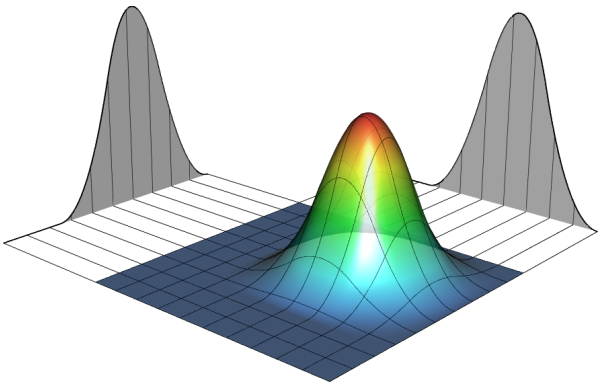
apts.ac.uk
Academy for PhD Training in Statistics

Adrian Bowman, Ludger Evers
School of Mathematics & Statistics
The University of Glasgow

Nonparametric Smoothing

Lecture Notes



1

Introduction

1.1 What this course is about

This APTS course will cover a variety of methods which enable data to be modelled in a flexible manner. It will use and extend a variety of topics covered in earlier APTS courses, including

– linear models, including the Bayesian version;
– generalised linear models;
– R programming;
– Taylor series expansions and standard asymptotic methods.

The main emphasis will be on regression settings, because of the widespread use and application of this kind of data structure. However, the material of the first chapter will include the simple case of density estimation, also covered in the preliminary material, to introduce some of the main ideas of nonparametric smoothing and to highlight some of the main issues involved.

As with any statistical topic, a rounded treatment involves a variety of approaches, including

– clear understanding of the underlying concepts;
– technical understanding of methods, with an exploration of their properties;
– appreciation of the practical computational issues;
– some knowledge of the tools available to fit relevant models in R;
– understanding of how these models can bring insight into datasets and applications.

The aim is to reflect all of these aspects in the course, but to varying degrees in different sections. There will not be time to cover all the material in the notes and some of the material is intended to provide pointers to topics which it might be of interest to explore in the context of your own research.

1.2 Broad concepts and issues of smoothing

The simple case of density estimation highlights features and issues which are common to a wide range of problems involving the estimation of functions, relationships or patterns which are *nonparametric* but *smooth*. The term nonparametric is used in this context to mean that the relationships or patterns of interest cannot be expressed in specific formulae which involved a fixed number of unknown parameters. This means that the parameter space is the space of functions, whose dimensionality is infinite. This takes us outside of the standard framework for parametric models and the main theme of the course will be to discuss how we can do this while producing tools which are highly effective for modelling and analysing data from a wide variety of contexts and exhibiting a wide variety of structures.

On a side note, the term nonparametric is sometimes used in the narrower setting of simple statistical methods based on the ranks of the data, rather than the original measurements. This is not the sense in which it will be used here.

Further details on density estimation are given in Silverman (1986), Scott (1992), Wand and Jones (1995) & Simonoff (1996).

The issues raised by our brief discussion of density estimation include

- how to construct estimators which match the type of data we are dealing with;
- how to find a suitable balance between being faithful to the observed data and incorporating the underlying regularity or smoothness which we believe to be present;
- how to construct and make use of suitable inferential tools which will allow the models to weigh the evidence for effects of interest, in a setting which takes us outside of standard parametric methods.

These broad issues will be explored in a variety of contexts in the remainder of the course.

1.3 Nonparametric regression

Regression is one of the most widely used model paradigms and this will be the main focus in the remainder of the course. Here is an example which will be used to illustrate the initial discussion.

Example 1.1 (Great Barrier Reef data). A survey of the fauna on the sea bed lying between the coast of northern Queensland and the Great Barrier Reef was carried out. The sampling region covered a zone which was closed to commercial fishing, as well as neighbouring zones where fishing was permitted. The variables are:

`Zone`	an indicator for the closed (1) and open (0) zones
`Year`	an indicator of 1992 (0) or 1993 (1)
`Latitude`	latitude of the sampling position
`Longitude`	longitude of the sampling position
`Depth`	bottom depth
`Score1`	catch score 1
`Score2`	catch score 2

The details of the survey and an analysis of the data are provided by Poiner et al. (1997), *The effects of prawn trawling in the far northern section of the Great Barrier Reef*, CSIRO Division of Marine Research, Queensland Dept. of Primary Industries. ◁

The relationship between catch score (Score1) and longitude is of particular interest because, at this geographical location, the coast runs roughly north-south and so longitude is a proxy for distance offshore. We might therefore reasonably expect the abundance of marine life to change with longitude. The left-hand panel of figure 1.1 summarises this in a simple linear regression which captures much of this relationship. However, if we allow our regression model to be more flexible then a more complex relationship is suggested in the right hand panel, with a broadly similar mean level for some distance offshore followed by a marked decline, possibly followed by some levelling off thereafter. This gives valuable informal and graphical insight into the data but how can flexible regression models can be constructed and how can we use them to evaluate whether there is really evidence of non-linear behaviour in the data?

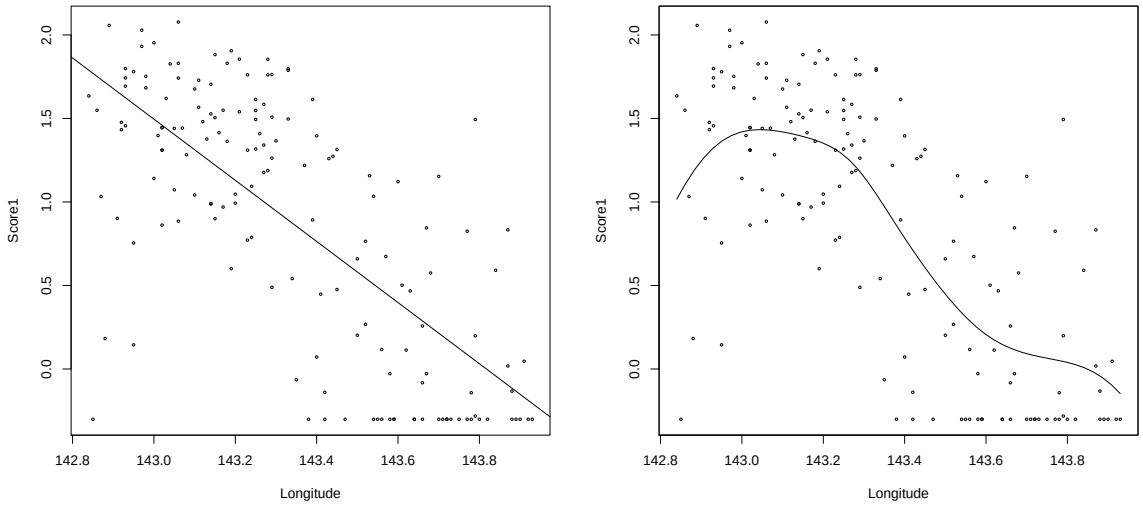


Figure 1.1. Linear model and smooth function fitted to the Great Barrier Reef data.

1.3.1 A local fitting approach

A simple nonparametric model has the form

$$y_i = m(x_i) + \varepsilon_i,$$

where the data (x_i, y_i) are described by a smooth curve m plus independent errors ε_i. One approach to fitting this is to take a model we know and fit it locally. For example, we can construct a *local linear regression*. This involves solving the least squares problem

$$\min_{\alpha,\beta} \sum_{i=1}^{n} \{y_i - \alpha - \beta(x_i - x)\}^2 \, w(x_i - x \, ; h)$$

and taking as the estimate at x the value of $\hat{\alpha}$, as this defines the position of the local regression line at the point x. This has an appealing simplicity and it can be generalised quite easily to other situations. This was the approach used to produce the nonparametric regression of the Reef data in the plot above.

There is a variety of other ways in which smooth curve estimates can be produced and a further approach is outlined in the next section. It can sometimes reasonably be argued that the precise mechanism usually isn't too important and can be chosen for convenience.

1.3.2 Basis function approaches

Basis function approachaes are not based on local weights, but based on expanding the design matrix used in linear regression. To fix notation, we quickly state the simple linear regression model

$$\mathbb{E}(Y_i) = m(x_i) = \beta_0 + \beta_1 x_i \qquad \text{for } i = 1, \ldots, n,$$

or equivalently, in matrix-vector notation,

$$\mathbb{E}(\mathbf{y}) = \mathbf{B}\boldsymbol{\beta} \qquad \text{with } \mathbf{y} = (Y_1, \ldots, Y_n)^\top \text{ and } \mathbf{B} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

Basis function approaches effectivly consist of introducing functions of $\mathbf{x}$ (other than just the identity) into the design matrix $\mathbf{B}$.

Basis approaches to function approximation have a very long history. One of the first was Fourier series, which is based on the expansion

$$m(x_i) \approx \frac{a_0}{2} + \sum_{j=1}^{r} a_j \cos\left(\frac{2\pi j x_i}{P}\right) + b_j \sin\left(\frac{2\pi j x_i}{P}\right),$$

where $x_i \in (0, P)$. This approximation corresponds to using the design matrix

$$\mathbf{B} = \begin{pmatrix} \frac{1}{2} & \cos\left(\frac{2\pi x_1}{P}\right) & \sin\left(\frac{2\pi x_1}{P}\right) & \dots & \cos\left(\frac{2\pi r x_1}{P}\right) & \sin\left(\frac{2\pi r x_1}{P}\right) & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2} & \cos\left(\frac{2\pi x_n}{P}\right) & \sin\left(\frac{2\pi x_n}{P}\right) & \dots & \cos\left(\frac{2\pi r x_1}{P}\right) & \sin\left(\frac{2\pi r x_1}{P}\right) & \end{pmatrix}$$

with $\boldsymbol{\beta} = (a_0, a_1, b_1, \dots, a_r, b_r)$.

Panel (a) in figure 1.2 shows the basis functions of a Fourier basis with $r = 3$.

Each basis function in a Fourier expansion has effects across the entire range of the data. Another approach with this – as we will find out unfortunate – property is polynomial regression, which correponds to the expansion

$$m(x_i) \approx \beta_0 + \beta_1 x_i + \ldots + \beta_r x_i^r,$$

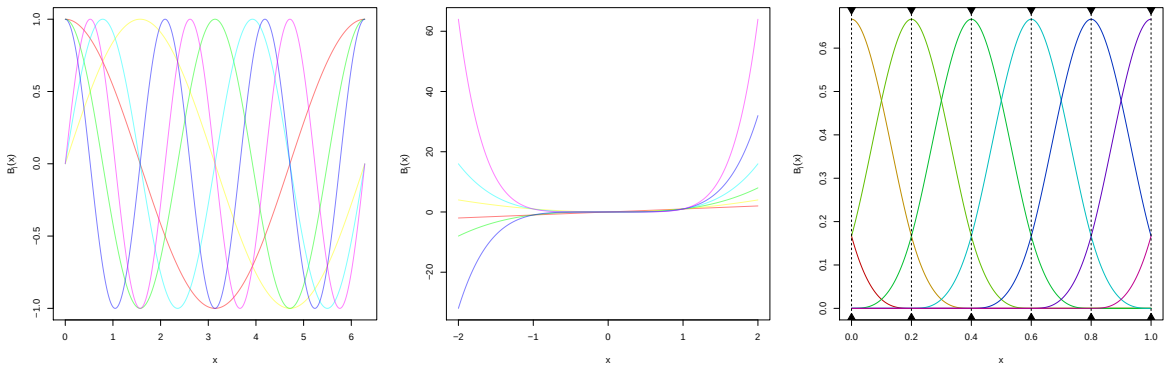
corresponding to the design matrix

$$\mathbf{B} = \begin{pmatrix} 1 & x_1 & \dots & x_1^r \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^r \end{pmatrix}.$$

With both of the aforementioned approaches each basis function acts "globally" rather than "locally", thus fitting data well in one part of the sample space can create artefacts elsewhere. We will thus study more locally-acting bases in chapter 3, as the one shown in panel (c).

One key advatage of basis-expansion methods is that we can then estimate $\boldsymbol{\beta}$ using the same techniques as used in multiple linear regression, i.e. the least-squares estimator is

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{y}$$



(a) Fourier basis (b) Polynomial basis (monomials) (c) B-spline basis

Figure 1.2. Examples of a Fourier basis, a polynomial basis consisting of monomials and a B-spline basis.

1.4 Further illustrations of smoothing

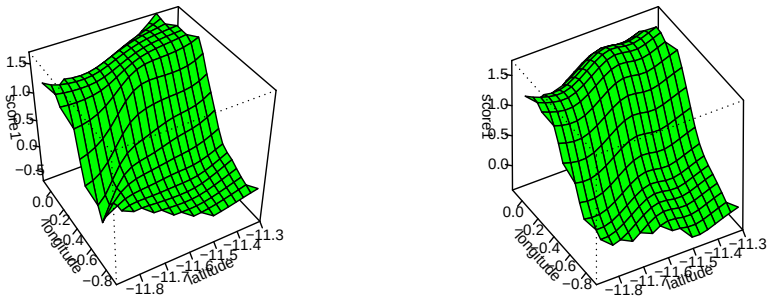
This section gives a few brief illustrations of how the ideas outlined in simple settings can be extended to much more complex situations and data structures. Some of these will be revisited in greater detail later in the course.

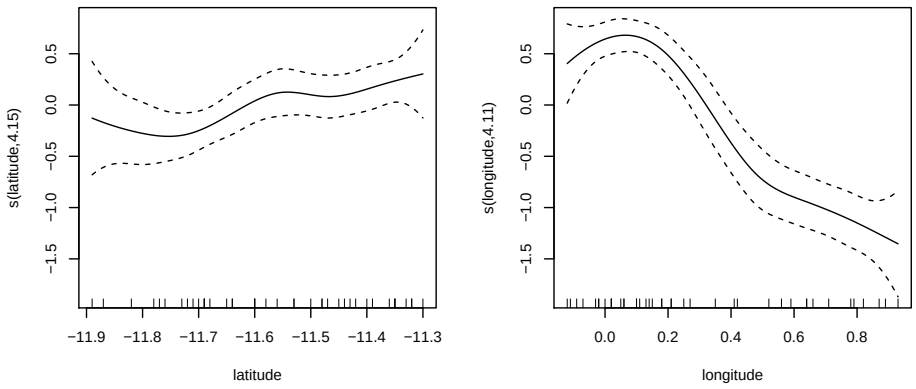
1.4.1 Regression with more than one covariate

It is rare to have problems which involve only a single covariate. For the Reef data a natural extension is to look at the relationship between the catch score and both latitude (x_1) and longitude (x_2), in a model

$$y_i = m(x_{1i}, x_{2i}) + \varepsilon_i.$$

the top left hand panel of the plot below shows the effect of this. The effect of longitude dominates, as we see from the earlier nonparametric regression. However, a small effect of latitude is also suggested. The methods by which surfaces of this type can be constructed will be discussed in the next two chapters.





It would be unrealistic to generalise this much further, by modelling additional covariates through functions of ever-increasing dimension. A very powerful approach is to construct *additive models* of the form

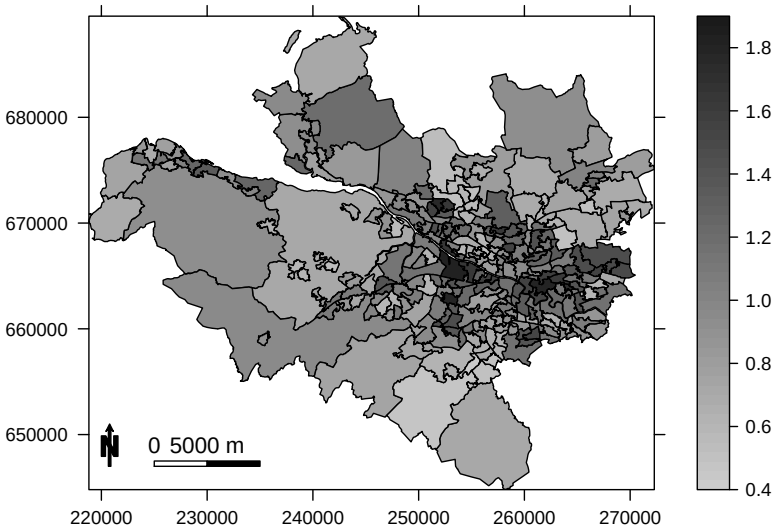
$$y_i = m_1(x_{1i}) + m_2(x_{2i}) + \varepsilon_i,$$

where the component functions m_1 and m_2 describe the separate and additive effects of the two covariates. Estimated additive model components, and their combined effects as an additive surface, are displayed in the other panels of the figure above. Methods for fitting models of this type will also be discussed later.

1.4.2 Areal data

Data quantifying population-level summaries of disease prevalence for n non-overlapping areal units are available from both the English and Scottish Neighbourhood Statistics databases. They are used in many different applications, including quantifying the effect of an exposure on health, and identifying clusters of areal units that exhibit elevated risks of disease. The health data are denoted by $\mathbf{Y} = (Y_1, \ldots, Y_n)$ and $\mathbf{E} = (E_1, \ldots, E_n)$, which are the observed and expected numbers of disease cases in each areal unit. The covariates are denoted by an $n \times p$ matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, and could include environmental exposures or measures of socio-economic deprivation.

One example concerns the prevalence of respiratory disease in Glasgow in 2010, where interest focusses on the hospitalisation risk $\text{SIR}_k = Y_k/E_k$. The data are displayed in the plot below.



A suitable statistical model is

$$\begin{aligned} Y_k &\sim \text{Poisson}(E_k R_k), \\ \log(R_k) &= \mathbf{x}_k^{\mathrm{T}}\boldsymbol{\beta} + \phi_k, \\ \phi_k|\boldsymbol{\phi}_{-k}, \tau^2, W &\sim \text{N}\left(\frac{\sum_{i=1}^n w_{ki}\phi_i}{\sum_{i=1}^n w_{ki}}, \frac{\tau^2}{\sum_{i=1}^n w_{ki}}\right), \end{aligned}$$

where

– R_k quantifies disease risk in area k.

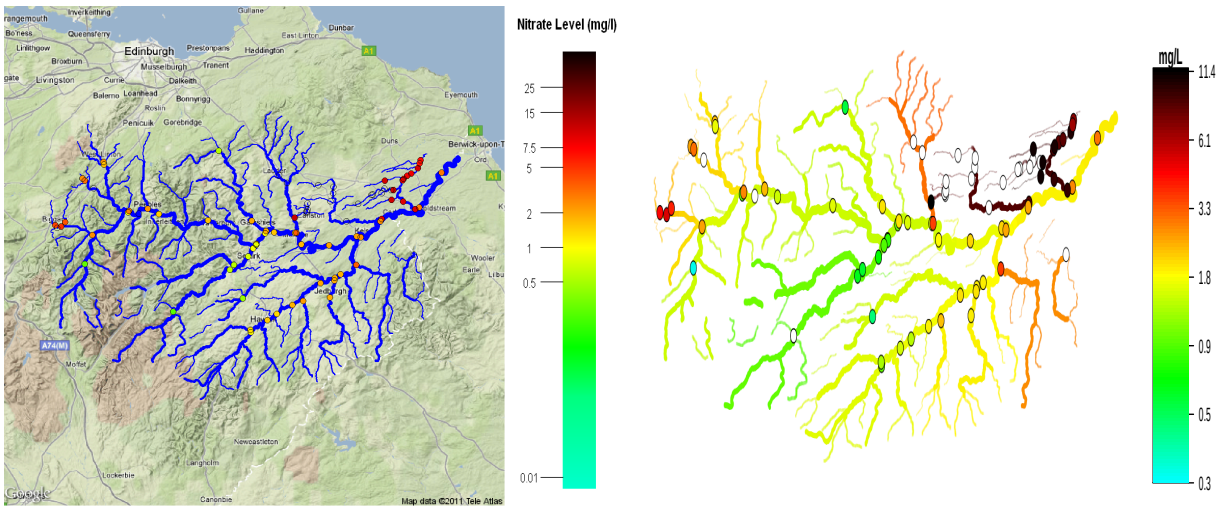
– $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$ are random effects that model residual spatial autocorrelation and improve estimation of disease risk by smoothing the risks across neighbouring areas.

– Commonly, conditional autoregressive (CAR) models are used for this smoothing, where $W = (w_{ki})$ is a binary $n \times n$ neighbourhood matrix.

Material kindly provided by Duncan Lee, University of Glasgow.

1.4.3 Network data

Models for spatial data often involve some form of smoothing, either implicitly or explicitly. An interesting forms of spatial data arises from river networks, where models for the variation in measurements should respect both the spatial pattern of the interconnecting water channels as well as the mixing effect of confluence points where channels meet. The data below refer to nitrate pollution in the River Tweed. The point measurements on the left can be used to construct a spatial model for the whole network, whose predictions are shown in the panel on the right.



This example arises from joint work with Alastair Rushworth, David O'Donnell, Marian Scott, Mark Hallard (SEPA).

2

Density estimation

2.1 Motivating example

A probability density function is a key concept through which variability can be expressed precisely. In statistical modelling its role is often to capture variation sufficiently well, within a model where the main interest lies in structural terms such as regression coefficients. However, there are some situations where the shape of the density function itself is the focus of attention. The example below illustrates this.

Example 2.1 (Aircraft data). These data record six characteristics of aircraft designs which appeared during the twentieth century. The variables are:

`Yr`	year of first manufacture
`Period`	a code to indicate one of three broad time periods
`Power`	total engine power (kW)
`Span`	wing span (m)
`Length`	length (m)
`Weight`	maximum take-off weight (kg)
`Speed`	maximum speed (km/h)
`Range`	range (km)

A brief look at the data suggests that the six measurements on each aircraft should be expressed on the log scale to reduce skewness. Span is displayed on a log scale in figure 2.1, for Period 3 which corresponds to the years after the second World War. ◁

The pattern of variability shown in both the histogram and the density estimate exhibits some skewness. There is perhaps even a suggestion of a subsidiary mode at high values of log span, although this is difficult to evaluate.

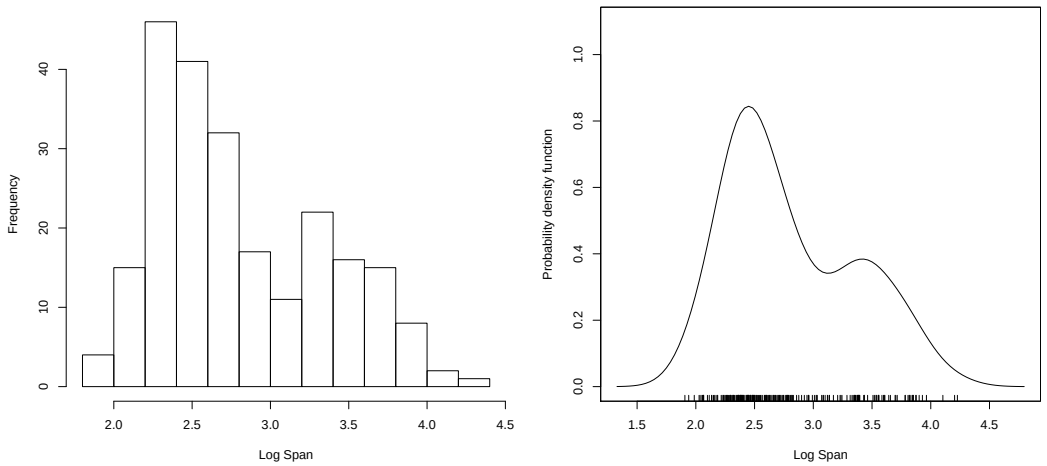


Figure 2.1. Histogram and kernel density estimator for log span of the aircraft data.

2.2 Kernel density estimation

The histogram is a very familiar object. It can be written as

$$\tilde{f}(y) = \sum_{i=1}^{n} I(y - \tilde{y}_i; h),$$

where $\{y_1, \ldots, y_n\}$ denote the observed data, $\tilde{y}_i$ denotes the centre of the interval in which y_i falls and $I(z; h)$ is the indicator function of the interval $[-h, h]$. The form of the construction of $\tilde{f}$ highlights some feature which are open to criticism if we view the histogram as an estimator of the underlying density function. Firstly the histogram is not smooth, when we expect that the underlying density usually will be. Secondly, some information is lost when we replace each observation y_i by the bin mid-point $\tilde{y}_i$. Both of the issues can be addressed by using a density estimator in the form

$$\hat{f}(y) = \frac{1}{n} \sum_{i=1}^{n} w(y - y_i; h),$$

where w is a *kernel function*, whose variance is controlled by the *smoothing parameter* h.

We want to use the data close to y to estimate the density $f(y)$ at y, thus we want the kernel function to have the following properties:

– $w(\delta; h)$ should have a mode at $\delta = 0$ and be symmetric $w(\delta; h) = w(-\delta; h)$.
– $w(\delta; h)$ should decay as we move away from zero, i.e. $\delta w(\delta; h) \to 0$ for $\delta \to \pm\infty$.
We typically require the stronger property that $w(\cdot; k)$ has a finite second moment, i.e. $\int_{-\infty}^{+\infty} \delta^2 w(\delta; h)\, d\delta < \infty$.
– For convenience we want $w(\delta; h)$ to be a valid density in δ, i.e. $\int_{-\infty}^{+\infty} w(\delta; h)\, d\delta = 1$.

The two most popular choices for the kernel function are

– the Gaussian kernel, $w(\delta; h) = \frac{1}{\sqrt{2\pi} \cdot h} \exp\left(-\frac{\delta^2}{h^2}\right)$, and

– the Epanechnikov kernel, $w(\delta; h) = \begin{cases} \frac{3}{4h}\left(1 - \frac{\delta^2}{h^2}\right) & \text{for } -h < \delta < h \\ 0 & \text{otherwise.} \end{cases}$

The Epanechnikov kernel can be shown to minimise the mean square error, however it leads to a non-differentiable density estimate.

– The uniform kernel, $w(\delta; h) = \frac{1}{2h} 21_{(-h;h)}(\delta)$, yields the histogram estimator from above.

Large changes in the value of the smoothing parameter h have large effects on the smoothness of the resulting estimates, as figure 2.2 illustrates.

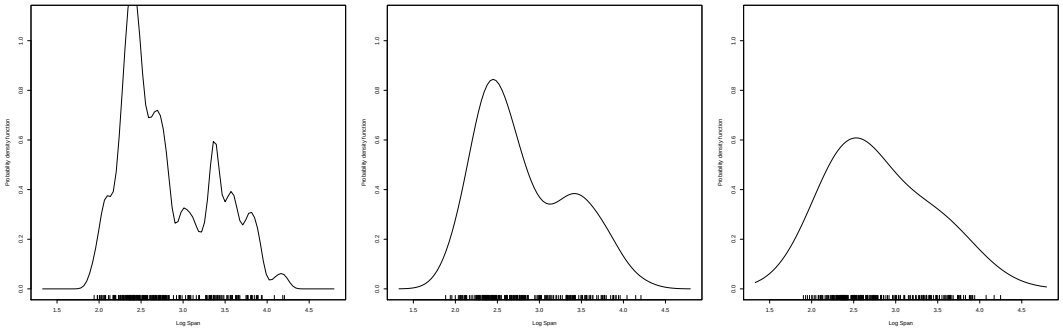


Figure 2.2. Effect of the smoothing parameter h on the estimated density.

One advantage of density estimates is that it is a simple matter to superimpose these to allow different groups to be compared. Figure 2.3 compares the distribution of the log-span for the three different time periods. It is interesting that the 'shoulder' appears in all three time periods.

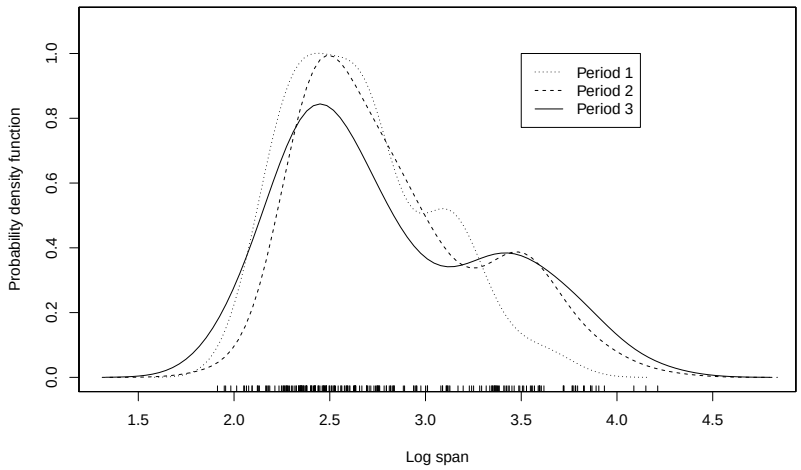


Figure 2.3. Estimated density of the log-span for the three time periods.

2.2.1 Link to the characteristic function

We can understand the kernel density estimate as the density of the sum of two random variables. Denote by $f_n(y)$ the empirical probability mass function of the sample $y_1, \ldots, y_n$, i.e.

$$f_n(y) = \frac{1}{n}|\{i : \ y_i = y\}|$$

The corresponding characteristic function[1] is

$$\phi_{f_n}(t) = \mathbb{E}(\exp(\imath t Y)) = \frac{1}{n}\sum_{i=1}^{n} \exp(\imath t y_i) \tag{2.1}$$

Assume that $\varepsilon \sim w(\cdot; h)$ and then consider the convolution of these two distributions.[2] The convolution has density

$$(f_n * w)(y) = \sum_{i=1}^{n} w(y - y_i; h) \cdot \frac{1}{n},$$

which is exactly the formula for the kernel density estimate.

The characteristic function of the convolution (and thus of the kernel density estimate) is

$$\phi_{f_n * w}(t) = \phi_{f_n}(t) \cdot \phi_w(t) = \frac{1}{n}\sum_{i=1}^{n} \exp(\imath t y_i)\phi_w(t), \tag{2.2}$$

Thus kernel density estimation can be viewed as estimating the characteristic function by multiplyting its empirical estimate by a dampening function $\phi_w(t)$, which makes the estimate more stable for large t.

This link between kernel density estimation and characteristic functions is not just of theoretical interest. Computing the kernel density estimation can be slow for large sample sizes n. For every point y at which we want to evaluate the kernel density estimate we have to evaluate $w(\cdot; h)$ for n times.

If we would like to evaluate the kernel density estimate on a regular grid of 2^m values for some $m \in \mathbb{N}$, then we can exploit (2.2). We can first use the fast Fourier transform to calculate (a discrete approximation to) $\phi_{f_n}(t)$ and $\phi_w(t)$. Then can simply multiply these together and compute the inverse fast Fourier transform to obtain. This is significantly faster than using the usual formula for the kernel density estimate and this is how the R function `density` calculates the kernel density estimate

[1] The characteristic function of a random variable X is $\phi_{f_X}(t) = \mathbb{E}(\exp(\imath t X))$ is the Fourier transform of its probability density function.

[2] The convolution of two distributions is the distribution of the sum of two independent random variables with those two distributions. The convolution of two continuous distributions with densities $f_X(\cdot)$ and $f_Y(\cdot)$ is given by $(f_X * f_Y)(z) = \int_{-\infty}^{+infty} f_X(z-y)f_Y(y)\ dy$. If X is continuous and Y is discrete with p.m.f. $f_Y(y)$ then the distribution of their sum has probability density function $(f_X * p_Y)(z) = \sum y \in R_Y f_X(z-y)f_Y(y)$

2.2.2 Simple asymptotic properties

Without any real restriction, we can assume that the kernel function can be written in the simple form $w(y - y_i; h) = \frac{1}{h} w\left(\frac{y-y_i}{h}\right)$. The preliminary material showed that the mean and variance of a density estimator can then be expressed as

$$\begin{aligned} \mathbb{E}\left\{\hat{f}(y)\right\} &= f(y) + \frac{h^2}{2}\sigma_w^2 f''(y) + o(h^2), \\ \mathsf{var}\left\{\hat{f}(y)\right\} &= \frac{1}{nh} f(y)\,\alpha(w) + o\left(\frac{1}{nh}\right), \end{aligned}$$

where we assume that the kernel function is symmetric so that $\int uw(u)du = 0$, and where σ_w^2 denotes the variance of the kernel, namely $\int u^2 w(u)du$, and $\alpha(w) = \int w^2(u)du$.

These expressions capture the essential features of smoothing. In particular, bias is incurred and we can see that this is controlled by f'', which means that where the density has peaks and valleys the density estimate will underestimate and overestimate respectively. This makes intuitive sense.

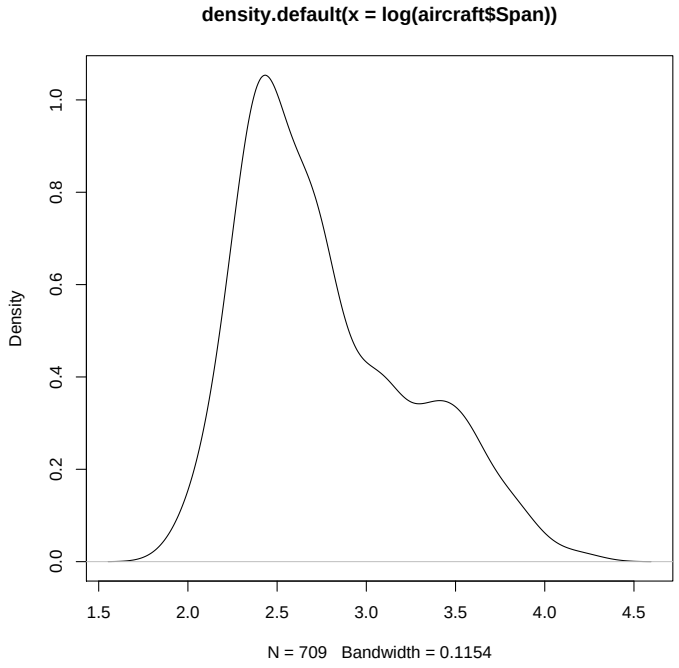
If we need it, a useful global measure of performance is the *mean integrated squared error* (MISE) which balances squared bias and variance.

$$\text{MISE}(\hat{f}) = \frac{1}{4} h^4 \sigma_w^4 \int f''(y)^2 dy + \frac{1}{nh}\,\alpha(w) + o\left(h^4 + \frac{1}{nh}\right).$$

2.2.3 Kernel density estimation in R

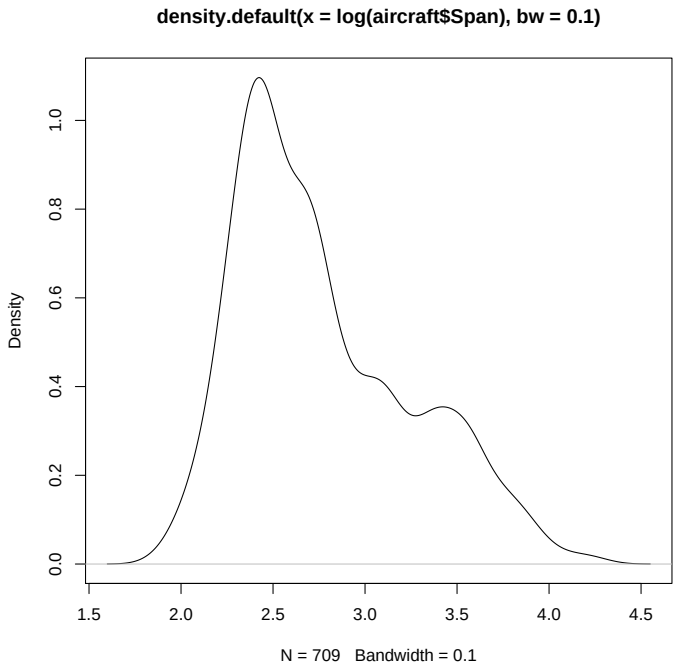
Kernel density estimation can be performed using the R function `density`

```
plot(density(log(aircraft$Span)))
```



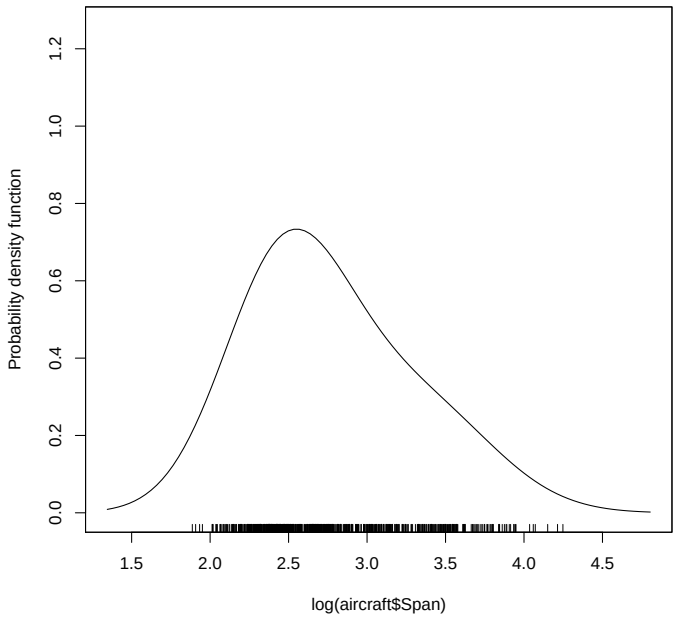
The bandwidth parameter is tuned automatically, but can can also be set manually using the option `bw`.

```
plot(density(log(aircraft$Span), bw=0.1))
```



Alternatively the function `sm.density` from the package `sm` can be used.

```
sm.density(log(aircraft$Span))
```



`sm` uses a different rule from `density` for choosing the bandwidth. When adding the argument `panel=TRUE` one can tune the bandwidth using a slider.

2.2.4 Extension to other sample spaces

The simple idea of density estimation is to place a kernel function, which in fact is itself a density function, on top of each observation and average these functions. This extends very naturally to a wide variety of other types of data and sample spaces.

For example, a two-dimensional density estimate can be constructed from bivariate data $\{(y_{1i}, y_{2i}) : i = 1, \ldots, n\}$ by employing a two-dimensional kernel function in the form

$$\hat{f}(y_1, y_2) = \frac{1}{n} \sum_{i=1}^{n} w(y_1 - y_{1i}; h_1)\, w(y_1 - y_{2i}; h_2).$$

Notice that there are now two smoothing parameters, (h_1, h_2). A more general two-dimensional kernel function could be used, but the simple product form is very convenient and usually very effective.

Figure 2.4 shows an example which uses the scores from the first two principal components of the aircraft data, again focussing on the third time period. The left hand scatterplot shows the individual scores while the right hand plot shows a density estimate, from which suggests three separate modes. This feature is not so easily seen from the raw scatterplot.

The lower two plots show alternative ways of presenting a two-dimensional estimate, using a coloured image on the left and contour lines on the right. Notice that the contours on the right have been chosen carefully to contain the quarters of the data with successively higher density, in a manner which has some similarities with a box plot.

This principle extends to all kinds of other data structures and sample spaces by suitable choice of an appropriate kernel function. However kernel density suffers especially badly from the curse of dimensionality and is unlikely to work well even in medium dimensions.

2.2.5 Deciding how much to smooth

It is not too hard to show that the value of h which minimizes MISE in an asymptotic sense is

$$h_{\text{opt}} = \left\{ \frac{\gamma(w)}{\beta(f) n} \right\}^{1/5},$$

where $\gamma(w) = \alpha(w)/\sigma_w^4$, and $\beta(f) = \int f''(y)^2 dy$. Of course, this is of rather limited use because it is a function of the unknown density. However, there are two practical approaches which can be taken to deciding on a suitable smoothing parameter to use. One is to construct an estimate of MISE and minimise this. Another is to estimate the optimal smoothing parameter. These two approaches are outlined below.

Cross-validation

The integrated squared error (ISE) of a density estimate is

$$\int \{\hat{f}(y) - f(y)\}^2 dy = \int \hat{f}(y)^2 dy - 2 \int f(y) \hat{f}(y) dy + \int f(y)^2 dy.$$

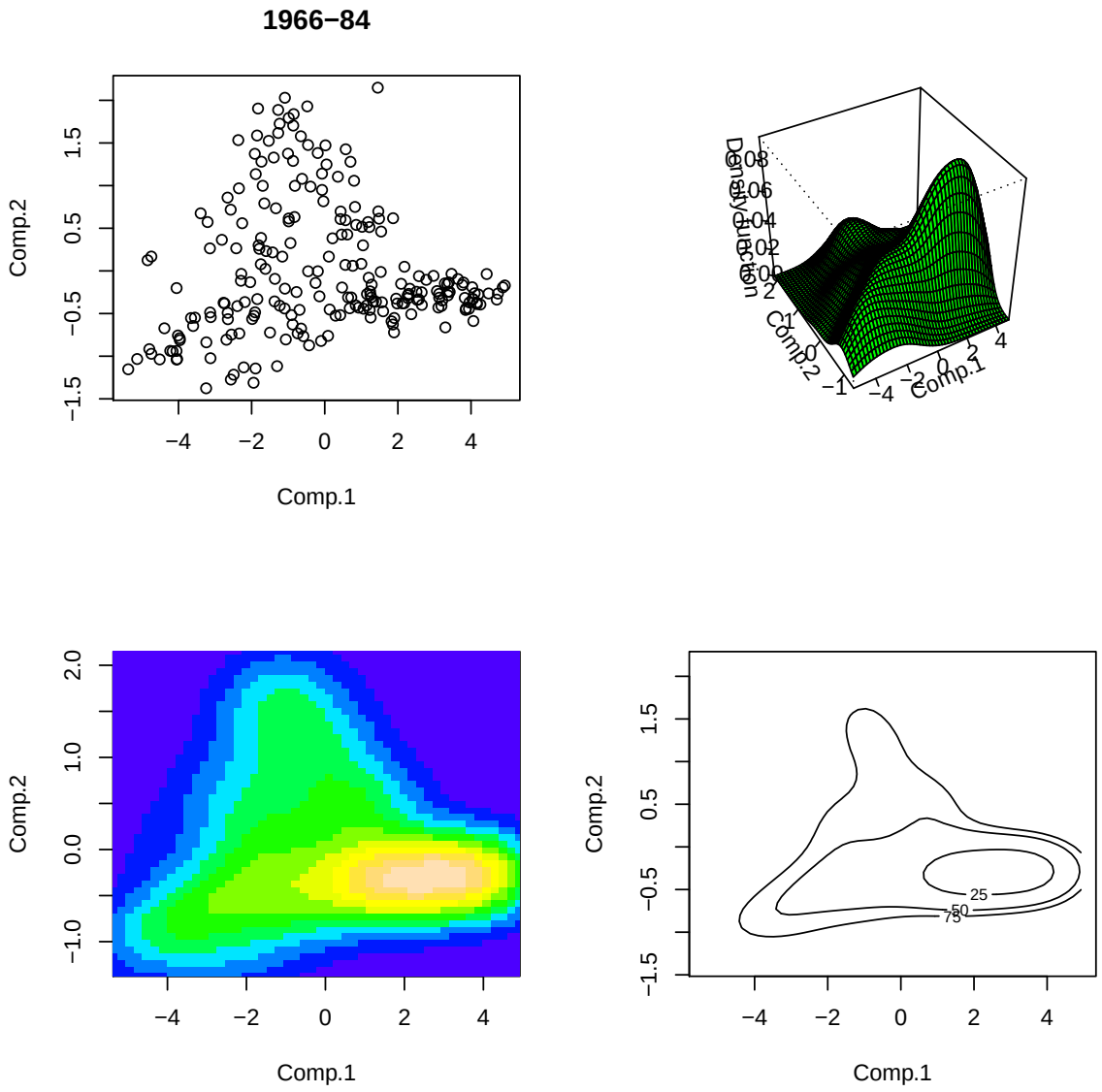


Figure 2.4. Bivariate kernel density estimate of the first two principal components of the aircraft data.

Only the first two of these terms involve h and these terms can be estimated by

$$\frac{1}{n}\sum_{i=1}^{n}\int \hat{f}^2_{-i}(y)dy - \frac{2}{n}\sum_{i=1}^{n}\hat{f}_{-i}(y_i),$$

where $\hat{f}_{-i}(y)$ denotes the estimator constructed from the data without the observation y_i. The value of h which minimises this expression is known as the *cross-validatory* smoothing parameter.

Plug-in methods

By inserting suitable estimates of the unknown quantities in the formula for the optimal smoothing parameter, a *plug-in* choice can be constructed. The difficult part is the estimation of $\beta(f)$ as this involves the second derivative of the density function. Sheather & Jones (JRSSB 53, 683–90) came up with a good, stable way of doing this. The Sheather-Jones method remains one of the most effective strategies for choosing the smoothing parameter.

