Chapter 3
Things Are Not Always Linear; Additive Modelling

3.1 Introduction

In the previous chapter, we looked at linear regression, and although the word linear implies modelling only linear relationships, this is not necessarily the case. A model of the form $Y_i = \alpha + \beta_1 \times X_i + \beta_2 \times X_i^2 + \varepsilon_i$ is a linear regression model, but the relationship between Y_i and X_i is modelled using a second-order polynomial function. The same holds if an interaction term is used. For example, in Chapter 2, we modelled the biomass of wedge clams as a function of length, month and the interaction between length and month. But a scatterplot between biomass and length may not necessarily show a linear pattern.

The word ‘linear’ in linear regression basically means linear in the parameters. Hence, the following models are all linear regression models.

- $Y_i = \alpha + \beta_1 \times X_i + \beta_2 \times X_i^2 + \varepsilon_i$
- $Y_i = \alpha + \beta_1 \times \log(X_i) + \varepsilon_i$
- $Y_i = \alpha + \beta_1 \times (X_i \times W_i) + \varepsilon_i$
- $Y_i = \alpha + \beta_1 \times \exp(X_i) + \varepsilon_i$
- $Y_i = \alpha + \beta_1 \times \sin(X_i) + \varepsilon_i$

In all these models, we can define a new explanatory variable Z_i such that we have a model of the form $Y_i = \alpha + \beta_1 \times Z_i + \varepsilon_i$. However, a model of the form

$$Y_i = \alpha + \beta_1 \times X_{1i} \times e^{\beta_2 \times X_{2i} + \beta_3 \times X_{3i}} + \varepsilon_i$$

is not linear in the parameters. In Chapter 2, we also discussed assessing whether the linear regression model is suitable for your data by plotting the residuals against fitted values, and residuals against each explanatory variable. If in the biomass wedge clam example, the residuals are plotted against length, and there are clear patterns, then you have a serious problem. Options to fix this problem are as follows:

- Extend the model with interactions terms.
- Extend the model with a non-linear length effect (e.g. use length and length to the power of two as explanatory variables).

- Add more explanatory variables.
- Transform the data to linearise the relationships. You can either transform the response variables or the explanatory variables. See, for example, Chapter 4 in Zuur et al. (2007) for guidance on this. An interesting discussion with arguments against transformations can be found in Keele (pg. 6–7, 2008). One of the arguments is that a transformation affects the entire Y – X relationship, whereas maybe the relationship is partly linear and also partly non-linear along the X gradient.

Now suppose you have already added all possible explanatory variables, and interactions, but you still see patterns in the graph of residuals against individual explanatory variables, and you do not want to transform the variables. Then you need to move on from the linear regression model, and one alternative is to use smoothing models, the subject of this chapter. These models allow for non-linear relationships between the response variable and multiple explanatory variables and are also called additive models. They are part of the family of generalised additive models (GAM) that we discuss in Chapters 8, 9, and 10, and an additive model can also be referred to as a GAM with a Gaussian distribution (and an identity link, but this is something we will explain in Chapter 9).

References on additive modelling, or more generally on GAM, in R, are Chambers and Hastie (1992), Bowman and Azzalini (1997), Venables and Ripley (2002), Faraway (2006), and Keele (2008). More advanced books are Hastie and Tibshirani (1990), Schimek (2000), Ruppert et al. (2003) or Wood (2006). Other books on GAM, but without R code are Fox (2000), Quinn and Keough (2002), and Zuur et al. (2007), among others. The 'must cite' books are Hastie and Tibshirani (1990) and Wood (2006). The Keele (2008) is an 'easy to read' book, though it aims at the social scientist. The Zuur et al. (2007) book has various case studies, showing applications of GAMs on various types of data, including time series. Its supporting website at www.highstat.com contains the required R code.

3.2 Additive Modelling

The linear regression model using only one explanatory variable is given by

$$Y_i = \alpha + \beta \times X_i + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.1}$$

The relationship between Y_i and X_i is summarised by the parameter β. In additive modelling, we use a smoothing function to link Y_i and X_i. Figure 3.1A shows a scatterplot of pelagic bioluminescence along a depth gradient in the northeast Atlantic Ocean for a particular station (number 16). These data were introduced in Chapter 2 and the graph shows a non-linear pattern. One approach to model these data is to try to linearise the relationship between the two variables by applying a square root, or logarithmic, transformation, and then applying a linear regression. Although this approach may work for the data from this particular station, it will not work for data

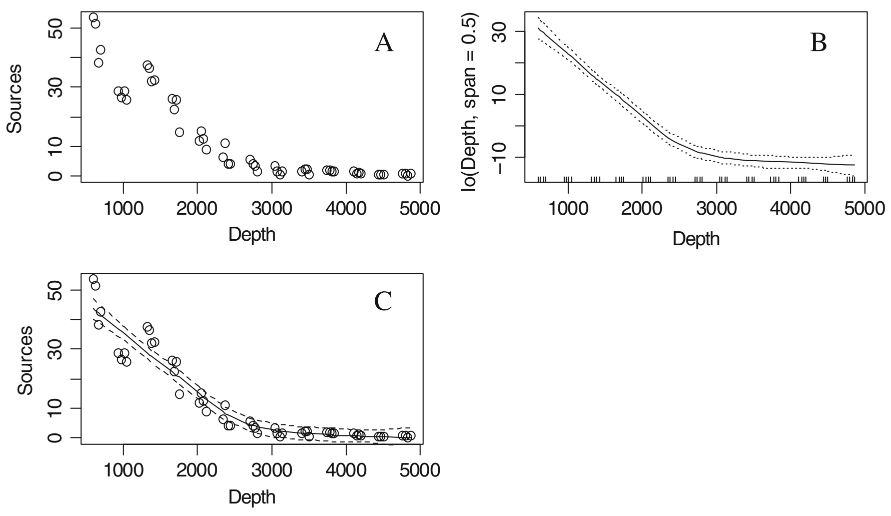

Fig. 3.1 **A**: Scatterplot of pelagic bioluminescence versus depth gradient for cruise 16. **B**: Estimated LOESS curve and pointwise 95% confidence bands obtained by the `gam` function in the `gam` package. **C**: Fitted values obtained by the LOESS smoother and observed data

of all stations in this data set, especially those showing a non-monotonic decreasing source-depth relationship (e.g. stations 6–10, see Fig. 2.11). We therefore need something that can cope with non-linear patterns, for example, the GAM.

3.2.1 GAM in gam and GAM in mgcv

The additive model fits a smoothing curve through the data. There are as many smoothing techniques as there are roads to Rome. In R, there are two main packages for GAM: The `gam` package written by Hastie and Tibshirani and the `mgcv` package produced by Wood. Each package has its own charms. Readers familiar with the classical textbook from Hastie and Tibshirani (1990) may prefer the `gam` package as it follows the theory described in the book. Estimation of smoothers is done using a method called the back-fitting algorithm (a simple but robust way to estimate one smoother at a time). The GAM in `mgcv` uses splines and these require slightly more mathematical understanding than the methods in `gam`.

The main advantage of GAM in the `gam` package is its simplicity; it is easy to understand and explain. The main advantage of GAM in `mgcv` is that it can do cross-validation and allows for generalised additive mixed modelling (GAMM) including spatial and temporal correlations as well as nested data and various heterogeneity patterns. GAMM will be discussed later in this book. Cross-validation is a process that automatically determines the optimal amount of smoothing.

3.2.2 GAM in gam with LOESS

Instead of going straight into the mathematical background of GAM, we show how to run it and explain what it does with help of an example. The following R code was used to generate Fig. 3.1A:

```
> library(AED); data(ISIT)
> op <- par(mfrow = c(2, 2), mar = c(5, 4, 1, 2))
> Sources16 <- ISIT$Sources[ISIT$Station == 16]
> Depth16 <- ISIT$SampleDepth[ISIT$Station == 16]
> plot(Depth16, Sources16, type = "p")
```

We access the variables directly from the ISIT object by using the $ symbol. The additive model applied on the source (response variable Y_i) and depth (explanatory X_i) variable is

$$Y_i = \alpha + f(X_i) + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.2}$$

Note that the only difference between models (3.1) and (3.2) is the replacement of $\beta \times X_i$ by the smoothing curve $f(X_i)$. However, it is fundamentally important to understand the difference. The linear regression model gives us a formula and the relationship between Y_i and X_i is quantified by an estimated regression parameter plus confidence intervals. In a GAM, we do not have such an equation. We have a smoother, and the only thing we can do with it, is plot it.[1] This does not mean that we cannot predict from this model; we can, but not with a simple equation. The smoother is the curve in Fig. 3.1B. The dotted lines are 95% point-wise confidence bands. For the moment, it is sufficient to know that the curve is a LOESS smoother obtained with the gam package. We will explain later what the abbreviation LOESS means. The R code is given below:

```
> library(gam)
> M1 <- gam(Sources16 ~ lo(Depth16, span = 0.5))
> plot(M1, se = TRUE)                    #Fig. 3.1B
```

The gam package does not come with the base installation of R, and you will need to download and install it. The plot command in the code above gives the LOESS smoother in Fig. 3.1B. To obtain a graph with fitted and observed values (Fig. 3.1C), use the following code:

[1]This is not entirely true as we will see later. The smoothers used in this chapter consist of a series of local regression-type models, which do allow for prediction. We just don't get one overall equation.

```
> M2 <- predict(M2, se = TRUE)
> plot(Depth16, Sources16, type = "p")
> I1 <- order(Depth16)
> lines(Depth16[I1], M2$fit[I1], lty = 1)
> lines(Depth16[I1], M2$fit[I1] + 2 * M2$se[I1], lty = 2)
> lines(Depth16[I1], M2$fit[I1] - 2 * M2$se[I1], lty = 2)
> par(op)
```

The `predict` command creates an object containing two variables: fitted values and standard errors. These can be accessed by typing `M2$fit` and `M2$se`. The `order` command avoids a spaghetti plot as the data were not sorted by depth. The `par(op)` sets the graphical parameters back to default values.

3.2.2.1 LOESS Smoothing

LOESS smoothing is discussed in several textbooks, e.g. Cleveland (1993), Hastie and Tibshirani (1990), Fox (2000), Zuur et al. (2007), and Keele (2008) among many others. Here, we only give a short, conceptual explanation, and we strongly advise that you consult Hastie and Tibshirani (1990) if you decide to work with the GAM from the `gam` package. The principle of LOESS smoothing is illustrated in Fig. 3.2.

