reduces to

$$\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \hat{\mu} = 0,$$

i.e. to $\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \hat{\mu}$. Now consider the case in which \mathbf{X} contains a column of 1's: this implies that one of the equations in this system is simply $\sum_i y_i = \sum_i \hat{\mu}_i$. Similarly any other weighted summation, where the weights are given by model matrix

¶ Recall that if $\{Z_i : i = 1 \dots n\}$ are a set of i.i.d. $N(0, 1)$ r.v.'s then $\sum Z_i^2 \sim \chi^2_n$.

columns (or a linear combination of these), is conserved between the raw data and the fitted values.

One practical upshot of this is that, for any GLM with an intercept term and canonical link, the residuals will sum to zero: this ‘observed unbiasedness’ is a reassuring property. Another practical use of the result is in categorical data analysis using log-linear models, where it provides a means of ensuring, via specification of the model, that totals which were built into the design of a study can be preserved in any model.

2.1.9 Residuals

Model checking is perhaps the most important part of applied statistical modelling. In the case of ordinary linear models, this is based on examination of the model residuals, which contain all the information in the data, not explained by the systematic part of the model. Examination of residuals is also the chief means for model checking in the case of GLMs, but in this case the standardization of residuals is both necessary and a little more difficult.

For GLMs the main reason for not simply examining the raw residuals, $\hat{\epsilon}_i = y_i - \hat{\mu}_i$, is the difficulty of checking the validity of the assumed mean variance relationship from the raw residuals. For example, if a Poisson model is employed, then the variance of the residuals should increase in direct proportion to the size of the fitted values ($\hat{\mu}_i$). However if raw residuals are plotted against fitted values it takes an extra-ordinary ability to judge whether the residual variability is increasing in proportion to the mean, as opposed to, say, the square root or square of the mean. For this reason it is usual to standardize GLM residuals, in such a way that, if the model assumptions are correct, the standardized residuals should have approximately equal variance, and behave, as far as possible, like residuals from an ordinary linear model (although see figure 6.9 in section 6.5 for an alternative plotting approach).

Pearson Residuals

The most obvious way to standardize the residuals is to divide them by a quantity proportional to their standard deviation according to the fitted model. This gives rise to the *Pearson residuals*

$$\hat{\epsilon}_i^P = \frac{y_i - \hat{\mu}_i}{\sqrt{V(\hat{\mu}_i)}},$$

which should have approximately zero mean and variance ϕ , if the model is correct. These residuals should not display any trend in mean or variance when plotted against the fitted values, or any covariates (whether included in the model or not). The name ‘Pearson residuals’ relates to the fact that the sum of squares of the Pearson residuals gives the Pearson statistic discussed in section 2.1.7.

Note that the Pearson residuals are the residuals of the working linear model from the converged IRLS method, divided by the square roots of the converged IRLS weights.

Deviance Residuals

In practice the distribution of the Pearson residuals can be quite asymmetric around zero, so that their behaviour is not as close to ordinary linear model residuals as might be hoped for. The *deviance residuals* are often preferable in this respect. The deviance residuals are arrived at by noting that the deviance plays much the same role for GLMs that the residual sum of squares plays for ordinary linear models: indeed for an ordinary linear model the deviance *is* the residual sum of squares. In the ordinary linear model case, the deviance is made up of the sum of the squared residuals. That is the residuals are the square roots of the components of the deviance with the appropriate sign attached.

So, writing d_i as the component of the deviance contributed by the i^{th} datum (i.e. the i^{th} term in the summation in (2.7)) we have

$$D = \sum_{i=1}^n d_i$$

and, by analogy with the ordinary linear model, we can define

$$\hat{\epsilon}_i^d = \text{sign}(y_i - \hat{\mu}_i) \sqrt{d_i}.$$

As required the sum of squares of these ‘deviance residuals’ gives the deviance itself.

Now if the deviance were calculated for a model where all the parameters were known, then (2.8) would become $D^* \sim \chi_n^2$, and this might suggest that for a single datum $d_i \sim \chi_1^2$, implying that $\epsilon_i^d \sim N(0, 1)$. Of course (2.8) can not reasonably be applied to a single datum, but non the less it suggests that we might expect the deviance residuals to behave something like $N(0, 1)$ random variables, for a well fitting model, especially in cases for which (2.8) is expected to be a reasonable approximation.

2.1.10 Quasi-likelihood

The treatment of GLMs has so far assumed that the distribution of the response variable is a known member of the exponential family. If there is a good reason to suppose that the response follows a particular distribution then it is appealing to base models on that distribution, but in many cases the nature of the response distribution is not known so precisely, and it is only possible to specify what the relationship between the variance of the response and its mean should be. That is, the function $V(\mu)$ can be specified, but little more. The question then arises of whether it is possible to develop theory for fitting and inference with GLMs, starting from the position of specifying only the mean variance relationship.

It turns out that it is possible to develop satisfactory methods, based on the notion of **quasi-likelihood**. Consider an observation, y_i , of a random variable with mean μ_i and known variance function, $V(\mu_i)$. Then the log quasi likelihood for μ_i given y_i is

defined to be

$$q_i(\mu_i) = \int_{y_i}^{\mu_i} \frac{y_i - z}{\phi V(z)} dz. \quad (2.12)$$

As we will see, the key feature of this function is that it shares many useful properties of l_i , the log-likelihood corresponding to a single observation, but only requires knowledge of V rather than the full distribution of Y_i . Provided that the data are observations of independent random variables, we can define a log quasi likelihood for the mean vector, μ , of all the response data, or any parameter vector defining μ as

$$q(\mu) = \sum_{i=1}^n q_i(\mu_i).$$

The key property of q is that, for the purposes of inference with GLMs, it behaves in a very similar manner to the log-likelihood, but only requires knowledge of the variance function in order to define it.

Consider, for example, obtaining maximum quasi-likelihood parameter estimates of the GLM parameters β . Differentiating q w.r.t. β_j yields

$$\frac{\partial q}{\partial \beta_j} = \sum_{i=1}^n \frac{y_i - \mu_i}{\phi V(\mu_i)} \frac{\partial \mu_i}{\partial \beta_j},$$

so that the parameter estimates are solutions to the equations

$$\sum_{i=1}^n \frac{(y_i - \mu_i)}{V(\mu_i)} \frac{\partial \mu_i}{\partial \beta_j} = 0 \quad \forall j,$$

but this is exactly the system (2.3), that must be solved to find the m.l.e.s for a GLM. Hence the maximum quasi-likelihood parameter estimates can be found by the usual GLM IRLS method, which in any case only requires knowledge of $V(\mu)$.

Furthermore, the log-quasi likelihood shares just enough properties with the log-likelihood that the results on the large sample distribution of the parameter estimators, given in section 2.1.5, also hold for the maximum quasi-likelihood estimators of the parameters. Similarly, the large sample distributional results of section 2.1.6, underpinning hypothesis testing with GLMs, hold when the log-likelihood, l , is replaced by the log quasi-likelihood, q . The theoretical basis for these assertions is provided in section 2.4.8.

Note that the log quasi-likelihood of the saturated model is always zero, so the quasi-deviance of a GLM is simply

$$D_q = -2q(\hat{\mu})\phi.$$

Obviously the discussion of residuals and scale parameter estimation also carries over from the likelihood to the quasi-likelihood case, again with no more than the replacement of l by q .

The practical use of the quasi-likelihood approach requires that the integral in (2.12) be evaluated, but this is possible for most practically useful forms of V : McCullagh

and Nelder (1989) give examples, or in R you can type e.g.

```
quasi(variance="mu^3")$dev.resids
```

to access the form of q_i for any particular mean variance relationship there implemented. For mean variance relationships corresponding to an exponential family distribution from table 2.1, the form of the quasi-deviance corresponds exactly to the form of the deviance for that family.

One major practical use of quasi-likelihood is to provide a means of modelling count data that are more variable than the Poisson or binomial distributions (with their fixed scale parameters) predict: the quasi-likelihood approach assumes that ϕ is unknown. Such ‘over-dispersed’ data are common in practice. Another practical use is to provide a means of modelling data with a mean variance relationship for which there is no obvious exponential family distribution: for example continuous data for which the variance is expected to be proportional to the mean.

2.2 Geometry of GLMs

The geometry of GLMs and GLM fitting is less straightforward than the geometry of ordinary linear models, since the likelihood used to judge model fit does not generally mean that the fit can be judged by Euclidian distance between model and data. Figure 2.1 illustrates the geometric situation that prevails for GLMs, using the example of the fit to 3 data of a 2 parameter GLM with a Gamma distribution and a log link. The flat model subspace of section 1.4 is now replaced by a curved ‘model manifold’, consisting of all the possible fitted value vectors predictable by the model. Since Euclidean distance between model manifold and data is no longer the measure of fit being used then different means must be employed to illustrate the geometry of estimation. The black lines, in the right panel of figure 2.1, show all the combinations of the response variables, which give rise to the same estimated model. Notice how these lines are not generally parallel, and are not generally orthogonal to the model manifold.

To fully understand figure 2.1, it may help to consider what the figure would look like for some different 2 parameter models.

1. For an ordinary linear model, the model manifold would be a flat plane, to which all the lines of equal fit would be orthogonal (and hence parallel to each other).
2. For a GLM assuming a normal distribution (but non-identity link) the lines of equal fit would be orthogonal to the (tangent space of the) model manifold where they meet it.
3. For a 2 parameter fit to 4 data, the lines of equal fit would become planes of equal fit.

In general, the geometric picture presented in figure 2.1 applies to any GLM. With more data the lines of equal fit become $n - p$ dimensional planes of equal fit, where n and p are the number of data and parameters respectively: for any fixed β , equation (2.3) gives the restrictions on y defining such a plane. Note that these planes

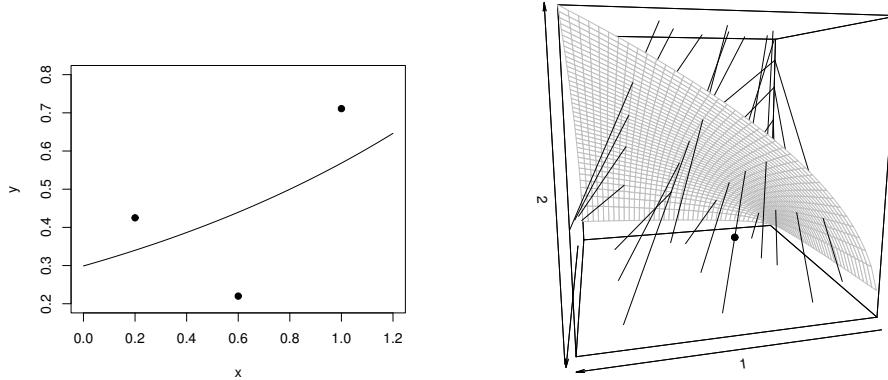


Figure 2.1 *The geometry of GLMs*. The left panel illustrates the best fit of the generalized linear model $\mathbb{E}(y) \equiv \mu = \exp(\beta_0 + \beta_1 x)$ to the three x, y data shown, assuming that each y_i is an observation of a Gamma distributed random variable with mean given by the model. The right panel illustrates the geometry of GLM fitting using this model as an example. The unit cube shown, represents a space within which the vector $(y_1, y_2, y_3)^T$ defines a single point, \bullet . The grey surface shows all possible predicted values (within the unit cube) according to the model, i.e. it represents all possible $(\mu_1, \mu_2, \mu_3)^T$ values. As the parameters β_0 and β_1 are allowed to vary, over all their possible values, this is the surface that the corresponding model ‘fitted values’ trace out: the ‘model manifold’. The continuous lines, which each start at one face of the cube and leave at another, are lines of equivalent fit: the values of the response data $(y_1, y_2, y_3)^T$ lying on such a line, each result in the same maximum likelihood estimates of β_0, β_1 and hence the same $(\mu_1, \mu_2, \mu_3)^T$. Notice how the equivalent fit lines are neither parallel to each other nor orthogonal to the model manifold.

can intersect — a point which will be returned to later. For discrete response data the pictures are no different, although the lines of equal fit strictly make sense only under continuous generalizations of the likelihoods (generally obtainable by replacing factorials by appropriate gamma functions in the probability functions). Only for the normal distribution are the lines/planes of equal fit orthogonal to the model manifold where-ever they meet it. For other distributions the lines/planes of equal fit may sometimes be parallel to each other, but are never all orthogonal to the model manifold.

2.2.1 The geometry of IRLS

The geometry of the IRLS estimation algorithm is most easily appreciated by considering the fit of a one parameter model to 2 response data. Figure 2.2 illustrates the geometry of such a model: in this case a GLM with a log link and Gamma errors,

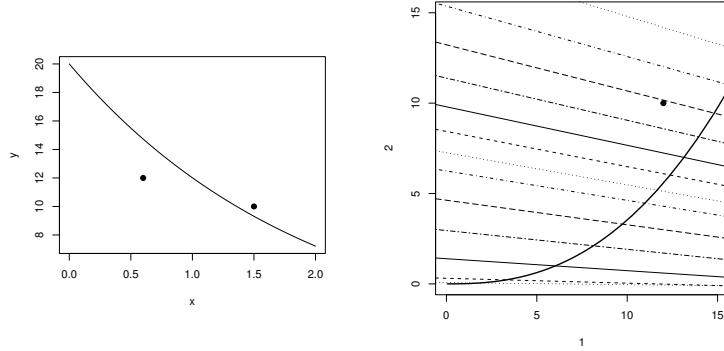


Figure 2.2 *Geometry of the GLM* $\mathbb{E}(y_i) \equiv \mu_i = 20 \exp(-\beta x_i)$ where $y_i \sim \text{Gamma}$ and $i = 1, 2$. The left panel illustrates the maximum likelihood estimate of the model (continuous line) fitted to the 2 x, y data shown as \bullet . The right panel illustrates the fitting geometry. The 15×15 square is part of the space \Re^2 in which (y_1, y_2) defines a single point, \bullet . The bold curve is the ‘model manifold’: it consists of all possible points (μ_1, μ_2) according to the model (i.e. as β varies (μ_1, μ_2) traces out this curve). The fine lines are examples of lines of equal fit. All points (y_1, y_2) lying on one of these lines share the same MLE of β and hence (μ_1, μ_2) : this MLE is where the equal fit line cuts the model manifold. The lines of equal fit are plotted for $\beta = .1, .2, .3, .4, .5, .6, .7, .8, .9, 1, 1.2, 1.5, 2, 3, 4$. ($\beta = .1, .7$ and 2 are represented by unbroken lines, with the $\beta = 2$ line being near the bottom of the plot. The $\beta = .1$ line is outside the plotting region in this plot, but appears in subsequent plots.)

but similar pictures can be constructed for a GLM with any combination of link and distributional assumption.

Now the key problems in fitting a GLM are that the model manifold is not flat, and that the lines of equal fit are not orthogonal to the model manifold where they meet it. The IRLS method linearly translates and rescales the fitting problem, so that at the current estimate of μ , the model manifold and intersecting line of equal fit are orthogonal, and, in the rescaled space, the location of the current estimate of μ is given by \mathbf{X} multiplied by the current β estimate. This rescaling results in a fitting problem that can be treated as locally linear, so that the β estimate can be updated by least squares.

Figure 2.3 illustrates how the IRLS steps involved in forming pseudodata and weighting it, effectively transform the fitting problem into one that can be approximately solved by linear least squares. The figure illustrates the transformations involved in one IRLS step, which are redone repeatedly, as the IRLS method is iterated to convergence.

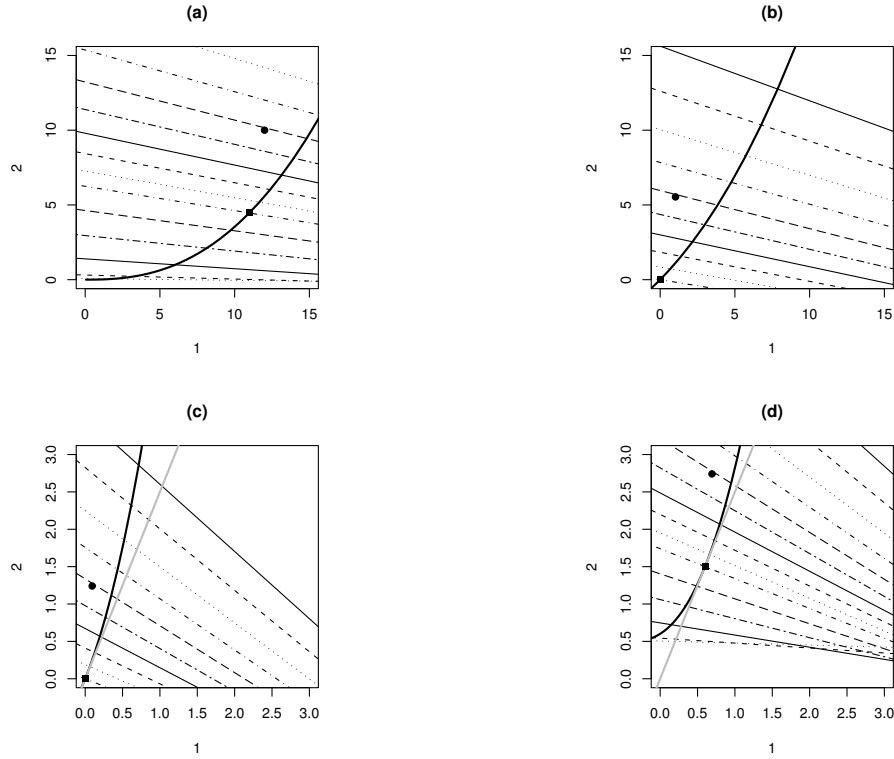


Figure 2.3 *Geometry of the IRLS estimation of a GLM, based on the example shown in figure 2.2.* (a) shows the geometry of the fitting problem — the model manifold is the thick black curve, the equal fit lines are the thin lines (as figure 2.2), the data are at \bullet and the current estimates of the fitted values, $\mu^{[k]}$, are at \blacksquare . (b) The problem is re-centred around the current fitted values (y_i is replaced by $y_i - \mu_i^{[k]}$). (c) The problem is linearly re-scaled so that the columns of \mathbf{X} now span the tangent space to the model manifold at \blacksquare . The tangent space is illustrated by the grey line (this step replaces $y_i - \mu_i^{[k]}$ by $g'(\mu_i^{[k]})(y_i - \mu_i^{[k]})$). (d) The problem is linearly translated so that the location of \blacksquare is now given by $\mathbf{X}\beta^{[k]}$. For most GLMs the problem would now have to be rescaled again by multiplying the components relative to each axis by $\sqrt{W_i}$, where the W_i are the iterative weights: this would ensure that the equal estimate line through \blacksquare , is orthogonal to the tangent space. In the current example these weights are all 1, so that the required orthogonality already holds. Now for the transformed problem, in the vicinity of \blacksquare , the model manifold can be approximated by the tangent space, to which the equal fit lines are approximately orthogonal: hence an updated estimate of μ and β can be obtained by finding the least squares projection of the transformed data, \bullet , onto the tangent space (grey line).

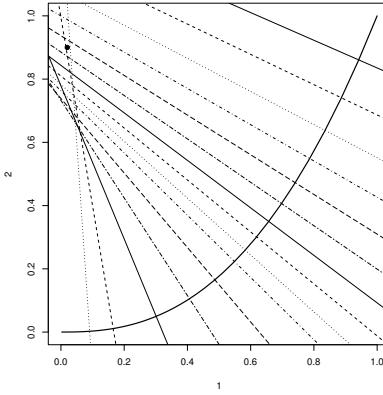


Figure 2.4 *Geometry of fitting and convergence problems.* The geometry of a 1 parameter GLM with a log link and normal errors is illustrated. The thick curve is the model manifold — within the unit square, it contains all the possible fitted values of the data, according to the model. The thin lines are the equal fit lines (levels as in figure 2.2). Notice how the lines of equal fit meet and cross each other at the top left of the plot. Data in this overlap region will yield model likelihoods having local minima at more than one parameter value. Consideration of the operation of the IRLS fitting method reveals that, in this situation, it may converge to different estimates depending on the initial values used to start the fitting process. • illustrates the location of a problematic response vector, used to illustrate non-unique convergence in the text.

2.2.2 Geometry and IRLS convergence

Figure 2.4 illustrates the geometry of fitting a model, $\mathbb{E}(y_i) \equiv \mu_i = \exp(-\beta x_i)$, where the y_i are normally distributed and there are two data, y_i , to fit, for which $x_1 = .6$ and $x_2 = 1.5$. As in the previous two sections, lines of equal fit are shown on a plot in which a response vector $(y_1, y_2)^\top$ would define a single point and the set of all possible fitted values $(\mu_1, \mu_2)^\top$, according to the model, is shown as a thick curve. In this example, the lines of equal fit intersect and cross in the top left hand corner of the plot (corresponding to very poor model fit). This crossing is problematic: in particular, the results of IRLS fitting to data lying in the upper left corner will depend on the initial parameter estimate from which the IRLS process is started, since each such data point lies on the intersection of two equal fit lines. If the IRLS iteration is started from fitted values in the top right of the plot then fitted values nearer the top right will be estimated, while starting the iteration with fitted values at the bottom left of the plot will result in estimated fitted values that are different, and closer to the bottom left of the plot.

That this indeed happens, in practice, is easily demonstrated in R, by fitting to the data $y_1 = .02$, $y_2 = .9$, illustrated as • in figure 2.4.

```
> ms<-exp(-x*4)      # set initial values at lower left
```

```
> glm(y~X-1,family=gaussian(link=log),mustart=ms)
Coefficients:
5.618
Residual Deviance: 0.8098      AIC: 7.868
> ms <- exp(-x*0.1) # set initial values at upper right
> glm(y~X-1,family=gaussian(link=log),mustart=ms)
Coefficients:
0.544
Residual Deviance: 0.7017      AIC: 7.581
```

Notice that the second fit here actually has higher likelihood (lower deviance) — the fits are not equivalent in terms of likelihood. The type of fitting geometry that gives rise to these ambiguities does not always occur: for example some models have parallel lines/planes of equal fit, but for any model with intersecting lines/planes of equal fit there is some scope for ambiguity. Fortunately, if the model is a good model, it is often the case that data lying in the region of ambiguity is rather improbable. In the example in figure 2.4, the problematic region consists entirely of data that the model can only fit very poorly. It follows that very poor models of data may yield estimation problems of this sort: but it is not uncommon for very poor models to be a feature of early attempts to model any complex set of data. If such problems are encountered then it can be better to proceed by linear modelling of transformed response data, until good enough candidate models have been identified to switch back to GLMs.

Of course, if reasonable starting values are chosen, then the ambiguity in the fitting process is unlikely to cause major problems when fitting GLMs: the algorithm will converge to one of the local minima of the likelihood, after all. However the ambiguity can cause more serious convergence problems for GAM estimation by “performance iteration”, when it becomes possible to cycle between alternative minima without ever converging.

2.3 GLMs with R

The `glm` function provides the means for using GLMs in R. Its use is similar to that of the `lm` function but with two differences. The right hand side of the model formula, specifying the form for the linear predictor, now gives the link function of the mean of the response, rather than the mean of the response directly. Also `glm` takes a `family` argument, which is used to specify the distribution from the exponential family to use, and the link function that is to go with it. In this section the use of the `glm` function with a variety of simple GLMs will be presented, to illustrate the wide variety of model structures that the GLM encompasses.

2.3.1 Binomial models and heart disease

Early diagnosis of heart attack is important if the best care is to be given to patients. One suggested diagnostic aid is the level of the enzyme creatinine kinase (CK) in

CK value	Patients with Heart attack	Patients without heart attack
20	2	88
60	13	26
100	30	8
140	30	5
180	21	0
220	19	1
260	18	1
300	13	1
340	19	1
380	15	0
420	7	0
460	8	0

Table 2.2 *Data (from Hand et al., 1994) on heart attack probability as a function of CK level.*

the blood stream. A study was conducted (Smith, 1967) in which the level of CK was measured for 360 patients suspected of suffering from a heart attack. Whether or not each patient had really suffered a heart attack was established later, after more prolonged medical investigation. The data are given in table 2.2. The original paper classified patients according to ranges of CK level, but in the table only midpoints of the range have been given.

It would be good to be able to base diagnostic criteria on data like these, so that CK level can be used to estimate the probability that a patient has had a heart attack. We can go some way towards such a goal, by constructing a model which tries to explain the proportion of patients suffering a heart attack, from the CK levels. In the following the data were read into a data.frame called `heart`. It contains variables `ha`, `ok` and `ck`, giving numbers of patients who subsequently turned out to have had, or not to have had, heart attacks, at each CK level. It makes sense to plot the observed proportions against CK level first.

```
p<-heart$ha/ (heart$ha+heart$ok)
plot (heart$ck,p,xlab="Creatinine kinase level",
      lab="Proportion Heart Attack")
```

The resulting plot is figure 2.5.

A particularly convenient model for describing these proportions is

$$\mathbb{E}(p_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}},$$

where p_i is the proportion with heart attacks at CK level x_i . This curve is sigmoid in

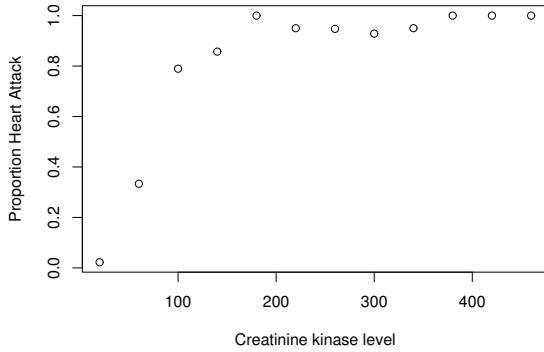


Figure 2.5 *Observed proportion of patients subsequently diagnosed as having had a heart attack, against CK level at admittance.*

shape, and bounded between 0 and 1. (Obviously the heart data do not show the lower tail of this proposed sigmoid curve.) This means that the expected number of heart attack sufferers is given by

$$\mu_i \equiv \mathbb{E}(p_i N_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} N_i,$$

where N_i is the known total number of patients at each CK level. This model is somewhat non-linear in its parameters, but if the ‘logit’ link,

$$g(\mu_i) = \log \left(\frac{\mu_i}{N_i - \mu_i} \right),$$

is applied to it we obtain

$$g(\mu_i) = \beta_0 + \beta_1 x_i,$$

the r.h.s. of which is linear in the model parameters. The logit link is the canonical link for binomial models, and hence the default in R.

In R there are two ways of specifying binomial models with `glm`.

1. The response variable can be the observed proportion of successful binomial trials, in which case an array giving the number of trials must be supplied as the `weights` argument to `glm`. For binary data, no weights vector need be supplied, as the default weights of 1 suffice.
2. The response variable can be supplied as a two column array, in which the first column gives the number of binomial ‘successes’, and the second column is the number of binomial ‘failures’.

For the current example the second method will be used. Supplying 2 arrays of the

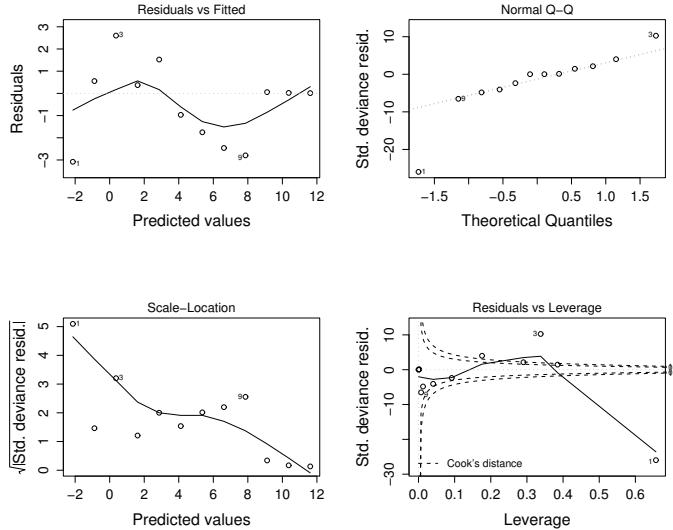


Figure 2.6 *Model checking plots for the first attempt to fit the CK data.*

r.h.s. of the model formula involves using `cbind`. Here is a `glm` call which will fit the heart attack model:

```
> mod.0<-glm(cbind(ha,ok) ~ ck, family=binomial(link=logit),
+ data=heart)
```

or we could have used

```
mod.0<-glm(cbind(ha,ok) ~ ck, family=binomial, data=heart)
```

since the logit link is canonical for the binomial and hence the R default. Here is the default information printed about the model:

```
> mod.0

Call: glm(formula=cbind(ha,ok) ~ ck, family=binomial, data=heart)

Coefficients:
(Intercept)          ck
-2.75834        0.03124

Degrees of Freedom: 11 Total (i.e. Null); 10 Residual
Null Deviance:    271.7
Residual Deviance: 36.93      AIC: 62.33
```

The Null deviance is the deviance for a model with just a constant term, while

the Residual deviance is the deviance of the fitted model (and also the scaled deviance in the case of a binomial model). These can be combined to give the *proportion deviance explained*, a generalization of r^2 , as follows:

```
> (271.7-36.93)/271.7
[1] 0.864078
```

AIC is the Akaike Information Criteria for the model, discussed in sections 2.1.4 and 2.4.7 (it could also have been extracted using `AIC(mod.)`).

Notice that the deviance is quite high for the χ^2_{10} random variable that it should approximate if the model is fitting well. In fact

```
> 1-pchisq(36.93,10)
[1] 5.819325e-05
```

shows that there is a very small probability of a χ^2_{10} random variable being as large as 36.93. The residual plots (shown in figure 2.6) also suggest a poor fit.

```
> op<-par(mfrow=c(2,2))
> plot(mod.0)
```

The plots have the same interpretation as the model checking plots for an ordinary linear model, discussed in detail in section 1.5.1, except that it is now the deviance residuals that are plotted, the Predicted values are on the scale of the linear predictor rather than the response, and some departure from a straight line relationship in the Normal QQ plot is often to be expected. The plots are not easy to interpret when there are so few data, but there appears to be a trend in the mean of the residuals plotted against fitted value, which would cause concern. Furthermore, the first point has very high influence. Note that the interpretation of the residuals would be much more difficult for binary data: exercise 2 explores simple approaches that can be taken in the binary case.

Notice how the problems do not stand out so clearly from a plot of the fitted values overlayed on the raw estimated probabilities (see figure 2.7):

```
> plot(heart$ck,p,xlab="Creatinine kinase level",
+ ylab="Proportion Heart Attack")
> lines(heart$ck,fitted(mod.0))
```

Note also that the fitted values provided by `glm` for binomial models are the estimated p_i 's, rather than the estimated μ_i 's.

The residual plots suggest trying a cubic linear predictor, rather than the initial straight line.

```
> mod.2<-glm(cbind(ha,ok) ~ ck+I(ck^2)+I(ck^3),family=binomial,
+ data=heart)
> mod.2
```

```
Call: glm(formula=cbind(ha,ok) ~ ck+I(ck^2)+I(ck^3),
```

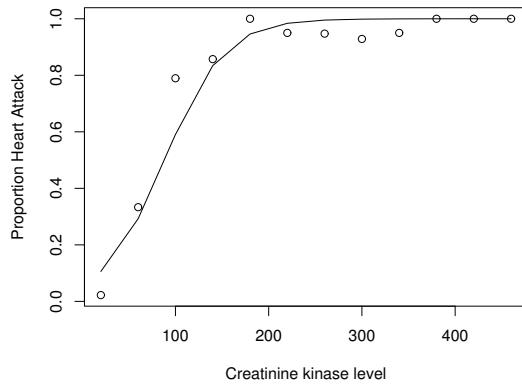


Figure 2.7 Predicted and observed probability of heart attack against CK level.

```

family=binomial,data=heart)

Coefficients:
(Intercept)          ck      I(ck^2)      I(ck^3)
-5.786e+00  1.102e-01 -4.648e-04  6.448e-07

Degrees of Freedom: 11 Total (i.e. Null);  8 Residual
Null Deviance:    271.7
Residual Deviance: 4.252      AIC: 33.66
> par(mfrow=c(2,2))
> plot(mod.2)

```

Clearly 4.252 is not too large for consistency with a χ^2_8 distribution (it is less than the expected value, in fact) and the AIC has improved substantially. The residual plots (figure 2.8) now show less clear patterns than for the previous model, although if we had more data then such a departure from constant variance would be a cause for concern. Furthermore the fit is clearly closer to the data now (see figure 2.9):

```

par(mfrow=c(1,1))
plot(heart$ck,p,xlab="Creatinine kinase level",
      ylab="Proportion Heart Attack")
lines(heart$ck,fitted(mod.2))

```

We can also get R to test the null hypothesis that `mod.0` is correct against the alternative that `mod.2` is required. Somewhat confusingly the `anova` function is used to do this, although it is an analysis of **deviance** (i.e. a generalized likelihood ratio test) that is being performed, and not an analysis of variance.

```

> anova(mod.0,mod.2,test="Chisq")
Analysis of Deviance Table

```

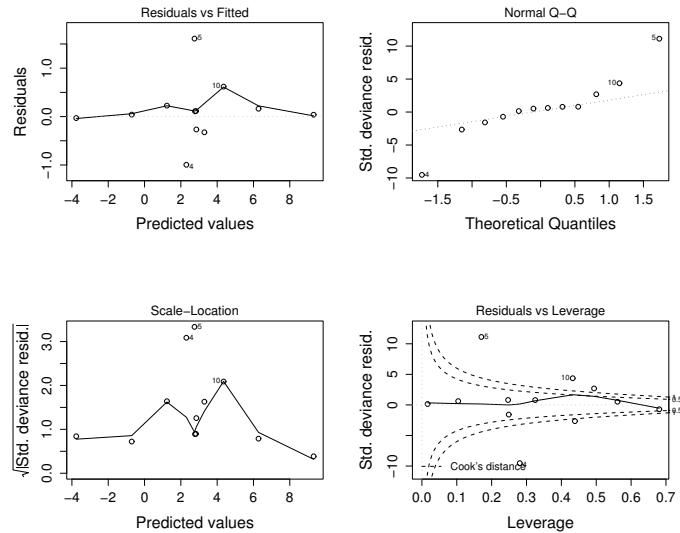


Figure 2.8 *Model checking plots for the second attempt to fit the CK data.*

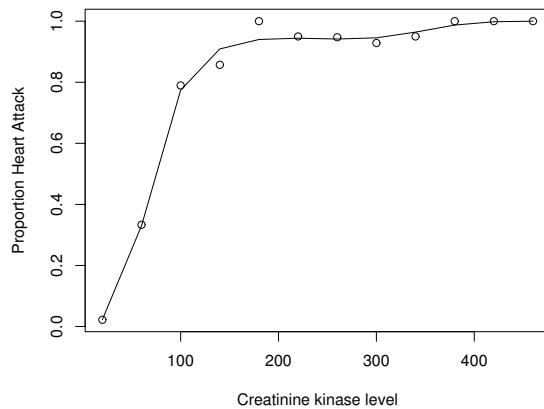


Figure 2.9 *Predicted and observed probability of heart attack against CK level.*

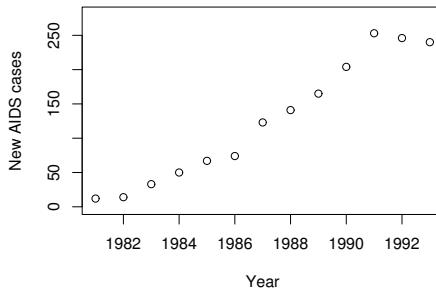


Figure 2.10 AIDS cases per year in Belgium

```

Model 1: cbind(ha, ok) ~ ck
Model 2: cbind(ha, ok) ~ ck + I(ck^2) + I(ck^3)
      Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1          10     36.929
2          8      4.252  2    32.676 8.025e-08

```

A p-value this low indicates very strong evidence against the null hypothesis - we really do need model 2. Recall that this comparison of models has a much firmer theoretical basis than the examination of the individual deviances had.

2.3.2 A Poisson regression epidemic model

The introduction to this chapter included a simple model for the early stages of an epidemic. Venables and Ripley (2003) provide some data on the number of new AIDS cases each year, in Belgium, from 1981 onwards. The data can be entered into R and plotted as follows.

```

y<- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t<-1:13
plot (t+1980,y,xlab="Year",ylab="New AIDS cases",ylim=c(0,280))

```

Figure 2.10 shows the resulting plot. The scientifically interesting question, relating to such data, is whether they provide any evidence that the increase in the underlying rate of new case generation is slowing. The simple model from the introduction might provide a plausible model from which to start investigating this question. The model assumes that the underlying expected number of cases per year, μ_i , increases according to:

$$\mu_i = c \exp(bt_i)$$

where c and b are unknown parameters, and t_i is time in years since the start of the

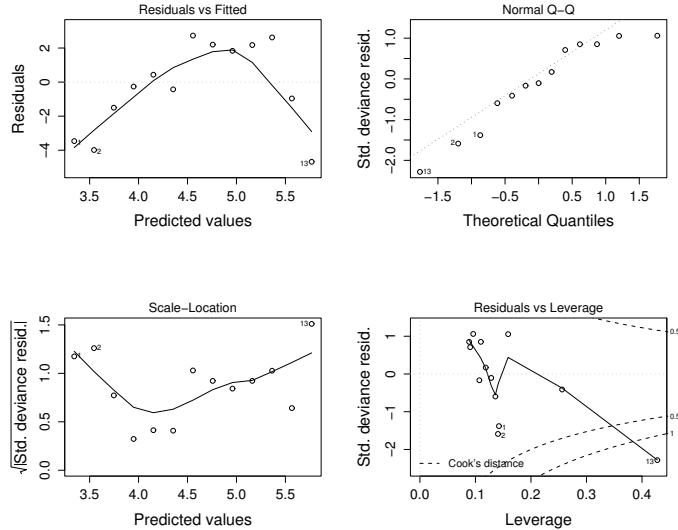


Figure 2.11 *Residual plots for m0 fitted to the AIDS data.*

data. A log link turns this into a GLM,

$$\log(\mu_i) = \log(c) + bt_i = \beta_0 + t_i\beta_1,$$

and we assume that $y_i \sim \text{Poi}(\mu_i)$ where y_i is the observed number of new cases in year t_i . The y_i are assumed independent. This is essentially a model of unchecked spread of the disease.

The following fits the model (the log link is canonical for the Poisson distribution, and hence the R default) and checks it.

```
> m0 <- glm(y ~ t, poisson)
> m0

Call: glm(formula = y ~ t, family = poisson)

Coefficients:
(Intercept)          t
            3.1406     0.2021

Degrees of Freedom: 12 Total (i.e. Null); 11 Residual
Null Deviance:      872.2
Residual Deviance:  80.69      AIC: 166.4
> par(mfrow=c(2,2))
> plot(m0)
```

The deviance is very high for the observation of a χ^2_{11} random variable that it ought

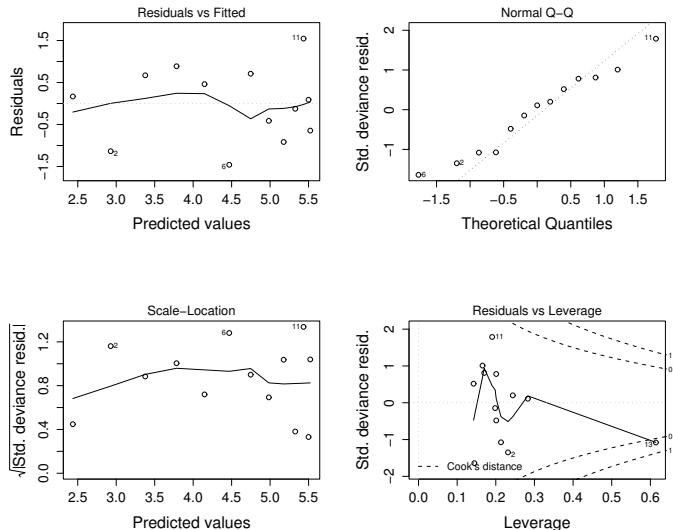


Figure 2.12 *Residual plots for m1 fitted to the AIDS data.*

to approximate, if the model is a good fit. The residual plots shown in figure 2.11 are also worrying. In particular the clear pattern in the mean of the residuals, plotted against the fitted values, shows violation of the independence assumption, and probably results from omission of something important from the model. Since, for this model, the fitted values increase monotonically with time — i.e. it appears that a quadratic term in time could usefully be added to the model. The very high influence of the final year's data, evident in the residuals versus leverage plot, is also worrying. Note that the interpretation of residual plots can become difficult if the Poisson mean is low, so that the data are mostly zeroes and ones. In such cases the simulation approaches covered in exercise 2 can prove useful, if adapted to the Poisson case.

It seems sensible to amend the model by adding a quadratic term to obtain:

$$\mu_i = \exp(\beta_0 + \beta_1 t_i + \beta_2 t_i^2).$$

This model allows situations other than unrestricted spread of the disease to be represented. The following fits and checks it:

```
> m1 <- glm(y~t+I(t^2),poisson)
> plot(m1)
> summary(m1)

Call:
glm(formula = y ~ t + I(t^2), family = poisson)
```

```

Deviance Residuals:
    Min      1Q   Median      3Q     Max
-1.45903 -0.64491  0.08927  0.67117  1.54596

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.901459  0.186877 10.175 < 2e-16 ***
t           0.556003  0.045780 12.145 < 2e-16 ***
I(t^2)      -0.021346  0.002659 -8.029 9.82e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 872.2058 on 12 degrees of freedom
Residual deviance: 9.2402 on 10 degrees of freedom
AIC: 96.924

Number of Fisher Scoring iterations: 4

```

Notice how the residual plots shown in figure 2.12 are now much improved: the clear trend in the mean has gone, the (vertical) spread of the residuals is reasonably even, the influence of point 13 is much reduced and the QQ plot is straighter. The fuller model summary shown for this model also indicates improvement: the deviance is now quite reasonable, i.e. close to what is expected for a χ^2_{10} r.v., and the AIC has dropped massively. All in all this model appears to be quite reasonable.

Notice also, how the structure of the `glm` summary is similar to an `lm` summary. The standard errors and p-values in the table of coefficient estimates is now based on the large sample distribution of the parameter estimators given in section 2.1.5. The `z value` column simply reports the parameter estimates divided by their estimated standard deviations. Since no dispersion parameter estimate is required for the Poisson, these z-values should be observations of $N(0,1)$ r.v.s, if the true value of the corresponding parameter is zero (at least in the large sample limit), and the reported p-value is based on this distributional approximation. For `mod.1` the reported p-values are very low: i.e for each parameter there is clear evidence that it is not zero. Note that `Fisher Scoring iterations` are another name for IRLS iterations in the GLM context.

Examination of the coefficient summary table indicates that the hypothesis that $\beta_2 = 0$ can be firmly rejected, providing clear evidence that `mod.1` is preferable to `mod.0`. The same question can also be addressed using a generalized likelihood ratio test:

```

> anova(m0,m1,test="Chisq")
Analysis of Deviance Table

Model 1: y ~ t
Model 2: y ~ t + I(t^2)

```

	Resid.	Df	Resid.	Dev	Df	Deviance	P(> Chi)
1		11		80.686			
2		10		9.240	1	71.446	2.849e-17

The conclusion is the same as before: the tiny p-value indicates that `mod.0` should be firmly rejected in favour of `mod.1`. Notice that the p-value from the summary and the analysis of deviance table are different, since they are based on fundamentally different approximate distributional results. The `test="Chisq"` argument to `anova` is justified because the scale parameter is known for this model, had it been estimated it would be preferable to set `test` to "F".

The hypothesis testing approach to model selection is appropriate here, as the main question of interest is whether there is evidence, from these data, that the epidemic is spreading un-checked, or not. It would be prudent not to declare that things are improving if the evidence is not quite firm that this is true. If we had been more interested in simply finding the best model for predicting the data then comparison of AIC would be more appropriate, but leads to the same conclusion for these data.

The parameter β_1 can be interpreted as the rate of spread of the disease at the epidemic start: that is, as a sort of intrinsic rate of increase of the disease in a new population where no control measures are in place. Notice how the estimate of this parameter has actually increased substantially between the first and second models: it would have been possible to be quite badly misled if we had stuck with the first poorly fitting model. An approximate confidence interval for β_1 can be obtained in the usual manner, based on the large sample results from section 2.1.5. The required estimate and standard error are easily extracted using the `summary` function, as the following illustrates:

```
> beta.1 <- summary(m1)$coefficients[2,]
> ci <- c(beta.1[1]-1.96*beta.1[2],beta.1[1]+1.96*beta.1[2])
> ci # print 95% CI for beta_1
0.4662750 0.6457316
```

The use of the critical points of the standard normal distribution is appropriate, because the scale parameter is known for this model. Had it been estimated, then we would have had to use critical points from the *t* distribution, with degrees of freedom set to the residual degrees of freedom of the model (i.e number of data less number of estimated β parameters).

Another obvious thing to want to do, is to use the model to find a confidence interval for the underlying rate of case generation at any time. The following R code illustrates how to use the `predict.glm` function to find CI's for the underlying rate, over the whole period of the data, and plot these.

```
new.t<-seq(1,13,length=100)
fv <- predict(m1,data.frame(t=new.t),se=TRUE)
plot(t+1980,y,xlab="Year",ylab="New AIDS cases",ylim=c(0,280))
lines(new.t+1980,exp(fv$fit))
lines(new.t+1980,exp(fv$fit+2*fv$se.fit),lty=2)
lines(new.t+1980,exp(fv$fit-2*fv$se.fit),lty=2)
```

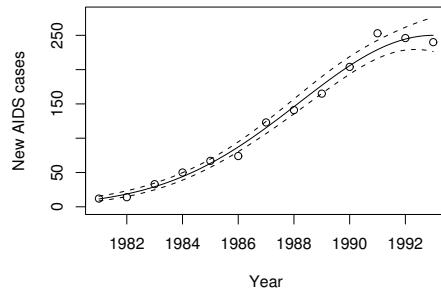


Figure 2.13 *Underlying AIDS case rate according to model m1 shown as a continuous curve with 95% confidence limits shown as dashed curves.*

The plot is shown in figure 2.13. Notice that by default the `predict.glm` function predicts on the scale of the linear predictor: we have to apply the inverse of the link function to get back onto the original response scale.

So the data provide quite firm evidence to suggest that the unfettered exponential increase model is overly pessimistic: by the end of the data there is good evidence that the rate of increase is slowing. Of course this model contains no mechanistic content — it says nothing about how or why the slowing might be occurring: as such it is entirely in-appropriate for prediction beyond the range of the data. The model allows us to be reasonably confident that the apparent slowing in the rate of increase in new cases is real, and not just the result of chance variation, but it says little or nothing about what may happen later.

2.3.3 Log-linear models for categorical data

The following table classifies a sample of women and men according to their belief in the afterlife:

	Believer	Non-Believer
Female	435	147
Male	375	134

The data (reported in Agresti, 1996) come from the US General Social Survey (1991), and the ‘non-believer’ category includes ‘undecideds’. Are there differences between males and females in the holding of this belief? We can address this question by using analysis of deviance to compare the fit of 2 competing models of these data: one in which belief is modelled as independent of gender, and a second in which there is some interaction between belief and gender. First consider the model of indepen-

dence. If y_i is an observation of the counts in one of the cells of the table, then we could model the expected number of counts as

$$\mu_i \equiv \mathbb{E}(Y_i) = n\gamma_k\alpha_j \text{ if } y_i \text{ is data for gender } k, \text{ and faith } j$$

where n is the total number of people surveyed, α_1 the proportion of believers, α_2 the proportion of non-believers and γ_1 and γ_2 the proportions of women and men respectively. Taking logs of this model yields

$$\eta_i \equiv \log(\mu_i) = \log(n) + \log(\gamma_k) + \log(\alpha_j).$$

So defining $\tilde{n} = \log(n)$, $\tilde{\gamma}_k = \log(\gamma_k)$ and $\tilde{\alpha}_j = \log(\alpha_j)$ the model can be written as

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{n} \\ \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \end{bmatrix}$$

This is clearly a GLM structure, but is obviously not identifiable. Dropping $\tilde{\gamma}_1$ and $\tilde{\alpha}_1$ solves the identifiability problem yielding

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{n} \\ \tilde{\gamma}_2 \\ \tilde{\alpha}_2 \end{bmatrix}.$$

Note how gender and faith are both factor variables with two levels in this model.

If the counts in the contingency table occurred independently at random, then the obvious distribution to use would be Poisson. In fact even when the total number of subjects in the table, or even some other marginal totals are fixed, then it can be shown that the correct likelihood can be written as a product of Poisson p.m.f.s, conditional on the various fixed quantities. Hence provided that the fitted model is forced to match the fixed total, and any fixed marginal totals, the Poisson is still the distribution to use. As was shown in section 2.1.8, forcing the model to match certain fixed totals in the data is simply a matter of insisting on certain terms being retained in the model.

The simple ‘independence’ model is easily estimated in R. First enter the data and check it:

```
> al<-data.frame(y=c(435,147,375,134),
+ gender=as.factor(c("F","F","M","M")),
+ faith=as.factor(c(1,0,1,0)))
> al
   y gender faith
1 435      F     1
2 147      F     0
3 375      M     1
4 134      M     0
```

Since gender and faith are both factor variables, model specification is very easy. The following fits the model and checks that the model matrix is as expected:

```
> mod.0<-glm(y~gender+faith,data=al,family=poisson)
> model.matrix(mod.0)
  (Intercept) genderM faith1
1             1         0       1
2             1         0       0
3             1         1       1
4             1         1       0
```

Now look at the fitted model object mod.0

```
> mod.0

Call: glm(formula=y~gender+faith,family=poisson,data=al)

Coefficients:
(Intercept)      genderM        faith1
      5.0100      -0.1340       1.0587

Degrees of Freedom: 3 Total (i.e. Null);  1 Residual
Null Deviance:    272.7
Residual Deviance: 0.162          AIC: 35.41

> fitted(mod.0)
     1      2      3      4
432.099 149.901 377.901 131.099
```

The fit appears to be quite close, and it would be somewhat surprising if a model with interactions between faith and gender did significantly better. Never the less such a model could be:

$$\eta_i \equiv \log(\mu_i) = \tilde{n} + \tilde{\gamma}_k + \tilde{\alpha}_j + \tilde{\zeta}_{kj} \text{ if } y_i \text{ is data for gender } k \text{ and belief } j$$

where ζ_{kj} is an ‘interaction parameter’. This model allows each combination of faith and gender to vary independently. As written, the model has rather a large number of un-identifiable terms.

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{n} \\ \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \\ \tilde{\zeta}_{11} \\ \tilde{\zeta}_{12} \\ \tilde{\zeta}_{21} \\ \tilde{\zeta}_{22} \end{bmatrix}.$$

But this is easily reduced to something identifiable:

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\gamma}_2 \\ \tilde{\alpha}_2 \\ \tilde{\zeta}_{22} \end{bmatrix}.$$

The following fits the model, checks the model matrix and prints the fitted model object:

```
> mod.1<-glm(y~gender*faith,data=al,family=poisson)
> model.matrix(mod.1)
  (Intercept) genderM faith1 genderM:faith1
1             1       0       1           0
2             1       0       0           0
3             1       1       1           1
4             1       1       0           0
> mod.1

Call: glm(formula=y~gender*faith,family=poisson,data=al)

Coefficients:
  (Intercept)      genderM      faith1   genderM:faith1
        4.99043     -0.09259      1.08491     -0.05583

Degrees of Freedom: 3 Total (i.e. Null);  0 Residual
Null Deviance:    272.7
Residual Deviance: 9.659e-14   AIC: 37.25
```

To test whether there is evidence for an interaction between gender and faith the null hypothesis that mod.0 is correct is tested against the more general alternative that mod.1 is correct, using analysis of deviance.

```
> anova(mod.0,mod.1,test="Chisq")
Analysis of Deviance Table

Model 1: y ~ gender + faith
Model 2: y ~ gender * faith
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1            1     0.16200
2            0  9.659e-14  1   0.16200  0.68733
```

A p-value of 0.69 suggests that there is no evidence to reject model 0 and the hypothesis of no association between gender and belief in the afterlife.

Notice that, in fact, the model with the interaction is the saturated model, which is why its deviance is numerically zero, and there was not really any need to fit it and compare it with the independence model explicitly — in this case we could just as well have examined the deviance of the independence model. However the general approach taken for this simple 2 way contingency table can easily be generalized to

multi-way tables and to arbitrary number of groups. In other words, the approach outlined here can be extended to produce a rather general approach for analyzing categorical data using log-linear GLMs.

Finally, note that the fitted values for `mod.0` had the odd property that although the fitted values and original data are different, the total number of men and women is conserved between data and fitted values, as is the total number of believers and non-believers. This results from the fact that the log link is canonical for the Poisson distribution, so by the results of section 2.1.8 $\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \hat{\boldsymbol{\mu}}$. The summations equated on the two sides of this last equation are the total number of subjects, the total number of males and the total number of believers: this explains the match between fitted values and data in respect of these totals.

2.3.4 Sole eggs in the Bristol channel

Fish stock assessment is difficult because adult fish are not easy to survey: they tend to actively avoid fishing gear, so that turning number caught into an assessment of the number in the sea is rather difficult. To get around this problem, fisheries biologists sometimes try and count fish eggs, and work back to the number of adult fish required to produce the estimated egg population. These ‘egg production methods’ are appealing because eggs are straightforward to sample. This section concerns a simple attempt to model data on sole eggs in the Bristol channel. The data (available in Dixon, 2003) are measurements of density of eggs per square metre of sea surface in each of 4 identifiable egg developmental stages, at each of a number of sampling stations in the Bristol channel on the west coast of England. The samples were taken during 5 cruises spaced out over the spawning season. Figure 2.14 shows the survey locations and egg densities for stage I eggs for each of the 5 surveys. Similar plots could be produced for stages II-IV. For further information on this stock, see Horwood (1993) and Horwood and Walker (1990).

The biologists’ chief interest is in estimating the rate at which eggs are spawned at any time and place within the survey arena, so this is the quantity that needs to be estimated from the data. To this end it helps that the durations of the egg stages are known (they vary somewhat with temperature, but temperature is known for each sample). Basic demography suggests that a reasonable model for the density of eggs (per day per square metre of sea surface), at any age a and location-time with covariates \mathbf{x} , would be

$$d(a, \mathbf{x}) = S(\mathbf{x}) e^{-\delta(\mathbf{x})a}.$$

That is, the density of eggs of age a is given by the product of the local spawning rate S and the local survival rate. δ is the per capita mortality rate, and, given this rate, we expect a proportion $\exp(-\delta a)$ of eggs to reach age a . Both S and δ are assumed to be functions of some covariates.

What we actually observe are not egg densities per unit age, per m^2 sea surface, but egg densities *in stages* per m^2 sea surface: y_i , say. To relate the model to the data we need to integrate the model egg density over the age range to which any particular

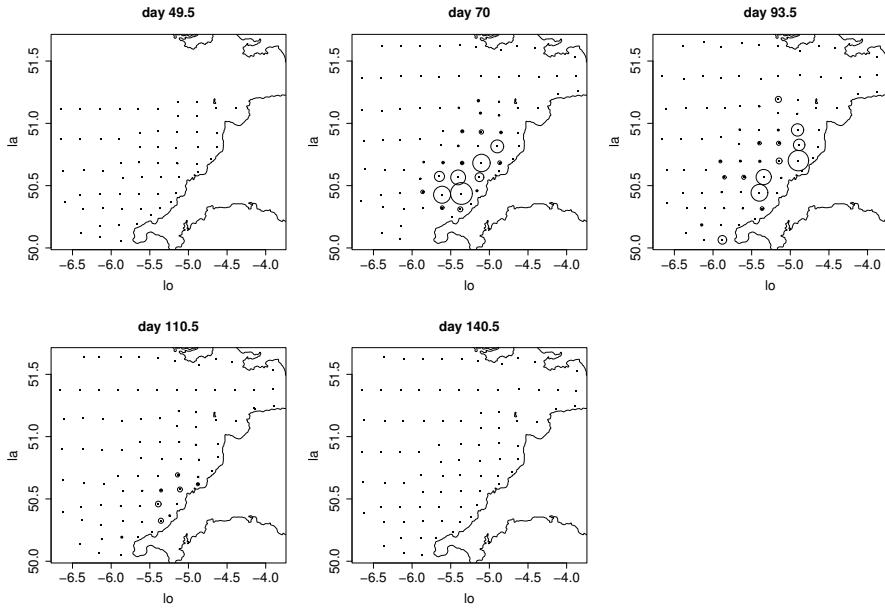


Figure 2.14 Density per m^2 sea surface of stage I sole eggs in the Bristol channel. The days given are the Julian day of the survey midpoint (day 1 is January 1). The symbol sizes are proportional to egg density and a simple dot indicates a station where no eggs were found.

datum relates. That is, if a_i^- and a_i^+ are the lower and upper age limits for the egg stage to which y_i relates, then the model should be

$$\mathbb{E}(y_i) \equiv \mu_i = \int_{a_i^-}^{a_i^+} d(z, \mathbf{x}_i) dz.$$

Evaluation of the integral would be straightforward, but does not enable the model to be expressed in the form of a GLM. However, if the integral is approximated so that the model becomes

$$\mu_i = \Delta_i d(\bar{a}_i, \mathbf{x}_i)),$$

where $\Delta_i = a_i^+ - a_i^-$ and $\bar{a}_i = (a_i^+ + a_i^-)/2$, then progress can be made, since in that case

$$\log(\mu_i) = \log(\Delta_i) + \log(S(\mathbf{x})) - \delta(\mathbf{x})\bar{a}_i. \quad (2.13)$$

The right hand side of this model can be expressed as the linear predictor of a GLM, with terms representing $\log(S)$ and δ as functions of covariates and with $\log(\Delta)$ treated as an ‘offset’ term — essentially a column of the model matrix with associated parameter fixed at 1.

For the sole eggs, a reasonable starting model might represent $\log(S)$ as a cubic function of longitude, lo , latitude, la , and time, t . Mortality might be modelled by

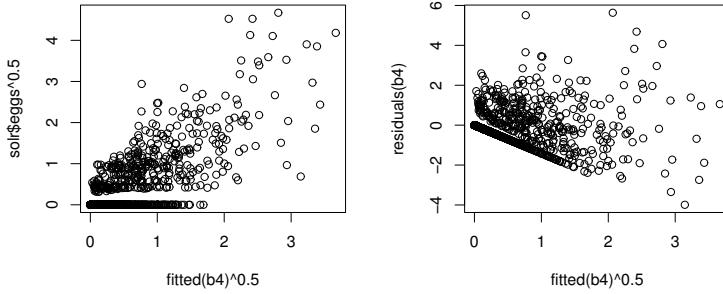


Figure 2.15 *Residual plots for the final Sole egg model.*

a simpler function — say a quadratic in t . It remains only to decide on a distributional assumption. The eggs are sampled by hauling a net vertically through the water and counting the number of eggs caught in it. This might suggest a Poisson model, but most such data display overdispersion relative to Poisson, and additionally, the data are not available as raw counts but rather as densities per m^2 sea surface. These considerations suggest using quasi-likelihood, with the variance proportional to the mean.

The following R code takes the `sole` data frame, calculates the mean ages and offset terms required, and fits the suggested model. Since polynomial models can lead to numerical stability problems, if not handled carefully, the covariates are all translated and scaled before fitting.

```
> sole$off <- log(sole$a.1-sole$a.0) # model offset term
> sole$a<-(sole$a.1+sole$a.0)/2      # mean stage age
> solr<-sole                         # make copy for rescaling
> solr$t<-solr$t-mean(sole$t)
> solr$t<-solr$t/var(sole$t)^0.5
> solr$la<-solr$la-mean(sole$la)
> solr$lo<-solr$lo-mean(sole$lo)
> b <- glm(eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+I(la^2)
+           +I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+I(t^3) +
+           I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+I(lo^2*t) +
+           I(la^2*t)+I(la*t^2)+I(lo*t^2)+a+I(a*t)+I(t^2*a),
+           family=quasi(link=log,variance="mu"),data=solr)
> summary(b)

Call:
glm(formula = eggs~offset(off)+lo+la+t+I(lo*la)+
    I(lo^2)+I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+
    I(la^3)+I(t^3)+I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+
    I(lo^2*t)+I(la^2*t)+I(la*t^2)+I(lo*t^2)+a+I(a*t)+I(t^2*a),
    family = quasi(link=log,
```

```

variance = "mu"), data = solr)

Deviance Residuals:
    Min      1Q   Median      3Q     Max
-4.10474 -0.35127 -0.10418 -0.01289  5.66956

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03836  0.14560 -0.263 0.792202
lo           5.22548  0.39436 13.251 < 2e-16 ***
la          -5.94345  0.50135 -11.855 < 2e-16 ***
t            -2.43222  0.25761 -9.442 < 2e-16 ***
I(lo * la)   3.38576  0.61797  5.479 4.99e-08 ***
I(lo^2)      -3.98406  0.36744 -10.843 < 2e-16 ***
I(la^2)      -4.21517  0.56228 -7.497 1.10e-13 ***
I(t^2)       -1.77607  0.26279 -6.758 1.97e-11 ***
I(lo * t)    0.20029  0.35117  0.570 0.568518
I(la * t)    1.82637  0.47332  3.859 0.000119 ***
I(lo^3)      -3.46452  0.49554 -6.991 4.03e-12 ***
I(la^3)      8.53152  1.28587  6.635 4.48e-11 ***
I(t^3)       0.70085  0.12397  5.653 1.87e-08 ***
I(lo * la * t) -1.10150  0.90738 -1.214 0.224959
I(lo^2 * la)  5.20779  0.88873  5.860 5.65e-09 ***
I(lo * la^2) -12.87497 1.24298 -10.358 < 2e-16 ***
I(lo^2 * t)   0.79928  0.54238  1.474 0.140774
I(la^2 * t)   5.42159  1.08911  4.978 7.14e-07 ***
I(la * t^2)   -1.14220  0.46440 -2.459 0.014021 *
I(lo * t^2)   0.65862  0.36929  1.783 0.074705 .
a             -0.12285  0.02184 -5.624 2.21e-08 ***
I(a * t)      0.09456  0.04615  2.049 0.040635 *
I(t^2 * a)   -0.18310  0.05998 -3.053 0.002306 **
---
Sig. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasi family taken to be 1.051635)

Null deviance: 3108.86 on 1574 degrees of freedom
Residual deviance: 913.75 on 1552 degrees of freedom
AIC: NA

```

Number of Fisher Scoring iterations: 7

The summary information suggests dropping the `lo*t` term (it seems unreasonable to drop the constant altogether). Rather than re-type the whole `glm` command again it is easier to use:

```
b1<-update(b, ~.-I(lo*t))
```

which re-fits the model, dropping the term specified. Repeating the process, suggests

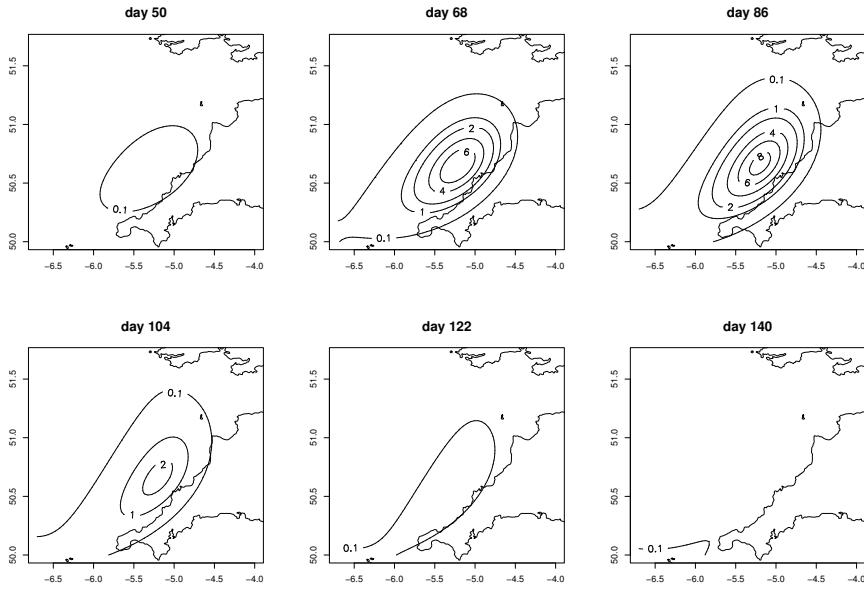


Figure 2.16 Model predicted Sole spawning rates in the Bristol Channel at various times in the spawning season.

dropping $\text{lo} * \text{la} * \text{t}$, $\text{lo} * \text{t}^2$ and finally $\text{lo}^2 * \text{t}$, after which all the remaining terms are significant at the 5% level. If $b4$ is the final reduced model, then it can be tested against the full model:

```
> anova(b,b4,test="F")
Analysis of Deviance Table
[edited]
  Resid. Df Resid. Dev    Df Deviance      F Pr(>F)
1      1552     913.75
2      1556     919.28    -4     -5.54 1.3161 0.2618
```

which gives no reason not to accept the simplified model.

The default residual plots are unhelpful for this model, because of the large number of zeroes in the data, corresponding to areas where there really are no eggs. This tends to lead to some very small values for the linear predictor, corresponding to zero predictions which in turn lead to rather distorted plots. The following residual plots are perhaps more useful.

```
> par(mfrow=c(1,2)) # split graph window into 4 panels
> plot(fitted(b4),solr$eggs) # fitted vs. data plot
> plot(fitted(b4)^0.5,residuals(b4)) # resids vs. sqrt(fitted)
```

The plots are shown in figure 2.15. The most noticeable features of both plots relate

to the large number of zeros in the data, with the lower boundary line in the right hand plot corresponding entirely to zeros, for which the raw residual is simply the negative of the fitted value. The plots are clearly far from perfect, but it is unlikely that great improvements can be made to them, with models of this general type.

The fitted model can be used for prediction of the spawning rate over the Bristol channel, by setting up a data frame containing the times and locations at which predictions are required, the age at which prediction is required — always zero — and the offset required — zero if spawning rate per square metre is the desired output. The time and location co-ordinates must be scaled in the same way as was done for fitting, of course. Figure 2.16 shows model predicted spawning rates produced in this way from model b4. It has been possible to get surprisingly far with the analysis of these data using a simple GLM approach, but the fitting did become somewhat unwieldy when it came to specifying that spawning rate should be a smooth function of location and time. For a less convenient spatial spawning distribution, it is doubtful that a satisfactory model could have been produced in this manner. This is part of the motivation for seeking to extend the way in which GLMs are specified, to allow a more compact and flexible way of specifying smooth functional relationships within the models: i.e. part of the motivation for developing GAMs.

2.4 Likelihood

This final, slightly more advanced, section covers the general theory of likelihood and quasi-likelihood, used for inference with GLMs. In particular the results invoked in section 2.1 are derived here. The emphasis is on explaining the key ideas as simply as possible, so some of the results are proved only for the simple case of a single parameter and i.i.d. data with the generalizations merely stated. To emphasize that the results given here apply much more widely than GLMs, the parameter that is the object of inference is denoted by θ in this section: for GLMs this will usually be β . Good references on the topics covered here are Cox and Hinkley (1974) and Silvey (1970), which are followed quite closely below.

Proofs are only given in outline and two general statistical results are used repeatedly: the ‘law of large numbers’ (LLN) and the ‘central limit theorem’ CLT. In the i.i.d. context these are as follows. Let X_1, X_2, \dots, X_n be i.i.d. random variables with mean μ and variance σ^2 (both of which are finite). The LLN states that as $n \rightarrow \infty$ $\bar{X} \rightarrow \mu$ (in probability)^{||}. The CLT states that as $n \rightarrow \infty$ the distribution of \bar{X} tends to $N(\mu, \sigma^2/n)$ whatever the distribution of the X_i . Both results generalize to multivariate and non i.i.d. settings.

^{||} tending to a limit in probability basically means that the probability of being further than any positive constant ϵ from the limit tends to zero.

2.4.1 Invariance

Consider an observation, $\mathbf{y} = [y_1, y_2, \dots, y_n]^\top$, of a vector of random variables, with joint p.m.f. or p.d.f. $f(\mathbf{y}, \theta)$, where θ is a parameter with m.l.e. $\hat{\theta}$. If γ is a parameter such that $\gamma = g(\theta)$, where g is any function, then the maximum likelihood estimate of γ is $\hat{\gamma} = g(\hat{\theta})$, and this property is known as *invariance*.

Invariance holds for any g , but a proof is easiest for the case in which g is a one to one function, so that g^{-1} is well defined. In this case $\theta = g^{-1}(\gamma)$ and maximum likelihood estimation would proceed by maximizing the likelihood

$$L(\gamma) = f(\mathbf{y}, g^{-1}(\gamma))$$

w.r.t. γ . But we know that the maximum of f occurs at $f(\mathbf{y}, \hat{\theta})$, by definition of $\hat{\theta}$, so it must be the case that L 's maximum w.r.t. γ occurs when $\hat{\theta} = g^{-1}(\hat{\gamma})$. i.e.

$$\hat{\gamma} = g(\hat{\theta})$$

is the m.l.e. of γ . So, when working with maximum likelihood estimation, we can adopt whatever parameterization is most convenient for performing calculations, and simply transform back to the most interpretable parameterization at the end. Note that invariance holds for vector parameters as well.

2.4.2 Properties of the expected log-likelihood

The key to proving and understanding the large sample properties of maximum likelihood estimators lies in obtaining some results for the expectation of the log-likelihood, and then using the convergence in probability of the log-likelihood to its expected value, which results from the law of large numbers. In this section, some simple properties of the expected log likelihood are derived.

Let y_1, y_2, \dots, y_n be independent observations from a p.d.f. $f(y, \theta)$, where θ is an unknown parameter with true value θ_0 . Treating θ as unknown, the log-likelihood for θ is

$$l(\theta) = \sum_{i=1}^n \log[f(y_i, \theta)] = \sum_{i=1}^n l_i(\theta),$$

where l_i is the log-likelihood, given only the single observation y_i . Treating l as a function of random variables, Y_1, Y_2, \dots, Y_n , means that l is itself a random variable (and the l_i are independent random variables). Hence we can consider expectations of l and its derivatives.

Result 1:

$$\mathbb{E}_0 \left(\frac{\partial l}{\partial \theta} \Big|_{\theta_0} \right) = 0 \quad (2.14)$$

The subscript on the expectation is to emphasize that the expectation is w.r.t. $f(y, \theta_0)$. The proof goes as follows (where it is to be taken that all derivatives are evaluated at

θ_0 , and there is sufficient regularity that the order of differentiation and integration can be exchanged)

$$\begin{aligned}\mathbb{E}_0\left(\frac{\partial l_i}{\partial \theta}\right) &= \mathbb{E}_0\left(\frac{\partial}{\partial \theta} \log[f(Y, \theta)]\right) = \int \frac{1}{f(y, \theta_0)} \frac{\partial f}{\partial \theta} f(y, \theta_0) dy \\ &= \int \frac{\partial f}{\partial \theta} dy = \frac{\partial}{\partial \theta} \int f dy = \frac{\partial 1}{\partial \theta} = 0.\end{aligned}$$

That the same holds for l follows immediately.

Result 1 has the following obvious consequence:

Result 2:

$$\text{var}\left(\left.\frac{\partial l}{\partial \theta}\right|_{\theta_0}\right) = \mathbb{E}_0\left[\left(\left.\frac{\partial l}{\partial \theta}\right|_{\theta_0}\right)^2\right]. \quad (2.15)$$

It can further be shown that

Result 3:

$$\mathcal{I} \equiv \mathbb{E}_0\left[\left(\left.\frac{\partial l}{\partial \theta}\right|_{\theta_0}\right)^2\right] = -\mathbb{E}_0\left[\left.\frac{\partial^2 l}{\partial \theta^2}\right|_{\theta_0}\right] \quad (2.16)$$

where \mathcal{I} is referred to as the **information** about θ contained in the data. The terminology refers to the fact that, if the data tie down θ very closely (and accurately), then the log likelihood will be sharply peaked in the vicinity θ_0 (i.e. high \mathcal{I}), whereas data containing little information about θ will lead to an almost flat likelihood, and low \mathcal{I} .

The proof of result 3 is simple. For a single observation, result 1 says that

$$\int \frac{\partial \log(f)}{\partial \theta} f dy = 0.$$

Differentiating again w.r.t. θ yields

$$\int \frac{\partial^2 \log(f)}{\partial \theta^2} f + \frac{\partial \log(f)}{\partial \theta} \frac{\partial f}{\partial \theta} dy = 0$$

but

$$\frac{\partial \log(f)}{\partial \theta} = \frac{1}{f} \frac{\partial f}{\partial \theta}$$

and so

$$\int \frac{\partial^2 \log(f)}{\partial \theta^2} f dy = - \int \left[\frac{\partial \log(f)}{\partial \theta} \right]^2 f dy$$

which is

$$\mathbb{E}_0\left[\left.\frac{\partial^2 l_i}{\partial \theta^2}\right|_{\theta_0}\right] = -\mathbb{E}_0\left[\left(\left.\frac{\partial l_i}{\partial \theta}\right|_{\theta_0}\right)^2\right].$$

The result follows very easily (given the independence of the l_i).

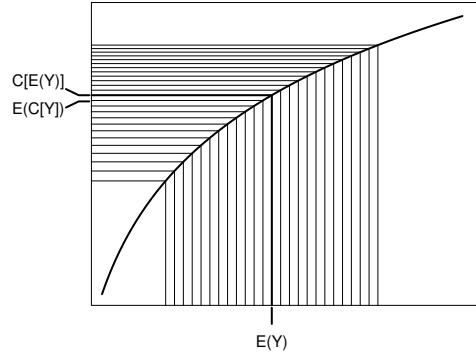


Figure 2.17 *Schematic illustration of Jensen's inequality which says that if c is a concave function then $\mathbb{E}[c(Y)] \leq c(\mathbb{E}[Y])$. The curve shows a concave function, while the lines connect the values of a discrete uniform random variable Y on the horizontal axis to the values of the discrete random variable $c(Y)$ on the vertical axis. It is immediately clear that the way that c spreads out $c(Y)$ values corresponding to low Y values, while bunching together $c(Y)$ values corresponding to high Y values implies Jensen's inequality. Further reflection suggests (correctly) that Jensen's inequality holds for any distribution.*

Note that by result 1 the expected log likelihood has a turning point at θ_0 , and since \mathcal{I} is positive, result 3 indicates that this turning point is a maximum. So the expected log likelihood has a maximum at the true parameter value. Unfortunately results 1 and 3 don't establish that this maximum is a global maximum, but a slightly more involved proof shows that this is in fact the case.

Result 4:

$$\mathbb{E}_0[l(\theta_0)] \geq \mathbb{E}_0[l(\theta)] \quad \forall \theta \quad (2.17)$$

The proof is based on Jensen's inequality, which says that if c is a concave function (i.e. has negative second derivative) and Y is a random variable, then

$$\mathbb{E}[c(Y)] \leq c(\mathbb{E}[Y]).$$

The inequality is almost a statement of the obvious as figure 2.17 illustrates. Now consider the concave function, \log , and the random variable, $f(Y, \theta)/f(Y, \theta_0)$. Jensen's inequality implies that

$$\mathbb{E}_0 \left[\log \left(\frac{f(Y, \theta)}{f(Y, \theta_0)} \right) \right] \leq \log \left[\mathbb{E}_0 \left(\frac{f(Y, \theta)}{f(Y, \theta_0)} \right) \right].$$

Consider the right hand side of the inequality.

$$\mathbb{E}_0 \left(\frac{f(Y, \theta)}{f(Y, \theta_0)} \right) = \int \frac{f(y, \theta)}{f(y, \theta_0)} f(y, \theta_0) dy = \int f(y, \theta) dy = 1.$$

So, since $\log(1) = 0$ the inequality becomes

$$\begin{aligned}\mathbb{E}_0 \left[\log \left(\frac{f(Y, \theta)}{f(Y, \theta_0)} \right) \right] &\leq 0 \\ \Rightarrow \mathbb{E}_0[\log(f(Y, \theta))] &\leq \mathbb{E}_0[\log(f(Y, \theta_0))]\end{aligned}$$

from which the result follows immediately.

The above results were derived for continuous Y , but also hold for discrete Y : the proofs are almost identical, but with $\sum_{\text{all } y_i}$ replacing $\int dy$. Note also that, although the results presented here were derived assuming that the data were independent observations from the same distribution, this is in fact much more restrictive than is necessary, and the results hold more generally.

Similarly, the results generalize to vector parameters. Let \mathbf{u} be the vector such that $u_i = \partial l / \partial \theta_i$, and \mathbf{H} be the hessian matrix of the log-likelihood w.r.t. the parameters so that $H_{i,j} = \partial^2 l / \partial \theta_i \partial \theta_j$. Then

Result 1 (vector parameter)

$$\mathbb{E}_0(\mathbf{u}) = \mathbf{0} \quad (2.18)$$

and

Result 3 (vector parameter)

$$\mathcal{I} \equiv \mathbb{E}_0(\mathbf{u}\mathbf{u}^T) = -\mathbb{E}_0(\mathbf{H}). \quad (2.19)$$

2.4.3 Consistency

Maximum likelihood estimators are often not unbiased, but under quite mild regularity conditions they are *consistent*. This means that as the sample size, on which the estimate is based, tends to infinity, the maximum likelihood estimator tends in probability to the true parameter value. Consistency therefore implies asymptotic** unbiasedness, but it actually implies slightly more than this — for example that the variance of the estimator is decreasing with sample size.

Formally, if θ_0 is the true value of parameter θ , and $\hat{\theta}_n$ is its m.l.e. based on n observations, y_1, y_2, \dots, y_n , then consistency means that

$$\Pr[|\hat{\theta}_n - \theta_0| < \epsilon] \rightarrow 1$$

as $n \rightarrow \infty$ for any positive ϵ .

To see why m.l.e.s are consistent, consider an outline proof for the case of a single parameter, θ , estimated from independent observations y_1, y_2, \dots, y_n on a random variable with p.m.f. or p.d.f. $f(y, \theta_0)$. The log-likelihood in this case will be

$$l(\theta) \propto \frac{1}{n} \sum_{i=1}^n \log(f(y_i, \theta))$$

** ‘asymptotic’ here meaning ‘as sample size tends to infinity’.

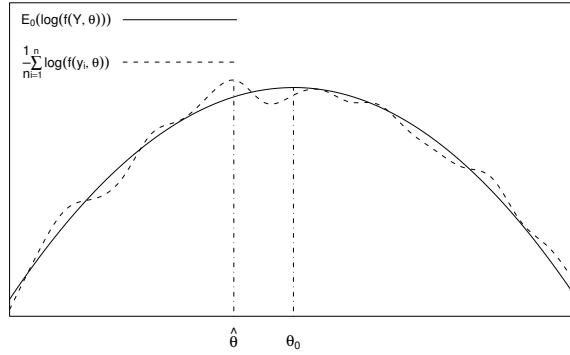


Figure 2.18 *Illustration of the idea behind the derivation of consistency of MLEs.* The dashed curve is proportional to the log likelihood while the solid curve is the expectation of the log likelihood. The solid curve has a maximum at the true parameter value, and as sample size tends to infinity the dashed curve tends to the solid curve by the LOLN, hence in the large sample limit the log likelihood has a maximum at the true parameter value, and we expect $\hat{\theta}_n \rightarrow \theta_0$ as $n \rightarrow \infty$.

where the factor of $1/n$ is introduced purely for later convenience. We need to show that in the large sample limit $l(\theta)$ achieves its maximum at the true parameter value θ_0 , but in the previous section it was shown that the expected value of the log likelihood for a single observation attains its maximum at θ_0 . The law of large numbers tells us that as $n \rightarrow \infty$, $\sum_{i=1}^n \log[f(Y_i, \theta)]/n$ tends (in probability) to $\mathbb{E}_0[\log(f(Y, \theta))]$. So in the large sample limit we have that

$$l(\theta_0) \geq l(\theta)$$

i.e. that $\hat{\theta}$ is θ_0 .

To show that $\hat{\theta} \rightarrow \theta_0$ in some well ordered manner as $n \rightarrow \infty$ requires that we assume some regularity (for example, at minimum, we need to be able to assume that if θ_1 and θ_2 are ‘close’ then so are $l(\theta_1)$ and $l(\theta_2)$), but in the vast majority of practical situations such conditions hold. Figure 2.18 can help illustrate how the argument works: as the sample size tends to infinity the dashed curve, proportional to the log likelihood, tends in probability to the solid curve, $\mathbb{E}_0[\log(f(Y, \theta))]$, which has its maximum at θ_0 , hence $\hat{\theta} \rightarrow \theta_0$.

For simplicity of presentation, the above argument dealt only with a single parameter and data that were independent observations of a random variable from one distribution. In fact consistency holds in much more general circumstance: for vector parameters, and non-independent data that do not necessarily all come from the same distribution.

2.4.4 Large sample distribution of $\hat{\theta}$

To obtain the large sample distribution of the m.l.e., $\hat{\theta}$, we make a Taylor expansion of the derivative of the log likelihood around the true parameter, θ_0 , and evaluate this at $\hat{\theta}$.

$$\frac{\partial l}{\partial \theta} \Big|_{\hat{\theta}} \simeq \frac{\partial l}{\partial \theta} \Big|_{\theta_0} + (\hat{\theta} - \theta_0) \frac{\partial^2 l}{\partial \theta^2} \Big|_{\theta_0}$$

and from the definition of the m.l.e. the left hand side must be zero, so we have that

$$(\hat{\theta} - \theta_0) \simeq \frac{-\partial l / \partial \theta|_{\theta_0}}{\partial^2 l / \partial \theta^2|_{\theta_0}},$$

with equality in the large sample limit (by consistency of $\hat{\theta}$). Now the top of this fraction has expected value zero and variance \mathcal{I} (see results (2.14) and (2.15)), but it is also made up of a sum of i.i.d. random variables, $l_i = \log[f(x_i, \theta)]$, so that by the central limit theorem, as $n \rightarrow \infty$, its distribution will tend to $N(0, \mathcal{I})$. By the law of large numbers we also have that, as $n \rightarrow \infty$, $\partial^2 l / \partial \theta^2|_{\theta_0} \rightarrow \mathcal{I}$ (in probability). So in the large sample limit $(\hat{\theta} - \theta_0)$ is distributed as an $N(0, \mathcal{I})$ r.v. divided by \mathcal{I} . i.e. in the limit as $n \rightarrow \infty$

$$(\hat{\theta} - \theta_0) \sim N(0, \mathcal{I}^{-1}).$$

The result generalizes to vector parameters:

$$\hat{\theta} \sim N(\theta_0, \mathcal{I}^{-1}) \tag{2.20}$$

in the large sample limit. Again the result holds generally and not just for the somewhat restricted form of the likelihood which we have assumed here.

Usually \mathcal{I} will not be known, any more than θ is, and will have to be estimated by plugging $\hat{\theta}$ into the expression for \mathcal{I} . Often this *empirical information matrix*, which is just the negative of the hessian ($-\mathbf{H}$) of the log-likelihood evaluated at the m.l.e., is an adequate approximation to the information matrix \mathcal{I} itself (this follows from the law of large numbers).

2.4.5 The generalized likelihood ratio test (GLRT)

Consider an observation, \mathbf{y} , on a random vector of dimension n with p.d.f. (or p.m.f.) $f(\mathbf{y}, \theta)$, where θ is a parameter vector. Suppose that we want to test:

$$H_0 : \mathbf{R}(\theta) = \mathbf{0} \quad \text{vs.} \quad H_1 : \mathbf{R}(\theta) \neq \mathbf{0},$$

where \mathbf{R} is a vector valued function of θ , such that H_0 imposes r restrictions on the parameter vector. If H_0 is true then in the limit as $n \rightarrow \infty$

$$2\lambda = 2(l(\hat{\theta}_{H_1}) - l(\hat{\theta}_{H_0})) \sim \chi_r^2, \tag{2.21}$$

where l is the log-likelihood function and $\hat{\theta}_{H_1}$ is the m.l.e. of θ . $\hat{\theta}_{H_0}$ is the value of θ satisfying $\mathbf{R}(\theta) = \mathbf{0}$, which maximizes the likelihood (i.e. the restricted m.l.e.). This result is used to calculate approximate p-values for the test.

Tests performed in this way are known as ‘generalized likelihood ratio tests’, since λ is the log of the ratio of the maximized likelihoods under each hypothesis. In the context of GLMs, the null hypothesis is usually that the correct model is a simplified version of the model under the alternative hypothesis.

2.4.6 Derivation of $2\lambda \sim \chi_r^2$ under H_0

To simplify matters, first suppose that the parameterization is such that $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\psi} \\ \boldsymbol{\gamma} \end{bmatrix}$, where $\boldsymbol{\psi}$ is r dimensional, and the null hypothesis can be written $H_0 : \boldsymbol{\psi} = \boldsymbol{\psi}_0$. In principle it is always possible to re-parameterize a model so that the null has this form^{††}.

Now let the unrestricted m.l.e. be $\begin{bmatrix} \hat{\boldsymbol{\psi}} \\ \hat{\boldsymbol{\gamma}} \end{bmatrix}$, and let $\begin{bmatrix} \boldsymbol{\psi}_0 \\ \hat{\boldsymbol{\gamma}}_0 \end{bmatrix}$ be the m.l.e. under the restrictions defining the null hypothesis. The key to making progress is to be able to express $\hat{\boldsymbol{\gamma}}_0$ in terms of $\hat{\boldsymbol{\psi}}$, $\hat{\boldsymbol{\gamma}}$ and $\boldsymbol{\psi}_0$. This is possible, in general, in the large sample limit, provided that the null hypothesis is true, so that $\hat{\boldsymbol{\psi}}$ is close to $\boldsymbol{\psi}_0$. Taking a Taylor expansion of the log likelihood around the unrestricted m.l.e. $\hat{\boldsymbol{\theta}}$ yields

$$l(\boldsymbol{\theta}) \simeq l(\hat{\boldsymbol{\theta}}) - \frac{1}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{H} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \quad (2.22)$$

where $H_{i,j} = -\partial^2 l / \partial \theta_i \partial \theta_j|_{\hat{\boldsymbol{\theta}}}$ (note the factor of -1 introduced here). Exponentiating this expression, the likelihood can be written

$$L(\boldsymbol{\theta}) \simeq L(\hat{\boldsymbol{\theta}}) \exp \left[-(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{H} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) / 2 \right].$$

i.e. the likelihood can be approximated by a function proportional to the p.d.f. of an $N(\hat{\boldsymbol{\theta}}, \mathbf{H}^{-1})$ random variable. By standard properties of the multivariate normal p.d.f., if

$$\begin{bmatrix} \boldsymbol{\psi} \\ \boldsymbol{\gamma} \end{bmatrix} \sim N \left(\begin{bmatrix} \hat{\boldsymbol{\psi}} \\ \hat{\boldsymbol{\gamma}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\psi\psi} & \boldsymbol{\Sigma}_{\psi\gamma} \\ \boldsymbol{\Sigma}_{\gamma\psi} & \boldsymbol{\Sigma}_{\gamma\gamma} \end{bmatrix} \right)$$

then

$$\boldsymbol{\gamma} | \boldsymbol{\psi} \sim N(\hat{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_{\gamma\psi} \boldsymbol{\Sigma}_{\psi\psi}^{-1} (\boldsymbol{\psi} - \hat{\boldsymbol{\psi}}), \boldsymbol{\Sigma}_{\gamma\gamma} - \boldsymbol{\Sigma}_{\gamma\psi} \boldsymbol{\Sigma}_{\psi\psi}^{-1} \boldsymbol{\Sigma}_{\psi\gamma}).$$

Hence if $\boldsymbol{\psi}$ is fixed at $\boldsymbol{\psi}_0$ by hypothesis, then the approximation to the likelihood will be maximized when $\boldsymbol{\gamma}$ takes the value

$$\hat{\boldsymbol{\gamma}}_0 = \hat{\boldsymbol{\gamma}} + \boldsymbol{\Sigma}_{\gamma\psi} \boldsymbol{\Sigma}_{\psi\psi}^{-1} (\boldsymbol{\psi}_0 - \hat{\boldsymbol{\psi}}). \quad (2.23)$$

If the null hypothesis is true, then in the large sample limit $\hat{\boldsymbol{\psi}} \rightarrow \boldsymbol{\psi}_0$ (in probability) so that the approximate likelihood tends to the true likelihood, and we can expect (2.23) to hold for the maximizers of the exact likelihood.

^{††} Of course, to use the result no re-parameterization is necessary — it’s only being done here for theoretical convenience when deriving the result. Invariance ensures that reparameterization is a legitimate thing to do.

Expressing (2.23) in terms of a partitioning of $\Sigma = \mathbf{H}^{-1}$, is not as useful as having the results in terms of the equivalent partitioning of \mathbf{H} itself. Writing $\Sigma\mathbf{H} = \mathbf{I}$ in partitioned form

$$\begin{bmatrix} \Sigma_{\psi\psi} & \Sigma_{\psi\gamma} \\ \Sigma_{\gamma\psi} & \Sigma_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{\psi\psi} & \mathbf{H}_{\psi\gamma} \\ \mathbf{H}_{\gamma\psi} & \mathbf{H}_{\gamma\gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

and multiplying out, results in four matrix equations, of which two are useful:

$$\Sigma_{\psi\psi}\mathbf{H}_{\psi\psi} + \Sigma_{\psi\gamma}\mathbf{H}_{\gamma\psi} = \mathbf{I}, \quad (2.24)$$

$$\Sigma_{\psi\psi}\mathbf{H}_{\psi\gamma} + \Sigma_{\psi\gamma}\mathbf{H}_{\gamma\gamma} = \mathbf{0}. \quad (2.25)$$

Re-arranging (2.25) while noting that, by symmetry, $\mathbf{H}_{\psi\gamma}^T = \mathbf{H}_{\gamma\psi}$ and $\Sigma_{\psi\gamma}^T = \Sigma_{\gamma\psi}^{\ddagger\dagger}$, yields

$$-\mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi} = \Sigma_{\gamma\psi}\Sigma_{\psi\psi}^{-1}$$

and hence

$$\hat{\gamma}_0 = \hat{\gamma} + \mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi}(\hat{\psi} - \psi_0). \quad (2.26)$$

For later use it is also worth eliminating $\Sigma_{\psi\gamma}$ from (2.24) and (2.25), which results in

$$\Sigma_{\psi\psi}^{-1} = \mathbf{H}_{\psi\psi} - \mathbf{H}_{\psi\gamma}\mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi}. \quad (2.27)$$

Now provided that the null hypothesis is true, so that $\hat{\psi}$ is close to ψ_0 , we can re-use the expansion (2.22) and write the log-likelihood at the restricted m.l.e. as

$$l(\psi_0, \hat{\gamma}_0) \simeq l(\hat{\psi}, \hat{\gamma}) - \frac{1}{2} \begin{bmatrix} \psi_0 - \hat{\psi} \\ \hat{\gamma}_0 - \hat{\gamma} \end{bmatrix}^T \mathbf{H} \begin{bmatrix} \psi_0 - \hat{\psi} \\ \hat{\gamma}_0 - \hat{\gamma} \end{bmatrix}.$$

Hence

$$2\lambda = 2(l(\hat{\psi}, \hat{\gamma}) - l(\psi_0, \hat{\gamma}_0)) \simeq \begin{bmatrix} \psi_0 - \hat{\psi} \\ \hat{\gamma}_0 - \hat{\gamma} \end{bmatrix}^T \mathbf{H} \begin{bmatrix} \psi_0 - \hat{\psi} \\ \hat{\gamma}_0 - \hat{\gamma} \end{bmatrix}.$$

Substituting for $\hat{\gamma}_0$ from (2.26) and writing out \mathbf{H} in partitioned form gives

$$2\lambda \simeq \begin{bmatrix} \psi_0 - \hat{\psi} \\ \mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi}(\hat{\psi} - \psi_0) \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_{\psi\psi} & \mathbf{H}_{\psi\gamma} \\ \mathbf{H}_{\gamma\psi} & \mathbf{H}_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} \psi_0 - \hat{\psi} \\ \mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi}(\hat{\psi} - \psi_0) \end{bmatrix}$$

and a short routine slog results in

$$2\lambda \simeq (\hat{\psi} - \psi_0)^T [\mathbf{H}_{\psi\psi} - \mathbf{H}_{\psi\gamma}\mathbf{H}_{\gamma\gamma}^{-1}\mathbf{H}_{\gamma\psi}] (\hat{\psi} - \psi_0).$$

But given (2.27), this means that

$$2\lambda \simeq (\hat{\psi} - \psi_0)^T \Sigma_{\psi\psi}^{-1} (\hat{\psi} - \psi_0). \quad (2.28)$$

Now if H_0 is true, then as $n \rightarrow \infty$ this expression will tend towards exactness as $\hat{\psi} \rightarrow \psi_0$. Furthermore, by the law of large numbers and (2.19), $\mathbf{H} \rightarrow \mathcal{I}$ as $n \rightarrow \infty$ (recall that in this section \mathbf{H} is the negative second derivative matrix), which means

$\ddagger\dagger$ and of course remembering that $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$.

that Σ tends to \mathcal{I}^{-1} , and hence $\Sigma_{\psi\psi}$ tends to the covariance matrix of $\hat{\psi}$ (see result 2.20). Hence, by the asymptotic normality of the m.l.e. $\hat{\psi}$,

$$2\lambda \sim \chi_r^2$$

under H_0 . Having proved the asymptotic distribution of λ under H_0 , you might be wondering why it was worth bothering, when we could simply have used the right hand side of (2.28) directly as the test statistic. This approach is indeed possible, and is known as the Wald test, but it suffers from the disadvantage that at finite sample sizes the magnitude of the test statistic depends on how we choose to parameterize the model. The GLRT, on the other hand, is invariant to the parameterization we choose to use, irrespective of sample size. This invariance seems much more satisfactory — we do not generally want our statistical conclusions to depend on details of how we set up the model, if those details could never be detected by observing data from the model.

2.4.7 AIC in general

As we have seen, selecting between (nested) models on the basis of which has higher likelihood is generally unsatisfactory, because the model with more parameters always has the higher likelihood. Indeed, the previous section shows that, if we add a redundant parameter to an already correct model, the expected increase in likelihood is $\simeq 1/2$. This problem arises because each additional parameter allows the model to get a little closer to the observed data, by fitting the noise component of the data, as well as the signal. If we were to judge between models on the basis of their fit to new data, not used in estimation, then this problem would not arise. The Akaike Information Criterion (AIC) is an attempt to provide a way of doing this.

The derivation of AIC basically has two parts:

- i) The average ability of a maximum likelihood estimated model to predict new data is measured by the expected Kulbeck-Leibler (K-L) discrepancy between the estimated model and the true model. This can be shown to be approximately the minimum K-L discrepancy that the model could possibly achieve, for any parameter values, plus an estimable constant.
- ii) The expected negative log likelihood of the model can be approximately expressed as the minimum K-L distance that could possibly be achieved *minus* the same estimable constant (and another ignorable constant). This immediately suggests an estimator for the expected KL discrepancy in terms of the negative log likelihood and the constant.

Superficially similar looking expectations appear to evaluate to rather different quantities in parts (i) and (ii), which can cause confusion. The key difference is that in part (i) we are interested in the estimated model's ability to predict new data, so that the parameter estimators are not functions of the data over whose distribution the expectations are taken. In part (ii), when examining the expected log likelihood, the parameter estimators are most definitely functions of the data.

Suppose then, that our data were really generated from a density $f_0(y)$ and that our model density is $f_\theta(y)$, where θ denotes the parameter(s) of f_θ , and y and θ will generally be vectors, with θ having dimension p .

$$K(f_\theta, f_0) = \int \{\log[f_0(y)] - \log[f_\theta(y)]\} f_0(y) dy \quad (2.29)$$

provides a measure of how badly f_θ matches the truth, known as the Kullbeck-Leibler discrepancy. So, if $\hat{\theta}$ is the m.l.e. of θ , then $K(f_{\hat{\theta}}, f_0)$ could be used to provide a measure of how well our model is expected to fit a new set of data, not used to estimate $\hat{\theta}$. Note that because we are interested in new data, $\hat{\theta}$ is treated as fixed and not as a function of y when evaluating (2.29).

Of course, (2.29) can not be used directly, since we are not sure of f_0 , but progress can be made by considering a truncated Taylor expansion of f_θ about the (unknown) parameters θ_K which would minimize (2.29).

$$\log[f_{\hat{\theta}}(y)] \simeq \log[f_{\theta_K}(y)] + (\hat{\theta} - \theta_K)^\top g + \frac{1}{2}(\hat{\theta} - \theta_K)^\top H(\hat{\theta} - \theta_K) \quad (2.30)$$

where g and H are the gradient vector and Hessian matrix of first and second derivatives of f_θ w.r.t. θ evaluated at θ_K . It is easy to see that if θ_K minimizes (2.29) then $\int g f_0 dy = 0$, so that substituting (2.30) into (2.29) yields

$$K(f_{\hat{\theta}}, f_0) \simeq K(f_{\theta_K}, f_0) + \frac{1}{2}(\hat{\theta} - \theta_K)^\top \mathcal{I}_K(\hat{\theta} - \theta_K) \quad (2.31)$$

where \mathcal{I}_K is the information matrix at θ_K . The dependence on θ_K is not helpful here, but can be removed by taking expectations over the distribution of $\hat{\theta}$, which yields

$$\mathbb{E}[K(f_{\hat{\theta}}, f_0)] \simeq K(f_{\theta_K}, f_0) + p/2, \quad (2.32)$$

where it has been assumed that the model is close enough to correct that consistency ensures closeness of $\hat{\theta}$ and θ_K and the large sample distribution of $\hat{\theta}$ implies that twice the last term in (2.31) has a χ_p^2 distribution.

Now (2.32) still depends on f_0 , and we need to be able to estimate it using what we have available, which does not include knowledge of f_0 . A useful estimator of $K(f_{\theta_K}, f_0)$ can be based on $-l(\hat{\theta}) = -\log[f_{\hat{\theta}}(y)]$, but to ensure that it is approximately unbiased, we need to consider $\mathbb{E}[-l(\hat{\theta})]$, where the expectation is now taken allowing for the fact that $\hat{\theta}$ is a function of y .

$$\begin{aligned} \mathbb{E}\{-l(\hat{\theta})\} &= \mathbb{E}\{-l(\theta_K) - [l(\hat{\theta}) - l(\theta_K)]\} \\ &\simeq - \int \log[f_{\theta_K}(y)] f_0(y) dy - p/2 \\ &= K(f_{\theta_K}, f_0) - p/2 - \int \log[f_0(y)] f_0(y) dy \end{aligned}$$

where again it has been assumed that the model is close enough to correct that we can use the large sample result $[l(\hat{\theta}) - l(\theta_K)] \sim \chi_p^2$. Re-arrangement yields the estimator

$$\widehat{K(f_{\theta_K}, f_0)} = -l(\hat{\theta}) + p/2 + \int \log[f_0(y)] f_0(y) dy,$$

which can be substituted into (2.31) to obtain the estimator

$$\mathbb{E} \{ \widehat{K}(f_{\hat{\theta}}, f_0) \} \simeq -l(\hat{\theta}) + p + \int \log[f_0(y)] f_0(y) dy.$$

The final term on the RHS depends only on the unknown true model, and will be the same for any set of models to be compared using a given data set. Hence, it can safely be dropped, and $-l(\hat{\theta}) + p$ can be used for model comparison purposes. Twice this quantity is known as the Akaike Information Criterion,

$$AIC = 2[-l(\hat{\theta}) + p]. \quad (2.33)$$

Generally we expect estimated models with lower AIC scores to be closer to the true model, in the K-L sense, than models with higher AIC scores. The p term in the AIC score penalizes models with more parameters than necessary, thereby counteracting the tendency of the likelihood to favour ever larger models.

The forgoing derivation assumes that the estimated model is ‘close’ to the true model. If a sequence of nested models includes a model that is acceptably close, then all larger models will also be close. On the other hand if we start to violate this assumption, by oversimplifying a model, then the resulting decrease in $l(\hat{\theta})$ will typically be much larger than the decrease in p , resulting in a substantial drop in AIC score: basically the likelihood part of the AIC score is well able to discriminate between models that are over-simplified, and those that are not.

A derivation of AIC which deals more carefully with the case in which the model is ‘wrong’ is given in Davison (2003), from which the above derivation borrows heavily. A more careful accounting makes more precise the nature of the approximation made by using the penalty term, p , when the model is incorrect: but it doesn’t make p any less approximate in that case.

2.4.8 Quasi-likelihood results

The results of sections 2.4.3 to 2.4.6 also apply if the log likelihood l is replaced by the log quasi-likelihood q . The key to demonstrating this lies in deriving equivalents to results 1 to 4 of section 2.4.2, for the log quasi likelihood function. Again, for clarity, only a single parameter θ will be considered, but the results generalize.

