Generalized Additive Models with P-splines

UC3M - Universidad Carlos III de Madrid, Statistics Department

Maria Durban < marialuz.durban@uc3m.es >

II Workshop em Bioestatística, Universidade Estadual de Maringá (Dezembro 2018)

Contents

1 Introduction

Smoothing models, and in particular, generalized additive models (GAM) are a collection of non-parametric regression techniques that attempt to estimate complex relationships between the response variable and the covariates. The singularity of these models is that there is no need to establish beforehand the relationship between the variables, the data are the ones that determine the shape of the relationship. This has made GAM models a fundamental tool in any area of research. The aim of this course is to introduce these techniques and show their use in an applied research context.We will introduce basic theoretical concepts, but the course will be mostly focused on learning through data analysis

Required packages

We will employ several packages that are not contained in R (note: list to be updated). These can be installed as:

```r
# Installation of required packages
install.packages(c("Semipar","car","lattice","splines","mgcv","nlme","MASS",
                   "latticeExtra","fields","MBA","sm","gamair","maps"))
```

1.1 Linear models (LMs)

Remember in the ordinary least squares linear regression model, we have some response/variable y we want to study assumed to be normally distributed with mean μ and variance σ^2. We also have a set of predictors/covariates or explanatory variables $X's$. The model is

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$$

where $\boldsymbol{X} = [\mathbf{1} : x_1 : x_2 : ... : x_p]$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_p)'$.

One of the issues with this model is that, in its basic form it can be very limiting in its assumptions about the data generating process for the response variable of interest.

1.2 Generalized Linear Models (GLMs)

GLMs incorporate other types of distributions for the response variable y (of the exponential family e.g.: Binomial, Poisson, Gamma, etc ...), and include a link function $g(\cdot)$ relating the mean μ through the so-called *linear predictor* η. The general form is:

$$g(\mu) = \eta = \boldsymbol{X}\boldsymbol{\beta},$$

where the estimated fitted values $\mathbb{E}[y] = \mu = g^{-1}(\eta)$. For the Poisson distribution, the (canonical) link function $g(\cdot)$, is the natural log, for Binomial is the *logit* and for Gaussian distribution is the identity.

1.3 Generalized Additive Models (GAMs)

GAMs are a generalization of GLM's to incorporate non-linear forms of the predictors.

$$y \sim \text{ExpoFam}(\mu, \sigma^2, ...)$$

$$\mu = \mathbb{E}[y]$$

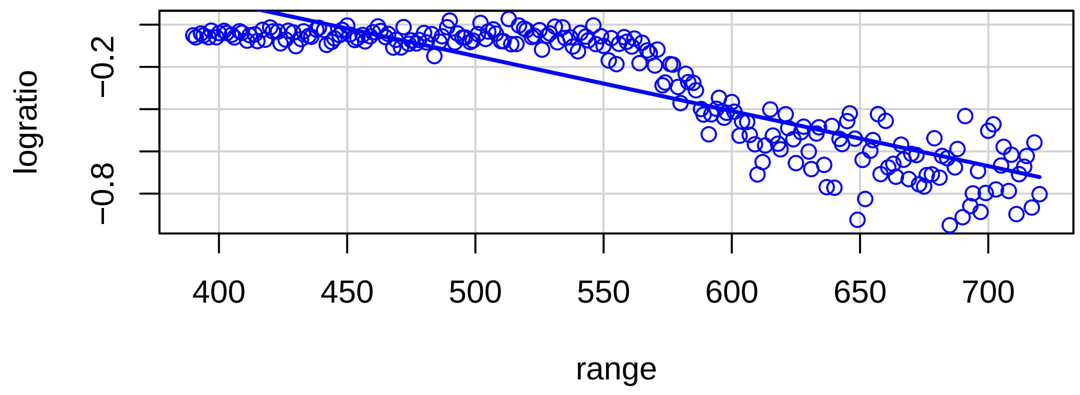


Figure 1: *LIDAR data*

$$g(\mu) = \eta = \beta_0 + f(x_1) + f(x_2) + ... + f(x_p)$$

Now we use *smooth* functions $f(\cdot)$ of our predictor variables, which can take more flexible forms. The observed values are assumed to be of some exponentinal family distribution, and μ is related to the predictors via a *link* function.

1.4 Scatterplot smoothing

Scatterplot smoothing consists of highlighting the *underlying trend* in the data. The underlying trend would be a function such as:

$$f(x) = \mathbb{E}(y|x)$$

That can also be written as:

$$y_i = f(x_i) + \epsilon_i, \qquad \mathbb{E}(\epsilon_i) = 0,$$

in which case the problem is often referred to as *non-parametric regression*, where $f(\cdot)$ os some unespecified *smooth* function that needs to be estimated from the pair (x_i, y_i)

There are several methods for smoothing a scatterplot, including *splines*, *kernel regression*, *running means*, *loess* or the weighted version *lowess*.

The blue line is the linear model fit (to remove it set `reg.line=FALSE`).

The `lidar` data frame has 221 observations from a light detection and ranging (LIDAR) experiment.

1.4.1 Polynomial regression

Polynomial regression consists of transforming the predictor X, adding p degree polynomials such that

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_p X^p + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

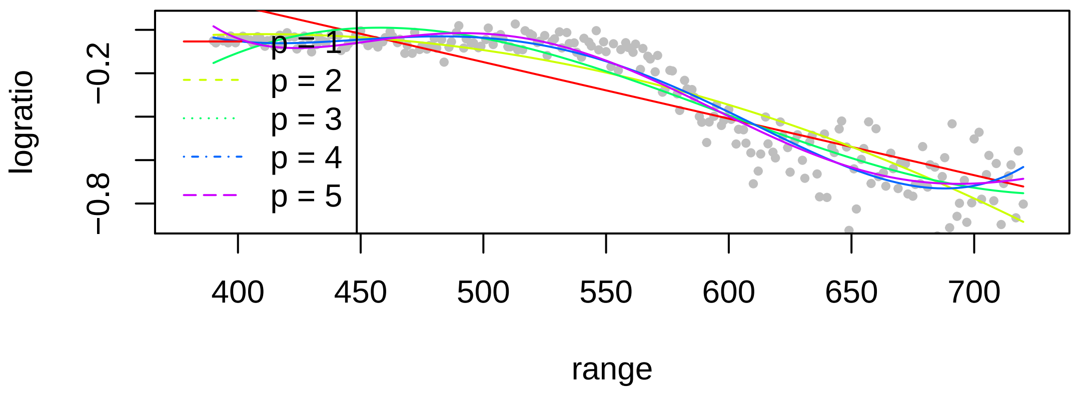


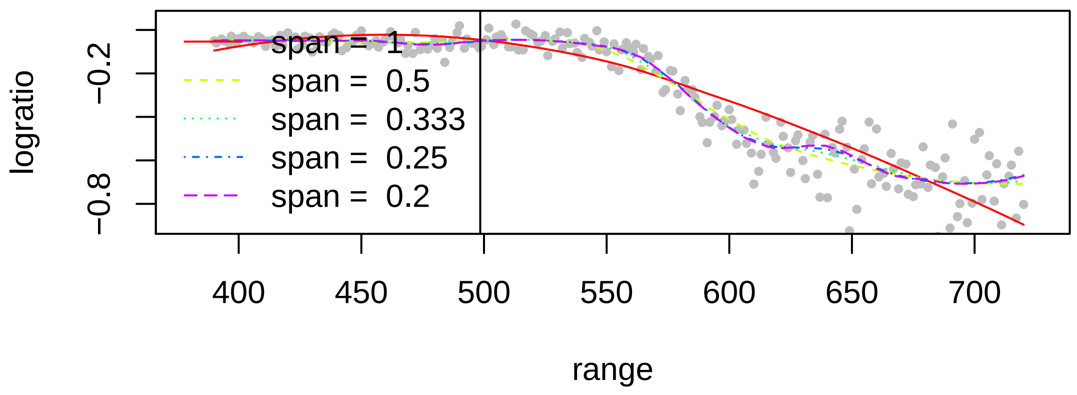
Figure 2: *Scatterplot smoothing of LIDAR data with polynomial regression of different degrees*

We need to handle such non-linear relationships effectively through more flexible techniques.

1.4.2 Local regression

Local regression

LOESS and LOWESS (locally weighted scatterplot smoothing) are two strongly related non-parametric regression methods that combine multiple regression models in a k-nearest-neighbor-based mode.



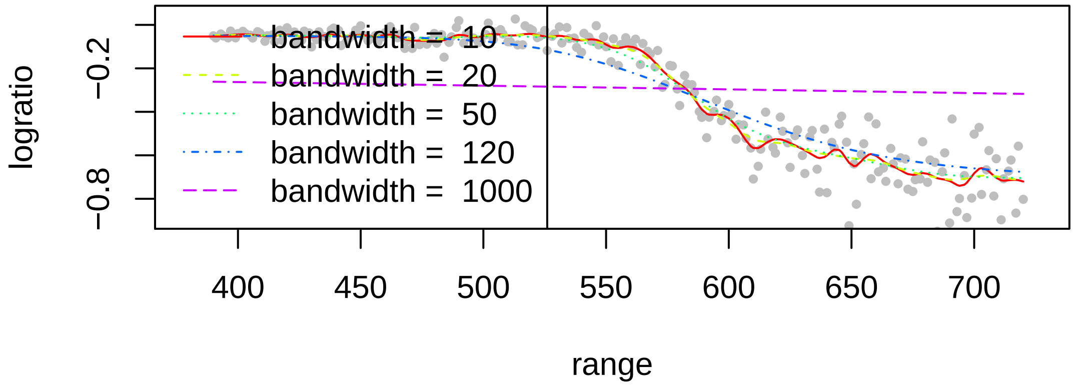


Figure 3: *Kernel smoothing with different bandwiths and Gaussian kernel*

A **kernel smoother** is of the form

$$\hat{y}_i = \frac{\sum_{j=1}^{n} y_i K\left(\frac{x_i - x_j}{b}\right)}{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{b}\right)},$$

where b is a *bandwidth* parameter, and K a kernel function, as in density estimation. There different choices for the kernel K (e.g.: Gaussian), the critical choice parameter is the bandwidth b.

In `R` the function `ksmooth` in the `library(stats)` can be used for kernel smoothing (other options are `locpoly` in `library(KernSmooth)`)

1.4.3 Splines

Splines is a modern alternative (Green and Silverman (1994)). We only need cubic splines. Divide the real line by and ordered set of points $\{z_i\}$ known as *knots*. On the interval $[z_i, z_{i+1}]$ the spline is a cubic polynomial, and its continuous and has continous first and second derivatives, imposing conditions at each knot. Boor (1978) proposed a computationally efficient to compute the so-called B-splines with desirable properties (`bs` in `R`).

A restricted form of B-splines are *natural splines* (implemented in `R` by function `ns` in `library(splines)`) is linear on $(-\infty, z_1]$ and $[z_n, \infty)$ and this have n parameters. However, `ns` adds an extra knot at each of the maximum and minimum of the data points, and so has $n+2$ parameters, dropping the requirement for the derivative to be continuous at z_1 and z_n.

The arguments of `ns` (or `bs`) are `df` (degrees of freedom) and `knots`. One can supply `df` rather than `knots`, then `df-1-intercept` inner knots at suitable chosen quantiles of x. Additionaly an `intercept` can be added to the model (default is `FALSE`)

The selection of the number of knots (or `df`) is not an easy task.

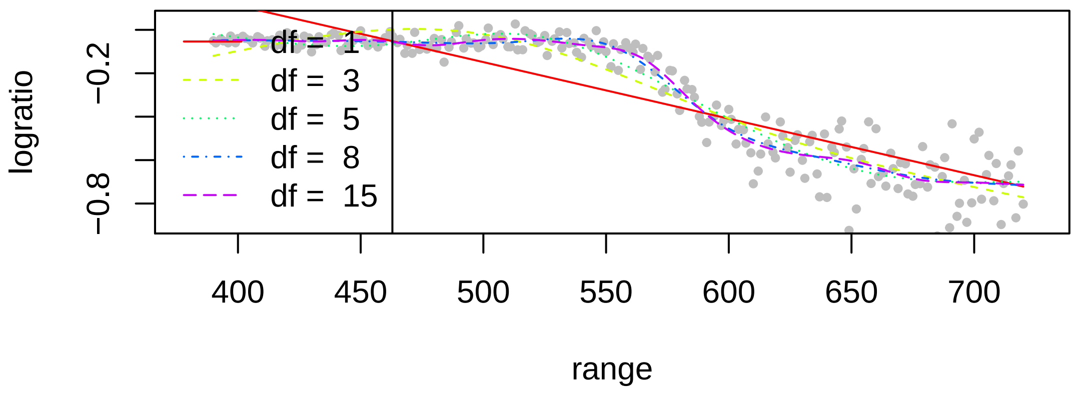


Figure 4: *Natural splines smoothing with different degrees of freedom*

1.4.4 Smoothing splines

A *smoothing spline* minimizes a compromise between the fit and the degree of smoothness of the form of a residual sum of squares:

$$RSS(f,\lambda) = \sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int_{x_1}^{x_n} (f''(x))^2 dx$$

over all (measurably twice-differentiable) functions. It is a cubic spline with knots at the x_i, but does not interpolate the data points for $\lambda > 0$ (*smoothing parameter*) and the degree of the fit is controlled by λ. The **smoothing parameter** $0 < \lambda < \infty$

`smooth.spline` function chooses automatically the degrees of freedom by cross-validation if the `df` argument is not specified. The degrees of freedom are the trace of the so-called smoother or hat-matrix H, as fitting a smoothing splines is a linear operation, there is an $n \times n$ matrix H such that $\hat{y} = Hy$.

For $\lambda = 0$ the smoothing spline will interpolate the data points if the x_i is distinct.

As a linear smoother, for each value of x_i there are a set of basis functions $B_j(x)$ such that:

$$f_\lambda(x) = \sum_{j=1}^{n} \theta_j B_j(x)$$

where θ_j are the coefficients and $B_j(x)$ are the basis functions. Let $\boldsymbol{B}$ be the $n \times n$ matrix defined by

$$B_{ij} = B_j(x_i)$$

and penalty matrix $\boldsymbol{\Omega}$

$$\boldsymbol{\Omega}_{ij} = \int_{x_1}^{x_n} B_i^{''}(x) B_j^{''} dx$$

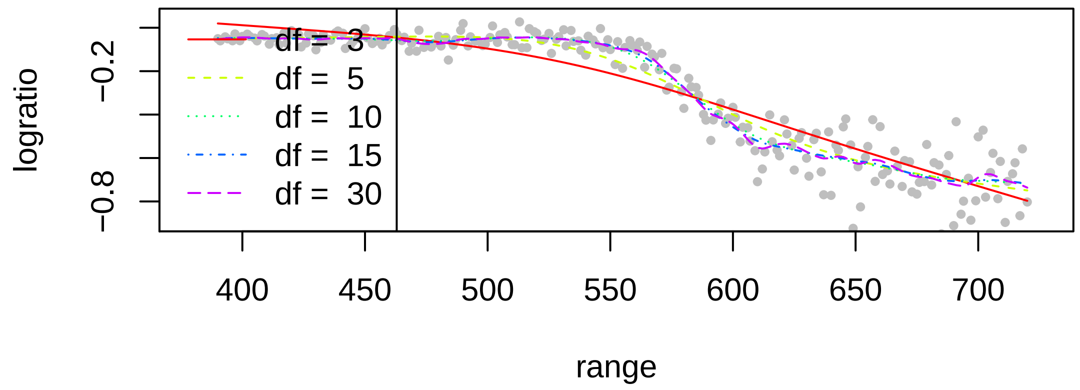


Figure 5: *Smoothing splines fit with different degrees of freedom*

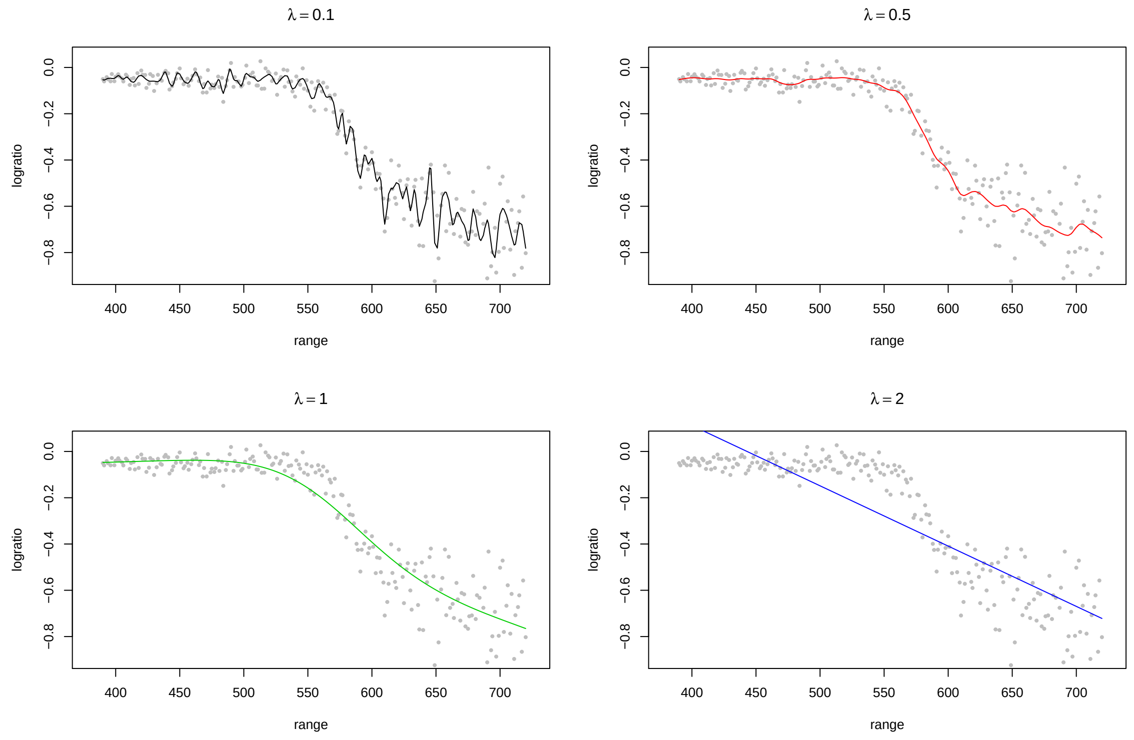


Figure 6: *Smoothing splines fit with different values of* `spar`

The penalized criterion is the minimization of the penalized residual sum of squares is defined as

$$\text{PRSS}(\boldsymbol{\theta}, \lambda) = (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta})'(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta}) + \lambda\boldsymbol{\theta}'\boldsymbol{\Omega}\boldsymbol{\theta}$$

The solution to the minimization of PRSS is

$$\hat{\boldsymbol{\theta}}_\lambda = (\boldsymbol{B}'\boldsymbol{B} + \lambda\boldsymbol{\Omega})^{-1}\boldsymbol{B}'\boldsymbol{y}$$

1.4.4.1 Choice of the smoothing parameter Cross-validation

The cross-validation is a general procedure that can be applied to estimate smoothing parameters in a wide variety of problems. Let $\boldsymbol{y}^{-i}$ be the $n-1$ vector with the $ i$th observation, y_i, removed from the original response vector $\boldsymbol{y}$. Let $\widehat{f}_\lambda^{-i}$ be the estimate based on $ n-1$ observations $\boldsymbol{y}^{-i}$. The ordinary cross-validation (CV) estimate of the prediction error is

$$\text{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda^{-i}(t_i))^2 \,.$$

A cross-validation estimate of λ is the minimizer of CV. The CV reduces to

$$\text{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{f}_\lambda(x_i)}{1 - h_{ii}}\right)^2 \,.$$

Thus one only needs to fit the model once with the full data and compute the diagonal elements of the H_λ hat-matrix.