Suppose we have a target value of depth = 1500 m. Choose a window around this target value of a certain size, let us say 500 m on both sides of the target value so the window goes from 1000 to 2000 m. All the observations inside this window are plotted as black dots, the ones outside the window as open circles. The aim is now to obtain a value for the number of sources at the target value (denoted by the triangle). Multiple options exist; we can take the mean value of the number of sources of all

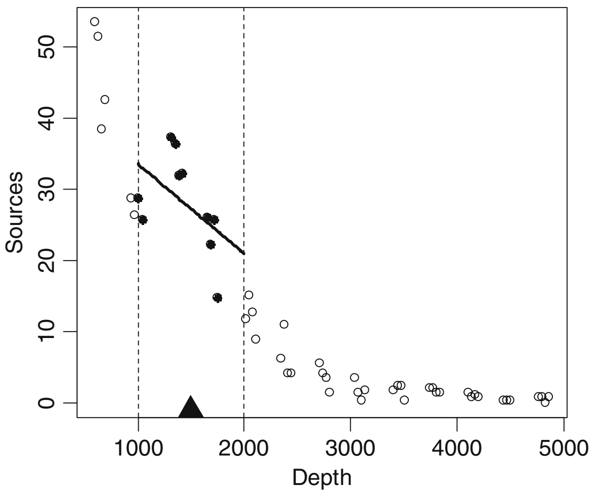

Fig. 3.2 Illustration of LOESS smoothing. We want to predict a value of sources at the target value of 1500 m depth (denoted by the *triangle*). A window around this target value is chosen (denoted by the *dotted lines*) and all points in this window (the *black dots*) are used in the local linear regression analysis

the points inside the window or the median. Alternatively, we can apply a linear regression using only the black points and predict the number of sources at depth = 1500 m from the resulting equation. Or we can apply a weighted linear regression, where the weights are determined by the distance (along the horizontal direction) of the observations from the target value. If we use the linear regression option (also called local linear regression), we get the thick linear regression line in Fig. 3.2. Using the underlying equation for this line, we can easily predict the sources at the target depth of 1500 m, which is 27.2. Repeating this whole process for a sequence of depth values (e.g. from the smallest to the largest depth value, with a total of 100 intermediate equidistant values) and each time storing the predicted value is called LOESS smoothing. In a statistical context, the abbreviation LOESS stands for local regression smoother, which now makes sense. If weights are used in the local regression, we talk about local weighted linear regression abbreviated as LOWESS. Instead of a linear regression, it is also possible to use polynomial models of order *p*; a typical value for *p* is two. You may see the name local polynomial regression in the literature for this.

The R function to use is `loess` (or `lo`). By default, it fits a local polynomial model of order two, but you if you use as option `degree = 1`, it fits a local linear regression. Confusingly, the function `loess` is actually doing local weighted linear regression (see its help file obtained by typing `?loess`). R also has a routine that is called `lowess`, which is an older version of `loess`. The function `loess` has better support for missing values and has different default settings. Keele (2008) shows that the differences between LOESS and LOWESS are marginal.

The process outlined above produces a similar curve as the smoothing curve in Fig. 3.1. The only difference between the smoothing curve in Fig. 3.1 and the one produced by the approach above is the size of the window (resulting in a less smooth curve in Fig. 3.2).

There are two problems associated with moving a window along the depth gradient: the size of the window (also called span width) and what happens at the edges. There is not much you can do about the edges, except to be careful with the interpretation of the smoother at the edges. The size of the window is a major headache. Instead of specifying a specific size (e.g. 500 m), it is expressed as the percentage of the data inside the window. The command in R

```
> lo(Depth, span = 0.5)
```

means that the size of the window is chosen so 50% of the data is inside each window. It is the default value. If this value is chosen as 1, we obtain a nearly straight line; setting it to a very small value gives a smoother that joins every data point. So, is there any point in changing the span width in Fig. 3.1? Note that the sources at the depth range around 1000 m show groups of points being under- and over-fitted. So, perhaps we should decrease the span a little bit, and try 0.4 or 0.3. But how do you assess the optimal span width? One option is trial and error. Figure 3.3 shows the smoother for the ISIT data of station 16. It seems that the smoother obtained with span = 0.1 over-fits the data (this is called under-smoothing). On the other extreme,

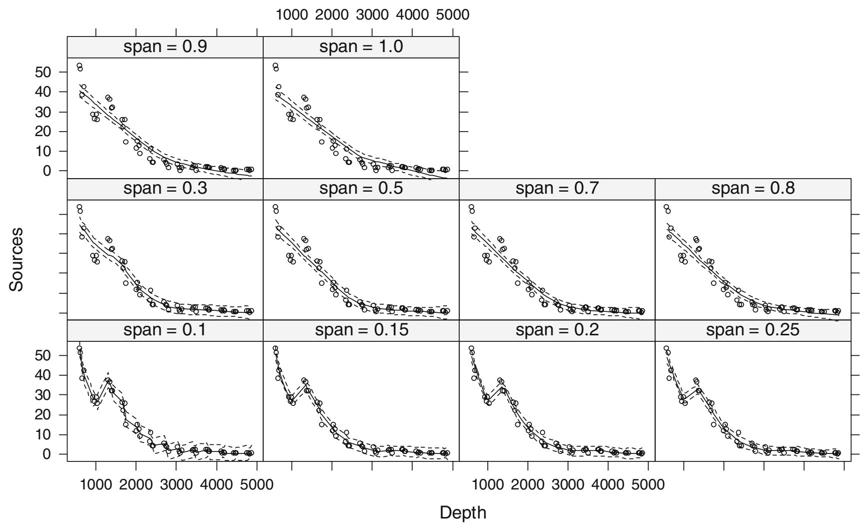

Fig. 3.3 LOESS smoothers for different span values. The solid line is the LOESS smoother and the dotted lines are 95% confidence bands. A visual inspection of the fitted lines and observed values indicates that a span of 0.2 seems to be optimal. The R code to make this graph is about two pages and is therefore presented on the book website

the model with span = 1 shows a clear misfit (this is called over-smoothing). It seems that the smoother obtained with span = 0.2 follows the pattern of the data reasonably well.

Finding the optimal span width is also a matter of bias – variance tradeoff. The smoother obtained with span = 0.1 has only a few data points in each window for the local regression; hence, the uncertainty around the predicted value at the target value will be large (and as a result the confidence bands around the smoothers are large). But the fit will be good! On the other hand, a smoother with a span of 0.9 or 1 will have small variances (lots of data are used to estimate the *Y* value at each target point), but the fit is not good.

Another way to find the optimal amount of smoothing is inspecting residual graphs. If there is still a pattern in the residuals, it may be an option to increase the span width. Yet, another option is to use the Akaike Information Criterion (AIC); see also Appendix A. In this case, the model with a span of 0.15 has the lowest AIC.

The first two options, visual inspection of smoothers and residuals, sound subjective, though with common sense they work fine in practice. It is also possible to apply automatic selection of the amount of smoothing, but instead of presenting this approach for the LOESS smoother, we will discuss this using the GAM in the `mgcv` package. Automatic estimation of the amount of smoothing has not been implemented (at the time of writing this book) in the GAM function in the `gam` package. This provides a good motivation for using the `mgcv` package!

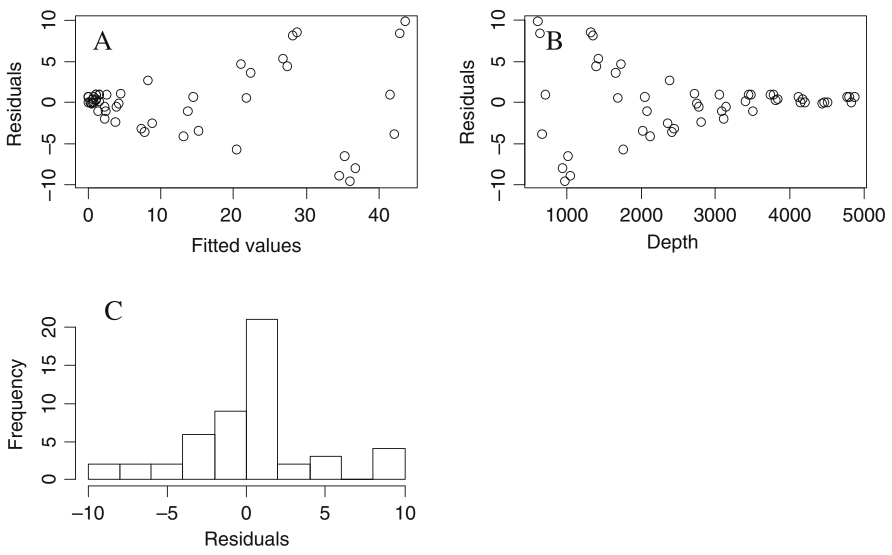

Fig. 3.4 **A**: Residuals versus fitted values. **B**: Residuals versus Depth. **C**: Histogram of residuals. The panels show that there are residual patterns, heterogeneity and non-normality

Hastie and Tibshirani (1990) show how the LOESS smoother can be written in simple matrix algebra, mimicking the matrix algebra of linear regression, and then give a justification for using hypothesis testing procedures that provide F-statistics and p-values. But just as in linear regression, we need to apply a whole series of model validation tests: homogeneity, normality, fixed X, independence, and no residual patterns. The same graphs can (and should) be used; see Fig. 3.4, where it can be seen from panels A and B that there are patterns in the residuals and the residual spread is smaller for deeper depths. The patterns may disappear if we use a span of 0.4 or 0.2. But instead of going this route, we apply GAM with the routines from the `mgcv` package in the next section (and remaining part of this book) as it is considerably more flexible.

3.2.3 GAM in mgcv with Cubic Regression Splines

Before explaining anything about GAM from `mgcv`, we show how to run it and show that it produces very similar results to results from the `gam` package. You can then decide whether it is worth your while digging a bit deeper into the underlying mathematics of splines, which are slightly more complicated than the LOESS smoother! Figure 3.5 contains the same graphs as in Fig. 3.1, except that we used the GAM from `mgcv`

The following R code was used to generate Fig. 3.5. As both packages use the same function `gam`, it may be wise to restart R; alternatively, type `detach("package:gam")`. This avoids R choosing the wrong `gam` function.

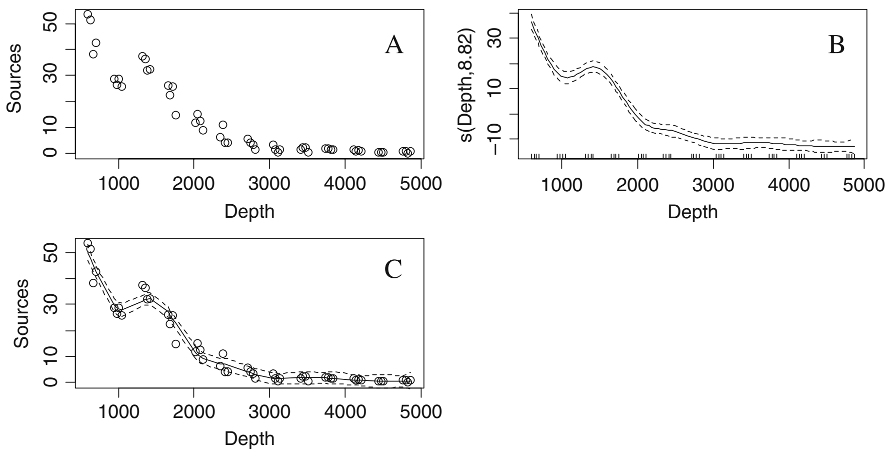

Fig. 3.5 **A**: Scatterplot of pelagic bioluminescence versus depth gradient for cruise 16. **B**: Estimated smoothing curve (cubic regression spline) and point-wise 95% confidence bands. **C**: Fitted values and observed data

```
> library(AED); data(ISIT)
> library(mgcv)
> op <- par(mfrow = c(2, 2), mar = c(5, 4, 1, 2))
> Sources16 <- ISIT$Sources[ISIT$Station == 16]
> Depth16 <- ISIT$SampleDepth[ISIT$Station == 16]
> plot(Depth16, Sources16, type = "p")
> M3 <- gam(Sources16 ~ s(Depth16, fx = FALSE, k=-1,
            bs = "cr"))
> plot(M3, se = TRUE)
> M3pred <- predict(M3, se = TRUE, type = "response")
> plot(Depth16, Sources16, type = "p")
> I1 <- order(Depth16)
> lines(Depth16[I1], M3pred$fit[I1], lty=1)
> lines(Depth16[I1], M3pred$fit[I1]+2*M3pred$se[I1],lty=2)
> lines(Depth16[I1], M3pred$fit[I1]-2*M3pred$se[I1],lty=2)
```