A very simple plug-in approach is to use the normal density function in the expression for the optimal smoothing parameter. This yields the simple formula

$$h = \left(\frac{4}{3n}\right)^{1/5}\sigma,$$

where σ denotes the standard deviation of the distribution. This is a surprisingly effective means of smoothing data, in large part because it is very stable.

2.2.6 Some simple inferential tools

Once an estimate has been constructed, a natural next step is to find its standard error. The earlier result on the variance of $\hat{f}$ is a natural starting point, but this expression involves the unknown density. A helpful route is to consider a 'variance stabilising' transformation. For any transformation $t(\cdot)$, a Taylor series argument shows that

$$\mathsf{var}\left\{t(\hat{f}(y))\right\} \approx \mathsf{var}\left\{\hat{f}(y)\right\}\left[t'\left(\mathbb{E}\left\{\hat{f}(y)\right\}\right)\right]^2.$$

When $t(\cdot)$ is the square root transformation, the principal term of this expression becomes

$$\mathsf{var}\left\{\sqrt{\hat{f}(y)}\right\} \approx \frac{1}{4}\frac{1}{nh}\alpha(w),$$

which does not depend on the unknown density f. This forms the basis of a useful *variability band*. We cannot easily produce proper confidence intervals because of the bias present in the estimate. However, if the standard error is constructed and the intervals corresponding to two s.e.'s on the square root scale are transformed back to the origin scale, then a very useful indication of the variability of the density estimate can be produced. This is shown in figure 2.5 for the aircraft span data from period 3.

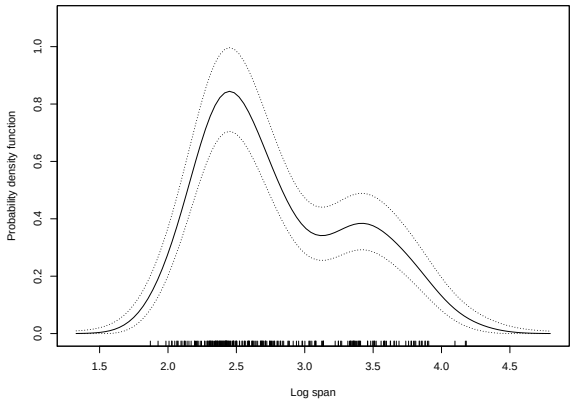


Figure 2.5. Reference bands for the density estimate of log-span.

A useful variation on this arises when the true density function is assumed to be normal with mean μ and variance σ^2, and the kernel function w is also normal. If the standard normal density function is denoted by ϕ, then the mean and variance of the density estimate at the point y are then

$$\begin{aligned} \mathbb{E}\left\{\hat{f}(y)\right\} &= \phi\left(y-\mu; \sqrt{h^2+\sigma^2}\right) \\ \mathsf{var}\left\{\hat{f}(y)\right\} &= \frac{1}{n}\phi\left(0; \sqrt{2}\,h\right)\ \phi\left(y-\mu; \sqrt{\sigma^2+\frac{1}{2}h^2}\right) \\ &\quad -\frac{1}{n}\phi\left(y-\mu; \sqrt{\sigma^2+h^2}\right)^2 \end{aligned}$$

These expressions allow the likely range of values of the density estimate to be calculated, under the assumption that the data are normally distributed. This can be expressed graphically through a *reference band*.

Example 2.2 (Icelandic tephra layer). Data on the percentages of aluminium oxide found in samples from a tephra layer resulting from a volcanic eruption in Iceland around 3500 years ago are available in the `tephra` dataset in the `sm` package. To deal with the percentage scale, apply the logit transformation

```
logit <- log(tephra$Al2O3/(100-tephra$Al2O3))
```

.

Can the variation in the tephra data be adequately modelled by a normal distribution on this scale? ◁

The density estimate shown in figure 2.6 does not fit comfortably within the reference band at all points and this effect persists across a wide range of smoothing parameters. A global test

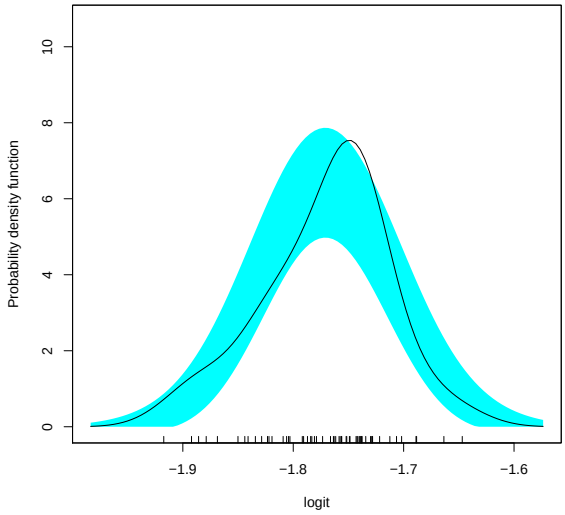


Figure 2.6. Kernel density estimate of the percentage of aluminion oxide, together with reference bands for an informal visual test of Gaussianity.

of normality could be developed but the graphical device of a reference band offers very useful informal insight.

The preliminary material also discussed the role of the bootstrap in capturing the variability, but not immediately the bias, of a density estimate.

2.3 Density estimation using mixtures

2.3.1 Introduction

Kernel density estimation is not the only way of estimating a density. When using a Gaussian kernel for density estimation we have constructed the estimate by adding up n Gaussian densities (where n is the sample size). When using mixtures we try to achieve the same using fewer Gaussian densities. However, in order for this to work, we need to estimate the mean and dispersion parameter of each Gaussian rather than considering Gaussian densities of the same dispersion anchored at each observation. Mixture-based methods for density estimation typically fare better than kernel-based methods for medium-dimensional data (none of the approaches work particularly well for high-dimensional data).

The key idea of mixture models is that whilst the density of interest might not necessarily be of a simple form, we can split the data into groups such that within each group the density has a simple form (e.g. Gaussian). This idea is illustrated by the example below.

Example 2.3 (Fisher's iris data). Consider the measurements of the petal width in the iris dataset. Figure 2.7 shows a plot of the (estimated) densities of the petal width for the three different

species and a plot of the density for the whole dataset. The plots suggest that it is reasonable to assume that the petal width in each group is from a Gaussian distribution. Denote with Y_i the petal width and with S_i the species name of the i-th flower. Then Y_i is (assumed to be) Gaussian in each group and the mean and the variance can be estimated by the empirical mean and variance in the different populations:

$(Y_i|S_i = se) \sim N(\mu_{se}, \sigma^2_{se})$ for *iris setosa* (estimates: $\hat{\mu}_{se} = 0.246$, $\hat{\sigma}^2_{se} = 0.01133$)
$(Y_i|S_i = ve) \sim N(\mu_{ve}, \sigma^2_{ve})$ for *iris versicolor* (estimates: $\hat{\mu}_{ve} = 1.326$, $\hat{\sigma}^2_{ve} = 0.03990$)
$(Y_i|S_i = vi) \sim N(\mu_{vi}, \sigma^2_{vi})$ for *iris virginica* (estimates: $\hat{\mu}_{vi} = 2.026$, $\hat{\sigma}^2_{vi} = 0.07697$)

The density of Y (species confounded) is then

$$f(y) = \pi_{se} f_{\mu_{se},\sigma^2_{se}}(y) + \pi_{ve} f_{\mu_{ve},\sigma^2_{ve}}(y) + \pi_{vi} f_{\mu_{vi},\sigma^2_{vi}}(y)$$

The density of Y is thus a *mixture* of three normal distributions. The π_{se}, π_{ve} and π_{vi} denote the probability that a randomly picked iris is of the corresponding species. These probabilities are sometimes referred to as *mixing weights*.

In the typical context of density estimation we would not have the grouping information provided by the species labels, i.e. we now assume we only know the petal widths. Nonetheless, we can stick to our idea that the density of Y is a mixture of Gaussians. Thus we assume

$$f(y) = \pi_1 f_{\mu_1,\sigma^2_1}(y) + \pi_2 f_{\mu_2,\sigma^2_2}(y) + \pi_3 f_{\mu_3,\sigma^2_3}(y).$$

Note that we had to replace the species names with indexes, because we don't possess the species information any more. This makes the estimation of the means μ_1, μ_2, μ_3 and variances $\sigma^2_1, \sigma^2_2, \sigma^2_3$ far more difficult. The species name S_i is now a *latent* variable, as we are unable to observe it. Once estimated, the S_i however provide potentially interesting information: they show how the data can be clustered into three groups. ◁

2.3.2 Model

We will for now assume that the component densities are Gaussian distributions, as this is by far the most common model. However any other distribution could be used. If we want the mixture to be able to approximate an arbitrary distribution we would have to choose a component density that can, in the limit, be reduced to a point mass (just like a Gaussian is reduced to a point mass if $\sigma^2 \to 0$).

We assume that the data comes from K groups ("clusters"). For the moment K is assumed to be fixed and known. We assume that the data in cluster k is from a (multivariate) Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Thus the density of $\mathbf{y}$ in cluster k is

$$f_{\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k}(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^p \cdot |\boldsymbol{\Sigma}_k|}} e^{-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\mathbf{y}-\boldsymbol{\mu}_k)}.$$

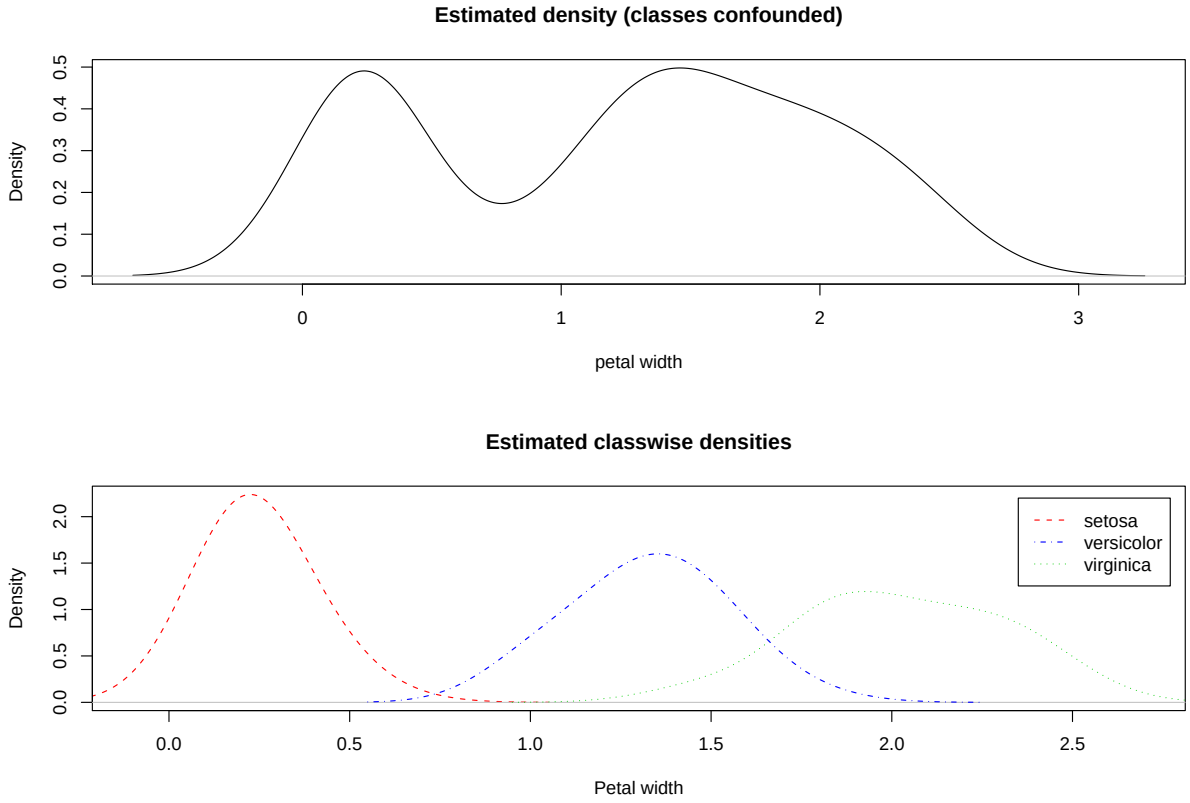


Figure 2.7. Estimated density of the petal width for all species confounded (above) and by species (below)

Often one assumes that all clusters have the same orientation and spread, i.e. $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$, or that all clusters are spherical, i.e. $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$. Another popular assumption is that the covariance matrices are diagonal, i.e. $\boldsymbol{\Sigma}_k = \begin{bmatrix} {\sigma_1^{(k)}}^2 & \cdots 0 & \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {\sigma_p^{(k)}}^2 \end{bmatrix}$. In this case the clusters are ellipses with axes parallel to to coordinate axes. Figure 2.8 visualises these different assumptions.

The density of $\mathbf{y}$ (all clusters confounded) is the *mixture* of the distributions in the clusters:

$$f(\mathbf{y}) = \sum_{k=1}^{K} \pi_k f_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}(\mathbf{y}).$$

with $\pi_k \geq 0$ such that $\sum_{k=1}^{K} \pi_k = 1$. This yields the likelihood

$$\ell(\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K) = \prod_{i=1}^{n} \left(\sum_{k=1}^{K} \pi_k f_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}(\mathbf{y}_i) \right). \tag{2.3}$$

2.3.3 Inference

Estimating the parameters of the mixture model appears difficult at first sight. Each contribution to the likelihood (2.3) is a sum rather than a product, which is usually the case. Often the loglikelihood is a simpler expression than the likelihood itself (think of a Gaussian), however

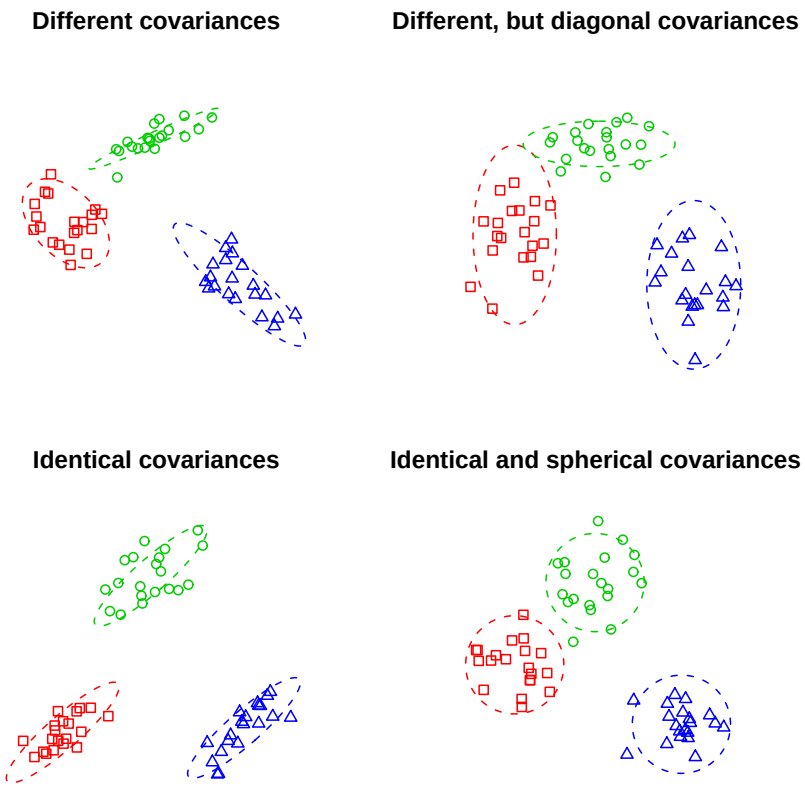


Figure 2.8. Examples for the different assumptions on the covariance structure

for mixtures this is not the case: each contribution to the loglikelihood is the logarithm of a sum, which cannot be simplified any further.

The problem can be overcome by considering an EM algorithm, which introduces auxiliary variables S_i which indicate the cluster allocation. The EM algorithm now consists of the following iterative process.

For $h = 1, 2, \ldots$ iterate the E-Step and the M-Step:

E-Step Estimate the distribution of S_i given the data and the previous values of the parameters $\hat{\pi}_k^{(h-1)}$, $\hat{\boldsymbol{\mu}}_k^{(h-1)}$ and $\hat{\boldsymbol{\Sigma}}_k^{(h-1)}$ $(k = 1, \ldots, K)$. This yields the "responsibilities" $w_{ik}^{(h)}$:

$$w_{ik}^{(h)} := \frac{\pi_k^{(h-1)} f_{\hat{\boldsymbol{\mu}}_k^{(h-1)}, \hat{\boldsymbol{\Sigma}}_k^{(h-1)}}(\mathbf{y}_i)}{\sum_{\zeta=1}^{K} \pi_\zeta^{(h-1)} f_{\hat{\boldsymbol{\mu}}_\zeta^{(h-1)}, \hat{\boldsymbol{\Sigma}}_\zeta^{(h-1)}}(\mathbf{y}_i)}$$

M-Step Compute the updated $\hat{\pi}_k^{(h)}$, $\hat{\boldsymbol{\mu}}_k^{(h)}$, and $\hat{\boldsymbol{\Sigma}}_k^{(h)}$ $(k = 1, \ldots, K)$ by estimating them from the data using the vector $\mathbf{w}_k^{(h)} = (w_{1k}^{(h)}, \ldots, w_{nk}^{(h)})$ of responsibilities as weights: This leads to the updated estimators

$$\hat{\pi}_k^{(h)} = \frac{1}{n} \sum_{i=1}^{n} w_{ik}^{(h)},$$

$$\hat{\boldsymbol{\mu}}_k^{(h)} := \frac{1}{\sum_{i=1}^{n} w_{ik}^{(h)}} \sum_{i=1}^{n} w_{ik}^{(h)} \mathbf{y}_i,$$

and

$$\hat{\boldsymbol{\Sigma}}_k^{(h)} := \frac{1}{\sum_{i=1}^{n} w_{ik}^{(h)}} \sum_{i=1}^{n} w_{ik}^{(h)} (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_k^{(h)})(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_k^{(h)})^\top$$

for the covariances $\boldsymbol{\Sigma}_k$ (assuming different covariances in the classes).

Figure 2.9 visualises the flow of the algorithm for an easy toy example.

Derivation of the EM algorithm for mixtures

The EM algorithm is based on the likelihood that is obtained when assuming the class labels S_i are known. In this case the likelihood is

$$\prod_{i=1}^{n} \pi_{S_i} f_{\boldsymbol{\mu}_{S_i}, \boldsymbol{\Sigma}_{S_i}}(\mathbf{y})$$

Note that the sum has now disappeared. This is due to the fact that we know the cluster labels now, so we just multiply the densities of the distribution in the corresponding clusters instead of multiplying the unpleasant mixture density. For simplification of the notation denote with $\boldsymbol{\theta}$ the whole parameter vector, i.e.

$$\boldsymbol{\theta} := (\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K)$$

The log-likelihood is thus

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\log \pi_{S_i} + \log f_{\boldsymbol{\mu}_{S_i}, \boldsymbol{\Sigma}_{S_i}}(\mathbf{y})) \tag{2.4}$$

In reality however, the S_i are unknown. So the EM-algorithm uses the trick of taking the (conditional) expectation of the likelihood.

In the E-step we have to compute the conditional expectation of the log-likelihood (2.4) given the data using the previous values of the parameters $\hat{\boldsymbol{\theta}}^{(h-1)}$. Thus we have to compute the conditional distribution of S_i (using the Bayes formula) yielding the responsibilities

$$w_{ik}^{(h)} := \mathbb{P}_{\hat{\boldsymbol{\theta}}^{(h-1)}}(S_i = k) = \frac{\hat{\pi}_k^{(h-1)} f_{\hat{\boldsymbol{\mu}}_k^{(h-1)}, \hat{\boldsymbol{\Sigma}}_k^{(h-1)}}(\mathbf{y}_i)}{\sum_{\zeta=1}^{K} \hat{\pi}_\zeta^{(h-1)} f_{\hat{\boldsymbol{\mu}}_\zeta^{(h-1)}, \hat{\boldsymbol{\Sigma}}_\zeta^{(h-1)}}(\mathbf{y}_i)}$$

The conditional expectation is thus

$$\mathbb{E}_{\hat{\boldsymbol{\theta}}^{(h-1)}}\left(L(\boldsymbol{\theta}) | \mathbf{y}_1, \ldots, \mathbf{y}_n\right) = \sum_{i=1}^{n} \sum_{k=1}^{K} (w_{ik}^{(h)} \log \pi_k + w_{ik}^{(h)} \log f_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}(\mathbf{y})). \tag{2.5}$$

In the M-step we have to find $\hat{\boldsymbol{\theta}}^{(h)}$ by maximising the conditional expectation (2.5) ober $\boldsymbol{\theta}$. One can easily see that the updated parameter estimates $\hat{\pi}_k^{(h)}$, $\hat{\boldsymbol{\mu}}_k^{(h)}$, and $\boldsymbol{\Sigma}_k^{(h)}$ maximise (2.5).

2.3.4 Estimating the number of components

Estimating the number of components is, at least from a theoretical point of view, difficult. We cannot use classical statistical tests for this. To illustrate this consider a mixture model with two univariate Gaussian components

$$f(y) = \pi f_{\mu_1, \sigma_1^2}(y) + (1 - \pi) f_{\mu_2, \sigma_2^2}(y).$$

If we want to test whether we need one or two components, we could test the null hypothesis that $\pi = 1$. The first obvious issue is that $\pi = 1$ is on the boundary of the parameter space for π, which is $[0, 1]$, so standard approaches for likelihood ratio tests would fail. The even bigger problem however is that if $\pi = 1$, the parameters μ_2 and σ_2^2 do not influence the likleihood any more. In other words, the effective dimension of the problem collapses on the boundary, which makes deriving theoretical results for likelihood ratio tests very challenging.

For this reason, it is common practice to use model selection criteria such as BIC to select the number of components.

However, if our only intention is to estimate the density rather than interpret the clustering identified, getting the number of components exactly right is less of an issue.

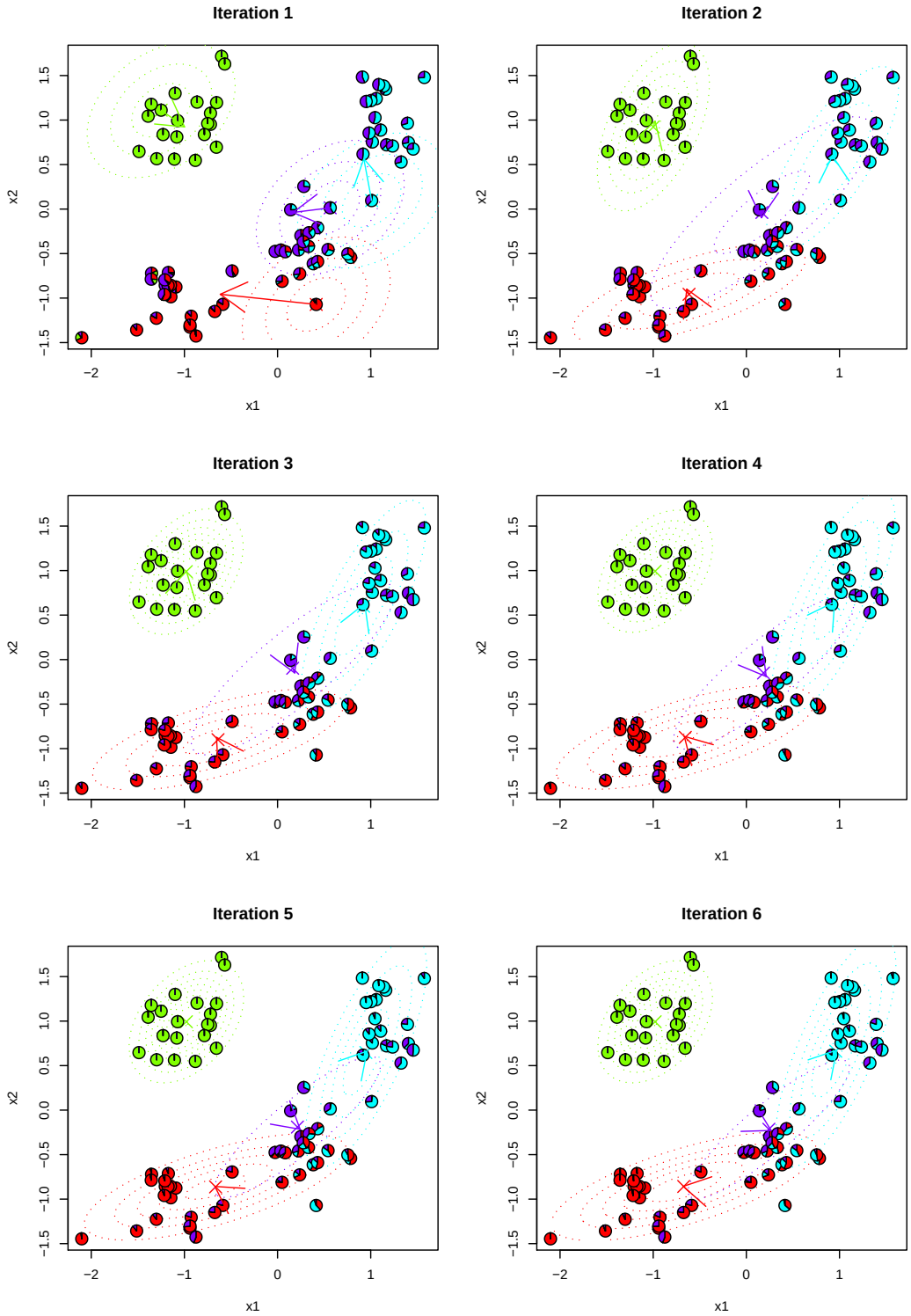


Figure 2.9. First 6 iterations of the EM algorithm for fitting a Gaussian mixture model to a toy example. The colouring of the data points corresponds to the responsibilities, the dotted lines show the isodensites of the Gaussians, and the arrow indicates where the means of the clusters are moved.

2.3.5 Example

Example 2.4 (Aircraft data (ctd.)). Figure 2.10 shows the results obtained when using mixtures of Gaussians to estimate the density of the first two principal components of the aircrcraft data for the third period. Due to the nature of the mixture model, the peaks are much more pronounced and most importantly there is a very concentrated fourth peak that was not apparent in the results obtained using kernel density estimation. Note that when using a mixture model there would have been no need to compute principal components before estimating the density. We could have used the raw data of six measurements per aircraft. The principal components were used to allow better comparison to the results obtained using a kernel density estimate, which would not be able to cope well with data of that dimension. ◁

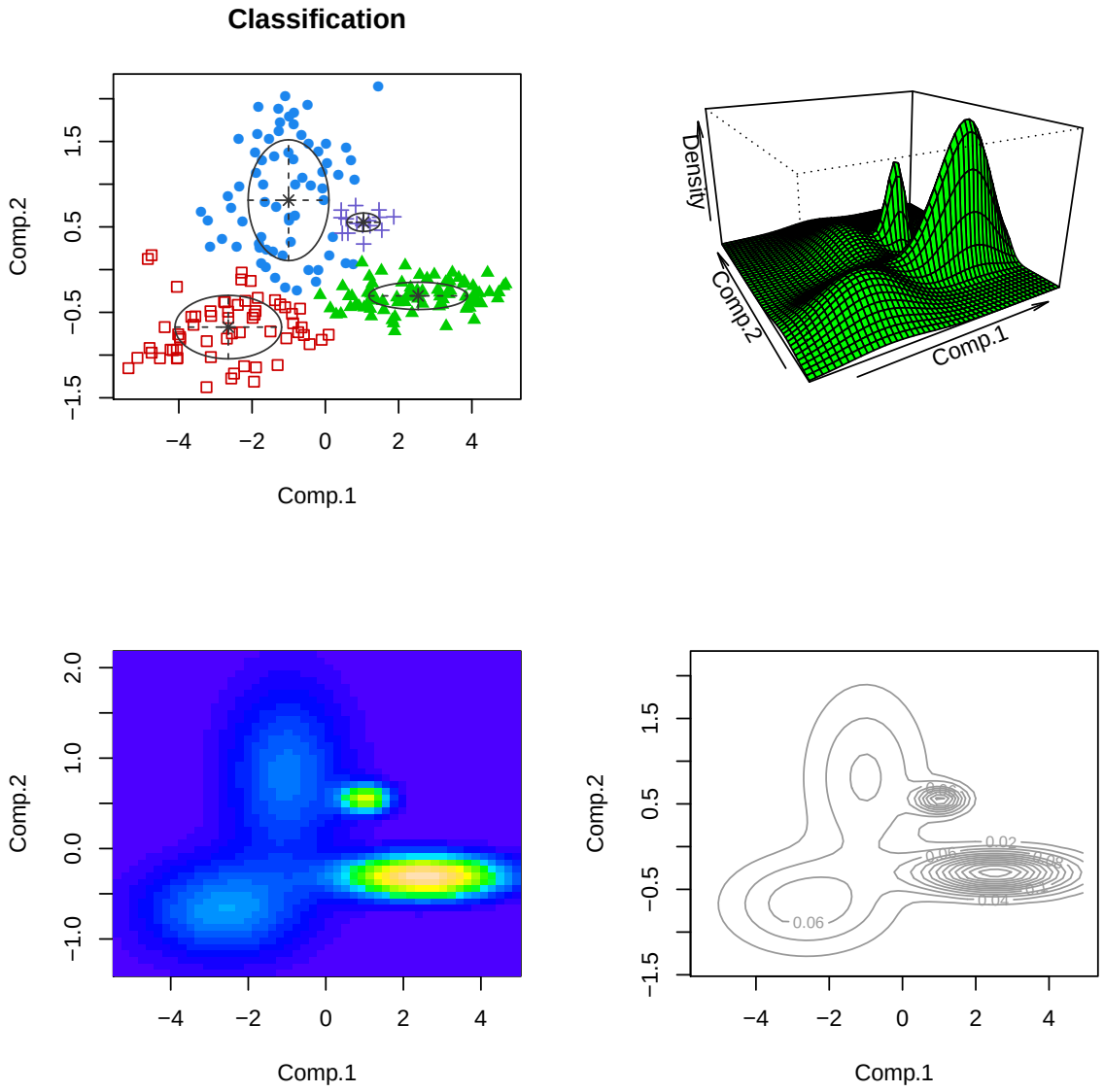
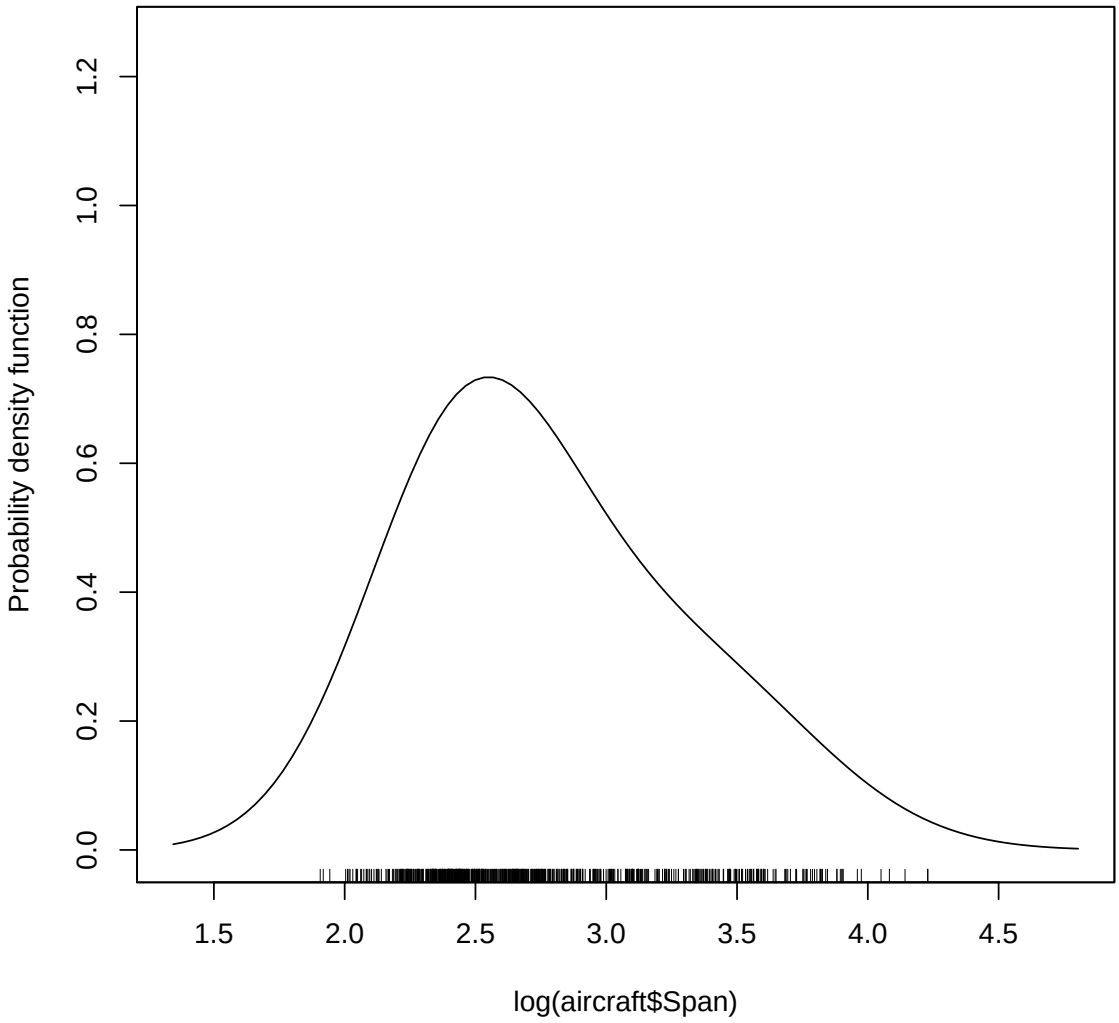


Figure 2.10. Estimated density using mixtures of normals for the first two principal components of aircraft data (using the R package `mclust`)

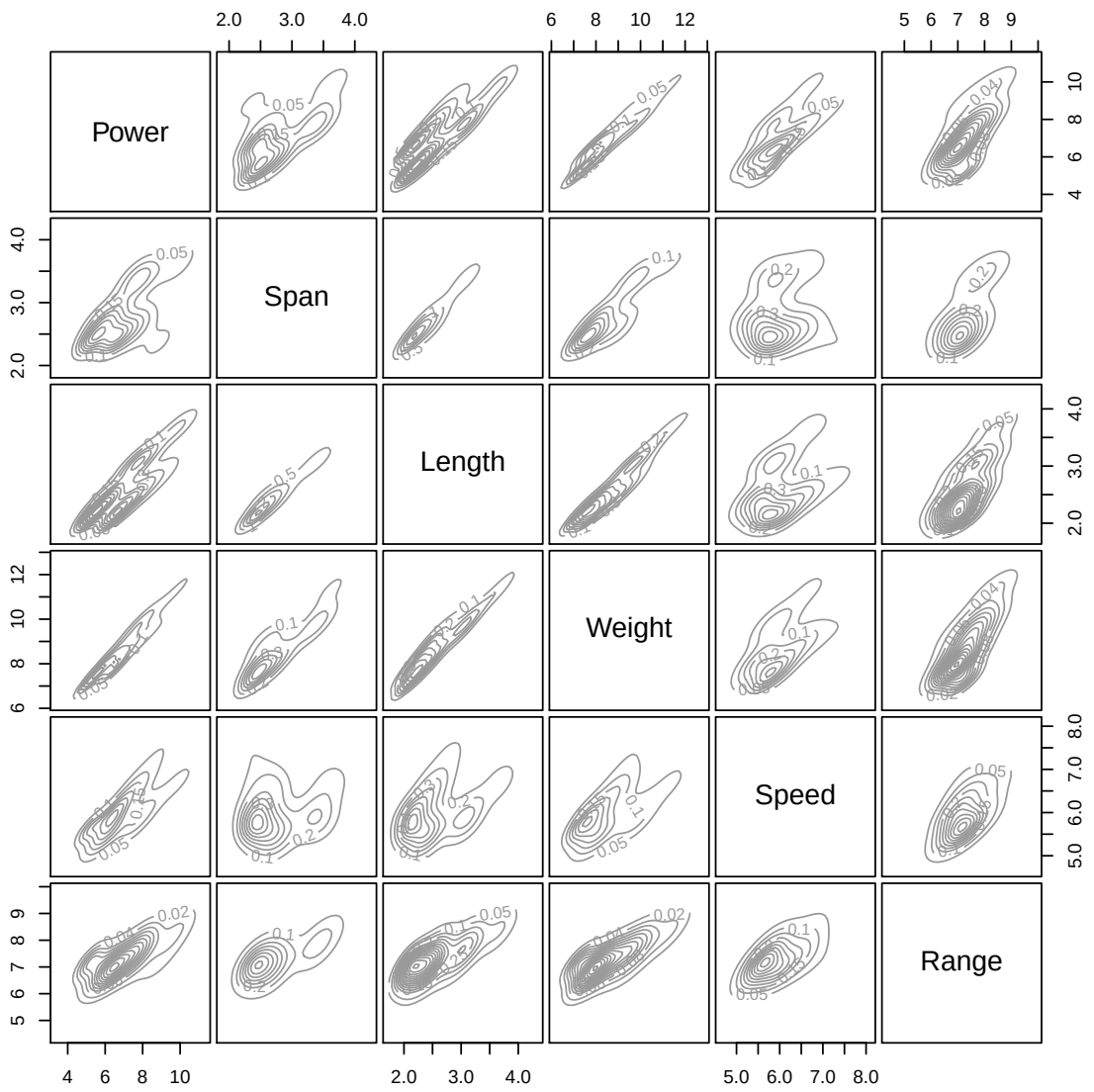
2.3.6 Mixtures of normals in R

Mixtures of normals can be fitted in R using the package `mclust`. `mclust` automatically chooses an appropriate number of components and a suitable model for the variance structure.

```
logaircraft <- log(aircraft[,-(1:2)])
m <- Mclust(logaircraft)
plot(m, what="classification")
```




```
plot(m, what="density")
```


Power
Span
Length
Weight
Speed
Range

3

Splines

3.1 Introduction

This chapter covers splines, which are one of the most popular tools for flexible modelling. This section discusses a number of more philosophical concepts, some of which we have already touched upon.

In parametric modelling (*e.g.* estimating the rate of a Poisson distribution, linear regression) we assume we know the data generating process up to a finite number of parameters. In flexible modelling we want to fit a function to data, without making such a strict parametric assumption. All we are willing to assume is typically that the function of interest is sufficiently smooth. More formally speaking, this corresponds to working with an infinite-dimensional parameter space. This flexible approach has a number of advantages, most importantly it is less likely that the model is mis-specified. However there is a price to pay. Estimation becomes more difficult.

Example 3.1. Figure 3.1 shows two smooth functions describing the relationship between the response Y_i and the covariate x_i. In this example both functions yield the same fitted values $\hat{y}_i = \hat{m}(x_i)$. This also implies that the least-squares loss $\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2$ is the same for both functions, i.e. the data alone does not tell us which function does a better job. There is no global answer to this question.

Which of the two functions appears better suited to us depends on the context and also to some extent our subjective choice. In most circumstances we would prefer the function in the left-hand panel as it is the "simpler" function. However, if we expect the signal to have a periodic component (say we are expecting a day-of-the-week effect) then we might prefer the function shown in the right-hand panel. ◁

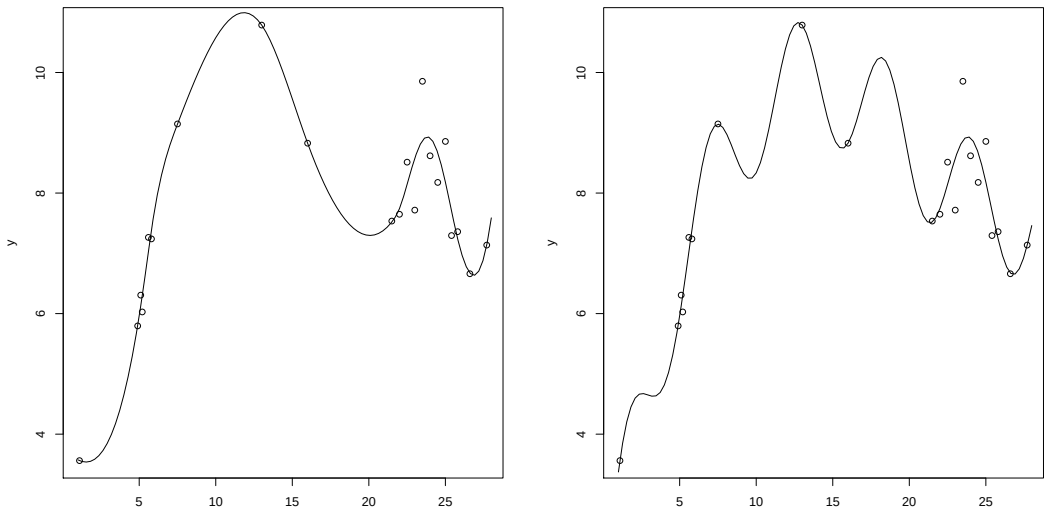


Figure 3.1. Two possible smooth functions modelling the relationship between the response Y_i and the covariate x_i. Note that both functions yield the same fitted values $\hat{y}_i = \hat{m}(x_i)$ and thus the same least-squares loss $\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2$.

What we have seen in the example is simply that the family of smooth functions is so large that observing a finite sample alone will not tell us enough to learn the function of interest $m(\cdot)$.

We need to provide additional information, which can be of different types:

– We can assume that the function of interest $m(\cdot)$ comes from a more restricted family of functions. We might even assume a rich class of parametric models. We will use this idea when we are looking at splines based on truncated power series and B-splines in section 3.2.4.
– We express a preference for some functions over others (without looking at the data) and use this in the model fitting procedure. Typically we prefer a smooth function to a more wiggly function. In a frequentist setting, this leads to penalty-based approach, or can be viewed as a Bayesian prior over the space of functions. We will discuss this in sections 3.2.3 and 3.3.

3.2 Univariate splines

3.2.1 Polynomial regression

We will start by revising polynomial regression. To fix notation, we quickly state the simple linear regression model

$$\mathbb{E}(Y_i) = \beta_0 + \beta_1 x_i \qquad \text{for } i = 1, \ldots, n,$$

or equivalently, in matrix-vector notation,

$$\mathbb{E}(\mathbf{y}) = \mathbf{B}\boldsymbol{\beta} \qquad \text{with } \mathbf{y} = (Y_1, \ldots, Y_n)^\top \text{ and } \mathbf{B} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

The simple linear regression model can be extended into a polynomial regression model by including powers of the covariates x_i in the design matrix. The polynomial regression model

$$\mathbb{E}(Y_i) = \beta_0 + \beta_1 x_i + \ldots + \beta_r x_i^r \qquad \text{for } i = 1, \ldots, n,$$

just corresponds to linear regression using the expanded design matrix

$$\mathbf{B} = \begin{pmatrix} 1 & x_1 & \ldots & x_1^r \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^r \end{pmatrix}.$$

We can then estimate $\boldsymbol{\beta}$ using the same techniques as used in multiple linear regression, i.e. the least-squares estimator is

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{y}$$

Polynomial regression is a very simple example of a basis expansion technique. We have simply replaced the design matrix of simple linear regression by an augmented design matrix. In the case of polynomial regression we have simply added powers of the x_i's.

Many of the techniques covered in this chapter will be based in this idea of basis expansions.

Polynomial regression can be a useful tool if a polynomial of very low order yields a sufficient fit to the data.

Example 3.2 (Glucose levels in potatoes). Figure 3.2 shows a quadratic regression model fitted to a simple data set from an experiment in which the glucose level in potatoes was measured over the course of several weeks. Given the small number of observations there is little need to go beyond a simple quadratic regression model. ◁

However, polynomial regression is not very well suited for modelling more complex relationships, as the following example shows.