Generalized Cross-validation

Replacing h_{ii} in CV by the average of all diagonal elements, Craven and Wahba (1979) proposed the following generalized cross-validation (GCV) criterion

$$\text{GCV}(\lambda) = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda(t_i))^2}{(1 - trH(\lambda)/n)^2} \,.$$

```
#default mode chooses df by generalized cross-validation
sm.gcv <- smooth.spline(range,logratio)
sm.cv <- smooth.spline(range,logratio,cv=TRUE)   # cv
```

2 Penalized splines

In this course we focus on *penalized splines* (*P*-splines) by Eilers and Marx (1996). In the prevoius section we discussed two types of smoothing methods with splines:

1. *Regression splines* are fitted by least squares once the number and position of the knots are choosen (very complex in most situations).

2. *Smoothing splines* use as many parameters as observations (computationally unefficient for large data sets).

A better solution consists of combining both approaches by *Penalized splines*: they use less parameters than *smoothing splines*, and the selection of the position and the number of knots is not an issue. There are 3 main reasons why use *P*-splines:

1. They are *low-rank smoothers*, i.e. the size of the basis $\boldsymbol{B}$ is $n \times c$ where n is the number of data and c is the number of regression coefficients, usually $c <<< n$. The number of regression coefficients depends on the number of knots, usually with P-splines the number of knots is less than 40. This is specially important with large data sets.

2. Including penalizations relaxes the need of choosing the number and position of the knots compared to regression splines,

3. There is a correspondence of P-splines and BLUP (best linear unbiased predictor) in mixed models. This fact allows us to apply the methodology of mixed models and the use of standard software such as `PROC MIXED` in `SAS` and `lme` in `librar(nlme)`.

2.1 Basis and penalties

Example

Suppose n pairs (x_i, y_i) and we want to fit the model

$$y_i = f(x_i) + \epsilon_i \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Let us consider the next simulated example

```r
set.seed(2015)
n <- 200
x <- seq(0,1,l=n)
# original function
f <- sin(3*pi*x)
y <- f + rnorm(n,.33)

plot(x,y,col="grey",pch=19)
lines(x,f,col=1)
```

Our aim is to estimate the function $f(x) = \sin(3\pi x)$ from the observed data (x_i, y_i). P-splines were proposed by Eilers and Marx (1996) although many other authors have popularized the techniques in different applications (See Ruppert, Wand, and Carroll (2003) and Wood (2006)).

The methodology can be summarized as follows:

a. A regression basis

b. A discrete penalty term added to the likelihood function that acts directly on the regression coefficients.

In the case of Gaussian data, we have the model

$$\boldsymbol{y} = \boldsymbol{B\theta} + \boldsymbol{\epsilon}, \quad \text{with} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$$

where $\boldsymbol{B}(x)$ is the regression basis constructed from the x variable. In order to estimate the regression coefficients, we minimize the penalized sum of squares function:

$$\text{PSS}(\boldsymbol{\theta}, \boldsymbol{y}, \lambda) = (\boldsymbol{y} - \boldsymbol{B\theta})'(\boldsymbol{y} - \boldsymbol{B\theta}) + \lambda \boldsymbol{\theta}' \boldsymbol{P\theta}$$

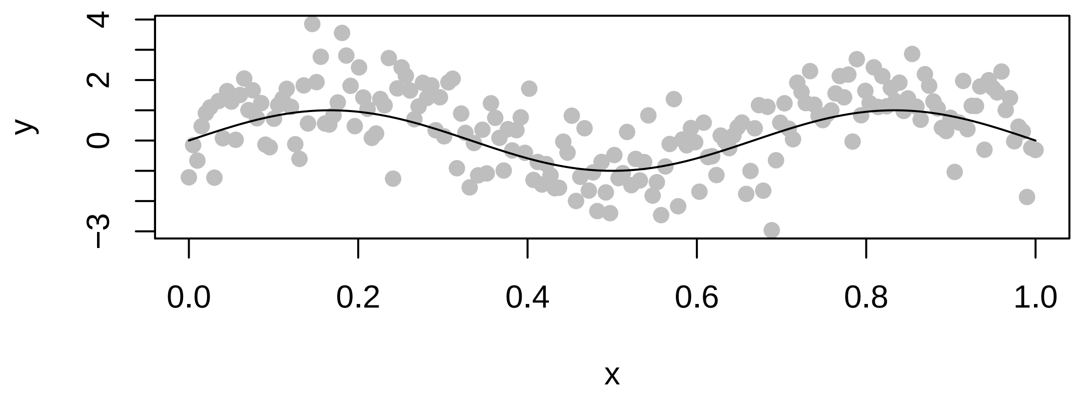


Figure 7: *Simulated data example*

where $\boldsymbol{P}$ is a matrix that penalizes the coefficients and λ is the smoothing parameter. For a fixed value of λ the minimization of PSS gives

$$(\boldsymbol{B}'\boldsymbol{B} + \lambda \boldsymbol{D}'\boldsymbol{D})\hat{\boldsymbol{\theta}} = \boldsymbol{B}'\boldsymbol{y}$$

It is easy to show that

$$\hat{\boldsymbol{y}} = \boldsymbol{B}(\boldsymbol{B}'\boldsymbol{B} + \lambda \boldsymbol{D}'\boldsymbol{D})\boldsymbol{B}' = \boldsymbol{H}\boldsymbol{y}$$

$\boldsymbol{H}$ is not a proyection matrix (it is not idempotent, $\boldsymbol{H}\boldsymbol{H} \neq \boldsymbol{H}$). The trace of the hat-matrix $\mathrm{tr}(\boldsymbol{H})$ is the effective degrees of freedom of the model.

2.1.1 Basis functions and knots

There are several options for regression bases with P-splines, there are 2 main approaches:

- Truncated polynomials
- B-splines

Other options are *Thin plate splines*

2.1.1.1 Truncated polynomials Consider the pairs (x_i, y_i). Consider $x \in [0, 1]$. Let $t_j = (j-1)/k$, $j = 2, ..., k+1$ be a set of k equally-space knots. The simplest version of Truncated Polynomial Functions (TPFs) of degree p is defined as:

$$1, x, x^2, \{(x-t_1)_+\}^p, ..., \{(x-t_k)_+\}^p$$

where $x_+ = \max(0, x)$. The function $\{(x-t_1)_+\}^p$ has $p-1$ continuous derivatives, such that the higher degree p the smoother the basis function.

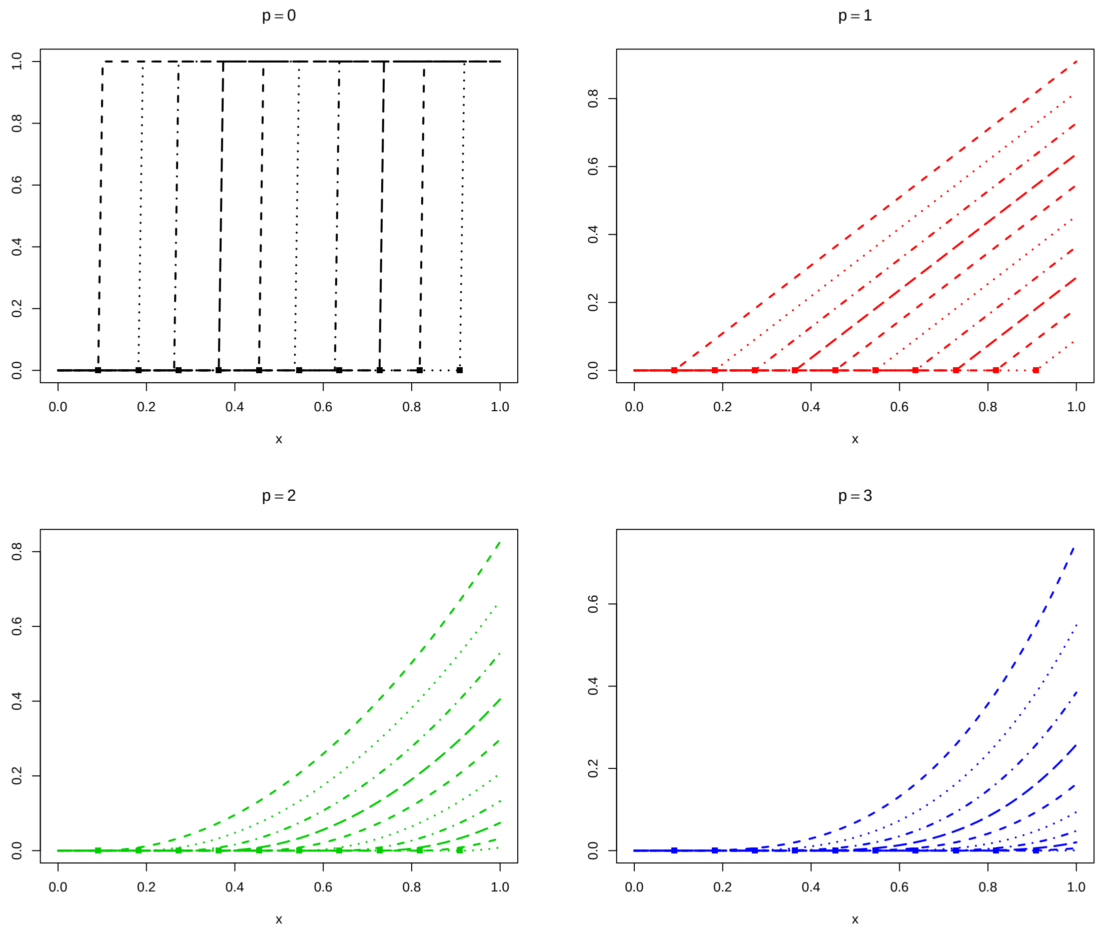


Figure 8: *Truncated Polynomial Functions with different degrees p*

```r
# Function to create TPFs
tpoly <- function(x,t,p){
  B = NULL
  for (i in 1:length(t)){
    B <- cbind(B,(x-t[i])^p*(x>t[i]))
  }
  B
}
```

Nex Figure illustrates the TPFs with different degrees with equally-spaced knots

2.1.1.2 ***B*-splines** Basic references are Boor (1978) and Dierckx (1993)

B-splines are formed by pieces of polynomials connected by knots.

The general properties of a B-spline of degree q

- Consists of $q+1$ polynomial pieces of each of degree q.
- The polynomial pieces join at q *inner knots*.
- at the joining points, derivatives up to order $q-1$ are continuous.
- The B-spline is positive on a domain spanned by $q+2$ knots; everywhere else is zero.
- Except at the boundaries, it overlaps with $2q$ polynomial pieces of its neighbors
- at given x, $a+1$ B-splines are non-zero.

The next function creates a B-spline basis function:

```r
library(splines)
bspline <- function(x,xl,xr,ndx,bdeg){
  dx <- (xr-xl)/ndx
  knots <- seq(xl-bdeg*dx,xr+bdeg*dx,by=dx)
  B <- spline.des(knots,x,bdeg+1,0*x,outer.ok=TRUE)$design
  output <- list(knots=knots,B=B)
  return(output)
}
```

where * `xl` is the left bound of x * `xr` is the right bound of x * `ndx` is the number of segments (*inner knots*) in which we divide the range of x * `bdeg` degree of the piecewise polynomial (usually cubic splines `q=3`)

The next Figure illustrates B-spline bases with different degrees of the polynomials.

2.2 Penalties and coefficients

Suppose a B-spline regression with a basis function $\boldsymbol{B}$ constructed by k knots. By means of least squares the fitted curve would be given by the solution of the mininization of:

$$\min \mathrm{SS}(\boldsymbol{\theta}, \boldsymbol{y}) = (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta})'(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta})$$

where $\hat{\boldsymbol{\theta}} = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{y}$. The fitted curve $\hat{f}(x) = \boldsymbol{B}\hat{\boldsymbol{\theta}}$ depends on the size of the basis $\boldsymbol{B}$. Next figure illustrates the effect of the size of the basis in the curve fit. The larger the basis, the fit tends to interpolate the data points.

O'Sullivan (1986) suggested including a penaly on the second derivative of the curve. Hence the objective function becomes:

$$\min \mathrm{SS}(\boldsymbol{\theta}, \boldsymbol{y}, \lambda) = (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta})'(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta}) + \lambda \int_{\boldsymbol{x}} (\boldsymbol{B}''\boldsymbol{\theta})^2 d\boldsymbol{x}$$

This penalty is similar to the one used in smoothing splines, but any derivative order can be used.

The advance in P-splines is that the penalty term is discrete and directly applied to the regression coefficients (instead of the curve), reducing the dimensionality of the problem.

The type of penalization depends on the type of regression basis. For instance, in the case of TPFs, the penalty is a *ridge* type penalty independent to the degree of the polynomial, i.e.

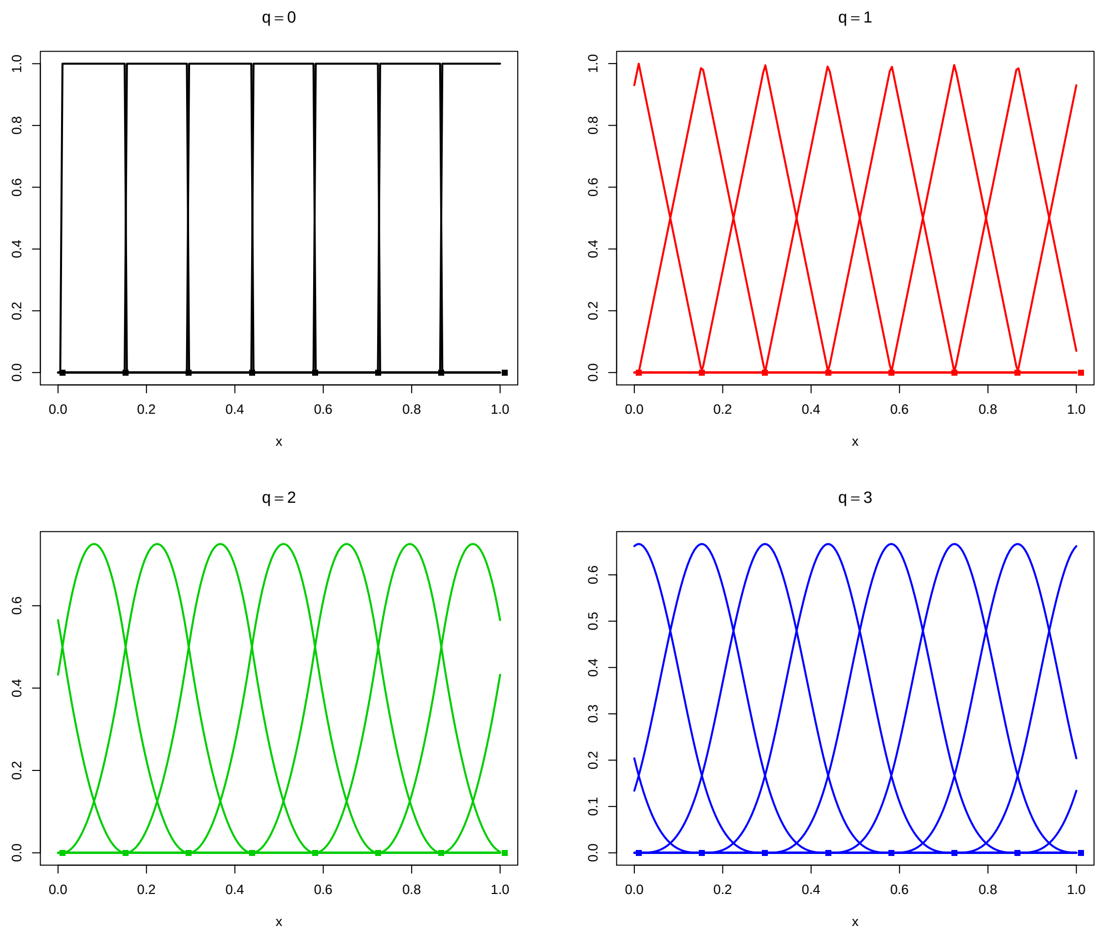


Figure 9: *B-spline bases with different degrees q*

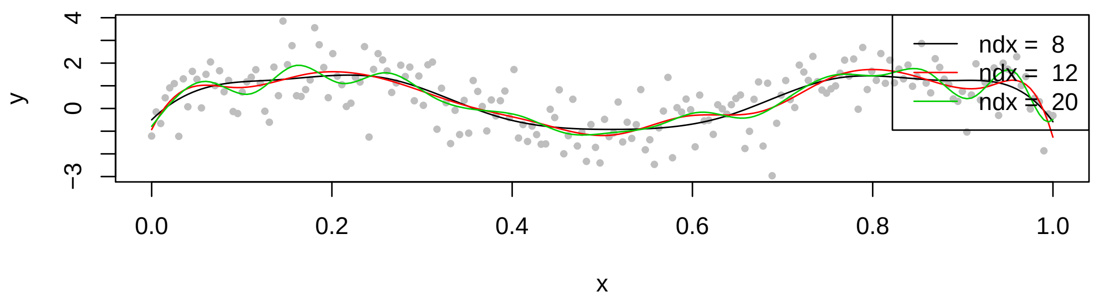


Figure 10: *B-spline regression with different number of segments*

$$\min \mathrm{SS}(\boldsymbol{\theta}, \boldsymbol{y}, \lambda) = (\boldsymbol{y} - \boldsymbol{B\theta})'(\boldsymbol{y} - \boldsymbol{B\theta}) + \lambda \boldsymbol{\theta}'\boldsymbol{\theta}$$

which is equivalent to impose a penalty on the $p+1$ derivative of the curve.