This is nearly the same code as before, except it uses the GAM from the `mgcv` package. This package is part of the base distribution; so you do not have to download anything, but the code to run the `gam` function is slightly different. The format is now similar to that of linear regression (Chapter 2). The expression `Y ~ s(X)` means that a smoothing function is used for the explanatory variable `X`. The `fx = FALSE, k = -1` bit means that the amount of smoothing is not fixed to a preset value; hence, cross-validation is used to estimate the optimal amount of smoothing. We will explain later what cross-validation is. The `bs = "cr"` code tells R that a cubic regression spline should be used. A cubic regression spline is slightly more

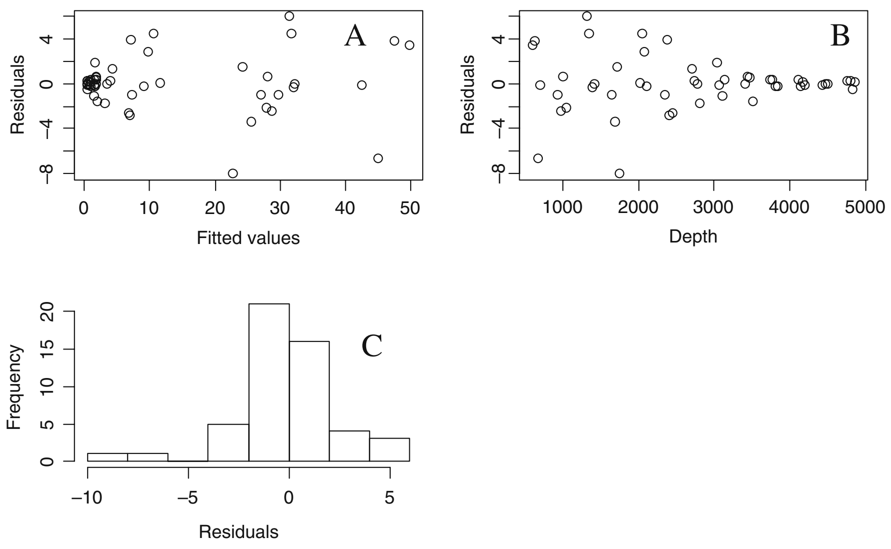

Fig. 3.6 **A**: Residuals versus fitted values using the gam function in mgcv. **B**: Residuals versus Depth. **C**: Histogram of residuals. The panels show that there are no residual patterns (independence), but there is heterogeneity (differences in spread)

complicated to understand than a LOESS smoother and a technical explanation is given in the next section. For now, it suffices to know that for the cubic regression spline method, the *X* gradient (depth) is divided into a certain number of intervals. In each interval, a cubic polynomial (this is a model of the form $Y_i = \alpha + \beta_1 \times X_i + \beta_2 \times X_i^2 + \beta_3 \times X_i^3$) is fitted, and the fitted values per segment are then glued together to form the smoothing curve. The points where the intervals connect are called knots. To obtain a smooth connection at the knots, certain conditions are imposed. These conditions involve first- and second-order derivates and require high-school mathematics to understand. The `gam` function in `mgcv` allows for other types of splines, but for most data sets, the visual differences are small.

This method, again, gives *F*-statistics and *p*-values, and as before, we need to check the assumptions on data characteristics. The validation plots for this example are presented in Fig. 3.6 and show that the residual patterns have disappeared (except for the heterogeneity). The reason for this is that `mgcv` has produced a less smooth curve due to the cross-validation. The R code that was used to create Fig. 3.6 is identical as that for Fig. 3.4 and is not shown again.

3.3 Technical Details of GAM in mgcv*

This section gives a more technical discussion on regression splines (hence the * in the section title) and some other aspects of GAM in the `mgcv` package. It is heavily based on Chapters 3 and 4 in Wood (2006), but at a considerably lower mathematical level. We advise readers to have a look at his chapters after reading this section, and to avoid confusion, we have used the same mathematical notation as Wood (2006), where possible.

The family of splines is rather large, e.g. cubic splines, B-splines, natural splines, thin-splines, and smoothing splines. Here, we discuss a few of them.

We mentioned in the previous section that for a cubic polynomial, the *x*-axis is divided into various segments, and on each segment, a cubic regression spline is fitted. The word cubic refers to 3. In a more general context, we can consider any order for the polynomial. Recall that the smoother is given by the function $f(X_i)$ in

$$Y_i = \alpha + f(X_i) + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.3}$$

We now give an expression for the function $f(X_i)$ so that it can be written as a linear regression model. This is done by using a 'basis' for $f(X_i)$. This means that $f(X_i)$ is built up in basic components, called the basis functions $b_j(X_i)$, such that:

$$f(X_i) = \sum\nolimits_{j=1}^{p} \beta_j \times b_j(X_i) \tag{3.4}$$

This may look magic to many readers, but the principle is fairly simple. Suppose that $p = 4$. This gives

$$f(X_i) = \beta_1 \times b_1(X_i) + \beta_2 \times b_2(X_i) + \beta_3 \times b_3(X_i) + \beta_4 \times b_4(X_i) \tag{3.5}$$

Ok, this may still look magic, but let us assume for the moment that $b_1(X_i) = 1$, $b_2(X_i) = X_i$, $b_3(X_i) = X_i^2$, and $b_4(X_i) = X_i^3$. This results in

$$f(X_i) = \beta_1 \times \beta_2 \times X_i + \beta_3 \times X_i^2 + \beta_4 \times X_i^3 \tag{3.6}$$

This is a cubic polynomial, and it can produce a wide range of possible shapes, depending on the values β_1, β_2, β_3, and β_4. Six examples of such shapes are given in Fig. 3.7. We took fixed values for X_i between 0 and 1 (with equidistance values), randomly generated some values for β_1, β_2, β_3, and β_4, and each time calculated the value of the function *f* using the expression in Equation (3.6).

The problem is that in reality, we do not know the values β_1, β_2, β_3, and β_4, and the shapes of the function *f* in Fig. 3.7 are not flexible enough to model more complicated patterns. Let us solve these two problems. Note that we do know the values of X_i, as it is a measured explanatory variable.

Suppose we divide the depth gradient into four segments (identified by the dotted lines in Fig. 3.8), and on each segment, we fit the model in Equation (3.6) using ordinary least squares (OLS). The fitted curves obtained by this approach are given in Fig. 3.8. At least, we now know the betas per segment (thanks to OLS), and because we used multiple segments, more complicated patterns than in Fig. 3.7 can be fitted. So, these two problems are solved.

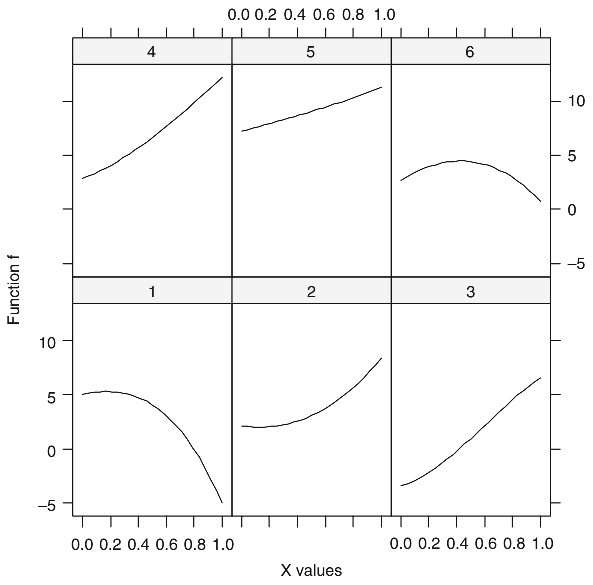

Fig. 3.7 Examples of the function *f* for six different sets of values for β_1, β_2, β_3, and β_4. The *X* values were chosen to have values between 0 and 1. Note the variety in possible shapes for the smoother. R code to create this graph is on the book website

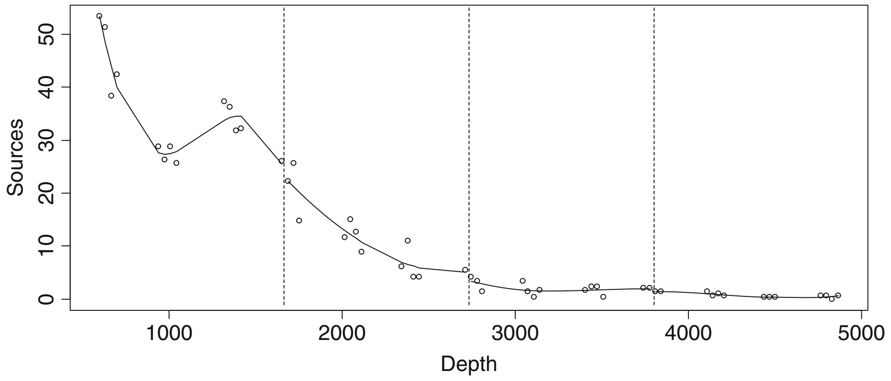

Fig. 3.8 Illustration of fitting a cubic polynomial on four segments of data using the ISIT data from station 19. We arbitrarily choose four segments along the depth gradient. The dotted lines mark these segments, and the line in each segment is the fit from the cubic polynomial model. R code to create this graph can be found on the book website

The problem with Fig. 3.8 is that, if we omit the dotted lines, any editor or referee will wonder why the line has gaps, and it seems to have discontinuities at three points (where the lines come together). A cubic regression spline ensures that the line will look smooth at the points where the individual lines (from the segments) connect. This is done with first-order and second-order derivatives. And here is where things get technical. If you are interested, you may read on or you can skip the next subsection; it is not essential for using GAMs.

3.3.1 A (Little) Bit More Technical Information on Regression Splines

In the previous paragraph, we ended with the statement that the cubic regression spline fits a third-order polynomial on segments of data, and where it connects the resulting fitted lines, it ensures that at the connection points (also called knots), the connections are smooth. This requires some high-school mathematics. It involves first-order and second-order derivatives. It can be shown that this affects the definition of the b_js in Equation (3.4), see Wood (2006). Before showing how the b_js look, we need to define the position of the knots (points along the x-axis where segments are defined). In Fig. 3.7, the knots are at 1665, 2732, and 3799 m. They were obtained by determining the maximum depth (4866 m) minus the minimum depth (598 m) and dividing this by 4 to obtain the segment width (= 1067 m). Once you have the segment width, you can easily calculate the position of the knots. The first knot is at 598 + 1067, the second at 598 + 2 × 1067, and the third at 598 + 3 × 1067. Let the vector $\mathbf{x}^*$ contain these knot positions, making $\mathbf{x}^* = (1665, 2732, 3799)$. The only catch is that the formulae presented below are defined for gradients scaled between 0 and 1. Obviously, this process also influences the values of $\mathbf{x}^*$, which means they become $\mathbf{x}^* = (0.25, 0.5, 0.75)$; but to understand the underlying principle, it is not relevant that the gradient is scaled or not.

