Chapter 2
LMs, GLMs and GAMs

. . . I have long been a proponent of the following unified field theory for statistics: "Almost all of statistics is linear regression, and most of what is left over is non-linear regression."
—R. I. Jennrich (discussion of Green (1984))

2.1 Introduction

This chapter reviews a few details about two building blocks for this book: LMs and smoothing. These topics naturally draw in two others, viz. GLMs and GAMs, whose inclusion poses the risk of distracting the reader from the main thrust of this book by having to include some material that is not quite necessary. Some justification and details, such as computational, are deferred to the next two chapters, where they are described under more general conditions.

2.2 LMs

LMs operate on data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where y_i is the ith response, $\boldsymbol{x}_i$ is explanatory, and $p \leq n$ (classically, it is usually not considered good practice if n is not much more than p). The data is assumed to follow the model

$$Y_i \;=\; \beta_1 \, x_{i1} + \cdots + \beta_p \, x_{ip} + \varepsilon_i, \qquad \varepsilon_i \;\sim\; N(0,\; \sigma^2) \;\text{ independently.} \tag{2.1}$$

Note that R has the first column of the model matrix $\mathbf{X}$ equal to ones if there is an intercept, i.e., $x_{i1} = 1$. Often, people call (2.1) a *multiple linear regression*. In the wide literature, the ε_i are known under various names, such as the (statistical) error, white noise, innovations, random noise. The mean of Y_i, given $\boldsymbol{x}_i$, is

$$E(Y_i|\boldsymbol{x}_i) \;=\; \mu_i(\boldsymbol{x}_i) \;=\; \boldsymbol{\beta}^T \boldsymbol{x}_i \;=\; \sum_{k=1}^{p} \beta_k \, x_{ik} \, . \tag{2.2}$$

T.W. Yee, *Vector Generalized Linear and Additive Models*,
Springer Series in Statistics, DOI 10.1007/978-1-4939-2818-7_2

Confusingly, some people call the LM a "general linear model", which has the same acronym as a "generalized linear model"(GLM). The general linear model includes generalized least squares, whereas the LM usually refers to ordinary least squares.

It is more compact to write the LM using vectors and matrices:

$$\boldsymbol{Y} \;=\; \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon}_i \;\sim\; N_n(\mathbf{0},\, \sigma^2\, \mathbf{I}_n), \tag{2.3}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$ and $x_{ik} = (\mathbf{X})_{ik}$. Later, we will sometimes write $\mathbf{X}$ as $\mathbf{X}_{\text{LM}}$ to distinguish it from another type of *model matrix* (*design matrix*).

The LM makes strong assumptions. Embedded in the above equations, these are the following.

(i) *Independence of the errors*: data such as time series often violate this assumption because the errors are correlated with each other, in particular, with observations from the past. Independence of the ε_i is not explicitly stated in (2.3) because of a property of the multivariate normal distribution that $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0$ iff ε_i and ε_j are independent.

(ii) *Linearity*: the mean function defines a hyperplane in p-dimensional space. Each covariate x_k is usually modelled having a linear effect on the mean response, keeping the other variables in $\boldsymbol{x}$ fixed. Actually, LMs are called LMs because the mean is linear with respect to the parameters, so that a polynomial in x_k is a LM.

(iii) *Errors that are normal with mean zero and constant variance*: for example, a common form of *heteroscedasticity* is when $\text{Var}(\varepsilon_i)$ increases with increasing mean μ_i.

In practice, all these assumptions should be checked as part of the model-building process; see Sect. 2.3.2. Relaxing the linearity assumption is the main motivation of *additive models*, e.g., see Sect. 2.5 and Chap. 4, and is a major theme of this book.

To estimate $\boldsymbol{\beta}$, it is most common to minimize the residual sum of squares

$$\text{ResSS}(\boldsymbol{\beta}) \;=\; \sum_{i=1}^{n} \varepsilon_i^2 \;=\; (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) \tag{2.4}$$

as a function of $\boldsymbol{\beta}$. Of course, this is why the name 'least squares' (LS) is used, and the estimator corresponds to the MLE because the errors are assumed to be normally distributed. Setting the derivative of ResSS with respect to $\boldsymbol{\beta}$ as $\mathbf{0}$ gives the *normal equations*

$$\mathbf{X}^T\mathbf{X}\,\boldsymbol{\beta} \;=\; \mathbf{X}^T\boldsymbol{Y}. \tag{2.5}$$

Provided that $\mathbf{X}$ is of full column-rank, then

$$\widehat{\boldsymbol{\beta}} \;=\; \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\boldsymbol{y} \tag{2.6}$$

is the LS estimate, and so

$$\widehat{\boldsymbol{y}} \;=\; \mathbf{X}\widehat{\boldsymbol{\beta}} \;=\; \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\boldsymbol{y} \;=\; \boldsymbol{\mathcal{H}}\,\boldsymbol{y}, \ \text{ say,} \tag{2.7}$$

where $\boldsymbol{\mathcal{H}}$ is known as the *hat matrix*. Then $\widehat{y}_i = \boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}}$ are the *fitted values*, and $r_i = y_i - \widehat{y}_i$ the *residuals*. It then follows that $\text{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$, and

$$\widehat{\boldsymbol{\beta}} \;\sim\; N_p\left(\boldsymbol{\beta},\; \sigma^2 \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right). \tag{2.8}$$

Provided that an estimate of σ can be obtained, (2.8) can be used for inference, e.g., to calculate the standard errors for the $\widehat{\beta}_k$, and construct confidence intervals. We shall see below that the quantity S^2, defined as $\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2/(n-p)$, is unbiased for σ^2, therefore it is natural to use $\widehat{\sigma} = s$.

Suppose we wish to test the null hypothesis $H_0 : \mathbf{A}\boldsymbol{\beta} = \boldsymbol{c}$ for some $q \times p$ matrix of known constants $\mathbf{A}$ having rank-q, and $\boldsymbol{c}$ is a vector of known constants. For example, this can be used to test that a subset of the β_k are all 0. Fitting the LM under this constrained (smaller) model results in a residual sum of squares which can be denoted by ResSS_0. Likewise, the unconstrained (larger or full) model has a ResSS_1 which will be generally lower. Then the test statistic F_0 is distributed as

$$F_0 \;=\; \frac{(\text{ResSS}_0 - \text{ResSS}_1)/q}{\text{ResSS}_1/(n-p)} \;\sim\; F_{q,n-p}. \tag{2.9}$$

The null hypothesis is rejected at the α-significance level if $F_0 > F_{q,n-p}(1-\alpha)$. The function `linearHypothesis()` in `car` may be used to test hypotheses of this form. The special cases of testing $H_0 : \beta_k = 0$ versus $H_1 : \beta_k \neq 0$, for all k one-at-a-time, is printed out in the `summary(lmObject)` output. For these, the test statistics are labelled "`t values`" because $T^2 \sim F_{1,\nu}$ for $T \sim t_\nu$, and have the form $t_{0k} = (\widehat{\beta}_k - 0)/\text{SE}(\widehat{\beta}_k)$. The 2-sided p-values are printed as `Pr(>|t|)`, and any controversial 'significance stars' adjacent to them.

A $100(1-\alpha)\%$ confidence ellipsoid for $\boldsymbol{\beta}$ comprises values of $\boldsymbol{\beta}$ such that

$$\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right)^T \mathbf{X}^T\mathbf{X} \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right) \;\leq\; p\,\widehat{\sigma}^2 F_{p,n-p}(1-\alpha).$$

Often we wish to look at one coefficient of $\boldsymbol{\beta}$ at a time, then a $100(1-\alpha)\%$ confidence interval for β_k is $\widehat{\beta}_k \pm t_{n-p}(1-\alpha/2)\,\text{SE}(\widehat{\beta}_k)$, where the standard error is $s \cdot [(\mathbf{X}^T\mathbf{X})^{-1}]_{kk}$. A call of the form `confint(lmObject)` returns such intervals.

For prediction at a value $\boldsymbol{x}_0$, say, a $100(1-\alpha)\%$ prediction interval for $y(\boldsymbol{x}_0)$ is

$$\boldsymbol{x}_0^T\,\widehat{\boldsymbol{\beta}} \pm t_{n-p}(1-\alpha/2)\,\widehat{\sigma}\,\sqrt{1 + \boldsymbol{x}_0^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\boldsymbol{x}_0}.$$

Similarly, a $100(1-\alpha)\%$ confidence interval for $\mu(\boldsymbol{x}_0)$ is

$$\boldsymbol{x}_0^T\,\widehat{\boldsymbol{\beta}} \pm t_{n-p}(1-\alpha/2)\,\widehat{\sigma}\,\sqrt{\boldsymbol{x}_0^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\boldsymbol{x}_0}.$$

Prediction intervals focus on a future (random) value of y and are consequently wider than a confidence interval for the (fixed) conditional mean $E(Y|\boldsymbol{x}_0)$. This is intuitively so because confidence intervals simply need to account for the uncertainty in $\widehat{\boldsymbol{\beta}}$, whereas prediction intervals have the additional randomness due to $\text{Var}(\varepsilon_i)$ as well.

2.2.1 The Hat Matrix

From (2.7), we have the $n \times n$ matrix

$$\boldsymbol{\mathcal{H}} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T, \tag{2.10}$$

which is referred to as the 'hat' matrix because it adds a 'hat' to $\boldsymbol{y}$ (i.e., $\widehat{\boldsymbol{y}}$) when $\boldsymbol{y}$ is premultiplied by it. Assuming that $\mathbf{X}$ is of rank-p, the hat matrix has the following properties.

(i) $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}^T$, i.e., symmetric.
(ii) $\boldsymbol{\mathcal{H}}^2 = \boldsymbol{\mathcal{H}}$, i.e., idempotent.
(iii) $\boldsymbol{\mathcal{H}}\mathbf{1} = \mathbf{1}$ if the LM has an intercept, i.e., all rows sum to unity.
(iv) $\text{rank}(\boldsymbol{\mathcal{H}}) = \text{trace}(\boldsymbol{\mathcal{H}}) = p$.
(v) $\boldsymbol{\mathcal{H}}$ has p unit eigenvalues and $n-p$ zero eigenvalues.
(vi) $0 \leq h_{ii} \leq 1$, where $h_{ij} = (\boldsymbol{\mathcal{H}})_{ij}$ is the (i,j)-element of $\boldsymbol{\mathcal{H}}$. If the LM has an intercept, then $n^{-1} \leq h_{ii} \leq 1$.
(vii) $\text{Var}(r_i) = \sigma^2(1-h_{ii})$. This serves as motivation for (2.12).

The proofs are not difficult and are left as an exercise (Ex. 2.1). The hat matrix $\boldsymbol{\mathcal{H}}$ is also known as a *projection matrix* because it is the orthogonal projection of $\boldsymbol{Y}$ onto the column (range) space of $\mathbf{X}$; it has the two properties of being symmetric and idempotent.

To show that S^2 is unbiased for σ^2, $E[\text{ResSS}] = E[(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})^T(\boldsymbol{Y} - \widehat{\boldsymbol{Y}})] = E[\boldsymbol{Y}^T(\mathbf{I}-\boldsymbol{\mathcal{H}})^T(\mathbf{I}-\boldsymbol{\mathcal{H}})\boldsymbol{Y}] = E[\boldsymbol{Y}^T(\mathbf{I}-\boldsymbol{\mathcal{H}})\boldsymbol{Y}] = \text{trace}((\mathbf{I}-\boldsymbol{\mathcal{H}})\,\sigma^2\mathbf{I}) + \boldsymbol{\mu}(\mathbf{I}-\boldsymbol{\mathcal{H}})\boldsymbol{\mu} = \sigma^2(n-p) + 0 = \sigma^2(n-p)$, by formulas given in Sect. A.2.5. These results are generalized later for linear smoothers in, e.g., Sect. 2.4.7.4.

The importance of the hat matrix, especially for diagnostic checking, is due to its interpretation as containing the weights of all the observations in obtaining the fitted value at a particular point. This can be seen by focusing on the ith row of (2.7):

$$\widehat{y}_i = \sum_{j=1}^{n} h_{ij}\, y_j, \tag{2.11}$$

so that h_{ij} can be interpreted as the weight associated with datum (x_j, y_j) to give the fitted value for datum (x_i, y_i). For the $p = 2$ case, plotting the h_{ij} versus x_{j2} for $j = 1, \ldots, n$, is analogous to the equivalent kernels for smoothers considered in the next chapter (e.g., Sect. 2.4.7.3).

However, it is usually the diagonal elements of $\boldsymbol{\mathcal{H}}$ that are of greatest relevance to people fitting LMs. Element h_{ii} measures how much impact y_i has on $\widehat{y}_i$, and consequently it quantifies the amount of influence that observation i has on the fit. Indeed, the h_{ii} are called *leverages* or *leverage scores*, and they measure how far $\boldsymbol{x}_i$ is away from the centre of all the data ($\overline{\boldsymbol{x}}$). Intuitively, as ResSS is being minimized, observations $\boldsymbol{x}_i$ that are isolated from the rest cause the regression plane $\mu(\boldsymbol{x})$ to be 'pulled' unduly towards them. The leverages can be defined as $\partial\widehat{y}_i/\partial y_i$. Since the sum of the h_{ii} is p, any individual diagonal element that is substantially higher than the mean value can be considered influential. It is common to use the rule-of-thumb: if $h_{ii} > 2p/n$, say, then observation i is influential or has high leverage. Others use $h_{ii} > 3p/n$ instead. As high-leverage points may 'pull' the regression line or plane towards them, they do not necessarily have a large residual.

2.2.1.1 LM Residuals and Diagnostics

A major component in checking the underlying LM assumptions is the examination of the residuals. Indeed, the use of diagnostic plots and other tools to check the adequacy of a fitted regression model separates competent practitioners from amateurs. From above, the ordinary residuals do not have equal variances, therefore

$$r_i^{\text{std}} \;=\; \frac{y_i - \widehat{y}_i}{s\,\sqrt{1-h_{ii}}} \tag{2.12}$$

are used commonly, called *standardized residuals* or *(internally) Studentized residuals*. Here, s follows from the result that S^2 is an unbiased estimator for σ^2. However, if $(\boldsymbol{x}_i, y_i)$ is an outlier, then s may be affected, therefore it is safer to use the *(externally) Studentized residuals* (or simply *Studentized residuals*)

$$r_i^{\text{stu}} \;=\; \frac{y_i - \widehat{y}_i}{s^{[-i]}\,\sqrt{1-h_{ii}}} \tag{2.13}$$

where $s^{[-i]}$ is the estimate of σ by deleting observation i. The function `hatvalues()` returns $\text{diag}(\boldsymbol{\mathcal{H}})$, and `rstandard()` and `rstudent()` return (2.12) and (2.13), respectively.

LM diagnostics are quite a large subject and beyond the scope of this book. Here is a small listing of popular diagnostic plots for detecting violations in some of the underlying LM assumptions. Here, 'residuals' are best standardized or Studentized residuals, although ordinary residuals are often used.

1. When the residuals are plotted against their fitted values, ideally one would want a patternless horizontal band. A common form of departure from this is a 'funnel-effect', where there is less spread at lower fitted values—this suggests non-constant variance of the errors.
2. Plot the *partial residuals* against each x_k, e.g., $r_i + \widehat{\beta}_k x_{ik}$ versus x_{ik}. These types of residuals attempt to remove the effect of x_k from the regression temporarily. If indeed x_k has a linear effect on the response, then removing $\widehat{\beta}_k x_{ik}$ from the residual should leave white-noise. Any nonlinear trend would suggest that the $\beta_k x_k$ term might be generalized to some smooth function $f_k(x_k)$, i.e., an additive model. Not surprising, partial residuals are central to the backfitting algorithm for fitting VGAMs (Sect. 4.3.2).
3. Check the normality of the errors by a normal Q-Q plot of the residuals or a histogram, e.g., with `qqnorm()` and `hist()`.
4. As a check of the independence of the errors, plotting $\widehat{\varepsilon}_{i-1}$ versus $\widehat{\varepsilon}_i$ and obtaining a pattern with a nonzero slope suggests a violation of the independence assumption. For this, the Durbin-Watson test is popular, and there are implementations for this in, e.g., car and lmtest. More fundamentally, how the data was generated and collected needs to be considered in the broader context of the analysis.

Some of the above are produced by a call of the form `plot(lmObject)`. Common remedies to violations of the LM assumptions include transforming the response or the x_k, adding polynomial terms in x_k, using weighted least squares (WLS) or generalized least squares (GLS), and fitting additive models.

From an inference point-of-view, it is generally thought that independence of the errors is the most crucial assumption, followed by constant variance of the errors and linearity with respect to each x_k. Normality of the errors is the least important assumption, and the Shapiro-Wilk test can be used for this (`shapiro.test()`). Of course, gross features such as outliers and high-leverage points must be attended to. Other potential problems include

- multicollinearity (sets of x_k which are highly correlated or almost linearly dependent, e.g., $x_2 = \log$ `width` and $x_3 = \log$ `breadth` would be highly correlated with $x_4 = \log$ `area` if regions were approximately rectangular in shape),
- interactions (e.g., the effect of x_s on Y changes with the value of another explanatory variable x_t); these can be complicated and hard to deal with,
- variable selection, e.g., trying to determine which variables should be included in the model. This problem is exacerbated in an age of Big Data, where many variables are collected routinely.

Consequently, good linear modelling requires skill and diligence; it is an art as well as a science.

Regarding variable selection, a simple technique for moderate p that can be performed manually, called *backward elimination*, involves fitting a model with all the explanatory variables in, and then removing the least significant variable (i.e., the one with the largest p-value) and refitting a new model. This procedure is repeated until all remaining variables are 'significant'. The criterion for a 'significant' variable might be one whose p-value less than 5% or 10%; it should be decided upon beforehand.

For `lm()` fits, `influence.measures()` is the primary high-level function for diagnostics. It returns a number of quantities such as DFBETAS, DFFITS, Cook's distances, and diag($\boldsymbol{\mathcal{H}}$). These are 'leave-one-out' diagnostics because they measure the effect of removing one observation at a time. In particular, DFBETA are the quantities $\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}_{[-i]}$, the **df**ference in the **beta** vector of coefficients. DFBETAS is the **s**caled version of DFBETA, DFFITS is a scaled version of DFFIT (which is $\widehat{y}_i - \boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}}_{[-i]} = \boldsymbol{x}_i^T(\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}_{[-i]})$). These quantities plus more are defined in Belsley et al. (1980, Chap.2). Cook's distances measure the effect of the ith observation on $\widehat{\boldsymbol{\beta}}$, in a way that picks up high-leverage points and observations with large Studentized residuals—`cooks.distance()` can return these quantities.

2.2.2 WLS and GLS

The assumption in (2.3) that $\mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2 \mathbf{I}_n$ is commonly unrealistic in real data analysis, as the previous section indicated. A generalization that relaxes homoscedastic errors is to allow $\mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2 \operatorname{diag}(w_1, \ldots, w_n)^{-1}$ for positive known weights w_i. Then fitting an LM by minimizing $\sum_i w_i (y_i - \widehat{y}_i)^2$ is known as *weighted least squares* (WLS). Note that the ε_i are still independent, as in *ordinary least squares* (OLS).

It is necessary to generalize WLS further. If $\mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2 \boldsymbol{\Sigma}$ for any known positive-definite matrix $\boldsymbol{\Sigma}$, then estimating $\boldsymbol{\beta}$ by minimizing $(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})$ is called *generalized least squares* (GLS). GLS is needed when considering VGLMs and VGAMs, whereas WLS is sufficient for fitting GLMs. GLS allows the errors ε_i to be correlated.

The following formulas hold for GLS, and it is left to the reader to supply their proofs (Ex. 2.5):

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{X}\right)^{-1} \mathbf{X}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{y}, \tag{2.14}$$

$$\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2 \left(\mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{X}\right)^{-1}, \tag{2.15}$$

$$\frac{(\boldsymbol{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})}{\sigma^2} \sim \chi^2_{n-p}. \tag{2.16}$$

Also, its hat matrix (defined as $\boldsymbol{\mathcal{H}}$ such that $\widehat{\boldsymbol{y}} = \boldsymbol{\mathcal{H}} \boldsymbol{y}$), is idempotent, but not symmetric in general. Equation (2.16) implies that $\widehat{\sigma}^2 = \mathrm{ResSS}/(n-p)$ is an unbiased estimator of σ^2.

2.2.3 *Fitting LMs in* R

The fitting function `lm()` is used to fit LMs. It has arguments

```
> args(lm)

  function (formula, data, subset, weights, na.action, method = "qr",
      model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
      contrasts = NULL, offset, ...)
  NULL
```

Arguments `formula` and `data` should be used all the time. Arguments `subset`, `weights` and `na.action` can be very useful at times, and `offset` can be incorporated into the formula instead by addition of the term `offset(<value>)`.

LMs may be fitted using `glm()`, albeit with slightly less efficiency. The normal distribution being known as the *Gaussian* distribution, `gaussian()` is the default for its `family` argument.

2.3 GLM Basics

This section gives a bare-bones and incomplete overview of GLMs. Such is not really required for an understanding of the VGLMs described in Chap. 3, however it does provide some background for such. Hence this section is given more for completion than for necessity.

GLMs as proposed by Nelder and Wedderburn (1972) provide a unifying framework for a number of important models in the exponential family. In particular, these include the normal (Gaussian), binomial and Poisson distributions. One consequence is that a single algorithm (IRLS) can be used to fit them all.

As with LMs, we have independent sample data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where y_i is the response (more general now), n is the sample size, and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ is a vector of p explanatory variables ($x_{i1} = 1$ is the intercept if there is one). A GLM is composed of three parts:

(i) a *random component* $f(y;\mu)$ specifying the distribution of Y,
(ii) a *systematic component* $\eta = \boldsymbol{\beta}^T \boldsymbol{x}$ specifying the variation in Y accounted for by known covariates, and
(iii) a *link function* $g(\mu) = \boldsymbol{\beta}^T \boldsymbol{x}$ that ties the two together. Often, g is simply called the *link*.

The η is known as the *linear predictor*, and the random component $f(y;\mu)$ is typically an exponential family distribution with $E(Y|\boldsymbol{x}) = \mu(\boldsymbol{x})$ being the *mean function*. GLMs thus fit

$$g(\mu(\boldsymbol{x}_i)) \;=\; \eta_i \;=\; \boldsymbol{\beta}^T \boldsymbol{x}_i \;=\; \beta_1\, x_{i1} + \cdots + \beta_p\, x_{ip}, \tag{2.17}$$

where g is known. The required properties of g are strict monotonicity and being twice-differentiable in the range of μ. The main purpose of g is to transform the mean, which is usually bounded, into an unbounded parameter space where the optimization problem is unfettered and therefore simpler. Another purpose is that it often aids interpretability. In the VGLM framework described in Chap. 3, we write (2.17) as

$$g_1(\mu(\boldsymbol{x}_i)) \;=\; \eta_{i1} \;=\; \boldsymbol{\beta}_1^T \boldsymbol{x}_i \;=\; \beta_{(1)1}\, x_{i1} + \cdots + \beta_{(1)p}\, x_{ip} \tag{2.18}$$

to allow for more than one linear predictor in a model. The linear predictor (2.18) is central to this book. It takes the form of a weighted average of the covariate values $x_{i1}, \ldots, x_{ip}$ for object i—it is a plane in p-dimensional space.