In contrast, Eilers and Marx (1996) proposed a penalty based on discrete differences of order d between adjacent coefficients of the B-spline basis. This is a more flexible penalty as it is independent from the degree of the polynomial used to construct the regression basis. Moreover, it is a good discrete approximation to the integral of the squqare of the dth derivative. The minimization criteria becomes:

$$\min \mathrm{SS}(\boldsymbol{\theta}, \boldsymbol{y}, \lambda) = (\boldsymbol{y} - \boldsymbol{B\theta})'(\boldsymbol{y} - \boldsymbol{B\theta}) + \lambda \boldsymbol{\theta}'\boldsymbol{P}_d\boldsymbol{\theta}$$

where $\hat{\boldsymbol{\theta}} = (\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P}_d)^{-1}\boldsymbol{B}'\boldsymbol{y}$, with $\boldsymbol{P}_d = (\boldsymbol{\Delta}^d)'\boldsymbol{\Delta}^d$, for $d = 0$ we have a ridge penalty. In general, we use $d = 2$, although other orders can be used, according to the variability of the curve and the amount of noise in the data.

A second order penalty $d = 2$ is equivalent to

$$(\theta_1 - 2\theta_2 + \theta_3)^2 + ... + (\theta_{k-2} - 2\theta_{k-1} + \theta_k)^2 = \boldsymbol{\theta}'\boldsymbol{D}'\boldsymbol{D}\boldsymbol{\theta}$$

where

$$\boldsymbol{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & ... \\ 0 & 1 & -2 & 1 & ... \\ 0 & 0 & 1 & -2 & ... \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \qquad \text{of size } (c-d) \times c$$

where $c = k + deg + 1$ the number of columns of $\boldsymbol{B}$.

The next Figure illustrates the P-spline fit with different values of the smoothing parameter λ and second order penalty $d = 2$.

The next Figure illustrates the P-spline fit with different values of λ. The basis function multiplied by the coefficients and the regression coefficients are also represented in the plots. The equally-spaced knots are represented by squares (■) and the coefficients by ∘. It results clear to visualize the fact that as the penalty increases it forces the coefficient to be smooth.

Among the properties of P-splines with B-spline bases we can highlight:

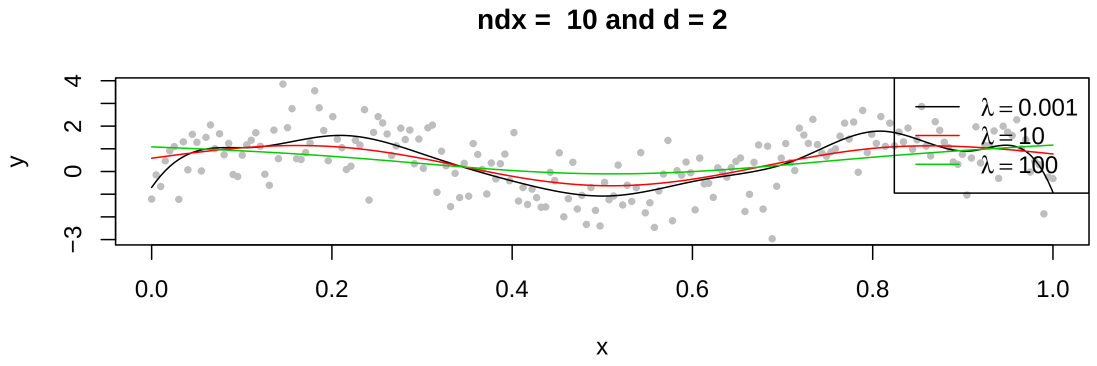


Figure 11: *B-spline regression with different values of the smoothing parameter*

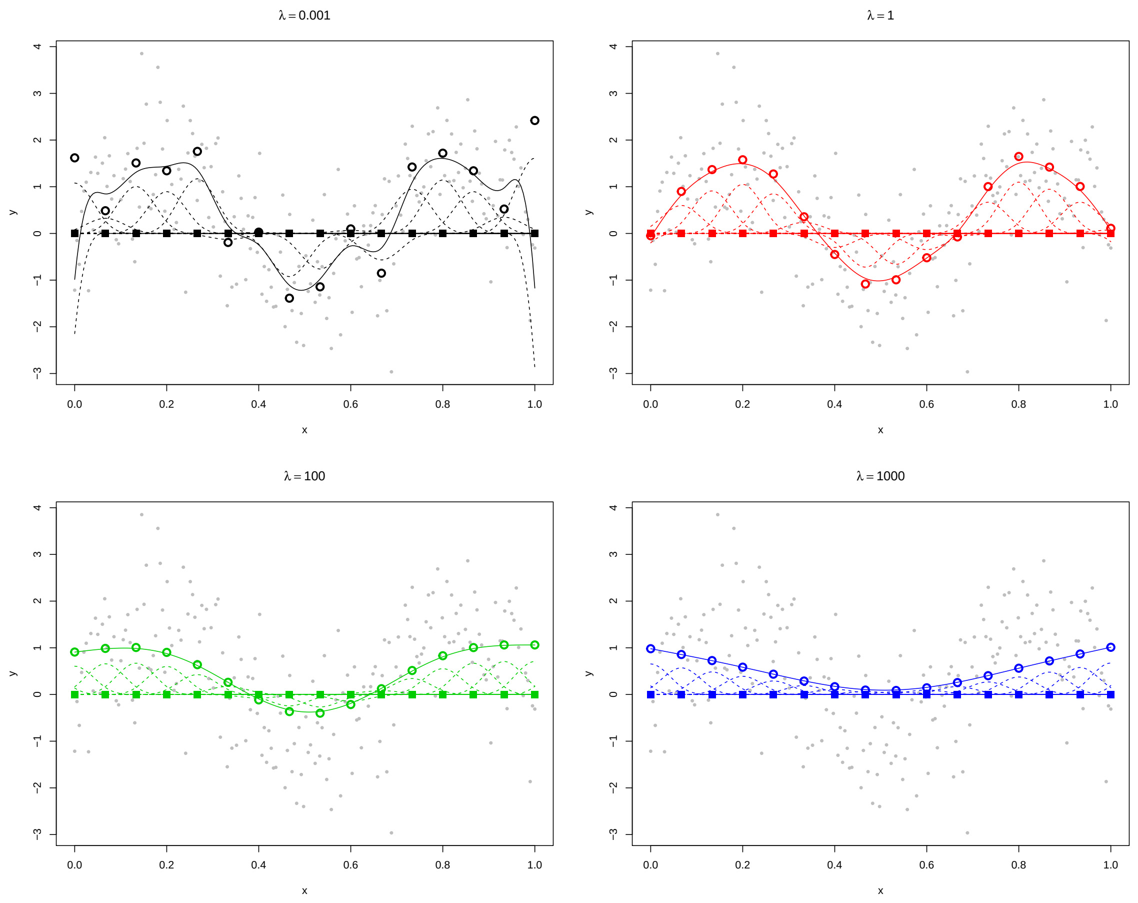


Figure 12: *P-spline regression fit with basis and coefficients*

- P-splines show no boundary effects, as many types of kernel smoothers do. By this we mean the spreading of a fitted curve or density outside of the (physical) domain of the data, generally accompanied by bending toward zero.
- P-splines can fit polynomial data exactly. Let data (x_i, y_i) be given. If the y_i are a polynomial in x of degree p, then B-splines of degree p or higher will exactly fit the data.
- P-splines can conserve moments of the data. This property is especially useful in the context of density smoothing: the mean and variance of the estimated density will be equal to mean and variance of the data, for any amount of smoothing. This is an advantage compared to kernel smoothers: these inflate the variance increasingly with stronger smoothing.
- The limit of a P-splines fit with strong smoothing is a polynomial.
- For large values of λ and a penalty of order q, the fitted series will approach a polynomial of degree $q-1$, if the degree of the B-splines is equal to, or higher than, q.
- For the selection of the position and the number of knots, equally spaced knots and a relatively large number k can be chosen, usually < 40.

2.3 Parameters estimation and degrees of freedom

We already know that for a given regression basis function $\boldsymbol{B}$ and penalty $\boldsymbol{P}$, the solution for the regression coefficients is given by

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P}_d)^{-1}\boldsymbol{B}'\boldsymbol{y}$$

for a given value of λ.

Effective degrees of freedom

In a P-spline model with $\lambda = 0$ the degrees of freedom of the model corresponds to the number of columns (regression coefficients) of the basis function $\boldsymbol{B}$, in contrast, with a large $\lambda \to \infty$ the model is not very flexible and there are few degrees of freedom.

The degrees of freedom are computed analogously to linear models as the trace of the so-called hat matrix

$$\boldsymbol{H} = \boldsymbol{B}(\boldsymbol{B}'\boldsymbol{B} + \lambda\boldsymbol{D}'\boldsymbol{D})^{-1}\boldsymbol{B}',$$

where

$$\text{tr}(\boldsymbol{H}) = \text{tr}(\boldsymbol{B}'\boldsymbol{B} + \lambda\boldsymbol{D}'\boldsymbol{D})^{-1}\boldsymbol{B}'\boldsymbol{B}.$$

Residual variance Another parameter of interest is the estimation of the residual variance $\hat{\sigma}^2$. For Gaussian errors, we use

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - \text{tr}(\boldsymbol{H})},$$

where RSS is the residual sum of squares, i.e. $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

The function `psfit` below, estimates a P-spline for a given λ using the function `ls.fit` and giving the generalized cross validation criteria value `gcv`.

```r
# Function to fit a Gaussian P-spline for given lambda
psfit <- function(x,y,pord=2,ndx=10,lambda=1){
  xl = min(x)
  xr = max(x)
  n <- length(y)
  B <- bspline(x,xl,xr,ndx,bdeg=3)$B
  nb <- ncol(B)
  P <- diff(diag(nb),differences=pord)

  # Construct penalty stuff
  P <- sqrt(lambda) * diff(diag(nb), diff = pord)
  nix = rep(0, nb - pord)

  # Fit
  f = lsfit(rbind(B, P), c(y, nix), intercept = FALSE)
  h = hat(f$qr)[1:n]
  theta = f$coef
  f.hat = B %*% theta

  # Cross-validation and dispersion
  trH <- sum(h)
  rss <- sum((y-f.hat)^2)
  gcv <- n*rss/(n-trH)^2
  sigma = sqrt(rss / (n - trH))

  # Error bands ("Bayesian estimate")
  Covb = solve(t(B) %*% B + t(P) %*% P)
  Covz = sigma ^ 2 * B %*% Covb %*% t(B)
  seb = sqrt(diag(Covz))

  output <- list(gcv=gcv,sigma=sigma,
                 f=f,theta=theta,f.hat=f.hat,seb=seb,trH=trH)
  return(output)
}
```

2.4 Choosing λ

We can use generalized cross-validation criteria:

$$\text{GCV} = \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{n - \text{tr}(\boldsymbol{H})},$$

or AIC

$$\text{AIC} = 2\log\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right) - 2\log(n) + 2\log(\text{tr}\boldsymbol{H})$$

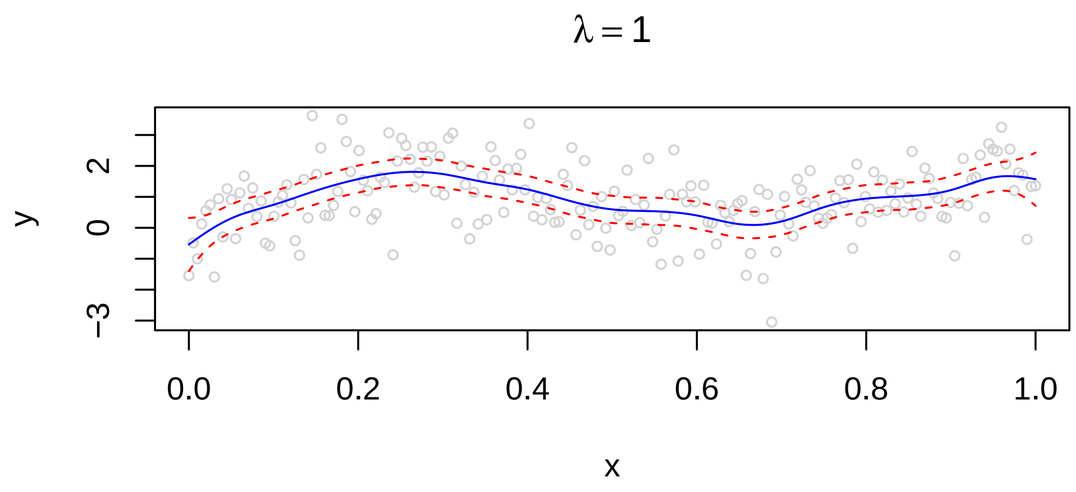


Figure 13: *P-spline fit with confidence bands computed with* ***psfit***

```
# plot of GCV criteria is more useful
lla = seq(-2, 2, by = 0.10)
cvs = 0 * lla
for (k in 1:length(lla)) {
  lambda = 10 ^ lla[k]
  pn =  psfit(x,y, lambda = lambda)
  cvs[k] = pn$gcv
}
#
lam.cv <- 10^(lla[which.min(cvs)])
lam.cv # lambda chosen by generalized cv
```

```
## [1] 5.011872
```

```
fit.cv <- psfit(x,y,lambda=lam.cv)

fit.cv$sigma # estimated sigma error
```

```
## [1] 0.9630074
```

2.5 Other penalized regression basis

Thin plate regression splines (TPRS)

Thin plate splines can be used to estimate the smooth function $f(\cdot)$ by finding the function f minimizing

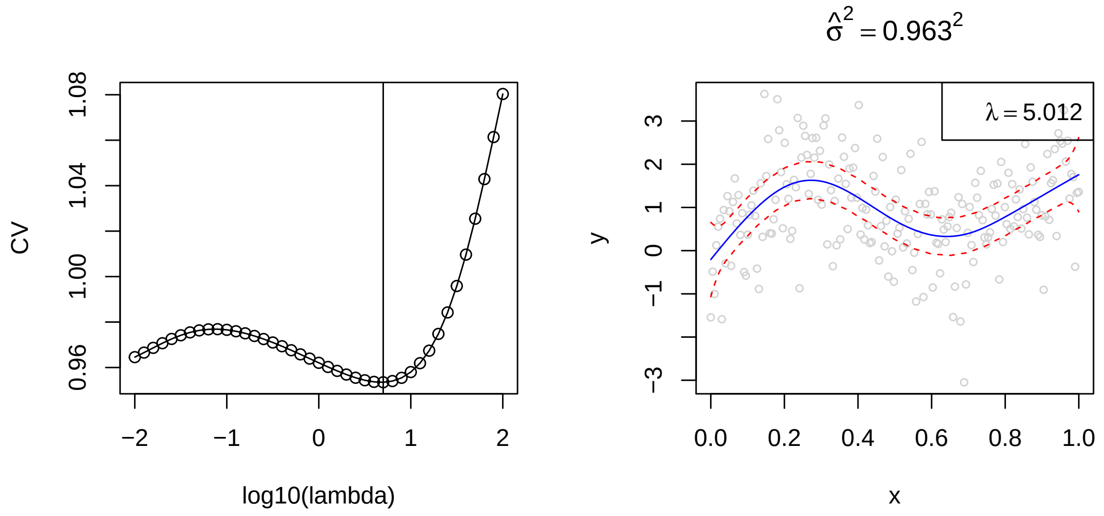


Figure 14: *P-spline fit with confidence bands computed with* **`psfit`** *and smoothing parameter selected by minimizing gcv*

$$\sum (y_i - f(x_i))^2 + \lambda \int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 dx$$

It is possible to rewrite the equation as:

$$\|\boldsymbol{y} - \boldsymbol{E}\boldsymbol{\delta} - \boldsymbol{T}\boldsymbol{\gamma}\| + \lambda\boldsymbol{\delta}'\boldsymbol{E}\boldsymbol{\delta}$$

subject to $\boldsymbol{T}'\boldsymbol{\delta} = 0$. See Green and Silverman (1994) for details.

The main problem is the number of knots (i.e. the number of coefficients to estimate) equal to the number of observations. Wood (2003) proposed the use of low-rank Thin plate regression splines with a much number of knots.

2.6 Penalized splines as mixed models

The great popularity of P-splines during the last decade is due to the possibility of rewriting the non-parametric model as a mixed model (with random effects). Hence, we are able to use the methodology developed in the context of mixed models to smoothing and use the available mixed model software for estimation and inference.

The connection between smoothing and mixed model was initially addressed by several authors (Speed (1991), Verbyla et al. (1999) and Brumback and Rice (1998)). The idea is to reformulate the smoothing model as a mixed-effects model:

$$\begin{array}{rcll} \boldsymbol{y} & = & \boldsymbol{B}\boldsymbol{\theta} + \boldsymbol{\epsilon} & \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}) \quad \text{is reformulated as} \\ \boldsymbol{y} & = & \boldsymbol{X}\beta + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\epsilon} & \boldsymbol{u} \sim \mathcal{N}(0, \boldsymbol{G}) \text{ and } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}) \end{array}$$

where $\boldsymbol{G} = \sigma_u^2 \boldsymbol{Q}^{-1}$ is the variance-covariance matrix of the random effects $\boldsymbol{u}$ that depends on σ_u^2 and a matrix $\boldsymbol{Q}$.

Wand (2003) uses truncated power functions for the construction of $\boldsymbol{X}$ and $\boldsymbol{Z}$ where:

$$\boldsymbol{X} = [1, x, x^2, ..., x^p] \text{ and } \boldsymbol{Z} = [(x_i - k_\kappa)_+^p] \quad 1 \leq i \leq n \text{ and } 1 \leq k \leq \kappa$$

Using B-splines we have different alternatives, e.g.:

- Eilers (1999) proposed

$$\boldsymbol{X} = [1, x, x^2, ..., x^p] \text{ and } \boldsymbol{Z} = \boldsymbol{B}\boldsymbol{D}'(\boldsymbol{D}'\boldsymbol{D})^{-1}$$

- Currie and Durbán (2002) proposed the use of the singular value decomposition of the matrix $\boldsymbol{D}'\boldsymbol{D}$, i.e.:

$$\boldsymbol{X} = [1, x, x^2, ..., x^p] \text{ and } \boldsymbol{Z} = \boldsymbol{B}\boldsymbol{U}_s\boldsymbol{\Sigma}_s^{-1/2}$$

  where $\boldsymbol{U}_s$ are the left singular vectors (of the *span*) of the SVD of $\boldsymbol{D}'\boldsymbol{D} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$.

In both reparameterizations we obtain the mixed model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\epsilon} \quad \boldsymbol{u} \sim \mathcal{N}(0, \boldsymbol{G}) \text{ and } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$$

where $\boldsymbol{Q} = \boldsymbol{I}_{c-q}$ and then $\boldsymbol{G} = \sigma_u^2\boldsymbol{I}_{c-q}$ is multiple of a diagonal matrix. c is the number of columns of the original basis $\boldsymbol{B}$ and q is the order of the penalty (usually $q = 2$). Then the smoothing parameter becomes the ratio $\lambda = \sigma^2/\sigma_u^2$.