Example 3.3. Consider the data set simulated using the model

$$y_i = 1 - x_i^3 - 2\exp(-100x_i^2) + \varepsilon_i$$

with $\mathbf{x} = (-1, -0.98, \ldots, 0.98, 1)$ and $\varepsilon_i \sim \mathsf{N}(0, 0.1^2)$. Figure 3.3(a) shows the data together with the fitted function obtained for a polynomial regression model of degree 10. The polynomial model of degree 10 is not flexible enough to capture the sharp dip around 0. If we

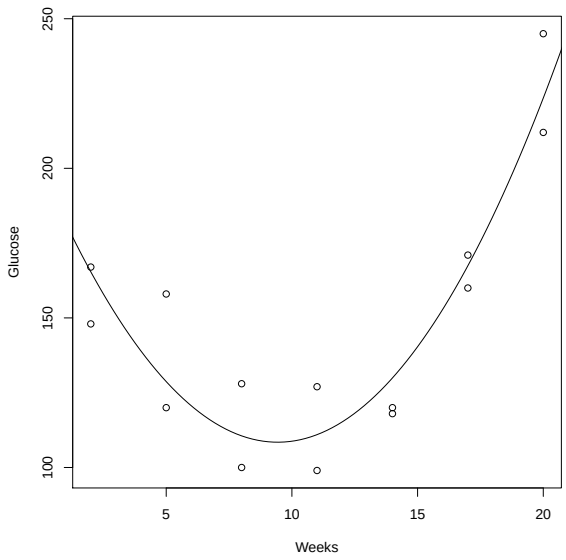


Figure 3.2. Glucose level in potatoes. The solid line is the fitted regression function obtained from quadratic regression.

increase the degree to 17 we can capture the dip better (panel (b)). However, the polynomial fit of degree 17 shows strong oscillations which are not supported by the data. Panel (c) shows the fitted regression function using a spline based model, which we will discuss later on in this chapter. The spline-based approach can capture the sharp dip much better and without yielding any oscillations.

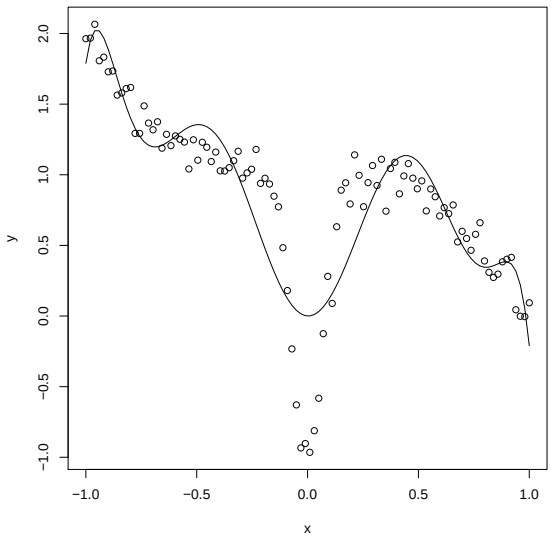
Figure 3.4 allows some insight into why the polynomial model struggles. It shows `image` plots of the hat matrix $\mathbf{S} = \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top$ for the three models under consideration. The hat matrix maps the observed response to the fitted response, i.e.

$$\hat{\mathbf{y}} = \mathbf{B}\hat{\boldsymbol{\beta}} = \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{y} = \mathbf{S}\mathbf{y}$$

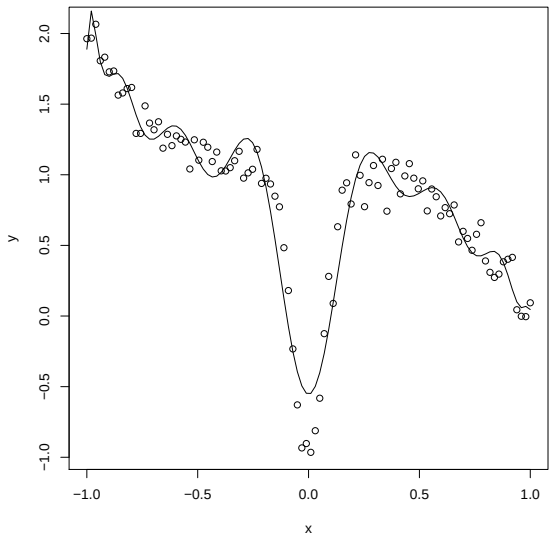
When performing flexible regression we would expect the prediction at x_i to almost only depend on observations close to x_i, i.e. we would expect the hat matrix $\mathbf{S}$ to be largely band-diagonal with a rather narrow band width. However, polynomials are not "local". As one can see from a Taylor series expansion, the coefficients of the polynomial can be learnt from higher order derivatives observed at a single point. The problem is that sharp dip provides more information than the data on either side of it, yielding to a poor fit on both sides. This is known as Runge's phenomenon in Numerical Analysis.

Figure 3.3(a) and figure 3.3(b) shows another drawback of polynomial regression. As $x \to \pm\infty$ the polynomial must go to $\pm\infty$ as well. This often leads to very high curvature at both ends of the range, which is typically not supported by the data.

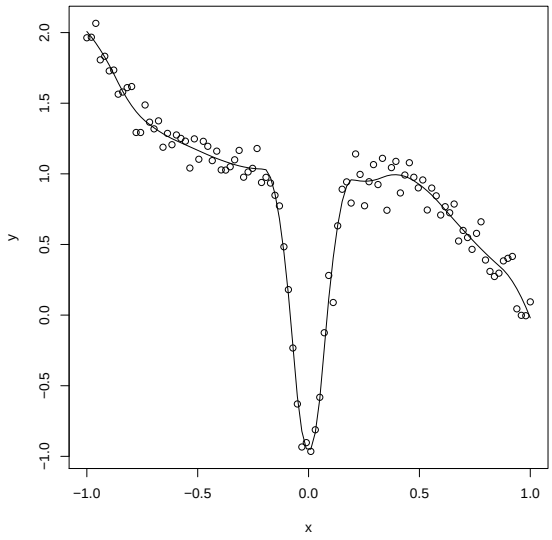
Yet another reason for avoiding polynomial regression is that it is highly likely to be numerically unstable. Due to the large correlations between the powers of the x_i, which make up the columns of the design matrix, the design matrix $\mathbf{B}$ and the matrix of cross-products



(a) Polynomial regression of degree 10

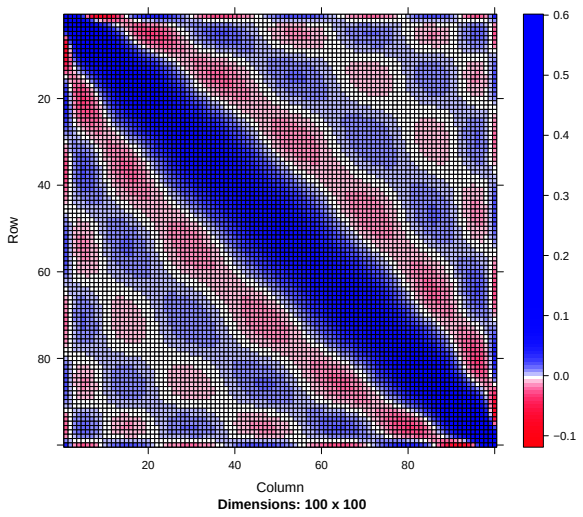


(b) Polynomial regression of degree 17

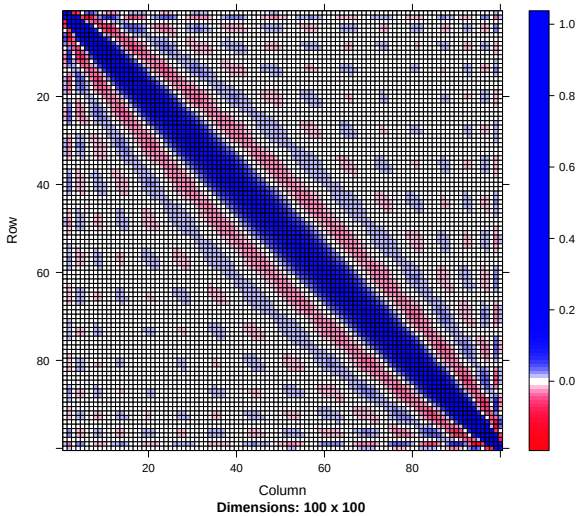


(c) Quadratic-spline-based regression

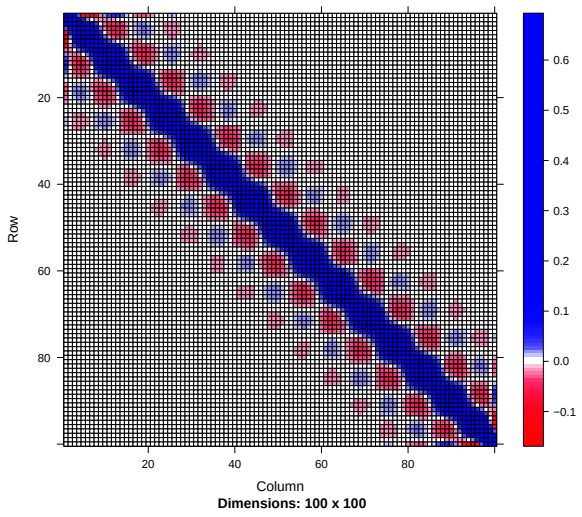
Figure 3.3. Data and fitted function for the simulated data from example 3.3 for polynomial regression of degrees 10 and 17 as well as for a spline-based model.



(a) Polynomial regression of degree 10



(b) Polynomial regression of degree 17



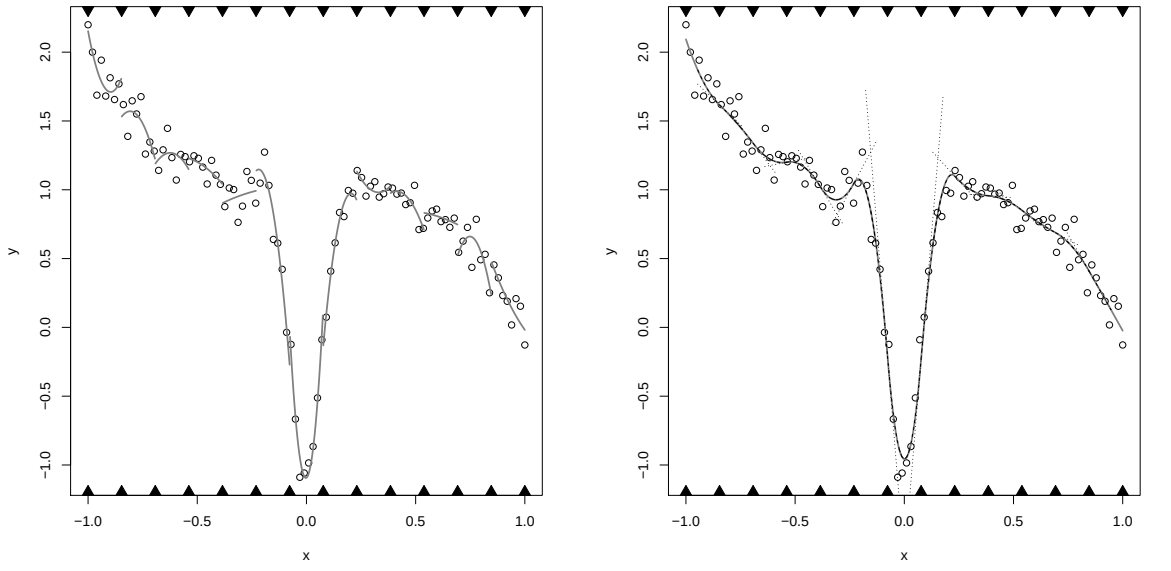
(c) Quadratic-spline-based regression

Figure 3.4. Hat matrix $\mathbf{S} = \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top$ for polynomial regression of degrees 10 and 17 as well as for splines applied to the simulated data from example 3.3.

$\mathbf{B}^\top\mathbf{B}$ is very likely to be ill-conditioned. The condition number[1] of $\mathbf{B}^\top\mathbf{B}$ for the polynomial regression model of degree 17 is 1.56×10^{12}, i.e. $\mathbf{B}^\top\mathbf{B}$ is barely invertible. For comparison, the corresponding condition number for the spline-based model is 32.49.

Instead of using monomials it would be numerically more stale to use so-called Tchebychev polynomials (as produced for example by the R function `poly`). Both sets of basis functions are equivalent, i.e. they span the same linear subspace and thus yield identical predictions. Though numerically more stable, Tchebychev polynomials suffer from all the other problems just as much as monomials. ◁

As we have seen in the example above, polynomial regression is, unless modelling very simple relationships, not a suitable tool for flexible regression. In the next section we will consider piecewise polynomial models, which are better suited for flexible regression. These are based on the idea of splitting the input domain and fitting low-order polynomials in each interval. As we can see from figure 3.5(a) fitting polynomials independently of each other does not yield satisfactory results. We will thus introduce additional constraints which make the function continuous and (potentially) differentiable (cf. panel (b)).



(a) Discontinuous piecewise polynomials

(b) Piecewise polynomials which form a continuously differentiable function (derivatives at knots shown as dashed lines)

Figure 3.5. Piece-wise polynomials fitted to the data from example 3.3 with an without smoothness constraints. The back triangles show the positions of the knots.

[1] The condition number of a matrix is defined as the ratio of the largest singular value divided by the smallest singular value. For a symmetric positive-definite matrix this is the same as the ratio of the largest over the smallest eigenvalue. The condition number is of measure of how numerically unstable matrix operations like taking the inverse will be.

3.2.2 Polynomial splines

In this section we will introduce polynomial splines which are piecewise polynomials, which "glued together" at the knots so that the resulting function is r-times continuously differentiable.

Definition 3.1 (Polynomial spline). *Given a set of knots $a = \kappa_1 < \kappa_2 < \ldots < \kappa_l = b$, a function $m : [a, b] \to \mathbb{R}$ is called a (polynomial) spline of degree r if*

– *$m(\cdot)$ is a polynomial of degree r on each interval (κ_j, κ_{j+1}) ($j = 1, \ldots, l-1$).*
– *$m(\cdot)$ is $r-1$ times continuously differentiable.*[2]

Historically, a spline was an elastic ruler used to draw technical designs, notably in shipbuilding and the early days of aircraft engineering. Figure 3.6 shows such a spline.[3]

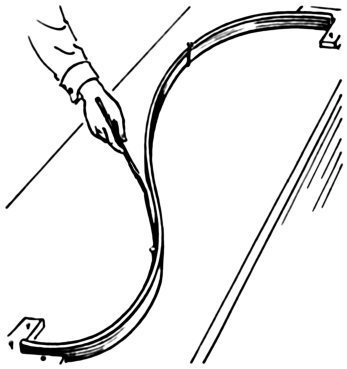


Figure 3.6. A spline.

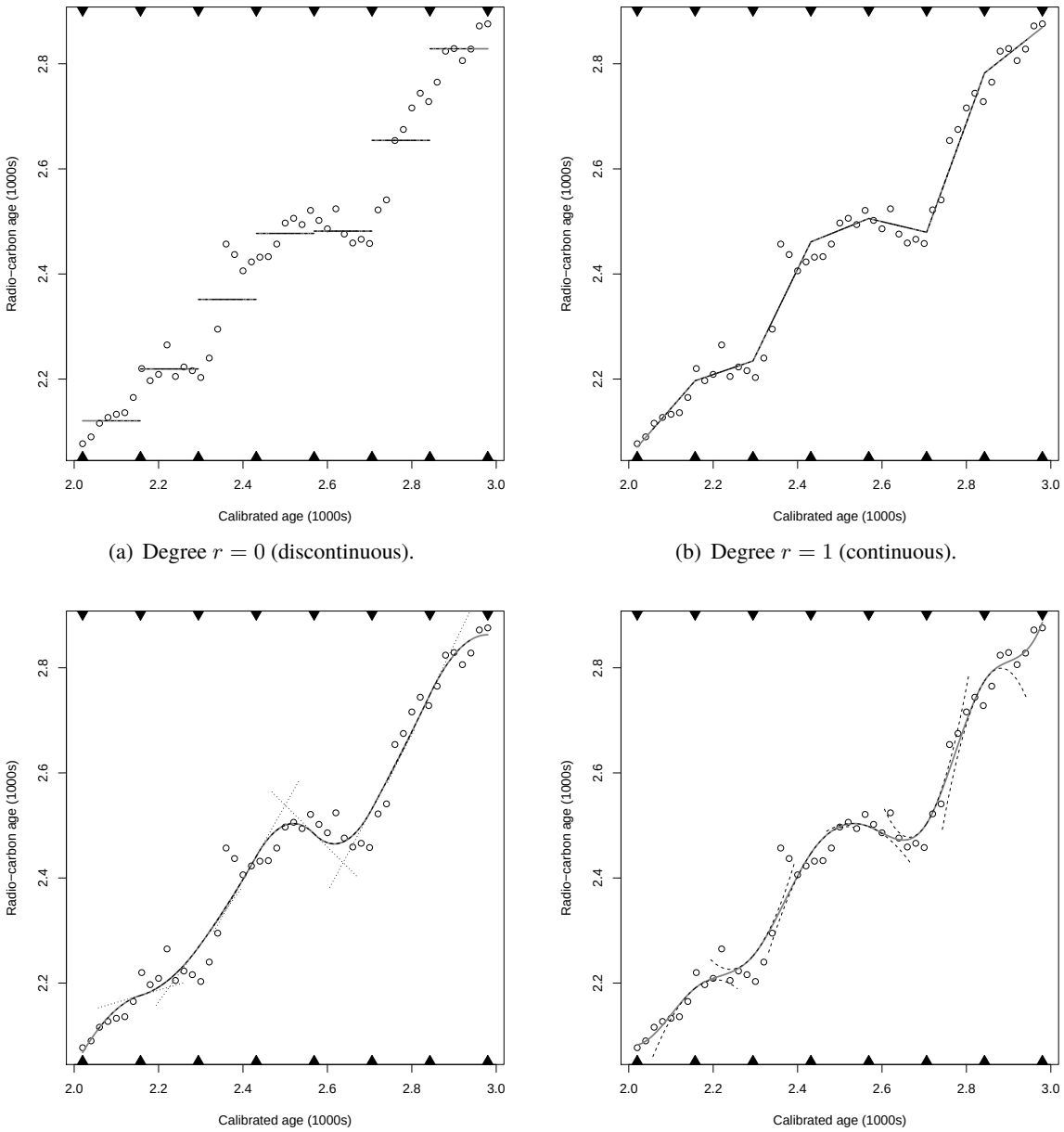
Choice of degree r. The degree r of the spline controls the smoothness in the sense of controlling its differentiability. For $r = 0$ the spline is a discontinuous step function. For $r = 1$ the spline is a polygonal line. For larger values of r the spline is increasingly smooth, but also behaves more and more like one global polynomial. It is worth noting that assuming too smooth a function can have significant detrimental effects on the fitted regression function (*e.g.* oscillations, ballooning). In practice it is rarely necessary to go beyond $r = 3$.

Example 3.4 (Radiocarbon dating). In a scientific experiment high-precision measurements of radiocarbon were performed on Irish oak. To construct a calibration curve we need to learn the

[2] For a spline of degree 0 the function $m(\cdot)$ does not need to be continuous. For a spline of degree 1 the function $m(\cdot)$ needs to be continuous, but does not need to be differentiable.

[3] See `http://pages.cs.wisc.edu/~deboor/draftspline.html` for a picture (probably from the 1960's) of a Boeing engineer using a spline.

relationship between the radiocarbon age and the calendar age. Figure 3.7 shows spline fits to the data using splines of different degrees. ◁



(a) Degree $r = 0$ (discontinuous).

(b) Degree $r = 1$ (continuous).

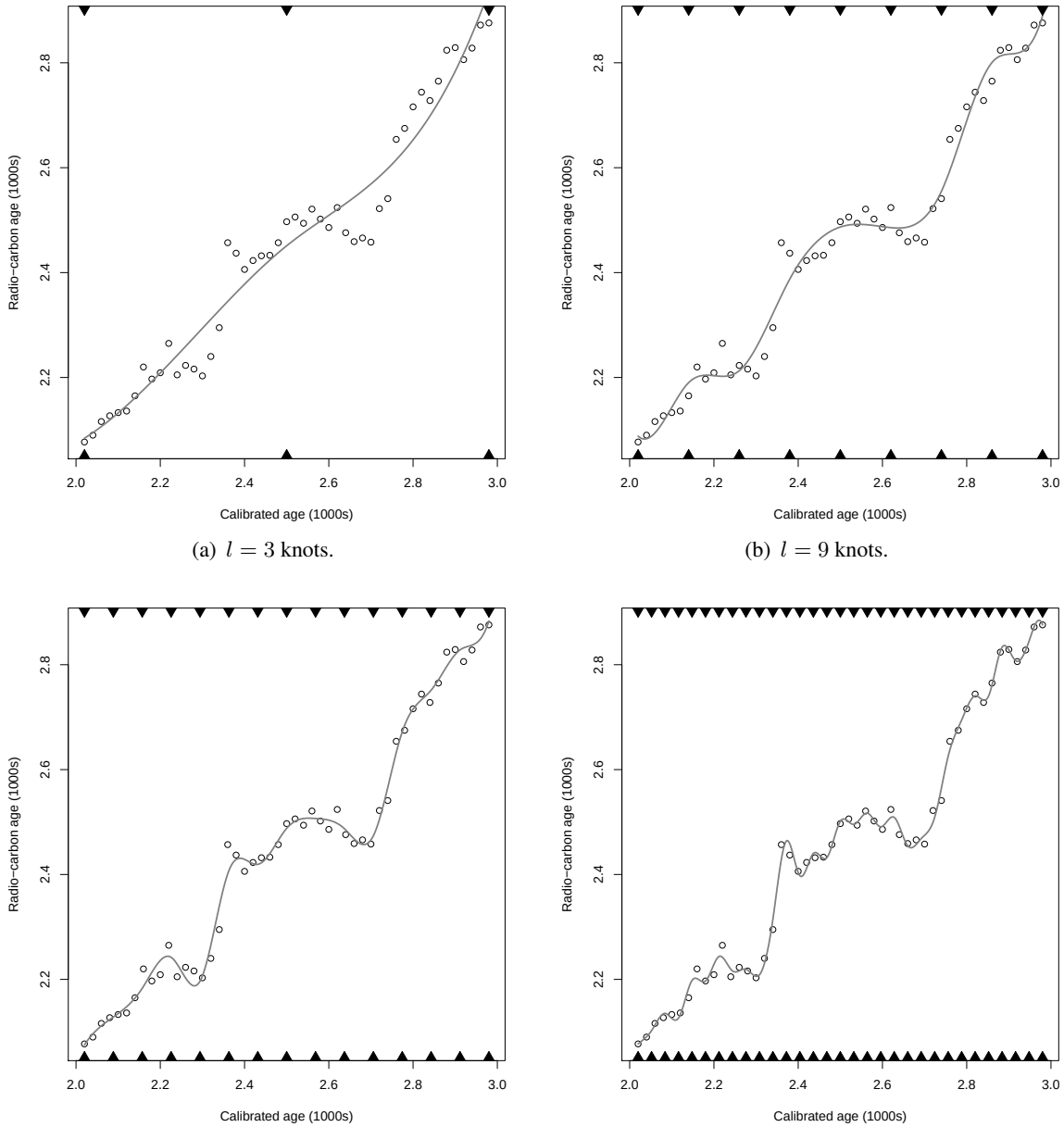
(c) Degree $r = 2$ (continuous first derivative).

(d) Degree $r = 3$ (continuous second derivative).

Figure 3.7. Splines of degree $r \in \{0, 1, 2, 3\}$ fitted to the radiocarbon data.

Choice of the number of knots l**.** In an (unpenalised) spline the number of knots acts as a smoothing parameter. The more knots are used, the more flexible the regression function can become. A more flexible regression function has a lower bias, but a higher variance.

Example 3.5 (Radiocarbon dating (continued)). Figure 3.8 shows a cubic spline fitted to the radiocarbon data using an increasing number of knots. Too few knots lead to an underfit to the



(a) $l = 3$ knots.

(b) $l = 9$ knots.

(c) $l = 15$ knots.

(d) $l = 31$ knots.

Figure 3.8. Cubic spline with different number of knots $l \in \{3, 9, 15, 31\}$ fitted to the radiocarbon data.

data: the fitted function does not fully represent the relationship between radiocarbon age and calendar age. Too many knots on the other hand lead to an overfit: the spline does not only pick up the signal, but also adapts to artefacts in the noise. ◁

Especially when the number of knots is small, the positioning of the knots can be important. The simplest strategy consist of using a set of equally spaced knots; this is computationally the simplest. Alternatively, we can place the knots according to the quantiles of the covariate. This makes the spline more flexible in regions with more data (and thus potentially more information) and less flexible in areas with less data (and potentially less information). A third strategy consists of trying to find an optimal placement of the knots. This usually is computationally very demanding.

Yet another approach consists of using "too many" knots — one knot per observation in the most extreme case — and use a penalty term to control for the smoothness. This avoid the need to select the number of knots altogether. We will study two such approaches in sections 3.2.3 and 3.3.

Splines as a vector space. For a given set of l knots and given degree r, the space of polynomial splines is a vector space, i.e. the sum of two splines as well as a scalar multiples of each spline are again splines. To find the dimension of the vector space have to find the number of "free parameters".

- Each polynomial has $r+1$ parameters and there are $l-1$ polynomials. Thus the spline model has $(r+1)\cdot(l-1)$ parameters. However we cannot choose all these parameters freely, as the resulting function needs to be $r-1$ times continuously differentiable.
- At the $l-2$ inside knots we have to guarantee that $m(\cdot)$ is $r-1$ times continuously differentiable. This corresponds to r constraints ($r-1$ constraints for each derivative and one for $m(\cdot)$ to be continuous). Thus there are $r\cdot(l-2)$ constraints (which are all linearly independent).

Thus there are $(r+1)\cdot(l-1)-r\cdot(l-2)=r+l-1$ free parameters. Thus the vector space of polynomial splines of degree r with l knots is $r+l-1$.

In section 3.2.4 we will explore different ways of constructing a basis for this space. The dimension will come in handy when proving that a given set of basis functions is indeed a basis of this space, as we only need to show that the basis functions are independent and that we use the correct number of basis functions.

Natural cubic splines. Finally, we will introduce the concept of a natural cubic spline. It is based on the idea that it is "safer" (or more "natural") to assume that the curvature of the spline at the first and last knot is zero. If we were to extrapolate, we would then extrapolate linearly.

Definition 3.2 (Natural cubic spline). *A polynomial spline $m : [a, b] \to \mathbb{R}$ of degree 3 is called a natural cubic spline if $m''(a) = m''(b) = 0$.*

Given a set of l knots the vector space of all cubic splines has dimension $l + 2$. Natural cubic splines introduce two additional constraints, thus they form a vector space of dimension l. This makes natural cubic splines perfectly suited for interpolation.

Proposition 3.3. *A set of l points (x_i, y_i) can be exactly interpolated using a natural cubic spline with the $x_1 < \ldots < x_l$ as knots. The interpolating natural cubic spline is unique.*

Proof. The space of natural cubic splines with knots at $x_1, \ldots, x_l$ is vector space of dimension l. Introducing l additional constraints ($y_i = m(x_i)$ for $i = 1, \ldots, l$) yields a system of l equations and l free parameters, which yields a unique solution.[4] □

Natural cubic splines can be generated using the function `ns` in R.

In the next section we will show that natural cubic spline have an important optimality property.

3.2.3 Optimality of splines

This section provides a theoretical justification for the choice of splines for flexible regression.

In this section we will ask a rather general question. Given a data set (x_i, y_i) with $a \leq x_i \leq b$ we try to find, amongst all twice continuously differentiable functions, the function which "best" models the relationship between response y_i and covariate x_i.

First of all, we need to specify what we mean by "best". We could look for the function $m(\cdot)$ which yields the smallest least-squares criterion

$$\sum_{i=1}^{n} (y_i - m(x_i))^2$$

This is however not a good idea. Any function which interpolates all the observations (x_i, y_i) would be optimal in this sense, yet such a function would typically not describe the relationship between x_i and y_i but rather model the artefacts of the random noise. Thus we will consider a so-called *penalised* (or *regularised*) criterion which tries to balance out two aspects which are important to us:

[4] Strictly speaking, we would need to show that the system of equations cannot be rank-deficient, which could cause the solution to be either non-unique or non-existing.

Fit to the data. We want $m(\cdot)$ to follow the data closely.

Simplicity/smoothness. We want the function $m(\cdot)$ not to be too complicated so that it generalises well to future unseen data.

We will thus the following penalised fitting criterion

$$\underbrace{\sum_{i=1}^{n}(y_i - m(x_i))^2}_{\text{Fit to the data}} + \lambda \underbrace{\int_a^b m''(x)^2 \, dx}_{\text{Roughness penalty}}, \tag{3.1}$$

where $\lambda > 0$ is a tuning parameter which controls the trade off between following the data and preventing $m(\cdot)$ from being too rough.

We will now establish that the minimiser of (3.2) over all twice continuously differentiable functions has to be a natural cubic spline, i.e. natural cubic splines with knots at each of the unique x_i are in this sense the optimal class functions.

We will start by stating that natural cubic splines are optimal interpolators, in the sense that they minimise the roughness penalty $\int_a^b m''(x)^2 \, dx$.

Lemma 3.4. *Amongst all functions on $[a, b]$ which are twice continuously differentiable and which interpolate the set of points (x_i, y_i), a natural cubic spline with knots at the x_i yields the smallest roughness penalty*

$$\int_a^b m''(x)^2 \, dx.$$

Spline-based interpolation is implemented in the Rfunctions `spline` and `splinefun`.

We will now generalise the result about interpolation to the case of smoothing.

$$\underbrace{\sum_{i=1}^{n}(y_i - m(x_i))^2}_{\text{Model fit}} + \lambda \underbrace{\int_a^b m''(x)^2 \, dx}_{\text{Roughness penalty}}, \tag{3.2}$$

where λ is a tuning parameter which controls the trade off between following the data and preventing $m(\cdot)$ from being too rough.

Theorem 3.5. *The minimiser of*

$$\sum_{i=1}^{n}(y_i - m(x_i))^2 + \lambda \cdot \int_a^b m''(x)^2 \, dx$$

amongst all twice continuously differentiable functions on $[a, b]$ is given by a a natural cubic spline with knots in the unique x_i.

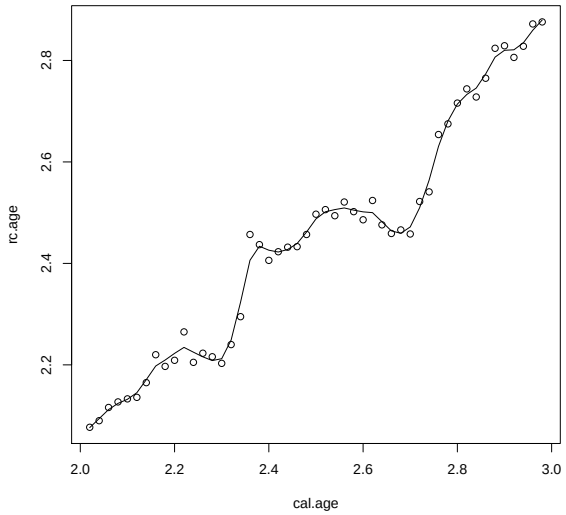
This is an extremely powerful theorem. Even though we consider the entire infinite-dimensional vector space of all twice continuously differentiable functions, we only need to consider the finite-dimensional vector space of natural cubic splines. We have thus reduced the complexity of the optimisation problem to the comparatively simple problem of finding the optimal coefficients of the natural cubic spline. This can be done using least-squares. Note that the proof did not make use of the fact that we have used the least-squares loss function. In fact, the theorem holds for any pointwise loss function.

The technique of *smoothing splines* is based on this theoretical result and finds the natural cubic spline minimising (3.2), and, due to the theorem, the optimal function amongst all twice continuously differentiable functions. This approach is implemented in the Rfunction `smooth.spline`.

```
smsp <- with(radiocarbon, {
    plot(cal.age, rc.age)
    smooth.spline(cal.age, rc.age)
})
smsp
```

```
## Call:
## smooth.spline(x = cal.age, y = rc.age)
##
## Smoothing Parameter  spar= 0.3734837  lambda= 3.249006e-06 (15 iterations)
## Equivalent Degrees of Freedom (Df): 23.07235
## Penalized Criterion (RSS): 0.0105881
## GCV: 0.00077177
```

```
lines(smsp)
```



We will revisit the idea of regularisaton in more detail in section 3.3.

3.2.4 Constructing splines

In this section we will studies two ways of constructing a basis for the vector space of polynomial splines: the truncated power basis and the B-spline basis. We will only cover the case of generic polynomial splines. However one can modify these bases to only span the space of natural cubic splines.

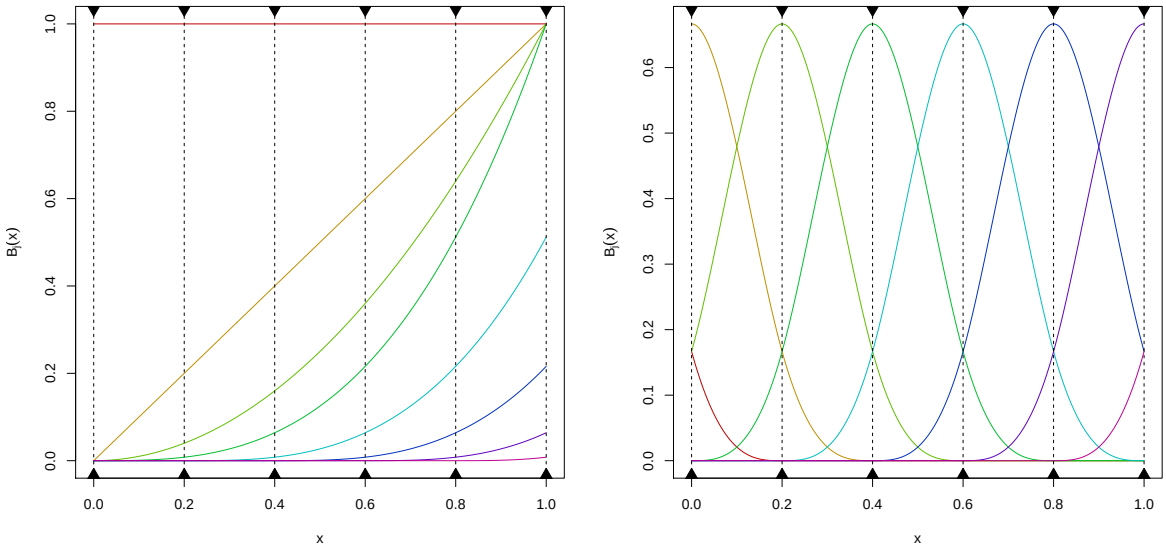
Truncated power basis. The simplest basis for polynomial splines is the truncated power basis.

Definition 3.6 (Truncated power basis). *Given a set of knots* $a = \kappa_1 < \ldots < \kappa_l = b$ *the truncated power basis of degree* r *is given by*

$$\left(1, x, \ldots, x^{r-1}, (x-\kappa_1)_+^r, (x-\kappa_2)_+^r, \ldots, (x-\kappa_{l-1})_+^r\right),$$

where $(z)_+^r = \begin{cases} z^r & \text{for } z > 0 \\ 0 & \text{otherwise.} \end{cases}$

The truncated power basis has $r+l-1$ basis functions. It is easy to see that they are linearly independent. Thus the truncated power basis is indeed a basis of the vector space of polynomial splines. Figure 3.9 shows the truncated power series basis of degree 3 for six equally spaced knots.



(a) Truncated power series (b) B-splines

Figure 3.9. Basis functions $B_j(x)$ of the cubic truncated power series basis (left panel) and B-splines (right panel). The vertical lines indicate the location of the knots.

To fit a polynomial spline to data we can exploit the fact the truncated power basis is a basis of the vector space of polynomial splines of the given degree and with the given set of knots.

Thus we can write any spline $m(\cdot)$ as a linear combination of the basis functions, i.e.

$$m(x) = \beta_0 + \beta_1 x + \ldots + \beta_{r-1} x^{r-1} + \beta_r (x - \kappa_1)_+^r + \ldots + \beta_{r+l-2} (x - \kappa_{l-1})_+^r$$

We can thus find the optimal spline $m(\cdot)$ by just finding the optimal set of coefficients β_j, which is nothing other than a linear regression problem with design matrix

$$\mathbf{B} = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{r-1} & (x_1 - \kappa_1)_+^r & \ldots & (x_1 - \kappa_{l-1})_+^r \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^{r-1} & (x_n - \kappa_1)_+^r & \ldots & (x_n - \kappa_{l-1})_+^r \end{pmatrix}$$

We can use the design matrix $\mathbf{B}$ in exactly the same way as we would use the design matrix of a classical linear model.

We can interpret the truncated power series as a regression model in which the leading coefficient changes at each knot. At each knot, the remaining coefficients change as well. However they are fully constrained by the condition that the spline has to be $r - 1$ times continuously differentiable at each knot.

Example 3.6 (Radiocarbon data (continued)). Figure 3.10 illustrates the use of a truncated power series basis for fitting a spline-based flexible regression model for the radiocarbon data.

As one can see from the middle panel of figure 3.10 and from figure 3.11, some of the estimated coefficients are very large: some of the basis functions are scaled up by a factor of more than 1000, with "neighbouring" basis functions having opposite signs. The reason for this is the high correlation between the columns of the design matrix of the truncated power series. The largest correlation between columns is 0.99921, which is very close to 1.

Figure 3.12 shows a scree plot of the singular values of the design matrix $\mathbf{B}$. The condition number of the matrix $\mathbf{B}$ is 225333.0, with the condition number of $\mathbf{B}^\top \mathbf{B}$ being $5,857,413,839$, i.e. $\mathbf{B}^\top \mathbf{B}$ is close to being numerically singular. This suggests that finding the least-squares estimate of the coefficients is close to being numerically unstable. ◁

We can generate a truncated power basis in R as follows.

```
tpower <- function(x, t, p)
    (x - t) ^ p * (x > t)

tbase <- function(x, xl = min(x), xr = max(x), n.knots = 10, deg = 3) {
  nseg <- n.knots - 1
  dx <- (xr - xl) / nseg
  knots <- seq(xl, xr,  len=n.knots)
  B <- cbind(outer(x-xl,0:(deg-1),"^"),
             outer(x,knots[-length(knots)],function(x,y) pmax(x-y,0))^deg)
  B
}
```

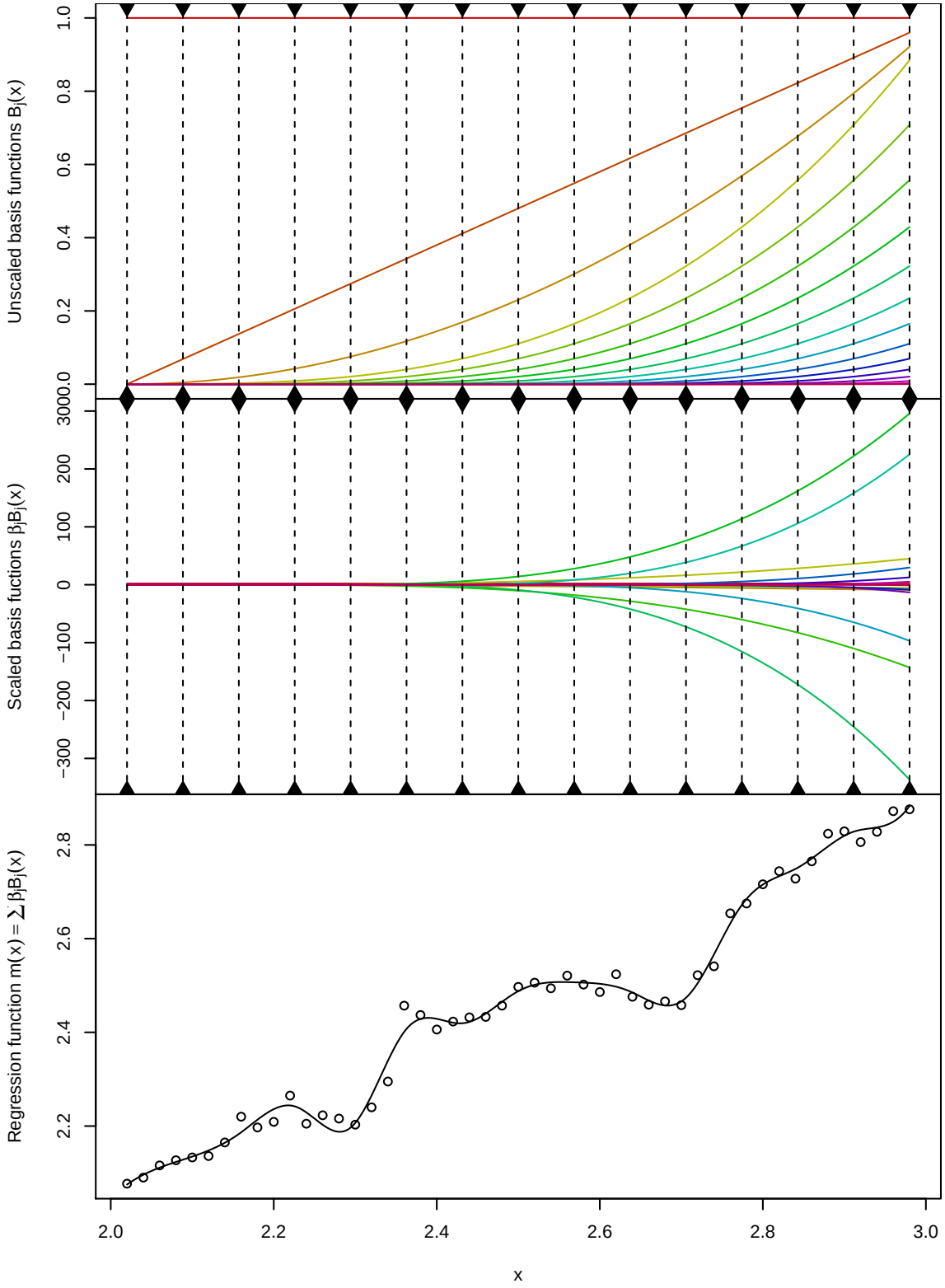


Figure 3.10. Illustration of flexible regression using the truncated power series basis of degree 3 applied to the radiocarbon data. The top panel shows the unscaled basis functions $B_j(x)$. The middle panel shows the scaled basis functions $\hat{\beta}_j B_j(x)$. The bottom panel shows a scatter plot of the data together with the fitted function $\hat{m}(x) = \sum_j \hat{\beta}_j B_j(x)$.

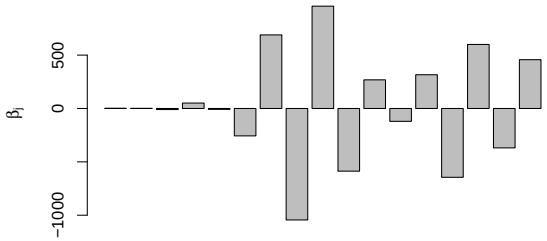


Figure 3.11. Bar plot of the coefficients $\hat{\boldsymbol{\beta}}$ estimated using the truncated power series regression model shown in figure 3.10.

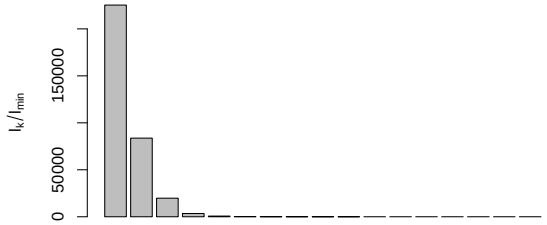
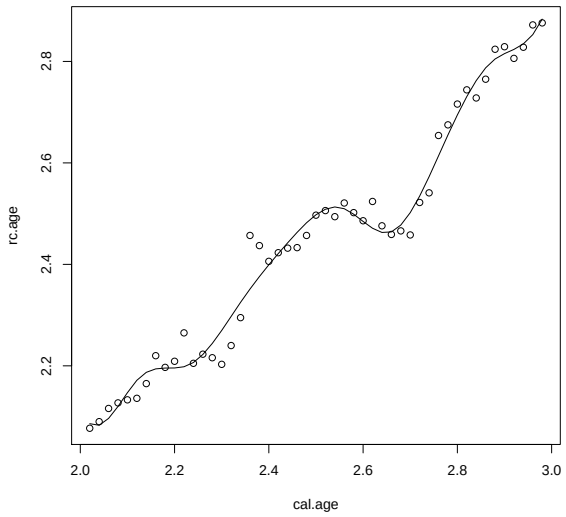


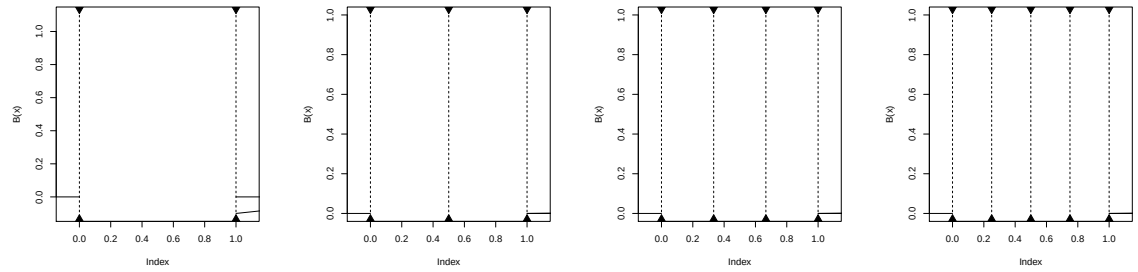
Figure 3.12. Scree plot of the singular values of the design matrix $\mathbf{B}$ (square root of the eigenvalues of the cross-product matrix $\mathbf{B}'\mathbf{B}$) for the truncated power series regression model shown in figure 3.10.

```
B <- tbase(radiocarbon$cal.age, n.knots=10)
y <- radiocarbon$rc.age
beta <- qr.coef(qr(B), y)
y.hat <- B%*%beta
with(radiocarbon, {
    plot(cal.age, rc.age)
    lines(cal.age, y.hat)
})
```



As we have seen in the above example the truncated power basis can easily lead to numerical instability. Thus we will turn to an alternative basis, the so-called B-spline basis.