Consider observations y_1, y_2, \dots, y_n on independent random variables, each with expectation μ_i . Given a single y_i , let the log quasi likelihood of μ_i be $q_i(\mu_i)$ (as defined by (2.12) in section 2.1.10), and suppose that all the μ_i depend on a parameter θ , which therefore has log quasi likelihood function

$$q(\theta) = \sum_{i=1}^n q_i(\mu_i).$$

Let θ_0 denote the true parameter value, \mathbb{E}_0 denote the expectation operator given that parameter value, and let μ_i^0 denote the true expected value of y_i .

Result 1:

$$\mathbb{E}_0 \left(\frac{\partial q}{\partial \theta} \Big|_{\theta_0} \right) = 0 \quad (2.34)$$

The proof is simple:

$$\mathbb{E}_0 \left(\frac{\partial q_i}{\partial \theta} \Big|_{\theta_0} \right) = \mathbb{E}_0 \left(\frac{\partial q_i}{\partial \mu_i} \Big|_{\mu_i^0} \right) \frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} = \frac{\mathbb{E}_0(Y_i) - \mu_i^0}{\phi V(\mu_i^0)} \frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} = 0,$$

from which the result follows immediately.

Given result 1 it is clear that,

Result 2:

$$\text{var} \left(\frac{\partial q}{\partial \theta} \Big|_{\theta_0} \right) = \mathbb{E}_0 \left[\left(\frac{\partial q}{\partial \theta} \Big|_{\theta_0} \right)^2 \right]. \quad (2.35)$$

Furthermore it can be shown that

Result 3:

$$\mathcal{I}_q \equiv \mathbb{E}_0 \left[\left(\frac{\partial q}{\partial \theta} \Big|_{\theta_0} \right)^2 \right] = -\mathbb{E}_0 \left[\frac{\partial^2 q}{\partial \theta^2} \Big|_{\theta_0} \right] \quad (2.36)$$

where \mathcal{I}_q might be termed the **quasi-information** about θ . The proof is straightforward.

$$\begin{aligned} \mathbb{E}_0 \left[\left(\frac{\partial q_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \right] &= \mathbb{E}_0 \left[\left(\frac{\partial q_i}{\partial \mu_i} \Big|_{\mu_i^0} \right)^2 \right] \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \\ &= \mathbb{E}_0 \left[\frac{(Y_i - \mu_i^0)^2}{\phi^2 V(\mu_i^0)^2} \right] \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \\ &= \frac{1}{\phi V(\mu_i^0)} \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \end{aligned}$$

Which can be shown to be equal to $-\mathbb{E}_0 \left[\partial^2 q_i / \partial \theta^2 \Big|_{\theta_0} \right]$ as follows

$$\begin{aligned} \mathbb{E}_0 \left[\frac{\partial^2 q_i}{\partial \theta^2} \Big|_{\theta_0} \right] &= \mathbb{E}_0 \left[\frac{\partial^2 q_i}{\partial \mu_i^2} \Big|_{\mu_i^0} \right] \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 + \mathbb{E}_0 \left(\frac{\partial q_i}{\partial \mu_i} \Big|_{\mu_i^0} \right) \frac{\partial^2 \mu_i}{\partial \theta^2} \Big|_{\theta_0} \\ &= \mathbb{E}_0 \left[\frac{\partial^2 q_i}{\partial \mu_i^2} \Big|_{\mu_i^0} \right] \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \\ &= \mathbb{E}_0 \left(\frac{-1}{\phi V(\mu_i^0)} - \frac{Y_i - \mu_i^0}{\phi^2 V(\mu_i^0)^2} \frac{\partial V}{\partial \mu} \Big|_{\mu_i^0} \right) \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \\ &= \frac{-1}{\phi V(\mu_i^0)} \left(\frac{\partial \mu_i}{\partial \theta} \Big|_{\theta_0} \right)^2 \end{aligned}$$

The result now follows easily given the independence of the Y_i and hence of the q_i .

Finally we have

Result 4:

$$\mathbb{E}_0[q(\theta_0)] \geq \mathbb{E}_0[q(\theta)] \quad \forall \theta \quad (2.37)$$

Proof is considerably easier than was the case for the equivalent likelihood result. By result 1 we know that $\mathbb{E}_0(q)$ has a turning point at θ_0 . Furthermore

$$\mathbb{E}_0\left(\frac{\partial q}{\partial \theta}\right) = \sum_{i=1}^n \mathbb{E}_0\left(\frac{\partial q_i}{\partial \mu_i}\right) \frac{\partial \mu_i}{\partial \theta} = \sum_{i=1}^n \frac{\mu_i^0 - \mu_i}{\phi V(\mu_i)} \frac{\partial \mu_i}{\partial \theta}$$

implying that q decreases monotonically away from θ_0 provided that each $\partial \mu_i / \partial \theta$ has the same sign for all θ (different μ_i can have derivatives of different sign, of course). The restriction is always met for a GLM (since the signs of these derivatives are controlled by the signs of the corresponding elements of the model matrix, which are fixed).

As in the likelihood case, these quasi-likelihood results can be generalized to vector parameters.

The consistency, large sample distribution and GLRT results of sections 2.4.3 to 2.4.6 can now be re-derived for models estimated by quasi-likelihood maximization. The arguments of sections 2.4.3 to 2.4.6 are unchanged except for the replacement of l by q , l_i by q_i , \mathcal{I} by \mathcal{I}_q and results 1 to 4 of section 2.4.2 by results 1 to 4 of the current section.

2.5 Exercises

1. A Bernoulli random variable, Y , takes the value 1 with probability p and 0 with probability $1 - p$, so that its probability function is:

$$f(y) = p^y(1-p)^{1-y}, \quad y = 0 \text{ or } 1$$

(a) Find $\mu \equiv E(Y)$.

(b) Show that the Bernoulli distribution is a member of the exponential family of distributions, by showing that its probability function can be written as

$$f(y) = \exp [\{y\theta - b(\theta)\}/a(\phi) + c(y, \phi)],$$

for appropriately defined θ , $b(\theta)$, ϕ , a and c . (See section 2.1.1.)

(c) What will the canonical link be for the Bernoulli distribution?

2. Residual checking for non-Gaussian error models is not always as straightforward as it is in the Gaussian case, and the problems are particularly acute in the case of binary data. This question explores this issue.

(a) The following code fits a GLM to data simulated from a simple binomial model and examines the default residual plots.

```

n<-100;m<-10
x <- runif(n)
lp <- 3*x-1
mu <- binomial()$linkinv(lp)
y <- rbinom(1:n,m,mu)
par(mfrow=c(2,2))
plot(glm(y~x,family=binomial,weights=rep(m,n)))

```

Run the code several times to get a feel for the range of results that are possible even when the model is correct (as it clearly is in this case).

- (b) Explore how the plots change as m (the number of binomial trials) is reduced to 1. Also examine the effect of sample size n .
 - (c) By repeatedly simulating data from a fitted model, and then refitting to the simulated data, you can build up a picture of how the residuals should behave when the distributional assumption is correct, and the data are really independent. Write code to take a `glm` object fitted to binary data, simulate binary data given the model fitted values, refit the model to the resulting data, and extract residuals from the resulting fits. Functions `fitted` and `rbinom` are useful here.
 - (d) If `rsd` contains your residual vector then


```
plot(sort(rsd), (1:length(rsd)-.5)/length(rsd))
```

 produces a plot of the ‘empirical CDF’ of the residuals, which can be useful for characterizing their distribution. By repeatedly simulating residuals, as in the previous part, you can produce a ‘simulation envelope’ showing e.g. where the middle 95% of these ‘empirical CDFs’ should lie, if the model assumptions are met: such envelopes provide a guide to whether an observed ‘real’ residual distribution is reasonable, or not. Based on your answer to the previous part, write code to do this.
 - (e) Plots of the residuals against fitted values, or predictors, are also hard to interpret for models of binary data. A simple check for lack of independence in the residuals can be obtained by ordering the residuals according the the values of the fitted values or a predictor, and checking whether these ordered residuals show fewer (or more) runs of values above and below zero than they should if independent. The command


```
rsd <- rsd[sort(fv,return.index=TRUE)$ix]
```

 will put `rsd` into the order corresponding to increasing `fv`. It is possible to simulate from the distribution of runs of residuals that should occur, under independence, as part of the simulation loop used in the previous part: modify your code to check whether the residuals appear non-independent with respect to fitted values.
3. This question looks at data covering guilty verdicts involving multiple murders in Florida between 1976 and 1987. The data are classified by skin ‘colour’ of victim and defendant and by whether or not the sentence was death.

Victim's Race	Defendant's Race	Death Penalty	No Death Penalty
White	White	53	414
	Black	11	37
Black	White	0	16
	Black	4	139

Data are from Radelet and Pierce *Florida Law Review*:431-34 (1991), as reported in Agresti (1996).

- (a) What proportion of black defendants and what proportion of white defendants were sentenced to death?
 - (b) Find the log-linear model which best explains these data.
 - (c) How would you interpret your best fit model?
4. If a random variable, Y_i , has expected value μ_i , and variance $\phi V(\mu_i)$, where ϕ is a scale parameter and $V(\mu_i) = \mu_i$, then find the quasi-likelihood of μ_i , given an observation y_i , by evaluation (2.12). Confirm that the corresponding deviance is equivalent to the deviance obtained if $Y_i \sim \text{Poi}(\mu_i)$.
5. If a linear model is to be estimated by minimizing $\sum_i w_i(y_i - \mathbf{X}_i\beta)^2$ w.r.t. β , show that the formal expression for the resulting parameter estimates is $\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$ where \mathbf{W} is a diagonal matrix such that $W_{i,i} = w_i$. (Note that the expression is theoretically useful, but would not be used for practical computation).
6. This question relates to the IRLS method of section 2.1.2, and gives an alternative insight into the distribution of the parameter estimators $\hat{\beta}$. Let y_i be independent random variables with mean μ_i , such that $g(\mu_i) = \eta_i \equiv \mathbf{X}_i\beta$, where g is a link function, \mathbf{X} a model matrix and β a parameter vector. Let the variance of y_i be $V(\mu_i)\phi$, where V is a known function, and ϕ a scale parameter. Define

$$z_i = g'(\mu_i)(y_i - \mu_i) + \eta_i \quad \text{and} \quad w_i = \{V(\mu_i)g'(\mu_i)^2\}^{-1}.$$

- (a) Show that $\mathbb{E}(z_i) = \mathbf{X}_i\beta$.
 - (b) Show that the covariance matrix of \mathbf{z} is $\mathbf{W}^{-1}\phi$, where \mathbf{W} is a diagonal matrix with $W_{i,i} = w_i$.
 - (c) If β is estimated by minimization of $\sum_i w_i(z_i - \mathbf{X}_i\beta)^2$ show that the covariance matrix of the resulting estimates, $\hat{\beta}$, is $(\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1}\phi$, and find $\mathbb{E}(\hat{\beta})$.
 - (d) The multivariate version of the central limit theorem implies that as the dimension of \mathbf{z} tends to infinity, $\mathbf{X}^T \mathbf{W} \mathbf{z}$ will tend to multivariate gaussian. What does this imply about the large sample distribution of $\hat{\beta}$?
7. Write R code to implement an IRLS scheme to fit the GLM defined in section 2.3.2 to the data given there. Use the `lm` function to fit the working linear model at each iteration.
8. This question is about GLMs for quite unusual models, handling non-linear parameters, and direct likelihood maximization as a flexible alternative to using

GLMs. Data frame `harrier` has 2 columns: `Consumption`.`Rate` of Grouse by Hen Harriers (per day), and the corresponding `Grouse.Density` (per km²). They have been digitized from figure 1 of Asseburg et al. (2005). Ecological theory suggests that the expected consumption rate, c , should be related to grouse density, d , by the model,

$$\mathbb{E}(c_i) = \frac{ad_i^m}{1 + atd_i^m},$$

where a , t and m are unknown parameters. It is expected that the variance in consumption rate is proportional to the mean consumption rate.

- (a) Show that, for fixed m , a GLM relating c_i and d_i can be obtained by use of the reciprocal link.
- (b) For $m = 1$ estimate the model using `glm` with the `quasi` family.
- (c) Plot the model residuals against Grouse density, and interpret the plot.
- (d) Search for the approximate value of m which minimizes the model deviance, by repeatedly re-fitting the model with alternative trial values.
- (e) For your best fit model, with the optimal m , produce a plot showing the curve of predicted consumption against density overlaid on the raw data. Using the R function `predict`, with the `se` argument set to `TRUE`, add approximate 95% confidence limits to the plot.
- (f) A more systematic way fitting this model is to write a function, which takes the model parameters, and the consumption and density vectors as arguments, and evaluates the model likelihood (or quasi-likelihood in the current context). This likelihood can then be maximized using the R built in optimizer, `optim`. Write R code to do this (see question 4 for the quasi-likelihood).

Note that the `optim` will optionally return the approximate Hessian of a likelihood, so this general approach gives you easy access to everything you need for approximate inference about the parameters of the model, using the results covered in section 2.4. The approach is very general: it would be easy to estimate the full model discussed in Asseburg et al. (2005) in the same way.

- 9. R data frame `ldeaths` contains monthly death rates from 3 lung diseases in the UK over a period of several years (see `?ldeaths` for details and reference). One possible model for the data is that they can be treated as Poisson random variables with a seasonal component and a long term trend component, as follows:

$$\mathbb{E}(\text{deaths}_i) = \beta_0 + \beta_1 t_i + \alpha \sin(2\pi \text{toy}_i / 12 + \phi),$$

where β_0 , β_1 , α and ϕ are parameters t_i is time since the start of the data, and `toy` is time of year, in months (January being month 1).

- (a) By making use of basic properties of sines and cosines, get this model into a form suitable for fitting using `glm`, and fit it. Use `as.numeric(ldeaths)` to treat `ldeaths` as a regular numeric vector rather than a time series object.
- (b) Plot the raw data time series on a plot, with the predicted timeseries overlaid.
- (c) Is the model an adequate fit?

10. In 2.3.2 an approximate confidence interval for β_1 was found using the large sample distribution of the GLM parameter estimators. An alternative method of confidence interval calculation is sometimes useful, based on inversion of the generalized likelihood ratio test (see sections 2.1.6, 2.4.5 and 2.4.6). This works by finding the range of values of β_1 that would have been accepted as null hypotheses about β_1 , using a GLRT. If the threshold for acceptance is 5% the resulting range gives a 95% confidence interval, if the threshold is 1% we get a 99% interval, and so on.
 - (a) Using `glm`, refit the AIDS model, with the quadratic time dependence, and save the resulting object.
 - (b) Write a loop which refits the same model for a sequence of *fixed* β_1 values centered on the MLE from part (a) and stores the resulting model log likelihoods. In this step you are fitting the model under a sequence of null hypotheses about β_1 . The way to fix β_1 in the model fit is to use an `offset` term in your model. e.g. if you want to fix β_1 at 0.5, you would replace the `t` term in your model formula, with the term, `offset (.5*t)`.
 - (c) Plot the log likelihoods from the last part against the corresponding β_1 values, and add to your plot the line above which the log-likelihoods are high enough that the corresponding β_1 values would be included in a 95% confidence interval. From your plot, read off the 95% CI for β_1 and compare it to the interval obtained previously.

CHAPTER 3

Introducing GAMs

3.1 Introduction

A generalized additive model (Hastie and Tibshirani, 1986, 1990) is a generalized linear model with a linear predictor involving a sum of smooth functions of covariates. In general the model has a structure something like

$$g(\mu_i) = \mathbf{X}_i^* \boldsymbol{\theta} + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}) + \dots \quad (3.1)$$

where

$$\mu_i \equiv \mathbb{E}(Y_i) \text{ and } Y_i \sim \text{some exponential family distribution.}$$

Y_i is a response variable, \mathbf{X}_i^* is a row of the model matrix for any strictly parametric model components, $\boldsymbol{\theta}$ is the corresponding parameter vector, and the f_j are smooth functions of the covariates, x_k . The model allows for rather flexible specification of the dependence of the response on the covariates, but by specifying the model only in terms of ‘smooth functions’, rather than detailed parametric relationships, it is possible to avoid the sort of cumbersome and unwieldy models seen in section 2.3.4, for example. This flexibility and convenience comes at the cost of two new theoretical problems. It is necessary both to represent the smooth functions in some way and to choose how smooth they should be.

This chapter illustrates how GAMs can be represented using penalized regression splines, estimated by penalized regression methods, and how the appropriate degree of smoothness for the f_j can be estimated from data using cross validation. To avoid obscuring the basic simplicity of the approach with a mass of technical detail, the most complicated model considered here will be a simple GAM with two univariate smooth components. Furthermore, the methods presented will not be those that are most suitable for general practical use, being rather the methods that enable the basic framework to be explained simply. The ideal way to read this chapter is sitting at a computer working through the statistics, and its implementation in R, side by side. If adopting this approach recall that the help files for R functions can be accessed by typing `? followed by the function name, at the command line (e.g. ?lm, for help on the linear modelling function).`

3.2 Univariate smooth functions

The representation of smooth functions is best introduced by considering a model containing one smooth function of one covariate,

$$y_i = f(x_i) + \epsilon_i, \quad (3.2)$$

where y_i is a response variable, x_i a covariate, f a smooth function and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. To further simplify matters, suppose that the x_i lie in the interval $[0, 1]$.

3.2.1 Representing a smooth function: regression splines

To estimate f , using the methods covered in chapters 1 and 2, requires that f be represented in such a way that (3.2) becomes a linear model. This can be done by choosing a *basis*, defining the space of functions of which f (or a close approximation to it) is an element. Choosing a basis, amounts to choosing some *basis functions*, which will be treated as completely known: if $b_j(x)$ is the j^{th} such basis function, then f is assumed to have a representation

$$f(x) = \sum_{j=1}^q b_j(x)\beta_j, \quad (3.3)$$

for some values of the unknown parameters, β_j . Substituting (3.3) into (3.2) clearly yields a linear model.

A very simple example: a polynomial basis

As a simple example, suppose that f is believed to be a 4th order polynomial, so that the space of polynomials of order 4 and below contains f . A basis for this space is $b_1(x) = 1$, $b_2(x) = x$, $b_3(x) = x^2$, $b_4(x) = x^3$ and $b_5(x) = x^4$, so that (3.3) becomes

$$f(x) = \beta_1 + x\beta_2 + x^2\beta_3 + x^3\beta_4 + x^4\beta_5,$$

and (3.2) becomes the simple model

$$y_i = \beta_1 + x_i\beta_2 + x_i^2\beta_3 + x_i^3\beta_4 + x_i^4\beta_5 + \epsilon_i.$$

Figures 3.1 and 3.2 illustrate a basis function representation of a function, f , using a polynomial basis.

Polynomial bases tend to be very useful for situations in which interest focuses on properties of f in the vicinity of a single specified point, but when the questions of interest relate to f over its whole domain (currently $[0, 1]$), the polynomial bases have some problems (see exercise 1). The *spline* bases perform well in such circumstances, largely because they can be shown to have good approximation theoretic properties.

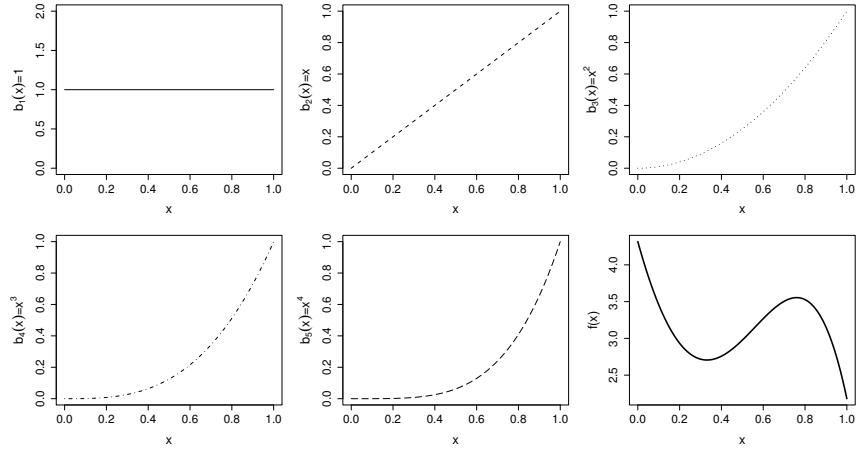


Figure 3.1 *Illustration of the idea of representing a function in terms of basis functions, using a polynomial basis. The first 5 panels (starting from top left), illustrate the 5 basis functions, $b_j(x)$, for a 4th order polynomial basis. The basis functions are each multiplied by a real valued parameter, β_j , and are then summed to give the final curve $f(x)$, an example of which is shown in the bottom right panel. By varying the β_j , we can vary the form of $f(x)$, to produce any polynomial function of order 4 or lower. See also figure 3.2*

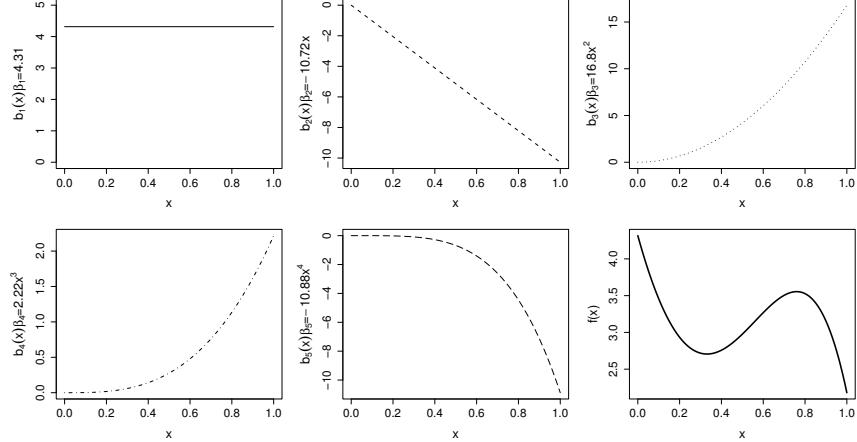


Figure 3.2 *An alternative illustration of how a function is represented in terms of basis functions. As in figure 3.1, a 4th order polynomial basis is illustrated. In this case the 5 basis function, $b_j(x)$, each multiplied by its coefficient β_j , are shown in the first five figures (starting at top left). Simply summing these 5 curves yields the function, $f(x)$, shown at bottom right.*

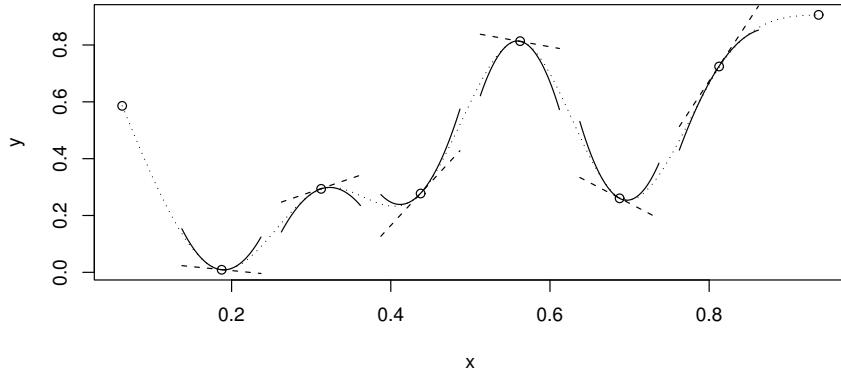


Figure 3.3 A *cubic spline* is a curve constructed from sections of cubic polynomial joined together so that the curve is continuous up to second derivative. The spline shown (dotted curve) is made up of 7 sections of cubic. The points at which they are joined (\circ) (and the two end points) are known as the *knots* of the spline. Each section of cubic has different coefficients, but at the knots it will match its neighbouring sections in value and first two derivatives. Straight dashed lines show the gradients of the spline at the knots and the curved continuous lines are quadratics matching the first and second derivatives at the knots: these illustrate the continuity of first and second derivatives across the knots. This spline has zero second derivatives at the end knots: a ‘natural spline’. Note that there are many alternative ways of representing such a cubic spline, using basis functions: although all are equivalent, the link to the piecewise cubic characterization is not always transparent.

Another example: a cubic spline basis

A univariate function can be represented using a *cubic spline*. A cubic spline is a curve, made up of sections of cubic polynomial, joined together so that they are continuous in value as well as first and second derivatives (see figure 3.3). The points at which the sections join are known as the *knots* of the spline. For a conventional spline, the knots occur wherever there is a datum, but for the regression splines of interest here, the locations of the knots must be chosen. Typically the knots would either be evenly spaced through the range of observed x values, or placed at quantiles of the distribution of unique x values. Whatever method is used, let the knot locations be denoted by $\{x_i^* : i = 1, \dots, q - 2\}$.

Given knot locations, there are many alternative, but equivalent, ways of writing down a basis for cubic splines. A simple basis to use, results from the very general approach to splines that can be found in the books by Wahba (1990) and Gu (2002), although the basis functions are slightly intimidating when written down and the link to the definition of a cubic spline given in figure 3.3 is rather opaque. For this basis:

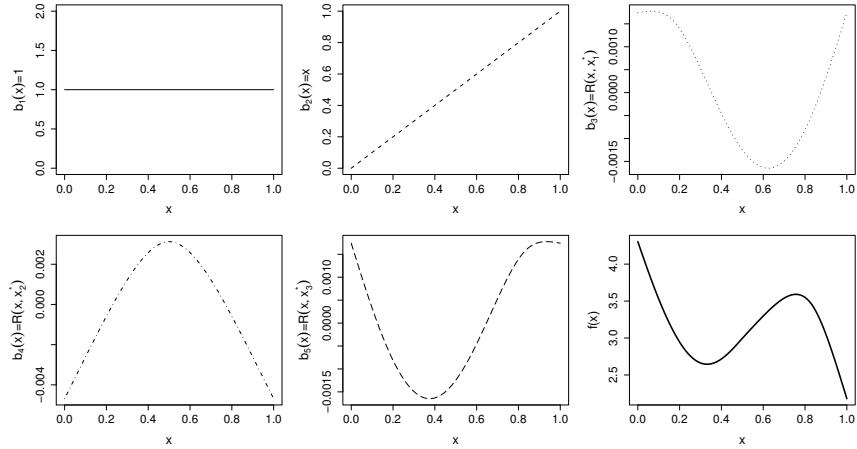


Figure 3.4 Illustration of the representation of a smooth function using the a rank 5 cubic regression spline basis (with knot locations $x_1^* = 1/6$, $x_2^* = 3/6$ and $x_3^* = 5/6$). The first 5 panels (starting from top left), illustrate the 5 basis functions, $b_j(x)$, for a rank 5 cubic spline basis. The basis functions are each multiplied by a real valued parameter, β_j , and are then summed to give the final curve $f(x)$, an example of which is shown in the bottom right panel. By varying the β_j we can vary the form of $f(x)$. See also figure 3.5

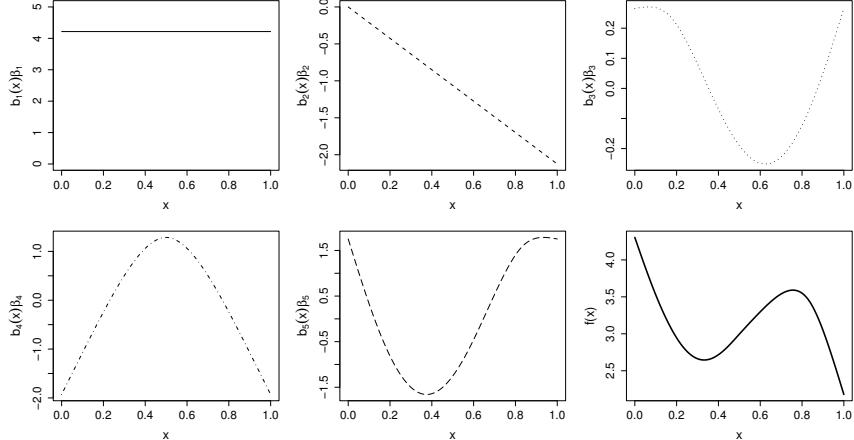


Figure 3.5 An alternative illustration of how a function is represented in terms of cubic spline basis functions. This figure shows the same rank 5 cubic regression spline basis that is shown in figure 3.4, but in this case the basis functions, $b_j(x)$, are each shown multiplied by corresponding coefficients β_j (first five figures, starting at top left). Simply summing these 5 curves yields the function, $f(x)$, shown at bottom right.

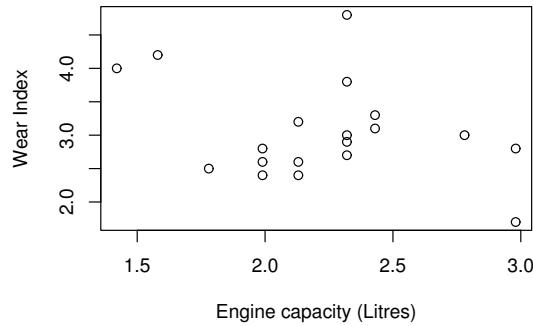


Figure 3.6 Data on engine wear index versus engine capacity for 19 Volvo car engines, obtained from http://www3.bc.sympatico.ca/Volvo_Books/engine3.html

$b_1(x) = 1$, $b_2(x) = x$ and $b_{i+2} = R(x, x_i^*)$ for $i = 1 \dots q - 2$ where

$$R(x, z) = \left[(z - 1/2)^2 - 1/12 \right] \left[(x - 1/2)^2 - 1/12 \right] / 4 \\ - \left[(|x - z| - 1/2)^4 - 1/2 (|x - z| - 1/2)^2 + 7/240 \right] / 24. \quad (3.4)$$

(see Gu, 2002, p.37 for further details). Using this cubic spline basis for f means that (3.2) becomes a linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$, where the i^{th} row of the model matrix is

$$\mathbf{X}_i = [1, x_i, R(x_i, x_1^*), R(x_i, x_2^*), \dots, R(x_i, x_{q-2}^*)].$$

Hence the model can be estimated by least squares. A rank 5 example of this basis is illustrated in figures 3.4 and 3.5.

Using the cubic spline basis

Now consider an illustrative example. It is often claimed, at least by people with little actual knowledge of engines, that a car engine with a larger cylinder capacity will wear out less quickly than a smaller capacity engine. Figure 3.6 shows some data for 19 Volvo engines. The pattern of variation is not entirely clear, so (3.2) might be an appropriate model.

First read the data into R and scale the engine capacity data to lie in [0,1].

```
size<-c(1.42,1.58,1.78,1.99,1.99,1.99,2.13,2.13,2.13,
2.32,2.32,2.32,2.32,2.32,2.43,2.43,2.78,2.98,2.98)
wear<-c(4.0,4.2,2.5,2.6,2.8,2.4,3.2,2.4,2.6,4.8,2.9,
3.8,3.0,2.7,3.1,3.3,3.0,2.8,1.7)
x<-size-min(size);x<-x/max(x)
plot(x,wear,xlab="Scaled engine size",ylab="Wear index")
```

Now write an R function defining $R(x, z)$

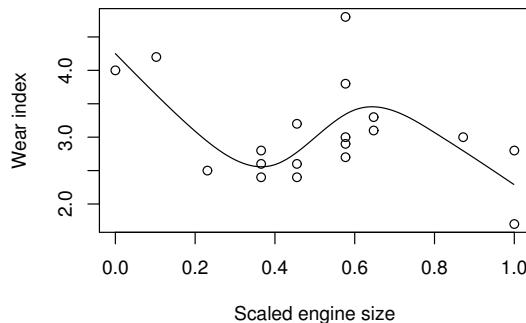


Figure 3.7 Regression spline fit (continuous line) to data (o) on engine wear index versus (scaled) engine capacity for 19 Volvo car engines.

```
rk<-function(x,z) # R(x,z) for cubic spline on [0,1]
{ ((z-0.5)^2-1/12)*((x-0.5)^2-1/12)/4-
  ((abs(x-z)-0.5)^4-(abs(x-z)-0.5)^2/2+7/240)/24
}
```

and use it to write an **R** function which will take a sequence of knots and an array of x values to produce a model matrix for the spline.

```
spl.X<-function(x,xk)
# set up model matrix for cubic penalized regression spline
{ q<-length(xk)+2 # number of parameters
  n<-length(x) # number of data
  X<-matrix(1,n,q) # initialized model matrix
  X[,2]<-x # set second column to x
  X[,3:q]<-outer(x,xk,FUN=rk) # and remaining to R(x,xk)
  X
}
```

All that is required now is to select a set of knots, x_i^* , and the model can be fitted. In the following a rank 6 basis is used, meaning that $q = 6$ and there are 4 knots: these have been evenly spread over $[0,1]$.

```
xk<-1:4/5 # choose some knots
X<-spl.X(x,xk) # generate model matrix
mod.1<-lm(wear~X-1) # fit model
xp<-0:100/100 # x values for prediction
Xp<-spl.X(xp,xk) # prediction matrix
lines(xp,Xp%*%coef(mod.1)) # plot fitted spline
```

The model fit looks quite plausible (figure 3.7), but the choice of degree of model

smoothness, controlled here by the basis dimension, q (i.e. the number of knots + 2), was essentially arbitrary. This issue must be addressed if a satisfactory theory for modelling with smooth functions is to be developed.

3.2.2 Controlling the degree of smoothing with penalized regression splines

One obvious possibility, for choosing the degree of smoothing, is to try to make use of the hypothesis testing methods from chapter 1, to select q by backwards selection. However such an approach is problematic, since a model based on $k - 1$ evenly spaced knots will not generally be nested within a model based on k evenly spaced knots. It is possible to start with a fine grid of knots and simply drop knots sequentially, as part of backward selection, but the resulting uneven knot spacing can itself lead to poor model performance. Furthermore, for these *regression spline* models, the fit of the model tends to depend quite strongly on the locations chosen for the knots.

An alternative to controlling smoothness by altering the basis dimension, is to keep the basis dimension fixed, at a size a little larger than it is believed could reasonably be necessary, but to control the model's smoothness by adding a “wigginess” penalty to the least squares fitting objective. For example, rather than fitting the model by minimizing,

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2,$$

it could be fit by minimizing,

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int_0^1 [f''(x)]^2 dx,$$

where the integrated square of second derivative penalizes models that are too “wiggly”. The trade off between model fit and model smoothness is controlled by the *smoothing parameter*, λ . $\lambda \rightarrow \infty$ leads to a straight line estimate for f , while $\lambda = 0$ results in an un-penalized regression spline estimate.

Because f is linear in the parameters, β_i , the penalty can always be written as a quadratic form in $\boldsymbol{\beta}$ (see exercise 7),

$$\int_0^1 [f''(x)]^2 dx = \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta},$$

where \mathbf{S} is a matrix of known coefficients. It is now that the somewhat complicated form of the spline basis, used here, proves its worth, for it turns out that $S_{i+2,j+2} = R(x_i^*, x_j^*)$ for $i, j = 1, \dots, q - 2$ while the first two rows and columns of \mathbf{S} are 0 (Gu, 2002, p.34).

Therefore, the penalized regression spline fitting problem is to minimize

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta} \tag{3.5}$$

w.r.t. $\boldsymbol{\beta}$. The problem of estimating the degree of smoothness for the model is now

the problem of estimating the smoothing parameter λ . But before addressing λ estimation, consider β estimation, given λ .

It is fairly straightforward to show (see exercise 4) that the formal expression for the minimizer of (3.5), the penalized least squares estimator of β , is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{S})^{-1} \mathbf{X}^T \mathbf{y}. \quad (3.6)$$

Similarly the influence, or hat matrix, \mathbf{A} , for the model can be written

$$\mathbf{A} = \mathbf{X} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{S})^{-1} \mathbf{X}^T.$$

Recall that $\hat{\mu} = \mathbf{Ay}$. Of course, these expressions are not the ones to use for computation, for which the greater numerical stability offered by orthogonal methods is to be preferred.

For practical computation therefore, note that

$$\left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{B} \end{bmatrix} \beta \right\|^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \beta^T \mathbf{S} \beta.$$

where \mathbf{B} is any square root of the matrix \mathbf{S} such that $\mathbf{B}^T \mathbf{B} = \mathbf{S}$. The sum of squares term, on the right hand side, is just a least squares objective for a model in which the model matrix has been augmented by a square root of the penalty matrix, while the response data vector has been augmented with q zeros. \mathbf{B} can be obtained easily by spectral decomposition or pivoted Choleski decomposition (see A.7 or A.8), and once obtained the augmented least squares problem can be solved using orthogonal methods, in order to solve the penalized least squares problem and fit the model.

To see a penalized regression spline in action, involves first writing a function to obtain \mathbf{S} .

```
spl.S<-function(xk)
# set up the penalized regression spline penalty matrix,
# given knot sequence xk
{ q<-length(xk)+2; S<-matrix(0,q,q) # initialize matrix to 0
  S[3:q,3:q]<-outer(xk,xk,FUN=rk) # fill in non-zero part
  S
}
```

A square root function is also needed in order to find $\mathbf{B} = \sqrt{\mathbf{S}}$: using the spectral decomposition is simple, if a little inefficient (see section A.8).

```
mat.sqrt<-function(S) # A simple matrix square root
{ d<-eigen(S, symmetric=TRUE)
  rS<-d$vectors %*% diag(d$values^0.5) %*% t(d$vectors)
}
```

Now the ingredients are in place to write a simple function for fitting a penalized regression spline.

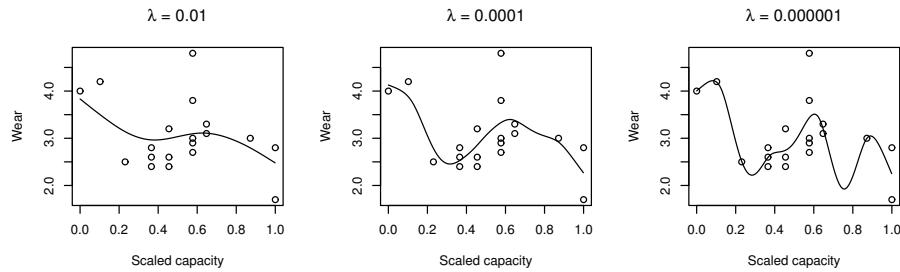


Figure 3.8 Penalized regression spline fits to the engine wear versus capacity data using three different values for the smoothing parameter.

```
prs.fit<-function(y,x,xk,lambda)
# function to fit penalized regression spline to x,y data,
# with knots xk, given smoothing parameter, lambda.
{ q<-length(xk)+2           # dimension of basis
  n<-length(x)               # number of data
  # create augmented model matrix ....
  Xa <- rbind(spl.X(x,xk),mat.sqrt(spl.S(xk)))*sqrt(lambda))
  y[(n+1):(n+q)]<-0 # augment the data vector
  lm(y~Xa-1) # fit and return penalized regression spline
}
```

To use this function, we need to choose the basis dimension, q , the knot locations, x_j^* , and a value for the smoothing parameter, λ . Provided that q is large enough that the basis is more flexible than we expect to *need* to represent $f(x)$, then neither the exact choice of q , nor the precise selection of knot locations, has a great deal of influence on the model fit. Rather it is the choice of λ that now plays the crucial role in determining model flexibility, and ultimately the estimated shape of $\hat{f}(x)$. In the following example $q = 9$ and the knots are evenly spread out over $[0,1]$. But it is the smoothing parameter, $\lambda = 10^{-4}$, which really controls the behaviour of the fitted model.

```
xk<-1:7/8 # choose some knots
mod.2<-prs.fit(wear,x,xk,0.0001) # fit pen. reg. spline
Xp<-spl.X(xp,xk) # matrix to map params to fitted values at xp
plot(x,wear);lines(xp,Xp%*%coef(mod.2)) # plot data & spl. fit
```

By changing the value of the smoothing parameter, λ , a variety of models of different smoothness can be obtained. Figure 3.8 illustrates this, but begs the question, which value of λ is ‘best’?

3.2.3 Choosing the smoothing parameter; λ : cross validation

If λ is too high then the data will be over smoothed, and if it is too low then the data will be under smoothed: in both cases this will mean that the spline estimate \hat{f} will not be close to the true function f . Ideally, it would be good to choose λ so that \hat{f} is as close as possible to f . A suitable criterion might be to choose λ to minimize

$$M = \frac{1}{n} \sum_{i=1}^n (\hat{f}_i - f_i)^2,$$

where the notation $\hat{f}_i \equiv \hat{f}(x_i)$ and $f_i \equiv f(x_i)$ have been adopted for conciseness. Since f is unknown, M cannot be used directly, but it is possible to derive an estimate of $\mathbb{E}(M) + \sigma^2$, which is the expected squared error in predicting a new variable. Let $\hat{f}^{[-i]}$ be the model fitted to all data except y_i , and define the *ordinary cross validation* score

$$\mathcal{V}_o = \frac{1}{n} \sum_{i=1}^n (\hat{f}_i^{[-i]} - y_i)^2.$$

This score results from leaving out each datum in turn, fitting the model to the remaining data and calculating the squared difference between the missing datum and its predicted value: these squared differences are then averaged over all the data. Substituting $y_i = f_i + \epsilon_i$,

$$\begin{aligned} \mathcal{V}_o &= \frac{1}{n} \sum_{i=1}^n (\hat{f}_i^{[-i]} - f_i - \epsilon_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\hat{f}_i^{[-i]} - f_i)^2 - (\hat{f}_i^{[-i]} - f_i)\epsilon_i + \epsilon_i^2. \end{aligned}$$

Since $\mathbb{E}(\epsilon_i) = 0$, and ϵ_i and $\hat{f}_i^{[-i]}$ are independent, the second term in the summation vanishes if expectations are taken:

$$\mathbb{E}(\mathcal{V}_o) = \frac{1}{n} \mathbb{E} \left(\sum_{i=1}^n (\hat{f}_i^{[-i]} - f_i)^2 \right) + \sigma^2.$$

Now, $\hat{f}^{[-i]} \approx \hat{f}$ with equality in the large sample limit, so $\mathbb{E}(\mathcal{V}_o) \approx \mathbb{E}(M) + \sigma^2$ also with equality in the large sample limit. Hence choosing λ in order to minimize \mathcal{V}_o is a reasonable approach if the ideal would be to minimize M . Choosing λ to minimize \mathcal{V}_o is known as ordinary cross validation.

Ordinary cross validation is a reasonable approach, in its own right, even without a mean square (prediction) error justification. If models are judged only by their ability to fit the data from which they were estimated, then complicated models are always selected over simpler ones. Choosing a model in order to maximize the ability to predict data to which the model was not fitted, does not suffer from this problem, as figure 3.9 illustrates.

It is computationally inefficient to calculate \mathcal{V}_o by leaving out one datum at a time,

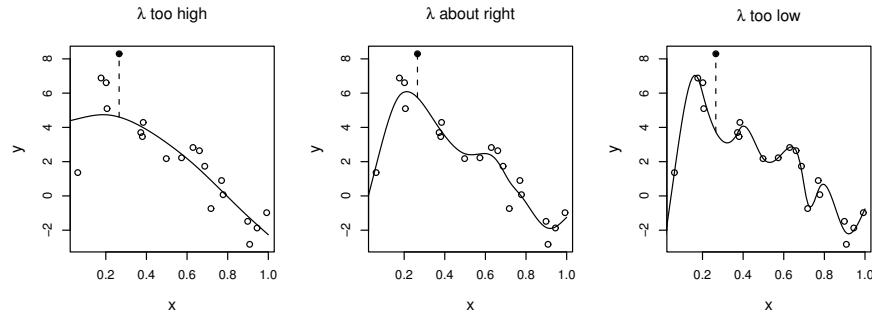


Figure 3.9 *Illustration of the principle behind cross validation. In this case the fifth datum (●) has been omitted from fitting and the continuous line shows a penalized regression spline fitted to the remaining data (○). When the smoothing parameter is too high the spline fits many of the data poorly and does no better with the missing point. When λ is too low the spline fits the noise as well as the signal and the extra variability that this induces causes it to predict the missing datum rather poorly. For the intermediate λ the spline is fitting the underlying signal quite well, but smoothing through the noise: as a result the missing datum is reasonably well predicted. Cross validation leaves out each datum from the data in turn and considers the average ability of models fitted to the remaining data to predict the left out datum.*

and fitting the model to each of the n resulting data sets, but fortunately it can be shown that

$$\mathcal{V}_o = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}_i)^2 / (1 - A_{ii})^2,$$

where \hat{f} is the estimate from fitting to all the data, and \mathbf{A} is the corresponding influence matrix (see section 4.5.2). In practice the weights, $1 - A_{ii}$, are often replaced by the mean weight, $\text{tr}(\mathbf{I} - \mathbf{A})/n$, in order to arrive at the *generalized cross validation* score

$$\mathcal{V}_g = \frac{n \sum_{i=1}^n (y_i - \hat{f}_i)^2}{[\text{tr}(\mathbf{I} - \mathbf{A})]^2}.$$

GCV has computational advantages over OCV, and it also has advantages in terms of invariance (see Wahba, 1990, p.53 or sections 4.5.2 and 4.5.3). In any case, it can also be shown to minimize $\mathbb{E}(M)$ in the large sample limit.

Returning to the engine wear example, a simple direct search for the GCV optimal smoothing parameter can be made as follows.

```
lambda<-1e-8;n<-length(wear);V<-0
for (i in 1:60) {
  mod<-prsr.fit(wear,x,xk,lambda)      # fit model, given lambda
  trA<-sum(influence(mod)$hat[1:n])    # find tr(A)
  rss<-sum((wear-fitted(mod)[1:n])^2) # residual sum of squares
  V[i]<-n*rss/(n-trA)^2                # obtain GCV score
  lambda<-lambda*1.5                     # increase lambda
}
```

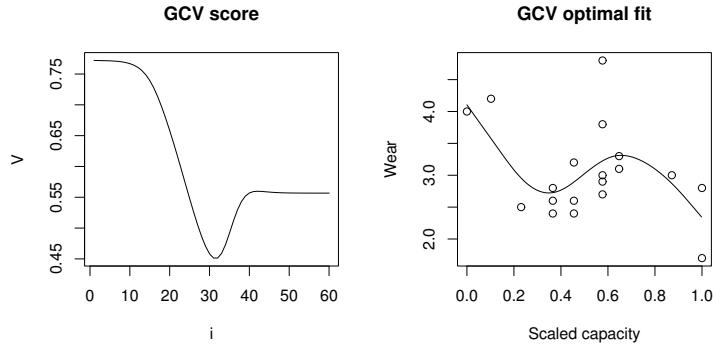


Figure 3.10 *Left panel:* the GCV function for the engine wear example. The smoothing parameters are shown on a log scale on the x axis, such that $\lambda = 1.5^i \times 10^{-8}$. *Right panel:* the fitted model which minimizes the GCV score.

```
plot(1:60,V,type="l",main="GCV score",xlab="i") # plot score
```

Note that the `influence()` function returns a list of diagnostics including `hat`, an array of the elements on the leading diagonal of the influence/hat matrix of the augmented model. The first n of these are the leading diagonal of the influence matrix of the penalized model (see exercise 5).

For the example, $V[31]$ is the lowest GCV score, so that the optimal smoothing parameter, from those tried, is $\hat{\lambda} = 1.5^{30} \times 10^{-8} \approx 0.002$. A plot of the optimal model is easily produced

```
i<-(1:60)[V==min(V)] # extract index of min(V)
mod.3<-prsr.fit(wear,x,xk,1.5^(i-1)*1e-8) # fit optimal model
Xp<-spl.X(xp,xk) # .... and plot it
plot(x,wear);lines(xp,Xp%*%coef(mod.3))
```

The GCV function and the GCV optimal model are shown in figure 3.10.

3.3 Additive Models

Now suppose that two explanatory variables, x and z , are available for a response variable, y , and that a simple additive model structure

$$y_i = f_1(x_i) + f_2(z_i) + \epsilon_i \quad (3.7)$$

is appropriate. The f_j are smooth functions, and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. Again, for simplicity, assume that all z_i and x_i lie in $[0, 1]$.

There are two points to note about this model. Firstly, the assumption of additive effects is a fairly strong one: $f_1(x) + f_2(z)$ is a quite restrictive special case of the general smooth function of two variables $f(x, z)$. Secondly, the fact that the model

now contains more than one function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant. To see this, note that any constant could be simultaneously added to f_1 and subtracted from f_2 , without changing the model predictions. Hence identifiability constraints have to be imposed on the model before fitting.

Provided the identifiability issue is addressed, the additive model can be represented using penalized regression splines, estimated by penalized least squares and the degree of smoothing estimated by cross validation, in the same way as the simple univariate model.

3.3.1 Penalized regression spline representation of an additive model

Each smooth function in (3.7) can be represented using a penalized regression spline basis. Using the spline basis from section 3.2.1

$$f_1(x) = \delta_1 + x\delta_2 + \sum_{j=1}^{q_1-2} R(x, x_j^*)\delta_{j+2}$$

and

$$f_2(z) = \gamma_1 + z\gamma_2 + \sum_{j=1}^{q_2-2} R(z, z_j^*)\gamma_{j+2}$$

where δ_j and γ_j are the unknown parameters for f_1 and f_2 respectively. q_1 and q_2 are the number of unknown parameters for f_1 and f_2 , while x_j^* and z_j^* are the knot locations for the two functions.

The identifiability problem with the additive model means that δ_1 and γ_1 are confounded. The simplest way to deal with this is to constrain one of them to zero, say $\gamma_1 = 0$. Having done this, it is easy to see that the additive model can be written in the linear model form, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where the i^{th} row of the model matrix is now

$$\mathbf{x}_i = [1, x_i, R(x_i, x_1^*), R(x_i, x_2^*), \dots, R(x_i, x_{q_1-2}^*), z_i, R(z_i, z_1^*), \dots, R(z_i, z_{q_2-2}^*)]$$

and the parameter vector is $\boldsymbol{\beta} = [\delta_1, \delta_2, \dots, \delta_{q_1}, \gamma_2, \gamma_3, \dots, \gamma_{q_2}]^\top$.

The wigginess of the functions can also be measured exactly as in section 3.2.2.

$$\int_0^1 f_1''(x)^2 dx = \boldsymbol{\beta}^\top \mathbf{S}_1 \boldsymbol{\beta} \quad \text{and} \quad \int_0^1 f_2''(x)^2 dx = \boldsymbol{\beta}^\top \mathbf{S}_2 \boldsymbol{\beta}$$

where \mathbf{S}_1 and \mathbf{S}_2 are zero everywhere except for $\mathbf{S}_{1,i+2,j+2} = R(x_i^*, x_j^*)$ for $i, j = 1, \dots, q_1 - 2$ and $\mathbf{S}_{2,i+q_1+1,j+q_1+1} = R(z_i^*, z_j^*)$ for $i, j = 1, \dots, q_2 - 2$.

It is, of course, perfectly possible to use any of a large number of alternative bases in place of the regression spline basis used here — only the details are changed by doing this, not the general principle that, once a basis has been chosen, model matrices and penalty coefficient matrices can immediately be obtained.

3.3.2 Fitting additive models by penalized least squares

The parameters β of the model (3.7) are obtained by minimization of the penalized least squares objective

$$\|y - X\beta\|^2 + \lambda_1\beta^T S_1\beta + \lambda_2\beta^T S_2\beta,$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of making f_1 and f_2 smooth, relative to the objective of closely fitting the response data. For the moment, assume that these smoothing parameters are given.

Defining $S \equiv \lambda_1 S_1 + \lambda_2 S_2$, the objective can be re-written as

$$\|y - X\beta\|^2 + \beta^T S \beta = \left\| \begin{bmatrix} y \\ 0 \end{bmatrix} - \begin{bmatrix} X \\ B \end{bmatrix} \beta \right\|^2.$$

where B is any matrix square root such that $B^T B = S$. As in the single smooth case, the right hand expression is simply the un-penalized least squares objective for an augmented version of the model and corresponding response data: hence the model can be fitted by standard linear regression.

Here is a function to set up a simple two term additive model, if x and z are the two predictor variables.

```
am.setup<-function(x,z,q=10)
# Get X, S_1 and S_2 for a simple 2 term AM
{ # choose knots ...
  xk <- quantile(unique(x),1:(q-2)/(q-1))
  zk <- quantile(unique(z),1:(q-2)/(q-1))
  # get penalty matrices ...
  S <- list()
  S[[1]] <- S[[2]] <- matrix(0,2*q-1,2*q-1)
  S[[1]][2:q,2:q] <- spl.S(xk)[-1,-1]
  S[[2]][(q+1):(2*q-1),(q+1):(2*q-1)] <- spl.S(zk)[-1,-1]
  # get model matrix ...
  n<-length(x)
  X<-matrix(1,n,2*q-1)
  X[,2:q]<-spl.X(x,xk)[,-1] # 1st smooth
  X[, (q+1):(2*q-1)]<-spl.X(z,zk)[,-1] # 2nd smooth
  list(X=X,S=S)
}
```

The same number of knots is assumed for each term in this case, and they have been evenly spread through the data by using the `quantile` function. The penalty matrices are obtained using `spl.S`, while the model matrix is constructed by combining columns of the model matrices obtained by calling `spl.X` for each predictor. Note that the rows and columns of S_1 and S_2 corresponding to the intercept of each term have been dropped, as have the intercept columns of the component matrices of X . The routine returns a two item list containing the model matrix for the additive model and a list containing the two penalty matrices.

It is now a straightforward matter to write a function that will take the response variable, \mathbf{X} , the penalty matrix list, and the smoothing parameters, as arguments, calculate the corresponding augmented model matrix and data vector, fit the model and calculate its GCV score:

```
fit.am<-function(y,X,S,sp)
# function to fit simple 2 term additive model
{ # get sqrt of total penalty matrix ...
  rS <- mat.sqrt(sp[1]*S[[1]]+sp[2]*S[[2]])
  q.tot <- ncol(X)                      # number of params
  n <- nrow(X)                          # number of data
  X1 <- rbind(X,rS)                     # augmented X
  y1<-y;y1[(n+1):(n+q.tot)]<-0      # augment data
  b<-lm(y1~X1-1)                       # fit model
  trA<-sum(influence(b)$hat[1:n])    # tr(A)
  norm<-sum((y-fitted(b)[1:n])^2)     # RSS
  list(model=b,gcv=norm*n/(n-trA)^2,sp=sp)
}
```

Let us use the routine to estimate an additive model for the data in R data frame `trees`. The data are `Volume`, `Girth` and `Height` for 31 felled cherry trees. Interest lies is in predicting `Volume`, and we can try estimating the model

$$\text{Volume} = f_1(\text{Girth}) + f_2(\text{Height}) + \epsilon_i$$

using the simple functions just produced. Given the simple smoothers being used here, we must first rescale the predictors onto [0,1]

```
data(trees)
rg <- range(trees$Girth)
trees$Girth <- (trees$Girth - rg[1])/(rg[2]-rg[1])
rh <- range(trees$Height)
trees$Height <- (trees$Height - rh[1])/(rh[2]-rh[1])
```

Then the model matrix and penalty matrices can be obtained.

```
am0 <- am.setup(trees$Girth,trees$Height)
```

Given these, a grid search can be performed to find the model fit that approximately minimizes the GCV score.

```
sp<-c(0,0)  # initialize smoothing parameter (s.p.) array
for (i in 1:30) for (j in 1:30)          # loop over s.p. grid
{ sp[1]<-1e-5*2^(i-1);sp[2]<-1e-5*2^(j-1) # s.p.s
  b<-fit.am(trees$Volume,am0$X,am0$S,sp)  # fit using s.p.s
  if (i+j==2) best<-b else                  # store 1st model
  if (b$gcv<best$gcv) best<-b             # store best model
}
best$sp  # GCV best smoothing parameter found
[1] 0.01024 5368.70912
```

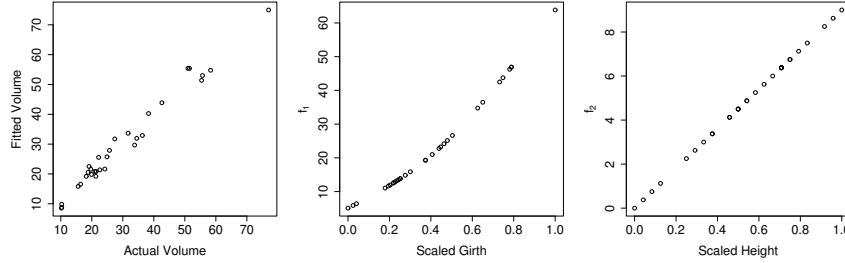


Figure 3.11 *The best fit two term additive model for the tree data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of Girth, evaluated at the observed Girths. The right panel is the estimate of the smooth function of Height, evaluated at the observed Heights.*

So the smooth of girth has a fairly low smoothing parameter, presumably allowing f_1 some curvature, while the f_2 has a very high smoothing parameter corresponding to a straight line estimate. The values of the smooths at the predictor variable values can be obtained quite easily by zeroing all model coefficients, except those corresponding to the term of interest, and using `predict` as the following code shows.

```
# plot fitted against data ...
plot(trees$Volume,fitted(best$model)[1:31],
      xlab="Fitted Volume",ylab="Actual Volume")
# evaluate and plot f_1 against Girth ...
b<-best$model
b$coefficients[1]<-0      # zero the intercept
b$coefficients[11:19]<-0 # zero the second smooth coeffs
f0<-predict(b)           # predict f_1 only, at data values
plot(trees$Girth,f0[1:31],xlab="Scaled Girth",
      ylab=expression(hat(f[1])))
```

Similar code can be produced in order to plot \hat{f}_2 : figure 3.11 shows the model fitted values against data, as well as \hat{f}_1 against girth and \hat{f}_2 against height.

3.4 Generalized Additive Models

Generalized additive models (GAMs) follow from additive models, as generalized linear models follow from linear models. That is, the linear predictor now predicts some known smooth monotonic function of the expected value of the response, and the response may follow any exponential family distribution, or simply have a known mean variance relationship, permitting the use of a quasi-likelihood approach. The resulting model has a general form something like (3.1) in section 3.1.

As an illustration, suppose that we would like to model the `trees` data using a GAM

of the form:

$$\log\{\mathbb{E}(\text{Volume}_i)\} = f_1(\text{Girth}_i) + f_2(\text{Height}_i), \quad \text{Volume}_i \sim \text{Gamma}.$$

This model is perhaps more natural than the additive model, as we might expect volume to be the product of some function of girth and some function of height, and it is probably reasonable to expect the variance in volume to increase with mean volume.

Now it is tempting to suppose that all that is needed, to fit this GAM, is to replace the call to `lm` with a call to `glm` in `fit.am`, and perhaps tweak the definition of the GCV score a little. Unfortunately, further reflection reveals that this is not the case. Whereas the additive model was estimated by penalized least squares, the GAM will be fitted by penalized likelihood maximization: in practice this will be achieved by penalized iterative least squares, but there is no simple trick to produce an unpenalized GLM whose likelihood is equivalent to the penalized likelihood of the GAM that we wish to fit.

To fit the model we simply iterate the following penalized iteratively re-weighted least squares (P-IRLS) scheme to convergence.

- Given current parameter estimates $\beta^{[k]}$, and corresponding estimated mean response vector $\mu^{[k]}$, calculate:

$$w_i \propto \frac{1}{V(\mu_i^{[k]})g'(\mu_i^{[k]})} \quad \text{and} \quad z_i = g(\mu_i^{[k]})(y_i - \mu_i^{[k]}) + \mathbf{X}_i \beta^{[k]}$$

where $\text{var}(Y_i) = V(\mu^{[k]})\phi$ as in section 2.1.2, and \mathbf{X}_i is the i^{th} row of \mathbf{X} .

- Minimize

$$\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\beta)\|^2 + \lambda_1 \beta^\top \mathbf{S}_1 \beta + \lambda_2 \beta^\top \mathbf{S}_2 \beta$$

w.r.t. β to obtain $\beta^{[k+1]}$. \mathbf{W} is a diagonal matrix such that $W_{ii} = w_i$.

Step 2 can be replaced by the equivalent:

- Minimize

$$\left\| \begin{bmatrix} \sqrt{\mathbf{W}} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \left(\begin{bmatrix} \mathbf{z} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix} \beta \right) \right\|^2$$

w.r.t. β to obtain $\beta^{[k+1]}$, where \mathbf{B} is a matrix square root such that $\mathbf{B}^\top \mathbf{B} = \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2$.

In the current case, the link function, g , is the log, so $g'(\mu_i) = \mu_i^{-1}$, while for the gamma distribution, $V(\mu_i) = \mu_i^2$. Hence, for the log-link, gamma errors model, we have:

$$w_i = 1 \quad \text{and} \quad z_i = (y_i - \mu_i^{[k]}) / \mu_i^{[k]} + \mathbf{X}_i \beta^{[k]}.$$

What should be used for the GCV score for this model? A natural choice is to use the GCV score for the final linear model in the P-IRLS iteration (although we will see, in chapter 4, that there may be better choices than this).

It is now a straightforward matter to modify `fit.am`, in order to produce a function that will fit this GAM.

```

fit.gamG<-function(y,X,S,sp)
# function to fit simple 2 term generalized additive model
# Gamma errors and log link
{ # get sqrt of combined penalty matrix ...
  rS <- mat.sqrt(sp[1]*S[[1]]+sp[2]*S[[2]])
  q <- ncol(X)    # number of params
  n <- nrow(X)    # number of data
  X1 <- rbind(X,rS)    # augmented model matrix
  b <- rep(0,q);b[1] <- 1    # initialize parameters
  norm <- 0;old.norm <- 1    # initialize convergence control
  while (abs(norm-old.norm)>1e-4*norm)  # repeat un-converged
  { eta <- (X1%*%b)[1:n]    # 'linear predictor'
    mu <- exp(eta)          # fitted values
    z <- (y-mu)/mu + eta    # pseudodata (recall w_i=1, here)
    z[(n+1):(n+q)] <- 0    # augmented pseudodata
    m <- lm(z~X1-1)        # fit penalized working model
    b <- m$coefficients    # current parameter estiamtes
    trA <- sum(influence(m)$hat[1:n]) # tr(A)
    old.norm <- norm        # store for convergence test
    norm <- sum((z-fitted(m))[1:n]^2) # RSS of working model
  }
  list(model=m,gcv=norm*n/(n-trA)^2,sp=sp)
}

```

To use this function to find the GCV optimum fit, we simply replace ‘`fit.am`’ by ‘`fit.gamG`’ in the smoothing parameter grid search loop given in section 3.3.2. Again, the smooth of `Girth` is estimated to be more flexible than the smooth of `Height`. The same code as was used to plot the model fit and estimated components of the fit, for the additive model, can be used to produce equivalent plots for the GAM (although the inverse of the link — the exponential function — must be applied to the fitted values from the working linear model when producing the first plot). Figure 3.12 shows the results of the fitting exercise.

3.5 Summary

This chapter has illustrated how models based on smooth functions of predictor variables can be represented, and estimated, once a basis and wiggliness penalty have been chosen for each smooth in the model. Model estimation is by penalized versions of the least squares or maximum-likelihood/IRLS methods, by which linear models or generalized linear models are fitted, since, given a basis, an additive model or GAM is simply a linear model or GLM, with one or more associated penalties. The additional problem, in working with additive models and GAMs, is that we have to choose how much to penalize the fitting process, but GCV seems to provide a quite reasonable solution.

Everything in this chapter has deliberately been kept as straightforward as possible, in order to try and emphasize the basic simplicity of this sort of modelling. If the

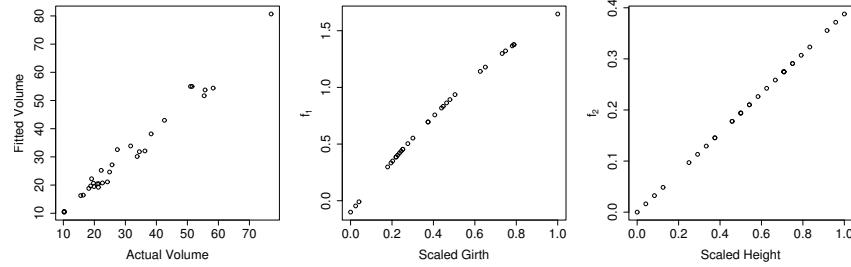


Figure 3.12 *The best fit two term generalized additive model for the tree data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of Girth, evaluated at the observed Girths. The right panel is the estimate of the smooth function of Height, evaluated at the observed Heights.*

material here has been thoroughly understood, then most of what follows in the next chapter simply adds detail to the general framework. It should be clear, for example: that we could use a wide range of alternative bases in place of the bases employed here; that representing smooth functions of more than one variable requires that we choose basis functions of more than one variable, but changes nothing else; that generalizing to more smooth functions in a model is entirely trivial; that dealing with other link functions and distributions involves programming, but nothing conceptually new; that deciding to move GCV optimization to within the linear model call of the P-IRLS is a change in detail, but not concept; that model checking will be similar to what is done for linear models and GLMs, and so on.

3.6 Exercises

1. This question is about illustrating the problems with polynomial bases. First run

```
set.seed(1)
x<-sort(runif(40)*10)^.5
y<-sort(runif(40))^0.1
```

to simulates some apparently innocuous x, y data.

- (a) Fit 5th and 10th order polynomials to the simulated data using e.g.
 $\text{lm}(y \sim \text{poly}(x, 5))$.
- (b) Plot the x, y data, and overlay the fitted polynomials. (Use the `predict` function to obtain predictions on a fine grid over the range of the x data: only predicting at the data, fails to illustrate the polynomial behaviour adequately).
- (c) One particularly simple basis for a cubic regression spline is $b_1(x) = 1$, $b_2(x) = x$ and $b_{j+2}(x) = |x - x_j^*|^3$ for $j = 1 \dots q - 2$, where q is the basis dimension, and the x_j^* are knot locations. Use this basis to fit a rank 11 cubic regression spline to the x, y data (using `lm` and evenly spaced knots).

- (d) Overlay the predicted curve according to the spline model, onto the existing x, y plot, and consider which basis you would rather use.
2. Polynomial models of the data from question 1 can also provide an illustration of why orthogonal matrix methods are preferable to fitting models by solution of the ‘normal equations’ $\mathbf{X}^T \mathbf{X} \boldsymbol{\beta} = \mathbf{X}^T \mathbf{y}$. The bases produced by `poly` are actually orthogonal polynomial bases, which are a numerically stable way of representing polynomial models, but if a naive basis is used then a numerically badly behaved model can be produced:

```
form<-paste("I(x^",1:10,")",sep="",collapse="+")
form <- as.formula(paste("y~",form))
```

produces the model formula for a suitably ill-behaved model. Fit this model using `lm`, extract the model matrix from the fitted model object using `model.matrix`, and re-estimate the model parameters by solving the ‘normal equations’ given above (see `?solve`). Compare the estimated coefficients in both cases, along with the fits. It is also instructive to increase the order of the polynomial by one or two and examine the results (and to decrease it to 5, say, in order to confirm that the QR and normal equations approaches agree if everything is ‘well behaved’). Finally, note that the singular value decomposition (see A.9) provides a reliable way of diagnosing the linear dependencies that can cause problems when model fitting. `svd(X)$d` obtains the singular values of a matrix X . The largest divided by the smallest gives the ‘condition number’ of the matrix — a measure of how ‘close’ it is to a matrix with non-independent columns.

3. Splines are not the only basis-penalty smoothers. Quite reasonable ‘piecewise linear smoothers’ can be constructed based on simple linear interpolation. This question is about implementing such a smoother, and using it to smooth the `mcycle` data from the `MASS` package. Consider smoothing x, y data. A piecewise linear smoother is based on the piecewise linear interpolant through a set of function values f_j^* at a set of evenly space x values, x_j^* : the f^* would be the coefficients of the smooth, the x_j^* are ‘knot locations’.
- (a) In order to obtain the model matrix for a smooth, there is actually no need to explicitly write down its basis functions: all that is required is to be able to evaluate the smooth $f(x)$ for any x value, given its coefficients, $\boldsymbol{\beta}$. This is because $f(x) = \sum_j \beta_j b_j(x)$, so that if all β_j ’s are set to zero, except for β_k , which is set to one, then $f(x) = b_k(x)$: this fact provides an easy way of evaluating the basis functions of f given a function evaluating f itself.
 Make use of this idea to write a function which will take an array of x values and obtain the model matrix corresponding to a piecewise linear smoother, having m knots spread evenly through the range of the x values. Do this by making use of the `approx` function in `R`.
- (b) Modify your function so that it also takes a y vector argument of response data to be smoothed, and fits the piecewise linear smoother to the supplied data. Have the function return the estimated model coefficients and fitted values in a list.
- (c) Use your function to model the `mcycle` data (`accel` is the response): plot the

fitted values over the raw data, trying values of m of 10 and 20. You will probably find that controlling the smoothness by modifying m is a bit unsatisfactory: the estimates either wiggle too much or miss the data.

- (d) To improve the smoother it helps to introduce a wigginess penalty. Given the meaning of the model coefficients, very simple difference penalties on the coefficients can be used. For example, it might be appropriate to penalize:

$$P_1 = \sum_{j=2}^m (\beta_{i+1} - \beta_i)^2 \quad \text{or} \quad P_2 = \sum_{j=2}^{m-1} (\beta_{i+1} - 2\beta_i + \beta_{i-1})^2.$$

(recall that $\beta_j = f(x_j^*)$.) The `diff` function makes computational work with such penalties very easy. If `diff(diag(m), differences=j)` returns the matrix \mathbf{D}_j , show that $P_j = \boldsymbol{\beta}^T \mathbf{D}_j^T \mathbf{D}_j \boldsymbol{\beta}$ for $j = 1, 2$.

- (e) Modify your piecewise linear smoother function to fit the smoother by penalized regression using the P_2 penalty. The function should now take a smoothing parameter as an argument (and should use `diff` to find the square root of the penalty directly).
- (f) Try out your function on the `mcycle` data, using different values for the smoothing parameter and $m = 20$.
- (g) Modify your functions once more, so that it returns the GCV score for the model, in addition to the coefficients and fitted values. Search for the GCV optimal smoothing parameter, and produce a final plot illustrating its fit.
4. Show that the $\boldsymbol{\beta}$ minimizing (3.5), in section 3.2.2, is given by (3.6).
5. Let \mathbf{X} be an $n \times p$ model matrix, \mathbf{S} a $p \times p$ penalty matrix, and \mathbf{B} any matrix such that $\mathbf{B}^T \mathbf{B} = \mathbf{S}$. If

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix}$$

is an augmented model matrix, show that the sum of the first n elements on the leading diagonal of $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T$ is $\text{tr}(\mathbf{X}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T)$.

6. The ‘obvious’ way to estimate smoothing parameters is by treating smoothing parameters just like the other model parameters, $\boldsymbol{\beta}$, and to choose λ to minimize the residual sum of squares for the fitted model. What estimate of λ will such an approach always produce?
7. Show that for any function f , which has a basis expansion

$$f(x) = \sum_j \beta_j b_j(x),$$

it is possible to write

$$\int f''(x)^2 dx = \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta},$$

where the coefficient matrix \mathbf{S} can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives). As usual $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.

8. Show that for any function f which has a basis expansion

$$f(x, z) = \sum_j \beta_j b_j(x, z),$$

it is possible to write

$$\int \left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial z} \right)^2 + \left(\frac{\partial^2 f}{\partial z^2} \right)^2 dx dz = \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta},$$

where the coefficient matrix \mathbf{S} can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives w.r.t. x and z). Again, $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.



CHAPTER 4

Some GAM theory

In the last chapter, it was demonstrated how the problem of estimating a generalized additive model, becomes the problem of estimating smoothing parameters and model coefficients for a penalized likelihood maximization problem, once a basis for the smooth functions has been chosen, together with associated measures of function wigginess. In practice the penalized likelihood maximization problem is solved by penalized iteratively re-weighted least squares (P-IRLS), while the smoothing parameters can be estimated using cross validation or related criteria. The purpose of this chapter is to justify and extend the methods introduced in chapter 3, and to add some distribution theory to facilitate confidence interval calculation and hypothesis testing. Table 4.1 lists the main elements of the approach, and where they can be found within the chapter.

The methods discussed in this chapter are almost all built around penalized regression smoothers, based on splines. This type of smoother goes back at least as far as Wahba (1980) and Parker and Rice (1985). The suggestion of representing GAMs using spline like penalized regression smoothers was made in section 9.3.6 of Hastie and Tibshirani (1990) and was given renewed impetus by Marx and Eilers (1998), but it is not the only possibility, as will briefly be covered at the chapter's end.

The chapter starts by introducing several different penalized regression smoothers useful for practical work, including smooth functions of several covariates. Since these are all spline based, some discussion of why splines are useful smoothers is also presented. There follows a short explanation of how these can be assembled into an estimable GAM, and the P-IRLS estimation scheme is then justified, before moving on to the important topic of smoothing parameter estimation. Having covered model representation and estimation, a Bayesian model useful for deriving confidence intervals is then introduced, before considering practical performance of such intervals, and the calculation of approximate p-values for model terms. Some further topics of theoretical interest are then touched on before finishing with a very brief presentation of some key ideas underpinning two alternative frameworks for GAM estimation and inference. A review of the matrix algebra used in this chapter is provided in Appendix A.

What	How	Where
Turn GAM into penalized GLM with coefficients β and smoothing parameters λ	Choose bases and wiggliness measures for the smooth terms	4.1, 4.2
Select λ	By GCV, UBRE or AIC using efficient, robust Newton methods	4.5 4.6, 4.7
Estimate β	By P-IRLS	4.3
Find confidence intervals/ credible intervals for (functions of) β	Use Bayesian smoothing model	4.8, 4.9
Test hypotheses about GAMs	Use frequentist approximations, or GLM methods on unpenalized GAM	4.8.5, 4.10.1

Table 4.1 *The main components of the framework for generalized additive modelling covered in this chapter, and where they can be found.*

4.1 Smoothing bases

For simplicity of presentation, only one very simple type of penalized regression smoother was presented in Chapter 3. For practical work a variety of alternative smoothers are available, and this section introduces a useful subset of the possibilities, starting with smooths of one covariate, and then moving on to smooths of one or more covariates. Since all the smooths presented are based on splines (although the tensor product smooths need not be), the section starts by addressing the question: what's so special about splines?

4.1.1 Why splines?

Almost all the smooths considered in this book are based in some way on splines, so it is worth spending a little time on the theoretical properties that make these functions so appealing for penalized regression. Rather than attempt full generality, the flavour of the theoretical ideas can be gleaned by considering some properties of cubic splines, first in the context of interpolation, and then of smoothing.

Natural cubic splines are smoothest interpolators

Consider a set of points $\{x_i, y_i : i = 1, \dots, n\}$ where $x_i < x_{i+1}$. The *natural cubic spline*, $g(x)$, interpolating these points, is a function made up of sections of cubic polynomial, one for each $[x_i, x_{i+1}]$, which are joined together so that the whole

spline is continuous to second derivative, while $g(x_i) = y_i$ and $g''(x_1) = g''(x_n) = 0$. Figure 3.3 illustrates such a cubic spline.

Of all functions that are continuous on $[x_1, x_n]$, have absolutely continuous first derivatives and interpolate $\{x_i, y_i\}$, $g(x)$ is the one that is smoothest in the sense of minimizing:

$$J(f) = \int_{x_1}^{x_n} f''(x)^2 dx.$$

Green and Silverman (1994) provide a neat proof of this, based on the original work of Schoenberg (1964). Let $f(x)$ be an interpolant of $\{x_i, y_i\}$, other than $g(x)$, and let $h(x) = f(x) - g(x)$. We seek an expression for $J(f)$ in terms of $J(g)$.

$$\begin{aligned} \int_{x_1}^{x_n} f''(x)^2 dx &= \int_{x_1}^{x_n} \{g''(x) + h''(x)\}^2 dx \\ &= \int_{x_1}^{x_n} g''(x)^2 dx + 2 \int_{x_1}^{x_n} g''(x)h''(x) dx + \int_{x_1}^{x_n} h''(x)^2 dx \end{aligned}$$

and integrating the second term on the second line, by parts, yields

$$\begin{aligned} \int_{x_1}^{x_n} g''(x)h''(x) dx &= g''(x_n)h'(x_n) - g''(x_1)h'(x_1) - \int_{x_1}^{x_n} g'''(x)h'(x) dx \\ &= - \int_{x_1}^{x_n} g'''(x)h'(x) dx \\ &= - \sum_{i=1}^{n-1} g'''(x_i^+) \int_{x_i}^{x_{i+1}} h'(x) dx \\ &= - \sum_{i=1}^{n-1} g'''(x_i^+) \{h(x_{i+1}) - h(x_i)\} \\ &= 0, \end{aligned}$$

where equality of lines 1 and 2 follows from the fact that $g''(x_1) = g''(x_n) = 0$. Equality of lines 2 and 3 results from the fact that $g(x)$ is made up of sections of cubic polynomial, so that $g'''(x)$ is constant over any interval (x_i, x_{i+1}) . The final equality to zero follows from the fact that both $f(x)$ and $g(x)$ are interpolants, and are hence equal at x_i , implying that $h(x_i) = 0$.

So we have shown that

$$\int_{x_1}^{x_n} f''(x)^2 dx = \int_{x_1}^{x_n} g''(x)^2 dx + \int_{x_1}^{x_n} h''(x)^2 dx \geq \int_{x_1}^{x_n} g''(x)^2 dx$$

with equality only if $h''(x) = 0$ for $x_1 < x < x_n$. However, $h(x_1) = h(x_n) = 0$, so in fact we have equality if and only if $h(x) = 0$ on $[x_1, x_n]$. In other words any interpolant that is not identical to $g(x)$ will have a higher integrated squared second derivative. So there is a well defined sense in which the cubic spline is the smoothest possible interpolant through any set of data.

The smoothest interpolation property is not the only good property of cubic spline

interpolants. In de Boor (1978, Chapter 5) a number of results are presented showing that cubic spline interpolation is optimal, or at least very good, in various respects. For example, if a ‘complete’ cubic spline, \tilde{g} , is used to approximate a function, \tilde{f} , by interpolating a set of points $\{x_i, \tilde{f}(x_i) : i = 1, \dots, n\}$ and matching $\tilde{f}'(x_1)$ and $\tilde{f}'(x_n)$ then if $\tilde{f}(x)$ has 4 continuous derivatives:

$$\max|\tilde{f} - g| \leq \frac{5}{384} \max(x_{i+1} - x_i)^4 \max|\tilde{f}'''|,$$

and this can be shown to be the best achievable.

These properties of spline interpolants, suggest that splines ought to provide a good basis for representing smooth terms in statistical models. Whatever the true underlying smooth function is, a spline ought to be able to approximate it closely, and if we want to construct models from smooth functions of covariates, then representing those functions from smoothest approximations is intuitively appealing.

Cubic smoothing splines

In statistical work, y_i is usually measured with noise, and it is generally more useful to smooth x_i, y_i data, rather than interpolating them. To this end, rather than setting $g(x_i) = y_i$, it might be better to treat the $g(x_i)$ as n free parameters of the cubic spline, and to estimate them in order to minimize

$$\sum_{i=1}^n \{y_i - g(x_i)\}^2 + \lambda \int g''(x)^2 dx,$$

where λ is a tuneable parameter, used to control the relative weight to be given to the conflicting goals of matching the data and producing a smooth g . The resulting $g(x)$ is a *smoothing spline* (Reinsch, 1967). In fact, of *all functions*, f , that are continuous on $[x_1, x_n]$, and have absolutely continuous first derivatives, $g(x)$ is the function minimizing:

$$\sum_{i=1}^n \{y_i - f(x_i)\}^2 + \lambda \int f''(x)^2 dx. \quad (4.1)$$

The proof is easy. Suppose that some other function, $f^*(x)$, minimized (4.1). In that case we could interpolate $\{x_i, f^*(x_i)\}$ using a cubic spline, $g(x)$. Now $g(x)$ and $f^*(x)$ have the same sum of squares term in (4.1), but by the properties of interpolating splines, $g(x)$ must have the lower integrated squared second derivative. Hence $g(x)$ yields a lower (4.1) than $f^*(x)$, and a contradiction, unless $f^* = g$.

So, the cubic spline basis arises naturally from the specification of the smoothing objective (4.1), in which, what is meant by model fit is defined precisely, what is meant by smoothness is defined precisely, and the basis for representing smooth functions is not chosen in advance, but rather emerges from seeking the function minimizing (4.1).

Smoothing splines, then, seem to be somewhat ideal smoothers. The only substantial problem, is the fact that they have as many free parameters as there are data to be

smoothed. This seems wasteful, given that, in practice, λ will almost always be high enough that the resulting spline is much smoother than n degrees of freedom would suggest. Indeed, in section 4.10.4 we will see that many degrees of freedom of a spline are often suppressed completely by the penalty. For univariate smoothing with cubic splines, the large number of parameters turns out not to be problematic, but as soon as we try to deal with more covariates, the computational expense becomes severe.

An obvious compromise between retaining the good properties of splines, and computational efficiency, is to use penalized regression splines, as introduced in Chapter 3. At its simplest, this involves constructing a spline basis (and associated penalties) for a much smaller data-set than the one to be analyzed, and then using that basis (plus penalties) to model the original data set. The covariate values in the smaller data set should be arranged to nicely cover the distribution of covariate values in the original data set. This penalized regression spline idea is presented in Wahba (1980) and Parker and Rice (1985), for example. In the rest of this section, some spline based penalized regression smoothers will be presented, starting with univariate smoothers, and then moving on to smooths of several variables.

4.1.2 Cubic regression splines

The basis used in Chapter 3 was one way of defining a cubic regression spline basis, but there are other ways of defining such smoothers, which have some advantages in terms of interpretability of the parameters. One approach is to parameterize the spline in terms of its values at the knots.

Consider defining a cubic spline function, $f(x)$, with k knots, $x_1 \dots x_k$. Let $\beta_j = f(x_j)$ and $\delta_j = f''(x_j)$. Then the spline can be written as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\delta_j + c_j^+(x)\delta_{j+1} \text{ if } x_j \leq x \leq x_{j+1} \quad (4.2)$$

where the basis functions a_j^- , a_j^+ , c_j^- and c_j^+ are defined in table 4.2. The conditions that the spline must be continuous to second derivative, at the x_j , and should have zero second derivative at x_1 and x_k , can be shown to imply (exercise 1) that

$$\mathbf{B}\boldsymbol{\delta}^- = \mathbf{D}\boldsymbol{\beta}. \quad (4.3)$$

where $\boldsymbol{\delta}^- = (\delta_2, \dots, \delta_{k-1})^\top$ (since $\delta_1 = \delta_k = 0$) and \mathbf{B} and \mathbf{D} are defined in table 4.2.

Defining $\mathbf{F}^- = \mathbf{B}^{-1}\mathbf{D}$, and

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} \\ \mathbf{F}^- \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{0}$ is a row of zeros, we have that $\boldsymbol{\delta} = \mathbf{F}\boldsymbol{\beta}$. Hence, the spline can be re-written entirely in terms of $\boldsymbol{\beta}$ as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\mathbf{F}_j\boldsymbol{\beta} + c_j^+(x)\mathbf{F}_{j+1}\boldsymbol{\beta} \text{ if } x_j \leq x \leq x_{j+1},$$

Basis functions for a cubic spline

$$a_j^-(x) = (x_{j+1} - x)/h_j \quad c_j^-(x) = [(x_{j+1} - x)^3/h_j - h_j(x_{j+1} - x)]/6$$

$$a_j^+(x) = (x - x_j)/h_j \quad c_j^+(x) = [(x - x_j)^3/h_j - h_j(x - x_j)]/6$$

Non-zero matrix elements — non cyclic spline

$$\begin{array}{lll} D_{i,i} = 1/h_i & D_{i,i+1} = -1/h_i - 1/h_{i+1} & D_{i,i+2} = 1/h_{i+1} \\ B_{i,i} = (h_i + h_{i+1})/3 & & i = 1 \dots k-2 \end{array}$$

$$B_{i,i+1} = h_{i+1}/6 \quad B_{i+1,i} = h_{i+1}/6 \quad i = 1 \dots k-3$$

Non-zero matrix elements — cyclic spline

$$\begin{array}{lll} \tilde{B}_{i-1,i} = \tilde{B}_{i,i-1} = h_{i-1}/6 & \tilde{B}_{i,i} = (h_{i-1} + h_i)/3 & \\ \tilde{D}_{i-1,i} = \tilde{D}_{i,i-1} = 1/h_{i-1} & \tilde{D}_{i,i} = -1/h_{i-1} - 1/h_i & i = 2 \dots k-1 \end{array}$$

$$\begin{array}{lll} \tilde{B}_{1,1} = (h_{k-1} + h_1)/3 & \tilde{B}_{1,k-1} = h_{k-1}/6 & \tilde{B}_{k-1,1} = h_{k-1}/6 \\ \tilde{D}_{1,1} = -1/h_1 - 1/h_{k-1} & \tilde{D}_{1,k-1} = 1/h_{k-1} & \tilde{D}_{k-1,1} = 1/h_{k-1} \end{array}$$

Table 4.2 *Definitions of basis functions and matrices used to define a cubic regression spline.*
 $h_j = x_{j+1} - x_j$.

which can be re-written, once more, as

$$f(x) = \sum_{i=1}^k b_i(x)\beta_i$$

by implicit definition of new basis functions $b_i(x)$: figure 4.1 illustrates the basis. Hence, given a set of x values, at which to evaluate the spline, it is easy to obtain a model matrix mapping β to the evaluated spline. It can further be shown (e.g. Lancaster and Šalkauskas, 1986, or exercise 2) that

$$\int_{x_1}^{x_k} f''(x)^2 dx = \mathbf{\beta}^\top \mathbf{D}^\top \mathbf{B}^{-1} \mathbf{D} \beta$$

i.e. $\mathbf{S} \equiv \mathbf{D}^\top \mathbf{B}^{-1} \mathbf{D}$ is the penalty matrix for this basis.

Notice that in addition to having directly interpretable parameters, this basis does not require any re-scaling of the predictor variables before it can be used to construct a GAM, although, as with the chapter 3 basis, we do have to choose the locations of the knots x_j . See Lancaster and Šalkauskas (1986) for more details about this basis.

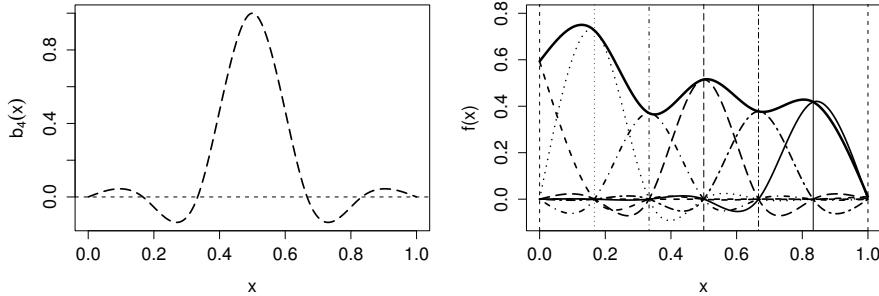


Figure 4.1 The left hand panel illustrates one basis function, $b_4(x)$, for a cubic regression spline of the type discussed in section 4.1.2: this basis function takes the value one at one knot of the spline, and zero at all other knots (such basis functions are sometimes called ‘cardinal basis functions’). The right hand panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions, $b_j(x)$, of a cubic regression spline, each multiplied by its associated coefficient β_j : these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve. The vertical thin lines show the knot locations.

4.1.3 A cyclic cubic regression spline

It is quite often appropriate for a model smooth function to be ‘cyclic’, meaning that the function has the same value and first few derivatives at its upper and lower boundaries. For example, in most applications, it would not be appropriate for a smooth function of time of year to change discontinuously at the year end. The penalized cubic regression spline, of the previous section, can be modified to produce such a smooth. The spline can still be written in the form (4.2), but we now have that $\beta_1 = \beta_k$ and $\delta_1 = \delta_k$. In this case then, we define vectors $\boldsymbol{\beta}^\top = (\beta_1, \dots, \beta_{k-1})$ and $\boldsymbol{\delta}^\top = (\delta_1, \dots, \delta_{k-1})$. The conditions that the spline must be continuous to second derivative at each knot, and that $f(x_1)$ must match $f(x_k)$, up to second derivative, are equivalent to

$$\tilde{\mathbf{B}}\boldsymbol{\delta} = \tilde{\mathbf{D}}\boldsymbol{\beta}$$

where $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{D}}$ are defined in table 4.2. Similar reasoning to that employed in the previous section implies that the spline can be written as

$$f(x) = \sum_{i=1}^{k-1} \tilde{b}_i(x)\beta_i,$$

by appropriate definition of the basis functions $\tilde{b}_i(x)$: figure 4.2 illustrates this basis. A second derivative penalty also follows:

$$\int_{x_1}^{x_k} f''(x)^2 dx = \boldsymbol{\beta}^\top \tilde{\mathbf{D}}^\top \tilde{\mathbf{B}}^{-1} \tilde{\mathbf{D}} \boldsymbol{\beta}.$$

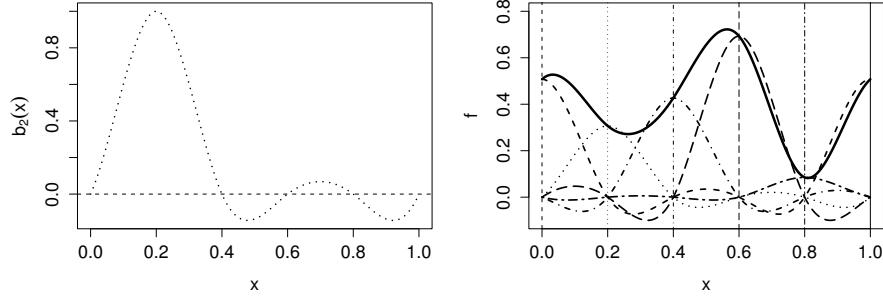


Figure 4.2 The left hand panel illustrates one basis function, $b_2(x)$, for a cyclic cubic regression spline of the type discussed in section 4.1.3: this basis function takes the value one at one knot of the spline, and zero at all other knots - notice how the basis function values and first two derivatives match at $x = 0$ and $x = 1$. The right hand panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions, $b_j(x)$, of a cubic regression spline, each multiplied by its associated coefficient β_j : these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve. The vertical thin lines show the knot locations.

4.1.4 P-splines

Yet another way to represent cubic splines (and indeed splines of higher or lower order), is by use of the B-spline basis. The B-spline basis is appealing because the basis functions are strictly local — each basis function is only non-zero over the intervals between $m + 3$ adjacent knots, where $m + 1$ is the order of the basis (e.g. $m = 2$ for a cubic spline*). To define a k parameter B-spline basis, we need to define $k + m + 1$ knots, $x_1 < x_2 < \dots < x_{k+m+1}$, where the interval over which the spline is to be evaluated lies within $[x_{m+2}, x_k]$ (so that the first and last $m + 1$ knot locations are essentially arbitrary). An $(m + 1)^{\text{th}}$ order spline can then be represented as

$$f(x) = \sum_{i=1}^k B_i^m(x)\beta_i,$$

where the B-spline basis functions are most conveniently defined recursively as follows:

$$B_i^m(x) = \frac{x - x_i}{x_{i+m+1} - x_i} B_i^{m-1}(x) + \frac{x_{i+m+2} - x}{x_{i+m+2} - x_{i+1}} B_{i+1}^{m-1}(x) \quad i = 1, \dots, k$$

and

$$B_i^{-1}(x) = \begin{cases} 1 & x_i \leq x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

(see e.g. de Boor, 1978; Lancaster and Šalkauskas, 1986). For example, the following R code can be used to evaluate single B-spline basis functions at a series of x values:

* The somewhat inconvenient definition of order is for compatibility with the notation usually used for normal splines.

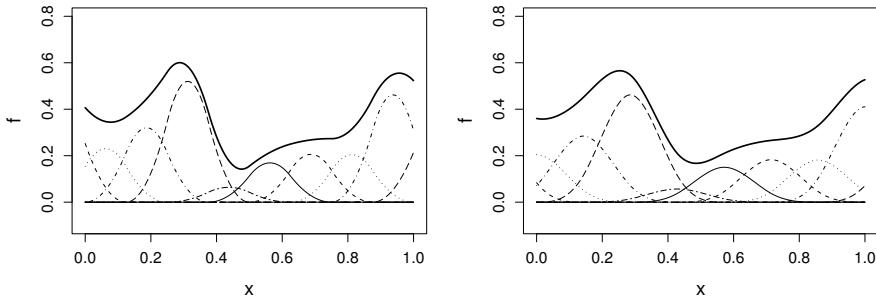


Figure 4.3 *Illustration of the representation of a smooth curve by rank 10 B-spline bases. The left plot shows a B-spline basis with $m = 1$. The thin curves show B-spline basis functions multiplied by their associated coefficients, each is non-zero over only 3 intervals. The sum of the coefficients multiplied by the basis functions gives the spline itself, represented by the thicker continuous curve. The right panel is the same, but for a basis for which $m = 2$: in this case each basis function is non-zero over 4 adjacent intervals. In both panels the knot locations are where each basis function peaks.*

```

bspline <- function(x,k,i,m=2)
# evaluate ith b-spline basis function of order m at the values
# in x, given knot locations in k
{
  if (m==1) # base of recursion
    { res <- as.numeric(x<k[i+1]&x>=k[i])
    } else      # construct from call to lower order basis
    { z0 <- (x-k[i])/(k[i+m+1]-k[i])
      z1 <- (k[i+m+2]-x)/(k[i+m+2]-k[i+1])
      res <- z0*bspline(x,k,i,m-1)+ z1*bspline(x,k,i+1,m-1)
    }
  res
}

```

Figure 4.3 illustrates the representation of functions using B-spline bases of two different orders.

B-splines were developed as a very stable basis for large scale spline interpolation (see de Boor, 1978, for further details), but for most statistical work with low rank penalized regression splines, you would have to be using very poor numerical methods before the enhanced stability of the basis became noticeable. The real statistical interest in B-splines has resulted from the work of Eilers and Marx (1996) in using them to develop what they term *P-splines*.

P-splines are low rank smoothers using a B-spline basis, usually defined on evenly spaced knots, and a *difference penalty* applied directly to the parameters, β_i , to control function wigginess. How this works is best seen by example. If we decide to penalize the squared difference between adjacent β_i values then the penalty would

be

$$\mathcal{P} = \sum_{i=1}^{k-1} (\beta_{i+1} - \beta_i)^2 = \beta_1^2 - 2\beta_1\beta_2 + 2\beta_2^2 - 2\beta_2\beta_3 + \dots + \beta_k^2,$$

and it is straightforward to see that this can be written

$$\mathcal{P} = \boldsymbol{\beta}^T \begin{bmatrix} 1 & -1 & 0 & \dots \\ -1 & 2 & -1 & \dots \\ 0 & -1 & 2 & \dots \\ \vdots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \boldsymbol{\beta}.$$

Such penalties are very easily generated in R. For example the penalty matrix for \mathcal{P} can be generated by:

```
k<-6                                # example basis dimension
P <- diag(k), differences=1           # sqrt of penalty matrix
S <- t(P) %*% P                      # penalty matrix
```

Higher order penalties are produced by increasing the differences parameter. The only lower order penalty is the identity matrix.

P-splines are extremely easy to set up and use, and allow a good deal of flexibility, in that any order of penalty can be combined with any order of B-spline basis, as the user sees fit. Their disadvantage is that the simplicity is somewhat diminished if uneven knot spacing is required, and that the penalties are less easy to interpret in terms of the properties of the fitted smooth, than the more usual spline penalties. See exercises 7 to 9, for further coverage of P-splines.

4.1.5 Thin plate regression splines

The bases covered so far are each useful in practice, but are open to some criticisms.

1. It is necessary to choose knot locations, in order to use each basis: this introduces an extra degree of subjectivity into the model fits.
2. The bases are only useful for representing smooths of one predictor variable.
3. It is not clear to what extent the bases are better or worse than any other basis that might be used.

In this section, an approach is developed which goes some way to addressing these issues, by producing knot free bases, for smooths of any number of predictors, that are in a certain limited sense ‘optimal’: the thin plate regression splines.

Thin plate splines

Thin plate splines (Duchon, 1977) are a very elegant and general solution to the problem of estimating a smooth function of multiple predictor variables, from noisy

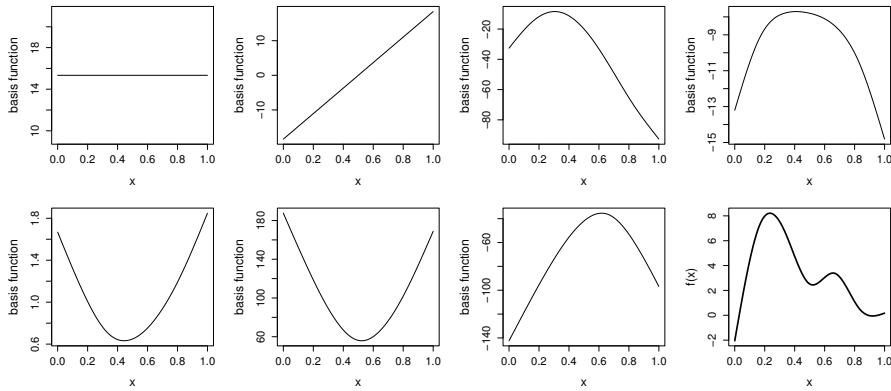


Figure 4.4 *Illustration of a thin plate spline basis for representing a smooth function of one variable fitted to 7 data with penalty order $m = 2$. The first 7 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth curve in the lower right panel. The first two basis functions span the space of functions that are completely smooth, according to the wiggliness measure. The remaining basis functions represent the wiggly component of the smooth curve: these latter functions are shown after absorption of the thin plate spline constraints $\mathbf{T}^\top \boldsymbol{\delta} = \mathbf{0}$ into the basis.*

observations of the function, at particular values of those predictors. Consider then, the problem of estimating the smooth function $g(\mathbf{x})$, from n observations (y_i, \mathbf{x}_i) such that

$$y_i = g(\mathbf{x}_i) + \epsilon_i$$

where ϵ_i is a random error term and where \mathbf{x} is a d -vector ($d \leq n$). Thin-plate spline smoothing estimates g by finding the function \hat{f} minimizing:

$$\|\mathbf{y} - \mathbf{f}\|^2 + \lambda J_{md}(f) \quad (4.4)$$

where \mathbf{y} is the vector of y_i data and $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_n))^\top$. $J_{md}(f)$ is a penalty functional measuring the ‘wiggliness’ of f , and λ is a smoothing parameter, controlling the tradeoff between data fitting and smoothness of f . The wiggliness penalty is defined as

$$J_{md} = \int \dots \int_{\mathbb{R}^d} \sum_{\nu_1+\dots+\nu_d=m} \frac{m!}{\nu_1! \dots \nu_d!} \left(\frac{\partial^m f}{\partial x_1^{\nu_1} \dots \partial x_d^{\nu_d}} \right)^2 dx_1 \dots dx_d.^\dagger \quad (4.5)$$

Further progress is only possible if m is chosen so that $2m > d$, and in fact for ‘visually smooth’ results it is preferable that $2m > d + 1$. Subject to the first of these

[†] The general form of the penalty is somewhat intimidating, so an example is useful. In the case of a smooth of two predictors with wiggliness measured using second derivatives, we have

$$J_{22} = \int \int \left(\frac{\partial^2 f}{\partial x_1^2} \right)^2 + \left(\frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f}{\partial x_2^2} \right)^2 dx_1 dx_2.$$

restrictions, it can be shown that the function minimizing (4.4) has the form,

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \delta_i \eta_{md}(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^M \alpha_j \phi_j(\mathbf{x}), \quad (4.6)$$

where $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$ are vectors of coefficients to be estimated, $\boldsymbol{\delta}$ being subject to the linear constraints that $\mathbf{T}^\top \boldsymbol{\delta} = \mathbf{0}$ where $T_{ij} = \phi_j(\mathbf{x}_i)$. The $M = \binom{m+d-1}{d}$ functions, ϕ_i , are linearly independent polynomials spanning the space of polynomials in \Re^d of degree less than m . The ϕ_i span the space of functions for which J_{md} is zero, i.e. the ‘null space’ of J_{md} : those functions that are considered ‘completely smooth’. For example, for $m = d = 2$ these functions are $\phi_1(\mathbf{x}) = 1$, $\phi_2(\mathbf{x}) = x_1$ and $\phi_3(\mathbf{x}) = x_2$. The remaining basis functions used in (4.6) are defined as

$$\eta_{md}(r) = \begin{cases} \frac{(-1)^{m+1+d/2}}{2^{2m-1}\pi^{d/2}(m-1)!(m-d/2)!} r^{2m-d} \log(r) & d \text{ even} \\ \frac{\Gamma(d/2-m)}{2^{2m}\pi^{d/2}(m-1)!} r^{2m-d} & d \text{ odd.} \end{cases}$$

Now defining matrix \mathbf{E} by $E_{ij} \equiv \eta_{md}(\|\mathbf{x}_i - \mathbf{x}_j\|)$, the thin plate spline fitting problem becomes,

$$\text{minimize } \|\mathbf{y} - \mathbf{E}\boldsymbol{\delta} - \mathbf{T}\boldsymbol{\alpha}\|^2 + \lambda \boldsymbol{\delta}^\top \mathbf{E}\boldsymbol{\delta} \text{ subject to } \mathbf{T}^\top \boldsymbol{\delta} = \mathbf{0}, \quad (4.7)$$

with respect to $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$. Wahba (1990) or Green and Silverman (1994) provide further information about thin-plate splines, and figure 4.4 illustrates a thin plate spline basis in one dimension.