For one observation, the probability density or mass function (PDF or PMF) of the exponential family can be written

$$f(y;\theta,\phi) \;=\; \exp\left\{ \frac{y\,\theta - b(\theta)}{\phi} + c\,(y,\phi) \right\}, \tag{2.19}$$

where θ is called the *natural parameter* or *canonical parameter*, ϕ is a possibly-known *dispersion parameter* (or *scale parameter*), and b and c are known functions. When ϕ is known, the distribution of Y is a one-parameter canonical exponential family member. When ϕ is unknown, it is often a nuisance parameter and then it is estimated by the method of moments. In most of GLM theory, the role of ϕ is curious and unfortunate: it is often treated as an unknown constant but not as a parameter. The VGLM framework views this as a deficiency, because of the original framework's inability to handle more than one parameter gracefully. The VGLM framework tends to estimate all parameters by full maximum likelihood estimation. This makes life easier in general, and estimation and inference is simpler.

At this stage, it is a good idea to handle known prior weights, A_i say, which may be entered into modelling functions such as `glm()` and `vglm()` via the `weights` argument. We will write $\phi_i = \phi/A_i$, where usually $A_i = 1$. In the case of the Y_i being binomial proportions, $N_i Y_i \sim \text{Binomial}(N_i, \mu_i)$ where the N_i can be assimilated into the A_i by $A_i = N_i$. Another reason is because we want to maximize a log-likelihood of the form $\sum_{i=1}^{n} A_i\, \ell_i$; most of this book dwells on maximizing the log-likelihood (3.7) so that the A_i here can be absorbed into the prior weights w_i there (this is largely a change of notation).

Noting that $\ell(\mu; y) = \log\, f(y; \mu)$, the log-likelihood is

$$\ell(\boldsymbol{\theta}, \phi; y) \;=\; \sum_{i=1}^{n} \frac{y_i\, \theta_i - b(\theta_i)}{\phi/A_i} + c\left(y_i, \frac{\phi}{A_i}\right), \tag{2.20}$$

and the score function is

$$U(\boldsymbol{\theta}) \;=\; \frac{\partial \ell}{\partial \boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} \frac{y_i - b'(\theta_i)}{\phi/A_i}. \tag{2.21}$$

Then, using (A.17) and (A.18),

$$E(Y_i) \;=\; b'(\theta_i) \;\text{ and }\; \mathrm{Var}(Y_i) \;=\; \frac{\phi}{A_i}\, b''(\theta_i). \tag{2.22}$$

The *variance function* is $V(\mu) = b''(\theta(\mu))$, i.e., the variance of Y as a function of the mean.

It is noted at this stage that the MLE $\widehat{\boldsymbol{\beta}}$ is obtained by solving the estimating equation

$$\boldsymbol{U}_{\boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} \frac{\partial \mu_i}{\partial \boldsymbol{\beta}}\, \mathrm{Var}(Y_i)^{-1}\, (y_i - \mu_i) \;=\; \mathbf{0}. \tag{2.23}$$

To see this,

$$\begin{aligned}
\mathbf{0} \;&=\; \frac{\partial \ell}{\partial \boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} \frac{\partial \ell_i}{\partial \theta_i} \frac{\partial \theta_i}{\partial \boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} U(\theta_i)\, \frac{\partial \theta_i}{\partial \boldsymbol{\beta}} \\
&=\; \sum_{i=1}^{n} \frac{Y_i - b'(\theta_i)}{\phi/A_i} \frac{1}{(\partial \mu_i/\partial \theta_i)} \frac{\partial \mu_i}{\partial \theta_i} \frac{\partial \theta_i}{\partial \boldsymbol{\beta}} \\
&=\; \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\phi/A_i} \frac{1}{b''(\theta_i)} \frac{\partial \mu_i}{\partial \boldsymbol{\beta}} \\
&=\; \sum_{i=1}^{n} \left\{ \frac{Y_i - \mu_i}{\mathrm{Var}(Y_i)} \right\} \frac{\partial \mu_i}{\partial \boldsymbol{\beta}},
\end{aligned} \tag{2.24}$$

which is the LHS of (2.23). This equation serves as the motivation for quasi-likelihood models described later because it only depends on the first two moments of Y.

Table 2.3 lists the most common members of the exponential family and how they are fitted in VGAM. For GLMs, the canonical parameter is a link function applied to the mean. Consequently, this particular link is known as the *canonical link*. With this link and given ϕ, $\mathbf{X}^T \boldsymbol{y}$ are a set of sufficient statistics for $\boldsymbol{\beta}$.

Iteratively reweighted least squares (IRLS) forms the core algorithm behind GLMs and GAMs. It is described under more general conditions in Sect. 3.2. For GLMs, it involves regressing at each iteration (the ath iteration, say) the *adjusted dependent variable* (or *modified dependent variable* or *working dependent variate* or *pseudo-response*) $z_i^{(a-1)}$ against $\boldsymbol{x}_i$ with (working) weights $w_i^{(a-1)}$; these are given by

$$z_i^{(a-1)} = \eta_i^{(a-1)} + \frac{y_i - \mu_i^{(a-1)}}{d\mu_i^{(a-1)}/d\eta_i} \quad \text{and} \quad w_i^{(a-1)} = \frac{A_i}{V\left(\mu_i^{(a-1)}\right)} \left(\frac{d\mu_i^{(a-1)}}{d\eta_i}\right)^2 . \tag{2.25}$$

It can be achieved by WLS: $\mathbf{X}$ is the model matrix, and $\mathbf{W}^{(a-1)}$ is $\mathrm{diag}(w_1^{(a-1)}, \ldots, w_n^{(a-1)})$.

The above weights are actually obtained by the *expected* negative Hessian (rather than the *observed*), so this is Fisher scoring. For some models this equates to Newton-Raphson. With the new $\boldsymbol{\beta}^{(a)}$, a new $\boldsymbol{\eta}^{(a)}$ and $\boldsymbol{\mu}^{(a)}$ are computed, and the cycle is continued till convergence. When close to the solution, the convergence rate is quadratic for GLMs with a canonical link, meaning that the number of correct decimal places doubles (asymptotically) at each iteration.

At convergence, we have

$$\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\beta}}) = \widehat{\phi}\left(\mathbf{X}^T \mathbf{W}^{(a)} \mathbf{X}\right)^{-1}, \tag{2.26}$$

which is returned by the function `vcov()`.

Convergence problems with GLMs can occur in practice, although they are not particularly common. Section 3.5.4 gives an example.

2.3.1 Inference

There is an elegant body of theory for inference pertaining to GLM models that naturally extend that of ordinary linear theory. Under certain conditions (e.g., grouped binary data), the *deviance* $D(\boldsymbol{y};\boldsymbol{\mu})$ can be used to measure goodness-of-fit of a model. The *scaled deviance* satisfies

$$\frac{D(\boldsymbol{y};\boldsymbol{\mu})}{\phi} = 2\left\{\ell(\boldsymbol{y};\boldsymbol{y}) - \ell(\boldsymbol{\mu};\boldsymbol{y})\right\} \tag{2.27}$$

which is non-negative. The term $\ell(\boldsymbol{y};\boldsymbol{y})$ corresponds to a *saturated model*—one where $\boldsymbol{\mu}$ maximizes ℓ over $\boldsymbol{\mu}$ unconstrained. For GLMs, a saturated model has $b'(\widehat{\theta}_i) = \widehat{\mu}_i = y_i$. The opposite extreme is a *null model*, which is what we call intercept-only, i.e., the R formula is of the form `y ~ 1`. If $\phi = 1$ then $D =$

$$2\sum_{i=1}^{n} A_i \left[\{y_i\,\theta(y_i) - b(\theta(y_i))\} - \left\{y_i\,\widehat{\theta}_i - b(\widehat{\theta}_i)\right\}\right] \quad \left(= \sum_{i=1}^{n} A_i\, d_i, \text{say}\right), \tag{2.28}$$

where $\widehat{\theta}$ is the MLE. This is called the deviance of a model even when the scale parameter is unknown, or is known to have a value other than one. The scaled deviance is sometimes called the *residual deviance*.

Smaller values of D indicate a better fit. The deviance, which is a generalization of the residual sum of squares for the LM, is a function of the data and of the fitted values, and when divided by a dispersion parameter ϕ, it is sometimes asymptotically χ^2 (e.g., as $n \to \infty$, or as the number of binomial trials $N_i \to \infty$). More generally, to test if a smaller model (i.e., one with fewer variables) is applicable given a larger model, it is only necessary to examine the increase in the deviance, and to compare it to a χ^2 distribution with degrees of freedom equal to the difference in the numbers of independent parameters in the two models (as each parameter has 1 degree of freedom). In the model-building process, this enables a test to be carried out as to which variables can be deleted to form a smaller model, or which variables need to be added to form a larger model.

For a Gaussian family with identity link, ϕ is the variance σ^2, and D is the residual sum of squares, i.e.,

$$D \;=\; \sum_{i=1}^{n} A_i \, (y_i - \mu_i)^2.$$

Hence

$$D/\phi \;\sim\; \chi^2_{n-p},$$

leading to the unbiased estimator

$$\widehat{\phi} \;=\; D/(n-p) \tag{2.29}$$

because, with an abuse of notation, $E(\chi^2_\nu) = \nu$.

An alternative estimator is the sum of squares of the standardized residuals divided by the residual degrees of freedom:

$$\tilde{\phi} \;=\; \frac{1}{n-p} \sum_{i=1}^{n} \frac{(y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)/A_i}. \tag{2.30}$$

This formula may be used to estimate the scale parameter ϕ. It has much less bias than (2.29). For the Gaussian errors, $\tilde{\phi} = \widehat{\phi}$.

If $\mathcal{M}_0$ is a submodel within a model $\mathcal{M}$ (that is, nested) with $q < p$ parameters, and if ϕ is known, then

$$\frac{D_{\mathcal{M}_0} - D_{\mathcal{M}}}{\phi} \;\dot{\sim}\; \chi^2_{p-q}.$$

If ϕ is unknown, then

$$\frac{D_{\mathcal{M}_0} - D_{\mathcal{M}}}{\tilde{\phi}\,(p-q)} \;\dot{\sim}\; F_{p-q,n-p}.$$

2.3.2 GLM Residuals and Diagnostics

As with LMs, diagnostics are available for GLMs to help check the underlying assumptions. These tend to be based on residuals, however, GLM residuals are more difficult to utilize compared to LMs because they are harder to interpret.

There are several residual-types that may be defined for GLMs. The `resid()` (or `residuals()`) method function returns one of five types of residuals for `glm()`

objects, depending on the `type` argument. For simplicity we set $A_i = 1$ here, but more general formulas are given in Sect. 3.7 for VGLMs. The five residual types are:

(i) *Deviance residuals* are

$$r_i^D = \operatorname{sign}(y_i - \widehat{\mu}_i)\,\sqrt{d_i}, \quad \text{where} \quad D = \sum_{i=1}^{n} d_i \tag{2.31}$$

(cf. (2.28)). This residual type is the default, and is the most useful for diagnostic purposes.

(ii) *Pearson residuals* are related to the working residuals, and are

$$r_i^P = \frac{y_i - \widehat{\mu}_i}{\sqrt{V(\widehat{\mu}_i)}}. \quad \text{Note that} \quad X^2 = \sum_{i=1}^{n} \left(r_i^P\right)^2 = \sum_{i=1}^{n} \frac{(y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)}$$

is the Pearson chi-squared statistic.

(iii) *Working residuals* are

$$r_i^W = (y_i - \widehat{\mu}_i)\,\frac{\partial \widehat{\eta}_i}{\partial \widehat{\mu}_i}. \tag{2.32}$$

They arise from the final IRLS iteration (cf. (2.25)).

(iv) *Response residuals* are simply $r_i^R = y_i - \widehat{\mu}_i$.

(v) *Partial residuals* are used for enhancing plots of the component functions of GAMs. For $\eta_i = \beta_1 + \beta_2\,x_{i2} + \cdots + \beta_p\,x_{ip}$, these are $\beta_k(x_{ik} - \overline{x}_k) + r_i^W$. For GAMs having η_i of the form $\beta_1 + f_2(x_{i2}) + \cdots + f_p(x_{ip})$, these are $\tilde{f}_k(x_{ik}) + r_i^W$ for $k = 2, \ldots, p$ and where the $\tilde{f}_k$ are centred f_ks.

The first four types of residuals coincide for the Gaussian family. For the types of plots listed in Sect. 2.2.1.1, some people maintain that deviance residuals are the most informative for GLMs. Figure 2.21 is an example of the first four types of residuals.

2.3.3 Estimation of ϕ

In the VGLM/VGAM framework, it is usually preferable to estimate the dispersion parameter by full maximum likelihood because it is simpler, inference is simplified too, and the models can be more flexible. However, for some GLM families such as the gamma, there are problems such as bias, and extreme sensitivity in very small values (McCullagh and Nelder, 1989, Chap. 8).

The most common GLM estimation method for ϕ is to use (2.30) which is based on the method of moments. It is unbiased for the LM, and is generally consistent for GLMs.

2.3.4 Fitting GLMs in R

GLMs are well-served by the modelling function `glm()`. It has arguments

```
> args(glm)

  function (formula, family = gaussian, data, weights, subset,
      na.action, start = NULL, etastart, mustart, offset, control = list(...),
      model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL,
      ...)
  NULL
```

They may be also fitted by `vglm()` with family functions having the same name as `glm()` but with an "`ff`" appended (they must be different because they are incompatible), e.g., `gaussianff()`.

2.3.5 Quasi-Likelihood Models

It is not uncommon for one to be unsure about the full distribution (2.19) of the response, e.g., when the variance of the data is much greater than the model suggests (*overdispersion*). Wedderburn (1974) proposed the use of the quasi-likelihood to help alleviate this problem. Specifically, it replaces the assumptions tied in with (2.19) by the weaker variance assumption that

$$\mathrm{Var}(Y) \;=\; \frac{\phi}{A}\, V(\mu), \tag{2.33}$$

where ϕ is assumed constant across samples. This can be very useful in applied work when the data are limited and information on the distribution of Y is lacking. However, one may have enough prior knowledge to specify, or data to reliably estimate, a relationship between the first two moments, as required by the quasi-likelihood model. In the absence of a likelihood function, one may estimate $\boldsymbol{\beta}$ by solving (2.23) because it only depends on the first two moments. What is the justification for using this? We have already seen that it yields the MLE of $\boldsymbol{\beta}$ for families belonging to the exponential family (2.19).

Now the term in braces in (2.24) is an expression for $\partial\ell_i/\partial\mu_i$. Coupled with (2.33), this suggests that

$$q(\mu; y) \;=\; \int_y^{\mu} \frac{y-t}{(\phi/A)\; V(t)}\, dt \tag{2.34}$$

might behave like a log-likelihood function for μ. Indeed, if

$$U_i \;=\; \frac{Y_i - \mu_i}{(\phi/A_i)\; V(\mu_i)}$$

then $E(U_i) = 0$ and $\mathrm{Var}(U_i) = 1/\{(\phi/A_i)\, V(\mu_i)\} = -E(\partial\, U_i/\partial\, \mu_i)$. These are the same properties that a log-likelihood derivative has (see (A.17)–(A.18)). So the overall conclusion is to solve the estimating equation

Table 2.1 Some quasi-likelihood functions. VGAM families are `quasibinomialff()` and `quasipoissonff()` (adapted from McCullagh and Nelder, 1989, Table 9.1).

Distribution	$V(\mu)$	$\phi\, q(\mu; y)$	`glm()` family
Gaussian	1	$-(y-\mu)^2/2$	
Binomial	$\mu(1-\mu)$	$y \operatorname{logit} \mu + \log(1-\mu)$	`quasibinomial()`
Poisson	μ	$y \log \mu - \mu$	`quasipoisson()`
Gamma	μ^2	$-y/\mu - \log \mu$	
Inverse Gaussian	μ^3	$-y/(2\mu^2) + 1/\mu$	

$$\sum_{i=1}^{n} \frac{\partial\, q(\mu_i; y_i)}{\partial\, \boldsymbol{\beta}} \;=\; \mathbf{0}. \tag{2.35}$$

A list of some quasi-likelihood functions is given in Table 2.1 and the `glm()` family functions for estimating them. In R, the `glm()` family functions `quasibinomial()` and `quasipoisson()` solve for $\boldsymbol{\beta}$ in (2.35) for the binary and Poisson cases, and $\widetilde{\phi}$ in (2.30) is printed out in the `summary()` as the '`Dispersion parameter`'. For `vglm()` in VGAM, the family functions are called `quasibinomialff()` and `quasipoissonff()`.

2.3.6 Binary Responses

We now dwell a little on one specific GLM, viz. the binomial family. In contrast, the Gaussian family is well-served in multitudes of books, and count data is described in Sect. 11.3 via negative binomial regression.

It is noted that two popular models for handling overdispersion with respect to the Poisson and binomial distributions are the negative binomial (Sect. 11.3) and beta-binomial (Sect. 11.4) distributions. Also, the variants positive-binomial, zero-inflated binomial and zero-altered binomial (Chap. 17) are available.

2.3.6.1 Links

Figure 2.1a,d plots four of the most commonly used link functions. The default is the logit link, which is not only very interpretable in terms of log-odds, it is usually indistinguishable in practice from the probit link unless n is large. Of these, only the complementary log–log link is asymmetric; the other three satisfy $p(\frac{1}{2}-c) = p(\frac{1}{2}+c)$ for $0 < c < \frac{1}{2}$. The cloglog link ties in with Poisson regression very simply because if $Y \sim \text{Poisson}(\mu)$ with $\eta = \log \mu$, then $\operatorname{cloglog} P[Y>0] = \log(-\log P[Y=0]) = \log(-\log e^{-\mu}) = \eta$.

The links correspond to the CDFs of some standardized distributions (Table 12.3): the logit link of a logistic distribution, the probit of a normal, the cauchit of a Cauchy, and the cloglog of a extreme-value (log-Weibull) distribution (Chap. 16).

Figure 2.1b are plots of the reciprocal first derivatives, $1/g_j'(p)$, which are relevant to (2.25) because of the term $d\mu/d\eta$. For grouped binomial data

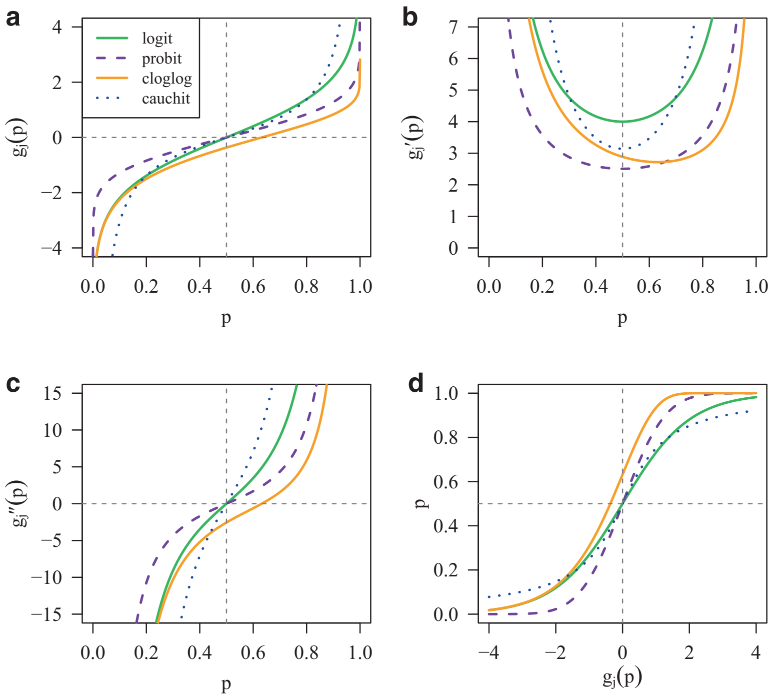

Fig. 2.1 Properties of some common link functions g_j suitable for a probability. (**a**) $g_j(p)$; (**b**) $g_j'(p)$; (**c**) $g_j''(p)$; (**d**) $g_j^{-1}(p)$. The legend in (**a**) is common for all plots. The calls to (**a**)–(**c**) are of the form `link.function(p, deriv = d)` for `d = 0`, `1` and `2` (Table 1.2).

where Y_i represents the proportion of successes out of N_i trials, we have $N_i Y_i \sim \text{Binomial}(N_i,\ \mu_i)$, so that (2.25) with a logit link becomes

$$z_i \;=\; \boldsymbol{\beta}^T \boldsymbol{x}_i + \frac{y_i - \mu_i}{\mu_i(1-\mu_i)}, \quad \text{with working weights} \quad w_i \;=\; N_i\,\mu_i(1-\mu_i).$$

2.3.6.2 The Hauck-Donner Phenomenon

This effect, which was first observed by Hauck and Donner (1977), gives one reason why the likelihood ratio test is to be preferred over the Wald test (these tests are described in Sect. A.1.4.2). They used the following example. Suppose that $n = 200$ for a logistic regression involving two groups of 100 observations each. The observed proportion of 1s in group k is p_k, and let $x_2 = 0$ and 1, denote groups 1 and 2 respectively. Then the coefficient of x_2 is the log odds ratio, and we wish to test equality of the population proportions in the two groups via $H_0 : \beta_{(1)2} = 0$ for the model logit $P(Y = 1) = \beta_{(1)1} + \beta_{(1)2}\, x_2$.

For two illustrative values of p_1, and allowing p_2 to vary as 0.01(0.01)0.99 (i.e., 0.01 to 0.99 in steps of 0.01), plotted in Fig. 2.2 is the square of the usual Wald statistic for x_2, which is χ^2_1 under H_0. The LRT statistic $-2\log\lambda$ has the same asymptotic distribution. The glaring feature is the quadratic shape about p_1 of both test statistics. Another feature is that the LRT increases as $|p_2 - p_1|$ increases.