B-splines. B-splines form a numerically more stable basis. They also make the definition of meaningful penalty matrices easier, which we will exploit in section 3.3.



(a) One basis function of degree $r = 0$ (b) One basis function of degree $r = 1$ (c) One basis function of degree $r = 2$ (d) One basis function of degree $r = 3$

Figure 3.13. One basis function of a B-spline basis with degree $r \in \{0, 1, 2, 3\}$ using $r + 1$ knots.

The key idea of B-splines is to use basis functions which are local, i.e. only non-zero for a "small" proportion of the range of the covariate and which are bounded above. We can think of B-splines as a sequence of "bumps". Figure 3.13 shows a B-spline basis function for degrees $r \in \{0, 1, 2, 3\}$. We will define B splines recursively.

Definition 3.7 (B-spline basis). *(a) Given a set of l knots the B-spline basis of degree 0 is given by the functions $(B_1^0(x), \ldots, B_{l-1}^0)$ with*

$$B_j^0(x) = \begin{cases} 1 & \text{for } \kappa_j \leq x < \kappa_{j+1} \\ 0 & \text{otherwise.} \end{cases}$$

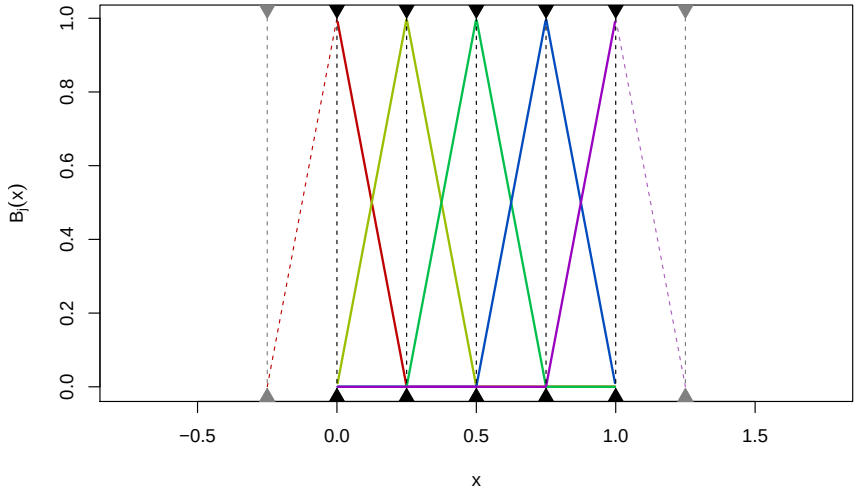
(b) Given a set of l knots the B-spline basis of degree $r > 0$ is given by the functions $(B_1^r(x), \ldots, B_{l+r-1}^r)$ with

$$B_j^r(x) = \frac{x - \kappa_{j-r}}{\kappa_j - \kappa_{j-r}} B_{j-1}^{r-1}(x) + \frac{\kappa_{j+1} - x}{\kappa_{j+1} - \kappa_{j+1-r}} B_j^{r-1}(x).$$

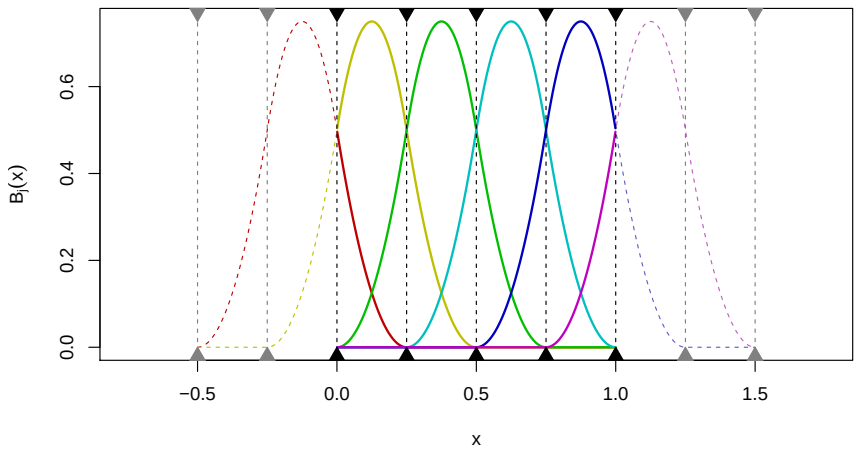
In order to be able to construct the splines recursively we have to introduce additional outside knots to the left of κ_1 and to the right of κ_l. In order to be able to construct a basis of degree r we need r additional outside knots on each side. Figure 3.15 illustrates this idea. These outside knots are just used to construct the basis.

From their recursive definition one can derive that B-splines have the following properties. These can also be seen in figure 3.15.

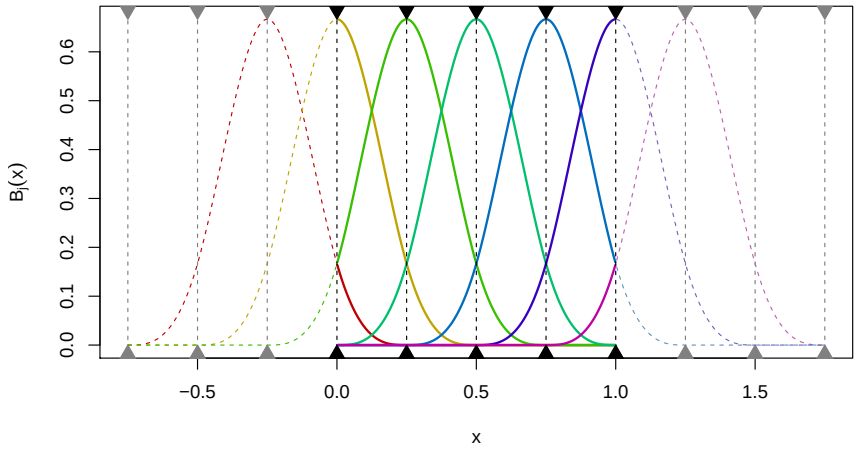
- A B-spline basis function of degree r is made up of $r + 1$ polynomials of degree r. Outside these $r + 1$ intervals, the basis function is zero. This makes the basis functions local.
- At every $x \in (a, b)$ only $r + 1$ basis functions are non-zero.
- The basis functions sum to 1 for all $x \in [a, b]$. This implies that we do not need to include an intercept in the design matrix.



(a) Degree $r = 1$



(b) Degree $r = 2$



(c) Degree $r = 3$

Figure 3.14. B spline bases for degrees $r \in \{1, 2, 3\}$.

– One can show (homework exercise) that the derivative of a B-spline of degree r is a B-spline of degree $r-1$.

We can fit a B-spline model to data by using the design matrix

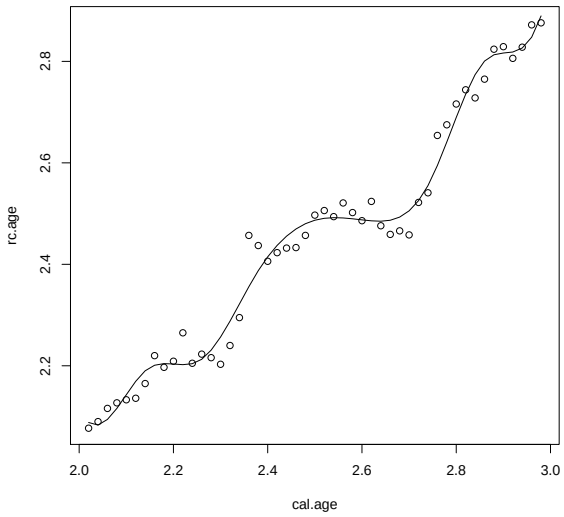
$$\mathbf{B} = \begin{pmatrix} B_1^r(x_1) & \dots & B_{l+r-1}^r(x_1) \\ \vdots & \ddots & \vdots \\ B_1^r(x_n) & \dots & B_{l+r-1}^r(x_n) \end{pmatrix}.$$

Example 3.7 (Radiocarbon data (continued)). Figure 3.15 illustrates the use of a B-spline basis for fitting a spline-based flexible regression model for the radiocarbon data.

The B-spline basis is numerically much better behaved. The coefficient values (cf. figure 3.16) are not too large and the columns of the design matrix $\mathbf{B}$ are much less correlated than the columns of the truncated power basis; the maximum correlation is 0.8309. The condition number of $\mathbf{B}$ is 25.664 (cf. figure 3.17) and the condition number of $\mathbf{B}^\top\mathbf{B}$ is 358.263. ◁

The R function `bs` from the package `splines` can generate a B-spline basis and can be used inside `lm`. The number of basis functions needs to be chosen manually when using `bs`.

```
model <- lm(rc.age~bs(cal.age, df=10), data=radiocarbon)
with(radiocarbon, {
    plot(cal.age, rc.age)
    lines(cal.age, predict(model))
})
```



However, in terms of scaling and properties on the boundary, the basis returned by `bs` differs slightly from the defintions above. The function given below (based on a function written by Paul Eilers) generates a B spline basis which looks exactly like the ones shown above.

```
bbase <- function(x, xl = min(x), xr = max(x), n.knots = 10, deg = 3) {
# Construct B-spline basis (based on a function written by Paul Eilers)
```

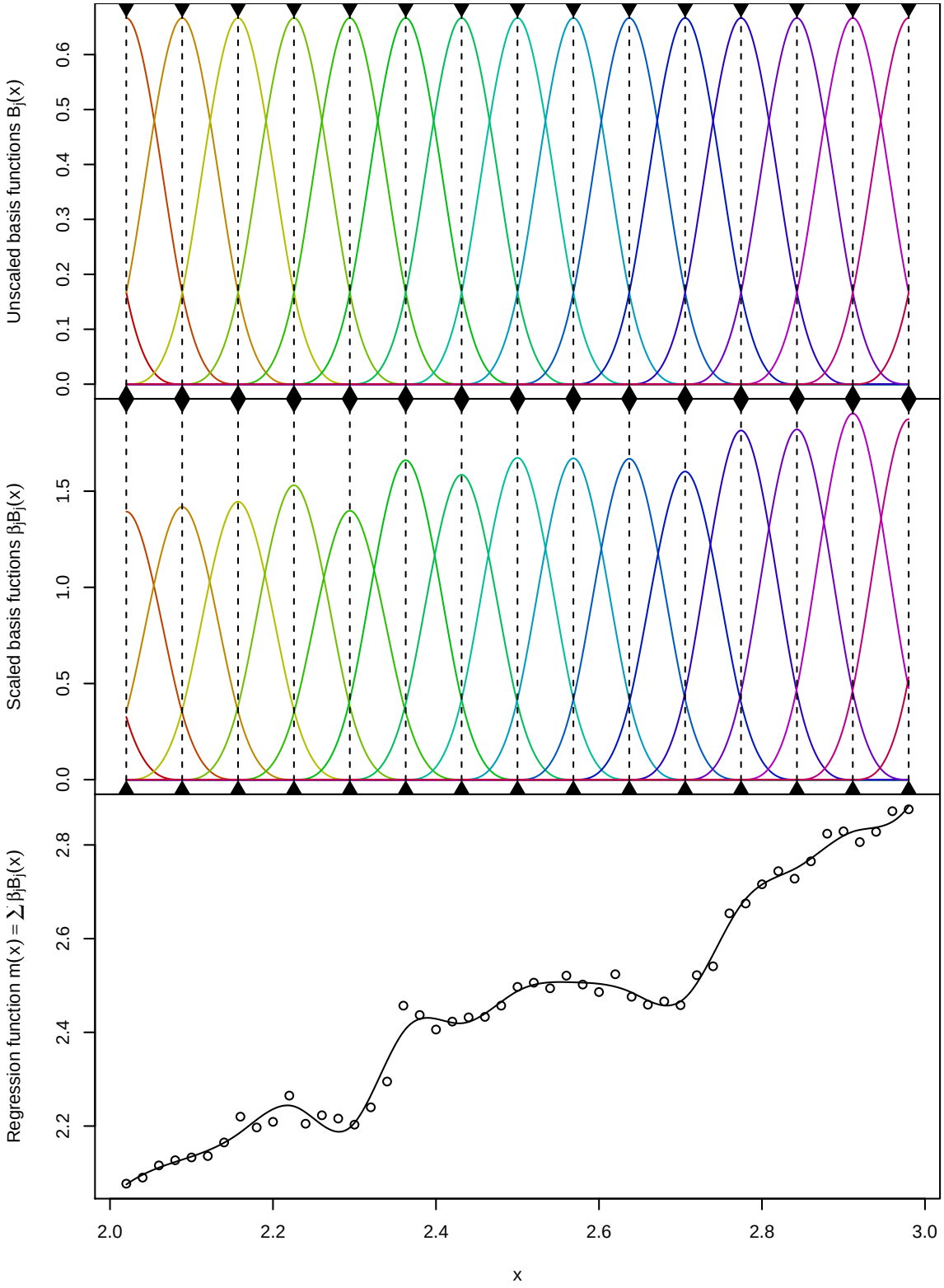


Figure 3.15. Illustration of flexible regression using the B-spline basis applied to the radiocarbon data. The top panel shows the unscaled basis functions $B_j(x)$. The middle panel shows the scaled basis functions $\hat{\beta}_j B_j(x)$. The bottom panel shows a scatter plot of the data together with the fitted function $\hat{f}(x) = \sum_j \hat{\beta}_j B_j(x)$.

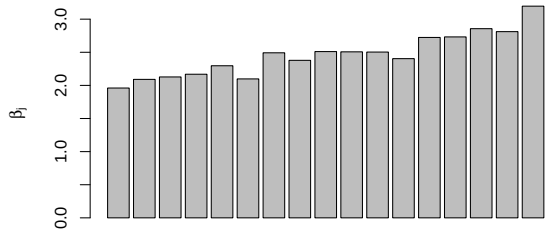


Figure 3.16. Bar plot of the coefficients $\hat{\boldsymbol{\beta}}$ estimated using the B-spline regression model shown in figure 3.15.

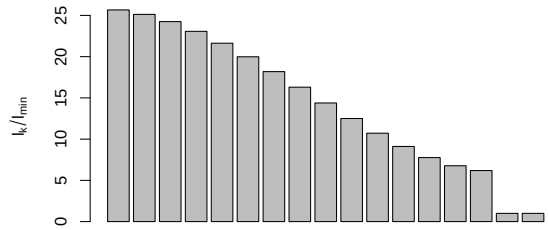


Figure 3.17. Scree plot of the singular values of the design matrix $\mathbf{B}$ (square root of the eigenvalues of the cross-product matrix $\mathbf{B}'\mathbf{B}$) for the B-spline regression model shown in figure 3.15. The condition number of $\mathbf{B}'\mathbf{B}$ is 395.661.

```
    nseg <- n.knots-1
    dx <- (xr - xl) / nseg
    knots <- seq(xl - deg * dx, xr + deg * dx, len = n.knots + 2*deg )
    P <- outer(x, knots, tpower, deg)
    n <- dim(P)[2]
    D <- diff(diag(n), diff = deg + 1) / (gamma(deg + 1) * dx ^ deg)
    B <- (-1) ^ (deg + 1) * P %*% t(D)
    B
}
```

The function `bbase` can be used in the same way as the function `tbase`.

3.3 Penalised splines (P-splines)

A reminder of ridge regression

Ridge regression solves the penalised (or regularised) least-squares criterion

$$\|\mathbf{y} - \mathbf{B}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2,$$

where $\mathbf{B}$ is the matrix of covariates. The solution of this problem is given by

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{I}_p)^{-1}\mathbf{B}^\top\mathbf{y}$$

To compute $\hat{\boldsymbol{\beta}}_{\text{ridge}}$ it is numerically more stable to use a QR decomposition to minimise the augmented system

$$\left\| \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{B} \\ \sqrt{\lambda}\mathbf{I} \end{pmatrix} \boldsymbol{\beta} \right\|^2$$

When using splines the positioning of the knots can have a large influence on the fitted function, especially if a comparatively small number of basis functions is used. One way of avoiding this problem is to use *penalised splines*. They are based on the idea of *not* using the number of basis functions to control the smoothness of the estimate, but to use a roughness penalty to this end. This is similar in spirit to the approach discussed in section 3.2.3, though in most cases it is not necessary to use one basis function per observation. Around 20 to 30 basis functions should be sufficient. Without including a penalty in the fitting criterion this would most likely lead to an overfit to the data. Thus we need to consider a penalised criterion which, just like in section 3.2.3, contains a roughness penalty. In this section we will use $\|\mathbf{D}\boldsymbol{\beta}\|^2$ as roughness penalty, i.e. we choose the regression coefficients $\boldsymbol{\beta}$ by minimising

$$\sum_{i=1}^{n}(y_i - m(x_i))^2 + \lambda\|\mathbf{D}\boldsymbol{\beta}\|^2. \tag{3.3}$$

This objective function is, with the exception of the inclusion of the matrix $\mathbf{D}$, the objective function of ridge regression. As before, λ controls the trade-off between following the data (small λ) and obtaining a strongly regularised curve (large λ). In analogy with ridge regression one can show that the optimal $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta} = (\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{D}^\top\mathbf{D})^{-1}\mathbf{B}^\top\mathbf{y},$$

where $\mathbf{B}$ is the design matrix corresponding to the B-spline basis used for $m(\cdot)$. Numerically, it is more advantageous to represent the penalty term $\lambda\|\mathbf{D}\boldsymbol{\beta}\|^2$ by including it into an expanded design matrix, i.e. to solve

$$\left\| \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{B} \\ \sqrt{\lambda}\mathbf{D} \end{pmatrix} \boldsymbol{\beta} \right\|^2$$

using a QR decomposition.

There are (at least) two possible approaches for choosing $\mathbf{D}$. We can choose $\mathbf{D}$ to be a difference matrix, or we can choose $\mathbf{D}$ such that $\|\mathbf{D}\boldsymbol{\beta}\|^2 = \int_a^b m''(x)^2\,dx$. The former is both conceptually and computationally simpler; the latter is closer to what the theory suggests as optimal.

3.3.1 Difference penalties

The simplest choice of $\mathbf{D}$ is to use a difference penalty. Using the identity matrix for $\mathbf{D}$, as we would in ridge regression, is usually not appropriate: it shrinks all coefficients to zero, i.e. it shrinks the regression function $m(\cdot)$ to zero as well, which rarely desirable (cf. figure

3.18(a)). As we can see from the middle panel of figure 3.15, we obtain a smooth function when neighbouring β_j's are similar.

This can be achieved by using one of the following choices. We assume that we are using equally-spaced knots.

First-order differences. We can set

$$\mathbf{D}_1 = \begin{pmatrix} 1 & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 1 & -1 \end{pmatrix}.$$

This calculates the roughness penalty as the sum of the squared first-order differences between the neighbouring β_j, i.e.

$$\|\mathbf{D}_1\boldsymbol{\beta}\|^2 = \sum_{j=1}^{l+r-2} (\beta_{j+1} - \beta_j)^2$$

This penalty shrinks the coefficients towards a common constant (cf. figure 3.18(b)) and thus shrinks the regression function $m(\cdot)$ towards a constant function. Adding a constant to $m(\cdot)$ does thus not change the penalty.

This penalty is the natural choice if B-splines of order 2 are used.

Second-order differences. We can set

$$\mathbf{D}_2 = \begin{pmatrix} 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 1 & -2 & 1 \end{pmatrix}.$$

This calculates the roughness penalty as the sum of the squared second-order differences between the neighbouring β_j, i.e.

$$\|\mathbf{D}_2\boldsymbol{\beta}\|^2 = \sum_{j=1}^{l+r-3} (\beta_{j+2} - 2\beta_{j+1} + \beta_j)^2$$

This penalty shrinks the coefficients towards a linear sequence (cf. figure 3.18(c)) and thus shrinks the regression function $m(\cdot)$ towards a linear function. Adding a linear function to $m(\cdot)$ does thus not change the penalty.

This penalty is the natural choice if B-splines of order 3 are used.

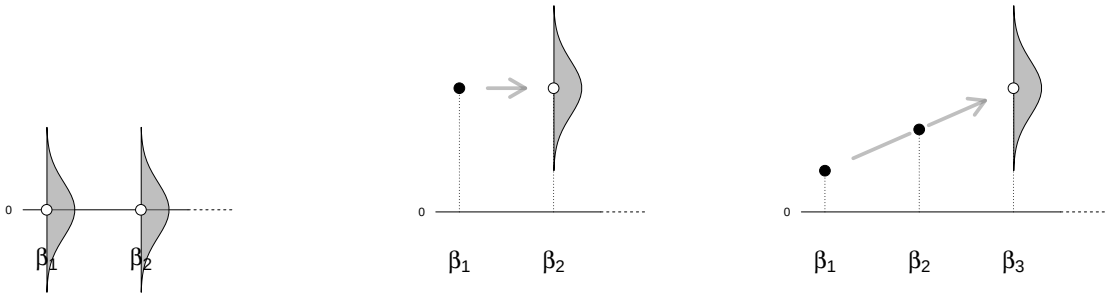
Higher-order differences. Higher-order difference matrices can be constructed using the recursive formula $\mathbf{D}_r = \mathbf{D}_1\mathbf{D}_{r-1}$ where $\mathbf{D}_r$ denotes the penalty matrix of order r.

Example 3.8 (Radiocarbon dating (continued)). Figure 3.19 shows the model fit obtained when fitting a P-spline model with different values of the smoothing parameter λ. The smaller λ the closer the fitted function $\hat{m}(\cdot)$ is to the data, which leads for very small values of λ to an overfit to the data. ◁

Penalty interpretation: Only an all-zero coefficient vector incurs no penalty.
Bayesian interpretation: Independent zero-mean Gaussian prior.

Penalty interpretation: Only an all constant coefficient vector incurs no penalty.
Bayesian interpretation: Conditional distribution of β_2 given β_1 is Gaussian with mean β_1.
(First-order random walk)

Penalty interpretation: Only a coefficient vector which forms a linear sequence incurs no penalty.
Bayesian interpretation: Conditional distribution of β_3 given β_1 and β_2 is Gaussian with mean $2 \cdot \beta_2 - \beta_1$.
(Second-order random walk)



(a) Illustration of a 0-th order penalty (ridge regression).

(b) Illustration of a first-order penalty.

(c) Illustration of a second-order penalty.

Figure 3.18. Illustration of difference penalties of order 0 to 2.

3.3.2 Other penalties

Difference penalties are not the only choice of penalty matrix. An alternative choice consists of choosing $\mathbf{D}$ such that $\|\mathbf{D}\boldsymbol{\beta}\|^2 = \int_a^b m''(x)^2 \, dx$, which is the roughness penalty we have used in section 3.2.3.

Using that $m''(x) = \sum_{j=1}^{l+r-1} \beta_j B_j''(x)$ we have that

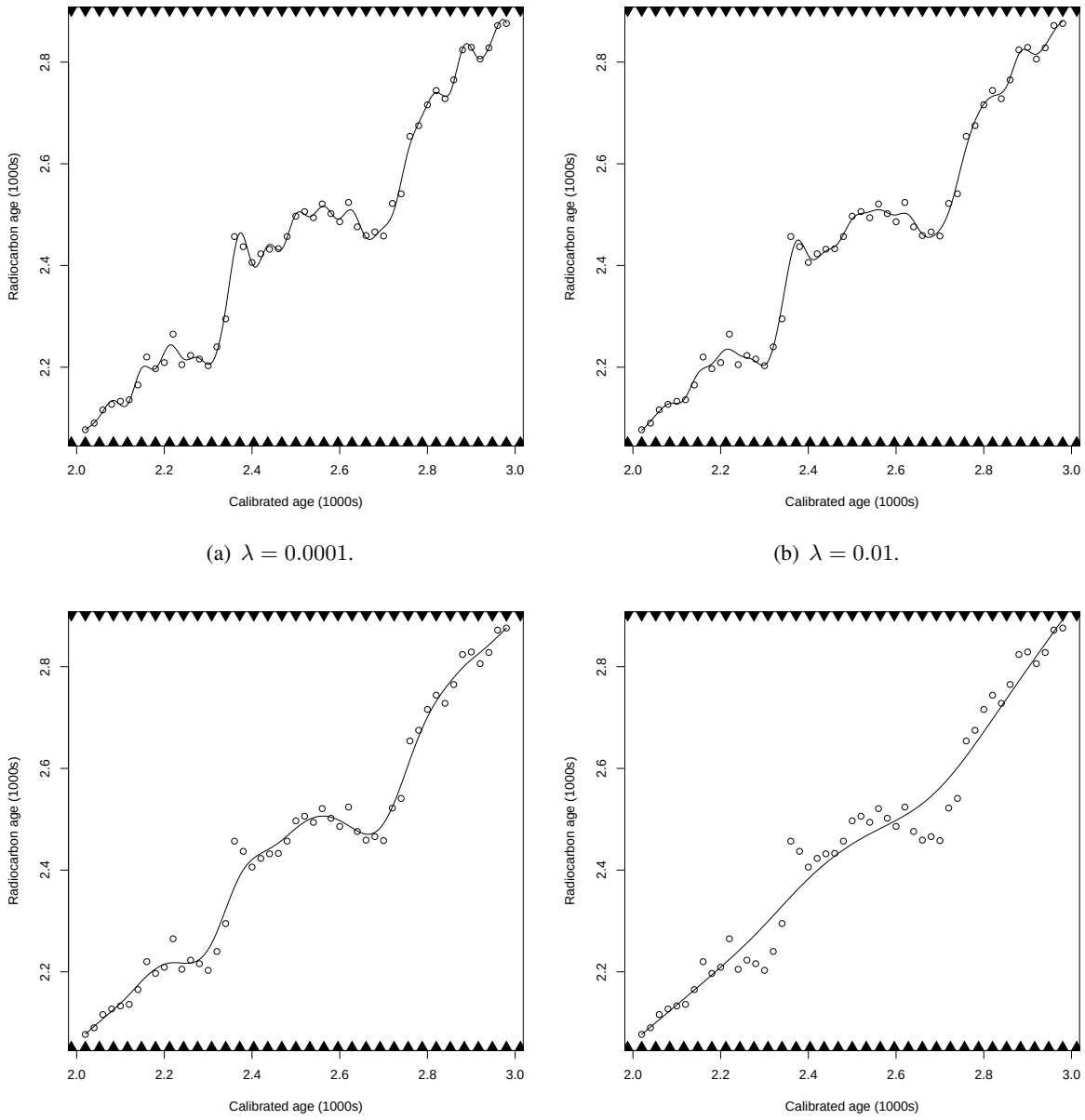
$$\begin{aligned}\int_a^b m''(x)^2 \, dx &= \sum_{j=1}^{l+r-1} \sum_{k=1}^{l+r-1} \beta_j \beta_k \int_a^b B_j''(x) B_k''(x) \, dx \\ &= \boldsymbol{\beta}^\top \begin{pmatrix} \int_a^b B_1''(x) B_1''(x) \, dx & \dots & \int_a^b B_1''(x) B_{l+r-1}''(x) \, dx \\ \vdots & \ddots & \vdots \\ \int_a^b B_1''(x) B_{l+r-1}''(x) \, dx & \dots & \int_a^b B_{l+r-1}''(x) B_{l+r-1}''(x) \, dx \end{pmatrix} \boldsymbol{\beta}\end{aligned}$$

Thus we just need to choose $\mathbf{D}$ such that

$$\mathbf{D}^\top \mathbf{D} = \begin{pmatrix} \int_a^b B_1''(x) B_1''(x) \, dx & \dots & \int_a^b B_1''(x) B_{l+r-1}''(x) \, dx \\ \vdots & \ddots & \vdots \\ \int_a^b B_1''(x) B_{l+r-1}''(x) \, dx & \dots & \int_a^b B_{l+r-1}''(x) B_{l+r-1}''(x) \, dx \end{pmatrix}.$$

3.3.3 Effective degrees of freedom

Finally we introduce the notion of effective degrees of freedom, also sometimes called the effective number of parameters. In an un-penalised regression problem, the number of parameters provides us with information about the complexity of the model. More complex models have more parameters than simpler models. For penalised regression problems counting the



(a) $\lambda = 0.0001$.

(b) $\lambda = 0.01$.

(c) $\lambda = 1$.

(d) $\lambda = 10$.

Figure 3.19. P-spline with different values of the smoothing parameter λ fitted to the radiocarbon data.

parameters is however not meaningful. Due to the roughness penalty not all parameters are "free". Recall that in linear regression the hat matrix $\mathbf{S} = \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top$ is a projection matrix and thus the trace $\mathrm{tr}(\mathbf{S})$ equals the number of parameters. We can generalise this to penalised models and define the effective degrees of freedom as

$$\mathrm{edf}(\lambda) = \mathrm{tr}(\mathbf{S}_\lambda),$$

where $\mathbf{S}_\lambda = \mathbf{B}(\mathbf{B}^\top\mathbf{B} + \lambda\mathbf{D}^\top\mathbf{D})^{-1}\mathbf{B}^\top$.

3.3.4 Random effects interpretation

Random effect models – Likelihood

In the random effects model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\varepsilon}$$

with error term $\boldsymbol{\varepsilon} \sim \mathsf{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and random effect $\boldsymbol{\gamma} \sim \mathsf{N}(\mathbf{0}, \tau^2\mathbf{I})$ twice the loglikelihood is (ignoring the variance parameters) given by

$$-\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top\boldsymbol{\alpha} - \mathbf{z}_i^\top\boldsymbol{\gamma})^2 - \frac{1}{\tau^2}\sum_{j=1}^{q}\gamma_j^2$$

Comparing the penalised least squares criterion (3.3) to the loglikelihood suggests that we can interpret the penalised regression model as a random effects model with no fixed effect and random effect $\boldsymbol{\beta}$. However the problem is that, at least for difference matrices, $\mathbf{D}^\top\mathbf{D}$ is not of full rank, thus we cannot take its inverse matrix square root. In order to obtain a proper random-effects representation we need to "split" $\boldsymbol{\beta}$ into an (unpenalised) fixed effect and a (penalised) random effect.

In the following we will only consider the case of a difference penalty of order 1 or 2. In the case of a first-order difference penalty we define $\mathbf{G} = (1, \ldots, 1)$. For a second-order difference penalty we define $\mathbf{G} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & 2 & \ldots & l+r-1 \end{pmatrix}$. The rows in $\mathbf{G}$ are parameter sequences which do not incur a penalty, i.e. $\mathbf{GD} = \mathbf{0}$. We also define $\mathbf{H} = \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top)^{-1}$. We can now write

$$\boldsymbol{\beta} = \mathbf{G}\boldsymbol{\alpha} + \mathbf{H}\boldsymbol{\gamma}$$

Because $\mathbf{D}\mathbf{D}^\top$ is of full rank we have that $\mathbf{DH} = \mathbf{D}\mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top)^{-1} = \mathbf{I}$. Plugging this into the objective function (3.3) gives

$$\|\mathbf{y}-\mathbf{BG}\boldsymbol{\alpha}-\mathbf{BH}\boldsymbol{\gamma}\|^2+\lambda\Big(\underbrace{\boldsymbol{\alpha}\mathbf{G}^\top\mathbf{D}^\top\mathbf{DG}\boldsymbol{\alpha}}_{=0}+2\underbrace{\boldsymbol{\alpha}\mathbf{G}^\top\mathbf{D}^\top\mathbf{DH}\boldsymbol{\gamma}}_{=0}+\boldsymbol{\gamma}^\top\underbrace{\mathbf{H}^\top\mathbf{D}^\top\mathbf{DH}}_{=\mathbf{I}}\boldsymbol{\gamma}\Big)$$
$$=\|\mathbf{y}-\mathbf{BG}\boldsymbol{\alpha}-\mathbf{BH}\boldsymbol{\gamma}\|^2+\lambda\|\boldsymbol{\gamma}\|^2$$

Defining $\mathbf{X}=\mathbf{BG}$ and $\mathbf{Z}=\mathbf{BH}$ and denoting rows of $\mathbf{X}$ and $\mathbf{Z}$ by $\mathbf{x}_i$ and $\mathbf{z}_i$ respectively, this is equivalent to

$$\sum_{i=1}^{n}(y_i-\mathbf{x}_i^\top\boldsymbol{\alpha}-\mathbf{z}_i^\top\boldsymbol{\gamma})^2+\lambda\sum_{j=1}^{q}\gamma_j^2,$$

which is $-\sigma^2$ times the loglikelihood of a random-effects model, which we have stated above. Hereby we have used $\lambda=\sigma^2/\tau^2$.

Thus the penalised regression model is nothing other than a random effects effect and we can use standard mixed model software to fit these models. Most importantly we can estimate the variances σ^2 and τ^2 is a mixed model (using (restricted) maximum likelihood, which gives us a way of estimating the otherwise rather elusive smoothing parameter $\hat{\lambda}=\hat{\sigma}^2/\hat{\tau}^2$.

3.3.5 Bayesian interpretation

Rather than interpreting the penalised fitting criterion as a random effects model we can treat the penalised regression model as a fully Bayesian model with the following prior and data model.

$$\mathbf{D}\boldsymbol{\beta}|\tau^2\sim\mathsf{N}(\mathbf{0},\tau^2\mathbf{I})$$
$$\mathbf{y}|\boldsymbol{\beta},\sigma^2\sim\mathsf{N}(\mathbf{B}\boldsymbol{\beta},\sigma^2\mathbf{I})$$

The prior distribution of $\boldsymbol{\beta}$ is improper if $\mathbf{D}$ is not of full rank, which is the case for all difference penalties. However in the case of difference penalties the prior distribution of $\boldsymbol{\beta}$ can be expressed in terms of random walks (cf. figure 3.18).

First-order random walk The first-order penalty corresponds to an improper flat prior on β_1 and $\beta_j|\beta_{j-1}\sim\mathsf{N}(\beta_{j-1}|\tau^2)$ (for $j\geq 2$).

Second-order random walk The second-order penalty corresponds to an improper flat prior on β_1 and β_2 and $\beta_j|\beta_{j-1},\beta_{j-2}\sim\mathsf{N}(2\beta_{j-1}-\beta_{j-2}|\tau^2)$ (for $j\geq 3$).

It seems natural to complement the model with priors for σ^2 and τ^2

$$\sigma^2\sim\mathsf{IG}(a_{\sigma^2},b_{\sigma^2})$$

$$\tau^2\sim\mathsf{IG}(a_{\tau^2},b_{\tau^2})$$

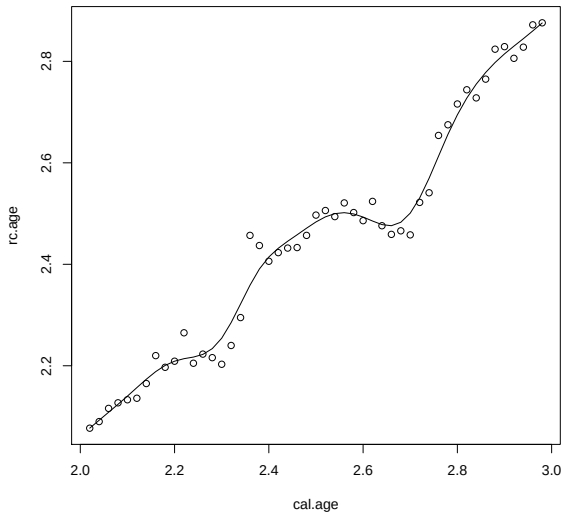
Inference can then be carried out efficiently using a Gibbs sampler. This model and many other Bayesian smoothing models are implemented in the software `BayesX`.

Rather than placing a independent inverse-gamma prior on τ^2 we can set $\tau^2 = \sigma^2/\lambda$ and place a prior of our choice on λ. In this model the posterior distribution distribution of λ does not follow a known distribution, but can be evaluated efficiently, as all the other parameters can be integrated out in closed form. The drawback is that the integration over λ would need to be carried out numerically, which suggests that this approach is better suited for an empirical Bayes strategy for estimating λ.

3.3.6 Penalised splines in R

We can fit a penalised spline from "first principles" in R as follows.

```
B <- bbase(radiocarbon$cal.age, n.knots=25)
D <- diff(diag(ncol(B)), diff=2)
y <- radiocarbon$rc.age
lambda <- 1
beta <- qr.coef(qr(rbind(B, lambda*D)), c(y, rep(0, nrow(D))))
y.hat <- B%*%beta
with(radiocarbon, {
    plot(cal.age, rc.age)
    lines(cal.age, y.hat)
})
```



The parameter λ would need to be tuned manually.

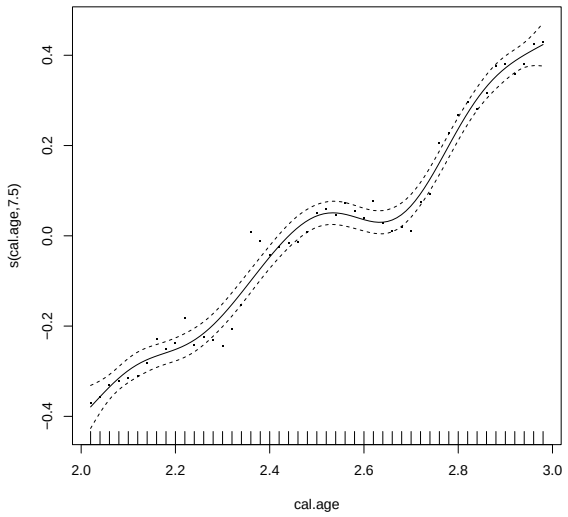
It is however simpler to use the function `gam` from the package `mgcv`, which automatically tunes the smoothing parameters (though these can also be set manually, if needed).

```
model <- gam(rc.age~s(cal.age), data=radiocarbon)
model

   ##
   ## Family: gaussian
   ## Link function: identity
   ##
   ## Formula:
```

```
   ## rc.age ~ s(cal.age)
   ##
   ## Estimated degrees of freedom:
   ## 7.5  total = 8.5
   ##
   ## GCV score: 0.001497071

plot(model, residuals=TRUE)
```



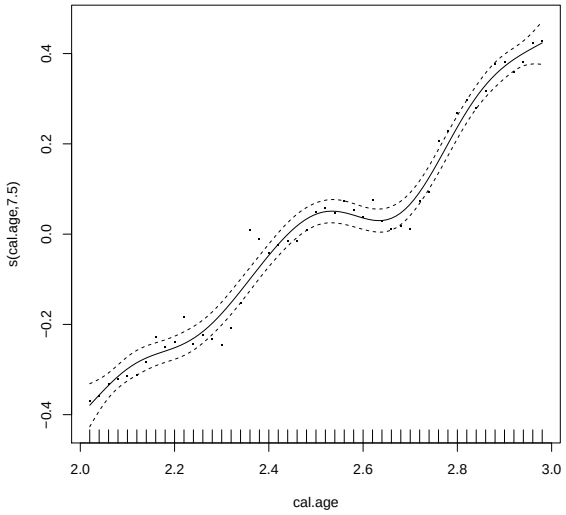
The function `s` uses by default a penalty based on the integrated squared second derivative, but can be set to use a difference penalty by using the additional argument `bs='ps'`,

We can also use *BayesX* to estimate a penalised spline model in Bayesian framework (using the package `R2BayesX`).

```
model <- bayesx(rc.age ~ sx(cal.age), data = radiocarbon)
model

   ## Call:
   ## bayesx(formula = rc.age ~ sx(cal.age), data = radiocarbon)
   ## Summary:
   ## N = 49  burnin = 2000  DIC = 70.3954  pd = 15.9458
   ## family = gaussian  iterations = 12000  step = 10

plot(model)
```



3.4 Splines in more than one dimension

3.4.1 Tensor-product splines

So far we have only covered the construction of spline bases in one dimension. In this section we will see how we can turn a one-dimensional spline basis into a spline basis of any dimension. To keep things simple we shall start with the bivariate case.

Suppose we have two covariates and want to fit a regression model of the form

$$\mathbb{E}(Y_i) = m(x_{i1}, x_{i2}),$$

where $m(\cdot, \cdot)$ is a bivariate surface.

We start by placing a basis on each dimension separately. Denote by $B_1^{(1)}(x_1), \ldots, B_{l_1+r-1}^{(1)}(x)$ the basis functions placed on the first covariate and by $B_1^{(2)}(x_1), \ldots, B_{l_2+r-1}^{(2)}(x)$ the basis functions placed on the second covariate. We now define a set of basis functions

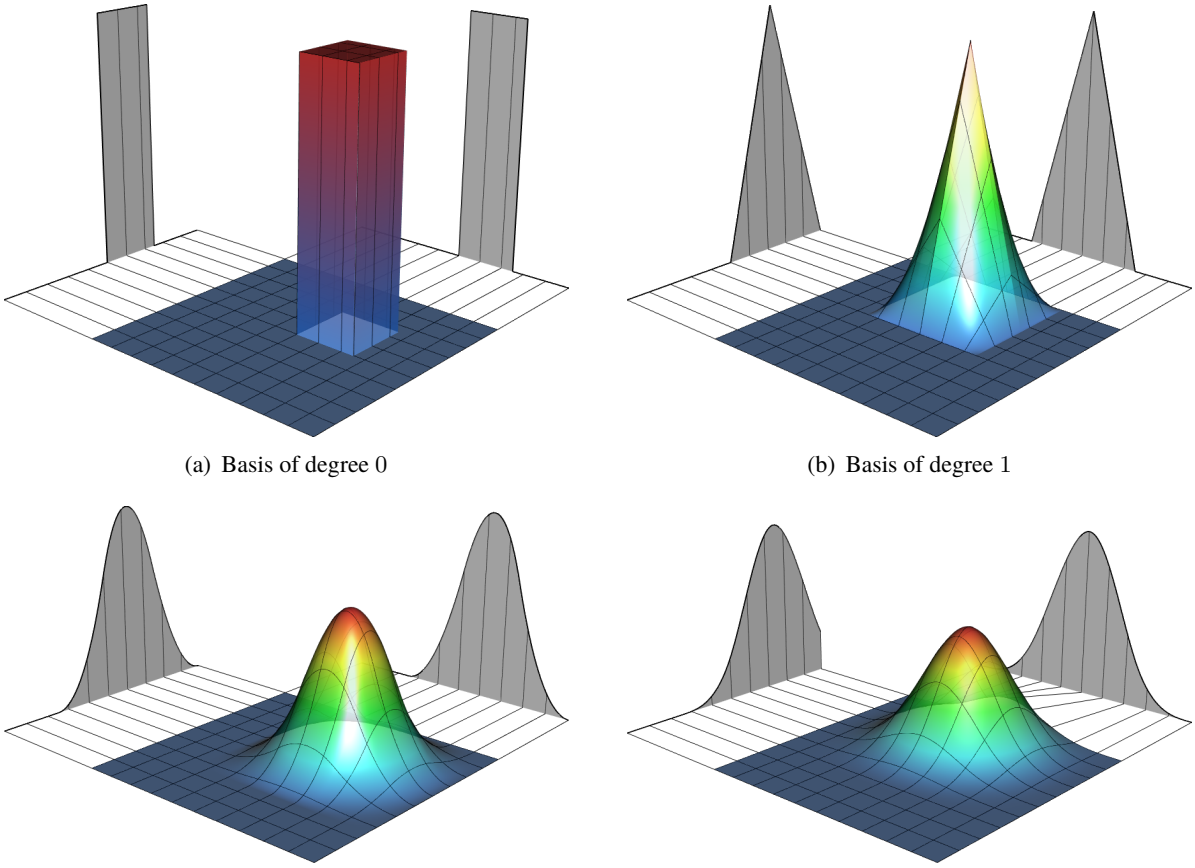
$$B_{jk}(x_1, x_2) = B_j^{(1)}(x_1) \cdot B_k^{(2)}(x_2)$$

for $j \in 1, \ldots, l_1 + r - 1$ and $k \in 1, \ldots, l_2 + r - 1$. Figure 3.20 shows how one such bivariate basis function looks like for different degrees of the underlying univariate B-spline. Figure 3.21 shows all 36 bivariate basis functions resulting from two B-spline bases with six basis functions each.