The thin plate spline, \hat{f} , is something of an ideal smoother: it has been constructed by defining exactly what is meant by smoothness, exactly how much weight to give to the conflicting goals of matching the data and making \hat{f} smooth, and finding the *function* that best satisfies the resulting smoothing objective. Notice that in doing this we did not have to choose knot positions or select basis functions, both of these emerged naturally from the mathematical statement of the smoothing problem. In addition, thin plate splines can deal with any number of predictor variables, and allow the user some flexibility to select the order of derivative used in the measure of function wigginess. So, at first sight it might seem that the problems listed at the start of this section are all solved, and thin plate spline bases and penalties should be used to represent all the smooth terms in the model.

The problem with thin plate splines is computational cost: these smoothers have as many unknown parameters as there are data (strictly, number of unique predictor combinations), and, except in the single predictor case, the computational cost of model estimation is proportional to the cube of the number of parameters. This is a very high price to pay for using such smooths. Given that the effective degrees of freedom estimated for a model term is usually a small proportion of n , it seems wasteful to use so many parameters to represent the term, and this begs the question of whether a low rank approximation could be produced which is as close as possible to the thin plate spline smooth, without incurring prohibitive computational cost.

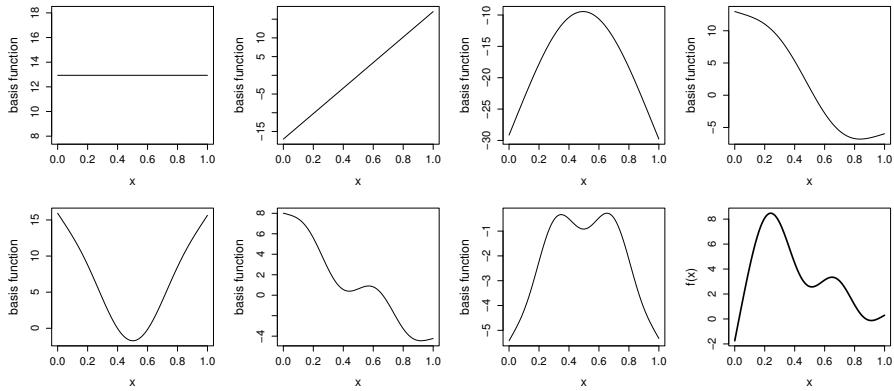


Figure 4.5 *Illustration of a rank 7 thin plate regression spline basis for representing a smooth function of one variable, with penalty order $m = 2$. The first 7 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth curve in the lower right panel. The first two basis functions span the space of functions that are completely smooth, according to the wiggliness measure. The remaining basis functions represent the wiggly component of the smooth curve: notice how these functions become successively more wiggly while generally tending to contribute less and less to the overall fit.*

Thin plate regression splines

Thin plate regression splines are based the idea of truncating the space of the wiggly components of the thin plate spline (the components with parameters δ), while leaving the components of ‘zero wiggliness’ unchanged (the α components). Let $\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ be the eigen-decomposition of \mathbf{E} , so that \mathbf{D} is a diagonal matrix of eigenvalues of \mathbf{E} arranged so that $|D_{i,i}| \geq |D_{i-1,i-1}|$ and the columns of \mathbf{U} are the corresponding eigenvectors. Now let \mathbf{U}_k denote the matrix consisting of the first k columns of \mathbf{U} and \mathbf{D}_k denote the top right $k \times k$ submatrix of \mathbf{D} . Restricting δ to the columns space of \mathbf{U}_k , by writing $\delta = \mathbf{U}_k\delta_k$, means that (4.7) becomes

$$\text{minimise } \|\mathbf{y} - \mathbf{U}_k\mathbf{D}_k\delta_k - \mathbf{T}\alpha\|^2 + \lambda\delta_k^\top\mathbf{D}_k\delta_k \text{ subject to } \mathbf{T}^\top\mathbf{U}_k\delta_k = \mathbf{0}$$

w.r.t. δ_k and α . The constraints can be absorbed in the usual manner, described in section 1.8.1. We first find any orthogonal column basis, \mathbf{Z}_k , such that $\mathbf{T}^\top\mathbf{U}_k\mathbf{Z}_k = \mathbf{0}$. One way to do this is to form the QR decomposition of $\mathbf{U}_k^\top\mathbf{T}$: the final M columns of the orthogonal factor give a \mathbf{Z}_k (see sections 1.8.1 and A.6). Restricting δ_k to this space, by writing $\delta_k = \mathbf{Z}_k\tilde{\delta}$, yields the unconstrained problem that must be solved to fit the rank k approximation to the smoothing spline:

$$\text{minimise } \|\mathbf{y} - \mathbf{U}_k\mathbf{D}_k\mathbf{Z}_k\tilde{\delta} - \mathbf{T}\alpha\|^2 + \lambda\tilde{\delta}^\top\mathbf{Z}_k^\top\mathbf{D}_k\mathbf{Z}_k\tilde{\delta}$$

with respect to $\tilde{\delta}$ and α . This has a computational cost of $O(k^3)$. Having fitted the model, evaluation of the spline at any point is easy: simply evaluate $\delta = \mathbf{U}_k\mathbf{Z}_k\tilde{\delta}$ and use (4.6).

Now, the main problem is how to find \mathbf{U}_k and \mathbf{D}_k sufficiently cheaply. A full eigen-decomposition of \mathbf{E} requires $O(n^3)$ operations, which would somewhat limit the utility of the TPRS approach. Fortunately the method of Lanczos iteration can be employed to find \mathbf{U}_k and \mathbf{D}_k at the substantially lower cost of $O(n^2k)$ operations. See Appendix A, section A.11, for a suitable Lanczos algorithm.

Properties of thin plate regression splines

It is clear that thin plate regression splines avoid the problem of knot placement, are relatively cheap to compute, and can be constructed for smooths of any number of predictor variables, but what of their optimality properties? The thin-plate splines are optimal in the sense that no smooth function will better minimize (4.4), but to what extent is that optimality inherited by the TPRS approximation? To answer this it helps to think about what would make a good approximation. An ideal approximation would probably result in the minimum possible perturbation of the fitted values of the spline, at the same time as making the minimum possible change to the ‘shape’ of the fitted spline. It is difficult to see how both these aims could be achieved, for all possible response data, without first fitting the full thin plate spline. But if the criteria are loosened somewhat to minimizing the worst possible changes in shape and fitted value then progress can be made, as follows.

The basis change and truncation can be thought of as replacing \mathbf{E} , in the norm in (4.7), by the matrix $\hat{\mathbf{E}} = \mathbf{E}\mathbf{U}_k\mathbf{U}_k^\top$, while replacing \mathbf{E} , in the penalty term of (4.7), by $\tilde{\mathbf{E}} = \mathbf{U}_k^\top\mathbf{U}_k\mathbf{E}\mathbf{U}_k\mathbf{U}_k^\top$. Now since the fitted values of the spline are given by $\mathbf{E}\hat{\boldsymbol{\delta}} + \mathbf{T}\boldsymbol{\alpha}$, the worst possible change in fitted values could be measured by:

$$\hat{e}_k = \max_{\boldsymbol{\delta} \neq \mathbf{0}} \frac{\|(\mathbf{E} - \hat{\mathbf{E}}_k)\boldsymbol{\delta}\|}{\|\boldsymbol{\delta}\|}.$$

(dividing by $\|\boldsymbol{\delta}\|$ is necessary since the upper norm otherwise has a maximum at infinity.) The ‘shape’ of the spline is measured by the penalty term in (4.7), so a suitable measure of the worst possible change in the shape of the spline caused by the truncation might be:

$$\tilde{e}_k = \max_{\boldsymbol{\delta} \neq \mathbf{0}} \frac{\boldsymbol{\delta}^\top(\mathbf{E} - \tilde{\mathbf{E}}_k)\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|^2}.$$

It turns out to be quite easy to show that \hat{e}_k and \tilde{e}_k are simultaneously minimized by the choice of \mathbf{U}_k , as the truncated basis for $\boldsymbol{\delta}$. i.e. there is no matrix of the same dimension as \mathbf{U}_k which would lead to lower \hat{e}_k or \tilde{e}_k , if used in place of \mathbf{U}_k (see Wood, 2003).

Note that \hat{e}_k and \tilde{e}_k are really formulated in too large a space. Ideally we would impose the constraints $\mathbf{T}^\top\boldsymbol{\delta} = \mathbf{0}$ on both, but in that case different bases minimize the two criteria. This in turn leads to the question of whether it would not be better to concentrate on just one of the criteria, but this is unsatisfactory, as it leads to results that depend on how the original thin plate spline problem is parameterized.

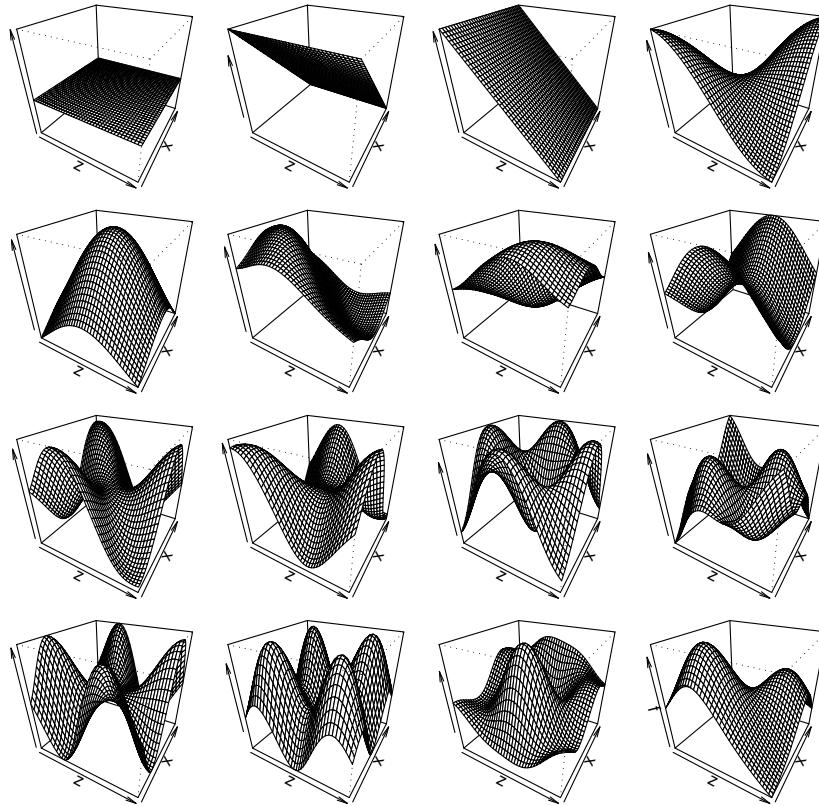


Figure 4.6 *Illustration of a rank 15 thin plate regression spline basis for representing a smooth function of two variables, with penalty order $m = 2$. The first 15 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth surface in the lower right panel. The first three basis functions span the space of functions that are completely smooth, according to the wiggleness measure, J_{22} . The remaining basis functions represent the wiggly component of the smooth curve: notice how these functions become successively more wiggly.*

Furthermore, these results can be extremely poor for some parameterizations. For example, if the thin plate spline is parameterized in terms of the fitted values, then the \hat{e}_k optimal approximation is not smooth. Similarly, very poor fitted values result from an \hat{e}_k optimal approximation to a thin plate spline, if that thin plate spline is parameterized so that the penalty matrix is an identity matrix, with some leading diagonal entries zeroed.

To sum up: thin plate regression splines are probably the best that can be hoped for in terms of approximating the behaviour of a thin plate spline using a basis of any given low rank. They have the nice property of avoiding having to choose ‘knot locations’, and are reasonably computationally efficient, if Lanczos iteration is used to find the truncated eigen-decomposition of \mathbf{E} . They also retain the rotational invari-

ance (isotropy) of full thin plate spline. Figures 4.5 and 4.6 provide examples of the bases functions that result from adopting a t.p.r.s approach.

Knot based approximation

If one is prepared to forgo optimality, and choose knot locations, then a simpler approximation is available, which avoids the truncated eigen-decomposition. If knot locations $\{\mathbf{x}_i^* : i = 1 \dots k\}$ are chosen, then the spline can be approximated by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^k \delta_i \eta_{md}(\|\mathbf{x} - \mathbf{x}_i^*\|) + \sum_{j=1}^M \alpha_j \phi_j(\mathbf{x}) \quad (4.8)$$

where $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$ are estimated by minimizing

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta} \text{ subject to } \mathbf{C}\boldsymbol{\beta} = \mathbf{0}$$

w.r.t. $\boldsymbol{\beta}^\top = (\boldsymbol{\delta}^\top, \boldsymbol{\alpha}^\top)$. \mathbf{X} is an $n \times k + M$ matrix such that

$$X_{ij} = \begin{cases} \eta_{md}(\|\mathbf{x}_i - \mathbf{x}_j^*\|) & j = 1, \dots, k \\ \phi_{j-k}(x_i) & j = k + 1, \dots, k + M. \end{cases}$$

\mathbf{S} is a $(k + M) \times (k + M)$ matrix with zeroes everywhere except in its upper left $k \times k$ block where $S_{ij} = \eta_{md}(\|\mathbf{x}_i^* - \mathbf{x}_j^*\|)$. Finally, \mathbf{C} is an $M \times (k + M)$ matrix such that

$$C_{ij} = \begin{cases} \phi_i(\mathbf{x}_j^*) & j = 1, \dots, k \\ 0 & j = k + 1, \dots, k + M. \end{cases}$$

This approximation goes back at least to Wahba (1980). Some care is required to choose the knot locations carefully. In one dimension it is usual to choose quantiles of the empirical distribution of the predictor, or even spacing, but in more dimensions matters are often more difficult. One possibility is to take a random sample of the observed predictor variable combinations, another to take a ‘spatially stratified’ sample of the predictor variable combinations. Even spacing is sometimes appropriate, or more sophisticated space filling schemes can be used: Ruppert et al. (2003) provide a useful discussion of the alternatives.

4.1.6 Shrinkage smoothers

A disadvantage of the smooths discussed so far, is that no matter how large their associated smoothing parameter becomes, the smooth is never completely eliminated in the sense of having all its parameters estimated to be zero. On the contrary, some functions are treated as completely smooth by the penalty, and hence functions of this class are always completely un-penalized. From the point of view of model selection with GAMs it would be more convenient if smooths could be zeroed by adjustment of smoothing parameters. One way to do this would be to add an extra penalty, with associated smoothing parameter which acted only on the unpenalized functions, but this would open up the possibility of penalizing the smooth components of a function

more than the wiggly components, which seems unsatisfactory, as well as requiring an extra smoothing parameter per smooth. A fairly crude alternative, is simply to add a small multiple of the identity matrix to the penalty matrix of the smooth, i.e.

$$\mathbf{S} \rightarrow \mathbf{S} + \epsilon \mathbf{I}$$

so that the penalty will now shrink all parameters to zero if its associated smoothing parameter is large enough. If ϵ is small enough, the identity part of the penalty will have almost no impact when a function is ‘wiggly’: only once it becomes close to ‘completely smooth’ will the identity component start to become important, and really start shrinking the parameters towards zero.

4.1.7 Choosing the basis dimension

When using penalized regression splines the modeller chooses the basis dimension as part of the model building process. Typically, this substantially reduces the computational burden of modelling, relative to full spline methods, and recognizes the fact that, usually, something is seriously wrong if a statistical model really *requires* as many coefficients as there are data. Working with fixed basis dimensions also makes it rather trivial to demonstrate large sample consistency, and other properties, of the smoothing methods, but only at the cost of a slightly artificial assumption that the truth is really in the space spanned by the reduced basis.

The main challenge introduced, by this low rank approach, is that a basis dimension has to be chosen. In the context of spline smoothing, Kim and Gu (2004) showed that the basis size should scale as $n^{2/9}$, where n is the number of data. Based on simulation they suggested using $10n^{2/9}$ as the basis dimension, but it is hard to see how one can really know what the constant of proportionality should be, without knowing the truth that is being estimated. Chapter 5 includes several examples where the rule appears to give too small a basis dimension, for example in section 5.6.2. Wood (2006) also suggests that the basis dimension should depend on the number of covariates of a smooth, as well as the sample size.

In practice, then, choice of basis dimension is something that probably has to remain a part of model specification. However, it is important to note that the exact size of basis dimension is really not that critical. The basis dimension is only setting an upper bound on the flexibility of a term: it is the smoothing parameter that controls the actual effective degrees of freedom. Hence the model fit is usually rather insensitive to the basis dimension, provided that it is not set restrictively low for the application concerned. The only caveat to this point is the slightly subtle one, that a function space with basis dimension 20 will contain a larger space of functions with EDF 5 than will a function space of dimension 10 (the numbers being arbitrary): it is this fact that causes model fit to retain some sensitivity to basis dimension, even if the appropriate EDF for a term is well below the basis dimension.

In practice, the modeller needs to decide roughly how large a basis dimension is fairly certain to provide adequate flexibility, in any particular application, and use that.

4.1.8 Tensor product smooths

A major feature of the thin plate (regression) spline approach of section 4.1.5 is the isotropy of the wigginess penalty: wigginess in all directions is treated equally, with the fitted spline entirely invariant to rotation of the co-ordinate system for the predictor variables. For example, suppose we were to measure air pollution at a fixed set of points in Southern England, measuring the location of the points relative to the UK national grid and modelling pollution levels as a smooth function of the two spatial co-ordinates. Now suppose that the locations were instead measured on the French grid, and the modelling exercise repeated: the model fit would be identical (provided that the earth is flat).

This isotropy is often considered to be desirable when modelling things as a smooth function of geographic co-ordinates[†], but it has some disadvantages. Chief among them is the difficulty of knowing how to scale predictors relative to one another, when both are arguments of the same smooth, but they are measured in fundamentally different units. For example, consider a smooth function of a single spatial co-ordinate and time: the implied relative importance of smoothness in time versus smoothness in space, is very different between a situation in which the units are metres and hours, compared to that in which the units are light-years and nanoseconds. One pragmatic approach is to scale all predictors into the unit square, as is often done in loess smoothing, but this is essentially arbitrary. A more satisfactory approach uses *tensor product smooths*.

Tensor product bases

The basic approach of this section is to start from smooths of single covariates, represented using any basis with associated quadratic penalty measuring ‘wigginess’ of the smooth. From these ‘marginal smooths’ a ‘tensor product’ construction is used to build up smooths of several variables. See de Boor (1978) for an important early reference on tensor product spline bases.

The methods developed here can be used to construct smooth functions of *any* number of covariates, but the simplest introduction is via the construction of a smooth function of 3 covariates, x , z and v , the generalization then being trivial. The process starts by assuming that we have low rank bases available, for representing smooth functions f_x , f_z and f_v of each of the covariates. That is we can write:

$$f_x(x) = \sum_{i=1}^I \alpha_i a_i(x), \quad f_z(z) = \sum_{l=1}^L \delta_l d_l(z) \quad \text{and} \quad f_v(v) = \sum_{k=1}^K \beta_k b_k(v),$$

where the α_i , δ_l and β_k are parameters, and the $a_i(x)$, $d_l(z)$ and $b_k(v)$ are known basis functions.

[†] Although it’s possible to overstate the case for doing this: in many applications at many locations North-South is not the same as East-West.

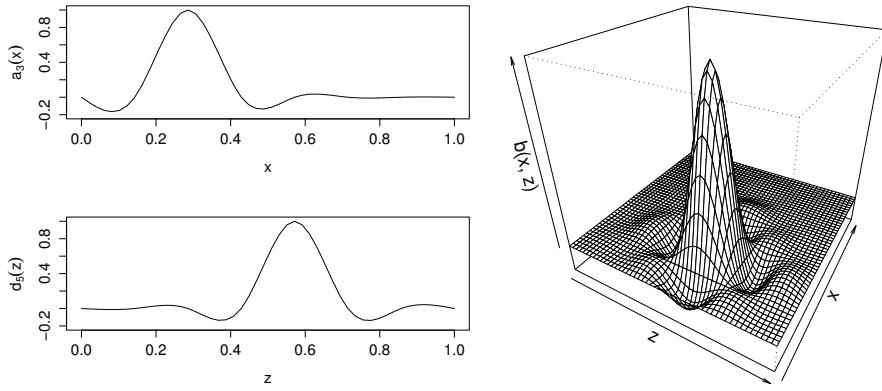


Figure 4.7 *How the product of two marginal basis functions for smooth functions of x and z , separately, results in a basis function for a smooth function of x and z together.* The two left panels show the 3rd and 5th basis functions for rank 8 cubic regression spline smooths of x and z respectively. The right hand plot shows $a_3(x)b_5(z)$, one of 64 similar basis functions of the tensor product smooth derived from these two marginal smooths.

Now consider how the smooth function of x , f_x , could be converted into a smooth function of x and z . What is required is for f_x to vary smoothly with z , and this can be achieved by allowing its parameters, α_i , to vary smoothly with z . Using the basis already available for representing smooth functions of z we could write:

$$\alpha_i(z) = \sum_{l=1}^L \delta_{il} d_l(z)$$

which immediately gives

$$f_{xz}(x, z) = \sum_{i=1}^I \sum_{l=1}^L \delta_{il} d_l(z) a_i(x).$$

Figure 4.7 illustrates this construction. Continuing in the same way, we could now create a smooth function of x , z and v by allowing f_{xz} to vary smoothly with v . Again, the obvious way to do this is to let the parameters of f_{xz} vary smoothly with v , and following the same reasoning as before we get

$$f_{xzx}(x, z, v) = \sum_{i=1}^I \sum_{l=1}^L \sum_{k=1}^K \beta_{ilk} b_k(v) d_l(z) a_i(x).$$

For any particular set of observations of x , z and v , there is a simple relationship between the model matrix, \mathbf{X} , evaluating the tensor product smooth at these observations, and the model matrices \mathbf{X}_x , \mathbf{X}_z and \mathbf{X}_v that would evaluate the marginal smooths at the same observations. If \otimes is the usual Kronecker product (see section A.4), then it is easy to show that, given appropriate ordering of the β_{ilk} into a vector

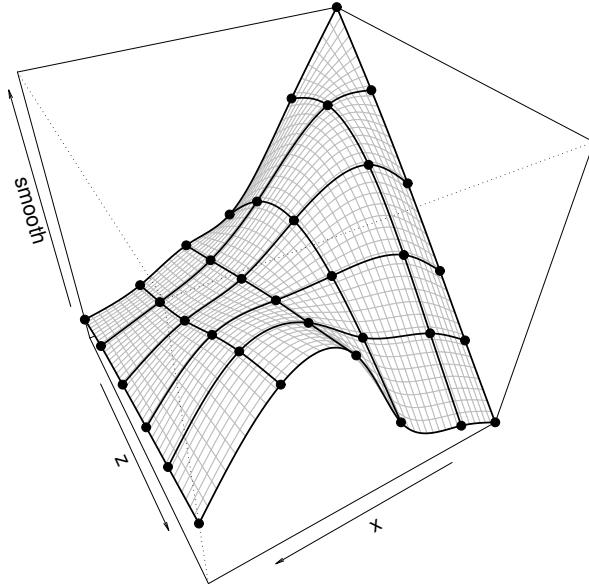


Figure 4.8 Illustration of a tensor product smooth of two variables x and z , constructed from 2 rank 6 marginal bases. Following section 4.1.8 a tensor product smooth can always be parameterized in terms of the values of the function at a set of ‘knots’ spread over the function domain on a regular mesh: i.e. in terms of the heights of the \bullet ’s shown. The basis construction can be thought of as follows: start with a smooth of x parameterized in terms of function values at a set of ‘knots’; to make the smooth of x vary smoothly with z , simply allow each of its parameters to vary smoothly with z : this can be done by representing each parameter using a smooth of z , also parameterized in terms of function values at a set of ‘knots’. Exactly the same smooth arises if we reverse the roles of x and z in this construction. The tensor product smooth penalty in the x direction, advocated in section 4.1.8, is simply the sum of the the marginal wigginess measure for the smooth of x applied to the thick black curves parallel to the x axes: the z penalty is similarly defined in terms of the marginal penalty of the smooth of z applied to the thick black curves parallel to the z axis.

β , the i^{th} row of \mathbf{X} is simply:

$$\mathbf{X}_i = \mathbf{X}_{xi} \otimes \mathbf{X}_{zi} \otimes \mathbf{X}_{vi}.$$

Clearly (i) this construction can be continued for as many covariates as are required; (ii) the result is independent of the order in which we treat the covariates and (iii) the covariates can themselves be vector covariates. Figure 4.8 attempts to illustrate the tensor product construction for a smooth of two covariates.

Tensor product penalties

Having derived a ‘tensor product’ basis for representing smooth functions, it is also necessary to have some way of measuring function ‘wiggleness’, if the basis is to be useful for representing smooth functions in a GAM context. Again, it is possible to start from wiggleness measures associated with the marginal smooth functions, and again the three covariate case provides sufficient illustration. Suppose then, that each marginal smooth has an associated functional that measures function wiggleness, and can be expressed as a quadratic form in the marginal parameters. That is

$$J_x(f_x) = \boldsymbol{\alpha}^\top \mathbf{S}_x \boldsymbol{\alpha}, \quad J_z(f_z) = \boldsymbol{\delta}^\top \mathbf{S}_z \boldsymbol{\delta} \text{ and } J_v(f_v) = \mathcal{B}^\top \mathbf{S}_v \mathcal{B}.$$

The \mathbf{S}_\bullet matrices contain known coefficients, and $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and \mathcal{B} are vectors of coefficients of the marginal smooths. An example of a penalty functional is the cubic spline penalty, $J_x(f_x) = \int (\partial^2 f_x / \partial x^2)^2 dx$. Now let $f_{x|zv}(x)$ be $f_{xvz}(x, z, v)$ considered as a function of x only, with z and v held constant, and define $f_{z|xv}(z)$ and $f_{v|xz}(v)$ similarly. A natural way of measuring wiggleness of $f_{x|zv}$ is to use:

$$J(f_{x|zv}) = \lambda_x \int_{z,v} J_x(f_{x|zv}) dz dv + \lambda_z \int_{x,v} J_z(f_{z|xv}) dx dv + \lambda_v \int_{x,z} J_v(f_{v|xz}) dx dz$$

where the λ_\bullet are smoothing parameters controlling the tradeoff between wiggleness in different directions, and allowing the penalty to be invariant to the relative scaling of the covariates. As an example, if cubic spline penalties were used as the marginal penalties, then

$$J(f) = \int_{x,z,v} \lambda_x \left(\frac{\partial^2 f}{\partial x^2} \right)^2 + \lambda_z \left(\frac{\partial^2 f}{\partial z^2} \right)^2 + \lambda_v \left(\frac{\partial^2 f}{\partial v^2} \right)^2 dx dz dv.$$

Hence, if the marginal penalties are easily interpretable, in terms of function shape, then so is the induced penalty. Numerical evaluation of the integrals in J is straightforward. As an example consider the penalty in the x direction. The function $f_{x|zv}(x)$ can be written as

$$f_{x|zv}(x) = \sum_{i=1}^I \alpha_i(z, v) a_i(x),$$

and it is always possible to find the matrix of coefficients $\mathbf{M}_{z,v}$ such that $\boldsymbol{\alpha}(z, v) = \mathbf{M}_{z,v} \boldsymbol{\beta}$ where $\boldsymbol{\beta}$ is the vector of β_{ilk} arranged in some appropriate order. Hence

$$J_x(f_{x|zv}) = \boldsymbol{\alpha}(z, v)^\top \mathbf{S}_x \boldsymbol{\alpha}(z, v) = \boldsymbol{\beta}^\top \mathbf{M}_{zv}^\top \mathbf{S}_x \mathbf{M}_{zv} \boldsymbol{\beta}$$

and so

$$\int_{z,v} J_x(f_{x|zv}) dz dv = \boldsymbol{\beta}^\top \int_{z,v} \mathbf{M}_{zv}^\top \mathbf{S}_x \mathbf{M}_{zv} dz dv \boldsymbol{\beta}.$$

The last integral can be performed numerically, and it is clear that the same approach can be applied to all components of the penalty. However, a simple reparameterization can be used to provide an approximation to the terms in the penalty, which performs well in practice, and avoids the need for explicit numerical integration.

To see how the approach works, consider the marginal smooth f_x . Let $\{x_i^* : i =$

$1, \dots, I\}$ be a set of values of x spread evenly through the range of the observed x values. In this case we can always re-parameterize f_x in terms of new parameters

$$\alpha'_i = f_x(x_i^*).$$

Clearly under this re-parameterization $\boldsymbol{\alpha}' = \boldsymbol{\Gamma}\boldsymbol{\alpha}$ where $\Gamma_{ij} = a_i(x_j^*)$. Hence the marginal model matrix becomes $\mathbf{X}'_x = \mathbf{X}_x\boldsymbol{\Gamma}^{-1}$ and the penalty coefficient matrix becomes $\mathbf{S}'_x = \boldsymbol{\Gamma}^{-T}\mathbf{S}_x\boldsymbol{\Gamma}^{-1}$.

Now suppose that the same sort of re-parameterization is applied to the marginal smooths f_v and f_z . In this case we have that

$$\int_{z,v} J_x(f_{x|zv}) dz dv \approx h \sum_{lk} J_x(f_{x|z_l^* v_k^*}),$$

where h is some constant of proportionality related to the spacing of the z_l^* 's and v_k^* 's. Similar expressions hold for the other integrals making up J . It is straightforward to show that the summation in the above approximation is:

$$J_x^*(f_{x|zv}) = \boldsymbol{\beta}^T \tilde{\mathbf{S}}_x \boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_x = \mathbf{S}'_x \otimes \mathbf{I}_L \otimes \mathbf{I}_K$$

and \mathbf{I}_L is the rank L identity matrix. Exactly similar definitions hold for the other components of the penalty so that

$$J_z^*(f_{x|zv}) = \boldsymbol{\beta}^T \tilde{\mathbf{S}}_z \boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_z = \mathbf{I}_I \otimes \mathbf{S}'_z \otimes \mathbf{I}_K$$

and

$$J_v^*(f_{x|zv}) = \boldsymbol{\beta}^T \tilde{\mathbf{S}}_v \boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_v = \mathbf{I}_I \otimes \mathbf{I}_L \otimes \mathbf{S}'_v.$$

Hence

$$J(f_{x|zv}) \approx J^*(f_{x|zv}) = \lambda_x J_x^*(f_{x|zv}) + \lambda_z J_z^*(f_{x|zv}) + \lambda_v J_v^*(f_{x|zv}),$$

where any constants, h , have been absorbed into the λ_j . Again, this penalty construction clearly generalizes to any number of covariates. Figure 4.8 attempts to illustrate what the penalties actually measure, for a smooth of two variables.

Given its model matrix and penalties, the coefficients and smoothing parameters of a tensor product smooth can be estimated as GAM components using the methods of sections 4.3 and 4.6 or 4.7. These smooths have the nice property of being invariant to rescaling of the covariates, provided only that the marginal smooths are similarly invariant (which is always the case in practice).

Note that it is possible to omit the reparameterization of the marginal smooths, in terms of function values, and to work with penalties of the form

$$\boldsymbol{\beta}^T \bar{\mathbf{S}}_z \boldsymbol{\beta} \text{ where } \bar{\mathbf{S}}_z = \mathbf{I}_I \otimes \mathbf{S}_z \otimes \mathbf{I}_K$$

for example: Eilers and Marx (2003) successfully used this approach to smooth with respect to two variables using tensor products of B-splines. A potential problem with the approach is that the penalties no-longer have the interpretation in terms of (averaged) function shape, that is inherited from the marginal smooths when re-parameterization is used. Another proposal in the literature is to use single penalties

of the form:

$$\boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta} \text{ where } \mathbf{S} = \mathbf{S}_1 \otimes \mathbf{S}_2 \otimes \cdots \otimes \mathbf{S}_d.$$

but this often leads to severe undersmoothing. The reason for the undersmoothing is straightforward: the rank of \mathbf{S} is the product of the ranks of the \mathbf{S}_j , and in practice this is often far too low for practical work. For example, consider a smooth of 3 predictors constructed as a tensor product smooth of 3 cubic spline bases, each of rank 5. The resulting smooth would have 125 free parameters, but a penalty matrix of rank 27. This means that varying the weight given to the penalty would only result in the effective degrees of freedom for the smooth varying between 98 and 125: not a very useful range. By contrast, for the same marginal bases, the multiple term penalties would have rank 117, leading to a much more useful range of effective degrees of freedom of between 8 and 125.

4.2 Setting up GAMs as penalized GLMs

As we saw in Chapter 3, a GAM models a response variable, y_i , using a model structure of a form like:

$$g(\mu_i) = \mathbf{X}_i^* \boldsymbol{\theta} + f_1(x_{1i}) + f_2(x_{2i}, x_{3i}) + f_3(x_{4i}) + \cdots \quad (4.9)$$

where $\mu_i \equiv \mathbb{E}(y_i)$ and $y_i \sim$ ‘an exponential family distribution’[§]. Here g is a known, monotonic, twice differentiable, link function; \mathbf{X}_i^* is the i^{th} row of a model matrix for any strictly parametric model components, with parameter vector $\boldsymbol{\theta}$; the f_j are smooth functions of the covariates x_j .

To estimate such a model we can specify a basis for each smooth function, along with a corresponding definition of what is meant by smoothness/wigginess of the function. Starting with the bases, we choose a set of basis functions, b_{ji} , for each function, so that it can be represented as:

$$f_j(x_j) = \sum_{i=1}^{q_j} \beta_{ji} b_{ji}(x_j)$$

where x_j may be a vector quantity and the β_{ji} are coefficients of the smooth, which will need to be estimated as part of model fitting.

Given a basis, it is straightforward to create a model matrix, $\tilde{\mathbf{X}}_j$, for each smooth. If \mathbf{f}_j is the vector such that $\mathbf{f}_j = f_j(x_{ji})$ and $\tilde{\boldsymbol{\beta}}_j = [\beta_{j1}, \beta_{j2}, \dots, \beta_{jq_j}]^T$, then

$$\mathbf{f}_j = \tilde{\mathbf{X}}_j \tilde{\boldsymbol{\beta}}_j$$

where $\tilde{\mathbf{X}}_{j,ik} = b_{jk}(x_{ji})$, and the covariate, x_j , may sometimes be a vector quantity.

Typically, (4.9) is not an identifiable model, unless each smooth is subject to a ‘centering constraint’. A suitable constraint is that the sum (or mean) of the elements of

[§] although if we take a quasi-likelihood approach we can relax the distributional assumption somewhat and only specify a mean variance relationship for y_i .

f_j should be zero, which can be written as

$$\mathbf{1}^T \tilde{\mathbf{X}}_j \tilde{\beta}_j = 0.$$

Using the approach taken in section 1.8.1, this constraint can easily be absorbed by re-parameterization. Specifically we find a matrix \mathbf{Z} , the $q_j - 1$ columns of which are orthogonal, and which satisfies:

$$\mathbf{1}^T \tilde{\mathbf{X}}_j \mathbf{Z} = 0.$$

Now reparameterizing the smooth in terms of $q_j - 1$ new parameters, β_j , such that $\tilde{\beta}_j = \mathbf{Z}\beta_j$, we obtain a new model matrix for the j^{th} term, $\mathbf{X}_j = \tilde{\mathbf{X}}_j \mathbf{Z}$, such that $f_j = \mathbf{X}_j \beta_j$ automatically satisfies the centering constraint. \mathbf{Z} is never formed explicitly, since it can be represented by a single Householder matrix (see section A.5).

Given centered model matrices, for each smooth term, (4.9) can now be re-written as

$$g(\mu_i) = \mathbf{X}_i \beta \quad (4.10)$$

where $\mathbf{X} = [\mathbf{X}^* : \mathbf{X}_1 : \mathbf{X}_2 : \dots]$ and $\beta^T = [\theta^T, \beta_1^T, \beta_2^T, \dots]$. Clearly (4.10) is just a GLM, and we can therefore write down its likelihood, $l(\beta)$, say. Equally clearly, if the q_j are large enough that we have a reasonable chance of accurately representing the unknown f_j 's, and β is estimated by ordinary likelihood maximization, then there is a good chance of substantially overfitting. For this reason, GAMs are usually estimated by penalized likelihood maximization, where the penalties are designed to suppress overly wiggly estimates of the f_j terms.

The most convenient penalties to work with are those which measure function wiggliness as a quadratic form in the coefficients of the function. For example, the wiggliness of the j^{th} function might be measured by $\tilde{\beta}_j^T \tilde{\mathbf{S}}_j \tilde{\beta}_j$, where $\tilde{\mathbf{S}}_j$ is a matrix of known coefficients. Sometimes $\tilde{\mathbf{S}}_j$ may itself be a weighted sum of simpler matrices of known coefficients, where the weights are parameters to be estimated (see section 4.1.8). The centering reparameterization would convert this penalty to the form $\beta_j^T \bar{\mathbf{S}}_j \beta_j$ where $\bar{\mathbf{S}}_j = \mathbf{Z}^T \tilde{\mathbf{S}}_j \mathbf{Z}$. Notationally it is convenient to re-write the penalty in terms of the full coefficient vector β , so that it becomes $\beta^T \mathbf{S}_j \beta$, where \mathbf{S}_j is just $\bar{\mathbf{S}}_j$ padded with zeroes so that $\beta^T \mathbf{S}_j \beta \equiv \beta_j^T \tilde{\mathbf{S}}_j \tilde{\beta}_j$.

Given a wiggliness measure for each function, we can define a penalized likelihood for the model,

$$l_p(\beta) = l(\beta) - \frac{1}{2} \sum_j \lambda_j \beta^T \mathbf{S}_j \beta, \quad (4.11)$$

where the λ_j are smoothing parameters, controlling the tradeoff between goodness of fit of the model and model smoothness. Given values for the λ_j , then l_p is maximized to find $\hat{\beta}$, but the λ_j must themselves be estimated.

4.2.1 Variable coefficient models

Hastie and Tibshirani (1993) proposed a class of models, which they dubbed ‘variable coefficient models’. These models are basically GAMs, in which the smooths

may be multiplied by some known covariate. An example is

$$g(\mu_i) = \mathbf{X}_i^* \boldsymbol{\theta} + f_1(x_{1i})x_{2i} + f_2(x_{3i}, x_{4i})x_{5i} + f_3(x_{6i})x_{7i} + \dots$$

Setting up these models for estimation by penalized regression methods is straightforward. Each row of the model matrix for the smooth is multiplied by the corresponding value of the covariate. For example, the formal expression for the model matrix for the term $f_1(x_{1i})x_{2i}$, in the above example, is simply $\text{diag}(\mathbf{x}_2)\mathbf{X}_1$, where \mathbf{X}_1 is the model matrix for $f_1(x_{1i})$, and $\text{diag}(\mathbf{x}_2)$ is a diagonal matrix with x_{2i} at the i^{th} position on its leading diagonal (in the terminology of the `mgcv` package, covered in Chapter 5, variables like x_2 are known as ‘by’ variables). No other modification of the GAM framework presented in this chapter is necessary.

Note that such models make it easy to condition smooths on factors. For example, if a smooth of x should depend on which of two levels of a factor, a , pertains for a particular response observation, we could write a model as

$$g(\mu_i) = f(x_i)z_{1i} + f(x_i)z_{2i},$$

where z_j is an indicator variable for whether the corresponding factor level is j .

4.3 Justifying P-IRLS

As we saw in chapter 3, the GAM penalized likelihood, (4.11), can be maximized by penalized iteratively re-weighted least squares, and in this section some justification for this approach is provided. For notational compactness (4.11) can be re-written as

$$l_p(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - \frac{1}{2}\boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta}$$

where $\mathbf{S} = \sum_j \lambda_j \mathbf{S}_j$, and for the moment the λ_j are taken as known. To maximize l_p we set its derivatives with respect to the β_j to zero:

$$\frac{\partial l_p}{\partial \beta_j} = \frac{\partial l}{\partial \beta_j} - [\mathbf{S}\boldsymbol{\beta}]_j = \frac{1}{\phi} \sum_{i=1}^n \frac{y_i - \mu_i}{V(\mu_i)} \frac{\partial \mu_i}{\partial \beta_j} - [\mathbf{S}\boldsymbol{\beta}]_j = 0,$$

where $[\cdot]_j$ denotes the j^{th} row of a vector. But by the same argument used in section 2.1.2 these equations are exactly those that would have to be solved to maximize the penalized non-linear least squares problem

$$\mathcal{S}_p = \sum_{i=1}^n \frac{(y_i - \mu_i)^2}{\text{var}(Y_i)} + \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta},$$

assuming that the $\text{var}(Y_i)$ terms were known. Again following section 2.1.2 it is easy to show that, in the vicinity of some parameter vector estimate $\hat{\boldsymbol{\beta}}^{[k]}$,

$$\mathcal{S}_p \simeq \left\| \sqrt{\mathbf{W}^{[k]}} (\mathbf{z}^{[k]} - \mathbf{X}\boldsymbol{\beta}) \right\|^2 + \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta}, \quad (4.12)$$

where, if g is the model link function, $\mathbf{z}^{[k]}$ is a vector of pseudodata and $\mathbf{W}^{[k]}$ is a diagonal matrix with diagonal elements $w_i^{[k]}$ then

$$w_i^{[k]} = \frac{1}{V(\mu_i^{[k]})g'(\mu_i^{[k]})^2} \text{ and } z_i = g(\mu_i^{[k]})(y_i - \mu_i^{[k]}) + \mathbf{X}_i \hat{\beta}^{[k]}.$$

Hence given smoothing parameters, the maximum penalized likelihood estimates, $\hat{\beta}$, are obtained by iterating the steps

1. Given the current $\hat{\beta}^{[k]}$ calculate the pseudodata $\mathbf{z}^{[k]}$ and weights $w_i^{[k]}$.
2. Minimize 4.12 w.r.t. β to find $\hat{\beta}^{[k+1]}$. Increment k .

to convergence. See O'Sullivan et al. (1986) for an early reference on penalized likelihood maximization for smooth models.

4.4 Degrees of freedom and residual variance estimation

Before covering λ estimation, it is helpful to consider the notion of degrees of freedom for a GAM, and this will also lead on naturally to the question of scale parameter estimation. How many degrees of freedom does a fitted GAM have? Clearly, if the smoothing parameters were all set to zero then the degrees of freedom of the model would be the dimension of β (less the number of identifiability constraints). At the opposite extreme, if all the smoothing parameters are very high then the model will be quite inflexible and will hence have very few degrees of freedom. One way of measuring the flexibility of the fitted model is to define the *effective degrees of freedom* as $\text{tr}(\mathbf{A})$, by analogy with section 1.3.5. It is fairly easy to show that the maximum of $\text{tr}(\mathbf{A})$ is just the number of parameters less the number of constraints, and similarly that the minimum values is $\text{rank}(\sum_i \mathbf{S}_i)$ less than this. As the smoothing parameters vary, from zero to infinity, the effective degrees of freedom moves smoothly between these limits.

Now the degrees of freedom of the model are, in effect, reduced by the application of the penalties during fitting, and penalties for different model terms will have different smoothing parameters, and will hence penalize their smooth functions differently. It is therefore natural to want to break the effective degrees of freedom down, into effective degrees of freedom for each smooth. In fact one might as well go further still, and try to ascertain the effective degrees of freedom associated with each $\hat{\beta}_i$, separately. Again this is natural, since the penalties generally penalize each element of β differently.

To this end, first define[¶] $\mathbf{P} \equiv (\mathbf{X}^\top \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^\top$, so that $\hat{\beta} = \mathbf{P} \mathbf{y}$ (in the un-weighted additive model case). Hence $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{XP})$. Now define \mathbf{P}_i^0 to be \mathbf{P} with all its

[¶] Again writing $\mathbf{S} = \sum_j \lambda_j \mathbf{S}_j$.

rows zeroed except the i^{th} , which is left unchanged. In this case the vector $\mathbf{P}_i^0 \mathbf{y}$ has $\hat{\beta}_i$ for its i^{th} element, and zero elsewhere, while

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^p \text{tr}(\mathbf{X} \mathbf{P}_i^0).$$

So, it is natural to interpret $\text{tr}(\mathbf{X} \mathbf{P}_i^0)$ as the effective degrees of freedom associated with the i^{th} parameter. However,

$$\text{tr}(\mathbf{X} \mathbf{P}_i^0) = (\mathbf{P} \mathbf{X})_{i,i},$$

so the vector of effective degrees of freedom for the model parameters is given by the leading diagonal of

$$\mathbf{F} = \mathbf{P} \mathbf{X} = (\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{X}.$$

For an intuitive insight into the meaning of effective degrees of freedom, an alternative argument is perhaps more useful. Without penalization, the parameter estimates would be,

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

With penalization the estimates are

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= \mathbf{F} \tilde{\boldsymbol{\beta}}\end{aligned}$$

i.e. \mathbf{F} is the matrix mapping the un-penalized estimates to the penalized ones. Now $\partial \hat{\beta}_i / \partial \tilde{\beta}_i = F_{ii}$, meaning that F_{ii} measures how much the penalized $\hat{\beta}_i$ will change, as a result of a unit change in the un-penalized $\tilde{\beta}_i$. This is why F_{ii} measures the effective degrees of freedom of the i^{th} penalized parameters: the un-penalized parameter has one degree of freedom, but the penalties effectively shrink that freedom to vary, by a factor F_{ii} . Actually things are not quite as straightforward as this implies, since there is actually no general guarantee that $F_{ii} > 0$, although for most reasonable bases and penalties the F_{ii} are positive, unless autocorrelated data are being modelled (as in chapter 6).

In the general, weighted, case the degrees of freedom matrix is easily shown to be

$$\mathbf{F} = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{X}.$$

4.4.1 Residual variance or scale parameter estimation

In the identity link, normal errors case, then by analogy with linear regression, σ^2 could be estimated by the residual sum of squares divided by the residual degrees of freedom:

$$\hat{\sigma}^2 = \frac{\|\mathbf{y} - \mathbf{A} \mathbf{y}\|^2}{n - \text{tr}(\mathbf{A})}, \quad (4.13)$$

where $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{F})$. In fact (4.13) is not unbiased, since it is readily shown that

$$\mathbb{E}(\|\mathbf{y} - \mathbf{Ay}\|^2) = \sigma^2 [n - 2\text{tr}(\mathbf{A}) + \text{tr}(\mathbf{A}^\top \mathbf{A})] + \mathbf{b}^\top \mathbf{b}, \quad (4.14)$$

where $\mathbf{b} = \boldsymbol{\mu} - \mathbf{A}\boldsymbol{\mu}$ represents the smoothing bias. Re-arranging (4.14) and substituting $\hat{\boldsymbol{\mu}}$ for $\boldsymbol{\mu}$, yields an alternative estimator for $\hat{\sigma}^2$, but the need to estimate \mathbf{b} means that it is still a biased estimator, and a rather complicated one to work with. For this reason (4.13) is usually preferred.

In the *generalized* additive model case, the scale parameter is usually estimated by the Pearson-like scale estimator,

$$\hat{\phi} = \frac{\sum_i V(\hat{\mu}_i)^{-1}(y_i - \hat{\mu}_i)^2}{n - \text{tr}(\mathbf{A})}.$$

4.5 Smoothing Parameter Estimation Criteria

Penalized likelihood maximization can only estimate model coefficients, $\boldsymbol{\beta}$, given smoothing parameters $\boldsymbol{\lambda}$, so this section covers the topic of smoothing parameter estimation introduced in section 3.2.3. Two basic approaches are useful: when the scale parameter is known then attempting to minimize the expected mean square error leads to estimation by Mallow's C_p /UBRE; when the scale parameter is unknown then attempting to minimize prediction error leads to cross validation or GCV. As usual it helps to start with the additive model case, and to generalize it at the end.

4.5.1 Known scale parameter: UBRE

An appealing way of estimating smoothing parameters would be to choose them in order that $\hat{\boldsymbol{\mu}}$ is as close as possible to the true $\boldsymbol{\mu} \equiv \mathbb{E}(\mathbf{y})$. An appropriate measure of this proximity might be M , the expected Mean Square Error (MSE) of the model, and in section 1.8.5, the argument^{||} leading to (1.14) implies that this is:

$$\mathbb{E}(M) = \mathbb{E}(\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2/n) = \mathbb{E}(\|\mathbf{y} - \mathbf{Ay}\|^2)/n - \sigma^2 + 2\text{tr}(\mathbf{A})\sigma^2/n. \quad (4.15)$$

Hence it seems reasonable to choose smoothing parameters which minimize an estimate of this expected MSE, that is to minimize the Un-Biased Risk Estimator (Craven and Wahba, 1979),

$$\mathcal{V}_u(\boldsymbol{\lambda}) = \|\mathbf{y} - \mathbf{Ay}\|^2/n - \sigma^2 + 2\text{tr}(\mathbf{A})\sigma^2/n, \quad (4.16)$$

which is also Mallow's C_p (Mallows, 1973). Note that the r.h.s. of (4.16) depends on the smoothing parameters through \mathbf{A} .

If σ^2 is known then estimating $\boldsymbol{\lambda}$ by minimizing \mathcal{V}_u works well, but problems arise

^{||} which makes no assumptions that are invalidated by *penalized* least squares estimation.

if σ^2 has to be estimated. For example, substituting the approximation

$$\mathbb{E}(\|\mathbf{y} - \mathbf{Ay}\|^2) = \sigma^2(n - \text{tr}(\mathbf{A})), \quad (4.17)$$

implied by (4.13), into (4.15) yields,

$$M = \mathbb{E}(\|\boldsymbol{\mu} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2/n) = \frac{\text{tr}(\mathbf{A})}{n}\sigma^2 \quad (4.18)$$

and the MSE estimator $\tilde{M} = \text{tr}(\mathbf{A})\hat{\sigma}^2/n$. Now consider comparison of 1 and 2 parameter models using \tilde{M} : the 2 parameter model has to reduce $\hat{\sigma}^2$ to less than half the one parameter σ^2 estimate before it would be judged to be an improvement. Clearly, therefore, \tilde{M} is not a suitable basis for model selection.

4.5.2 Unknown scale parameter: Cross Validation

As we have seen, naively attempting to minimize the average square error in model predictions of $\mathbb{E}(\mathbf{y})$, will not work well when σ^2 is unknown. An alternative is to base smoothing parameter estimation on mean square *prediction error*: that is on the average squared error in predicting a new observation y using the fitted model. The expected mean square prediction error is readily shown to be

$$P = \sigma^2 + M.$$

The direct dependence on σ^2 tends to mean that criteria based P are much more resistant to over-smoothing, which would inflate the σ^2 estimate, than are criteria based on M alone.

The most obvious way to estimate P is to use cross validation (e.g. Stone, 1974). By omitting a datum, y_i , from the model fitting process, it becomes independent of the model fitted to the remaining data. Hence the squared error in predicting y_i is readily estimated, and by omitting all data in turn we arrive at the ordinary cross validation estimate of P , given in section 3.2.3:

$$\mathcal{V}_o = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{\mu}_i^{[-i]})^2$$

where $\hat{\mu}_i^{[-i]}$ denotes the prediction of $\mathbb{E}(y_i)$ obtained from the model fitted to all data except y_i .

Fortunately, calculating \mathcal{V}_o by performing n model fits, to obtain the n terms $\hat{\mu}_i^{[-i]}$, is unnecessary. To see this, first consider the penalized least squares objective which in principle has to be minimized to find the i^{th} term in the OCV score:

$$\sum_{\substack{j=1 \\ j \neq i}}^n (y_j - \hat{\mu}_j^{[-i]})^2 + \text{Penalties.}$$

Clearly, adding zero to this objective will leave the estimates that minimize it com-

pletely unchanged. So we can add the term $(\hat{\mu}_i^{[-i]} - \hat{\mu}_i^{[-i]})^2$ to obtain

$$\sum_{j=1}^n (y_j^* - \hat{\mu}_j^{[-i]})^2 + \text{Penalties}, \quad (4.19)$$

where $\mathbf{y}^* = \mathbf{y} - \bar{\mathbf{y}}^{[i]} + \bar{\boldsymbol{\mu}}^{[i]}$: $\bar{\mathbf{y}}^{[i]}$ and $\bar{\boldsymbol{\mu}}^{[i]}$ are vectors of zeroes except for their i^{th} elements which are y_i and $\hat{\mu}_i^{[-i]}$, respectively.

Fitting, by minimizing (4.19), obviously results in i^{th} prediction $\hat{\mu}_i^{[-i]}$, and also in an influence matrix \mathbf{A} , which is just the influence matrix for the model fitted to all the data (since (4.19) has the structure of the fitting objective for the model of the whole data). So considering the i^{th} prediction we have that:

$$\hat{\mu}_i^{[-i]} = \mathbf{A}_i \mathbf{y}^* = \mathbf{A}_i \mathbf{y} - A_{ii} y_i + A_{ii} \hat{\mu}_i^{[-i]} = \hat{\mu}_i - A_{ii} y_i + A_{ii} \hat{\mu}_i^{[-i]},$$

where $\hat{\mu}_i$ is from the fit to the full \mathbf{y} . Subtraction of y_i from both sides, and a little rearrangement then yields

$$y_i - \hat{\mu}_i^{[-i]} = (y_i - \hat{\mu}_i) / (1 - A_{ii}),$$

so that the OCV score becomes

$$\mathcal{V}_o = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}, \quad (4.20)$$

which can clearly be calculated from a single fit of the original model. Stone (1977) demonstrates the asymptotic equivalence of cross validation and AIC.

Problems with Ordinary Cross Validation

OCV is a reasonable way of estimating smoothing parameters, but suffers from two potential drawbacks. Firstly, it is computationally expensive to minimize in the additive model case, where there may be several smoothing parameters. Secondly, it has a slightly disturbing lack of invariance (see Golub et al., 1979; Wahba, 1990, p. 53).

To appreciate the invariance problem, consider the additive model fitting objective,

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{i=1}^m \lambda_i \boldsymbol{\beta}^\top \mathbf{S}_i \boldsymbol{\beta},$$

again. Given smoothing parameters, all inferences about $\boldsymbol{\beta}$, made on the basis of minimizing this objective, are identical to the inferences that would be made by using the alternative objective:

$$\|\mathbf{Q}\mathbf{y} - \mathbf{Q}\mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{i=1}^m \lambda_i \boldsymbol{\beta}^\top \mathbf{S}_i \boldsymbol{\beta},$$

where \mathbf{Q} is any orthogonal matrix of appropriate dimension. However, the two objectives generally give rise to *different* OCV scores.

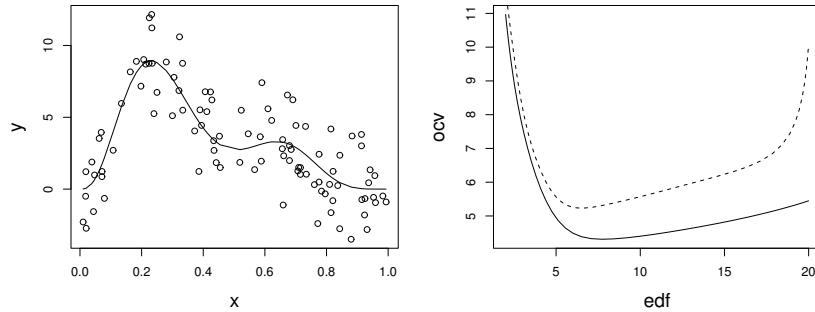


Figure 4.9 *Lack of invariance or ordinary cross validation*. The left panel shows a smooth function (continuous line) and some data sampled from it at random x values with noise added in the y direction. The right panel shows OCV scores against estimated degrees of freedom for a rank 20 penalized regression spline fitted to the data: the continuous curve is the OCV score for the model fitted using the original objective; the dashed line shows the OCV score for the model fitted by an alternative objective. Both objectives result in identical inferences about the model coefficients for any given smoothing parameters, so the difference in OCV score is somewhat unsatisfactory. GCV and UBRE do not suffer from this problem.

Figure 4.9 illustrates this problem for a smooth of x, y data. The right hand side of figure 4.9 shows \mathcal{V}_o plotted against effective degrees of freedom, for the same rank 20 penalized regression spline fitted to the data in the left panel: the continuous curve is \mathcal{V}_o corresponding to the original objective, while the dashed curve is \mathcal{V}_o for the orthogonally transformed objective. For this example, \mathbf{Q} was obtained from the QR decomposition of \mathbf{X} , but similar differences occur for arbitrary orthogonal \mathbf{Q} . Note how the ‘optimal’ degrees of freedom differ between the two OCV scores, and that this occurs despite the fact that the fitted models at any particular EDF are identical.

4.5.3 Generalized Cross Validation

The problems with ordinary cross validation arise because, despite parameter estimates, effective degrees of freedom and expected prediction error being invariant to rotation of $\mathbf{y} - \mathbf{X}\beta$ by any orthogonal matrix \mathbf{Q} , the elements, A_{ii} , on the leading diagonal of the influence matrix, are not invariant and neither are the individual terms in the summation in (4.20). This sensitivity to an essentially arbitrary choice about how fitting is done is unsatisfactory, but what can be done to improve matters?

One approach is to consider what might make for ‘good’ or ‘bad’ rotations of $\mathbf{y} - \mathbf{X}\beta$ and to decide to perform cross validation on a particularly nicely rotated problem. One thing that would appear to be undesirable is to base cross validation on data in which a few points have very high leverage relative to the others. That is highly uneven A_{ii} values are undesirable, as they will tend to cause the cross validation score

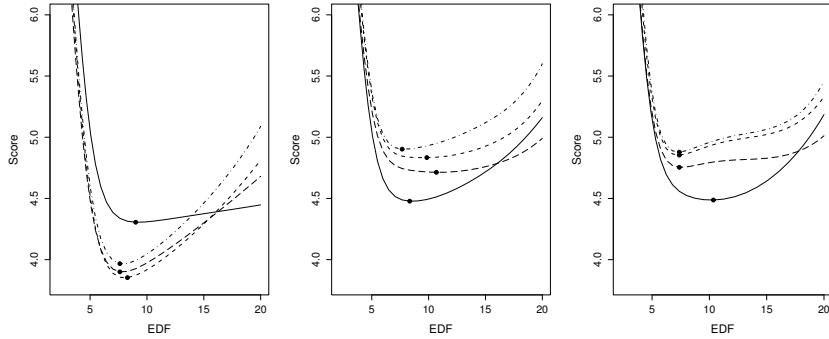


Figure 4.10 *Comparison of different smoothing parameter estimation criteria for rank 20 penalized regression splines fitted to three replicates of the simulated data shown in figure 4.9. The curves are as follows (in ascending order at the 20 EDF end of the left hand panel): the continuous curve is observed mean square error + known σ^2 ; the long dashed curve is $V_u + \sigma^2$; the short dashed curve is V_g , the GCV criterion; the alternating dashes and dots are V_o , the OCV criterion. The dot on each curve shows the location of its minimum.*

(4.20) to be dominated by a small proportion of the data. This suggests choosing the rotation \mathbf{Q} in order to make the A_{ii} as even as possible.

In fact it is possible to choose \mathbf{Q} in order to make all the A_{ii} equal. To see this note that if \mathbf{A} is the influence matrix for the original problem, then the influence matrix for the rotated problem is

$$\mathbf{A}_Q = \mathbf{Q} \mathbf{A} \mathbf{Q}^\top$$

but if \mathbf{B} is any matrix such that $\mathbf{B}\mathbf{B}^\top = \mathbf{A}$ then the influence matrix can be written:

$$\mathbf{A}_Q = \mathbf{Q} \mathbf{B} \mathbf{B}^\top \mathbf{Q}^\top.$$

Now if the orthogonal matrix \mathbf{Q} is such that each row of $\mathbf{Q}\mathbf{B}$ has the same Euclidean length, then it is clear that all the elements on the leading diagonal of the influence matrix, \mathbf{A}_Q , have the same value, which must be $\text{tr}(\mathbf{A})/n$, since $\text{tr}(\mathbf{A}_Q) = \text{tr}(\mathbf{Q}\mathbf{A}\mathbf{Q}^\top) = \text{tr}(\mathbf{A}\mathbf{Q}^\top\mathbf{Q}) = \text{tr}(\mathbf{A})$.

Does a \mathbf{Q} with this neat row-length-equalizing property actually exist? It is easy to see that it does, by construction. Firstly note that it is always possible to produce an orthogonal matrix to be applied from the left, whose action is to perform a rotation which affects only two rows of a target matrix, such as \mathbf{B} : a matrix with this property is known as a Givens rotation. As the angle of rotation, θ , increases smoothly from zero, the Euclidean lengths of the two rows varies smoothly, although the sum of their squared lengths remains constant, as befits a rotation. Once θ reaches 90 degrees, the row lengths are interchanged, since the magnitudes of the elements of the rows have been interchanged. Hence there must exist an intermediate θ at which the row lengths are exactly equal. Let the Givens rotation with this θ be termed the ‘averaging rotation’ for the two rows. If we now apply an averaging rotation to the pair of rows of

\mathbf{B} having smallest and largest Euclidean lengths, then the range of row lengths in the modified \mathbf{B} will automatically be reduced. Iterating this process must eventually lead to all row lengths being equal, and the product of the averaging rotations employed in the iteration yields \mathbf{Q} .

With this best rotation of the fitting problem, the ordinary cross validation score (4.20) can be written

$$\mathcal{V}_g = \frac{n\|\mathbf{y} - \hat{\boldsymbol{\mu}}\|^2}{[n - \text{tr}(\mathbf{A})]^2}. \quad (4.21)$$

which is known as the *Generalized Cross Validation* score (GCV, Craven and Wahba, 1979; Golub et al., 1979). Notice that we do not have to actually perform the rotation in order to use GCV. Also, since the expected prediction error is unaffected by the rotation, and GCV is just OCV on the rotated problem, GCV must be as valid an estimate of prediction error as OCV, but GCV has the nice property of being invariant.

Figure 4.10 compares GCV, OCV and UBRE scores for some simulated data. Unsurprisingly the criteria tend to be in quite close agreement, and are all in closer agreement with each other than with the observed $\text{MSE} + \sigma^2$.

4.5.4 GCV/UBRE/AIC in the Generalized case

The previous sections have only dealt with additive models rather than generalized additive models, but the generalization is straightforward. The GAM fitting objective can be written in terms of the model *deviance* (equation (2.7), section 2.1.6), as

$$D(\boldsymbol{\beta}) + \sum_{j=1}^m \lambda_j \boldsymbol{\beta}^\top \mathbf{S}_j \boldsymbol{\beta}$$

which is minimized w.r.t. $\boldsymbol{\beta}$. Given $\boldsymbol{\lambda}$ then this objective can be quadratically approximated, by

$$\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\|^2 + \sum_{j=1}^m \lambda_j \boldsymbol{\beta}^\top \mathbf{S}_j \boldsymbol{\beta}, \quad (4.22)$$

where the approximation should reasonably capture the dependence of the penalized deviance on $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$, in the vicinity of the current $\boldsymbol{\lambda}$, and the corresponding minimizing values of $\boldsymbol{\beta}$. This approximation is justified in section 2.1.3, which shows that the first two derivatives of $D(\boldsymbol{\beta})$ and $\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\|^2$ are approximately equal, while the approximate distributional results for the Pearson statistic and Deviance (e.g. section 2.1.7) suggest that they have approximately equal expected value.

Now, for the problem (4.22), it is straightforward to re-use the arguments of sections 4.5.2 and 4.5.3 in order to derive a GCV score for smoothing parameter selection:

$$\mathcal{V}_g^w = \frac{n\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\|^2}{[n - \text{tr}(\mathbf{A})]^2}. \quad (4.23)$$

Of course, this approximation is only valid locally to the $\boldsymbol{\lambda}$ used to find \mathbf{z} and \mathbf{W} ,

but re-using the approximation from section 2.1.3, a globally applicable GCV score (Hastie and Tibshirani, 1990) can be obtained

$$\mathcal{V}_g = \frac{nD(\hat{\beta})}{(n - \text{tr}(\mathbf{A}))^2}. \quad (4.24)$$

Applying similar arguments to the UBRE criterion, yields

$$\mathcal{V}_u^w = \frac{1}{n}\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\beta)\|^2 - \sigma^2 + \frac{2}{n}\text{tr}(\mathbf{A})\sigma^2. \quad (4.25)$$

and hence

$$\mathcal{V}_u = \frac{1}{n}D(\hat{\beta}) - \sigma^2 + \frac{2}{n}\text{tr}(\mathbf{A})\sigma^2. \quad (4.26)$$

Notice how this criterion is effectively just a linear transformation of AIC.

Although the definition of \mathcal{V}_g is intuitively very reasonable (since the deviance is specifically constructed to behave like the residual sum of squares term it is replacing) the given justification for \mathcal{V}_g is not strong. In particular the value of $D(\hat{\beta})$ is often not all that close to that of $\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\hat{\beta})\|^2$, so that the final approximation in the argument leading to \mathcal{V}_g is rather poor. One way of dealing with this would be to recognize that in reality $D(\hat{\beta}) + k \approx \|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\hat{\beta})\|^2$, where k is a constant which could be estimated from an initial model fit in which it is neglected. A second fit could then be performed in which the GCV score used $D(\hat{\beta}) + k$ in place of $D(\hat{\beta})$. However, in practice this seems not to result in large enough improvements to be worthwhile (see exercise 9). Clearly the whole issue presents no problem for the UBRE score, for which the value of k is immaterial.

Intuitively, it might also seem reasonable** to replace the deviance by the Pearson statistic in order to obtain GCV and UBRE scores for the generalized case. The resulting scores are:

$$\mathcal{V}_g^p = \frac{n \sum_{i=1}^n V(\hat{\mu}_i)^{-1}(y_i - \hat{\mu}_i)^2}{[n - \text{tr}(\mathbf{A})]^2}$$

and

$$\mathcal{V}_u^p = \frac{1}{n} \sum_{i=1}^n V(\hat{\mu}_i)^{-1}(y_i - \hat{\mu}_i)^2 - \sigma^2 + \frac{2}{n}\text{tr}(\mathbf{A})\sigma^2.$$

Actually, these scores are a little problematic: they are more difficult to justify using the sorts of arguments presented in previous sections, because of the dependence of V on $\hat{\mu}_i$ (which also means that the Pearson statistic is not a quadratic approximation to the deviance). More seriously, they can lead to oversmoothing, which is particularly severe for binary data. It's easy to show that for binary data $\mathbb{E}(\mathcal{V}_g^p) = n^2/(n - \text{tr}(\mathbf{A}))^2$, which is clearly minimized by adopting the smoothest possible model (exercise 4).

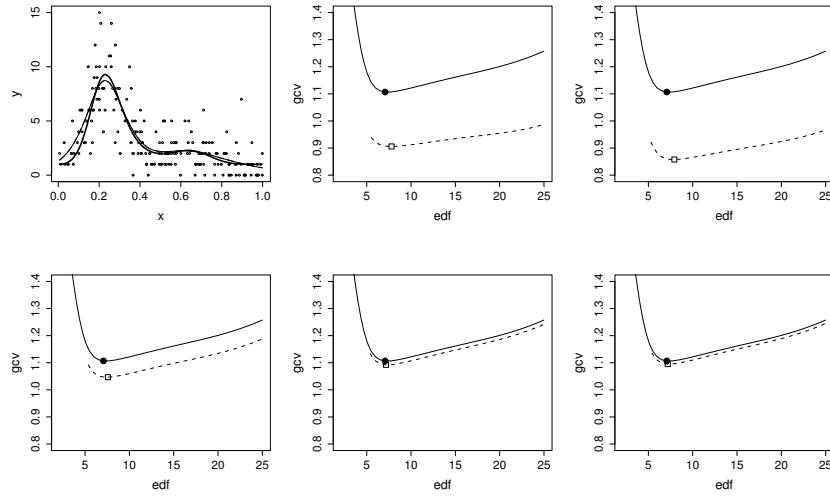


Figure 4.11 *Illustration of performance iteration.* The top left panel shows 200 simulated x, y data: the x co-ordinates are uniform random deviates and the y co-ordinates are Poisson random variables with x dependent mean given by the thick curve. The data are modelled as $\log(\mu_i) = f(x_i)$, $y_i \sim \text{Poi}(\mu_i)$ with f represented using a rank 25 penalized regression spline. The effective degrees of freedom for f were chosen by performance iteration. Working from middle top to bottom right, the remaining panels illustrate the progress of the performance iteration. In each panel the continuous curve is the (deviance based) GCV score for the model plotted against EDF (this curve is the same in each panel), while the dashed curve is the GCV score for the working linear model of the current P-IRLS iteration, against EDF. In each panel the minimum of each curve is marked. Notice how the two curves become increasingly close in the vicinity of the finally selected model, as iteration progresses: in fact the correspondence is unusually close in the replicate shown, often the dashed curve is a further below the continuous curve, with the performance iteration therefore suggesting a slightly more complex model. The thin continuous curve, in the upper left panel, shows the model fits corresponding to the minimum of the model GCV score and the model corresponding to the performance iteration estimate: they are indistinguishable.

Approaches to GAM GCV/UBRE minimization

In the GAM case there are two possible numerical strategies for estimating smoothing parameters using \mathcal{V}_g or \mathcal{V}_u minimization:

- $\mathcal{V}_{g/u}$ can be minimized directly, which means that the P-IRLS scheme must be iterated to convergence for each trial set of smoothing parameters. This is sometimes termed *outer* iteration, since estimation is outer to the P-IRLS loop.
- $\mathcal{V}_{g/u}^w$ can be minimized and smoothing parameters selected for each working pe-

** at least to those as foolish as the author of this volume.

nalized linear model of the P-IRLS iteration. This is known as *performance iteration*, since it is typically rather computationally efficient.

Performance iteration (originally proposed by Gu, 1992) usually converges, and requires only that we have available a reliable and efficient method for score minimization in the context of penalized least squares problems. It typically requires no more P-IRLS iterations than are required to fit a model with known smoothing parameters. Figure 4.11 illustrates how performance iteration works using a simple model of Poisson data. Note how the performance iteration results do not exactly match the results obtained by direct outer iteration.

The fact that performance iteration does not exactly minimize $\mathcal{V}_{u/g}$, of the actual model, is slightly unsatisfactory when it comes to comparing alternative models on the basis of their $\mathcal{V}_{u/g}^w$ scores — if the scores of two models are quite similar then there is always a possibility that the model with the lower score, from performance iteration, might actually have the higher score, if you minimized the scores directly. Of course in practice this is only likely to occur when models are essentially indistinguishable, but it is still not entirely satisfactory. A more substantial concern with performance iteration is that it can fail to converge at all.

The simplest sort of failure for performance iteration is as follows. At some stage in the P-IRLS a set of smoothing parameter estimates and coefficient estimates, $\{\hat{\lambda}, \hat{\beta}\}$ is obtained; this set in turn implies a working linear model and GCV score which yields a new set of estimates, $\{\hat{\lambda}, \hat{\beta}\}$; this new set of estimates itself yields a new working model and GCV score, which in turn yield a new set of estimates, but these turn out to be $\{\hat{\lambda}, \hat{\beta}\}$. This cycling never ends and convergence therefore never occurs. Similar problems involving cycling through a much larger number of sets of estimates also occur, so that simple tactics for detecting and dealing with this problem are not viable. The cycling is related to the fitting geometry issues discussed in section 2.2.2, which are exacerbated by the way in which changing smoothing parameters modify that geometry (see section 4.10.3). This sort of problem is particularly common in the presence of ‘concurvity’, for example when a model includes terms such as $f_1(x_i) + f_2(z_i)$, but z is itself a smooth function of x : in this circumstance the smooth functions, f_1 and f_2 , are confounded, and there may be very little information in $\mathcal{V}_{u/g}$ from which to estimate their smoothing parameters separately: neglected dependencies in the performance iteration approach then become critical. Concurvity problems like this are very common in models which include a smooth of spatial location, and a number of covariates which themselves vary fairly smoothly over space.

A final technical problem, with performance iteration, relates to divergence of the P-IRLS scheme. Such divergence can occasionally occur with all GLM fitting by iteratively reweighted least squares, but is easily dealt with by reducing the parameter step length taken, if divergence is detected. Unfortunately divergence is not easily detected with performance iteration, because model likelihood, GCV/UBRE score and penalized likelihood may all legitimately increase as well as decrease from one iteration to the next (see e.g. steps 2 to 3 in figure 4.11).

Direct outer iteration (which was proposed in O’Sullivan et al., 1986) suffers none of these disadvantages of performance iteration, but it is more computationally costly, since the P-IRLS scheme must be iterated to convergence in order to evaluate $\mathcal{V}_{u/g}$, for each trial set of smoothing parameters. Reliable and reasonably efficient optimization, in this case, requires that at least first derivatives of $\mathcal{V}_{u/g}$ w.r.t. smoothing parameters be calculated, as part of the P-IRLS process.

In the following sections the computations required for performance and outer iteration will be explained.

4.6 Numerical GCV/UBRE: performance iteration

Estimation of smoothing parameters for an additive model, or for a generalized additive model using performance iteration, requires that we can minimize the GCV score (4.21) or UBRE score (4.16) with respect to multiple smoothing parameters, for models estimated by penalized least squares. This section explains the computational method by which this can be achieved in an efficient and stable manner. Gu and Wahba (1991) provided the first algorithm for this sort of problem, which exploited the special structure of the smoothing spline models in which they were interested. Wood (2000) provided a method, based on a similar optimization strategy, for the type of models considered here, but this section will follow the more stable approach of Wood (2004).

4.6.1 Minimizing the GCV or UBRE score

In order to fit GAMs, it will eventually be necessary to estimate smoothing parameters for weighted penalized least squares problems, possibly subject to linear equality constraints. But, as we saw in sections 1.8.1 and 1.8.4, a constrained weighted problem can always be transformed to an un-weighted, unconstrained one, so it is with such problems that this section will deal. The penalized least squares objective considered here will therefore be

$$\mathcal{S}_p = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^\top \mathbf{H}\boldsymbol{\beta} + \sum_{i=1}^m \lambda_i \boldsymbol{\beta}^\top \mathbf{S}_i \boldsymbol{\beta}, \quad (4.27)$$

and for a given $\boldsymbol{\lambda}$, this is minimized w.r.t. $\boldsymbol{\beta}$ to obtain $\hat{\boldsymbol{\beta}}$. \mathbf{H} is any positive semi-definite matrix, which may be zero, but may also be used to allow lower bounds to be imposed on the smoothing parameters, or to regularize an ill-conditioned problem. For example, if it is required that $\lambda_1 \geq 0.1$ and $\lambda_2 \geq 10$, then we could set $\mathbf{H} = 0.1\mathbf{S}_1 + 10\mathbf{S}_2$.

One slight practical modification of the GCV and UBRE scores is worth including at this stage. Sometimes the GCV or UBRE selected model is deemed to be too wiggly, and a smoother model is desired. One way to achieve this, in a systematic way, is to increase the amount that each model effective degree of freedom counts, in the GCV

or UBRE score, by a factor $\gamma \geq 1$ (see e.g. Chambers and Hastie, 1993, p. 574). The slightly modified criteria that will be considered here are therefore:

$$\mathcal{V}_g = \frac{n\|\mathbf{y} - \mathbf{Ay}\|^2}{[n - \gamma \text{tr}(\mathbf{A})]^2} \quad (4.28)$$

and

$$\mathcal{V}_u = \frac{1}{n}\|\mathbf{y} - \mathbf{Ay}\|^2 + \frac{2}{n}\sigma^2\gamma \text{tr}(\mathbf{A}) - \sigma^2. \quad (4.29)$$

The dependence of \mathcal{V}_g and \mathcal{V}_u on $\boldsymbol{\lambda}$, is through the influence matrix \mathbf{A} , which is in turn obtained via the minimization of \mathcal{S}_p .

In the multiple smoothing parameter case there is no efficient direct method for minimizing the GCV and UBRE scores: their evaluation is made numerically costly by the presence of the $\text{tr}(\mathbf{A})$ terms in both. It is therefore necessary to adopt a numerical approach based on Newton's method. That is, the score, \mathcal{V} , is approximated in the vicinity of the current best estimate of the smoothing parameters, by a quadratic function.

$$\mathcal{V}(\boldsymbol{\lambda}) \simeq \mathcal{V}(\boldsymbol{\lambda}^{[k]}) + (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{[k]})^\top \mathbf{m} + \frac{1}{2}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{[k]})^\top \mathbf{M}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{[k]})$$

where \mathbf{m} and \mathbf{M} are the first derivative vector and second derivative matrix of \mathcal{V} w.r.t. the smoothing parameters. It is easy to show that the minimum of the approximating quadratic is at

$$\boldsymbol{\lambda}^{[k+1]} = \boldsymbol{\lambda}^{[k]} - \mathbf{M}^{-1}\mathbf{m},$$

and this can be used as the next estimate of the smoothing parameters. A new approximating quadratic is then found by expansion about $\boldsymbol{\lambda}^{[k+1]}$ and this is minimized to find $\boldsymbol{\lambda}^{[k+2]}$, with the process being repeated until convergence.

In fact several things can go wrong with Newton's method, so some modifications are needed. Firstly, the method does not 'know' that the smoothing parameters must be positive, and may step to negative values, but this is easily avoided by using $\rho_i = \log(\lambda_i)$ as the optimization parameters. Secondly \mathbf{M} may not be positive definite, so that the quadratic has no unique minimum: in this case the best that can be done is to search in the steepest descent direction, $-\mathbf{m}$, for parameter values that will reduce the score. Finally the quadratic approximation may simply be very poor, so that stepping to its minimum actually increases the real \mathcal{V} : in this case it is worth trying to successively half the length of the step, from $\rho^{[k]}$, until a step is found that actually decreases \mathcal{V} ; if this fails then steepest descent can be tried. See Gill et al. (1981) for a general coverage of optimization.

Stable and efficient evaluation of the scores and derivatives

The main challenge, in implementing the numerical approach outlined above, is to be able to evaluate the GCV and UBRE scores, and their derivatives, in a manner that is both computationally efficient and stable. Stability can be an issue, as the model matrix for a complex GAM can become close to column rank deficient, which can

cause problems with the estimation of the β parameters, let alone the smoothing parameters, if care is not taken.

The expensive part of evaluating the GCV or UBRE/AIC criteria is the evaluation of the trace of the influence matrix of the problem (4.27), so it is this influence matrix that must be considered first:

$$\mathbf{A} = \mathbf{X} \left(\mathbf{X}^T \mathbf{X} + \mathbf{H} + \sum_{i=1}^m \lambda_i \mathbf{S}_i \right)^{-1} \mathbf{X}^T.$$

The process of getting this into a more useful form starts with a QR decomposition of \mathbf{X} ,

$$\mathbf{X} = \mathbf{Q}\mathbf{R},$$

where the columns of \mathbf{Q} are columns of an orthogonal matrix, and \mathbf{R} is an upper triangular matrix (see A.6). For maximum stability a pivoted QR decomposition should actually be used here (Golub and van Loan, 1996): this has the consequence that the parameter vector and rows and columns of the \mathbf{S}_i matrices have to be permuted before proceeding, with the inverse permutation applied to the parameter vector and covariance matrix at the end of the estimation procedure.

The next step is to define $\mathbf{S} = \mathbf{H} + \sum_{i=1}^m \lambda_i \mathbf{S}_i$ and \mathbf{B} as any matrix square root of \mathbf{S} such that $\mathbf{B}^T \mathbf{B} = \mathbf{S}$. \mathbf{B} can be obtained efficiently by pivoted Choleski decomposition (see ?chol in R, for example) or by eigen-decomposition of the symmetric matrix \mathbf{S} (see e.g. Golub and van Loan, 1996). Augmenting \mathbf{R} with \mathbf{B} , a singular value decomposition (Golub and van Loan, 1996, LAPACK contains a suitable version) is then obtained:

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{B} \end{bmatrix} = \mathbf{U}\mathbf{D}\mathbf{V}^T.$$

The columns of \mathbf{U} are columns of an orthogonal matrix, and \mathbf{V} is an orthogonal matrix. \mathbf{D} is the diagonal matrix of singular values: examination of these is the most reliable way of detecting numerical rank deficiency of the fitting problem (Golub and van Loan, 1996; Watkins, 1991). Rank deficiency of the fitting problem is dealt with at this stage by removing from \mathbf{D} the rows and columns containing any singular values that are ‘too small’, along with the corresponding columns of \mathbf{U} and \mathbf{V} . This has the effect of recasting the original fitting problem into a reduced space in which the model parameters are identifiable. ‘Too small’ is judged with reference to the largest singular value: for example, singular values less than the largest singular value multiplied by the square root of the machine precision might be deleted.

Now let \mathbf{U}_1 be the sub matrix of \mathbf{U} such that $\mathbf{R} = \mathbf{U}_1 \mathbf{D} \mathbf{V}^T$. This implies that $\mathbf{X} = \mathbf{Q} \mathbf{U}_1 \mathbf{D} \mathbf{V}^T$, while $\mathbf{X}^T \mathbf{X} + \mathbf{S} = \mathbf{V} \mathbf{D}^2 \mathbf{V}^T$ and consequently

$$\begin{aligned} \mathbf{A} &= \mathbf{Q} \mathbf{U}_1 \mathbf{D} \mathbf{V}^T \mathbf{V} \mathbf{D}^{-2} \mathbf{V}^T \mathbf{V} \mathbf{D} \mathbf{U}_1^T \mathbf{Q}^T \\ &= \mathbf{Q} \mathbf{U}_1 \mathbf{U}_1^T \mathbf{Q}^T. \end{aligned}$$

Hence, $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{U}_1 \mathbf{U}_1^T \mathbf{Q}^T \mathbf{Q}) = \text{tr}(\mathbf{U}_1 \mathbf{U}_1^T)$. Notice that the main computational cost is the forming the QR decomposition, but thereafter evaluation of $\text{tr}(\mathbf{A})$ is relatively cheap for new trial values of λ .

For efficient minimization of the smoothness selection criteria, we also need to find the derivatives of the criteria w.r.t. the smoothing parameters. To this end, it helps to write the influence matrix as $\mathbf{A} = \mathbf{X}\mathbf{G}^{-1}\mathbf{X}^\top$ where $\mathbf{G} = \mathbf{X}^\top\mathbf{X} + \mathbf{S} = \mathbf{V}\mathbf{D}^2\mathbf{V}^\top$ and hence $\mathbf{G}^{-1} = \mathbf{V}\mathbf{D}^{-2}\mathbf{V}^\top$. Letting $\rho_i = \log(\lambda_i)$ we then have that

$$\frac{\partial \mathbf{G}^{-1}}{\partial \rho_i} = -\mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial \rho_i} \mathbf{G}^{-1} = -\lambda_i \mathbf{V}\mathbf{D}^{-2}\mathbf{V}^\top \mathbf{S}_i \mathbf{V}\mathbf{D}^{-2}\mathbf{V}^\top$$

and so

$$\frac{\partial \mathbf{A}}{\partial \rho_i} = \mathbf{X} \frac{\partial \mathbf{G}^{-1}}{\partial \rho_i} \mathbf{X}^\top = -\lambda_i \mathbf{Q}\mathbf{U}_1 \mathbf{D}^{-1} \mathbf{V}^\top \mathbf{S}_i \mathbf{V}\mathbf{D}^{-1} \mathbf{U}_1^\top \mathbf{Q}^\top.$$

Turning to the second derivatives, we have

$$\frac{\partial^2 \mathbf{G}^{-1}}{\partial \rho_i \partial \rho_j} = \mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial \rho_j} \mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial \rho_i} \mathbf{G}^{-1} - \mathbf{G}^{-1} \frac{\partial^2 \mathbf{G}}{\partial \rho_i \partial \rho_j} \mathbf{G}^{-1} + \mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial \rho_i} \mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial \rho_j} \mathbf{G}^{-1}$$

and, of course,

$$\frac{\partial^2 \mathbf{A}}{\partial \rho_i \partial \rho_j} = \mathbf{X} \frac{\partial^2 \mathbf{G}^{-1}}{\partial \rho_i \partial \rho_j} \mathbf{X}^\top.$$

This becomes

$$\frac{\partial^2 \mathbf{A}}{\partial \rho_i \partial \rho_j} = \lambda_i \lambda_j \mathbf{Q}\mathbf{U}_1 \mathbf{D}^{-1} \mathbf{V}^\top [\mathbf{S}_j \mathbf{V}\mathbf{D}^{-2}\mathbf{V}^\top \mathbf{S}_i]^\dagger \mathbf{V}\mathbf{D}^{-1} \mathbf{U}_1^\top \mathbf{Q}^\top + \delta_j^i \frac{\partial \mathbf{A}}{\partial \rho_i}$$

where $\mathbf{B}^\dagger \equiv \mathbf{B} + \mathbf{B}^\top$ and $\delta_j^i = 1$, if $i = j$, and zero otherwise.

Writing $\alpha = \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2$, we can now find convenient expressions for the component derivatives needed in order to find the derivatives of the GCV or UBRE/AIC scores.

First define: (i) $\mathbf{y}_1 = \mathbf{U}_1^\top \mathbf{Q}^\top \mathbf{y}$; (ii) $\mathbf{M}_i = \mathbf{D}^{-1} \mathbf{V}^\top \mathbf{S}_i \mathbf{V}\mathbf{D}^{-1}$ and (iii) $\mathbf{K}_i = \mathbf{M}_i \mathbf{U}_1^\top \mathbf{U}_1$. Some tedious manipulation then shows that:

$$\text{tr} \left(\frac{\partial \mathbf{A}}{\partial \rho_i} \right) = -\lambda_i \text{tr} (\mathbf{K}_i)$$

$$\text{tr} \left(\frac{\partial^2 \mathbf{A}}{\partial \rho_i \partial \rho_j} \right) = 2\lambda_i \lambda_j \text{tr} (\mathbf{M}_j \mathbf{K}_i) - \delta_j^i \lambda_i \text{tr} (\mathbf{K}_i)$$

while

$$\frac{\partial \alpha}{\partial \rho_i} = 2\lambda_i [\mathbf{y}_1^\top \mathbf{M}_i \mathbf{y}_1 - \mathbf{y}_1^\top \mathbf{K}_i \mathbf{y}_1]$$

and

$$\frac{\partial^2 \alpha}{\partial \rho_i \partial \rho_j} = 2\lambda_i \lambda_j \mathbf{y}_1^\top [\mathbf{M}_i \mathbf{K}_j + \mathbf{M}_j \mathbf{K}_i - \mathbf{M}_i \mathbf{M}_j - \mathbf{M}_j \mathbf{M}_i + \mathbf{K}_i \mathbf{M}_j] \mathbf{y}_1 + \delta_j^i \frac{\partial \alpha}{\partial \rho_i}.$$

The above derivatives can be used to find the derivatives of \mathcal{V}_g or \mathcal{V}_u w.r.t. the ρ_i . Defining $\delta = n - \gamma \text{tr}(\mathbf{A})$, so that

$$\mathcal{V}_g = \frac{n\alpha}{\delta^2} \quad \text{and} \quad \mathcal{V}_u = \frac{1}{n}\alpha - \frac{2}{n}\delta\sigma^2 + \sigma^2,$$

then

$$\frac{\partial \mathcal{V}_g}{\partial \rho_i} = \frac{n}{\delta^2} \frac{\partial \alpha}{\partial \rho_i} - \frac{2n\alpha}{\delta^3} \frac{\partial \delta}{\partial \rho_i}$$

and

$$\frac{\partial^2 \mathcal{V}_g}{\partial \rho_i \partial \rho_j} = -\frac{2n}{\delta^3} \frac{\partial \delta}{\partial \rho_j} \frac{\partial \alpha}{\partial \rho_i} + \frac{n}{\delta^2} \frac{\partial^2 \alpha}{\partial \rho_i \partial \rho_j} - \frac{2n}{\delta^3} \frac{\partial \alpha}{\partial \rho_j} \frac{\partial \delta}{\partial \rho_i} + \frac{6n\alpha}{\delta^4} \frac{\partial \delta}{\partial \rho_j} \frac{\partial \delta}{\partial \rho_i} - \frac{2n\alpha}{\delta^3} \frac{\partial^2 \delta}{\partial \rho_i \partial \rho_j}.$$

Similarly

$$\frac{\partial \mathcal{V}_u}{\partial \rho_i} = \frac{1}{n} \frac{\partial \alpha}{\partial \rho_i} - 2 \frac{\partial \delta}{\partial \rho_i} \frac{\sigma^2}{n}$$

and

$$\frac{\partial^2 \mathcal{V}_u}{\partial \rho_i \partial \rho_j} = \frac{1}{n} \frac{\partial^2 \alpha}{\partial \rho_i \partial \rho_j} - 2 \frac{\partial^2 \delta}{\partial \rho_i \partial \rho_j} \frac{\sigma^2}{n}.$$

For each trial λ , these derivatives can be obtained at quite reasonable computational cost, so that Newton's method backed up by steepest descent can be used to find the optimum λ fairly efficiently. Given the estimated $\hat{\lambda}$, the best fit β vector is simply,

$$\hat{\beta} = \mathbf{V}\mathbf{D}^{-1}\mathbf{y}_1,$$

while the Bayesian covariance matrix for the parameters (see section 4.8) is

$$\mathbf{V}_\beta = \mathbf{V}\mathbf{D}^{-2}\mathbf{V}^\top.$$

The weighted constrained case

So far weights and constraints have been neglected. GAM estimation, by performance iteration, requires that smoothing parameters be estimated for the working linear models at each P-IRLS iteration, which are generally weighted. Similarly, in most cases, some linear constraints on a GAM are required to ensure identifiability of its smooth components, and, if these have not been absorbed into the basis as described in section 4.2, they must be imposed during fitting. Hence, in general the problems of interest will be of the form

$$\text{minimize } \|\sqrt{\mathbf{W}}(\mathbf{y} - \mathbf{X}\beta)\|^2 + \beta^\top \mathbf{H}\beta + \sum_{i=1}^m \lambda_i \beta^\top \mathbf{S}_i \beta \text{ w.r.t. } \beta \text{ subject to } \mathbf{C}\beta = \mathbf{0}$$

where \mathbf{W} typically contains the iterative weights of a P-IRLS iteration, although it could also be a general positive definite matrix^{††}, such as the inverse of the covariance matrix of the response data.

Exactly as described in section 1.8.1, the constraints can be eliminated from the problem by forming the QR decomposition of \mathbf{C}^\top , in order to find a basis for the null space of the constraints, \mathbf{Z} . Writing $\beta = \mathbf{Z}\beta_z$ ensures that the constraints are

^{††} in which case $\sqrt{\mathbf{W}}$ is any matrix square root such that $\sqrt{\mathbf{W}}^\top \sqrt{\mathbf{W}} = \mathbf{W}$.

met. Hence, letting $\tilde{\mathbf{y}} = \sqrt{\mathbf{W}}\mathbf{y}$, $\tilde{\mathbf{X}} = \sqrt{\mathbf{W}}\mathbf{X}\mathbf{Z}$, $\tilde{\mathbf{H}} = \mathbf{Z}^\top \mathbf{H}\mathbf{Z}$ and $\tilde{\mathbf{S}}_i = \mathbf{Z}^\top \mathbf{S}_i \mathbf{Z}$ the problem becomes

$$\text{minimize } \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}_z\|^2 + \boldsymbol{\beta}_z^\top \tilde{\mathbf{H}}\boldsymbol{\beta}_z + \sum_{i=1}^m \lambda_i \boldsymbol{\beta}_z^\top \tilde{\mathbf{S}}_i \boldsymbol{\beta}_z \text{ w.r.t. } \boldsymbol{\beta}_z$$

which is in exactly the un-weighted, unconstrained form, required for the smoothing parameter selection method described above.

Transforming the parameters and their variances back to the original parameter space, after fitting, is straightforward:

$$\hat{\boldsymbol{\beta}} = \mathbf{Z}\hat{\boldsymbol{\beta}}_z \text{ and } \mathbf{V}_{\boldsymbol{\beta}} = \mathbf{Z}\mathbf{V}_{\boldsymbol{\beta}_z}\mathbf{Z}^\top.$$

4.7 Numerical GCV/UBRE optimization by outer iteration

The previous section detailed an effective numerical method for use with performance iteration, or to estimate a simple additive model. In this section a numerical method for ‘outer iteration’ is presented. Our aim is to minimize the GCV score,

$$\mathcal{V}_g = \frac{nD(\hat{\boldsymbol{\mu}})}{[n - \gamma \text{tr}(\mathbf{A})]^2}, \quad (4.30)$$

or UBRE score (scaled AIC),

$$\mathcal{V}_u = \frac{1}{n}D(\hat{\boldsymbol{\mu}}) - \sigma^2 + \frac{2\gamma}{n} \text{tr}(\mathbf{A})\sigma^2, \quad (4.31)$$

with respect to the model smoothing parameters. Recall that $\hat{\mu}_i$ is a maximum penalized likelihood estimate of μ_i given smoothing parameters, and $\text{tr}(\mathbf{A})$ is the trace of the influence matrix of the working weighted linear model, at convergence of the P-IRLS scheme used to find $\hat{\mu}_i$.

The basic approach is to use a Newton type method, as outlined in section 4.6.1, to minimize (4.30) or (4.31), directly. To do this, we need to evaluate the gradient vector, and Hessian matrix, of \mathcal{V}_g or \mathcal{V}_u , which involves either iterating various derivatives alongside the P-IRLS model fitting iterations, or estimating the derivatives by finite differencing. In this section first derivatives are calculated exactly, as part of a modified P-IRLS iteration, while the Hessian is calculated by finite differencing of these first derivatives. Given exact first derivatives, Newton methods are usually very effective, even if second derivatives are quite crudely approximated, so this is a safe approach to take.

4.7.1 Differentiating the GCV/UBRE function

The derivatives of the GCV and UBRE functions, w.r.t. smoothing parameters, have similar components, so it suffices to concentrate on the GCV score. All quantities in

this section are estimates at the current $\hat{\beta}$, but to avoid clutter I will omit hats from all quantities except $\hat{\beta}$. Writing (4.30) as $\mathcal{V}_g = nD/\delta^2$, we have that

$$\frac{\partial D}{\partial \beta_j} = -2 \sum_{i=1}^n \omega_i \frac{y_i - \hat{\mu}_i}{V(\hat{\mu}_i)g'(\hat{\mu}_i)} X_{ij}$$

and, of course,

$$\frac{\partial D}{\partial \rho_k} = \sum_{j=1}^p \frac{\partial D}{\partial \beta_j} \frac{\partial \beta_j}{\partial \rho_k}.$$

where $\rho_k = \log(\lambda_k)$. The key component in this expression is $\partial \beta_j / \partial \rho_k$ which has to be calculated iteratively as part of the P-IRLS calculation. $\partial \delta / \partial \rho_k$ depends on the derivative of $\text{tr}(\mathbf{A})$ w.r.t. ρ_k , which must also be obtained.

To calculate these derivatives, the P-IRLS iteration of section 4.3 is generalized, so that one iteration step becomes:

1. Evaluate the pseudodata ,

$$z_i = \frac{\partial \eta_i}{\partial \mu_i} (y_i - \mu_i) + \eta_i,$$

and corresponding derivatives,

$$\frac{\partial z_i}{\partial \rho_k} = (y_i - \mu_i) g''(\mu_i) \frac{\partial \mu_i}{\partial \eta_i} \frac{\partial \eta_i}{\partial \rho_k}.$$

Note that $\eta_i = \mathbf{X}_i \hat{\beta}$, is the ‘linear predictor’ for the i^{th} datum.

2. Evaluate the weights,

$$w_i = [V(\mu_i)g'(\mu_i)^2]^{-1/2},$$

and derivatives,

$$\frac{\partial w_i}{\partial \rho_k} = -\frac{1}{2} w_i^3 \left[\frac{\partial V}{\partial \mu_i} \frac{\partial \mu_i}{\partial \eta_i} + 2V(\mu_i)g''(\mu_i) \right] \frac{\partial \eta_i}{\partial \rho_k}.$$

3. Drop any observations (for this iteration only) for which $w_i = 0$ or $\partial \mu_i / \partial \eta_i = 0$.

4. Find the parameters, $\hat{\beta}$, minimizing

$$\sum_i w_i^2 (z_i - \mathbf{X}_i \beta)^2 + \beta^\top \mathbf{H} \beta + \sum_k e^{\rho_i} \beta^\top \mathbf{S}_i \beta$$

and the derivative vector, $\partial \hat{\beta} / \partial \rho_k$, for the next iteration.

Notice that the iteration requires the second derivative of the link w.r.t. μ_i , and the first derivative of the variance function: these are not required by the usual P-IRLS iteration.

The penalized least squares step (4), and corresponding derivative update, need to be spelled out. This is facilitated by first transforming the pseudodata so that

$$z'_i = w_i z_i \quad \text{and} \quad \frac{\partial z'_i}{\partial \rho_k} = \frac{\partial w_i}{\partial \rho_k} z_i + w_i \frac{\partial z_i}{\partial \rho_k}.$$

Now define $\mathbf{W} = \text{diag}(\mathbf{w})$ and $\mathbf{S} = \mathbf{H} + \sum_k e^{\rho_k} \mathbf{S}_k$. Formally, we have that

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{W}^2 \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{z}' = \mathbf{B} \mathbf{z}'$$

by definition of \mathbf{B} . Similarly the influence matrix is formally:

$$\mathbf{A} = \mathbf{W} \mathbf{X} (\mathbf{X}^\top \mathbf{W}^2 \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^\top \mathbf{W}.$$

Of course these expressions are not suitable for computation. Instead, find the QR decomposition

$$\mathbf{W} \mathbf{X} = \mathbf{Q} \mathbf{R}$$

(as in the methods for performance iteration, this would usually be done with pivoting) and find a square root \mathbf{E} , such that $\mathbf{E}^\top \mathbf{E} = \mathbf{S}$. Now form the SVD,

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{E} \end{bmatrix} = \mathbf{U} \mathbf{D} \mathbf{V}^\top.$$

Again some truncation of this SVD may be performed at this stage, exactly as for the performance iteration method. Let \mathbf{U}_1 be the first p rows of \mathbf{U} , so that $\mathbf{R} = \mathbf{U}_1 \mathbf{D} \mathbf{V}^\top$. Now

$$\mathbf{B} = \mathbf{V} \mathbf{D}^{-1} \mathbf{U}_1^\top \mathbf{Q}^\top \quad (4.32)$$

and

$$\mathbf{A} = \mathbf{Q} \mathbf{U}_1 \mathbf{U}_1^\top \mathbf{Q}^\top. \quad (4.33)$$

It is also useful to define $\mathbf{G} = (\mathbf{X}^\top \mathbf{W}^2 \mathbf{X} + \mathbf{S})^{-1} = \mathbf{V} \mathbf{D}^{-2} \mathbf{V}^\top$.

$$\frac{\partial \hat{\boldsymbol{\beta}}}{\partial \rho_k} = \frac{\partial \mathbf{B}}{\partial \rho_k} \mathbf{z}' + \mathbf{B} \frac{\partial \mathbf{z}'}{\partial \rho_k}$$

and some effort then yields:

$$\frac{\partial \mathbf{B}}{\partial \rho_k} = -2 \mathbf{B} \mathbf{T}_k \mathbf{A} - e^{\rho_k} \mathbf{G}^{-1} \mathbf{S}_k \mathbf{B} + \mathbf{B} \mathbf{T}_k$$

where $T_k = \text{diag}\{\partial w_i / \partial \rho_k / w_i\}$. Similarly

$$\frac{\partial \mathbf{A}}{\partial \rho_k} = \mathbf{T}_k \mathbf{A} - 2 \mathbf{A} \mathbf{T}_k \mathbf{A} - e^{\rho_k} \mathbf{B}^\top \mathbf{S}_k \mathbf{B} + \mathbf{A} \mathbf{T}_k.$$

Actual evaluation of the trace of the derivatives of \mathbf{A} , and of the product of \mathbf{B} 's derivatives and \mathbf{z}' uses the representations of \mathbf{A} and \mathbf{B} given in (4.33) and (4.32).

In practice $\partial \hat{\boldsymbol{\beta}} / \partial \rho_k$ must be updated at each step of the P-IRLS iteration, while the slightly more expensive, $\partial \text{tr}(\mathbf{A}) / \partial \rho_k$, need only be evaluated once the iteration has converged.