We are now in a position to specify how the b_js looks for a cubic regression spline. They become:

$$\begin{aligned} b_1(X_i) &= 1 \\ b_2(X_i) &= X_i \\ b_3(X_i) &= R(X_i, x_1^*) \\ b_4(X_i) &= R(X_i, x_2^*) \end{aligned} \tag{3.7}$$

The notation $x_1{}^*$ refers to the first element of $\mathbf{x}^*$, $x_2{}^*$ to the second element, etc. The form of $R(X_i, z)$ is rather intimidating, and it is given in Wood (2006). We decided to print it here as well, but the computer calculates it for you.

$$R(X, z) = \frac{1}{4} \times \left(\left(z - \frac{1}{2}\right)^2 - \frac{1}{12}\right) \times \left(\left(X - \frac{1}{2}\right)^2 - \frac{1}{12}\right) - \frac{1}{24} \times \left(\left(|X - z| - \frac{1}{2}\right)^4 - \frac{1}{2}\left(|X - z| - \frac{1}{2}\right)^2 + \frac{7}{240}\right)$$

The computer fills in the knot values of $z = 0.25$, or 0.5, or 0.75 and calculates the values of R for given values of the covariate X. Summarising, instead of using a basis of the form $b_1(X_i) = 1$, $b_2(X_i) = X_i$, $b_3(X_i) = X_i{}^2$, and $b_4(X_i) = X_i{}^3$, we use more complicated ones that involve $R(X_i, \mathbf{x}^*)$, and all that this does is ensure that at the knots, the smoothers are neatly connected.

Substituting Equation (3.4) into (3.3) results in the following expression for the additive model.

$$Y_i = \alpha + \sum\nolimits_{j=1}^{p} \beta_j \times b_j(X_i) + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.8}$$

Recall that the b_js are known values determined by the definitions in Equation (3.7) and depend on the number and the values of the knots. This means that the expression in Equation (3.8) forms a linear regression model, and the regression parameters β_j can be estimated by ordinary least squares. The number of regression parameters is equal to the number of knots plus four (for cubic bases).

In a *natural* cubic regression spline, a linear regression model is fitted in the outer two segments. This avoids spurious behaviour of the smoother at the edges.

The next question is how many knots to use. Figure 3.9 shows the effect of the number of knots on the smoothness of the smoother. We used the data from transect 19 as it shows a more non-linear pattern than the data for transect 16. The number of knots is defined as the number of splits (indicated by a vertical line) and the two endpoints. The lower left panel shows the smoother if we use two segments (three knots), the panel next to it three segments, etc. Clearly, the more knots we use, the less smooth the curve becomes.

Just as for the LOESS smoother, we can choose the 'optimal' amount of knots based on a visual comparison of the smoothers. It seems that the smoother with 6 knots follows the pattern of the data reasonably well. Using more knots does not seem to change much. It is also possible to use the AIC (Eilers and Marx, 1996).

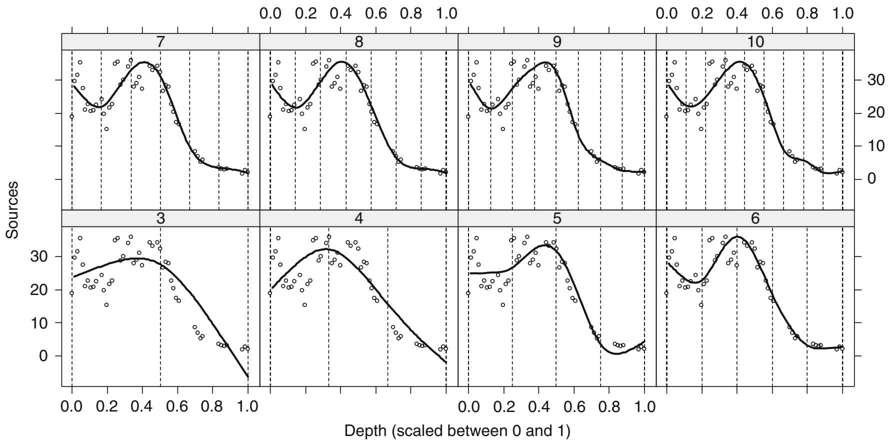

Fig. 3.9 Smoothing curves obtained by cubic regression splines using 3 knots, (*lower left panel*), 4 knots, 5 knots, etc., and 10 knots (*upper right panel*). Both end points also count as knots. The vertical lines show the position of the knots. The thick line in each panel is the smoother, and the dots the observed data for transect 19. The more knots used, the more variation in the smoother. R code to calculate the smoothing curves can be found in Wood (2006), and extended R code to present the graphs in an `xyplot` from the lattice package is on the book website

Keele (2008) gives as general recommendation to use 3 knots if there are less than 30 observations and 5 knots if there are more than 100 observations.

As to the placement of the knots, this is typically done by the software using quartiles and equidistant positions.

3.3.2 *Smoothing Splines Alias Penalised Splines*

Because the model in Equation (3.8) can be written as a linear regression model, it is tempting to use hypothesis testing procedures or backwards selection methods to find the optimal number of knots. After all, linear regression implies that we can find the parameters β_j by minimising the sum of squares:

$$S = \sum\nolimits_{i=1}^{n} (Y_i - \mu_i)^2$$

where μ_i is the fitted value for observation i, and $i = 1, \ldots, 789$ (there are 789 observations). A typical mathematical notation for this is

$$S = \sum\nolimits_{j=1}^{n} (Y_i - \mu_i)^2 = \|\mathbf{Y} - \boldsymbol{\mu}\|^2 = \|\mathbf{Y} - \mathbf{X} \times \boldsymbol{\beta}\|^2$$

The notation || || stands for the Euclidean norm, **Y** contains all the observed data in vector format, $\boldsymbol{\beta}$ all the parameters in vector format, and **X** all the b_js. It looks intimidating, but it is nothing more than some matrix notation for the residual sum of squares in linear regression. This type of matrix notation for linear regression can be found in many statistical textbooks, e.g. Montgomery and Peck (1992) or Draper and Smith (1998).

However, Wood (2006) argues that it is unwise to go the route of hypothesis testing and backwards selection to obtain the optimal number of knots for a regression spline. Instead, smoothing splines (also called penalised splines) are used. These are obtained by finding the parameters $\boldsymbol{\beta}$ (and therefore the smoothers) that minimise the following criteria.

$$\|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int f''(x)^2 dx \qquad (3.9)$$

The second part of this equation is a penalty (hence the name penalised least squares and penalised smoothers) and is new. It contains λ and an integral over the second-order derivatives. Remember that the second-order derivatives of the smoothing function *f*, denoted by f'', tell you how smooth a curve is. A high value of f'' means that the smoother *f* is highly non-linear, whereas a straight line (the perfect smooth curve) has a second-order derivative of 0. So, if λ is very large, the penalty for having a non-smooth curve is large, and the resulting smoother will be a straight line. On the other hand, if λ is small, then there is a low penalty for non-smoothness, and we are likely to end up with a considerably less smooth curve.

We started with the question of how to find the optimal amount of smoothing, and it is still not answered. Instead, we have obtained a different way of quantifying the amount of smoothing. We are now no longer looking for the optimal number of knots. Instead, our new aim is find the optimal amount of smoothing by choosing a fixed and large number of knots (and keep them fixed during the analysis) and by focusing the analysis on finding the optimal value for λ.[2] The word 'optimal' means we want to minimise the expression in Equation (3.9), where the integral measures the amount of wiggliness of the smoother, which is then controlled by the parameter λ. It is just a different approach to tackle the same problem, albeit a better one according to Wood (2006); based on mathematics, the cubic smoothing spline gives the best possible fit with the least amount of error. Minimising the expression in Equation (3.9), also called penalised least squares, is just much simpler, faster, and more robust than hypothesis testing for the number of knots.

So, how do we find the optimal value of λ? Before addressing this question, we look at the effects of changing the value of λ on the smoother. Figure 3.10 shows the cubic regression spline for λ = *10e–6*, λ = *10e–5*, λ = *10e–4* up to λ = 10. As might be expected, the larger the value of λ, the more linear the smoother. Changing the number of knots has a considerably smaller effect than changing λ.

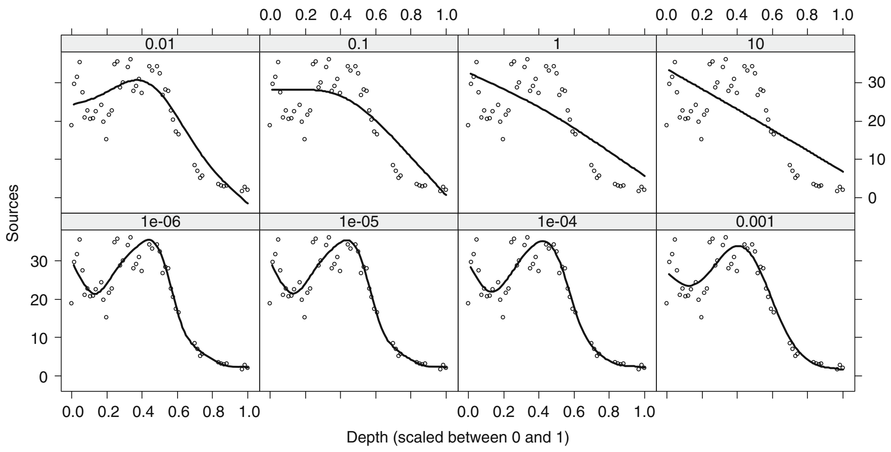

Fig. 3.10 Smoothing curve for different values of λ. The lower left panel shows the estimated smoother obtained by minimising the expression in Equation (3.9) for λ = *10e–6*, the upper left panel for λ = *10e–2*, and the upper right panel for λ = *10.* R code to calculate the smoothing curves can be found in Wood (2006), and modified code to present the graphs in an `xyplot` from the lattice package is on the book website

[2]Keele (pg. 69, 2008) shows an example in which the fits of two smoothing splines with the same amount of smoothing (λ) are compared; one smoother uses four knots and the other uses sixteen knots; the difference between the curves is minimal.

Now that we have seen the effect of changing λ, we can ask the original question again: 'What is the optimal amount of smoothing?', or even better: 'What is the optimal value for λ?' This question is answered with a process called cross-validation and is explained below.

3.3.3 Cross-Validation

To obtain the smoothers in Fig. 3.10, we *chose* values for λ and then minimised the expression in Equation (3.9). Now, we consider the minimisation of this function if *both* $\boldsymbol{\beta}$ and λ are unknown.