2.6.1 Estimation of the variance components, and fixed and random effects β and u

In the context of mixed models the standard method for estimating the variance components is *restricted or residual maximum likelihood* (REML)

$$\ell_R(\sigma_u^2, \sigma^2) = -\frac{1}{2}\log|\boldsymbol{V}| - \frac{1}{2}\log|\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X}| - \frac{1}{2}\boldsymbol{y}'(\boldsymbol{V}^{-1} - \boldsymbol{V}^{-1}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{V}^{-1})\boldsymbol{y},$$

where $\boldsymbol{V} = \sigma^2\boldsymbol{I} + \boldsymbol{Z}\boldsymbol{G}\boldsymbol{Z}'$ and

$$\begin{array}{rcl} \boldsymbol{\beta} & = & (\boldsymbol{X}\boldsymbol{V}^{-1}\boldsymbol{X})\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{y} \\ \boldsymbol{u} & = & \sigma_u^2\boldsymbol{Z}'\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) \end{array}$$

with $\boldsymbol{V}^{-1} = 1/\sigma^2(\boldsymbol{I} - \boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z} + (\sigma^2/\sigma_u^2)\boldsymbol{I}_{c-q})^{-1}\boldsymbol{Z}')$.

In the mixed models context, we can consider more complex structures, e.g. correlated data, with $\epsilon \sim \mathcal{N}(0, \sigma^2\boldsymbol{\Omega})$, where $\boldsymbol{\Omega}$ is a correlation matrix (e.g. $AR(1)$). We will discuss this situation later on.

3 Software

In this section we will focus on the use of penalized splines with `R` software. Although many `R` packages are available, `library(mgcv)` has become a very popular package which implements a wide variety of flexible models for smoothing via the function `gam` (for generalized additive models). For the mixed model reparameterization of a smooth model, the function `gamm` (for additive mixed models). However, first we will start introducing the function `spm` in `SemiPar` which uses the mixed model representation with truncated polynomials by Ruppert, Wand, and Carroll (2003).

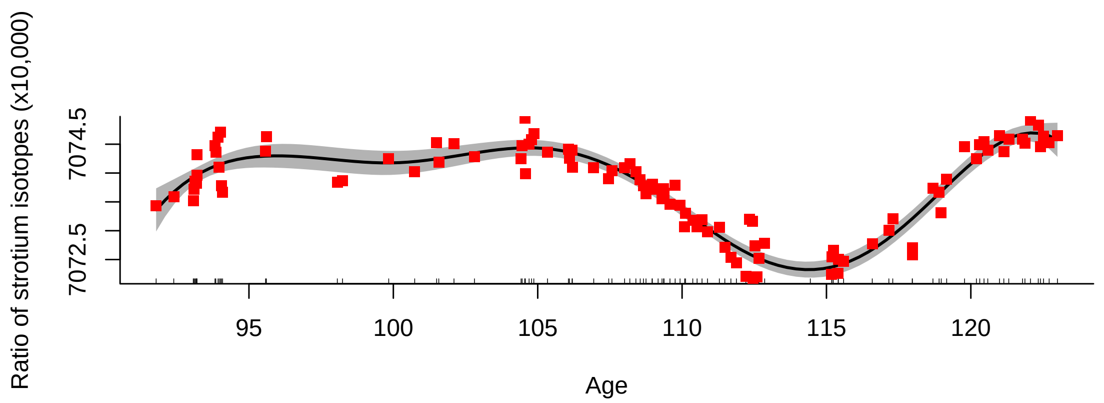


Figure 15: *P-splines fit using the mixed model reparameterization by Ruppert, Wand, and Carroll (2003)*

3.1 spm function in `library(SemiPar)`

For illustration we consider the fossil data described by Chaudhuri and Marron (1999)

```
library(SemiPar)
data(fossil)
attach(fossil)
y <- 10000*strontium.ratio
x <- age
```

We can fit the model with `spm` function, specifyng the smooth term with `f()`, i.e.:

```
fit.spm <- spm(y ~ f(x, basis="trunc.poly", degree=3))
summary(fit.spm)
```

```
##
##
## Summary for non-linear components:
##
##         df  spar knots
## f(x) 8.87 3.419    25
##
## Note this includes 1 df for the intercept.
```

Fitted curve can be simply plotted with

```
plot(fit.spm, xlab="Age", ylab="Ratio of strotium isotopes (x10,000)")
points(x,y,pch=15,col="red") # add points
```

The object fitted by `spm` has three components:

```r
names(fit.spm)
```

```
## [1] "fit"  "info" "aux"
```

- `$fit` containing the same information as a `lme` object

```r
names(fit.spm$fit)
```

```
##  [1] "modelStruct"  "dims"         "contrasts"    "coefficients"
##  [5] "varFix"       "sigma"        "apVar"        "logLik"
##  [9] "numIter"      "groups"       "call"         "terms"
## [13] "method"       "fitted"       "residuals"    "fixDF"
## [17] "na.action"    "data"         "sigma"        "coef"
```

- `$info` with information about the model, the bases, the knots, degree of the polynomial, etc…

```r
names(fit.spm$info)
```

```
## [1] "formula"   "y"         "intercept" "lin"       "pen"       "krige"
## [7] "off.set"   "trans.mat"
```

- `$aux` with the covariance matrix of the fixed and random effects (`$cov.mat`), the estimated variance of the random effects $\hat{\sigma}^2$ (`$random.var`), the residual variance $\hat{\sigma}^2$ (`$error.var`) and the approximated degrees of freedom for each component `$df`.

```r
names(fit.spm$aux)
```

```
## [1] "cov.mat"    "df"         "block.inds" "resid.var"  "random.var"
## [6] "df.fit"     "df.res"     "mins"       "maxs"
```

For more details see Ngo and Wand (2004).

3.2 mgcv package

The main reference for this section is the book by Wood (2006).

3.2.1 The gam function

```r
library(mgcv)
fit.gam <- gam(y ~ s(x))
```

Now we plot the fitted curve with confidence bands

```r
# See ?plot.gam for plotting options
plot(fit.gam,shade=TRUE,seWithMean=TRUE,pch=19,1,cex=.55)
```

The main function for smoothing with `mgcv` is `gam`. For example:

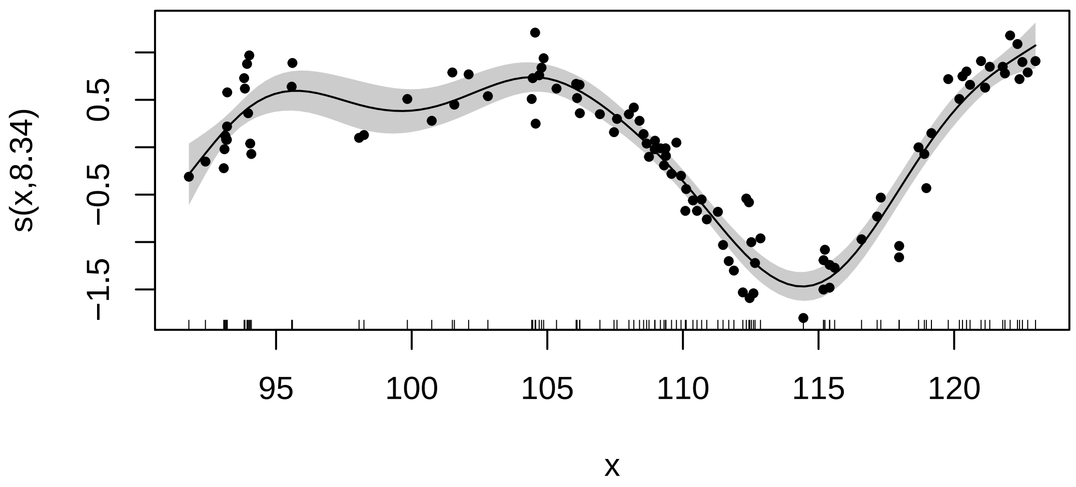


Figure 16: *Smooth fit with `mgcv`'s `gam` function*

```
gam(formula,method="",select="",family=gaussian())
```

The main arguments for this function must be specified as a `formula`:

- The first choice is the basis used to represent the smooth terms `s(x)` (See `?s` or `?smooth.terms`). The default is to use the thin plate splines. The type of basis function can be modified using `bs` argument inside `s(x,bs="ps")`, where `ps` uses P-splines. Other alternatives are describe in the next Table (from Wood (2006), page 226)

`bs`	**Description**
`"tp"`	Thin Plate Regression Splines
`"ts"`	Thin Plate Regression Splines with Shrinkage
`"cr"`	Cubic regression spline
`"crs"`	Cubic regression spline with Shrinkage
`"cc"`	Cyclic cubic regression spline
`"ps"`	P-splines (see `?p.spline`)

- `m` The order of the penalty.
- `k` the dimension of the basis used to represent the smooth term. `k` should not be less than the dimension of the null space of the penalty for the term (the order of the penalty `m`).
- `by` a numeric or factor variable of the same dimension as each covariate.
- `sp` any supplied smoothing parameter for the smooth term.
- `fx` indicates whether the term is a fixed e.d.f. regression spline (`TRUE`) or a penalized regression spline (`FALSE`).
- `id` used to allow different smooths to be forced to use the same basis and smoothing parameter

For more option in function `gam` see `?gam`, i.e. `method` for the selection of the smoothing parameter (in general likelihood-based methods tend to be more robust).

```
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7073.7412     0.0255  277398   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
## s(x) 8.339  8.876 87.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.881   Deviance explained =   89%
## GCV = 0.075588  Scale est. = 0.068928  n = 106
```

gam.check function produces some basic residual plots, and provides some information related to the fitting procces

```
par(mfrow=c(2,2))
gam.check(fit.gam)
```

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 4.638002e-06 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##        k'  edf k-index p-value
## s(x) 9.00 8.34    0.94    0.23
```

Other residual plots should be examined, e.g.:

```
plot(fitted(fit.gam),residuals(fit.gam))
```

As you may have noticed, there are some choices to make, specially the dimension `k` of the basis used to represent the smooth term. The choice of basis dimensions amounts to setting the maximum possible degress of freedom allowed to the smooth term. Usually, the choice of `k` makes sligth (almost negligible) difference as long as enough flexibility is provided (see `?choose.k`). Sometimes the choice of `k` depends on the computational efficiency (size of the basis for large data sets).

3.2.2 The gamm function

`mgcv` library includes the function `gamm` which fits a generalized additive mixed model (GAMM) based on linear mixed models as implemented in the `nmle` library. The function is a wrapper of `lme`

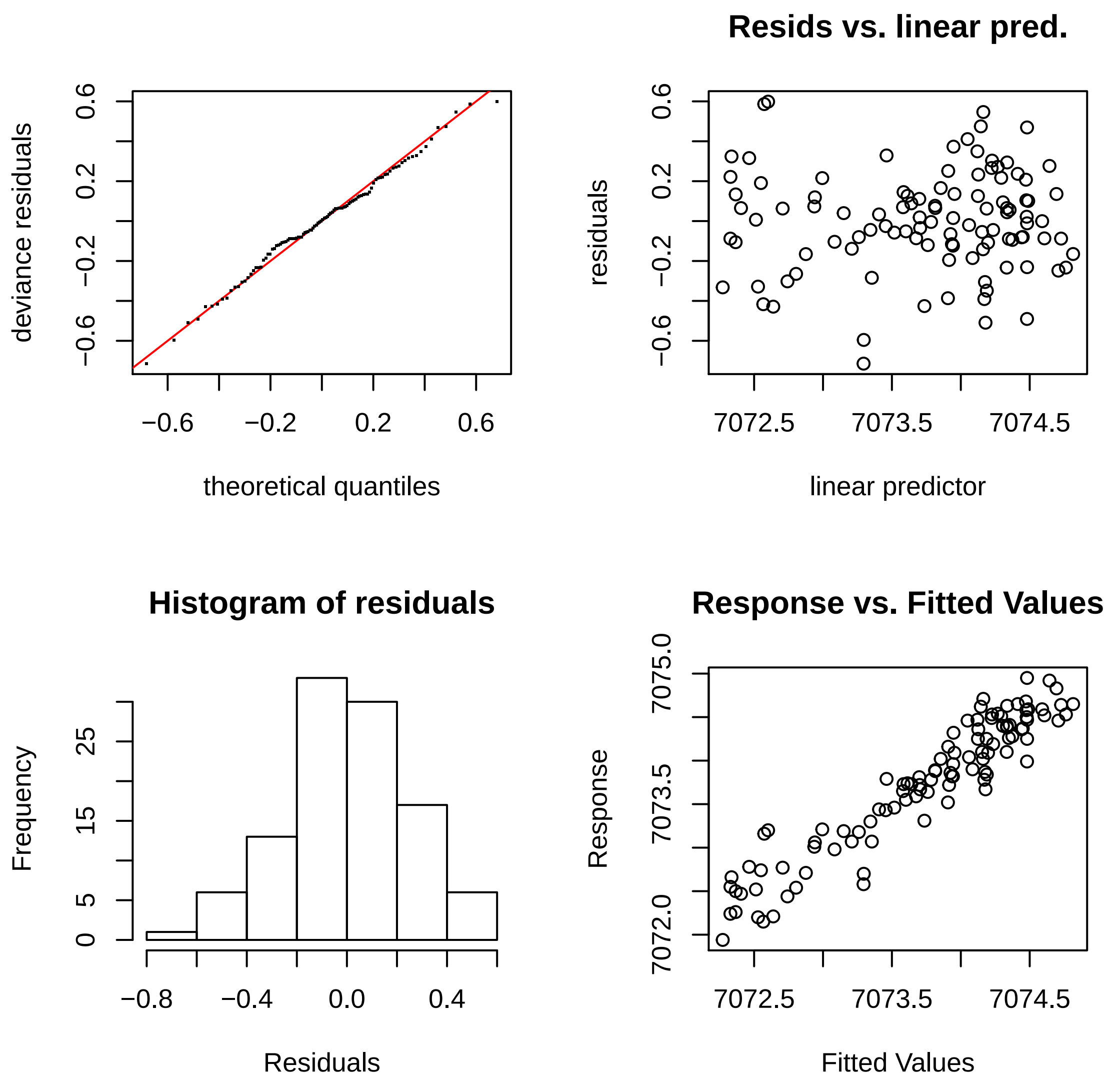


Figure 17: *Diagnostics for fitted gam model with* `gam.check`, *see* `?gam.check` *for details*

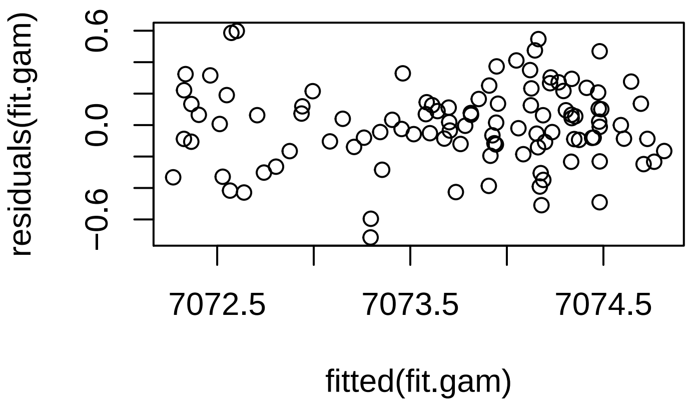


Figure 18: *Residuals plots*

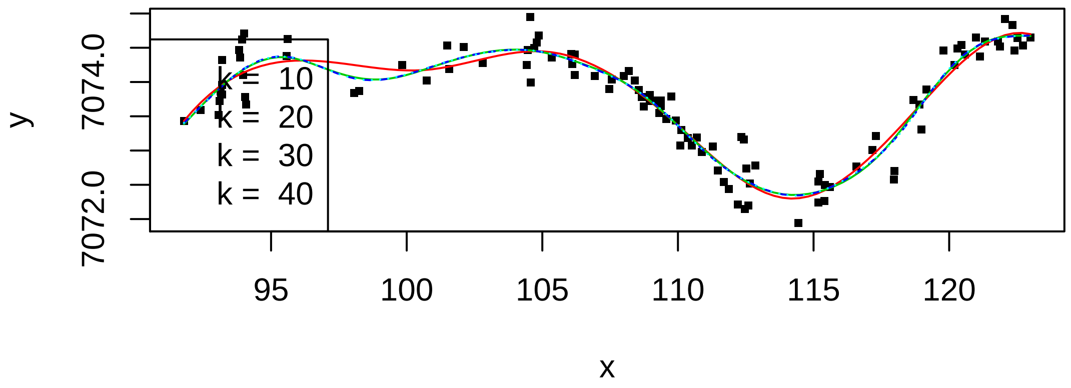


Figure 19: *Different smooth fits for* $k = 10, 20, 30, 40$

and `glmmPQL` in `MASS` library. Its purpose is to perform the reparametrization of the smooth model into a mixed models as shown in Section 2.6.

```
gamm(formula,method="",random=NULL,correlation=NULL,select="",family=gaussian())
```

```
fit.gamm <- gamm(y ~s(x,bs="ps",k=20), method="REML")
```

The fitted object with `gamm` returns a list with two components: `$lme` is the object returned by `lme`; `$gam` is the complete object of class `gam` which can be treated like a `gam` object for prediction, plotting etc . . .