Given this numerically efficient and stable method for evaluating GCV or UBRE scores, and their derivatives w.r.t. smoothing parameters, smoothing parameter estimation can proceed by Quasi-Newton methods, or by the Newton method, with finite differencing to obtain an estimate of the Hessian. It is possible to extend the derivative calculations further, in order to obtain an exact Hessian, but the resulting expressions become somewhat involved, and the benefits of an exact Hessian are nothing like as substantial as the benefits of exact first derivatives.

4.8 Distributional results

The previous few sections have shown how point estimates for the model parameters, β , and smoothing parameters, λ , can be obtained, but it is also of interest to quantify the uncertainty of those estimates. In particular it would be useful to be able to find confidence intervals for the parameters, β , and quantities derived from them, such as the estimates of the smooth terms themselves. Similarly it would be useful to be able to test whether terms were required in a model at all.

There are two approaches to uncertainty estimation. Firstly, writing $\mathbf{S} = \mathbf{H} + \sum_i \lambda_i \mathbf{S}_i$, and recalling that the parameter estimators are of the form,

$$\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y},$$

where the data or pseudodata, \mathbf{y} , have covariance matrix $\mathbf{W}^{-1} \phi$, we have that

$$\mathbf{V}_e = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{X} (\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S})^{-1} \phi$$

is the covariance matrix for the estimators $\hat{\beta}$. From the normality of \mathbf{y} , or large sample multivariate normality of $\mathbf{X}^T \mathbf{W} \mathbf{y}$, (see section 4.8.3), it follows that, approximately

$$\hat{\beta} \sim N(\mathbb{E}(\hat{\beta}), \mathbf{V}_e). \quad (4.34)$$

Generally $\mathbb{E}(\hat{\beta}) \neq \beta$, so that there are problems in using this result for calculating confidence intervals. However, if $\beta = \mathbf{0}$ then $\mathbb{E}(\hat{\beta}) = \mathbf{0}$, with the same holding approximately for some subsets of β : hence the result can be useful for testing model terms for equality to zero.

An alternative is to use a Bayesian approach to uncertainty estimation, which results in a Bayesian posterior covariance matrix for the parameters,

$$\mathbf{V}_\beta = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S})^{-1} \phi$$

and a corresponding posterior distribution for those parameters,

$$\beta \sim N(\hat{\beta}, \mathbf{V}_\beta). \quad (4.35)$$

Once again, for non normal data, posterior normality of the parameters is an approximation justified by large sample results. This latter result can be used directly to calculate credible intervals for parameters.

The remainder of this section, derives the Bayesian results, and considers the calculation of p-values from the frequentist distribution of the estimators.

4.8.1 Bayesian model, and posterior distribution of the parameters, for an additive model

Consider an additive model, for which we have selected smoothing bases and penalties, so that the model can be written as

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon, \quad \epsilon \sim N(\mathbf{0}, \mathbf{W}^{-1} \sigma^2), \quad (4.36)$$

to be fitted by minimization of the penalized least squares objective

$$\|\mathbf{W}^{1/2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\|^2 + \sum_{i=1}^m \lambda_i \boldsymbol{\beta}^\top \mathbf{S}_i \boldsymbol{\beta}. \quad (4.37)$$

where \mathbf{W} is some positive definite weight matrix. Assume that the model has already been re-parameterized to eliminate any identifiability constraints.

Following Wahba (1983) and Silverman (1985), we can recognize that, by imposing a particular penalty, we are effectively imposing some prior beliefs about the likely characteristics of the correct model. That is, the model structure allows considerably more flexibility than we believe is really likely, and we choose to penalize models that are in some sense too wiggly. It is natural to give a Bayesian structure to this approach, by specifying a prior distribution on the parameters $\boldsymbol{\beta}$.

Specifically let the (generally improper) prior for $\boldsymbol{\beta}$ be:

$$f_{\boldsymbol{\beta}}(\boldsymbol{\beta}) \propto e^{-\frac{1}{2}\boldsymbol{\beta}^\top \sum \mathbf{S}_i / \tau_i \boldsymbol{\beta}}$$

where the τ_i are parameters controlling the dispersion of the prior. The prior is appropriate since it makes explicit the fact that we believe smooth models to be more likely than wiggly ones, but it gives equal probability density to all models of equal smoothness.

From the original model specification we know that the conditional distribution of \mathbf{y} given $\boldsymbol{\beta}$ is

$$f(\mathbf{y}|\boldsymbol{\beta}) \propto e^{-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{W}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})/\sigma^2}.$$

So using Bayes rule we have:

$$\begin{aligned} f(\boldsymbol{\beta}|\mathbf{y}) &\propto e^{-\frac{1}{2}(\mathbf{y}^\top \mathbf{W}\mathbf{y}/\sigma^2 - 2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{W}\mathbf{y}/\sigma^2 + \boldsymbol{\beta}^\top (\mathbf{X}^\top \mathbf{W}\mathbf{X}/\sigma^2 + \sum \mathbf{S}_i / \tau_i)\boldsymbol{\beta})} \\ &\propto e^{-\frac{1}{2}(-2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{W}\mathbf{y}/\sigma^2 + \boldsymbol{\beta}^\top (\mathbf{X}^\top \mathbf{W}\mathbf{X}/\sigma^2 + \sum \mathbf{S}_i / \tau_i)\boldsymbol{\beta})} \end{aligned}$$

Now, if $\boldsymbol{\alpha} \sim N([\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \mathbf{X}^\top \mathbf{W}\mathbf{y}, [\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \sigma^2)$, then the probability density function for $\boldsymbol{\alpha}$ is:

$$\begin{aligned} f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) &\propto \\ &e^{-\frac{1}{2}(\boldsymbol{\alpha} - (\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i)^{-1} \mathbf{X}^\top \mathbf{W}\mathbf{y})^\top (\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i)(\boldsymbol{\alpha} - (\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i)^{-1} \mathbf{X}^\top \mathbf{W}\mathbf{y})/\sigma^2} \\ &\propto e^{-\frac{1}{2}(-2\boldsymbol{\alpha}^\top \mathbf{X}^\top \mathbf{W}\mathbf{y}/\sigma^2 + \boldsymbol{\alpha}^\top (\mathbf{X}^\top \mathbf{W}\mathbf{X}/\sigma^2 + \sum \lambda_i \mathbf{S}_i / \sigma^2)\boldsymbol{\alpha})} \end{aligned}$$

Comparing $f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$ and $f(\boldsymbol{\beta}|\mathbf{y})$, it is clear that if we choose $\tau_i = \sigma^2 / \lambda_i$ then

$$\boldsymbol{\beta}|\mathbf{y} \sim N([\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \mathbf{X}^\top \mathbf{W}\mathbf{y}, [\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \sigma^2).$$

That is

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^\top \mathbf{W}\mathbf{X} + \sum \lambda_i \mathbf{S}_i)^{-1} \sigma^2). \quad (4.38)$$

This result yields a self consistent basis for constructing Bayesian ‘confidence intervals’, or more correctly ‘credible intervals’, for any quantity derived from $\boldsymbol{\beta}$.

Clearly the problem of choosing the τ_i is equivalent to the problem of choosing the λ_i , and in practice it is usual to simply plug the GCV or UBRE estimates, $\hat{\lambda}_i$ into

(4.38). An estimate of σ^2 is also usually required and we can use (4.13) from section 4.4.1, for this purpose.

4.8.2 Structure of the prior

To some extent, the prior on β has been chosen to give the rather convenient form for the distribution of $\beta|y$, that intuition might suggest is sensible. However the structure of the prior is rather reasonable in any case. Firstly, note that the prior is equivalent to assuming that each of the components of model wigginess, $\beta^\top S_i \beta$, is an independent exponentially distributed random variable with expected value τ_i . The independence assumption is quite natural in situations in which the penalties are “non-overlapping”, for example when $\sum S_i$ is block-diagonal, as in the case for most of the GAMs considered here.

To explore the prior structure further, first define $S \equiv \sum S_i / \tau_i$. Since S will generally not be of full rank, the prior f_β is generally improper, being “almost” multivariate normal. Re-parameterizing in terms of the eigen-basis of S clarifies this. Let $S = UDU^\top$ where the columns of U are the eigenvectors of S , and the diagonal matrix D has the eigenvalues of S arranged in order of decreasing magnitude on its leading diagonal. The model fitting problem, and Bayesian analysis, are invariant^{‡‡} to re-parameterization in terms of $\beta_u = U^\top \beta$.

Given this basis change $\beta^\top S \beta \equiv \beta_u^\top D \beta_u$, but the rank deficiency of S generally means that the lower right portion of D is zero. Hence the last few elements of β_u are completely un-penalized. Accordingly β_u can be partitioned into parts corresponding to the wiggly and smooth components of the model: $\beta_u^\top = [\beta_w^\top, \beta_s^\top]^\top$. Now, given the prior for β , the prior for β_u is:

$$f_{\beta_u}(\beta_u) \propto e^{-\frac{1}{2}\beta_u^\top D \beta_u / \sigma^2}$$

but if D^+ is the largest sub matrix of D having strictly positive elements on its leading diagonal, then

$$f_{\beta_u}(\beta_u) \propto e^{-\frac{1}{2}\beta_w^\top D^+ \beta_w / \sigma^2}.$$

i.e. the prior is a multivariate normal for the non-smooth component of the model (as measured by the penalty), multiplied by a completely uninformative improper prior on the parameters corresponding to the components of the model that are of zero wigginess, according to the penalty.

4.8.3 Posterior distribution for a GAM

Now consider a GAM, so that the model becomes

$$g(\mu_i) = \mathbf{X}_i \beta, \quad \mu_i \equiv \mathbb{E}(Y_i), \quad Y_i \sim \text{exponential family}, \quad (4.39)$$

^{‡‡} The invariance is by orthogonality of U : it may readily be verified, for example, that the Bayesian covariance matrix for β_u is $(U^\top X^\top W X U + \sigma^2 D)^{-1} \sigma^2$ corresponding to the Bayesian covariance matrix for β already given.

where g is a known link function, and it is estimated by minimization of

$$-l(\beta) + \frac{1}{2} \sum_i^m \lambda_i \beta^\top S_i \beta, \quad (4.40)$$

with respect to β . $l(\beta)$ is the log likelihood of the model.

As we have seen repeatedly (4.40) is minimized by iteratively solving the problem,

$$\text{minimise } \left\| \sqrt{\mathbf{W}^{[k]}} (\mathbf{X}\beta - \mathbf{z}^{[k]}) \right\|^2 + \sum_{i=1}^m \lambda_i \beta^\top S_i \beta \text{ w.r.t. } \beta,$$

where k is the iteration index,

$$\mathbf{z}^{[k]} = \mathbf{X}\beta^{[k]} + \mathbf{G}^{[k]}(\mathbf{y} - \boldsymbol{\mu}^{[k]}),$$

$\mu_i^{[k]}$ is the current model estimate of $\mathbb{E}(Y_i)$, $\mathbf{G}^{[k]}$ is a diagonal matrix such that $G_{ii}^{[k]} = g'(\mu_i^{[k]})$, and \mathbf{W} is a diagonal weight matrix where

$$W_{ii} = \left[G_{ii}^{[k]2} V \left(\mu_i^{[k]} \right) \right]^{-1}.$$

$V(\mu_i)$ gives the variance of Y_i to within a scale parameter. The penalized least squares problem can be viewed as a quadratic approximation to the penalized likelihood in the vicinity of the true parameters.

The iterative least squares approach suggests working in terms of the random vector

$$\mathbf{z} = \mathbf{X}\beta + \mathbf{G}(\mathbf{y} - \boldsymbol{\mu}).$$

Then, $\mathbb{E}(\mathbf{z}|\beta) = \mathbf{X}\beta$, and the covariance matrix of $\mathbf{z}|\beta$ is $\mathbf{W}^{-1}\phi$ (where ϕ is the scale parameter). Defining $\mathbf{v} = \mathbf{X}^\top \mathbf{W} \mathbf{z}$, it follows that $\mathbb{E}(\mathbf{v}|\beta) = \mathbf{X}^\top \mathbf{W} \mathbf{X}\beta$ and the covariance matrix of $\mathbf{v}|\beta$ is $\mathbf{X}^\top \mathbf{W} \mathbf{X}\phi$. It can further be shown that as sample size, n , tends to infinity the distribution of $\mathbf{v}|\beta$ tends to the multivariate normal

$$N(\mathbf{X}^\top \mathbf{W} \mathbf{X}\beta, \mathbf{X}^\top \mathbf{W} \mathbf{X}\phi). \quad (4.41)$$

This last result follows from the Central Limit Theorem (Lindeberg, 1922) and the fact that a random vector \mathbf{v} has a multivariate normal distribution if and only if $\mathbf{c}^\top \mathbf{v}$ has a univariate normal distribution, for any vector of constants \mathbf{c} .

Specifically, if \mathbf{c} is any vector of constants then, subject to some conditions, the distribution of

$$\mathbf{c}^\top \mathbf{v} = \mathbf{c}^\top \mathbf{X}^\top \mathbf{W} \mathbf{z}$$

tends to normality as $n \rightarrow \infty$, by the Central Limit Theorem of Lindeberg (1922). Hence the distribution of \mathbf{v} tends to a multivariate normal distribution as $n \rightarrow \infty$.

Before using (4.41) it is important to examine the conditions for validity of the approximation, which follow from Lindeberg's conditions for the validity of the CLT (see e.g Feller, 1957). First let,

$$a_i = \sum_j c_j X_{ij} W_i,$$

so that,

$$\mathbf{c}^\top \mathbf{v} = \sum_{i=1}^n a_i z_i.$$

Defining

$$s_n^2 = \sum_{i=1}^n a_i^2 \phi / w_i$$

and

$$U_i = \begin{cases} a_i z_i - a_i \mu_i & \text{if } |a_i z_i - a_i - \mu_i| \leq \epsilon s_n \\ 0 & \text{if } |a_i z_i - a_i - \mu_i| > \epsilon s_n \end{cases}$$

for all ϵ , Lindeberg's conditions are that:

$$\frac{1}{s_n^2} \sum \mathbb{E}(U_i^2) \rightarrow 1$$

and $s_n \rightarrow \infty$ as $n \rightarrow \infty$. In the current context $s_n \rightarrow \infty$ is a given, and provided that the a_i are bounded, then the first condition will be satisfied if $\Pr[U_i = 0] \rightarrow 0$ as $n \rightarrow \infty$. The elements of \mathbf{c} do not change as the limit is taken, so boundedness of the a_i is given by boundedness of the $X_{ij} W_i$. Hence the condition will be met if:

$$\lim_{n \rightarrow \infty} \Pr(|a_i z_i - a_i \mu_i| > \epsilon \sum_i a_i^2 \phi / w_i) = 0 \quad \forall \epsilon.$$

By Chebyshev's inequality

$$\Pr(|a_i z_i - a_i \mu_i| > \epsilon \sum_i a_i^2 \phi / w_i) < \frac{a_i^2 \phi / w_i}{[\sum_i a_i^2 \phi / w_i]^2 \epsilon^2}$$

so it is sufficient that

$$\frac{a_i^2 \phi / w_i}{[\sum_i a_i^2 \phi / w_i]^2} \rightarrow 0$$

as $n \rightarrow \infty$ for the result to hold. Since \mathbf{c} does no more than form weighted sums over columns of \mathbf{X} , a sufficient condition for the last condition to hold is that as $n \rightarrow \infty$:

$$\frac{(X_{ij} W_i^{1/2})^2}{\left[\sum_i (X_{ij} W_i^{1/2})^2 \right]^2} \rightarrow 0 \quad \forall i, j.$$

This last condition is actually interpretable, and says that no element of \mathbf{X} should dominate the fit, as the sample size tends to infinity: it is a fairly mild condition in most penalized regression settings.

Having established its validity, we can now use the large sample approximation (4.41), for the distribution of $\mathbf{v}|\beta$, and proceed as in section 4.8.1, to obtain the posterior for β :

$$\beta | \mathbf{v} \sim N([\mathbf{X}^\top \mathbf{W} \mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \mathbf{v}, [\mathbf{X}^\top \mathbf{W} \mathbf{X} + \sum \lambda_i \mathbf{S}_i]^{-1} \phi), \quad (4.42)$$

which is (4.38). Plugging in the \mathbf{v} and \mathbf{W} estimates at convergence of the P-IRLS algorithm, along with the estimated smoothing parameters, and if necessary an estimate

for ϕ , these results can be used to set approximate Bayesian confidence intervals for any quantity derived from β . For many exponential family distributions the scale parameter ϕ is known, but if an estimate is needed then $\hat{\phi} = \|\mathbf{W}^{1/2}(\mathbf{y} - \hat{\mu})\|^2 / \text{tr}(\mathbf{I} - \mathbf{A})$ can be used, where \mathbf{A} is the influence (or hat) matrix for the model.

4.8.4 Bayesian confidence intervals for non-linear functions of parameters

Given the results of the previous sections, it is straightforward to find confidence intervals for linear functions of the model parameters, including the component functions of a GAM, for example. Bayesian confidence intervals for non-linear functions of the parameters can also be obtained by simulation from the posterior distribution of β . Specifically if $G(\beta)$ is the function of interest, then the approximate posterior cumulative distribution function $\hat{F}(g)$ for G can be obtained by simulating a set of random vectors, $\{\beta_i^* : i = 1, \dots, N\}$, from the multivariate normal posterior for β so that:

$$\hat{F}(g) = \frac{1}{N} \sum_{i=1}^N H(g - G(\beta_i^*))$$

where H is the Heaviside function (jumping from 0 to 1 at g). Bayesian ‘confidence intervals’ are obtained from the quantiles of this distribution in the obvious manner, and are of considerable practical use. Previous approaches to finding confidence intervals for $G(\beta)$ have used bootstrapping (e.g. Borchers et al., 1997; Augustin et al., 1998): but in the usual case, in which evaluation of G is very much cheaper than model fitting, the cost of the intervals proposed here will be about the same as the cost of performing one bootstrap replicate.

4.8.5 P-values

Now consider the problem of testing whether some subset, β_j , of β is identically zero. In a GAM context, if β_j contains the coefficients for a single smooth term, then $\mathbb{E}(\hat{\beta}_j) \approx \mathbf{0}$ if $\beta_j = \mathbf{0}$. In fact if the covariates of the smooth are uncorrelated with other smooth terms in the model then $\mathbb{E}(\hat{\beta}_j) = \mathbf{0}$, but otherwise there is a (usually) weak dependence of $\mathbb{E}(\hat{\beta}_j)$ on the bias in the other penalized model terms.

Extracting the frequentist covariance matrix of $\hat{\beta}_j$, $\mathbf{V}_{\hat{\beta}_j}$, from \mathbf{V}_e (see (4.34)), we have that under the null hypothesis $\beta_j = \mathbf{0}$ and so,

$$\hat{\beta}_j \sim N(\mathbf{0}, \mathbf{V}_{\hat{\beta}_j}).$$

From this it follows that, if $\mathbf{V}_{\hat{\beta}_j}$ is of full rank, then under the null hypothesis

$$\hat{\beta}_j^T \mathbf{V}_{\hat{\beta}_j}^{-1} \hat{\beta}_j \sim \chi_d^2,$$

where $d = \dim(\beta_j)$. In fact the action of the penalty often suppresses some dimensions of the parameter space, so that $\mathbf{V}_{\hat{\beta}_j}$ is not of full rank (for example if a cubic

spline term, subject to a centering constraint, is heavily penalized it becomes effectively a straight line, with only one degree of freedom). If $r = \text{rank}(\mathbf{V}_{\hat{\beta}_j})$ and $\mathbf{V}_{\hat{\beta}_j}^{r-}$ is the rank r pseudoinverse of the covariance matrix, then testing is performed using the result that under the null,

$$\hat{\beta}_j^T \mathbf{V}_{\hat{\beta}_j}^{r-} \hat{\beta}_j \sim \chi_r^2.$$

Specifically the p-value for the test that $\beta_j = \mathbf{0}$ is $\Pr[X > \hat{\beta}_j^T \mathbf{V}_{\hat{\beta}_j}^{r-} \hat{\beta}_j]$ where $X \sim \chi_r^2$. r is usually determined numerically, while forming the pseudoinverse of the covariance matrix.

If \mathbf{V}_e , and hence $\mathbf{V}_{\hat{\beta}_j}$, contain an unknown scale parameter, ϕ , then it is usually better to base p-value calculations on the approximate result,

$$\frac{\hat{\beta}_j^T \mathbf{V}_{\hat{\beta}_j}^{r-} \hat{\beta}_j / r}{\hat{\phi} / (n - \text{edf})} \sim F_{r, \text{edf}},$$

where ‘edf’ denotes the estimated degrees of freedom for the model (the denominator here is effectively the GCV score).

The p-values, calculated in this manner, behave correctly for un-penalized models, or models with known smoothing parameters, but when smoothing parameters have been estimated, the p-values are typically lower than they should be, meaning that the tests reject the null too readily. This is because smoothing parameter uncertainty has been neglected in the reference distributions used for testing. As a result these distributions are typically too narrow, so that they ascribe too low a probability to moderately high values of the test statistics. Limited simulation experience suggests that the p-values can be as little as half the correct value at around the 5% level, when the null is true, although they may be more accurate than this in other circumstances. Note that this problem is in no way unique to GAMs. If you perform model selection on any model, and then hypothesis test using the selected model, the p-values associated with model terms will not be strictly correct, since they neglect model selection uncertainty. The advantage in performing model selection before hypothesis testing is that the elimination of unnecessary model degrees of freedom increases power.

In practical terms, if these p-values suggest that a term is not needed in a model, then this is probably true, but if a term is deemed ‘significant’ it is important to be aware that this significance may be overstated. If hypothesis testing is a key aim of an analysis, then it may sometimes be preferable to base tests on overspecified unpenalized models, so that although the fits may be excessively wiggly, the p-values will be correct: the price to pay will be some loss of power. If adopting this approach, it is quite important to avoid allowing the smooths excessively flexibility, otherwise tests will have low power. Hence the choice of basis dimension for a smooth becomes quite important: it needs to be large enough that the model structure can include a reasonable approximation to the truth, but small enough to avoid loss of power. See question 11 in Chapter 5 for further discussion.

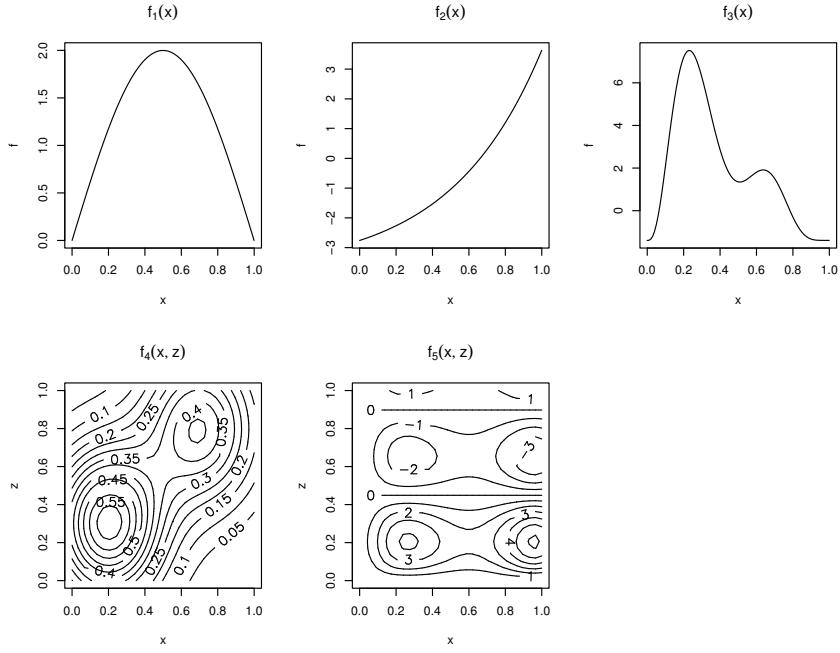


Figure 4.12 *The five test functions used in the simulation study reported in section 4.9.1.*

4.9 Confidence interval performance

Confidence intervals based on section 4.8 rely on large sample results to deal with non-Gaussian distributions, and treat the smoothing parameters as fixed, when in reality they are estimated from the data. It is not clear how these assumptions will affect the coverage probabilities of the intervals, so it is worth looking at simulation evidence. This section first examines coverage probabilities of single smooths and then GAMs, via simulation.

4.9.1 Single smooths

This section examines the performance of the intervals for models involving only single smooth terms, examining Gaussian and non-Gaussian cases. Five test functions, plotted in figure 4.12, were employed:

$$\begin{aligned}
 f_1(x) &= 2 \sin(\pi x), & f_2(x) &= e^{2x} - 3.75, \\
 f_3(x) &= x^{11}(10[1-x])^6 + 10(10x)^3(1-x)^{10} - 1.4, \\
 f_4(x, z) &= \pi \sigma_x \sigma_z \left[1.2e^{-(x-0.2)^2/\sigma_x^2 - (z-0.3)^2} + 0.8e^{-(x-0.7)^2/\sigma_x^2 - (z-0.8)^2/\sigma_z^2} \right], \\
 f_5(x, z) &= 1.9 [1.45 + e^x \sin(13[x-0.6]^2)] e^{-z} \sin(7z).
 \end{aligned}$$

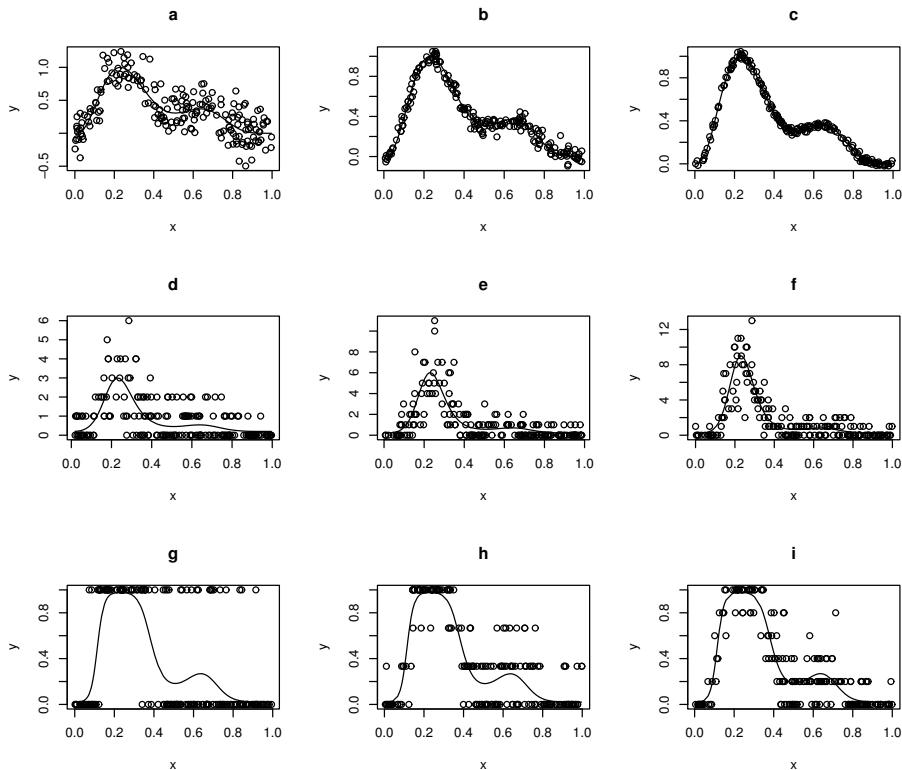


Figure 4.13 Illustration of the 3 noise levels employed for each of the 3 error families examined. (a) Gaussian, $\sigma = 0.2$. (b) Gaussian, $\sigma = 0.05$. (c) Gaussian, $\sigma = 0.02$. (d) Poisson, $p_{max} = 3$. (e) Poisson, $p_{max} = 6$. (f) Poisson, $p_{max} = 9$. (g) Binomial $n_{bin} = 1$. (h) Binomial $n_{bin} = 3$. (i) Binomial $n_{bin} = 5$.

For each function, data were simulated from 3 error models (normal, Poisson and binomial) at each of 3 signal to noise ratios, at each of two sample sizes ($n = 200$ or $n = 500$). In each case, covariates were simulated from a uniform distribution on the unit interval or unit square. The functions were linearly scaled, as detailed below, and each was then applied to the simulated covariate to give the ‘true’ linear predictor for each function. The inverse of the canonical link for the distribution was applied to the linear predictor, to obtain the true response means, and data were simulated from the appropriate distribution, with that mean.

In the normal case the functions were scaled to have range [0,1] and normal random deviates with one of three noise levels ($\sigma = 0.02, 0.05$ or 0.2) were then added to the true expected values. In the Poisson case the functions were scaled so that the true means lay in $[0.2, p_{max}]$ where p_{max} was one of 3, 6 or 9, and Poisson deviates with the required mean were simulated. In the binomial case the functions were scaled so

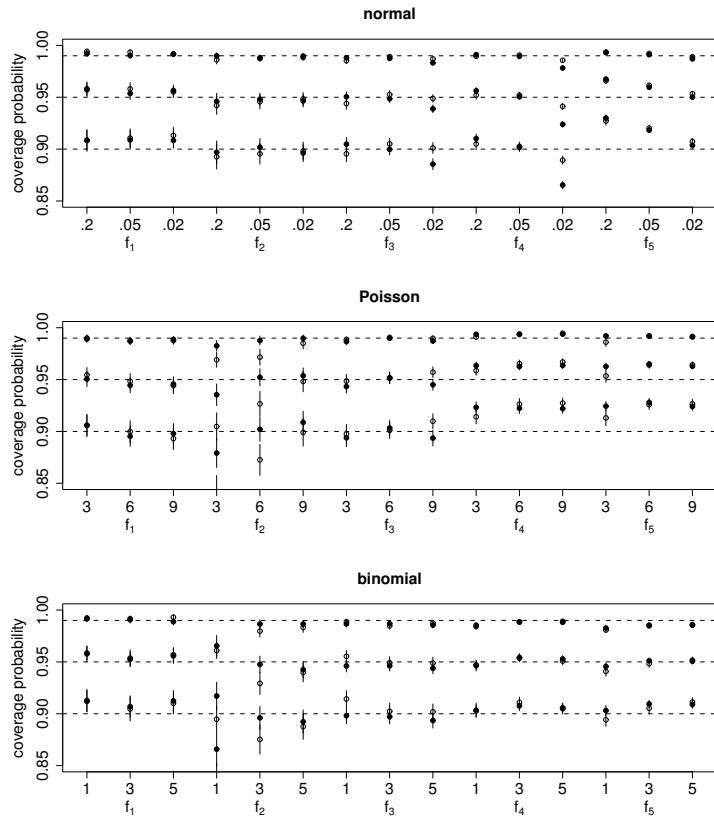


Figure 4.14 Results from the confidence interval performance simulation study detailed in section 4.9.1. The three figures show results for the three noise models. Results are shown for each function at each noise level (see text for explanation of numbers). \circ shows average realized coverage probabilities for sample sizes of 200, while \bullet is the equivalent for $n = 500$. Dashed horizontal lines show the nominal coverage probabilities for the intervals examined, and vertical lines show ± 2 standard error bands for the realized coverage probabilities.

that the binomial probabilities lay in the range [0.02, 0.98], and data were simulated from binomial distributions with denominator n_{bin} of 1, 3 or 5. Figure 4.13 shows data simulated in this way for f_3 , with each noise level shown for each distribution.

500 replicate data sets were generated for each function at each combination of sample size, distribution and noise level. Each replicate data set was modelled using a thin plate regression spline, fitted by penalized likelihood maximization, with smoothing parameter chosen by GCV in the normal case and UBRE via performance iteration otherwise. TPRS basis dimensions of 10, 10, 20, 40 and 100 were used for functions 1 to 5 respectively: these values were chosen to be as small as possible subject to the constraint that inability of the basis to fit the underlying truth should have a

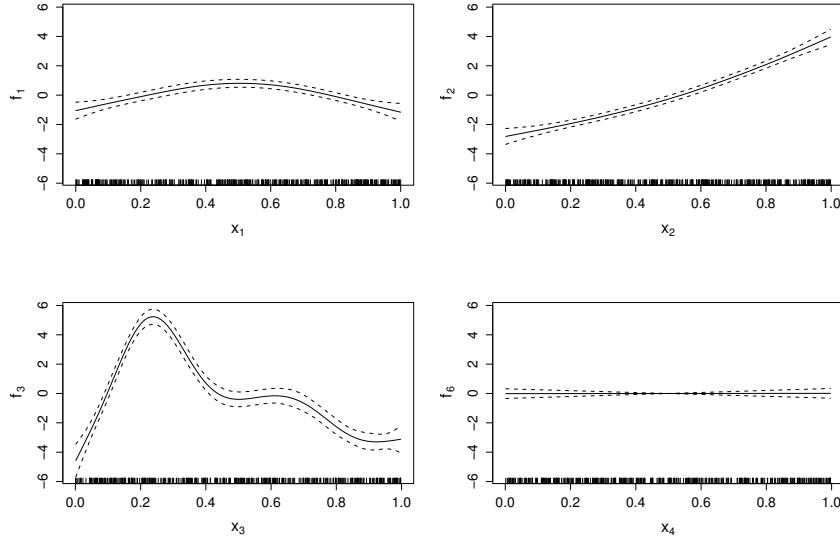


Figure 4.15 *Example component wise Bayesian confidence intervals for an additive model in which the linear predictor is the sum of (unscaled) functions 1, 2, 3 and 6 (see section 4.9.2) applied to independent covariates simulated from a uniform distribution. The gaussian noise standard deviation was 2 and the sample size was 400. Solid curves are the function estimates, and dashed curves delimit the 95 % confidence regions (credible regions) for each function. Smoothing parameter selection was by GCV.*

negligible affect on the realized coverage probabilities. For each replicate, coverage proportions were obtained for 90%, 95% and 99% confidence intervals for the functions evaluated at the covariate values. From the resulting 500 ‘across the function’ coverage proportions, an overall mean coverage probability, and its standard error, were calculated.

Figure 4.14 shows the results. Clearly, for these single functions the confidence intervals perform well: the realized coverage probabilities are usually close to the nominal. The exception is instructive: f_2 sometimes shows low coverage, and this seems to relate to a tendency to wrongly select a straight line model, for this function, when the signal to noise ratio is low.

4.9.2 GAMs and their components

A further study was conducted to investigate the effectiveness of these confidence intervals in a GAM setting. Two sets of simulations were carried out. In each, n independent values were simulated from $U(0, 1)$ to simulate 4 independent covariates. In the first example, the linear predictor was made up of a sum of linearly scaled ver-

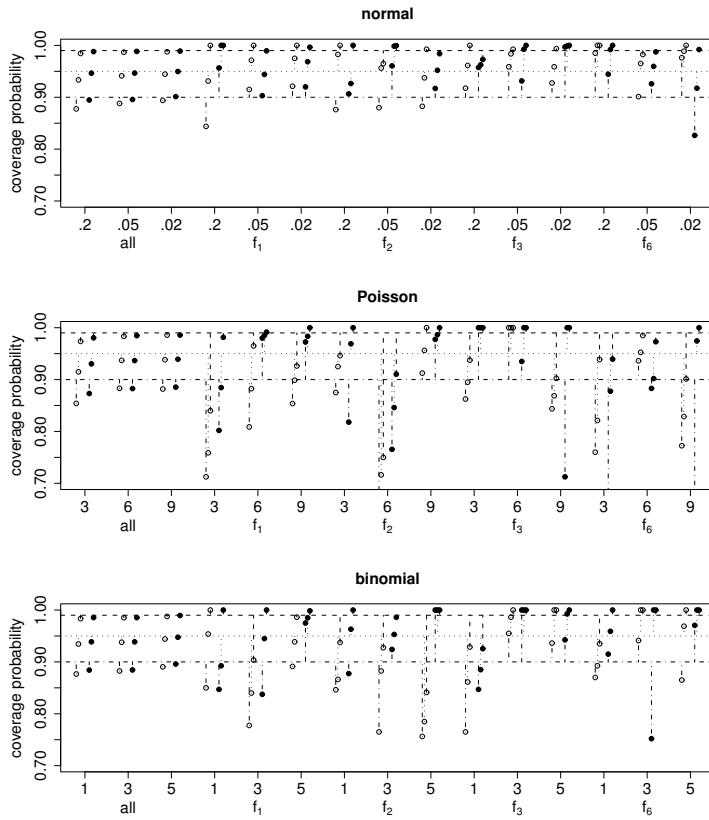


Figure 4.16 *Results from the first GAM confidence interval performance simulation study detailed in section 4.9.2. The three figures show results for the three noise models. Results are shown for the whole model (left) and each component function at each noise level (see text for explanation of numbers). \circ shows average realized coverage probabilities for sample sizes of 200, while \bullet is the equivalent for $n = 500$. Each realized coverage probability is joined to its corresponding nominal probability by a vertical line. Dashed horizontal lines show the nominal coverage probabilities for the intervals examined.*

sions of functions 1, 2 and 3, from section 4.9.1, plus the null function 0 (referred to as f_6 below), applied to these simulated covariate values. Figure 4.15 shows component wise intervals calculated from the fit to one such replicate (but without function scaling in the data simulation).

Except for f_6 , the functions were scaled to have the same range before being summed. This linear predictor was then treated in exactly the same manner as in the univariate cases of section 4.9.1, so that the noise levels have broadly the same interpretation as before. The second set of simulations was carried out in a similar manner, but with true linear predictor given by a sum of scaled versions of functions 1, 3 and 5.

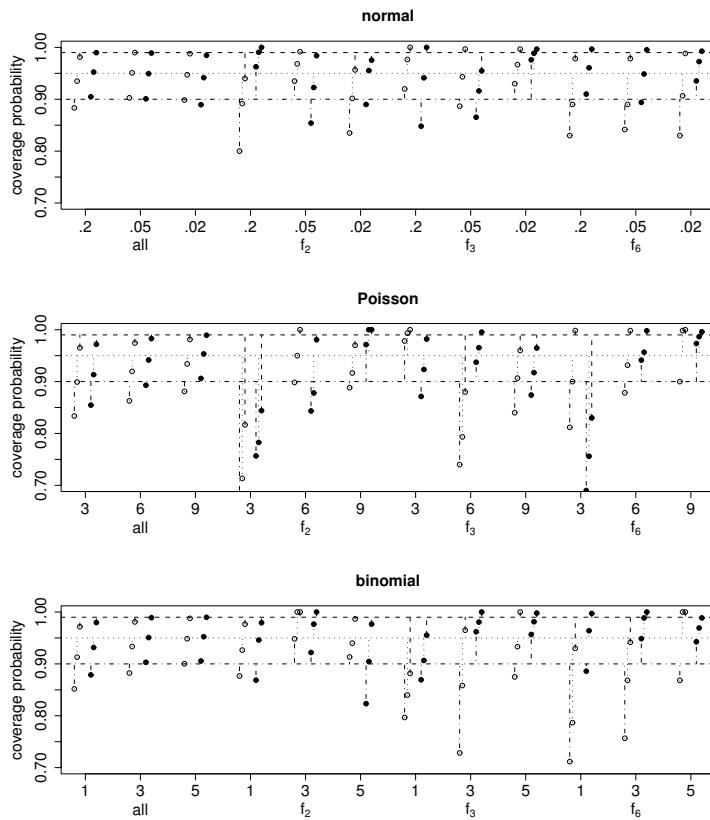


Figure 4.17 As figure 4.16, but for the second GAM simulation study detailed in section 4.9.2.

GAMs were fitted to each of 500 replicates at each sample size ($n = 200$ or 500), model structure, distribution and error level combination. The model terms were represented using penalized thin plate regression splines, and the models were fitted by penalized likelihood maximization with smoothing parameters selected by GCV in the normal cases and UBRE via performance iteration in the Poisson and binomial cases. Confidence intervals for the linear predictor, at the simulated covariate values, were obtained, along with confidence intervals for each component function, evaluated at the simulated values of its covariate argument(s). 99%, 95% and 90% intervals were calculated in all cases. The proportion of these intervals including the truth was calculated for each replicate (i.e. the assessment was made ‘across the function’), and was eventually averaged across all replicates.

Figure 4.16 summarizes the results from the first model structure. The results demonstrate that while the overall coverage for the whole model is reasonably close to nominal, the component wise coverages are quite unreliable. This pattern is confirmed by

the results for the second model structure shown in figure 4.17, where again the coverage probabilities, realized across the model, for the expected values of the response are close to nominal, while the component wise coverages are poor.

In summary: confidence intervals for the ‘whole model’ appear to be reliable, while component-wise intervals can only be used as a rough guide to the uncertainty of the components. A likely explanation for this problem is that the intervals are conditional on the smoothing parameters. Because it is relatively easy to get the overall amount of smoothing right, whole model intervals behave well. However, getting smoothing parameters for individual components right is more difficult, and this may be what causes the poor performance of component wise intervals.

4.9.3 Unconditional Bayesian confidence intervals

One possibility for improving the performance of the intervals is to extend the Bayesian treatment further, and base intervals on the joint posterior density,

$$f(\boldsymbol{\beta}, \hat{\boldsymbol{\lambda}}|\mathbf{y}) = f(\boldsymbol{\beta}|\hat{\boldsymbol{\lambda}}, \mathbf{y})f(\hat{\boldsymbol{\lambda}}|\mathbf{y}).$$

By accounting for smoothing parameter uncertainty, the performance of the component-wise intervals should be improved. Provided that we are only interested in obtaining confidence intervals for quantities that are functions of $\boldsymbol{\beta}$, but not $\boldsymbol{\lambda}$, there is no difficulty in working in terms of $\hat{\boldsymbol{\lambda}}$ rather than $\boldsymbol{\lambda}$ itself. However, $f(\hat{\boldsymbol{\lambda}}|\mathbf{y})$ is unknown. To obtain this distribution would probably require a fully Bayesian treatment, in which priors were also specified for the τ_i (or λ_i), but then one might as well explore the full posterior distribution $f(\boldsymbol{\beta}, \boldsymbol{\lambda}|\mathbf{y})$. While possible, (see e.g. Fahrmeir and Lang, 2001) this is a computer intensive undertaking, requiring MCMC simulations that are substantially less routine to use than the penalized regression methods employed here.

A pragmatic alternative is to replace $f(\hat{\boldsymbol{\lambda}}|\mathbf{y})$ by a parametric bootstrap approximation to the sampling distribution of $\hat{\boldsymbol{\lambda}}$, $f_{\hat{\boldsymbol{\lambda}}}(\hat{\boldsymbol{\lambda}})$, say, so that

$$f(\boldsymbol{\beta}, \hat{\boldsymbol{\lambda}}|\mathbf{y}) \approx f(\boldsymbol{\beta}|\hat{\boldsymbol{\lambda}}, \mathbf{y})f_{\hat{\boldsymbol{\lambda}}}(\hat{\boldsymbol{\lambda}}).$$

If this is done, then the following practical algorithm can be used:

1. Fit the penalized regression model to response data \mathbf{y} , selecting $\boldsymbol{\lambda}$ by GCV or similar, to obtain estimates $\hat{\boldsymbol{\lambda}}^{[1]}$. Let the corresponding parameter estimates be $\hat{\boldsymbol{\beta}}^{[1]}$ and the estimated parameter covariance matrix be $\hat{\mathbf{V}}_{\boldsymbol{\beta}}^{[1]}$.
2. Repeat steps 3 to 5 for $k = 2 \dots N_b$.
3. Based on the fitted model from step 1, simulate a parametric bootstrap response vector $\mathbf{y}^{[k]}$. That is, simulate random deviates with the appropriate response distribution, and mean given by the fitted values from step 1.
4. Fit the penalized regression model to the bootstrap data, $\mathbf{y}^{[k]}$, to obtain a bootstrap estimate of the smoothing parameters, $\hat{\boldsymbol{\lambda}}^{[k]}$.

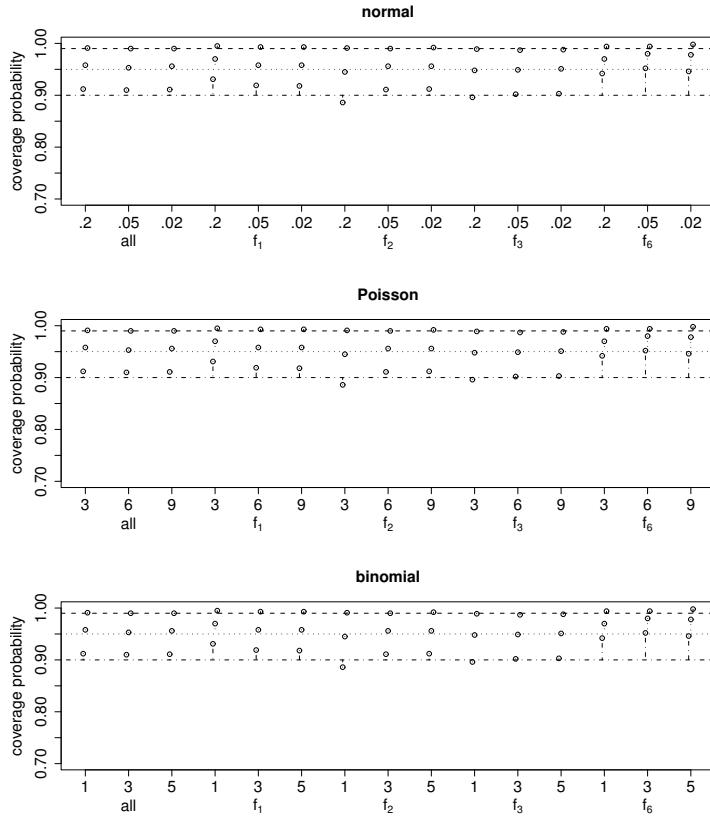


Figure 4.18 Coverage probability results for the first GAM from section 4.9.2, for a sample size of 200, but now using the unconditional intervals of section 4.9.3. Details are described in the caption of figure 4.16, to which this figure should be compared. Notice the huge improvement in the performance of the component wise intervals, when smoothing parameter uncertainty is accounted for.

5. Fit the penalized regression model to \mathbf{y} , using the smoothing parameters $\hat{\lambda}^{[k]}$, to obtain parameter estimates and estimated covariance matrix, $\hat{\beta}^{[k]}$ and $\hat{\mathbf{V}}_{\beta}^{[k]}$.
6. To simulate, approximately, from the posterior $f(\boldsymbol{\beta}, \hat{\lambda} | \mathbf{z})$, generate a random integer, j , from a discrete uniform distribution on $\{1, \dots, N_b\}$ and simulate a random $\boldsymbol{\beta}$ from $N(\boldsymbol{\beta}^{[j]}, \hat{\mathbf{V}}_{\beta}^{[j]})$.

Given steps 1 to 5, repeated simulations using step 6 can be used to find approximate Bayesian confidence intervals for any function of the parameters, $\boldsymbol{\beta}$. The method offers substantial advantages over direct bootstrapping to find confidence intervals. Since bootstrapping is only used to approximate the distribution of the smoothing pa-

rameters, there is no need to worry about the smoothing induced bias in the bootstrap $\hat{\beta}$, since they are not going to be used to obtain confidence intervals.

Similarly, since we are typically only interested in confidence intervals on quantities that are a function of β , and not $\hat{\lambda}$, a rather small N_b will usually be tolerable, offering substantial computational savings, relative to a pure bootstrapping method (simulation from $f(\beta|\hat{\lambda}, \mathbf{y})$ is very cheap computationally, at least once the square root of the covariance matrix has been evaluated for a given $\hat{\lambda}$).

Figure 4.18 is the equivalent to figure 4.16, but with confidence intervals calculated by the above method, using $N_b = 20$. Notice that the intervals now show a tendency to over cover a little, but that the component wise confidence intervals are substantially improved. The only very poor result is for the nuisance function f_6 , where the intervals over cover at all confidence levels. This is likely to result from the fact that the true smoothing parameter is infinite for this function. Hence the only error that can be made is to under-smooth and therefore produce intervals that over-cover.

4.10 Further GAM theory

This section covers some further results that are useful for applied modelling with GAMs, as well as some material that is just interesting.

4.10.1 Comparing GAMs by hypothesis testing

Sometimes it is desirable to test a null hypothesis that the simpler of two nested GAMs is correct, against the alternative that the larger of the two is correct. As in the GLM case discussed in section 2.1.6, it is sensible to proceed by using the log-likelihood ratio (i.e. difference in deviance between the two models) as a test statistic, where the likelihoods are now evaluated at the maximum penalized likelihood estimates. If $\hat{\eta}_0$ is the estimated linear predictor for the null model, and $\hat{\eta}_1$ is the equivalent for the alternative, then the test statistic is

$$\lambda = 2[l(\hat{\eta}_1) - l(\hat{\eta}_0)].$$

Unfortunately the sampling distribution of λ , under the null, is unknown, and we are forced back onto rather crude approximations.

In particular, if we condition on the smoothing parameters (i.e. treat them as known, rather than estimated) then a rough approximation is that under the null

$$\lambda \sim \chi^2_{EDF_1 - EDF_0}. \quad (4.43)$$

i.e. that λ follows a χ^2 distribution with degrees of freedom given by the difference in effective degrees of freedom between the two models.

This approximation is most easily justified for penalized regression spline smooths.

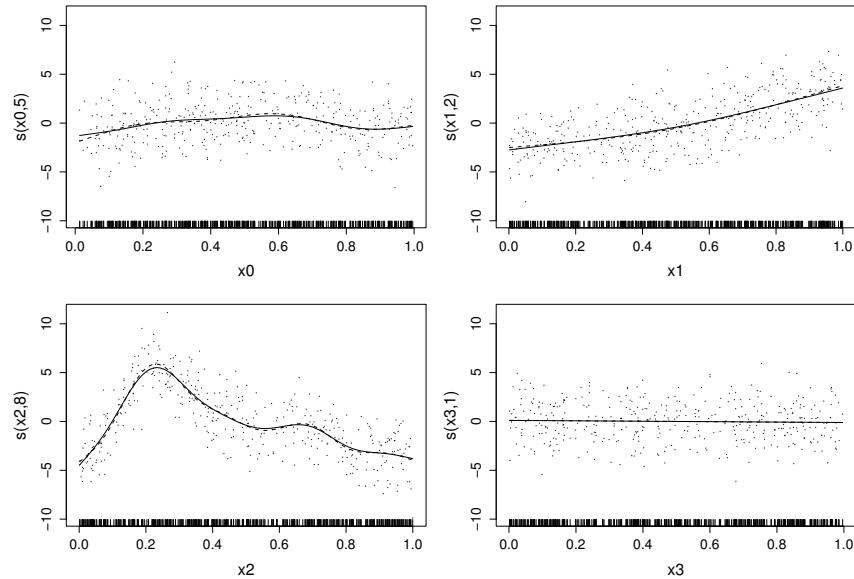


Figure 4.19 *Illustration of the similarity between the estimated terms of a GAM represented using penalized regression splines (solid curves) and pure regression splines (dashed curves — based on thin plate regression splines). Both models are fitted to the same data, and the effective degrees of freedom of each penalized term matches the degrees of freedom of its pure regression equivalent. Partial residuals are shown to give an indication of the scale of variability in the data. In this example the mean square difference between the fitted values from the two models is 1.2% of the residual variance. The correlation between the fitted values from the two models is 0.998.*

As we saw in section 4.1.5, it is possible to come up with an optimal rank k approximation to a full spline smooth for any k (above some detail dependent technical minimum). Such approximations perform rather well as pure regression splines. Hence for any penalized regression spline model, with given termwise effective degrees of freedom, there is a very similar model based on pure regression splines, with similar true termwise degrees of freedom. i.e for any set of data, the two models will produce very similar estimated linear predictors, and similar estimated smooth terms. Figure 4.19 illustrates the similarity between fits for a 4 term GAM, represented using penalized regression splines and pure regression splines.

Hence any comparison of two GAMs represented using penalized regression splines, can be approximated by the comparison of two GAMs represented using pure regression splines. However GAMs represented using pure regression splines are simply GLMs, and for this case the distribution of λ under the null is well approximated by $\chi^2_{p_1-p_0}$, where p_0 and p_1 are the numbers of parameters for the null and alternative models, as we saw in sections 2.1.6 and 2.4.6. However, since the two versions (penalized regression or pure regression) of each GAM produce very similar linear predictors for any replicate set of data, it follows that they must produce sim-

ilar statistics λ , which must therefore follow similar distributions (by construction $EDF_1 - EDF_0 \approx p_1 - p_0$). i.e. there is some reason to expect (4.43) to be not too far from the truth.

In the case in which the scale parameter is unknown, an F-ratio test would be used exactly as in section 2.1.6, with the modifications that maximum penalized likelihood estimates replace MLEs and effective degrees of freedom are used in place of real degrees of freedom. The justification follows from that for (4.43).

Note that strictly, the foregoing argument requires the approximating regression models to be properly nested, suggesting that (4.43) will be most reliable if we are careful to ensure that each smooth term in the estimated null model has no more effective degrees of freedom than same term in the alternative model (this can easily be forced in fitting). Note also, that the less like a spline a smoother is, the less clear it is that the argument in this section applies. However, for penalized regression smoothers of any sort it is probably the case that a pure regression approximation based on truncating the natural parameterization (section 4.10.4) of the smoother would be adequate for the purposes of the argument. Hastie and Tibshirani (1990) provide an alternative justification of (4.43).

As was mentioned in section 4.8.5, if hypothesis testing is a key aim of an analysis, then it is sometimes preferable to forgo penalization altogether in order to ensure that p-values are correct (i.e. have a uniform distribution under the null). This works because an un-penalized GAM is simply a GLM, for which Generalized Likelihood Ratio Tests work reliably. However, some care is required to ensure that smoothing bases are sufficiently small that test power is maintained, while being sufficiently large that the test is not compromised by model mis-specification. Of course having tested hypotheses in this way, it is still preferable to use penalized versions of the model(s) for point estimation and confidence interval calculations.

4.10.2 ANOVA decompositions and Nesting

It is sometimes of interest to fit models with a linear predictor containing terms like,

$$f_1(x) + f_2(z) + f_3(x, z),$$

which can be thought of as an ANOVA decomposition of a function of x and z (there is a rich literature on such models in the context of smoothing splines, for example Wahba et al., 1995; Gu, 2002). If we use the tensor product methods of section 4.1.8, then it is quite easy to ensure that bases for the different terms are appropriately nested. For example the basis for $f_1(x) + f_2(z)$ will be strictly nested within the basis for $f_3(x, z)$, if f_3 is represented using a tensor product basis which uses the bases for f_1 and f_2 as marginal bases. By strict nesting is meant that $f_3(x, z)$ could exactly represent any possible $f_1(x) + f_2(z)$, given the bases used. Of course we do not *have* to ensure this exact nesting occurs, but it makes some aspects of model interpretation easier if we do.

It is worth thinking quite carefully before using such models, for, even if we set

things up to ensure strict nesting of the bases for f_1 and f_2 within the basis for f_3 , the notion of smoothness implied by using the ANOVA decomposition will be different to the notion of smoothness employed by using f_3 alone. The ANOVA decomposition model has a more complicated wigginess penalty than the simple smooth of x and z , with 2 extra smoothing parameters to be estimated. One way of viewing this, is as a way of allowing flexibility in the form of the wigginess measure, so that the final choice of what smoothness means is somewhat data driven.

If such nested models are used, then it is necessary to impose some identifiability conditions, if they are to be estimable. A general method, which can deal with any choice of bases for the smooths, is as follows. Working through all smooths, starting from smooths of two variables and working up through smooths of more variables:

1. Identify all smooths of fewer or the same number of variables sharing covariates with the current smooth.
2. Numerically identify any linear dependence of the basis for the current smooth on the other smooths sharing its covariates, and constrain the current smooth to remove this.

Numerical identification of dependency is fairly straightforward. Let \mathbf{X}_1 be the combined model matrix for all the lower order model terms sharing covariates with the smooth of interest, and \mathbf{X}_2 be the model matrix of the smooth of interest. Provided \mathbf{X}_1 and \mathbf{X}_2 are separately of full column rank, then we could test for dependence between them by forming the QR decomposition,

$$[\mathbf{X}_1 : \mathbf{X}_2] = \mathbf{Q}\mathbf{R},$$

where the columns of \mathbf{Q} are columns of an orthogonal matrix, and \mathbf{R} is full rank upper triangular if $[\mathbf{X}_1 : \mathbf{X}_2]$ is of full column rank, and reduced rank otherwise. Order d rank deficiency of \mathbf{R} is identified by a $d \times d$ zero block at its lower right corner, but to identify which columns of \mathbf{X}_2 to remove in order to fix this requires identification of the rows of \mathbf{R} at which the non-zero elements ‘step-in’ from the leading diagonal of the matrix. Furthermore, for the best performance, with rank deficient matrices the QR decomposition should really be done with column pivoting, but this may result in columns of \mathbf{X}_1 being identified as candidates for removal, which makes no sense in the current context.

Given these considerations, a practical approach is based on performing the QR decomposition of $[\mathbf{X}_1 : \mathbf{X}_2]$ with pivoting, but in two parts, so that columns of \mathbf{X}_1 can not be ‘pivoted past’ columns of \mathbf{X}_2 . This is achieved as follows. Form the QR decomposition

$$\mathbf{X}_1 = \mathbf{Q}_1 \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}.$$

Now let \mathbf{B} be $\mathbf{Q}_1^\top \mathbf{X}_2$ with the first r rows removed, where r is the number of columns of \mathbf{X}_1 . Now form a second QR decomposition with pivoting

$$\mathbf{B} = \mathbf{Q}_2 \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{0} \end{bmatrix}.$$

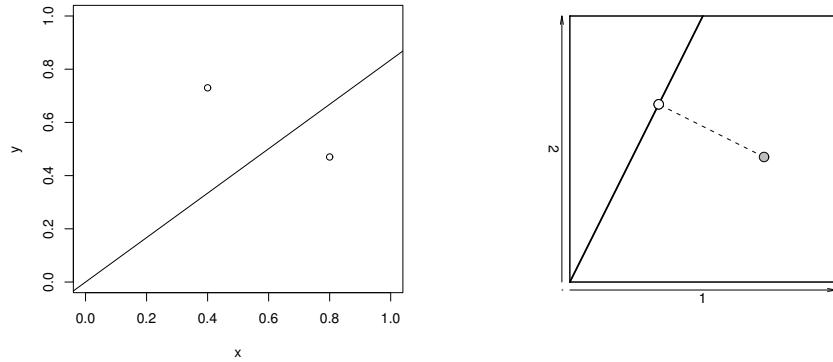


Figure 4.20 *The simple, one parameter, linear regression used to illustrate the geometry of penalized regression in section 4.10.3. The left panel shows the least square fit to two response data of a straight line through the origin. The right panel shows the corresponding un-penalized fitting geometry in a space in which there is an axis for each response datum, and the response data are therefore represented by a single point (the grey circle). The thick line shows the model-subspace: the model fitted values must lie on this line. The open circle is the orthogonal projection of the response data onto the model space: i.e. the fitted values.*

If some columns of \mathbf{X}_2 depend on columns of \mathbf{X}_1 then there will be a zero block at the lower right corner of \mathbf{R}_2 , and columns of \mathbf{X}_2 responsible for this block will be identifiable from the record of which columns of \mathbf{X}_2 have been pivoted to these final columns. These dependent columns can be removed, to make the model identifiable, or equivalently, the corresponding parameters can be constrained to zero.

If the basis of this algorithm is unclear, note that the implied QR decomposition used is

$$[\mathbf{X}_1 : \mathbf{X}_2] = \mathbf{Q}_1 \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & \bar{\mathbf{B}} \\ \mathbf{0} & \mathbf{R}_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $\bar{\mathbf{B}}$ is the first r rows of $\mathbf{Q}_1^T \mathbf{X}_2$.

4.10.3 The geometry of penalized regression

The geometry of linear and generalized linear model fitting, covered in sections 1.4 and 2.2, becomes more complicated when quadratically penalized estimation is used. In the unpenalized cases a response data vector, \mathbf{y} , which is exactly representable by the model (i.e. $g(y_i) = \mathbf{X}_i\beta$ for some β and all i), results in model fitted values $\hat{\mathbf{y}} = \mathbf{y}$. When penalized estimation is used, then this is generally not the case (except for \mathbf{y} that are exactly representable by the model, with parameter values for which the penalty is exactly zero). However, there is a geometric interpretation of penal-

ized fitting, in terms of projections in a larger space than the ‘data space’ that was considered in sections 1.4 and 2.2.

Consider the model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

to be estimated by minimization of the penalized objective function

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\beta^T \mathbf{S}\beta. \quad (4.44)$$

A geometric interpretation of penalized estimation can be obtained by re-writing (4.44) as

$$\left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{S} \end{bmatrix} \beta \right\|^2$$

i.e. as the ordinary least squares objective for an augmented model, fitted to the response data augmented with zeroes. So penalized regression amounts to orthogonally projecting $\begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}$ onto the space spanned by the columns of $\begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{S} \end{bmatrix}$ and then orthogonally projecting back onto the space spanned by \mathbf{X} , to get the fitted values corresponding to \mathbf{y} .

It’s difficult to draw pictures that really illustrate all aspects of this geometry in a satisfactory way, but some insight can be gained by considering the model

$$y_i = x_i\beta + \epsilon_i$$

for 2 data. Figure 4.20 illustrates the unpenalized fitting geometry for this model, as in section 1.4. Now consider fitting this model by penalized regression: in fact by simple ridge regression. The model fitting objective is then

$$\sum_{i=1}^2 (y_i - x_i\beta)^2 + \lambda\beta^2 = \left\| \begin{bmatrix} y_1 \\ y_2 \\ 0 \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \\ \sqrt{\lambda} \end{bmatrix} \beta \right\|^2.$$

The geometry of this model fit is shown in figure 4.21. Notice the way in which increasing λ results in fitted values closer and closer to zero.

For the simple penalty, illustrated in figure 4.21, only $\beta = 0$ results in a zero penalty, and hence only the response data $(0, 0)$ would result in identical fitted values $(0, 0)$. For penalties on multiple parameters, the penalty coefficient matrix, \mathbf{S} , is generally d short of full rank, where d is some integer. There is then a rank d subspace of the model space, for which the fitting penalty is identically zero, and if the data happen to be in that space (i.e. with no component outside it) then the fitted values will be exactly equal to the response data. For example, data lying exactly on a straight line are left unchanged by a cubic spline smoother.

4.10.4 The “natural” parameterization of a penalized smoother

Penalized smoothers, with a single penalty, can always be parameterized in such a way that the parameter estimators are independent, with unit variance, in the absence

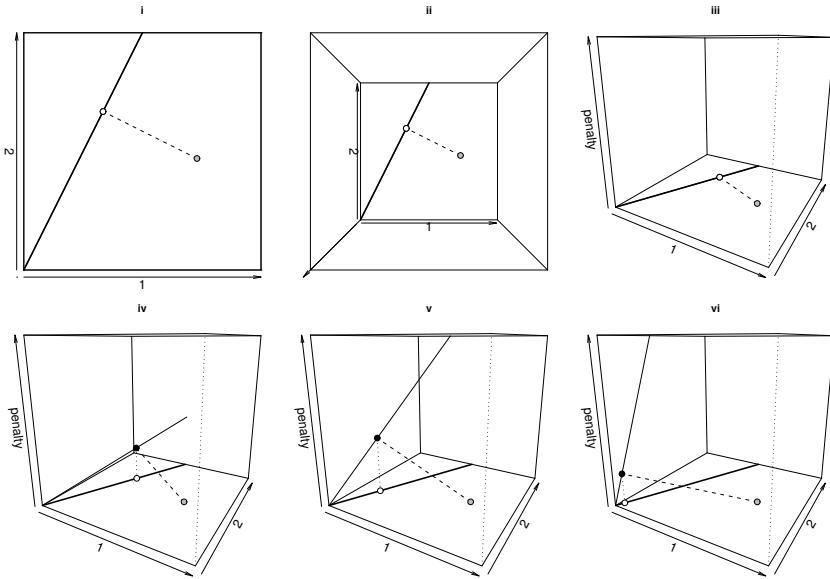


Figure 4.21 *Geometry of penalized linear regression for the model shown in figure 4.20.* (i) The unpenalized geometry, exactly as in figure 4.20. (ii) The space shown in panel (i) is augmented by a space corresponding to the penalty term. (iii) Another view of the augmented space from (ii), still corresponding to the unpenalized, $\lambda = 0$ case. (iv) penalized geometry for $\lambda = 0.1$. The thin line now shows the model subspace spanned by the columns of the augmented model matrix. Fitting amounts to finding the orthogonal projection from the (zero augmented) response data vector (grey circle) onto the augmented model subspace (giving the black circle). The model fitted values for the (un-augmented) response data are obtained by then projecting back onto the original model subspace to get the model fitted values (the open circle). (v) Same as (iv) for the $\lambda = 1$. (vi) same as (iv) for $\lambda = 10$. Notice how even data lying exactly on the original model subspace, results in fitted values that are different from those data: only zero data are not shrunk in this way.

of the penalty, and the penalty matrix is diagonal. This parameterization is particularly helpful for understanding the way in which the penalty suppresses model degrees of freedom.

Consider a smooth with model matrix \mathbf{X} , parameter vector, β , wigginess penalty coefficient matrix, \mathbf{S} and smoothing parameter λ . Suppose that the model is to be estimated by penalized least squares from data with variance σ^2 . Forming the QR decomposition

$$\mathbf{X} = \mathbf{QR},$$

we can re-parameterize in terms of $\beta' = \mathbf{R}\beta$, so that the model matrix is now \mathbf{Q} , and the penalty matrix becomes $\mathbf{R}^{-T}\mathbf{SR}^{-1}$. Eigen-decomposing the penalty matrix yields

$$\mathbf{R}^{-T}\mathbf{SR}^{-1} = \mathbf{UDU}^T,$$

where \mathbf{U} is an orthogonal matrix, the columns of which are the eigenvectors of

$\mathbf{R}^{-T}\mathbf{S}\mathbf{R}^{-1}$, while \mathbf{D} is a diagonal matrix of the corresponding eigenvalues, arranged in decreasing order. Reparameterization, via a rotation/reflection of the parameter space, now yields parameters $\beta'' = \mathbf{U}^T\beta'$, and correspondingly a model matrix $\mathbf{Q}\mathbf{U}$ and penalty matrix \mathbf{D} . If the penalty is not applied then the covariance matrix for these parameters is $\mathbf{I}\sigma^2$, since \mathbf{U} is orthogonal, and the columns of \mathbf{Q} are columns of an orthogonal matrix.

If the penalty is applied, then the Bayesian covariance matrix of the parameters is simply the diagonal matrix $(\mathbf{I} + \lambda\mathbf{D})^{-1}\sigma^2$, from which the role of the penalty in limiting parameter variance is rather clear. The frequentist equivalent would simply be the square of the Bayesian covariance matrix.

The effective degrees of freedom matrix, $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^T\mathbf{X}$, in the original parameterization, now becomes the diagonal matrix:

$$(\mathbf{I} + \lambda\mathbf{D})^{-1}.$$

Hence the effective degrees of freedom for the i^{th} parameter is now given by $(1 + \lambda D_{ii})^{-1}$.

So, in the natural parameterization, all parameters have the same degree of associated variability, when un-penalized, and the penalty acts on each parameter independently (i.e. the degree of penalization of one parameter has no affect on the other parameters). In this parameterization the relative magnitudes of different D_{ii} terms directly indicate the relative degree of penalization of different components of the model space. Note that \mathbf{D} is uniquely defined — no matter what parameterization you start out with, the elements of \mathbf{D} are always the same.

Figure 4.22 illustrates how parameters get ‘shrunk’ by the penalty, using the natural parameterization. The fact that at most levels of penalization, some subspace of the model space is almost completely suppressed, while some other subspace is left almost unpenalized, is clear from these figures, and lends some support to the idea that a penalized fit is somehow equivalent to an unpenalized fit with degrees of freedom close to the effective degrees of freedom of the penalized model. However the fact that many parameters have intermediate penalization, means that this support is only limited.

The natural parameterization makes the penalized smoothers behave more like full spline models than is otherwise immediately apparent. For example, for a full spline smoother the EDF matrix is simply the influence matrix, \mathbf{A} , which also defines the Bayesian posterior covariance matrix $\mathbf{A}\sigma^2$ and the equivalent frequentist matrix $\mathbf{A}^2\sigma^2$. In other words, the relationship between these matrices, which holds for smoothing splines, also holds for general penalized regression smoothers with the natural parameterization.

The penalty’s action in effectively suppressing some dimensions of the model space is also readily apparent in the natural parameterization. For most smoothers the penalty assesses the “wiggliness” of the fitted model, in such a way that smoother functions are generally not simply closer to the zero function. In this circumstance,

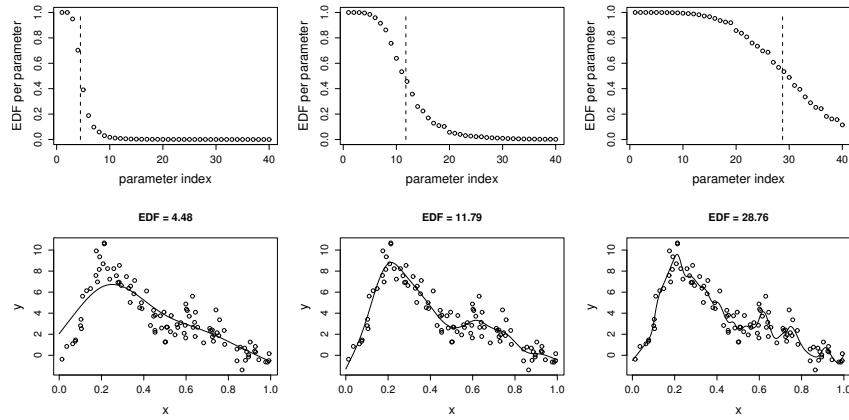


Figure 4.22 *How parameters are shrunk to zero by the penalty in the natural parameterization. All plots relate to a rank 40 cubic regression spline fit to the data shown in the bottom row plots. The top row plots the degrees of freedom associated with each parameter against parameter index using the natural parameterization, for 3 levels of penalization. On each plot the effective degrees of freedom (i.e. the effective number of free parameters) is shown by a vertical dashed line: note that this is simply the sum of the plotted degrees of freedom per parameter. The lower row shows the smooth fit corresponding to each of the top row plots. For this smooth the first two ‘natural’ parameters are never penalized. Notice how some potentially penalized parameters are effectively un-penalized, some are effectively suppressed completely, while some suffer intermediate penalization. Clearly, it is only approximately true that a penalized fit is equivalent to an unpenalized fit with the number of parameters given by the penalized fit effective degrees of freedom.*

if increased penalization is to lead to continuously smoother models then, in the natural parameterization, it is clear that the elements of \mathbf{D} must have a spread of (non-negative) values, with the higher values penalizing coefficients corresponding to more wiggly components of the model function. If this not clear, consider the converse situation in which all elements on the leading diagonal of \mathbf{D} are the same. In this case increased penalization amounts to simply multiplying each model coefficient by the same constant, thereby shrinking the fitted function towards zero without changing its shape.

4.11 Other approaches to GAMs

The name “Generalized Additive Model” was coined by Hastie and Tibshirani, who first proposed this class of models, along with methods for their estimation, and associated inference (Hastie and Tibshirani, 1986, 1990). Their proposed GAM estimation technique of “backfitting” has the advantage that it allows the component functions of an additive model to be represented using almost any smoothing or mod-

elling technique (regression trees for example). The disadvantage is that estimation of the degree of smoothness of a model is hard to integrate into this approach.

The generalized smoothing spline methods of Wahba, Gu and co-workers also encompass GAMs, within a rather complete theoretical framework, which does allow smoothness estimation, but is restricted to splines as model components (Wahba, 1990; Gu, 2002, see e.g.). Until recently, this framework had the drawback of rather high computational cost, but recent work by Kim and Gu (Gu and Kim, 2002; Kim and Gu, 2004) has much improved this situation.

Another alternative, is to take the fully Bayesian approach to GAMs of Fahrmeir, Lang and co-workers (e.g. Fahrmeir and Lang, 2001; Fahrmeir et al., 2004; Lang and Bresger, 2004). In this case the representation of GAMs is rather similar to what has been presented in this chapter, but the Bayesian model of section 4.8.1 is pushed to its logical conclusion and all inference is fully Bayesian, using MCMC.

The aim of the remainder of this section is to give a brief and partial sketch of the key ideas underpinning the backfitting and generalized smoothing spline frameworks. The aim is not to be comprehensive, but simply to give starting points for understanding these approaches. For full treatments, see the books by Hastie and Tibshirani (1990) and Gu (2002), and for an introduction to practical computation with these methods see section 5.6.

4.11.1 Backfitting GAMs

Backfitting is a beautifully simple way of fitting GAMs, which allows an enormous range of possibilities for representing the component functions of a GAM, including many that are not really smooth at all, in the sense that we have been considering so far. In this section only the basic principles of the simplest version of backfitting will be presented.

The basic idea behind backfitting is to estimate each smooth component of an additive model by iteratively smoothing *partial residuals* from the AM, with respect to the covariate(s) that the smooth relates to. The partial residuals relating to the j^{th} smooth term are the residuals resulting from subtracting all the current model term estimates from the response variable, *except* for the estimate of j^{th} smooth. Almost any smoothing method (and mixtures of methods) can be employed to estimate the smooths.

Here is a more formal description of the algorithm. Suppose that the object is to estimate the additive model:

$$y_i = \alpha + \sum_{j=1}^m f_j(x_{ji}) + \epsilon_i$$

where the f_j are smooth functions, and the covariates x_j , may sometimes be vector covariates. Let \hat{f}_j denote the vector whose i^{th} element is the estimate of $f_j(x_{ji})$. The basic backfitting algorithm is as follows.

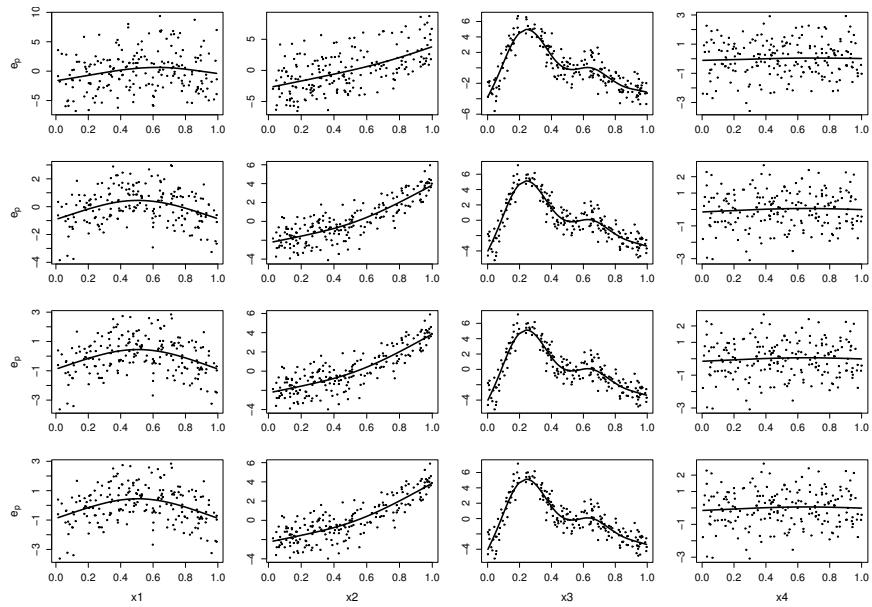


Figure 4.23 *Fitting a 4 term Additive Model using backfitting. Iteration proceeded across the columns and down the rows (i.e. starts at top left and finishes at bottom right). The j^{th} column relates to the j^{th} smooth and its covariate. The points shown in each plot are the partial residuals for the term being estimated, plotted against the corresponding covariate. By smoothing these residuals, an estimate of the corresponding smooth function (evaluated at the covariate values) is produced: these estimates are shown as thick curves. For example, the third column, second row, shows the partial residuals for f_3 , at the second iteration, plotted against x_3 : the thick curve is therefore the second iteration estimate of f_3 , and results from smoothing the partial residuals with respect to x_3 . The estimated functions change moving down the rows, but have stabilized by the last row.*

1. Set $\hat{\alpha} = \bar{y}$ and $\hat{\mathbf{f}}_j = \mathbf{0}$ for $j = 1, \dots, m$.
2. Repeat steps 3 to 5 until the estimates, $\hat{\mathbf{f}}_j$, stop changing.
3. For $j = 1, \dots, m$ repeat steps 4 and 5.
4. Calculate partial residuals:

$$\mathbf{e}_p^j = \mathbf{y} - \hat{\alpha} - \sum_{k \neq j} \hat{\mathbf{f}}_k$$

5. Set $\hat{\mathbf{f}}_j$ equal to the result of smoothing e_p^j with respect to x_j .

As an example, here is some R code implementing the basic backfitting algorithm, using R routine `smooth.spline` as the smoothing method. It is assumed that `edf[j]` contains the required degrees of freedom for the j^{th} smooth, that `x` is an `m` column array, with j^{th} column containing the (single) covariate for the j^{th} smooth, and that the response is in `y`.

```

f<-x*0;alpha<-mean(y);ok <- TRUE
while (ok) { # backfitting loop
  for (i in 1:m) { # loop through the smooth terms
    ep <- y - rowSums(f[,-i]) - alpha
    b <- smooth.spline(x[,i],ep,df=edf[i])
    f[,i] <- predict(b,x[,i])$y
  }
  rss <- sum((y-rowSums(f))^2)
  if (abs(rss-rss0)<1e-6*rss) ok <- FALSE
  rss0 <- rss
}

```

Figure 4.23 shows 4 iterations of the backfitting algorithm applied to the estimation of an additive model with 4 smooth components. In this case the data were simulated using the code given in the help file for `gam` in the `mgcv` package. The `edf` array was set to 3, 3, 8 and 3.

To fit GAMs by penalized iteratively re-weighted least squares, a weighted version of the backfitting algorithm is used, and there are some refinements which speed up convergence (which can otherwise be slow if covariates are highly correlated). Notice how the cost of the algorithm depends on the cost of the componentwise smoothers: `smooth.spline` is a very efficient way of smoothing with respect to one variable, for example. Clearly the elegance of backfitting is appealing, as is the flexibility to choose a very wide variety of smoothing methods, for component estimation, but this does come at a price. It is not easy to efficiently integrate smoothness estimation methods with backfitting: the obvious approach of applying GCV to the smoothing of partial residuals is definitely not recommendable, while direct estimation of the influence/hat matrix of the whole model has computational cost of at least $O(mn^2)$, where n is the number of data, so that cross validation or GCV would both be quite expensive.

4.11.2 Generalized smoothing splines

The generalized smoothing spline approach is built around some very elegant and general methods for smoothing, based on the theory of reproducing kernel Hilbert spaces. This section will only give the briefest introduction to this theory, but even that is somewhat tougher going than the rest of this book.

To reduce the level of abstraction, a little, let us revisit the cubic smoothing spline. To this end, first construct a space of functions of x , say, which are ‘wiggly’ according to the cubic spline penalty. That is consider a space of functions, \mathcal{F} , where the inner product of two functions, f and $g \in \mathcal{F}$ is $\langle g, f \rangle = \int g''(x)f''(x)dx$, and consequently a norm on the space is the cubic spline penalty, $\int f''(x)^2dx$: except for the zero function, functions for which this norm is zero are currently excluded from \mathcal{F} .

There is a rather remarkable theorem, the Reisz representation theorem, which says that there exists a function $R_z \in \mathcal{F}$, such that $f(z) = \langle R_z, f \rangle$, for any $f \in \mathcal{F}$: i.e. if

we want to evaluate f at some particular value z , then one way of doing it is to take the inner product of the function, f , with the ‘representor of evaluation’ function, R_z .

Suppose further that we construct a function of two variables $R(z, x)$, such that for any z , $R(z, x) = R_z(x)$. This function is known as the *reproducing kernel* of the space, since $\langle R(z, \cdot), R(\cdot, t) \rangle = R(z, t)$, i.e.

$$\int \frac{\partial^2 R(z, x)}{\partial x^2} \frac{\partial^2 R(x, t)}{\partial x^2} dx = R(z, t),$$

by the Reisz representation theorem. Basically, if you take an appropriately defined inner product of R , with itself, you get back R : hence the terminology. If you must know what R actually looks like see (3.4) in section 3.2.1.

Now, consider the cubic smoothing spline problem of finding the *function* minimizing:

$$\sum_{i=1}^n \{y_i - f(x_i)\}^2 + \lambda \int f''(x)^2 dx. \quad (4.45)$$

It turns out that the minimizer is in the space of functions that can be represented as:

$$\hat{f}(x) = \beta_0 + \beta_1 x + \sum_{i=1}^n \delta_i R(x_i, x).$$

where the β_j 's and δ_i 's are coefficients to be estimated. The first two terms on the r.h.s simply span the space of functions for which $\int f''(x)^2 dx = 0$, the null space of the penalty: clearly β_0 and β_1 can be chosen to minimize the sum of squares term without worrying about the effect on the penalty. The terms in the summation represent the part of $\hat{f}(x)$ that is in \mathcal{F} . It is clear that not all functions in \mathcal{F} can be represented as $\sum_{i=1}^n \delta_i R(x, x_i)$, so how do we know that the minimizer of 4.45 can?

The answer lies in writing the minimizer, \hat{f} 's, component in \mathcal{F} as $r + \eta$ where $r = \sum_{i=1}^n \delta_i R(x, x_i)$, and η is any function in the part of \mathcal{F} which is orthogonal to r . Orthogonality means that each inner product $\langle R(x_i, x), \eta(x) \rangle = 0$, but these inner products evaluate $\eta(x_i)$, so we have that $\eta(x_i) = 0$ for each i : i.e. $\eta(x)$ can have no effect on the sum of squares term in (4.45). In the case of the penalty term, orthogonality of r and η means that:

$$\int \hat{f}''(x)^2 dx = \int r''(x)^2 dx + \int \eta''(x)^2 dx.$$

Obviously, the member of \mathcal{F} which minimizes this, is the zero function, $\eta(x) = 0$.

Having demonstrated what form the minimizer has, we must now actually find the minimizing β_j 's and δ_i 's. Given that we now have a basis, this is not difficult, although it is necessary to derive two linear constraints ensuring identifiability between the null space components and the reproducing kernel terms in the model, before this can be done.

The above argument is excessively complicated if all that is required is to derive the

cubic smoothing spline, but its utility lies in how it generalizes. The cubic spline penalty can be replaced by any other derivative based penalty with associated inner product, and x can be a vector. In fact this *reproducing kernel Hilbert space* approach generalizes in all sorts of wonderful directions, and in contrast to the other methods described in this book, the basis for representing component functions of a model emerges naturally from the specification of fitting criteria like (4.45), rather than being a more or less arbitrary choice made by the modeller. See Wahba (1990) or Gu (2002) for a full treatment of this approach. The down sides are theoretical difficulty, less freedom to choose smoothers, greater difficulty in generalizing the models in some directions, and for full models, computational cost. However the work of Kim and Gu (2004) has made this point much less significant than it was.

4.12 Exercises

1. It is easy to see that the cubic regression spline defined by equation (4.2) in section 4.1.2 has value β_j and second derivative δ_j at knot x_j , and that value and second derivative are continuous across the knots. Show that the condition that the first derivative of the spline be continuous across x_j (and that the second derivative be zero at x_1 and x_k), leads to equation (4.3).
2. This question refers, again, to the cubic regression spline (4.2) of section 4.1.2.
 - (a) Show that the second derivative of the spline can be expressed as

$$f''(x) = \sum_{i=2}^{k-2} \delta_i d_i(x),$$

where

$$d_i(x) = \begin{cases} (x - x_i)/h_{i-1} & x_{i-1} \leq x \leq x_i \\ (x_{i-1} - x)/h_i & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

- (b) Hence show that, in the notation of section 4.1.2,

$$\int f''(x)^2 dx = \boldsymbol{\delta}^{-\top} \mathbf{B} \boldsymbol{\delta}^-.$$

(It may be helpful to review exercise 7 in Chapter 3, before proceeding with this part.)

- (c) Finally show that

$$\int f''(x)^2 dx = \boldsymbol{\beta}^\top \mathbf{D}^\top \mathbf{B}^{-1} \mathbf{D} \boldsymbol{\beta}.$$

3. Prove that equation (4.14) in section 4.4.1 is true.
4. This question follows on from section 4.5.4. Show that, in the notation of that section,

$$\mathbb{E} \left(\frac{n \sum_i V(\mu_i)^{-1} (y - \mu_i)^2}{\{n - \text{tr}(\mathbf{A})\}^2} \right) = \frac{n^2}{\{n - \text{tr}(\mathbf{A})\}^2}$$

if the y_i are binary random variables with mean μ_i and the model smoothing parameters are treated as known. What does this indicate about \mathcal{V}_g^p ?

5. The natural parameterization of section 4.10.4 is particularly useful for understanding the way in which penalization causes bias in estimates, and this question explores this issue.
- Find an expression for the bias in a parameter estimator $\hat{\beta}_i''$ in the natural parameterization (bias being defined as $\mathbb{E}\{\hat{\beta}_i'' - \beta_i''\}$). What does this tell you about the bias in components of the model which are unpenalized, or only very weakly penalized, and in components for which the ‘true value’ of the corresponding parameter is zero or nearly zero?
 - The Mean Square Error in a parameter estimator (MSE) is defined as $\mathbb{E}\{(\hat{\beta}_i - \beta_i)^2\}$ (dropping the primes for notational convenience). Show that the MSE of the estimator is in fact the estimator variance plus the square of the estimator bias.
 - Find an expression for the mean square error of the i^{th} parameter of a smooth in the natural parameterization.
 - Show that the lowest achievable MSE, for any natural parameter, is bounded above by σ^2 , implying that penalization always has the potential to reduce the MSE of a parameter *if the right smoothing parameter value is chosen*. Comment on the proportion of the minimum achievable MSE that is contributed by the squared bias term, for different magnitudes of parameter value.
 - Under the prior from the Bayesian model of section 4.8.1, but still working in the natural parameterization, show that the expected squared value of the i^{th} parameter is $\sigma^2/(\lambda_i D_{ii})$. Using this result as a guide to the typical size of β_i^2 , find an expression for the typical size of the squared bias in a parameter, and find a corresponding upper bound on the squared bias as a proportion of σ^2 .
6. Cross validation can fail completely for some problems: this question explores why.
- Consider attempting to smooth some response data y_i by solving the ‘ridge regression’ problem

$$\text{minimise } \sum_{i=1}^n (y_i - \mu_i)^2 + \lambda \sum_{i=1}^n \mu_i^2 \text{ w.r.t. } \boldsymbol{\mu},$$

where λ is a smoothing parameter. Show that for this problem the GCV and OCV scores are identical, and are independent of λ .

- By considering the basic principle underpinning ordinary cross validation, explain what causes the failure of cross validation in part (a).
- Given the explanation of the failure of cross validation for the ridge regression problem in part (a), it might be expected that the following modified approach would work better. Suppose that we have an x_i covariate observed for each y_i (and for convenience $x_i < x_{i+1} \forall i$). Define the function $\mu(x)$ to be the piecewise linear interpolant of the points (x_i, μ_i) . In this case we could estimate the μ_i by minimizing the following penalized least squares objective

$$\sum_{i=1}^n (y_i - \mu_i)^2 + \lambda \int \mu(x)^2 dx$$

w.r.t. the μ_i .

Now consider 3 equally spaced points x_1, x_2, x_3 with corresponding μ values μ_1, μ_2, μ_3 . Suppose that $\mu_1 = \mu_3 = \mu^*$, but that we are free to choose μ_2 . Show that in order to minimize $\int_{x_1}^{x_3} \mu(x)^2 dx$ we should set $\mu_2 = -\mu^*/2$. What does this imply about trying to choose λ by cross validation? (Hint: think about what the penalty will do to μ_i if we ‘leave out’ y_i .)

- (d) Would the penalty:

$$\int \mu'(x)^2 dx$$

suffer from the same problem as the penalty in part (c)?

- (e) Would you expect to encounter these sorts of problems with penalized regression smoothers? Explain.

7. This question concerns the P-splines of section 4.1.4, the eigen-basis of the penalty discussed in section 4.8.2 and the natural parameterization of section 4.10.4.

- (a) Write an R function which will return the model matrix and square root penalty matrix for a P-spline, given a sequence of knots, covariate vector x , the required basis dimension q , and the orders of B-spline and difference penalty required.
- (b) Simulate 100 uniform x data on $(0, 1)$, use your routine to evaluate the basis functions of a rank 9 B-spline basis at the x values, and plot these 9 basis functions. (Use cubic B-splines with a second order difference penalty.)
- (c) Following section 4.8.2 and using the data and basis from the previous part, re-parameterize in terms of the eigen-basis of the P-spline penalty matrix, and plot the 9 evaluated basis functions corresponding to the re-parameterization.
- (d) Re-parameterize the P-spline smoother using its natural parameterization (see section 4.10.4). Again plot the 9 evaluated basis functions corresponding to this parameterization.

8. This question covers P-IRLS and smoothing parameter estimation, using the P-spline set up in the previous question. The following simulates some x, y data:

```
f <- function(x) .04*x^11*(10*(1-x))^6+2*(10*x)^3*(1-x)^10
n <- 100;x <- sort(runif(n))
y <- rpois(rep(1,n),exp(f(x)))
```

An appropriate model for the data is $y_i \sim \text{Poi}(\mu_i)$ where $\log(\mu_i) = f(x_i)$: f is a smooth function, representable as a P-spline, and the y_i are independent.

- (a) Write an R function to estimate the model by penalized likelihood maximization, given a smoothing parameter. Re-use the function from the first part of the previous question, in order to set up the P-spline for f .
- (b) Modify your routine to return the model deviance and model degrees of freedom, so that a GCV score can be calculated for the model.
- (c) Write a loop to fit the model and evaluate its (deviance based) GCV score, on a grid of smoothing parameter values. Plot the fitted values from the GCV optimal model, over the original data.

9. In section 4.5.4 it was pointed out that the approximations underpinning the deviance based GCV score are not very good, and that in principle a better GCV score might be obtained by adding a constant to the deviance in the GCV score, the constant being estimated from an initial model fit in which it is taken to be zero. Following on from the previous question, apply this approach and compare the best fit models, with and without the correction to the GCV score.
10. The following R code simulates some data sampled with noise from a function of two covariates:

```
test1<-function(x,z,sx=0.3,sz=0.4)
{ 1.2*exp(-(x-0.2)^2/sx^2-(z-0.3)^2/sz^2) +
  0.8*exp(-(x-0.7)^2/sx^2-(z-0.8)^2/sz^2)
}
n <- 200
x <- matrix(runif(2*n),n,2)
f <- test1(x[,1],x[,2])
y <- f + rnorm(n)*.1
```

Write an R function to fit a thin plate spline of two variables (penalty order $m = 2$) to the simulated $\{y_i, \mathbf{x}_i\}$ data, given a smoothing parameter value. Write the function so that it can fit either a full thin plate spline or a ‘knot based’ thin plate regression spline (see first and last subsections of section 4.1.5 for details). To deal with the linear constraints on the spline, see section 1.8.1 and `?qr`. The simple augmented linear model approach introduced in section 3.2.2 can be used for actual fitting. Write another function to evaluate the fitted spline at new covariate values. Use your functions to fit thin plate splines to the simulated data and produce contour plots of the results, for several different smoothing parameter values.

11. The natural parameterization of section 4.10.4 also allows the GCV score for a single smoothing parameter model to be evaluated very efficiently, for different trial values of the smoothing parameter. Consider such a single smooth to be fitted to response data \mathbf{y} : this question examines the efficient calculation of the components of the GCV score.
 - (a) In the notation of section 4.10.4, find an expression for the influence matrix \mathbf{A} in terms of \mathbf{Q} , \mathbf{U} , \mathbf{D} and λ .
 - (b) Show that the effective degrees of freedom of the model is simply:

$$\sum_{i=1}^k \frac{1}{1 + \lambda D_{ii}}.$$

- (c) Again using the natural parameterization, show how calculations can be arranged so that, after some initial setting up, each new evaluation of $\|\mathbf{y} - \mathbf{Ay}\|^2$ costs only $O(k)$ arithmetic operations.

CHAPTER 5

GAMs in practice: mgcv

This chapter covers use of the generalized additive modelling functions provided by R package `mgcv`: the design of these functions is based largely on Hastie (1993), although to facilitate smoothing parameter estimation, their details have been modified. It is also well worth being aware of other packages available for GAM type modelling in R. At time of writing, two other packages stand out: `gss`, written by Chong Gu and `gam`, written by Trevor Hastie. There is not space in this book to cover these in detail, but section 5.6 offers brief introductions to both. Packages `assist` and `gamlss` are also available at cran.r-project.org, and the `vgam` package is also worth seeking out.

The `gam` function from library `mgcv` is very much like the `glm` function covered in chapter 2. The main difference is that the `gam` model formula can include smooth terms, `s()` and `te()`, and there are a number of options available for controlling automatic smoothness selection, or for directly controlling model smoothness. Some simple examples are helpful for introducing the main features, so this chapter starts with the cherry tree data from chapter 3, before moving on to some more realistic examples. When reading this chapter note that R and the `mgcv` package are subject to continuing efforts to improve them. Sometimes this may involve modifications of numerical optimization behaviour, which may result in noticeable, but (hopefully) statistically unimportant, differences between the output given in this chapter, and the corresponding results with more recent versions. Sometimes the exact formatting of output can also change a little.

5.1 Cherry trees again

The example with which chapter 3 ended is easily re-done.

```
library(mgcv)
data(trees)
ct1<-gam(Volume~s(Height)+s(Girth),
           family=Gamma(link=log), data=trees)
```

This fits the Generalized additive model

$$\log(\mathbb{E}[\text{Volume}_i]) = f_1(\text{Height}_i) + f_2(\text{Girth}_i) \text{ where } \text{Volume}_i \sim \text{Gamma}$$

and the f_j are smooth functions. The degree of smoothness (within certain limits) of the f_j is estimated by GCV. The results can be checked by typing the name of the fitted model object to invoke the `print.gam` print method, and by plotting the fitted model object. For example

```
> ctl
Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
1.076070 2.408379 total = 4.484449

GCV score: 0.008103299
> plot(ctl,residuals=TRUE)
```

The resulting plot is displayed in the upper two panels of figure 5.1. Notice that the default print method reports the model distribution family, link function and formula, before displaying the effective degrees of freedom for each term (in the order that the terms appear in the model formula) and the whole model: in this case a straight line, corresponding to one degree of freedom, is estimated for the effect of height, while the effect of Girth is estimated as a smooth curve with 2.4 degrees of freedom; the total degrees of freedom is the sum of these two, plus one degree of freedom for the model intercept. Finally the GCV score for the fitted model is reported.

The plots show the estimated effects as solid lines/curves, with 95% confidence limits (strictly Bayesian credible intervals, based on section 4.8) shown as dashed lines. The coincidence of the confidence limits and the estimated straight line, at the point where the line passes through zero on the vertical axis, is a result of the identifiability constraints applied to the smooth terms*. The points shown on the plots are *partial residuals*. These are simply the Pearson residuals added to the smooth terms evaluated at the appropriate covariate values. For example, the residuals plotted in the top left panel of figure 5.1 are given by

$$\hat{\epsilon}_{1i}^{\text{partial}} = f_1(\text{Height}_i) + \hat{\epsilon}_i^p$$

plotted against Height_i . For a well fitting model the partial residuals should be evenly scattered around the curve to which they relate. The ‘rug plots’, along the bottom of each plot, show the values of the covariates of each smooth, while the number in each y-axis caption is the effective degrees of freedom of the term being plotted.

* The identifiability constraint is that the sum of the values of each curve, at the observed covariate values, must be zero: for a straight line, this condition exactly determines where the line must pass through zero, so there can be no uncertainty about this point.

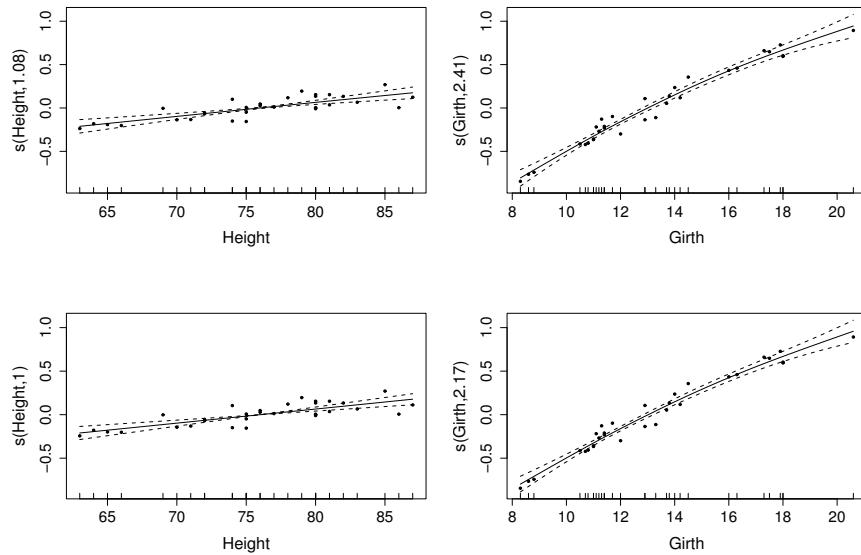


Figure 5.1 Components of GAM model fits to the cherry tree data. The upper two panels are from `ct1` and the lower 2 from `ct4`.

5.1.1 Finer control of `gam`

The simple form of the `gam` call producing `ct1` hides a number of options that have been set to default values. The first of these is the choice of basis used to represent the smooth terms. The default is to use thin plate regression splines, which have some appealing properties, but can be computationally costly for large data sets. In the following this is modified by using `s(..., bs="cr")` to select penalized cubic regression splines to represent the same cherry tree model.

```
> ct2<-gam(Volume~s(Height,bs="cr")+s(Girth,bs="cr"),
+ family=Gamma(link=log),data=trees)
> ct2

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height, bs = "cr") + s(Girth, bs = "cr")

Estimated degrees of freedom:
1.000126 2.418591    total = 4.418718

GCV score: 0.008080546
```

As you can see, the change in basis has made very little difference to the fit. Plots are in fact indistinguishable to those for `ct1`. This is re-assuring: it would be unfortunate if the model depended very strongly on details like the exact choice of basis. However, larger changes to the basis, such as using P-splines, can make an appreciable difference.

Another choice hidden, in the previous two model fits, is the choice of the *dimension*, k , of the basis used to represent smooth terms. In the previous two fits, the default, $k = 10$, was used. The choice of basis dimensions amounts to setting the maximum possible degrees of freedom allowed for each model term. The actual effective degrees of freedom, for each term, will usually be estimated from the data, by GCV or UBRE, but the upper limit on this estimate is $k - 1$: the basis dimension, less one degree of freedom due to the identifiability constraint on each smooth term. The following example sets k to 20 for the smooth of `Girth` (and illustrates, by the way, that there is no problem in mixing different bases).

```
> ct3 <- gam(Volume ~ s(Height)+s(Girth,bs="cr",k=20),
+               family=Gamma(link=log),data=trees)
> ct3

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth, bs = "cr", k = 20)

Estimated degrees of freedom:
1.000003 2.424226  total = 4.424229

GCV score: 0.00808297
```

Again, this change makes boringly little difference in this case, and the plots (not shown) are indistinguishable from those for `ct1`. This insensitivity to basis dimension is not universal, of course, and one quite subtle point is worth being aware of. This is that a space of functions of dimension 20, will contain a larger subspace of functions with effective degrees of freedom 5, than will a function space of dimension 10 (the particular numbers being arbitrary here). Hence it is often the case that increasing k will change the effective degrees of freedom estimated for a term, even though both old and new estimated degrees of freedom are lower than the original $k - 1$.

A final default choice, that it is worth being aware of, is the value for γ in the GCV or UBRE scores (expressions (4.28) or (4.29), respectively) optimized in order to select the degree of smoothness of each term. The default value is 1, but GCV is known to have some tendency to overfitting on occasion, and it has been suggested that using $\gamma \approx 1.4$ can largely correct this without compromising model fit (Kim and Gu, 2004). Applying this idea to the current model, results in the bottom row of figure 5.1 and the following output.

```

> ct4 <- gam(Volume ~ s(Height) + s(Girth),
+               family=Gamma(link=log), data=trees, gamma=1.4)
> ct4

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
1.00011 2.169248 total = 4.169358

GCV score: 0.00922805

> plot(ct4, residuals=TRUE)

```

Clearly the heavier penalty on each degree of freedom in the GCV score has resulted in a model with fewer degrees of freedom, but the figure indicates that the change in estimates that this produces is barely perceptible.

5.1.2 Smooths of several variables

`gam` is not restricted to models containing only smooths of one predictor. In principle, smooths of any number of predictors are possible via two types of smooth. Within a model formula, `s()`, terms using the "`tp`" or "`ts`" bases, produce isotropic smooths of multiple predictors, while `te()` terms produce smooths of multiple predictors from tensor products of *any* bases available for use with `s()` (including mixtures of different bases). The tensor product smooths are invariant to linear rescaling of covariates, and can be quite computationally efficient.