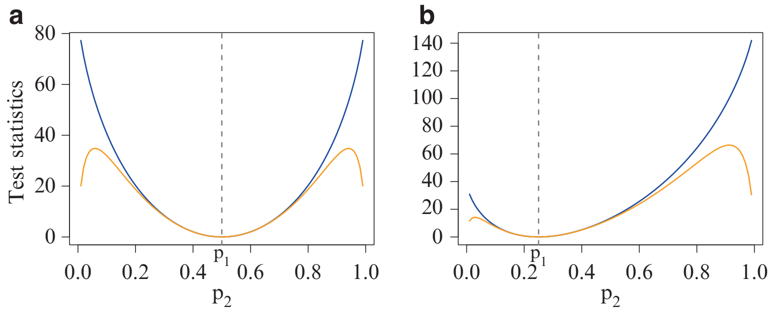

Fig. 2.2 Wald (*orange*) and likelihood ratio test (*blue*) statistics plotted against p_2, for: (**a**) $p_1 = 0.5$, (**b**) $p_1 = 0.25$ (*vertical dashed lines*). Actually, the Wald statistic here is the square of the usual Wald statistic, and $p_2 = 0.01(0.01)0.99$ is discrete. The data follows Hauck and Donner (1977).

This monotonicity is a good thing: there is increasing evidence against the null hypothesis the more the two sample proportions differ. However, the Wald statistic initially increases but then decreases near the boundaries. Thus Wald tests of binomial (and Poisson GLMs) can be highly unreliable, because a moderate p-value indicates no effect or a very large effect. Not only does it show aberrant behaviour, it is also less powerful than a LRT.

2.3.6.3 Problems Due to Complete Separation

It is well-known that multiple maximums of the log-likelihood function cannot occur with logistic regression because ℓ is globally concave, meaning that the function can have at most one maximum (Amemiya, 1985). However, it is possible for the likelihood function to have *no* maximum, in which case the MLE is said to not exist. The problem occurs when there is *complete separation* (Albert and Anderson, 1984). For example, the data used by Allison (2004) is

```
> cs.data <- data.frame(y = rep(0:1, each = 5), x2 = c((-5):(-1), 1:5))
```

These data are plotted in Fig. 2.3 as solid blue circles. Suppose that we fit logit $P(Y = 1|x_2) = \beta_{(1)1} + \beta_{(1)2}\, x_2$. Then it may easily be shown that the log-likelihood function increases as a function of $\beta_{(1)2}$ and that it flattens out as $\beta_{(1)2} \longrightarrow \infty$, i.e., the MLE does not exist. Complete separation occurs when there exists some vector $\boldsymbol{\beta}$ such at $y_i = 1$ whenever $\boldsymbol{\beta}^T \boldsymbol{x}_i > 0$, and $y_i = 0$ whenever $\boldsymbol{\beta}^T \boldsymbol{x}_i < 0$.

There is a related problem called *quasi-complete separation*. This occurs if there exists a $\boldsymbol{\beta}$ such that $\boldsymbol{\beta}^T \boldsymbol{x}_i \geq 0$ whenever $y_i = 1$ and $\boldsymbol{\beta}^T \boldsymbol{x}_i \leq 0$ whenever $y_i = 0$, and when equality holds for at least one observation in each category of the response variable. Adding $(0, 0)$ and $(0, 1)$ to the previous data set will result in quasi-complete separation (Fig. 2.3). Once again, the MLE does not exist.

In practice, quasi-complete separation is far more common than complete separation. It most often occurs when an explanatory variable x_k is a dummy variable, and for one value of x_k, either every observation has $y = 1$ or $y = 0$. In general, consider the 2×2 table of y versus every dichotomous explanatory variable. If there is a zero in any cell, then the MLE will not exist. Convergence failure in

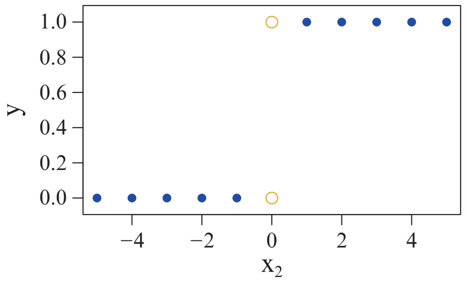

Fig. 2.3 Completely separable data (*blue circles*). Adding the two orange hollow points results in quasi-completely separable data. The logistic regression estimate of the slope will tend to infinity in both cases.

logistic regression is most commonly caused by this. It occurs more often when the sample size is small, however it is certainly possible in large data sets too.

Allison (2004) gives practical advice about a course of action to take if there is complete separation or quasi-complete separation. Actually, the two cases are best considered separately. One possibility is to use bias-reduction. Heinze and Schemper (2002) have shown that this method always yields finite estimates of parameters under complete or quasi-complete separation. However, the result that bias-reduction of $\boldsymbol{\beta}$ gives a finite answer as a by-product is not a sufficient reason for the blind application of the technique.

Of course, the problem can occur regardless of the link function chosen. And it can also occur for the multinomial logit and cumulative link models described in Chap. 14. Bias-reducing techniques have been developed for GLMs, which have the effect that each coefficient in $\widehat{\boldsymbol{\beta}}$ is finite for binary responses; see Sect. 9.4 for an appetizer.

Complete separation is the subject of Altman et al. (2004, Chap.10).

2.4 Univariate Smoothing Methods

2.4.1 *The Classical Smoothing Problem*

Smoothing is a powerful tool for exploratory data analysis, and it allows a *data-driven* approach to the more *model-driven* LM modus operandi. The simplest form of smoothing operates on scatter plot data. As a preliminary (and imperfect) example, consider Fig. 2.5a which is a scatter plot of the proportion of rainbow trout caught from Lake Otamangakau (`lakeO`) between the years 1974 to 1989 by a certain angler. A smoother called a regression spline has been fitted (Fig. 2.5b). The main feature of the data has been picked up, namely it is flat on the LHS, and then is followed by a decrease over the years. However, the smooth $\widehat{f}(x)$ is probably *too* flexible here and it overfits. (Another weakness with this example is that smoothing proportions directly is not a good idea, because the smooth can sometimes assume negative values or values greater than 1. A logistic regression GAM would be recommended instead.)

Four broad categories of smoothers are:

[1.] Regression or series smoothers (polynomial regression, regression splines, P-splines, Fourier regression, filtering, wavelets),

[2.] Smoothing splines (with roughness penalties, e.g., cubic smoothing splines, O-splines, P-splines),
[3.] Local regression (Nadaraya-Watson estimator, kernel smoothers, Lowess, Loess, it generalizes to local likelihood),
[4.] Nearest-neighbour smoothers (running means, running lines, running medians).

We will only be concerned with a small selection of these for two reasons: not all so naturally generalize to the vector-$\boldsymbol{y}$ case, and VGAM currently implements only two methods (regression splines and smoothing O-splines). However, we also consider two other methods, local regression and P-splines, because of their contribution to our understanding of vector smoothing as a whole, and because they have favourable properties, respectively. P-splines ("P" for "penalized") can be considered a hybrid method, therefore they appear twice in the list as they share similarities with regression splines and smoothing splines. Section 4.1.1 gives an overview of how smoothing relates to VGLMs and VGAMs.

Smoothing has many general uses, e.g., data visualization and exploratory data analysis, prediction, derivative estimation (e.g., growth curves, acceleration), and it is used as a building block for many modern statistical techniques, such as in Chap. 4. Examples of each of these uses can be found throughout this book.

For our purposes, the *classical smoothing problem* is to estimate an arbitrary smooth function f based on the model

$$y_i \;=\; f(x_i) + \varepsilon_i, \qquad \varepsilon_i \sim (0, \sigma_i^2) \;\text{ independently,} \tag{2.36}$$

for data $(x_i, y_i, w_i = 1)$, $i = 1, \ldots, n$. If there is no *a priori* function form for f, then it may be estimated by a smoother. They do not impose any particular form on the function apart from being smooth. Since the errors have mean 0, we are modelling the conditional mean $E(Y|x) = f(x)$. Ordinarily, it is usually assumed that all the σ_i are equal. For this, the data sets of Figs. 2.4a and 2.5 appear to violate this assumption. This section describes conventional smoothing methods for the univariate problem (2.36). Indeed, hundreds of journal articles have addressed this problem and its direct extensions.

It is needful to generalize (2.36) to the weighted case so that $\text{Var}(\varepsilon_i) = \sigma_i^2 = \sigma^2 w_i^{-1}$, where the w_i are known and positive, and σ is unknown and may be estimated. This can be written $\text{Var}(\boldsymbol{\varepsilon}) = \sigma^2\, \mathbf{W}^{-1}$ where $\mathbf{W} = \text{diag}(w_1, \ldots, w_n) = \boldsymbol{\Sigma}^{-1}$. For example, in the `lakeO` example, one might assign $w_i = \{y_i(1-y_i)/N_i\}^{-1}$ where $N_i = \texttt{total.fish}_i$ because y_i is a sample proportion (however, in this case, some $y_i = 1$, which is problematic).

Without loss of generality, let the data be ordered so that $x_1 < x_2 < \cdots < x_n$. Consider the unweighted case of $w_i = 1$ for all i. The fundamental idea behind all smoothing methods is the concept of a *neighbourhood*. Here, only observations whose x_i values are 'close' (are neighbours) to a *target point* x_0, say, are used to estimate f at x_0, and the more closer x_i is to x_0, the more influence or weight that (x_i, y_i) has for $\widehat{f}(x_0)$. As an extreme case, observation (x_1, y_1) has the least effect on $\widehat{f}(x_n)$ because x_1 is the furthest away from x_n. This concept is especially easy to see for local regression, e.g., in Fig. 2.13 the shaded region denotes the effective neighbourhood, and the kernel function at the bottom provides the relative weights given to the (x_i, y_i). Rather than 'neighbourhood', many writers use the word *window* to describe the *localness* idea because observations lying outside the window are effectively ignored. This hypothetical window glides along the x-axis to estimate the entire f. The window may or may not have distinct sides depending on whether the weights are strictly zero past a certain distance away from the target point. In Table 2.2, all but one kernel function vanishes beyond a certain distance away from its centre, hence such 'windows' have distinct sides. The window of Fig. 2.13 has blurry edges because the Gaussian kernel function is strictly positive.

We shall see later that the size of the window or neighbourhood about a target point x_0 is crucial because it controls how smooth or wiggly the smooth is. It is fundamentally related to bias and variance. For example, a very large neighbourhood that effectively includes all the data corresponds to little or no smoothing at all, and this has relatively little variance but much bias.

While kernel function smoothing methods are probably the easiest to understand, other smoothing methods such as splines are motivated completely differently, however its asymptotic properties can be shown to be based on the neighbourhood idea, e.g., Sect. 2.4.7.3 shows that, under certain conditions, a cubic smoothing spline operates like a kernel function smoother.

This section describes a few common methods for fitting the classical smoothing problem (2.36). The purpose is to lay a foundation for methods that apply to the vector $\boldsymbol{y}$ case (Sect. 4.2). Some books covering this large topic can be found in the bibliographic notes.

The reader should be aware that this section largely adopts commonly used notation from the smoothing literature, and consequently there is some recyling of notation, e.g., K denotes the number of knots for splines, as well as the kernel function for local smoothers. This however should not present any severe problems because these topics are quite separate and their context is easily grasped.

2.4.2 Polynomial Regression

Polynomial regression is a common technique that involves fitting polynomial functions of each x_k in order to provide more flexibility than the usual linear $\beta_k x_k$ term. One reason for its widespread use is that polynomials are easy to work with in just about every way—mathematically, computationally and they have high interpretability for low degrees. For (2.36), one may use an S formula having the term `poly(x2, degree)` for `degree` $= 1, 2, \ldots,$ else explicit terms such as `I(x2^2)` (or `poly(x2, degree, raw = TRUE)`). The former has the advantage of using orthogonal polynomials which are numerically stable, but at the expense of having coefficients that are not so interpretable.

The Weierstrass approximation theorem, which states that every continuous function on a closed interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial, might lead us to believe that fitting a sufficiently high order polynomial will be a good idea for estimating most fs in general, however this is not the case. The reasons include the following.

- Polynomials are not very local but have a global nature. Their derivatives are continuous functions for all orders. For example, a small perturbation on the LHS of the curve may result in a large change on the RHS. Consequently the concept of a local neighbourhood does not really exist. Polynomials are thus flexible but not flexible enough. This can be seen in Fig. 2.4a; polynomials of degree 1–4 are unable to conform to the trend (which is admittedly complex).

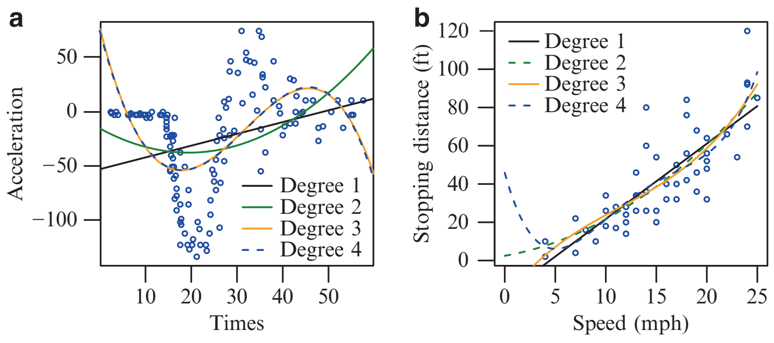

Fig. 2.4 Polynomials of degree 1–4 fitted to two data sets. (**a**) `mcycles` from MASS. (**b**) `cars` from datasets.

The cubic and quartics are almost indistinguishable. It is left to the reader to confirm that polynomials up to the 10th degree do not offer much improvement (Ex. 2.17).

- They usually have edge effects: they do not model the boundaries well, especially if the degree of the polynomial is high. This results in significant bias in regions of the x-space. Figure 2.4b illustrates this. The trend should be monotonically increasing, and if a polynomial of degree that is too high is fitted then often the boundaries are not modelled well. Here, while the linear and quadratic functions appear well-behaved over the range of x, the quartic at the LHS corner curls upwards, and therefore would be dangerous for prediction.
- They are sensitive to outliers and high-leverage points.
- The polynomial degree is discrete rather than continuous.

It is probably a safe general recommendation that one should avoid fitting quartics or higher, and even fitting a cubic should be done cautiously and with trepidation.

2.4.3 Regression Splines

Usually a better alternative to polynomial regression is the use of *regression splines*. These are *piecewise* polynomials, hence the neighbourhood concept is built in directly. Each piece, usually of low degree, is defined on an x-region that is delimited by *knots* (or *breakpoints*). The (x, y) positions where each pair of *segments* join are called *joints*. The more knots, the more flexible the family of curves become. It is customary to force the piecewise-polynomials to join smoothly at the knots, e.g., a popular choice called *cubic splines* are piecewise-cubic polynomials with continuous zeroth, first and second derivatives. By forcing the first few derivatives to be continuous at the knots, the entire curve has the appearance of one nice smooth curve. Using splines of degree > 3 seldom confers any additional benefit. Figures 2.5b and 2.6 are examples of cubic regression splines. The coloured segments of Fig. 2.6 join up at the joints, and the vertical lines mark the knots.

The word ‘spline’ comes from a thin flexible strip used by engineers and architects in the pre-computer days to construct ship hulls and the aerofoils of wings. Splines were attached to important positions on a 2-dimensional plan (e.g., floor of a design loft or on enlarged graph paper) using lead weights called “ducks”

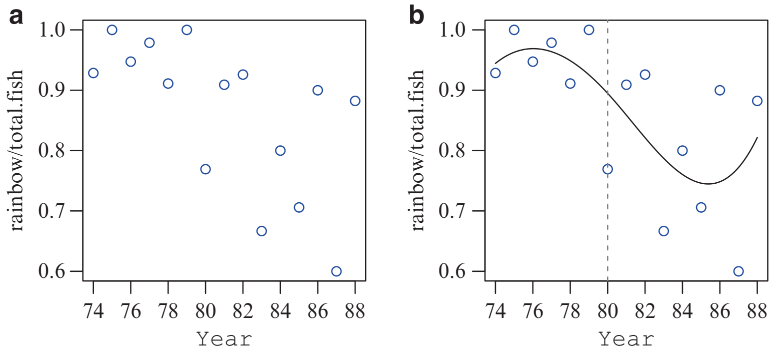

Fig. 2.5 (**a**) Proportion of fish caught that are rainbow trout from Lake Otamangakau (`lakeO`) caught by an angler who frequented the spot. The variable `Year` is `year-1900`. (**b**) Smoothing the same data with a cubic regression spline (truncated power basis) with one knot located at the year 1980. A boundary effect on the RHS is evident.

and then released. The resting shapes assumed by the splines would minimize the strain energy according to some calculus of variations criterion. Splines were used by ancient Greek mathematicians (including Diocles) for drawing curves in diagrams (e.g., conic sections). In more modern times, I. J. Schoenberg is attributed to be the first to use 'splines' in the mathematical literature, and is known as the father of splines. The physical meaning of splines is especially relevant to the smoothing spline (Sect. 2.4.4), where it is related to curvature and Hooke's Law for elastic bodies such as springs.

Regression splines are one example of multiple regression on a family of *basis vectors*. A simpler example is polynomial regression where the set $\mathcal{S} = \{1, x, x^2, \ldots, x^r\}$ form the usual basis of polynomials of degree r. We say $\mathcal{S}$ *spans* this function space. There are two common bases for cubic splines:

1. *Truncated power series* These are easy to understand, but are not recommended in practice because they may be ill-conditioned. Some details are in Sect. 2.4.3.1.
2. *B-splines* These are more complex but are used in practice because they are numerically more stable. The functions `bs()` and `ns()` are two implementations of B-splines. Some details are in Sect. 2.4.3.2. Called "B-splines" by Schoenberg, "B" is usually taken to stand for "basic" or "basis", and for some others, "beautiful".

From the glossary, a function $f \in \mathcal{C}^k[a,b]$ if derivatives $f', f'', \ldots, f^{(k)}$ all exist and are continuous in an interval $[a,b]$. For example, ordinary polynomials $\in \mathcal{C}^k[a,b]$ for all k, a and b, but $|x| \notin \mathcal{C}^1[a,b]$ if $0 \in [a,b]$. Then a *spline of degree* r (some given positive integer) with knots $\xi_1, \ldots, \xi_K$ (such that $a < \xi_1 < \xi_2 < \cdots < \xi_K < b$) is a function $f(x)$ defined over an interval (a,b) if it satisfies the following properties:

(i) for any subinterval (ξ_j, ξ_{j+1}), $f(x)$ is a polynomial of degree r (*order* $r+1$);
(ii) $f \in \mathcal{C}^{r-1}(a,b)$, i.e., $f(x), f'(x), \ldots, f^{(r-1)}(x)$ are continuous;
(iii) the rth derivative of $f(x)$ is a step function with jumps at $\xi_1, \ldots, \xi_K$.

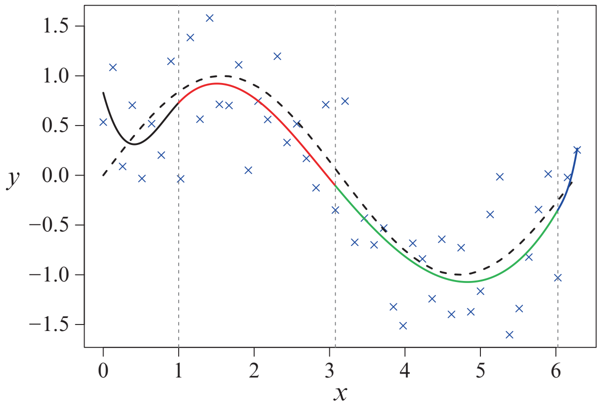

Fig. 2.6 Smoothing some data with a regression spline (B-spline). Each segment of the spline is coloured differently. The term is effectively `bs(x, knots = c(1, 3.08, 6.03))`. The true function is the sine function (*dashed*) and $n = 50$.

The name *cubic spline* is used for the curve arising from the special case of $r = 3$. An example of a plot of the first few derivatives of a cubic spline fit is Fig. 2.11—a step function can be seen in the latter plot.

Regression splines have at least two advantages: they are computationally and statistically simple, and standard parametric inferences are available. The first advantage is partly because they are an LM representation of a smooth function. An example of the second advantage is testing whether a particular knot can be removed and the same polynomial equation used to explain two adjacent segments (by using $H_0 : \beta_j = 0$ in (2.40)). In R, this corresponds to one of the t-test statistics printed by `summary()`. However, they do have drawbacks such as the difficulty choosing the number of knots and their locations, and their smoothness cannot be controlled continuously as a function of a single smoothing parameter.

Regarding the first drawback, in a paper reflecting a lot of experience fitting regression splines, Wold (1974) made the following recommendations for cubic splines:

(i) They should have as few knots as possible, ensuring that a minimum of 4 or 5 observations lie between knot points.
(ii) No more than one stationary point and one inflexion point should fall between two knots (because a cubic is not flexible enough to allow for too many of such points).
(iii) Stationary points should be centred in intervals, and inflexion points should be located near knot points.

These recommendations might be actioned using the `knots` argument of `bs()` and `ns()`. As an illustration, consider Fig. 2.6. Overall, the fit is alright except at the left-hand side boundary. This example illustrates how regression splines may be poor at the boundaries, especially if the knot placement is careless. How `bs()` and `ns()` choose their knots by default is described on p.58.

2.4.3.1 Truncated Power Series

Following the notation of Green and Silverman (1994), a cubic spline may be written as

$$f(x) \;=\; d_s\,(x-\xi_s)^3 + c_s\,(x-\xi_s)^2 + b_s\,(x-\xi_s) + a_s, \qquad \xi_s \le x \le \xi_{s+1}, \tag{2.37}$$

for $s = 0, \dots, K$. For given constants a_s, b_s, c_s and d_s, we define $\xi_0 = a$ and $\xi_{K+1} = b$. The coefficients are interrelated because of the various continuity conditions, e.g., f is continuous at ξ_{s+1} implies

$$d_s\,(\xi_{s+1} - \xi_s)^3 + c_s\,(\xi_{s+1} - \xi_s)^2 + b_s\,(\xi_{s+1} - \xi_s) + a_s \;=\; a_{s+1} \tag{2.38}$$

for $s = 0, \dots, K-1$. Thus there are $4(K+1) - 3K = K + 4$ parameters. The *truncated power series basis* for a cubic spline with K knots is

$$\{1,\ x,\ x^2,\ x^3,\ (x-\xi_1)_+^3, \dots, (x-\xi_K)_+^3\} \tag{2.39}$$

where $u_+ = \max(0, u)$ is the positive part of u. Figure 2.7 gives an example of these functions with $\xi_k = k$ for $k = 1, \dots, 5$. When x is large the curves $(x - \xi_k)_+^3$ become almost vertical and parallel, therefore ill-conditioning occurs.