```
summary(fit.gamm$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x, bs = "ps", k = 20)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7073.7412     0.0245  288713   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
## s(x) 10.33  10.33 82.07  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.89
##   Scale est. = 0.063631  n = 106
```

```
summary(fit.gamm$lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: strip.offset(mf)
##        AIC     BIC    logLik
##   56.57453 67.1521 -24.28727
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##               Xr1       Xr2       Xr3       Xr4       Xr5       Xr6
## StdDev: 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468
##                Xr7       Xr8       Xr9      Xr10      Xr11      Xr12
## StdDev: 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468
##               Xr13      Xr14      Xr15      Xr16      Xr17      Xr18
## StdDev: 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468 0.1277468
##          Residual
## StdDev: 0.2522526
##
## Fixed effects: y.0 ~ X - 1
##                   Value Std.Error  DF   t-value p-value
## X(Intercept) 7073.741 0.0245009 104 288713.12  0.0000
## Xs(x)Fx1        0.108 0.3215396 104      0.34  0.7372
##  Correlation:
##          X(Int)
## Xs(x)Fx1 0
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.38374441 -0.59772858  0.02233536  0.60088298  2.41386908
##
## Number of Observations: 106
## Number of Groups: 1
```

Note that the main difference between `fit.gamm` fitted with `gamm` and `fit20.gam` with `gam` is how the smoothing parameter is estimated (REML vs GCV), and hence the results are basically the same.

3.3 `lme` package

We saw that the P-splines has a mixed model representation in Section 2.6. In this section we will see how we can use the function `lme` in `library(nlme)` for smoothing. Note, that this is what `gamm` function is doing internally, but we will show how you can construct all the elements by yourself.

The `R` functions `mixed.model.B` and `mixed.model.T` creates the mixed model bases $\boldsymbol{X}$ and $\boldsymbol{Z}$ based on the methods described in Section 2.6.

First we create the bases

```r
# Create MM bases
nseg = 20
MM <- mixed.model.B(x,min(x)-0.01,max(x)+0.01,ndx=nseg,bdeg=3,pord=2,type="Eilers")
names(MM)
```

```
## [1] "X" "Z"
```

```r
X <- MM$X
Z <- MM$Z
```

In order to use `lme` we need the to provide the information in the correct way:

```
library(nlme)
n <- length(y)

Id <- factor(rep(1,n))  # create an Id factor for each observation (individual)
#  hence there is no nesting

# random effects has a cov matrix multiple of the Identity
Z.block <- list(list(Id=pdIdent(~Z-1)))
Z.block <- unlist(Z.block,recursive=FALSE)

data.fr <- groupedData(y ~ X[,-1] | Id,
                       data=data.frame(y,X,Z))

fit.lme <- lme(y ~ X[,-1],data=data.fr,
               random=Z.block) # method="REML" by default

class(fit.lme)
```

```
## [1] "lme"
```

What can we get from `fit.lme`?

- $\hat{\sigma}^2$

```
sig2 <- fit.lme$sigma^2
```

- $\hat{\sigma}^2_\alpha$

```
sig2.alpha <- sig2*exp(2*unlist(fit.lme$modelStruct))
```

- $REML$ score

```
reml <- fit.lme$logLik
```

- $\hat{\beta}$

```
beta.hat <- fit.lme$coeff$fixed
```

- $\hat{\alpha}$

```
alpha.hat <- unlist(fit.lme$coeff$random)
```

- $\hat{f}$

```
f.hat <- c(X%*%beta.hat + Z%*%alpha.hat[1:ncol(Z)])
d <- ncol(fit.lme$fitted)
f.hat <- fit.lme$fitted[,d]
```

- Confidence Intervals of the fitted curve $\hat{f}$

```
# 1st we create a function to obtain 95% CI's from a fitted lme object
Int.Conf<-function(X,Z,f.hat,sigma2,sigma2.alpha)
{
  C=cbind(X,Z)
  D=diag(c(rep(0,ncol(X)),rep(sigma2/sigma2.alpha,ncol(Z))))
  S=sigma2*rowSums(C%*%solve(t(C)%*%C+D)*C)
  IC1=f.hat-1.96*sqrt(S)
  IC2=f.hat+1.96*sqrt(S)
  IC=cbind(IC1,IC2)
  colnames(IC) <- c("lower","upper")
  IC
}

ci <- Int.Conf(X,Z,f.hat,sig2,sig2.alpha)
```

We can now plot the fitted curve and the CI's

```
plot(x,y,cex=.65,pch=15,col="grey")
lines(x[order(x)],f.hat[order(x)],col=2)
matlines(x[order(x)],ci[order(x)],col=4,lty=2)
```

Or on a finer grid

```
# Predict
x.grid <- seq(min(x),max(x),l=200)
MM.grid <- mixed.model.B(x.grid,min(x.grid)-0.01,max(x.grid)+0.01,
                      ndx=nseg,bdeg=3,pord=2,type="Eilers")
X.grid <- MM.grid$X
Z.grid <- MM.grid$Z

# Compute the fitted curve on a finer grid of x points
f.hat.grid <- c(X.grid%*%beta.hat+Z.grid%*%alpha.hat)

ci.grid <- Int.Conf(X.grid,Z.grid,f.hat.grid,sig2,sig2.alpha)
plot(x,y,cex=.65,col="grey")
lines(x.grid,f.hat.grid,col=2)
matlines(x.grid,ci.grid,lty=2,col=4)
```

We can use `gam` with GLM's by specifying the argument `family` as in the `glm` function. For instance:

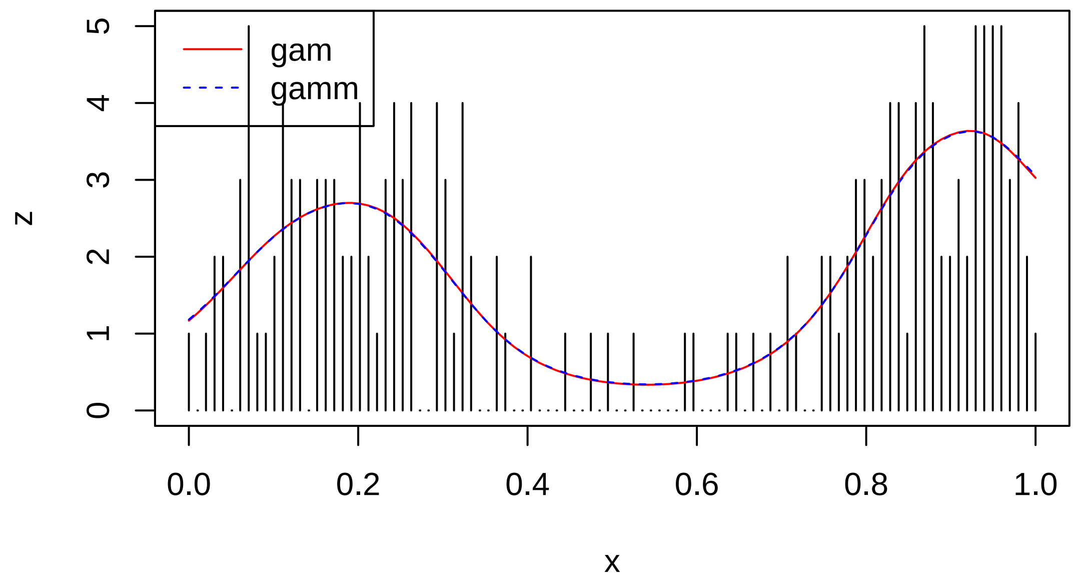


Figure 20: *Simulated example with Poisson data*

```
library(mgcv)
# Poisson example
n = 100
x <- seq(0,1,l=n)
f <- sin(2.8*x*pi)
g <- exp(f)
z <- rpois(rep(1,n),g) # simulate a Poisson response vble with mean exp(f)

# gam fit
fit.pois <- gam(z ~ s(x,bs="ps",m=2,k=40),family=poisson,
                method="REML")

# gamm fit (using PQL)
fit.pois2 <- gamm(z ~ s(x,bs="ps",m=2,k=40),family=poisson,
                  method="REML")
```

```
##
##  Maximum number of PQL iterations:  20
```

As mentioned before,

```r
library(MASS)

# Create MM bases
nseg = 20
MM <- mixed.model.B(x,min(x)-0.01,max(x)+0.01,
                    ndx=nseg,bdeg=3,pord=2,type="Eilers")
names(MM)
X <- MM$X
Z <- MM$Z
n <- length(z)

# create Id factor and Z.block as shown in lme
Id <- factor(rep(1,n))  # create an Id factor for each observation (individual)
#  hence there is no nesting

Z.block <- list(list(Id=pdIdent(~Z-1)))
Z.block <- unlist(Z.block,recursive=FALSE)

data.fr <- groupedData(z ~ X[,-1] | Id,
                       data=data.frame(z,X,Z))

# fit with glmmPQL function in library(MASS)

fit.pois.glmmPQL <- glmmPQL(z ~ X[,-1],
                            data=data.fr, random=Z.block, family=poisson)
```

4 Applications

In this section we will present some examples using the smoothing techniques presented previously applied to some real examples.

4.1 Additive models

4.1.1 Air quality data

In this Section, we analyze the `data(airquality)` (see `?airquality`) consisting of air quality measurements in New York, from Maty to September 1973. The data frame contains with 154 observations on 6 variables.

- `[,1]` `Ozone` numeric Ozone (ppb)
- `[,2]` `Solar.R` numeric Solar R (lang)
- `[,3]` `Wind` numeric Wind (mph)
- `[,4]` `Temp` numeric Temperature (degrees F)
- `[,5]` `Month` numeric Month (1–12)

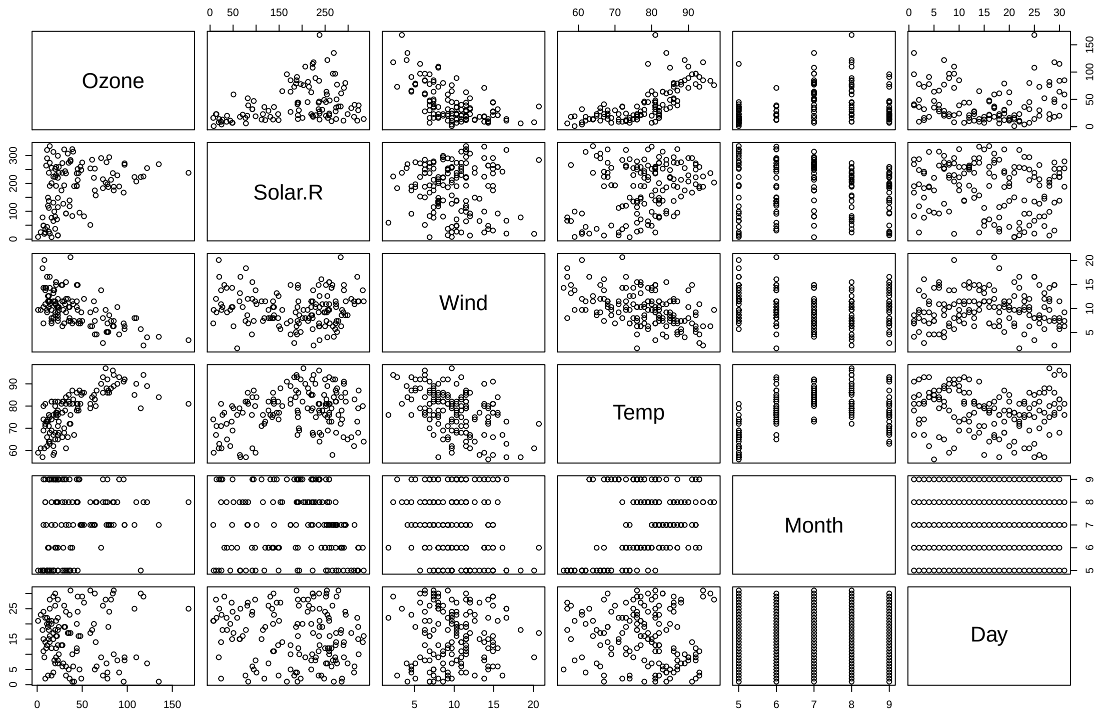


Figure 21: *Scatterplot matrix*

- `[,6]` `Day` numeric Day of month (1–31)

For more details `?airquality`

```
data(airquality)
pairs(airquality)
```

A simple scatterplot shows that some of the variables have a non-linear relationship. Let us consider a linear model for the response variable `Ozone`

```
airq.lm <- lm(Ozone ~ Temp + Wind + Solar.R, data=airquality)
summary(airq.lm)
```

```
##
## Call:
## lm(formula = Ozone ~ Temp + Wind + Solar.R, data = airquality)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.485 -14.219  -3.551  10.097  95.619
```

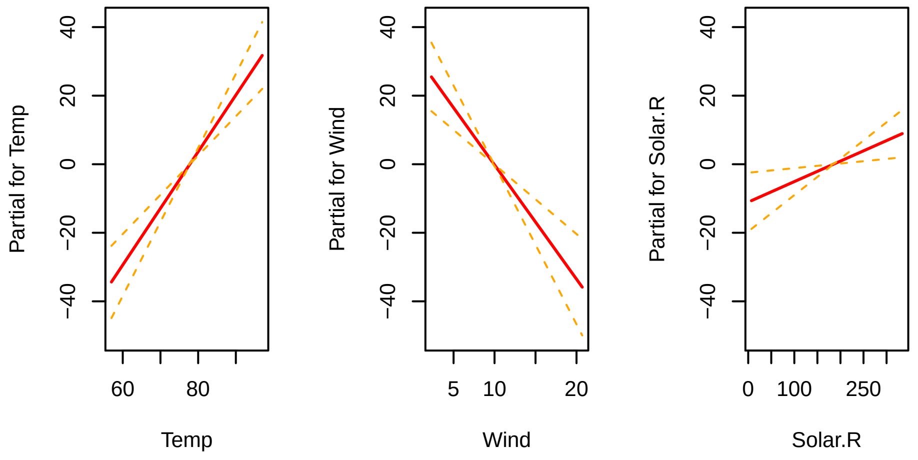


Figure 22: *Plots of `lm` fit*

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -64.34208   23.05472  -2.791  0.00623 **
## Temp          1.65209    0.25353   6.516 2.42e-09 ***
## Wind         -3.33359    0.65441  -5.094 1.52e-06 ***
## Solar.R       0.05982    0.02319   2.580  0.01124 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.18 on 107 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.5948
## F-statistic: 54.83 on 3 and 107 DF,  p-value: < 2.2e-16
```

Let us plot the results

```
par(mfrow=c(1,3))
termplot(airq.lm,se=TRUE)
```

We can also plot the residuals of the model in order to check the model hypothesis

```
par(mfrow=c(1,2))
plot(airq.lm,which=1:2)
```

The lack of normality in the residuals is due to the asymmetry of the response variable `Ozone`, we can apply a `log` transform to achieve asymmetry, i.e.:

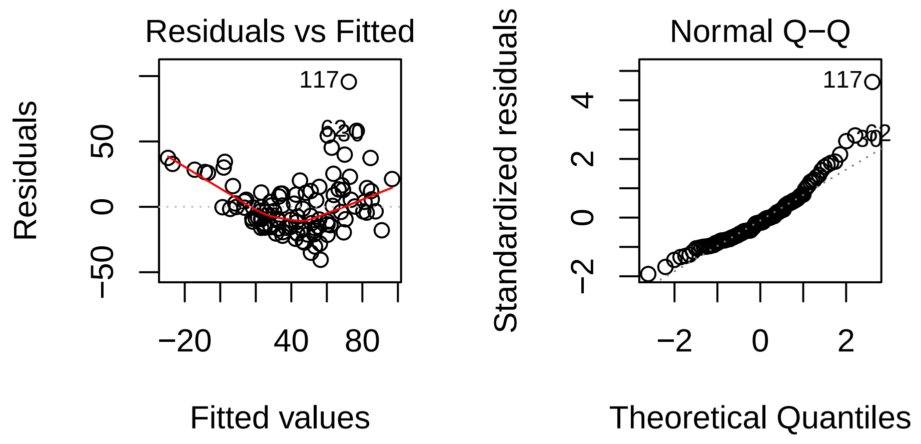


Figure 23: *Plots of* `lm` *residuals*

```
par(mfrow=c(1,2))
hist(airquality$Ozone, main="Ozone")
hist(log(airquality$Ozone), main ="log(Ozone)")
```

We can now fit a linear model of the $\log(Ozone)$, does the model look more adecuate?

```
lairq.lm <- lm(log(Ozone)~ Temp + Wind + Solar.R, data=airquality)
summary(lairq.lm)
plot(lairq.lm)
```

Fitting a linear model we can conclude that there might be some heterokedasticity not accounted by the model. However, the most possible cause is that the relationship between the response variable and the covariates are far from linear.

Let us fit a GAM model, firstly with a single variable, i.e,:

```
library(mgcv)
airq.gam1 <- gam(log(Ozone) ~ s(Wind,bs="ps",m=2,k=10),
                 method="REML", select=TRUE,data=airquality)
summary(airq.gam1)
```

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

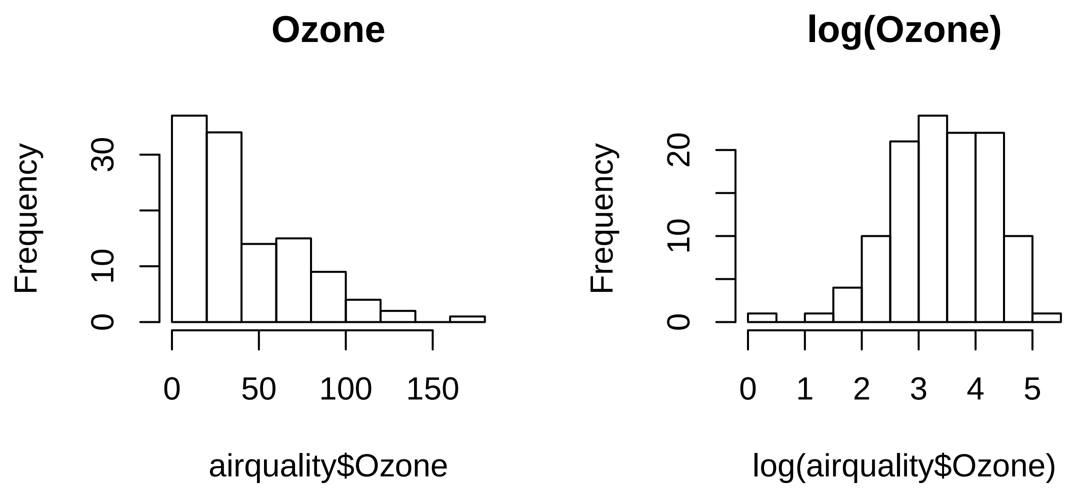


Figure 24: *histograms of `Ozone` and `log(Ozone)`*

```
## log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.4185     0.0655   52.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F  p-value
## s(Wind) 2.565      9 6.453 2.76e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.336   Deviance explained =   35%
## -REML = 128.69  Scale est. = 0.49769   n = 116
```

The results show that the smooth effect `s(Wind)` is significative (with an approximate p-value close to zero and `edf` 2.56). The next Figure shows the smooth wind effect. The increment of the wind speed decreases the log Ozone levels (dramatically until 10mph). Note that including the intercept, the smooth effects are centered at zero.

```
plot(airq.gam1,residuals=TRUE,scheme=1)
```

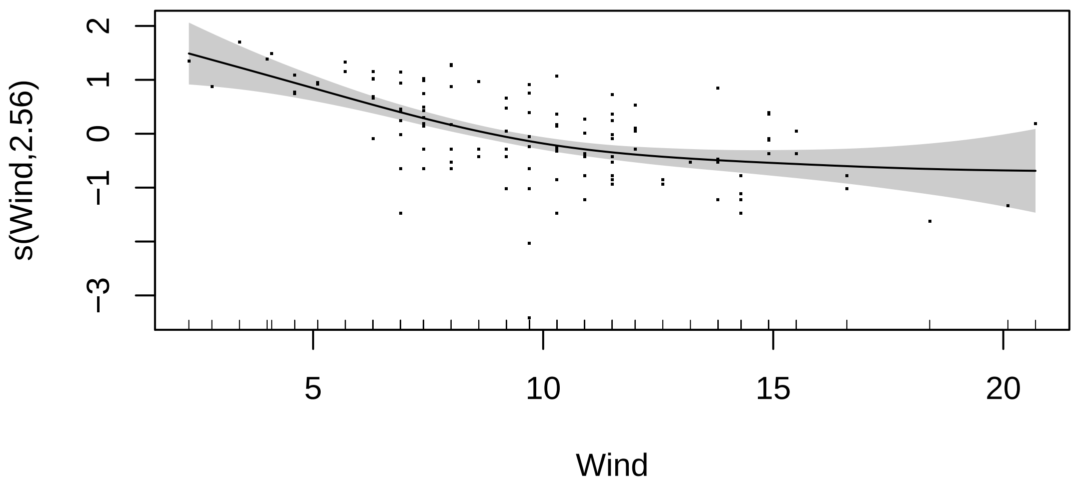


Figure 25: *Estimated smooth effect of `Wind` on `log(Ozone)`. Data points are the residuals*

```
gam.check(airq.gam1)
```