The following code fragments both fit the model

$$\log(\mathbb{E}[\text{Volume}_i]) = f(\text{Height}_i, \text{Girth}_i) \text{ where } \text{Volume}_i \sim \text{Gamma},$$

and f is a smooth function. Firstly an isotropic thin plate regression spline is used:

```

> ct5 <- gam(Volume ~ s(Height, Girth, k=25),
+               family=Gamma(link=log), data=trees)
> ct5

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height, Girth, k = 25)

Estimated degrees of freedom:
4.668129 total = 5.668129

```

bs	Description	Advantages	Disadvantages
"tp"	Thin plate regression splines. (TPRS)	Can smooth w.r.t. any number of covariates. Invariant to rotation of covariate axes. Can select penalty order No 'knots' and some optimality properties.	Computationally costly for large data sets. Not invariant to covariate rescaling.
"ts"	TPRS with shrinkage	As TPRS, but smoothness selection can zero term completely	as TPRS
"cr"	cubic regression spline (CRS)	Computationally cheap Directly interpretable parameters	Can only smooth w.r.t 1 covariate. Knot based Doesn't have TPRS optimality.
"cs"	CRS with shrinkage	As CRS, but smoothness selection can zero term completely.	As CRS
"cc"	cyclic CRS	As CRS, but start point same as end point	As CRS
"ps"	P-splines	Any combination of basis and penalty order possible Perform well in tensor products	Based on equally spaced knots Penalties awkward to interpret. No optimality properties available.

Table 5.1 *Smoothing bases built in to package mgcv, and a summary of their advantages and disadvantages. (To use P-splines see ?p.spline.)*

```
GCV score: 0.009358786
```

```
> plot(ct5,too.far=0.15)
```

yielding the left hand panel of figure 5.2. Secondly a tensor product smooth is used. Note that the k argument to te specifies the dimension for each marginal basis: if different dimensions are required for the marginal bases then k can also be supplied as an array.

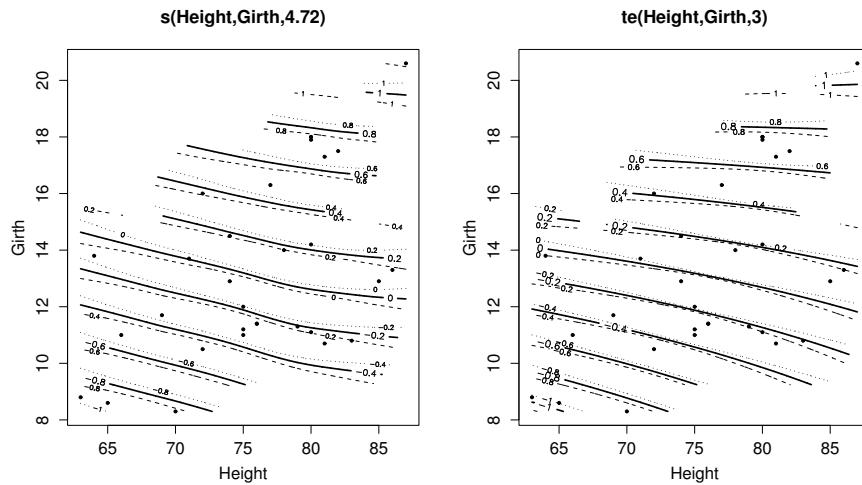


Figure 5.2 *Smooth functions of height and girth fitted to the cherry tree data, with degree of smoothing chosen by GCV. The left hand panel shows a thin plate regression spline fit (ct 5), while the right panel shows a tensor product spline fit (ct 6). For both plots the bold contours show the estimate of the smooth; the dashed contours show the smooth plus the standard error of the smooth and the dotted contours show the smooth less its standard error. The symbols show the locations of the covariate values on the height - girth plane. Parts of the smooths that are far away from covariate values have been excluded from the plots using the too.far argument to plot.gam.*

```

> ct6 <- gam(Volume ~ te(Height,Girth,k=5),
+               family=Gamma(link=log), data=trees)
> ct6

Family: Gamma
Link function: log

Formula:
Volume ~ te(Height, Girth, k = 5)

Estimated degrees of freedom:
3.000175 total = 4.000175

GCV score: 0.008197151

> plot(ct6,too.far=0.15)

```

Notice how the tensor product model has fewer degrees of freedom and a lower GCV score, than the TPRS smooth here. In fact, with just 3 degrees of freedom, the tensor

Tensor product, <code>te</code>	<code>TPRS, s(...,bs="tp")</code>
Invariant to linear rescaling of covariates, but not to rotation of covariate space.	Invariant to rotation of covariate space (isotropic), but not to rescaling of covariates.
Good for smooth interactions of quantities measured in different units, or where very different degrees of smoothness appropriate relative to different covariates.	Good for smooth interactions of quantities measured in same units, such as spatial co-ordinates, where isotropy is appropriate.
Computationally inexpensive, provided TPRS bases are not used as marginal bases.	Computational cost can be high as it increases with square of number of data (can be avoided by approximation).
Apart from scale invariance, not much supporting theory.	Some optimality results available.

Table 5.2 *Comparison of the tensor product (`te`) and thin plate regression spline (`s(...,bs="tp")` or `s(...,bs="ts")`) approaches to smoothing with respect to multiple covariates.*

product smooth model amounts to

$$\log(\mathbb{E}[\text{Volume}_i]) = \beta_0 + \beta_1 \text{Height}_i + \beta_2 \text{Girth}_i + \beta_3 \text{Height}_i \text{Girth}_i,$$

the ‘wiggly’ components of the model having been penalized away altogether.

5.1.3 Parametric model terms

So far, only models consisting of smooth terms have been considered, but there is no difficulty in mixing smooth and parametric model components. For example, given that the smooth of height, in model `ct1`, is estimated to be a straight line, we might as well fit the model:

```
gam(Volume~Height+s(Girth), family=Gamma(link=log), data=trees)
```

but to make the example more informative, let us instead suppose that the `Height` is actually only measured as a categorical variable. This can easily be arranged, by creating a factor variable which simply labels each tree as small, medium or large:

```
trees$Hclass <- factor(floor(trees$Height/10)-5,
                         labels=c("small", "medium", "large"))
```

Now we can fit a generalized additive model to these data, using the `Hclass` variable as a factor variable.

```
ct7 <- gam(Volume ~ Hclass+s(Girth),
            family=Gamma(link=log), data=trees)
```

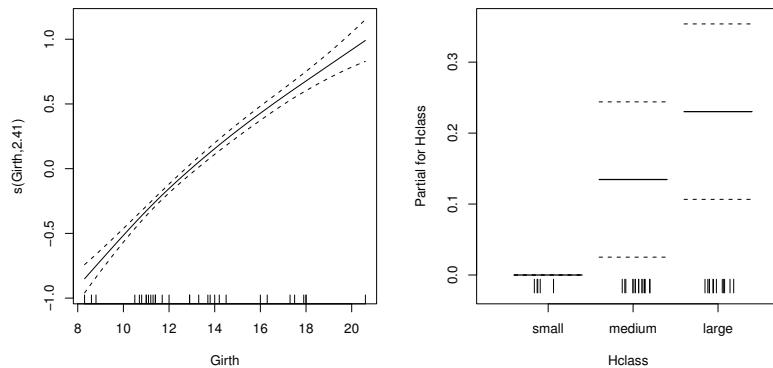


Figure 5.3 Plot of model `ct7`, a semi-parametric model of cherry tree volume, with a factor for height and a smooth term for the dependence on Girth. The left plot shows the smooth of Girth, with 95% confidence interval, while the right panel shows the estimated effect, for each level of factor `Hclass`. The effect of being in the small height class is shown as zero, because the default contrasts have been used here, which set the parameter for the first level of each factor to zero.

```
par(mfrow=c(1, 2))
plot(ct7, all.terms=T)
```

The resulting plot is shown in figure 5.3

Often, more information about a fitted model is required than is supplied by plots or the default print method, and various utility functions exist to provide this. For example the `anova` function can be used to investigate the approximate significance of model terms.

```
> anova(ct7)

Family: Gamma
Link function: log

Formula:
Volume ~ Hclass + s(Girth)

Parametric Terms:
  df      F p-value
Hclass  2 7.076 0.00358

Approximate significance of smooth terms:
  edf Est.rank   F p-value
s(Girth) 2.414    9.000 54.43 1.98e-14
```

Clearly there is quite strong evidence that both Height and Girth matter (see section 4.8.5, for information on the p-value calculations for the smooth terms). Similarly, an approximate AIC value can be obtained for the model:

```
> AIC(ct7)
[1] 154.9411
```

The `summary` method provides considerable detail.

```
> summary(ct7)

Family: Gamma
Link function: log

Formula:
Volume ~ Hclass + s(Girth)

Parametric coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.12693 0.04814 64.949 < 2e-16 ***
Hclassmedium 0.13459 0.05428 2.479 0.020085 *
Hclasslarge 0.23024 0.06137 3.752 0.000908 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
edf Est.rank F p-value
s(Girth) 2.414 9.000 54.43 1.98e-14 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq. (adj) = 0.967 Deviance explained = 96.9%
GCV score = 0.012076 Scale est. = 0.0099671 n = 31
```

Notice that, in this case, the significance of individual parameters of the parametric terms is given, rather than whole term significance. Other measures of fit are also reported, such as the adjusted r^2 and percentage deviance explained, along with the GCV score, an estimate of the scale parameter of the model, and the number of data fitted.

5.2 Brain Imaging Example

This section examines a more substantial example, in particular covering issues of model selection and checking in more detail. The data are from brain imaging by functional magnetic resonance scanning, and were reported in Landau et al. (2003). The data are available in data frame `brain`, and are shown in figure 5.4. Each row

scale	Controls whether to use UBRE or GCV for smoothness estimation. $\text{scale} > 0$ is taken as known scale parameter: UBRE used. $\text{scale} < 0 \Rightarrow$ scale parameter unknown: GCV used. $\text{scale} = 0 \Rightarrow$ UBRE for Poisson or binomial, otherwise GCV.
gamma	This multiplies the model degrees of freedom in the GCV or UBRE criteria. Hence as <code>gamma</code> is increased from 1 the ‘penalty’ per degree of freedom increases in the GCV or UBRE criterion and increasingly smooth models are produced. Increasing <code>gamma</code> to around 1.4 can usually reduce over-fitting, without much degradation in prediction error performance.
sp	An array of supplied smoothing parameters. When this array is non null, a negative element signals that a smoothing parameter should be estimated, while a non-negative value is used as the smoothing parameter for the corresponding term. This is useful for directly controlling the smoothness of some terms.
min.sp	Minimum values for smoothing parameters can be supplied here.
method	An argument usually set with <code>gam.method</code> , which controls the numerical method used to estimate smoothing parameters. The most important choice is between ‘outer’, and ‘performance’ iteration in the <i>generalized additive modelling</i> case.

Table 5.3 *Main arguments to `gam` for controlling the smoothness estimation process.*

of the data frame corresponds to one voxel. The columns are: `X` and `Y`, giving the locations of each voxel; `medFPQ`, the brain activity level measurement (median ‘Fundamental Power Quotient’ over three measurements); `region`, a code indicating which region the voxel belongs to (0 for ‘base’ region; 1 for region of experimental interest and 2 for region subjected to direct stimulation) — there are some NA’s in this column; `meanTheta` is the average phase shift at each voxel, which we will not consider further. This section will consider models for `medFPQ` as a function of `X` and `Y`.

Clearly the `medFPQ` data are quite noisy, and the main purpose of the modelling in this section is simply to partition this very variable signal into a smooth trend component and a ‘random variability’ component, so that the pattern in the image becomes a bit clearer. For data such as these, where the discretization (into voxels) is essentially arbitrary, there is clearly a case to be made for employing a model which includes local correlation in the ‘error’ terms. Chapter 6 covers methods for doing this, but for the moment we will proceed by treating the randomness as being independent between voxels, and letting all between voxel correlation be modelled by the trend. This is not simply a matter of convenience, but relates closely to the purpose of

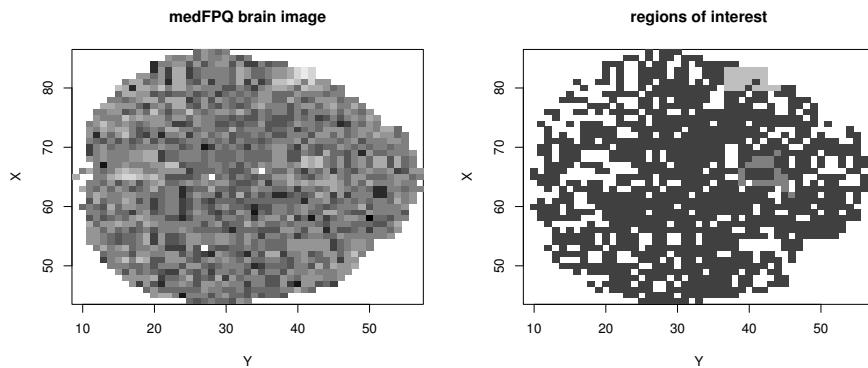


Figure 5.4 The raw data for the brain imaging data discussed in section 5.2. The left hand plot shows the median of 3 measurements of Fundamental Power Quotient values at each voxel making up this slice of the scan: these are the measurements of brain activity. The right hand panel shows the regions of interest: the ‘base region’ is the darkest shade, the region directly stimulated experimentally is shown in light grey and the region of experimental interest is shown as dark grey. Unclassified voxels are white.

the analysis: for these data the main interest lies in cleaning up this particular image, that is in removing the component of variability that appears to be nothing more than random variation, at the level of the individual voxel. The fact that the underlying mechanism generating features in the image, may include a component attributable to correlated noise across voxels, is only something that need be built into the model if the objective is to be able to remove this component of the pattern from the image, in addition to removing the un-correlated noise component.

5.2.1 Preliminary Modelling

An appropriate initial model structure would probably involve modelling `medFPQ` as a smooth function of `X` and `Y`, but before attempting to fit models, it is worth examining the data itself to look for possible problems. When this is done, 2 voxels appear problematic. These voxels have `medFPQ` values recorded as 3×10^{-6} and 4×10^{-7} , while the remaining 1565 voxels have values in the range 0.003 to 20. Residual plots from all attempts to model the data set, including these two voxels, consistently show them as grotesque outliers. For these reasons, these two voxels were excluded from the following analysis:

```
brain <- brain[brain$medFPQ>5e-3,] # exclude 2 outliers
```

The fairly skewed nature of the response data, `medFPQ`, and the fact that it is a necessarily positive quantity, suggest that some transformation may be required if a Gaussian error model is to be used. Attempting to use a Gaussian model without transformation confirms this:

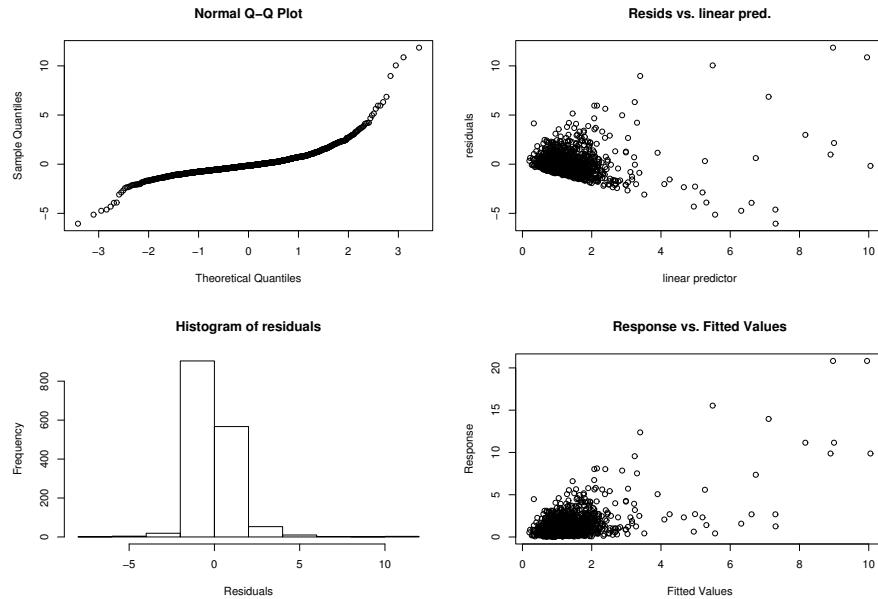


Figure 5.5 Some basic model checking plots for model $m0$ fitted to the brain scan data. The upper left normal QQ plot clearly shows a problem with the Gaussian assumption. This is unsurprising when the upper right plot of residuals versus fitted values (linear predictor) is examined: the constant variance assumption is clearly untenable. The lower left histogram of residuals confirms the pattern evident in the QQ plot: there are too many residuals in the tails and centre of the distribution relative to its shoulders. In this case the plot of response variable against fitted values, shown in the lower right panel, emphasizes the failure of the constant variance assumption.

```
> m0 <- gam(medFPQ ~ s(Y, X, k=100), data=brain)
> gam.check(m0)
Smoothing parameter selection converged after 6 iterations.
The RMS GCV score gradient at convergence was 6.460904e-06 .
The Hessian was positive definite.
The estimated model rank was 100 (maximum possible: 100)
```

`gam.check` is a routine that produces some basic residual plots, and a little further information about the success or otherwise of the fitting process. The plots produced for $m0$ are shown in figure 5.5. As explained in the caption of that figure, there are clear problems with the constant variance assumption: variance is increasing with the mean here. From the plots it is not easy to gauge the rate at which the variance of the data is increasing with the mean, but, in the absence of a good physical model of the mechanism underlying the relationship, some guidance can be obtained by a simple informal approach. If we assume that $\text{var}(y_i) \propto \mu_i^\beta$, where $\mu_i = \mathbb{E}(y_i)$ and β is some parameter, then a simple regression estimate of β can be obtained as follows:

```
> lm(log(e^2) ~ log(fv))

Call:
lm(formula = log(e^2) ~ log(fv))

Coefficients:
(Intercept)      log(fv)
-1.961          1.912
```

i.e. $\beta \approx 2$. That is, from the residuals of the simple fit, it appears that the variance of the data increases with the square of the mean. This in turn suggests using the Gamma distribution, which has this mean-variance relationship, or of treating the 4th root of medFPQ as the response for a Gaussian model (since such a transformation should approximately stabilize the variance, given the apparent mean-variance relationship on the original scale). With the Gamma model, it might also be appropriate to use a log link, in order to ensure that all model predicted FPQ values are positive.

The following fits models based first on transforming the data, and then on use of the Gamma distribution.

```
m1<-gam(medFPQ^.25~s(Y,X,k=100),data=brain)
gam.check(m1)
gm <- gam.method(gam="perf.magic")
m2<-gam(medFPQ~s(Y,X,k=100),data=brain,family=Gamma(link=log),
method=gm)
```

The plots from `gam.check` are shown in figure 5.6, and now show nothing problematic. `gam.check` plots for `m2` are equally good (but not shown). Note that, for this example, I have used the performance iteration to estimate smoothing parameters, rather than outer iteration. The two methods are unlikely to differ greatly, in this case, since for the Gamma distribution with a log link the iterative weights are simply 1: hence the computationally quicker performance iteration is preferable.

The major difference between `m1` and `m2` is in their biasedness on different scales. The model of the transformed data is approximately unbiased on the 4th root of the response scale (*approximately* because the variance stabilization can only be approximate): this means that it is biased downwards on the response scale itself. The log-Gamma model is approximately unbiased on the response scale (only approximately because maximum penalized likelihood estimation is not generally unbiased, but is consistent). This can be seen if we look at the mean of the fitted values (response scale) for the two models, and compare this to the mean of the raw data:

```
> mean(fitted(m1)^4);mean(fitted(m2));mean(brain$medFPQ)
[1] 0.985554 # m1 tends to under-estimate
[1] 1.212545 # m2 substantially better
[1] 1.250302
```

Clearly, if the response scale is the scale of prime interest, then the Gamma model is to be preferred to the a model based on normality of the transformed data. So far,

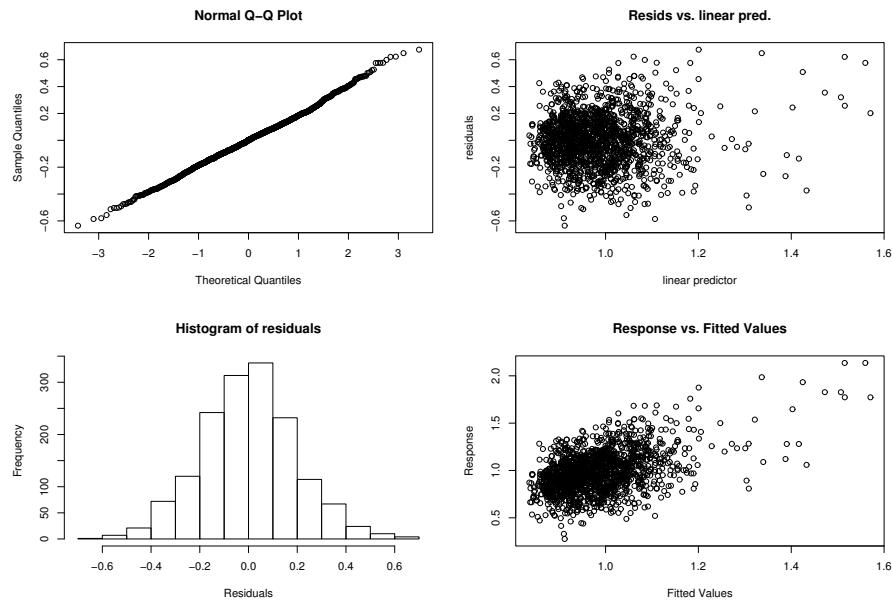


Figure 5.6 Some basic model checking plots for model $m1$ fitted to the transformed brain scan data. The upper left normal QQ plot is very close to a straight line, suggesting that the distributional assumption is reasonable. The upper right plot suggests that variance is approximately constant as the mean increases. The histogram of residuals at lower left appears approximately consistent with normality. The lower right plot of response against fitted values shows a positive linear relationship with a good deal of scatter: nothing problematic.

then, the best model seems to be the Gamma log-link model $m2$, which should be examined a little further.

```
> m2
Family: Gamma
Link function: log

Formula:
medFPQ ~ s(Y, X, k = 100)

Estimated degrees of freedom:
63.17224  total = 64.17224

GCV score: 0.5994399
> vis.gam(m2,plot.type="contour",too.far=0.03,
+ color="gray",n.grid=60,zlim=c(-1,2))
```

So a relatively complex fitted surface has been estimated, with 64 degrees of freedom. The function `vis.gam` provides quite useful facilities for plotting predictions from a `gam` fit against pairs of covariates, either as coloured perspective plots, or as coloured contour plots. The plot it produces for `m2` is shown in figure 5.7(a). Notice how the activity in the directly stimulated region and the region of interest stand out clearly in this plot.

5.2.2 Would an additive structure be better?

Given the large number of degrees of freedom, employed by the model `m2`, the question naturally arises of whether a different, simpler, model structure might achieve a more parsimonious fit. Since this is a book about GAMs, the obvious candidate is the additive model,

$$\log(\mathbb{E}[\text{medFPQ}_i]) = f_1(Y_i) + f_2(X_i), \quad \text{medFPQ}_i \sim \text{Gamma}.$$

```
> m3 <- gam(medFPQ ~ s(Y, k=30) + s(X, k=30), data=brain,
+   family=Gamma(link=log), method=gm)
> m3

Family: Gamma
Link function: log

Formula:
medFPQ ~ s(Y, k = 30) + s(X, k = 30)

Estimated degrees of freedom:
9.50822 19.56363  total = 30.07185

GCV score: 0.6817362
```

Clearly the GCV score is higher for this model, suggesting that it is not an improvement, and a comparison of explained deviances using `summary(m2)` and `summary(m3)` also suggests that the additive model is substantially worse.

It is also possible to test whether `m3` generated the data against the alternative that `m2` did using

```
> anova(m3, m2, test="F")
Analysis of Deviance Table

Model 1: medFPQ ~ s(Y, k = 30) + s(X, k = 30)
Model 2: medFPQ ~ s(Y, X, k = 100)
      Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
1  1533.928     970.91
2  1499.828     894.27     34.100     76.64 3.9098 5.727e-13 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

which suggests that the additive structure can be firmly rejected. Note that the test performed here involves three types of approximation (in addition to the usual use of asymptotic results). Firstly the test is based on treating the penalized fits as if they were un-penalized fits, with degrees of freedom given by the effective degrees of freedom of the models; secondly, the tests are conditional on smoothing parameters that have in fact been estimated; and thirdly, although, conceptually the smooth function $f_1(Y) + f_2(X)$ is nested within the smooth function $f(Y, X)$, this is not exactly true, given the bases actually used to represent the smooth functions. Given the clear cut nature of the rejection of the additive model, these approximations need not cause much concern here, but care would be needed if the p-value was anywhere near some pre-defined significance level for acceptance or rejection.

The fact that the models are not strictly nested can be addressed by replacing `m2` by

```
m4 <- gam(medFPQ~s(Y, k=30)+s(X, k=30)+s(Y, X, k=100), data=brain,
            family=Gamma(link=log), method=gm)
```

but the additive structure is still firmly rejected, if this is done.

Perhaps the most persuasive argument against the additive structure is provided by the plot of the predicted log activity levels provided in figure 5.7 (b): the additive structure produces horizontal and vertical ‘stripes’ in the plot that have no real support from the data.

5.2.3 Isotropic or tensor product smooths?

As discussed in chapter 4, isotropic smooths, of the sort produced by `s(Y, X)` terms, are usually good choices when the covariates of the smooth are naturally on the same scale, and we expect that the same degree of smoothness is appropriate with respect to both covariate axes. For the brain scan data these conditions probably hold, so the isotropic smooths are probably a good choice.

Nevertheless, it is worth checking what the results look like if we use a scale invariant tensor product smooth of `Y` and `X`, in place of the isotropic smooth (see section 4.1.8). Such smooths are computationally rather efficient (if the marginal bases of a tensor product smooth are cheap to construct, then so is the tensor product basis itself). Additionally, as discussed in section 4.10.2, there is always at least one additive model structure which is *strictly* nested within a tensor product smooth.

For example

```
tm<-gam(medFPQ~te(Y, X, k=10), data=brain, family=Gamma(link=log),
          method=gm)
tm1<-gam(medFPQ~s(Y, k=10, bs="cr") + s(X, bs="cr", k=10),
           data=brain, family=Gamma(link=log), method=gm)
```

fits a tensor product smooth to the FPQ data, storing the result in `tm`, and then fits a purely additive model to the same data, storing the result in `tm1`. The smooth in `tm`

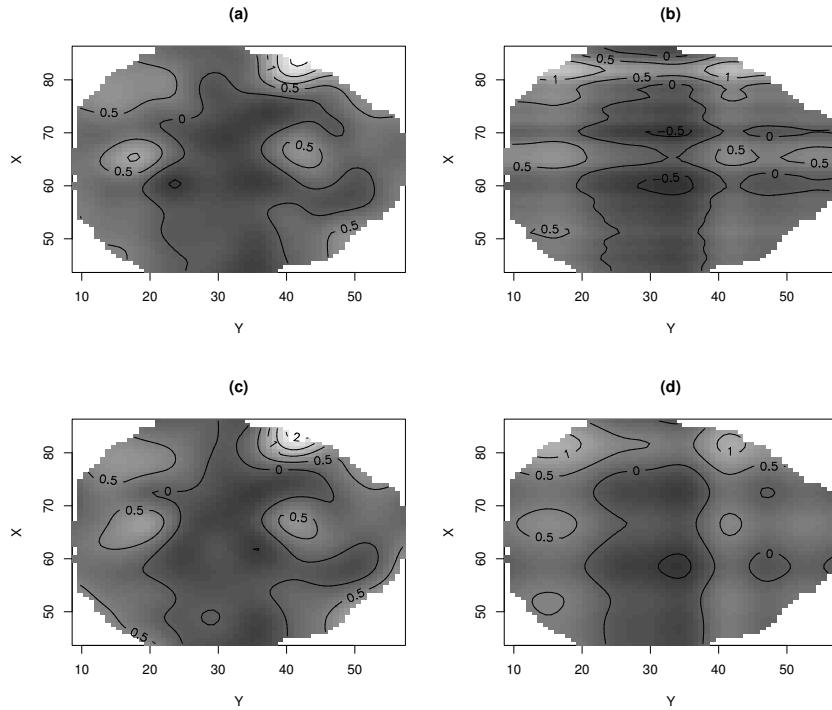


Figure 5.7 *Comparison of 4 models of the brain scan data. All plots show image plots and overlaid contour plots, on the scale of the linear predictor.* (a) is model m_2 based on an isotropic smooth of Y and X . (b) is model m_3 based on a sum of smooth functions of Y and X . Notice the apparent artefacts in (b), relating to the assumption of an additive structure. (c) plots model tm and is basically as (a), but using a rank 100 tensor product smooth in place of the isotropic smooth. (d) is for model tm_1 , which is as model m_3 except that the bases used ensure that this model is strictly nested within tm . All plots were produced with something like: `vis.gam(m2,plot.type="contour",too.far=0.03,color="gray",n.grid=60,zlim=c(-1,2),main="(a)")`

is a tensor product of two cubic regression spline bases, each of rank 10, and is hence of rank 100. The bases for the two rank 10 cubic regression splines, used in tm_1 , are also part of the tensor product basis used in tm , so tm_1 is strictly nested within tm , thereby removing one potential difficulty in model comparison.

As with the previous models, a comparison of GCV scores suggests that the additive model is not the best.

```
> tm1
[edited]
Estimated degrees of freedom:
 8.44507 8.081686  total = 17.52676
```

```
GCV score: 0.7390905
> tm
[edited]
Estimated degrees of freedom:
 57.88622 total = 58.88622
GCV score: 0.6166449
```

Similarly, an approximate test of the null hypothesis that the additive structure is appropriate, strongly suggests accepting `tm`.

```
> anova(tml,tm,test="F")
Analysis of Deviance Table

Model 1: medFPQ~s(Y, k=10, bs="cr") + s(X, bs="cr", k=10)
Model 2: medFPQ~te(Y, X, k=10)
Resid. Df Resid. Dev          Df Deviance      F     Pr(>F)
1   1546.473    1004.34
2   1505.114     906.04    41.359    98.31 4.0053 9.882e-16 ***
```

Plots of `tm` and `tml` are shown in figure 5.7 (c) and (d). The plot for `tm` (c) is rather similar to the plot for `m2`, while once again the additive model, `tml`, produces a plot with horizontal and vertical stripes, which are almost certainly artefacts (the stripes are less pronounced than for `m3` because `tml` has been forced to be a smoother model).

5.2.4 Detecting symmetry (with by variables)

It is sometimes of interest to test whether the image underlying some noisy data is symmetric, and this is quite straightforward to accomplish, with the methods covered here. For the brain data we might want to test whether the underlying activity levels are symmetric about the mean `X` value of 64.5. The symmetry model in this case would be

$$\log(\mathbb{E}[\text{medFPQ}_i]) = f(Y_i, |X_i - 64.5|), \quad \text{medFPQ}_i \sim \text{Gamma}$$

and this could be compared with model `m2`, which does not assume symmetry, although for strict nested models we would need to be slightly more sophisticated and use an asymmetry model with a form such as,

$$\log(\mathbb{E}[\text{medFPQ}_i]) = f(Y_i, |X_i - 64.5|) + f_r(Y_i, |X_i - 64.5|).\text{right}_i,$$

where f_r is represented using the same basis as f , and right_i is a dummy variable, taking the values 1 or 0, depending on whether medFPQ_i is from the right or left side of the brain.

The following code creates variables required to fit these two model, estimates them, and prints the results.

```
> brain$Xc <- abs(brain$X - 64.5)
```

```

> brain$right <- as.numeric(brain$X<64.5)
> m.sy <- gam(medFPQ~s(Y,Xc,k=100),data=brain,
+ family=Gamma(link=log),method=gm)
> m.as <- gam(medFPQ~s(Y,Xc,k=100)+s(Y,Xc,k=100,by=right),
+ data=brain,family=Gamma(link=log),method=gm)
> m.sy
[edited]
Estimated degrees of freedom:
 52.54246  total = 53.54246
GCV score: 0.6498528
> m.as
[edited]
Estimated degrees of freedom:
 59.80448 46.20071  total = 107.0052
GCV score: 0.5967997

```

The GCV scores suggest that the asymmetric model is better. Approximate hypothesis testing can proceed either by comparing the two models using an F test (see section 4.10.1), or by the Wald testing approach (section 4.8.5):

```

> anova(m.sy,m.as,test="F")
Analysis of Deviance Table

Model 1: medFPQ~s(Y,Xc,k=100)
Model 2: medFPQ~s(Y,Xc,k=100)+s(Y,Xc,k=100,by=right)
  Resid. Df Resid. Dev      Df Deviance    F   Pr(>F)
1  1510.458     946.71
2  1456.995     857.57     53.463     89.14 2.999 8.677e-12 ***
> anova(m.as)

Family: Gamma
Link function: log

Formula:
medFPQ ~ s(Y, Xc, k = 100) + s(Y, Xc, k = 100, by = right)

Approximate significance of smooth terms:
          edf Est.rank    F p-value
s(Y,Xc)      59.8      99.0 4.172 < 2e-16
s(Y,Xc):right 46.2      99.0 2.104 6.19e-09

```

The two approaches are in agreement that there is very strong evidence against symmetry, but the approximate p-values are not in very close numerical agreement. This is unsurprising, given that each is obtained from rather extreme tails of different distributional approximations. At the cost of another ‘nesting approximation’ we could also have compared `m.sy` with `m2`, in which case the reported p-value is of the order of 10^{-14} .

Note that, as mentioned in sections 4.8.5 and 4.10.1, the p-values produced by this

analysis ignore the fact that we have selected the model effective degrees of freedom. As a result, the distribution of the p-values, under the null, deviates a little from the uniform distribution that p-values should follow. This problem can be fixed, at the cost of some loss of power, by working with un-penalized models for hypothesis testing purposes. Using model terms of the form `s(..., fx=TRUE)` is the way to achieve this. If the above analysis is repeated without penalization the p-values change somewhat, but, unsurprisingly, still firmly reject the null hypothesis of symmetry. It is important to be aware that the un-penalized approach is not a panacea: more care is needed in the choice of the `k` argument of `s()` when it is used, since excessively high `k` will lead to low testing power. Also, penalization definitely is required for CI calculation and point estimation.

Plots of the different model predictions help to show why symmetry is so clearly rejected (figure 5.8).

```
vis.gam(m.sy,plot.type="contour",view=c("Xc","Y"),too.far=.03,
        color="gray",n.grid=60,zlim=c(-1,2),main="both sides")
vis.gam(m.as,plot.type="contour",view=c("Xc","Y"),
        cond=list(right=0),too.far=.03,color="gray",n.grid=60,
        zlim=c(-1,2),main="left side")
vis.gam(m.as,plot.type="contour",view=c("Xc","Y"),
        cond=list(right=1),too.far=.03,color="gray",n.grid=60,
        zlim=c(-1,2),main="right side")
```

5.2.5 Comparing two surfaces

The symmetry testing, considered in the last section, is an example of comparing surfaces — in that case one half of an image, with a mirror image of the other half. In some circumstances, it is also interesting to compare completely independent surfaces in a similar way. To see how this might work, a second set of brain scan data can be simulated, using the fitted model `m2`, and perturbed, somewhat, as follows.

```
brain1 <- brain
mu <- fitted(m2)
n<-length(mu)
ind <- brain1$X<60 & brain1$Y<20
mu[ind] <- mu[ind]/3
set.seed(1)
brain1$medFPQ <- rgamma(rep(1,n),mu/m2$sig2,scale=m2$sig2)
```

Now the data sets can be combined, and dummy variables created to identify which dataset each row of the combined data frame relates to.

```
brain2=rbind(brain,brain1)
brain2$sample1 <- c(rep(1,n),rep(0,n))
brain2$sample0 <- 1 - brain2$sample1
```

After which it is straightforward to fit a model with a single combined surface for

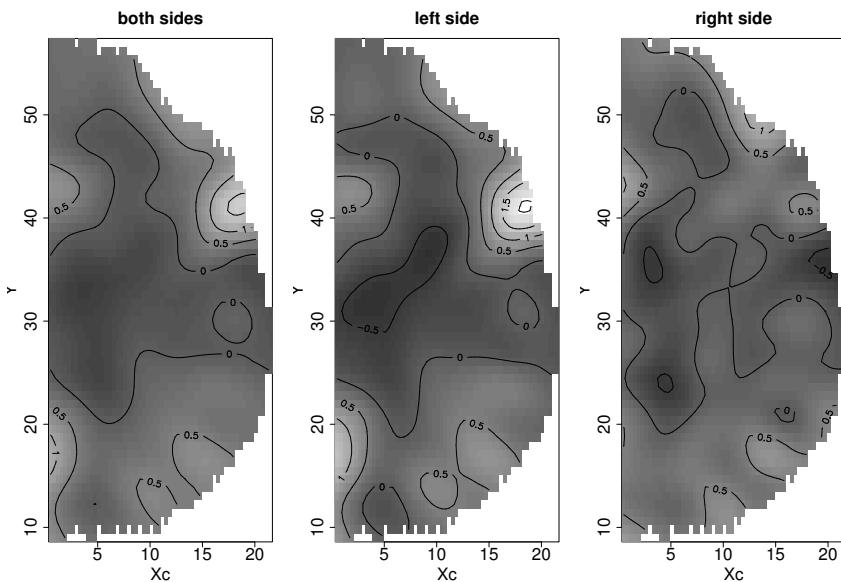


Figure 5.8 *Half brain images for the two models involved in examining possible symmetry in the brain image data. The left panel shows the predictions of log activity for the symmetry model: the left side of the image would be a mirror image of this plot. The middle plot shows a mirror image of the left side brain image under the asymmetry model, while the right plot shows the right side image under the asymmetry model. Clearly the asymmetry model suggests rather different activity levels in the two sides of the image.*

both data sets, and a second model where the surfaces are allowed to differ. Note that in the latter case a single common surface is estimated, with a difference surface for the second sample. This approach is often preferable to a model with two completely separate surfaces, for reasons of parsimony. If there is no difference between the surfaces, it will still be necessary to estimate two quite complex surfaces, if we adopt separate surfaces for each data set. With the common surface plus difference approach, only one complex surface need be estimated: the difference being close to flat. Similarly, any time that the surfaces are likely to differ in a fairly simply way, it makes sense to use the common surface plus difference model, thereby reducing the number of degrees of freedom needed by the model.

```
m.same<-gam(medFPQ~s(Y,X,k=100), data=brain2,
              family=Gamma(link=log), method=gm)
m.diff<-gam(medFPQ~s(Y,X,k=100)+s(Y,X,by=sample1,k=100),
             data=brain2, family=Gamma(link=log), method=gm)
```

Examination of the GCV scores for the two models suggests that the second model `m.diff` is slightly preferable to `m.same`, as do the AIC values, and an approximate test of the hypothesis that a single surface is correct, against the two surface alternative, tends to confirm this:

```
> anova(m.same,m.diff,test="F")
Analysis of Deviance Table

Model 1: medFPQ~s(Y,X,k=100)
Model 2: medFPQ~s(Y,X,k=100)+s(Y,X,by=sample1,k=100)
      Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
1   3058.646   1937.36
2   3044.218   1906.11   14.428     31.25 3.7877 1.575e-06 ***
```

Repeating this analysis without penalization still suggests rejecting the null of no difference, but with a much increased p-value. This emphasizes the price paid for the well behaved p-values that come from not penalizing. The fact that the difference term is estimated to have only 14 degrees of freedom in the penalized fit means that using a basis dimension of 100 in the un-penalized fit was really excessive: and this degree of over-specification leads to greatly reduced power. Repeating the analysis again with a basis dimension of 50 for the difference smooth gives a much reduced p-value, but of course this p-value is no longer strictly valid, since the model has been modified in the light of an initial fit.

In this example, the data for the two surfaces was available on exactly the same regular mesh, but the approach is equally applicable for irregular data where the data are not at the same covariate values for the two surfaces (although of course the regions of covariate space covered should overlap, for a comparison of this sort to be meaningful).

5.2.6 Prediction with predict.gam

The `predict` method function, `predict.gam`, enables a `gam` fitted model object to be used for prediction at new values of the model covariates. It is also used to provide estimates of the uncertainty of those predictions, and the user can specify whether predictions should be made on the scale of the response or of the linear predictor. For predictions on the scale of the linear predictor, `predict.gam` also allows predictions to be decomposed into their component terms, and it is also possible to extract the matrix, which when multiplied by the model parameter vector, yields the vector of predictions at the given set of covariate values.

Usually the covariate values, at which predictions are required, are supplied as a data frame in argument `newdata`, but if this argument is not supplied then predictions/fitted values are returned for the original covariate values, used for model estimation. Here are some examples. Firstly on the scale of the linear predictor.

```
> predict(m2)[1:5]
      1         2         3         5         6
0.3024547 0.3418227 0.3474573 0.2769854 0.4541785
> pv <- predict(m2,se=TRUE)
> pv$fit[1:5]
      1         2         3         5         6
```

```
0.3024547 0.3418227 0.3474573 0.2769854 0.4541785
> pv$se[1:5]
      1          2          3          5          6
0.2640762 0.2164298 0.2158971 0.2237674 0.2275705
```

and then on the response scale

```
> predict(m2,type="response") [1:5]
      1          2          3          5          6
1.353176 1.407511 1.415464 1.319147 1.574879
> pv <- predict(m2,type="response",se=TRUE)
> pv$se[1:5]
      1          2          3          5          6
0.3573416 0.3046273 0.3055945 0.2951821 0.3583961
```

For both sets of examples, predictions are produced for all 1564 voxels, but I have only printed the first 5. Note that the standard errors provided on the response scale are approximate, being obtained by the usual Taylor expansion approach.

Usually, predictions are required for new covariate values, rather than simply for the values used in fitting. Suppose, for example, that we are interested in two points in the brain scan image, firstly in the directly stimulate area, ($X = 80.1, Y = 41.8$), and secondly in the region of experimental interest, ($X = 68.3, Y = 41.8$). A data frame can be created, containing the covariate values for which predictions are required, and this can be passed to `predict.gam`, as the following couple of examples show.

```
> pd <- data.frame(X=c(80.1,68.3),Y=c(41.8,41.8))
> predict(m2,newdata=pd)
      1          2
1.2931442 0.6116455
> predict(m2,newdata=pd,type="response",se=TRUE)
$fit
      1          2
3.644227 1.843462
$se.fit
      1          2
0.5444432 0.2693135
```

It is also possible to obtain the contributions that each model term, excluding the intercept, makes to the linear predictor, as the following example shows. The additive model `m3` has been used to illustrate this, since it has more than one term.

```
> predict(m3,newdata=pd,type="terms",se=TRUE)
$fit
      s(Y)      s(X)
1 0.2496231 0.521442293
2 0.2496231 -0.007236053

$se.fit
      s(Y)      s(X)
1 0.05841173 0.09677576
2 0.05841173 0.08109702
```

As you can see, named arrays have been returned, the columns of which correspond to each model term. There is one array for the predicted values, and a second for the corresponding standard errors.

Prediction with lpmatrix

Because the GAMs discussed in this book have an underlying parametric representation, it is possible to obtain a ‘prediction matrix’, \mathbf{X}_p , say, which maps the model parameters, $\hat{\beta}$, to the predictions of the linear predictor, $\hat{\eta}_p$, say. That is, to find the \mathbf{X}_p such that

$$\hat{\eta}_p = \mathbf{X}_p \hat{\beta}.$$

`predict.gam` can return \mathbf{X}_p , if its `type` argument is set to "lpmatrix".

The following example illustrates this. Since the returned matrix is of dimension 2×101 , it has not been printed out, but rather it is demonstrated that it does indeed give the required linear predictor values, when multiplied by the coefficients of the fitted model.

```
> Xp <- predict(m2, newdata=pd, type="lpmatrix")
> fv <- Xp %*% coef(m2)
> fv
[1]
1 1.2931442
2 0.6116455
```

Why is \mathbf{X}_p useful? A major use, is in the calculation of variances for combinations of linear predictor values. Clearly, if $\hat{\mathbf{V}}_\beta$ is the estimate of the parameter covariance matrix, then from standard probability theory, the estimated covariance matrix of $\hat{\eta}_p$ must be:

$$\hat{\mathbf{V}}_{\eta_p} = \mathbf{X}_p \hat{\mathbf{V}}_\beta \mathbf{X}_p^T.$$

Now suppose that we are really interested in, for example, the difference, δ , between the linear predictor values at the points in the two regions. This difference could be written as, $\delta = \mathbf{d}^T \hat{\eta}_p$, where $\mathbf{d}^T = [1, -1]^T$. In that case, standard theory says that:

$$\widehat{\text{var}}(\delta) = \mathbf{d}^T \hat{\mathbf{V}}_{\eta_p} \mathbf{d} = \mathbf{d}^T \mathbf{X}_p \hat{\mathbf{V}}_\beta \mathbf{X}_p^T \mathbf{d}$$

The following code illustrates this.

```
> d<-t(c(1, -1))
> d%*%fv
[1]
[1,] 0.6814987
> d%*%Xp%*%m2$Vp%*%t(Xp)%*%t(d)
[1]
[1,] 0.04321413
```

So, the ability to obtain an explicit ‘predictor matrix’, makes some variance calculations rather straightforward. Of course the neat linear theory, facilitating these calculations, is only applicable if we are interested in inference about linear functions

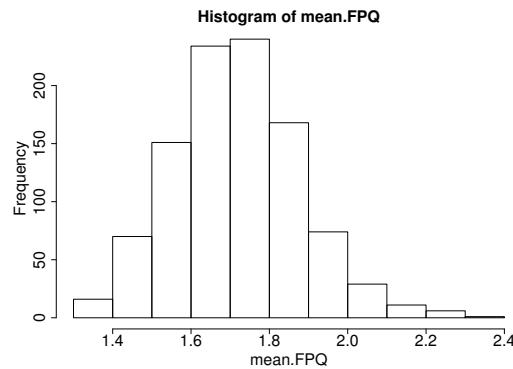


Figure 5.9 *Histogram of 1000 draws from the posterior distribution of average medFPQ in the region of experimental interest in the brain scan example.*

of the linear predictor. As soon as the functions of interest become non-linear, as is generally the case when working on the response scale, we need different methods, which are covered next.

5.2.7 Variances of non-linear functions of the fitted model

Prediction matrices, in conjunction with simulation from the posterior distribution of the parameters, β , give a simple and general method for obtaining variance estimates (or indeed distribution estimates), for any quantity derived from the fitted model, not simply those quantities that are linear in β . The idea of this sort of posterior simulation was discussed in section 4.8.4.

As an example, suppose that we would like to estimate the posterior distribution of the average medFPQ, in the region of experimental interest. Since the quantity of interest is on the response scale, and not the scale of the linear predictor, it is not linear in β , and the simple approach, taken at the end of the last section, is not applicable.

To approach this problem, a prediction matrix is first obtained, which will map the parameters to the values of the linear predictor that will be needed to form the average.

```
ind <- brain$region==1 & ! is.na(brain$region)
Xp <- predict(m2,newdata=brain[ind,],type="lpmatrix")
```

Next, a large number of replicate parameter sets are simulated from the posterior distribution of β , using the `mvrnorm` function from the MASS library.

```
library(MASS)
br <- mvrnorm(n=1000,coef(m2),m2$Vp) # simulate from posterior
```

Each column of the matrix \mathbf{br} , is a replicate parameter vector, drawn from the posterior distribution of β . Given these replicate parameter vectors, and the matrix \mathbf{X}_p , it is a simple matter to obtain the linear predictor implied by each replicate, from which the required averages can easily be obtained, as follows.

```
mean.FPQ<-rep(0,1000)
for (i in 1:1000)
{ lp <- Xp%*%br[i,] # replicate linear predictor
  mean.FPQ[i] <- mean(exp(lp)) # replicate region 1 mean FPQ
}
```

or more efficiently, but less readably

```
mean.FPQ <- colMeans(exp(Xp%*%t(br)))
```

So `mean.FPQ` now contains 1000 replicates from the posterior distribution of the mean FPQ measurement in region 1. The results of `hist(mean.FPQ)` are shown in figure 5.9.

Clearly, this simulation approach is rather general: samples from the posterior distribution of any quantity that can be predicted from the fitted model, are rather easily obtained. Notice also that, in comparison to bootstrapping, the approach is very computationally efficient. The only real disadvantage is that the results are conditional on the estimated smoothing parameters, but this is something that can be addressed using the ideas discussed in section 4.9.3: a practical implementation of that approach is given in section 5.4.2.

5.3 Air Pollution in Chicago Example

The relationship between air-pollution and health is a moderately controversial topic, and there is a great deal of epidemiological work attempting to elucidate the links. In this section a variant of a type of analysis that has become quite prevalent in air-pollution epidemiology is presented. The data are from Peng and Welty (2004) and are contained in a data frame `chicago`. The response of interest is the daily death rate in Chicago, `death`, over a number of years. Possible explanatory variable for the observed death rate are levels of ozone, `o3median`, levels of sulphur dioxide, `so2median`, mean daily temperature, `tmpd`, and levels of particulate matter, `pm10median`, (as generated by diesel exhaust, for example). In addition to these air quality variables, the underlying death rate tends to vary with `time` (in particular throughout the year), for reasons having little or nothing to do with air quality.

A conventional approach to modelling these data would be to assume that the observed numbers of deaths are Poisson random variables, with an underlying mean that is the product of a basic, time varying, death rate, modified through multiplication by pollution dependent effects. That is, the model would be

$$\begin{aligned} \log(\mathbb{E}[\text{death}_i]) = f(\text{time}_i) + \beta_1 \text{pm10median}_i + \beta_2 \text{so2median}_i \\ + \beta_3 \text{o3median}_i + \beta_4 \text{tmpd}_i, \end{aligned}$$

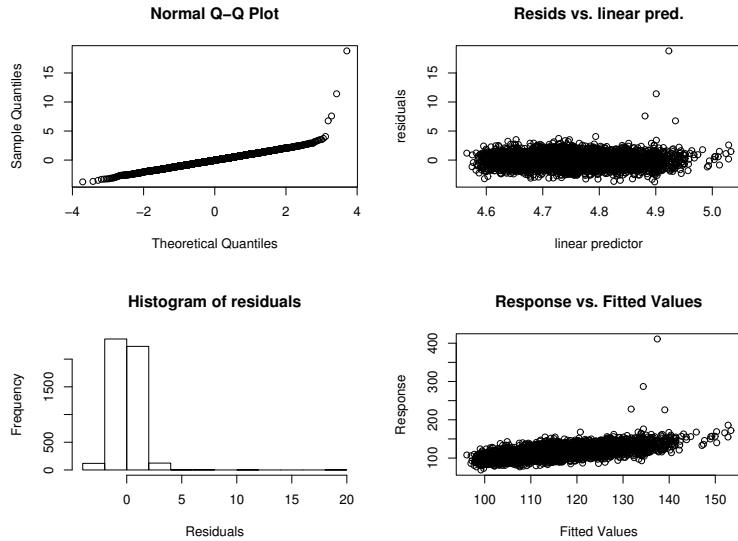


Figure 5.10 *Basic model checking plots for the ap0 air pollution mortality model. For Poisson data with moderately high means the distribution of the standardized residuals should be quite close to normal, so that the QQ-plot is obviously problematic. As all the plots make clear there are a few gross outliers that are very problematic in this fit.*

where death_i follows a Poisson distribution, and f is a smooth function.

The model is easily fitted and checked.

```
ap0 <- gam(death~s(time,bs="cr",k=200)+pm10median+so2median+
o3median+tmpd,data=chicago,family=poisson)
gam.check(ap0)
```

A cubic regression spline has been used for f , since 5000 observations is too many for straightforward use of the TPRS basis. The checking plots are shown in figure 5.10, and show clear problems as a result of some substantial outliers. Plotting the estimated smooth, with and without partial residuals, emphasizes the size of the outliers.

```
par(mfrow=c(2,1))
plot(ap0,n=1000) # n increased to make plot smooth
plot(ap0,residuals=TRUE,n=1000)
```

The plots are shown in figure 5.11, and 4 gross outliers, in close proximity to each other, are clearly visible. Examination of the data indicates that the outliers are the four highest daily death rates occurring in the data, and that they occurred on consecutive days,

```
> chicago$death[3111:3125]
[1] 112 97 122 119 116 121 226 411 287 228 159 142 123 102 94
```

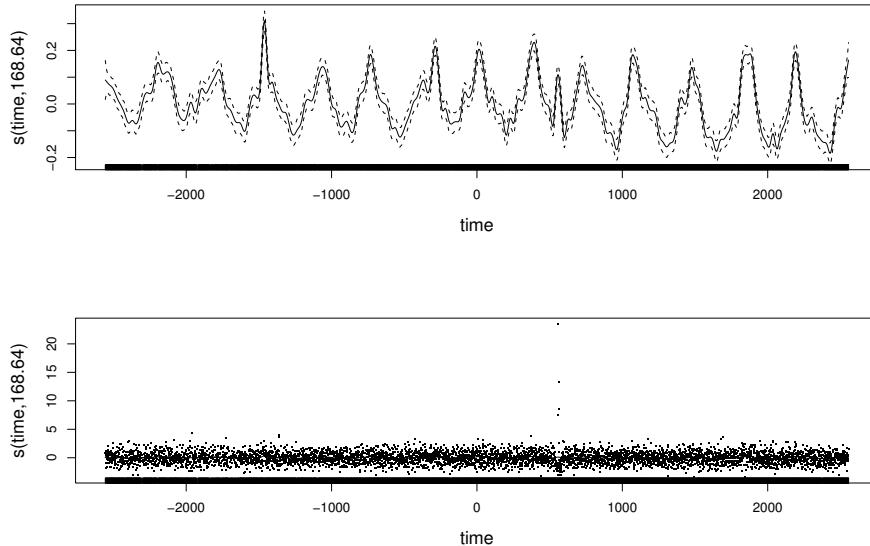


Figure 5.11 *The estimate of the smooth from model ap0 shown with and without partial residuals. Note the 4 enormous outliers apparently in close proximity to each other.*

Plotting this section of data also indicates that this peak is associated with a period of very high temperatures and high ozone. One immediate possibility is that the model is simply too inflexible, and that some non-linear response of death rate to temperature and ozone is required. This might suggest replacing the linear dependencies on the air quality covariates, with smooth functions, so that the model structure becomes:

$$\log(\mathbb{E}[\text{death}_i]) = f_1(\text{time}_i) + f_2(\text{pm10median}_i) + f_3(\text{so2median}_i) \\ + f_4(\text{o3median}_i) + f_5(\text{tmpd}_i)$$

where the f_j are smooth functions. This model is easily fitted

```
ap1<-gam(death~s(time,bs="cr",k=200)+s(pm10median,bs="cr")+
  s(so2median,bs="cr")+s(o3median,bs="cr")+s(tmpd,bs="cr"),
  data=chicago,family=poisson)
```

but the `gam.check` plots are almost indistinguishable from those shown in figure 5.10. Figure 5.12 shows the estimated smooths, and indicates a problem with the distribution of `pm10median` values, in particular, which might be expected to cause leverage problems. Similar plots, with partial residuals, again indicate a severe failure to fit the 4 day run of record death rates.

More detailed examination of the data, surrounding the 4 day mortality surge, shows

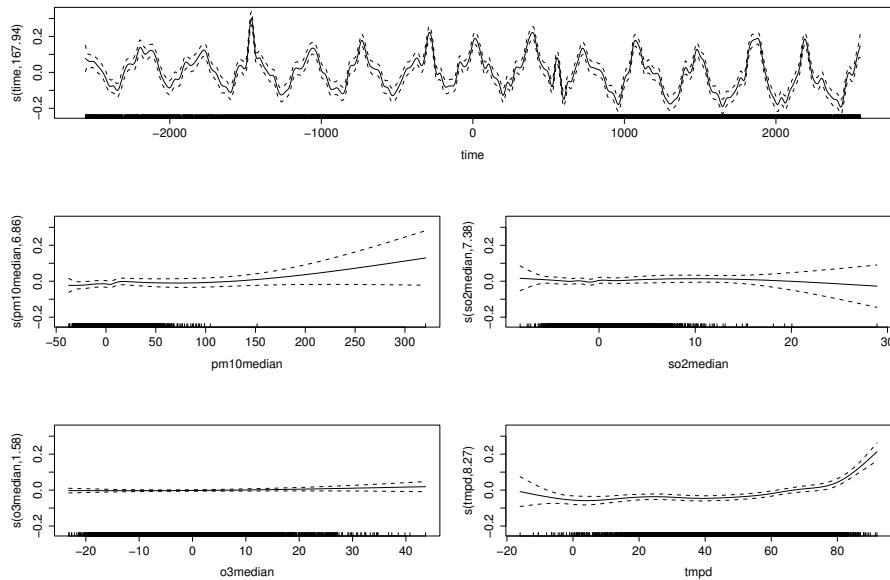


Figure 5.12 *The estimate of the smooth from model ap1 shown without partial residuals. Note the gap in the pm10median values. Similar plots with partial residuals highlight exactly the same gross outliers as were evident in figure 5.11.*

that the highest temperatures in the temperature record where recorded in the few days preceding the high mortalities, when there were also high ozone levels recorded. This suggests that average temperature and pollution levels, over the few days preceding a given mortality rate, might better predict it than the temperature and levels only on the day itself. On reflection, such a model might be more sensible on biological/medical grounds: the pollution levels and temperatures recorded in the data are no where near high enough to cause immediate acute disease and mortality, and it seems more plausible that any effects would take some time to manifest themselves via, for example, the aggravation of existing medical conditions.

The high mortality episode provides a useful way of trying to identify appropriate aggregations and lags for the predictor variable. If the extremely high mortalities are related to the pollutants, then, presumably, these high mortalities lie at the extreme of some combination of predictors. Some experimentation with aggregating the predictor variables in different ways suggests that the sum (or mean) of the predictors on the day in question, and the three preceding days, might be appropriate. Figure 5.13 illustrates this.

Given these considerations the following simple function was used to sum the pollutant and temperature levels over the 4 days up to and including each daily mortality reading. Corresponding portions of the time and death rate vectors were also selected.

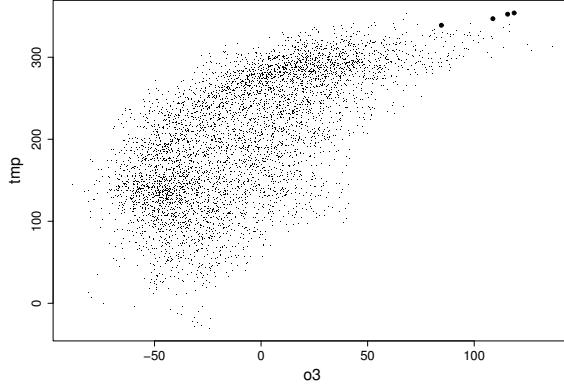


Figure 5.13 Plots of all observed combinations of temperature and ozone, summed over 4 day periods. The large symbols show the 4 day periods ending in each of the 4 extreme mortality events. The suggestion is that a combination of persistent high temperatures and high ozone over several days might lead to higher mortality risk

```

lag.sum <- function(a,10,11)
## 10 is the smallest lag, 11 the largest
{ n<-length(a)
  b<-rep(0,n-11)
  for (i in 0:(11-10)) b <- b + a[(i+1):(n-11+i)]
  b
}
death<-chicago$death[4:5114]
time<-chicago$time[4:5114]
o3 <- lag.sum(chicago$o3median,0,3)
tmp <- lag.sum(chicago$tmpd,0,3)
pm10 <- lag.sum(log(chicago$pm10median+40),0,3)
so2 <- lag.sum(log(chicago$so2median+10),0,3)

```

So, for example, the aggregate variable,

$$o3_i = \sum_{j=i-3}^i o3\text{median}_j$$

is to be used as the predictor for death_i , with similar definitions for the other predictor variables (except `time`, of course).

Given the suggestion, from examination of the data, that a combination of high ozone and high temperature might lead to very high death rates, the following model was tried next,

$$\log(\mathbb{E}[\text{death}_i]) = f_1(\text{time}_i) + f_2(o3_i, \text{tmp}_i, \text{pm10}_i),$$

where f_1 and f_2 are smooth functions. Isotropy is probably not an appropriate assumption for f_2 , so it was represented by a tensor product smooth as follows

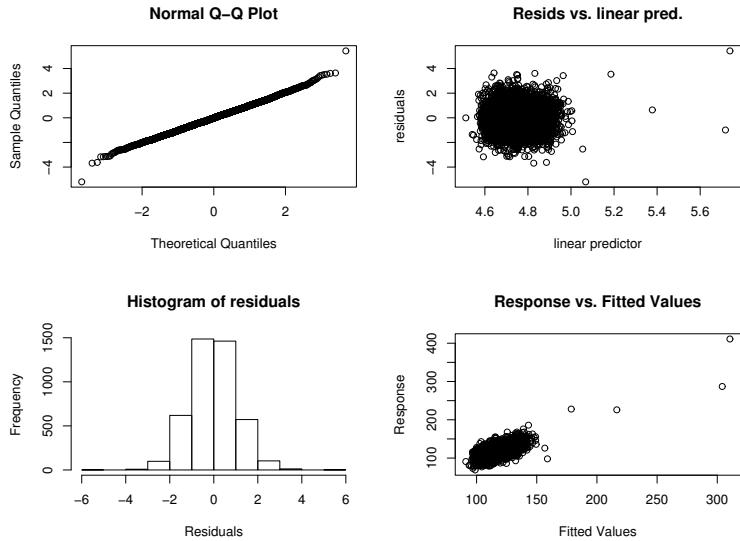


Figure 5.14 Basic checking plots for model ap2. Note the substantial improvement in comparison to Figure 5.10.

```
ap2 <- gam(death ~ s(time, bs="cr", k=200) +
              te(o3, tmp, pm10, k=c(8, 8, 6)), family=poisson)
```

(This model is slow to converge, and fails to converge altogether if performance iteration is used).

The default checking plots, for this model, are shown in figure 5.14, and are now greatly improved. Having reached the stage of having a model that is not obviously wrong, it is worth proceeding to see if it can be simplified, and one obvious simplification to try is,

$$\log(\mathbb{E}[\text{death}_i]) = f_1(\text{time}_i) + f_2(o3_i, \text{tmp}_i) + f_3(pm10_i).$$

```
ap3 <- gam(death ~ s(time, bs="cr", k=200) + te(o3, tmp, k=8) +
              s(pm10, bs="cr", k=6), family=poisson)
```

fits the model, and the `gam.check` plots are very similar to figure 5.14. By default, smoothness selection for these models has been performed using UBRE, and the simpler model, ap3, has a slightly lower UBRE score, suggesting that it is to be preferred. In this case UBRE is effectively AIC, so a comparison of AIC scores leads to the same conclusion (although in this example approximate hypothesis testing points in the other direction). Further simplification, to an additive structure in smooths of each covariate separately, worsens the fit quite radically, since the model is again unable to fit the peak death rates. Adding `so2` improves the model marginally, in that the UBRE score is reduced a little, but the estimated effect is of small magnitude,

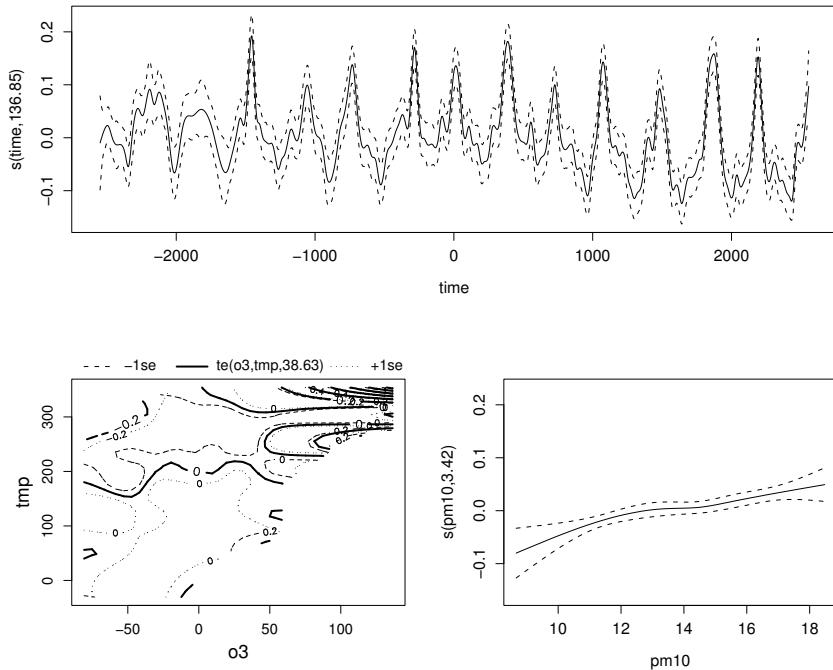


Figure 5.15 A reasonable model for the Chicago air-pollution mortality data. The panels show the estimates of the terms in model ap3. The upper panel is the smooth function of time; the lower left panel is the smooth function of aggregated lagged temperature and ozone; the lower right panel is the function of (transformed) aggregated lagged particulate matter.

with only one degree of freedom and a confidence interval that barely excludes 0. Further experimentation with alternative additive decompositions only worsens the model fit.

Estimates of the components of model ap3 are shown in figure 5.15. Clearly the notion that several days of high ozone and temperature can cause elevated death rates can explain these data, but given the way that the data have been used to develop a model, we would really need to see if the same model works well in other locations, or time periods, before giving it too much credence.

5.4 Mackerel egg survey example

World wide, most commercially exploitable fish stocks are over exploited, with some stocks, such as Newfoundland Cod, having famously collapsed. Effective management of stocks rests on sound fish stock assessment, but this is not easy to achieve.

The main difficulty is that counting the number of catchable fish of any given species is all but impossible. A standard statistical sampling approach, based on trying to catch fish, fails because, for large mobile organisms, there is no way of relating what is caught to what was available to catch, in a given area. To some extent, surveys with sonar avoid such catchability problems, but suffer from other problems, chiefly that it is often impossible to determine which fish species cause a given sonar signal. Another alternative is to attempt to reconstruct past fish stocks, on the basis of records of what fish get landed commercially, an approach known as ‘virtual population analysis’, but this suffers from the problem that it tells you quite accurately what the stock was several years ago, but is rather imprecise about its current or recent state.

Egg production methods are a very different means of assessing stocks. The basic idea is to try and assess the number of eggs produced by a stock, or the rate at which a stock is producing eggs, and then to work out the number (or more often mass) of adult fish required to produce this number or production rate. This works because egg production rates per kg of adult fish *can* be assessed from adults caught in trawls, while eggs can be sampled in an unbiased manner. Egg data are obtained, over the area occupied by a stock, by sending out scientific cruise ships to sample eggs, at each station of some predefined sampling grid. Eggs are usually sampled by hauling a fine meshed net up from the sea bed to the sea surface (or at least from well below the depth at which eggs are expected to the surface). The number of eggs, of the target species, in the sample is then counted, and, of course, the volume of water sampled is known.

5.4.1 Model development

To get the most out of the egg data, it is helpful to model the egg distribution, and here GAMs can be useful. The example considered in this section concerns data from a 1992 mackerel egg survey. The data were first modelled using GAMs by Borchers et al. (1997), and first entered the public domain as data sets `mack` and `smacker` in the `sm` library of Bowman and Azzalini (1997). Here they have been combined into one data set `mack`. The left hand panel of figure 5.16 shows the location at which samples were taken, and the egg densities found there. As well as longitude and latitude, a number of other possible predictors of egg abundance are available: salinity; surface temperature of the ocean, `temp.surf`; water temperature at a depth of 20m, `temp.20m`; depth of the ocean, `b.depth`; and finally, distance from the 200m seabed contour, `c.dist`. The latter predictor reflects the biologists’ belief that the fish like to spawn near the edge of the continental shelf, conventionally considered to end at a seabed depth of 200m. At each sampling location, a net was hauled vertically through the water column from well below the depth at which eggs are found, to the surface: the mackerel eggs caught in the net were counted, to give the response variable, `egg.count`.

The Poisson distribution might be a reasonable model for the egg counts, with a mean

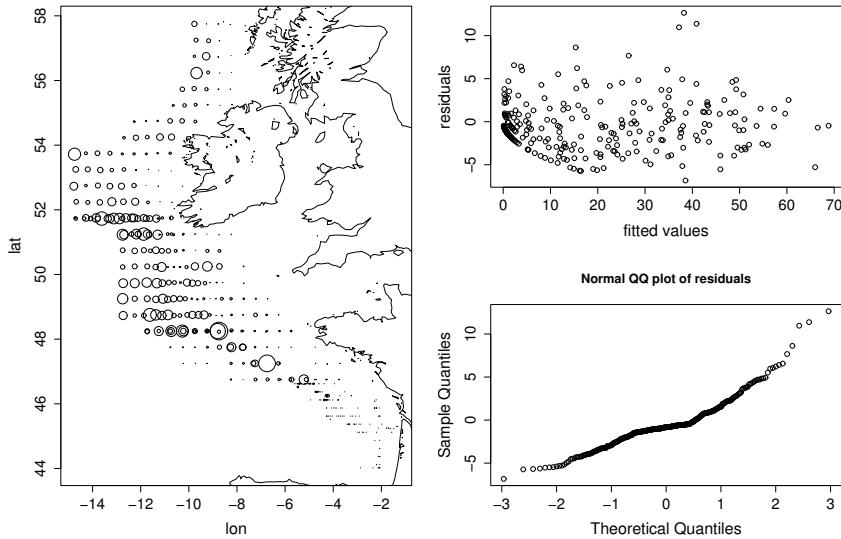


Figure 5.16 The left hand plot shows the density of stage I mackerel eggs per square metre of sea surface as assessed by net samples in the 1992 Mackerel egg survey. Circle areas are proportional to egg density and are centered on the location at which the egg samples were obtained. The right hand plots show residual plots for the gm GAM fitted to these data.

given by

$$\mathbb{E}[\text{egg.count}_i] = g_i \times [\text{net area}]_i$$

where g_i is the density of eggs, per square metre of sea surface, at the i^{th} sampling location. Taking logs of this equation we get,

$$\log(\mathbb{E}[\text{egg.count}_i]) = f_i + \log([\text{net area}]_i)$$

where $f_i = \log(g_i)$ will be modelled as a function of predictor variables, using an additive structure, and $\log([\text{net area}]_i)$ will be treated as a model ‘offset’: that is, as a column of the model matrix with associated parameter fixed at 1.

For the purposes of this exercise, a simple additive structure will be assumed, with the first model being fitted as follows:

```
mack$log.net.area <- log(mack$net.area)

gm <- gam(egg.count ~ s(lon, lat, bs="ts") + s(I(b.depth^.5), bs="ts")
          + s(c.dist, bs="ts") + s(salinity, bs="ts") + s(temp.surf, bs="ts")
          + s(temp.20m, bs="ts") + offset(log.net.area),
          data=mack, family=poisson, scale=-1, gamma=1.4)
```

Here, shrinkage smoothers have been used, which are constructed in such a way that smooth terms can be penalized away altogether, making no contribution to the

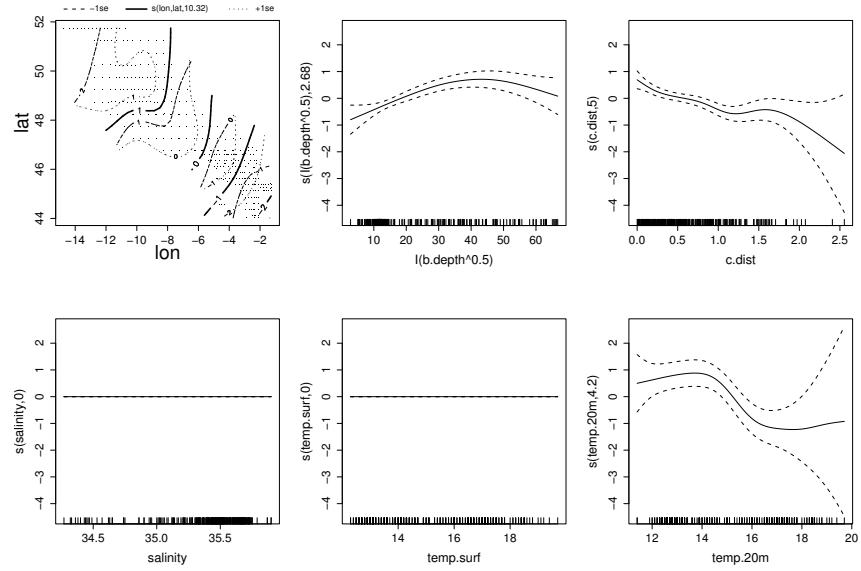


Figure 5.17 *Estimated model terms for the simple additive gm mackerel model.*

model (see section 4.1.6). The argument `scale=-1` forces the scale parameter of the Poisson to be treated as unknown, and smoothing parameters to be estimated by GCV, rather than UBRE, which is the Poisson default: hence the model is employing an ‘overdispersed Poisson’ structure. The argument `gamma=1.4`, forces each model effective degree of freedom to count as 1.4 degrees of freedom in the GCV score, which forces models to be a little smoother than they might otherwise be, and is an ad hoc way of avoiding overfitting (Kim and Gu, 2004). Sea bed depth has been square root transformed, to achieve an even spread of covariate values, without a few high depths having undue leverage. The right hand panels of figure 5.16 show residual plots for this model, which are not too bad, considering the high proportion of zeroes in these data.

Figure 5.17 shows the estimated smooth terms for model `gm`. The surface temperature and salinity effects have been shrunk to zero. Refitting without salinity enables the full data set to be used for estimation, so surface temperature should be left in for the moment (some experimentation is needed to find a reasonable `k` for the smooth of `lon` and `lat`).

```
> gml<-gam(egg.count ~ s(lon,lat,bs="ts",k=100) +
+           s(l(b.depth^.5),bs="ts") + s(c.dist,bs="ts") +
+           s(temp.surf,bs="ts") + s(temp.20m,bs="ts") +
+           offset(log.net.area),
+           data=mack,family=poisson,scale=-1,gamma=1.4)
> gml
```

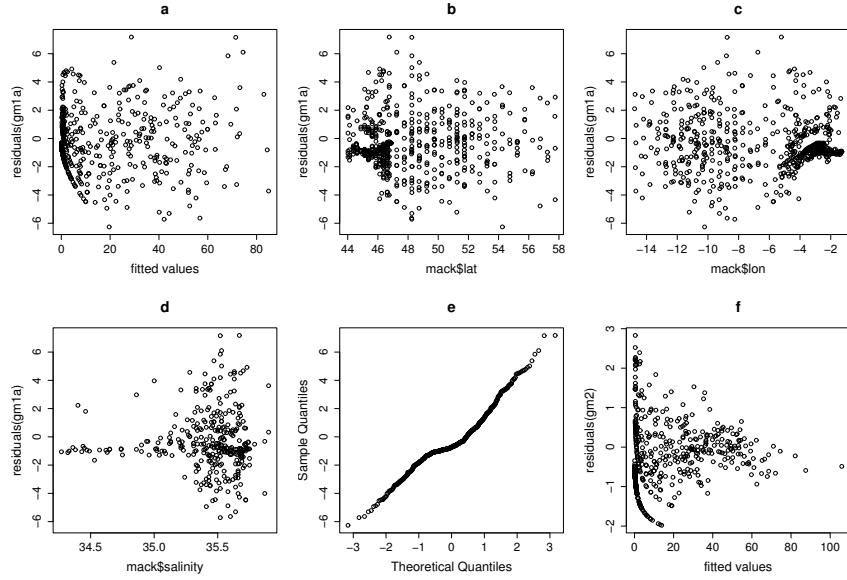


Figure 5.18 *a-e* are residual plots for model gmla. Only *d* appears problematic, but in fact the pattern is simply the result of both data and model predicting zeroes (or near zeroes) in the areas of low salinity. *f* is a residual plot for a negative binomial model and is clearly not satisfactory.

Family: poisson
Link function: log

Formula:
`egg.count ~ s(lon, lat, bs="ts", k=100) + s(I(b.depth^0.5),
 bs="ts") + s(c.dist, bs="ts") + s(temp.surf, bs="ts") +
 s(temp.20m, bs="ts") + offset(log.net.area)`

Estimated degrees of freedom:
`77.3188 1.6469 1.49499e-15 4.0150e-09 6.24424 total = 86.2099`

GCV score: 5.834046

Clearly their effective degrees of freedom are so small that the smooths of `c.dist` and `temp.surf` have effectively been eliminated from the model, so we may as well refit without them.

```
gmla<-gam(egg.count ~ s(lon, lat, bs="ts", k=100) +  

  s(I(b.depth^.5), bs="ts") + s(temp.20m, bs="ts") +  

  offset(log.net.area), data=mack, family=poisson,  

  scale=-1, gamma=1.4)
```

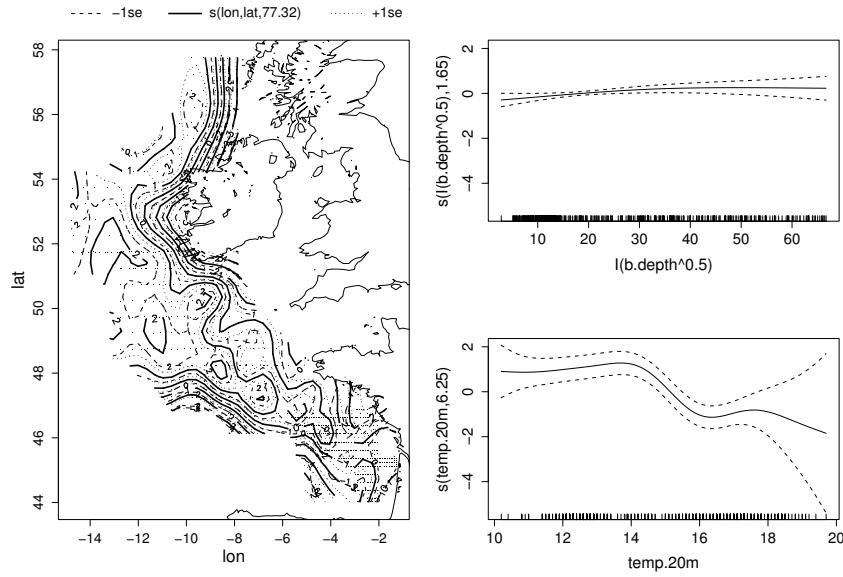


Figure 5.19 *Estimated smooth terms for the mackerel model gm1a. The left panel shows the smooth of location. The upper right panel shows the smooth of seabed depth and the lower right panel shows the smooth of temperature at 20 metres depth.*

As expected the estimates of the smooths included in gm1a are almost identical to the estimates of the equivalent smooths from gm1. Residual plots for gm1a are shown in figure 5.18, and suggest no problems with the model. In some respects the high degrees of freedom estimated for the spatial smooth is disappointing: biologically it would be more satisfactory for the model to be based on predictors to which spawning fish might be responding directly. Spatial location can really only be a proxy for something else, or the result of a process in which much of the pattern is driven by spatial correlation.

The fitted model produces a scale parameter estimate of 4.8, indicating a fair degree of over-dispersion, relative to Poisson data. Rather than adopting the quasi-likelihood over dispersed Poisson approach, of the models used so far, it might be worth trying the negative binomial distribution. gam can use the negative binomial family from the MASS library, although, at time of writing, only with performance iteration estimation of smoothing parameters.

```
> gm2<-gam(egg.count ~ s(lon, lat, bs="ts", k=40) +
+           s(l(b.depth^.5), bs="ts") + s(c.dist, bs="ts") +
+           s(temp.surf, bs="ts") + s(temp.20m, bs="ts") +
+           offset(log.net.area),
+           data=mack, family=negative.binomial(1),
+           control=gam.control(maxit=100), gamma=1.4)
> gm2
```

```

Family: Negative Binomial(0.6226)
Link function: log

Formula:
egg.count ~ s(lon, lat, bs="ts", k=40) + s(I(b.depth^0.5),
    bs="ts") + s(c.dist, bs="ts") + s(temp.surf, bs="ts") +
    s(temp.20m, bs="ts") + offset(log.net.area)

Estimated degrees of freedom:
17.230 2.2881 0.93996 6.6139e-10 5.1026 total = 26.562

GCV score: 1.126673

```

The model ascribes considerably more variability to random variation than do the Poisson based models, but far fewer degrees of freedom to the spatial smooth (even if k is increased substantially). However the residual plot, figure 5.18f, shows a clear pattern, with residual variability declining sharply with increasing mean: the model appears to overstate the variance of data corresponding to high mean densities. Hence, from the models considered here, `gmla` appears to me least inappropriate, and its component smooths are shown in figure 5.19. The rather gentle nature of the depth effect might suggest that it is not really significant, however

```

> anova(gmla)
[edited]
Approximate significance of smooth terms:
      edf Est.rank   F p-value
s(lon, lat)    77.313 99.000 6.583 < 2e-16
s(I(b.depth^0.5)) 1.656   9.000 2.302  0.0152
s(temp.20m)     6.250   9.000 6.572 6.24e-09

```

implies otherwise. Note however that the p-value probably overstates the significance of sea bed depth, although it is highly unlikely that an exact p-value would be greater than 0.05 (what simulation evidence there is suggests it is probably < 0.03).

5.4.2 Model predictions

The purpose of this sort of modelling exercise is assessment of the total stock of eggs, which means that predictions are required from the model. In the first instance a simple map of predicted densities is useful. The data frame `mackp` contains the model covariates on a regular grid, over the survey area, as well as an indexing column indicating which grid square the covariates belong to, in an appropriate 2D array. The following code produces the plot on the left hand side of figure 5.20

```

mackp$log.net.area <- 0*mackp$lon # make offset column
lon<-seq(-15,-1,1/4);lat<-seq(44,58,1/4)
zz<-array(NA,57*57)
zz[mackp$area.index]<-predict(gmla,mackp)

```

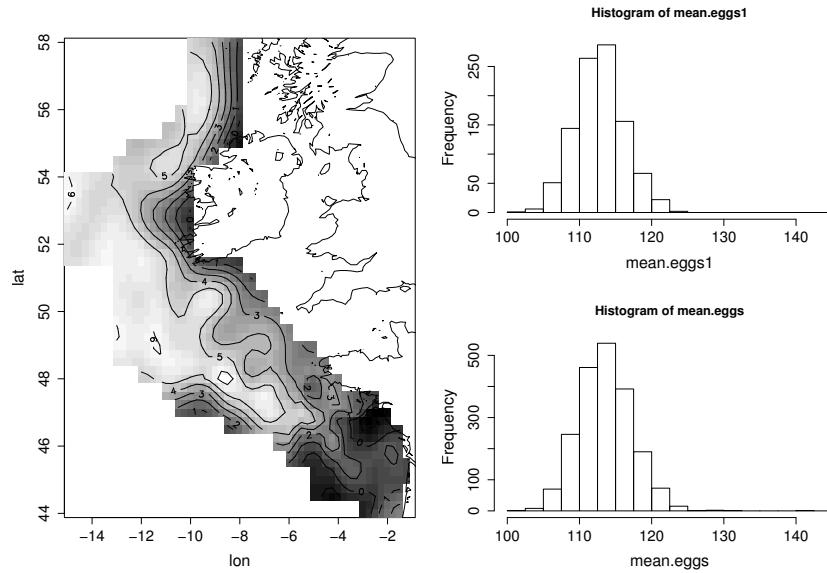


Figure 5.20 The left hand panel shows predicted log densities of mackerel eggs over the 1992 survey area, according to the model gmla. The upper right panel shows the posterior distribution of mean egg densities per square metre sea surface, conditional on the estimated smoothing parameters. The lower left panel is the same, but unconditional.

```
image(lon,lat,matrix(zz,57,57),col=gray(0:32/32),
      cex.lab=1.5,cex.axis=1.4)
contour(lon,lat,matrix(zz,57,57),add=TRUE)
lines(coast$lon,coast$lat,col=1)
```

Notice the substantial problem that the egg densities remain high at the western boundary of the survey area.

Typically, uncertainty estimates are required for quantities derived from fitted model predictions, and as in the brain imaging example, these can be obtained by simulation from the posterior distribution of the model coefficients. For example, the following obtains a sample from the posterior distribution of mean density of mackerel eggs across the survey area, shown in the upper right panel of figure 5.20.

```
library(MASS)
br1 <- mvtnorm(n=1000,coef(gmla),gmla$Vp)
Xp <- predict(gmla,newdata=mackp,type="lpmatrix")
mean.eggs1 <- colMeans(exp(Xp%*%t(br1)))
hist(mean.eggs1)
```

A disadvantage of such simulations, from the posterior distribution of the parameters,

β , is that they are conditional on the estimated smoothing parameters, $\hat{\lambda}$. That is, we are simulating from the posterior $f(\beta|\hat{\lambda})$, when we would really like to simulate from $f(\beta)$. Following section 4.9.3, we could improve matters by approximating $f(\beta)$ by its bootstrap sampling distribution, and then using the fact that $f(\beta) = f(\beta|\hat{\lambda})f(\hat{\lambda})$, to simulate from an approximate version of $f(\beta)$. The following code does just that, and plots the results in the lower right panel of figure 5.20.

```
f<-fitted(gmla)
form<-egg$count~offset(log.net.area)+s(lon,lat,bs="ts",k=100) +
      s(I(b.depth^.5),bs="ts") +s(temp.20m,bs="ts")
mack.bs <- mack
n <- nrow(mack)
br <- matrix(0,0,length(coef(gmla)))
for (i in 1:19)
{ e <- rpois(rep(1,n),f) - f
  y <- round(f+e*gm2$sig2^.5)
  y[y<0] <- 0
  mack.bs$egg.count <- y
  sp <-
    gam(form,data=mack.bs,family=poisson,scale=-1,gamma=1.4)$sp
  b <- gam(form,data=mack,family=poisson,sp=sp,scale=-1)
  br <- rbind(br,mvrnorm(n=100,coef(b),b$Vp))
}
br <- rbind(br,mvrnorm(n=100,coef(gmla),gmla$Vp))
mean.eggs <- colMeans(exp(Xp%*%t(br)))
hist(mean.eggs)
```

The unconditional distribution is a little wider than the conditional one, but in this case the differences are not large.

5.5 Portuguese larks

Figure 5.21 shows data on the presence or absence of Crested Lark in each of a set of sampled $2\text{km} \times 2\text{km}$ squares, gathered as part of the compilation of the Portuguese Atlas of Breeding Birds. The whole of Portugal was divided into $10\text{km} \times 10\text{km}$ squares, each square was further subdivided into 25, $2\text{km} \times 2\text{km}$ ‘tetrads’, and a number of tetrads (usually 6) were selected from each square. Each selected tetrad[†] was then surveyed, to establish which bird species were breeding within it: Crested Lark is one of the species surveyed (although it should be noted that there are some problems distinguishing Crested Lark from Thekla Lark).

The compilers of the Atlas would like to summarize the information in figure 5.21, as a map, showing how the probability that a tetrad contains breeding Crested Larks varies over Portugal. An obvious approach is to model the presence absence data, c_i , as

$$\text{logit}(\mu_i) = f(x_i, y_i)$$

[†] Actually at time of writing the field work was not quite finished, so a few tetrads had yet to be surveyed.

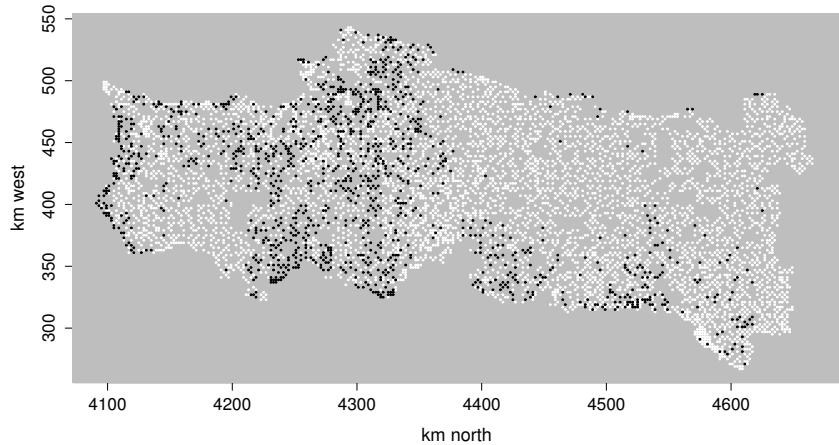


Figure 5.21 Presence (black) and absence (white) of Crested Lark in sampled 2km by 2km squares in Portugal.

where $\text{logit}(\mu_i) = \mu_i/(1 - \mu_i)$, f is a smooth function of location variables x and y , and $c_i \sim \text{binomial}(1, \mu_i)$. The geographic nature of the data, suggests using an isotropic smoother to represent f . Given the rather large number of data, it speeds things up to base the TPRS on rather fewer spatial locations than those contained in the entire data set: this can be done by making use of the `knots` argument of `gam`. In the following, 2000 randomly chosen tetrad locations are used as initial knots from which to produce a TPRS basis.

```
ind <- sample(1:25100, 2000, replace=FALSE)
m2 <- gam(crestlark ~ s(x, y, k=100), data=bird, family=binomial,
          knots=bird[ind, ], gamma=1.4)
```

The model is fitted with $\gamma = 1.4$, since for this application slight over-smoothing is much preferable to under-smoothing. By default UBRE/AIC is used for smoothing parameter estimation in the binomial case, and in this example a relatively complex model is selected.

```
> m2
Family: binomial
Link function: logit

Formula:
crestlark ~ s(x, y, k = 100)

Estimated degrees of freedom:
82.58799  total = 83.58799

UBRE score: -0.2416895
```

Model checking with binary data is somewhat awkward,(see exercise 2 in Chapter

2) especially for spatial data, but for this application we can simplify matters considerably by choosing to model the data at the 10km by 10km square level, in order to obtain a binomial response, with easier to interpret residuals. It turns out that, in terms of predicted probabilities, the estimated models are almost indistinguishable, whether we model raw data, or the aggregated data. The `bird` data frame contains a column `QUADRICULA` identifying which 10 km square each tetrad belongs to, so aggregation is easy.

```
bird$n <- bird$y*0+1 # when summed, gives binomial denominator
bird$n[is.na(bird$crestlark)] <- NA
ba <- aggregate(data.matrix(bird),by=list(bird$QUADRICULA),
                 FUN=sum,na.rm=TRUE)
ba$x <- ba$x / 25 # don't want locations summed!
ba$y <- ba$y / 25
```

The model can now be fitted.

```
> m10 <- gam(cbind(crestlark,n-crestlark)~s(x,y,k=100),
              data=ba,family=binomial,gamma=1.4)
> m10

Family: binomial
Link function: logit

Formula:
cbind(crestlark, n - crestlark) ~ s(x, y, k = 100)

Estimated degrees of freedom:
75.22112 total = 76.22112

UBRE score: 0.5183815
```

As usual, the (deviance) residuals should be plotted against fitted values to check model assumptions, but conventional residual plots are unlikely to pick up one potential problem with spatial data: namely spatial auto-correlation in the residuals, and consequent violation of the independence assumption. To check this, it is useful to examine the *variogram* of the residuals, and the `geoR` package has convenient functions for doing this.

```
library(geoR)
coords<-matrix(0,1004,2);coords[,1]<-ba$x;coords[,2]<-ba$y
gb<-list(data=residuals(m10,type="d"),coords=coords)
plot(variog(gb,max.dist=1e5))
plot(fitted(m10),residuals(m10))
```

The plotted variogram is shown at the top left of figure 5.22. Uncorrelated residuals should give a more or less flat variogram, while un-modelled spatial auto-correlation usually results in a variogram which increases sharply before eventually plateau-ing. In the current case, the the variogram suggests no autocorrelation, or even slight negative autocorrelation which might suggest very slight overfitting (although see exercise 4).

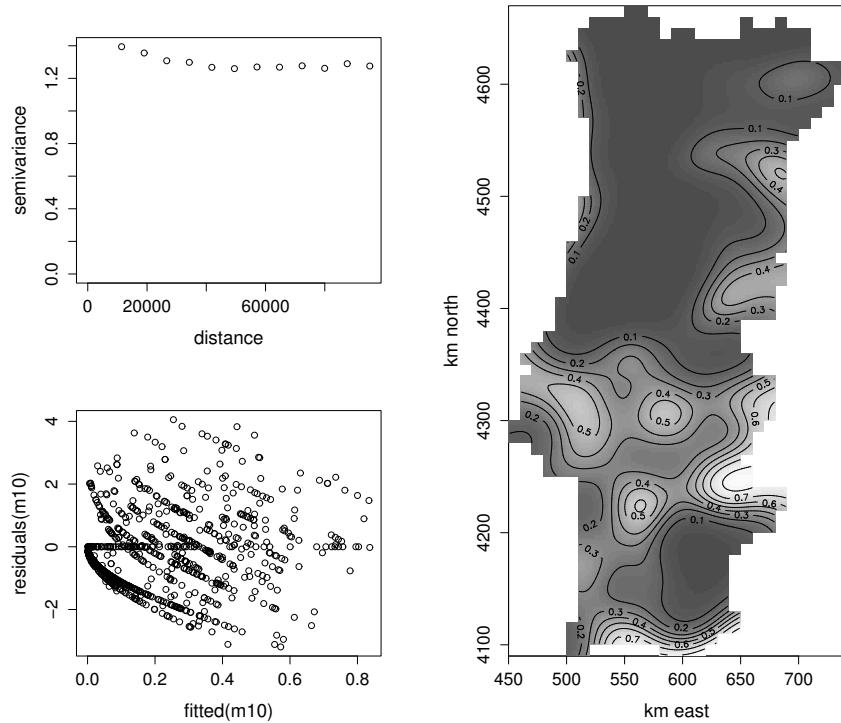


Figure 5.22 Plots related to model m10 for Crested Lark in Portugal. Top left: variogram of the deviance residuals: there is no evidence that the model is missing auto-correlation in these data. Lower left: residuals against fitted values, showing no problems, except for minor overdispersion. Right: model predicted probability that Crested Lark is breeding in a 2km square at any location in Portugal.

The residual plot against fitted values shown at the lower left of figure 5.22 is very good, for binomial data, although there is a suggestion of overdispersion (i.e. the data seem slightly more variable than truly binomial data).

The end product of this modelling exercise should be a map of breeding probabilities. In this case the `bird` data frame actually contains the locations for all 2km tetrads in Portugal, so it is a fairly simply matter to produce such a map. The only fiddly part is embedding the predictions, for each tetrad in Portugal, in a larger square array of tetrads, where any tetrad not in Portugal is assigned the value NA: this is necessary to facilitate plotting. The following code suffices.

```
mx <- sort(unique(bird$x)); my<-sort(unique(bird$y))
nx<-length(mx); ny<-length(my)
ixm <- 1:nx; names(ixm)<-mx
ix <- ixm[as.character(bird$x)]
iym <- 1:ny; names(iym)<-my
iy <- iym[as.character(bird$y)]
```

```

um <- matrix(NA,nx,ny)
fv10 <- predict(m10,bird,type="response")
my<-my/1000;mx <- mx/1000
um[iy+(iy-1)*nx] <- fv10
image(mx,my,matrix(um,nx,ny),xlab="km east",ylab="km north")
contour(mx,my,matrix(um,nx,ny),add=TRUE)

```

The results are shown on the right hand side of figure 5.22. Note that the probabilities plotted are interpretable as follows: if you pick a 10km square in Portugal, then the plotted probability for that square, is the probability that a randomly selected 2km tetrad within the square will contain breeding Crested Larks.

5.6 Other packages

There are a number of other packages implementing GAMs, and related models for R, and this section offers a very brief introduction to two of them: Trevor Hastie's `gam` package, which is a port to R of the original `gam` in S-PLUS, and Chong Gu's `gss` package, which is a rather complete package for general smoothing spline models. In both cases, a model from the Chicago air pollution example will be re-estimated.

Two packages not covered here are the `assist` package, which implements the approach of e.g. Wang (1998b) and Wang (1998a), and the `gamlss` package which offers the generalization of GAMs presented in Rigby and Stasinopoulos (2004).

5.6.1 Package `gam`

Package `gam` is an implementation of the GAM framework of Hastie and Tibshirani (1990). The `mgcv` package was an attempt to provide GAMs for R, before the `gam` package was available, and its functions are based closely on the S equivalents designed by Hastie (1993). For this reason, basic use of `gam` is rather similar to basic use of `mgcv`. The main differences are: (i) `s()` terms in a `gam`:`:gam` formula denote cubic smoothing spline smooths of one variable; (ii) smooths of any number of variables are provided by `lo` terms, and are loess smooths; (iii) `gam`:`:gam` does not estimate the degree of smoothness automatically.

In the following, a version of the final Chicago air pollution model is fitted, using package `gam`. The flexibility of the smooths (controlled by the `df` and `span` arguments) has been selected to give a fit with terms of similar complexity to those estimated using `mgcv`:`:gam`.

```

> library(gam)
> bfm <- gam(death~s(time,df=140)+lo(o3,tmp,span=.1),
   family=poisson,control=gam.control(bf.maxit=150))

```

The control argument has to be modified a little, in this case, to achieve convergence of the back fitting iterations. Here is a default summary of the fit.

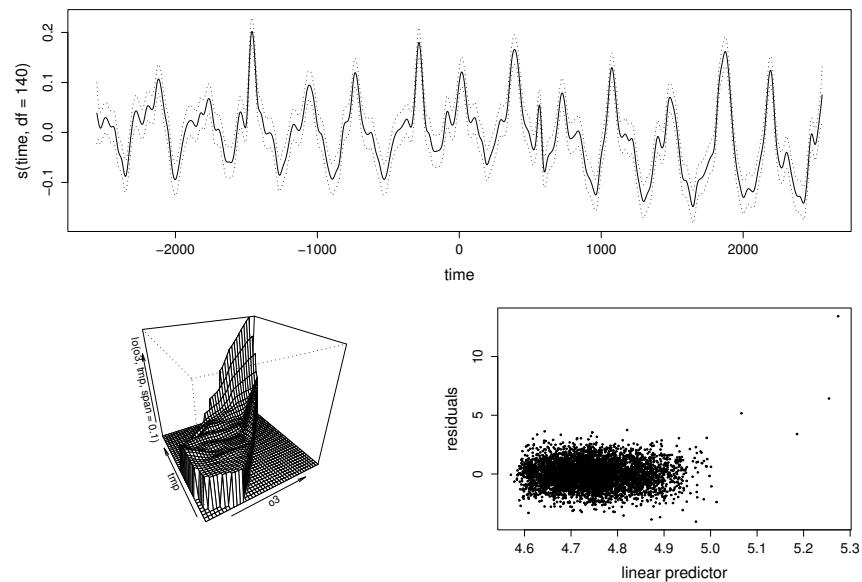


Figure 5.23 A GAM fitted to the Chicago air pollution data, using `gam` from package `gam`. The upper panel is the estimated smooth of time while the lower left panel is a perspective plot of the estimated ozone, temperature interaction. The lower right plot is a residual plot.

```
> summary(bfm)

Call: gam(formula=death~s(time,df=140)+lo(o3,tmp,span=0.1),
  family=poisson,control=gam.control(maxit=100,bf.maxit=150))
Deviance Residuals:
    Min      1Q      Median      3Q      Max 
-4.05604 -0.72077 -0.02738  0.67355 13.43195 

(Dispersion Parameter for poisson family taken to be 1)

Null Deviance: 9860.715 on 5110 degrees of freedom
Residual Deviance: 5823.157 on 4929.065 degrees of freedom
AIC: 39815.83

Number of Local Scoring Iterations: 20

DF for Terms and Chi-squares for Nonparametric Effects

          Df Npar Df Npar Chisq  P(Chi)
(Intercept)     1.0
s(time, df = 140) 1.0   139.0   1336.27  0.00
lo(o3, tmp, span = 0.1) 2.0    38.9    645.51  0.00
```

Plotting of model terms is easily.

```
plot(bfm, se=T, rug=F, phi=30, theta=-30)
```

produces the upper and lower left panels in figure 5.23. The residual plot, at lower right of the figure, shows that this model still misses the really high deaths by a little way. This may be because of the way the relative scaling of ozone and temperature has been handled: these variables are scaled into the unit square, and then smoothed with a locally weighted regression. The scaling step is somewhat arbitrary, and imposes an implicit assumption about how smoothly death rate varies with temperature as opposed to ozone: this assumption maybe behind the slightly worse fit, relative to the previous version of this model.

5.6.2 Package gss

The `gss` package is a comprehensive implementation of the general smoothing spline approach to modelling described in the monographs by Wahba (1990) and Gu (2002). The modelling approach is somewhat different to the `gam` functions met so far, being based strongly on the notion of ANOVA decompositions of functions (see section 4.10.2). Again the Air pollution model provides a useful example:

```
library(gss)
ssm <- gssanova(death~time+o3*tmp, family="poisson", nbasis=200)
```

The `gssanova1` function is a computationally efficient reduced rank version of the function `gssanova`, which fits generalized smoothing spline ANOVA models based on the methods reported in Kim and Gu (2004). The model formula specifies a linear predictor, with the following structure:

$$f_1(\text{time}) + f_2(\text{o3}) + f_3(\text{tmp}) + f_4(\text{o3}, \text{tmp}).$$

i.e. smooth main effect functions of the three covariates, plus a smooth interaction of ozone and temperature. Notice that the size of approximating basis used by `gssanova1` has been adjusted here, using the `nbasis` argument: the default basis size performs rather poorly in this example. Here is a summary of the fitted model.