With (2.39) we can express the spline as

$$f(x) \;=\; \beta_1 + \beta_2\,x + \beta_3\,x^2 + \beta_4\,x^3 + \sum_{s=1}^{K} \beta_{4+s}\,(x - \xi_s)_+^3 \,. \tag{2.40}$$

As an example, here is the essential code behind the `lakeO` example of Fig. 2.5:

```
> Pos <- function(x) pmax(x, 0)  # Same as ifelse(x > 0, x, 0)
> lakeO <- transform(lakeO, Year = year - 1900)  # Because of ill-conditioning
> knot <- 80  # For the year 1980; a prespecified knot
> fit.trout <- lm(rainbow / total.fish ~ Year + I(Year^2) + I(Year^3) +
                  I(Pos(Year-knot)^3), data = lakeO)
> model.matrix(fit.trout)

     (Intercept) Year I(Year^2) I(Year^3) I(Pos(Year - knot)^3)
  1            1   74      5476    405224                     0
  2            1   75      5625    421875                     0
  3            1   76      5776    438976                     0
  4            1   77      5929    456533                     0
  5            1   78      6084    474552                     0
  6            1   79      6241    493039                     0
  7            1   80      6400    512000                     0
  8            1   81      6561    531441                     1
  9            1   82      6724    551368                     8
  10           1   83      6889    571787                    27
  11           1   84      7056    592704                    64
  12           1   85      7225    614125                   125
  13           1   86      7396    636056                   216
  14           1   87      7569    658503                   343
  15           1   88      7744    681472                   512
  attr(,"assign")
  [1] 0 1 2 3 4
```

The variable `Year = year-1900` has been used to ameliorate the ill-conditioning, e.g., 1974^3 and 1988^3 are both large numbers, and are treated almost as having the same value since the computations are performed using finite arithmetic. The example reflects the recommendation that the truncated power series is unsuitable for general use because the model matrices may be ill-conditioned (the columns almost linearly dependent). In general, a B-spline basis is superior.

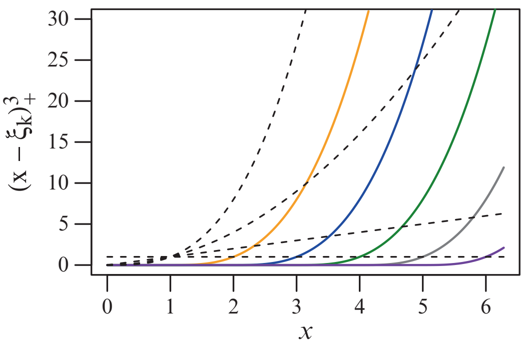

Fig. 2.7 Truncated power series basis for cubic splines (2.39). The *black dashed lines* are $1, x, x^2, x^3$. The *coloured solid lines* are $(x-\xi_k)^3_+$ for knots $\xi_1 = 1, \xi_2 = 2, \xi_3 = 3, \xi_4 = 4$ and $\xi_5 = 5$.

2.4.3.2 B-Splines

The reason why B-splines are used is mainly computational: their very good numerical properties arise from the fact that B-splines have minimal support. That is, $B_{s,q}(x) > 0$ for x belonging to the smallest overall region defined by the knots. With minimal support, the concept of a neighbourhood is made computationally more stable because the coefficient of one B-spline is related to the fewest number of coefficients associated with the other B-splines, i.e., the amount of overlap is minimal. This can be seen in Fig. 2.8: a B-spline of order q consists of q segments only. Hence a B-spline on the LHS of a scatter plot is only concerned with data around the neighbourhood there. In contrast, the value $x = 6$ in Fig. 2.7 is associated with all the truncated power series basis functions.

It is convenient to consider splines of a general order, Q say. Some special cases are as follows.

$Q = 1$: These are similar to a shifted unit rectangle function or boxcar function.
$Q = 2$: *Linear spline* which has continuous derivatives up to order $Q - 2 = 0$ at the knots—i.e., the function is continuous and is piecewise-linear. In fact, it is a scaled density of a triangle distribution (Table 12.10).
$Q = 3$: *Quadratic spline* (*parabolic spline*) which has continuous derivatives up to order $Q - 2 = 1$ at the knots.
$Q = 4$: *Cubic spline*, these are very popular, and have been described as the lowest-order spline for which the discontinuities in the $f^{(Q-1)}(\xi_s)$ are imperceptible (Hastie et al., 2009).

Let ξ_s for $s = 1, \ldots, K$, be K *interior knots*, and let ξ_0 and ξ_{K+1} be the two *boundary knots*. Then we can augment these knots with $2Q$ others to obtain a vector $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{K+2Q})^T$ satisfying the following inequality:

$$\tau_1 \le \tau_2 \le \cdots \le \tau_Q \quad \le \xi_0 \tag{2.41}$$
$$< \xi_1 \le \cdots \le \xi_K \tag{2.42}$$
$$< \xi_{K+1} \le \tau_{K+Q+1} \le \cdots \le \tau_{K+2Q}, \tag{2.43}$$

where $\tau_{Q+s} = \xi_s$ for $s = 1, \ldots, K$. Usually $\tau_1 = \cdots = \tau_Q = \xi_0$ and $\tau_{K+Q+1} = \cdots = \tau_{K+2Q} = \xi_{K+1}$ is chosen.

Denote by $B_{s,q}(x)$ the sth B-spline basis function of order q (degree $q-1$) for the knot sequence $\boldsymbol{\tau}$ for $q = 1, \ldots, Q$. They are defined recursively as follows (de Boor, 2001):

(1) For $s = 1, \ldots, K + 2Q - 1$,

$$B_{s,1}(x) \;=\; \begin{cases} 1, & \tau_s \le x < \tau_{s+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.44}$$

(2) Then for $s = 1, \ldots, K + 2Q - q$ and $q > 1$,

$$\begin{aligned} B_{s,q}(x) &= \omega_{s,q}\, B_{s,q-1}(x) + (1 - \omega_{s+1,q})\, B_{s+1,q-1}(x) &(2.45)\\ &= \frac{x - \tau_s}{\tau_{s+q-1} - \tau_s}\, B_{s,q-1}(x) + \frac{\tau_{s+q} - x}{\tau_{s+q} - \tau_{s+1}}\, B_{s+1,q-1}(x), &(2.46) \end{aligned}$$

where $\omega_{s,q} \equiv (x - \tau_s)/(\tau_{s+q-1} - \tau_s)$ for $\tau_{s+q-1} > \tau_s$, while $\omega_{s,q} \equiv 0$ if $\tau_{s+q-1} = \tau_s$. These may be computed using stable and efficient recursive algorithms. Note that $B_{s,q}$ only depends on the $q+1$ knots $\tau_s, \ldots, \tau_{s+q}$, and vanishes outside the interval $[\tau_s, \tau_{s+q})$ and is positive in its interior. If $\tau_s = \tau_{s+q}$, then $B_{s,q} = 0$.

Thus with $Q = 4$, $B_{s,4}$ (for $s = 1, \ldots, K + 4$) are the $K + 4$ cubic B-spline basis functions for $\boldsymbol{\tau}$.

Some B-splines of orders 1 to 4 are plotted in Fig. 2.8. Essentially, the code is

```
knots <- c(1:3, 5, 7, 8, 10)  # Interior knots
x.vector <- seq(0, 11, by = 0.01)
for (ord in 1:4) {
  B.matrix <- bs(x = x.vector, degree = ord-1, knots = knots, intercept = TRUE)
  matplot(x.vector, B.matrix, type = "l")
}
```

The significance of the argument `intercept` is due to the $B_{s,Q}$ in (2.46) including the intercept because $\sum_{s=1}^{K+Q} B_{s,Q}(x) = 1$ for $x \in [\xi_0, \xi_{K+1}]$. Function `bs()` has `intercept = FALSE` as the default, because usually it is called within an S formula that has an intercept by default. Figure 2.10 shows B-spline basis functions corresponding to a regression spline LM fitted without an intercept term and with a `bs(x, intercept = TRUE)`-type term. Also, note however that `bs()` presently does not accept a value of `degree = 0`, hence the first case `ord = 1` might be computed as follows (it is assumed that `intercept = TRUE`).

```
  allknots <- sort(c(Boundary.knots, knots))
  B1.matrix <- matrix(0, length(x), length(knots) + intercept)
  for (s in 1:(length(allknots)-1))
    B1.matrix[, s] <- as.numeric(allknots[s] <= x & x < allknots[s+1])  # 0 or 1
```

Here are some additional notes.

1.
```
> args(bs)

  function (x, df = NULL, knots = NULL, degree = 3, intercept = FALSE,
      Boundary.knots = range(x))
  NULL
```

If argument `knots` is supplied, then the function returns a matrix of dimension `c(length(x), df = length(knots) + degree + intercept)`. Alternatively, the second dimension may be inputted directly by the `df` argument.

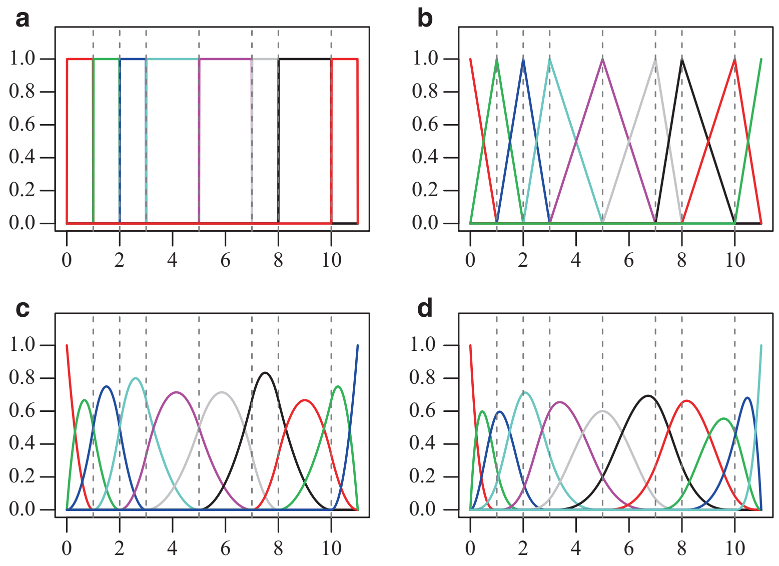

Fig. 2.8 B-splines of order 1–4 ((**a**)–(**d**)), where the interior knots are denoted by *vertical lines*. The boundary knots are at 0 and 11. The basis functions have been plotted from left to right.

It can be seen from the argument `Boundary.knots` that $\xi_0 = \min(x_i)$ and $\xi_{K+1} = \max(x_i)$. By default, the internal knots selected by `bs()` and `ns()` are of the form `quantile(x.inside, probs)` with equally spaced `probs` values. In fact, if `nIknots` is the number of internal knots, then `probs = (1:nIknots)/(nIknots + 1)`, and `x.inside` are the `x` values inside the interval described by the 2-vector `Boundary.knots`.

Note that predicting `bs()` outside the boundary knots is not recommended, because $B_{s,Q}(x)$ is not well defined outside of $[\xi_0, \xi_{K+1}]$. In fact a warning is issued if this is attempted.

2. A well-known type of cubic spline on $[\xi_0, \xi_{K+1}]$ called a *natural cubic spline* (NCS) has second and third derivatives, which are zero at ξ_0 and ξ_{K+1}:

$$f''(\xi_0) \;=\; f'''(\xi_0) \;=\; f''(\xi_{K+1}) \;=\; f'''(\xi_{K+1}) \;=\; 0. \qquad (2.47)$$

These are called the *natural boundary conditions*. NCSs are implemented by `ns()`, which has defaults

```
> args(ns)

  function (x, df = NULL, knots = NULL, intercept = FALSE,
      Boundary.knots = range(x))
  NULL
```

The lack of a `degree` argument is due to only *cubic* NCSs being implemented. Given knots $\xi_1, \ldots, \xi_K$, an NCS is linear on $(-\infty, \xi_0]$ and $[\xi_{K+1}, \infty)$. Function `ns()` has $K + 2$ parameters including the intercept because $K + 2 = (K+4) - 2 \times 2 + 2$: each boundary constraint in (2.47) deducts one parameter from the total number, and there are two extra knots at ξ_0 and ξ_{K+1}.

In practice, often there is not a huge difference between `bs()` and `ns()` terms, when they are calibrated to be as similar to each other as possible. However,

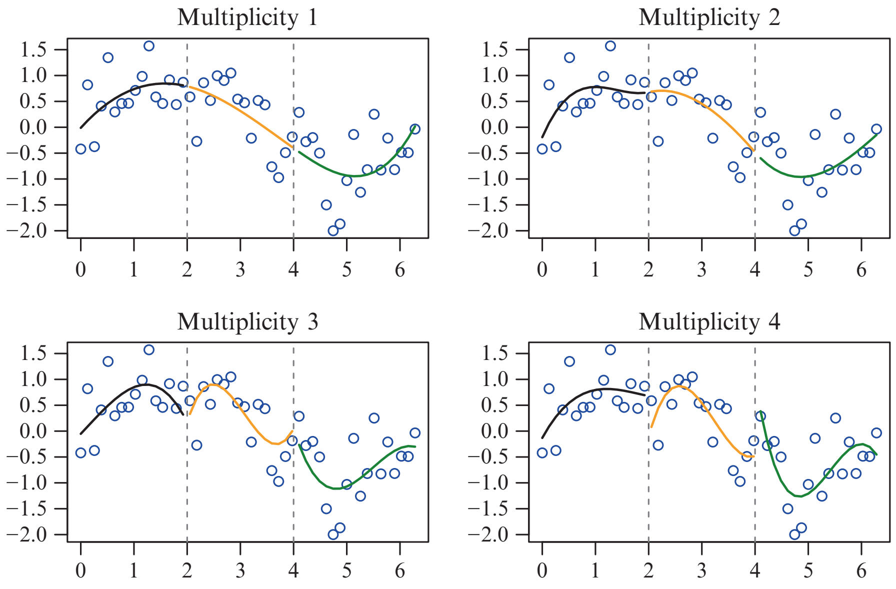

Fig. 2.9 Smoothing some data with cubic regression splines with knots of varying multiplicities. The knots are at $x = 2$ and 4.

usually the natural boundary constraints means `ns()` behaves better at the edges than `bs()`. The `ns()` function is also a double-edged sword compared to `bs()`, because it is defined for all x, but as with all smoothers, prediction beyond the range of the data poses more potential danger.

3. It is possible to move adjacent knots closer and closer together until they coincide. We then say the set of distinct knots has varying multiplicities. Then we need to define $B_{s,1} \equiv 0$ if $\tau_s = \tau_{s+1}$, and use the maxim *anything times zero is zero* in (2.46) to avoid division by 0. It transpires that if a knot is duplicated then it loses one continuous derivative for each new knot there. It can be shown that *the number of continuity conditions at a knot ξ_s, plus the multiplicity of knots at ξ_s, equals Q*. The effect of this important formula can be seen in Fig. 2.9: the cubic spline ($Q = 4$) with a knot of multiplicity m means that only $f^{(0)}, f^{(1)}, \ldots, f^{(Q-m-1)}$ exist there.
4. Although *safe prediction* in R will be sufficient for most users, it will not handle nested expressions involving data-dependent functions such as `I(bs(x))` and `poly(scale(x), 2)`. In such cases, *smart prediction* will work; see Sect. 18.6 for details.

2.4.4 Smoothing Splines

For regression splines, the user typically controls the flexibility of the smoother by selecting a small number of basis functions, e.g., by assigning the `df` argument an appropriate value that is less than 10, say. In contrast, the regularization approach to smoothing is to start off with many basis functions (e.g., n of them) and penalize some characteristic of these basis functions in order to control the flexibility of the fit. A popular example of this approach is the *cubic smoothing spline*. These are defined as minimizers of the objective function

$$S(f) \;=\; \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \int_a^b \{f''(x)\}^2 \, dx, \tag{2.48}$$

over a space of "smooth" functions. In fact, it is an infinite-dimensional space of functions known as $\mathcal{W}_2^2[a,b]$ (a *Sobolev space* of order 2 on $[a,b]$ described in Table A.3). Here, $a < x_1 < \cdots < x_n < b$ for some a and b is again assumed, and the *smoothing parameter* satisfies $\lambda \geq 0$.

The first term of $S(f)$ penalizes lack-of-fit since it is a residual sum of squares. The second term penalizes the wiggliness or lack of smoothness, e.g., the integral equals zero for constant and linear functions. These two opposing quantities are balanced with each other by λ. Larger values of λ produce more smooth curves, indeed, as $\lambda \to \infty$, $f''(x) \to 0$ and the solution becomes the least squares line. The other extreme is as $\lambda \to 0^+$, and the solution tends to a twice-differentiable function that interpolates the data (x_i, y_i). These two extremes are often unacceptable as a solution, so it is surmised that there is some λ value which balances the two adequately. The quantity (2.48) fits into the *"penalty function"* approach described in Sect. 1.5.1, and is expounded by Green and Silverman (1994) specifically for splines.

Let $\boldsymbol{\Sigma} = \mathbf{W}^{-1} = \operatorname{diag}(w_1^{-1}, \ldots, w_n^{-1})^T$ to handle known prior weights as in the weighted classical smoothing problem (2.36). Then the *penalized least squares* criterion can be written

$$S(f) \;=\; (\boldsymbol{y} - \boldsymbol{f})^T \; \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{f}) + \lambda \boldsymbol{f}^T \mathbf{K} \boldsymbol{f} \tag{2.49}$$

where $\mathbf{K}$ is a roughness penalty matrix described below. Setting its derivative with respect to $\boldsymbol{f}$ to $\mathbf{0}$ yields the solution

$$\widehat{\boldsymbol{f}} \;=\; \mathbf{S}(\lambda)\, \boldsymbol{y} \tag{2.50}$$

where $\mathbf{S}(\lambda) = (\mathbf{I}_n + \lambda \boldsymbol{\Sigma}\mathbf{K})^{-1}$ is known as the *influence* or *smoother matrix.* We shall see that it has properties similar to the LM hat matrix $\boldsymbol{\mathcal{H}}$ (2.10).

Here are some notes.

1. One can select λ by trial-and-error such as by eye, however, more objective methods such as cross-validation are described below.
2. Following on from the description of a spline as a thin wooden strip in Sect. 2.4.3, one justification for the penalty term of $S(f)$ is that the energy to bend it is proportional to $\int_a^b$ curvature^2 with[1] respect to arc length, which is approximately proportional to $\int_a^b f''(t)^2 \, dt$. From Hooke's Law, springs exert an energy that is proportional to $\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$. Hence (2.48) does have a real physical meaning.

[1] The *curvature* of a curve $y = f(x)$ is $|f''(x)| \left\{1 + [f'(x)]^2\right\}^{-3/2}$. If the { } term is dropped (because the assumption $|f'(x)| \ll 1$ is almost always made in physics and engineering), then $|f''(x)|$ is left as an approximation to the curvature. In natural cubic spline interpolation, we are finding a curve with minimal (approximate) curvature over an interval, for the quantity $\int [f''(x)]^2 \, dx$ is being minimized.

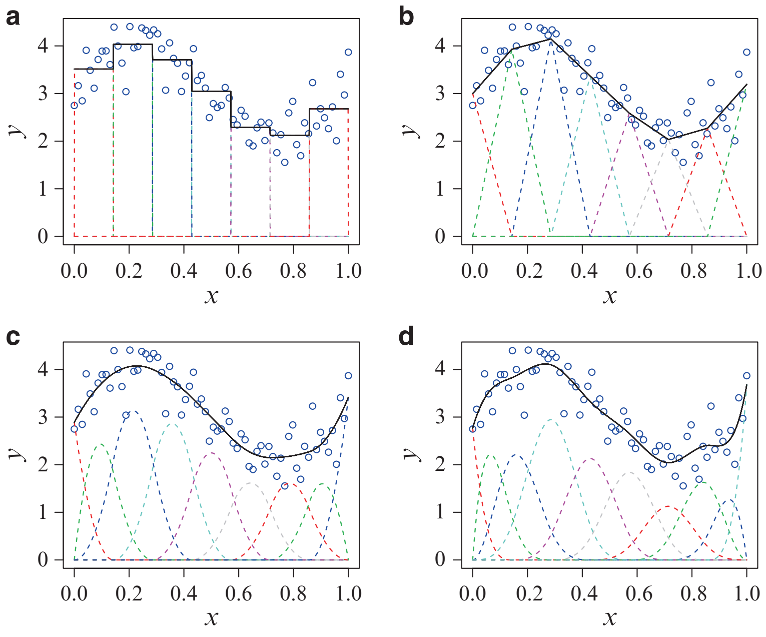

Fig. 2.10 (**a**)–(**d**) Linear combinations of B-splines of degrees 0–3 fitted to some scatter plot data; the formula is similar to (2.55). The knots are equally spaced on the unit interval.

3. Importantly, Reinsch (1967) used the calculus of variations to show that the solution of (2.48) is a cubic spline with knots at the distinct values of the x_i (provided that $n \geq 3$ and $\lambda > 0$). Another important result is that if NCSs interpolate the data (x_i, y_i) then they uniquely minimize $\int \{f''\}^2$ over all functions in $\mathcal{W}_2^2[a, b]$ (for $a < x_1 < \cdots < x_n < b$). Both these results are stated in, e.g., Green and Silverman (1994, Thms 2.3, 2.4).
4. Actually, the optimization problem (2.48) derives from minimizing

$$\int_a^b \{f''(x)\}^2\, dx \quad \text{subject to the constraint} \quad \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 \leq A$$

for some A. Then (2.48) arises from applying the Lagrange multipliers technique to this (Reinsch, 1967).
5. As $n \to \infty$, λ should become smaller, consequently some authors replace λ in (2.48) by λ/n.
6. There are alternative regularizations to the penalty of (2.48), e.g.,

$$\int_a^b f'(x)^2\, dx, \tag{2.51}$$

whose solution is a linear spline. In general, using $\int_a^b [f^{(\nu)}(x)]^2\, dx$ produces a spline solution of degree $2\nu - 1$.
7. More generally, a Sobolev space of order m on $[a, b]$ is written $\mathcal{W}_2^m[a, b]$. The case of $m = 2$ corresponds to cubic splines. Absolutely continuity is a stronger

condition than uniform continuity, given the definition of $\mathcal{C}^k[a,b]$ on p.53, it can be shown that

$$\mathcal{C}^2[a,b] \;\subset\; \mathcal{W}_2^2[a,b]. \tag{2.52}$$

As an example, Fig. 2.11a is a plot of a cubic smoothing spline fitted to the `lakeO` data. Only a little nonlinearity is afforded to it. It suggests a gradual decline in the proportion of rainbow trout caught there over time. The first 3 derivatives are also shown in Fig. 2.11b–d, and these become increasingly more jagged. The third derivative is a step function, and the second derivative is a piecewise-linear function.

2.4.4.1 Computation by the Reinsch Algorithm

Cubic smoothing splines may be computed in several ways. All of the following methods except for the first can be efficiently computed in $O(n)$ operations.

1. Direct method (2.50). *Not* recommended because it involves $O(n^3)$ operations due to an order-n matrix inversion.
2. B-splines—this numerically stable method is probably the most commonly used algorithm nowadays, and is implemented in R by splines.
3. Reinsch algorithm—using clever linear algebra, one can transform the problem into a banded system that can be efficiently solved. Green and Silverman (1994, Sect.2.3.3) gives a succinct description and this is summarized even more below. It forms the basis of the Fessler (1991) algorithm for vector splines (Sect. 4.2.1).
4. State-space approach—this is based on Kalman filter computations in time series analysis (Wecker and Ansley, 1983; Kohn and Ansley, 1987).