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-1.455961e-06,1.183099e-06]
## (score 128.6926 & scale 0.4976891).
## Hessian positive definite, eigenvalue range [0.4379927,57.51548].
## Model rank =  10 / 10
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'  edf k-index p-value
## s(Wind) 9.00 2.56    0.69   0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that, in model `airq.gam1` we used `k=10` knots, the variable `Wind` has 31 unique values, and hence 10 knots should be enough. We fit a model for the residuals of the fitted gam model

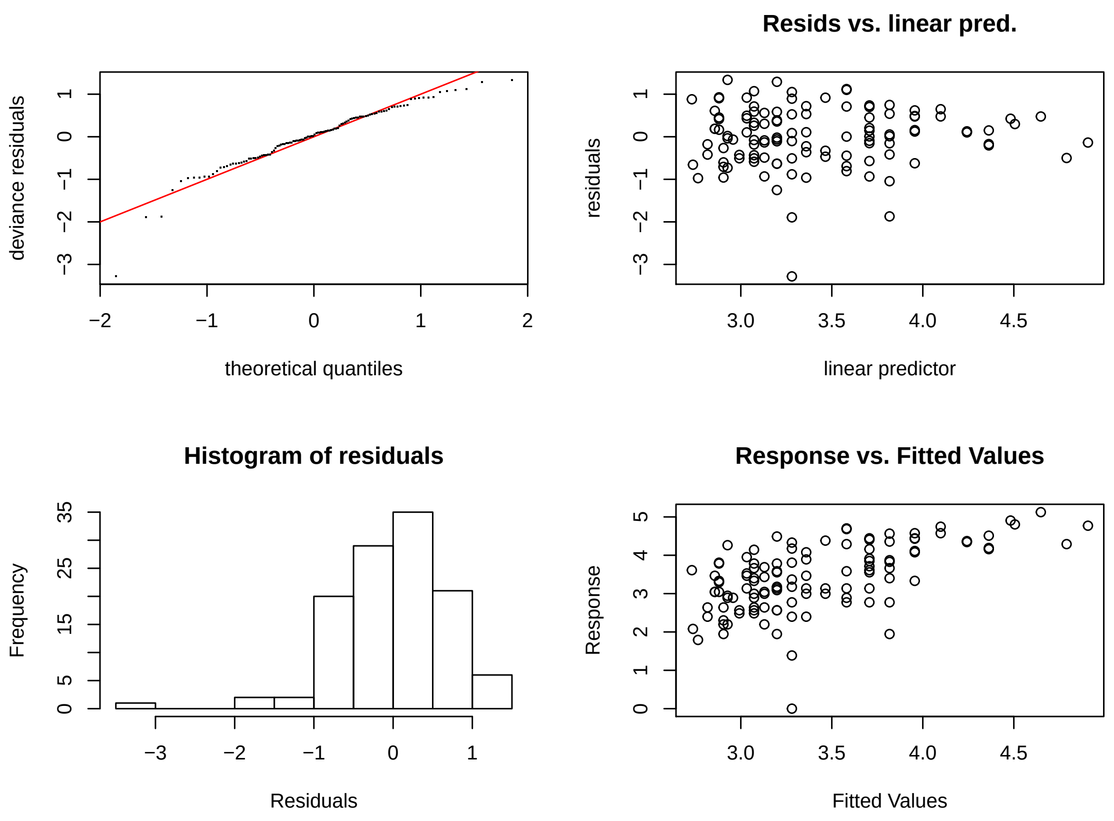


Figure 26: *Check plots by* `gam.check`

```r
resids <- residuals(airq.gam1)
resids.gam <- gam(resids~s(Wind,k=20,m=2),method="REML",
                  select=TRUE,data=airq.gam1$model)
summary(resids.gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## resids ~ s(Wind, k = 20, m = 2)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.137e-15  6.477e-02       0        1
##
## Approximate significance of smooth terms:
##               edf Ref.df F p-value
## s(Wind) 7.419e-05     19 0       1
##
## R-sq.(adj) =  -5.66e-07   Deviance explained = 7.89e-06%
## -REML = 124.14  Scale est. = 0.48659    n = 116
```

The results show that there is no unexplained variability between the variable and the residuals. Hence, we can conclude that we do not need more knots. The next Figure supports this statement.

```r
plot(resids.gam)
```

```r
airq.pred <- data.frame(Wind=seq(min(airquality$Wind),max(airquality$Wind)),
                        length.out=200)
p <- predict(airq.gam1, newdata = airq.pred, type="response", se.fit = TRUE)

plot(airq.pred$Wind,p$fit, xlab="Wind", ylab="log(Ozone)",
     type="l", ylim=c(0,6))
lines(airq.pred$Wind,p$fit + 1.96 * p$se.fit, lty=2)
lines(airq.pred$Wind,p$fit - 1.96 * p$se.fit, lty=2)
points(airquality$Wind,log(airquality$Ozone),cex=.55,col="grey", pch=15)
```

Now we add the variable `Temp`:

```r
airq.gam2 <- gam(log(Ozone)~s(Wind,bs="ps",m=2,k=10)+s(Temp,bs="ps",m=2,k=10),
                 method="REML",select=TRUE,data=airquality)
summary(airq.gam2)
```

```
##
## Family: gaussian
```

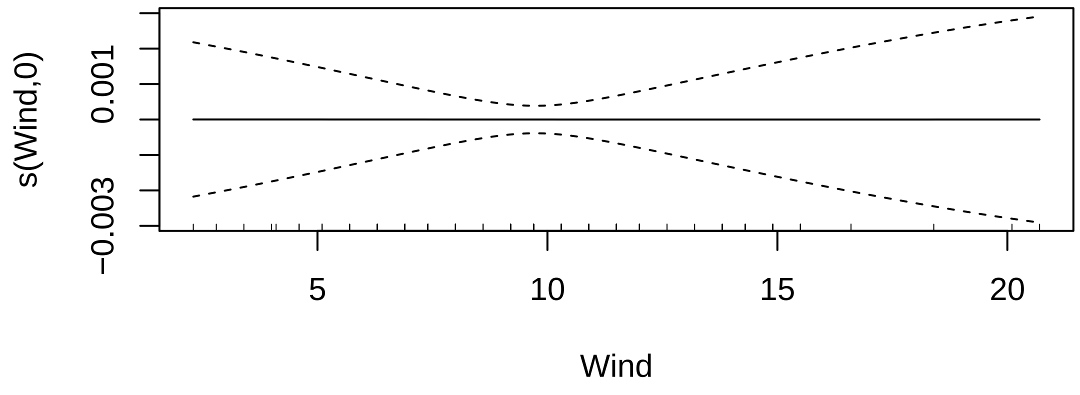


Figure 27: *Wind effect vs the residuals of* `airq.gam1`

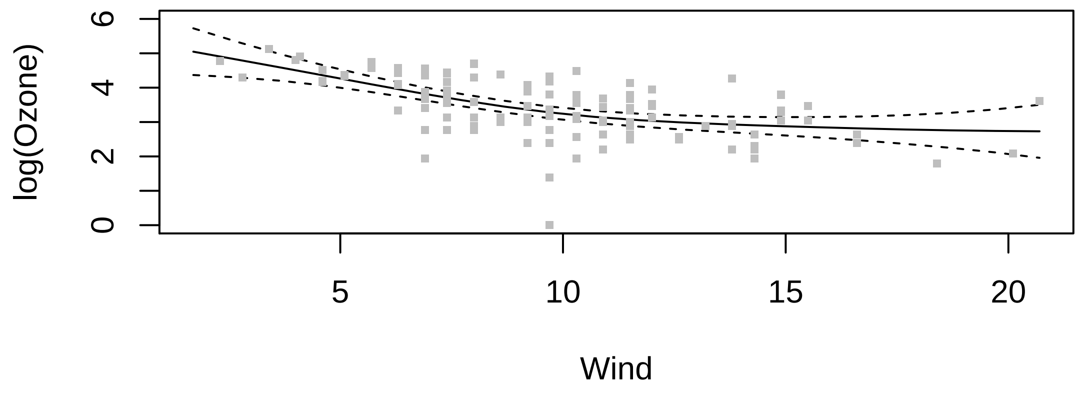


Figure 28: *Predicted curve and CI's*

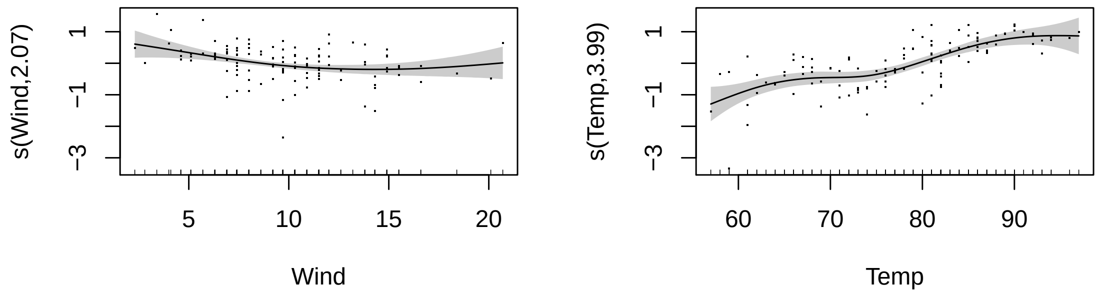


Figure 29: *Estimated smooth effect of `Wind` and `Temp` on `log(Ozone)`. Data points are the residuals*

```
## Link function: identity
##
## Formula:
## log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10) + s(Temp, bs = "ps",
##     m = 2, k = 10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.41852    0.04963   68.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df      F  p-value
## s(Wind) 2.068      9  1.353 0.000981 ***
## s(Temp) 3.990      9 10.496  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.619   Deviance explained = 63.9%
## -REML = 101.64  Scale est. = 0.28568   n = 116
```

Hence, the `Temp` effect is significative. The next Figure show both smooth effects.

```r
par(mfrow=c(1,2))
plot(airq.gam2,residuals=TRUE,scheme=1)
```

We can compare both models using the function `anova` for a F-test:

```
anova(airq.gam1,airq.gam2)
```

```
## Analysis of Deviance Table
##
## Model 1: log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10)
## Model 2: log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10) + s(Temp, bs = "ps",
##     m = 2, k = 10)
##   Resid. Df Resid. Dev     Df Deviance
## 1    111.16     55.958
## 2    105.98     31.123 5.1832   24.835
```

We can conclude that `Temp` variable is relevant. Note that, strictly, in this case both models are not nested, as the effective dimension of the variable `Wind` is different when the variable `Temp` is present. Hence, we use the AIC to confirm the results.

```
AIC(airq.gam1)
```

```
## [1] 255.0315
```

```
AIC(airq.gam2)
```

```
## [1] 195.6561
```

Now, we include the covariate `Solar.R`. Notice that this variables contains missing values (`NA`'s).

```
sum(is.na(airquality$Solar.R))
```

```
## [1] 7
```

In order to compare the previous model `airq.gam2` and the new model that includes `Solar.R` we will use the same data, so we will omit the missing values and refit the models.

```
new.airquality <- na.omit(airquality)

airq.gam22=gam(log(Ozone)~s(Wind,bs="ps",m=2,k=10)+s(Temp,bs="ps",m=2,k=10),
               method="REML",select=TRUE,data=new.airquality)

airq.gam3=gam(log(Ozone)~s(Wind,bs="ps",m=2,k=10)+s(Temp,bs="ps",m=2,k=10)+s(Solar.R,bs="ps",m=2,k=20),
              method="REML",select=TRUE,data=new.airquality)

summary(airq.gam22)
```

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

```
## log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10) + s(Temp, bs = "ps",
##     m = 2, k = 10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.41593    0.05128   66.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F  p-value
## s(Wind) 2.1879      9 1.919    1e-04 ***
## s(Temp) 0.9874      9 8.601 7.73e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.611   Deviance explained = 62.2%
## -REML = 95.215  Scale est. = 0.29194   n = 111
```

```r
summary(airq.gam3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(Ozone) ~ s(Wind, bs = "ps", m = 2, k = 10) + s(Temp, bs = "ps",
##     m = 2, k = 10) + s(Solar.R, bs = "ps", m = 2, k = 20)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.41593    0.04586   74.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F  p-value
## s(Wind)    2.318      9 2.255 2.44e-05 ***
## s(Temp)    1.852      9 6.128 1.12e-12 ***
## s(Solar.R) 2.145     19 1.397 1.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.689   Deviance explained = 70.7%
## -REML = 86.106  Scale est. = 0.23342   n = 111
```

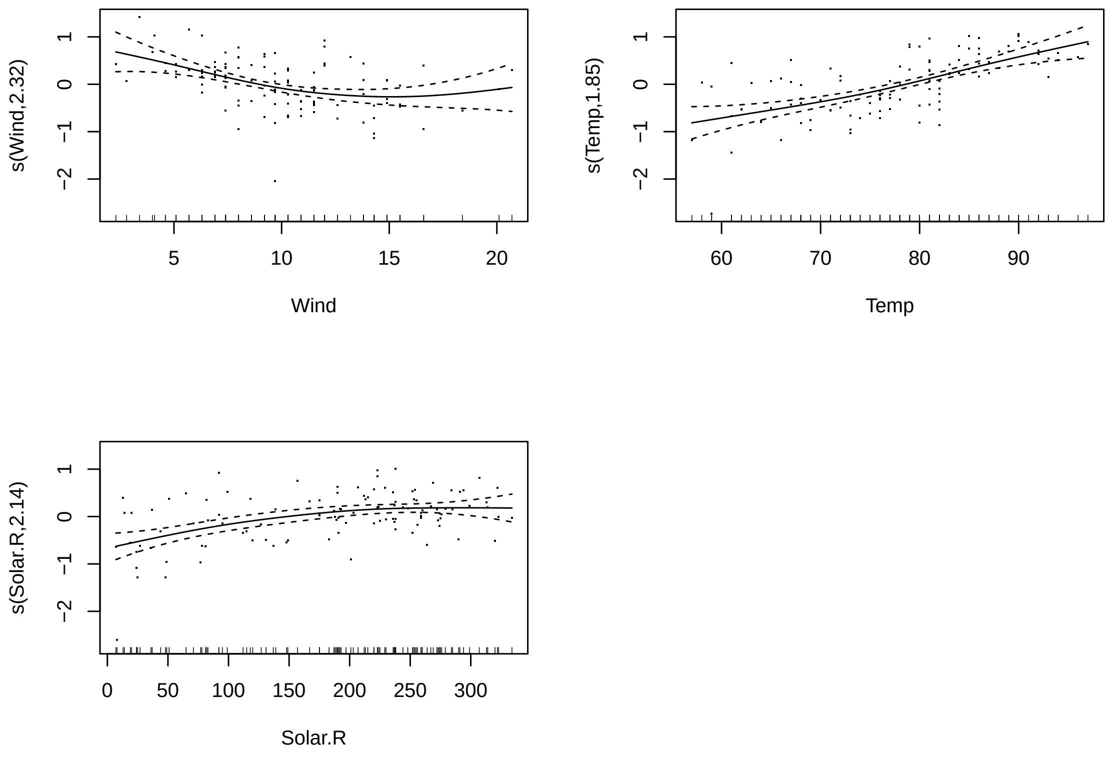


Figure 30: *Smooth effects of `Wind`, `Temp` and `Solar.R`*

```r
AIC(airq.gam22)
```

```
## [1] 185.7769
```

```r
AIC(airq.gam3)
```

```
## [1] 166.0423
```

Is the `Solar.R` variable relevant?

The next Figure shows the estimated smooth effects for `Wind`, `Temp` and `Solar.R` covariables (partial residuals are also plotted).

```r
par(mfrow=c(2,2))
plot(airq.gam3,residuals=TRUE)
```

Variable selection in GAMs

The function `gamSim` simulates several data for GAMs. See `?gamSim` for details.

```
set.seed(666)
data=gamSim(eg=1,n=400,dist="normal")
```

```
## Gu & Wahba 4 term additive model
```

```
fit=gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=data,method="REML",select=TRUE)
summary(fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x0) + s(x1) + s(x2) + s(x3)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.81866    0.09661   80.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df      F  p-value
## s(x0) 2.588e+00      9  2.633 5.67e-06 ***
## s(x1) 2.436e+00      9 39.163  < 2e-16 ***
## s(x2) 7.735e+00      9 87.972  < 2e-16 ***
## s(x3) 6.087e-05      9  0.000    0.999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.743   Deviance explained = 75.1%
## -REML = 856.23  Scale est. = 3.7337    n = 400
```

```
par(mfrow=c(2,2))
plot(fit,residuals=TRUE)
```

The `summary(fit)` table shows the estimated degrees of freedom (`edf`) and the approximated F-test *p*-values. Clearly, the `s(x3)` effect is not significant as we can also see in the estimated effect plot.