The function $f(X_i)$ is for the population, and its estimator (based on a sample) is written as:

$$\hat{f}(X_i)$$

The subtle difference is the hat ˆ on f, which denotes that it is an estimator. The estimator needs to be estimated to be as close as possible to the real value, and one criterion to judge this is (Wood, 2006)

$$M = \frac{1}{n}\sum_{i=1}^{n}(f(X_i) - \hat{f}(X_i))^2$$

If we would know $f(X_i)$, we could just choose a λ such that M is as small as possible. The problem is that $f(X_i)$ is unknown. Therefore, the strategy is to replace M by 'something' that can be calculated and minimise this 'something' as a function of λ. There are different options for the 'something' bit, and here is where things like cross-validation (CV), generalised cross-validation (GCV), unbiased risk estimator (UBRE), or Mallow's C_p pop up. You will see these things in the numerical output of the GAM from the mgcv package if you apply automatic selection of the amount of smoothing. Let us start explaining how cross-validation, also called *ordinary* cross-validation (OCV), works. Define the OCV score as

$$V_0 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}^{[-i]}(X_i))^2$$

The notation [–i] means that the smoother is calculated using all observations except for observation i. Observation i is dropped, the smoother is estimated using the remaining $n - 1$ observations, the value for observation i is predicted from the estimated smoother, and the difference between the predicted and real value is calculated. This process is then repeated for each observation in turn, resulting in n prediction residuals. As always, residuals are squared and added up, which gives V_0. Wood (2006) shows that the expected value of V_0 is approximately equal to the expected value of M plus σ^2:

$$E[V_0] \approx E[M] + \sigma^2$$

If the aim is to find a λ that minimises M, then minimising V_0 is a reasonable approach. The process of minimising V_0 is called ordinary cross-validation. However, for a large data set, this is a time consuming process as we have to repeat the analysis n times. Luckily, a mathematical shortcut exists to calculate V_0 in one analysis! The new equation for V_0 is

$$V_0 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}(X_i))^2/(1 - A_{ii})^2$$

The A_{ii} are the diagonal elements of the so-called influence matrix. Explaining what exactly this is requires a certain degree of matrix algebra and the interested reader is referred to Section 4.5.2 of Wood (2006). However, instead of working with V_0, we use a slightly modified version, namely, the generalised cross validation (GCV). The $1 - A_{ii}$ bit is replaced by something that involves the trace of $\mathbf{I} - \mathbf{A}$, and the justification for this requires fairly complex matrix algebra. It suffices to know that GCV is a modified version of OCV with computational advantages.

To illustrate the process, we calculated GCV for a large range of values for λ (Fig. 3.11). The lowest GCV value is obtained for $\lambda = 0.000407$ (we took this from the numerical output underlying the graph). Hence, the smoother in the third panel (from the left) at the bottom of Fig. 3.10 is close to the optimal smoother.

It is advisable not to follow blindly the results of the cross-validation. Violation of collinearity and independence and application of cross-validation on small (< 50) data sets can cause trouble (e.g. over-smoothing). It may be wise to verify the cross-validation results with smoothers in which the amount of smoothing is selected manually (or modify the results from the optimal model a little bit).

Instead of providing the output of the cross-validation in terms of λ, the `gam` function in the mgcv package uses a term called the *effective degrees of freedom*

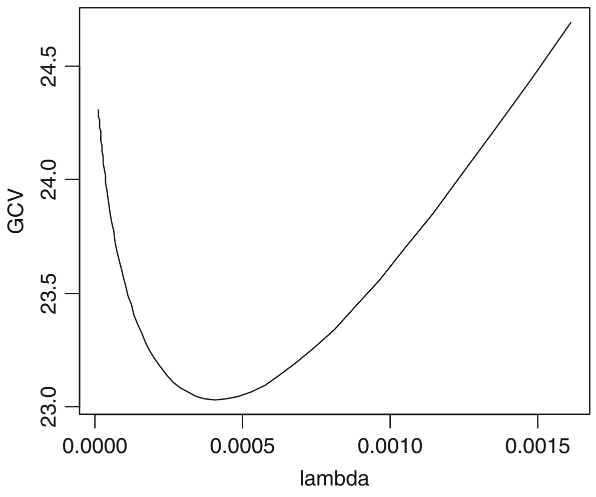

Fig. 3.11 GCV score plotted versus λ. The optimal smoother has $\lambda = 0.000407$. R code to produce this graph is given on the book website

(edf). This is a value between 0 and infinity and is a sort of mathematical transformation of λ. The higher the edf, the more non-linear is the smoothing spline.

In the cross-validation process, we assumed that the residual variance σ^2 is unknown. The UBRE approach minimizes a slightly different criterion, and it works well if σ^2 is known. Its underlying formulae are identical to Mallow's C_p.

Another issue is the actual estimation of the smoothing spline and λ. Both require iterative numerical algorithms. We can choose λ and find the smoothing spline by minimizing the expression in Equation (3.9) and put the generalised cross-validation procedure on top of this to find optimal λ. This is called outer iteration. The alternative is to do it the other way around: put the generalised cross-validation inside the iteration process of minimising the expression in Equation (3.9). This is called performance iteration. Probably you don't want to know these numerical details; they only become an issue if you get convergence problems and want to try different settings.

3.3.4 Additive Models with Multiple Explanatory Variables

Equation (3.2) showed an additive model with one explanatory variable. The same model can easily be extended to two explanatory variables:

$$Y_i = \alpha + f_1(X_i) + f_2(Z_i) + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.10}$$

The functions $f_1(X_i)$ and $f_2(Z_i)$ are smoothing functions of the explanatory variables X_i and Z_i respectively. Both functions can be written as in Equation (3.4). Just as the model with one explanatory variable was written as a linear regression model; so the model in Equation (3.10) is written as a linear regression model. Its form is exactly the same as in (3.9), except that the matrix **X** now also contain information on the second smoother. This also means that we can have hybrid models of the form

$$Y_i = \alpha + f_1(X_i) + \beta \times Z_i + factor(W_i) + \varepsilon_i \qquad \text{where} \quad \varepsilon_i \sim N(0, \sigma^2) \tag{3.11}$$

Examples of interaction in smoothers are given in the case study chapters, and an introductory example is discussed in the next section.

3.3.5 Two More Things

Before we give an example, there are two more things we need to discuss: degrees of freedom of a smoother and other types of smoothers. As to other type of smoothers, in the previous section, we discussed the cubic regression spline. Recall that this was a smoother that fits third order polynomials on segments of data, and to ensure a nice looking curve with no sudden jumps at the knots, certain conditions were imposed.

These conditions resulted in a basis system with rather intimidating expressions of the form

$$f(X_i) = \sum\nolimits_{j=1}^{p} \beta_j \times b_j(X_i)$$

There is actually a large collection of related smoothers, and Chapter 4 in Wood (2006) gives a detailed mathematical explanation on different types of smoothers. We do not repeat the mathematical detail here. The main difference between these smoothers is the definition of the b_js, and also a few differ with respect to the optimisation criterion defined in Equation (3.9).

A useful smoother is the cyclic cubic regression spline. It ensures that the value of the smoother at the far left point of the gradient is the same as at the far right point of the gradient. This comes in handy if you use a smoother for month (with 12 values) or a smoother for day of the year; it wouldn't make sense to have a big jump between the January value and the December value for the month smoother. Shrinkage smoothers are also useful; they can have 0 amount of smoothing. This means that if you do a backwards selection to find the optimal model, all smoothers with 0 amount of smoothing can be dropped simultaneously from the model.

The thin plate regression spline is yet another smoother, one that apparently does quite well (except for larger data sets). The thin plate regression spline involves higher order derivates in the integral in Equation (3.9). In practise, the difference between cubic regression splines, thin plate regression splines, B-splines, P-splines, etc., is rather small.

The final point we want to discuss in this section is the amount of smoothing for smoothing splines. If a model has two smoothers, say

$$Y_i = \alpha + f_1(X_i) + f_2(Z_i) + \varepsilon_i$$

then these two smoothers have the form

$$f_1(X_i) = \sum\nolimits_{j=1}^{p} \beta_j \times b_j(X_i) \quad \text{and} \quad f_2(Z_i) = \sum\nolimits_{j=1}^{p} \gamma_j \times b_j(Z_i)$$

We mentioned this earlier in this section. Using two smoothers instead of one smoother has an effect on the definitions of the **Y**, **X**, and $\boldsymbol{\beta}$ in Equation (3.9), but in essence the form stays the same. The optimisation criterion with the penalty for the wiggliness becomes

$$\|\mathbf{Y} - \mathbf{X} \times \boldsymbol{\beta}\|^2 + \lambda_1 \int f_1''(x)^2 dx + \lambda_2 \int f_2''(x)^2 dx \qquad (3.12)$$

This looks rather intimidating, but all it does is allow for different amounts of wiggliness per smoothing spline. As a result, some smoothers can be smooth (large λ_j), whereas others are not (small λ_j). Hence, the values of the λ_js determine the amount of smoothing. To get these λ_js, the optimisation criterion in Equation (3.12) can be written in matrix notation as

$$\|\mathbf{Y} - \mathbf{X} \times \boldsymbol{\beta}\|^2 + \boldsymbol{\beta}' \times \mathbf{S} \times \boldsymbol{\beta} \tag{3.13}$$

This looks a little bit friendlier, but the average reader may still be utterly confused by this stage. But you need not be too concerned as the rules for filling in the elements of **S** are given in Wood (2006). Now we come to the reason why we explained all of this detail. The amount of smoothing of a smoother is not expressed in terms of the λ_js, as you would expect, but as *effective degrees of freedom* for a smoother. A high value (8–10 or higher) means that the curve is highly non-linear, whereas a smoother with 1 degree of freedom is a straight line. You can think about it as a calibration parameter. Technically, the matrix **S**, which depends on the λs, is involved in determining the effective degrees of freedom (edf) and it mirrors the algebra underpinning linear regression.

We now give two examples, and once we are more familiar with the graphical and numerical output of GAMs, we discuss how much trust we can place in the *p*-values coming out of the `anova` and `summary` commands.

3.4 GAM Example 1; Bioluminescent Data for Two Stations

Earlier in this section, we used bioluminescent data, obtained from one particular station. A `xyplot` from the lattice package was given in Chapter 2, showing the data from all 19 stations along each depth gradient. In this section, we combine the data from two stations and show how GAM can be used to analyse the data. We start by presenting the data in a graph, see Fig. 3.12. The patterns in both graphs are

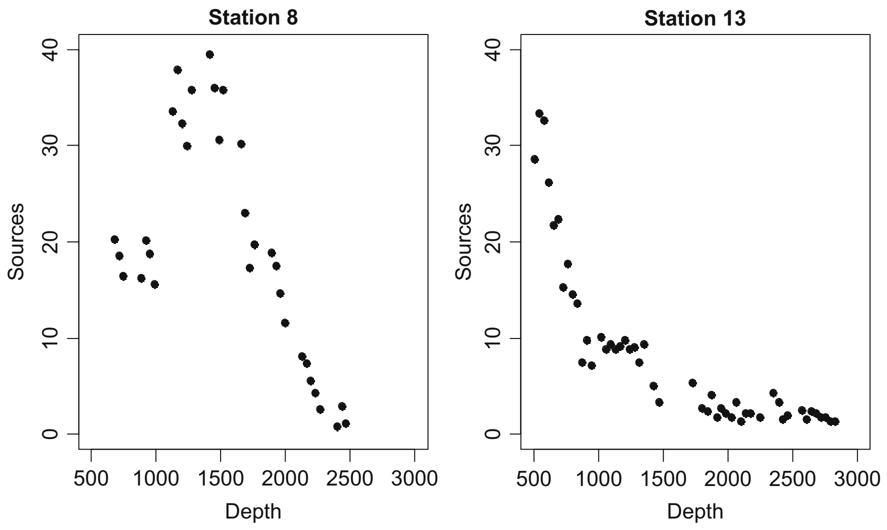

Fig. 3.12 Bioluminescent data for stations 8 and 13. Sources are plotted against depth. The axes limits were set to be equal

similar (decreasing pattern along depth), but there are also clear differences. The question that now arises is whether we can model the data with one smoother or perhaps we need two smoothers, one for each station? To answer this, we need to fit a model with one smoother and then a model with two smoothers and compare them with each other. We start with a GAM that only contains one smoother, go over the graphical and numerical output, and then return to the question of choosing the best model.