Elements of the Reinsch (1967) algorithm are as follows. Firstly, it may be shown that the roughness penalty matrix can be expressed[2] as $\mathbf{K} = \lambda \mathbf{Q}\mathbf{T}^{-1}\mathbf{Q}^T$, where $\mathbf{Q}$ is a banded $n \times (n-2)$ matrix and $\mathbf{T}$ is symmetric tridiagonal of order $n-2$. Also, it may be shown that $\mathbf{Q}^T\boldsymbol{f} = \mathbf{T}\boldsymbol{\gamma}$ for some vector $\boldsymbol{\gamma}$. Secondly, starting at (2.50),

$$\boldsymbol{f} \;=\; \left(\mathbf{I}_n + \lambda\,\mathbf{W}^{-1}\,\mathbf{K}\right)^{-1}\boldsymbol{y} \;=\; \left(\mathbf{W} + \lambda\,\mathbf{K}\right)^{-1}\mathbf{W}\,\boldsymbol{y}. \tag{2.53}$$

Then $\boldsymbol{f} = \boldsymbol{y} - \lambda\,\mathbf{W}^{-1}\mathbf{Q}\mathbf{T}^{-1}\mathbf{Q}^T\boldsymbol{f}$ and hence $\boldsymbol{f} = \boldsymbol{y} - \lambda\,\mathbf{W}^{-1}\mathbf{Q}\boldsymbol{\gamma}$. Premultiply both sides by $\mathbf{Q}^T$ and substitute $\mathbf{Q}^T\boldsymbol{f} = \mathbf{T}\boldsymbol{\gamma}$ to give

$$\left(\mathbf{T} + \lambda\,\mathbf{Q}^T\,\mathbf{W}^{-1}\,\mathbf{Q}\right)\gamma \;=\; \mathbf{Q}^T\,\boldsymbol{y}. \tag{2.54}$$

This is the key equation. The LHS is a symmetric positive-definite band matrix with bandwidth 5 (half-bandwidth 3). One can decompose this into the rational Cholesky decomposition $\mathbf{L}\mathbf{D}\mathbf{L}^T$, where $\mathbf{L}$ is a unit lower diagonal band matrix and $\mathbf{D}$ is a diagonal matrix with positive diagonal elements. The matrices $\mathbf{Q}$ and $\mathbf{T}$ can be found in $O(n)$ operations, and hence $\mathbf{L}$ and $\mathbf{D}$ require only linear time for their computation.

[2] Explicitly, letting $h_i = x_{i+1} - x_i$ for $i = 1, \ldots, n-1$, their nonzero elements are: $(\mathbf{T})_{ii} = (h_i + h_{i+1})/3$, $(\mathbf{T})_{i,i-1} = (\mathbf{T})_{i,i+1} = h_i/6$, $(\mathbf{Q})_{ii} = h_i^{-1}$, $(\mathbf{Q})_{i+1,i} = -(h_i^{-1} + h_{i+1}^{-1})$ and $(\mathbf{Q})_{i+2,i} = h_{i+1}^{-1}$.

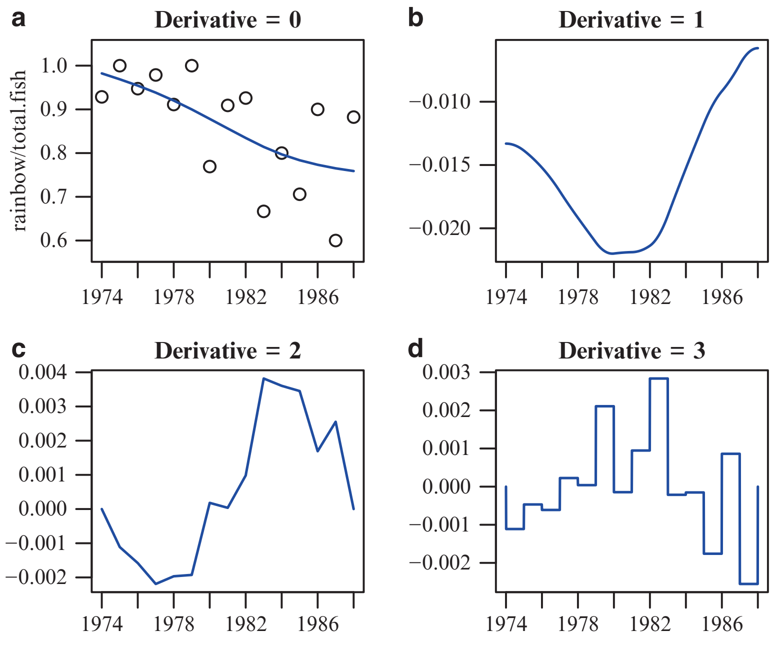

Fig. 2.11 (**a**) Cubic smoothing spline fitted to the proportion of fish caught that are rainbow trout from `lake0`. The x-axis is `year`. The smoother has 1 nonlinear degrees of freedom. (**b**)–(**d**) Derivatives of the smooth of orders 1–3. In contrast, Fig. 2.15 fits a local linear regression to these data.

So the steps are:

(i) compute $\mathbf{Q}^T \boldsymbol{y}$,

(ii) find the non-zero bands of $\left(\mathbf{T} + \lambda\, \mathbf{Q}^T \mathbf{W}^{-1} \mathbf{Q}\right)$ and hence its rational Cholesky decomposition factors $\mathbf{L}$ and $\mathbf{D}$,

(iii) solve $\mathbf{L}\mathbf{D}\mathbf{L}^T \boldsymbol{\gamma} = \mathbf{Q}^T \boldsymbol{y}$,

(iv) compute $\widehat{\boldsymbol{f}} \;=\; \boldsymbol{y} - \lambda\, \mathbf{W}^{-1} \mathbf{Q} \boldsymbol{\gamma}$.

Unfortunately the Reinsch algorithm becomes numerically unstable as n gets very large and/or if the x_i are very unequally spaced. Like regression splines, a more numerically stable algorithm can be devised, based on B-splines.

2.4.4.2 B-Splines

Here, we express $\widehat{f}$ as a linear combination of B-splines like Sect. 2.4.3.2 and Fig. 2.10:

$$\widehat{f}(x) \;=\; \sum_{s=1}^{K+Q} \beta_s \, B_{s,Q}(x) \tag{2.55}$$

so that the elements of the roughness penalty matrix from (2.49) are $(\mathbf{K})_{st} = \int_a^b B''_{s,Q}(x)\, B''_{t,Q}(x)\, dx$. These integrals are not difficult to compute because the integrands are merely quadratics.

2.4.4.3 O-Splines

As stated above, an important property of cubic smoothing splines is that the knots are the (distinct) x_i. However, for large n, having so many knots is overkill. Consequently, O-splines are used to reduced the computational cost by choosing an 'effective' number of knots ($K \ll n$, say) that hopefully results in a fitted curve that does not differ appreciably from the *full-knot* solution. The result has been called a *low-rank* spline smoother (e.g., Ruppert et al., 2003) or *reduced-knot* smoother.

How might the K knots be chosen? Ideally, they should 'mimic' the x_is, hence one technique is to take a simple random sample of them. Another suggestion is to place relatively more knots in regions where f is wiggly as opposed to simple. A good strategy would be to choose quantile-based knots, and another to use equally spaced knots. For these two, it is possible to construct f and distributions of the x_i that cause the other strategy to perform poorly. O-splines use quantile-based knots, whereas P-splines (Sect. 2.4.5) choose equally spaced knots. The former is implemented in the R function `smooth.spline()` and also in VGAM as a whole.

As for the value of K itself, the upper function of Fig. 2.12 is a plot of K versus n used by `smooth.spline()`. As $n \to \infty$, $K = 200 + (n - 3200)^{1/5}$ grows very slowly. To 'fill the space' of the x_is, the software selects the sth knot to be approximately the $s/(K + 1)$th sample quantiles of the unique x_is. (In contrast, P-splines choose equally spaced knots). But $K = n$ for $n \leq 50$ because of the light computational cost. It should be noted that O-splines use the natural boundary constraints (2.47) so that the solution is linear beyond the range of the data. Some more details are given in Wand and Ormerod (2008).

The function `vsmooth.spline()` described in Sect. 4.4.2 also follows a similar idea. However, it reduces K with greater severity because $M > 1$ increases the computational cost quickly as M grows. Currently,

$$K \;=\; \begin{cases} n, & n \leq 40, \\ \lfloor 40 + (n - 400)^{1/4} \rfloor, & n > 40, \end{cases} \tag{2.56}$$

which is the lower function of Fig. 2.12.

Incidentally, the "O" in "O-splines" is due to F. O'Sullivan, the author of a software implementation of the above, named BART, which was written in the mid-1980s. It forms the innards of `smooth.spline()`. By default, this function will implement O-splines, but if argument `all.knots = TRUE` then the full-knot solution to (2.48) will be returned. The early S-PLUS `gam()` function was built on BART, as is `gam()` in gam presently. More details about the O-spline algorithm are given in Sect. 4.2.1.3 for the general M case.

2.4.5 P-Splines

Rather than using smoothing splines, it is more convenient to smooth using the "penalized B-splines" of Eilers and Marx (1996), also known as "P-splines". They are another example of a low-rank smoother and have several compelling advantages. Their solution can be conveniently computed because it involves straightforward linear algebra computations, therefore estimation can proceed in a similar

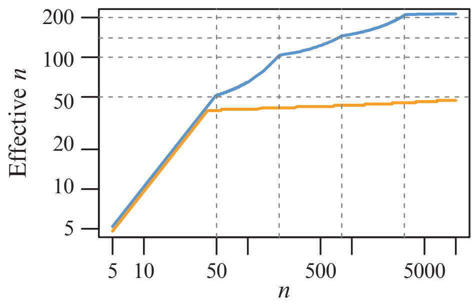

Fig. 2.12 O-splines: number of knots K selected from n unique x_i, for `smooth.spline()` is the top function. The lower function is (2.56) for `vsmooth.spline()`. Both axes are on a logarithmic scale. The top function intersects with the *dashed lines* at $(50, 50)$, $(200, 100)$, $(800, 140)$ and $(3200, 200)$; logarithmic interpolation is used for other n values.

manner to GLMs. There is no need for backfitting (Sect. 4.3) and all functions are estimated simultaneously. Furthermore, inference is straightforward, and automatic smoothing parameter selection is a less daunting problem. The R package mgcv conveys P-splines to the common GAM class plus about a dozen more distributions.

P-splines extend regression splines by penalizing the coefficients of adjacent B-splines. Suppose that $x_i \in [a, b]$ for some simple scatter plot data. Let the regression spline be

$$f(x) \;=\; \sum_{s=1}^{K+Q-1} \beta_s \, B_{s,q}(x), \tag{2.57}$$

where there are $K + 1$ equidistant knots $\xi_s = a + s(b - a)/K$ (for $s = 0, 1, \ldots, K$) in $[a, b]$ (i.e., $K - 1$ internal knots). We can write (2.57) as $\boldsymbol{f} = \mathbf{X}\boldsymbol{\beta}$ where $(\mathbf{X})_{ij} = B_{j,Q}(x_i)$. Then $\boldsymbol{\beta}$ can be estimated by minimizing

$$S(\boldsymbol{\beta}) \;=\; (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{W} (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \, \boldsymbol{\beta}^T \, \mathbf{D}_{[d]}^T \, \mathbf{D}_{[d]} \, \boldsymbol{\beta} \tag{2.58}$$

where $\lambda > 0$ is the smoothing parameter, and $\mathbf{D}_{[d]}$ $((K + Q - 1 - d) \times (K+Q-1))$ is the matrix representation of the dth-order differencing operator Δ^d, e.g., $\Delta^1 \beta_s = \beta_s - \beta_{s-1}$ and $\Delta^2 \beta_s = \Delta(\Delta\beta_s) = \Delta\beta_s - \Delta\beta_{s-1} = \beta_s - \beta_{s-1} - (\beta_{s-1} - \beta_{s-2}) = \beta_s - 2\beta_{s-1} + \beta_{s-2}$. In practice, the values $d = 2$ and 3 are common. In general, the roughness penalty term in (2.58) is $\lambda \sum_{s=d+1}^{K+Q-1} \left(\Delta^d \beta_s\right)^2$, and this penalty may not make sense with non-equidistant knots. The form the $\mathbf{D}_{[d]}^T$ takes on is similar to:

```
> (D_1 <- diff(diag(4)))

       [,1] [,2] [,3] [,4]
  [1,]   -1    1    0    0
  [2,]    0   -1    1    0
  [3,]    0    0   -1    1

> (D_2 <- diff(diff(diag(4))))  # Same as diff(diag(4), diff = 2)

       [,1] [,2] [,3] [,4]
  [1,]    1   -2    1    0
  [2,]    0    1   -2    1
```

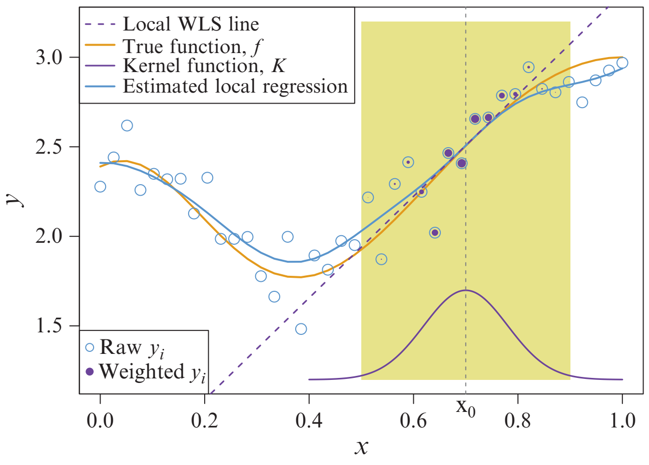

Fig. 2.13 Local linear regression with $n = 40$ points. The kernel weights have been divided by 10 for scaling purposes. The *vertical line* is at the target point $x_0 = 0.7$ so that three *curves/lines* intersect at $(x_0, \widehat{f}(x_0))$. The *shaded region* is effectively the window for computing $\widehat{f}(x_0)$.

It may be seen in (2.58) that $\boldsymbol{\beta}$ appears in both terms. This means that the coefficients controls both the amount of goodness-of-fit and the wiggliness. The solution obtained by setting $\partial S/\partial \boldsymbol{\beta} = \mathbf{0}$ is

$$\widehat{\boldsymbol{\beta}} \;=\; \left(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\,\mathbf{D}_{[d]}^T\mathbf{D}_{[d]}\right)^{-1}\mathbf{X}^T\mathbf{W}\boldsymbol{y}. \qquad (2.59)$$

Then the variance-covariance matrix of $\widehat{\boldsymbol{\beta}}$ is easy: $\mathrm{Var}(\widehat{\boldsymbol{\beta}}) =$

$$\sigma^2\left(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\,\mathbf{D}_{[d]}^T\mathbf{D}_{[d]}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\,\mathbf{D}_{[d]}^T\mathbf{D}_{[d]}\right)^{-1}, \qquad (2.60)$$

and so

$$\mathrm{Var}(\widehat{\boldsymbol{y}}) \;=\; \mathrm{Var}(\mathbf{X}\widehat{\boldsymbol{\beta}}) \;=\;$$
$$\sigma^2\,\mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\,\mathbf{D}_{[d]}^T\mathbf{D}_{[d]}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda\,\mathbf{D}_{[d]}^T\mathbf{D}_{[d]}\right)^{-1}\mathbf{X}^T. \quad (2.61)$$

2.4.6 Local Regression

Local regression refers to a major class of smoothers that includes the Nadaraya-Watson smoother (2.63), local polynomial kernel estimators (2.65), and variants such as Loess and Lowess. No local regression smoother is currently implemented in VGAM, so we describe it here mainly for completeness and for preparation of some theoretical properties in the vector case (Sect. 4.2.2.1). We give scant attention to any practical aspects, and only briefly mention that a popular smoother is `loess()` (Cleveland et al., 1991) and its older variant `lowess()`—see Sect. 2.4.6.5.

Consider the classical smoothing problem (2.36) with $\sigma_i^2 = \sigma^2$ and $w_i = 1$ as related to Fig. 2.13. To estimate $f(x_0)$, one computes a WLS fit to the (x_i, y_i) with weights determined by the distance x_0 is from the x_i. In fact, these weights are $K_h(x_0 - x_i)$ where K is some *kernel function*, h is the positive smoothing parameter known as the *bandwidth*, and

$$K_h(u) \quad = \quad h^{-1} \cdot K\Big(\frac{u}{h}\Big) \tag{2.62}$$

is a scaled version of K that integrates to unity for all h. The bandwidth scales the distance by adjusting the window size in a similar manner that the standard deviation does to a normal distribution. Small/large values of h mean a small/large effective window size about x_0. An h that is too low results in too few observations, therefore is prone to overfit. As $h \to \infty$, the solution becomes an essentially unweighted LS fit (because all weights are equal) to all the data, e.g., $\widehat{f}(x) = \overline{y}$ for all x if a polynomial of degree $r = 0$ is fitted.

Some popular kernel functions are given in Table 2.2 and are plotted in Fig. 2.14. For convenience they possess the following properties:

(i) symmetric,
(ii) have unit area,
(iii) centred at the origin,
(iv) nonincreasing going away from the origin.

Regarding the latter property, apart from the uniform kernel which assigns an equal weight to observations within the window, other kernel functions strictly decrease as the distance from the origin increases. This is called the unimodal property. The significance of the Epanechnikov kernel is that it minimizes the asymptotic mean integrated squared error (2.84).

Given the kernel weights, the WLS fit is a polynomial of degree r (Fig. 2.13), hence the name *local polynomial kernel estimator* is sometimes used. The case $r = 1$ is known as a *local linear regression* or *local linear kernel* smoother. The case $r = 0$ gives the *local constant* or simple *Nadaraya-Watson estimator*

$$\widehat{f}_{nw}(x_0) \quad = \quad \frac{\sum_{i=1}^{n} K\left(\frac{x_0 - x_i}{h}\right) y_i}{\sum_{i=1}^{n} K\left(\frac{x_0 - x_i}{h}\right)} \quad = \quad \sum_{i=1}^{n} \left\{ \frac{K_h(x_0 - x_i)}{\sum_{t=1}^{n} K_h(x_0 - x_t)} \right\} y_i. \tag{2.63}$$

Clearly, it takes a weighted average of the y_i, and more weight is assigned to those x_i that are closer to x_0. The quantities in braces are normalized kernel weights.

More generally, an explicit expression for the rth-degree local polynomial kernel estimator can be obtained as follows. At a target point x, the estimator $\widehat{f}(x; r, h)$ is obtained by fitting the polynomial $\beta_1 + \beta_2(\cdot - x) + \cdots + \beta_{r+1}(\cdot - x)^r$ to the (x_i, y_i) using weighted least squares with kernel weights $K_h(x_i - x)$. The value of $\widehat{f}(x; r, h)$ is the intercept $\widehat{\beta}_1$ because of the centring, where $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_1, \ldots, \widehat{\beta}_{r+1})^T$ minimizes the WLS criterion

$$\sum_{i=1}^{n} \left\{ y_i - \beta_1 - \beta_2(x_i - x) - \cdots - \beta_{r+1}(x_i - x)^r \right\}^2 K_h(x_i - x). \tag{2.64}$$

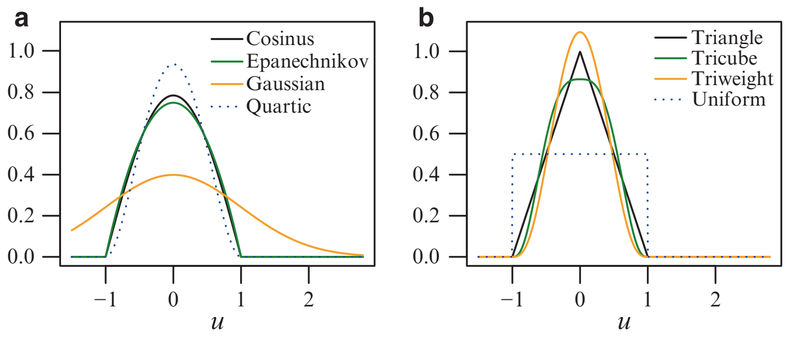

Fig. 2.14 Kernel functions from Table 2.2. All but one has compact support.

Table 2.2 Some popular kernel functions for local regression. All but one is defined on $[-1, 1]$, and the quartic is also known as the biweight. They are graphed in Fig. 2.14.

Kernel	$K(u)$	Kernel	$K(u)$
Cosinus	$\frac{\pi}{4}\cos(\pi u/2) \cdot I(\lvert u\rvert \leq 1)$	Triangle	$(1 - \lvert u\rvert) \cdot I(\lvert u\rvert \leq 1)$
Epanechnikov	$\frac{3}{4}(1 - u^2) \cdot I(\lvert u\rvert \leq 1)$	Tricube	$\frac{70}{81}(1 - \lvert u\rvert^3)^3 \cdot I(\lvert u\rvert \leq 1)$
Gaussian	$\phi(u) = \exp(-\frac{1}{2}u^2)/\sqrt{2\pi}$	Triweight	$\frac{35}{32}(1 - u^2)^3 \cdot I(\lvert u\rvert \leq 1)$
Quartic	$\frac{15}{16}(1 - u^2)^2 \cdot I(\lvert u\rvert \leq 1)$	Uniform	$\frac{1}{2} \cdot I(\lvert u\rvert \leq 1)$

The centring about x is for mathematical convenience. The solution is

$$\widehat{\boldsymbol{\beta}}_x \;=\; \left(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W}_x \, \boldsymbol{y} \tag{2.65}$$

where $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, $\mathbf{W}_x = \text{diag}(K_h(x_1 - x), \ldots, K_h(x_n - x))$ and the model matrix specific to x is

$$\mathbf{X}_x \;=\; \begin{pmatrix} 1 & (x_1 - x) & \ldots & (x_1 - x)^r \\ \vdots & \vdots & & \vdots \\ 1 & (x_n - x) & \ldots & (x_n - x)^r \end{pmatrix}, \tag{2.66}$$

which is $n \times (r + 1)$. Since the estimator of $f(x)$ is the intercept, we have

$$\widehat{f}(x; r, h) \;=\; \boldsymbol{e}_1^T \widehat{\boldsymbol{\beta}}_x \;=\; \boldsymbol{e}_1^T \left(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W}_x \, \boldsymbol{y}. \tag{2.67}$$

Substituting $r = 0$ into this yields the Nadaraya-Watson estimator (2.63). Similarly, the local linear estimator ($r = 1$) can be written as

$$\widehat{f}(x; 1, h) \;=\; n^{-1} \sum_{i=1}^{n} \frac{\{\widehat{s}_2(x; h) - \widehat{s}_1(x; h) \cdot (x_i - x)\} \; K_h(x_i - x) \; y_i}{\widehat{s}_2(x; h) \; \widehat{s}_0(x; h) - \widehat{s}_1(x; h)^2} \tag{2.68}$$

where

$$\widehat{s}_r(x; h) \;=\; n^{-1} \sum_{i=1}^{n} (x_i - x)^r \; K_h(x_i - x). \tag{2.69}$$

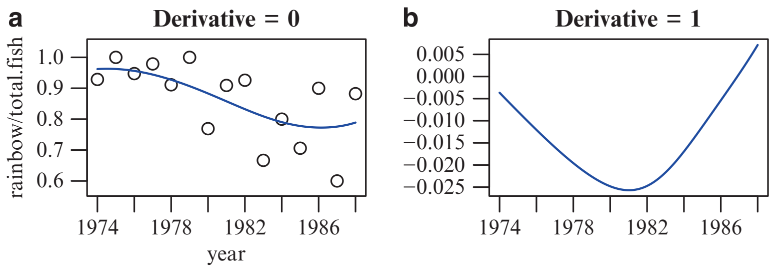

Fig. 2.15 (**a**) Local linear regression fitted to the proportion of fish caught that are rainbow trout from `lakeO`. The smoother has a bandwidth of $h = 2.5$ and uses the Gaussian kernel function. (**b**) $\widehat{f}'(x)$. In contrast, Fig. 2.11 fits a cubic smoothing spline to these data.