We will now use the basis expansion

$$m(x_{i1}, x_{i2}) = \sum_{j=1}^{l_1+r-1} \beta_{jk} B_{jk}(x_1, x_2)$$

which corresponds to the design matrix



(a) Basis of degree 0 (b) Basis of degree 1

(c) Basis of degree 2 (d) Basis of degree 3

Figure 3.20. Illustration of the construction of a single bivariate B-spline basis function $B_{jk}(x_1, x_2) = B_j(x_1) \cdot B_k(x_2)$ for B-spline bases of different degree.

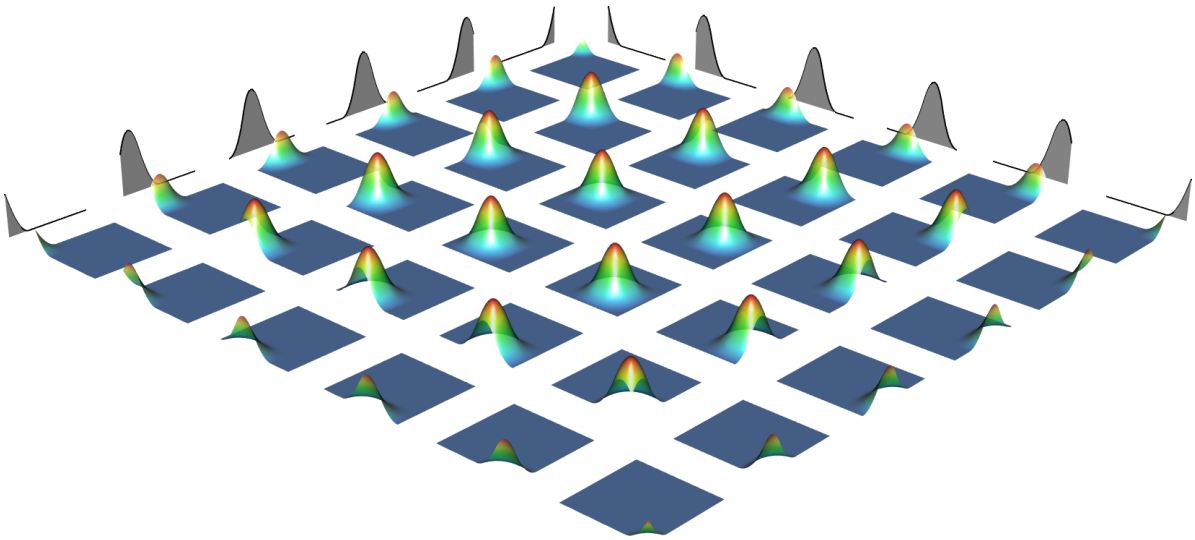


Figure 3.21. Illustration of the construction of a bivariate B-spline basis created from a univariate B-spline basis.

$$\mathbf{B} = \begin{pmatrix} B_{11}(x_{11},x_{12}) & \dots & B_{l_1+r-1,1}(x_{11},x_{12}) & B_{12}(x_{11},x_{12}) & \dots & B_{l_1+r-1,2}(x_{11},x_{12}) & \dots & B_{l_1+r-1,l+2+r-1}(x_{11},x_{12}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{11}(x_{n1},x_{n2}) & \dots & B_{l_1+r-1,1}(x_{n1},x_{n1}) & B_{12}(x_{n1},x_{n2}) & \dots & B_{l_1+r-1,2}(x_{n1},x_{n2}) & \dots & B_{l_1+r-1,l+2+r-1}(x_{n1},x_{n2}) \end{pmatrix}$$

and coefficient vector $\boldsymbol{\beta} = (\beta_{11}, \dots, \beta_{l_1+r-1,1}, \beta_{12}, \dots, \beta_{l_2+r-1,2}, \dots, \beta_{l_1+r-1,l_2+r-1})^\top$.

We can generalise this principle of constructing a basis to dimension p by multiplying all combinations of basis functions of the p covariates.

Finally, we need to explain how a penalty matrix can be constructed for this bivariate spline basis. We will explain the basic idea using figure 3.21. A simple way of constructing a roughness penalty consist of applying the univariate roughness penalties to the rows and columns of the basis functions. More mathematically, this corresponds to taking Kronecker products, i.e. using the difference matrix

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}^{(2)} \otimes \mathbf{I}_{l_1+r-1} \\ \mathbf{I}_{l_2+r-1} \otimes \mathbf{D}^{(1)} \end{pmatrix},$$

where $\mathbf{D}^{(1)}$ is the univariate difference matrix used for the first dimension and $\mathbf{D}^{(2)}$ is the univariate difference matrix used for the second dimension.

Example 3.9 (Great Barrier Reef (continued)). Figure 3.22 shows the result of fitting a tensor-product-spline model to the data from example 4.1. The objective is to model a score which represents the composition of the catch as a function of longitude and latitude. ◁

In principle, Tensor-product spline bases can be constructed for any dimension, however the number of basis functions scales exponentially in the dimension, so

Tensor-product splines do not scale well as the dimension is increased. The number of basis function increases exponentially in the dimension. Thus they cannot be used for dimensions beyond three (and in some cases even two).

The R code below illustrates how tensor product P-splines can be fit from first principles.

3.4.2 Thin-plate splines

In this section we generalise natural cubic splines to the bivariate case, which provides an alternative way of bivariate spline smoothing. In section 3.2.3 we have seen that the minimiser of

$$\underbrace{\sum_{i=1}^{n}(y_i - m(x_i))^2}_{\text{Fit to the data}} + \lambda \underbrace{\int_a^b m''(x)^2\, dx}_{\text{Roughness penalty}},$$

has to be a natural cubic spline.

Generalising this variational problem to the bivariate case leads to the objective function

$$\underbrace{\sum_{i=1}^{n}(y_i - m(x_{i1},x_{i2}))^2}_{\text{Fit to the data}} + \lambda \underbrace{\int\int \left(\frac{\partial^2}{\partial x_1^2}m(x_1,x_2) + 2\frac{\partial^2}{\partial x_1 \partial x_2}m(x_1,x_2) + \frac{\partial^2}{\partial x_2^2}m(x_1,x_2)\right)^2 dx_2\, dx_1}_{\text{Roughness penalty}},$$

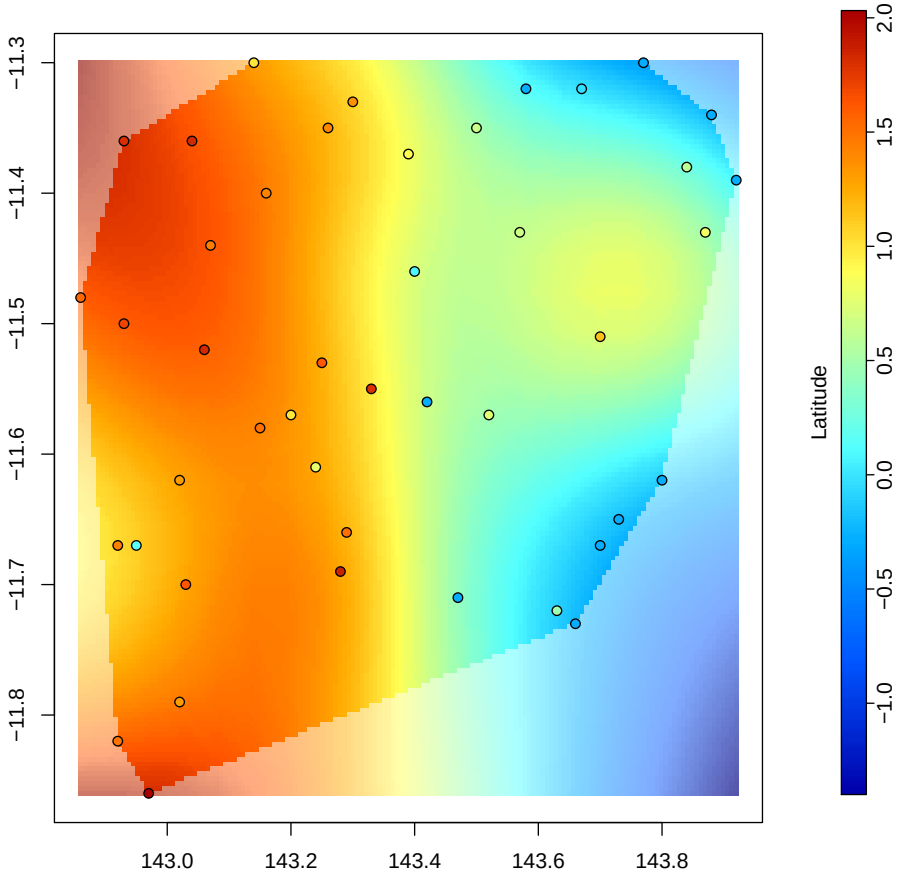


Figure 3.22. Predicted score obtained from a tensor-product-spline model fitted to the Great Barrier Reef data.

The roughness penalty can be interpreted as the bending energy of thin plate of metal. One can show that the solution to his problem has to be a so-called *thin-plate* spline of the form

$$m(\xi_1, \xi_2) = \beta_0 + \beta_1 \xi_1 + \beta_2 \xi_2 + \sum_{i=1}^{n} \beta_{2+i} K\left((\xi_1, \xi_2), (x_{i1}, x_{i2})\right),$$

where $K\left((\xi_1, \xi_2), (\zeta_1, \zeta_2)\right) = \frac{1}{2}\left((\zeta_1 - \xi_1)^2 + (\zeta_2 - \xi_2)^2\right) \cdot \log\left((\zeta_1 - \xi_1)^2 + (\zeta_2 - \xi_2)^2\right)$.

Similar to what we have discussed in section 3.3 we can estimate the coefficients β_j using a penalised least squares criterion. In fact, we need to minimise the objective function

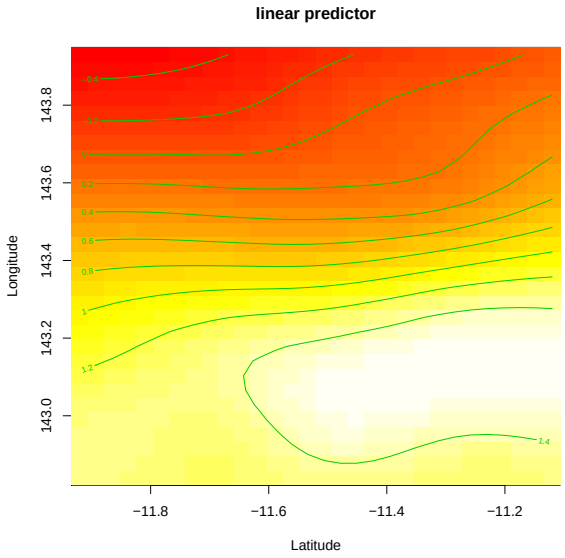
$$\sum_{i=1}^{n} \left(y_i - m(x_{i1}, x_{i2})\right)^2 + \lambda \boldsymbol{\beta}' \mathbf{P} \boldsymbol{\beta}$$

subject to the constraints that $\sum_{i=1}^{n} \beta_{2+i} = \sum_{i=1}^{n} x_{i1}\beta_{2+i} = \sum_{i=1}^{n} x_{i2}\beta_{2+i} = 0$, where

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & K\left((x_{11}, x_{12}), (x_{11}, x_{12})\right) & \dots & K\left((x_{11}, x_{12}), (x_{n1}, x_{n2})\right) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & K\left((x_{n1}, x_{n2}), (x_{11}, x_{12})\right) & \dots & K\left((x_{n1}, x_{n2}), (x_{n1}, x_{n2})\right) \end{pmatrix}$$

Thin-plate splines scale much better in dimensionality, however they do not scale as well as tensor-product splines in the number of data points. Thin-plate splines are the default in `mgcv`'s function `gam`.

```
model <- gam(Score1~s(Latitude, Longitude), data=trawl)
vis.gam(model, plot.type="contour")
```



4

Flexible regression in more than one dimension

In this chapter, methods of extending flexible regression to more than one covariate will be explored. For one covariate, spline methods have been discussed in some detail in earlier chapters while a local fitting approach was outlined in the *preliminary material*. We will revisit the local linear approach and see how that can be extended, before returning to splines. A more general approach known as *additive modeling* will then be described. First of all there is a reminder of the example which was used earlier.

Example 4.1 (Great Barrier Reef data). A survey of the fauna on the sea bed lying between the coast of northern Queensland and the Great Barrier Reef was carried out. The sampling region covered a zone which was closed to commercial fishing, as well as neighbouring zones where fishing was permitted. The variables are:

Zone	an indicator for the closed (1) and open (0) zones
Year	an indicator of 1992 (0) or 1993 (1)
Latitude	latitude of the sampling position
Longitude	longitude of the sampling position
Depth	bottom depth
Score1	catch score 1
Score2	catch score 2

The details of the survey and an analysis of the data are provided by Poiner et al. (1997), *The effects of prawn trawling in the far northern section of the Great Barrier Reef*, CSIRO Division of Marine Research, Queensland Dept. of Primary Industries. ◁

4.1 The local fitting approach with a single covariate

We will briefly rehearse the idea of local fitting in the context of regression data on a response variable y and a single covariate x. A simple nonparametric model has the form

$$y_i = m(x_i) + \varepsilon_i,$$

where the data (x_i, y_i) are described by a smooth curve m plus independent errors ε_i. One approach to estimating m is to use a model we know and fit it locally. For example, we can construct a *local linear regression*. This involves solving the least squares problem

$$\min_{\alpha,\beta} \sum_{i=1}^{n} \{y_i - \alpha - \beta(x_i - x)\}^2 \, w(x_i - x \, ; h)$$

and taking as the estimate at x the value of $\hat{\alpha}$, as this defines the position of the local regression line at the point x. An even simpler approach is to fit a local mean. Specifically, at any point of interest x, we choose our estimator of the curve there as the value of μ which minimises

$$\sum_{i=1}^{n} \{y_i - \mu\}^2 w(x_i - x; h)$$

and this is easily shown to produce the 'running mean'

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} w(x_i - x; h) \, y_i}{\sum_{i=1}^{n} w(x_i - x; h)}.$$

If we do the algebra to minimise the sum-of-squares in the local linear approach, then an explicit formula for the local estimator can be derived as

$$\hat{m}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{\{s_2(x; h) - s_1(x; h)(x_i - x)\} w(x_i - x; h) y_i}{s_2(x; h) s_0(x; h) - s_1(x; h)^2},$$

where $s_r(x; h) = \{\sum (x_i - x)^r w(x_i - x; h)\}/n$.

In both the local mean and the local linear cases, the estimator is seen to be of the form $\sum_i \kappa_i y_i$, where the weights κ_i sum to 1. There is a broad sense then in which even the local linear method is 'locally averaging' the data. In fact, many other forms of nonparametric regression can also be formulated in a similar way.

The *preliminary material* explored the properties of these estimators and showed that the local linear approach has some advantages. Some useful descriptions of the behaviour of the local linear approach is that

$$\mathbb{E}\{\hat{m}(x)\} \approx m(x) + \frac{h^2}{2} m''(x),$$
$$\mathsf{var}\{\hat{m}(x)\} \approx \frac{1}{nh} \left\{ \int w(u)^2 du \right\} \sigma^2 \frac{1}{f(x)},$$

where σ^2 denotes the variance of the error terms ε_i.

It is helpful to express the fitted values of the nonparametric regression as

$$\hat{m} = Sy,$$

where $\hat{m}$ denotes the vector of fitted values, S denotes a *smoothing matrix* whose rows consist of the weights appropriate to estimation at each evaluation point, and y denotes the observed responses in vector form. This linear structure applies with both local fitting and spline approaches and it is very helpful.

For example, it gives us a route to defining *degrees of freedom* by analogy with what happens with the usual linear model, where the number of parameters is the trace of the projection matrix. An approximate version of these can be constructed for nonparametric models as

$$\text{df} = \text{tr}\left\{S\right\}.$$

Similarly, we can construct an estimate of the error variance σ^2 through the residual sum-of-squares, which in a nonparametric setting is simply $\text{RSS} = \sum\{y_i - \hat{m}(x_i)\}^2$. This leads to the estimator of the error variance $\hat{\sigma}^2 = \text{RSS}/\text{df}$. The linear structure of the fitted values also makes it very easy to produce standard errors which quantify the variability of the estimate at any value of x.

```
library(sm)
sm.regression(trawl$Longitude, trawl$Score1, se = TRUE)
```

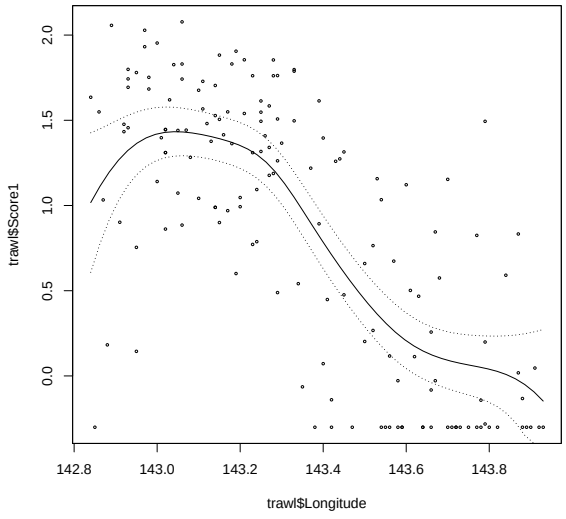


Figure 4.1. A flexible regression curve for the Reef data, with variability bands indicated.

Unfortunately, we can't easily produce confidence intervals for the curve because of the bias mentioned above. However, by adding and subtracting two standard errors at each point on the curve we can produce *variability bands* which express the variation in the curve estimate. In fact, we don't need to rely on the asymptotic formula for variance. If $\hat{m}$ denotes the estimated values of m at a set of evaluation points then

$$\mathsf{var}\{\hat{m}\} = \mathsf{var}\{Sy\} = SS^T\sigma^2$$

and so, by plugging in $\hat{\sigma}^2$ and taking the square root of the diagonal elements, the standard errors at each evaluation point are easily constructed. The plot below illustrates this on the Reef data.

4.2 Local fitting with more than one covariate

It is rare to have problems which involve only a single covariate. For the Reef data a natural extension is to look at the relationship between the catch score and both latitude (x_1) and longitude (x_2), in a model

$$y_i = m(x_{1i}, x_{2i}) + \varepsilon_i.$$

The local linear approach is particularly easy to extend to this setting. If the observed data are denoted by $\{x_{1i}, x_{2i}, y_i; i = 1, \ldots, n\}$, then for estimation at the point (x_1, x_2) the weighted least squares formulation is

$$\min_{\alpha,\beta,\gamma} \sum_{i=1}^{n} \{y_i - \alpha - \beta(x_{1i} - x_1) - \gamma(x_{2i} - x_2)\}^2 \, w(x_{1i} - x_1; h_1) \, w(x_{2i} - x_2; h_2).$$

The value of the fitted surface at (x_1, x_2) is simply $\hat{\alpha}$. With careful thought, the computation can be performed efficiently.

This is illustrated below with one year of Reef data. The effect of longitude dominates, as we see from the earlier nonparametric regression. However, a small effect of latitude is also suggested.

Notice that two smoothing parameters, h_1 and h_2, are now required - one for each covariate.

4.3 Splines with more than one covariate

As we have seen earlier, a b-spline basis provides a set of building blocks for a flexible regression function. The basic idea is to use the combination of a set of overlapping functions and a set of weights for each component to control the shape of the surface created. If we have two covariates then these basis functions need to be functions of both covariates.

$$y = \sum_{i=1}^{p} \sum_{j=1}^{p} \beta_{ij} b_{ij}(x_1, x_2) + \varepsilon.$$

A very simple way of achieving this is to create the basis functions from pairwise products of functions of a single covariate. Specifically,

$$b_{ij}(x_1, x_2) = b_i(x_1) b_j(x_2).$$

```
trawl1 <- subset(trawl, Year == 0)
sm.regression(trawl1[ , c("Longitude", "Latitude")], trawl1$Score1, theta = 120)
```

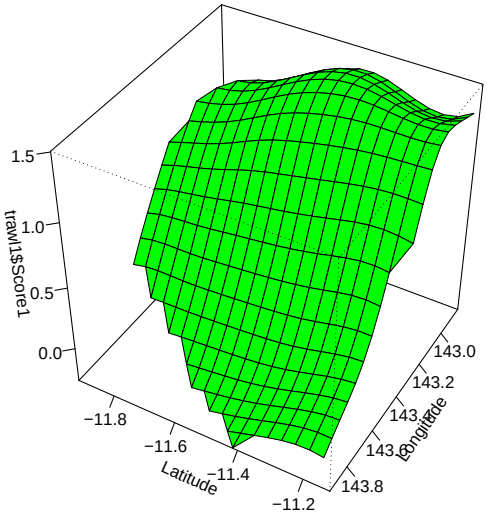


Figure 4.2. Reef data with two covariates for one year.

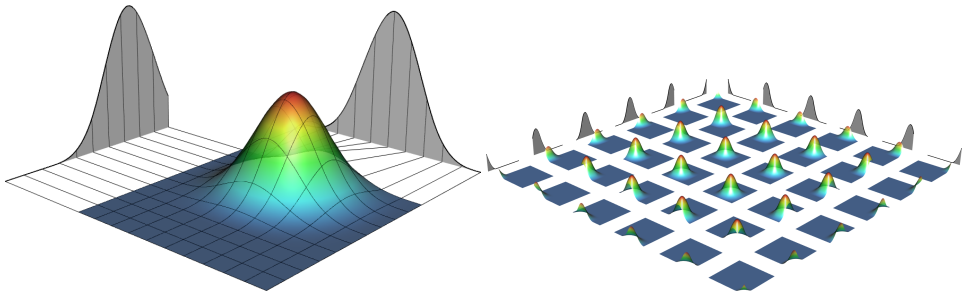


Figure 4.3. A graphical illustration of b-spline basis functions in the two-dimensional case.

The graphics in Figure 4.3 try to illustrate what is going on in terms of the basis functions.

In terms of a matrix formulation, a b-spline representation for a flexible function of x_1 is $y = B_1\alpha_1 + \varepsilon$, where y now represents the vector of observed responses, x_1 represents a vector of observed covariate values, B_1 is a matrix with columns which evaluate each basis function at all the values of x_1, and ε represents a vector of errors. Similarly $y = B_2\alpha_2 + \varepsilon$ is a model for a flexible regression based on x_2. A simple way of combining both covariates is as $y = B_{12}\alpha_{12} + \varepsilon$, where the columns of the matrix B_{12} are constructed from all the pairwise products of the columns of B_1 and B_2 and α_{12} is the corresponding set of parameters. In fact, the parameters in α_{12} could be annotated as α_{ij}, where i and j correspond to the rows and columns of the layout of basis functions. That notation makes it easier to consider suitable penalty functions, if we wish to employ them. A simple option for a penalty function is

$$\lambda_1 \sum_{i=2}^{p}\sum_{j=1}^{p}(\alpha_{ij} - \alpha_{i-1,j})^2 + \lambda_2 \sum_{i=1}^{p}\sum_{j=2}^{p}(\alpha_{ij} - \alpha_{i,j-1})^2.$$

This is based on first differences but clearly second differences could also be used. The first term penalises roughness down the 'columns' while the second term penalises roughness across the 'rows'.

Notice again that two penalty parameters, λ_1 and λ_2, are required - one for each covariate.

4.4 How much to smooth

One of the key questions with nonparametric models is how much smoothing to apply to the data. For exploratory work, it can often be helpful simply to experiment with different degrees of smoothing. One appealing way to do that is to specify how many *degrees of freedom* (see discussion above) you would like to have. This puts things on a natural scale.

However, in more complicated situations that can be difficult and it is helpful to have an automatic way of producing a suitable level of smoothing. There are several ways to do this, some of which are carefully tailored to particular models. Here we will outline a method called *cross-validation* which, although it has some difficulties, has the advantage that the generality of its definition allows it to be applied to quite a wide variety of settings. In the present setting, the idea is to choose the smoothing parameter h (possibly a vector) to minimise

$$\text{CV:} \quad \sum_{i=1}^{n}\{y_i - \hat{m}_{-i}(x_i)\}^2.$$

The subscript $-i$ denotes that the estimate of the smooth curve at x_i is constructed from the remainder of the data, excluding x_i. The aim then is to evaluate the level of smoothing through the extent to which each observation is predicted from the smooth curve produced by the rest of the data. The value of h which minimises the expression above should provide a suitable level

of smoothing. The linearity of smoothing operations allows the computations to be performed in a very efficient manner.

It is often convenient to use an approximation known as *generalised cross-validation* (GCV) which has the efficient computational form

$$\text{GCV:} \quad n\text{RSS}/\{\text{tr}\,\{I-S\}^2\}.$$

Altering the smoothing parameters changes the entries of S, which in turns affects the value of GCV.

The degree of smoothing can also be selected automatically by minimising a quantity based on *Akaike's information criterion*, namely

$$\text{AIC:} \quad \frac{\text{RSS}}{n} + 1 + \frac{2(\nu+1)}{(n-\nu-2)},$$

where ν denotes the degrees of freedom.

4.5 A simple additive model

It would be unrealistic to generalise this much further, by modelling additional covariates through functions of ever-increasing dimension. However, now that we have tools available to estimate smooth curves and surfaces, linear regression models can be extended to *additive models* as

$$y_i = \beta_0 + m_1(x_{1i}) + \ldots + m_p(x_{pi}) + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where the m_i are functions whose shapes are unrestricted, apart from an assumption of smoothness. This gives a very flexible set of modelling tools. To see how these models can be fitted, consider the case of only two covariates,

$$y_i = \beta_0 + m_1(x_{1i}) + m_2(x_{2i}) + \varepsilon_i, \qquad i = 1, \ldots, n,$$

A rearrangement of this as $y_i - \beta_0 - m_2(x_{2i}) = m_1(x_{1i}) + \varepsilon_i$ suggests that an estimate of component m_1 can then be obtained by smoothing the residuals of the data after fitting $\hat{m}_2$,

$$\hat{m}_1 = S_1(y - \bar{y} - \hat{m}_2)$$

and that, similarly, subsequent estimates of m_2 can be obtained as

$$\hat{m}_2 = S_2(y - \bar{y} - \hat{m}_1).$$

Repetition of these steps gives a simple form of the *backfitting* algorithm. The same idea applies when we have more than two components on the model. At each step we smooth over a particular variable using as response the y variable with the current estimates of the other components subtracted.

If a spline basis is used, then the backfitting algorithm is not required as we have a form of linear model with a penalty term. This can be written as

$$y_i = B\alpha + \varepsilon_i$$

where, as usual, the columns of the matrix B evaluate the basis functions at each observation. This time B is constructed by stacking together the columns of a basis matrix for each covariate. The model is fitted by choosing the vector of weights α to minimise

$$(y - B\alpha)^T(y - B\alpha) + \alpha^T P\alpha, \tag{4.1}$$

where the penalty matrix P is of block-diagonal form, constructed from the penalties from the individual model components, with the jth component $\lambda_j D_j^T D_j$, where D_j is a differencing matrix. This leads to the direct solution

$$\hat{\alpha} = \left(B^T B + P\right)^{-1} B^T y.$$

The terms of an additive model are unidentifiable without imposing some constraint, as a constant can be added and subtracted from the individual components without changing the resulting value. A simple solution is to require that $\sum_i m_j(x_{ij}) = 0$ for each component j.

A simple example of an additive model for the Reef data is shown below.

4.6 More general additive models

A more general formulation of an additive model is:

$$y_i = \alpha + m_1(x_{1i}) + \ldots + m_p(x_{pi}) + \varepsilon_i.$$

Further generality can be achieved by the use of a link function to create a *generalised additive model* or GAM for short. At the moment we will consider additive models, with link functions deferred to a later session, but it is convenient to use the terminology GAM in this case too. A simple extension of the steps outlined for two covariates gives a form of the *backfitting* algorithm. In order to ensure identifiability, we assume that $\sum_i m_j(x_{ji}) = 0$, for each j. At each step we smooth over a particular variable using as response the y variable with the current estimates of the other components subtracted.

The backfitting algorithm can be expressed as:

$$\hat{m}_j^{(r+1)} = S_j\left(y - \hat{\alpha}\mathbf{1} - \sum_{k<j} \hat{m}_k^{(r+1)} - \sum_{k>j} \hat{m}_k^{(r)}\right).$$

We can also express these in terms of projection matrices.

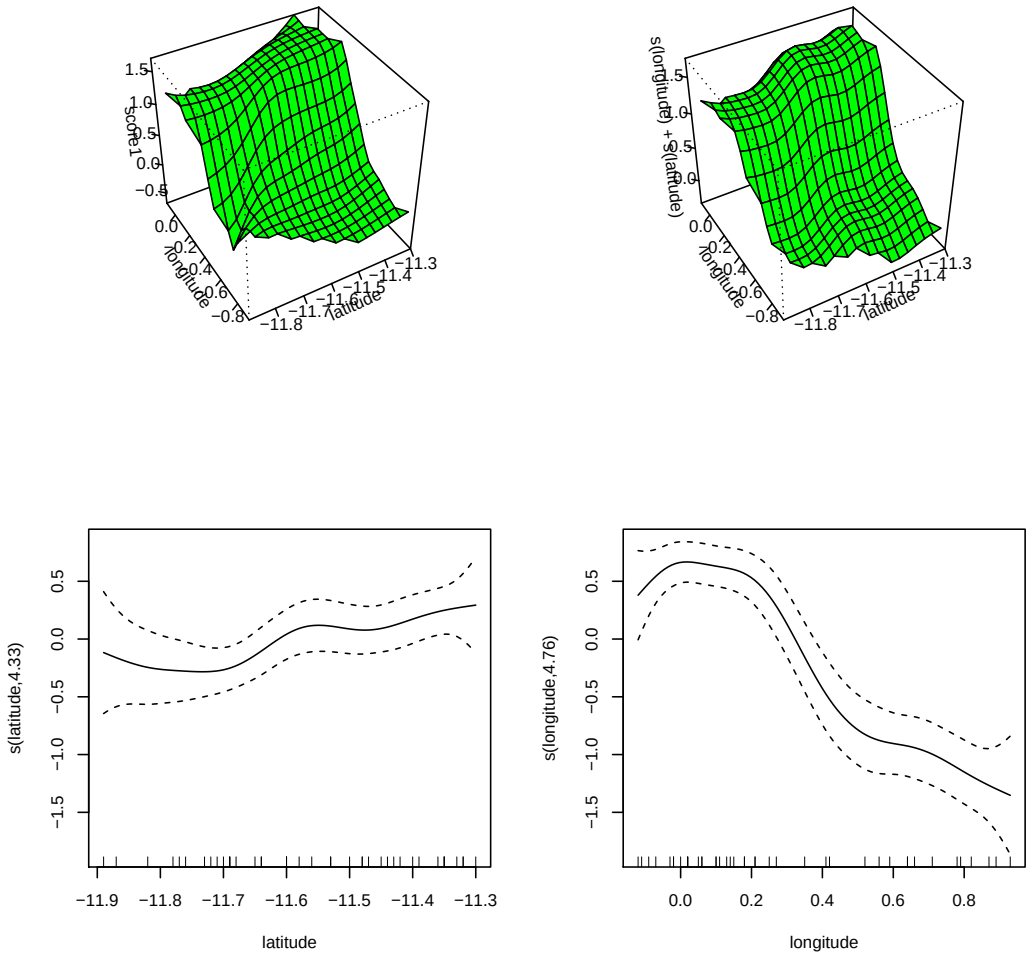


Figure 4.4. The top left hand plot shows a two-dimensional smooth estimate of the combined effects of latitude and longitude on the catch score for the Reef data. The lower plots show the estimated components from a GAM model. The top right hand plot shows the surface produced by the combination of the two GAM components.

$$P_j^{(l)} = (I_n - P_0)S_j(I_n - \sum_{k<j} P_k^{(l)} - \sum_{k>j} P_k^{(l-1)}),$$

$$\hat{y} = Py = (P_0 + \sum_{j=1}^{p} P_j)y$$

If a regression splines or p-splines model is adopted, the each of the functions $m_i(x)$ is represented by a linear expression and so the model itself remains linear. It can then be fitted by standard linear regression, incorporating a set of penalties in the p-splines case. This has the advantage of direct, rather than iterative, fitting but it has the potential disadvantage of needing to invert very large matrices if the model has many terms.

The plots below show data from a survey of dissolved oxygen (DO) in the River Clyde at a single sampling station, related to potential explanatory variables of interest. The additive terms usefully capture the underlying trends.

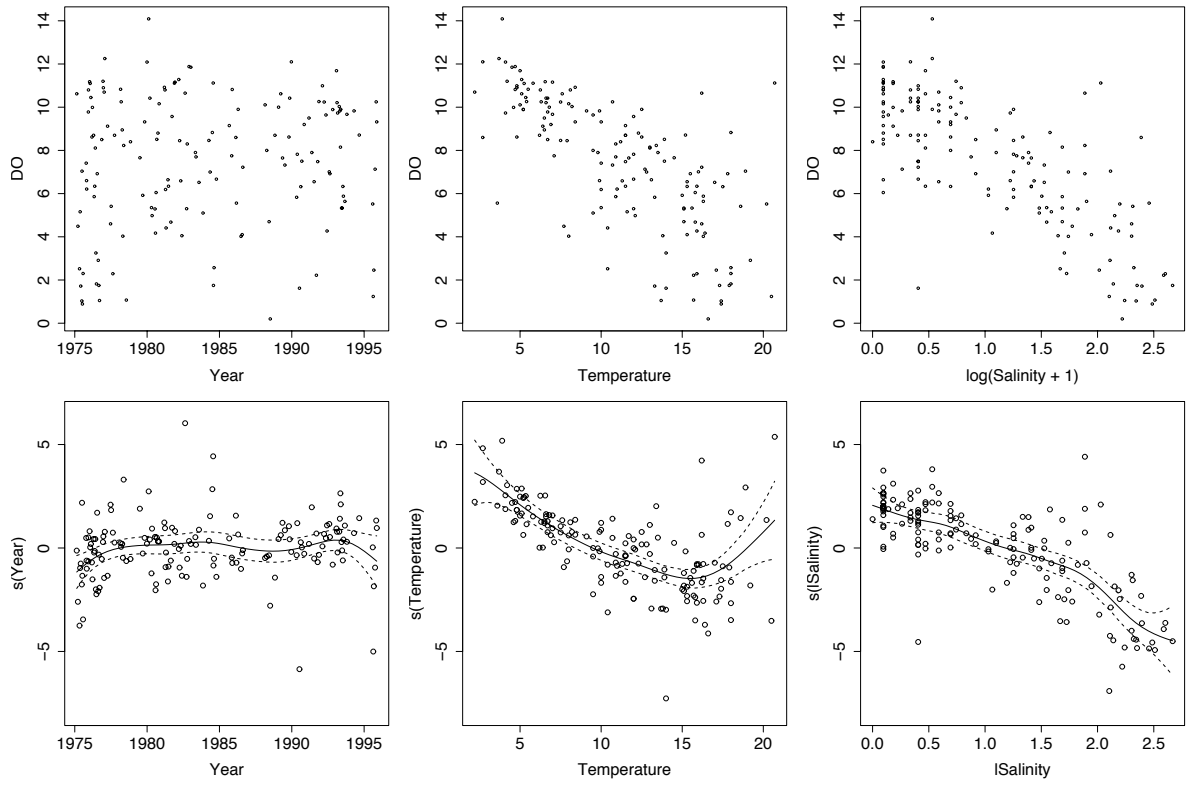


Figure 4.5. The top row of plots show DO against three covariates. The bottom row of plots show the fitted components, and partial residuals, of a GAM model.

As ever, a method of determining the level of smoothing in an additive model is required. There are several potential approaches and some of these will be discussed later. For the moment, cross-validation provides a convenient option. The `mgcv` package for `R` has efficient algorithms for identifying the optimal smoothing parameters.

4.7 Comparing additive models

While models of this type provide very flexible and visually informative descriptions of the data, it is also necessary to consider how models can be compared and inferences drawn. Hastie & Tibshirani (1990) recommend the use of residual sums-of-squares and their associated approximate degrees of freedom to provide guidance for model comparisons.

For an additive model, the residual sum-of-squares can easily be defined as

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where $\hat{y}_i$ denotes the fitted value, produced by evaluating the additive model at the observation x_i. We can write the residual sum-of-squares as

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = y^\top (I - P)^\top (I - P) y,$$

where P denotes the projection matrix discussed earlier. The approximate degrees of freedom for error can be defined as

$$\text{df} = \text{tr}\left\{ (I - P)^\top (I - P) \right\}.$$

In an obvious notation, comparisons of two models can expressed quantitatively in

$$F = \frac{(\text{RSS}_2 - \text{RSS}_1)/(\text{df}_2 - \text{df}_1)}{\text{RSS}_1/\text{df}_1},$$

by analogy with the F-statistic used to compare linear models. Unfortunately, this analogy does not extend to distributional calculations and no general expression for the distribution of this test statistic is available. However, Hastie and Tibshirani (1990, sections 3.9 and 6.8) suggest that at least some approximate guidance can be given by referring the observed nonparametric F-statistic to an F distribution with $(\text{df}_2 - \text{df}_1)$ and df_1 degrees of freedom.

There are corresponding analogies for the Wald approach to testing, using quadratic forms associated with individual terms in an additive model to assess their significance. Wood (2006) describes the details in the context of testing whether relevant spline coefficients might be 0.

The reef data provide a simple illustration of how model comparisons may be made, using the `mgcv` package. The table below indicates that the evidence for a latitude effect is not compelling.

```
## anova(model2)

   Approximate significance of smooth terms:
                  edf Ref.df      F p-value
   s(latitude)  4.329  5.284  2.131  0.0822
   s(longitude) 4.763  5.791 29.386  <2e-16
```

4.8 Further examples of additive models

4.8.1 Mackerel eggs in the Eastern Atlantic

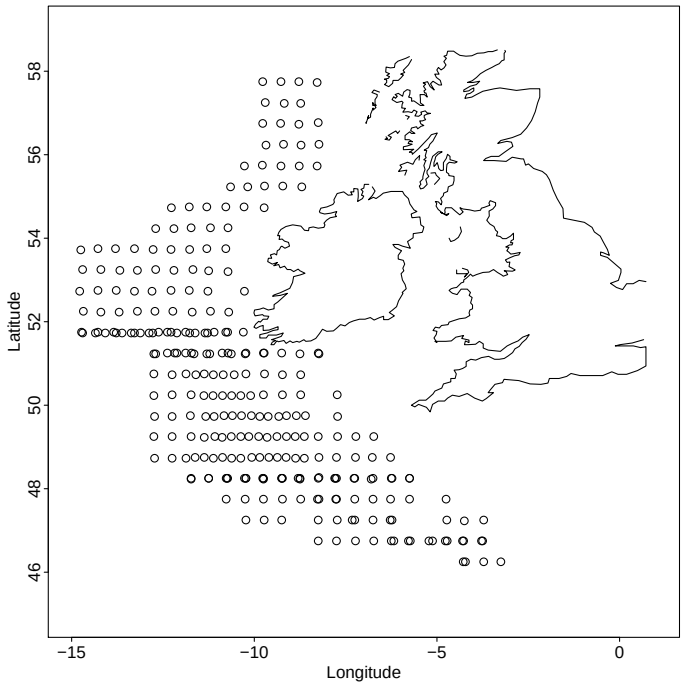


Figure 4.6. Locations of mackerel egg samples.

A further example uses data from a multi-country survey of mackerel eggs in the Eastern Atlantic. Figure 4.6 shows the locations at which samples were taken. An additive model for egg density might reasonably contain terms for depth and temperature, plus a joint term for latitude and longitude, to reflect spatial position. This leads to the model

$$y = \beta_0 + m_{12}(x_1, x_2) + m_3(x_3) + m_4(x_4) + \varepsilon,$$

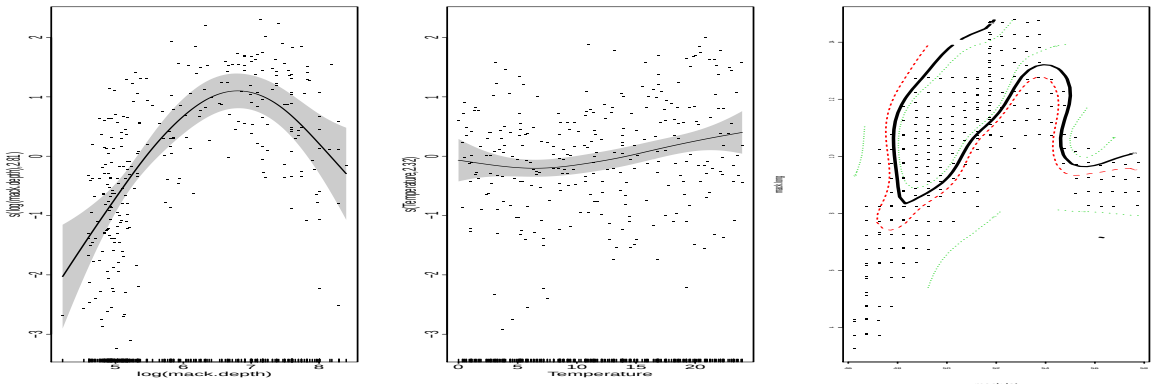
where m_{12} represents a smooth two-dimensional function of latitude (x_1) and longitude (x_2), and m_3 and m_4 represent additive terms of the usual type for depth (x_3) and temperature (x_4). Two-dimensional terms require restrictions to define the functions uniquely, as in the one-dimensional case. A simple convention is $\sum_{i=1}^{n} m_{12}(x_{1i}, x_{2i}) = 0$.

Figure 4.7 gives the details of a fitted GAM model for the mackerel data.

4.8.2 Interactions in GAM models

What does an interaction mean in a GAM model? A broad interpretation of an interaction between two covariates is that the effect of one depends on the setting of the other. For a GAM,

```
model1  <- gam(log(Density) ~ s(log(mack.depth)) + s(Temperature)
            + s(mack.lat, mack.long), data = mackerel)
par(mfrow=c(1,3), mar = c(3, 3, 1, 1), mgp = c(1.2, 0.2, 0), tcl = -0.2)
plot.gam(model1, se = TRUE, shade = TRUE, residuals = TRUE)
```




```
anova(model1)
```

```
Family: gaussian
Link function: identity

Formula:
log(Density) ~ s(log(mack.depth)) + s(Temperature) + s(mack.lat,
    mack.long)

Approximate significance of smooth terms:
                           edf Ref.df      F  p-value
s(log(mack.depth))     2.815  3.538 18.055 9.55e-12
s(Temperature)         2.316  2.904  3.872   0.0147
s(mack.lat,mack.long) 20.197 24.788  5.060 1.03e-12
```

Figure 4.7. A GAM model for the density of mackerel eggs. The right hand plot uses contours to indicate the spatial effect, with coloured contours to indicate variability.

this means that we need a smooth surface to describe the combined effects of the two covariates (just as we used for the spatial term in the mackerel data above). Two one-dimensional functions to capture the effects of the separate (marginal) covariates is no longer enough.

A model for the dissolved oxygen in the River Clyde illustrates this, expressed here in R syntax:

```
DO ~ s(lSalinity, Station) + s(Temperature, Station) + s(Year, Station)
```

This builds a model for the whole river, using data at many sampling stations. (Some care has to be taken here because of the repeated measures nature of the data. We will ignore this complication for the moment.) We might reasonably expect that the effects of salinity, temperature and year will be different at different locations on the river. The interaction terms are shown in the surface plots.