The R code used to create the graph is given below.

```
> library(AED); data(ISIT)
> S8  <- ISIT$Sources[ISIT$Station == 8]
> D8  <- ISIT$SampleDepth[ISIT$Station == 8]
> S13 <- ISIT$Sources[ISIT$Station == 13]
> D13 <- ISIT$SampleDepth[ISIT$Station == 13]
> So <- c(S8, S13); De <- c(D8, D13)
> ID <- rep(c(8, 13), c(length(S8), length(S13)))
> mi <- max(min(D8), min(D13))
> ma <- min(max(D8), max(D13))
> I1 <- De > mi & De < ma
> op <- par(mfrow = c(1, 2))
> plot(D8[I1], S8[I1], pch = 16, xlab = "Depth",
    ylab = "Sources", col = 1, main = "Station 8",
    xlim = c(500, 3000), ylim = c(0, 40))
> plot(D13[I1], S13[I1], pch = 16, xlab = "Depth",
    ylab = "Sources", col = 1, main = "Station 13",
    xlim = c(500, 3000), ylim = c(0, 40))
> par(op)
```

The first part of the code accesses the data, extracts the relevant data (`Sources` and `SampleDepth`) from stations 8 and 13, concatenates them in a long vector `So` and `De` using the concatenate command `c`, creates a vector that identities which row corresponds to a certain station using the `rep` (repeat) command, and determines the common depth ranges for both stations (the `min` and `max` commands). These data are then plotted in two panels on the same graphical window (see the `par` command). The `[I1]` bit in the `plot` command ensures that we only use those data with the same depth ranges.

The underlying model for a GAM with one smoother is given by

$$Sources_i = \alpha + f(Depth_i) + factor(Station_i) + \varepsilon_i \qquad \varepsilon_i \sim N(0, \sigma^2) \qquad (3.14)$$

Station is a nominal explanatory variable with two levels, and depth is a continuous explanatory variable. The index i runs from 1 to 75. To apply this GAM model in the mgcv package, use the following code.

```
> library(mgcv)
> M4 <- gam(So ~ s(De) + factor(ID), subset = I1)
> summary(M4)
> anova(M4)
```

The `subset = I` part is merely used to ensure that we use observations within the same depth ranges. The `library` command loads the mgcv package. The argument within the `gam` function is similar to linear regression; `So` (sources of both stations) is modelled as a smoothing function (using the `s` function) of `De` (concatenated depth data of both stations) and the nominal variable `ID` (identifying the station). Both the `summary` and `anova` commands provide useful numerical output. The `summary` output is as follows.

```
Parametric coefficients:
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)   19.198      1.053  18.236  < 2e-16
factor(ID)13 -12.296      1.393  -8.825 6.86e-13

Approximate significance of smooth terms:
        edf Est.rank     F  p-value
s(De) 4.849        9 10.32 7.37e-10

R-sq.(adj) =  0.695   Deviance explained = 71.9%
GCV score = 38.802   Scale est. = 35.259    n = 75
```

The explained deviance (R^2) is 71.9%, the variance of the residuals is 35.26, the smoother for depth is significant at the 5% level, the estimated degrees for the smoother is 4.8, the intercept has a value of 19.2, and station 13 is 12.3 units lower than station 8. The output from the `anova` command is more compact and is useful if the model contains nominal variables with more than two levels as it gives one *p*-value for all the levels using an *F*-test. It is given below.

```
Parametric Terms:
           df     F  p-value
factor(ID)  1 77.88 6.86e-13

Approximate significance of smooth terms:
        edf Est.rank     F  p-value
s(De) 4.849    9.000 10.32 7.37e-10
```

In practice, both the `summary` and `anova` commands would be used. The `plot (M4)` command produces the fitted smoother in Fig. 3.13. To get the fitted values for observations from station 8, we use

$$Sources_i = 19.2 + f(Depth_i) + \varepsilon_i \qquad \varepsilon_i \sim N(0, 5.9^2) \tag{3.15A}$$

And for observations of station 13 we use

$$Sources_i = 19.2 + f(Depth_i) - 12.3 + \varepsilon_i \qquad \varepsilon_i \sim N(0, 5.9^2) \tag{3.15B}$$

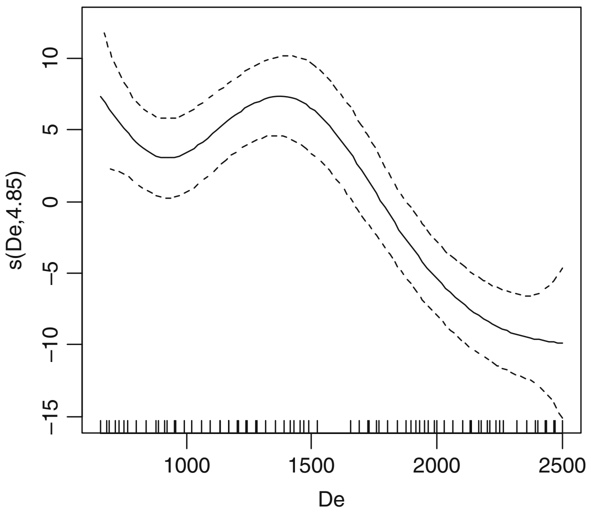

Fig. 3.13 Estimated smoothing curve. The *x*-axis shows the values of depth (*vertical lines*) and the *y*-axis the contribution of the smoother to the fitted values. The solid line is the smoother and the dotted lines 95% confidence bands. The *little vertical lines* along the *x*-axis indicate the depth values of the observations

The values 19.2 and 12.3 can be subtracted to give a new intercept of 6.9. In both cases, the smoothing function *f(Depth$_i$)* is the solid curve in Fig. 3.13. The mgcv package has several useful tools to visualise the results. The following two commands produce the 3-dimensional plot in Fig. 3.14.

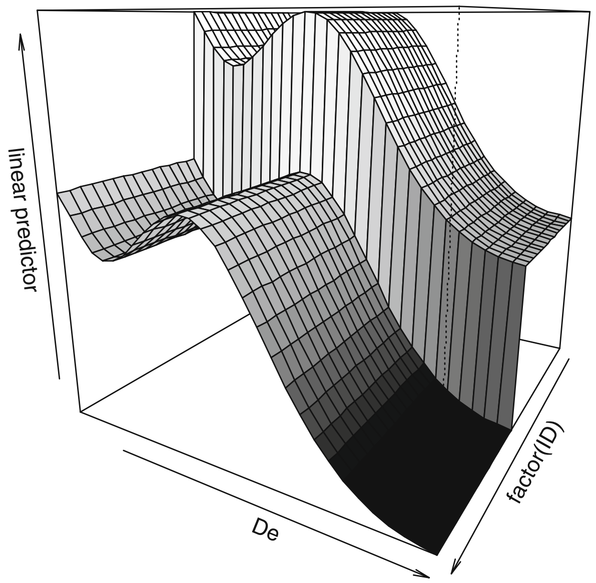

Fig. 3.14 Three-dimensional graph showing the fitted values for both stations. The graph shows the effect of the nominal variable ID (representing station)

```
> par(mar = c(2, 2, 2, 2))
> vis.gam(M4, theta = 120, color = "heat")
```

The `par` command adjusts the white space around the figure and the `vis.gam` function visualises what the gam is doing; it produces two parallel smoothing curves with different intercepts. The `theta` option changes the angle the window is viewed from.

Another useful tool is the `gam.check (M4)` command. It produces graphical diagnostics for the model; see Fig. 3.15. These can be used to (i) assess normality (the QQ-plot and the histogram), (ii) homogeneity (residuals versus fitted values, (also called the linear predictor for the Gaussian distribution with identity link – see Chapter 7)), and (iii) model fit (fitted values versus observed values). In this case, the results in this graph do not look very promising. There are clear patterns in the right two panels. Before concluding that more complicated models (e.g. additive mixed modelling) are needed, we should first try to extend the current model with more covariates, or as in this case, add an interaction term between depth and station.

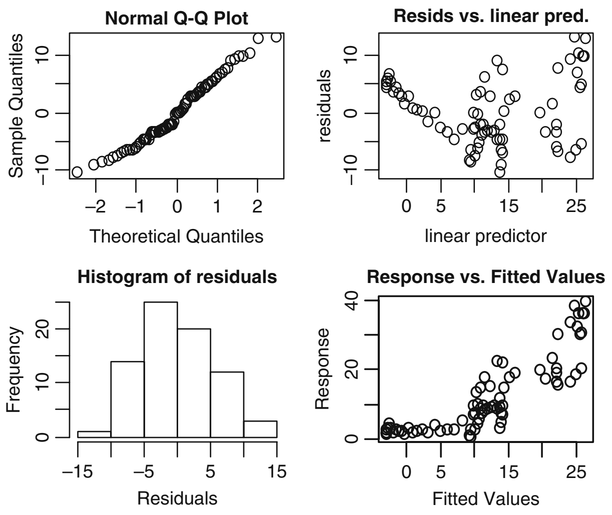

Fig. 3.15 Validation tools for the GAM model that contains one smoother for depth and a nominal variable ID (= station). The QQ-plot and the histogram are used to assess normality and the residuals versus fitted values homogeneity. The response against fitted values should ideally show a straight line

3.4.1 Interaction Between a Continuous and Nominal Variable

Recall that the model in Equation (3.14) assumes that both stations have the same depth-source relationship. However, the scatterplots in Fig. 3.12 clearly indicate that there are differences in this relationship per station. The term `factor(ID)` in the

GAM is only adding or subtracting a constant value to the smoother; it does not allow for a *change* in the source-depth relationship. In a GAM, interaction is not the same interaction we know from linear regression. To understand this, we first write down the R code required for the 'interaction' in the mgcv package. There are two ways of doing this. The first option is as follows.

```
> M5<-gam(So ~ s(De)+
                s(De, by = as.numeric(ID == 13)) +
                factor(ID), subset = I1)
> anova(M5)

Parametric Terms:
           df     F p-value
factor(ID)  1 573.6  <2e-16

Approximate significance of smooth terms:
                              edf  Est.rank     F p-value
s(De)                        8.073    9.000 98.88  <2e-16
s(De):as.numeric(ID == 13) 6.696    9.000 48.39  <2e-16
```

The `s(De)` part applies a smoother along depth for all data points. Hence, it represents the overall depth effect at both stations. The second smoother along depth contains a `by` argument. The `as.numeric` is used to convert the vector with TRUE (an observation is from station 13) and FALSE (it is not from station 13) into a vector with ones (station 13) and zeros (not from station 13). This smoother is then only using the observations for which there is a 1 in the `by` command. In this case, the second depth smoother represents the deviation at station 13 from the overall source–depth relationship. The output from the `anova` indicates that the second depth smoother is highly significant. Hence, there are different depth patterns at each station. To see what the model is doing, both smoothers are plotted in Fig. 3.16. The smoother in the left panel represents the overall source–depth relationship, and the right panel the deviation from this at station 13. Hence, for station 13, you need to add up both the smoothers, and add the contribution from the factor to get the fitted values. Comparing Figs. 3.12 and 3.16 should give some clues what the second smoother (obtained with the `by` command) is doing; it is adjusting the pattern along the depth gradient for the second station.