2.4.6.1 Derivative Estimation

Sometimes the first or second derivative of f is of more interest than f itself. For example, in the study of human growth curves of height as a function of age, the "speed" and "acceleration" of growth have important biological significance.

The νth derivative of f is easily estimated from above. As

$$\frac{d^{\nu}}{du^{\nu}} \; \beta_{p+1} \cdot (u-x)^{p} \bigg|_{u=x} \;=\; \begin{cases} p!\,\beta_{p+1}, & \nu = p, \\ 0, & \text{otherwise}, \end{cases}$$

we simply extract the $(\nu + 1)$th coefficient of $\boldsymbol{\beta}_x$ to give the estimate

$$\widehat{f}^{(\nu)}(x; r, h) \;=\; \nu!\; \boldsymbol{e}_{\nu+1}^{T} \left(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W}_x \boldsymbol{y} \tag{2.70}$$

for all $\nu = 0, \ldots, r$. Of course, (2.67) is a special case of this. Note that $\widehat{f}^{(\nu)}(x; r, h)$ is not in general equal to the νth derivative of $\widehat{f}(x; r, h)$.

As an example, Fig. 2.15 fits a local linear regression to the `lakeO` data. In contrast, Fig. 2.11 fits a cubic smoothing spline to these data. Both estimates of $f(x)$ and $f'(x)$ are similar for the two smoothers, which is not surprising for this small simple scatter plot. As $r = 1$, $\widehat{f''}(x)$ is not available through (2.70), however it might be estimated using a local quadratic polynomial kernel estimator.

2.4.6.2 Bias and Variance †

Compared to some other smoothers, the large-sample properties of local polynomial kernel estimators are readily derived. This section demonstrates some of this ability. From (2.65), write

$$\widehat{\boldsymbol{\beta}}_x \;=\; \boldsymbol{\Delta}_x^{-1} \boldsymbol{\theta}_x \tag{2.71}$$

where $\boldsymbol{\Delta}_x = n^{-1}\,\mathbf{X}_x^T\mathbf{W}_x\,\mathbf{X}_x$ and $\boldsymbol{\theta}_x = n^{-1}\,\mathbf{X}_x^T\mathbf{W}_x\,\boldsymbol{y}$, so that their means are finite. If $\mathrm{Var}(y_i) = \sigma^2$ then it follows from (2.65) that

$$E\left[\widehat{\boldsymbol{\beta}}_x\right] = \left(\mathbf{X}_x^T\mathbf{W}_x\,\mathbf{X}_x\right)^{-1}\mathbf{X}_x^T\mathbf{W}_x\,\boldsymbol{f}, \tag{2.72}$$

$$\mathrm{Var}\left[\widehat{\boldsymbol{\beta}}_x\right] = \sigma^2\left(\mathbf{X}_x^T\mathbf{W}_x\,\mathbf{X}_x\right)^{-1}\mathbf{X}_x^T\mathbf{W}_x^2\,\mathbf{X}_x\left(\mathbf{X}_x^T\mathbf{W}_x\,\mathbf{X}_x\right)^{-1}, \tag{2.73}$$

where $\widehat{\boldsymbol{\beta}}_x = (\widehat{f}(x), \ldots, \widehat{f}^{(r)}(x)/r!)^T$ and $\boldsymbol{f} = (f(x_1), \ldots, f(x_n))^T$.

Suppose the *design points* x_i are a random sample from some distribution with density function g (this is called a *random design*). Define

$$\mu_q = \int_{-\infty}^{\infty} u^q\,K(u)\,du \quad\text{and}\quad \nu_q = \int_{-\infty}^{\infty} u^q\,K^2(u)\,du, \tag{2.74}$$

so that $\mu_0 = 1$, and $\mu_q = \nu_q = 0$ for odd $q \geq 1$. We assume that $\nu_2 < \infty$ and $\mu_4 < \infty$. For the purposes of Sect. 4.2.2.1, we shall mainly consider the $r = 1$ case. Then it can be shown, subject to regularity conditions (e.g., Sect. 4.2.2.1), that when the x_i are uniformly distributed and x_0 is away from the boundaries, then asymptotically

$$\mathrm{Bias}[\widehat{f}(x_0)] \sim \frac{h^2}{2}\,\mu_2\,f''(x_0), \tag{2.75}$$

$$\mathrm{Bias}[\widehat{f'}(x_0)] \sim \frac{h^2}{3!\,\mu_2}\mu_4\,f'''(x_0), \tag{2.76}$$

$$\mathrm{Var}\left[\widehat{f}(x_0)\right] \sim \frac{\nu_0\,\sigma^2}{n\,h\,g(x_0)}, \tag{2.77}$$

$$\mathrm{Var}\left[\widehat{f'}(x_0)\right] \sim \frac{\nu_2\,\sigma^2}{n\,h^3\,\mu_2^2\,g(x_0)}. \tag{2.78}$$

In these formulas, the bias-variance trade-off can be seen immediately, e.g., as h decreases, the biases decrease and the variances increase. Another observation is that in order for the estimator of $f(x_0)$ to be consistent, it is necessary for $h \to 0$ and $nh \to \infty$ as $n \to \infty$. In fact, it can be shown that to minimize the asymptotic mean integrated squared error (2.84), the optimum rate is $h = O(n^{-1/5})$. Additionally, it can be shown that the asymptotic bias of $\widehat{f}(x_0)$ from a local polynomial regression of degree r is $O(h^{r+1})$ for odd r, and $O(h^{r+2})$ for even r. This suggests that a higher degree r should be chosen for large samples if f is very wiggly.

To verify (2.75)–(2.76), one needs to show that, for example,

$$n^{-1}\sum_{i=1}^{n}\alpha(x_i)\,(x_i - x)\,K_h(x_i - x) \sim h^2\,\{\alpha'(x)\,g(x) + \alpha(x)\,g'(x)\}\,\mu_2$$

for $h > 0$ and some smooth function $\alpha(x)$. The following standard argument is used to obtain the asymptotic mean of the LHS. Call the LHS I_1, say, and let $z = (x_i - x)/h$. Then apply two Taylor series about x:

$$
\begin{aligned}
I_1 \;&\sim\; \int_{-\infty}^{\infty} \left[\alpha(x) + \alpha'(x)\,(x_i - x) + \alpha''(x)\,\frac{(x_i - x)^2}{2} + \cdots\right](x_i - x)\,\cdot \\
&\qquad \frac{1}{h}\,K\!\left(\frac{x_i - x}{h}\right)\left[g(x) + g'(x)\,(x_i - x) + g''(x)\,\frac{(x_i - x)^2}{2} + \cdots\right] dx_i \\
&=\; \frac{1}{h}\int_{-\infty}^{\infty} (zh)\,K(z)\left[\alpha(x) + \alpha'(x)\,zh + \frac{1}{2}\,\alpha''(x)\,(zh)^2 + \cdots\right]\cdot \\
&\qquad \left[g(x) + g'(x)\,zh + \frac{1}{2}\,g''(x)\,(zh)^2 + \cdots\right] h\,dz \\
&\sim\; h\int_{-\infty}^{\infty} z\,K(z)\,\{\alpha'(x)\,g(x)\,zh + \alpha(x)\,g'(x)\,zh\}\,dz \\
&=\; h^2\,\{\alpha'(x)\,g(x) + \alpha(x)\,g'(x)\}\int_{-\infty}^{\infty} z^2\,K(z)\,dz.
\end{aligned}
$$

A similar argument to the above can be used to show the following:

$$
n^{-1}\sum_{i=1}^{n} \alpha(x_i)\,K_h(x_i - x)\,(x_i - x)^t \;\sim
$$

$$
\begin{cases}
\alpha(x)\,g(x) + & \\
h^2\,\mu_2\left\{\frac{1}{2}\,\alpha(x)\,g''(x) + \alpha'(x)\,g'(x) + \frac{1}{2}\,\alpha''(x)\,g(x)\right\}, & t = 0, \\
h^2\,\mu_2\,\{\alpha(x)\,g'(x) + \alpha'(x)\,g(x)\} + & \\
h^4\,\mu_4\left\{\frac{1}{6}\,\alpha\,g''' + \frac{1}{2}\,\alpha'\,g'' + \frac{1}{2}\,\alpha''\,g' + \frac{1}{6}\,\alpha'''\,g\right\}, & t = 1, \\
h^2\,\mu_2\,\alpha(x)\,g(x) + & \\
h^4\,\mu_4\left\{\frac{1}{2}\,\alpha(x)\,g''(x) + \alpha'(x)\,g'(x) + \frac{1}{2}\,\alpha''(x)\,g(x)\right\}, & t = 2.
\end{cases}
\tag{2.79}
$$

One makes good use of the above when working out the elements of $\boldsymbol{\Delta}_x$ and $E[\boldsymbol{\theta}_x]$ (the latter uses $\alpha = f$). For the local linear case,

$$
\boldsymbol{\Delta}_x^{-1} \;\sim\; \frac{1}{g(x)}\begin{pmatrix} 1 & -g'(x)/g(x) \\ -g'(x)/g(x) & 1/\{h^2\,\mu_2\} \end{pmatrix}, \tag{2.80}
$$

which can be used to premultiply the 2-vector $E[\boldsymbol{\theta}_x]$. This gives the first element

$$
f(x) + \frac{h^2\,\mu_2}{g^2(x)}\left\{\frac{1}{2}\,f(x)\,g(x)\,g''(x) + \frac{1}{2}\,f''(x)\,g^2(x) - f(x)\,[g'(x)]^2\right\}.
$$

Subtracting $f(x)$ from this gives the asymptotic bias. If the x_i are uniformly distributed (a *fixed design*), then $g(x)$ is a constant, leading to the bias term (2.75). Similarly, deriving the second element gives Bias$[\widehat{f'}(x_0)] \sim$

$$
\begin{aligned}
&\frac{h^2}{g(x_0)}\left[\frac{\mu_4}{\mu_2}\left\{\frac{1}{6}\,f(x_0)\,g'''(x_0) + \frac{1}{2}\,f'(x_0)\,g''(x_0) + \frac{1}{2}\,f''(x_0)\,g'(x_0) + \frac{1}{6}\,f'''(x_0)\,g(x_0)\right\} - \right. \\
&\left. \mu_2\,\frac{g'(x_0)}{g(x_0)}\left\{\frac{1}{2}\,f(x_0)\,g''(x_0) + f'(x_0)\,g'(x_0) + \frac{1}{2}\,f''(x_0)\,g(x_0)\right\}\right].
\end{aligned}
$$

Then uniformly distributed x_i implies (2.76).

The variance terms (2.77)–(2.78) follow from a similar standard argument that shows $n^{-1} \sum_{i=1}^{n} \alpha(x_i)\, K_h^2(x_i - x)\,(x_i - x)^t \sim$

$$\begin{cases} h^{-1}\,\nu_0\,\alpha(x)\,g(x) + \\ h\,\nu_2 \left\{\frac{1}{2}\,\alpha(x)\,g''(x) + \alpha'(x)\,g'(x) + \frac{1}{2}\,\alpha''(x)\,g(x)\right\}, & t = 0, \\ h\,\nu_2 \left\{\alpha(x)\,g'(x) + \alpha'(x)\,g(x)\right\} + \\ h^3\,\nu_4 \left\{\frac{1}{6}\,\alpha\,g''' + \frac{1}{2}\,\alpha'\,g'' + \frac{1}{2}\,\alpha''\,g' + \frac{1}{6}\,\alpha'''\,g\right\}, & t = 1, \\ h\,\nu_2\,\alpha(x)\,g(x) + \\ h^3\,\nu_4 \left\{\frac{1}{2}\,\alpha(x)\,g''(x) + \alpha'(x)\,g'(x) + \frac{1}{2}\,\alpha''(x)\,g(x)\right\}, & t = 2. \end{cases} \tag{2.81}$$

These are used in the $n^{-1}\left(\mathbf{X}_x^T \mathbf{W}_x^2\, \mathbf{X}_x\right)$ part of the formula of (2.73). Multiplying $\boldsymbol{\Delta}_x^{-1} \mathbf{X}_x^T \mathbf{W}_x^2\, \mathbf{X}_x \boldsymbol{\Delta}_x^{-1}$ together and setting $g' = 0$ gives the required results.

Of further interest, the equivalent kernel (Sect. 2.4.7.3) of a smoother are the weights assigned to y_i in order to obtain $\widehat{f}(x)$. That is, $\widehat{f}(x) = \sum_{i=1}^{n} \omega_i^*\, y_i$ where the ω_i^* are known as the *equivalent kernel* of $\widehat{f}(x)$. For local linear regression, the equivalent kernels are easily found by

$$\begin{pmatrix} \widehat{f}(x) \\ \widehat{f}'(x) \end{pmatrix} = \boldsymbol{\Delta}_x^{-1}\, \boldsymbol{\theta}_x = n^{-1} \sum_{i=1}^{n} K_h(x_i - x)\, \boldsymbol{\Delta}_x^{-1} \begin{pmatrix} 1 \\ x_i - x \end{pmatrix} y_i,$$

therefore the ith vector of this sum which multiplies y_i is

$$n^{-1}\, K_h(x_i - x) \begin{pmatrix} \dfrac{1}{g(x)} - \dfrac{g'(x)}{g^2(x)}\,(x_i - x) \\ \\ -\dfrac{g'(x)}{g^2(x)} + \dfrac{x_i - x}{h^2\, g(x)\, \mu_2} \end{pmatrix}. \tag{2.82}$$

The first element is the asymptotic equivalent kernel for $\widehat{f}(x)$ [cf. (4.44)]. For uniformly distributed x_i, this is proportional to $K_h(x_i - x)$, which makes intuitive sense.

The second element of (2.82) is the asymptotic equivalent kernel for $\widehat{f}'(x)$. For a simple example of $n = 101$ equally spaced points on $[0, 1]$ with $h = 0.2$ and a Gaussian kernel, Fig. 2.16 is a plot of these for three values of x_0. The weights for the y_i are positive to the immediate RHS of x_0, and negative on the LHS; this makes sense given the central finite-difference formula in Sect. 9.2.5: $f'(x) \approx [f(x + h/2) - f(x - h/2)]/h$, whose error is $O(h^2)$.

The above argument may be simplified for the $r = 0$ case to show that the Nadaraya-Watson estimator also has $O(h^2)$ bias in the interior. But it can also be shown that the bias at the boundaries is $O(h)$, which may be quite severe. This can be seen quite simply by smoothing data of the form $y_i = \alpha + \beta\, x_i$ where the x_i are not equally spaced: the Nadaraya-Watson estimate will be nonlinear! If $f(x)$ is quite flat, then the Nadaraya-Watson estimator can perform better than local linear regression, but if $f(x)$ is steep and curved, then local linear regression should be the better choice.

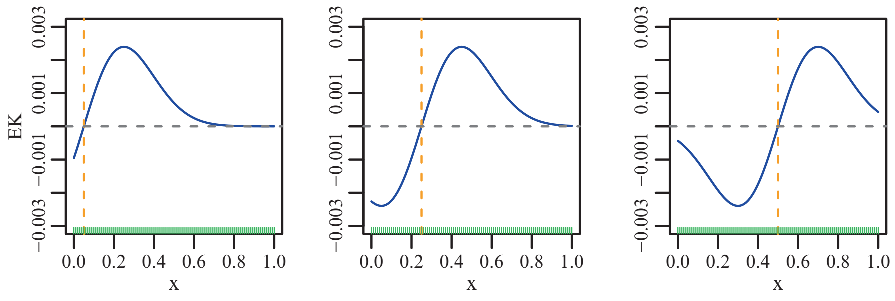

Fig. 2.16 Asymptotic equivalent kernel for $\widehat{f}'(x)$ from (2.82). The 101 x_i are equally spaced on $[0, 1]$ (as shown by the rugplot). The x_0 values are 0.05, 0.25, 0.5 (*vertical dashed lines*), and the bandwidth is 0.2. The kernel function $K = \phi(\cdot)$.

2.4.6.3 On Choosing r, h and K

In the early 1990s, Fan and co-workers showed that, for the νth derivative of f, the case of even $r - \nu$ had the same amount of variability as the next odd $r - \nu$ value. It was therefore recommended that the lowest odd degree $r = \nu + 1$ be chosen, and occasionally, $r = \nu + 3$. Thus, for most applications where f is of primary interest, a local linear regression was suggested. Ruppert et al. (2003, pp.85–6) suggest $r = 1$ when f is monotonically increasing, otherwise $r = 2$ is a good choice (partly supported by simulations). In conclusion, probably $r = 1$ and/or $r = 2$ are a good choice for many data sets, and occasionally $r = 3$.

However, in practice, the choice of the bandwidth h is the most crucial. Much research has been directed towards this very difficult problem, and ideas such as variable bandwidths have been investigated. Some packages reflecting bandwidth selection are bbefkr, KernSmooth, lokern, and np. The choice of the kernel function has long been known to be less important than bandwidth selection.

2.4.6.4 Further Comments

Practically, a major drawback of local regression as described above is the sparse data problem: if some of the (sorted) x_i have big gaps between them and the bandwidth is too small, then there may not be any observations at all within the window. For $r = 0$, this results in the denominator of (2.63) being 0 or nearly so, hence the estimate is unstable or undefined. This problem does not occur so much with splines. Hence the standard local regression formulation needs modification for coal-face general practice. One such modification is to use a nearest neighbourhood such as (2.86).

Since $\widehat{f}(x_0)$ is essentially some WLS fit, many properties of local regression estimators are consequently naturally defined for vector responses. For example, the equivalent kernel, influential measures, bias and variance, degrees of freedom, etc. Some of these are considered for the vector case in Sect. 4.2.2.

2.4.6.5 Lowess and Loess

In passing, it mentioned that two popular methods based on local regression are Lowess (Cleveland, 1979) and Loess (Cleveland and Devlin, 1988). Lowess stands for *lo*cally *w*eighted *s*catterplot *s*moother, and it robustifies the locally WLS method described above. Loess is the more modern of the two (Cleveland et al., 1991) and it can perform multivariate smoothing for $\boldsymbol{x}$.

The basic idea of Loess is to fit a polynomial of degree r locally (the window sizes of which are determined by a nearest-neighbours scheme) and obtain the fitted values. Then the residuals are assigned weights: larger/smaller residuals receive small/large weights respectively. Another local polynomial of degree r (with weights given by the product of the initial weight and new weight) is fitted. Thus observations showing large residuals at the initial fit are *downweighted* in the second fit. The above process is repeated a few times. Cleveland (1979) recommended 3 iterations and $r = 1$, which are the software defaults.

Loess can be invoked simply, e.g.,

```
fit.lo <- loess(y ~ x, data = ldata)
plot(y ~ x, data = ldata)
lines(predict(fit.lo) ~ x, data = ldata)   # The variable x is assumed sorted here
```

and for additive models, it is implemented in gam, e.g.,

```
gam(y ~ lo(x2) + lo(x3), binomial, data = bdata)
```

Both Lowess and Loess measure the size of a neighbourhood using the ‘span’; the larger the value, the larger the neighbourhood.

2.4.7 Some General Theory

In this subsection, a sprinkling of general theory relating to scatter plot smoothing is provided. Here, there is a fundamental trade-off between the bias and variance of the estimator, and this phenomenon is governed by the smoothing parameter. One criterion that compares the two quantities directly at a value x is the (pointwise) *mean squared error* (MSE; Sect. A.1.3.1)

$$\mathsf{MSE}(\widehat{f}(x)) \;=\; E\left[\left(\widehat{f}(x) - f(x)\right)^2\right] \;=\; \mathrm{Var}\left(\widehat{f}(x)\right) + \left(E\,\widehat{f}(x) - f(x)\right)^2. \quad (2.83)$$

A similar quantity to the above, known as the *mean integrated squared error* (MISE), is

$$\mathsf{MISE}(\widehat{f}(\cdot)) \;=\; \int_{-\infty}^{\infty} \mathsf{MSE}(\widehat{f}(x))\, g(x)\, W(x)\, dx, \quad (2.84)$$

which is a global measure of precision. Here, $W(x)$ is a weighting function that might be needed for the integral to exist; it is assumed that $W(x) = 1$ unless otherwise stated. This MISE weighs the MSE of $\widehat{f}$ by the density of the design points g. For some smoothers, this criterion can be minimized with respect to the smoothing parameter.

2.4.7.1 Linear Smoothers

A smoother is said to be *linear* if

$$\widehat{\boldsymbol{y}} \;=\; \mathbf{S}\boldsymbol{y}, \tag{2.85}$$

where the influence matrix $\mathbf{S}$ can depend on $\boldsymbol{x}$ but *not* on $\boldsymbol{y}$. The rank of $\mathbf{S}$ might be n or much less than n—called *full-rank* and *low-rank smoothers*, respectively.

All four smoothers described in this section (regression splines, cubic smoothing splines, P-splines and local polynomial kernel smoothers) are linear smoothers, provided that the smoothing parameters are fixed. Strictly speaking, the use of automatic smoothing parameter selection procedures makes a smoother nonlinear because then $\mathbf{S}$ does depend on $\boldsymbol{y}$. However, as Ruppert et al. (2003) confess, we commonly pretend the smoothing parameter is fixed and, as an approximation, treat the smoother as linear. Other linear smoothers not discussed here include bin smoothers, running-mean smoothers and running-line smoothers.