```
> summary(ssm)

Call:
gssanova1(formula=death~time+o3*tmp, family="poisson",
nbasis=200)

(Dispersion parameter for poisson family taken to be 1)

Working residuals (weighted):
      Min        1Q    Median        3Q       Max
-0.34584431 -0.06654956 -0.00357026  0.06297024  0.48446762
Residual sum of squares: 5699.994
```

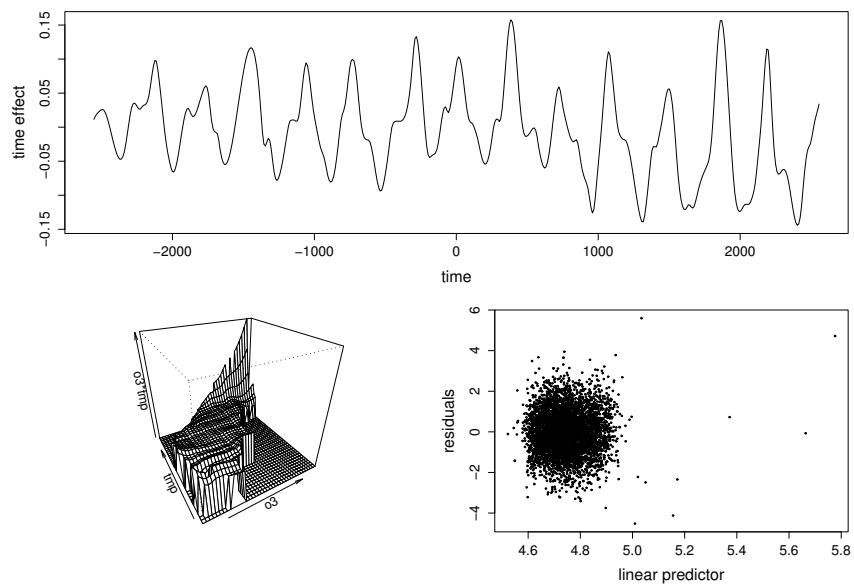


Figure 5.24 A GAM fitted to the Chicago air pollution data, using `ssanova1` from package `gss`. The upper panel is the estimated smooth of time while the lower left panel is a perspective plot of the estimated ozone, temperature interaction. The lower right plot is a residual plot.

Deviance residuals:

Min	1Q	Median	3Q	Max
-4.5208983	-0.7249406	-0.0373412	0.6706598	5.5970047

Deviance: 5695.739
Null deviance: 9860.715

Penalty associated with the fit: 81.43146

`gss` does not provide a default plot function for such models, but it does provide a `predict` method function, so that it is easy to create the required plots. The following creates the upper panel in figure 5.24.

```
tp <- seq(min(time), max(time), length=500)
fvt <- predict(ssm, newdata=data.frame(time=tp,
                                         o3=rep(mean(o3), 500), tmp=rep(mean(tmp), 500)),
               include=list("time"))
plot(tp, fvt, type="l", xlab="time", ylab="time effect")
```

Notice the `include` argument to `predict.ssanova1`: only model terms on this list are included in the predictions: so only the smooth function of time has been evaluated and returned here[†].

[†] `predict.ssanova1` allows the calculation of standard errors for predictions, but for this particular model this failed (a very unusual occurrence).

The code for producing the perspective plot, at the lower left of figure 5.24, showing the dependence on ozone and temperature, is as follows.

```
m <- 40
o3m <- seq(min(o3), max(o3), length=m)
tmpm <- seq(min(tmp), max(tmp), length=m)
tmpp <- rep(tmpm, rep(m, m))
o3p <- rep(o3m, m)
pd <- data.frame(time=rep(0, m*m), o3=o3p, tmp=tmpp)
fv <- predict(ssm, newdata=pd, include=list("o3", "tmp", "o3:tmp"))
library(mgcv)
ind <- exclude.too.far(o3p, tmpp, o3, tmp, dist=0.04)
fv[ind] <- 0
persp(o3m, tmpm, matrix(fv, m, m), phi=30, theta=-30, zlab="o3*tmp",
      xlab="o3", ylab="tmp")
```

Notice the modified `include` argument to `predict`: we have to specify the smooth main effects of ozone and temperature *and* the smooth interaction of the two, in order to obtain the complete dependence on these two variables. The `exclude.too.far` function, from package `mgcv`, is used to remove parts of the perspective plot that are too distant from supporting data.

The residual plot for this model is shown in the lower right panel of figure 5.24 and is rather encouraging.

5.7 Exercises

1. This question re-examines the `hubble` data from Chapter 1.

- (a) Use `gam` to fit the model:

$$V_i = f(D_i) + \epsilon_i$$

to the `hubble` data, where f is a smooth function and the ϵ_i are i.i.d. $N(0, \sigma^2)$. Does a straight line model appear to be most appropriate? How would you interpret the best fit model?

- (b) Examine appropriate residual plots and refit the model with more appropriate distributional assumptions. How are your conclusions from part (a) modified?

2. This question is about using `gam` for univariate smoothing, the advantages of penalized regression and weighting a smooth model fit. The `mcycle` data in the `MASS` package are a classic dataset in univariate smoothing, introduced in Silverman (1985). The data measure the acceleration of the rider's head, against time, in a simulated motorcycle crash.

- (a) Plot the acceleration against time, and use `gam` to fit a univariate smooth to the data, selecting the smoothing parameter by GCV (k of 30 to 40 is plenty for this example). Plot the resulting smooth, with partial residuals, but without standard errors.

- (b) Use `lm` and `poly` to fit a polynomial to the data, with approximately the same degrees of freedom as was estimated by `gam`. Use `termplot` to plot

- the estimated polynomial and partial residuals. Note the substantially worse fit achieved by the polynomial, relative to the penalized regression spline fit.
- (c) It's possible to overstate the importance of penalization in explaining the improvement of the penalized regression spline, relative to the polynomial. Use `gam` to refit an un-penalized thin plate regression spline to the data, with basis dimension the same as that used for the polynomial, and again produce a plot for comparison with the previous two results.
 - (d) Redo part (c) using an un-penalized cubic regression spline. You should find a fairly clear ordering of the acceptability of the results for the 4 models tried — what is it?
 - (e) Now plot the model residuals against time, and comment.
 - (f) To try and address the problems evident from the residual plot, try giving the first 20 observations the same higher weight, α , while leaving the remaining observations with weight one. Adjust α so that the variance of the first 20 residuals matches that of the remaining residuals. Recheck the residuals plots.
 - (g) Experiment with the order of penalty used in the smooth. Does increasing it affect the model fit?
3. This question uses the `mcycle` data again, in order to explore the influence matrix of a smoother, more fully.
- (a) Consider a model for response data, y , which has the influence matrix, \mathbf{A} , mapping the response data to the fitted values, given a smoothing parameter, λ . i.e. $\hat{\mu} = \mathbf{Ay}$. Show that if we fit the same model, with the same λ to the response data \mathbf{I}_j (the j^{th} column of the identity matrix) then the resulting model fitted values are the j^{th} column of \mathbf{A} .
 - (b) Using the result from part (a) evaluate the influence matrix, \mathbf{A} , for the model fitted in question 2(a).
 - (c) What value do all the rows of \mathbf{A} sum to? Why?
 - (d) Any smoothing model that can be represented as $\hat{\mu} = \mathbf{Ay}$, simply replaces each value y_j by a weighted sum of neighbouring y_i values. e.g. $\hat{\mu}_j = \sum_i A_{ji}y_i$, where the A_{ji} are the weights in the summation. It is instructive to examine the weights in the summation, so plot the weights used to form $\hat{\mu}_{65}$ against `mcycle$time`. What you have plotted is the ‘equivalent kernel’ of the fitted spline (at the 65th datum).
 - (e) Plot all the equivalent kernels, on the same plot. Why do their peak heights vary?
 - (f) Now vary the smoothing parameter around the GCV selected value. What happens to the equivalent kernel for the 65th datum?
4. This question follows on from questions 2 and 3, and examines the auto-correlation of residuals that results from smoothing.
- (a) Based on the best model from question 2, produce a plot of the residual auto-correlation at lag 1 (average correlation between each residual and the previous residual, see `?acf`) against the model effective degrees of freedom. Vary the degrees of freedom between 2 and 40 by varying the `sp` argument to `gam`.

- (b) Why do you see positive autocorrelation at very low EDF, and what causes this to reduce as the EDF increases?
- (c) The explanation of why autocorrelation becomes negative is not quite so straightforward. Given the insight from question 3, that smoothers operate by forming weighted averages of neighbouring data, the cause of the negative autocorrelation can be understood by examining the simplest weighted average smoother: the k -point running mean. The fitted values from such a smoother are a simple average of the k nearest neighbours of the point in question (including the point itself). k is an odd integer.
- Write out the form of a typical row, j , of the influence matrix, \mathbf{A} , of the simple running mean smoother. Assume that row j is not near the beginning or end of \mathbf{A} , and is therefore unaffected by edge-effects.
 - It is easy to show that the residuals are given by $\hat{\epsilon} = (\mathbf{I} - \mathbf{A})\mathbf{y}$ and hence that their covariance matrix is $\mathbf{V}_{\hat{\epsilon}} = (\mathbf{I} - \mathbf{A})(\mathbf{I} - \mathbf{A})^T \sigma^2$. Find expressions for the elements of $\mathbf{V}_{\hat{\epsilon}}$ on its leading diagonal and on the sub- and super-diagonal, in terms of k . Again, only consider rows/columns that are unaffected by being near the edge of the data.
 - What do your expressions suggest about residual auto-correlation as the amount of smoothing is reduced?
5. This question is about modelling data with seasonality, and the need to be very careful if trying to extrapolate with GAMs (or any statistical model). The data frame `co2s` contains monthly measurements of CO₂ at the south pole from January 1957 onwards. The columns are `co2`, the month of the year, `month`, and the cumulative number of months since January 1957, `c.month`. There are missing `co2` observations in some months.
- Plot the CO₂ observations against cumulative months.
 - Fit the model, $co2_i = f(c.month_i) + \epsilon_i$ where f is a smooth function and the ϵ_i are i.i.d. with constant variance, using the `gam` function. Use the `cr` basis, and a basis dimension of 300.
 - Obtain the predicted CO₂ for each month of the data plus 36 months after the end of the data as well as associated standard errors. Produce a plot of the predictions with twice standard error bands. Are the predictions in the last 36 months credible?
 - Fit the model $co2_i = f_1(c.month_i) + f_2(month_i) + \epsilon_i$ where f_1 and f_2 are smooth functions, but f_2 is cyclic (you will need to use the `knots` argument of `gam` to ensure that f_2 wraps appropriately: it's important to make sure that January is the same as January, not that December and January are the same!).
 - Repeat the prediction and plotting in part (c) for the new model. Are the predictions more credible now? Explain the differences between the new results and those from part (c).
6. The data frame `ipo` contains data from Lowry and Schwert (2002) on the number of ‘Initial Public Offerings’ (IPOs) per month in the US financial markets between 1960 and 2002. IPOs are the process by which companies go public:

ownership of the company is sold, in the form of shares, as a means of raising capital. Interest focuses on exploring the effect that several variables may have on numbers of IPOs per month, `n.ipo`. Of particular interest are the variables:

`ir` the average initial return from investing in an IPO. This is measured as percentage difference between the offer price of shares, and the share price after the first day of trading in the shares: this is basically a reflection of by how much the offer price undervalues the company. One might expect companies to pay careful attention to this when deciding on IPO timing.

`dp` is the average percentage difference between the middle of the share price range proposed when the IPO is first filed, and the final offer price. This might be expected to carry information about direction of changes in market sentiment.

`reg.t` the average time (in days) it takes from filing to offer. Obviously fluctuations in the length of time it takes to register has a direct impact on the number of companies making it through to offering in any given month.

Find a suitable possible model for explaining the number of IPOs in terms of these variables (as well as time, and time of year). Note that it is probably appropriate to look at lagged versions of `ir` and `dp`, since the length of the registration process precludes the number of IPOs in a month being driven by the initial returns in that same month. In the interests of interpretability it is probably worth following the advice of Kim and Gu (2004) and setting `gamma=1.4` in the `gam` call. Look at appropriate model checking plots, and interpret the results of your model fitting.

7. The data frame `wine` contains data on prices and growing characteristics of 25 high quality Bordeaux wines from 1952 to 1998 as reported in:

<http://schwert.ssb.rochester.edu/a425/a425.htm>.

`price` gives the average price as a percentage of 1961; `s.temp` is the average temperature (Celsius) over the summer preceding harvest, while `h.temp` is the average temperature at harvest; `w.rain` is the mm of rain in the preceding winter, while `h.rain` give the mm of rain in the harvest month; `year` is obvious. Create a GAM to model `price` in terms of the given predictors. Interpret the effects and use the model to predict the missing prices.

8. The `mgcv` package allows users to add their own smoother classes for use with `gam`. Two functions must be written in order to do this. The first is a smooth constructor function with the name `smooth.construct.xy.smooth.spec`, where `xy` is the two letter code which will be used to identify the smoother class when referring to it in a `gam` formula: something like `s(..., bs="xy")` would be used to invoke this class. This function has access to the model covariates, any supplied knots and the information on setting up the smooth supplied to `s()`. It must return an object containing model and penalty matrices, and some other supplementary information. It must also assign that object a class: in the following the class is assumed to be called `xy.smooth`, but you are free to choose. The second function is called `Predict.matrix.xy.smooth`, and is used to evaluate a ‘prediction matrix’ which will map the coefficients of the smooth to evaluations of the smooth at a new set of data — this function is used for prediction and plotting.

Full specification for the two functions can be found in the help file `?p.spline`, which also includes example functions implementing P-spline smoothers for `gam`. Further examples can be found in the functions for handling smooth types `cr`, `cc`, `tp`, while the functions for smooths `cs` and `ts` show how slight variations on existing smoothers can be implemented. Once you have defined a (single penalty) smoother it can automatically be used for tensor product smoothing (the constructor `smooth.construct.tensor.smooth.spec` sets up tensor product smooths, and is somewhat from other constructors).

Ruppert et al. (2003) discuss the use of penalized regression splines based on simple ridge penalties applied to the coefficients of a spline represented using the ‘truncated power basis’ for splines. For a spline of order m with $k-m-1$ ‘knots’, x_i^* , a spline f is represented as:

$$f(x) = \sum_{i=0}^m x^i \beta_{i+1} + \sum_{i=1}^{k-m-1} \beta_{i+m+1} |x - x_i^*|_+^m$$

where the β_i are unknown parameters and $|z|_+ = z$ if z is positive and zero otherwise. (Note that a cubic spline has $m = 3$, in this question.) The suggested penalty on the spline is simply

$$\lambda \sum_{i=m+2}^k \beta_i^2.$$

Using the P-spline functions as templates, implement this class of smoothers for use with `gam`. Try it out on the first example given in the help file `?gam`.

9. Sometimes rather unusual models can be expressed as GAMs. For example the data frame `blowfly` contains `counts` (not samples!) of adults in a laboratory population of blowflies, as reported in the classic ecological papers of Nicholson (1954a,b). One possible model for these data (basically Ellner et al., 1997) is that they are governed by the equation,

$$\Delta n_{t+1} = f_b(n_{t-k})n_{t-k} - f_d(n_t)n_t + \epsilon_t,$$

where n_t is the population at time t , $\Delta n_{t+1} = n_{t+1} - n_t$, f_b and f_d are smooth functions and the ϵ_t are i.i.d. errors with constant variance. The idea is that the change in population is given by the difference between the birth and death rates plus a random term. *per capita* birth rates and death rates are smooth functions of populations, and the delayed effect of births on the adult population is because it takes around 12 days to go from egg to adult.

- (a) Plot the `blowfly` data against time.
- (b) Fit the proposed model, using `gam`. You will need to use `by` variables to do this. Comment on the estimates of f_b and f_d . It is worth making f_b and f_d depend on log populations, rather than the populations themselves, to avoid leverage problems.
- (c) Using the beginning of the real data as a starting sequence, iterate your estimated model forward in time, to the end of the data, and plot it. First do this

with the noise terms set to zero, and then try again with an error standard deviation of up to 500 (much larger and the population tends to explode). Comment on the results. You will need to artificially restrict the population to the range seen in the real data, to avoid problems caused by extrapolating the model.

- (d) Why is the approach used for these data unlikely to be widely applicable?
10. This question is about creating models for calibration of satellite remote sensed data. The data frame `chl` contains direct ship based measurements of chlorophyll concentrations in the top 5 metres of ocean water, `chl`, as well as the corresponding satellite estimate `chl.sw` (actually a multi-year average measurement for the location and time of year), along with ocean depth, `bath`, day of year, `jul.day` and location `lon`, `lat`. The data are from the world ocean database (see <http://seawifs.gsfc.nasa.gov/SEAWIFS/> for information on SeaWifs). `chl` and `chl.sw` do not correlate all that well with each other, probably because the reflective characteristics of water tend to change with time of year and whether the water is near the coast (and hence full of particulate matter) or open ocean. One way of improving the predictive power of the satellite observations might be to model the relationship between directly measured chlorophyll and remote sensed chlorophyll, viewing the relationship as a smooth one that is modified by other factors. Using ocean depth as an indicator for water type (near shore vs. open) a model something like:

$$\mathbb{E}(\text{chl}_i) = f_1(\text{chl.sw}_i)f_2(\text{bath}_i)f_3(\text{jul.day}_i)$$

might be a reasonable starting point.

- (a) Plot the response data and predictors against each other using `pairs`. Notice that some of predictors have very skewed distributions. It is worth trying some simple power transformations in order to get a more even spread of predictors, and reduce the chance of a few points being over influential: find appropriate transformations.
- (b) Using `gam`, try modelling the data using a model of the sort suggested (but with predictors transformed as in part (a)). Make sure that you use an appropriate family. It will probably help to increase the default basis dimensions used for smoothing, somewhat (especially for `jul.day`). Use the "`cr`" basis to avoid excessive computational cost.
- (c) In this sort of modelling the aim is to improve on simply using the satellite measurements as predictions of the direct measurements. Given the large number of data, it is easy to end up using rather complex models after quite a lot of examination of the data. It is therefore important to check that the model are not overfitting. A sensible way of doing this is to randomly select, say, 90% of the data to be fitted by the model, and then to see how well the model predicts the other 10% of data. Do this, using proportion deviance explained as the measure of fit/prediction. Note that the family you used will contain a function `dev.resids`, which you can use to find the deviance of and set of predictions.

11. Investigate the performance of the p-values reported by `summary.gam` and described in section 4.8.5, by a simulation study based on the first example in the help file `?gam`.
 - (a) If a term should not be in the model, then its associated p-values should follow a uniform distribution on (0,1). Check this by simulation. Compare the performance of the p-values when smoothing parameters are estimated, and when the term is left un-penalized (using `s(..., fx=TRUE)`). What can you conclude?
 - (b) From the results of part (a) it is tempting to do all hypothesis testing without penalization. Investigate whether this reduces the power of the tests (i.e. the ability to detect terms that are significant.)



CHAPTER 6

Mixed models and GAMMs

A different approach to estimation and inference with GAMs is based on representing GAMs as mixed models with the smooth terms as random effects. To facilitate the explanation of this approach, this chapter first introduces linear mixed models, starting with simple mixed models for balanced experimental data and then moving on to general linear mixed models. Note that Pinheiro and Bates (2000) offers fuller coverage of linear mixed modelling in R, while Ruppert et al. (2003) includes a clear explanation of smoothers as mixed model components.

In general, linear mixed models extend the linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2)$$

to

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}), \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\Lambda}\sigma^2)$$

where random vector, \mathbf{b} , contains *random effects*, with zero expected value and covariance matrix $\boldsymbol{\psi}$, and \mathbf{Z} is a model matrix for the random effects. $\boldsymbol{\Lambda}$ is a positive definite matrix, of simple structure, which is typically used to model residual autocorrelation: its elements are usually determined by some simple model, with few (or no) unknown parameters. Often $\boldsymbol{\Lambda}$ is simply the identity matrix. This extension allows the model a more complex stochastic structure than the ordinary linear model, and, in particular, implies that the elements of the response vector, \mathbf{y} , are no longer independent.

6.1 Mixed models for balanced data

Some details of the general, likelihood based, approach to mixed modelling require an understanding of linear mixed models for balanced data, so this section will briefly cover this topic. First consider an example.

6.1.1 A motivating example

Plant leaves have tiny holes, called ‘stomata’ through which they take up air, but also lose water. Most non-tropical plants photosynthesize in such a way that, on sunny

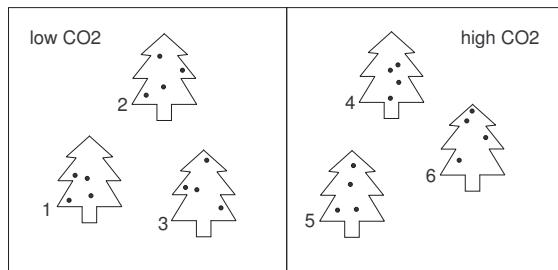


Figure 6.1 *Schematic diagram of the CO₂ experiment.*

days, they are limited by how much carbon dioxide they can obtain through these stomata. The ‘problem’, for a plant, is that if its stomata are too small, it will not be able to get enough carbon dioxide, and if they are too large it will lose too much water on sunny days. Given the importance of this to such plants, it seems likely that stomatal size will depend on the concentration of carbon dioxide in the atmosphere. This may have climate change implications, if increasing the amount of CO₂ in the atmosphere causes plants to release less water: water vapour is the most important green house gas.

Consider an experiment* in which tree seedlings are grown under two levels of carbon dioxide concentration, with 3 trees assigned to each treatment, and suppose that after 6 months growth, stomatal area is measured at each of 4 random locations on each plant (the sample sizes here are artificially small). Figure 6.1 shows the experimental layout schematically.

The wrong approach: a fixed effects linear model

A model of these data should include a (2 level) factor for CO₂ treatment, but also a (6 level) factor for individual tree, since we have multiple measurements on each tree and must expect some variability in stomatal area from tree to tree. So a suitable linear model is

$$y_i = \alpha_j + \beta_k + \epsilon_i \text{ if observation } i \text{ is for CO}_2 \text{ level } j, \text{ plant } k,$$

where y_i is the i^{th} stomatal area measurement, α_j is the population mean stomatal area at CO₂ level j , β_k is the deviation of tree k from that mean and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. Now if this is a fixed effects model, we have two problems:

1. The α_j ’s and β_k ’s are completely confounded. Trees are ‘nested’ within treatment, with 3 trees in one treatment and 3 in the other: any number you like could be

* One important part of the design of such an experiment would be to ensure that the trees are grown under natural, *variable* light levels. At constant *average* light levels the plants are not CO₂ limited.

added to α_1 and simultaneously subtracted from β_1 , β_2 and β_3 , without changing the model predictions at all, and the same goes for α_2 and the remaining β_k 's.

2. We really want to learn about trees in general, but this is not possible with a model in which there is a fixed effect for each particular tree: we cannot use the model to predict what happens to a tree other than the 6 in the experiment.

The following R session illustrates problem 1. First compare models with and without the tree factor (β_k 's):

```
> m1 <- lm(area ~ CO2 + tree,stomata)
> m0 <- lm(area ~ CO2,stomata)
> anova(m0,m1)
Analysis of Variance Table

Model 1: area ~ CO2
Model 2: area ~ CO2 + tree
  Res.Df   RSS Df Sum of Sq    F    Pr(>F)
1     22 2.1348
2     18 0.8604  4     1.2744 6.6654 0.001788 **

> m2 <- lm(area ~ tree,stomata)
> anova(m2,m1)
Analysis of Variance Table

Model 1: area ~ tree
Model 2: area ~ CO2 + tree
  Res.Df   RSS Df Sum of Sq F Pr(>F)
1     18 0.8604
2     18 0.8604  0 -2.220e-16
```

The confounding of the CO₂ and tree factors, means that the models being compared here are really the same model: as a result, they give the same residual sum of squares and have the same residual degrees of freedom — ‘comparing’ them tells us nothing about the effect of CO₂.

In many ways this problem comes about because our model is simply too flexible. Individual tree effects are allowed to take any value whatsoever, which amounts to saying that each individual tree is completely different to every other individual tree: e.g. having results for 6 trees will tell us nothing whatsoever about a 7th. This is not a sensible starting point for a model aimed at analyzing data like these. We really expect trees of a particular species to behave in broadly similar ways, so that a representative (preferably random) sample of trees from the wider population of such trees, *will* allow us to make inferences about that wider population of trees. Treating the individual trees, not as completely unique individuals, but as a random sample from the target population of trees, will allow us to estimate the CO₂ effect, and to generalize beyond the six trees in the experiment.

The right approach: a mixed effects model

The key to establishing whether CO₂ has an effect is to recognize that the CO₂ factor and tree factors are different in kind. The CO₂ effects are fixed characteristics of the whole population of trees that we are trying to learn about. In contrast, the tree effect will vary randomly from tree to tree in the population. We are not primarily interested in the values of the tree effect for the particular trees used in the experiment: if we had used a different 6 trees these effects would have taken different values anyway. But we can not simply ignore the tree effect without inducing dependence between the response observations (area), and hence violating the independence assumption of the linear model. In this circumstance, it makes sense to model the distribution of tree effects across the population of trees, and to suppose that the particular tree effects that occur in the experiment are just independent observations from this distribution. That is, the CO₂ effect will be modelled as a fixed effect, but the tree effect will be modelled as a random effect. Here is a model set up in this way:

$$y_i = \alpha_j + b_k + \epsilon_i \text{ if observation } i \text{ is for CO}_2 \text{ level } j \text{ plant } k, \quad (6.1)$$

where $b_k \sim N(0, \sigma_b^2)$, $\epsilon_i \sim N(0, \sigma^2)$ and all the b_j and ϵ_i are mutually independent random variables. Now testing for tree effects can proceed exactly as it did in the fixed effects case, by comparing the least squares fits of models with and without the tree effects. But this mixed effects model also lets us test CO₂ effects, whether or not there is evidence for a tree effect.

All that is required is to average the data at each level of the random effect, i.e. at each tree. For balanced data, such as we have here, the key feature of a mixed effects model is that this ‘averaging out’ of a random effect, automatically implies a simplified mixed effects model for the aggregated data: the random effect is absorbed into the independent residual error term. It is easy to see that the model for the average stomatal area per tree must be,

$$\bar{y}_k = \alpha_j + e_k \text{ if plant } k \text{ is for CO}_2 \text{ level } j, \quad (6.2)$$

where the e_k are independent $N(0, \sigma_b^2 + \sigma^2/4)$ random variables.

Now it is a straightforward matter to test for a CO₂ effect in R. First aggregate the data for each tree:

```
> st <- aggregate(data.matrix(stomata),
+                   by=list(tree=stomata$tree), mean)
> st$CO2 <- as.factor(st$CO2); st
   tree      area CO2 tree
1    1 1.623374   1    1
2    2 1.598643   1    2
3    3 1.162961   1    3
4    4 2.789238   2    4
5    5 2.903544   2    5
6    6 2.329761   2    6
```

and then fit the model implied by the aggregation.

```
> m3 <- lm(area~CO2, st)
> anova(m3)
Analysis of Variance Table

Response: area
          Df  Sum Sq Mean Sq F value    Pr(>F)
CO2        1 2.20531 2.20531  27.687 0.006247 ***
Residuals  4 0.31861 0.07965
```

There is strong evidence for a CO₂ effect here, and we would now proceed to examine the estimate of this fixed effect (e.g. using `summary(m3)`). Usually, with a mixed model, the variances of the random effects are of more interest than the effects themselves, so in this example σ_b^2 should be estimated.

Let RSS_i stand for the residual sum of squares for model *i*. From the usual theory of linear models we have that:

$$\hat{\sigma}^2 = \text{RSS}_1/18$$

(RSS₁ is the residual sum of squares from fitting (6.1)) and

$$\widehat{\sigma_b^2 + \sigma^2}/4 = \text{RSS}_3/4$$

(RSS₃ is the residual sum of squares from fitting (6.2)). Both estimators are unbiased. Hence, an unbiased estimator for σ_b^2 is

$$\hat{\sigma}_b^2 = \text{RSS}_3/4 - \text{RSS}_1/72$$

This can easily be evaluated in R.

```
> summary(m3)$sigma^2 - summary(m1)$sigma^2/4
[1] 0.06770177
```

6.1.2 General principles

To see how the ideas from the previous section generalize, consider data from a designed experiment and an associated linear mixed model for the data, in which the response variable depends only on factor variables and their interactions (which may be random or fixed). Assume that the data are *balanced* with respect to the model, meaning that for each factor or interaction in the model, the same number of data have been collected at each of its levels. In this case:

- Aggregated data, obtained by averaging the response at each level of any factor or interaction, will be described by a mixed model, derived from the original mixed model by the averaging process.
- Models for different aggregations will enable inferences to be made about different fixed and random factors, using standard methods for ordinary linear models. Note that not all aggregations will be useful, and the random effects themselves can not be ‘estimated’ in this way.

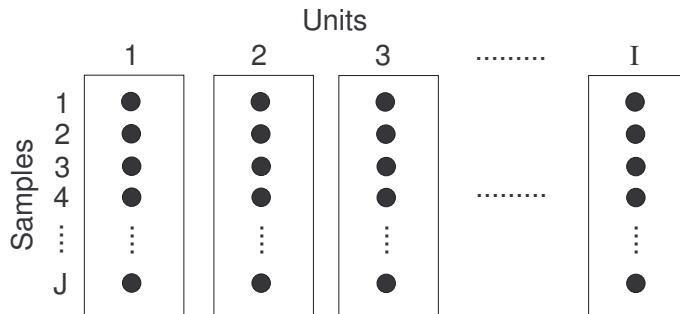


Figure 6.2 *Schematic illustration of the balanced one-way experimental layout discussed in section 6.1.3. Rectangles are experimental units and •’s indicate measurements.*

- The variances of the random effects can be estimated from combinations of the usual residual variance estimates from models for different aggregations.

These principles are useful for two reasons. Firstly, the classic mixed model analyses for designed experiments can be derived using them. Secondly, they provide a straightforward explanation for the degrees of freedom of the reference distributions used in mixed model hypothesis testing: the degrees of freedom are always those that apply to the aggregated model appropriate for testing hypotheses about the effect concerned. For example, in the CO₂ analysis, the hypothesis tests about the CO₂ effect were conducted with reference to an $F_{1,4}$ distribution, with these degrees of freedom being those appropriate to the aggregated model, used for the test.

To illustrate and reinforce these ideas, two further simple examples of the analysis of ‘standard designs’ will be covered, before returning to the general mixed models that are of more direct relevance to GAMs.

6.1.3 A single random factor

Consider an experimental design in which you have J measurements from each of I units, illustrated schematically in figure 6.2. Suppose that we are interested in establishing whether there are differences between the units, but are not really interested in the individual unit effects: rather in quantifying how much variability can be ascribed to differences between units. This would suggest using a random effect term for units.

A concrete example comes from animal breeding. For a breeding program to be successful, we need to know that variability in the trait, which is to be modified by the program, has a substantial enough genetic component that we can expect to alter it by selective breeding. Consider a pig breeding experiment in which I pregnant sows are fed a standard diet, and the fat content of J of each of their piglets is measured.

The interesting questions here are: is there evidence for litter to litter variability in fat content (which would be consistent with genetic variation in this trait) and if so how large is this component, in relation to the piglet to piglet variability within a litter? Notice here that we are not interested in how piglet fat content varies from particular sow to particular sow in the experiment, but rather in the variability between sows in general. This suggests using a random effect for sow, in a model for such data.

So, a suitable model is

$$y_{ij} = \alpha + b_i + \epsilon_{ij}, \quad (6.3)$$

where α is the fixed parameter for the population mean, $i = 1 \dots I$, $j = 1 \dots J$, $b_i \sim N(0, \sigma_b^2)$, $\epsilon_{ij} \sim N(0, \sigma^2)$ and all the b_i and ϵ_{ij} terms are mutually independent.

The first question of interest is whether $\sigma_b^2 > 0$, i.e. whether there is evidence that the factor variable contributes to the variance of the response. Formally we would like to test $H_0 : \sigma_b^2 = 0$ against $H_1 : \sigma_b^2 > 0$. To do this, simply note that the null hypothesis is exactly equivalent to $H_0 : b_i = 0 \forall i$, since both formulations of H_0 imply that the data follow,

$$y_{ij} = \alpha + \epsilon_{ij}. \quad (6.4)$$

Hence we can test the null hypothesis by using standard linear modelling methods to compare (6.4) to (6.3) by using an F-ratio test (ANOVA).

Fitting (6.3) to the data will also yield the usual estimate of σ^2 ,

$$\hat{\sigma}^2 = \text{RSS}_1 / (n - I),$$

where RSS_1 is the residual sum of squares from fitting the model to data, $n = IJ$ is the number of data, and $n - I$ is the residual degrees of freedom from this model fit.

So far the analysis with the mixed model has been identical to what would have been done with a fixed effects model, but now consider estimating σ_b^2 . The ‘obvious’ method of just using the sample variance of the \hat{b}_i ’s, ‘estimated’ by least squares, is not to be recommended, as such estimators are biased. Instead we make use of the model that results from averaging at each level of the factor:

$$\bar{y}_{i \cdot} = \alpha + b_i + \frac{1}{J} \sum_{j=1}^J \epsilon_{ij}.$$

Now define a new set of I random variables,

$$e_i = b_i + \frac{1}{J} \sum_{j=1}^J \epsilon_{ij}.$$

The e_i ’s are clearly mutually independent, since their constituent random variables are independent and no two e_i ’s share a constituent random variable. They are also zero mean normal random variables, since each is a sum of zero mean normal random variables. It is also clear that

$$\text{var}(e_i) = \sigma_b^2 + \sigma^2/J.$$

Hence, the model for the aggregated data becomes,

$$\bar{y}_{i\cdot} = \alpha + e_i, \quad (6.5)$$

where the e_i are i.i.d. $N(0, \sigma_b^2 + \sigma^2/J)$ random variables. If RSS₂ is the residual sum of squares when this model is fitted by least squares, then the residual variance estimate gives

$$\text{RSS}_2/(I - 1) = \hat{\sigma}_b^2 + \hat{\sigma}^2/J.$$

Re-arrangement implies that

$$\hat{\sigma}_b^2 = \text{RSS}_2/(I - 1) - \hat{\sigma}^2/J$$

is an unbiased estimator of σ_b^2 .

Now consider a practical industrial example. An engineering test for longitudinal stress in rails, involves measuring the time it takes certain ultrasonic waves to travel along the rail. To be a useful test, engineers need to know the average travel time for rails, and the variability to expect between rails, as well as the variability in the measurement process. The `Rail` data frame available with R package `nlme` provides 3 measurements of travel time for each of 6 randomly chosen rails. This provides an obvious application for model (6.3). First examine the data.

```
> library(nlme) # load nlme 'library', which contains data
> data(Rail)    # load data
> Rail
   Rail travel
1     1    55
2     1    53
3     1    54
4     2    26
5     2    37
.
.
.
17    6    85
18    6    83
```

Now fit model (6.3) as a fixed effects model, and use this model to test $H_0 : \sigma_b^2 = 0$, i.e. to test for evidence of differences between rails.

```
> m1 <- lm(travel ~ Rail, Rail)
> anova(m1)
Analysis of Variance Table

Response: travel
          Df Sum Sq Mean Sq F value    Pr(>F)
Rail       5 9310.5 1862.1 115.18 1.033e-09 ***
Residuals 12 194.0    16.2
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So there is strong evidence to reject the null hypothesis and accept rail to rail differences as real. As we saw theoretically, so far the analysis does not differ from that for

a fixed effects model, but to estimate σ_b^2 involves averaging at each level of the random effect and fitting model (6.5) to the resulting averages. R function aggregate will achieve the required averaging.

```
> rt <- # average over Rail effect
+ aggregate(data.matrix(Rail), by=list(Rail$Rail), mean)
> rt
  Group.1 Rail   travel
1       1   1 31.66667
2       2   2 50.00000
3       3   3 54.00000
4       4   4 82.66667
5       5   5 84.66667
6       6   6 96.00000
```

It is now possible to fit (6.5) and calculate $\hat{\sigma}_b$ and $\hat{\sigma}$, as described above:

```
> m0 <- lm(travel ~ 1, rt) # fit model to aggregated data
> sigb <- (summary(m0)$sigma^2 - summary(m1)$sigma^2/3)^0.5
> # sigb^2 is variance component for rail
> sig <- summary(m1)$sigma # sig^2 is resid. var. component
> sigb
[1] 24.80547
> sig
[1] 4.020779
```

So, there is a fairly large amount of rail to rail variability, whereas the measurement error is relatively small. In this case the model intercept, α , is confounded with the random effects, b_j , so α must be estimated from the fit of model (6.5).

```
> summary(m0)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 66.50      10.17   6.538 0.00125 **
```

Model checking proceeds by looking at residual plots, from the fits to both the original and the aggregated data, since, approximately, these should look like samples of i.i.d. normal random variables. However there would have to be a really grotesque violation of the normality assumption for the b_j 's, before you could hope to pick it up from examination of 6 residuals.

6.1.4 A model with two factors

Now consider an experiment in which each observation is grouped according to two factors. A schematic diagram of such a design is shown in figure 6.3. Suppose that one factor is to be modelled as a fixed effect and one as a random effect. A typical example is a randomized block design, for an agricultural field trial, testing different fertilizer formulations. The response variable would be yield of the crop concerned,

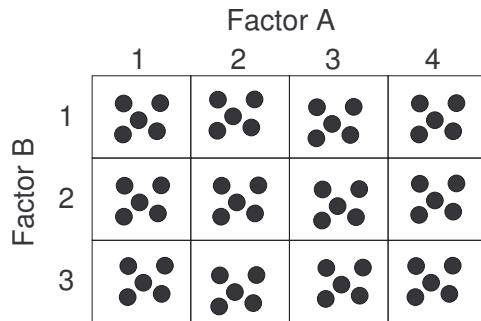


Figure 6.3 A schematic diagram of a two factor design of the sort discussed in section 6.1.4, with 3 levels of one factor, 4 levels of another and 5 observations for each combination of factor levels. Note that this diagram is not intended to represent the actual physical layout of any experiment.

assessed by harvesting at the end of the experiment. Because crop yields depend on many uncontrolled soil related factors, it is usual to arrange the experiment in blocks, within which it is hoped that the soil will be fairly homogeneous. Treatments are randomly arranged within the blocks. For example, a field site might be split into 4 adjacent blocks, with 15 plots in each block, each plot being randomly assigned one of five replicates of each of 3 fertilizer treatments. The idea is that differences within blocks should be smaller than differences between blocks — i.e. variability in conditions within a block will be smaller than variability across the whole field. A suitable model for the data would include a block effect, to account for this block to block variability, and since we are not in the least interested in the particular values of the block effects, but view them as representing variability in environment with location, it makes sense to treat them as random effects. The treatments, on the other hand, would be modelled as fixed effects, since the values of the treatment effects are fixed properties of the crop population in general. (If we repeated the experiment in a different location, the particular values of the block effects would be unrelated to the block effects in the first experiment, whereas the fertilizer effects should be very similar.)

So, a model for the k^{th} observation at level i of fixed effect A and level j of random effect B is

$$y_{ijk} = \mu + \alpha_i + b_j + (\alpha b)_{ij} + \epsilon_{ijk}, \quad (6.6)$$

where $b_j \sim N(0, \sigma_b^2)$, $(\alpha b)_{ij} \sim N(0, \sigma_{\alpha b}^2)$ and $\epsilon_{ijk} \sim N(0, \sigma^2)$, and all these random variables are mutually independent. μ is the overall population mean, the α_i 's are the I fixed effects for factor A, the b_j 's are the J random effects for factor B, and the $(\alpha b)_{ij}$'s are the IJ interaction terms for the interaction between the factors (an interaction term involving a random effect must also be a random term).

Testing $H_0 : \sigma_{\alpha b}^2 = 0$ is logically equivalent to testing $H_0 : (\alpha b)_{ij} = 0 \forall ij$, in a fixed effects framework. Hence this hypothesis can be tested by the usual ANOVA/F-

ratio test comparison of models, with and without the interaction terms. If RSS_1 now denotes the residual sum of squares from fitting (6.6) then:

$$\hat{\sigma}^2 = \text{RSS}_1 / (n - IJ).$$

In a purely fixed effects context it only makes sense to test for main effects if the interaction terms are not significant, and can hence be treated as zero. In the mixed effects case, because the interaction is a random effect, it is possible to make inferences about the main effects whether or not the interaction terms are significant. This can be done by averaging the K data at each level of the interaction. The averaging, together with model (6.6), implies the following model for the averages:

$$\bar{y}_{ij\cdot} = \mu + \alpha_i + b_j + (\alpha b)_{ij} + \frac{1}{K} \sum_{k=1}^K \epsilon_{ijk}.$$

Defining

$$e_{ij} = (\alpha b)_{ij} + \frac{1}{K} \sum_{k=1}^K \epsilon_{ijk},$$

it is clear that, since the e_{ij} 's are each sums of zero mean normal random variables, they are also zero mean normal random variables. Also, since the $(\alpha b)_{ij}$'s and ϵ_{ijk} 's are mutually independent random variables, and no $(\alpha b)_{ij}$ or ϵ_{ijk} is a component of more than one e_{ij} , the e_{ij} 's are mutually independent. Furthermore

$$\text{var}(e_{ij}) = \sigma_{\alpha b}^2 + \sigma^2/K.$$

Hence the simplified model is,

$$\bar{y}_{ij\cdot} = \mu + \alpha_i + b_j + e_{ij}, \quad (6.7)$$

where the e_{ij} 's are i.i.d. $N(0, \sigma_{\alpha b}^2 + \sigma^2/K)$ random variables. The null hypothesis, $H_0 : \alpha_i = 0 \forall i$, is tested by comparing the least squares fits of (6.7) and $\bar{y}_{ij\cdot} = \mu + b_j + e_{ij}$, in the usual way, by F-ratio testing. Similarly $H_0 : \sigma_b^2 = 0$ is logically equivalent to $H_0 : b_j = 0 \forall j$, and is hence tested by ANOVA comparison of (6.7) and $\bar{y}_{ij\cdot} = \mu + \alpha_i + e_{ij}$. The residual sum of squares for model (6.7), RSS_2 , say, is useful for unbiased estimation of the residual variance.

$$\hat{\sigma}_{\alpha b}^2 + \hat{\sigma}^2/K = \text{RSS}_2 / (IJ - I - J + 1)$$

and hence,

$$\hat{\sigma}_{\alpha b}^2 = \text{RSS}_2 / (IJ - I - J + 1) - \hat{\sigma}^2/K.$$

Averaging the data once more, over the levels of factor B, induces the model

$$\bar{y}_{\cdot j} = \mu + \frac{1}{I} \sum_{i=1}^I \alpha_i + b_j + \frac{1}{I} \sum_{i=1}^I e_{ij}.$$

Defining $\mu' = \mu + \frac{1}{I} \sum_i \alpha_i$ and $e_j = b_j + \frac{1}{I} \sum_i e_{ij}$ this model becomes

$$\bar{y}_{\cdot j} = \mu' + e_j, \quad (6.8)$$

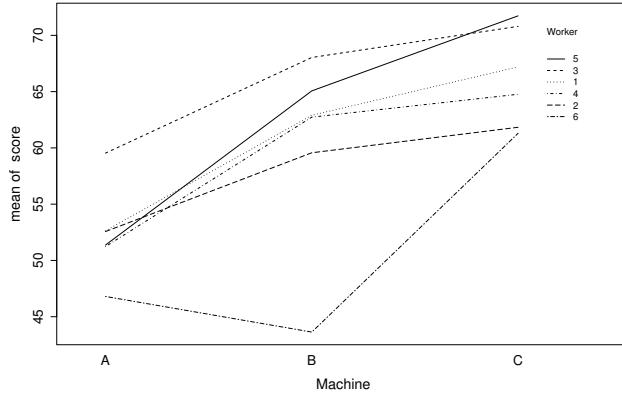


Figure 6.4 *Plot of the Machines data discussed in section 6.1.4.*

where

$$e_j \sim N(0, \sigma_b^2 + \sigma_{ab}^2/I + \sigma^2/(IK)).$$

Hence, if RSS_3 is the residual sum of squares of model (6.8), an unbiased estimator of σ_b^2 is given by

$$\hat{\sigma}_b^2 = \text{RSS}_3/(J - 1) - \hat{\sigma}_{ab}^2/I - \hat{\sigma}^2/(IK).$$

Now consider a practical example. The *Machines* data frame from the *nlme* package, contains data from an industrial experiment comparing 3 different machine types. The aim of the experiment was to determine which machine type resulted in highest worker productivity. 6 workers were randomly selected to take part in the trial, with each worker operating each machine 3 times (presumably after an appropriate period of training designed to eliminate any ‘learning effect’ from the data). The following produces the plot shown in figure 6.4

```
> library(nlme) # only needed in R, not S-PLUS
> data(Machines) # only needed in R, not S-PLUS
> names(Machines)
[1] "Worker" "Machine" "score"
> attach(Machines) # make data available without 'Machines$'
> interaction.plot(Machine, Worker, score)
```

From the experimental aims, it is clear that fixed machine effects and random worker effects are appropriate. We are interested in the effects of these particular machine types, but are only interested in the worker effects in as much as they reflect variability between workers in the population of workers using this type of machine. Put another way, if the experiment were repeated somewhere else (with different workers) we would expect the estimates of the machine effects to be quite close to the results obtained from the current experiment, while the individual worker effects

would be quite different (although with similar variability, we hope). So model (6.6) is appropriate, with α_i 's representing the fixed machine effects, b_j 's representing the random worker effects, and $(\alpha b)_{ij}$ representing the worker machine interaction (i.e. the fact that different workers may work better on different machines).

Fitting the full model, we can immediately test $H_0 : \sigma_{ab}^2 = 0$.

```
> m1 <- lm(score ~ Worker*Machine, Machines)
> m0 <- lm(score ~ Worker + Machine, Machines)
> anova(m0, m1)
Analysis of Variance Table
Model 1: score ~ Worker + Machine
Model 2: score ~ Worker + Machine + Worker:Machine
  Res.Df   RSS Df Sum of Sq    F    Pr(>F)
1     46 459.82
2     36 33.29 10    426.53 46.13 < 2.2e-16 ***
```

We must accept $H_1 : \sigma_{ab}^2 \neq 0$. There is very strong evidence for an interaction between machine and worker. σ^2 can now be estimated...

```
> summary(m1)$sigma^2
[1] 0.9246296
```

To examine the main effects we can aggregate at each level of the interaction,

```
Mach <- aggregate(data.matrix(Machines), by=
  list(Machines$Worker, Machines$Machine), mean)
Mach$Worker <- as.factor(Mach$Worker)
Mach$Machine <- as.factor(Mach$Machine)
```

and fit model (6.7) to the resulting data.

```
> m0 <- lm(score ~ Worker + Machine, Mach)
> anova(m0)
Analysis of Variance Table
Response: score
  Df Sum Sq Mean Sq F value    Pr(>F)
Worker      5 413.96   82.79  5.8232 0.0089495 **
Machine     2 585.09  292.54 20.5761 0.0002855 ***
Residuals  10 142.18   14.22
```

The very low p-values again indicate that $H_0 : \sigma_b^2 = 0$ and $H_0 : \alpha_1 = \alpha_2 = \alpha_3 = 0$ should be rejected in favour of the obvious alternatives. There is strong evidence for differences between machine types and for variability between workers. Going on to examine the fixed effect estimates, using standard fixed effects methods, indicates that machine 3 leads to substantially increased productivity.

Estimation of σ_{ab}^2 , the interaction variance, is straightforward.

```
> summary(m0)$sigma^2 - summary(m1)$sigma^2/3
[1] 13.90946
```

Finally, aggregate once more and fit (6.8) in order to estimate the worker variance component, σ_b^2 .

```
> M <- aggregate(data.matrix(Mach), by=list(Mach$Worker), mean)
> m00 <- lm(score ~ 1, M)
> summary(m00)$sigma^2 - (summary(m0)$sigma^2)/3
[1] 22.96118
```

Residual plots should be checked for $m1$, $m0$ and $m00$. If this is done, then it is tempting to try and see how robust the results are to the omission of Worker 6 on machine B (see figure 6.4), but this requires methods that can cope with unbalanced data.

6.1.5 Discussion

Although practical examples were presented in the preceding subsections, this theory for mixed models of balanced experimental data is primarily of theoretical interest for understanding the results used in classical mixed model ANOVA tables, and for motivating the use of particular reference distributions when conducting hypothesis tests for mixed models. In practice, the somewhat cumbersome analysis, based on aggregating data, would usually be eschewed in favour of using specialist mixed modelling software, such as that accessible via R function `lme` from the `nlme` library.

Before leaving the topic of balanced data altogether, it is worth noting the reason that ordinary linear model theory can be used for inference with balanced models. The first reason relates to the estimator of the residual variance, $\hat{\sigma}^2$. In ordinary linear modelling, $\hat{\sigma}^2$ does not depend in any way on the values of the model parameters β and is independent of $\hat{\beta}$ (see section 1.3.3). This fact is not altered if some elements of β are themselves random variables. Hence the usual estimator of $\hat{\sigma}^2$, based on the least squares estimate of a linear model, remains valid, and unbiased, for a linear mixed model.

The second reason relates to the estimators of the fixed effect parameters. In a fixed effects setting, consider two subsets β_1 and β_2 of the parameter vector, β , with corresponding model matrix columns \mathbf{X}_1 and \mathbf{X}_2 . If \mathbf{X}_1 and \mathbf{X}_2 are orthogonal, meaning that $\mathbf{X}_1^\top \mathbf{X}_2 = \mathbf{0}$, then the least squares estimators $\hat{\beta}_1$ and $\hat{\beta}_2$ will be independent: so inferences about β_1 do not depend in any way on the value of β_2 . This situation is unaltered if we now move to a mixed effects model, and suppose that the β_2 is actually a random vector. Hence, in a mixed model context, we can still use fixed effects least squares methods for inferences about any fixed effects whose estimators are independent of all the random effects in the model. So, when successively aggregating data (and models), we can use least squares methods to make inferences about a fixed effect as soon as the least squares estimator of that fixed effect becomes independent of all random effects in the aggregated model. Generally such independence only occurs for balanced data from designed experiments.

Generally, least squares methods are not useful for directly ‘estimating’ the values of random effects. This is, in part, because identifiability constraints are generally required in order to estimate effects, but imposing such constraints on random effects fundamentally modifies the model by changing the random effect distributions.

6.2 Linear mixed models in general

The general linear mixed model can conveniently be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \psi_{\theta}), \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \Lambda\sigma^2) \quad (6.9)$$

where ψ_{θ} is a positive definite covariance matrix for the random effects \mathbf{b} , and \mathbf{Z} is a matrix of fixed coefficients describing how the response variable, \mathbf{y} , depends on the random effects (it is a model matrix for the random effects). ψ_{θ} depends on some parameters, θ , which will be the prime target of statistical inference about the random effects (the exact nature of the dependence is model specific). Finally, Λ is a positive definite matrix which usually has a simple structure depending on few or no unknown parameters: it is sometimes used to model residual correlation, but is often simply an identity matrix.

As a simple example of this general formulation, recall the rails example from section 6.1.3. The model for the j^{th} response on the i^{th} rail is

$$y_{ij} = \alpha + b_i + \epsilon_{ij}, \quad b_i \sim N(0, \sigma_b^2), \quad \epsilon_{ij} \sim N(0, \sigma^2), \quad (6.10)$$

with all b_i ’s and ϵ_{ij} ’s mutually independent. There were 6 rails with 3 measurements on each. In the general linear mixed model form, the model is therefore

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \\ y_{41} \\ y_{42} \\ y_{43} \\ y_{51} \\ y_{52} \\ y_{53} \\ y_{61} \\ y_{62} \\ y_{63} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \left[\begin{array}{c} \alpha \end{array} \right] + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{31} \\ \epsilon_{32} \\ \epsilon_{33} \\ \epsilon_{41} \\ \epsilon_{42} \\ \epsilon_{43} \\ \epsilon_{51} \\ \epsilon_{52} \\ \epsilon_{53} \\ \epsilon_{61} \\ \epsilon_{62} \\ \epsilon_{63} \end{bmatrix}$$

where $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}_6\sigma_b^2)$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}_{18}\sigma^2)$. In this case the random effects parameter vector, θ , contains the single element σ_b^2 .

Returning to the general model, we could combine the residual vector and random effects into a single, non-independent, variable-variance residual vector, $\mathbf{e} = \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}$. It is obvious that \mathbf{e} is a zero mean multivariate normal vector, and its covariance matrix must be $\mathbf{Z}\psi_\theta\mathbf{Z}^\top + \mathbf{I}\sigma^2$. Hence (6.9) can be re-written as:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}, \quad \mathbf{e} \sim N(\mathbf{0}, \Sigma_\theta\sigma^2) \quad (6.11)$$

where $\Sigma_\theta = \mathbf{Z}\psi_\theta\mathbf{Z}^\top/\sigma^2 + \mathbf{I}$, and the subscript, θ , emphasizes the dependence of Σ_θ on the covariance parameter vector, θ (a dependence inherited from ψ_θ). So if θ were known then we could estimate β using the methods of section 1.8.4.

6.2.1 Estimation of linear mixed models

In general, of course, θ must be estimated, and maximum likelihood estimation provides the basic framework for doing this. As first discussed in section 1.8.3 the likelihood of β , θ and σ^2 will be

$$L(\beta, \theta, \sigma^2) = \frac{1}{\sqrt{(2\pi\sigma^2)^n |\Sigma_\theta|}} \exp \left[-(\mathbf{y} - \mathbf{X}\beta)^\top \Sigma_\theta^{-1} (\mathbf{y} - \mathbf{X}\beta) / (2\sigma^2) \right] \quad (6.12)$$

and maximizing L w.r.t. β , θ and σ^2 will provide $\hat{\beta}$, $\hat{\theta}$ and $\hat{\sigma}^2$. Usually this maximization can be simplified by *profiling* the likelihood. The idea is that since we already know, from section 1.8.4, exactly how to find the maximum likelihood estimates of β and σ^2 , for a given θ , these estimators can be plugged into the likelihood as implicit functions of θ , to yield the *profile* likelihood

$$L_p(\theta) = \frac{1}{\sqrt{(2\pi\hat{\sigma}_\theta^2)^n |\Sigma_\theta|}} \exp \left[-(\mathbf{y} - \mathbf{X}\hat{\beta}_\theta)^\top \Sigma_\theta^{-1} (\mathbf{y} - \mathbf{X}\hat{\beta}_\theta) / (2\hat{\sigma}_\theta^2) \right].$$

Here $\hat{\beta}_\theta$ is the maximum likelihood/least squares estimate of β for a given θ (calculated as in section 1.8.4) and $\hat{\sigma}_\theta^2$ is the corresponding m.l.e. of σ^2 . For purposes of numerical maximization, L_p can then be treated as a function of θ alone: whatever values maximize L_p w.r.t. θ will automatically maximize L , of course. Obviously, the maximum likelihood estimates of the other parameters are simply $\hat{\beta}_\theta$ and $\hat{\sigma}_\theta^2$.

Why is profiling useful? Usually we must resort to iterative numerical methods to do the maximization with respect to θ , and while we *could* use the same iterative methods with all the parameters it would be very computationally slow to do so, relative to using the quick, one-step, methods available for obtaining $\hat{\beta}$ and $\hat{\sigma}^2$, given any particular values for θ .

The iterative method used to maximize L_p (or more usually $l_p = \log L_p$) is just the multivariate version of Newton's method, as described at the beginning of section 4.6.1, for example. Starting with a parameter guess θ_0 , l_p is approximated by a quadratic sharing l_p 's value and first and second derivatives at θ_0 . The value of θ maximizing this approximating quadratic is used as the next estimate of θ , and the process is iterated to convergence, at which point the maxima of the approximating quadratic and l_p will coincide.

6.2.2 Directly maximizing a mixed model likelihood in R

Usually linear mixed modelling is performed using specialist routines, but it is worth seeing how straightforward it is to estimate a simple linear mixed model. For example, it is easy to write an R function to evaluate the log-likelihood of a linear mixed model, and maximize that log likelihood w.r.t. the variance parameters of the model. Consider the mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}$$

where the ϵ_i are i.i.d. $N(0, \sigma^2)$ and the b_i are i.i.d. $N(0, \sigma_b^2)$. For estimation purposes, let us use the parameters $\gamma_1 = \log(\sigma_b)$ and $\gamma_2 = \log(\sigma)$. The following function will evaluate the log likelihood, profiling out $\boldsymbol{\beta}$.

```
ll <- function(gamma, X, Z, y)
{ sigma.b <- exp(gamma[1])
  sigma <- exp(gamma[2])
  n <- nrow(Z)
  # evaluate covariance matrix for y
  V <- Z %*% t(Z) * sigma.b^2 + diag(n) * sigma^2
  L <- chol(V) # L' L=V
  # transform dependent linear model to indep.
  y <- backsolve(L, y, transpose=TRUE)
  X <- backsolve(L, X, transpose=TRUE)
  b <- coef(lm(y ~ X - 1)) # estimate fixed effects
  # evaluate log likelihood
  logLik <- -n/2 * log(2*pi) - sum(log(diag(L))) -
    sum((y - X %*% b)^2)/2
  attr(logLik, "fixed") <- b # allow retrieval of beta
  logLik
}
```

Given variance parameters, the routine simply evaluates the covariance matrix of the response data, finds its square root, and then uses the inverse transpose of this square root to transform the dependent data linear modelling problem, to a linear modelling problem for independent data, with constant variance: this allows $\hat{\boldsymbol{\beta}}$ (given the variance parameters) to be found immediately. Then all that remains is to evaluate the log likelihood.

The routine can easily be employed with a general purpose function optimizer, such as `optim` in R. As an example, the parameters of model (6.10) from sections 6.1.3 and 6.2 can be estimated as follows.

```
> options(contrasts=c("contr.treatment", "contr.treatment"))
> Z <- model.matrix(~ Rail$Rail - 1)
> X <- matrix(1, 18, 1)
>
> rail.mod <- optim(c(0, 0), ll, control=list(fnscale=-1),
  X=X, Z=Z, y=Rail$travel)
>
> exp(rail.mod$par) # variance components
```

```
[1] 22.629166 4.024072
> attr(ll(rail.mod$par,X,Z,Rail$travel),"fixed")
  X
66.5 # fixed effect estimate
```

See `?optim` for details about the operation of `optim`. Notice that these maximum likelihood estimates are similar to those obtained in section 6.1.3, but that the variance component estimates are reduced, particularly that for σ_b . This is because maximum likelihood estimates are not always unbiased, and those for variance components tend to be biased downwards. This issue will be revisited in the subsequent sections.

6.2.3 Inference with linear mixed models

Except in the balanced data case, inference about the fixed effects and random effects is approximate. For fixed effects it is usual to condition on the random effects, and then use fixed effect methods for inference. Inference about the random effects is based on likelihood theory, but in the case of hypothesis testing is ‘very approximate’.

Fixed effects

Since linear mixed models are fitted using maximum likelihood estimation, we could, in principle, use generalized likelihood ratio tests for all model comparison/hypothesis testing (see section 2.4). In practice, however, it is usually better to perform tests about model fixed effects by conditioning on $\hat{\theta}$ (i.e. treating the θ estimates as if they were known parameters). Then our model can usefully be treated as a fixed effects model for non-independent data with variable variance, of the form (6.11). This fixed effects model, and the corresponding data, can be transformed to an equivalent model for independent data, as described in section 1.8.4, at which point all the standard fixed effects theory, described in chapter 1, can be applied to drawing inferences about the fixed effect parameters, β .

The reason that this conditional approach is usually preferable to an approach based on asymptotic likelihood theory, is that the approximations involved in conditioning on $\hat{\theta}$ are usually better than those involved in using the large sample likelihood results at finite sample sizes. In fact, for balanced designs, where $\hat{\theta}$ is independent of the other estimators, exact tests can be obtained by the conditional approach.

If the, $\hat{\theta}$ conditional, ordinary linear model approach, is used for inference about the fixed effects, then there is an approximation that can improve it somewhat. When F-ratio testing the significance of model terms, the denominator (lower) degrees of freedom for the F distribution would at first sight seem to be given by the residual degrees of freedom from the fit of the $\hat{\theta}$ conditional linear model. This is perfectly valid as an approximation, and is fine in the large sample limit, but if an alternative scheme is used, it is possible to partly compensate for the variability neglected by

conditioning on $\hat{\theta}$. Specifically, it is better to use a method for approximating the denominator degrees of freedom which, in the balanced data case, will coincide with the denominator degrees of freedom that would have been used in the classical mixed modelling approach, described in section 6.1. The justification for this is provided by the fact that in the balanced data cases the REML (see section 6.2.5) and classical estimators of the fixed effects and variance components coincide, and must therefore follow the same distributions. But, of course, these distributions are known exactly in the balanced case. There is more than one method for approximating the denominator degrees of freedom which will meet the given criteria: see section 2.4 of Pinheiro and Bates (2000) for the one used in R package `n1me` (which is covered in section 6.2.5).

Inference about the random effects

Inference about the random effects is more difficult, and does rely on large sample likelihood results (see section 2.4). In particular, in the large sample limit

$$\begin{bmatrix} \log \hat{\sigma} \\ \boldsymbol{\theta} \end{bmatrix} \sim N \left(\begin{bmatrix} \log \sigma \\ \boldsymbol{\theta} \end{bmatrix}, \mathcal{I}_{\sigma, \theta}^{-1} \right) \quad (6.13)$$

where

$$\mathcal{I}_{\sigma, \theta} = \begin{bmatrix} \frac{\partial^2 l_p}{\partial(\log \sigma)^2} & \frac{\partial^2 l_p}{\partial \log \sigma \partial \theta_1} & \frac{\partial^2 l_p}{\partial \log \sigma \partial \theta_2} & \dots & \dots \\ \frac{\partial^2 l_p}{\partial \log \sigma \partial \theta_1} & \frac{\partial^2 l_p}{\partial \theta_1^2} & \frac{\partial^2 l_p}{\partial \theta_1 \partial \theta_2} & \dots & \dots \\ \frac{\partial^2 l_p}{\partial \log \sigma \partial \theta_2} & \frac{\partial^2 l_p}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 l_p}{\partial \theta_2^2} & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots \end{bmatrix}$$

— the empirical information matrix (i.e. it is evaluated at the parameter estimators). The parameterization in terms of $\log \sigma$ tends to give better finite sample results than using the un-logged variance. (6.13) allows approximate confidence intervals for the variance parameters to be found.

Models differing in their random effects structure can be compared using generalized likelihood ratio tests. Specifically, if l_1 is the maximized log likelihood of a model with p_1 parameters, and l_0 is the maximized log likelihood of a reduced version of the model (i.e. one with a simplified random effects structure) with p_0 parameters, then if the reduced model is correct

$$2(l_1 - l_0) \sim \chi^2_{p_1 - p_0}. \quad (6.14)$$

Unfortunately there is a problem with this approach, which undermines the utility of this large sample approximation. Namely, the null hypothesis specified by the smaller model will often restrict some of the variance parameters to the edge of the feasible parameter space (i.e. the null hypotheses involve assumptions that some variance components are zero). This violates the conditions under which the log of the likelihood ratio will have the stated distribution.

In practice, the most sensible approach is to treat the p-values from the log-ratio

tests as ‘very approximate’. If the p-value is very large, or very small, there is no practical difficulty about interpreting it, but on the borderline of significance, more care is needed. Pinheiro and Bates (2000) give a simulation approach which could be used in such cases.

For practical work, some other points of view can also be useful:

1. The assumptions underpinning the asymptotic distribution of the log likelihood ratio test statistic are not violated when using this result to find confidence intervals for parameters (by test inversion). Hence, if you can sensibly define the size of a variance component that would count as ‘practically negligible’, there is no problem in assessing whether this value is inside or outside the confidence interval for the parameters. Put another way: if you have confidence intervals for the parameters, it is rather seldom that inability to exactly test a variance component, for precise equality to zero, is likely to be a serious practical problem.
2. In a sense, a p-value is just a way of measuring whether a test statistic is ‘large’ or ‘small’. When testing whether variance components should be zero, there is nothing fundamentally wrong with the log likelihood ratio test statistic itself: it is calculating a p-value from it that is problematic. But suppose we obtained a p-value of 0.7 using (6.14) to compare two mixed models. What this means is that the log-likelihood ratio statistic is so small, that if it occurred in a test where all the GLRT assumptions were met, the p-value would be 0.7: this small a log likelihood ratio can not possibly provide any evidence for the alternative model, even though the p-value itself is not right for our test. Similar arguments apply for very high log-likelihood ratio statistics.

These points are not made in order to imply that there is no problem with conducting hypothesis tests about variance components, but merely to point out that the situation is very far from hopeless.

6.2.4 Predicting the random effects

Since the b_i ’s are random effects, we do not *estimate* them, as we would fixed parameters, but we may want to *predict* them. The usual way to do this is to evaluate the expected value of \mathbf{b} , given the data, \mathbf{y} (obviously the unconditional expected value is $\mathbf{0}$, by construction of the model). The most straightforward derivation of $\mathbb{E}(\mathbf{b}|\mathbf{y})$ uses the following general result from probability.

If \mathbf{x} and \mathbf{z} are random vectors, with the joint normal distribution

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \sim N \left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_z \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xz} \\ \boldsymbol{\Sigma}_{zx} & \boldsymbol{\Sigma}_{zz} \end{bmatrix} \right)$$

(where $\boldsymbol{\Sigma}_{xz} = \boldsymbol{\Sigma}_{zx}^T$), then the mean of \mathbf{x} given \mathbf{z} is

$$\mathbb{E}(\mathbf{x}|\mathbf{z}) = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xz}\boldsymbol{\Sigma}_{zz}^{-1}(\mathbf{z} - \boldsymbol{\mu}_z) \quad (6.15)$$

and the covariance matrix of \mathbf{x} given \mathbf{z} is

$$\Sigma_{x|z} = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx}. \quad (6.16)$$

Now consider the general linear mixed effects model. We have that

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{y} \end{bmatrix} \sim N \left(\begin{bmatrix} \mathbf{0} \\ \mathbf{X}\beta \end{bmatrix}, \begin{bmatrix} \boldsymbol{\psi} & \Sigma_{yb} \\ \Sigma_{yb} & \Sigma_\theta \sigma^2 \end{bmatrix} \right)$$

where $\Sigma_\theta = \mathbf{Z}\boldsymbol{\psi}\mathbf{Z}^\top/\sigma^2 + \mathbf{I}$ (the subscript serving as a reminder of the dependence of this matrix on the variance parameters θ). From (6.15) we have $\mathbb{E}(\mathbf{y}|\mathbf{b}) = \mathbf{X}\beta + \Sigma_{yb}\boldsymbol{\psi}^{-1}(\mathbf{b} - \mathbf{0})$, while the original model structure gives $\mathbb{E}(\mathbf{y}|\mathbf{b}) = \mathbf{X}\beta + \mathbf{Z}\mathbf{b}$. Hence

$$\mathbf{Z}\mathbf{b} = \Sigma_{yb}\boldsymbol{\psi}^{-1}\mathbf{b},$$

and for this to hold for all \mathbf{b} requires that

$$\Sigma_{yb} = \mathbf{Z}\boldsymbol{\psi}.$$

Now we can use (6.15), once more, to obtain the mean for \mathbf{b} given \mathbf{y} ,

$$\mathbb{E}(\mathbf{b}|\mathbf{y}) = \boldsymbol{\psi}\mathbf{Z}^\top\Sigma_\theta^{-1}(\mathbf{y} - \mathbf{X}\beta)/\sigma^2,$$

which is estimated by plugging the estimates of the parameters β , θ and σ^2 into the expression on the r.h.s. to obtain:

$$\hat{\mathbf{b}} = \hat{\boldsymbol{\psi}}\mathbf{Z}^\top\Sigma_{\hat{\theta}}^{-1}(\mathbf{y} - \mathbf{X}\hat{\beta})/\hat{\sigma}^2 \quad (6.17)$$

(where $\hat{\boldsymbol{\psi}}$ follows from $\hat{\theta}$.)

For completeness, note that from the general result (6.16), the covariance matrix for \mathbf{b} given \mathbf{y} is

$$\Sigma_{b|y} = \boldsymbol{\psi} - \boldsymbol{\psi}\mathbf{Z}^\top\Sigma_\theta^{-1}\mathbf{Z}\boldsymbol{\psi}/\sigma^2,$$

and the distribution of \mathbf{b} conditional on \mathbf{y} is multivariate normal, of course.

Model ‘fitted values’ are predicted as

$$\hat{\mathbf{y}} = \widehat{\mathbb{E}(\mathbf{y}|\hat{\mathbf{b}})} = \mathbf{X}\hat{\beta} + \mathbf{Z}\hat{\mathbf{b}},$$

and residuals as

$$\hat{\epsilon} = \mathbf{y} - \hat{\mathbf{y}}.$$

Both are useful for model checking.

Note that, in practice, computation of $\hat{\mathbf{b}}$ often uses the results of section 6.2.6.

6.2.5 REML

The maximum likelihood estimators of variance parameters, tend to become quite badly biased, as the number of fixed parameters in a model increases, particularly if sample sizes are not large. This section covers an effective approach for dealing

with this issue. The problem itself is best seen in the context of estimating a single variance component, by maximum likelihood estimation: consider estimating the residual variance of a fixed effect linear model by maximum likelihood.

The likelihood of the parameters σ and β of the model $\mathbf{y} = \mathbf{X}\beta + \epsilon$, $\epsilon \sim N(\mathbf{0}, \mathbf{I}\sigma^2)$ is

$$L(\beta, \sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[\frac{-\|\mathbf{y} - \mathbf{X}\beta\|^2}{2\sigma^2}\right],$$

so that the log-likelihood is

$$l(\beta, \sigma) = \log(L(\beta, \sigma)) = -\frac{n}{2} \log(2\pi) - n \log(\sigma) - \|\mathbf{y} - \mathbf{X}\beta\|^2/(2\sigma^2).$$

Differentiating l w.r.t. σ , and setting the result to zero, yields the m.l.e.

$$\frac{\partial l}{\partial \sigma} = -\frac{n}{\sigma} + \|\mathbf{y} - \mathbf{X}\beta\|^2/\sigma^3 = 0 \Rightarrow \hat{\sigma}^2 = \|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2/n.$$

Comparing this with the unbiased estimator (1.8), derived in section 1.3.3, it is clear that

$$\mathbb{E}(\hat{\sigma}^2) = \frac{n-p}{n} \sigma^2,$$

where p is the dimension of β , and n the dimension of \mathbf{y} . Unfortunately this tendency to underestimate variance components, increasingly badly as p increases, is a general feature of maximum likelihood estimators (from normal distribution derived likelihoods). The difficulty is that the estimators take no account of the degrees of freedom ‘lost’ by estimating the fixed effects.

One way of alleviating this problem is to estimate the variance components, not by maximizing the likelihood, but by maximizing the so called ‘REML’ criterion, where ‘REML’ sometimes stands for ‘REstricted Maximum Likelihood’ (Patterson and Thompson, 1971). The REML approach measures the fit of the variance parameters, not from the joint likelihood of the variance parameters and β , but rather by the (scaled) average of the likelihood over all the possible values of β . This average likelihood is the REML criterion, and it is maximised to find the variance parameters. Since this approach does not ‘lose’ any degrees of freedom by estimating fixed effects, it tends not to suffer the under-estimation problems of maximum likelihood variance component estimation. Formally, if $L(\beta, \theta, \sigma^2)$ is the likelihood for the parameters of a mixed effects linear model, then

$$L_R(\theta, \sigma^2) = \int L(\beta, \theta, \sigma^2) d\beta$$

is the equivalent REML criterion.

The numerical methods for maximizing L_R are similar to those used for maximizing the profiled likelihood under maximum likelihood estimation. Once variance components have been estimated by REML, we can condition on the estimated θ and σ^2 , write the mixed model as (6.11), and use the methods of sections 1.8.4 to estimate the fixed parameters β . Given estimates of all the model parameters, inference about the fixed effects proceeds as for the MLE case.

Some care is needed for inference about the random effects. It turns out that L_R behaves much like the likelihood, L , from the point of view of inference. Specifically, the expressions for the asymptotic distributions $\hat{\theta}$ and $\hat{\sigma}^2$ are similar to the expressions under MLE, but with the log of L_R , l_R , replacing the log-likelihood, l . The one major difference is that generalized likelihood ratio testing, with l_R replacing l , only works if the models being compared have **identical fixed effects structures**.

The explicit form of the REML criterion

At first sight, the integral in the REML criterion, L_R , appears a little intimidating, but in fact it is straightforward to obtain L_R in closed form. Purely for notational compactness define $\Sigma = \Sigma_\theta \sigma^2$. Then plugging the likelihood (6.12) into the expression for L_R , we have

$$L_R(\boldsymbol{\theta}, \sigma^2) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \int \exp [-(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})/2] d\boldsymbol{\beta}$$

The quadratic form can be expanded as

$$\begin{aligned} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ &\quad + (\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta})^\top \Sigma^{-1} (\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}) \\ &\quad + 2(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) + (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \mathbf{X}^\top \Sigma^{-1} \mathbf{X} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \end{aligned}$$

where the least squares estimate of $\boldsymbol{\beta}$, given Σ , is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{y}$, and the final equality results from the fact that

$$\begin{aligned} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}(\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{y})^\top \Sigma^{-1} \mathbf{X} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \\ &= (\mathbf{y}^\top \Sigma^{-1} \mathbf{X} - \mathbf{y}^\top \Sigma^{-1} \mathbf{X})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = 0. \end{aligned}$$

Hence

$$L_R(\boldsymbol{\theta}, \sigma^2) = \frac{e^{-(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})/2}}{\sqrt{(2\pi)^n |\Sigma|}} \int e^{-(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^\top \mathbf{X}^\top \Sigma^{-1} \mathbf{X} (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})/2} d\boldsymbol{\beta}$$

Now if the expression inside the integral were divided by $(2\pi)^{p/2} |\mathbf{X}^\top \Sigma^{-1} \mathbf{X}|^{-1/2}$ (where p is the dimension of $\boldsymbol{\beta}$), then it would be immediately recognizable as a multivariate normal p.d.f., and the integral would be 1. Hence the integral is in fact $(2\pi)^{p/2} |\mathbf{X}^\top \Sigma^{-1} \mathbf{X}|^{-1/2}$, and

$$L_R(\boldsymbol{\theta}, \sigma^2) = \frac{e^{-(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})/2}}{\sqrt{(2\pi)^n |\Sigma|}} \sqrt{\frac{(2\pi)^p}{|\mathbf{X}^\top \Sigma^{-1} \mathbf{X}|}}.$$

Clearly the log REML criterion is therefore

$$l_R = \frac{p-n}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma| - \frac{1}{2} \log |\mathbf{X}^\top \Sigma^{-1} \mathbf{X}| - \frac{1}{2} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}).$$

6.2.6 A link with penalized regression

If ψ and σ^2 are known, or we condition on their estimates, then it turns out that the predicted values of the random effects, $\hat{\mathbf{b}}$, (see equation 6.17) and the maximum likelihood estimates of the parameters, $\hat{\beta}$, are the values minimizing the penalized sum of squares

$$\frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b}\|^2 + \mathbf{b}^\top \psi_\theta^{-1} \mathbf{b}.$$

This was proven by Harville (1976, 1977), but a much simpler proof follows Pinheiro and Bates (2000).

Start by writing down, $f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b}) = f_{\beta, \sigma^2}(\mathbf{y} | \mathbf{b}) f_\theta(\mathbf{b})$, the joint p.d.f. of \mathbf{y} and \mathbf{b} , from which the marginal p.d.f. of \mathbf{y} can be obtained by integration w.r.t. \mathbf{b} :

$$f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b}) = \frac{1}{(2\pi\sigma^2)^n} e^{-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b}\|^2} \frac{1}{(2\pi)^q} e^{-\frac{1}{2} \mathbf{b}^\top \psi_\theta^{-1} \mathbf{b}}.$$

Finding any matrix square root, \mathbf{B} , such that $\mathbf{B}^\top \mathbf{B} = \psi_\theta^{-1} \sigma^2$, we then have

$$f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b}) = \frac{|\mathbf{B}|}{(2\pi\sigma^2)^{\frac{n+q}{2}}} e^{-\frac{1}{2\sigma^2} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\mathbf{b}\|^2},$$

where $\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}$, $\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{0} \end{bmatrix}$ and $\tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{Z} \\ \mathbf{B} \end{bmatrix}$. Now let $\hat{\mathbf{b}}_\beta = (\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}})^{-1} \tilde{\mathbf{Z}}^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta)$, be the value of \mathbf{b} maximizing $f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b})$ for any given β . Since $f(\mathbf{b} | \mathbf{y}) = f(\mathbf{y}, \mathbf{b}) / f(\mathbf{y})$ it is clear that $f(\mathbf{b} | \mathbf{y})$ is also maximized by $\hat{\mathbf{b}}_\beta$. i.e. these are the ‘posterior modes’ of the random effects. Furthermore, since $f(\mathbf{b} | \mathbf{y})$ is a normal distribution, its mean is the same as its mode, so $\hat{\mathbf{b}}_\beta = \mathbb{E}(\mathbf{b} | \mathbf{y})$, assuming β is known.

Now turning to the fixed effects, β , it helps to re-write the norm in the joint p.d.f. as

$$\begin{aligned} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\mathbf{b}\|^2 &= \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta + \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta - \tilde{\mathbf{Z}}\mathbf{b}\|^2 \\ &= \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta\|^2 + \|\tilde{\mathbf{Z}}(\hat{\mathbf{b}}_\beta - \mathbf{b})\|^2 \end{aligned}$$

The final equality above is geometrically obvious, since the residual vector, from regressing $\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta$ on the columns of $\tilde{\mathbf{Z}}$, is clearly orthogonal to $\tilde{\mathbf{Z}}$, but in any case it follows from:

$$\begin{aligned} (\hat{\mathbf{b}}_\beta - \mathbf{b})^\top \tilde{\mathbf{Z}}^\top [\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta] &= (\hat{\mathbf{b}}_\beta - \mathbf{b})^\top \tilde{\mathbf{Z}}^\top [\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}})^{-1} \tilde{\mathbf{Z}}^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta)] \\ &= (\hat{\mathbf{b}}_\beta - \mathbf{b})^\top [\tilde{\mathbf{Z}}^\top \tilde{\mathbf{y}} - \tilde{\mathbf{Z}}^\top \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta)] \\ &= (\hat{\mathbf{b}}_\beta - \mathbf{b})^\top \mathbf{0} = 0. \end{aligned}$$

Hence the joint p.d.f. of \mathbf{y} and \mathbf{b} can be written

$$f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b}) = \frac{\text{abs}|\mathbf{B}|}{(2\pi\sigma^2)^{\frac{n+q}{2}}} e^{-\frac{1}{2\sigma^2} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta\|^2 - \frac{1}{2\sigma^2} (\mathbf{b} - \hat{\mathbf{b}}_\beta)^\top \tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}} (\mathbf{b} - \hat{\mathbf{b}}_\beta)}.$$

Now the marginal p.d.f. of \mathbf{y} is $f_{\beta, \theta, \sigma^2}(\mathbf{y}) = \int f_{\beta, \theta, \sigma^2}(\mathbf{y}, \mathbf{b}) d\mathbf{b}$ and from the properties of a normal p.d.f. we have that

$$\frac{|\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}}|^{1/2}}{(2\pi\sigma^2)^{q/2}} \int e^{-\frac{1}{2\sigma^2} (\mathbf{b} - \hat{\mathbf{b}}_\beta)^\top \tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}} (\mathbf{b} - \hat{\mathbf{b}}_\beta)} d\mathbf{b} = 1$$

so that

$$f_{\beta, \theta, \sigma^2}(\mathbf{y}) = \frac{\text{abs}|\mathbf{B}|}{(2\pi\sigma^2)^{\frac{n}{2}} |\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}}|^{\frac{1}{2}}} e^{-\frac{1}{2\sigma^2} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta\|^2}$$

which, when treated as a function of the parameters, is the likelihood of β , θ and σ^2 . Clearly the likelihood is maximized by the values of β minimizing

$$\|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\hat{\mathbf{b}}_\beta\|^2,$$

and hence the posterior modes $\hat{\mathbf{b}}$ and the maximum likelihood estimates $\hat{\beta}$ are the values of \mathbf{b} and β simultaneously minimizing

$$\|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta - \tilde{\mathbf{Z}}\mathbf{b}\|^2 = \frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b}\|^2 + \mathbf{b}^\top \psi_\theta^{-1} \mathbf{b}.$$

6.2.7 The EM algorithm

As discussed in section 6.2.1 and demonstrated in section 6.2.2, the likelihood of a linear mixed model can be maximized directly by Newton's method. Newton's method works very well *given good starting values* for the parameters, so that the initial quadratic approximation is a good one. It can be less effective if starting values are poor. This raises the question of how to find good starting values, and here an alternative optimization approach, known as the *EM algorithm* is useful.

The idea is that we base parameter estimation on a ‘likelihood’, L_b , in which the random effects, \mathbf{b} , are treated as if they were data. That is we base L_b on the joint distribution of the observed data \mathbf{y} and the unobserved random effects, \mathbf{b} , rather than on the marginal distribution of \mathbf{y} alone. For the linear mixed model we have

$$L_b(\theta, \beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b}\|^2} \frac{1}{(2\pi)^{q/2} |\psi_\theta|^{1/2}} e^{-\frac{1}{2} \mathbf{b}^\top \psi_\theta^{-1} \mathbf{b}}.$$

(where $q = \dim(\mathbf{b})$). Let $l_b = \log L_b$. Normally we would integrate L_b w.r.t. \mathbf{b} to obtain the likelihood of the parameters, and then maximize that, but the EM algorithm takes a different approach. Starting from parameter guesses, $\hat{\theta}, \hat{\beta}, \hat{\sigma}^2$, the following steps are iterated:

1. Find the distribution of $\mathbf{b}|\mathbf{y}$ according to the current parameter estimates.
2. Treating the distribution from 1 as fixed (rather than depending on the θ, β, σ^2), we can now find an expression for $\mathbb{E}_{\mathbf{b}|\mathbf{y}}\{l_b(\theta, \beta, \sigma^2)\}$ as a function of θ, β, σ^2 , using the distribution from 1. The \mathbf{y} are treated as fixed, here. (This is the E-step.)
3. Maximize the expression for $\mathbb{E}_{\mathbf{b}|\mathbf{y}}\{l_b(\theta, \beta, \sigma^2)\}$ w.r.t. the parameters to obtain updated estimates $\hat{\theta}, \hat{\beta}, \hat{\sigma}^2$. (This is the M-step.)

Note that the expectation in step 2 is taken with respect to the *fixed* distribution from step 1, which depends on the current parameter *estimates*. When evaluating $\mathbb{E}_{\mathbf{b}|\mathbf{y}}\{l_b(\theta, \beta, \sigma^2)\}$, we view $l_b(\theta, \beta, \sigma^2)$ as a function of θ, β, σ^2 , but do not treat the distribution of $\mathbf{b}|\mathbf{y}$ as depending on these parameters.

There are two key points about this algorithm.

1. Each step of the algorithm can be shown to increase the log-likelihood $l(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ (until the MLE is reached).
2. $\mathbb{E}_{\mathbf{b}|\mathbf{y}}\{l_b(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma^2)\}$ is often much easier to evaluate and maximize than $l(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma^2)$ itself.

The algorithm is a very reliable way of maximizing the likelihood and can also be used with the REML criterion, but is rather slow to converge near the MLEs. Hence it is often best to start optimization off using EM steps, before switching to Newton's method (Pinheiro and Bates, 2000).

To appreciate exactly what the E-step involves, it helps to write out l_b :

$$l_b(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma^2) = c - n \log(\sigma) - \frac{1}{2} \log |\psi_{\theta}| - \frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{b}\|^2}{2\sigma^2} - \frac{\mathbf{b}^T \psi_{\theta}^{-1} \mathbf{b}}{2},$$

where c is a parameter independent constant. Now the E-step simply replaces the two final terms on the right hand side, with their expectation according to the current estimate of the distribution of $\mathbf{b}|\mathbf{y}$ (treating \mathbf{y} as fixed). In fact, evaluation of these expectations requires only knowledge of the current estimates of $\mathbb{E}(\mathbf{b}|\mathbf{y})$ and the current estimate of the covariance matrix $\Sigma_{\mathbf{b}|\mathbf{y}}$. Expressions for both are given in section 6.2.4, although the methods of section 6.2.6 would usually be preferred for evaluating the estimates of $\mathbb{E}(\mathbf{b}|\mathbf{y})$ (the posterior modes). It turns out that the dependence of $\mathbb{E}_{\mathbf{b}|\mathbf{y}}(l_b)$ on $\boldsymbol{\theta}$ is through two simple expressions involving ψ^{-1} and no other parameters: this makes maximization w.r.t. $\boldsymbol{\theta}$ rather straightforward.

The EM algorithm applies much more widely than linear mixed models, and the key reference is Dempster et al. (1977). As ever, Davison (2003) provides a very clear explanation and examples.

6.3 Linear mixed models in R

There are several packages for linear mixed modelling in R, of which `nlme` and latterly `lme4` are particularly noteworthy. In this section I will concentrate on the `nlme` package: actually this package provides much more than just *linear* mixed models, but the rest is beyond the scope of this book (see Pinheiro and Bates, 2000).

The main model fitting function of interest is called `lme`. A call to the `lme` function is similar to a call to `lm`, except that an extra argument specifying the random effects structure must also be supplied to the model. By default, `lme` works with a slightly more restricted structure for linear mixed models than the very general form (6.9), given in section 6.2. Specifically `lme` assumes that your data are grouped according to the levels of some factor(s), and that the same random effects structure is required for each group, with random effects independent between groups. Assuming just one level of grouping, the model for the data in the i^{th} group is then

$$\mathbf{y}_i = \mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{b}_i + \boldsymbol{\epsilon}_i, \quad \mathbf{b}_i \sim N(\mathbf{0}, \psi), \quad \boldsymbol{\epsilon}_i \sim N(\mathbf{0}, \Lambda \sigma^2). \quad (6.18)$$

Careful attention should be paid to which terms have an i subscript, and hence depend on group, and which are common to all groups. Note, in particular, that β , ψ and Λ , the unknowns for the fixed effects and random effects respectively, are assumed to be the same for all groups, as is the residual variance, σ^2 . This form of mixed effects model, which was introduced by Laird and Ware (1982), is a sensible default, because it is very common in practical applications, and because model fitting for this structure is more efficient than for the general form (6.9). However it is important to realize that (6.18) is only a special form of (6.9). Indeed if we treat all the data as belonging to a single group then (6.18) is exactly (6.9), with no special structure imposed.

Because of `lme`'s default behaviour you need to provide two parts to the random effects specification: a part that specifies Z_i and a part specifying the grouping factor(s). By default, ψ is assumed to be a general positive definite matrix to be estimated, but it is also possible to specify that it should have a more restricted form (for example $I\sigma_b^2$). The simplest way to specify the random effects structure is with a one sided formula. For example $\sim x | g$ would set up Z_i according to the $\sim x$ part of the formula while the levels of the factor variable, g , would be used to split the data into groups (i.e. the levels of g are effectively the group index, i). The random effects formula is one sided, because there is no choice about the response variable — it must be whatever was specified in the fixed effects formula. So an example call to `lme` looks something like this:

```
> lme(y ~ x + z, dat, ~ x | g)
```

where the response is y , the fixed effects depend on x and z , the random effects depend only on x , the data are grouped according to factor g , and all data are in data frame `dat`. An alternative way of specifying the same model is:

```
> lme(y ~ x + z, dat, list(g = ~ x))
```

and in fact this latter form is the one we will eventually use with GAMMs.

As an example, model (6.10), from sections 6.1.3 and 6.2, can easily be fitted.

```
> library(nlme)
> lme(travel ~ 1, Rail, list(Rail = ~ 1))
Linear mixed-effects model fit by REML
  Data: Rail
  Log-restricted-likelihood: -61.0885
  Fixed: travel ~ 1
(Intercept)
          66.5

Random effects:
Formula: ~ 1 | Rail
        (Intercept) Residual
StdDev:     24.80547 4.020779

Number of Observations: 18
Number of Groups: 6
```

Note that, because REML has been used for estimation, the results are identical to those obtained in section 6.1.3. If we had used MLE, by specifying `method="ML"` in the call to `lme`, then the results would have corresponded to those obtained in section 6.2.2.

6.3.1 Tree Growth: an example using `lme`

The `nlme` package includes a data frame called `Loblolly`, containing growth data on Loblolly pine trees. `height`, in feet (data are from the US), and `age`, in years, are recorded for 14 individual trees. A factor variable `Seed`, with 14 levels, indicates the identity of individual trees. Interest lies in characterizing the, population level, mean growth trajectory of the Loblolly Pines, but it is clear that we would expect a good deal of tree to tree variation, and probably also some degree of autocorrelation in the random component of height.

From examination of data plots, the following initial model might be appropriate for the i^{th} measurement on the j^{th} tree:

$$\begin{aligned} \text{height}_{ji} = & \beta_0 + \beta_1 \text{age}_{ji} + \beta_2 \text{age}_{ji}^2 + \beta_3 \text{age}_{ji}^3 \\ & + b_0 + b_{j1} \text{age}_{ji} + b_{j2} \text{age}_{ji}^2 + b_{j3} \text{age}_{ji}^3 + \epsilon_{j,i} \end{aligned}$$

where the $\epsilon_{j,i}$ are zero mean normal random variables, with correlation given by $\rho(\epsilon_{j,i}, \epsilon_{j,i-1}) = \phi$, and ϕ is an unknown parameter: this $\epsilon_{j,i}$ model is an autoregressive model of order 1 (if the ages had been unevenly spaced then a continuous generalization of this is available, which would then be more appropriate). The ϵ terms are independent between different trees. As usual β denotes the fixed effects and $b_j \sim N(0, \psi)$ denotes the random effects.

This model can be estimated using `lme`, but to avoid convergence difficulties in the following analysis, two preparatory steps are useful. Firstly, it is worth centering the `age` variable as follows:

```
Loblolly$age <- Loblolly$age - mean(Loblolly$age)
```

without such centering, polynomial terms can become highly correlated which can cause numerical difficulties. An alternative to centering would be to use the `poly` function to set up an orthogonal polynomial basis.

Secondly, for this analysis the default fitting method fails, without some adjustment. `lme` fits start by using the EM algorithm to get reasonably close to the optimal parameter estimates, and then switch to Newton's method, which converges more quickly. The number of EM steps to take, and the maximum number of Newton steps to allow, are both controllable via the `control` argument of `lme`. The `lmeControl` function offers a convenient way of producing a `control` list, with some elements modified from their default[†]. For example

[†] At time of writing, not all arguments of `lmeControl` are actually used to control fitting: if in doubt, see the source code.

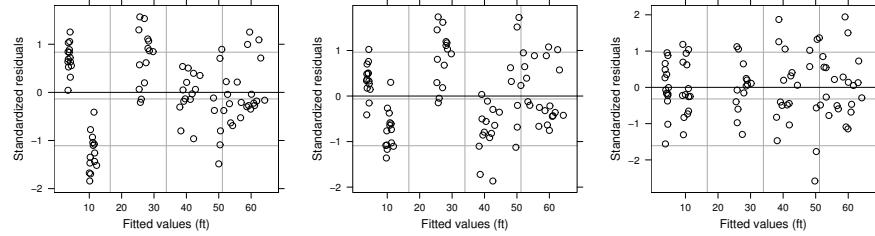


Figure 6.5 Default residual plots for models m_0 , m_1 and m_2 (left to right). There is a clear trend in the mean of the residuals for the first two models, which model m_2 eliminates.

```
lmc <- lmeControl(niterEM=500,msMaxIter=100)
```

produces a `control` list in which the number of EM iterations is set to 500, and the maximum number of Newton iterations is set to 100. For future reference, note that `niterEM` should rarely be increased from its default 0 when calling `gamm`.

The model can now be estimated.

```
m0 <- lme(height ~ age + I(age^2) + I(age^3), Loblolly,
           random=list(Seed=~age+I(age^2)+I(age^3)),
           correlation=corAR1(form=~age|Seed), control=lmc)
```

The `random` argument specifies that there should be a different cubic term for each tree, while the `correlation` argument specifies a continuous autoregressive model for the residuals for each tree. `form=~age|Seed` indicates that `age` is the variable determining the ordering of residuals, and that the correlation applies within measurements made on one tree, but not between measurements on different trees.

The command

```
plot(m0)
```

produces the default residual plot shown in the left panel of figure 6.5. The plot shows a clear trend in the mean of the residuals: the model seems to underestimate the first group of measurements, made at age 5, and then overestimate the next group, made at age 10, before somewhat underestimating the next group, which correspond to year 15. This suggests a need for a more flexible model, so fourth and fifth order polynomials were also tried.

```
m1 <- lme(height ~ age+I(age^2)+I(age^3)+I(age^4), Loblolly,
           list(Seed=~age+I(age^2)+I(age^3)),
           cor=corAR1(form=~age|Seed), control=lmc)
plot(m1)
m2 <- lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
           Loblolly, list(Seed=~age+I(age^2)+I(age^3)),
           cor=corAR1(form=~age|Seed), control=lmc)
plot(m2)
```

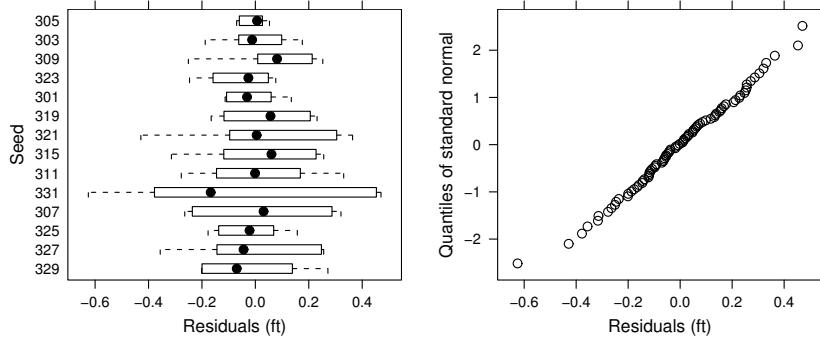


Figure 6.6 *Further residual plots for model m2. The left panel shows boxplots of the residuals for each tree, while the right plot is a normal QQ plot for the residuals.*

The resulting residuals plots are shown in the middle and right panels of figure 6.5. m1 does lead to a slight improvement, but only m2 is really satisfactory. Further model checking plots can now be produced for m2.

```
plot(m2, Seed~resid(.))
qqnorm(m2, ~resid(.))
qqnorm(m2, ~ranef(.))
```

The resulting plots are shown in figures 6.6 and 6.7, and suggest that the model is reasonable.

An obvious question is whether the elaborate model structure, with random cubic and autocorrelated within tree errors, is really required. First try dropping the auto-correlation component.

```
> m3<-lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
+       Loblolly, list(Seed=~age+I(age^2)+I(age^3)), control=lmc)
> anova(m3,m2)
      Model df     AIC     BIC   logLik   Test L.Ratio p-value
m3     1 17  250.46 290.53 -108.23
m2     2 18  239.36 281.78 -101.68 1 vs 2 13.1041    3e-04
```

The `anova` command is actually conducting a generalized likelihood ratio test here, which rejects m3 in favour of m2. Note that in this case the GLRT assumptions are met: m3 is effectively setting the autocorrelation parameter ϕ to zero, which is in the middle of its possible range, not on a boundary. `anova` also reports the AIC for the models, which also suggest that m2 is preferable. There seems to be strong evidence for auto-correlation in the within tree residuals.

Perhaps the random effects model could be simplified, by dropping the dependence of tree specific growth on the cube of age.

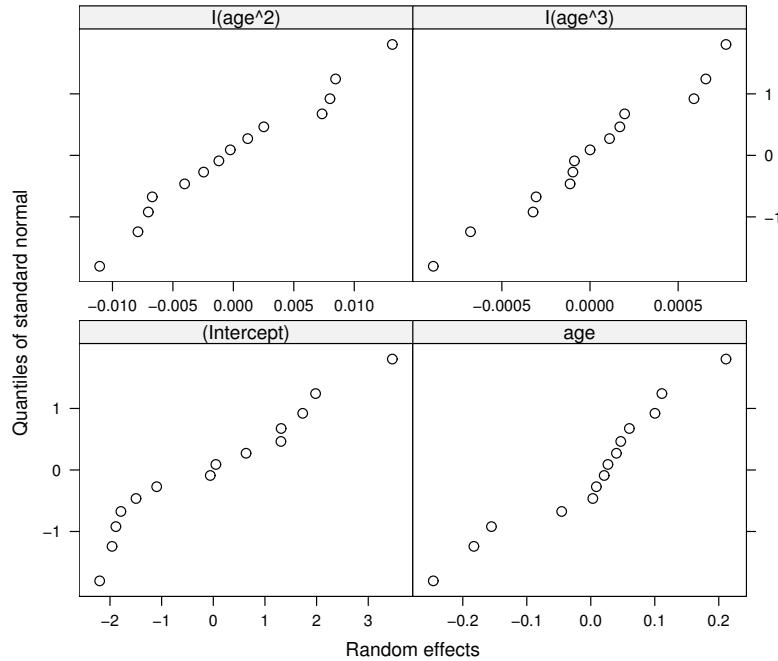


Figure 6.7 Normal QQ plots for the predicted random effects from model m2. The plots should look like random scatters around straight lines, if the normality assumptions for the random effects are reasonable: only \hat{b}_1 shows any suggestion of any problem, but it is not enough to cause serious concern.

```
> m4<-lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
+           Loblolly, list(Seed=~age+I(age^2)),
+           correlation=corAR1(form=~age|Seed), control=lmc)
>
> anova(m4,m2)
   Model df      AIC      BIC  logLik  Test L.Ratio p-value
m4     1 14  253.76  286.75 -112.88
m2     2 18  239.36  281.78 -101.68 1 vs 2 22.4004  2e-04
```

Recall that the GLRT test is somewhat problematic here, since m4 is m2 with some variance parameters set to the edge of the feasible parameter space: however a likelihood ratio statistic so large that it would have given rise to a p-value of .0002, for a standard GLRT, is strong grounds for rejecting m4 in favour of m2 in the current case. Comparison of AIC scores (which could also have been obtained using `AIC(m4, m2)`) suggests quite emphatically that m2 is the better model.