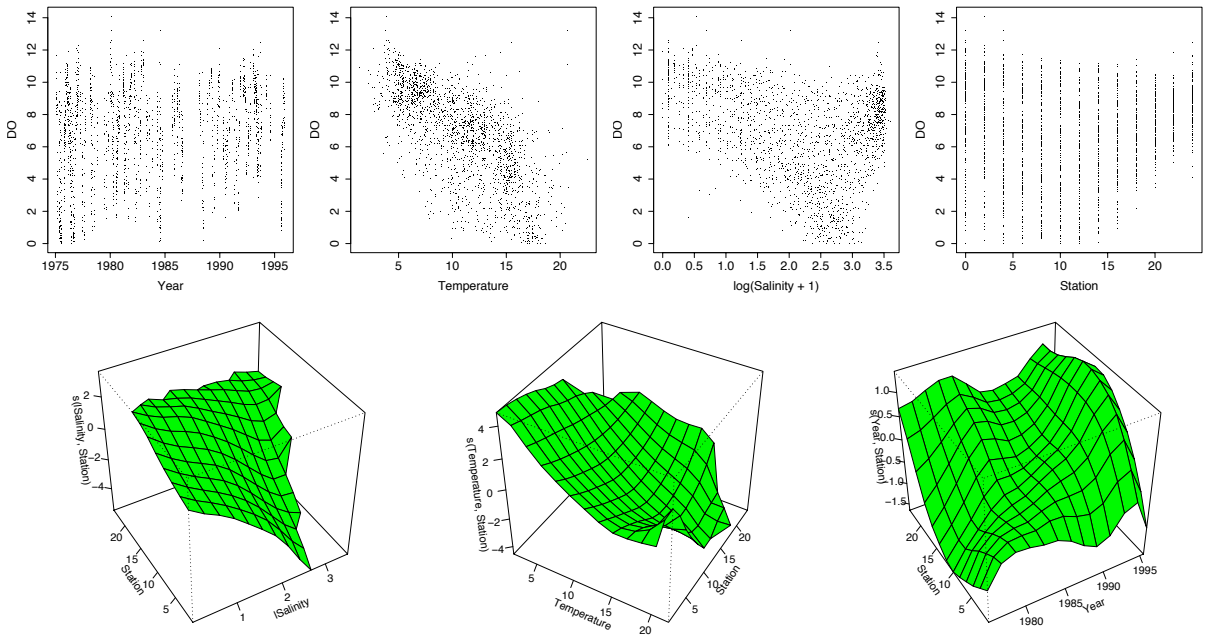


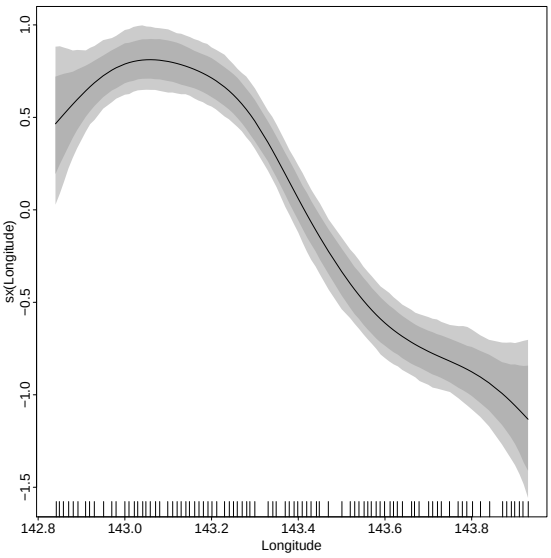
Figure 4.8. The top row of plots show DO against four covariates. The lower row of plots show interaction terms from a fitted GAM model.

4.8.3 Bayesian additive models

The penalised spline approach to fitting flexible regression curves and surfaces, and additive models, is strongly suggestive of a Bayesian approach. Expression (4.1) can be viewed as the combination of a log-likelihood (quantifying how well the model fits the data) and a prior for the parameters α (expressing correlation between neighbouring values). This can be developed into a fully Bayesian approach, including priors for the unknown hyperparameter λ. Some aspects of this will be discussed later in the course. However, for the moment we will use the BayesX package (see www.bayesx.org) to experiment with this approach. Figure 4.9 shows

two models for the Reef data which are (reassuringly) very similar to the models produced earlier.

```
library(R2BayesX)
model1 <- bayesx(Score1 ~ sx(Longitude), data = trawl)
```




```
model2 <- bayesx(Score1 ~ sx(Longitude) + sx(Latitude), data = trawl)
```

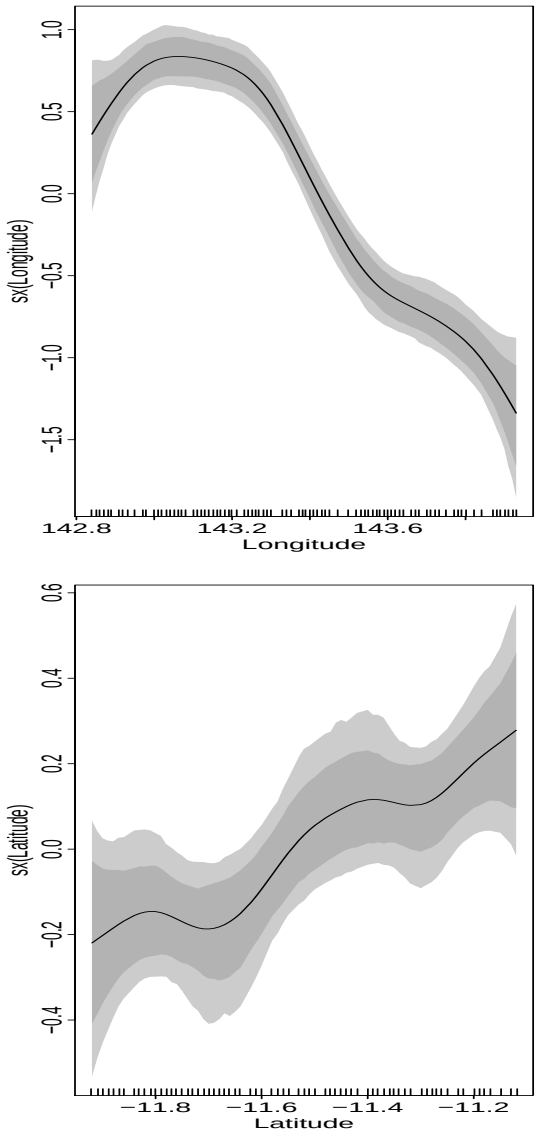


Figure 4.9. Flexible regression models for the Reef data, using a fully Bayesian approach implemented in the `BayesX` package. The upper plot shows a model for longitude alone, while the lower plots show the components of an additive model for longitude and latitude.

5

Bayesian nonparametrics and Kernel methods

In this section we will introduce two Bayesian nonparametric methods, Gaussian processes and Dirichlet processes. Gaussian processes are a Bayesian model for function estimation and Dirichlet processes are a Bayesian model for density estimation. They have in common that the object for which we want to perform inference is an infinite-dimensional object rather than a finite-dimensional vector of parameters.

5.1 Gaussian Processes

5.1.1 Bayesian Linear Model

In this section we will quickly revise the Bayesian Linear Model, i.e. we assume the linear regression model

$$y_i|\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{x}_i^\top \boldsymbol{\beta}, \sigma^2) \text{ i.i.d.}$$

or equivalently,

$$\mathbf{y}|\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \cdot \mathbf{I}) \tag{5.1}$$

Rather than using the usual normal-inverse-gamma prior jointly placed on $(\boldsymbol{\beta}, \Sigma^2)$, we will for the moment assume that σ^2 is known and just place a Gaussian prior on $\boldsymbol{\beta}$, i.e.

$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, \tau^2 \cdot \mathbf{I}). \tag{5.2}$$

We can write the p.d.f. of the posterior distribution of $\boldsymbol{\beta}$ as

$$f(\boldsymbol{\beta}|y_1,\ldots,y_n) \propto \underbrace{\left(\prod_{i=1}^{n} f(y_i|\boldsymbol{\beta})\right)}_{\text{Likelihood}} \cdot \underbrace{f(\boldsymbol{\beta})}_{\text{prior}}$$

$$= \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2}{2\sigma^2}\right)\right) \cdot \left(\frac{1}{\sqrt{2\pi\tau^2}}\right)^p \exp\left(-\frac{\sum_{j=1}^{p}\beta_j^2}{2\tau^2}\right)$$

Collecting terms, taking logs and keeping only terms involving $\boldsymbol{\beta}$ yields the log-posterior density

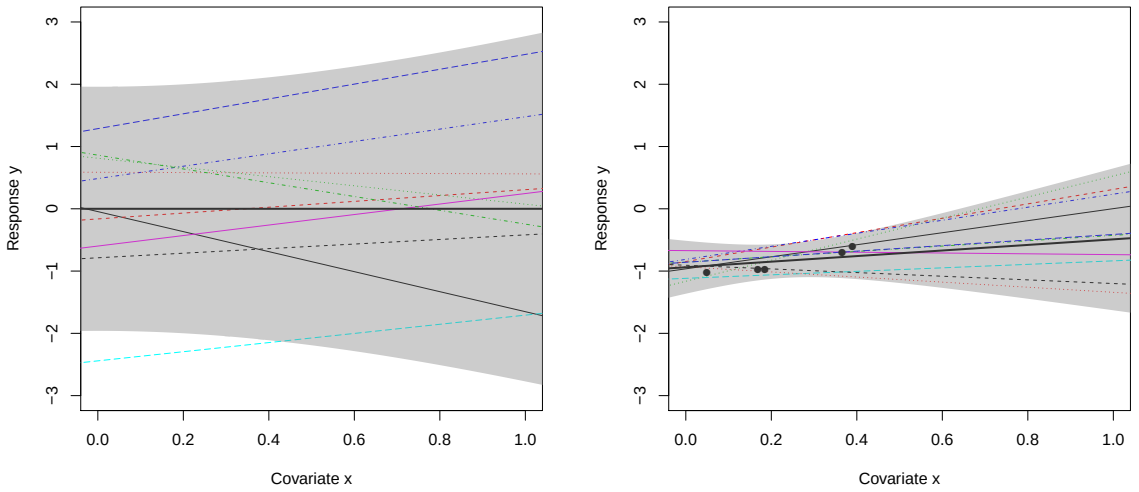
$$\log f(\boldsymbol{\beta}|y_1,\ldots,y_n) = \text{const} - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2 - \frac{1}{2\tau^2}\sum_{j=1}^{p}\beta_j^2,$$

which is, up to a multiplicative constant, the objective function used in ridge regression with $\lambda = \frac{\sigma^2}{\tau^2}$.

One can show (by completing the square) that the posterior distribution of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}|y_1,\ldots,y_n \sim \mathsf{N}\left(\left(\mathbf{X}^\top\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}, \left(\mathbf{X}^\top\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{I}\right)^{-1}\right)$$

Thus the ridge regression estimate $\hat{\boldsymbol{\beta}}^{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$ is the Bayesian maximum-a-posteriori (MAP) estimate of $\boldsymbol{\beta}$.



(a) Samples from the prior distribution.

(b) Data and samples from the posterior distribution.

Figure 5.1. Draws from the prior distribution and the posterior distribution of a Bayesian linear model. The bold line corresponds to the mean, the shaded area corresponds to pointwise 95% credible intervals.

Figure 5.1 illustrates this idea of Bayesian inference for a linear model with design matrix $\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$. Panel (a) shows ten draws from the prior distribution, whereas panel (b) shows draws from the posterior distribution given the data.

So far we have stated the model structure of the Bayesian linear model in terms of the mean, i.e. $\mathbb{E}(\mathbf{y}|\boldsymbol{\beta}) = \mathbf{X}\boldsymbol{\beta}$. We will now explain that we can rewrite this, such that exactly the same model structure can be expressed in terms of covariances.

To achieve this, we try to re-express the model without reference to $\boldsymbol{\beta}$, i.e. we suppose that we are not interested in the regression coefficients $\boldsymbol{\beta}$, but only in predictions for future observations. Essentially we have to combine (5.1) and (5.2) to find the marginal distribution of $\mathbf{y}$. The theory of the normal distribution tells us that the marginal distribution of $\mathbf{y}$ is also a normal distribution, so we only need to find its expected values and its variance.

$$\begin{aligned}\mathbb{E}(\mathbf{y}) &= \mathbb{E}_{\boldsymbol{\beta}}\left(\mathbb{E}_{\mathbf{y}|\boldsymbol{\beta}}(\mathbf{y})\right)\mathbf{0}\\ \mathbf{Var}(\mathbf{y}) &= \mathbf{Var}_{\boldsymbol{\beta}}\left(\mathbb{E}_{\mathbf{y}|\boldsymbol{\beta}}(\mathbf{y})\right) + \mathbb{E}_{\boldsymbol{\beta}}\left(\mathbf{Var}_{\mathbf{y}|\boldsymbol{\beta}}(\mathbf{y})\right)\\ &= \tau^2\mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\end{aligned}$$

thus the Bayesian linear model is equivalent to assuming that

$$\mathbf{y} \sim \mathsf{N}\left(\mathbf{0}, \tau^2\mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\right). \tag{5.3}$$

What we have achieved by eliminating $\boldsymbol{\beta}$[1] is to move the structural assumption of the Bayesian linear model regression from the mean into the covariance of the Gaussian distribution. The key idea which allows us to generalise the Bayesian linear model to Gaussian processes is that we can replace $\mathbf{X}\mathbf{X}^\top$ by a more general matrix.

5.1.2 Definition of a Gaussian process

We start by defining what a Gaussian process actually is. We define a Gaussian process to be a collection of random variables $y_i = y(\mathbf{x}_i)$ $(i = 1, 2, 3, \ldots)$ depending on covariates $\mathbf{x}_i$ such that any *finite* subset of random variables $\mathbf{y} = (y_1, \ldots, y_n) = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))$ has a multivariate normal distribution. In geostatistics, this model is known as a kriging model[2].

We assume that we can only make observations subject to noise and assume (without any loss of generality) a mean of 0, i.e.

$$\mathbf{y} \sim \mathsf{N}\left(\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}\right). \tag{5.4}$$

with

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}. \tag{5.5}$$

If we write $y_i = m_i + \varepsilon_i$ with $m_i = m(\mathbf{x}_i)$ and $\varepsilon_i \sim \mathsf{N}(0, \sigma^2)$ then (5.4) is equivalent to

$$\mathbf{y}|\mathbf{m} \sim \mathsf{N}(\mathbf{m}, \sigma^2\mathbf{I}) \qquad\qquad \mathbf{m} \sim \mathsf{N}(\mathbf{0}, \mathbf{K}).$$

i.e. $\mathrm{Cov}(m_i, m_j) = k(\mathbf{x}_i, \mathbf{x}_j)$

[1] To be mathematically more precise, we have integrated out $\boldsymbol{\beta}$.

[2] named after Daniel Gerhardus Krige, a South African mining engineer and professor at the University of the Witwatersrand, who first suggested kriging to model mineral deposits.

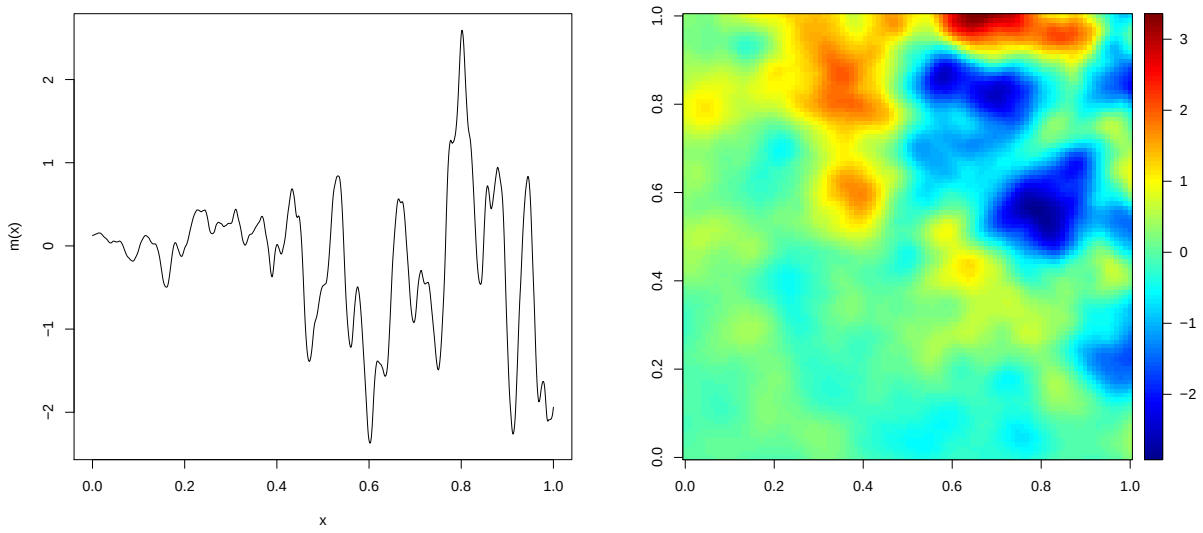
The function $k(\cdot, \cdot)$ is called *covariance function* or *kernel function*. We are free to choose any $k(\cdot, \cdot)$ as long as it is symmetric in its arguments and the matrix $\mathbf{K}$ from (5.5) is positive semi-definite.

Note that the Bayesian linear model (5.3) is a special case of a Gaussian process with covariance function $k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \cdot \mathbf{x}_i^\top \mathbf{x}_j$.

We often make the assumption that the Gaussian process is *stationary*, which is the case if and only if

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i - \mathbf{x}_j).$$

Figure 5.2 shows a draw from a non-stationary Gaussian process.



(a) The variance of $f(\cdot)$ is smaller to the left than to the right. (b) The variance of $f(\cdot)$ is smaller at the bottom-left.

Figure 5.2. Draw from a one-dimensional and two-dimensional non-stationary Gaussian process.

An additional simplifying assumption is that the process is *isotropic*, which is the case if and only if

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\|\mathbf{x}_i - \mathbf{x}_j\|),$$

i.e. only distance, but not direction matters. For the remainder we will assume that the Gaussian process is stationary and isotropic. Figure 5.3 shows a draw from a non-isotropic Gaussian process.

Furthermore, a process is called separable if

$$k(\mathbf{x}_i, \mathbf{x}_j) = k_1(x_{i1} - x_{j1}) \cdot k_2(x_{i2} - x_{j2}) \cdots k_p(x_{p1} - x_{p2}).$$

If the covariance function is separable and the data is observed on a regular grid then we can write the covariance matrix $\mathbf{K}$ of the process as a Kronecker product

$$\mathbf{K} = \mathbf{K}_1 \otimes \mathbf{K}_2 \otimes \ldots \otimes \mathbf{K}_m,$$

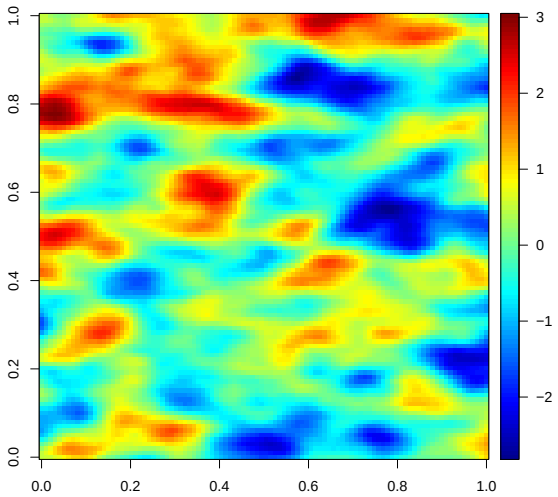


Figure 5.3. Draw from a non-isotropic two-dimensional Gaussian process. The variability in the horizontal direction is less than the one in the vertical direction.

where $\mathbf{K}_m$ is the covariance matrix constructed using the unique values of the m-th block of covariate only. In this case one can evaluate the posterior distribution without ever having to compute $\mathbf{K}$, which is a rather large matrix. The matrices $\mathbf{K}_j$ are of much smaller dimensions allowing for very efficient computations.

The idea of separability can also be used to define a covariance function by multiplying different covariance functions acting on separate sub-vectors of $\mathbf{x}_i$. Separability is often assumed in spatio-temporal models, where data is observed over time in space. In this case $\mathbf{x}_i = (s_{i1}, s_{i2}, t_i) = (\mathbf{s}_i, t_i)$, the covariates consist of the spatial coordinates $\mathbf{s}_i = (s_{i1}, s_{i2})$ and time t_i. In such models often makes the separability assumption that

$$k((\mathbf{s}_i, t_i), (\mathbf{s}_j, t_j)) = k_1(\mathbf{s}_i, \mathbf{s}_j)k_2(t_i, t_j)$$

with $k_1(\cdot, \cdot)$ being a covariance function for space and $k_2(\cdot, \cdot)$ being a covariance function for time.

We will discuss different choices of $k(\cdot, \cdot)$ later on in section 5.1.4. In geostatistics it is quite common to use a different parametrisation and work with the so-called *(semi-)variogram*

$$\gamma(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\text{Var}\,(m_i - m_j) = \frac{1}{2}(k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j)) - k(\mathbf{x}_i, \mathbf{x}_j)$$

instead of the covariance function. There is a one-to-one mapping between the two, so you can either work with the (semi-)variogram or the covariance function.

5.1.3 Predictions for Gaussian processes

Conditionals of Gaussian distributions

Assume that

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}\right)$$

Then the conditional distribution of $\mathbf{y}_2$ given $\mathbf{y}_1$ is

$$\mathbf{y}_2|\mathbf{y}_1 \sim \mathsf{N}\left(\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{y}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right)$$

We can compute predictions for a new observation with covariates $\mathbf{x}_0$ by looking at the joint distribution

$$\begin{pmatrix} \mathbf{y} \\ y_0 \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{k}_0 \\ \mathbf{k}_0^\top & k_{00} + \sigma^2 \end{pmatrix}\right),$$

where $\mathbf{K}$ is as defined in the preceding section, $\mathbf{k}_0 = (k(\mathbf{x}_0, \mathbf{x}_1), \ldots, k(\mathbf{x}_0, \mathbf{x}_n))$ is the covariance between the training data and the test case and $k_{00} = k(\mathbf{x}_0, \mathbf{x}_0)$. Then using the formula for the conditional distribution of a Gaussian we obtain

$$y_0|\mathbf{y} \sim \mathsf{N}\left(\mathbf{k}_0^\top \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}, \left(k_{00} - \mathbf{k}_0^\top \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_0\right) + \sigma^2\right)$$

The mean of the posterior distribution of y_0 can be shown to the best linear unbiased predictor (BLUP). The formula above gives the variance to be used for a prediction interval for a new observation. If we want to get the variance for a confidence interval for its mean we have to omit the "+σ^2" term accounting for the error on the unseen data, i.e. the variance of the predicted mean is $\left(k_{00} - \mathbf{k}_0^\top \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_0\right)$.

Figure 5.4 shows five draws each from the prior distribution (panel (a)) and the posterior distribution (panel (b)) from a simple Gaussian process fitted to data.

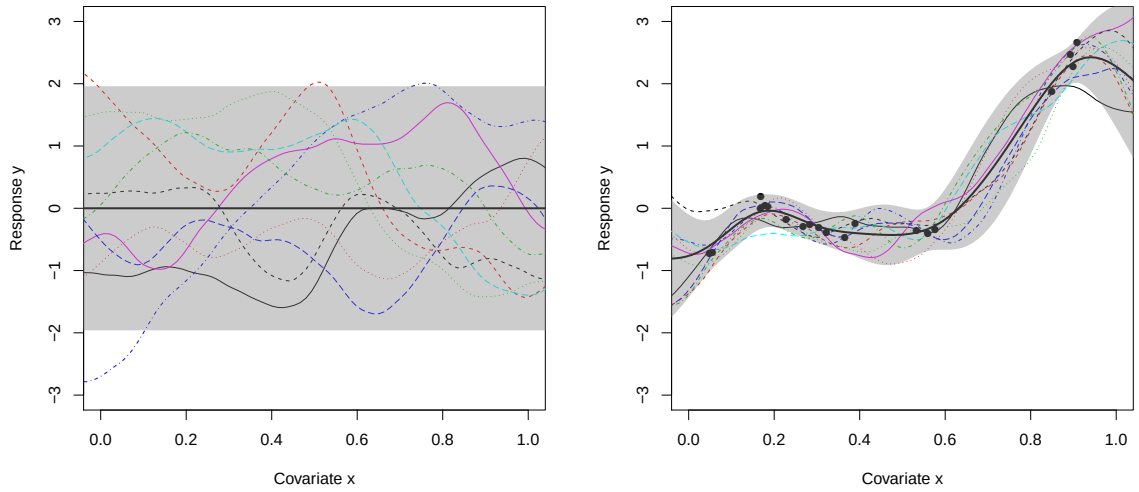
5.1.4 Covariance functions (kernel functions)

Squared exponential (SE) The squared exponential (or, Gaussian) covariance function is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \cdot \exp(-\rho\|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

The squared exponential covariance function generates very smooth processes: their paths are infinitely differentiable, which is often unrealistically smooth.

Exponential covariance function – Ornstein-Uhlenbeck (OU) process The exponential covariance function is defined as



(a) Samples from the prior distribution.

(b) Samples from the posterior distribution.

Figure 5.4. Draws from the prior distribution and the posterior distribution of a simple Gaussian process (Matérn covariance with $\kappa = 2.5$). The bold line corresponds to the mean, the shaded area corresponds to pointwise 95% credible intervals.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \cdot \exp(-\rho \|\mathbf{x}_i - \mathbf{x}_j\|)$$

It leads to a continuous, but not a differentiable process, which is often unrealistically rough. The OU process is the continuous equivalent of an $AR(1)$ process.

γ-exponential One can generalise the above two covariance functions by considering

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \cdot \exp(-\rho \|\mathbf{x}_i - \mathbf{x}_j\|^\gamma)$$

with $0 < \gamma \leq 2$, which allows choosing any model between the rough OU process and the squared exponential. However it is less flexible than the Mateérn class.

Matérn class The Matérn covariance function[3] is more flexible than the γ-exponential covariance function, however also much more complex.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \cdot \frac{1}{\Gamma(\kappa) 2^{\kappa-1}} (2\sqrt{\kappa}\rho \|\mathbf{x}_i - \mathbf{x}_j\|)^\kappa K_\kappa \left(2\sqrt{\kappa}\rho \|\mathbf{x}_i - \mathbf{x}_j\|\right),$$

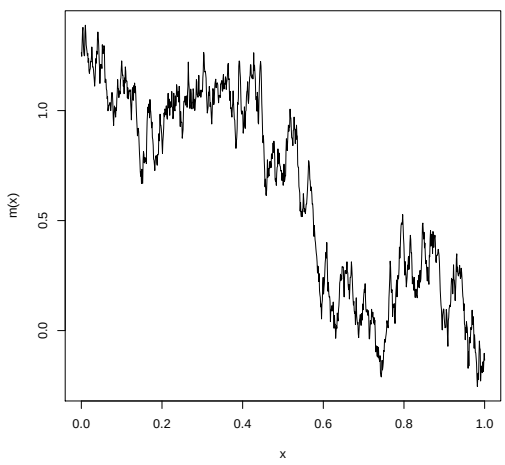
where $K_\kappa(\cdot)$ is the modified Bessel function of the second kind. Special cases of the Matérn covariance function are the OU process ($\kappa = \frac{1}{2}$) and the squared exponential ($\kappa \to +\infty$). Figure 5.5 shows functions drawn the from Matérn class for different values of κ.

For all of the above covariance functions, the parameter ρ controls how fast the correlation decays. The larger ρ the quicker the decay of the correlation. The parameter τ^2 controls the prior variance of the signal. All of the above covariance functions are stationary and isotropic, as all are based only on $\|\mathbf{x}_i - \mathbf{x}_j\|$.

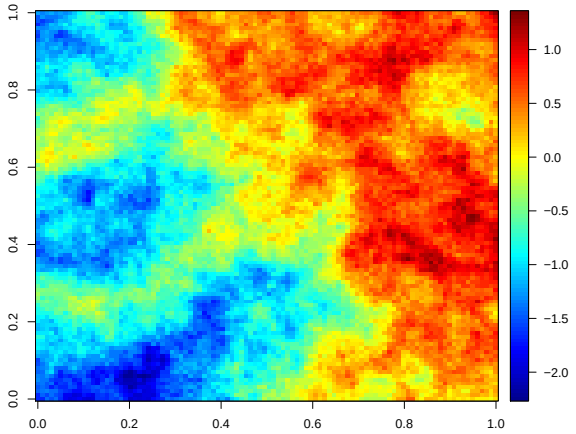
5.1.5 Estimation of hyperparameters

We have so far assumed that the hyperparameters (σ^2 and parameters of the kernel function) are known. In practice however, these need to be estimated from the data. This is best done

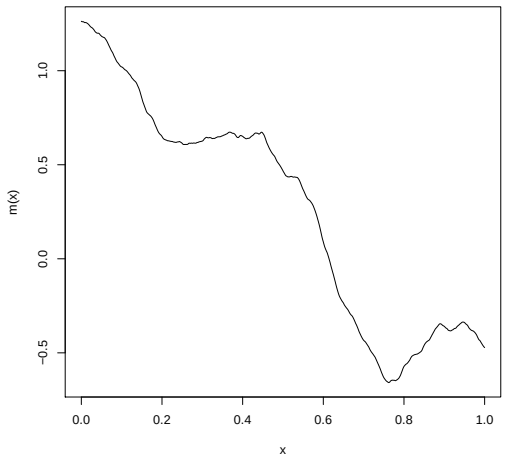
[3] named after Bertil Matén, a Swedish statistician.



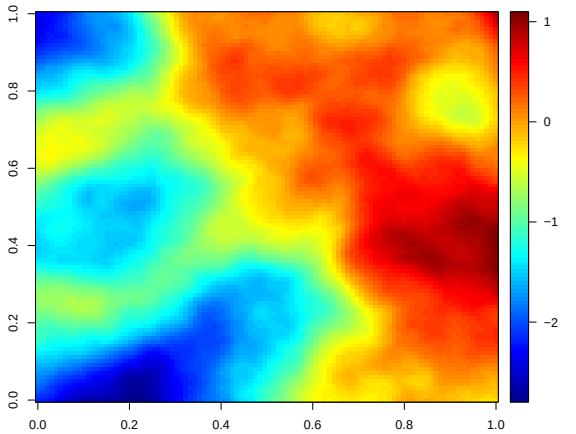
(a) $\kappa = 0.5$ (OU process)



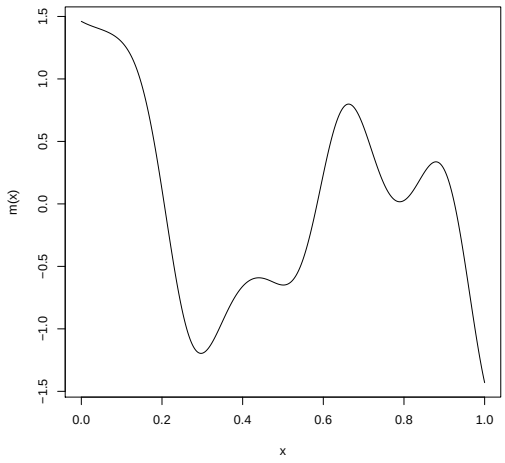
(b) $\kappa = 0.5$ (OU process)



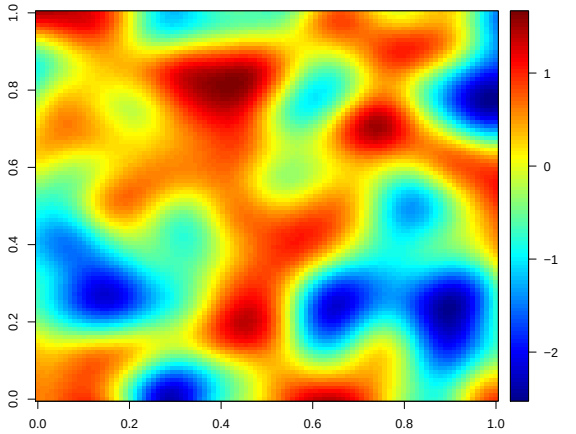
(c) $\kappa = 1.5$



(d) $\kappa = 1.5$



(e) $\kappa = +\infty$ (SE)



(f) $\kappa = +\infty$ (SE)

Figure 5.5. Samples drawn from the prior distribution of a Gaussian process with a Matérn covariance function for of $\kappa \in \{0.5, 1.5, +\infty\}$. The parameter ρ was chosen so that the covariance at lag $\frac{1}{2}$ is the same for all plots.

using the marginal log-density of $\mathbf{y}$,

$$\log f(\mathbf{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\det(\mathbf{K}+\sigma\mathbf{I}) - \frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y}$$

We could use an empirical Bayes strategy (sometime also referred to as maximum-likelihood) and maximise the density with respect to the hyperparameters.

However, a Gaussian process can use many hyperparameters and there is often little information in the data about the hyperparameters. This is especially true for the parameter κ of the Matérn covariance function. Full Bayesian models thus typically fare better as they take into account the uncertainty about the values of the hyperparameters. However, with the possible exception of σ^2, none of the hyperparameters can be integrated out in closed form, thus one has to resort to either using a discrete grid or sampling techniques such as Markov Chain Monte Carlo (MCMC).

5.1.6 Gaussian Processes in R

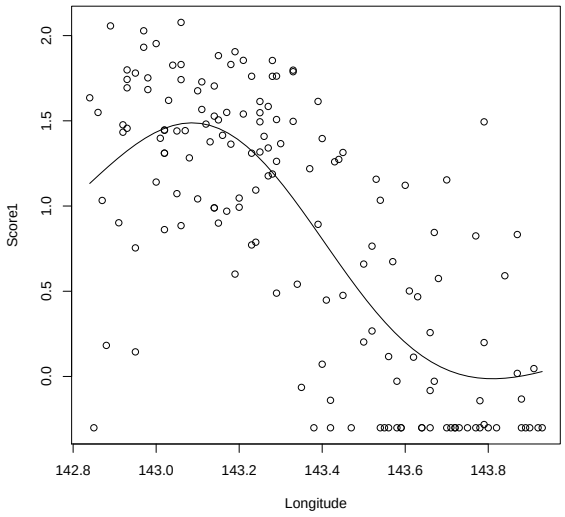
Gaussian processes (with maximum-likelihood estimation of the hyperparameters) can fit using the packages `GPfit` or `mlegp`. The example below uses the latter.

We fit a GP to the Great Barrier Reef data, initially using one covariate only.

```
fit <- mlegp(trawl$Longitude , trawl$Score1)

   ## reps detected - nugget will be estimated
   ##
   ## ========== FITTING GP # 1 ==============================
   ## intial_scaled nugget is 0.259148
   ## running simplex # 1...
   ## ...done
   ## ...simplex #1 complete, loglike = -121.200082 (convergence)
   ## running simplex # 2...
   ## ...done
   ## ...simplex #2 complete, loglike = -121.200082 (convergence)
   ## running simplex # 3...
   ## ...done
   ## ...simplex #3 complete, loglike = -121.200082 (convergence)
   ## running simplex # 4...
   ## ...done
   ## ...simplex #4 complete, loglike = -121.200082 (convergence)
   ## running simplex # 5...
   ## ...done
   ## ...simplex #5 complete, loglike = -121.200082 (convergence)
   ##
   ## using L-BFGS method from simplex #4...
   ##  iteration: 1,loglike = -121.200082
   ## ...L-BFGS method complete
   ##
   ## Maximum likelihood estimates found, log like =  -121.200082
   ## addNuggets...
   ## creating gp object......done
```

```
newdata <- data.frame(Longitude = seq(min(trawl$Longitude), max(trawl$Longitude), len=50))
predictions <- predict(fit, newdata)
plot(Score1~Longitude, data=trawl)
lines(newdata$Longitude, predictions)
```



5.2 Excursion: Kernel methods in more general

5.2.1 An alternative derivation of Gaussian processes

In this section we will look again at the derivation of Gaussian processes, more specifically at the transition from equation (5.3) to equation (5.4). We start with the Bayesian linear model, which we showed to be equivalent to

$$\mathbf{y} \sim \mathsf{N}\left(\mathbf{0}, \tau^2 \mathbf{X}\mathbf{X}^\top + \sigma^2 \mathbf{I}\right).$$

The matrix $\mathbf{X}\mathbf{X}^\top$ is nothing other than the matrix of inner products

$$\begin{aligned}\mathbf{X}\mathbf{X}^\top &= \begin{bmatrix} \sum_{j=1}^p x_{1j}^2 & \dots & \sum_{j=1}^p x_{1j}x_{nj} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^p x_{nj}x_{1j} & \dots & \sum_{j=1}^p x_{nj}^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{x}_1^\top\mathbf{x}_1 & \dots & \mathbf{x}_1^\top\mathbf{x}_n \\ \vdots & \ddots & \vdots \\ \mathbf{x}_n^\top\mathbf{x}_1 & \dots & \mathbf{x}_n^\top\mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1\rangle & \dots & \langle \mathbf{x}_1, \mathbf{x}_n\rangle \\ \vdots & \ddots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1\rangle & \dots & \langle \mathbf{x}_n, \mathbf{x}_n\rangle \end{bmatrix}\end{aligned}$$

What we have achieved so far is that we have shown that the model (5.3) from above only depends on the covariates through inner products. Note that this is only true for the Bayesian linear model or ridge regression, but not for classical frequentist linear models. We will now exploit this to turn the Bayesian linear model into a generic non-parametric methods, namely

Gaussian processes. The advantage of this derivation is that it is in no way tied to the Bayesian linear model and can be applied to a variety of other methods.

Inner products

An inner product space is a vector space with a function $\langle \cdot, \cdot \rangle$, called inner product, which satisfies the following three properties.

(Conjugate) symmetry $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$ (in case of a real-valued inner products $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$)

Linearity $\langle \alpha \mathbf{x}_1 + \beta \mathbf{x}_2, \mathbf{y} \rangle = \alpha \langle \mathbf{x}_1, \mathbf{y} \rangle + \beta \langle \mathbf{x}_2, \mathbf{y} \rangle$.

Together with the (conjugate) symmetry, linearity in the first argument implies (conjugate) symmetry in the second argument.

Positive-definiteness $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ with equality if and only if $\mathbf{x} = \mathbf{0}$.

One can show that then $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is a norm and thus $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{\langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle}$ is a distance. Examples of inner products are:

- In the vector space of $\mathbb{R}^p$ the classical inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^p x_i y_i$ satisfies the above definition.
- In the vector space of $\mathbb{C}^p$ the classical inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^p x_i \bar{y}_i$ satisfies the above definition.
- In the vector space of random variables with zero mean and finite variance, the covariance $\mathrm{Cov}(X, Y)$ satisfies the properties of an inner product.

Rather than working with the data $\mathbf{x}_i$ itself, we now consider a basis expansion $\mathbf{b}(\mathbf{x}_i)$, i.e. an extended design matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}(\mathbf{x}_1)^\top \\ \vdots \\ \mathbf{b}(\mathbf{x}_n)^\top \end{bmatrix} = \begin{bmatrix} b_1(\mathbf{x}_1) & \dots & b_q(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ b_1(\mathbf{x}_n) & \dots & b_q(\mathbf{x}_n) \end{bmatrix}.$$

Now

$$\begin{aligned} \mathbf{B}\mathbf{B}^\top &= \begin{bmatrix} \sum_{j=1}^q b_j(\mathbf{x}_1)^2 & \dots & \sum_{j=1}^q b_j(\mathbf{x}_1) b_j(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^q b_j(\mathbf{x}_1) b_j(\mathbf{x}_n) & \dots & \sum_{j=1}^q b_j(\mathbf{x}_n)^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{b}(\mathbf{x}_1)^\top \mathbf{b}(\mathbf{x}_1) & \dots & \mathbf{b}(\mathbf{x}_1)^\top \mathbf{b}(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \mathbf{b}(\mathbf{x}_1)^\top \mathbf{b}(\mathbf{x}_n) & \dots & \mathbf{b}(\mathbf{x}_n)^\top \mathbf{b}(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} \langle \mathbf{b}(\mathbf{x}_1), \mathbf{b}(\mathbf{x}_1) \rangle & \dots & \langle \mathbf{b}(\mathbf{x}_1), \mathbf{b}(\mathbf{x}_n) \rangle \\ \vdots & \ddots & \vdots \\ \langle \mathbf{b}(\mathbf{x}_1), \mathbf{b}(\mathbf{x}_n) \rangle & \dots & \langle \mathbf{b}(\mathbf{x}_n), \mathbf{b}(\mathbf{x}_n) \rangle \end{bmatrix} \end{aligned}$$

If we now define $k(\mathbf{x}_i, \mathbf{x}_j) = \tau^2 \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j) \rangle$, then

$$\tau^2 \mathbf{B}\mathbf{B}^\top = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_1, \mathbf{x}_n) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Plugging this into (5.3) yields equation (5.4), which is the definition of Gaussian processes.

Note that we do not need to know much about the basis expansion $\mathbf{b}(\cdot)$, all we need to be able to do is to compute inner products $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j) \rangle$. Actually, we don't even need to specify $\mathbf{b}(\cdot)$, we can simply write down the function $k(\cdot, \cdot)$. An important result from functional analysis, called Mercer's theorem[4], guarantees that if

i. $k(\cdot, \cdot)$ is symmetric in its arguments, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$, and
ii. $k(\cdot, \cdot)$ is positive semi-definite, i.e. for any choice of $\mathbf{x}_1, \dots, \mathbf{x}_n$ the matrix

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

is positive semi-definite,

then there exists a unique basis expansion ("feature map") $\mathbf{b}(\cdot)$ such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j) \rangle$. Note that the conditions of Mercer's theorem just say that $k(\cdot, \cdot)$ needs to be a valid covariance function, so we can use the same functions as discussed in section 5.1.4.

This way of turning a linear method (such as the Bayesian linear model) into a nonlinear method is known as the kernel trick in the Machine Learning community. This trick can be applied to any linear technique, as long as it only depends on the data through inner products.

5.2.2 Support Vector Machines

In this section we will introduce Support Vector Machines, a popular Machine Learning methods, which also make use of the kernel trick.

Introduction The history of support vector machines reaches back to the 1960s. The "Generalised Portrait" algorithm, which constructs a separating hyperplane with maximal margin, was originally proposed by the Soviet mathematicians Vapnik and Chernovenkis. Over last decade, support vector machines have become an increasingly popular learning algorithm.

Though this course is mostly concerned with regression we start with support vector machines for classification and then cover the regression case.

Support vector machine are implemented in the function `svm` in the package `e1071`. There are also other package implementing SSVMs, such as `kernlab`.

Support Vector Machines for Classification The basic idea of support vector classification is to maximise the margin between the two classes. The margin is the qidth of a band around

[4] named after James Mercer FRS (1883 –1932), a British mathematician.

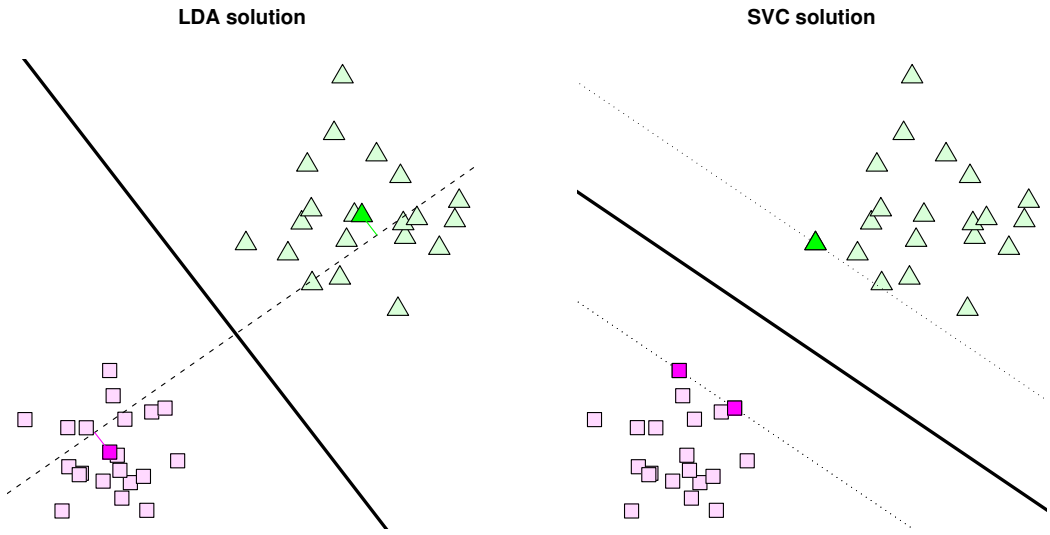


Figure 5.6. Decision boundary for linear discriminant analysis (left) and support vector classification (right) for a two-dimensional toy example: LDA maximises (after rescaling the data) the distance between the projected class means (filled symbols), whereas SVC maximises the margin between the two classes.

the decision boundary which does not contain any observations. Figure 5.6 compares this to the idea of linear discriminant analysis (LDA). As one can see from the figure, the solution depends only on the observations that lie on the margin. This makes the solution vector of the support vector machine extremely sparse.