One can define a symmetric nearest neighbourhood of x_i as the set of indices around about i as:

$$\begin{aligned}\mathcal{N}_i \;=\; \Big\{ & \min\left(i - \frac{\lfloor sn \rfloor - 1}{2}, 1\right), \ldots, i-1, i, i+1, \ldots, \\ & \max\left(i + \frac{\lfloor sn \rfloor - 1}{2}, n\right)\Big\},\end{aligned} \tag{2.86}$$

(Buja et al., 1989) where $0 < s < 1$ is known as the *span*. For $j \in N_i$, one computes the mean of observations (x_j, y_j) to get $\widehat{f}(x_i)$ for the running mean smoother.

An example of a nonlinear smoother is the running median smoother. To see this, suppose that n is large and the size of the symmetric nearest neighbourhood in the interior is 3 observations (these are x_{i-1}, x_i and x_{i+1}). In the absence of ties, the tridiagonal part of the smoother matrix will have two 0s and one 1 in order to pick off the median of three y_i observations. The position of the 1 can only be determined by looking at the y_i, therefore the influence matrix does not depend on $\boldsymbol{x}$ alone.

The theory for linear smoothers is much simpler than for nonlinear smoothers, and this is probably the reason why they are used much more commonly—their properties are well-understood. In probably all respects, linear smoothers generalize all the properties of simple linear regression.

2.4.7.2 Eigenvalues

Many properties of smoothers can be seen by examining the eigenvalues and eigenvectors of $\mathbf{S}$. For example, a cubic smoothing spline has all eigenvalues of $\mathbf{S}(\lambda)$ in $(0, 1]$, with exactly two unit eigenvalues with corresponding eigenvectors $\mathbf{1}$ and $\boldsymbol{x}$, i.e.,

$$\mathbf{S}\,\mathbf{1} \;=\; 1\,\mathbf{1} \quad \text{and} \quad \mathbf{S}\,\boldsymbol{x} \;=\; 1\,\boldsymbol{x}. \tag{2.87}$$

These correspond to constant and linear functions: smoothing y_i that are constant or lie on a line with respect to x_i results in fitted values equal to y_i because such

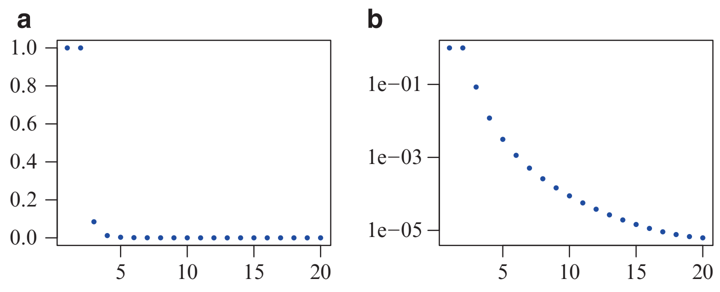

Fig. 2.17 (**a**) Eigenvalues of the smoother matrix of a cubic smoothing spline. Here, $n = 20$ and the x_i are equidistant on $[0, 1]$. (**b**) The same on a logarithmic scale. The two unit eigenvalues correspond to constant and linear functions. The corresponding eigenvectors are plotted in Fig. 2.18.

functions are not penalized by the roughness penalty criterion (2.48). Later, we shall see that such functions which are not penalized belong to the null space $\mathcal{H}_0$ in the RKHS framework of Sect. 4.2.1.7.

If $\mathbf{S}$ is symmetric, then we can write

$$\mathbf{S}\,\boldsymbol{v}_i \;=\; \theta_i\,\boldsymbol{v}_i, \qquad i = 1, \ldots, n, \tag{2.88}$$

where θ_i are real eigenvalues. Smoothers with some $0 < \theta_i < 1$ are called *shrinking smoothers*. If all the θ_i are 0 or 1, then the smoother is called a *regression smoother*. For cubic smoothing splines, the $\boldsymbol{v}_i$ are approximately orthogonal polynomials of increasing order, and

$$\theta_i \;=\; 1/(1 + \lambda\,\rho_i)$$

where $\rho_1 \le \rho_2 \le \cdots \le \rho_n$ so that $\theta_1 \ge \theta_2 \ge \cdots \ge \theta_n$. Figure 2.17 illustrates how quickly these eigenvalues can decay. Now $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ forms an orthonormal basis for $\mathbb{R}^n$, and the spectral decomposition of $\mathbf{S}$ is

$$\mathbf{S} \;=\; \mathbf{V}\,\mathrm{Diag}(\theta_1, \ldots, \theta_n)\mathbf{V}^T \;=\; \sum_{i=1}^{n} \theta_i\,\boldsymbol{v}_i\,\boldsymbol{v}_i^T \;\approx\; \sum_{i=1}^{n^*} \theta_i\,\boldsymbol{v}_i\,\boldsymbol{v}_i^T.$$

The approximation thereof holds for some appropriate $n^* \ll n$ because $\theta_i \approx 0$ for $i > n^*$. The predicted values then are

$$\widehat{\boldsymbol{y}} \;=\; \mathbf{S}\boldsymbol{y} \;\approx\; \sum_{i=1}^{n^*} \theta_i \cdot (\boldsymbol{v}_i^T\boldsymbol{y}) \cdot \boldsymbol{v}_i.$$

This shows that the fitted values are largely determined by the first few eigenvalues and eigenvectors. The high-frequency eigenvectors (Fig. 2.18) are not very important because their effect is dampened by those almost-zero eigenvalues. This suggests a low-rank approximation (e.g., Hastie, 1996) whereby a few of the largest eigenvalues are retained and the remainder set to zero. This idea can be used to motivate P-splines (Sect. 2.4.5).

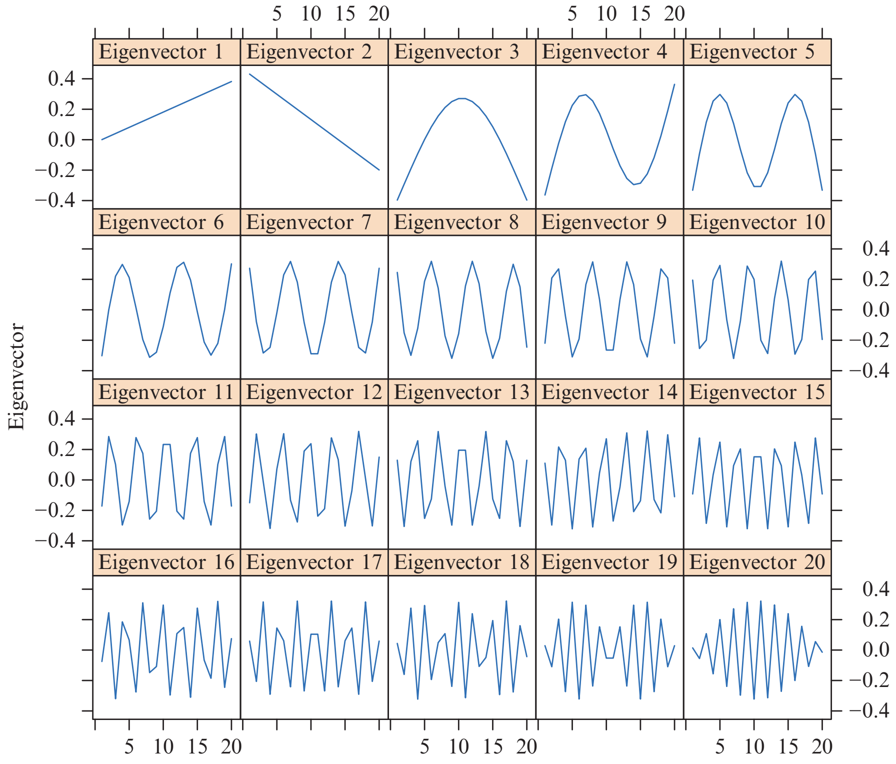

Fig. 2.18 Successive eigenvectors corresponding to the eigenvalues of Fig. 2.17.

2.4.7.3 Equivalent Kernels

Some properties of a smoother may be seen by considering its so-called *equivalent kernel*, e.g., these may be used to compare different types of linear smoothers (e.g., Buja et al., 1989). For a typical linear smoother, plotting a row of the influence matrix $\mathbf{S}$ (see (2.85)) against the x_i values gives the form of neighbourhood used and the weighting function.

For some smoothers, it is possible to derive analytical expressions for their equivalent kernel as $n \to \infty$. We saw this was the case for local linear smoothers in Sect. 2.4.6.2. This is also the case for the cubic smoothing spline: consider the weighted cubic smoothing problem

$$S(f) = \sum_{i=1}^{n} w_i \{y_i - f(x_i)\}^2 + \lambda \int_a^b \{f''(x)\}^2 \, dx, \qquad (2.89)$$

where $w_i > 0$ are known and they sum to unity. Silverman (1984) showed that

$$\widehat{f}(t) = \sum_{i=1}^{n} F(t, x_i) \, w_i \, y_i \qquad (2.90)$$

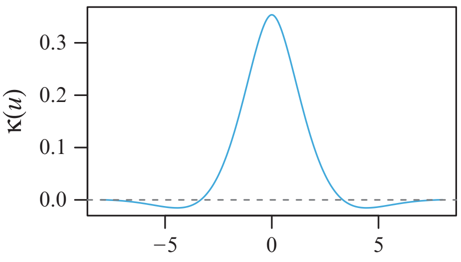

Fig. 2.19 Equivalent kernel of a cubic spline, $\kappa(u)$ (Eq. (2.92)).

asymptotically, with weighting function

$$F(t,x) \;\approx\; \frac{1}{g(x)\,h(x)}\;\kappa\left(\frac{t-x}{h(x)}\right) \tag{2.91}$$

with $h(x) = (\lambda/g(x))^{\frac{1}{4}}$, and kernel

$$\kappa(u) \;=\; \frac{1}{2}\,e^{-|u|/\sqrt{2}}\;\sin\left(\frac{|u|}{\sqrt{2}}+\frac{\pi}{4}\right). \tag{2.92}$$

The latter function is plotted in Fig. 2.19. Although κ is an even function that integrates to unity, its values are not all positive everywhere. The elements of the smoother matrix are given by $(\mathbf{S})_{ij} = w_j\,F(x_i, x_j)$.

Equation (2.90) holds for large n, small λ and x_i not too close to the boundary. That $g^{-1/4}(x)$ is bounded by $g^0(x)$ and $g^{-1}(x)$ indicates that the behaviour of the smoothing spline is between fixed-kernel smoothing and smoothing based on an average of a fixed number of neighbouring values.

2.4.7.4 Effective Degrees of Freedom

All smoothers allow the user to vary the amount of smoothing via the smoothing parameter, e.g., bandwidth h, λ, etc. However, it would be useful to have some measure of the amount of smoothing done that applies to *all* linear smoothers. One such measure is the *effective degrees of freedom* (EDF) of a smooth. It is useful for a number of reasons, e.g., comparing different types of smoothers while keeping the amount of smoothing roughly equal.

Using some basic results pertaining to the hat matrix of the linear model (Sect. 2.2.1) by replacing $\boldsymbol{\mathcal{H}}$ by $\mathbf{S}$, these results suggest the following three definitions for the effective degrees of freedom of a smooth:

$$df \;=\; \mathrm{trace}(\mathbf{S}), \tag{2.93}$$

$$df^{\,\mathrm{var}} \;=\; \mathrm{trace}(\mathbf{S}\mathbf{S}^T),\;\text{ and} \tag{2.94}$$

$$df^{\,\mathrm{err}} \;=\; n - \mathrm{trace}(2\mathbf{S} - \mathbf{S}^T\mathbf{S}). \tag{2.95}$$

More generally, with weights $\mathbf{W}$, these are

$$df \;=\; \mathrm{trace}(\mathbf{S}), \tag{2.96}$$

$$df^{\,\mathrm{var}} \;=\; \mathrm{trace}(\mathbf{W}\mathbf{S}\mathbf{W}^{-1}\mathbf{S}^T),\;\text{ and} \tag{2.97}$$

$$df^{\,\mathrm{err}} \;=\; n - \mathrm{trace}(2\mathbf{S} - \mathbf{S}^T\mathbf{W}\mathbf{S}\mathbf{W}^{-1}). \tag{2.98}$$

It can be shown that if $\mathbf{S}$ is a (symmetric) projection matrix then trace($\mathbf{S}$), trace($2\mathbf{S} - \mathbf{S}\mathbf{S}^T$) and trace($\mathbf{S}\mathbf{S}^T$) coincide. For cubic smoothing splines, it can be shown that

$$\text{trace}(\mathbf{S}\mathbf{S}^T) \;\leq\; \text{trace}(\mathbf{S}) \;\leq\; \text{trace}(2\mathbf{S} - \mathbf{S}\mathbf{S}^T), \tag{2.99}$$

and that all three of these functions are decreasing in λ.

Computationally, *df* is the most popular and the cheapest because only the diagonal elements of the smoother matrix are needed. Its cost is $O(n)$ for most smoothers.

Practically, the EDF lies in the interval $[2, n]$, where a linear fit corresponds to the smallest value and an interpolating function to the largest value. As the EDF increases, the fit becomes more wiggly. Very crudely, a smooth with an EDF of about 3 or 3.5 might have about the same flexibility as a quadratic, say. A value of 4 or 5 degrees of freedom is often used as the default value in software, as this can accommodate a reasonable amount of nonlinearity but without being excessive—it should handle f having one or two stationary points.

Unfortunately, there is scope for confusion when citing the EDF because some authors do not include the constant function because the function has already been centred. For example, `smooth.spline()` and `vsmooth.spline()` have a `df` argument that corresponds to the EDF above: the value 2 means a linear LS fit, etc. However, the `df` argument of `s()` in `gam`'s `gam()`, and `vgam()`, is such that the value 1 corresponds to a linear function. Its default is

```
> args(s)

   function (x, df = 4, spar = 0, ...)
   NULL
```

There may be less opportunity for confusion if the *effective nonlinear degrees of freedom* (ENDF) is cited, e.g., it has value 0 for a linear function.

Zhang (2003) examines calibration issues with regard to their EDF relating to local regression and spline smoothers.

2.4.7.5 Standard Errors

Plots of smooths are commonly supplemented with ± 2 pointwise standard error bands in order to prevent the over-interpretation of the estimated function. For example, Fig. 17.3 shows that the `weight` smooth has its widest pointwise standard errors at the boundaries. Such plots give the viewer some idea about how much to trust $\widehat{f}$, and which parts of the smooth have greater certainty.

From (2.85) it immediately follows that

$$\text{Var}(\widehat{\boldsymbol{f}}) \;=\; \sigma^2\, \mathbf{S}\mathbf{S}^T, \tag{2.100}$$

and so its diagonal elements may be extracted. However, this becomes impractical with large n because the entire $\mathbf{S}$ is needed. For cubic splines, the approximation $\sigma^2\,\mathbf{S}$ has been used instead (and justified by a Bayesian argument, e.g., Wahba (1990); Silverman (1985)). Its cost is $O(n)$, and the approximation has been found to work well in practice.

2.4.7.6 Automatic Smoothing Parameter Selection

Choosing the smoothing parameter is arguably the most important decision for a specified method. Ideally, we want an automated way of choosing the 'right' value. In this section, we restrict ourselves to linear smoothers.

Occasionally it is possible to estimate λ by maximum likelihood, e.g., Wecker and Ansley (1983) for smoothing splines. However, a more general and popular method is the *cross-validation* (CV) technique. The idea is to leave out point (x_i, y_i) one at a time, and estimate the smooth at x_i based on the remaining $n-1$ points. Then λ_{CV} can be chosen to minimize the *cross-validation sum of squares*

$$\mathsf{CV}(\lambda) \;=\; \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i - \widehat{f}_{\lambda}^{[-i]}(x_i) \right\}^2 , \tag{2.101}$$

where $\widehat{f}_{\lambda}^{[-i]}(x_i)$ is the fitted value at x_i, computed by leaving out (x_i, y_i).

While one could compute (2.101) naïvely, a more efficient way is to set the weight of the ith observation to zero and increasing the remaining weights so that they sum to unity. Then

$$\widehat{f}_{\lambda}^{(-i)}(x_i) \;=\; \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{s_{ij}}{1 - s_{ii}}\, y_j \; . \tag{2.102}$$

From this,

$$\widehat{f}_{\lambda}^{(-i)}(x_i) \;=\; \sum_{j=1, j\neq i}^{n} s_{ij}\, y_j + s_{ii}\, \widehat{f}_{\lambda}^{(-i)}(x_i)$$

and

$$y_i - \widehat{f}_{\lambda}^{(-i)}(x_i) \;=\; \frac{y_i - \widehat{f}_{\lambda}(x_i)}{1 - s_{ii}}.$$

Thus, $\mathsf{CV}(\lambda)$ can be written

$$\mathsf{CV}(\lambda) \;=\; \frac{1}{n} \sum_{i=1}^{n} \left\{ \frac{y_i - \widehat{f}_{\lambda}\,(x_i)}{1 - s_{ii}(\lambda)} \right\}^2 . \tag{2.103}$$

This only requires the addition of the diagonal elements of the smoother matrix.

In practice, CV sometimes gives questionable performance. A popular alternative is the *generalized cross validation* (GCV) technique, where

$$\mathsf{GCV}(\lambda) \;=\; \frac{n^{-1} \|(\mathbf{I} - \mathbf{S}(\lambda)\, \boldsymbol{y})\|^2}{[n^{-1}\ \mathrm{trace}(\mathbf{I} - \mathbf{S}(\lambda))]^2} \tag{2.104}$$

is minimized. The rationale for this expression is to replace s_{ii} by its average value, $\mathrm{trace}(\mathbf{S})/n$, which is easier to compute:

$$\mathsf{GCV}(\lambda) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{y_i - \widehat{f}_{\lambda}(x_i)}{1 - \mathrm{trace}(\mathbf{S})/n}\right\}^2 .$$

GCV enjoys several asymptotic optimality properties. However, neither method can be trusted always, especially with small n, e.g., an interpolating spline ($\lambda = 0$) has some positive probability of arising for a given data set.

Both CV and GCV are used when σ^2 is unknown. They are related to other criterion, such as Mallow's C_p (unbiased risk estimator; UBRE). When σ^2 is known, minimizing the UBRE is a popular choice. Another popular criterion is AIC.

2.4.7.7 Testing for Nonlinearity

Suppose we wish to compare two smooths $\widehat{\boldsymbol{f}}_1 = \mathbf{S}_1\boldsymbol{y}$ and $\widehat{\boldsymbol{f}}_2 = \mathbf{S}_2\boldsymbol{y}$, e.g., $\widehat{\boldsymbol{f}}_2$ might be less smooth than $\widehat{\boldsymbol{f}}_1$, and we wish to test if it picks up any significant bias. A standard case that often arises is when $\widehat{\boldsymbol{f}}_1$ is linear, in which case we want to test if the linearity is real. We must assume that $\widehat{\boldsymbol{f}}_2$ is unbiased, and that $\widehat{\boldsymbol{f}}_1$ is unbiased under H_0. Letting ResSS_j be the residual sum of squares for the jth smooth, and γ_j be $\mathrm{trace}(2\mathbf{S} - \mathbf{S}^T\mathbf{S})$, then

$$\frac{(\mathrm{ResSS}_1 - \mathrm{ResSS}_2)/(\gamma_2 - \gamma_1)}{\mathrm{ResSS}_2/(n - \gamma_1)} \;\dot\sim\; F_{\gamma_2-\gamma_1,n-\gamma_1} \tag{2.105}$$

approximately, which follows from a standard F test applied to a LM (2.9).

An approximate score test for VGAMs, given in Sect. 4.3.4, tests for the linearity of component functions.

2.5 Generalized Additive Models

GAMs are a nonparametric extension of GLMs, and they provide a powerful data-driven class of models for exploratory data analysis. GAMs extend (2.17) to

$$g(\mu(\boldsymbol{x}_i)) \;=\; \eta_i \;=\; \beta_1 + f_2(x_{i2}) + \cdots + f_p(x_{ip}), \tag{2.106}$$

a sum of smooth functions of the individual covariates. As usual with these types of models, an intercept is included because the f_k are centred for identifiability. GAMs loosen the linearity assumption of GLMs; this is very useful as it allows the data to 'speak for themselves'. Smoothers are used to estimate the f_k. They still assume additivity of the effects of the covariates, although interaction terms may be accommodated.

We will see later that the VGAM framework writes (2.106) as

$$g_1(\mu(\boldsymbol{x}_i)) \;=\; \eta_1(\boldsymbol{x}_i) \;=\; \beta_{(1)1} + f_{(1)2}(x_{i2}) + \cdots + f_{(1)d}(x_{id}), \tag{2.107}$$

to have provision for handling multiple additive predictors η_j. For VGAM's `vgam()` function, the `s()` function represents the smooths $f_{(j)k}(x_k)$, and it has arguments `df` and `spar` to regulate the amount of smoothness. However, `df` ≥ 1 only is allowed, with a value of unity corresponding to a linear fit.

2.5.1 Why Additive Models?

One of the reasons additive models are popular is that they do not suffer from the *curse of dimensionality* (Bellman, 1961) because all the smoothing is done univariately: $\eta_j(\boldsymbol{x}) = \sum_{k=1}^{d} f_{(j)k}(x_k)$. In one dimension, the concept of a neigbourhood poses the least problems because the x_i are spread out in only one dimension. However, as the dimension of $\boldsymbol{x}$ increases, the volume of the space increases so fast that the data rapidly becomes more and more isolated in d-space. Smoothers then require a larger neighbourhood to find enough data points, hence the estimate becomes less localized and can be severely biased. Theoretically, the sparsity problem might be overcome by a sample size that grows exponentially with the dimensionality, however, this is impractical in most applications.

Modelling the $\eta_j(\boldsymbol{x})$ additively has another advantage: they have simple interpretation. Each covariate has an additive effect, therefore each effect can be determined by keeping the other x_k fixed (although this may be unrealistic in the presence of multicollinearity). The fitted functions are easily plotted separately and examined. However, this simplicity comes at a cost, e.g., interactions are not so readily handled.

One family of models which hold additive models as a special case is based on classical analysis of variance (ANOVA) and called *smoothing spline ANOVA* (SS-ANOVA). Here, functions replace the usual parameters, e.g., one-way SS-ANOVA corresponds to an additive model. A simple example of a two-way SS-ANOVA with covariates x_2 and x_3 is

$$\mu \;=\; \beta_{(1)1} + f_2(x_2) + f_3(x_3) + f_{23}(x_2, x_3),$$

where f_k represents the main effects for x_k, and f_{23} is a second-order interaction between x_2 and x_3. More generally, the unique SS-ANOVA decomposition of a multivariable function f is

$$f(x_2, \ldots, x_d) \;=\; \beta_{(1)1} + \sum_{k=1}^{d} f_k(x_k) + \sum_{s<t} f_{st}(x_s, x_t) + \cdots \qquad (2.108)$$

with the constraints $E_k(f_k) = 0$, $E_s E_t(f_{st}) = 0$, etc. where the E_k are averaging operators. In practice, it is necessary to drop high-order interactions from the model space in order to avoid the curse of dimensionality. Additive models and models with second-order interactions are the most commonly used.