Another obvious model to try is one with a less general random effects structure. The models so far have allowed the random effects for any tree to be correlated in a very general way: it has simply been assumed that $b_j \sim N(0, \psi)$, where the only re-

restriction on the matrix ψ , is that it should be positive definite. Perhaps a less flexible model would suffice: for example, ψ might be a diagonal matrix (with positive diagonal elements). Such a structure (and indeed many other structures) can be specified in the call to `lme`.

```
> m5<-lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
+           Loblolly, list(Seed=pdDiag(~age+I(age^2)+I(age^3))),
+           correlation=corAR1(form=~age|Seed), control=lmc)
```

Here the `pdDiag` function indicates that the covariance matrix for the random effects at each level of `Seed` should have a (positive definite) diagonal structure. `m5` can be compared to `m2`.

```
> anova(m2,m5)
   Model df      AIC      BIC logLik  Test L.Ratio p-value
m2     1 18 239.3576 281.7783 -101.6788
m5     2 12 293.7081 321.9886 -134.8540 1 vs 2 66.3505 <.0001
```

Again, both the GLRT test and AIC comparison favour the more general model `m2`. In this case the GLRT assumptions are met: `m5` amounts to setting the random effects covariances in `m2` to zero, but since covariances can be positive or negative this is not on the boundary of the parameter space and the GLRT assumptions hold. The `nlme` package includes very many useful utilities for examining and plotting grouped data, one of which is the following, for plotting data and model predictions together on a unit by unit basis. See figure 6.8.

```
plot(augPred(m2))
```

6.3.2 Several levels of nesting

It is quite common, when using mixed models, to have several levels of nesting present in a model. For example, in the machine type and worker productivity model (6.6), of section 6.1.4, there are random effects for worker and each worker-machine combination. `lme` can accommodate such structures as follows

```
> lme(score~Machine,Machines, list(Worker=~1,Machine=~1))
Linear mixed-effects model fit by REML
  Data: Machines
  Log-restricted-likelihood: -107.8438
  Fixed: score ~ Machine
  (Intercept)    MachineB    MachineC
  52.355556    7.966667   13.916667

  Random effects:
  Formula: ~1 | Worker
              (Intercept)
  StdDev:     4.781049

  Formula: ~1 | Machine %in% Worker
```

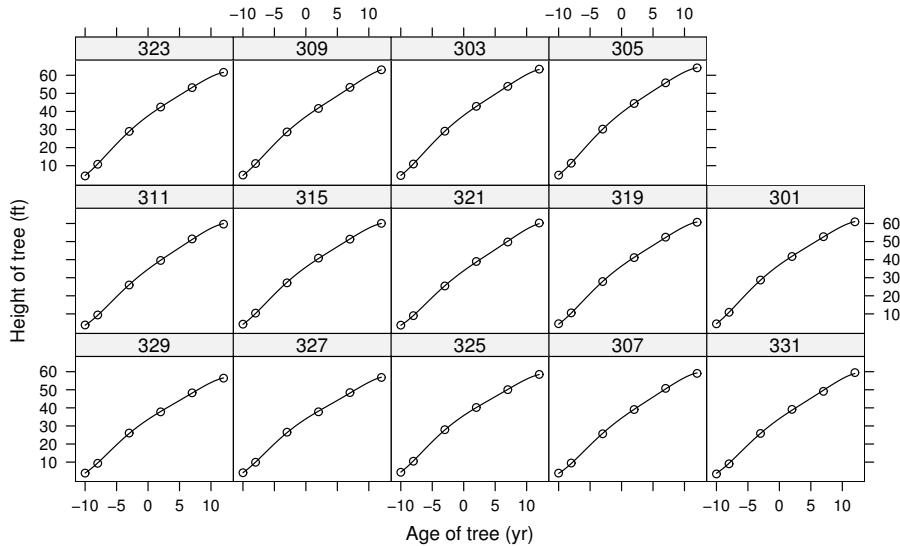


Figure 6.8 *Model predictions from m4 at the individual tree level, overlaid on individual Loblolly pine growth data. The panel titles are the value of the Seed individual tree identifier.*

```
(Intercept) Residual
StdDev:      3.729536 0.9615768

Number of Observations: 54
Number of Groups:
Worker Machine %in% Worker
6                      18
```

Notice how any grouping factor in the random effects list is assumed to be nested within the grouping factors to its left.

This section can only hope to scratch the surface of what is possible with `lme4`: for a much fuller account, see Pinheiro and Bates (2000).

6.4 Generalized linear mixed models

Generalized linear mixed models follow from linear mixed models, as GLMs followed from linear models. Let $\mu^b \equiv \mathbb{E}(y|b)$. Then a GLMM has the form

$$g(\mu_i^b) = \mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}) \text{ and } y_i | \mathbf{b} \sim \text{exponential family distribution}$$

where g is a monotonic link function, and the covariance matrix, ψ , of the random effects, is usually parameterized in terms of a parameter vector θ . The $y_i|\mathbf{b}$ are independent.

The likelihood for a GLMM is most helpfully obtained by considering the joint distribution of the response and the random effects:

$$f_{\beta, \theta, \phi}(\mathbf{y}, \mathbf{b}) \propto |\psi|^{-1/2} \exp \left(\log f(\mathbf{y}|\mathbf{b}) - \frac{1}{2} \mathbf{b}^T \psi^{-1} \mathbf{b} \right)$$

where $f(\mathbf{y}|\mathbf{b})$ is the joint distribution of the response conditional on the random effects, which is of the form given in chapter 2 for the GLM. Now the marginal distribution of \mathbf{y} , and hence the likelihood, is obtained by integrating out the random effects. i.e.

$$L(\beta, \theta, \phi) \propto |\psi|^{-1/2} \int \exp \left(l(\beta, \mathbf{b}) - \frac{1}{2} \mathbf{b}^T \psi^{-1} \mathbf{b} \right) d\mathbf{b}$$

where $l(\beta, \mathbf{b})$ is $f(\mathbf{y}|\mathbf{b})$ with the observed \mathbf{y} plugged in, considered as a function of β and \mathbf{b} , i.e. the log likelihood of the GLM that would result from treating both β and \mathbf{b} as fixed effects (see section 2.1.2). Unfortunately, this integral generally has to be either approximated, or evaluated numerically, with the latter becoming increasingly impractical as the dimension of \mathbf{b} increases.

A simple approximation is obtained by replacing $l_p = l(\beta, \mathbf{b}) - \mathbf{b}^T \psi^{-1} \mathbf{b}/2$ by a quadratic approximation about estimated values, $\hat{\beta}$ and $\hat{\mathbf{b}}$, that maximize l_p . The resulting integral can be evaluated, and the approximation is usually reasonable, since the integrand decays rapidly to zero away from $\hat{\mathbf{b}}$: this is a Laplace approximation to the integral.

Following this prescription, $l(\beta, \mathbf{b})$ can be approximated by $\mathcal{S}_m = -\|\mathbf{W}^{1/2}(\mathbf{z} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b})\|^2/(2\phi)$, to within an additive constant, where

$$z_i = g'(\hat{\mu}_i^b)(y_i - \hat{\mu}_i^b) + \mathbf{X}_i \hat{\beta} + \mathbf{Z}_i \hat{\mathbf{b}}$$

and

$$W_{ii} = \frac{1}{V(\hat{\mu}_i^b)g'(\hat{\mu}_i^b)^2}.$$

In these expressions, $\hat{\mu}_i^b = g^{-1}(\mathbf{X}_i \hat{\beta} + \mathbf{Z}_i \hat{\mathbf{b}})$, ϕ is the ‘scale parameter’ of the exponential family distribution concerned and V is the function characterizing the mean variance relationship for the distribution.

From the results of section 2.1.3, it follows that \mathcal{S}_m has the same first derivatives w.r.t. β and \mathbf{b} as $l(\beta, \mathbf{b})$, and its Hessian matrix with respect to these parameters is the expected hessian of $l(\beta, \mathbf{b})$, so that the quadratic approximation is justified by the law of large numbers. Hence the approximate likelihood of the model is given by

$$L^*(\beta, \theta, \phi) \propto |\psi|^{-1/2} \int \exp \left(\frac{-1}{2\phi} \|\mathbf{W}^{1/2}(\mathbf{z} - \mathbf{X}\beta - \mathbf{Z}\mathbf{b})\|^2 - \frac{1}{2} \mathbf{b}^T \psi^{-1} \mathbf{b} \right) d\mathbf{b},$$

which is simply the likelihood of a weighted linear mixed model.

Hence L may be maximized approximately by iteratively maximizing approximate likelihoods, L^* , using the methods already discussed for linear mixed models. Notice that the results are only approximately maximum likelihood estimates now, even as the sample size tends to infinity. The approximation depends on how good the Laplace approximation to the integral is, and also on the W_{ii} 's varying only slowly with β and b .

So the recipe for fitting a GLMM is:

1. Obtain initial estimates $\hat{\beta}^{[1]}$ and $\hat{b}^{[1]}$, for example by setting $\hat{b}^{[1]} = \mathbf{0}$ and fitting the resulting GLM to get $\hat{\beta}^{[1]}$.
2. Set $k = 1$ and iterate the following steps until convergence.
3. Given $\hat{\beta}^{[k]}$ and $\hat{b}^{[k]}$, find \mathbf{z} and \mathbf{W} as defined above.
4. Estimate the linear mixed effects model

$$\mathbf{z} = \mathbf{X}\beta + \mathbf{Z}b + \epsilon, \quad \mathbf{b} \sim N(\mathbf{0}, \psi), \quad \epsilon \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi)$$

to obtain estimates $\hat{\beta}^{[k+1]}$, $\hat{\theta}^{[k+1]}$ and $\hat{\phi}^{[k+1]}$, and predictions $\hat{b}^{[k+1]}$. Increment k by one.

This approximate method is usually known as Penalized Quasi Likelihood (PQL), as a result of the way in which Breslow and Clayton (1993) sought to justify it theoretically.

While approximate parameter estimator distributions for GLMMs are easily obtained, it is not clear whether general likelihood based inferential procedures can be made to work with the approximate likelihood used in estimation, or some variant of it. The development of theory for AIC or GLRT like statistics for these models is still an open research topic, but it is tempting to use the AIC of the working linear mixed model at convergence as a rough guide for model selection.

6.5 GLMMs with R

GLMMs as described in section 6.4 are implemented by routine `glmmPQL` supplied with Venables and Ripley's MASS library. `glmmPQL` calls are much like `lme` calls except that it is now necessary to supply a `family`. Also `glmmPQL` will accept offsets, unlike `lme` (at time of writing). As you would expect from section 6.4, `glmmPQL` operates by iteratively calling `lme`, and it returns the fitted `lme` model object for the working model at convergence.

To illustrate its use, consider again the Sole egg modelling undertaken in section 2.3.4. One issue with the data used in that exercise, is that, at any sampling station, the counts for the four different egg stages are all taken from the same net sample. It is therefore very likely that there is a 'sampling station' component to the variance of the data, which effectively means that the data for different stages at a station can not be treated as independent. This effect can easily be checked by examining the residuals from the final model in section 2.3.4 for evidence of a 'station effect'.

```

> rf <- residuals(b4,type="d") # extract deviance residuals
> ## create an identifier for each sampling station
> solr$station <- factor(with(solr,paste(-la,-lo,-t,sep="")))
>
> ## is there evidence of a station effect in the residuals?
> solr$rf <-rf
> rm <- lme(rf~1,solr,random=~1|station)
> rm0 <- lm(rf~1,solr)
> anova(rm,rm0)
  Model df      AIC      BIC logLik  Test L.Ratio p-value
rm     1  3 3319.57 3335.66 -1656.79
rm0    2  2 3582.79 3593.52 -1789.40 1 vs 2 265.220 <.0001

```

The above output compares two models for the residuals. `rm0` models them as i.i.d. normal random variables, with some unknown mean, while `rm` models the residuals as having a mean depending on a station dependent random effect. The GLRT test, performed by the `anova` function, clearly rejects the i.i.d. model, suggesting that there is a real sampling station effect.

One way of modelling the sampling station effect would be to suppose that the mean of each stage count, at each station, is multiplied by a log-normal random variable, $\exp(b_i)$, where $b_i \sim N(0, \sigma_b^2)$ is a station specific random effect. The resulting GLMM can be estimated as follows.

```

> b <- glmmPQL(eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2) +
+ I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+
+ I(t^3)+I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+I(lo^2*t)+
+ I(la^2*t)+I(la*t^2)+I(lo*t^2) # end log spawn
+ + a +I(a*t)+I(t^2*a),random=list(station=~1),
family=quasi(link=log,variance="mu"),data=solr)
> summary(b)
[edited]

```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.025506	0.1813214	1178	0.140665	0.8882
lo	4.643018	0.5179583	374	8.964077	0.0000
la	-4.878785	0.6313637	374	-7.727375	0.0000
t	-2.101037	0.3091183	374	-6.796872	0.0000
I(lo * la)	4.221226	0.8216752	374	5.137342	0.0000
I(lo^2)	-4.895147	0.4573844	374	-10.702480	0.0000
I(la^2)	-5.187457	0.7565845	374	-6.856415	0.0000
I(t^2)	-1.469416	0.1961255	374	-7.492220	0.0000
I(lo * t)	-0.246946	0.3646591	374	-0.677197	0.4987
I(la * t)	1.578309	0.4576964	374	3.448376	0.0006
I(lo^3)	-3.956541	0.6598010	374	-5.996567	0.0000
I(la^3)	5.524490	1.5175128	374	3.640490	0.0003
I(t^3)	0.633109	0.1359888	374	4.655593	0.0000
I(lo * la * t)	-0.040474	0.8643458	374	-0.046826	0.9627
I(lo^2 * la)	6.700204	1.0944157	374	6.122175	0.0000
I(lo * la^2)	-11.539919	1.5509149	374	-7.440717	0.0000

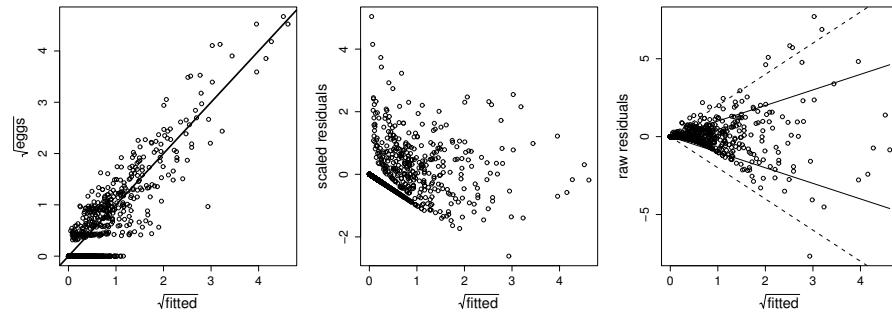


Figure 6.9 *Model checking plots for the GLMM of the Bristol Channel Sole data. The left panel shows the relationship between raw data and fitted values. The middle panel plots Pearson residuals against fitted values: there are a handful of rather high residuals at low predicted densities. The right panel shows raw residuals against fitted values, with reference lines illustrating where 1 residual standard deviation and 2 residual standard deviations from the residual mean should lie, for each fitted value.*

```

I(lo^2 * t)      0.517189 0.6008440 374   0.860770 0.3899
I(la^2 * t)      4.879013 1.0328137 374   4.724001 0.0000
I(la * t^2)      -0.548011 0.4099971 374  -1.336623 0.1822
I(lo * t^2)      0.822542 0.3576837 374   2.299635 0.0220
a                 -0.121312 0.0125463 1178  -9.669168 0.0000
I(a * t)          0.092769 0.0270447 1178   3.430218 0.0006
I(t^2 * a)        -0.187472 0.0362511 1178  -5.171498 0.0000
[edited]

```

The summary[‡] suggests dropping $I(lo * la * t)$. Refitting without this term, and then examining the significance of terms in the resulting model, suggests dropping $I(lo*t)$. Continuing in the same way we drop $I(lo^2*t)$ and $I(la*t^2)$, before all terms register as significant at the 5% level. Note that GLR testing is not possible with models estimated in this way — we do not actually know the value of the maximized likelihood for the model. Supposing that the final fitted model is again called `b4`, some residual plots can now be produced.

```

fv <- exp(fitted(b4)+solr$off) # note need to add offset
resid <- solr$egg-fv           # raw residuals
plot(fv^.5,solr$eggs^.5)
abline(0,1,lwd=2)
plot(fv^.5,resid/fv^.5)
plot(fv^.5,resid)
f1<-sort(fv^.5)
## add 1 s.d. and 2 s.d. reference lines

```

[‡] `summary(b)` could have been replaced by `anova(b, type="marginal")`, the latter being the more useful function for models with factors.

```
lines(f1,f1);lines(f1,-f1);lines(f1,2*f1,lty=2)
lines(f1,-2*f1,lty=2)
```

The resulting plots are shown in figure 6.9. Comparison of these plots with the equivalent plots in section 2.3.4 highlights how substantial the station effect appears to be, and this is emphasized by examining the estimated size effect.

```
> intervals(b4,which="var-cov")
Approximate 95% confidence intervals

Random Effects:
  Level: station
            lower      est.      upper
sd((Intercept)) 0.8398715 0.9599066 1.097097

  Within-group standard error:
            lower      est.      upper
0.5565463 0.5777919 0.5998485
```

Clearly the station effect is somewhat larger than the variability in the working residuals.

Figure 6.10 shows the predicted spawning rates over the Bristol Channel, at various times of year. Note that, relative to the predictions made using GLMs, in section 2.3.4, the peak spawning densities are slightly lower for this model, however, given the clear evidence for a station effect, the current model is probably better supported than the GLM.

6.6 Generalized Additive Mixed Models

A GAMM is just a GLMM in which part of the linear predictor is specified in terms of smooth functions of covariates (see e.g. Lin and Zhang, 1999). For example, an Additive Mixed Model has a structure something like

$$y_i = \mathbf{X}_i\boldsymbol{\beta} + f_1(x_{1i}) + f_2(x_{2i}, x_{3i}) + \dots + \mathbf{Z}_i\mathbf{b} + \epsilon_i \quad (6.19)$$

where y_i is a univariate response; $\boldsymbol{\theta}$ is a vector of fixed parameters; \mathbf{X}_i is a row of a fixed effects model matrix; the f_j s are smooth functions of covariates x_k ; \mathbf{Z}_i is a row of a random effects model matrix; $\mathbf{b} \sim N(0, \psi)$ is a vector of random effects coefficients, with unknown positive definite covariance matrix ψ ; $\epsilon \sim N(\mathbf{0}, \Lambda)$ is a residual error vector, with i^{th} element ϵ_i , and covariance matrix Λ , which is usually assumed to have some simple pattern.

The generalization from GLMs to GAMs required the development of theory for penalized regression, in order to avoid overfitting, but GLMM methods require no adjustment in order to cope with GAMMs: it is possible to write any of the penalized regression smoothers considered in this book, as components of a mixed model, while treating their smoothing parameters as variance component parameters, to be estimated by Likelihood, REML or PQL methods.

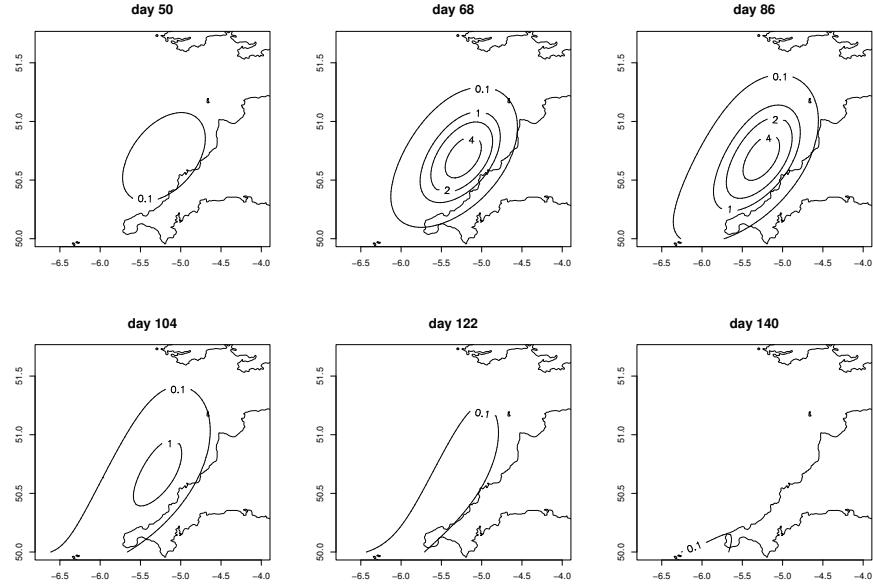


Figure 6.10 *Predicted spawning distributions of Bristol Channel Sole according to the GLMM of section 6.5. Notice how the spawning distributions are less peaked than those shown in figure 2.16.*

For example, (6.19) can be turned into a regular linear mixed model, by making use of the Bayesian model of smoothing covered in section 4.8.1, and in particular the material in section 4.8.2. Each smooth is treated as having a fixed effects (unpenalized) component, which can be absorbed into $\mathbf{X}_i\beta$, and a random effects effects (penalized) component, which can be absorbed into $\mathbf{Z}_i\mathbf{b}$. The random effects component of the smooth also has an associated Gaussian distributional assumption, based on the wigginess measure for the smooth, and having an unknown variance parameter, which is related to the smoothing parameter, as in section 4.8.2.

6.6.1 Smooths as mixed model components

This section explains, in more detail, how any quadratically penalized smooth can be used as a conventional component of a linear mixed model. The material here is closely related to that in section 4.8.2. Smooths with a single smoothing parameter are considered first, followed by the tensor product smooths of section 4.1.8.

First consider a smooth with a single smoothing parameter. For example,

$$f(\mathbf{x}) = \sum_{j=1}^J b_j(\mathbf{x})\beta_j$$

with associated wiggliness measure, $J(f) = \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta}$, where \mathbf{S} is a positive semi-definite matrix of coefficients (only semi-definite because most penalties treat some space of functions as having zero wigginess). Given (y_i, \mathbf{x}_i) data, it is straightforward to produce a model matrix \mathbf{X}^f , where $X_{ij}^f = b_j(\mathbf{x}_i)$, so that $\mathbf{X}^f \boldsymbol{\beta}$ is a vector of $f(\mathbf{x}_i)$ values.

Following section 4.8.1, the mixed model approach to estimating f , starts from the premise that, by stating that f is smooth, we really believe that it is more probable that f is smooth than that f is wiggly. This can be formalized by specifying a prior for the wigginess of the model which is $\propto \exp(-\lambda \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta}/2)$, say. Such a prior implies an improper Gaussian prior for $\boldsymbol{\beta}$ itself, but an improper distribution for $\boldsymbol{\beta}$ does not fit easily into standard linear mixed modelling approaches (e.g. Pinheiro and Bates, 2000). Some re-parameterization is therefore needed, so that the new parameters divide into a set with a proper distribution, to be treated as random effects, and a set (of size M) with an improper uniform distribution, which can be treated as fixed effects. In general, this can be achieved by using the eigen-decomposition, $\mathbf{S} = \mathbf{U} \mathbf{D} \mathbf{U}^\top$, where \mathbf{U} is an orthogonal matrix, the columns of which are the eigenvectors of \mathbf{S} , and \mathbf{D} is a diagonal matrix, with the corresponding eigenvalues arranged in descending order on the leading diagonal. Let \mathbf{D}_+ denote the smallest sub-matrix of \mathbf{D} containing all the strictly positive eigenvalues. Now re-parameterize, so that the new coefficient vector can be written $(\mathbf{b}_R^\top, \boldsymbol{\beta}_F^\top)^\top \equiv \mathbf{U}^\top \boldsymbol{\beta}$, where $\boldsymbol{\beta}_F$ is of dimension M . It is clear that $\boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta} = \mathbf{b}_R^\top \mathbf{D}_+ \mathbf{b}_R$, and that the coefficients $\boldsymbol{\beta}_F$ are unpenalized. Partitioning the eigenvector matrix so that $\mathbf{U} \equiv [\mathbf{U}_R : \mathbf{U}_F]$, where \mathbf{U}_F has M columns, and defining $\mathbf{X}_F \equiv \mathbf{X}^f \mathbf{U}_F$, while $\mathbf{X}_R = \mathbf{X}^f \mathbf{U}_R$, the mixed model representation of the smooth, in terms of a linear predictor and random effects distribution is now

$$\mathbf{X}_F \boldsymbol{\beta}_F + \mathbf{X}_R \mathbf{b}_R \text{ where } \mathbf{b}_R \sim N(\mathbf{0}, \mathbf{D}_+^{-1}/\lambda)$$

where λ and $\boldsymbol{\beta}_F$ are fixed parameters to be estimated. For convenient estimation with standard software, a further re-parameterization is useful. Defining $\mathbf{b} = \sqrt{\mathbf{D}_+^{-1}} \mathbf{b}_R$ and $\mathbf{Z} = \mathbf{X}_R \sqrt{\mathbf{D}_+}$, then the mixed model representation of the term, evaluated at its covariate values is

$$\mathbf{X}_F \boldsymbol{\beta}_F + \mathbf{Z} \mathbf{b} \text{ where } \mathbf{b} \sim N(\mathbf{0}, \mathbf{I}/\lambda)$$

Including such a term in a standard GLMM is simply a matter of appending the columns of \mathbf{X}_F to the fixed effect model matrix, appending the columns of \mathbf{Z} to the random effects model matrix, and specifying the given random effects covariance matrix. Obviously, the multiple smooth terms of an additive model are easily combined (although centering constraints are then required, which can most conveniently be absorbed into the basis before any of this section's re-parameterizations as in section 4.2).

When representing tensor product smooths (see section 4.1.8), which have multiple smoothing parameters, the only change is that the positive semi-definite pseudoinverse of the covariance matrix for β is now of the form $\sum_{i=1}^d \lambda_i \tilde{\mathbf{S}}_i$, where $\tilde{\mathbf{S}}_i$ is defined in section 4.1.8. The degree of rank deficiency of this matrix, M_T , is readily shown to be given by the product of the dimensions of the null spaces of the marginal penalty matrices, \mathbf{S}_i , (provided that $\lambda_i > 0 \forall i$). Again re-parameterization is needed, this time by forming,

$$\sum_{i=1}^d \tilde{\mathbf{S}}_i = \mathbf{U} \mathbf{D} \mathbf{U}^\top$$

where \mathbf{U} is an orthogonal matrix of eigenvectors, and \mathbf{D} is a diagonal matrix of eigenvalues, with M_T zero elements at the end of the leading diagonal. Notice that there are no λ_i parameters in the sum that is decomposed: this is reasonable since the null space of the penalty does not depend on these parameters (however given finite precision arithmetic it might be necessary to scale the $\tilde{\mathbf{S}}_i$ matrices in some cases).

It is not now possible to achieve the sort of simple representation of a term that was obtained with a single penalty, so the re-parameterization is actually simpler. Partitioning the eigenvector matrix so that $\mathbf{U} \equiv [\mathbf{U}_R : \mathbf{U}_F]$ where \mathbf{U}_F has M_T columns, it is necessary to define $\mathbf{X}_F \equiv \mathbf{X}^f \mathbf{U}_F$, $\mathbf{Z} \equiv \mathbf{X}^f \mathbf{U}_R$ and $\mathcal{S}_i = \mathbf{U}_R^\top \tilde{\mathbf{S}}_i \mathbf{U}_R$. A mixed model representation of the tensor product term (i.e. the linear predictor and random effects distribution) is now

$$\mathbf{X}_F \beta_F + \mathbf{Z} \mathbf{b} \text{ where } \mathbf{b} \sim N\left(\mathbf{0}, \left(\sum \lambda_i \mathcal{S}_i\right)^{-1}\right),$$

where the λ_i and β_F parameters have to be estimated. This covariance matrix structure is not a form that is available in standard software, but it turns out to be possible to implement, at least in the `nlme` software of Pinheiro and Bates (2000): `R` package `mgcv` provides an appropriate ‘pdMat’ class for `lme` called ‘pdTens’. Given such a class, incorporation of one or more tensor product terms into a (generalized) linear mixed model is straightforward.

6.6.2 Inference with GAMMs

When using GAMMs, it is often desirable to be able to calculate confidence/credible intervals, exactly as for a GAM. In particular, credible regions for the smooth components are required. To do this, let β now contain all the fixed effects *and* the random effects for the smooth terms (only), and let $\bar{\mathbf{X}}$ be the corresponding model matrix. Let $\bar{\mathbf{Z}}$ be the random effects model matrix *excluding* the columns relating to smooths, and let $\bar{\psi}$ be the corresponding random effects covariance matrix. Now define a covariance matrix

$$\mathbf{V} = \bar{\mathbf{Z}} \bar{\psi} \bar{\mathbf{Z}}^\top + \Lambda \sigma^2.$$

Essentially the Bayesian approach of section 4.8.1 implies that

$$\beta \sim N(\hat{\beta}, (\bar{\mathbf{X}}^\top \mathbf{V}^{-1} \bar{\mathbf{X}} + \mathbf{S})^{-1})$$

where $\mathbf{S} = \sum \lambda_i / \sigma^2 \mathbf{S}_i$. Similarly the leading diagonal of

$$\mathbf{F} = (\bar{\mathbf{X}}^\top \mathbf{V}^{-1} \bar{\mathbf{X}} + \mathbf{S})^{-1} (\bar{\mathbf{X}}^\top \mathbf{V}^{-1} \bar{\mathbf{X}})$$

gives the effective degrees of freedom for each element of β .

In the AMM case, the usual inferential framework for linear mixed models applies exactly, and can be used for model comparison: for example AIC based model selection is straightforward. In the GAMM case, model comparison is not so straightforward, since these issues are still somewhat open for GLMMs.

6.7 GAMMs with R

R library `mgcv` includes a `gamm` function which fits GAMMs based on linear mixed models, as implemented in the `nlme` library. The function is basically a wrapper function for `lme`, or the GLMM fitting routine `glmmPQL`, from the `MASS` library: its purpose is to perform the re-parameterizations detailed in section 6.6.1, call `lme`, or `glmmPQL`, to actually estimate the model, and then unscramble the returned object so that (i) it looks like a `gam` object and (ii) the posterior covariance matrix of section 6.6.2 is readily calculated. This section presents some examples of its use.

It should be noted that `gamm` seems to work `lme` quite hard, and it is not difficult to specify models which cause numerical problems in estimation, or failure of the PQL iterations in the generalized case. This seems to be particularly true when explicitly modelling correlation in the data, probably because of the inherent difficulty in separating correlation from trend, when the trend model is itself rather complex. Note also that changes in the underlying optimization methods may lead to slight differences between the results obtained with different `mgcv` and `nlme` versions: these should not be statistically important.

6.7.1 A GAMM for sole eggs

To start with, let us revisit the Bristol Channel Sole data, one more time. The GLMs and GLMMs considered in sections 2.3.4 and 6.5 were rather unwieldy, as a result of the large number of polynomial terms involved in specifying the models. If nothing else, a GAMM offers a way of reducing the clumsiness of the model formulation. It is straightforward to write the basic sole model (2.13), plus random effect, as a GAMM,

$$\log(\mu_i) = \log(\Delta_i) + f_1(\text{lo}_i, \text{la}_i, \text{t}_i) - f_2(\text{t}_i) \bar{a}_i + b_k,$$

if observation i is from sampling station k . Here f_1 and f_2 are smooth functions, $b_k \sim N(0, \sigma_b^2)$ are the i.i.d. random effects for station, and, as before, μ_i is the expected value for the i^{th} observation, while Δ_i and \bar{a}_i are the width and average age of the corresponding egg class. `lo`, `la` and `t` are location and time variables. Notice that, for simplicity, the mortality rate term, f_2 is assumed to depend only on time: this assumption could be relaxed, but it is unlikely that the data really contain enough information to say much about spatial variation in mortality rate.

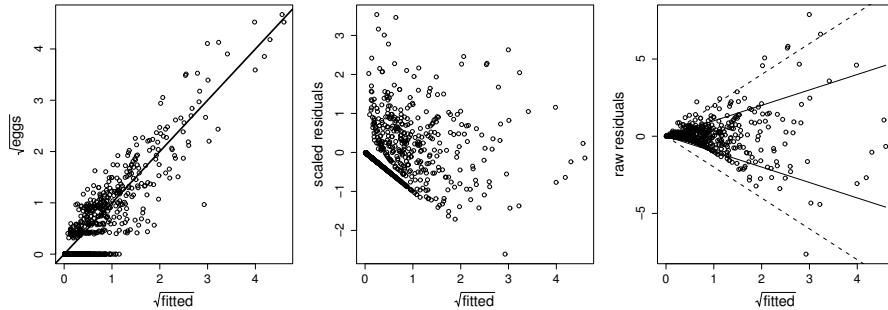


Figure 6.11 *Model checking plots for the GAMM of the Bristol Channel Sole data. The left panel shows the relationship between raw data and fitted values. The middle panel plots Pearson residuals against fitted values: there are a handful of rather high residuals at low predicted densities. The right panel shows raw residuals against fitted values, with reference lines illustrating where 1 residual standard deviation and 2 residual standard deviations from the residual mean should lie, for each fitted value. Note how plots are very similar to those from figure 6.9, although the residuals at low densities are less extreme in the current plots.*

We need to decide what sort of smooths to use to represent f_1 and f_2 . For f_1 , a tensor product of a thin plate regression spline of lo and la , with a thin plate regression spline (or any other spline) of t , is probably appropriate: isotropy is a reasonable assumption for spatial dependence (although in that case we should really use a more isotropic co-ordinate system than longitude and latitude), but not for the space-time interaction. Anything could be used for f_2 and the default TPRS suffices.

The multiplication of f_2 by \bar{a}_i is achieved using the `by` variable mechanism. We need to bear in mind that f_2 will be subject to centering constraints, when estimated as part of a GAM(M): this means that we will need to add an extra \bar{a}_i term into the right-hand side of the model, to allow for the fact that such a constraint is not required in the current case.

Here is the call used to fit the model. Note that, unlike `lme`, `gamm` will *only* accept the list form of the `random` argument.

```
bam<-gamm(eggs~te(lo,la,t,bs=c("tp","tp")),k=c(25,5),d=c(2,1))
+ a+s(t,k=5,by=a)+offset(off),family=quasi(link=log,
variance="mu"),data=solr,random=list(station=~1))
```

`gamm` returns a list with two components: `lme` is the object returned by `lme` or `glmmPQL`; `gam` is an incomplete object of class `gam`, which can be treated like a `gam` object for prediction, plotting etc. For example,

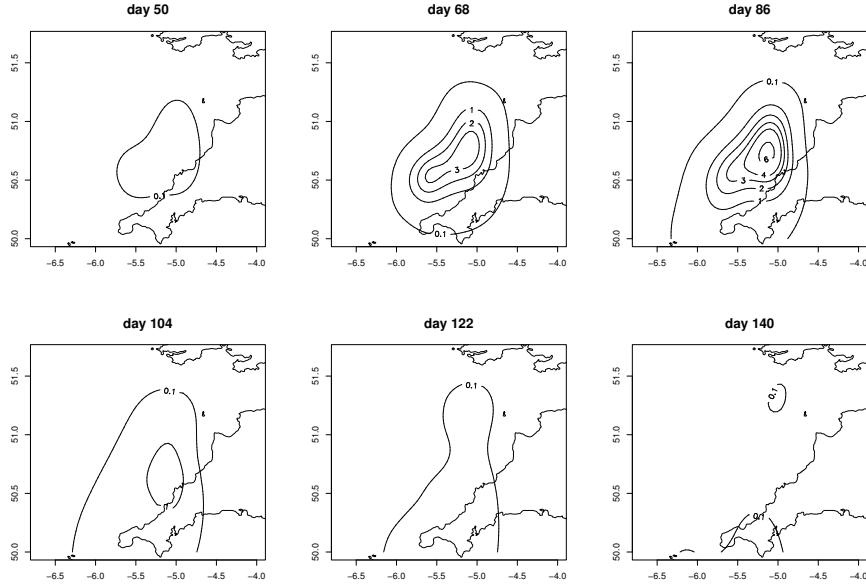


Figure 6.12 Predicted spawning rates at various times, using the GAMM of Bristol Channel sole eggs, presented in section 6.7. Note how plots are peakier and of rather different shape to those from figure 6.10.

```
> bam$gam

Family: quasi
Link function: log

Formula:
eggs ~ offset(off) + te(lo, la, t, bs = c("tp", "tp")), k = c(25,
5), d = c(2, 1)) + a + s(t, k = 5, by = a)

Estimated degrees of freedom:
49.96516 3.409959 total = 55.37512.
```

Residual plots for the model are shown in figure 6.11, these have been calculated from `bam$lme` so that the predictions of the random effects are included in fitted values and excluded from residuals, as is appropriate for model checking. The plots show a slight improvement relative to the equivalent plots from section 6.5. Figure 6.12 shows why some improvement in residual plots may be possible: the greater flexibility of the GAM allows more complicated shapes for the distributions, and allows a higher (and sharper) peak abundance.

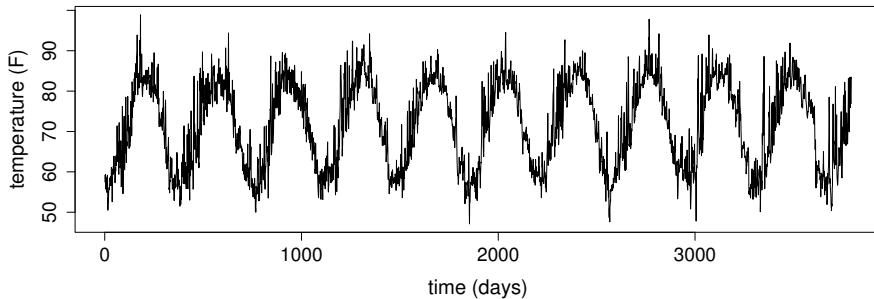


Figure 6.13 Daily air temperature in Cairo, Egypt from January 1st 1995.

6.7.2 The Temperature in Cairo

Figure 6.13 shows daily temperature in Cairo over nearly a decade. The data are from <http://www.engr.udayton.edu/weather/citylistWorld.htm>. It is clear that the data contain a good deal of noisy short term auto-correlation, and a strong yearly cycle. Much less clear, given these other sources of variation, is whether there is evidence for any increase in average mean temperature over the period: this is the sort of uncertainty that allows climate change skeptics to get away with it. A reasonable model of these data might therefore be

$$\text{temp}_i = f_1(\text{time.of.year}_i) + f_2(\text{time}_i) + e_i \quad (6.20)$$

where $e_i = \phi e_{i-1} + \epsilon_i$, the ϵ_i being i.i.d. $N(0, \sigma^2)$ random variables. f_1 should be a ‘cyclic’ function, with value and first 2 derivatives matching at the year ends. The basic idea is that if we model time of year and autocorrelation properly, then we should be in a good position to establish whether there is a significant overall temperature trend.

Model (6.20) is easily fitted.

```
ctamm<-gamm(temp~s(day.of.year,bs="cc",k=20)+s(time,bs="cr"),
  data=cairo,correlation=corAR1(form=~1|year))
```

Note a couple of details: the data covers some leap years, where `day.of.year` runs from 1 to 366. This means that by default the cyclic smooth matches at day 1 and 366, which is correct in non-leap years, but not quite right in the leap years themselves: a very fussy analysis might deal more carefully with this. Secondly, I have nested the AR model for the residuals within year: this vastly speeds up computation, but is somewhat arbitrary.

Examining the `gam` part of the fitted model first

```
> summary(ctamm$gam)
```

```

Family: gaussian
Link function: identity

Formula:
temp ~ s(day.of.year, bs = "cc", k = 20) + s(time, bs = "cr")

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 71.6463     0.1517   472.2 <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
          edf Est.rank    F p-value
s(day.of.year) 8.521    18.000 225.208 <2e-16 ***
s(time)        1.347     7.000   2.327  0.0228 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) = 0.849  Scale est. = 16.493    n = 3780

```

it seems that there is some evidence for a long term trend in temperature, and that the model fits fairly closely. We can also extract things from the `lme` representation of the fitted model, for example 95% confidence intervals for the variance parameters.

```

> intervals(ctamm$lme, which="var-cov")
Approximate 95% confidence intervals

Random Effects:
Level: g.1
      lower      est.      upper
sd(Xr.1 - 1) 3.799281 10.22356 27.41017
Level: g.2
      lower      est.      upper
sd(Xr.2 - 1) 0.0001544062 0.07988084 48.81436

Correlation structure:
      lower      est.      upper
Phi 0.6593677 0.683507 0.7062383
attr(,"label")
[1] "Correlation structure:"

Within-group standard error:
      lower      est.      upper
3.911106 4.061128 4.216904

```

The confidence interval for ϕ is easily picked out, and provides very strong evidence that the AR1 model is preferable to an independence model ($\phi = 0$), while the interval for σ is (3.92,4.23). Note that confidence intervals for the smoothing parameters

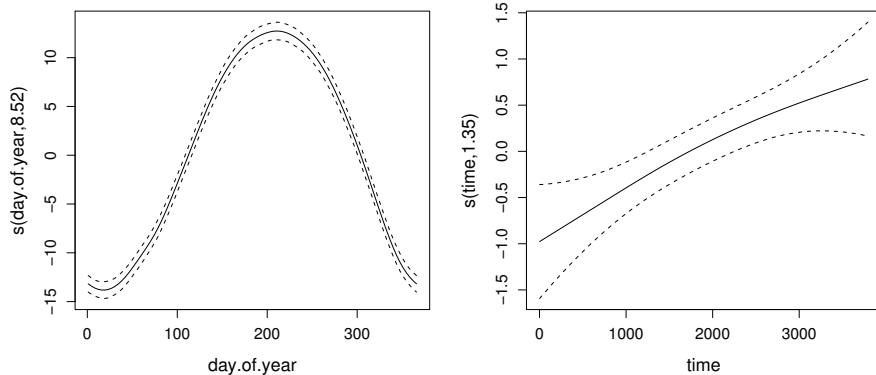


Figure 6.14 *GAMM terms for daily air temperature in Cairo, Egypt from January 1st 1995.* The left panel is the estimated annual cycle: note that it has a fatter peak and thinner trough than a sinusoid. The right pattern is the estimated long term trend: there appears to have been a rise of around 1.5 F over the period of the data.

are also available. Under the Random Effects heading the interval for $g.1$ relates to the smoothing parameter for the first smooth, while that for $g.2$ relates to the second smooth. The parameterization is not completely obvious: the reported parameters are σ^2/λ_i , as the following confirms

```
> ctamm$gam$sig2/ctamm$gam$sp
[1] 10.22356312  0.07988084
```

This ability to quantify the uncertainty associated with the smoothing parameters is a nice bonus from use of the mixed model approach.

The approximate p-values from the `summary` command suggested evidence for a long term temperature trend, but since this model is just a linear mixed model and we hence have access to its full likelihood, we can use likelihood based methods for testing, as well. For example

```
> ctamm0<-gamm(temp~s(day.of.year,bs="cc",k=20),data=cairo,
+ correlation=corAR1(form=~1|year))
> ctamm0$gam

Family: gaussian
Link function: identity

Formula:
temp ~ s(day.of.year, bs = "cc", k = 20)

Estimated degrees of freedom:
8.437449 total = 9.437449
> anova(ctamm0$lme,ctamm$lme)
```

Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
ctamm0\$lm	1	5	18990.8	19022.0	-9490.4		
ctamm\$lm	2	7	18983.5	19027.1	-9484.7	1 v 2	11.3108 0.0035

The hypothesis testing is again approximate here, but still suggests clear evidence for the trend. The AIC also clearly support the model with a long term trend.

To finish this short example, we can plot the estimated model terms

```
plot(ctamm$gam)
```

which results in figure 6.14. The temperature increase appears to be quite marked. The simulation techniques covered in Chapter 5 can be used to simulate from the posterior distribution of the modelled temperature rise, if required.

6.8 Exercises

1. A pig breeding company was interested in investigating litter to litter variability in piglet weight (after a fixed growth period). 6 sows were selected randomly from the companies breeding stock, impregnated and 5 (randomly selected) piglets from each resulting litter were then weighed at the end of a growth period. The data were entered into an R data frame, `pig`, with weights recorded in column `w` and a column, `sow`, containing a factor variable indicating which litter the piglet came from. The following R session is part of the analysis of these data using a simple mixed model for piglet weight.

```
> pig$w
[1] 9.6 10.1 11.2 11.1 10.5 9.5 9.6 9.4 9.5 9.5 11.5
[12] 10.9 10.8 10.7 11.7 10.7 11.2 11.2 10.9 10.5 12.3 12.1
[23] 11.2 12.3 11.7 11.2 10.3 9.9 11.1 10.5
> pig$sow
[1] 1 1 1 1 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6
[28] 6 6 6
Levels: 1 2 3 4 5 6
> m1<-lm(w~sow,data=pig)
> anova(m1)
Analysis of Variance Table

Response: w
          Df  Sum Sq Mean Sq F value    Pr(>F)
sow       5 15.8777  3.1755 14.897 1.086e-06 ***
Residuals 24  5.1160  0.2132

> piggy<-aggregate(data.matrix(pig),
+                      by=list(sow=pig$sow),mean)
> m0<-lm(w~1,data=piggy)
> summary(m1)$sigma^2
[1] 0.2131667
> summary(m0)$sigma^2
[1] 0.6351067
```

- (a) The full mixed model being used in the R session has a random effect for litter/sow and a fixed mean. Write down a full mathematical specification of the model.
- (b) Specify the hypothesis being tested by the `anova` function, both in terms of the parameters of the mixed model, and in words.
- (c) What conclusion would you draw from the printed ANOVA table. Again state your conclusions both in terms of the model parameters and in terms of what this tells you about pigs.
- (d) Using the given output, obtain an (unbiased) estimate of the between litter variance in weight, in the wider population of pigs.

2. Consider a model with 2 random effects of the form:

$$y_{ij} = \alpha + b_i + c_j + \epsilon_{ij}$$

where $i = 1, \dots, I$, $j = 1, \dots, J$, $b_i \sim N(0, \sigma_b^2)$, $c_j \sim N(0, \sigma_c^2)$ and $\epsilon_{ij} \sim N(0, \sigma^2)$ and all these r.v.'s are mutually independent. If the model is fitted by least squares then

$$\hat{\sigma}^2 = RSS/(IJ - I - J + 1)$$

is an unbiased estimator of σ^2 , where RSS is the residual sum of squares from the model fit.

- (a) Show that, if the above model is correct, the averages $\bar{y}_{i\cdot} = \sum_j y_{ij}/J$ are governed by the model:

$$\bar{y}_{i\cdot} = a + e_i$$

where the e_i are i.i.d. $N(0, \sigma_b^2 + \sigma^2/J)$ and a is a random intercept term. Hence suggest how to estimate σ_b^2 .

- (b) Show that the averages $\bar{y}_{\cdot j} = \sum_i y_{ij}/I$ are governed by the model:

$$\bar{y}_{\cdot j} = a' + e'_j$$

where the e'_j are i.i.d. $N(0, \sigma_c^2 + \sigma^2/I)$ and a' is a random intercept parameter. Suggest an estimator for σ_c^2 .

3. Data were collected on blood cholesterol levels and blood pressure for a group of patients regularly attending an outpatient clinic for a non heart disease related condition. Measurements were taken each time the patient attended the clinic. A possible model for the resulting data is,

$$y_{ij} = \mu + a_i + \beta x_{ij} + \epsilon_{ij}, \quad a_i \sim N(0, \sigma_a^2) \text{ and } \epsilon_{ij} \sim N(0, \sigma^2),$$

where y_{ij} is the j^{th} blood pressure measurement for the i^{th} patient and x_{ij} is the corresponding cholesterol measurement. β is a fixed parameter relating blood pressure to cholesterol concentration and a_i is a random coefficient for the i^{th} patient. Assume (somewhat improbably) that the same number of measurements are available for each patient.

- (a) Explain how you would test $H_0 : \sigma_a^2 = 0$ vs. $H_1 : \sigma_a^2 > 0$ and test $H_0 : \beta = 0$ vs $H_1 : \beta \neq 0$, using standard software for ordinary linear modelling.

- (b) Explain how β and σ_a^2 could be estimated. You should write down the models involved, but should assume that these would be fitted using standard linear modelling software.
4. Write out the following 3 models in the general form,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}) \text{ and } \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2),$$

where \mathbf{Z} is a matrix containing known coefficients which determine how the response, \mathbf{y} , depends on the random effects \mathbf{b} (i.e. it is a ‘model matrix’ for the random effects). $\boldsymbol{\psi}$ is the covariance matrix of the random effects \mathbf{b} . You should ensure that \mathbf{X} is specified so that the fixed effects are identifiable (you don’t need to do this for \mathbf{Z}) and don’t forget to specify $\boldsymbol{\psi}$.

- (a) The model from question 3, assuming 4 patients and 2 measurements per patient.
- (b) The mixed effects model from section 6.1.1, assuming only two measurements per tree.
- (c) Model (6.6) from section 6.1.4, assuming that $I = 2$, $J = 3$ and $K = 3$.
- 5.(a) Show that if \mathbf{X} and \mathbf{Z} are independent random vectors, both of the same dimension, and with covariance matrices $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_z$, then the covariance matrix of $\mathbf{X} + \mathbf{Z}$ is $\boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_z$.
- (b) Consider a study examining patients blood insulin levels 30 minutes after eating, y , in relation to sugar content, x , of the meal eaten. Suppose that each of 3 patients had their insulin levels measured for each of 3 sugar levels, and that an appropriate linear mixed model for the j^{th} measurement on the i^{th} patient is,

$$y_{ij} = \alpha + \beta x_{ij} + b_i + \epsilon_{ij}, \quad b_i \sim N(0, \sigma^2), \text{ and } \epsilon_{ij} \sim N(0, \sigma^2),$$

where all the random effects and residuals are mutually independent.

- i. Write this model out in matrix vector form.
ii. Find the covariance matrix for the response vector \mathbf{y} .
6. The R data frame Oxide from the nlme library contains data from a quality control exercise in the semiconductor industry. The object of the exercise was to investigate sources of variability in the thickness of oxide layers in silicon wafers. The dataframe contains the following columns:

`Thickness` is the thickness of the oxide layer (in nanometres, as far as I can tell).

`Source` is a two level factor indicating which of two possible suppliers the sample came from.

`Site` is a 3 level factor, indicating which of three sites on the silicon wafer the thickness was measured.

`Lot` is a factor variable with levels indicating which particular batch of Silicon wafers this measurement comes from.

`Wafer` is a factor variable with levels labelling the individual wafers examined.

The investigators are interested in finding out if there are systematic differences between the two sources, and expect that thickness may vary systematically across the three sites; they are only interested in the lots and wafers in as much as they are representative of a wider population of lots and wafers.

- (a) Identify which factors you would treat as random and which as fixed, in a linear mixed model analysis of these data.
 - (b) Write down a model that might form a suitable basis for beginning to analyze the Oxide data.
 - (c) Perform a complete analysis of the data, including model checking. Your aim should be to identify the sources of thickness variability in the data and any fixed effects causing thickness variability.
7. Starting from model (6.6) in section 6.1.4, re-analyse the Machines data using `lme`. Try to find the most appropriate model, taking care to examine appropriate model checking plots. Make sure that you test whether the interaction in (6.6) is appropriate. Similarly test whether a more complex random effects structure would be appropriate: specifically one in which the machine-worker interaction is correlated within worker. If any data appear particularly problematic in the checking plots, repeat the analysis, and see if the conclusions change.
8. This question follows on from question 7. *Follow up multiple comparisons* are a desirable part of some analyses. This question is about how to do this in practice. In the analysis of the Machines data the ANOVA table for the fixed effects indicates that there are significant differences between machine types, so an obvious follow up analysis would attempt to assess exactly where these differences lie. Obtaining Bonferroni corrected intervals for each of the 3 machine to machine differences would be one way to proceed, and this is easy to do.

First note that provided you have set up the default contrasts using

```
options(contrasts=c("contr.treatment", "contr.treatment"))
(before calling lme, of course) then lme will set your model up in such away that the coefficients associated with the Machine effect correspond to the difference between the second and first machines, and between the third and first machines. Hence the intervals function can produce two of the required comparisons automatically. However, by default the intervals function uses the 95% confidence level, which needs to be modified if you wish to Bonferroni correct for the fact that 3 comparisons are being made. If your model object is m1 then
```

```
intervals(m1, level=1-0.05/3, which="fixed")
```

will produce 2 of the required intervals. Note the Bonferroni correction ‘3’. The option `which="fixed"` indicates that only fixed effect intervals are required. The third comparison, between machines B and C can easily be obtained by changing the way that the factor variable `Machine` is treated, so that machine type B or C count as the ‘first machine’ when setting up the model. The MASS library provides a function, `relevel`, for doing this.

```
library(MASS)          # load MASS library
levels(Machines$Machine) # check the level names
## reset levels so that 'first level' is "B" ...
```

```
Machines$Machine<-relevel(Machines$Machine, "B")
```

Now refit the model and re-run the `intervals` function for the new fit. This will yield the interval for the remaining comparison (plus one of the intervals you already have, of course). What are the Bonferroni corrected 95% intervals for the 3 possible comparisons? How would you interpret them?

9. The data frame `Gun` (library `nlme`) is from a trial examining methods for firing naval guns. Two firing methods were compared, with each of a number of teams of 3 gunners; the gunners in each team were matched to have similar physique (Slight, Average or Heavy). The response variable `rounds` is rounds fired per minute, and there are 3 explanatory factor variables, `Physique` (levels Slight, Medium and Heavy); `Method` (levels M1 and M2) and `Team` with 9 levels. The main interest is in determining which method and/or physique results in the highest firing rate, and in quantifying team-to-team variability in firing rate.
 - (a) Identify which factors should be treated as random and which as fixed, in the analysis of these data.
 - (b) Write out a suitable mixed model as a starting point for the analysis of these data.
 - (c) Analyse the data using `lme` in order to answer the main questions of interest. Include any necessary follow up multiple comparisons (as in the previous question) and report your conclusions.
10. In an experiment comparing two varieties of Soybeans, plants were grown on 48 plots and measurements of leaf weight were taken at regular intervals as the plants grew. The `nlme` data frame `Soybean` contains the resulting data and has the following columns:

`Plot` is a factor variable with levels for each of the 48 plots.

`weight` is the leaf weight in grammes.

`Time` is the time in days since planting.

`Variety` is either F or P indicating the variety of Soybean.

There is one observation for each variety in each plot at each time. Interest focuses on modelling the growth of Soybeans over time and on establishing whether or not this differs between the varieties.

In this question you may have to increase `niterEM` in the `control` list for `lme` and `glmmPQL`: see `?lmeControl`.

- (a) A possible model for the weights is

$$w_{ijk} = \alpha_i + \beta_i t_k + \gamma_i t_k^2 + \delta_i t_k^3 + a_j + b_j t_k + \epsilon_{ijk}$$

where w_{ijk} is the weight measurement for the i^{th} variety in the j^{th} plot at the k^{th} time; $[a_j, b_j]^T \sim N(\mathbf{0}, \psi)$ where ψ is a covariance matrix, and $\epsilon_{ijk} \sim N(0, \sigma^2)$. The random effects are independent of the residuals and independent of random effects with different j . The residuals are i.i.d.

Fit this model using `lme` and confirm that the residual variance appears to increase with the (random effect conditional) mean.

- (b) To deal with the mean variance relationship, it might be appropriate to model the weights as being Gamma distributed, so that the model becomes a GLMM. e.g.

$$\log(\mu_{ijk}) = \alpha_i + \beta_i t_k + \gamma_i t_k^2 + \delta_i t_k^3 + a_j + b_j t_k$$

where $\mu_{ijk} \equiv \mathbb{E}(w_{ijk})$ and $w_{ijk} \sim \text{Gamma}$. Fit this GLMM, using `glmmPQL` from the `MASS` package, and confirm the improvement in residual plots that results.

- (c) Explore the whether further improvements to the model could be made by modifications of the random or fixed effects model structures.

11. This question follows on from question 10, on Soybean modelling.

- Using `gamm`, replace the cubic function of time in the GLMM of question 10, with a smooth function of time. You should allow for the possibility that the varieties depend differently on time, and examine appropriate model checking plots.
- Evaluate whether a model with or without a variety effect is more appropriate, and what kind of variety effect is most appropriate.
- Explain why a model with separate smooths for the two different varieties is different to a model with a smooth for one variety and a smooth correction for the other variety.



APPENDIX A

Some Matrix Algebra

This appendix provides some useful matrix results, and a brief introduction to some relevant numerical linear algebra.

A.1 Basic computational efficiency

Consider the numerical evaluation of the expression

$$\mathbf{ABy}$$

where \mathbf{A} and \mathbf{B} are $n \times n$ matrices and \mathbf{y} is an n -vector. The following code evaluates this expression in two ways (wrong and right).

```
> n <- 1000
> A <- matrix(runif(n*n), n, n)
> B <- matrix(runif(n*n), n, n)
> y <- runif(n)
> system.time((A%*%B) %*% y)      # wrong way
[1] 5.60 0.01 5.61   NA   NA
> system.time(A%*%(B%*%y))       # right way
[1] 0.02 0.00 0.02   NA   NA
```

So, in this case, the ‘wrong way’ took 5.6 seconds and the ‘right way’ less than 0.02 seconds. Why? The wrong way first multiplied \mathbf{A} and \mathbf{B} at a cost of n^3 floating point operations, and then multiplied \mathbf{AB} by \mathbf{y} at the cost of n^2 further operations: total cost $n^3 + n^2$ operations. The right way first formed the vector \mathbf{By} at a cost of n^2 operations and then formed \mathbf{ABy} at a further cost of n^2 operations: total cost $2n^2$ operations. So we should have got around a 500 fold speed up by doing things the right way.

This sort of effect is pervasive: the order in which matrix operations are performed can have a major impact on computational speed. An important example is in the evaluation of the trace of a matrix (the sum of the leading diagonal). It is easy to show that $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, and for non-square matrices, one of these is always cheaper to evaluate than the other. Here is an example

```
> m <- 500; n<-5000
```

```
> A <- matrix(runif(m*n), n, m)
> B <- matrix(runif(m*n), m, n)
> system.time(sum(diag(A%*%B)))
[1] 51.61 0.16 51.93 NA NA
> system.time(sum(diag(B%*%A)))
[1] 4.92 0.00 4.92 NA NA
```

In this case formation of \mathbf{AB} costs n^2m operations, while formation of \mathbf{BA} costs only m^2n : i.e. in the example, the second evaluation is 10 times faster than the first. Actually even the second evaluation is wasteful, as the following code is easily demonstrated to evaluate the trace, at a cost of only mn operations

```
> system.time(sum(t(B) * A))
[1] 0.11 0.00 0.11 NA NA
```

(somewhat similar code to this gets all the elements on the leading diagonal). Simple tricks like these can lead to very large efficiency gains in matrix computations.

A.2 Covariance matrices

If \mathbf{X} is a random vector and $\boldsymbol{\mu}_x = \mathbb{E}(\mathbf{X})$, then the *covariance matrix* of \mathbf{X} is

$$\mathbf{V}_x = \mathbb{E}\{(\mathbf{X} - \boldsymbol{\mu}_x)(\mathbf{X} - \boldsymbol{\mu}_x)^T\}.$$

\mathbf{V}_x is symmetric and positive semi definite (all its eigenvalues are ≥ 0). The i^{th} element on the leading diagonal of \mathbf{V}_x is the variance of X_i , while the $(i, j)^{\text{th}}$ element is the covariance of X_i and X_j .

If \mathbf{C} is a matrix of fixed real coefficients and $\mathbf{Y} = \mathbf{CX}$ then the covariance matrix of \mathbf{Y} is

$$\mathbf{V}_y = \mathbf{CV}_x\mathbf{C}^T.$$

This result is sometimes known as the ‘law of propagation of errors’. It is easily proven. First note that $\boldsymbol{\mu}_y \equiv \mathbb{E}(\mathbf{Y}) = \mathbf{C}\mathbb{E}(\mathbf{X}) = \mathbf{C}\boldsymbol{\mu}_x$. Then

$$\begin{aligned} \mathbf{V}_y &= \mathbb{E}\{(\mathbf{Y} - \boldsymbol{\mu}_y)(\mathbf{Y} - \boldsymbol{\mu}_y)^T\} \\ &= \mathbb{E}\{(\mathbf{CX} - \mathbf{C}\boldsymbol{\mu}_x)(\mathbf{CX} - \mathbf{C}\boldsymbol{\mu}_x)^T\} \\ &= \mathbf{C}\mathbb{E}\{(\mathbf{X} - \boldsymbol{\mu}_x)(\mathbf{X} - \boldsymbol{\mu}_x)^T\}\mathbf{C}^T \\ &= \mathbf{CV}_x\mathbf{C}^T. \end{aligned}$$

A.3 Differentiating a matrix inverse

Consider differentiation of \mathbf{A}^{-1} w.r.t. x . By definition of a matrix inverse we have that $\mathbf{I} = \mathbf{A}^{-1}\mathbf{A}$. Differentiating this expression w.r.t. x gives

$$0 = \frac{\partial \mathbf{A}^{-1}}{\partial x} \mathbf{A} + \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x}$$

which re-arranges to

$$\frac{\partial \mathbf{A}^{-1}}{\partial x} = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x} \mathbf{A}^{-1}.$$

A.4 Kronecker product

The Kronecker product of a $n \times m$ matrix \mathbf{A} and $p \times q$ matrix \mathbf{B} is the $np \times qm$ matrix

$$\begin{bmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \dots & \dots \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

In R this is implemented by the operator `%x%`: see `?kronecker` for details.

A.5 Orthogonal matrices and Householder matrices

Orthogonal matrices are square matrices which rotate/reflect vectors and have a variety of interesting and useful properties. If \mathbf{Q} is an orthogonal matrix then $\mathbf{Q}\mathbf{Q}^T = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}$, i.e. $\mathbf{Q}^{-1} = \mathbf{Q}^T$. If \mathbf{x} is any vector of appropriate dimension then $\|\mathbf{Q}\mathbf{x}\| = \|\mathbf{x}\|^*$, that is \mathbf{Q} changes the elements of \mathbf{x} without changing its length: i.e. it rotates/reflects \mathbf{x} . It follows immediately that all the eigenvalues of \mathbf{Q} are one.

A particularly important class of orthogonal matrices are the Householder/reflector matrices. Let \mathbf{u} be a non-zero vector. Then

$$\mathbf{H} = \mathbf{I} - \gamma \mathbf{u} \mathbf{u}^T \text{ where } \gamma = 2/\|\mathbf{u}\|^2$$

is a *Householder matrix* or *reflector matrix*. Note that \mathbf{H} is symmetric.

The multiplication of a vector by a householder matrix is a prime example of the need to think carefully about the order of operations when working with matrices. \mathbf{H} is never stored as a complete matrix, but rather \mathbf{u} and γ are stored. Then $\mathbf{H}\mathbf{x}$ is evaluated as follows (overwriting \mathbf{x} by $\mathbf{H}\mathbf{x}$).

1. $\alpha \leftarrow \mathbf{u}^T \mathbf{x}$
2. $\mathbf{x} \leftarrow \mathbf{x} - \alpha \gamma \mathbf{u}$

Notice how this requires only $O(n)$ operations, where $n = \dim(\mathbf{x})$ — explicit formation and use of \mathbf{H} is $O(n^2)$.

Householder matrices are important, because if \mathbf{x} and \mathbf{y} are two non-zero vectors of the same length (i.e. $\|\mathbf{x}\| = \|\mathbf{y}\|$), then the Householder matrix such that $\mathbf{H}\mathbf{x} = \mathbf{y}$ is obtained by setting $\mathbf{u} = \mathbf{x} - \mathbf{y}$. This property is exploited in the next section, but note that in practical computation it is important to guard against possible overflow or cancellation error when using these matrices: see e.g. Watkins (1991, section 3.2) for further details.

* recall than $\|\mathbf{x}\|^2 = \mathbf{x}^T \mathbf{x}$.

A.6 QR decomposition

Any real $n \times m$ ($m \leq n$, here) matrix \mathbf{X} can be written as

$$\mathbf{X} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

where \mathbf{Q} is an orthogonal matrix, and \mathbf{R} is an $m \times m$ upper triangular matrix. \mathbf{Q} can be made up of the product of m Householder matrices. Here is how.

First construct the Householder matrix \mathbf{H}_1 which reflects/rotates the first column of \mathbf{x} in such a way that all but its first element is zeroed. e.g.

$$\mathbf{H}_1 \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots \\ x_{21} & x_{22} & x_{23} & \dots \\ x_{31} & x_{32} & x_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} x'_{11} & x'_{12} & x'_{13} & \dots \\ 0 & x'_{22} & x'_{23} & \dots \\ 0 & x'_{32} & x'_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

x'_{11} will be the length of the first column of \mathbf{X} . The next step is to calculate the Householder matrix, \mathbf{H}_2 , which will transform the second column of \mathbf{X} , so that its first element is unchanged, and every element beyond the second is zeroed. e.g.

$$\mathbf{H}_2 \begin{bmatrix} x'_{11} & x'_{12} & x'_{13} & \dots \\ 0 & x'_{22} & x'_{23} & \dots \\ 0 & x'_{32} & x'_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} x'_{11} & x'_{12} & x'_{13} & \dots \\ 0 & x''_{22} & x''_{23} & \dots \\ 0 & 0 & x''_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Notice that the fact that \mathbf{H}_2 is only acting on elements from the second row down has two implications: (i) the first row is unchanged by \mathbf{H}_2 and (ii) the first column is unchanged, since in this case \mathbf{H}_2 simply transforms zero to zero.

Continuing in the same way we eventually reach the situation where

$$\mathbf{H}_m \mathbf{H}_{m-1} \cdots \mathbf{H}_1 \mathbf{X} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

Hence $\mathbf{Q}^T = \mathbf{H}_m \mathbf{H}_{m-1} \cdots \mathbf{H}_1$ implying that $\mathbf{Q} = \mathbf{H}_1 \mathbf{H}_2 \cdots \mathbf{H}_m$.

Note that it is quite common to write the QR decomposition as $\mathbf{X} = \mathbf{QR}$ where \mathbf{Q} is the first m columns of the orthogonal matrix just constructed. Whichever convention is used, \mathbf{Q} is always stored as a series of m Householder matrix components. See `?qr` in R for details about using QR decompositions in R.

A.7 Choleski decomposition

The Choleski decomposition is a very efficient way of finding the ‘square root’ of a positive definite matrix. A positive definite matrix, \mathbf{D} , is one for which $\mathbf{x}^T \mathbf{D} \mathbf{x} > 0$ for *any* non-zero vector \mathbf{x} (of appropriate dimension). Positive definite matrices are symmetric and have strictly positive eigenvalues.

The Choleski decomposition of \mathbf{D} is

$$\mathbf{D} = \mathbf{R}^T \mathbf{R}$$

where \mathbf{R} is an upper triangular matrix of the same dimension as \mathbf{D} . The algorithm for constructing the Choleski factor, \mathbf{R} , is quite simple, and can be found in any textbook on numerical linear algebra, for example Golub and van Loan (1996) or Watkins (1991). The Choleski decomposition costs around $n^3/6$ operations for an $n \times n$ matrix, which makes it a rather efficient way of solving linear systems involving positive definite matrices. For example, the \mathbf{x} satisfying $\mathbf{D}\mathbf{x} = \mathbf{y}$ where \mathbf{D} is the \mathbf{x} satisfying

$$\mathbf{R}^T \mathbf{R} \mathbf{x} = \mathbf{y}$$

where \mathbf{R} is the Choleski factor of \mathbf{D} . Because \mathbf{R} is triangular it is cheap to solve for $\mathbf{R}\mathbf{x}$ and then for \mathbf{x} . The following code provides an example in R.

```
n <- 1000
D <- matrix(runif(n*n),n,n)
D <- D %*% t(D) # create a +ve def. matrix
y <- runif(n) # and a y
## now solve Dx=y, efficiently, using the Choleski factor of D
R <- chol(D)
x <- backsolve(R,y,transpose=TRUE)
x <- backsolve(R,y)
```

A.8 Eigen-decomposition

Consider an $n \times n$ matrix \mathbf{A} . There exist n scalars λ_i and n corresponding vectors \mathbf{u}_i such that

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i,$$

i.e. multiplication of \mathbf{u}_i by \mathbf{A} results in a scalar multiple of \mathbf{u}_i . The λ_i are known as the *eigenvalues* (own-values) of \mathbf{A} , and the corresponding \mathbf{u}_i are the *eigenvectors* (strictly ‘right eigenvectors’, since ‘left eigenvectors’ can also be defined).

In this book we only need eigenvalues and eigenvectors of symmetric matrices, so suppose that \mathbf{A} is symmetric. In that case it is possible to write

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T$$

where the eigenvectors, \mathbf{u}_i , are the columns of the $n \times n$ orthogonal matrix \mathbf{U} , while the eigenvalues λ_i provide the diagonal elements of the diagonal matrix Λ . Such a decomposition is known as an *eigen-decomposition* or *spectral decomposition*.

The eigen-decomposition of \mathbf{A} provides an interesting geometric interpretation of how any n vector, \mathbf{x} , is transformed by multiplication by \mathbf{A} . The product \mathbf{Ax} can be written as $\mathbf{U}\Lambda\mathbf{U}^T\mathbf{x}$: so \mathbf{U}^T first rotates \mathbf{x} into the ‘eigenspace’ of \mathbf{A} , the resulting vector then has each of its elements multiplied by an eigenvector of \mathbf{A} , before being rotated back into the original space by \mathbf{U} .

It is easy to see that for any integer m , $\mathbf{A}^m = \mathbf{U}\Lambda^m\mathbf{U}^\top$, which, given the eigen-decomposition, is rather cheap to evaluate, since Λ is diagonal. This result also means that any function with a power series expansion has a natural generalization to the matrix setting. For example, the exponential of a matrix would be

$$\exp(\mathbf{A}) = \mathbf{U} \exp(\Lambda) \mathbf{U}^\top$$

where $\exp(\Lambda)$ is the diagonal matrix with $\exp(\lambda_i)$ as the i^{th} element on its leading diagonal.

Here are some other useful results related to eigenvalues:

- The ‘rank’ of a matrix is the number of non-zero eigenvalues that it possesses. The rank is the number of independent rows/columns of a matrix.
- A symmetric matrix is positive definite if all its eigenvalues are strictly positive. i.e. $\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0 \forall \mathbf{x} \neq 0$.
- A symmetric matrix is positive semi definite if all its eigenvalues are non-negative. i.e. $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0 \forall \mathbf{x} \neq 0$.
- The trace of a matrix is also the sum of its eigenvalues.
- The determinant of a matrix is the product of its eigenvalues.

Numerical calculation of eigenvalues and eigenvectors is quite a specialized topic. Algorithms usually start by reducing the matrix to tri-diagonal or bi-diagonal form by successive application of Householder rotations from the left and right. By orthogonality of the Householder rotations, the reduced matrix has the same eigenvalues as the original matrix. The eigenvalues themselves are then found by an iterative procedure, usually the QR-algorithm (not to be confused with the QR decomposition!) see Golub and van Loan (1996) or Watkins (1991) for details. See `?eigen` for details of the eigen routines available in R. Sometimes only the largest magnitude eigenvalues and corresponding eigenvectors are required, in which case see section A.11.

A.9 Singular value decomposition

The singular value decomposition is rather like the eigen-decomposition, except that it can be calculated for any real matrix (not just square ones), and its components are all real (eigenvalues and vectors can be complex for non-symmetric matrices). If \mathbf{X} is an $n \times m$ real matrix ($m \leq n$, say) then it is possible to write

$$\mathbf{X} = \mathbf{UDV}^\top$$

where the columns of the $n \times m$ matrix \mathbf{U} are the first m columns of an orthogonal matrix (so that $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_m$, but $\mathbf{U}\mathbf{U}^\top \neq \mathbf{I}_n$), \mathbf{V} is an $m \times m$ orthogonal matrix and \mathbf{D} is a diagonal matrix, the diagonal elements of which are the *singular values* of \mathbf{X} .

The i^{th} singular value of \mathbf{X} is the positive square root of the i^{th} eigenvalue of $\mathbf{X}^\top \mathbf{X}$, so for a symmetric matrix, the eigenvalues and singular values are the same. Singular values are so called, because there is a sense in which they indicate the ‘distance’

between a matrix and the ‘nearest’ singular matrix. The ratio of the largest to smallest singular values is known as the ‘condition number’ of a matrix, and provides a useful guide to the numerical stability of calculations involving the matrix: again, see Golub and van Loan (1996) or Watkins (1991) for details.

The number of non-zero singular values gives the rank of a matrix, and the singular value decomposition is the most reliable method for estimating the rank of a matrix numerically.

Again, the calculation of the singular value decomposition starts by transforming the matrix to a bi-diagonal form, after which, iteration is used to find the singular values themselves (once again, see Golub and van Loan, 1996; Watkins, 1991).

A.10 Pivoting

Some matrix decompositions have better numerical properties if they are applied, not to the original matrix, but to a matrix in which the columns and/or rows of the matrix have been permuted. This is known as pivoting. Some properties of decompositions are invariant to this pivoting, whilst for other uses the pivoting has to be unscrambled after the decomposition.

As an example, if pivoting is used with a Choleski decomposition, then it is possible to find a square root of a positive *semi-* definite matrix, by this method. Here is an example in R

```
> B <- matrix(runif(200*100),200,100)
> B <- B%*%t(B)    ## B is now +ve semi-definite...
> chol(B)          ## ... so un-pivoted Choleski fails
Error in chol(B) : the leading minor of order 101 is
not positive definite
> R <- chol(B, pivot = TRUE)      ## ok with pivoting
> piv <- order(attr(R, "pivot")) ## extract pivot index
> R <- R[, piv]      ## unscramble pivoting
> range(B-t(R)%*%R) ## check that result is valid sqrt
[1] -4.867218e-12  4.444445e-12 ## it is!
```

of course, what is lost here is that R is no-longer upper triangular.

A.11 Lanczos iteration

Lanczos iteration (see e.g. Demmel, 1997) is a method which can be used to obtain the rank k truncated eigen-decomposition of a symmetric matrix, \mathbf{E} , in $O(kn^2)$ operations, by iteratively building up a tridiagonal matrix the eigenvalues of which converge (in order of decreasing magnitude) to those required, as iteration proceeds.

The algorithm is iterative, and at the i^{th} iteration produces an $(i \times i)$ symmetric tri-diagonal matrix (\mathbf{K}_i , say), the eigenvalues of which approximate the i largest

magnitude eigenvalues of the original matrix: these eigenvalues converge as the iteration proceeds, with those of largest magnitude converging first. The eigenvalues and vectors of \mathbf{K}_i can be obtained in order i^3 operations, using the usual QR algorithm[†] with accumulation of the eigenvectors. In principle it should be possible to get away with $O(i^2)$ operations by only using the QR algorithm to find the eigenvalues and then inverse iteration to find the eigenvectors, but I experienced some stability problems when using a simple implementation of inverse iteration to do this, particularly for thin plate regression splines of one predictor variable. In any case $i^3 \ll n^2$ in most cases. The eigenvectors of the original matrix are easily obtained from the eigenvectors of \mathbf{K}_i .

A complete version of the algorithm, suitable for finding the truncated decomposition of \mathbf{E} is as follows.

1. Let \mathbf{b} be an arbitrary non-zero n vector[‡].
2. Set $\mathbf{q}_1 \leftarrow \mathbf{b}/\|\mathbf{b}\|$.
3. Repeat steps (4) to (12) for $j = 1, 2, \dots$ until enough eigenvectors have converged.
4. Form $\mathbf{c} \leftarrow \mathbf{E}\mathbf{q}_j$.
5. Calculate $\gamma_j \leftarrow \mathbf{q}_j^\top \mathbf{c}$
6. Reorthogonalize \mathbf{c} to ensure numerical stability, by performing the following step **twice**:

$$\mathbf{c} \leftarrow \mathbf{c} - \sum_{i=1}^{j-1} (\mathbf{c}^\top \mathbf{q}_i) \mathbf{q}_i$$

7. Set $\xi_j \leftarrow \|\mathbf{c}\|$
8. Set $\mathbf{q}_{j+1} \leftarrow \mathbf{c}/\xi_j$
9. Let \mathbf{K}_j be the $(j \times j)$ tridiagonal matrix with $\gamma_1, \dots, \gamma_j$ on the leading diagonal, and ξ_1, \dots, ξ_{j-1} on the leading sub- and super- diagonals.
10. If iteration has proceeded far enough to make it worthwhile, find the eigen-decomposition (spectral decomposition) $\mathbf{K}_j = \mathbf{V}\Lambda\mathbf{V}^\top$, where the columns of \mathbf{V} are eigenvectors of \mathbf{K}_j and Λ is diagonal with eigenvalues on leading diagonal.
11. Compute “error bounds” for each $\Lambda_{i,i}$: $|\xi_j V_{i,j}|$.
12. Use the error bounds to test for convergence of the k largest magnitude eigenvalues. Terminate the loop if all are converged.
13. The i^{th} eigenvalue of \mathbf{E} is $\Lambda_{i,i}$. The i^{th} eigenvector of \mathbf{E} is $\mathcal{Q}\mathbf{v}_i$, where \mathcal{Q} is the matrix whose columns are the \mathbf{q}_j (for all j calculated) and \mathbf{v}_i is the i^{th} column of \mathbf{V} (again calculated at the final iteration). Hence \mathbf{D}_k and \mathbf{U}_k can easily be formed.

[†] Not to be confused with the QR decomposition: they are completely different things.

[‡] It may be best to initialize this from a simple random number generator, to reduce the risk of starting out orthogonal to some eigenvector (exact repeatability can be ensured by starting from the same random number generator seed)

The given algorithm is stabilized by orthogonalization against all previous vectors \mathbf{q}_j : several selective orthogonalization schemes have been proposed to reduce the computational burden of this step, but I experienced convergence problems when trying to use these schemes with \mathbf{E} from the 1 covariate TPRS problem. In any case the computational cost of the method is dominated by the $O(n^2)$ step: $\mathbf{c} \leftarrow \mathbf{E}\mathbf{q}_j$, so the efficiency benefits of using a selective method are unlikely to be very great, in the current case (if \mathbf{E} were sparse then selective methods would offer a benefit).

Finally note that \mathbf{E} need not actually be formed and stored as a whole: it is only necessary that its product with a vector can be formed. For a fuller treatment of the Lanczos method see Demmel (1997), from which the algorithm given here has been modified.



APPENDIX B

Solutions to exercises

To avoid this appendix reaching the same length as the rest of the book, the following solutions do not generally include R output and plots, but only the code for generating them.

B.1 Chapter 1

1. $t_i = \beta d_i + \epsilon_i$ is a reasonable model. So the least squares estimate of β is

$$\hat{\beta} = \frac{\sum_i x_i y_i}{\sum_i x_i^2} = \frac{0.1 + 1.2 + 2 + 3}{1 + 9 + 16 + 25} = \frac{6.3}{51} \simeq .1235$$

implying a speed of $51/6.3 \simeq 8.1$ kilometres per hour.

- 2.

$$\frac{\partial}{\partial \beta} \sum_{i=1}^n (y_i - \beta)^2 = -2 \sum_{i=1}^n (y_i - \beta) = 0 \Rightarrow \hat{\beta} = \bar{y}$$

3. None of these.

- 4.(a)

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_{31} \\ y_{32} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{bmatrix}$$

The constraint $\beta_1 = 0$ ensures model identifiability (i.e. full column rank of \mathbf{X}): any other constraint doing the same is ok.

(b)

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{24} \\ y_{31} \\ y_{32} \\ y_{33} \\ y_{34} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_2 \\ \beta_3 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{14} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{24} \\ \epsilon_{31} \\ \epsilon_{32} \\ \epsilon_{33} \\ \epsilon_{34} \end{bmatrix}$$

Here $\beta_1 = \gamma_1 = 0$ have been chosen as constraints ensuring identifiability: of course there are many other possibilities (depending on what contrasts are of most interest).

(c)

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.1 \\ 1 & 0 & 0.4 \\ 1 & 1 & 0.5 \\ 1 & 1 & 0.3 \\ 1 & 1 & 0.4 \\ 1 & 1 & 0.7 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_2 \\ \gamma \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix}$$

5. The situation described in the question translates into a model of the form:

$$\mu_i = \begin{cases} kx_i + \alpha x_i^2 & \text{alloy 1} \\ kx_i + \beta x_i^2 & \text{alloy 2} \\ kx_i + \gamma x_i^2 & \text{alloy 3} \end{cases}$$

where $y_i \sim N(\mu_i, \sigma^2)$. Written out in full matrix-vector form this is

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \\ y_{16} \\ y_{17} \\ y_{18} \end{bmatrix} = \begin{bmatrix} x_1 & x_1^2 & 0 & 0 \\ x_2 & x_2^2 & 0 & 0 \\ x_3 & x_3^2 & 0 & 0 \\ x_4 & x_4^2 & 0 & 0 \\ x_5 & x_5^2 & 0 & 0 \\ x_6 & x_6^2 & 0 & 0 \\ x_1 & 0 & x_1^2 & 0 \\ x_2 & 0 & x_2^2 & 0 \\ x_3 & 0 & x_3^2 & 0 \\ x_4 & 0 & x_4^2 & 0 \\ x_5 & 0 & x_5^2 & 0 \\ x_6 & 0 & x_6^2 & 0 \\ x_1 & 0 & 0 & x_1^2 \\ x_2 & 0 & 0 & x_2^2 \\ x_3 & 0 & 0 & x_3^2 \\ x_4 & 0 & 0 & x_4^2 \\ x_5 & 0 & 0 & x_5^2 \\ x_6 & 0 & 0 & x_6^2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \\ \epsilon_9 \\ \epsilon_{10} \\ \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{14} \\ \epsilon_{15} \\ \epsilon_{16} \\ \epsilon_{17} \\ \epsilon_{18} \end{bmatrix}$$

6.

$$\hat{\mu} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \Rightarrow \mathbf{X}^\top \hat{\mu} = \mathbf{X}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{y}$$

If the model has an intercept then one column of \mathbf{X} is a column of ones, so that one of the equations in $\mathbf{X}^\top \hat{\mu} = \mathbf{X}^\top \mathbf{y}$ is simply $\sum_i \hat{\mu}_i = \sum_i y_i$. Re-arrangement of this shows that the residuals must sum to zero.

7. Given that $\mathbb{E}(r_i^2) = \sigma^2$,

$$\mathbb{E}(\|\mathbf{r}\|^2) = \sum_{i=0}^{n-p} \mathbb{E}(r_i^2) = (n-p)\sigma^2,$$

and the result follows.

8.(a) library(MASS)

```
m1 <- lm(loss~hard+tens+I(hard*tens)+I(hard^2)+I(tens^2)+
I(hard^2*tens)+I(tens^2*hard)+I(tens^3)+I(hard^3),Rubber)
plot(m1)    ## residuals OK
summary(m1) ## p-values => drop I(tens^2*hard)
m2 <- update(m1,.~.-I(tens^2*hard))
summary(m2)
m3 <- update(m2,.~.-hard)
summary(m3)
m4 <- update(m3,.~.-1)
summary(m4)
m5 <- update(m4,.~.-I(hard^2))
summary(m5) ## p-values => keep all remaining
plot(m5)    ## residuals OK
```

(b) AIC(m1,m2,m3,m4,m5)

```
m6 <- step(m1)
```

```
(c) m <- 40; attach(Rubber)
    mt <- seq(min(tens),max(tens),length=m)
    mh <- seq(min(hard),max(hard),length=m)
    lp <- predict(m6,data.frame(hard=rep(mh,rep(m,m))),
                  tens=rep(mt,m)))
    contour(mt,mh,matrix(lp,m,m),xlab="tens",ylab="hard")
    points(tens,hard)
    detach(Rubber)

9. > wm <- lm(breaks~wool*tension,warpbreaks)
> plot(wm)      # residuals OK
> anova(wm)
Analysis of Variance Table

Response: breaks
          Df Sum Sq Mean Sq F value    Pr(>F)
wool        1  450.7   450.7  3.7653 0.0582130 .
tension     2 2034.3   1017.1  8.4980 0.0006926 ***
wool:tension 2 1002.8    501.4  4.1891 0.0210442 *
Residuals   48 5745.1    119.7
---
## ... so there is evidence for a wool:tension interaction.

> with(warpbreaks,interaction.plot(tension,wool,breaks))
10.(a) > cm1 <- lm(dist ~ speed + I(speed^2),cars)
> summary(cm1)
## Intercept has very high p-value, so drop it
> cm2 <- lm(dist ~ speed + I(speed^2)-1,cars)
> summary(cm2)
## both terms now significant, but try the alternative of
## dropping 'speed'
> cm3 <- lm(dist ~ I(speed^2),cars)
> AIC(cm1,cm2,cm3)
      df      AIC
cm1  4 418.7721
cm2  3 416.8016
cm3  3 416.9860
> plot(cm2)

Clearly cm2, with speed and speed squared terms is to be preferred, but note
that variance seems to be increasing with mean a little: perhaps a GLM, better?

(b) In seconds, the answer is obtained as follows
> b <- coef(cm2)
> 5280/(b[1]*60^2)
1.183722

This is a long time, but would have a rather wide associated confidence interval.
The stopping distances in cars are quite long relative to those usually quoted.

(c) The usual argument is really nonsense, and seems to result from confusing
```

continuous and factor variables. In the current case it would condemn us to leaving in the constant, and tolerating very high estimator uncertainty, even though hypothesis testing, AIC, and the physical mechanisms underlying the data all suggest dropping it.

11. The following is a version of the function that you should end up with.

```
fitlm <- function(y,X)
{ qrx <- qr(X)                      ## get QR decomposition
  y <- qr.qty(qrx,y)                  ## form Q'y efficiently
  R <- qr.R(qrx)                     ## extract R
  p <- ncol(R); n <- length(y)       ## get dimensions
  f <- y[1:p]; r <- y[(p+1):n]## partition Q'y
  beta <- backsolve(R,f)            ## parameter estimates (a)
  sig2 <- sum(r^2)/(n-p)             ## resid variance estimate (c)
  Ri <- backsolve(R,diag(ncol(R))) ## inverse of R matrix
  Vb <- Ri%*%t(Ri)*sig2           ## covariance matrix
  se <- diag(Vb)^.5                 ## standard errors (c)
  F.ratio <- f^2/sig2               ## sequential F-ratios
  seq.p.val <- 1-pf(F.ratio,1,n-p) ## seq. p-values (e)
  list(beta=beta,se=se,sig2=sig2,seq.p.val=seq.p.val,df=n-p)
}
```

The following code uses the function to answer some of the question parts.

- ```
get example X ...
X <- model.matrix(dist ~ speed + I(speed^2),cars)
cm <- fitlm(cars$dist,X) # used fitting function
cm$beta;cm$se # print estimates and s.e.s (a,c)
cm1<-lm(dist ~ speed + I(speed^2),cars) # equiv. lm call
summary(cm1) # check estimates and s.e.s (b,c)
t.ratio <- cm$beta/cm$se # form t-ratios
p.val <- pt(-abs(t.ratio),df=cm$df)*2
p.val # print evaluated p-values (d)
print sequential ANOVA p-values, and check them (e)
cm$seq.p.val
anova(cm1)

12. X <- model.matrix(~spray-1,InsectSprays)
X <- cbind(rep(1,nrow(X)),X) # redundant model matrix
C <- matrix(c(0,rep(1,6)),1,7) # constraints
qrc <- qr(t(C)) # QR decomp. of C'
use fact that Q=[D:Z] and XQ = (Q'X')' to form XZ ...
XZ <- t(qr.qty(qrc,t(X)))[,2:7]
m1 <- lm(InsectSprays$count~XZ-1) # fit model
bz <- coef(m1) # estimates in constrained parameterization
form b = Z b_z, using fact that Q=[D:Z], again
b <- c(0,bz)
b <- qr.qy(qrc,b)
sum(b[2:7])

13.(a) EV.func <- function(b,g,h)
{ mu <- b[1]*g^b[2]*h^b[3]
```

```

J <- cbind(g^b[2]*h^b[3],mu*log(g),mu*log(h))
list(mu=mu,J=J)
}
(b) > attach(trees)
> b <- c(.002,2,1);b.old <- 100*b+100
> while (sum(abs(b-b.old))>1e-7*sum(abs(b.old))) {
+ EV <- EV.func(b,Girth,Height)
+ z <- (Volume-EV$mu) + EV$J%*%b
+ b.old <- b
+ b <- coef(lm(z~EV$J-1))
+ }
> b
0.001448827 1.996921475 1.087646524
(c) > sig2 <- sum((Volume - EV$mu)^2)/(nrow(trees)-3)
> Vb <- solve(t(EV$J) %*% EV$J)*sig2
> se <- diag(Vb)^.5;se
[1] 0.001366994 0.082077439 0.242158811

```

## B.2 Chapter 2

1.(a)  $\mu \equiv \mathbb{E}(Y) = (1-p) \times 0 + p \times 1 = p$ .

(b)

$$\begin{aligned}
f(y) &= \exp(\log(\mu^y(1-\mu)^{1-y})) \\
&= \exp(y \log(\mu) + (1-y) \log(1-\mu)) \\
&= \exp\left(y \log\left(\frac{\mu}{1-\mu}\right) + \log(1-\mu)\right).
\end{aligned}$$

Now let  $\theta = \log(\mu/(1-\mu))$  so that  $e^\theta = \mu/(1-\mu)$  and therefor

$$1 + e^\theta = \frac{1}{1-\mu} \Rightarrow -\log(1 + e^\theta) = \log(1-\mu) \Rightarrow b(\theta) = \log(1 + e^\theta).$$

Hence,

$$f(y) = \exp(y\theta - b(\theta)),$$

which is exponential form with  $a(\phi) = \phi = 1$  and  $c(y, \phi) \equiv 0$ . So the Bernoulli distribution is in the exponential family.

- (c) A canonical link is one which when applied to the mean,  $\mu$ , gives the canonical parameter, so for the Bernoulli it is clearly:

$$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$$

a special case of the logit link.

2. After completing all 4 parts, you get something like the following:

```

example glm fit...
b <- glm(y~x,family=binomial,weights=rep(m,n))
reps <- 200;mu <- fitted(b)

```

```

rsd <- matrix(0,reps,n) # array for simulated resids
runs <- rep(0,reps) # array for simulated run counts
for (i in 1:reps) { # simulation loop
 ys <- rbinom(1:n,m,mu) # simulate from fitted model
 ## refit model to simulated data
 br <- glm(ys~x,family=binomial,weights=rep(m,n))
 rs <- residuals(br) # simulated resids (meet assumptions)
 rsd[i,] <- sort(rs) # store sorted residuals
 fv.sort <- sort(fitted(br),index.return=TRUE)
 rs <- rs[fv.sort$ix] # order resids by sorted fit values
 rs <- rs > 0 # check runs of +ve, -ve resids
 runs[i] <- sum(rs[1:(n-1)]!=rs[2:n])
}
plot original ordered residuals, and simulation envelope
for (i in 1:n) rsd[,i] <- sort(rsd[,i])
par(mfrow=c(1,1))
plot(sort(residuals(b)),(1:n-.5)/n) # original
plot 95% envelope
lines(rsd[5,],(1:n-.5)/n);lines(rsd[reps-5,],(1:n-.5)/n)

compare original runs to distribution under independence
rs <- residuals(b)
fv.sort <- sort(fitted(b),index.return=TRUE)
rs <- rs[fv.sort$ix]
rs <- rs > 0
obs.runs <- sum(rs[1:(n-1)]!=rs[2:n])
sum(runs>obs.runs)

```

3. First read in the data:

```

> count <- c(53,414,11,37,0,16,4,139)
> death <- factor(c(1,0,1,0,1,0,1,0))
> defendant <- factor(c(0,0,1,1,0,0,1,1))
> victim <- factor(c(0,0,0,1,1,1,1))
> levels(death) <- c("no","yes")
> levels(defendant) <- c("white","black")
> levels(victim) <- c("white","black")

(a) > sum(count[death=="yes"&defendant=="black"])/
+ sum(count[defendant=="black"])
[1] 0.07853403
> sum(count[death=="yes"&defendant=="white"])/
+ sum(count[defendant=="white"])
[1] 0.1097308

(b) > dm <- glm(count~death*victim+death*defendant+
+ victim*defendant,family=poisson(link=log))
> summary(dm)
[edited]
Coefficients:
Estimate Std Err z value Pr(>|z|)
(Intercept) 6.02631 0.04913 122.669 < 2e-16

```

```

deathyes -2.05946 0.14585 -14.121 < 2e-16
victimblack -3.26517 0.25478 -12.816 < 2e-16
defendblack -2.42032 0.17155 -14.109 < 2e-16
deathyes:victimblack -2.40444 0.60061 -4.003 6.25e-05
deathyes:defendblack 0.86780 0.36707 2.364 0.0181
victimblack:defendblack 4.59497 0.31353 14.656 < 2e-16

> dm0 <- glm(count~death*victim+victim*defendant,
+ family=poisson(link=log))
> anova(dm0,dm,test="Chisq")
Analysis of Deviance Table

Model 1: count ~ death * victim + victim * defendant
Model 2: count ~ death * victim + death * defendant +
 victim * defendant
Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1 2 5.3940
2 1 0.3798 1 5.0142 0.0251

```

A model in which the counts depend on all two way interactions seems appropriate. Note that the least significant interaction is between the defendant's skin 'colour' and death penalty, but that dropping this term does not seem to be justified (AIC also suggests leaving it in).

- (c) From the summary table it is clear that the strongest and most significant association is between skin colour of victim and defendant: blacks kill blacks and whites kill whites, for the most part. Next in terms of significance and strength of effect is the association between victims colour, and death penalty: the death penalty is less likely if the victim is black. Finally, being black is associated with an increased likelihood of being sentenced to death, once these other associations have been taken into account. Most of these effects can be ascertained by careful examination of the original table, but their significance and the relative strength of the effects are striking results from the modelling.

4.

$$\begin{aligned} \int_{y_i}^{\mu_i} \frac{y_i - z}{\phi z} dz &= \frac{1}{\phi} [y_i \log(z) - z]_{y_i}^{\mu_i} \\ &= \frac{1}{\phi} [y_i \log(\mu_i/y_i) - \mu_i + y_i] \end{aligned}$$

Since the quasi likelihood of the saturated model is zero, the corresponding deviance is simply

$$2y_i \log(y_i/\mu_i) - 2(y_i - \mu_i),$$

which corresponds to the Poisson deviance.

5.  $\sum_i w_i(y_i - X_i\beta)^2 \equiv \sum_i (\tilde{y}_i - \tilde{X}_i\beta)^2$  where  $\tilde{y}_i = \sqrt{w_i}y_i$  and  $\tilde{X}_{ij} = \sqrt{w_i}X_{ij}$ . Hence re-using the results from section 1.3.7 we have  $(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{y}}$ . Re-writing this in terms of  $\mathbf{X}$  and  $\mathbf{W}$  yields the required result.

- 6.(a)  $\mathbb{E}(z_i) = g'(\mu_i)(\mathbb{E}(y_i) - \mu_i) + \eta_i = 0 + \mathbf{X}_i\beta$

(b)

$$\begin{aligned}\text{var}(z_i) &= \text{var}\{g'(\mu_i)y_i\} \\ &= g'(\mu_i)^2 \text{var}(y_i) \\ &= g'(\mu_i)^2 V(\mu_i)\phi = w_i^{-1}\phi.\end{aligned}$$

Furthermore, since the  $y_i$  are independent, then so are the  $z_i$ , so the covariance matrix of the  $z_i$ 's is  $\mathbf{W}^{-1}\phi$ , as required.

- (c) From the previous question we have that  $\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{z}$ . So, by the results of A.2 on transformation of covariance matrices, we have that the covariance matrix of  $\hat{\beta}$  is

$$(\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{W}^{-1} \mathbf{W} \mathbf{X} (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \phi = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \phi.$$

Similarly,

$$\mathbb{E}(\hat{\beta}) = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbb{E}(\mathbf{z}) = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{X} \beta = \beta.$$

- (d) If  $\mathbf{X}^T \mathbf{W} \mathbf{z}$  tends to multivariate normality than so must  $\hat{\beta}$ . Hence, in the large sample limit,  $\hat{\beta} \sim N(\beta, (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \phi)$ . Of course in practical application of GLMs we calculate  $\mathbf{z}$  and  $\mathbf{W}$  using  $\hat{\beta}$ , rather than  $\beta$ , but as sample size tends to infinity  $\hat{\beta} \rightarrow \beta$ , so the result still holds.

```
7. y<- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t<-1:13
x <- cbind(rep(1,13),t,t^2) # model matrix
mu <- y;eta <- log(mu) # initial values
ok <- TRUE
while (ok) {
 ## evalaute pseudodata and weights
 z <- (y-mu)/mu + eta
 w <- as.numeric(mu)
 ## fit weighted working linear model
 z <- sqrt(w)*z; WX <- sqrt(w)*x
 beta <- coef(lm(z~WX-1))
 ## evaluate new eta and mu
 eta.old <- eta
 eta <- X%*%beta
 mu <- exp(eta)
 ## test for convergence...
 if (max(abs(eta-eta.old))<1e-7*max(abs(eta))) ok <- FALSE
}
plot(t,y);lines(t,mu) # plot fit
```

8.(a)

$$\frac{1}{\mathbb{E}(c_i)} = \frac{1}{ad_i^m} + t$$

(b)

```
m <- 1
b <- glm(Consumption.Rate~I(1/Grouse.Density^m),
family=quasi(link=inverse,variance=mu),data=harrier)
```

(c) `plot(harrier$Grouse.Density, residuals(b))`  
...the plot shows a clear pattern if  $m = 1$ , and the parameter estimates lead to a rather odd curve.

(d) Re-using the code from (b) with different  $m$  values, suggests that  $m \approx 3.25$  produces the lowest deviance.

(e) `pd <- data.frame(Grouse.Density = seq(0,130,length=200))  
pr <- predict(b,newdata=pd,se=TRUE)  
with(harrier,plot(Grouse.Density,Consumption.Rate))  
lines(pd$Grouse.Density,1/pr$fit,col=2)  
lines(pd$Grouse.Density,1/(pr$fit-pr$se*2),col=3)  
lines(pd$Grouse.Density,1/(pr$fit+pr$se*2),col=3)`

(f) `ll <- function(b,cr,d)  
## evaluates -ve quasi- log likelihood of model  
## b is parameters, cr is consumption, d is density  
{ ## get expected consumption...
dm <- d^b[3]
Ec <- exp(b[1])*dm/(1+exp(b[1]))*exp(b[2])*dm
## appropriate quasi-likelihood...
ind <- cr>0 ## have to deal with cr==0 case
ql <- cr - Ec
ql[ind] <- ql[ind] + cr[ind]*log(Ec[ind]/cr[ind])
-sum(ql)
}
## Now fit model ...
fit <- optim(c(log(.4),log(10),3),ll,method="L-BFGS-B",
hessian=TRUE,cr=harrier$Consumption.Rate,
d=harrier$Grouse.Density)
## and plot results ...
b <- fit$par
d <- seq(0,130,length=200); dm <- d^b[3]
Ec <- exp(b[1])*dm/(1+exp(b[1]))*exp(b[2])*dm
with(harrier,plot(Grouse.Density,Consumption.Rate))
lines(d,Ec,col=2)`

9. `death <- as.numeric(ldeaths)  
month <- rep(1:12,6)  
time <- 1:72  
ldm <- glm(death ~ sin(month/12*2*pi)+cos(month/12*2*pi),
family=poisson(link=identity))  
plot(time,death,type="l");lines(time,fitted(ldm),col=2)  
summary(ldm)  
plot(ldm)`

The model is clearly inadequate: massive over-dispersion and clear pattern in the residuals. Probably residual autocorrelation should be modelled properly here, reflecting the time-series nature of the data.

10. `y<- c(12,14,33,50,67,74,123,141,165,204,253,246,240)  
t<-1:13  
b <- glm(y~t + I(t^2),family=poisson)`

```

log.lik <- b1 <- seq(.4,.7,length=100)
for (i in 1:100)
{ log.lik[i] <- logLik(glm(y~offset(b1[i]*t)+I(t^2),
 family=poisson))
}
plot(b1,log.lik,type="l")
points(coef(b)[2],logLik(b),pch=19)
abline(logLik(b)-qchisq(.95,df=1),0,lty=2)

```

From the resulting plot, the 95% CI is (0.43,0.68), slightly wider than the (0.47,0.65) found earlier, but in this case almost symmetric. One key difference between these intervals and the intervals covered in the text, is that they are not necessarily symmetric. Another is that parameter values within the interval always have higher likelihood than those outside it, and a third is that they are invariant to re-parameterization.

### B.3 Chapter 3

- ## polynomial fits ...

```

xx <- seq(min(x),max(x),length=200)
plot(x,y)
b<-lm(y~poly(x,5))
lines(xx,predict(b,data.frame(x=xx)))
b<-lm(y~poly(x,10))
lines(xx,predict(b,data.frame(x=xx)),col=2)
spline fits ...
sb <- function(x,xk) { abs(x-xk)^3}
q<-11
xk<-((1:(q-2)/(q-1))*10)^.5
lazy person's formula construction ...
form<-paste("sb(x,xk[,1:(q-2),])",sep="",collapse="+")
form <- paste("y~x+",form)
b<-lm(formula(form))
lines(xx,predict(b,data.frame(x=xx)),col=3)

```

Note the wild behaviour of the order 10 polynomial, compared to the much better behaviour of the spline model of the same rank.

The difference in behaviour of the two rank 11 models is perhaps unsurprising. The theoretical justification for polynomial approximations of unknown functions would probably be Taylor's theorem: but this is concerned with getting good approximations in the vicinity of some particular point of interest: it is clear from the theorem that the approximation will eventually become very poor as we move away from that point. The theoretical justifications for splines are much more concerned with properties of the function over the whole region of interest.