The graphical results obtained by the `gam.check(M5)` command are given in Fig. 3.17 and looks considerably better than in Fig. 3.15, especially the right two panels. However, the variation for larger fitted values (nearer the surface) seems to be larger than for the smaller fitted values (at deeper depths). Biologically, this makes sense, but it does violate the homogeneity assumption. Tools to solve this for these data are discussed in case study 17.

But which of the models is better? Although this question can be readily answered in favour of the second model based on the model validation plots (compare Figs. 3.15 and 3.17), it would be nice if we could also compare the models using a test or selection criteria as the graphical evidence may not always be as clear as it

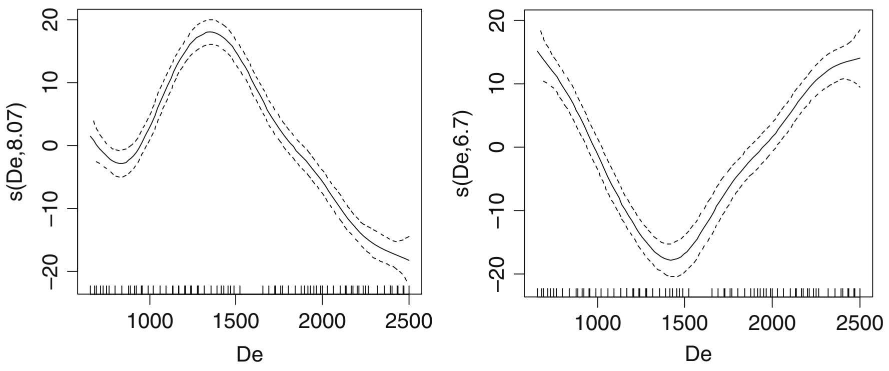

Fig. 3.16 Estimated smoothing curves for the model that contains one depth smoother for both stations (*left panel*) and a depth smoother using the by command, representing the deviation at station 13 (*right panel*)

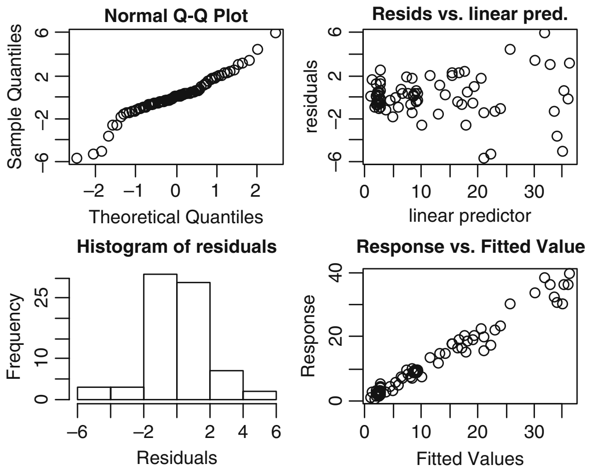

Fig. 3.17 Validation tools for the GAM model that contains one smoother for depth, a smoother for depth that represents the deviation at station 13, and a nominal variable ID. The QQ-plot and the histogram are used to assess normality, and the residuals versus fitted values homogeneity. The response versus fitted values should ideally show a *straight line*

is in this case. There are many ways of doing this. The first option is with the Akaike Information Criteria (AIC), which measures goodness of fit and model complexity. The lower the AIC, the better the model fits the data. The AIC can be obtained by the commands `AIC(M4)` and `AIC(M5)` and gives the values 488.56 and 345.26, respectively, clearly favouring the second model. A nearly identical option is to use the generalised cross validation score, indicated by GCV in the output and obtained by the `summary` command. For the first model, it is 38.80 and for the second model (not presented above) it is 6.03, giving the same conclusion as using the AIC. The third option is to use the F-statistic and associated p-value for the smoother using the by option, obtained by `summary(M5)`. These values are not presented here, but give $p < 0.001$. Yet, another option is to compare the model with and without the second smoother and apply an *F*-test (Wood, 2006). This is done with the command

```
> anova(M4, M5, test = "F")

Analysis of Deviance Table
Model 1: So ~ s(De) + factor(ID)
Model 2: So ~ s(De) + s(De, by = as.numeric(ID == 13))
              + factor(ID)

  Resid. Df Resid. Dev      Df Deviance     F    Pr(>F)
1   68.1507    2402.90
2   58.2309     272.94    9.91  2129.96  45.80  < 0.001
```

The results of this test confirm earlier results. Interpretation of all these tests and resulting *p*-values should be done with care as these *p*-values are 'approximate'. Don't be too confident with a *p*-value of 0.04. We will discuss this aspect further in Section 3.6.

Instead of fitting a model that contains one smoother for depth for all stations, and one that represents the deviation at station 13, it is also possible to fit a GAM with one smoother for the data of station 8, and one smoother for the data of station 13; both smoothers would use the `by` option in this case. The R code is as follows.

```
> M6 <- gam(So ~ s(De, by = as.numeric(ID == 8)) +
                 s(De, by = as.numeric(ID == 13)) +
                 factor(ID), subset = I1)
```

Application of this type of models is discussed in various case study chapters. Models constructed with the `by` command are also called variable coefficient models (Hastie and Tibshirani, 1990). Their general statistical model formulation is given by

$$Y_i = \alpha + f_1(X_{1i}) + f_2(X_{2i}) \times X_{3i} + \varepsilon_i$$

Y_i is the response variable, α the intercept, $f_1(X_{1i})$ is a smoother, and the values of the smoother $f_2(X_{2i})$ are multiplied with the values of X_{3i}. Note that we are not multiplying X_{2i} itself with X_{3i}, but the smoother. Hence, the equations for the models in `M5` and `M6` are, respectively,

$$M5 : Sources_i = \alpha + f_1(Depth_i) + f_2(Depth_i) \times ID_{13,i} + \varepsilon_i$$
$$M6 : Sources_i = \alpha + f_1(Depth_i) \times ID_{8,i} + f_2(Depth_i) \times ID_{13,i} + \varepsilon_i$$

The variable $ID_{8,i}$ is 1 if an observation is from station 8 and 0 else. The variable $ID_{13,i}$ is 1 if an observation is from station 13 and 0 else.

3.5 GAM Example 2: Dealing with Collinearity

The previous example was relatively simple as there were only two explanatory variables. This makes the model selection process relatively simple; just compare the model with and without the interaction term (obtained by the `by` option). Now we consider an example that contains considerably more explanatory variables. As explanatory variables increase, it becomes important to avoid using collinear explanatory variables in GAM. If you do use explanatory variables that are highly correlated with each other, using GAMs becomes a rather confusing experience. Data exploration tools to identify collinearity are discussed in Appendix A and can also be found in Zuur et al. (2007).

In this section, we show how confusing GAM becomes if you ignore this step of avoiding correlated explanatory variables. We use a plant vegetation data set for illustration. Sikkink et al. (2007) analysed grassland data from a monitoring programme from two temperate communities in Montana, USA: Yellowstone National Park and National Bison Range. The aim of the study was to determine whether the biodiversity of these bunchgrass communities changed over time and if they did, whether the changes in biodiversity relate to specific environmental factors. Here, we use the Yellowstone National Park data. Sikkink et al. (2007) quantified biodiversity using species richness to summarise the large number of species: ninety species were identified in the study. Richness is defined as the *different number* of species per site. The data were measured in eight different transects and each transect was measured repeatedly over time with time intervals of about four to ten years. For the moment, we ignore the temporal aspects of the data. And, instead of using all 20 or so explanatory variables, we use only those explanatory variables that Sikkink et al. (2007) identified as important. Figure 3.18 shows a scatterplot of all the variables used in this section. The response variable is species richness for the 64 observations, and the explanatory variables are rock content (ROCK), litter content (LITTER), bare soil (BARESOIL), rainfall in the fall (FallPrec), and maximum temperature in the spring (SprTmax). The correlation between ROCK and LITTER is reasonably high with a Pearson correlation of –0.7.

The following R code was used to run a GAM on these data. We ignore the fact that a Poisson distribution might be more appropriate, and that there are temporal aspects in this data set.

```
> library(AED); data(Vegetation)
> library(mgcv)
> M7 <- gam(Richness ~ s(ROCK, bs = "cs") +
          s(LITTER, bs = "cs") + s(BARESOIL, bs = "cs") +
          s(FallPrec, bs = "cs") + s(SprTmax, bs = "cs"),
          data = Vegetation)
```

The only new bit, compared to the previous section, is the `bs = "cs"` part. It ensures that a regression spline with shrinkage is applied. Shrinkage means that a smoother can have 0 degrees of freedom. All smoothers that have 0 degrees of

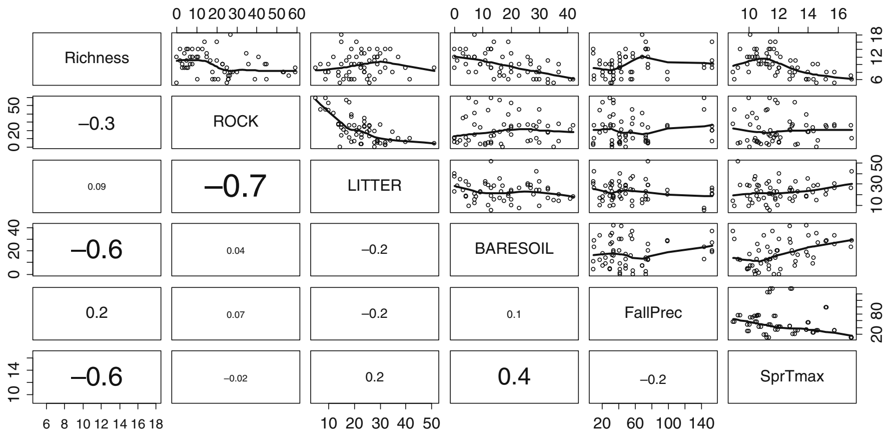

Fig. 3.18 Scatterplot of richness (response variable), rock content, litter content, baresoil, rainfall in the fall, and maximum temperature in the spring for the vegetation data set. The upper panels show scatterplots with a smoother added to visualise the patterns, and the lower panels contain the Pearson correlation coefficients. The font of the correlation coefficient is proportional to its estimated value. The code to produce this graph is given on the book website and is based on code presented in the help file of the `pairs` function

freedom can be dropped simultaneously from the model. The relevant output from the `anova(M7)` command is presented below, and shows that this is not happening.

```
Approximate significance of smooth terms:
                edf Est.rank     F p-value
s(ROCK)       1.750     4.000 4.634 0.00367
s(LITTER)     1.865     4.000 2.320 0.07376
s(BARESOIL)   4.167     9.000 2.991 0.00820
s(FallPrec)   4.699     9.000 2.216 0.04150
s(SprTmax)    5.103     9.000 3.508 0.00286
```

Not all terms are significant at the 5% level, and the estimated smoothing curves, with the wide confidence bands for some of the terms confirm this (Fig. 3.19). We can now do two things: either leave the model as it is or apply a backwards selection approach to find the optimal model. We prefer to use a model with only significant terms; so we go for the second option. We can either use a selection criterion like the AIC or CGV in a stepwise backwards selection process, like in linear regression or use hypothesis testing procedures. The first approach is better than the second, but the second approach takes less time; it drops the least significant term, refits the model, and continues this process until all terms are significant. If you do this, you end up with a GAM that only contains smoothing terms of ROCK, BARESOIL, and SprTmax: the estimated smoothing curves are presented in Fig. 3.20. The shape of the smoothers suggest that the higher the rock content, the lower the species richness, and the same relationship applies to bare soil.