4.2 Semi-parametric models

Semi-parametric models allows mixtures of linear (parametric) and non-parametric components, i.e.:

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x}_1 + \ldots + \beta_{j-1} \boldsymbol{x}_{j-1} + f(\boldsymbol{x}_j) + \boldsymbol{\epsilon}$$

The fitting procedure is the same as shown in previous sections. Simply the parametric effects are included in the fixed effects part, $\boldsymbol{X}$. An interested case is then the parametric part includes

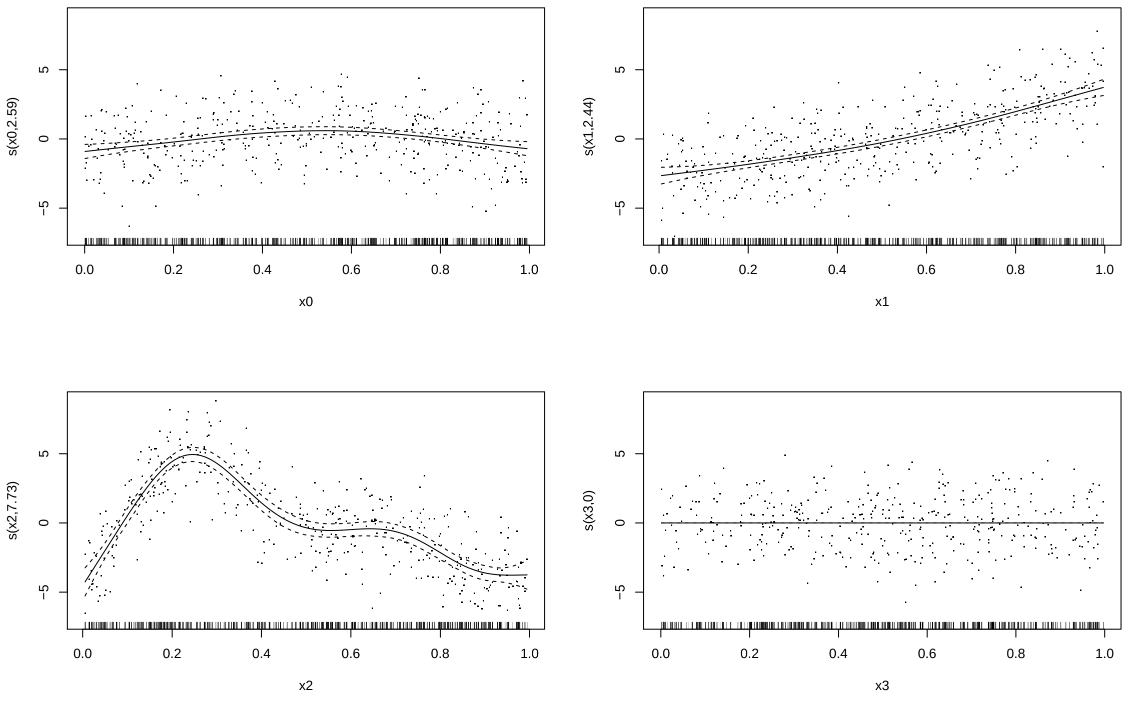


Figure 31: *Estimated smooth effects.*

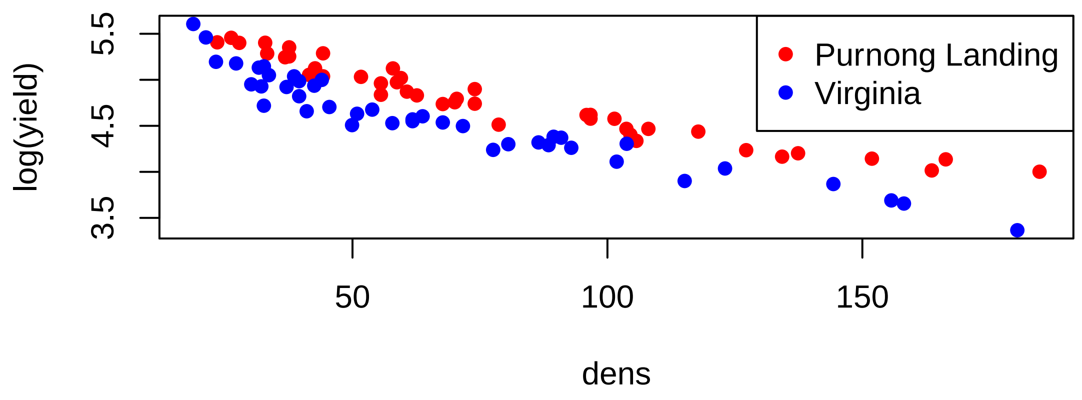


Figure 32: *Onions yield in two locations*

a factor variable with two or more levels. As we saw in the linear case, we can consider different situations: parallel smooth terms (additive effect) or non-parallel smooth terms (interaction effect). Moreover, we can consider the same amount of smoothing or different amount.

4.2.1 Onions data

To illustrate a simple case of a semi-parametric model, we consider the `data(onions)` in the `library(SemiPar)`. The data consist of 84 observations form an experiment involving the production of white Spanish onions in two South Australian locations. The variables are:

- `dens` areal density of plants (plants per square metre)
- `yield` onion yield (grammes per plant).
- `location` indicator of location: `0=Purnong Landing`, `1=Virginia`.

The next Figure shows that for higher density of plants the Purnong Landing yield of onios is greater.

```
library(SemiPar)
data(onions)
attach(onions)
points.cols <- c("red","blue")
plot(dens,log(yield),col=points.cols[location+1],pch=16)
legend("topright",c("Purnong Landing","Virginia"),col=points.cols,pch=rep(16,2))
```

The linear model will be

$$\log(\text{yield}_i) = \beta_0 + \beta_1 \text{location}_{ij} + \beta_2 \text{dens}_i + \epsilon_i$$

where

$$\text{location}_{ij} = \begin{cases} 0 & \text{if the } i\text{th observation is from Purnong Landing} \\ 1 & \text{if the } i\text{th observation is from Virginia} \end{cases}$$

The figure suggest a non-linear terms for `dens`, hence a better model would be:

$$\log(\text{yield}_i) = \beta_0 + \beta_1 \text{location}_{ij} + f(\text{dens}_i) + \epsilon_i$$

```
# create a factor for location
L <- factor(location)

levels(L) <- c("Purnong Landing","Virginia")

# fit a gam with location factor
fit1 <- gam(log(yield) ~ L + s(dens,k=20,m=2,bs="ps"),
            method="REML", select=TRUE)

summary(fit1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(yield) ~ L + s(dens, k = 20, m = 2, bs = "ps")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.85011    0.01688  287.39   <2e-16 ***
## LVirginia   -0.33284    0.02409  -13.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(dens) 4.568     19 72.76  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.946   Deviance explained = 94.9%
## -REML = -54.242  Scale est. = 0.011737  n = 84
```

The next Figure shows the fitted curves for each location with model `fit1`, we assumed both curves are parallel.

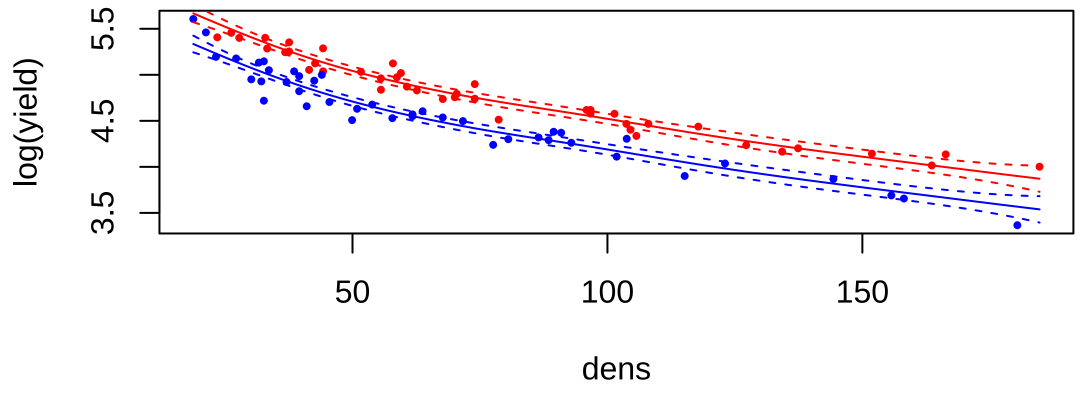


Figure 33: *Fitted curves for each location*

Assuming parallel curves for both locations implies that the decrease of the yield as the density increases is the same in both locations. Instead of fitting an additive effect model we can fit an interaction effect model such as:

$$\log(\text{yield}_i) = \beta_0 + \beta_1\beta_1\text{location}_{ij} + f(\text{dens}_i)_{L(j)} + \epsilon_i$$

where

$$L(j) = \begin{cases} 0 & \text{if the } i\text{th observation is from Purnong Landing} \\ 1 & \text{if the } i\text{th observation is from Virginia} \end{cases}$$

```
fit2 <- gam(log(yield) ~ L + s(dens,k=20,m=2,bs="ps",by=L),
            method="REML", select=TRUE)
summary(fit2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(yield) ~ L + s(dens, k = 20, m = 2, bs = "ps", by = L)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.84407    0.01603  302.12   <2e-16 ***
## LVirginia   -0.33003    0.02271  -14.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                              edf Ref.df     F p-value
## s(dens):LPurnong Landing 3.097     18 37.62  <2e-16 ***
## s(dens):LVirginia        4.728     17 52.10  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.952   Deviance explained = 95.7%
## -REML = -53.616  Scale est. = 0.01045   n = 84
```

Which model is better?

```
AIC(fit1)
```

```
## [1] -125.2307
```

```
AIC(fit2)
```

```
## [1] -131.1844
```

```
fit1$sp
```

```
##     s(dens)1     s(dens)2
## 164.66124239   0.00147493
```

```
fit2$sp
```

```
## s(dens):LPurnong Landing1 s(dens):LPurnong Landing2
##              4.264039e+02              1.604648e-03
##        s(dens):LVirginia1        s(dens):LVirginia2
##              6.437458e+01              1.053443e-03
```

```
# plot the smooth effects
par(mfrow=c(2,2))
plot(fit2, se=TRUE)

# In the same plot
fit2.P <- predict(fit2,newdata=data.frame(L=L.P,dens=dens.g),se.fit=TRUE)
fit2.V <- predict(fit2,newdata=data.frame(L=L.V,dens=dens.g),se.fit=TRUE)

plot(dens,log(yield),col=points.cols[location+1],pch=16,cex=.55)
lines(dens.g,fit2.P$fit,col=2)
lines(dens.g,fit2.P$fit+1.96*fit1.P$se.fit,col=2,lty=2)
```

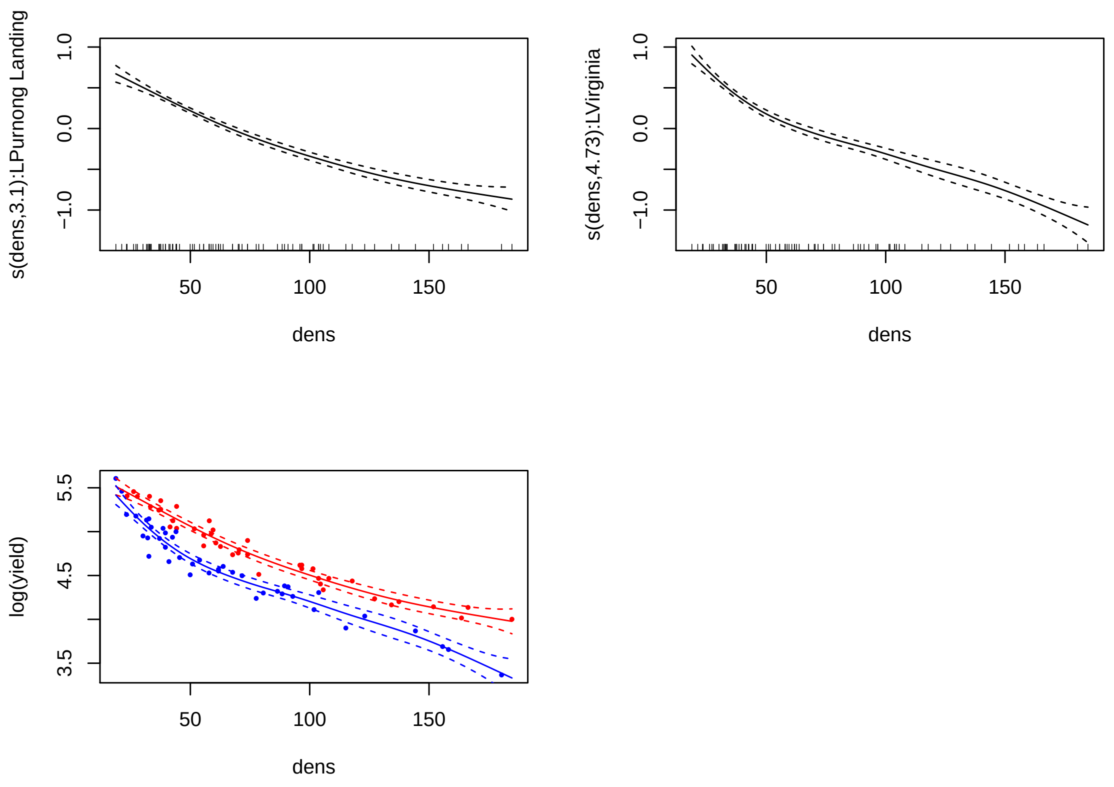


Figure 34: *Fitted curves by location*

```
lines(dens.g,fit2.P$fit-1.96*fit1.P$se.fit,col=2,lty=2)
lines(dens.g,fit2.V$fit,col=4)
lines(dens.g,fit2.V$fit+1.96*fit2.V$se.fit,col=4,lty=2)
lines(dens.g,fit2.V$fit-1.96*fit2.V$se.fit,col=4,lty=2)
```

4.3 Penalized splines for longitudinal data

In this section we will consider longitudinal data in which the individuals trajectories are modelled as a non-linear function of time.

4.3.1 Girls leukemia data

We will use data from a longitudinal study related to different therapies on girls. The data where slightly modified to preserve individuals privacy.

The variables are

- `case` individual
- `treatment` type of treatment received (`1=no radiation`, `2=conventional radiation` or `3=hyperfractioned radiation`)
- `height` girl height in cm
- `age` age in years

The total number of observations is 1988. The number of measurement lies between 1 and 21.

4.3.1.1 Random intercept model

The simplest model is of the form:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \sum_{l=2}^{L} \gamma_l \text{tr}_{il} + U_i + \epsilon_{ij} \quad U_i \sim \mathcal{N}(0, \sigma_U^2) \quad \epsilon_{ij} \sim N(0, \sigma_\epsilon) \quad \begin{matrix} 1 \leq i \leq 197 \\ 1 \leq j \leq n_i \end{matrix}$$

where

$$\text{tr}_{il} = \begin{cases} 1 \text{ if the } i\text{th girl received treatment } l \\ 0 \text{ otherwise} \end{cases}$$

This model assumes that all girls have the same linear growth rate and the variability among girls are accounted by the random effect U_i. In matrix form:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon} \tag{4.1}$$

where

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_1 & \boldsymbol{T}_1 \\ \vdots & \vdots \\ \boldsymbol{X}_m & \boldsymbol{T}_m \end{pmatrix}, \quad \boldsymbol{X}_i = \begin{pmatrix} 1 & x_{i1} \\ \vdots & \vdots \\ 1 & x_{in_i} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix}, \quad \boldsymbol{Z} = \begin{pmatrix} \mathbf{1}_1 & 0 & \dots & 0 \\ 0 & \mathbf{1}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{1}_m \end{pmatrix}, \quad \mathbf{1}_i = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{n_i \times 1}$$

$\boldsymbol{T}_i$ is a matrix that indicates if the ith girls receives treatment $1, 2$ or 3.

The `R` code is

```r
# read data
leukemia <- read.table("GAMs-data/leukemia.txt", header=TRUE)
head(leukemia)
```

```
##   case treatment height      age
## 1    1         3  105.5 4.708333
## 2    1         3  106.5 5.125000
## 3    1         3  108.0 5.708333
## 4    1         3  111.5 6.125000
## 5    1         3  113.5 6.708333
## 6    1         3  115.0 7.208333
```

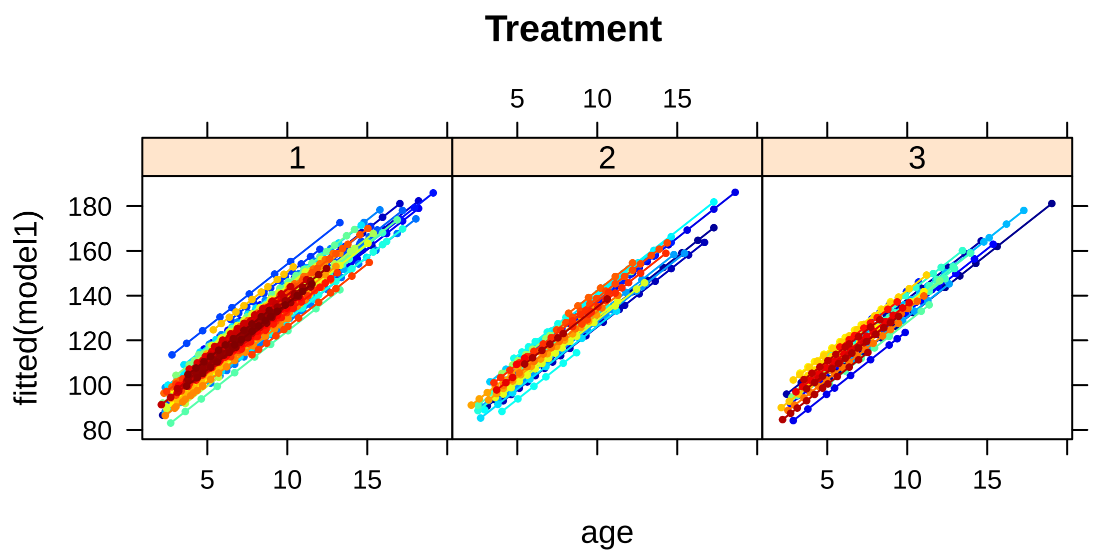


Figure 35: *Random intercept fitted model per treatments*

```
# random intercept model
model1 <- lme(height~factor(treatment)+age,random=list(case=pdIdent(~1)),
              data=leukemia)
fit.model1=model1$fitted
```

The next Figure shows the fitted straight lines for the random intercept model. This model is not able to describe the trajectories of the girls' height.

```
## Warning: package 'fields' was built under R version 3.4.4

## Warning: package 'spam' was built under R version 3.4.4

## Warning: package 'dotCall64' was built under R version 3.4.4

## Warning: package 'maps' was built under R version 3.4.4
```

4.3.1.2 Additive mixed model

A natural extension of the previous model is

$$y_{ij} = \sum_{l=2}^{L} \gamma_l \mathrm{tr}_{il} + f(x_{ij}) + U_i + \epsilon_{ij} \quad U_i \sim \mathcal{N}(0, \sigma_U^2) \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma_\epsilon) \tag{4.2}$$

where f is a smooth function accounting for the growth trend of the girls. The function f is estimated with P-splines (using the mixed model representation). Now,

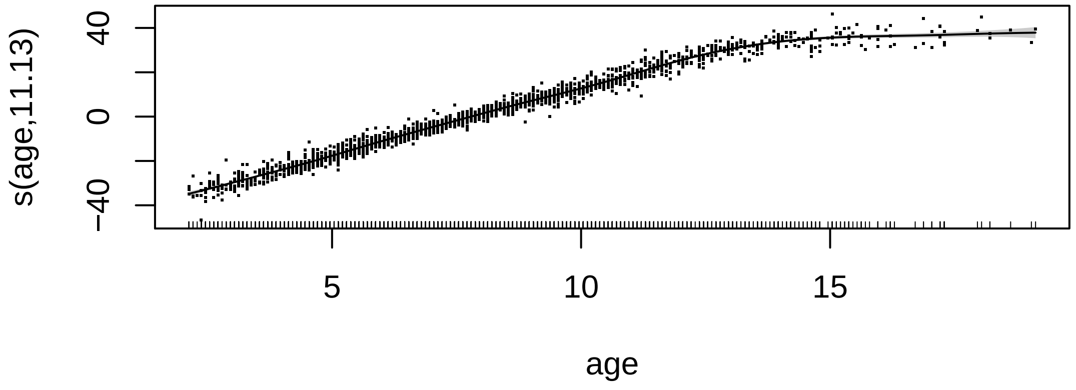


Figure 36: *Smooth effect for the random intercept model*

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{Z}_1 & \mathbf{1}_1 & 0 & \dots & 0 \\ \boldsymbol{Z}_2 & 0 & \mathbf{1}_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Z}_m & \vdots & \vdots & \dots & \mathbf{1}_m \end{pmatrix}$$

where $\boldsymbol{Z}_i$ is the random effect matrix obtained from the mixed model representation of a P-spline as shown in Section 2.6.

$$\boldsymbol{u} = (u_1, u_2, ..., u_k, U_m)' \quad \text{and} \quad \boldsymbol{G} = \text{Cov}(\boldsymbol{u}) = \begin{pmatrix} \sigma_u^2 \boldsymbol{I} & 0 \\ 0 & \sigma_U^2 \boldsymbol{I} \end{pmatrix}$$

Instead of straight lines, we fit smooth curves that only differs in their intercepts.