- ## x,y, and xx from previous question

```

b1 <- lm(form)
plot(x,y)
lines(xx,predict(b1,data.frame(x=xx)),col=4)
X <- model.matrix(b1) # extract model matrix

```

```

beta <- solve(t(X) %*% X, t(X) %*% y, tol=0)
b1$coefficients <- beta # trick for simple prediction
lines(xx, predict(b1, data.frame(x=xx)), col=5)
... upping the basis dimension to 11, makes the normal equations estimates perform very badly.

3. pls <- function(x,y,m=10,lambda=1,order=2)
{ n <- length(x)
 xk <- seq(min(x),max(x),length=m) # knot locations
 X <- matrix(0,n,m) # model matrix
 for (i in 1:m) X[,i] <- approx(xk,diag(m)[,i],x)$y
 D <- diff(diag(m),differences=order) # sqrt penalty matrix
 Xa <- rbind(X,lambda*D) # augmented model matrix
 b<-lm(c(y,rep(0,m-order))~Xa-1) # fit model
 edf<-sum(influence(b)$hat[1:n]) # effective d.o.f.
 fv<-fitted(b)[1:n]
 rss <- sum((y-fv)^2)
 list(gcv=rss/(n-edf)^2,edf=edf,fv=fv,coef=coef(b),xk=xk)
}
Example (at approx GCV optimum lambda)
library(MASS)
b <- pls(mcycle$time,mcycle$accel,m=20,lambda=.8)
plot(mcycle$time,accel);
lines(mcycle$time,b$fv)

```

4.

$$\begin{aligned}
S_p &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^T \mathbf{S}\boldsymbol{\beta} \\
&= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T \mathbf{S}\boldsymbol{\beta} \\
&= \mathbf{y}^T \mathbf{y} - 2\boldsymbol{\beta}^T \mathbf{X}^T \mathbf{y} + \boldsymbol{\beta}^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{S}) \boldsymbol{\beta}.
\end{aligned}$$

Differentiating  $S_p$  w.r.t.  $\boldsymbol{\beta}$  and setting to zero results in the system of equations

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{S}) \hat{\boldsymbol{\beta}} = \mathbf{X}^T \mathbf{y},$$

which yields the required result.

5. The result follows from the fact that the upper left  $n \times n$  submatrix of  $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T$  is  $\mathbf{X}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T$ , as the following shows:

$$\begin{aligned}
\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T &= \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix} (\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \begin{bmatrix} \mathbf{X}^T & \mathbf{B}^T \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{X}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T & \mathbf{X}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{B}^T \\ \mathbf{B}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{X}^T & \mathbf{B}(\mathbf{X}^T \mathbf{X} + \mathbf{S})^{-1} \mathbf{B}^T \end{bmatrix}
\end{aligned}$$

6. Zero. The most complex model will always be chosen, as this will allow the data to be fitted most closely (only for a set of data configurations of probability zero is this not true — for example data lying exactly on a straight line).  
7. Differentiating the basis expansion for  $f$ , we get  $f''(x) = \boldsymbol{\beta}^T \mathbf{d}(x)$  where  $d_j(x) =$

$b_j''(x)$ . Using the fact that a scalar is its own transpose we then have that:

$$\int f''(x)^2 dx = \int \beta^\top \mathbf{d}(x) \mathbf{d}(x)^\top \beta dx = \beta^\top \mathbf{S} \beta$$

where

$$\mathbf{S} = \int \mathbf{d}(x) \mathbf{d}(x)^\top dx.$$

8.

$$\begin{aligned} & \int \left( \frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left( \frac{\partial^2 f}{\partial x \partial z} \right)^2 + \left( \frac{\partial^2 f}{\partial z^2} \right)^2 dx dz = \\ & \int \left( \frac{\partial^2 f}{\partial x^2} \right)^2 dx dz + 2 \int \left( \frac{\partial^2 f}{\partial x \partial z} \right)^2 dx dz + \int \left( \frac{\partial^2 f}{\partial z^2} \right)^2 dx dz \end{aligned}$$

Treating each integral on the r.h.s. in a similar way to the integral in the previous question we find that the penalty can be written as  $\beta^\top \mathbf{S} \beta$  where

$$\mathbf{S} = \int \mathbf{d}_{xx}(x, z) \mathbf{d}_{xx}(x, z)^\top + 2 \mathbf{d}_{xz}(x, z) \mathbf{d}_{xz}(x, z)^\top + \mathbf{d}_{zz}(x, z) \mathbf{d}_{zz}(x, z)^\top dx dy.$$

Here the  $j^{\text{th}}$  element of  $\mathbf{d}_{xz}(s, z)$  is  $\partial^2 b_j / \partial x \partial z$ , with similar definitions for  $\mathbf{d}_{xx}$  and  $\mathbf{d}_{zz}$ . Obviously this sort of argument can be applied to all sorts of penalties involving integrals of sums of squared derivatives.

#### B.4 Chapter 4

1. Consider the spline defined by (4.2). At knot position  $x_j$ , we require that the derivative at  $x_j$  of the section of cubic to the left of  $x_j$  matches the derivative at  $x_j$  of the section of cubic to the right of  $x_j$ . Writing this condition out in full yields

$$\begin{aligned} -\frac{\beta_j}{h_j} + \frac{\beta_{j+1}}{h_j} + \delta_j \frac{h_j}{6} + \delta_{j+1} \frac{3h_j}{6} - \delta_{j+1} \frac{h_j}{6} = \\ -\frac{\beta_{j+1}}{h_{j+1}} + \frac{\beta_{j+2}}{h_{j+1}} - \delta_{j+1} \frac{3h_{j+1}}{6} + \delta_{j+1} \frac{h_{j+1}}{6} - \delta_{j+2} \frac{h_{j+1}}{6} \end{aligned}$$

and simple re-arrangement leads to

$$\begin{aligned} \frac{1}{h_j} \beta_j - \left( \frac{1}{h_j} + \frac{1}{h_{j+1}} \right) \beta_{j+1} + \frac{1}{h_{j+1}} \beta_{j+2} = \\ \frac{h_j}{6} \delta_j + \left( \frac{h_j}{3} + \frac{h_{j+1}}{3} \right) \delta_{j+1} + \frac{h_{j+1}}{6} \delta_{j+2}. \end{aligned}$$

With the additional restriction that  $\delta_1 = \delta_k = 0$ , this latter system repeated for  $j = 1, \dots, k-2$  is (4.3).

- 2.(a) Differentiating (4.2) twice, yields

$$f''(x) = \delta_j(x_{j+1} - x)/h_j + \delta_{j+1}(x - x_j)/h_j, \quad x_j \leq x \leq x_{j+1}.$$

By inspection this can be re-written in the required form.

- (b) Writing  $\mathbf{d}(x)$  as the vector with  $i^{\text{th}}$  element  $d_{i+1}(x)$  (the first and last  $d_i$ 's have coefficients zero, so are of no interest), it is easy to show (e.g. exercise 7, Chapter 3), that

$$\int f''(x)^2 dx = \boldsymbol{\delta}^{-\top} \int \mathbf{d}(x) \mathbf{d}(x)^\top dx \boldsymbol{\delta}^-.$$

Since each  $d_i(x)$  is non-zero over only 2 intervals, it is clear that  $\int \mathbf{d}(x) \mathbf{d}(x)^\top dx$  is tridiagonal, and it is also obviously symmetric, by construction. The  $(i-1)^{\text{th}}$  leading diagonal element is given by

$$\int_{x_{i-1}}^{x_{i+1}} d_i(x)^2 dx = \left[ \frac{(x - x_{i-1})^3}{3h_{i-1}^2} \right]_{x_{i-1}}^{x_i} - \left[ \frac{(x_{i+1} - x)^3}{3h_i^2} \right]_{x_i}^{x_{i+1}} = \frac{h_{i-1}}{3} + \frac{h_i}{3},$$

where  $i$  runs from 2 to  $k-1$ . In the same vein, the off diagonal elements  $(i-1, i)$  and  $(i, i-1)$  are given by:

$$\int_{x_{i-1}}^{x_i} d_i(x) d_{i-1}(x) dx = \int_{x_{i-1}}^{x_i} \frac{x - x_{i-1}}{h_{i-1}} \frac{x_i - x}{h_{i-1}} dx = \frac{h_{i-1}}{6}.$$

In other words  $\int \mathbf{d}(x) \mathbf{d}(x)^\top dx = \mathbf{B}$ , as required.

- (c) From (4.3) we have that  $\boldsymbol{\delta}^- = \mathbf{B}^{-1} \mathbf{D} \boldsymbol{\beta}$ , from which the result follows immediately by substitution.

3.

$$\begin{aligned} \mathbb{E}(\|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2) &= \mathbb{E}(\|\boldsymbol{\mu} + \boldsymbol{\epsilon} - \mathbf{A}\boldsymbol{\mu} - \mathbf{A}\boldsymbol{\epsilon}\|^2) \\ &= (\boldsymbol{\mu} + \boldsymbol{\epsilon} - \mathbf{A}\boldsymbol{\mu} - \mathbf{A}\boldsymbol{\epsilon})^\top (\boldsymbol{\mu} + \boldsymbol{\epsilon} - \mathbf{A}\boldsymbol{\mu} - \mathbf{A}\boldsymbol{\epsilon}) \\ &= \boldsymbol{\mu}^\top \boldsymbol{\mu} - \boldsymbol{\mu}^\top \mathbf{A}\boldsymbol{\mu} + \boldsymbol{\mu}^\top \mathbf{A}\mathbf{A}\boldsymbol{\mu} + \\ &\quad \mathbb{E}(\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}) - 2\mathbb{E}(\boldsymbol{\epsilon}^\top \mathbf{A}\boldsymbol{\epsilon}) + \mathbb{E}(\boldsymbol{\epsilon}^\top \mathbf{A}\mathbf{A}\boldsymbol{\epsilon}) \\ &= \mathbf{b}^\top \mathbf{b} + n\sigma^2 - 2\text{tr}(\mathbf{A})\sigma^2 + \text{tr}(\mathbf{A}\mathbf{A})\sigma^2 \end{aligned}$$

where  $\mathbb{E}(\boldsymbol{\epsilon}^\top \mathbf{A}\boldsymbol{\epsilon}) = \mathbb{E}(\text{tr}(\boldsymbol{\epsilon}^\top \mathbf{A}\boldsymbol{\epsilon})) = \mathbb{E}(\text{tr}(\mathbf{A}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top)) = \text{tr}(\mathbf{A}\sigma^2)$ , and similar have been used.

4. For binary random variables we have that,

$$\mathbb{E}\{(y_i - \mu_i)^2\} = \mathbb{E}(y_i^2) - 2\mu_i \mathbb{E}(y_i) + \mathbb{E}(\mu_i^2) = \mu - 2\mu^2 + \mu = \mu - \mu^2,$$

and  $V(\mu_i) = \mu_i(1 - \mu_i)$ . Substituting into the expression to be evaluated yields,

$$\frac{n \sum_i 1}{\{n - \text{tr}(\mathbf{A})\}^2} = \frac{n^2}{\{n - \text{tr}(\mathbf{A})\}^2},$$

as required. So the expected value of  $\mathbb{E}(\mathcal{V}_g^p) \rightarrow n^2/\{n - \text{tr}(\mathbf{A})\}^2$  as  $n \rightarrow \infty$ , provided that  $\hat{\mu}$  is consistent.

5.(a) In the regular parameterization:

$$\begin{aligned} \mathbb{E}(\hat{\boldsymbol{\beta}}) &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{S})^{-1} \mathbf{X}^\top \mathbb{E}(\mathbf{y}) \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{S})^{-1} \mathbf{X}^\top \mathbf{X} \boldsymbol{\beta}. \end{aligned}$$

In the natural parameterization this becomes:

$$\mathbb{E}(\hat{\beta}'') = (\mathbf{I} + \lambda \mathbf{D})^{-1} \beta''$$

So

$$\text{bias}(\hat{\beta}'') = -\beta''_i \lambda D_{ii} / (1 + \lambda D_{ii}).$$

So if  $\beta''_i = 0$  then its estimator is unbiased. Equally if the parameter is unpenalized because  $\lambda D_{ii} = 0$  then the estimator is unbiased. Clearly the bias will be small for small true parameter value or low penalization: it is only moderate or strongly penalized model components of substantial magnitude that are subject to substantial bias.

(b)

$$\begin{aligned}\mathbb{E}\{(\hat{\beta}_i - \beta_i)^2\} &= \mathbb{E}\{(\hat{\beta}_i - \mathbb{E}(\hat{\beta}_i) + \mathbb{E}(\hat{\beta}_i) - \beta_i)^2\} \\ &= \mathbb{E}\{(\hat{\beta}_i - \mathbb{E}(\hat{\beta}_i))^2\} + \mathbb{E}\{(\mathbb{E}(\hat{\beta}_i) - \beta_i)^2\} \\ &\quad + \mathbb{E}\{(\hat{\beta}_i - \mathbb{E}(\hat{\beta}_i))(\mathbb{E}(\hat{\beta}_i) - \beta_i)\} \\ &= \text{var}(\hat{\beta}_i) + \text{bias}(\hat{\beta}_i)^2 + 0\end{aligned}$$

(c) So the MSE,  $M$ , of  $\beta_i$  (natural parameterization) is

$$M = \frac{\sigma^2}{(1 + \lambda D_{ii})^2} + \frac{(\lambda D_{ii})^2 \beta_i^2}{(1 + \lambda D_{ii})^2} = \frac{\sigma^2 + (\lambda D_{ii} \beta_i)^2}{(1 + \lambda D_{ii})^2}$$

this expression makes it rather clear how increasing  $\lambda$  decreases variance while increasing bias.

(d) Writing

$$\frac{M}{\sigma^2} = \frac{1 + k^2 r}{(1 + k)^2}$$

where  $\lambda D_{ii} = k$  and  $\beta_i^2 / \sigma^2 = r$ , it's clear that  $M/\sigma^2$  (and hence  $M$ ) is minimized by choosing  $\lambda$  so that  $k = 1/r$ , in which case  $M = \sigma^2 r / (1 + r)$ . In the natural parameterization the unpenalized estimator variance (and hence unpenalized MSE) is  $\sigma^2$ , and  $\sigma^2 r / (1 + r)$  is clearly always less than this. If  $\lambda$  could be chosen to minimize the MSE for a particular parameter, then from the preceding formulae, it is clear that small magnitude  $\beta_i$ 's would lead to high penalization and MSE dominated by the bias term, while large magnitude  $\beta_i$ 's would be lightly penalized, with the MSE dominated by the variance.

(e) In the natural parameterization the Bayesian prior variance for  $\beta_i$  is  $\sigma^2 / (\lambda_i D_{ii})$ . Since the prior expected value (for penalized terms) is 0, this means that  $\mathbb{E}(\beta_i^2) = \sigma^2 / (\lambda_i D_{ii})$  according to the prior. If this is representative of the typical size of  $\beta_i^2$  then the typical size of the bias  $(\hat{\beta}_i)^2$  would be:

$$\frac{\lambda D_{ii} \sigma^2}{(1 + \lambda D_{ii})^2},$$

implying that the squared bias should typically be bounded above by something like  $\sigma^2/4$ .

- 6.(a) The influence matrix for this problem is obviously  $\mathbf{A} = (\mathbf{I} + \lambda\mathbf{I})^{-1}$ . Hence  $\text{tr}(\mathbf{A}) = n/(1 + \lambda)$  and  $\mu_i = y_i/(1 + \lambda)$ . Substituting into the expression for the OCV or GCV scores results in  $\mathcal{V}_o = \mathcal{V}_g = \sum_i y_i^2/n$ , which does not depend on  $\lambda$ .
- (b) If we were to drop a  $y_i$  from the model sum of squares term, then the only thing influencing the estimate of  $\mu_i$  would be the penalty term, which would be minimized by setting  $\mu_i = 0$ , whatever (positive) value  $\lambda$  takes. This complete decoupling, where each  $\mu_i$  is influenced only by its own  $y_i$ , will clearly cause cross validation to fail: if we leave out  $y_i$  then the corresponding  $\mu_i$  is always estimated as zero, since the other data have no influence on it, and this behaviour occurs for any possible value of  $\lambda$ . In a sense this is unsurprising, since there is actually no covariate in this problem, and hence nothing to indicate which  $y_i$  or  $\mu_i$  values should be close in value.
- (c) From the symmetry around  $x_2$  it suffices to examine only the integral of the penalty from  $x_1$  to  $x_2$ . Without loss of generality we can take  $x_1$  to be 0. If  $b$  is the slope of the line segment joining  $(x_1 = 0, \mu_1 = \mu^*)$  to  $(x_2, \mu_2)$  then  $\mu_2 = \mu^* + bx_2$ , and it is simpler to work in terms of  $b$ , initially. So

$$2P = \int_{x_1}^{x_3} \mu_i(x)^2 dx = 2 \int_0^{x_2} (\mu^* + bx)^2 dx$$

so that

$$P = \mu^{*2}x_2 + \mu^*bx_2^2 + b^2x_2^3/3.$$

To find the  $b$  minimizing  $P$ , set  $\frac{\partial P}{\partial b} = 0$ , which implies  $b = -3\mu^*/(2x_2)$  and hence  $\mu_2 = -\mu^*/2$ .

So, consider a set of 3 adjacent points with roughly similar  $y_i$  and  $\mu_i$  values: if we omit the middle point from the fitting, then the action of the penalty will send its  $\mu_i$  estimate to the other side of zero, from its neighbours. This is a rather unfortunate tendency. It means that the missing datum will always be better predicted with a high  $\lambda$  than with a lower one, since the higher  $\lambda$  will tend to shrink the  $\mu_i$  for the included data towards zero, which will mean that the  $\mu_i$  for the omitted datum will also be closer to zero, and hence less far from the omitted datum value. Hence cross validation will have the pathological tendency to always select the model  $\mu_i = 0 \forall i$ , an effect which can be demonstrated practically!

- (d) The first derivative penalty does not suffer from the problems of the other two penalties. In this case the action of the penalty is merely to try and flatten  $\mu(x)$  in the vicinity of an omitted datum: increased flattening with increased  $\lambda$  generally pulls  $\mu(x)$  away from the omitted datum, in the way that cross validation implicitly assumes will happen.
- (e) Generally, penalized regression smoothers can not decouple in the manner of the smoothers considered in this question: because each smoother has far fewer parameters than data, each  $\mu_i$  is necessarily dependent on several  $y_i$ , rather than just one: it is simply not possible for the penalty to do something bizarre to one  $\mu_i$  while leaving the others unchanged.

7.(a) library(splines)  
 pspline.XB <- function(x,q=10,m=2,p.m=2)  
 # Get model matrix and sqrt Penalty matrix for P-spline  
 { # first make knot sequence, k  
 k <- seq(min(x),max(x),length=q-m)  
 dk <- k[2]-k[1]  
 k <- c(k[1]-dk\*((m+1):1),k,k[q-m]+dk\*(1:(m+1)))  
 # now get model matrix and root penalty  
 X <- splineDesign(k,x,ord=m+2)  
 B <- diff(diag(q),difference=p.m)  
 list(X=X,B=B)  
 }  
 (b) n<-100  
 x <- sort(runif(n))  
 ps <- pspline.XB(x,q=9,m=2,p.m=2)  
 par(mfrow=c(3,3)) # plot the original basis functions  
 for (i in 1:9) plot(x,ps\$X[,i],type="l")  
 (c) S <- t(ps\$B) %\*% ps\$B  
 es <- eigen(S);U <- es\$vectors  
 XU <- ps\$X %\*% U # last p.m cols are penalty null space  
 par(mfrow=c(3,3)) # plot penalty eigenbasis functions  
 for (i in 1:9) plot(x,XU[,i],type="l")  
 (d) qrx <- qr(ps\$X) # QR of X  
 R <- qr.R(qrx)  
 RSR <- solve(t(R),S);RSR <- t(solve(t(R),t(RSR)))  
 ersr <- eigen(RSR)  
 U <- ersr\$vectors  
 Q <- qr.Q(qrx)  
 QU <- Q %\*% U  
 par(mfrow=c(3,3)) # plot the natural basis functions  
 for (i in 1:9) plot(x,QU[,i],type="l")

8.(a) The following answers (a) and (b). Note that rss is only returned in order to facilitate question 9.

```
fit.wPs <- function(y,X,B,lambda=0,w=rep(1,length(y)))

fit to y by weighted penalized least squares, X is

model matrix, B is sqrt penalty, lambda is smoothing p.

{ w <- as.numeric(w^.5)

n <- nrow(X)

X<-rbind(w*X,sqrt(lambda)*B)

y<-c(w*y,rep(0,nrow(B)))

b <- lm(y~X-1) # actually estimate model

trA <- sum(influence(b)$hat[1:n])

rss <- sum((y-fitted(b))[1:n]^2)

list(trA=trA,rss=rss,b=coef(b))

}

fitPoiPs <- function(y,X,B,lambda=0)

Fit Poisson model with log-link by P-IRLS
```

```

{ mu <- y;mu[mu==0] <- .1
 eta <- log(mu)
 converged <- FALSE
 dev <- ll.sat <- sum(dpois(y,y,log=TRUE))
 while (!converged) {
 z <- (y-mu)/mu + eta
 w <- mu
 fPs <- fit.wPs(z,X,B,lambda,w)
 eta <- X%*%fPs$b
 mu=exp(eta)
 old.dev <- dev
 dev <- 2*(ll.sat-sum(dpois(y,mu,log=TRUE)))
 if (abs(dev-old.dev)<1e-6*dev) converged <- TRUE
 }
 list(dev=dev,rss=fPs$rss,trA=fPs$trA,b=fPs$b,fv=mu)
}

(c) # set up P-spline using code from Q 7.(a)
library(splines)
ps <- pspline.XB(x,q=10,m=2,p.m=2)
lambda <- 1e-4;reps <- 60
sp <- trA <- gcv <- rep(0,reps)
for (i in 1:reps) { # loop through trial s.p.s
 fps <- fitPoiPs(y,psX,psB,lambda=lambda)
 trA[i] <- fps$trA;sp[i] <- lambda
 gcv[i] <- n*fps$dev/(n-trA[i])^2
 lambda <- lambda*1.3
}
plot(trA,gcv,type="l")
fps1 <- fitPoiPs(y,psX,psB,lambda=sp[gcv==min(gcv)])
plot(x,y);lines(x,fps1$fv)

```

9. Following on from Q.8, the following code estimates  $k$  and uses it in a modified GCV score.

```

k <- fps1$rss - fps1$dev
lambda <- 1e-4;reps <- 60
sp <- trA <- gcv <- rep(0,reps)
for (i in 1:reps) {
 fps <- fitPoiPs(y,psX,psB,lambda=lambda)
 trA[i] <- fps$trA;sp[i] <- lambda
 gcv[i] <- n*(fps$dev+k)/(n-trA[i])^2
 lambda <- lambda*1.3
}
fps2 <- fitPoiPs(y,psX,psB,lambda=sp[gcv==min(gcv)])
lines(x,fps2$fv,col=2) # added to Q8 plot

```

Repeated simulation of the question 8/9 example, suggests that the modified GCV score seldom gives very different results to the unmodified version, although, of course, it tends to smooth a little less.

10. eta <- function(r)  
 { # thin plate spline basis functions

```

ind <- r<=0
eta <- r
eta[!ind] <- r[!ind]^2*log(r[!ind])/(8*pi)
eta[ind] <- 0
eta
}
XSC <- function(x,xk=x)
{ # set up t.p.s., given covariates, x, and knots, xk
 n <- nrow(x);k <- nrow(xk)
 X <- matrix(1,n,k+3) # tps model matrix
 for (j in 1:k) {
 r <- sqrt((x[,1]-xk[j,1])^2+(x[,2]-xk[j,2])^2)
 x[,j] <- eta(r)
 }
 X[,j+2] <- x[,1];X[,j+3] <- x[,2]
 C <- matrix(0,3,k+3) # tps constraint matrix
 S <- matrix(0,k+3,k+3) # tps penalty matrix
 for (i in 1:k) {
 C[1,i]<-1;C[2,i] <- xk[i,1];C[3,i] <- xk[i,2]
 for (j in i:k) S[j,i]<-S[i,j] <-
 eta(sqrt(sum((xk[i,]-xk[j,])^2)))
 }
 list(X=X,S=S,C=C)
}
absorb.con <- function(X,S,C)
{ # get constraint null space, Z...
 qrc <- qr(t(C)) # QR=C', Q=[Y,Z]
 m <- nrow(C);k <- ncol(X)
 X <- t(qr.qty(qrc,t(X)))[, (m+1):k] # form XZ
 # now form Z'SZ ...
 S <- qr.qty(qrc,t(qr.qty(qrc,t(S))))[(m+1):k, (m+1):k]
 list(X=X,S=S,qrc=qrc)
}

fit.tps <- function(y,x,xk=x,lambda=0)
{ tp <- XSC(x,xk) # get tps matrices
 tp <- absorb.con(tpX,tpS,tp$C) # make unconstrained
 ev <- eigen(tp$S,symmetric=TRUE) # get sqrt penalty, rS
 rS <- ev$vectors%*%(ev$values^.5*ev$vectors)
 X <- rbind(tp$X,rS*sqrt(lambda)) # augmented model matrix
 z <- c(y,rep(0,ncol(rS))) # augmented data
 beta <- coef(lm(z~X-1)) # fit model
 beta <- qr.qy(tp$qrc,c(0,0,0,beta)) # backtransform beta
}

eval.tps <- function(x,beta,xk)
{ # evaluate tps at x, given parameters, beta, and knots, xk.
 k <- nrow(xk);n <- nrow(x)
 f <- rep(beta[k+1],n)

```

```

for (i in 1:k) {
 r <- sqrt((x[,1]-xk[i,1])^2+(x[,2]-xk[i,2])^2)
 f <- f + beta[i]*eta(r)
}
f <- f + beta[k+2]*x[,1] + beta[k+3]*x[,2]
}

select some 'knots', xk ...
ind <- sample(1:n, 100, replace=FALSE)
xk <- x[ind,]
fit model ...
beta <- fit.tps(y, x, xk=xk, lambda=.01)

contour truth and fit
par(mfrow=c(1,2))
xp <- matrix(0, 900, 2)
x1<-seq(0,1,length=30);x2<-seq(0,1,length=30)
xp[,1]<-rep(x1,30);xp[,2]<-rep(x2, rep(30,30))
truth<-matrix(test1(xp[,1],xp[,2]),30,30)
contour(x1,x2,truth)
fit <- matrix(eval.tps(xp,beta,xk),30,30)
contour(x1,x2,fit)
...obviously, many other solutions are possible.

```

11.(a)

$$\mathbf{A} = \mathbf{Q}\mathbf{U}(\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{U}^\top\mathbf{Q}^\top.$$

(b)

$$\begin{aligned} \text{tr}(\mathbf{A}) &= \text{tr}(\mathbf{Q}\mathbf{U}(\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{U}^\top\mathbf{Q}^\top) = \text{tr}((\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{U}^\top\mathbf{Q}^\top\mathbf{Q}\mathbf{U}) \\ &= \text{tr}((\mathbf{I} + \lambda\mathbf{D})^{-1}) = \sum_{i=1}^k \frac{1}{1 + \lambda D_{ii}}. \end{aligned}$$

(c)

$$\begin{aligned} \|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2 &= \mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{A}\mathbf{y} + \mathbf{y}^\top\mathbf{A}\mathbf{A}\mathbf{y} \\ &= \mathbf{y}^\top\mathbf{y} - 2\tilde{\mathbf{y}}^\top(\mathbf{I} + \lambda\mathbf{D})^{-1}\tilde{\mathbf{y}} + \tilde{\mathbf{y}}^\top(\mathbf{I} + \lambda\mathbf{D})^{-2}\tilde{\mathbf{y}} \end{aligned}$$

where  $\tilde{\mathbf{y}} = \mathbf{U}^\top\mathbf{Q}^\top\mathbf{y}$ . Once  $\mathbf{y}^\top\mathbf{y}$  and  $\tilde{\mathbf{y}}$  have been evaluated, it's clear that  $\|\mathbf{y} - \mathbf{A}\mathbf{y}\|^2$  can be evaluated in  $O(k)$  operations, since  $\mathbf{D}$  is diagonal, and  $\tilde{\mathbf{y}}$  is of dimension  $k$ . So the GCV score can be calculated in  $O(k)$  operations, for each trial  $\lambda$ .

## B.5 Chapter 5

1.(a) `data(hubble)`

```

h1 <- gam(V~s(D), data=hubble)
plot(h1) ## model is curved
h0 <- gam(V~D, data=hubble)

```

```
h1;h0
AIC(h1,h0)
```

The smooth (curved) model has a lower GCV and lower AIC score than the straight line model. On the face of it there is a suggestion that velocities are lower at very high distances than Hubble's law suggests. This would imply an accelerating expansion!

- (b) 

```
gam.check(h1) # oh dear
h2 <- gam(V~s(D), data=hubble, family=quasi(var=mu))
gam.check(h2) # not great, but better
h2
```

The residual plots for h1 are problematic: there is a clear relationship between the mean and the variance. Perhaps a quasi-likelihood approach might solve this. m2 does have somewhat better residual plots, although they are still not perfect. All evidence for departure from Hubble's has has now vanished.

- 2.(a) 

```
library(MASS)
par(mfrow=c(2,2))
mc <- gam(accel~s(times,k=40), data=mcycle)
plot(mc, residuals=TRUE, se=FALSE, pch=1)
```

Note the way the fitted curve dips too early.

- (b) 

```
mc1 <- lm(accel~poly(times,11), data=mcycle)
termplot(mc1, partial.resid=TRUE)
```

Notice the wild oscillations, unsupported by the data.

- (c) 

```
mc2 <- gam(accel~s(times,k=11,fx=TRUE), data=mcycle)
plot(mc2, residuals=TRUE, se=FALSE, pch=1)
```

  
...not much worse than the penalized fit.

- (d) 

```
mc3 <- gam(accel~s(times,k=11,fx=TRUE,bs="cr"), data=mcycle)
plot(mc3, residuals=TRUE, se=FALSE, pch=1)
```

So, mc is a bit better than mc2 which is a bit better than mc3 which is much better than mc1: i.e. the polynomial does much worse than any sort of spline, while regression splines are a bit worse than penalized splines, however the TPRS is almost as good as a penalized spline.

- (e) 

```
par(mfrow=c(1,1))
plot(mcycle$times, residuals(mc))
```

The first 20 residuals have much lower variance than the remainder. In addition, just after time 9 there is a substantial cluster of negative residuals, followed by a cluster of positive residuals, suggesting that the model is not capturing the mean acceleration correctly in this region: something which is also apparent in the default plot of mc: the model dips too early.

- (f) The following was run several times with different  $\alpha$  values, before settling on 400 as the weight which causes the final ratio to be approximately 1.

```
mcw <- gam(accel~s(times,k=40), data=mcycle,
 weights=c(rep(400,20),rep(1,113)))
plot(mcw, residuals=TRUE, pch=1)
rsd <- residuals(mcw)
```

```
plot(mcycle$times, rsd)
var(rsd[21:133])/var(rsd[1:20])
```

The model and residual plots are now very much better. Although this procedure was somewhat ad hoc it can only be better than ignoring the variance problem in this case.

- (g) The following uses the integrated squared third derivative as penalty ( $m=3$ ).

```
gam(accel~s(times, k=40, m=3), data=mcycle,
 weights=c(rep(400, 20), rep(1, 113)))
```

the original model perhaps failed to go quite deep enough at the minimum of the data: the new curve is fine. Further increase of  $m$  doesn't result in much further change.

- 3.(a) Fitting the model to  $\mathbf{I}_j$  and evaluating the fitted values,  $\hat{\mu}^*$ , is equivalent to forming  $\hat{\mu}^* = \mathbf{A}\mathbf{I}_j$ , but this is clearly just the  $j^{\text{th}}$  column of  $\mathbf{A}$ .

(b) 

```
library(MASS)
n <- nrow(mcycle)
A <- matrix(0, n, n)
for (i in 1:n) {
 mcycle$y<-mcycle$accel*0;mcycle$y[i] <- 1
 A[, i] <- fitted(gam(y~s(times, k=40), data=mcycle, sp=mc$sp))
}
```

(Actually this could be done more efficiently using the `fit=FALSE` option in `gam`.)

- (c) `rowSums(A)` shows that all the rows sum to 1. This has to happen, since by construction the model does not penalize the constant (or linear trend) part of the model. Hence if  $\mathbf{b}$  is a vector all elements of which have the same value,  $b$ , then we require that  $\mathbf{b} = \mathbf{Ab}$ . This means that  $b_i = \sum_{j=1}^n A_{ij}b_j \forall i$ , but this is equivalent to  $b = b \sum_{j=1}^n A_{ij} \forall i$ , which will only happen if  $\sum_{j=1}^n A_{ij} = 1 \forall i$ . i.e. the rows of  $\mathbf{A}$  must each sum to 1.

- (d) `plot(mcycle$times, A[, 65], type="l", ylim=c(-0.05, 0.15))`

Notice how the kernel peaks at the time of the datum that it relates to.

- (e) `for (i in 1:n) lines(mcycle$times, A[, i])`

The kernels all have rather similar typical widths: the heights vary according to the number of data within that width. If many data are making a substantial contribution to the weighted sum that defines the fitted value, then the weights must be lower than if fewer data contribute.

- (f) 

```
par(mfrow=c(2, 2))
mcycle$y<-mcycle$accel*0;mcycle$y[65] <- 1
for (k in 1:4) plot(mcycle$times, fitted(
 gam(y~s(times, k=40), data=mcycle, sp=mc$sp*10^(k-1.5)))
, type="l", ylab="A[65,]", ylim=c(-0.01, 0.12))
```

Low smoothing parameters lead to narrow, high kernels, while high smoothing parameters result in wide low kernels.

4.(a) 

```
w <- c(rep(400,20),rep(1,113))
m <- 40;par(mfrow=c(1,1))
sp <- seq(-13,12,length=m) ## trial log(sp)'s
AC1 <- EDF <- rep(0,m)
for (i in 1:m) { ## loop through s.p.'s
 b <- gam(accel~s(times,k=40),data=mcycle,weights=w,
 sp=exp(sp[i]))
 EDF[i] <- sum(b$edf)
 AC1[i] <- acf(residuals(b),plot=FALSE)$acf[2]
}
plot(EDF,AC1,type="l");abline(0,0,col=2)
```

So the lag 1 residual autocorrelation starts positive declines to zero around the GCV best fit model, and then becomes increasingly negative.

- (b) At low EDF the model oversmooths and fails to capture the mean of the data. This will lead to clear patterns in the mean of the residuals against time: the ACF picks this residual trend up as positive autocorrelation.
- (c) i. So the  $j^{\text{th}}$  row of the  $n \times n$  matrix  $\mathbf{A}$  is  $j - (k + 1)/2$  zeroes, followed by  $k$  values,  $1/k$ , followed by  $n - j - (k - 1)/2$  further zeroes. i.e. something like:

$$\begin{bmatrix} 0 & \dots & 0 & 1/k & 1/k & \dots & 1/k & 0 & \dots & 0 \end{bmatrix}$$

- ii. Using  $\mathbf{V}_{\hat{\epsilon}}/\sigma^2 = \mathbf{I} - 2\mathbf{A} + \mathbf{A}\mathbf{A}$  and slogging through, it turns out that the leading diagonal elements (in the interior) are  $\sigma^2(k - 1)/k$ , while the elements on the sub and super diagonal (the lag 1 covariances) are  $-\sigma^2(k + 1)/k^2$ .

- iii. The correlation at lag 1 is clearly

$$-\sqrt{\frac{k+1}{k(k-1)}}$$

So the residual autocorrelation is always negative, with magnitude decreasing as  $k$  increases. Once  $k$  is small enough, oversmoothing is avoided, so that the observed ACF reflects this residual autocorrelation, rather than the inadequately modelled trend. It is this increasingly negative auto-correlation that was seen in the `mcycle` example in part (a).

It is tempting to view negative autocorrelation in the residuals as an indication of overfitting, but the preceding analysis indicates that some care would be needed to do this, since the true residual auto-correlation (at lag 1) should always be negative.

5.(a) 

```
attach(co2s)
plot(c.month,co2,type="l")
```

(b) 

```
b<-gam(co2~s(c.month,k=300,bs="cr"))
```

(c) 

```
pd <- data.frame(c.month=1:(n+36))
fv <- predict(b,pd,se=TRUE)
plot(pd$c.month,fv$fit,type="l")
lines(pd$c.month,fv$fit+2*fv$se,col=2)
lines(pd$c.month,fv$fit-2*fv$se,col=2)
```

The prediction of smoothly and sharply decreasing CO<sub>2</sub> is completely out of line with the long term pattern in the data, and is driven entirely by the seasonal dip at the end of the data.

(d) 

```
b2 <- gam(co2~s(month,bs="cc") + s(c.month,bs="cr",k=300),
 knots=list(month=seq(1,13,length=10)))
```

Notice how the `knots` argument has been used to ensure that the smooth of month wraps round correctly: month 1 should be the same as month 13 (if it ever occurred), not month 12.

(e) 

```
pd2 <- data.frame(c.month=1:(n+36),
 month=rep(1:12,length.out=n+36))
fv <- predict(b2,pd2,se=TRUE)
plot(pd$c.month,fv$fit,type="l")
lines(pd$c.month,fv$fit+2*fv$se,col=2)
lines(pd$c.month,fv$fit-2*fv$se,col=2)
```

The predictions now look *much* more credible, since we now extrapolate the long terms trend, and simply repeat the seasonal cycle on top of it. However, it is worth noting that the smooth of cumulative month (the long term trend) is still estimated to have rather high effective degrees of freedom, and it does still wiggle a lot, suggesting that extrapolation is still a fairly dangerous thing to do, and we had better not rely on it very far into the future.

Note that an alternative way of extrapolating is to add knots beyond the range of the observed data. e.g. with the argument

```
knots=list(...,c.month=seq(1,n+36,length=300))
```

Since the function value at these extra knots can only very subtly alter the shape of the function in the range of the data, the resulting curves tend to have very high associated standard errors: this is probably realistic!

6. There is no unique ‘right’ answer to this, but here is an outline of how to get started. I looked at `ir` and `dp` lagged for 1 to 4 months. Everything suggested dropping the effect of `ir` at lag 4 from the model, but everything else marginally improved the model, so was left in. Here is the code.

```
n<-nrow(ipo)
create lagged variables ...
ipo$ir1 <- c(NA,ipo$ir[1:(n-1)])
ipo$ir2 <- c(NA,NA,ipo$ir[1:(n-2)])
ipo$ir3 <- c(NA,NA,NA,ipo$ir[1:(n-3)])
ipo$ir4 <- c(NA,NA,NA,NA,ipo$ir[1:(n-4)])
ipo$dp1 <- c(NA,ipo$dp[1:(n-1)])
ipo$dp2 <- c(NA,NA,ipo$dp[1:(n-2)])
ipo$dp3 <- c(NA,NA,NA,ipo$dp[1:(n-3)])
ipo$dp4 <- c(NA,NA,NA,NA,ipo$dp[1:(n-4)])
fit initial model and look at it ...
b<-gam(n.ip0~s(ir1)+s(ir2)+s(ir3)+s(ir4)+s(log(reg.t))+s(dp1)+s(dp2)+s(dp3)+s(dp4)+s(month,bs="cc") + s(t,k=20),
 data=ipo,knots=list(month=seq(1,13,length=10)),
 family=poisson,gamma=1.4)
par(mfrow=c(3,4))
```

```

plot(b,scale=0)
summary(b)
re-fit model dropping ir4 ...
b1 <- gam(n.ipo~s(ir1)+s(ir2)+s(ir3)+s(log(reg.t))+s(dp1) +
 s(dp2)+s(dp3)+s(dp4)+s(month,bs="cc")+s(t,k=20),
 data=ipo,knots=list(month=seq(1,13,length=10)),
 family=poisson,gamma=1.4)
par(mfrow=c(3,4))
plot(b1,scale=0)
summary(b1)
residual checking ...
gam.check(b1)
acf(residuals(b1))

```

The final model has good residual plots that largely eliminate the auto-correlation. In the above the degrees of freedom for  $t$  have been kept fairly low, in order to try and force the model to use the other covariates in preference to  $t$ , and because too much freedom for the smooth of time tends to lead to it being used to the point of overfitting (lag 1 residual auto-correlation becomes very negative, for example). The most noticeable feature of the plotted smooth effects is the way that IR at all significant lags tends to be associated with an increase in IPO volume up to around 20%, after which there is a decline. If this reflects company behaviour, it certainly makes sense. If IR averages are too low then there is a danger of investors not being interested in IPOs, while excessively high IRs suggest that companies are currently being undervalued excessively, so that unreasonably small amounts of capital will be raised by the IPO. The other plots are harder to interpret!

7. The following is the best model I found.

```

wm<-gam(price~s(h.rain)+s(s.temp)+s(h.temp)+s(year),
 data=wine,family=Gamma(link=identity),gamma=1.4)
plot(wm,pages=1,residuals=TRUE,pch=1,scale=0)
acf(residuals(wm))
gam.check(wm)
predict(wm,wine,se=TRUE)

```

A Gamma family seems to produce acceptable residual plots. `w.temp` appears to be redundant. Two way interactions between the weather variables only make the GCV and AIC scores worse. `gamma=1.4` is a prudent defence against overfitting, given that the sample size is so small. The effects are easy to interpret: (i) more recent vintages are worth less than older ones; (ii) low harvest rainfall and high summer temperatures are both associated with higher prices; (iii) there is some suggestion of an optimum harvest temperature at 17.5C, but the possible increases at very low and very high harvest temperatures make interpretation difficult (however this harvest effect substantially increases the proportion deviance explained).

8. `smooth.construct.tr.smooth.spec<-function(object,data,knots)`  
`## a truncated power spline constructor method function`  
`## object$p.order = null space dimension`  
`{ m <- object$p.order`

```

if (m<1) stop("silly m supplied")
nk<-object$bs.dim-m-1 # number of knots
if (nk<=0) stop("k too small for m")
x <- get.var(object$term,data) # find the data
x.shift <- mean(x) # shift used to enhance stability
if (!is.null(knots)) # are there supplied knots?
k <- get.var(object$term,knots)
else k<-NULL
if (is.null(k)) # space knots through data
{ n<-length(x)
 k<-quantile(x[2:(n-1)],seq(0,1,length=nk+2))[2:(nk+1)]
}
if (length(k)!=nk) # right number of knot?
stop(paste("there should be ",nk," supplied knots"))
x <- x - x.shift # basis stabilizing shift
k <- k - x.shift # knots treated the same!
X<-matrix(0,length(x),object$bs.dim)
for (i in 1:(m+1)) X[,i] <- x^(i-1)
for (i in 1:nk) X[,i+m+1]<- (x-k[i])^m*as.numeric(x>k[i])
object$X<-X # the finished model matrix
if (!object$fixed) # create the penalty matrix
{ object$S[[1]]<-diag(c(rep(0,m+1),rep(1,nk)))
}
object$rank<-nk # penalty rank
object>null.space.dim <- m+1 # dim. of unpenalized space
store "tr" specific stuff ...
object$knots<-k;object$m<-m;object$x.shift <- x.shift
get the centering constraint ...
object$C<-matrix(colSums(object$X),1,object$bs.dim)
object$df<-ncol(object$X)-1 # maximum DoF
if (object$by!="NA") # deal with "by" variable
{ by <- get.var(object$by,data) # find by variable
 if (is.null(by)) stop("Can't find by variable")
 object$X<-by*object$X # form diag(by)%*%X
}
class(object)<-"tr.smooth" # Give object a class
object
}

Predict.matrix.tr.smooth<-function(object,data)
prediction method function for the 'tr' smooth class
{ x <- get.var(object$term,data)
 x <- x - object$x.shift # stabilizing shift
 m <- object$m; # spline order (3=cubic)
 k<-object$knots # knot locations
 nk<-length(k) # number of knots
 X<-matrix(0,length(x),object$bs.dim)
 for (i in 1:(m+1)) X[,i] <- x^(i-1)
 for (i in 1:nk) X[,i+m+1] <- (x-k[i])^m*as.numeric(x>k[i])
}

```

```

 X # return the prediction matrix
}

Now run first data simulation in ?gam

Fit simulated data using the new class ...
m <- 2
b<-gam(y~s(x0,bs="tr",m=m)+s(x1,bs="tr",m=m) +
 s(x2,bs="tr",m=m)+s(x3,bs="tr",m=m))
plot(b, pages=1)

9.(a) plot(bfday, bfpop, type="l")
(b) ## prepare differenced and lagged data ...
bf$dn <- c(NA, bf$pop[2:n]-bf$pop[1:(n-1)])
lag <- 6
bf$n.lag <- c(rep(NA,lag),bf$pop[1:(n-lag)])
bf1 <- bf[(lag+1):n,] # strip out NAs, for convenience
fit model, note no intercept ...
b<-gam(dn~n.lag+pop+s(log(n.lag),by=n.lag) +
 s(log(pop),by=-pop)-1,data=bf1)
plot(b, pages=1, scale=-1, se=F) ## effects
plot(abs(fitted(b)), residuals(b))
acf(residuals(b))

So the per capita birth rate declines with increasing population, which is sensible, given likely competition for resources. The mortality rate increases with population, which is also plausible, although the decline at very high densities is surprising, and may not be real. Note that both functions are negative in places: this is biologically nonsensical. The residual plots are not brilliant, and there is residual auto-correlation at lag 1, so the model is clearly not great.

(c) fv <- bf$pop
e <- rnorm(n)*0 ## increase multiplier for noisy version
min.pop <- min(bf$pop); max.pop <- max(bf$pop)
for (i in (lag+1):n) { ## iteration loop
 dn <- predict(b, data.frame(n.lag=fv[i-lag], pop=fv[i-1]))
 fv[i] <- fv[i-1]+dn + e[i];
 fv[i]<-min(max.pop, max(min.pop, fv[i]))
}
plot(bf$day, fv, type="l")

```

The amplitude without noise is rather low, but does improve with noise, however the feature that cycles tend to be smoother at the trough and noisier at the peak is not re-captured, suggesting that something more complicated than a constant variance model may be needed. Ideally we would fit the model with constraints forcing the functions to be positive, and possibly monotonic: this is possible, but more complicated (see `?pcl.s`).

- (d) Census data like these are rather unusual. In most cases the model has to deal with measurement error as well, which rather invalidates the approach used in this question.

10.(a) `pairs(chl)`

Notice, in particular, that there is fairly good coverage of the predictors `bath` and `jul.day`. Plotting histograms of individual predictors indicates very skewed distributions for `chl.sw` and `bath`: raising these to the power of .4 and .25 respectively largely eliminates this problem.

```
(b) fam <- quasi(link=log,var=mu^2)
cm <- gam(chl ~ s(I(chl.sw^.4),bs="cr",k=20) +
 s(I(bath^.25),bs="cr",k=60)+s(jul.day,bs="cr",k=20),
 data=chl,family=fam,gamma=1.4)
gam.check(cm)
summary(cm)
```

The given `quasi` family seems to deal nicely with the mean variance relationship, and a log link is required to linearize the model.

```
(c) ## create fit and validation sets ...
set.seed(2)
n<-nrow(chl);nf <- floor(n*.9)
ind <- sample(1:n,nf,replace=FALSE)
chl1 <- chl[ind,];chl2 <- chl[-ind,]
fit to the fit set
smf<-gam(chl ~ s(I(chl.sw^.4),bs="cr",k=20) +
 s(I(bath^.25),bs="cr",k=60)+s(jul.day,bs="cr",k=20),
 data=chl1,family=fam,gamma=1.4)
evaluate prop. dev. explained for validation set
y <- chl2$chl;w <- y*0+1
mu <- predict(smf,chl2,type="response")
pred.dev <- sum(fam$dev.resids(y,mu,w))
null.dev <- sum(fam$dev.resids(y,mean(y),w))
1-pred.dev/null.dev # prop dev. explained
```

I got proportion deviance explained of about 46% for the fitted models and for predicting the validation set, suggesting that the model is not overfitting, and does provide a useful improvement on the raw satellite data. More sophisticated models based on tensor product smooths can be used to improve matters further. For practical use of these models, it is always important to look at their predictions over the whole spatial arena of interest, at a number of times through the year, to check for artefacts (for example from extrapolating outside the observed predictor space).

11. Based on `?gam` here is a function to simulate data.

```
sim.data <- function(n=400,sig=2) {
 x0 <- runif(n, 0, 1);x1 <- runif(n, 0, 1)
 x2 <- runif(n, 0, 1);x3 <- runif(n, 0, 1)
 f0 <- function(x) 2 * sin(pi * x)
 f1 <- function(x) exp(2 * x)
 f2<-function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
 f3 <- function(x) 0*x
 f <- f0(x0) + f1(x1) + f2(x2)
 e <- rnorm(n, 0, sig)
 y <- f + e}
```

```

 return(data.frame(y=y, f=f, x0=x0, x1=x1, x2=x2, x3=x3))
}

```

Note that  $x_3$  has no effect on the response, while  $x_0$  has the weakest effect of the significant predictors.

```

(a) reps <- 200
 n.func<-4 # which term to examine
 pv <- rep(0,reps) # array for observed p-values
 for (i in 1:reps) {
 dat <- sim.data(n=200,sig=2)
 b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3,fx=FALSE),data=dat)
 pv[i] <- summary(b)$s.pv[n.func]
 }
 pv <- sort(pv)
 true <- (1:reps-.5)/reps
 plot(true,pv);abline(0,1,col=2)

```

The resulting QQ-plot should look like a random scatter around the plotted straight line, if the p-values are really from a uniform distribution. In this case the points lie on a curve a little below the line: the p-values tend to be a little too small — they reject the null too easily. So if a p-value says that a term is not significant, it probably isn't, but significance is a little harder to be sure about.

If the code is re-run with  $fx=TRUE$  then the p-values behave as they should.

(b) The answer to (a) suggests basing hypothesis tests on un-penalized terms, in order to get the p-values right. This will mean that all terms have rather wiggly estimates, which might be expected to result in low power to reject a false null (of no effect). This can be investigated by slight modification of the code given for part (a). Set  $n.func<-1$ , in order to look at p-values for the first term, which should be in the model, but only has a weak effect. Then compare what proportion of p-values are less than, say, 0.05, (i) when the smoothing parameter for the first term is estimated, and (ii) when the first term is replaced by  $s(x0, fx=TRUE)$ , so that it is unpenalized. When I did this (i) gave 90% of p-values less than 0.05, while (ii) gave 80%. Hence there is some reduction in power, but this is perhaps modest given the better behaviour of the p-values without penalization.

## B.6 Chapter 6

1.(a)

$$y_{ij} = \alpha + b_i + \epsilon_{ij}, \quad b_i \sim N(0, \sigma_b^2) \quad \epsilon_{ij} \sim N(0, \sigma^2),$$

where  $y_{ij}$  is the weight of the  $j^{th}$  piglet in the  $i^{th}$  litter and the random variables  $b_i$  and  $\epsilon_{ij}$  are all mutually independent.

- (b)  $H_0 : \sigma_b^2 = 0$  against  $H_1 : \sigma_b^2 > 0$ . i.e. testing whether or not there is a maternal component of piglet weight variability.
- (c) There is very strong evidence ( $p \approx 10^{-6}$ ) that  $\sigma_b^2 > 0$ . i.e. very strong evidence for a maternal component to piglet weight variability.

(d)

$$\hat{\sigma}_b^2 = \hat{\sigma}_0^2 - \hat{\sigma}_1^2/5 = 0.592$$

2.(a)

$$\begin{aligned}\bar{y}_{i\cdot} &= \frac{1}{J} \sum_{j=1}^J y_{ij} = \alpha + b_i + \sum_{j=1}^J c_j + \frac{1}{J} \sum_{j=1}^J \epsilon_{ij} \\ &= a + e_i\end{aligned}$$

where  $a = \alpha + \sum_j c_j$  and  $e_i = b_i + \sum_j \epsilon_{ij}/J$ . The  $e_i$ 's are obviously independent since the  $b_i$ 's and  $\epsilon_{ij}$ 's are mutually independent and no two  $e_i$ 's share a  $b_i$  or  $\epsilon_{ij}$ . Normality of the  $e_i$ 's follows immediately from normality of the  $b_i$ 's and  $\epsilon_{ij}$ 's. Furthermore,

$$\mathbb{E}(e_i) = \mathbb{E}(b_i) + \frac{1}{J} \sum_{j=1}^J \mathbb{E}(\epsilon_{ij}) = 0$$

and (using independence of the terms)

$$\text{var}(e_i) = \text{var}(b_i) + \frac{1}{J^2} \sum_{j=1}^J \text{var}(\epsilon_{ij}) = \sigma_b^2 + \sigma^2/J.$$

So  $\sigma_b^2 + \sigma^2/J$  can be estimated from the residual sum of squares,  $RSS_1$ , of the least squares fit of the aggregated model to the  $\bar{y}_{i\cdot}$  data:

$$\hat{\sigma}_b^2 + \hat{\sigma}^2/J = RSS_1/(I-1)$$

$$\Rightarrow \hat{\sigma}_b^2 = RSS_1/(I-1) - \hat{\sigma}^2/J$$

where  $\hat{\sigma}^2$  is given in the question.

(b) Calculations tediously similar to part (a), culminate in:

$$\hat{\sigma}_c^2 = RSS_2/(J-1) - \hat{\sigma}^2/I$$

where  $RSS_2$  is the residual sum of squares for the 2<sup>nd</sup> aggregated model fit to the  $\bar{y}_{\cdot j}$  data.

3.(a)  $H_0 : \sigma_a^2 = 0$  would be tested by ANOVA comparison of the fit of the given full model, to the fit of the simplified model implied by  $H_0$ ,

$$y_{ij} = \mu + \beta x_{ij} + \epsilon_{ij}, \quad \epsilon_{ij} \sim N(0, \sigma^2).$$

$H_0 : \beta = 0$  would be tested by comparing the fit of the full model with the fit of the reduced model implied by this null,

$$y_{ij} = \mu + a_i + \epsilon_{ij}, \quad a_i \sim N(0, \sigma_a^2) \text{ and } \epsilon_{ij} \sim N(0, \sigma^2),$$

using ANOVA, in the same manner as if both were fixed effect linear models.

(b) To estimate  $\sigma_a^2$  and  $\beta$  the data would be averaged at each level of the random

effect  $a_i$  to yield

$$\begin{aligned}\bar{y}_{i \cdot} &= \mu + a_i + \beta \bar{x}_{i \cdot} + \frac{1}{J} \sum_{j=1}^J \epsilon_{ij} \\ &= \mu + \beta \bar{x}_{i \cdot} + e_i\end{aligned}$$

where  $e_i \sim N(0, \sigma_a^2 + \sigma^2/J)$ . The least squares fit of this model can be used to estimate  $\beta$  (the full model fit can not be used, since  $\hat{\beta}$  will not be independent of the  $a_i$ ). If  $RSS_1$  and  $RSS_0$  are the residual sums of squares for fitting the full and reduced models respectively to the  $y_{ij}$  and  $\bar{y}_{i \cdot}$  data, then the obvious estimator of  $\sigma_a^2$  is:

$$\hat{\sigma}_a^2 = \frac{RSS_0}{I-2} - \frac{RSS_1}{J(IJ-I-1)}.$$

4. (In this question the fixed effects model matrices resulting from any other valid identifiability constraints on the fixed effects are fine, of course.)

(a)

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_{31} \\ y_{32} \\ y_{41} \\ y_{42} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{21} \\ 1 & x_{22} \\ 1 & x_{31} \\ 1 & x_{32} \\ 1 & x_{41} \\ 1 & x_{42} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \\ \epsilon_{41} \\ \epsilon_{42} \end{bmatrix}$$

with

$$\psi = \mathbf{I}_4 \sigma_a^2$$

(b)

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_{31} \\ y_{32} \\ y_{41} \\ y_{42} \\ y_{51} \\ y_{52} \\ y_{61} \\ y_{62} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \\ \epsilon_{41} \\ \epsilon_{42} \\ \epsilon_{51} \\ \epsilon_{52} \\ \epsilon_{61} \\ \epsilon_{62} \end{bmatrix}$$

with

$$\psi = \mathbf{I}_6 \sigma_b^2$$

(c)

$$\begin{bmatrix} y_{111} \\ y_{112} \\ y_{113} \\ y_{121} \\ y_{122} \\ y_{123} \\ y_{131} \\ y_{132} \\ y_{133} \\ y_{211} \\ y_{212} \\ y_{213} \\ y_{221} \\ y_{222} \\ y_{223} \\ y_{231} \\ y_{232} \\ y_{233} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \mu \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ (\alpha b)_{11} \\ (\alpha b)_{12} \\ (\alpha b)_{13} \\ (\alpha b)_{21} \\ (\alpha b)_{22} \\ (\alpha b)_{23} \end{bmatrix} + \begin{bmatrix} \epsilon_{111} \\ \epsilon_{112} \\ \epsilon_{113} \\ \epsilon_{121} \\ \epsilon_{122} \\ \epsilon_{123} \\ \epsilon_{131} \\ \epsilon_{132} \\ \epsilon_{133} \\ \epsilon_{211} \\ \epsilon_{212} \\ \epsilon_{213} \\ \epsilon_{221} \\ \epsilon_{222} \\ \epsilon_{223} \\ \epsilon_{231} \\ \epsilon_{232} \\ \epsilon_{233} \end{bmatrix}$$

$$\psi = \begin{bmatrix} \mathbf{I}_3\sigma_b^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_6\sigma_{ab}^2 \end{bmatrix}$$

5.(a)

$$\begin{aligned} \Sigma_{x+z} &= \mathbb{E}[(\mathbf{X} + \mathbf{Z} - \boldsymbol{\mu}_x - \boldsymbol{\mu}_z)(\mathbf{X} + \mathbf{Z} - \boldsymbol{\mu}_x - \boldsymbol{\mu}_z)^T] \\ &= \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_x)(\mathbf{X} - \boldsymbol{\mu}_x)^T] + \mathbb{E}[(\mathbf{Z} - \boldsymbol{\mu}_z)(\mathbf{Z} - \boldsymbol{\mu}_z)^T] \\ &\quad + \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_x)(\mathbf{Z} - \boldsymbol{\mu}_z)^T] + \mathbb{E}[(\mathbf{Z} - \boldsymbol{\mu}_z)(\mathbf{Z} - \boldsymbol{\mu}_z)^T] \\ &= \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_x)(\mathbf{X} - \boldsymbol{\mu}_x)^T] + \mathbb{E}[(\mathbf{Z} - \boldsymbol{\mu}_z)(\mathbf{Z} - \boldsymbol{\mu}_z)^T] \text{ (by ind.)} \\ &= \Sigma_x + \Sigma_z \end{aligned}$$

(b) i.

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{13} \\ 1 & x_{21} \\ 1 & x_{22} \\ 1 & x_{23} \\ 1 & x_{31} \\ 1 & x_{32} \\ 1 & x_{33} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{31} \\ \epsilon_{32} \\ \epsilon_{33} \end{bmatrix}$$

where  $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$  and  $\epsilon \sim N(\mathbf{0}, \mathbf{I}\sigma^2)$ .

ii. The covariance matrix of  $\mathbf{y}$  is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \sigma_b^2 + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \sigma^2$$

which is

$$\begin{bmatrix} \sigma^2 + \sigma_b^2 & \sigma_b^2 & \sigma_b^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_b^2 & \sigma^2 + \sigma_b^2 & \sigma_b^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_b^2 & \sigma_b^2 & \sigma^2 + \sigma_b^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 + \sigma_b^2 & \sigma_b^2 & \sigma_b^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_b^2 & \sigma^2 + \sigma_b^2 & \sigma_b^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_b^2 & \sigma_b^2 & \sigma^2 + \sigma_b^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 + \sigma_b^2 & \sigma_b^2 & \sigma_b^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_b^2 & \sigma^2 + \sigma_b^2 & \sigma_b^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_b^2 & \sigma_b^2 & \sigma^2 + \sigma_b^2 \end{bmatrix}$$

6.(a) Site and source should be fixed; Lot and Wafer should be random.

(b) The thickness for wafer  $l$ , lot  $k$ , site  $j$  and source  $i$  is

$$y_{ijkl} = \mu + \alpha_i + \beta_j + b_k + c_{(k)l} + \epsilon_{ijkl}$$

where  $b_k \sim N(0, \sigma_b^2)$ ,  $c_{(k)l} \sim N(0, \sigma_c^2)$ ,  $\epsilon_{ijkl} \sim N(0, \sigma^2)$  and all these r.v.s are independent. The subscript,  $(k)l$ , is used to emphasize that wafer is nested within lot to help clarify the practical fitting of the model (I haven't indicated all such nestings this way!).

(c)

```

> options(contrasts=c("contr.treatment",
+ "contr.treatment"))
> m1 <- lme(Thickness~Site+Source,Oxide,~1|Lot/Wafer)
> plot(m1) # check resids vs. fitted vals
> qqnorm(residuals(m1)) # check resids for normality
> abline(0,sd(resid(m1)))# adding a "reference line"
> qqnorm(m1,~ranef(..,level=1)) # check normality of b_k
> qqnorm(m1,~ranef(..,level=2)) # check normality of c_(k)l
> m2 <- lme(Thickness~Site+Source,Oxide,~1|Lot)
> anova(m1,m2)

```

| Model | df | AIC | BIC    | logLik | Test    | L.Ratio | p-value        |
|-------|----|-----|--------|--------|---------|---------|----------------|
| m1    | 1  | 7   | 455.76 | 471.30 | -220.88 |         |                |
| m2    | 2  | 6   | 489.41 | 502.72 | -238.70 | 1 vs 2  | 35.6444 <.0001 |

The checking plots suggest no problems at all in this case. m1 is the full model fit and m2 is a reduced version, with no  $c_{(k)l}$  terms. The anova(m1, m2) command performs a generalized likelihood ratio test of the hypothesis that the data were generated by m2 against the alternative that they were generated by m1. The p value is so low in this case that, despite the problems with GLRT tests in this context, we can be very confident in rejecting  $H_0$  and concluding that m1 is necessary. i.e. there is evidence for wafer to wafer variability above what is explicable by lot to lot variability.

```
> anova(m1)
numDF denDF F-value p-value
(Intercept) 1 46 240197.01 <.0001
Site 2 46 0.60 0.5508
Source 1 6 1.53 0.2629
```

There appears to be no evidence for site or source effects, and hence no need for follow up multiple comparisons. Note that for unbalanced data you would usually refit without the redundant fixed effects at this point, in order to gain a little precision, but these data are balanced, so doing so would make no difference.

```
> intervals(m1)
Approximate 95% confidence intervals

Fixed effects:
 lower est. upper
(Intercept) 1983.237346 1994.9166667 2006.595987
Site2 -2.327301 -0.2500000 1.827301
Site3 -1.243967 0.8333333 2.910634
Source2 -9.888949 10.0833333 30.055615
attr(",label")
[1] "Fixed effects"

Random Effects:
 Level: Lot
 lower est. upper
sd((Intercept)) 5.831891 10.94954 20.55808
 Level: Wafer
 lower est. upper
sd((Intercept)) 4.057118 5.982933 8.822887

Within-group standard error:
 lower est. upper
2.914196 3.574940 4.385495
```

So the lot to lot variation is substantial, with a standard deviation of between 5.8 and 20.6, while the wafer to wafer variation, within a lot, is not much

less, with standard deviation between 4.1 and 8.8. The unaccounted for ‘within wafer’ variability seems to be a little less, having a standard deviation somewhere between 2.9 and 4.4. There is no evidence that site on the wafer or source have any influence on thickness.

```

7. library(nlme)
attach(Machines)
interaction.plot(Machine, Worker, score) # note 6B
base model
m1<-lme(score~Machine, Machines, ~1|Worker/Machine)
check it...
plot(m1)
plot(m1, Machine~resid(.), abline=0)
plot(m1, Worker~resid(.), abline=0)
qqnorm(m1, ~resid(.))
qqnorm(m1, ~ranef(., level=1))
qqnorm(m1, ~ranef(., level=2)) ## note outlier
try more general r.e. structure
m2<-lme(score~Machine, Machines, ~Machine|Worker)
check it...
qqnorm(m2, ~resid(.))
qqnorm(m2, ~ranef(., level=1)) ## still an outlier
simplified model...
m0 <- lme(score~Machine, Machines, ~1|Worker)
formal comparison
anova(m0, m1, m2) ## m1 most appropriate
anova(m1) ## significant Machine effect
intervals(m1) ## Machines B and C better than A
remove problematic worker 6, machine B
Machines <- Machines[-(34:36),]
re-running improves plots, but conclusions same.

```

It seems that (6.6) is the most appropriate model of those tried, and broadly the same conclusions are reached with or without Worker 6, on Machine B, which causes outliers on several checking plots. See next question for comparison of machines B and C.

#### 8. Using the data without worker 6 machine B:

```

> intervals(m1, level=1-0.05/3, which="fixed")
Approximate 98.333333333333% confidence intervals

```

```

Fixed effects:
 lower est. upper
(Intercept) 48.138854 52.35556 56.57226
MachineB 6.682296 10.40277 14.12325
MachineC 10.436430 13.91667 17.39690
attr(", "label")
[1] "Fixed effects:"
> levels(Machines$Machine)
[1] "A" "B" "C"
> Machines$Machine<-relevel(Machines$Machine, "B")

```

```
> mla<-lme(score~Machine, Machines, ~1|Worker/Machine)
> intervals(mla, level=1-0.05/3, which="fixed")
Approximate 98.333333333333% confidence intervals

Fixed effects:
 lower est. upper
(Intercept) 58.3930615 62.758326 67.123591
MachineA -14.1232461 -10.402771 -6.682296
MachineC -0.2065794 3.513896 7.234371
attr(),"label")
[1] "Fixed effects:"
```

So, there is evidence for differences between machine A and the other 2, but not between B and C, at the 5% level. However, depending on how economically significant the point estimate of the B-C difference is, it might be worth conducting a study with more workers in order to check whether the possible small difference might actually be real.

- Physique and method would be modelled as fixed effects: the estimation of these effects is of direct interest and we would expect to obtain similar estimates in a replicate experiment; the effect of these factors should be fixed properties of the population of interest. Team would be modelled as a random effect — team to team variation is to be expected, but the individual team effects are not of direct interest and would be completely different if the experiment were repeated with different gunners.
- If  $y_{ijk}$  denotes the number of rounds for team  $i$ , which physique  $j$  using method  $k$  then a possible model is

$$y_{ijk} = \mu + \alpha_j + \beta_k + b_i + \epsilon_{ijk}$$

where the  $b_i$  are i.i.d.  $N(0, \sigma_b^2)$  and the  $\epsilon_{ijk}$  are i.i.d.  $N(0, \sigma^2)$ . A more careful notation might make explicit the nesting of team within build by, for example, replacing  $b_i$  by  $b_{i(j)}$ . An interaction between physique and method might also be considered (as might a random two way interaction of method and team).

(c)

```
> library(nlme); data(Gun)
> options(contrasts=c("contr.treatment", "contr.treatment"))
> with(Gun, plot(Method, rounds))
> with(Gun, plot(Physique, rounds))
> m1 <- lme(rounds~Method+Physique, Gun, ~1|Team)
> plot(m1) # fitted vs. resid plot
> qqnorm(residuals(m1))
> abline(0,m1$sigma) # add line of "perfect Normality"
> anova(m1) # balanced data: don't need type="marginal"
 numDF denDF F-value p-value
(Intercept) 1 26 2056.5315 <.0001
Method 1 26 316.8426 <.0001
Physique 2 6 1.2266 0.3576
> m2 <- lme(rounds~Method, Gun, ~1|Team)
> intervals(m2)
Approximate 95% confidence intervals
```

```

Fixed effects:
 lower est. upper
(Intercept) 22.562754 23.588889 24.615024
MethodM2 -9.493961 -8.511111 -7.528261
attr(),"label")
[1] "Fixed effects:"
```

Random Effects:

| Level:          | Team | lower     | est.     | upper    |
|-----------------|------|-----------|----------|----------|
| sd((Intercept)) |      | 0.5437598 | 1.101839 | 2.232693 |

Within-group standard error:

| lower    | est.     | upper    |
|----------|----------|----------|
| 1.093067 | 1.434448 | 1.882447 |

The residual plots appear quite reasonable in this case (although if you check the random effects themselves these are not so good). Since there is no evidence for an effect of Physique the only sensible follow up comparison would compare methods: method one appears to lead to an average increase in firing rate of between 7.5 and 9.5 rounds per minute relative to method 2 (with 95% confidence). By contrast the team to team variability and within team variability are rather low with 95% CIs on their standard deviations of (0.5,2.2) and (1.1,1.9) respectively.

- 10.(a) 

```
library(nlme)
attach(Soybean)
lmc <- lmeControl(niterEM=300) ## needed for convergence
m1<-lme(weight~Variety*Time+Variety*I(Time^2) +
 Variety*I(Time^3),Soybean,~Time|Plot,control=lmc)
plot(m1) ## clear increasing variance with mean
```
- (b) 

```
library(MASS)
m2<-glmmPQL(weight~Variety*Time+Variety*I(Time^2) +
 Variety*I(Time^3),data=Soybean,random=~Time|Plot,
 family=Gamma(link=log),control=lmc)
plot(m2) ## much better
```
- (c) 

```
m0<-glmmPQL(weight~Variety*Time+Variety*I(Time^2) +
 Variety*I(Time^3),data=Soybean,random=~1|Plot,
 family=Gamma(link=log),control=lmc) # simpler r.e.'s
m3<-glmmPQL(weight~Variety*Time+Variety*I(Time^2) +
 Variety*I(Time^3),data=Soybean,random=~Time+
 I(Time^2)|Plot,family=Gamma(link=log),control=lmc)
... m3 has more complex r.e. structure
Following not strictly valid, but gives a rough
guide. Suggests m2 is best...
AIC(m0,m2,m3)
summary(m2) ## drop Variety:Time
m4<-glmmPQL(weight~Variety+Time+Variety*I(Time^2) +
 Variety*I(Time^3),data=Soybean,random=~Time|Plot,
```

```

family=Gamma(link=log),control=lmc)
summary(m4) ## perhaps drop Variety:I(Time^3)?
m5<-glmmPQL(weight~Variety+Time+Variety*I(Time^2)+

 I(Time^3),data=Soybean,random=~Time|Plot,

 family=Gamma(link=log),control=lmc)
summary(m5) ## don't drop any more
AIC(m2,m4,m5) ## supports m4
intervals(m5,which="fixed")

```

So m4 or m5 are probably the best models to use, and both suggest that variety P has a higher weight on average.

11.(a) 

```
g1<-gamm(weight ~ Variety + s(Time) +

 s(Time,by=as.numeric(Variety=="P")),data=Soybean,

 family=Gamma(link=log), random=list(Plot=~Time))

plot(g1$lme) ## standard mean variance plot

par(mfrow=c(1,3))

plot(g1$gam,residuals=TRUE,all.terms=TRUE) ## gam plot
```

The residual plots look fine. It seems that variety P increases its weight a little more slowly than variety F (don't forget that this is on the log scale).

(b) 

```
summary(g1$gam) ## evidence for variety dependence

could also do following

g2 <- gamm(weight~s(Time),family=Gamma(link=log),

 data=Soybean,random=list(Plot=~Time))

g3 <- gamm(weight~Variety+s(Time),family=Gamma(link=log),

 data=Soybean,random=list(Plot=~Time))

following only a rough guide, but also supports g1 ...

AIC(g1$lme,g2$lme,g3$lme)
```

The summary (in combination with the plotted effects) gives strong evidence that variety P gives higher weights, with the difference decreasing slightly with time. Fitting models without a smooth function of time as a correction for P, or without any effect of Variety both give model fits that seem worse than the original model fit according to the AIC of the working model at convergence of the PQL iterations, confirming the implications of the summary.

(c) If varieties F and P have only slightly different trajectories through time, but we use completely separate smooths for each, then both smooths will require a similar relatively large number of degrees of freedom in order to represent the time trajectory of each variety. On the other hand, if we model the trajectory of P as F's trajectory plus a correction, it is possible that this correction may be very smooth, so that a good fit can be achieved without using up as many degrees of freedom as would be needed for the same fit, using the completely separate smooths. This is in fact what happens for the Soybean data.

---

## Bibliography

---

- Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. New York: Wiley.
- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. Petran and F. Csaaki (Eds.), *International Symposium on Information Theory*, Akadeemiai Kiadó, Budapest, Hungary., pp. 267–281.
- Asseburg, C., S. Smout, J. Matthiopoulos, C. Fernández, S. Redpath, S. Thirgood, and J. Harwood (2005). The functional response of a generalist predator. *Preprint*.
- Augustin, N. H., D. L. Borchers, E. D. Clarke, S. T. Buckland, and M. Walsh (1998). Spatio-temporal modelling of annual egg production of fish stocks using generalized additive models. *Canadian Journal of Fisheries and Aquatic Science* 55, 2608–2621.
- Baker, R. R. and M. A. Bellis (1993). Human sperm competition: ejaculate adjustment by males and the function of masturbation. *Animal behaviour* 46, 861–885.
- Borchers, D. L., S. T. Buckland, I. G. Priede, and S. Ahmadi (1997). Improving the precision of the daily egg production method using generalized additive models. *Canadian Journal of Fisheries and Aquatic Science* 54, 2727–2742.
- Bowman, A. W. and A. Azzalini (1997). *Applied Smoothing Techniques for Data Analysis*. Oxford University Press.
- Breslow, N. E. and D. G. Clayton (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* 88, 9–25.
- Chambers, J. M. (1993). Linear models. In J. M. Chambers and T. J. Hastie (Eds.), *Statistical Models in S*, pp. 95–144. Chapman & Hall.
- Chambers, J. M. and T. J. Hastie (Eds.) (1993). *Statistical Models in S*. Chapman & Hall.
- Cox, D. R. and D. V. Hinkley (1974). *Theoretical Statistics*. Chapman & Hall.
- Craven, P. and G. Wahba (1979). Smoothing noisy data with spline functions. *Numerische Mathematik* 31, 377–403.
- Davison, A. C. (2003). *Statistical Models*. Cambridge, UK: Cambridge University Press.
- de Boor (1978). *A Practical Guide to Splines*. New York: Springer.
- Demmel, J. (1997). *Applied Numerical Linear Algebra*. Philadelphia: SIAM.
- Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Biometrika* 64, 511–526.

- plete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society, Series B* 39, 1–38.
- Dixon, C. E. (2003). *Multi-dimensional modelling of physiologically and temporally structured populations*. Ph. D. thesis, University of St Andrews.
- Dobson, A. J. (2001). *An Introduction to Generalized Linear Models* (2nd ed.). Chapman & Hall/CRC.
- Duchon, J. (1977). Splines minimizing rotation-invariant semi-norms in solobev spaces. In W. Schempp and K. Zeller (Eds.), *Construction Theory of Functions of Several Variables*, Berlin, pp. 85–100. Springer.
- Eilers, P. H. C. and B. D. Marx (1996). Flexible smoothing with B-splines and penalties. *Statistical Science* 11(2), 89–121.
- Eilers, P. H. C. and B. D. Marx (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and intelligent laboratory systems* 66, 159–174.
- Ellner, S. P., B. E. Kendall, S. N. Wood, E. McCauley, and C. J. Briggs (1997). Inferring mechanism from time-series data: delay differential equations. *Physica D* 110(3-4), 182–194.
- Fahrmeir, L., T. Kneib, and S. Lang (2004). Penalized structured additive regression for space-time data: a bayesian perspective. *Statistica Sinica* 14(3), 731–761.
- Fahrmeir, L. and S. Lang (2001). Bayesian inference for generalized additive mixed models based on markov random field priors. *Applied Statistics* 50, 201–220.
- Faraway, J. J. (2004). *Linear Models with R*. Chapman & Hall/CRC.
- Feller, W. (1957). *An Introduction to Probability Theory and its Applications* (2nd ed.), Volume 1. New York: Wiley.
- Freedman, W. L., B. F. Madore, B. K. Gibson, L. Ferrarese, D. D. Kelson, S. Sakai, J. R. Mould, R. C. Kennicutt, H. C. Ford, J. A. Graham, J. P. Huchra, S. M. Hughes, G. D. Illingworth, L. M. Macri, and P. B. Stetson (2001). Final results from the hubble space telescope key project to measure the hubble constant. *The Astrophysical Journal* (553), 47–72.
- Gill, P. E., W. Murray, and M. H. Wright (1981). *Practical Optimization*. London: Academic Press.
- Golub, G. H., M. Heath, and G. Wahba (1979). Generalized cross validation as a method for choosing a good ridge parameter. *Technometrics* 21(2), 215–223.
- Golub, G. H. and C. F. van Loan (1996). *Matrix Computations* (3rd ed.). Baltimore: Johns Hopkins University Press.
- Green, P. J. and B. W. Silverman (1994). *Nonparametric Regression and Generalized Linear Models*. Chapman & Hall.
- Gu, C. (1992). Cross-validating non-gaussian data. *Journal of Computational and Graphical Statistics* 1, 169–179.
- Gu, C. (2002). *Smoothing Spline ANOVA Models*. New York: Springer.

- Gu, C. and Y. J. Kim (2002). Penalized likelihood regression: general approximation and efficient approximation. *Canadian Journal of Statistics* 34(4), 619–628.
- Gu, C. and G. Wahba (1991). Minimizing gcv/gml scores with multiple smoothing parameters via the newton method. *SIAM Journal on Scientific and Statistical Computing* 12(2), 383–398.
- Hand, D. J., F. Daly, A. D. Lunn, K. J. McConway, and E. Ostrowski (1994). *A Handbook of Small Data Sets*. Chapman & Hall.
- Harville, D. A. (1976). Confidence intervals and sets for linear combinations of fixed and random effects. *Biometrics* 32(2), 403–407.
- Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association* 72(358), 320–340.
- Hastie, T. and R. Tibshirani (1986). Generalized additive models (with discussion). *Statistical Science* 1, 297–318.
- Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*. Chapman & Hall.
- Hastie, T. and R. Tibshirani (1993). Varying-coefficient models. *Journal of the Royal Statistical Society, Series B* 55(4), 757–796.
- Hastie, T. J. (1993). Generalized additive models. In J. Chambers and T. Hastie (Eds.), *Statistical Models in S*. London: Chapman & Hall.
- Horwood, J. (1993). The bristol channel sole (*solea solea* (L.)): A fisheries case study. *Advances in Marine Biology* 29, 215–367.
- Horwood, J. and M. G. Walker (1990). Determinacy of fecundity in sole (*solea solea*) from the bristol channel. *Journal of the Marine Biological Association of the United Kingdom* 70, 803–813.
- Kim, Y. J. and C. Gu (2004). Smoothing spline gaussian regression: more scalable computation via efficient approximation. *Journal of the Royal Statistical Society, Series B* 66, 337–356.
- Laird, N. M. and J. H. Ware (1982). Random-effects models for longitudinal data. *Biometrics* 38, 963–974.
- Lancaster, P. and K. Šalkauskas (1986). *Curve and Surface Fitting: an introduction*. London: Academic Press.
- Landau, S., I. C. Ellison-Wright, and E. T. Bullmore (2003). Tests for a difference in timing of physiological response between two brain regions measured by using functional magnetic resonance imaging. *Applied Statistics* 53(1), 63–82.
- Lang, S. and D. Bresger (2004). Bayesian p-splines. *Journal of Computational and Graphical Statistics* 13, 183–212.
- Lin, X. and D. Zhang (1999). Inference in generalized additive mixed models using smoothing splines. *Journal of the Royal Statistical Society, Series B* 61, 381–400.
- Lindeberg, J. W. (1922). Eine neue herleitung des exponentialgesetzes in der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift* 15, 211–225.

- Lowry, M. and G. W. Schwert (2002). Ipo market cycles: Bubbles or sequential learning? *The Journal of Finance* 67(3), 1171–1198.
- Mallows, C. L. (1973). Some comments on  $c_p$ . *Technometrics* 15, 661–675.
- Marx, B. D. and P. H. Eilers (1998). Direct generalized additive modeling with penalized likelihood. *Computational Statistics and Data Analysis* 28, 193–209.
- McCullagh, P. and J. A. Nelder (1989). *Generalized Linear Models* (2nd ed.). London: Chapman & Hall.
- Nelder, J. A. and R. W. M. Wedderburn (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A* 135, 370–384.
- Nicholson, A. J. (1954a). Compensatory reactions of populations to stresses and their evolutionary significance. *Australian Journal of Zoology* 2, 1–8.
- Nicholson, A. J. (1954b). An outline of the dynamics of animal populations. *Australian Journal of Zoology* 2, 9–65.
- O’Sullivan, F. B., B. Yandall, and W. Raynor (1986). Automatic smoothing of regression functions in generalized linear models. *Journal of the American Statistical Association* 81, 96–103.
- Parker, R. and J. Rice (1985). Discussion of silverman (1985). *Journal of the Royal Statistical Society, Series B* 47(1).
- Patterson, H. D. and R. Thompson (1971). Recovery of interblock information when block sizes are unequal. *Biometrika* 58, 545–554.
- Peng, R. D. and L. J. Welty (2004). The NMMAPls package. *R News* 4(2), 10–14.
- Pinheiro, J. C. and D. M. Bates (2000). *Mixed- Effects Models in S and S-PLUS*. New York: Springer-Verlag.
- R Development Core Team (2005). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.
- Reinsch, C. H. (1967). Smoothing by spline functions. *Numerische Mathematik* 10, 177–183.
- Rigby, R. A. and D. M. Stasinopoulos (2004). Generalized additive models for location, scale and shape (with discussion). *Applied Statistics* 54, 1–38.
- Ruppert, D., M. P. Wand, and R. J. Carroll (2003). *Semiparametric Regression*. Cambridge University Press.
- Schoenberg, I. J. (1964). Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences* 52, 947–950.
- Silverman, B. W. (1985). Some aspects of the spline smoothing approach to nonparametric regression curve fitting. *Journal of the Royal Statistical Society, Series B* 47(1), 1–53.
- Silvey, S. D. (1970). *Statistical Inference*. Chapman & Hall.
- Smith, A. F. (1967). Diagnostic value of serum-creatinine-kinase in a coronary care

- unit. *Lancet* 2, 178.
- Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society, Series B* 36, 111–147.
- Stone, M. (1977). An asymptotic equivalence of choise of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society, Series B* 39, 44–47.
- Venables, B. and B. R. Ripley (2003). *Modern Applied Statistics in S* (4th ed.). Springer.
- Wahba, G. (1980). Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of noisy data. In E. Cheney (Ed.), *Approximation Theory III*. London: Academic Press.
- Wahba, G. (1983). Bayesian confidence intervals for the cross validated smoothing spline. *Journal of the Royal Statistical Society, Series B* 45, 133–150.
- Wahba, G. (1990). *Spline models for observational data*. Philadelphia: SIAM.
- Wahba, G., Y. Wang, C. Gu, R. Klein, and B. Klein (1995). Smoothing spline ANOVA for exponential families, with application to the Wisconsin epidemiological *The Annals of Statistics* 23(6), 1865–1895.
- Wang, Y. (1998a). Mixed effects smoothing spline analysis of variance. *Journal of the Royal Statistical Society, Series B* 60, 159–174.
- Wang, Y. (1998b). Smoothing spline models with correlated random errors. *Journal of the American Statistical Association* 93(441), 341–348.
- Watkins, D. S. (1991). *Fundamentals of Matrix Computation*. New York: John Wiley and Sons.
- Wood, S. N. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *Journal of the Royal Statistical Society, Series B* 62, 413–428.
- Wood, S. N. (2003). Thin plate regression splines. *Journal of the Royal Statistical Society, Series B* 65, 95–114.
- Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association* 99, 673–686.
- Wood, S. N. (2006). Low rank scale invariant tensor product smooths for generalized additive mixed models. *Biometrics*.

---

# Index

---

- $\chi^2$  test, 70, 193
- additive model, 131–135  
fitting, 133  
model matrix, 132  
penalty, 132
- age of Universe, 5
- aggregate, 278
- AIC, 184  
GLMs, 67, 84  
in general, 110–112  
linear model, 51  
AIC, 228
- analysis of deviance, 68, 70, 85, 95
- ANOVA, 15, 44, 55, 58, 281  
decomposition of function, 204–206
- anova, 44, 228
- augmented linear model, 127, 133, 207, 218
- autocorrelation, 319
- B-spline, 150  
basis, 150
- backfitting, 211–213
- basis, 144  
B-spline, 150  
choice of dimension, 159, 222  
cubic regression, 123  
cubic regression spline, 148  
expansion, 120, 165  
polynomial, 120  
tensor product, 160–162  
thin plate spline, 154  
truncated power, 271
- Bayesian covariance matrix, 183, 209
- Bayesian smoothing model, 187–189, 313, 314  
prior structure, 189  
simulation from posterior, 192, 244, 258  
unconditional posterior simulation, 259
- binary data, 259  
residual checking, 114
- binomial distribution, 260
- binomial family, 260
- Bonferroni correction, 325
- by variable, 167, 237, 240  
and centering constraints, 317
- canonical parameter, 62, 71
- central limit theorem, 101, 190
- Chebyshev’s inequality, 191
- Choleski decomposition, 127, 332
- concurvity, 178
- confidence interval  
and transformation, 10  
Bayesian, 192  
Bayesian unconditional, 200–202, 204  
definition, 9  
for smoothing parameter, 321  
GLM, 68, 91  
GLRT inversion, 118  
linear model, 9, 14, 36  
performance for GAM, 194–200  
Wilk’s, 118
- confounding, 178, 276
- contingency tables, 92
- contour, 57
- contrast matrix, 47
- Cook’s distance, 26
- covariance matrix, 330  
of linear transformation, 330
- cross validation, 130  
failure of, 216  
general, *see* GCV  
ordinary, *see* OCV
- cubic regression spline, 148, 224  
fitting, 127  
model matrix, 124, 125  
penalty, 148  
penalty matrix, 127

- cubic spline, 122, 213  
approximation properties, 146  
natural, 144  
penalty, 126, 146  
quadratic penalty, 126  
r.k. basis, 124  
smoothest interpolant, 145
- cyclic spline, 149, 224, 319  
penalty, 149
- daily temperature, 319
- Datasets
- AIDS in Belgium, 87, 118
  - belief in afterlife, 92
  - bird, 261
  - blowfly, 271
  - brain, 228
  - cairo, 319
  - chicago, 245
  - chl, 272
  - co2s, 269
  - engine wear, 124
  - Florida death penalty, 116
  - harrier, 117
  - heart attacks and CK, 81
  - hubble, 5, 267
  - ipo, 269
  - mack, 252
  - MASS
    - mcycle, 139, 267
    - Rubber, 56
  - nlme
    - Gun, 326
    - Loblolly, 302
    - Machines, 286, 306, 325
    - Oxide, 324
    - Rail, 282, 291, 301
    - Soybean, 326
  - R
    - InsectSprays, 58
    - ldeaths, 117
    - PlantGrowth, 42
    - trees, 58, 134, 138, 219
    - warpbreaks, 57
    - sole, 96, 309, 317
    - sperm.comp1, 22
    - sperm.comp2, 22, 31
    - wine, 270
  - design matrix, 11
- deviance, 69  
approximate distribution, 69  
null, 83  
proportion explained, 84  
residual, 84  
scaled, 69
- EDF, 169, 202, 209  
and basis dimension, 159, 222  
matrix, 169
- effective degrees of freedom, *see* EDF
- eigen decomposition, 155, 333  
eigenvalue, 333  
eigenvector, 333  
EM algorithm, 299  
Euclidean norm, 12  
*exclude.too.far*, 267  
exponential family, 59, 62–63
  - mean, 62
  - variance, 63

extrapolation  
dangers of, 269, 361, 366

F-ratio statistic, 15, 30, 70  
F-test, 15, 44, 70, 193, 204, 235, 238, 281, 285

factor variables, 11, 37, 44  
contrasts, 46  
dummy variables, 11, 37  
identifiability, 38  
interactions, 40  
use in R, 41–44

follow up comparisons, 325

formula, 44  
 $I()$ , 27, 45

GAM, 119, 135–137, 165  
as penalized GLM, 165–166  
backfit, 210–213  
Bayesian model, 187–189
  - non-linear functions, 192
  - posterior, 187, 189–192
  - prior, 189

bias, 216  
coefficient estimators, 168, 183  
confidence interval, 192
  - performance, 194–200
  - termwise, 197–200, 206

- constraints, 183  
 centering, 165  
 identifiability, 205–206  
 degrees of freedom, 168–169  
 distributional results, 187–193  
 fitting, 168  
 frequentist approximation, 187  
 hypothesis testing, 192, 202–204, 235, 238  
 unpenalized, 193, 204  
 influence matrix, 181  
 interaction  
   smooth with factor, 167  
 large sample results, 189–192  
 model matrix, 166  
 motivation, 101  
 penalized log likelihood, 166, 167  
 proximity of penalized and unpenalized versions, 203  
 scale estimation, 169–170  
 termwise p-values, 192–193, 228, 238
- `gam`, 219, 222, 230, 232, 234, 235, 237, 240, 246, 247, 250, 253, 255, 257, 263  
`anova.gam`, 227, 235, 237, 238, 241, 257  
 arguments, 229  
 controlling, 221–223  
   fitting, 229  
 gamma argument, 222, 254  
 knots, 260  
 knots argument, 260  
 parametric terms, 226  
`plot.gam`, 220, 223–225  
   `all.terms`, 227  
   contour plot interpretation, 225  
   n argument, 246  
   `too.far`, 225  
`predict.gam`, 241, 242, 244, 258, 263  
   `lpmatrix`, 243  
`print.gam`, 220, 225  
 scale argument, 254  
`summary.gam`, 228, 263  
 tensor product smooths, 224  
 TPRS terms, 223  
 user defined smoothers, 270
- `gam` package, 263–265  
`gam`, 263  
`plot.gam`, 265  
`summary.gam`, 263
- `gam.check`, 231, 232, 246  
`gam.method`, 232  
 GAMM, 312–322  
   and AIC, 316  
   inference, 315–316  
   with R, 316–322
- `gamm`  
 convergence problems, 303  
 correlation structures, 319  
 random argument, 301, 317  
 use of, 316–322
- gamma distribution, 223, 234  
 Gamma family, 223, 234  
 Gauss Markov theorem, 17  
 Gauss-Newton method, 53  
 GCV, 130, 173–176  
   derivatives, 181–186  
   for GLM, 175–176, 218  
 GAM case, 136  
 Pearson based, 176, 215  
 score, 180, 184  
 single penalty case, 218  
 stable optimization, 180–186
- generalized additive mixed model, *see* GAMM  
 generalized additive model, *see* GAM  
 generalized cross validation, *see* GCV  
 generalized likelihood ratio test, 68, 85, 107–110, 202  
 assumptions, 293, 304–306
- generalized linear mixed model, *see* GLMM  
 generalized linear model, *see* GLM  
 generalized smoothing splines, 213–215  
 geometry  
   generalized linear model , 75–80  
   IRLS, 76  
   IRLS convergence, 79  
   least squares, 19  
   linear model, 18  
     general covariance structure, 50  
     nested models, 21  
     non-linear least squares, 54  
     orthogonal fitting, 20  
     penalized least squares, 206–207
- `geoR` package, 261  
 GLM, 59  
   estimation, 63  
   estimator distribution, 68  
   fitted values

- properties, 72  
hypothesis testing, 68  
in R, 80–96, 98–101  
likelihood, 64  
quasi-likelihood, 73–75  
residuals, 72  
    deviance, 73  
    Pearson, 72  
    working, 72  
theory, 60–75  
`glm`, 80, 83, 85, 88, 89, 94, 95, 98  
    AIC, 84  
    anova, 95, 100  
    summary.glm, 98  
    update, 99  
`GLMM`, 307–312  
    PQL, 309  
`glmmPQL`, 309  
    convergence problems, 303  
`gss` package, 265–267  
    predict.sanova1, 266  
    anova1, 265
- hat matrix, 16  
Householder matrix, 331  
Hubble’s law, 1  
hypothesis testing  
    and model selection, 193  
    GAM, 192, 202–204, 237–241  
    GLM, 68, 85, 90, 95  
    GLRT, 107–110  
    linear model, 7–9, 14, 15, 44  
    unpenalized GAM, 193, 204, 241, 273
- idempotency, 16  
influence matrix, 16, 127, 181  
    properties, 16, 268  
information, 103  
information matrix, 68, 105  
    empirical or observed, 107  
interaction  
    factors, 40  
    smooth, 160, 204  
    smooth with factor, 167  
invariance, 226  
    and cross validation, 172–175  
    GLRT, 110  
    of MLEs, 102
- rotational, 158, 160, 223, 260  
scale, 160, 223, 265
- IRLS**  
algorithm, 66  
convergence, 79  
derivation, 63–66  
divergence, 178  
geometry, 76  
initialization, 66  
iterative weights, 66  
penalized, 136  
properties, 66  
pseudodata, 66, 168, 190  
weights, 168, 190
- iteratively re-weighted least squares, *see* IRLS
- Jacobian, 53, 65  
Jensen’s inequality, 104
- knots, 122, 125, 133, 158  
Kronecker product, 331  
Kullbeck-Leibler discrepancy, 111
- Lanczos iteration, 156, 335  
Laplace approximation, 108, 308  
law of large numbers, 101  
least squares, 12  
    estimators, 12  
    non-independent data, 49  
    non-linear, 53  
    residual variance estimation, 56
- likelihood  
    consistency of MLEs, 105  
    distribution of estimators, 107  
    linear model, 48  
    properties of expected log, 102  
    theory, 101–114
- linear constraints, 38, 46, 58  
linear contrasts, 46  
linear mixed model, 275  
    general case, 289–307  
    and penalized regression, 298–299  
    estimation, 290–292  
    grouped data, 301  
    likelihood, 290  
    predicting random effects, 294–295, 298–299

- R functions, *see also* `lme`, 300–307
  - response covariance matrix, 290, 291
  - simple balanced case, 275–289
    - 2 way design, 283–288
    - aggregated model, 278, 282, 285
    - estimation in R, 282, 286
    - general principles, 279–280, 288–289
    - hypothesis testing, 287
    - interactions, 284
    - oneway ANOVA, 280–283
    - variance components, 279, 282, 286, 287
    - why bother, 275–280
  - linear model, 2, 10
    - ANOVA, 15
    - checking, 25
    - coefficients, 27
    - estimator distribution, 13
    - F-ratio, 15
    - fitted values, 16
      - properties, 56
    - `formula`, 5, 44
    - likelihood, 48
    - model matrix, 11, 24
    - model selection, 30–35
      - stepwise, 57
    - polynomial, 57
    - prediction, 36
    - residuals, 16
    - t-ratio, 14, 29
    - theory , 12–18
      - traditional results, 17
  - `lm`, 5, 6, 24, 28, 31, 33–35, 43, 44, 125, 128, 134, 137
    - associated functions, 23
    - for mixed models, 277
    - `plot.lm`, 25
    - `print.lm`, 27
    - `step`, 57
    - `summary.lm`, 28
    - use of, 24
  - `lme`
    - convergence problems, 303
    - correlation argument, 303
    - form of model, 301
    - intervals, 320
  - `lmeControl`, 303
  - `msMaxIter`, 303
  - `niterEM`, 303
  - `plot.lme`, 303
    - use of, 301
  - log-linear models, 92
  - `lpmatrix`, 243
  - Mallow’s statistic, 51, 170
  - matrix
    - square root, 291
    - determinant, 334
    - differentiating, 331
    - efficient computation, 329
    - positive definite, 334
    - positive semi-definite, 334
    - square root, 127
    - trace, 334
  - matrix square root, 181
  - mean square error, 129, 170
  - `mgcv`, 219
  - mixed effects model
    - general case, 302
  - model formula, *see* `formula`
  - model matrix, 11, 289
  - multiple comparisons, 325
  - natural spline, 144
  - negative binomial distribution, 256
  - `negative.binomial` family, 256
  - nesting, 204, 236, 276
  - Newton’s method, 180, 290
  - `nlme` package, 300
  - normal distribution
    - joint distribution results, 294
  - OCV, 129, 171–173
    - not invariant, 172
  - offset, 97
  - `offset`, 118, 253
  - `optim`, 292
  - ordinary cross validation, *see* OCV
  - orthogonal matrix, 331
  - outer iteration, 177, 184–186
  - outliers
    - dealing with, 27, 33, 230, 246–250
  - overdispersion, 75, 98
  - overfitting

- checking for, 272
- P-IRLS, 136, 167–168, 217  
differentiation of, 184–186  
divergence, 178
- P-spline, 150, 217, 224  
penalty, 152
- p-value, 29, 95  
and model selection, 193  
for GAM term, 192–193  
interpretation, 29  
properties, 273
- partial residuals, *see* residuals, partial
- Pearson statistic, 71, 170, 176
- penalized IRLS, *see* P-IRLS
- penalized quasi-likelihood, *see* GLMM, PQL
- performance iteration, 178–184  
convergence failure, 178
- pivoting, 335
- Poisson distribution, 252
- poisson family, 253
- polynomial, 120, 302
- polynomial model  
ensuring stability, 98, 302
- posterior simulation, 192, 244
- PQL, 309, 312
- prediction  
`lpmatrix`, 243  
response scale, 242  
term-wise, 242
- prediction error, 129, 171
- propagation of errors, 330
- QQ plot, 26
- QR decomposition, 13, 57, 206, 332
- quasi family, 310
- quasi-information, 113
- quasi-likelihood, 73  
theory, 112–114
- R model formulae, 44
- $R^2$ , 29  
adjusted, 29, 228
- random effects, 275, 278
- rank deficiency, 181, 186, 205–206
- reflector matrix, *see* Householder matrix
- REML, 295–297, 312  
and inference, 297
- explicit form, 297
- motivation, 296
- restrictions on use, 297
- reproducing kernel, 124, 214
- residual plots  
against fitted values, 26  
Cook’s distance, 27  
GLMM, 311  
GLMs, 84  
linear mixed model, 288  
linear model, 5  
QQ-plot, 26  
scale location, 26  
zero line, 101
- residual sum of squares, 13
- residuals  
autocorrelation, 269, 319  
checking binary, 114  
deviance, 73  
distribution, 73  
GAM, 231, 246  
GLMs, 72  
improvement by transformation, 230–233  
linear model, 16  
partial, 220  
Pearson, 72  
properties, 56  
standardized, 17, 26
- s, 223
- `bs` argument, 222, 224
- `fx` argument, 273
- `k` argument, 222  
choice of, 159
- scale parameter, 62, 70, 215  
estimation, 70, 169
- scope of statistics, 2
- semi-parametric models, 226–228
- shrinkage smoother, 158, 224, 253
- singular value decomposition, 334
- singularity, 181, 186
- smoothing bias, 170, 187
- smoothing parameter, 126, 146  
as variance component, 314  
confidence interval, 321
- smoothing parameter estimation, 129–131,  
134, 177–186, 217
- criteria, 170–176
- efficiently for single smooth, 218

- numerical strategy, 177  
 outer iteration, 177, 184–186  
 performance iteration, 178–184  
*smoothing spline ANOVA*, *see SS-ANOVA*  
*smooths*, *see also* *spline*  
 as mixed model terms, 313–315  
 built into *mgcv*, 224  
 of several variables, 223–226  
 running mean, 269  
*Sole eggs*, 96  
*spectral decomposition*, *see eigen decomposition*  
*sperm competition*, 22–36  
*spline*  
 B-spline, 150  
 basis size choice, 159  
 built into *mgcv*, 224  
 cubic, 122, 215  
 cubic regression, 122, 148  
 cubic smoothing, 146, 213  
 cyclic, 149  
 equivalent kernel, 268  
 fitting, 127  
 natural parameterization, 207–210, 216, 217  
 P-spline, 150, 217  
     truncated power, 271  
 penalized regression, 126, 147  
 penalized regression estimator, 127  
 reproducing kernel approach, 213–215  
 tensor product, 160  
 theoretical properties, 144–147  
 thin plate, 152  
 thin plate regression (TPRS), 152  
*SS-ANOVA*, 204, 265  
*summary*  
*gam*, 228  
*glm*, 90, 91  
*lm*, 28  
  
*te*, 223, 235  
     *bs* argument, 317  
     *k* argument, 224, 250  
 tensor product smooth, 160–165  
     alternative penalties, 164  
     as mixed model term, 314–315  
     basis, 160–162  
     comparison with TPRS, 226, 235–237  
     marginal bases, 160  
  
 marginal penalties, 163  
 penalty, 163, 164  
 penalty re-parameterization, 164  
 tensor product smooth penalty, 163–165  
*thin plate regression splines*, *see* TPRS  
*thin plate spline*, 152, 218  
     basis, 154  
     penalty, 153  
 TPRS, 152–158, 224  
     basis, 155, 157  
     comparison with tensor product, 226, 235–237  
     construction, 155  
     knot based, 158, 218  
     properties, 156, 268  
     speeding up, 260  
*transformation*  
     to reduce leverage, 254  
     to stabilize variance, 26, 230–233  
  
 UBRE, 170–171  
     derivatives, 181–186  
     equivalence to AIC, 176  
     for GLM, 176  
     score, 180, 184  
     stable optimization, 180–186  
*unbiased risk estimator*, *see* UBRE  
*update*, 99  
  
 validation set, 272  
*Variable coefficient model*, 166–167  
*variable coefficient model*, 237  
     mixed effects version, 316  
*variance function*, 63  
*variogram*, 261  
*vis.gam*, 233, 236, 239  
  
 Wald test, 110, 193, 238  
*weighted constrained penalized least squares*, 183  
*wigginess penalty*, 146