Hard-margin support vector machines Thus we look for separating hyperplane $\{\mathbf{x} : \langle \mathbf{x}, \mathbf{w}^\circ \rangle = -b^\circ\}$ ($\|\mathbf{w}^\circ\| = 1$) which maximises the margin ρ.[5] The margin is the largest number ρ satisfying

$$\langle \mathbf{x}_i, \mathbf{w}^\circ \rangle + b^\circ \geq \rho \text{ for } y_i = 1, \qquad \langle \mathbf{x}_i, \mathbf{w}^\circ \rangle + b^\circ \leq -\rho \text{ for } y_i = -1$$

To obtain a unique solution we have to restrict the norm of $\mathbf{w}^\circ$ to 1. Mathematically it is however easier to fix the margin to 1 and look for the hyperplane with minimal $\|\mathbf{w}\|^2/2$. Graphically this corresponds to zooming the data by a factor of $1/\rho$, and more mathematically this corresponds to dividing $\mathbf{w}^\circ$ and b° by ρ. We thus want to minimise

$$\frac{1}{2}\|\mathbf{w}\|^2$$

with respect to

$$\langle \mathbf{x}_i, \mathbf{w} \rangle + b \geq 1 \text{ for } y_i = 1, \qquad \langle \mathbf{x}_i, \mathbf{w} \rangle + b \leq -1 \text{ for } y_i = -1 \tag{5.6}$$

The corresponding Lagrangian is

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b \right) - 1 \right)$$

with Lagrangian multipliers $\alpha_i \geq 0$. The dual problem is to maximise

[5] $\mathbf{w}$ is the normal vector of the separating hyperplane and governs its orientation, whereas b controls the offset of the hyperplane

$$D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{5.7}$$

over all $\alpha_i \geq 0$ with $\sum_{i=1}^n \alpha_i y_i = 0$. $\mathbf{w}$ can be obtained from the α_i via $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$. b can be computed using any observation $x_{i_{sv}}$ with $\alpha_{i_{sv}} > 0$ via $y_{i_{sv}} (\langle \mathbf{x}_{i_{sv}}, \mathbf{w} \rangle + b) = 1$.

The prediction formula for a new observation with covariate $\mathbf{x}_0$ is

$$\hat{y}_0 = \operatorname{sign} (\langle \mathbf{x}_0, \mathbf{w} \rangle + b) = \operatorname{sign} \left(\sum_{i=1}^{r} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_0 \rangle + b \right) \tag{5.8}$$

The optimisation problem (5.7) has the inequality constraint $\alpha_i \geq 0$, thus it is highly likely that many α_i will be 0, i.e. the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is likely to be very sparse. Thus the solution only depends on the observations for which $\alpha_i > 0$. These observations are called support vectors.

Soft-margin support vector machines Obviously maximising the margin in the above sense is only possible if the dataset is separable. In practice however, most datasets are not separable.[6] In this case, slack variables ξ_i must be introduced so that the constraints can be met. The value of these slack variables indicates how far the data point lies outside the margin ("error"). Figure 5.7 visualises this idea. These "errors" are considered with a cost factor $C > 0$ in the objective function, which then becomes

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i.$$

The solution is thus a trade off between maximising the margin and not having too many points dropping outside the "margin". This optimisation problem can be solved using the standard tools of semi-definite quadratic programming (e.g. interior point algorithms).

For these "soft margin" support vector machines we must change (5.6) to

$$\langle \mathbf{x}_i, \mathbf{w} \rangle + b \geq 1 - \xi_i \text{ for } y_i = 1, \qquad \langle \mathbf{x}_i, \mathbf{w} \rangle + b \leq 1 + \xi_i \text{ for } y_i = -1 \tag{5.9}$$

with $\xi_i \geq 0$. The objective function is now

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i.$$

The corresponding dual problem is once again (5.7), now to be maximised over all $0 \leq \alpha_i \leq C$ with $\sum_{i=1}^n \alpha_i y_i = 0$.

We can compute $\mathbf{w}$ and b from $\boldsymbol{\alpha}$: $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ and b can be computed using any observation $\mathbf{x}_{i_{sv}}$ with $0 < \alpha_{i_{sv}} < C$ using equations (5.9) with a "=" sign instead of the $\leq$ and $\geq$. Note that $\xi_i > 0$, iff $\alpha_i = C$.

[6] Another way of dealing with non-separable data is mapping it into a high-dimensional feature space (see below).

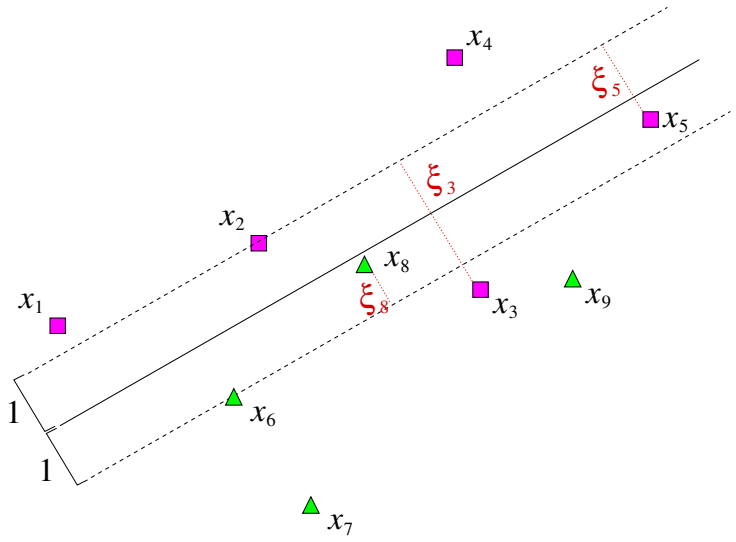


Figure 5.7. Slack variables ξ_i used in support vector classification for non-separable data

Kernelising the support vector machine So far, we have only studied generate separating hyperplanes. Many problems in real life are however *not* linearly separable. Thus we need to turn the support vector machine into a nonlinear classification method.

As we have seen in equation 5.8 the predictions of $\hat{y}_0$ and the dual (5.7) only depend on the covariates through the inner products, so we can use the kernel trick once again. Suppose that we now consider a transform $\mathbf{b}(\mathbf{x}_i)$ instead of $\mathbf{x}_i$. The decision boundary will now be linear in this feature space, and thus the decision boundary will be nonlinear in the original space.

Using $k(\mathbf{x}_i, \mathbf{x}_k) = \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j) \rangle$ the dual (5.7) becomes

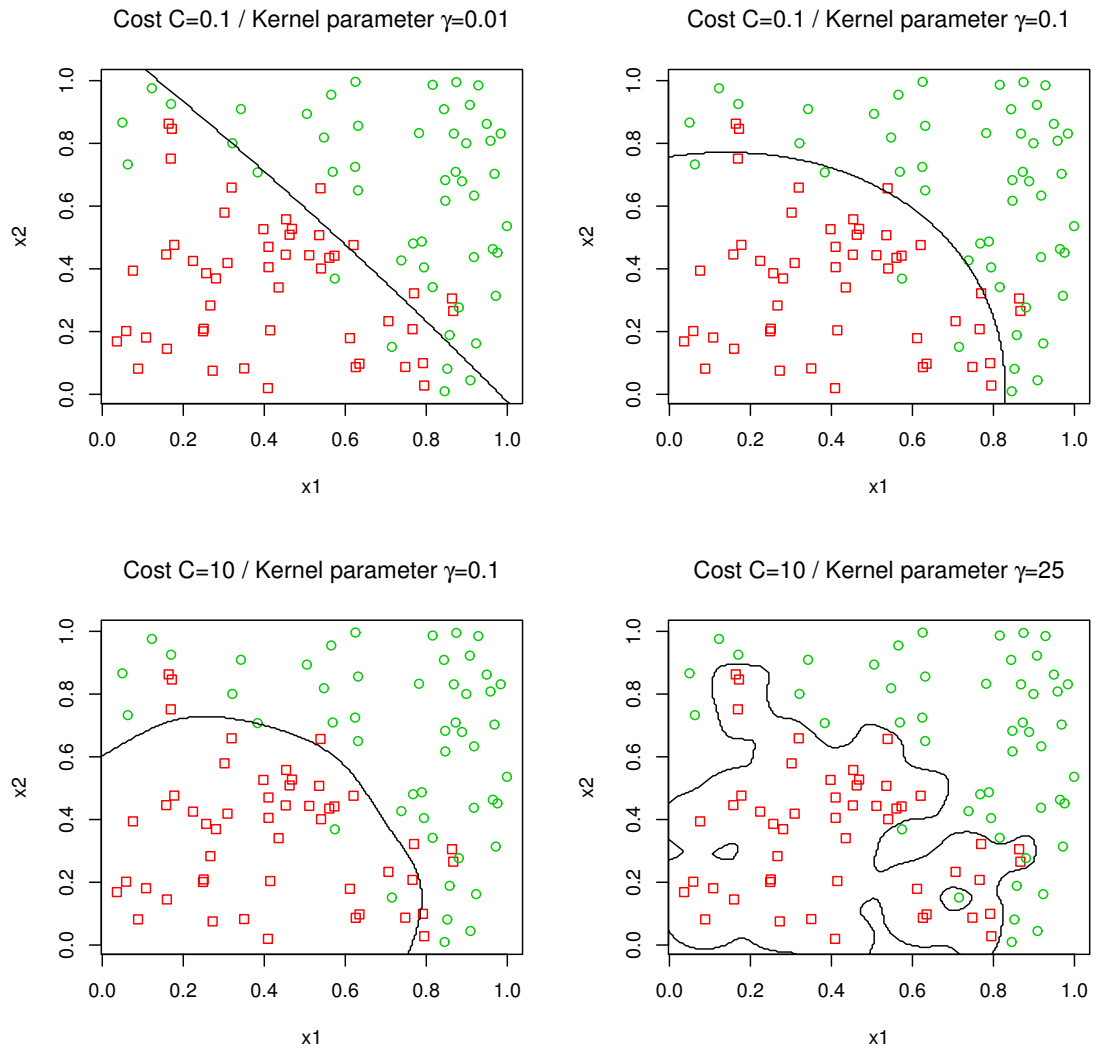
$$D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

and new predictions can be computed using

$$\hat{y}_0 = \text{sign} \left(\sum_{i=1}^{r} \alpha_i y_i k(\mathbf{x}_0, \mathbf{x}_i) + b \right)$$

Note again that the fitted function is a linear combination of the kernel functions evaluated at the observations $\mathbf{x}_i$, as the Representer Theorem requires.

Practical considerations The performance of a support vector machines depends crucially on the hyperparameters chosen: Big values of C generally yield an overfit to the data. The bigger the degree q of the polynomial or the smaller the "width" γ of the Gaussian kernel is selected, the more wigglier the decision boundary will be and the more likely the fit will result in an overfit. Figure 5.8 visualises the effect of the hyperparameters. Historically the Vapnik-Chervonenkis-(VC)-bound has been used to determine optimal values for the hyperparameters.



		Error rate	
		Training set	Test set
Cost $C = 0.1$,	kernel parameter $\gamma = 0.01$	0.15	0.15
Cost $C = 0.1$,	kernel parameter $\gamma = 0.1$	0.09	0.12
Cost $C = 10$,	kernel parameter $\gamma = 0.1$	0.11	0.19
Cost $C = 10$,	kernel parameter $\gamma = 25$	0.01	0.25

Figure 5.8. SVM fits (decision boundary) and error rates for different values of the cost C and the parameter γ of the Gaussian kernel

The VC bound is a (loose) upper bound for the actual risk. One just picks the value of the hyperparameter that leads to the lowest VC bound. The idea of minimising the VC bound is the central idea of the structural risk minimisation (SRM) principle. However, there seems to be some empirical evidence that the VC bound is *not* suitable for choosing optimal hyperparameters. Simulation studies suggest that it is best to determine the hyperparameters using either a validation set or cross-validation). Nonetheless the VC bound is an interesting theoretical concept and we will come back to the VC bound later on.

Support vector machines are not scale-invariant, so it is necessary to scale the data beforehand. However most implementations of SVM (like the one used in `R`) perform the standardisation automatically.

Support vector machines have been successfully trained classifiers with huge amounts of data (like speech or image (digit) recognition). This applies to the number of observations as well as to the number of covariates. Support vector machines are one of the most competitive methods is the area of two-class classification.

Support vector machines do not make any model assumption. This makes them a very versatile tool, but it makes assessing the uncertainty difficult to impossible: we cannot define proper confidence intervals or compute criteria like the AIC or the BIC. There are some probabilistic upper bounds like the VC bound, but these bounds are typically very loose and thus only of limited use.

The R function `tune.svm` from `e1071` can tune the hyperparameters automatically using cross-validation.

Support Vector Regression Though support vector machines are mostly used for classification, we will only cover support vector machines for regression.

Robust Statistics You probably remember from your introductory undergraduate Statistics course that the median is more robust than the mean. In other words, the median is less affected by outliers than the mean. In this section we will relate this to loss functions and use these to propose robust methods for regression.

One can show that the mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ of a sample minimises the sum of squares, i.e.

$$\sum_{i=1}^{n}(y_i - a)^2$$

is minimal for $a = \bar{y}$. Similarly one can show than the median $\tilde{y}$ minimises the sum of absolute differences, i.e.

$$\sum_{i=1}^{n}|y_i - a|$$

is minimal for $a = \tilde{y}$.

Remember that in standard linear regression we choose the regression coefficients $\boldsymbol{\beta}$ such that

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2,$$

is minimal, i.e. (standard) linear regression is using a quadratic loss function (just like the mean). One can obtain a robust version of linear regression by choosing $\boldsymbol{\beta}$ such that

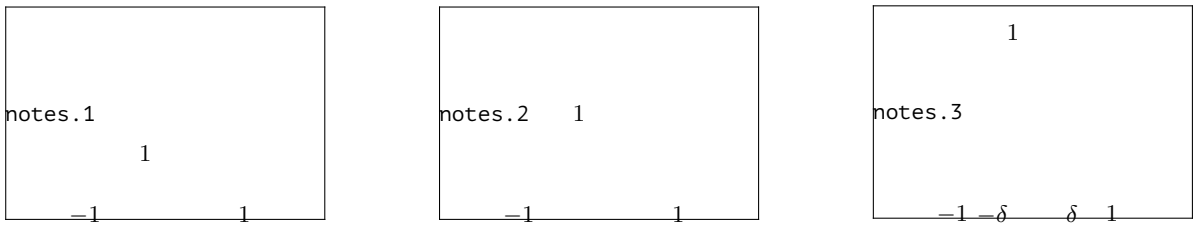
$$\sum_{i=1}^{n}|y_i - \mathbf{x}_i^\top \boldsymbol{\beta}|,$$

is minimal. This robust approach to regression yields an algorithm that can cope much better with outliers. However computing the regression coefficients is computationally more demanding and there is no "nice" theory for tests and confidence / prediction intervals.

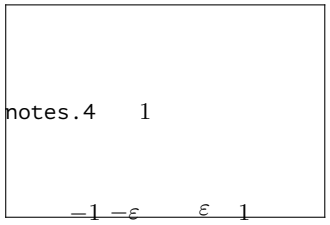
A compromise between two loss functions is Huber's loss function which is defined as

$$\sum_{i=1}^{n} L_\delta^H(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}) \qquad \text{where} \qquad L_\delta^H(z) = \begin{cases} \frac{z^2}{2} & \text{for } -\delta \le z \le \delta \\ \delta(|z| - \delta/2) & \text{otherwise,} \end{cases}$$

where $\delta > 0$ is a suitably chosen constant. Huber's loss function is implemented in the function `rlm` in `MASS`. Figure 5.9 (a) to (c) compares the three loss functions.



(a) Quadratic loss (b) Linear loss (c) Huber's loss



(d) ε-insensitive loss

Figure 5.9. Different loss functions for regression.

Support Vector Regression Support vector regression is yet another way of performing robust regression. All methods described in the previous section yield estimates of the regression coefficients which depend on all observations. In order to obtain a sparse solution which depends only on a small subset of the observations a modification of the above loss functions is used. This "ε-insensitive" loss function is defined as

$$L_\varepsilon(z) = (|z| - \varepsilon)_+ = \begin{cases} 0 & \text{for } -\varepsilon \le z \le \varepsilon \\ |z - \varepsilon| & \text{otherwise,} \end{cases}$$

where ε is a suitably chosen constant. Small errors (i.e. errors less than ε) will not incur any loss with this loss function, thus ε is typically chosen rather small (often as small as 10^{-3}).

As common in the support vector literature we will denote the regression coefficients by $\mathbf{w}$, rather than $\boldsymbol{\beta}$. In linear support vector regression we fit a linear function $b + \langle \mathbf{x}_i, \mathbf{w} \rangle$ to a response y_i by minimising the criterion

$$\underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{regularisation}} + \underbrace{C \sum_{i=1}^{n} L_\varepsilon \left(y_i - b - \langle \mathbf{x}_i, \mathbf{w} \rangle \right)}_{\text{training loss}}$$

Note that the objective function is almost the same as in ridge regression. The only difference is that we use the ε-insensitive loss function for the training loss rather than the quadratic loss used in ridge regression.

Before we go into the details of solving the optimisation problem we will first generalise the problem to the nonlinear case. Suppose we want to use a feature map $\mathbf{b}(\cdot)$, i.e. fit the function $b + \langle \mathbf{b}(\mathbf{x}_i), \mathbf{w} \rangle$. In this case the objective function becomes

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} L_\varepsilon \left(y_i - b - \langle \mathbf{b}(\mathbf{x}_i), \mathbf{w} \rangle \right)$$

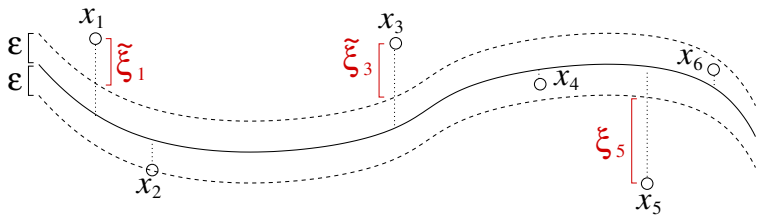


Figure 5.10. Slack variables $\tilde{\xi}_i$ and ξ_i used in support vector regression

The above optimisation can be written as an optimisation problem using slack variables $\tilde{\xi}_i$ and ξ_i. We want to minimise

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} (\tilde{\xi}_i + \xi_i)$$

subject to $y_i - (\langle \mathbf{b}(\mathbf{x}_i), \mathbf{w} \rangle + b) \leq \varepsilon + \tilde{\xi}_i$ and $(\langle \mathbf{b}(\mathbf{x}_i), \mathbf{w} \rangle + b) - y_i \leq \varepsilon + \xi_i$ with slack variables $\tilde{\xi}_i, \xi_i \geq 0$. This is illustrated in figure 5.10. The corresponding dual is

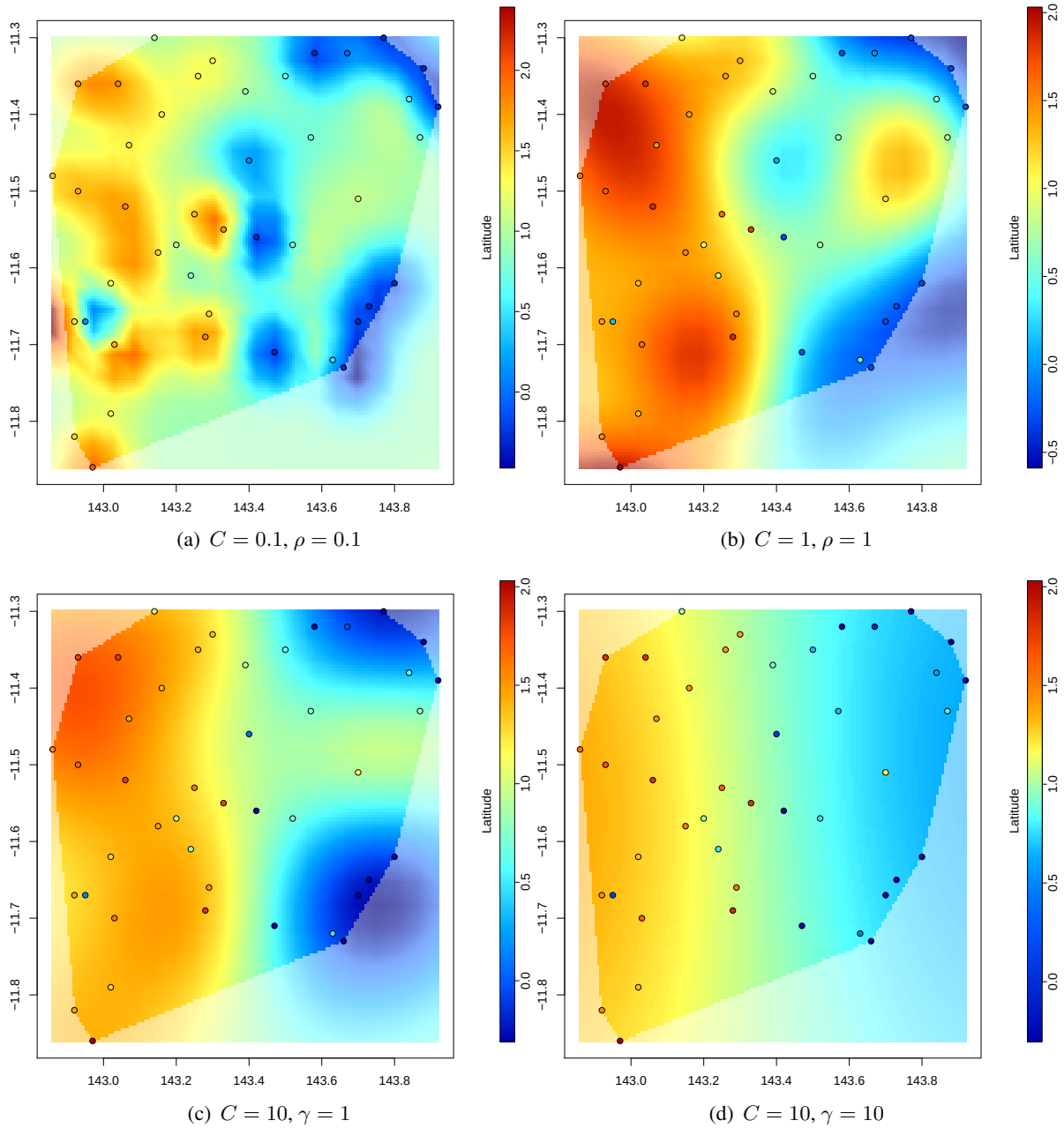
$$D(\tilde{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j} (\tilde{\alpha}_i - \alpha_i)(\tilde{\alpha}_j - \alpha_j) \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j) \rangle - \varepsilon \sum_{i=1}^{n} (\tilde{\alpha}_i + \alpha_i) + \sum_{i=1}^{n} y_i (\tilde{\alpha}_i - \alpha_i),$$

which is to be maximised over $\tilde{\alpha}_i, \alpha_i \in [0, C]$ with $\sum_{i=1}^{n} (\tilde{\alpha}_i - \alpha_i) = 0$ and $\tilde{\alpha}_i \alpha_i = 0$. The estimated regression curve can be expressed as a function of $(\tilde{\alpha}_i - \alpha_i)$ and b:

$$\hat{m}(\mathbf{x}_0) = \sum_{i=1}^{r} (\tilde{\alpha}_i - \alpha_i) \langle \mathbf{b}(\mathbf{x}_0), \mathbf{b}(\mathbf{x}_i) \rangle + b$$

Once again the optimisation problem and its solution only depend on a subset of the learning dataset (those vectors having $\tilde{\alpha}_i > 0$ or $\alpha_i > 0$) and only through inner products, i.e. we can use a kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{b}(\mathbf{x}_i), \mathbf{b}(\mathbf{x}_j)) \rangle$. In this case the estimated regression curve becomes

$$\hat{m}(\mathbf{x}_0) = \sum_{i=1}^{r} (\tilde{\alpha}_i - \alpha_i) k(\mathbf{x}_0, \mathbf{x}_i) + b.$$



(a) $C = 0.1, \rho = 0.1$ (b) $C = 1, \rho = 1$

(c) $C = 10, \gamma = 1$ (d) $C = 10, \gamma = 10$

Figure 5.11. Suppport vector machine fits to the Great Barrier Reef data for different values of the hyperparameters using a squared exponential kernel.

5.2.3 Reproducing kernel Hilbert spaces (RKHS)

We have seen that the posterior mean prediction for a Gaussian process is given by

$$\hat{y}_0 = \hat{m}(\mathbf{x}_0) = \mathbf{k}_0^\top \underbrace{\left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}}_{=\boldsymbol{\alpha}} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_0, \mathbf{x}_i)$$

The predicted class for a support vector classification machine is of the form

$$\hat{y}_0 = \text{sign}\left(\sum_{i=1}^{r} \alpha_i y_i k(\mathbf{x}_0, \mathbf{x}_i) + b\right)$$

where as the predicted value for a support vector regression machine is

$$\hat{m}(\mathbf{x}_0) = \sum_{i=1}^{r} (\tilde{\alpha}_i - \alpha_i) k(\mathbf{x}_0, \mathbf{x}_i) + b$$

What all three solutions have in common is that they are a linear combination of covariance/kernel functions. In this section we will explore in what way such solitions are optimal.

For this we start by generalising our objective. We will actually generalise it thus far that it well even cover the optimality of smoothing splines we have derived in chapter 3.

Suppose we have a loss function

$$\begin{aligned} L \;:\; & \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}, \\ & (y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) \mapsto L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) \end{aligned}$$

which is defined pointwise and which associates a loss L to a set of predictions $m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)$ with respect to the observed responses $y_1, \ldots, y_n$. One example of such a loss function is the least-squares loss

$$L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) = \sum_{i=1}^{n} (y_i - m(\mathbf{x}_i))^2,$$

but the theory we will derive is general enough to apply to any pointwise loss function.

Similarly we will consider a more general penalty $\Omega(\|\cdot\|^2)$, where $\Omega : [0, +\infty] \to \mathbb{R}$ be a strictly monotonic increasing function and $\|\cdot\|$ is a valid norm. In chapter 3 we have used

$$\begin{aligned} \langle m_1, m_2 \rangle &= \int_a^b m_1''(x) m_2''(x)\, dx \\ \|m\|^2 = \langle m, m \rangle &= \int_a^b m''(x)^2\, dx \end{aligned}$$

and $\Omega(z) = z$.

We now want to find the function $m(\cdot)$ which minimises the regularised loss ($\lambda \in \mathbb{R}^+$)

$$L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) + \lambda \cdot \Omega(\|m\|^2). \tag{5.10}$$

In chapter 3 we performed the optimisation over all functions on $[a, b]$, which are twice continuously differentiable. The space of these functions is a special type of vector space, called a representing kernel Hilbert space.[7] The Riesz representation theroem[8] tells us that

[7] A Hilbert space is an inner product space which is complete, i.e. every Cauchy sequence converges. A reproducing kernel Hilbert space is a Hilbert space of functions where the linear map δ_x which maps each function $m(\cdot)$ to the value $m(x)$ it takes at some x is continuous for every choice of x. Essentially, a reproducing kernel Hilbert space is a "reasonably well behaved" Hilbert space.

[8] named after Frigyes Riesz (1880 – 1956), a Hungarian mathematician

in a reproducing kernel Hilbert space $\mathcal{H}$ the kernel function k is the so-called representer of evaluation, i.e. for all $x \in \mathcal{X}$ and all functions $m \in \mathcal{H}$ we have that

$$m(\mathbf{x}) = \langle m, k(\cdot, x) \rangle$$

The so-called Representer Theorem now says that the minimiser of (5.10) can be written as a sum of kernel functions. In other words, the minimisation problem is only finite-dimensional as we just have to find a set of coefficients. This is an extremely powerful result, which was first dervied in he late 1970s by Kimeldorf and Wahba.

Theorem 5.1 (Representer Theorem). *Suppose $L : \mathbb{R}^n \times \mathbb{R}^b \to \mathbb{R}$ is a point-wise defined loss function and $\Omega : \mathbb{R} \to \mathbb{R}$ is a non-decreasing function. The minimiser of*

$$L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) + \lambda \cdot \Omega(\|m\|^2),$$

among all functions in the reproducing kernel Hilbert space $\mathcal{H}$ admits the representation

$$m(\mathbf{x}_0) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_0) \tag{5.11}$$

for suitable $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$.

Proof. Suppose we have a function $g \in \mathcal{H}$ which we think is optimal. We will now show that we can construct a "competitor" m of the form (5.11) which will outperform it.

i. We will first start by considering the sub-space of $\mathcal{H}$ which is spanned by $(k(x_1, \cdot), \ldots, k(x_n, \cdot))$ and define our competitor m as the projection of g into this subspace. We will also define a residual $h = g - m$, which will be orthogonal to m.
   More formally, as $k(x_i, \cdot) \in \mathcal{H}$, we can decompose the function g into a part $m \in$ span $(k(x_1, \cdot), \ldots, k(x_n, \cdot))$ and a residual part $h(\cdot) = g(\cdot) - m(\cdot)$.
   We can find projection coefficients $\alpha_1, \ldots, \alpha_n$ such that

$$m(x) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$$

   and the residual $h(\cdot)$ is orthogonal to the span (i.e. $\langle h, k(x_i, \cdot) \rangle = 0$), as it is the residual after projection.
   In other words, we have decomposed

$$g(\cdot) = \underbrace{\sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot)}_{=m(\cdot)} + h(\cdot),$$

   with $m(\cdot) \perp h(\cdot)$.

ii. We will now show that $m(\cdot)$ and $g(\cdot)$ take the same value at any of the training points $\mathbf{x}_j$, by showing that

$$g(\mathbf{x}_j) = \sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j) = m(\mathbf{x}_j).$$

We can show this by using the representer property

$$\begin{aligned} g(\mathbf{x}_j) &= \langle g, k(\cdot, \mathbf{x}_j)\rangle = \left\langle \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot) + h(\cdot), k(\cdot, \mathbf{x}_j) \right\rangle = \\ &= \sum_{i=1}^{n} \alpha_i \underbrace{\langle k(\mathbf{x}_i, \cdot), k(\cdot, \mathbf{x}_j)\rangle}_{=k(\mathbf{x}_i, \mathbf{x}_j)} + \underbrace{\langle h(\cdot), k(\cdot, \mathbf{x}_j)\rangle}_{=0} \\ &= \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) = m(\mathbf{x}_j) \end{aligned}$$

Thus both our competitor $m(\cdot)$ and $g(\cdot)$ incur the same training loss, i.e.

$$L(y_1, \ldots, y_n, g(\mathbf{x}_1), \ldots, g(\mathbf{x}_n)) = L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))$$

iii. We will finally show that $m(\cdot)$ cannot incur a larger penalty than $g(\cdot)$.

$$\Omega(\|g\|^2) = \Omega(\|m + h\|^2) \overset{m \perp h}{=} \Omega(\underbrace{\|m\|^2 + \|g\|}_{\geq \|m\|^2}) \geq \Omega(\|m\|^2)$$

iv. Putting together ii. and iii. we obtain

$$L(y_1, \ldots, y_n, g(\mathbf{x}_1), \ldots, g(\mathbf{x}_n)) + \lambda \cdot \Omega(\|g\|^2) = L(y_1, \ldots, y_n, m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)) + \lambda \cdot \Omega(\|m\|^2),$$

i.e. our competitor cannot perform worse. One can show that equality only holds if g also admits the form (5.11).

□

5.3 Dirichlet processes

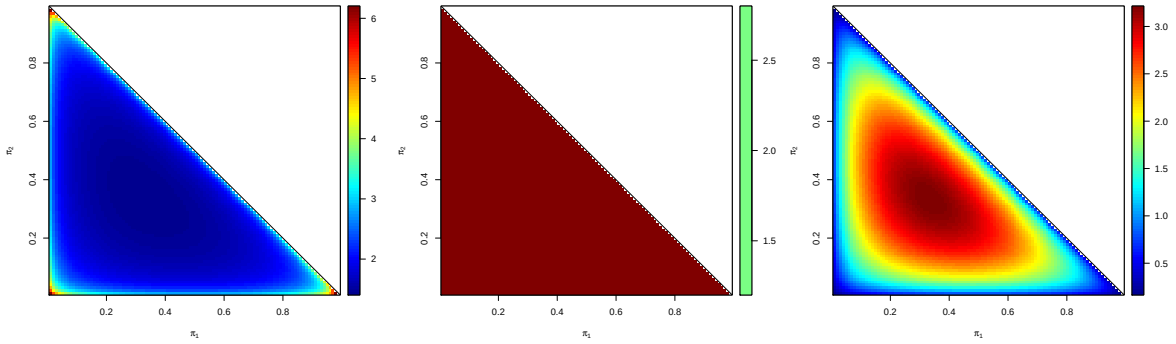
5.3.1 Bayesian mixture models

We have studied mixture models in section 2.3. One challenge was estimating the number of components. This problem can be overcome by considering a Bayesian approach. For this we need to place priors on the model parameters:

– We place a Dirichlet distribution on $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$. The Dirichlet distribution is a generalisation of the Beta distribution and generates realisations such that $\sum_{k=1}^{K} \pi_k = 1$ with $\pi_k > 0$. Its probability density function is, for $\pi_k > 0$ with $\sum_{k=1}^{K} \pi_k = 1$, given by

$$f(\pi_1, \ldots, \pi_K) = \frac{\Gamma(\delta_1 + \ldots + \delta_K)}{\Gamma(\delta_1) \cdots \Gamma(\delta_K)} \pi_1^{\delta_1 - 1} \cdots \pi_K^{\delta_K - 1}$$

In this context we typically choose all δ_k to be equal, i.e. $\delta_1 = \ldots = \delta_K = \delta$. Figure 5.12 shows the distribution for a trivariate Dirichlet distribution for $\delta \in \{0.75, 1, 1.5\}$. The figure shows the density only as a function of π_1 and π_2 as $\pi_3 = 1 - \pi_1 - \pi_2$.



(a) $\delta = 0.75$ (b) $\delta = 1$ (c) $\delta = 1.5$

Figure 5.12. Probability density function of the trivariate Dirichlet distribution for $\delta \in \{0.75, 1, 1.5\}$.

– We place a conjugate Normal-Wishart distribution on $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^{-1}$.

In order to perform posterior inference using a Gibbs sampler we introduce the same latent variables S_i (giving the cluster allocation) as we have for the EM algorithm. The Gibbs sampler iterates, just like the EM algorithm, between updating the cluster allocations and then updating the means and variances in each cluster.

Note that our model currently assumes a fixed K. If we want to infer K from the data, we have to perform model selection in an MCMC context.

There are essentially two ways of how one can perform model selection in this case. One is to run separate MCMC chains for each model (in our case each plausible number of clusters K) and then compute quantities like the deviance information criterion (DIC) for each chain. However, DIC is not suitable for mixture models. The alternative approach consists to run an MCMC algorithm that can jump between different models, i.e. we include K as a parameter in our model. This however introduces the challenge that the dimension of the parameter space changes as K is changed (we need to introduce a probability π_K and a new pair $(\boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K)$, or remove them). This requires the use of transdimensional MCMC techniques, such as reversible jump Markov Chain Monte Carlo (RJMCMC), which is rather complex and often leads to poorly mixing chains. In the next section we will consider a model which gets around this.

5.3.2 Dirichlet processes

An alternative model for estimating densities (and clustering) in a Bayesian context is given by the Dirichlet process. The key advantage of the the Dirichlet process does not require the specification of a number of clusters or transimensional MCMC if one wants to learn the number of clusters from data.

Dirichlet process prior The Dirichlet process prior is a Bayesian model for a random measure, i.e. it is a probability distribution on probability distributions.

Thse Dirichlet process (DP) can be characterised using a number of equivalent definitions. The first definition was given by Ferguson (1973). The Dirichlet process has two parameters, a so-called base measure (a probability distribution) G_0 around which the Dirichlet process is centred and a concentration parameter $\alpha > 0$. Note that G_0 is itself a probability distribution, i.e. it assigns probabilities to sets A, which are subsets of the domain of interest Ω. for example our base measure G_0 could be the $\mathsf{N}(0, 1)$ distribution in which case $\Omega = \mathbb{R}$ and $G_0((a, b)) = \int_a^b \phi(z)\, dz = \Phi(b) - \Phi(a)$.

Ferguson's definition The original definition given by Ferguson is based on considering finite partitions $(A_1, \ldots, A_q)$ of Ω.[9] G is a Dirichlet processes with parameters G_0 (base measure) and $\alpha > 0$ (concentration) if and only if

$$(G(A_1), \ldots, G(A_q)) \sim \mathsf{Dirichlet}(\alpha G_0(A_1), \ldots, \alpha G_0(A_q))$$

for every partition $(A_1, \ldots, A_q)$ of Ω.

Using the formulae for the mean and the variance of the Dirichlet distribution we can show that

$$\begin{aligned} \mathbb{E}(G(A_j)) &= \frac{\alpha G_0(A_j)}{\alpha G_0(A_1) + \ldots + \alpha G_0(A_q)} = G_0(A_j) \\ \text{Var}\,(G(A_j)) &= \frac{G_0(A_j)(1 - G_0(A_j))}{\alpha + 1}, \end{aligned}$$

i.e. on average the random measure G assigns probbability $G_0(A)$ to a set A, so G is indeed centered around G_0. α controls the spread of G: the smaller α the higher the variance of the random probabilities of a set A.

Though this is the original definition of the Dirichlet process, many of the key properties of Dirichlet processes are not obvious from this definition.

One, at first sight very surprising, feature of the Dirichlet process is that one can show that draws from the Dirichlet process exhibit ties, i.e. if we were to draw a sample from G, then we would draw some values more than once with non-zero probability. This not the case the continuous distributions.[10] So the distribution G is discrete with probability 1, even if G_0 is continuous.

Stick-breaking construction (Sethuraman,1994) An equivalent definition of the Dirichlet process is based on a "stick-breaking" construction. Instead of characterising G, we write down the stick-breaking construction in terms of drawing samples from G.

We can draw a sample $\phi_1, \phi_2, \ldots \sim G$ as follows.

[9] A finite partition of Ω is a finite set of subsets $A_j \subset \Omega$, which are disjoint ($A_j \cap A_k = \varnothing$ for $j \neq k$) and whose union spans Ω, i.e. $A_1 \cup \ldots \cup A_q = \Omega$. This way every $y \in \Omega$ is in exaclty one of the A_j's.

[10] If we draw two realisations from say the $\mathsf{N}(0, 1)$ distribution, these would be different with probability 1.

1. Draw a sequence of realisations θ_i from the base measure, i.e. $\theta_1, \theta_2, \ldots \sim G_0$.
2. Draw a sequence of weights $V_1, V_2, \ldots \sim \mathsf{Beta}(1, \alpha)$.
3. Construct a discrete probability distribution G using a stick breaking construction

$$p_G(\phi) = \begin{cases} V_1 & \text{for } \phi = \theta_1 \\ (1 - V_1)V_2 & \text{for } \phi = \theta_2 \\ (1 - V_1)(1 - V_2)V_3 & \text{for } \phi = \theta_3 \\ \ldots & \ldots \\ 0 & \text{otherwise} \end{cases}$$

4. We can draw samples $\phi_1, \phi_2, \ldots$ from p_G by drawing from the discrete distribution G from step 3.

One key advantage of the stick-breaking definition is that it makes it clear that the distribution G is discrete (albeit with random range and random probabilities), thus a sample $\phi_1, \ldots, \phi_n$ from G is likely to exhibit ties. How many ties we would expect depends on the concentration parameter α.

– If $\alpha \to 0$ all the ϕ_i are (almost surely) identical with $\phi_1 \sim G_0$.
– If $\alpha \to +\infty$ the ϕ_i are i.i.d. draws from G_0, i.e. we would observe no ties with probability 1.

In this representation we have used the Greek letters ϕ_i and θ_j to represent draws from the Dirichlet process. The ϕ_i stand from the individual draws from G, whereas the θ_j's stand for the unique values the ϕ_i's take (and which are draws from G_0).

Chinese Restaurant process (CRP) Yet another way of characterising the Dirichlet process is given by the Chinese restaurant process. One can show that samples from the Dirichlet process can be drawn recursively from G as follows:

1. Draw $\phi_1 \sim G_0$.
2. For $i = 2, 3, \ldots$
   a) Construct a frequency table of the values $\phi_1, \ldots, \phi_{i-1}$ drawn so far.

Value	θ_1	θ_2	θ_3	$\ldots$
Absolute frequency	n_1	n_2	n_3	$\ldots$

   Remember we use the θ_j's to denote the unique values of the ϕ_i's.
   b) Draw $\theta_0 \sim G_0$.
   c) Draw ϕ_i from the discrete distribution given by

$$p(\phi) = \begin{cases} \frac{\alpha}{i-1+\alpha} & \text{for } \phi = \theta_0 \\ \frac{n_1}{i-1+\alpha} & \text{for } \phi = \theta_1 \\ \frac{n_2}{i-1+\alpha} & \text{for } \phi = \theta_2 \\ \dots & \dots \\ 0 & \text{otherwise.} \end{cases}$$

We can imagine this as a sequence of customers entering a Chinese restaurant. Assume that in the restaurant all customers (representing random draws from G) sitting at a table will be served the same dish (a draw from G). We will denote by θ_j the dish on the j-th table and by ϕ_i the dish ordered by the i-th customer. If customer i sits at table j then $\phi_i = \theta_j$. Figure 5.13 illustrates this. Due to the exchangeability of the Dirichlet process[11], this view forms the basis of an efficient Gibbs sampler, referred to later on in this section.

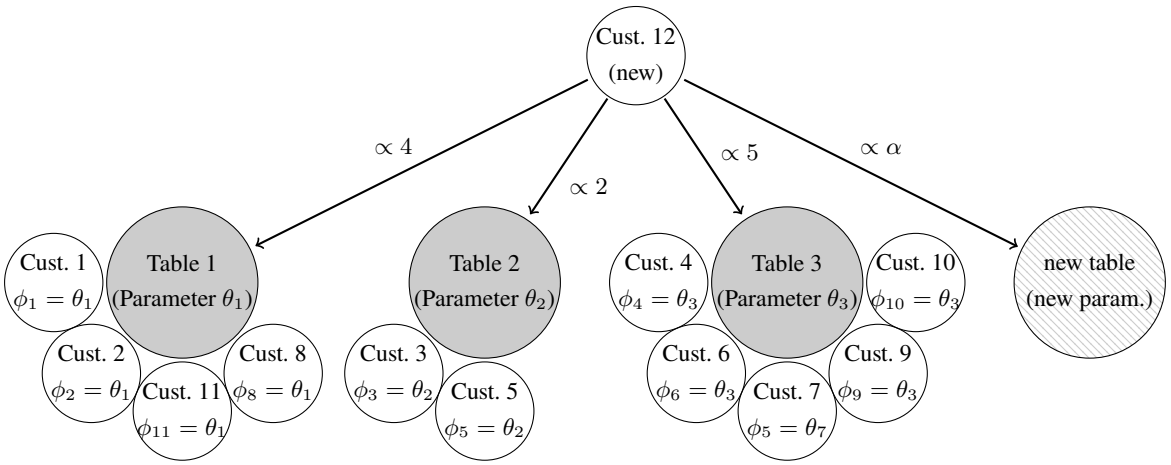


Figure 5.13. Illustration of the Chinese restaurant view of the Dirichlet process, assuming 11 customers have already chosen their table.

One equivalent view is to assume that the i-th customer entering the restaurant chooses an existing customer next to whom he wants to sit (each with probability $\frac{1}{i-1+\alpha}$). With probability $\frac{\alpha}{i-1+\alpha}$ the customer chooses to sit at a new table.

One key conclusion from this way of viewing the Dirichlet process is that it leads to a varying number of clusters (tables), depending on how the customers are seated.

The Dirichlet process has a "rich gets richer"property. Tables with many customers are more likely to attact future customers.