It should be noted that even bivariate smoothing of the form $f(x_s, x_t)$ raises difficulties: although possibly suffering from a mild case of the curse of dimensionality, plotting the functions meaningfully can require some effort, and their interpretation may be difficult.

SS-ANOVA has been extended to generalized SS-ANOVA (GSS-ANOVA), i.e., to η in the classical exponential family. It would be natural then to define the Vector SS-ANOVA class as those VGLM/VGAM families having an ANOVA decomposition (2.108) applied to each η_j.

Now just to show the simplest of GAMs, we now fit a nonparametric logistic regression with one covariate, albeit with a grain of salt.

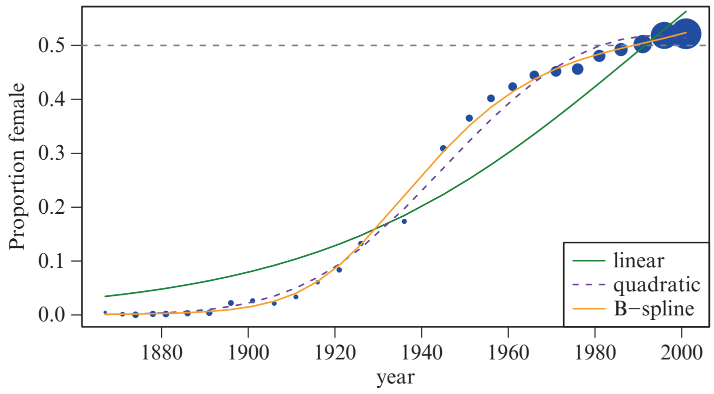

Fig. 2.20 Fitted values from some logistic regression models applied to `chinese.nz`. The response is the proportion of New Zealand Chinese who are female. The terms are `year`, `poly(year, 2)`, `bs(year, 4)`. Area sizes of the points are proportional to the number of people.

2.5.2 Binomial 'example'

In the following, we (controversially) illustrate how ordinary logistic regression can potentially misfit data. To do this, we do something that is not strictly correct, in order to make a point.

The *simple linear logistic regression* model logit $P(Y = 1) = \beta_{(1)1} + \beta_{(1)2}\, x_2$ for a single covariate x_2 results in a sigmoid curve that slopes upward or downward depending on the sign of $\beta_{(1)2}$. The limitation of sigmoid curves seems largely unappreciated by many practitioners. To illustrate the potential inadequacies of this model, consider the `chinese.nz` data frame, which gives the proportion of females in the Chinese population of New Zealand from the mid-1800s to the start of this century. These data are of historical interest, and the number of individuals involved is large enough for the sample proportions to be clearly seen.

Figure 2.20 plots the fitted values of the basic model, as well as some alternatives. Specifically, the underlying code are the first three models of:

```
vglm(cbind(female, male) ~ year,            binomialff, data = chinese.nz)
vglm(cbind(female, male) ~ poly(year, 2),   binomialff, data = chinese.nz)
vglm(cbind(female, male) ~ bs(year, 4),     binomialff, data = chinese.nz)
vgam(cbind(female, male) ~ s(year, df = 3), binomialff, data = chinese.nz)
```

It can be clearly seen that there is underfitting in the 'ordinary' logistic regression. Any predictions based on this model would have severe biases. Applying a quadratic yields a large improvement, and the regression spline is even more flexible.

The above is unwarranted for any formal inference because the data are longitudinal: most people appearing in one year will appear in adjacent years, hence the binomial independence assumption does not hold. Thus the plot should be used little more than for descriptive purposes. It is left to the reader to confirm that the last two models are very similar (Ex. 2.11, 2.23).

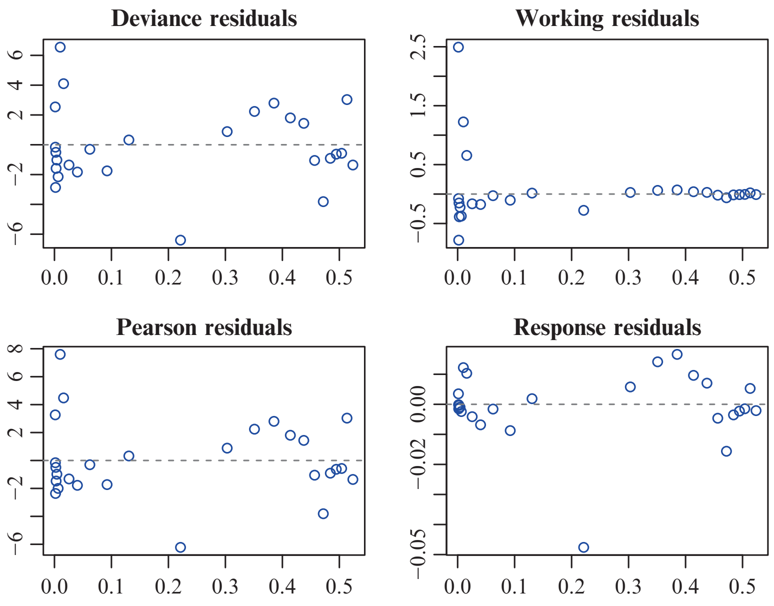

Fig. 2.21 Four residual types for the regression spline fit of Fig. 2.20. The fitted values are plotted on the x-axis.

To give some idea about what residuals can look like, Fig. 2.21 plots four types of residuals versus fitted values for the regression spline fit of Fig. 2.20. It shows that the response residuals are very different from the others, and there is much similarity between the Pearson and deviance residuals here.

Bibliographic Notes

Many books cover the theory of LMs, e.g., Rao (1973), Seber and Lee (2003), Rencher and Schaalje (2008), Christensen (2011). Books for linear modelling based on R include Fox and Weisberg (2011) and Faraway (2015). LM diagnostics are the focus of Belsley et al. (1980) and Cook and Weisberg (1982).

GLMs too are well-served in the literature. McCullagh and Nelder (1989) remains the standard text, and some other descriptions are Firth (1991), Azzalini (1996), Lindsey (1997), Fahrmeir and Tutz (2001), Venables and Ripley (2002), Faraway (2006), Dobson and Barnett (2008), Aitkin et al. (2009), Myers et al. (2010), Agresti (2015). Much of Hastie and Pregibon (1991) is still relevant to `glm()`. Some R package written to overcome situations where `glm()` may fail include biglm (to handle extra large data sets), and glm2 (which addressed possible convergence problems). The brglm package implements bias-reduction, which gives a finite solution to completely separated data (Sects. 2.3.6.3 and 9.4).

The smoothing literature is very large, due partly to it being an active research area during the 1980s and 1990s especially. General texts with accessible material on smoothing include Hastie and Tibshirani (1990), Ruppert et al. (2003) (especially P-splines), Hastie et al. (2009), James et al. (2013, Chap.7). Schimek (2000) surveys many topics related to smoothing, especially computational.

An additive model with at least one smooth term might be called a semiparametric regression model, and Harezlak et al. (2015) is a recent introductory book to such. Ruppert et al. (2009) reviews semiparametric regression from 2002–7. The class of partially linear models, as defined by $Y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + g(\boldsymbol{t}_i) + \varepsilon_i$, is the subject of Härdle et al. (2000).

B-splines are well-covered in the mathematical literature. Of these, some starters include de Boor (2001) and Schumaker (2007). For smoothing splines and their extensions specifically, a good introductory book is Green and Silverman (1994). More intermediate treatments include Eubank (1999), Ruppert et al. (2003). Advanced treatments based on RKHS theory (Sect. 4.2.1.7) include Wahba (1990), Wang (2011), Gu (2013). The two latter references cover the subject of SS-ANOVA models; they are implemented by `ssanova()` in gss. A book specifically on RKHS is Berlinet and Thomas-Agnan (2004), however, Nosedal-Sanchez et al. (2012) is the simplest introduction to RKHS and is specifically focused on smoothing splines.

For kernel smoothing, local regression and likelihood, see e.g., Härdle (1987), Härdle (1990), Wand and Jones (1995), Fan and Gijbels (1996), Loader (1999). Loess was first described within S3 by Cleveland et al. (1991).

The two most comprehensive references on GAMs are Hastie and Tibshirani (1990) and Wood (2006). The current approach of VGAM is much more similar to the former, with respect to the theory and its software (gam). The latter is more focused on automatic smoothing parameter selection based on P-splines and GCV, as implemented by mgcv. Another GAM book is Ruppert et al. (2003). An elementary applied GAM book for novice users only is Zuur (2012). Härdle et al. (2004) gives more examples and theory on a number of topics considered in this chapter, as does Gentle et al. (2012). An accessible overview of some of the ideas behind mgcv is Marra and Radice (2010). The utility of GAMs was recognized quite quickly and introduced into many fields during the 1990s, e.g., Yee and Mitchell (1991) into plant ecology.

Linear algebra and matrices for statisticians are presented at a moderate level by Banerjee and Roy (2014), and at a more advanced level by Harville (1997) and Seber (2008). Yanai et al. (2011) is an accessible introduction to projection matrices, generalized inverses and SVD.

Exercises

Ex. 2.1. Prove all the properties of the hat matrix $\boldsymbol{\mathcal{H}}$ listed in Sect. 2.2.1.

Ex. 2.2. **Hat Matrices**

(a) Prove that $\boldsymbol{\mathcal{H}}$ projects $\boldsymbol{Y}$ orthogonally onto the column (range) space of $\mathbf{X}$.
(b) Obtain an expression for h_{ii} for an LM through the origin: $y_i = \beta_1 x_i + \varepsilon_i$, for $i = 1, \ldots, n$.
(c) Repeat (b) for the simple linear regression model $y_i = \beta_1 + \beta_2 x_{i2} + \varepsilon_i$.

Ex. 2.3. Explain why $\boldsymbol{\mathcal{H}}\mathbf{1} = \mathbf{1}$ is a good idea on p.36. Why would $\boldsymbol{\mathcal{H}}\boldsymbol{x} = \boldsymbol{x}$ also be a good property?

Ex. 2.4. The degrees of freedom for an LM can be defined as $\sum_{i=1}^{n} \mathrm{Cov}(\widehat{y}_i, y_i)/\sigma^2$. Show that this equals p.

Ex. 2.5. GLS
Prove the results (2.14)–(2.16), as well as its hat matrix being idempotent but not symmetric in general.

Ex. 2.6. Show that the score function (2.21) leads to

$$\boldsymbol{U}_{\boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} A_i\, \boldsymbol{x}_i \left(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}\right) \;=\; \mathbf{0}$$

for multiple linear regression, and

$$\boldsymbol{U}_{\boldsymbol{\beta}} \;=\; \sum_{i=1}^{n} A_i\, \boldsymbol{x}_i \left(y_i - \frac{\exp\{\boldsymbol{x}_i^T \boldsymbol{\beta}\}}{1 + \exp\{\boldsymbol{x}_i^T \boldsymbol{\beta}\}}\right) \;=\; \mathbf{0}$$

for logistic regression ($A_i\, Y_i \sim \text{Binomial}(A_i, \mu_i)$ with $\eta = \operatorname{logit} \mu$).

Ex. 2.7. Given the exponential family (2.19), verify all the columns of Table 2.3 from θ to $b''(\theta)$. What are the $c(y, \phi)$ functions?

Ex. 2.8. Using (2.19) with $\phi_i = \phi / w_i$ and where the diagonal elements of $\mathbf{W} = \text{diag}(w_1, \ldots, w_n)$ are known prior weights, show that $\mathbf{X}^T \mathbf{W} \boldsymbol{y}$ are a set of sufficient statistics for $\boldsymbol{\beta}$ for a GLM having a canonical link and known ϕ. Hint: use (A.4).

Ex. 2.9. The moment generating function (MGF) of a random variable Y is defined as $M_Y(t) = E(e^{tY})$ for real t, wherever this expectation exists.

(a) Show that $E(Y) = M'_Y(0)$, $E(Y^2) = M''_Y(0)$, and deduce that $E(Y^k) = M_Y^{(k)}(0)$ for $k = 0, 1, 2, \ldots$.
(b) Obtain an expression for $M_Y(t)$ for Y belong to the exponential family (2.19).
(c) Apply (b) to the Poisson distribution to verify that $E(Y) = \text{Var}(Y) = \mu$.

Ex. 2.10. LRT, Score and Wald Tests

(a) Generate 5 observations from Poisson($\mu = 3$) with the random number seed initialized to some value. Then compute the MLE.
(b) Suppose we wish to test $H_0 : \mu = 3$ versus $H_1 : \mu \neq 3$. Compute the p-values from Wald, score and likelihood ratio tests for this. Comment.
(c) Reset the random number generator and generate 5 observations from Poisson($\mu = 30$). Test $H_0 : \mu = 30$ versus $H_1 : \mu \neq 30$. Repeat in a similar way to (b). Comment.

Ex. 2.11. Nonparametric Logistic Regressions
Fit the four models given in Sect. 2.5.2 to the `chinese.nz` data frame. Obtain the same as Fig. 2.20, and then add the fitted values of the `vgam()` model—hence, show its similarity with the regression spline fit.

Ex. 2.12. Suppose somebody wrote a `log10link()` function so that $\eta = \log_{10} \mu$ could be fitted somewhat like a Poisson regression. How would its estimate of $\boldsymbol{\beta}$ be related to the usual MLE $\widehat{\boldsymbol{\beta}}$?

Ex. 2.13. IRLS Initial Values

(a) Show that one iteration of IRLS starting from any value of $\boldsymbol{\mu}^{(0)}$ will give the LS solution for the normal case, e.g., `gaussianff()`.

(b) Write down expressions for z_i and the working weights w_i for the Poisson and binomial models (each with canonical link functions).

Ex. 2.14. GLM Residuals

(a) Show that the first four residual types in Sect. 2.3.2 simplify to $y_i - \widehat{\mu}_i$ for the Gaussian case.
(b) Obtain a formula for the deviance residuals of a standard Poisson regression.
(c) Obtain a formula for the deviance residuals of a standard logistic regression.

Ex. 2.15. Exponential Family Members
Consider models in the exponential family (2.19).

(a) Show that the two-parameter NB distribution (1.14) is not a standard member of the exponential family. Show that it is a member if k is a known.
(b) Find the canonical parameter θ of the NB, and show that the canonical link is $\log(\mu/(\mu+k))$.
(c) Consider a Pareto distribution (Table 12.8) where the scale parameter b is known. Is this distribution a member of the exponential family? If so, what is its canonical link?

Ex. 2.16. For $p \neq 0$, 1, 2, and $V(\mu) = \mu^p$, show that $q(\mu; y) = y\,\mu^{1-p}/(1-p) - \mu^{2-p}/(2-p)$ where $\mu > 0$. What is the canonical parameter? [McCullagh and Nelder (1989)]

Ex. 2.17. Polynomial Regression
Fit polynomials up to the 10th degree to `mcycles` in MASS, and add them to a scatter plot. Comment.

Ex. 2.18. Running-Mean Smoother

(a) Find $\mathbf{S}$ such that $\widehat{\boldsymbol{y}} = \mathbf{S}\,\boldsymbol{y}$ for a running-mean smoother with $n = 10$, and span $= 0.5$ as defined by (2.86).
(b) Compute the eigenvalues of $\mathbf{S}$ and show that a few are negative. How many unit eigenvalues are there? Comment.
(c) What range of span values would allow for a maximum of 3 values in a symmetric nearest neighbourhood? [Buja et al. (1989)]

Ex. 2.19.

(a) Determine the coefficients a, b and c so that $f(x)$ is a cubic spline, where

$$f(x) \;=\; \begin{cases} x^3, & x \in [0,2], \\ \frac{1}{3}(x-2)^3 + a\,(x-2)^2 + b\,(x-2) + c, & x \in [2,5]. \end{cases}$$

(b) Do the same for coefficients a–d, where

$$f(x) \;=\; \begin{cases} \frac{1}{3}x^3 + x - 1, & x \in [-3,0], \\ a\,x^3 + b\,x^2 + c\,x + d, & x \in [0,1], \end{cases}$$

with $f(1) = 7$.

(c) Do the same for coefficients a–h so that $f(x)$ is a natural cubic spline satisfying $f(-1) = 1$, $f(0) = 2$ and $f(2) = 2$, where

$$f(x) \;=\; \begin{cases} a\,x^3 + b\,x^2 + c\,x + d, & x \in [-1, 0], \\ e\,x^3 + f\,x^2 + g\,x + h, & x \in [0, 2]. \end{cases}$$

(d) Use R to plot the solution of (c) on $[-2, 3]$. What are $f(-2)$ and $f(3)$?

Ex. 2.20. Express the order-3 Bspline $B_{s,3}$ as a linear combination of order-1 B-splines. The coefficients should be in terms of $\omega_{s,3}$, etc. [de Boor (2001)]

Ex. 2.21. Parabolic B-Splines
Plot the 5 parabolic B-spline basis functions whose support is in $[0, 6]$ for the knot sequence $\{0, 1, 1, 3, 4, 6, 6, 6\}$. Comment. [de Boor (2001)]

Ex. 2.22. Regression Splines and `lakeO`

(a) Modify the `lakeO` code that produces Fig. 2.5b to 'work' for the raw variable `year` (taking on values 1974,...,1988). Verify that the LM fails to give finite regression estimates due to the ill-conditioning.
(b) Figure 2.5b is fitted with a *cubic* regression spline. Modify the code to use a *quadratic* regression spline—keep the same knot. Does the fitted curve change appreciably?

Ex. 2.23. Derivative Estimation for GAMs
Consider the `chinese.nz` data set, and let $x_0 = 1936$.

(a) Create a subset of the data frame *without* the year x_0. Fit 3 separate logistic regressions to the subset; the $\eta(x)$ should be (i) linear, (ii) quadratic, and (iii) a smooth function, of $x =$ `year`. Use `vgam()` for (iii). The response is the proportion that are female. Call the fitted curves $\widehat{p}_j(x)$ for $j = 1, 2, 3$.
(b) Predict the values for $p_j(x_0)$.
(c) Plot the sample proportions versus year. Predict the $p_j(x)$ along a fine grid of time, and add the fitted curves to your plot.
(d) For model (iii) compute $\widehat{p}_3'(x_0)$, the first derivative of $\widehat{p}_3(x)$ evaluated at x_0. Plot the sample proportions versus year again, then add $\widehat{p}_3(x)$ and the tangent line at $\widehat{p}_3(x_0)$ to the plot.

Ex. 2.24. Nadaraya-Watson Estimator: Bias and Variance

(a) Verify each entry of (2.79).
(b) Verify each entry of (2.81).
(c) Work out the expressions for the bias and variance, as in (2.72)–(2.73), for the Nadaraya-Watson estimator ($r = 0$).

Ex. 2.25. Variance Estimates in Local Linear Regression
Given (2.81), derive the results (2.77)–(2.78).

Ex. 2.26. Equivalent Degrees of Freedom—Projection Matrix
Show that the 3 definitions of the EDF of a smooth, based on the trace in (2.93)–(2.95), coincide when $\mathbf{S}$ is a projection matrix.

Ex. 2.27. Nadaraya-Watson Estimate

(a) Generate scatter plot data coming from $x_i = 0((n-1)^{-1})1$, $y_i = e^{-2x_i}\sin(5x_i) + \varepsilon_i$, where $\varepsilon_i \sim N(0,\ \sigma^2 = 0.01)$ i.i.d., for $n = 101$, i.e., the design points are equally spaced on the unit interval. Use `set.seed()` for reproducibility.
(b) By eye, determine a reasonable value for the bandwidth h so that your Nadaraya-Watson estimate fits reasonably well.
(c) For your bandwidth, what is the ENDF value? Comment on the how the smoother handles the boundaries.
(d) Determine the values of h so that ENDF $\approx$ 4, 5, and 6.

Ex. 2.28. GCV and CV

Show that $\mathsf{GCV}(\lambda)$ can be written as a weighted version of $\mathsf{CV}(\lambda)$, i.e., $\mathsf{CV}(\lambda) = n^{-1}\sum_i w_i^* \{\ldots\}^2$ starting from (2.101). Obtain an expression for the weights w_i^*.

We believe that the generalized linear models here developed could form a useful basis for courses in statistics.
—Nelder and Wedderburn (1972)

Table 2.3 Summary of GLMs supported by VGAM. A is a known prior weight. The "`ff`" stands for "family function", to avoid clashes with some functions associated with `glm()`. All families have `dpqr`-type functions supplied in standard R. The ϕ and $b(\theta)$ columns relate to (2.19), where θ is the natural parameter.

Distribution	PMF/PDF $f(y;\theta)$	Support	θ	Range of θ	$\mu(\theta)$	Var(Y)	ϕ	$b(\theta)$	VGAM family
Gaussian	$(2\pi\sigma^2)^{-\frac{1}{2}} \cdot \exp\left\{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right\}$	$(-\infty,\infty)$	μ	$\mu \in \mathbb{R},\ 0 < \sigma$	θ	$\frac{\sigma^2}{A}$	σ^2	$\theta^2/2$	`gaussianff()`
Binomial	$\binom{A}{Ay}\mu^{Ay}(1-\mu)^{A(1-y)}$	$0\left(\frac{1}{A}\right)1$	$\operatorname{logit}\mu$	$0 < \mu < 1$	$\frac{e^\theta}{1+e^\theta}$	$\frac{\mu(1-\mu)}{A}$	1	$\log\left(1+e^\theta\right)$	`binomialff()`
Poisson	$\frac{e^{-\mu}\mu^y}{y!}$	$0(1)\infty$	$\log\mu$	$0 < \mu$	e^θ	$\frac{\mu}{A}$	1	e^θ	`poissonff()`
Gamma	$\frac{(k/\mu)^k\, y^{k-1}}{\Gamma(k)}\exp\{-ky/\mu\}$	$(0,\infty)$	$-1/\mu$	$0 < \mu,\ 0 < k$	$-1/\theta$	$\frac{\mu^2}{Ak}$	$\frac{1}{k}$	$-\log(-\theta)$	`gammaff()`
Inverse Gaussian	$\left(\frac{\lambda}{2\pi\, y^3}\right)^{\frac{1}{2}} \cdot \exp\left\{-\frac{\lambda}{2\,\mu^2}\frac{(y-\mu)^2}{y}\right\}$	$(0,\infty)$	$\frac{-1}{2\mu^2}$	$0 < \mu,\ 0 < \lambda$	$\frac{1}{\sqrt{-2\theta}}$	$\frac{\mu^3}{\lambda A}$	$\frac{1}{\lambda}$	$-\sqrt{-2\theta}$	`inverse.gaussianff()`