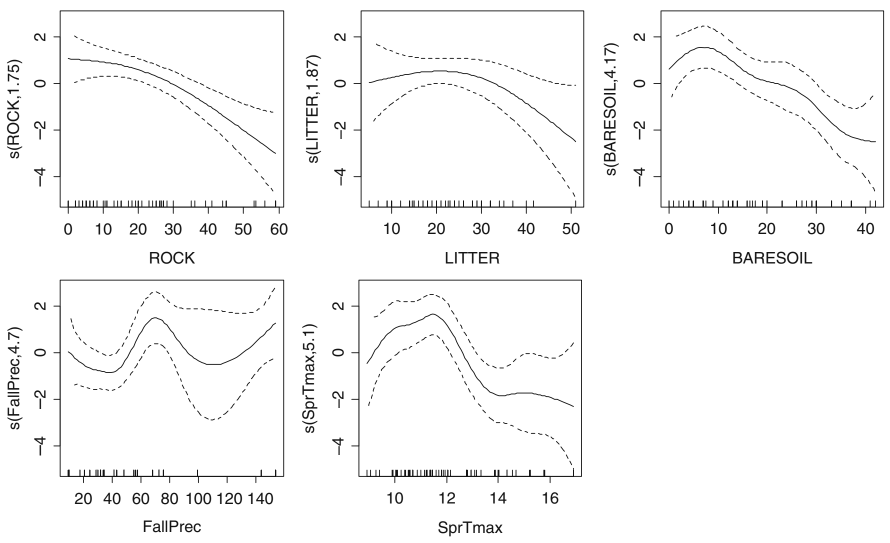

Fig. 3.19 Estimated smoothing curves for the GAM model containing all explanatory variables

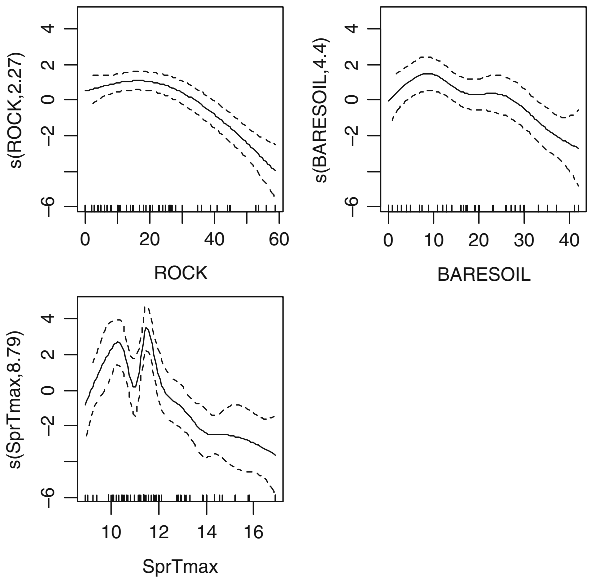

Fig. 3.20 Estimated smoothing curves for the optimal GAM model

The smoother for maximum temperature is rather difficult to interpret. During the model selection process it took various different shapes. This may indicate that the smoother for SprTmax may represent patterns that are actually due to other variables. Draper and Smith (1998) discuss various tools in linear regression analysis to identify the presence of collinearity. One of their recommendations is to see whether slopes change radically if a term is omitted as this is an indication for the presence of collinearity. Although we do not have slopes here, the amount of smoothing (or better: the associated degrees of freedom) for SprTmax did show considerable changes during the selection process (from 4 to 8 edf). This is probably caused by collinearity.

3.6 Inference*

In the previous section, the software gave us p-values for smoothers (these were derived from F-tests). There is a little catch here; these p-values are approximate and should be used with care.

The principle behind confidence intervals in GAM is relatively simple, and follows linear regression. A multiple linear regression model is given by

$$Y_i = \alpha + \beta_1 \times X_{1i} + \cdots + \beta_q \times X_{qi} + \varepsilon_i$$

Using matrix algebra, this can also be written as

$$\mathbf{Y} = \mathbf{X} \times \boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

The vector **Y** contains all observed data, the matrix **X** all q explanatory variables and its first column contains only ones (for the intercept), $\boldsymbol{\beta}$ contains all regression parameters (including the intercept α), and $\boldsymbol{\varepsilon}$ all the residuals ε_i. Using this notation, the ordinary least square estimator for $\boldsymbol{\beta}$ can be found using

$$\hat{\mathbf{b}} = (\mathbf{X} \times \mathbf{X})^{-1} \times \mathbf{X}' \times \mathbf{Y}$$

If you are not familiar with matrix algebra, then this may look intimidating, but this solution can be found in most statistical textbooks. Once you have the estimated regression parameters, the fitted values are given

$$\hat{\mathbf{Y}} = \mathbf{X} \times \hat{\mathbf{b}} = \mathbf{X} \times (\mathbf{X} \times \mathbf{X})^{-1} \times \mathbf{X}' \times \mathbf{Y} = \mathbf{H} \times \mathbf{Y}$$

The reason that we show all this matrix algebra is because of $\mathbf{H} \times \mathbf{Y}$. **H** is also called the hat matrix, and it plays an important role as it goes straight into equations for standard errors of predicted values, and it is also used to determine degrees of freedom.

The LOESS smoother and the polynomial and cubic regression splines are all local regression models, and in fact, can all be written in exactly the same form as the linear regression model:

*Means that it is difficult and that it can be skipped upon first reading.

$$\hat{\mathbf{Y}} = \mathbf{S} \times \mathbf{Y}$$

Instead of using an **H**, we used a matrix **S**. Depending on the type of smoother, the computer software will fill in the elements of **S**. An example can be found in Section 7.4 in Zuur et al. (2007) for a moving average smoother. Because polynomial and cubic regression splines are linear regression models, we can follow exactly the same theory as in linear regression. Hence, the variance of the fitted values are given by

$$\mathrm{var}(\hat{\mathbf{Y}}) = \sigma^2 \times \mathbf{S} \times \mathbf{S}'$$

where σ^2 is the variance of the Y_i. This all follows immediately from linear regression theory. The matrix **S** and its trace are then used to calculate degrees of freedom and residual degrees of freedom.

Recall that we decided to work with the smoothing splines (or penalised splines). Broadly speaking, the same approach can be followed, although the matrix **S** is now more complex, see Keele (pg. 75, 2008). Basically, the solution of Equation (3.13) is again of the form

$$\hat{f} = \mathbf{S} \times \mathbf{Y}$$

but this **S** is more complicated compared to regression splines as it is also a function of λ. The way **S** is used in the calculation of confidence bands, degrees of freedom, etc., is nearly the same, though. The 'nearly' bit refers to the fact the *p*-values behave reasonable well for smoothing splines with known degrees of freedom, but if these are estimated (e.g. using cross-validation), they can be misleading. This is because the uncertainty due to estimating the λs is neglected. Wood (2006) mentioned that based on limited simulation experience, the *p*-values close to 0.05 can be around half of their correct value when the null hypothesis is true. This means that smoothers with *p*-values smaller than 0.001 can be trusted, and we can also trust the *p*-value if it is 0.2, 0.5, or 0.8. It is the smoother with a *p*-value of 0.02, 0.03, 0.04, or 0.05 that gives trouble. The same holds for *F*-tests comparing nested GAMs. If this is a serious issue for your data, then you could do bootstrapping to get better *p*-values. Chapter 8 in Keele (2008) gives a nice introduction, and a detailed algorithm is presented in Davison and Hinkley (1997). When writing this chapter, we were tempted to add a bootstrapping example; however, it would only duplicate Chapter 8 from Keele (2008).

3.7 Summary and Where to Go from Here?

In this chapter, we started with LOESS smoothers. Recall that a weighted linear (or quadric) regression model is applied on all data in a window around the target value. The amount of smoothing is determined by the size of the window, also called the span width. We then introduced splines; simple linear splines, quadratic and cubic splines. The gradient is divided in segments using a certain number of knots, and a linear, quadratic, or cubic polynomial model is fitted on the data in each segment. Certain conditions are imposed ensuring a smooth connection at the edges. Finally, smoothing splines are introduced as the best option. It minimises the penalised least squares criterion. Table 3.1 summarises all smoothers.

In Section 3.6, we discussed that the p-values thrown at you by the software are approximate for smoothing splines. You can safely make biological statements based on smoothers with a p-value of 0.001 (or smaller) or a p-value of 0.1 (or larger), but be very careful with smoothers for which p is just below the magical 0.05 level. If you really want to say something about such smoothers, apply bootstrapping.

Some of the problems encountered for linear regression can also cause trouble in additive modelling, e.g. violation of independence, heterogeneity, and nested data. In fact, we have been cheating in most GAM examples presented in this chapter. For example, we used vegetation data that were measured repeatedly over time. The analysis carried out assumes (implicitly) that there is no correlation between observations made at the same transect. The lags between two observations at the same transect is approximately 4–10 years.

The same holds for the bioluminescent data; measurements at the surface may be related to measurements at deeper depths, simply because particles tend to move down from the surface to the bottom of the ocean. It is like rain; if it rains at 100 m, it

Table 3.1 Summary of all smoothers discussed in Chapter 3

Name of smoother	What is it?	Relevant options for smoothing
LOESS	Weighted linear regression on a window around the target value. Move target value.	Size of the span
Simple linear regression spline	Gradient is divided in segments using knots. Fit bivariate linear regression model on each segment.	Number and location of knots
Quadratic and cubic regression splines	Gradient is divided in segments using knots. Fit a quadratic or cubic polynomial on each segment, and ensure a smooth connection at the knots.	Number of knots, location of knots, and degree of polynomial
Smoothing splines (alias penalised splines)	Gradient is divided in a large number of segments. Fit a cubic polynomial model on each segment, and ensure a smooth connection at the knots. Minimise the penalised sum of squares in Equation (3.9).	Use large number of knots. Find optimal value of λ

will probably also rain at 50 m. Hence, there is also a correlation issue here. On top of this, we noticed that there is violation of homogeneity along the depth gradient (more variation towards the surface). Then there is another issue if we analyse data of all 19 stations; these are nested data. Data from the same station may be more similar than data from different stations.

All these issues (heterogeneity, nested data, and correlation) are addressed in Chapters 4, 5, 6, and 7.