The CRP view of the Dirichlet process makes it very easy come up with generalisations (just think of other ways of seating customers). It is also (due to the exchangeability of the Dirichlet process) the basis for constructing Gibbs samplers.

[11] A sequence of random variable is called exchangeable if the joint distribution of any permutation of them has teh same distribution. Any i.i.d. sample is exchangeable, but the converse is not necessarily the case.

Limit of finite mixture models The finite mixture model from section 5.3.1 converges to a Dirichlet process if we let the number of components $K \to +\infty$ and $\delta \to 0$ such that $K\delta \to \alpha$.

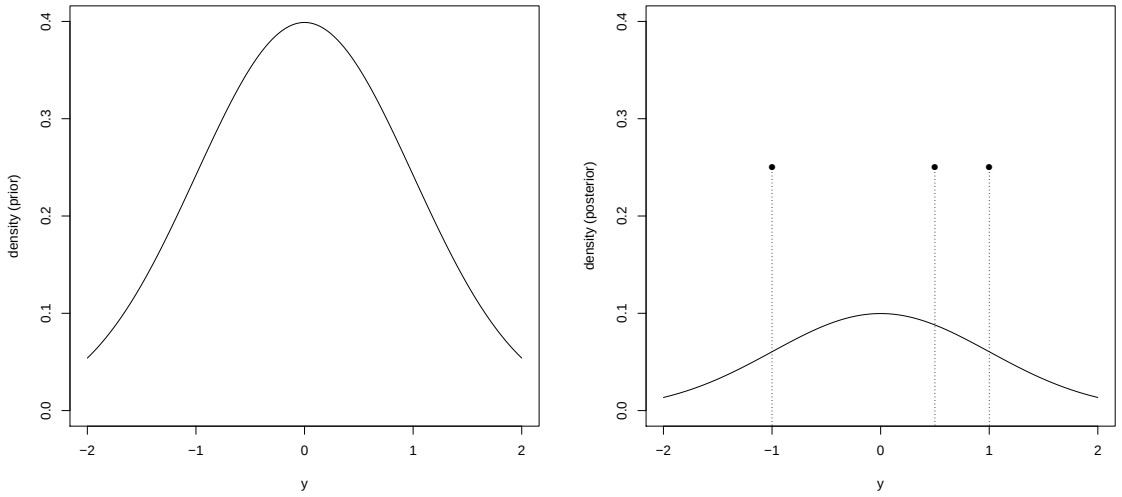
Posterior distribution So far we have just stated different (equivalent) ways of defining the Dirichlet process. We will now use it to perform Bayesian inference. Suppose we have now observed $\phi_1, \ldots, \phi_n$, what would the posterior distribution of probability distributions be?

We start by placing a Dirichlet process prior with base measure G_0 and concentration α on the distribution of distributions. One can now show that the posterior distribution is again given by a Dirichlet process, this time with concentration $\alpha + n$ and base measure

$$\frac{\alpha}{\alpha+n}G_0 + \frac{1}{\alpha+n}\sum_{i=1}^{n}\delta_{\phi_i} = \frac{\alpha}{\alpha+n}G_0 + \sum_j \frac{|\{i:\ \phi_i = \theta_j\}|}{\alpha+n}\delta_{\theta_j} =$$

where δ_{ϕ_i} denotes a distribution which takes the value ϕ_i with probability 1. In other words, the base measure of the posterior distribution is a mixture of the base distribution of the prior (with weight $\frac{\alpha}{\alpha+n}$) and the empirical distribution of the sample $\phi_1, \ldots, \phi_n$ (with weight $\frac{n}{\alpha+n}$). Figure 5.14 gives an example of this. This property seems rather unsatisfying: even if we assume that our data comes from a continuous distribution, the distribution resulting from posterior inference has discrete components.

This is why Dirichlet processes are usually not used to model the distribution of the $Y_1, \ldots, Y_n$ directly, which would correspond to using $\phi_i = Y_i$. Instead one typically assumes that Y_i comes from a distribution with parameter ϕ_i, where ϕ_i is modelled using a Dirichlet process. This setup effectively adds "jitter" such that the distributions resulting from posterior inference are continuous.



(a) Base measure of the prior distribution

(b) Base measure of the posterior distribution

Figure 5.14. Base measure of the prior and posterior distribution in a Dirichlet process model with the $\mathsf{N}(0, 1)$ distribution as prior base measure and a prior concentration parameter of 1. The observed sample was $\phi_1 = -1$, $\phi_2 = 1$ and $\phi_3 = 0.5$.

Dirichlet process mixture models In this section we will explain how a Dirichlet process can be used to model a mixture of normals.

We have seen in the preceding section that we do not want to use a Dirichlet process to directly model the $\mathbf{y}_i$. Instead we assume that the mean vector and covariance matrix for each observation come from a Dirichlet process, i.e.

$$\mathbf{y}_i|\boldsymbol{\phi}_i \sim \mathsf{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \qquad \text{using } \boldsymbol{\phi}_i = (\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

We now place a Dirichlet process prior on the distribution of the $\boldsymbol{\phi}_i$. This is likely to result in ties, i.e. more than one observation will have the same $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, which corresponds to nothing other than clusters.

Inference in this model can be performed efficiently using a Gibbs sampler (see e.g. Neal (2000)).

Figures 5.15 and 5.16 show the results when applying the method to the aircraft data from chapter 2.1.

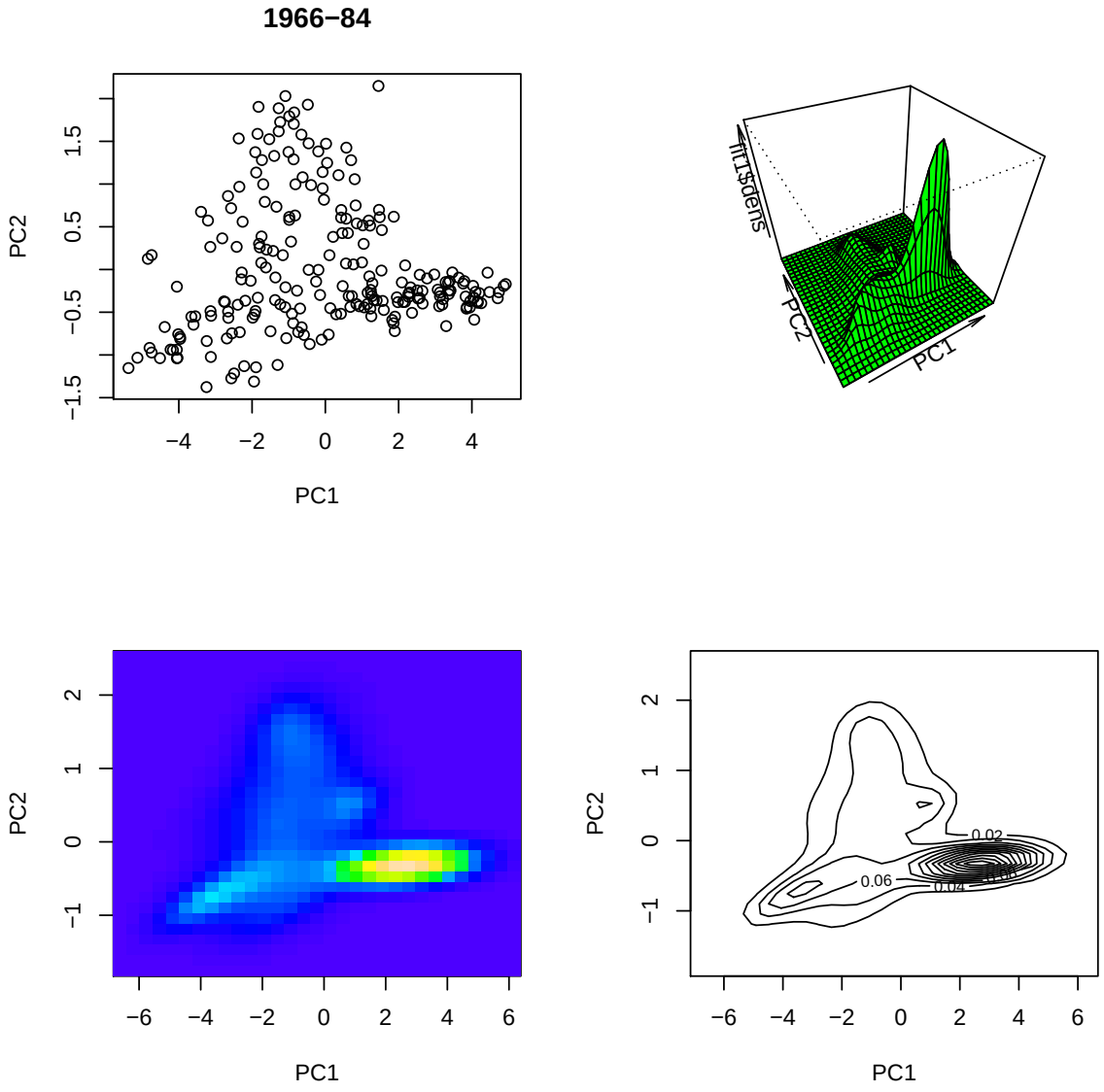


Figure 5.15. Estimated density using a Dirichlet process mixtures of normals models for the first two principal components of aircraft data (using the R package `DPpackage`).

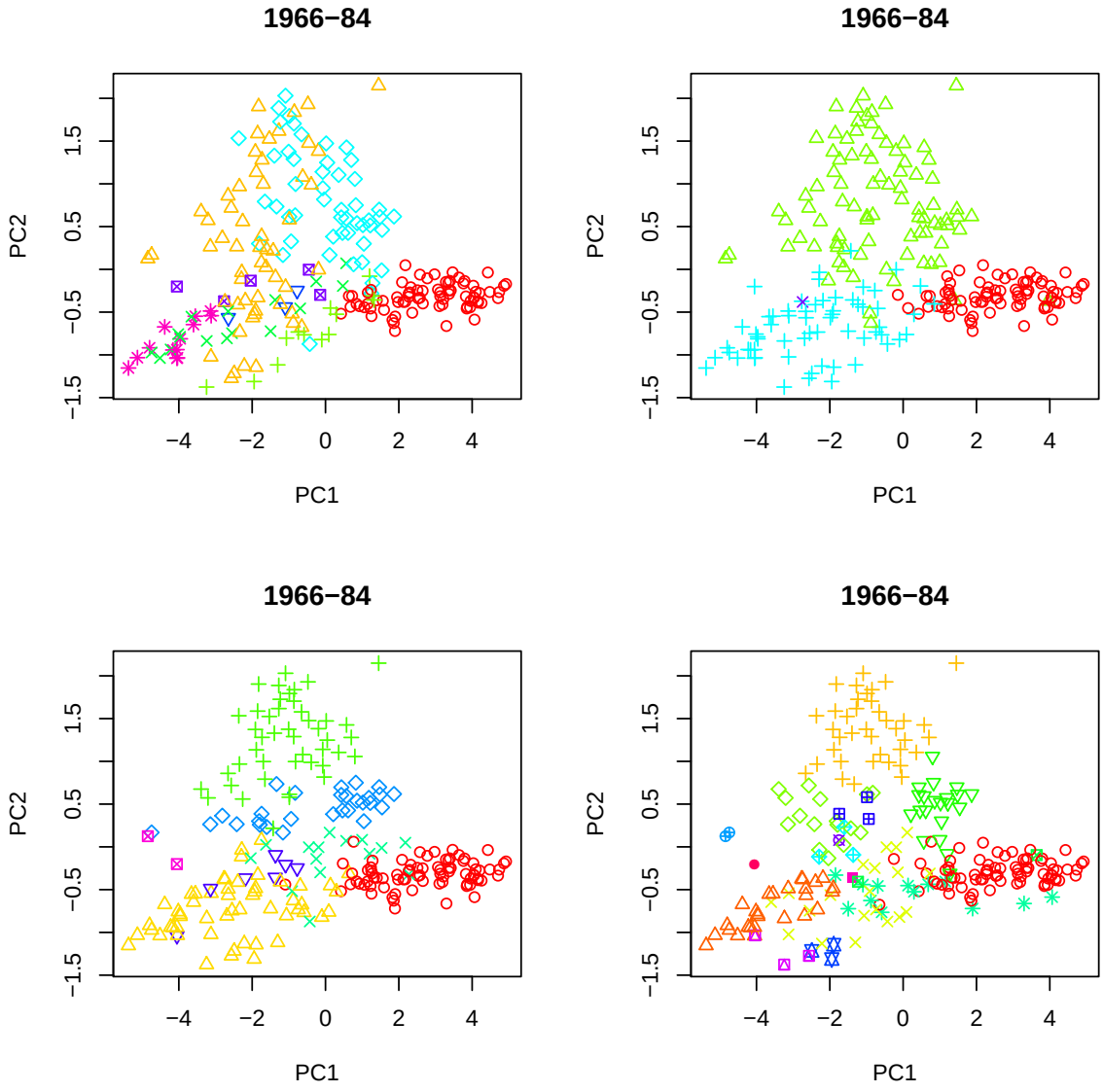


Figure 5.16. Clustering at four different iterations of the MCMC algorithm obtained for the principal components of aircraft data.

6

Extending the concepts

6.1 Non-Gaussian data

So far we have assumed that (conditional on the covariates) the response has a normal distribution. This however does not need to be the case. How we can accomodate a non-Gaussian observation model depends on the type of method we have used.

6.1.1 Basis expansion methods

Basis expansion methods such as splines are not based on an assumption of Gaussianity.

Basis expansion methods are based on replacing the usual design matrix $\mathbf{X}$ in linear regression by the matrix of basis expansions $\mathbf{B}$. When the response is from an exponential familiy we can simply fit a generalised linear model (GLM) and use $\mathbf{B}$ as the design matrix.

To use a B spline basis with a fixed degree of freedom (i.e. a fixed number of basis functions) we can simply use `bspline` inside the formula of a call to `glm`.

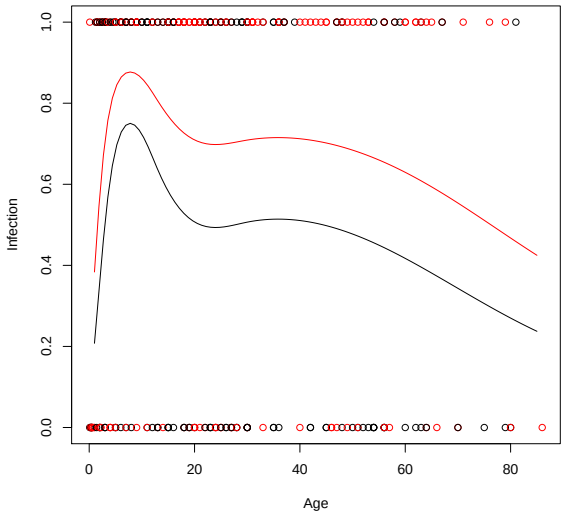
Example 6.1 (Ascaris lumbricoides). A survey of the occurence of the human parasitic worm infection *Ascaris lumbricoides* was carried out in residents of a rural community in China. The variables are:

`Age`	age of the resident
`Infection`	presence (1) or absence (0) of infection
`Sex`	male (1) or female (2)

The background to the data, and an analysis, are described by Weidong et al. (1996), *Ascaris, people and pigs in a rural community of Jiangxi province, China*, Parasitology 113, 545-57. The data (`worm`) is available in the package `sm`.

```
model <- glm(Infection~bs(Age, df=5) + as.factor(Sex), family='binomial', data=worm)
plot.model <- function(model) {
```

```
    plot(Infection~Age, data=worm, col=Sex)
    for (sex in 1:2) {
        newdata <- data.frame(Age=seq(1,85, len=100), Sex=sex)
        lines(newdata$Age, predict(model, newdata, type="response"), col=sex)
    }
}
plot.model(model)
```



◁

In the least-squares case we have included a penalty by considering the objective function

$$\sum_{i=1}^{n}(y_i - \mathbf{b}(\mathbf{x}_i)^\top \boldsymbol{\beta}) + \lambda \|\mathbf{D}\boldsymbol{\beta}\|^2 \tag{6.1}$$

When including a penalty in a likelihood-based method we simply subtract it from the loglikelihood to create the *penalised likelihood*

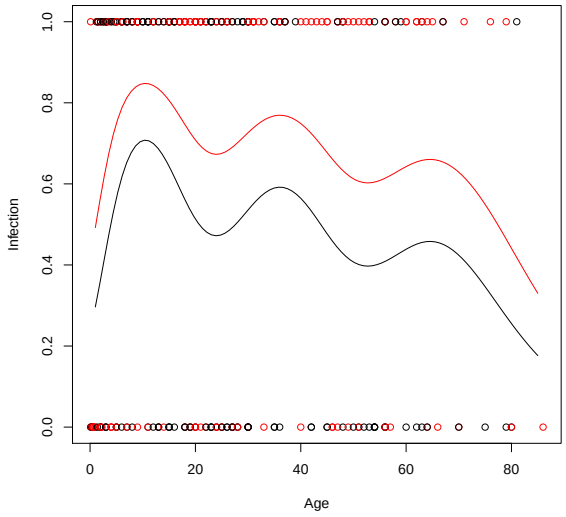
$$\sum_{i=1}^{n} \log f_{\mathbf{b}(\mathbf{x}_i)^\top \boldsymbol{\beta}}(y_i) - \lambda \|\mathbf{D}\boldsymbol{\beta}\|^2$$

The penalised negative loglikelihood becomes the scaled negative of (6.1) when assuming a normal distribution for the response.

Example 6.2 (Ascaris lumbricoides). A penalised spline mode is fitted most conveniently using the `gam` function from the package `mgcv`. We can make two different assumptions about how age and sex influence the probability of an infection.

We can assume an additive term for sex in the model, so that the fitted curves for men and women are just shifted by that estimated coefficient.

```
model <- gam(Infection~s(Age) + as.factor(Sex), family='binomial', data=worm)
plot.model(model)
```

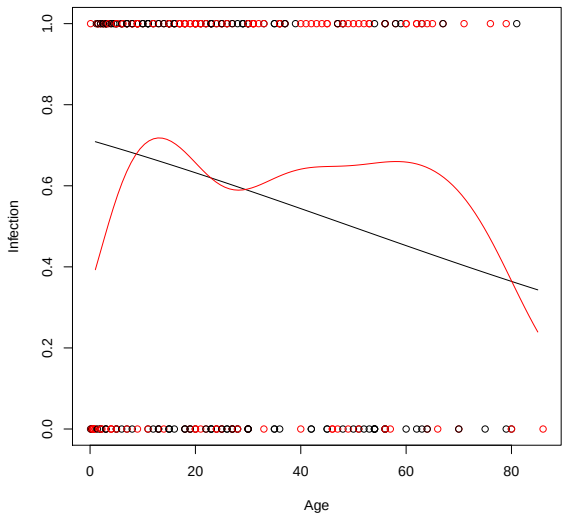



```
AIC(model)
```

```
## [1] 392.6313
```

Alternatively, we can assume that the relationship between age and infections is completely different in shape for mean and women.

```
model <- gam(Infection~s(Age, by=as.factor(Sex)), family='binomial', data=worm)
plot.model(model)
```




```
AIC(model)
```

```
## [1] 406.8489
```

We will simply use AIC for model comparison. The former model has the lower AIC and is thus preferred. ◁

6.1.2 Local estimators

In the local regression models from chapter 4 we have found the prediction at x_0 by considering the locally-weighted least squares criterion

$$\min_{\mu} \sum_{i=1}^{n} (y_i - \mu)^2 \, w(x_i - x_0 \, ; h) \tag{6.2}$$

for the running-mean estimator, or

$$\min_{\alpha,\beta} \sum_{i=1}^{n} (y_i - \alpha - \beta(x_i - x_0))^2 \, w(x_i - x_0 \, ; h) \tag{6.3}$$

for the locally linear estimator.

We generalise this to the case of non-Gaussian data by replacing the least squares loss by the loglikelihood, i.e. consider

$$\max_{\mu} \sum_{i=1}^{n} \log f_{\mu}(y_i) \, w(x_i - x_0 \, ; h)$$

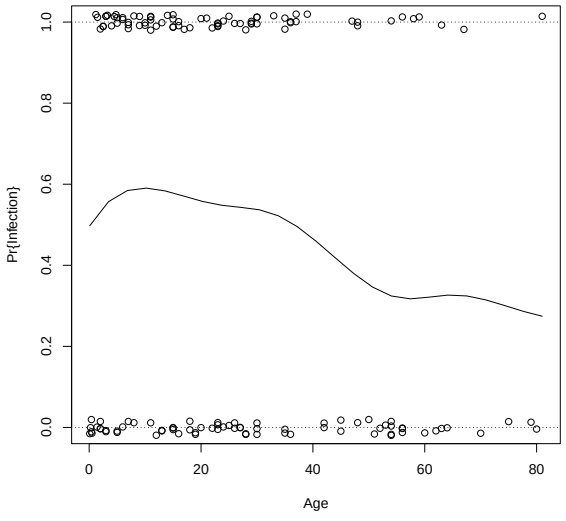
or

$$\max_{\alpha,\beta} \sum_{i=1}^{n} \log f_{\alpha+\beta(x_i-x_0)}(y_i) \, w(x_i - x_0 \, ; h)$$

When using a Gaussian model for the data these simplify to (6.1.2) and (6.3).

Example 6.3 (Ascaris lumbricoides (continued)). We can fit a local binomial regression model using the function `sm.binomial` from the package `sm`.

```
with(subset(worm, Sex==1), {
    sm.binomial(Age, Infection, h=10, group=Sex)
})
```



◁

6.1.3 From Gaussian models to latent Gaussian models

Gaussian processes are more difficult to generalise to other distributions. We cannot simply replace the assumption of a joint normal distribution by an assumption of another distribution.

One thus typically proceeds by introducing a latent Gaussian process. So, for example if we want to perform binary classification using a Gaussian process we would assume the following two-level model

$$Y_i|Z_i = z_i \sim \mathsf{Bi}(1, \pi_i) \text{ with } \pi_i = \frac{\exp(z_i)}{1 + \exp(z_i)}$$
$$\mathbf{Z} \sim \mathsf{N}(\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I})$$

The latent Gaussian process cannot be integrated out in closed form (unless the response is itself Gaussian). So inference has to either be based on an approximation or posterior simulation (MCMC). There are essentially two different, but equally succesful strategies that one can use for find an approximate solution. One is to use a Laplace approximation (used in a more sophisticated way by INLA) whereas the other is to resort to a technique called variational inference.

6.2 Generalised Additive Models for Location, Scale and Shape (GAMLSS)

In some circumstances standard regression methods based on a standard distributional assumption will not capture all aspects of the distribution of the response variable of interest.

Usually regression models are based on a covariate-based model assumption for the mean only. However in some situations not just the mean, but also the spread and the shape of the distribution of the response depend on covariates.

GAMLSS models (Rigby and Stasinopoulos, 2005) have up to four parameters which can be influenced by covariates. The μ parameter controls the location, the σ parameter controls the spread, the τ parameter controls the skewness and the ν parameter controls the kurtosis.

The `gamlss` package implements a very large number of distributions, one such distribution is the so-called Box-Cox Cole and Green distribution (BCCG) given by

$$f(y|\mu, \sigma, \nu) = \frac{1}{\sqrt{2\pi}\sigma} \frac{y^{\nu-1}}{\mu^\nu} \exp\left(-\frac{z^2}{2}\right)$$

where

$$z = \begin{cases} \frac{(y/\mu)^\nu - 1}{\nu\sigma} & \text{if } \nu \neq 0 \\ \frac{\log(y/\mu)}{\sigma} & \text{if } \nu = 0 \end{cases}$$

Example 6.4 (Effect of age on obesity in the US). The National Health and Nutrition Examination Survey (NHANES) is a program of studies designed to assess the health and nutritional status of adults and children in the United States. As part of the NHANES data (available in the R package `NHANES`) both the age and the body mass index (BMI) are collected.

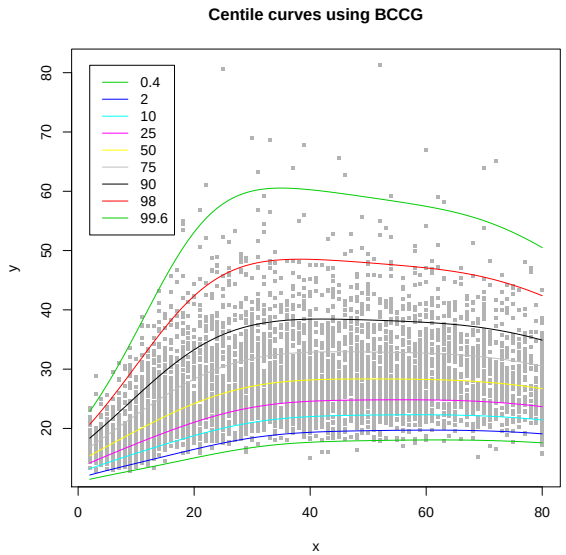
Suppose we want to study the effect of age on obesity. If the objective of our investigation is obesity, we are not really interested in how the mean BMI changes with age. Rather, we are interested in how large quantiles change with the mean. This requires modelling the effect of age on all aspects of the distribution of the BMI and not just its mean.

The function `gamlss` from the package `gamlss` lets us fit such a model.

```
model <- gamlss(BMI~ps(Age), sigma.formula=~ps(Age), tau.formula=~ps(Age), data=NHANES, family="BCCG")

   ## GAMLSS-RS iteration 1: Global Deviance = 59754.46
   ## GAMLSS-RS iteration 2: Global Deviance = 59288.42
   ## GAMLSS-RS iteration 3: Global Deviance = 59272.57
   ## GAMLSS-RS iteration 4: Global Deviance = 59272.07
   ## GAMLSS-RS iteration 5: Global Deviance = 59272.08
   ## GAMLSS-RS iteration 6: Global Deviance = 59272.09
   ## GAMLSS-RS iteration 7: Global Deviance = 59272.11
   ## GAMLSS-RS iteration 8: Global Deviance = 59272.11
   ## GAMLSS-RS iteration 9: Global Deviance = 59272.11
   ## GAMLSS-RS iteration 10: Global Deviance = 59272.11

centiles(model, xvar=NHANES$Age)
```




```
   ## % of cases below  0.4 centile is  0.2594976
   ## % of cases below  2 centile is  1.806103
   ## % of cases below  10 centile is  9.694831
   ## % of cases below  25 centile is  25.17127
   ## % of cases below  50 centile is  50.49824
   ## % of cases below  75 centile is  74.071
   ## % of cases below  90 centile is  89.72389
   ## % of cases below  98 centile is  98.0901
   ## % of cases below  99.6 centile is  99.62632
```

◁

6.3 Quantile and density regression

When the quantity of interest is just one quantile it is easiest to fit a quantile regression model. The idea of quantile regression was introduced by Koenker and Bassett (1978). Suppose we have data $\{(y_1, x_1), \ldots, (y_n, x_n)\}$ and a predictor function $m(\mathbf{x})$ which depends on parameters $\boldsymbol{\beta}$.

For Gaussian data we have used a least-squares loss function, i.e. we have chosen $\boldsymbol{\beta}$ to minimise

$$\sum_{i=1}^{n}(y_i - m(\mathbf{x}_i))^2.$$

Because the mean minimises the squared loss, standard regression is sometimes referred as to mean regression.

If we were to use the absolute loss

$$\sum_{i=1}^{n}|y_i - m(\mathbf{x}_i)|$$

we would obtain median regression.

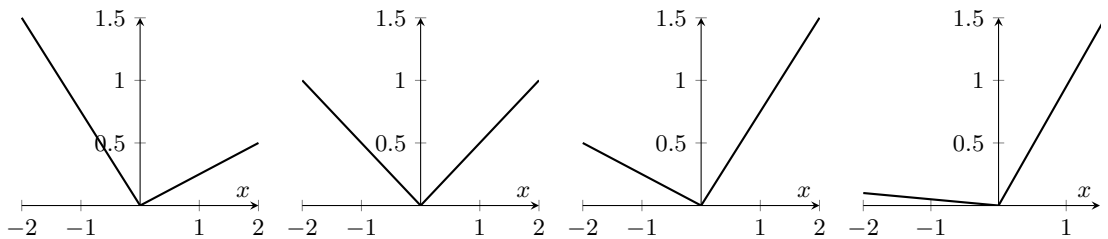
Quantile regression is based on minimising

$$\sum_{i=1}^{n}\rho_\tau(y_i - m(\mathbf{x}_i))$$

and results in an estimate of the τ-th quantile of the response distribution. $\rho_\tau(\cdot)$ is the so-called check function

$$\rho_\tau(z) = \begin{cases} \tau z & \text{if } z \geq 0 \\ (\tau - 1)z & \text{if } z < 0 \end{cases}$$

Figure 6.1 shows the check function for different $\tau \in \{0.25, 0.5m, 0.75, 0.95\}$.

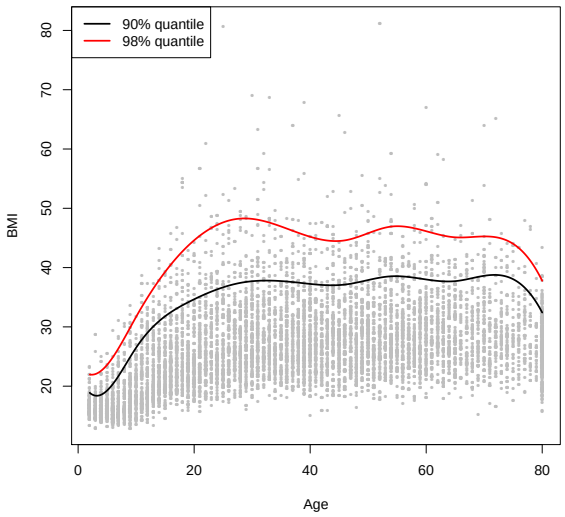


(a) $\tau = 0.25$ (b) $\tau = 0.5$ (c) $\tau = 0.75$ (d) $\tau = 0.95$

Figure 6.1. Check function for different values of τ.

Example 6.5 (Effect of age on obesity in the US (continued)). To find the 90% and 98% quantile of the conditional distribution of tge BMI given Age we need to fit two quantile regression models using the function `rq` from `quantreg`.

```
plot(BMI~Age, data=NHANES, col="grey", pch=16, cex=0.5)
newdata <- data.frame(Age=seq(2,80,len=500))
model <- rq(BMI~bs(Age, df=10), data=NHANES, tau=0.9)
lines(newdata$Age, predict(model, newdata), col=1, lwd=2)
model <- rq(BMI~bs(Age, df=10), data=NHANES, tau=0.98)
lines(newdata$Age, predict(model, newdata), col=2, lwd=2)
legend("topleft",col=1:2, lwd=2, c("90% quantile", "98% quantile"))
```



◁

In their vanilla form, quantile regression models are run separeately for each quantile. This can however sometimes lead to problem with estimated quantiles crrossing.

This can be avoided by estimating the entire conditional distribution in one go, just like GAMLSS has performed the estimation. GAMLSS however still is a (admittedly rather rich) parametric model. We can avoid such parametric assumption by using a density regression technique. Bayesian techniques for density regression are typically based on the Dirichlet process. There are a number of different classes of such models. One approach (implemented in the function `DPcdensity` in `DPpackage`) is based on estimating the point density of $(\mathbf{y}_i, \mathbf{x}_i)$ using a Dirichlet process and then computing the conditional distribution of $\mathbf{y}_i|\mathbf{x}_i$. Other approaches are based on modifying the stick breaking view or the Chinese restaurant view so that weights depend on covariates. Due to their nature these models are rather complex and go beyond the scope of this course.

6.4 Functional data analysis

The type of data which are now routinely collected can have quite complex structures, rather than simply having a single measurements of a response variable. For example, a response might be in the form of a function collected by a monitoring device which effectively collects data continuously over time. Although in practice the data may be discretised on a grid of time points, it can be helpful to think of this as representing a function. This leads to the concept of *functional data analysis* which has attracted considerable interest over the last couple of decades. There are strong links here with the techniques we have been discussing, as methods of flexible regression provide curve descriptions which reduce noise or have compact representations through basis functions.

Example 6.6 (Mediterranean fruit flies). A dataset containing the number of eggs laid from fifty Mediterranean fruit flies ("medflies", Ceratitis capitata) during the first 25 days of their lives. In addition to the number of eggs laid each data, the dataset also contains the lifespan of each fly. Our objective is to investigate whether fecundity can predict the future lifespan of a fly.

In this example the covariate is functional. Rather than having a single egg count we have a time series of 25 counts, i.e. our covariate is a function of time $x_i(t)$.

This suggests using a regression model of the form

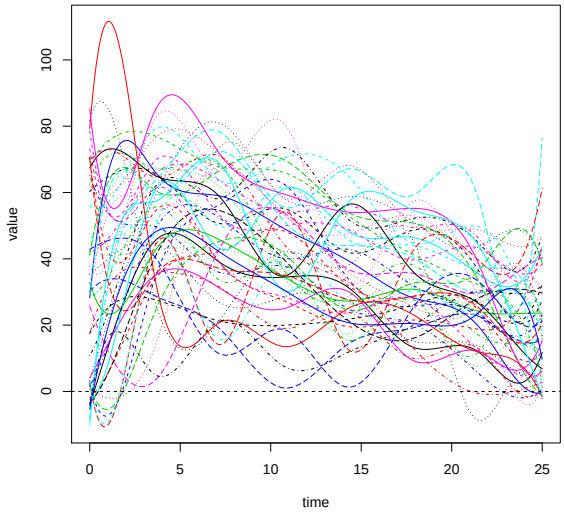
$$\mathbb{E}(y_i) = \int_0^{25} x_i(t)\beta(t)\, dt$$

to predict the future lifetime of the fly. Because the covariate is functional we also have to use a functional regression coefficient.

Such a model can be fitting using the package `fda`.

We start by creating an `fd` object to hold the functional data. Before we can create an `fd` object we need to create a set of basis functions which are then used to represent the data.

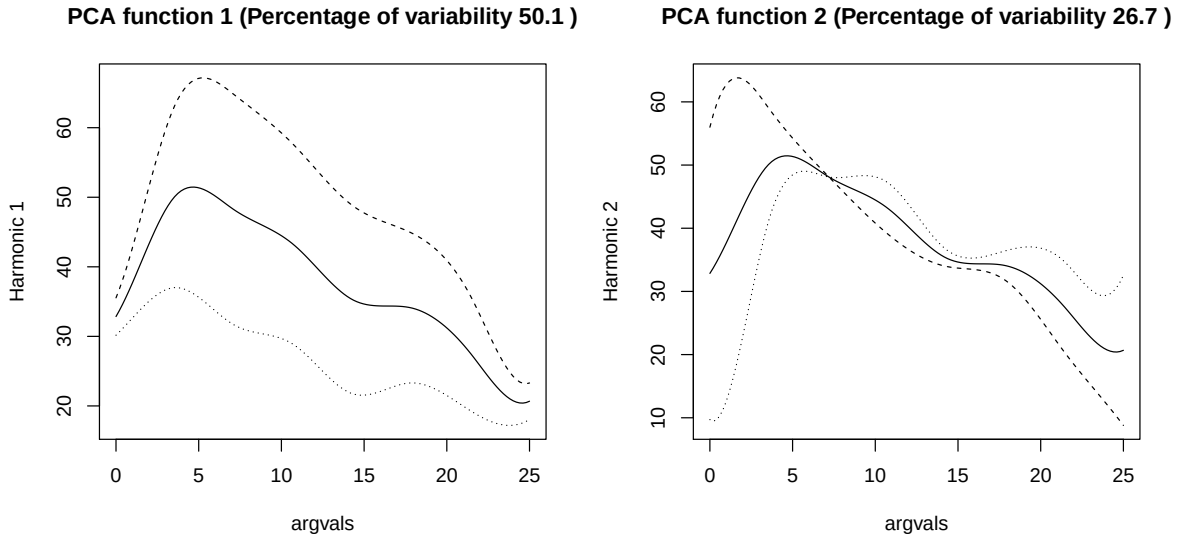
```
basisfd <- create.bspline.basis(rangeval=c(0, 25), 10)
xfd <- Data2fd(medfly$eggcount, argval=0:25, basisobj=basisfd)
lifetime <- as.numeric(medfly$lifetime)
plot(xfd)
```




```
## [1] "done"
```

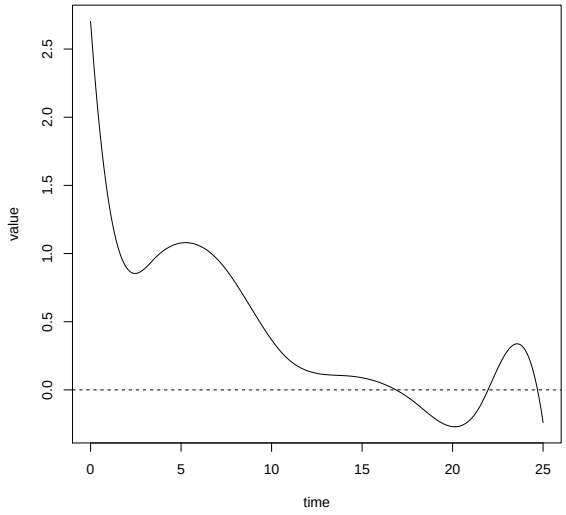
To further explore the data, we can compute the functional principal components.

```
par(mfrow=1:2)
plot(pca.fd(xfd),pointplot=FALSE)
```



Finally we fit the functional regression model. Note that the regression coefficient is now itself a function (represented as a B-spline).

```
betabasis <- create.bspline.basis(c(0,25), 10)
lifetime <- as.numeric(medfly$lifetime)
model <- fRegress(lifetime ~ xfd)
plot(model$betaestlist[[2]])
```




```
## [1] "done"
```

The the number of eggs laid towards the begin of the 25 day period is positively related to the further survival of the fly. ◁

6.5 Models for discrete space

We will finally consider the case of areal unit data. So far we have assumed that spatial data is recorded at a precise spatial location. However some data is only available at a coarser spatial resolution.

The left-hand plot in figure 6.2 shows the observed number of deaths from larynx cancer for males in each of 544 districts of Germany from 1986 to 1990. The right-hand plot shows the standardised incidence ratio (SIR) which is obtained by dividing the observed number of the cases by the number of cases expected based on the population size and age structure in each district.

It seems reasonable to assume that mortality rates in neighbouring districts are related to each other. If we assume that the mortality rate in each district is directly influenced only by its first-order neighbours we are assuming a graphical model like the one shown in figure 6.3.

We will use the following model, called the Convolution Conditional Autoregressive (CAR) model, which was first proposed by Besag *et al.* (1991).

$$
\begin{aligned}
Y_i|Z_i = z_i &\sim \mathsf{Poi}(\lambda_i E_i) \text{ with } \lambda_i = \exp(z_i^{(1)} + z_i^{(2)}) \\
z_i^{(1)}|\mathbf{z}_{-i}^{(1)} &\sim \mathsf{N}\left(\frac{\sum_{j\sim i} z_j^{(1)}}{n_i}, \frac{\tau_1^2}{n_i}\right) \\
z_i^{(2)} &\sim N(0, \tau_2^2) \; i.i.d.
\end{aligned}
$$

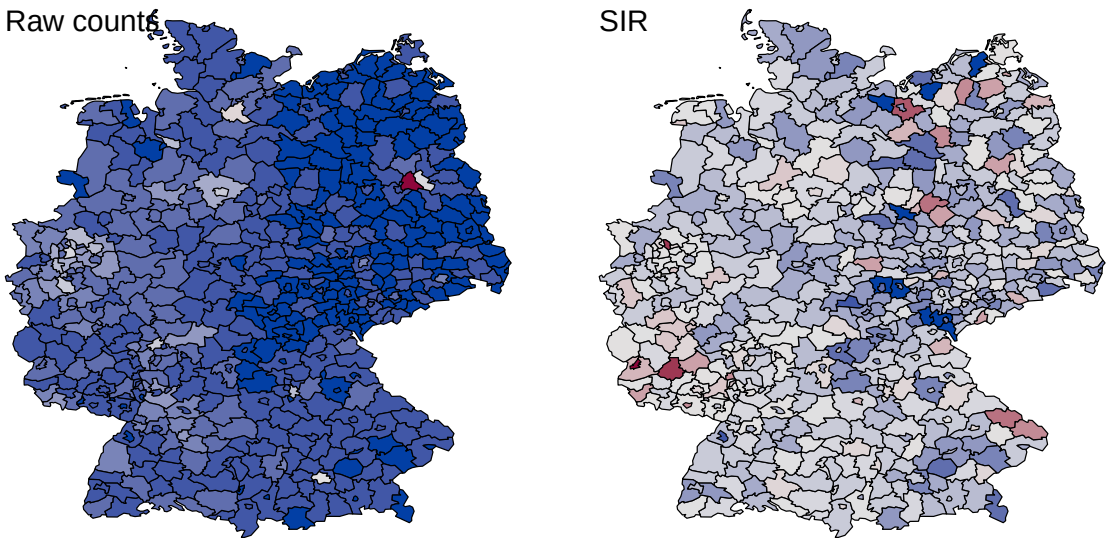


Figure 6.2. Raw counts and standardised incidence ratio (SIR) for larynx cancer in 544 German districts.

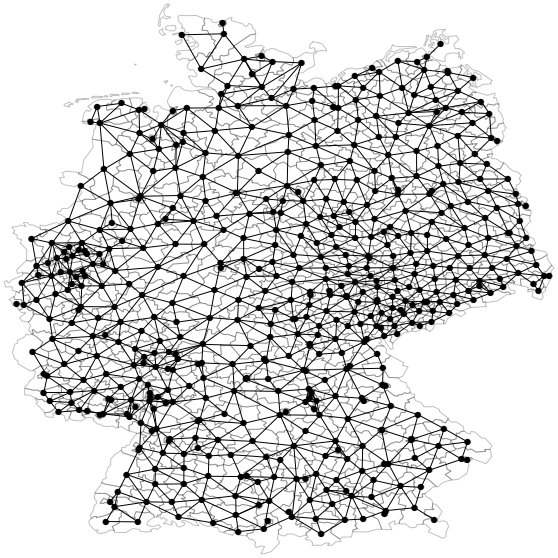


Figure 6.3. Graphical model assumed for the relationship between the latent Gaussian Markov random field for the larynx cancer counts.

where $j \sim i$ denotes all neighbours of i and n_i is the number of neighbours of district i. Y_i denotes the observed number of cases, whereas E_i denotes the expected number of cases.

Inference in this model has to be carried out using MCMC (for example using the package `CARBayes` or by performing approximate inference. Figure 6.4 shows the estimates of the SIR obtained by fitting the above CAR model using the package `INLA`.

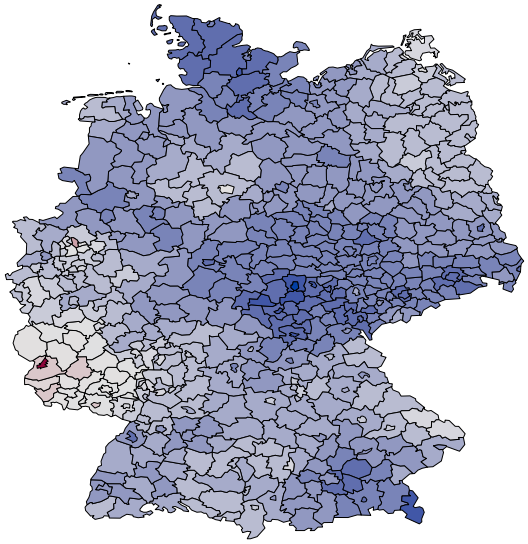


Figure 6.4. Model-based estimate of the SIR obtained using a CAR model

References

Besag, J., York, J., and Mollié, A. (1991). Bayesian image restoration, with two applications in spatial statistics (with discussion). *Annals of the Institute of Statistical Mathematics 43*, 1–59.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. London: Chapman and Hall.

Koenker, R. and Bassett, G. (1978). Regression quantiles. *Econometrica: journal of the Econometric Society*, 33–50.

Rigby, R. A. and Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape,(with discussion). *Applied Statistics 54*, 507–554.

Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: John Wiley.

Silverman, B. (1986). *Density estimation for statistics and data analysis*. London: Chapman and Halland Hall.

Simonoff, J. S. (1996). *Smoothing methods in statistics*. New York: Springer.

Wand, M. P. and Jones, M. C. (1995). *Kernel smoothing*. London: Chapman and Hall.

Wood, S. (2006). *Generalized Additive Models: an introduction with R*. London: Chapman and Hall/CRC.