In `R` we use the function `gamm`

```
library(mgcv)
# Smooth random intercept model
fit2.gamm <- gamm(height~factor(treatment)+s(age,k=40,bs="ps",m=2),
                  random=list(case=pdIdent(~1)),data=leukemia)
```

The smooth `age` effect is shown in the next figure.

```
plot(fit2.gamm$gam,2,scheme=1)
```

We can plot the fitted curves using the function `xyplot` from the library `lattice`, in order to plot the fitted curves and overlap the original data points we will use the library `latticeExtra` and the function `as.layer`.

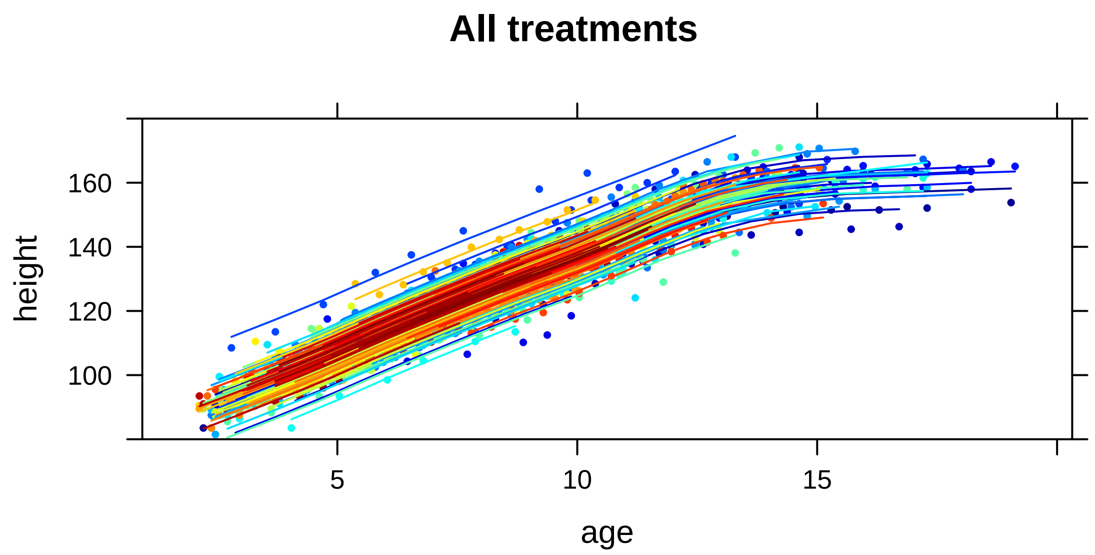


Figure 37: *Smooth effect for the random intercept model*

```
## Warning: package 'latticeExtra' was built under R version 3.4.4
```

The next plot shows the fitted trends by treatment.

Previous plots show that we account for the non-linear trend but we are not able to capture the individual trajectories.

4.3.1.3 Linear individual differences model (Random intercept and slope model) This model is an extension of the previous one. Now we allow not only for random intercept but also for random slopes.

$$y_{ij} = \sum_{l=2}^{L} \gamma_l \text{tr}_{il} + f(x_{ij}) + a_{i1} + a_{i2}x_{ij} + \epsilon_{ij} \quad \epsilon_{ij} \sim N(0, \sigma_\epsilon) \qquad (a_{i1}, a_{i2})' \sim \mathcal{N}(0, \Sigma) \tag{4.3}$$

In matrix notation, now the random effect matrix $\boldsymbol{Z}$ is

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{Z}_1 & \boldsymbol{X}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{Z}_2 & \boldsymbol{0} & \boldsymbol{X}_2 & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Z}_m & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{X}_m \end{pmatrix},$$

with random effects:

$$\boldsymbol{u} = (u_1, \dots, u_K, a_{11}, a_{12}, \dots, a_{m_1}, a_{m_2})'$$

and covariance matrix:

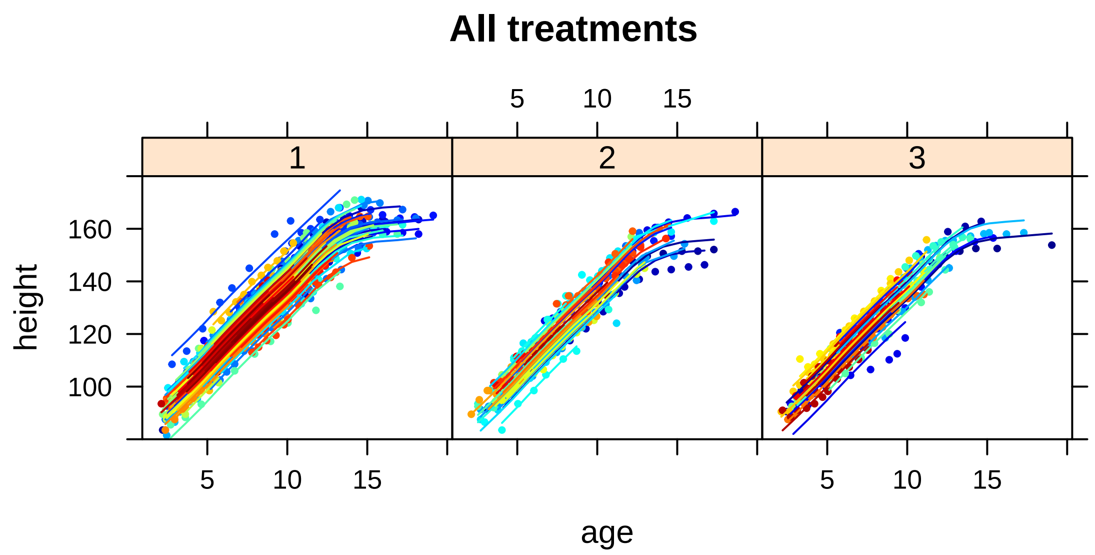


Figure 38: *Fitted trends for the smooth random intercept model by treatment*

$$\boldsymbol{G} = \mathrm{Cov}(\boldsymbol{u}) = \begin{pmatrix} \sigma_u^2 \boldsymbol{I} & 0 \\ 0 & \underset{1\leq i\leq m}{\mathrm{blockdiagonal}} \boldsymbol{\Sigma} \end{pmatrix}$$

```
# Random intercepts and slopes and smooth effect for age
fit3.gamm <- gamm(height~factor(treatment)+s(age,k=40,bs="ps",m=2),
                  random=list(case=pdSymm(~age)),data=leukemia)
summary(fit3.gamm$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## height ~ factor(treatment) + s(age, k = 40, bs = "ps", m = 2)
##
## Parametric coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        125.5430     0.4507 278.562   <2e-16 ***
## factor(treatment)2  -2.2864     0.9637  -2.373   0.0178 *
## factor(treatment)3  -1.3464     0.9734  -1.383   0.1668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

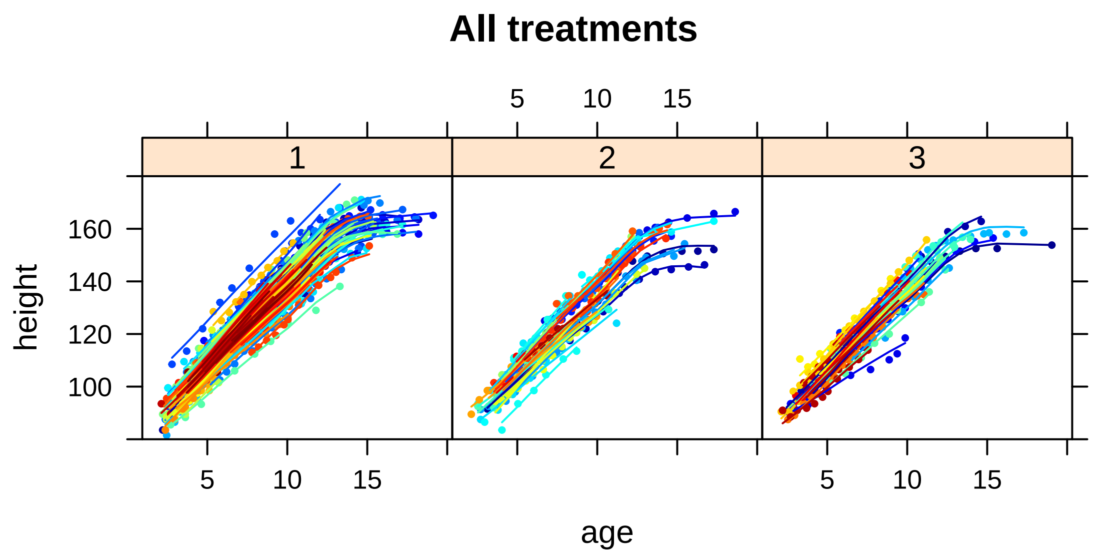


Figure 39: *Fitted individual trends for the smooth random intercept and slope model by treatment*

```
##           edf Ref.df    F p-value    
## s(age) 12.05  12.05 1198  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## R-sq.(adj) =  0.919   
##   Scale est. = 3.6616    n = 1988
```

This model adds more flexibility, allowing for non-parallel curves.

4.3.1.4 Curve by factor interaction The final aim of the study was the long term effects of the three different therapies. Hence, individual curves for each treatment is of interest. We can extend the model by allowing for a interaction of a factor variable with a continuous predictor. The model is of the form:

$$y_{ij} = f_{z_i}(x_{ij}) + a_{i1} + a_{i2}x_{ij} + \varepsilon_{ij} \tag{4.4}$$

more formally

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + Z_i u_k \sum_{l=2}^{L} \mathrm{tr}_{il}(\gamma_{0l} + \gamma_{1l}x_{ij}) + \sum_{l=2}^{L} \mathrm{tr}_{il} Z_i w_k^l + a_{i1} + a_{i2}x_{ij} + \varepsilon_{ij}$$

where

$$w_k^l \sim \mathcal{N}(0, \sigma_{wl}^2), \quad (a_{i1}, a_{i2})' \sim \mathcal{N}(0, \Sigma) \quad , \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

where $\mathrm{tr}_{il} = 1$ if $\mathrm{tr}_i = l$ and 0 otherwise.

In a model with factor variables, we must impose a constraint to ensure an identifiable model, in this case, we impose that $\gamma_{01} = \gamma_{11} = 0$, which means that $\beta_0 + \beta_1 x_{ij} + Z_i u_k$ is the fitted curve for $l = 1$, and $\gamma_{0l} + \gamma_{1l} x_{ij} + Z_i w_k^l$ is the difference between the fitted curves for therapy 2 (conventional radiation) and therapy 3 (hyperfractioned radiation) and therapy 1 (no radiation).

```
# Curve by treatment model
fit4.gamm <- gamm(height~factor(treatment)+s(age,k=40,bs="ps",m=2,
                                              by=factor(treatment)),
                  random=list(case=pdSymm(~age)),data=leukemia)
summary(fit4.gamm$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## height ~ factor(treatment) + s(age, k = 40, bs = "ps", m = 2,
##     by = factor(treatment))
##
## Parametric coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        125.8952     0.4606 273.358  < 2e-16 ***
## factor(treatment)2  -3.6383     1.0508  -3.463 0.000547 ***
## factor(treatment)3  -2.2342     1.0605  -2.107 0.035271 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                               edf Ref.df     F p-value
## s(age):factor(treatment)1 11.589 11.589 954.8  <2e-16 ***
## s(age):factor(treatment)2  8.986  8.986 250.8  <2e-16 ***
## s(age):factor(treatment)3  9.251  9.251 268.3  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.918
##   Scale est. = 3.41      n = 1988
```

Equivalent model with `lme`

```
attach(leukemia)
X=model.matrix(height~factor(treatment)*age)
treatment=factor(treatment)

MM=mixed.model.B(age,min(age)-0.5,max(age)+0.5,40,3,2,type="Eilers")

Z=MM[[2]]
```

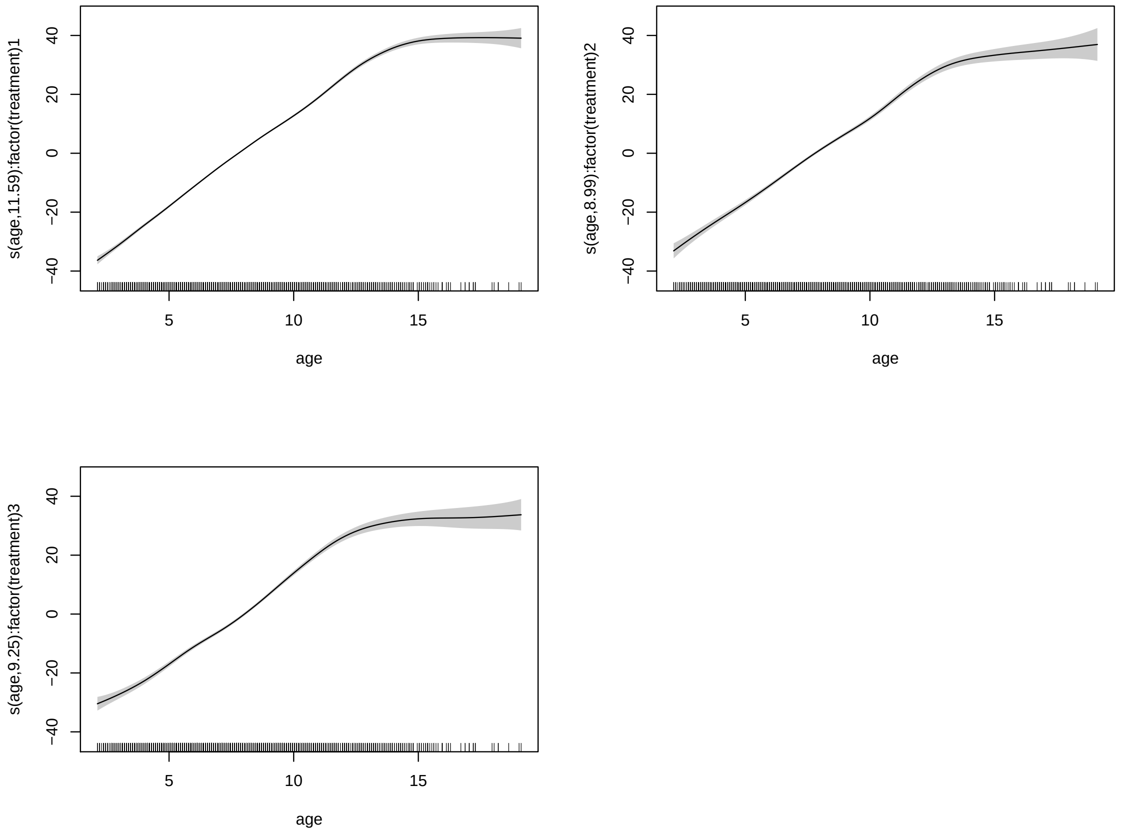


Figure 40: *Fitted global trends for the smooth random intercept and slope model by treatment*

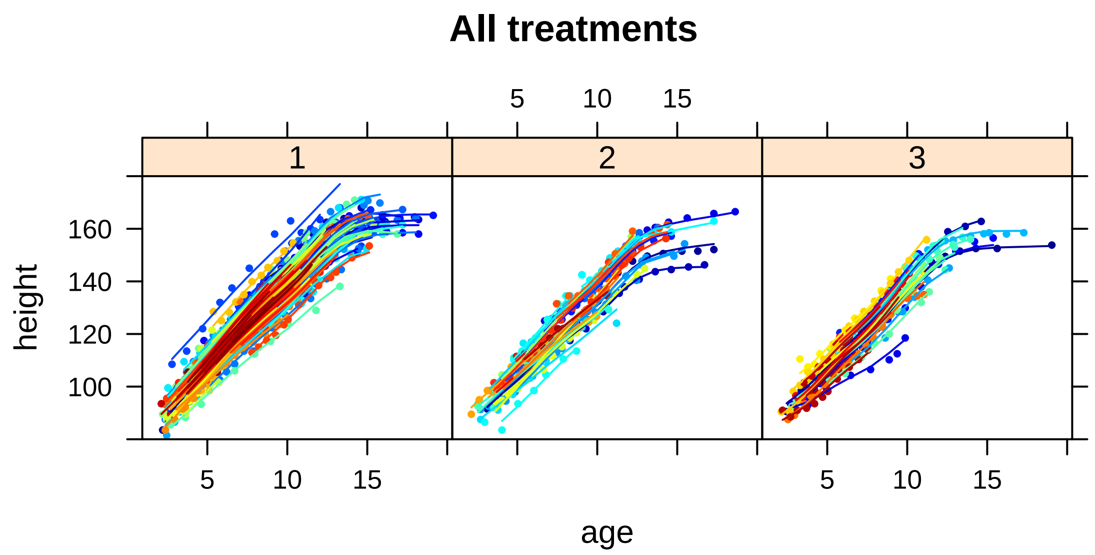


Figure 41: *Fitted individual trends for the smooth random intercept and slope model by treatment*

```
Id=factor(rep(1,length(height)))

Z.block4=list(treatment=pdIdent(~Z-1),case=pdSymm(~age))

data.fr <- groupedData(height ~ X[,-1] | Id, data = data.frame(height,X,Z,case,age))
model4 <- lme(height~X[,-1],data=data.fr,random=Z.block4)
```

4.3.1.5 Specific curves for individuals The more flexble model is the one that allows for specific differences among individuals using non-parametric smooth terms.

$$y_{ij} = \sum_{l=2}^{L} \gamma_l \text{tr}_{il} + f(x_{ij}) + g_i(x_{ij}) + \epsilon_{ij} \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma_\epsilon) \tag{4.5}$$

with

$$g(x_{ij}) = a_{i1} + a_{i2}x_{ij} + Z_i v_k \quad (a_{i1}, a_{i2})' \sim \mathcal{N}(0, \Sigma) \quad v_k \sim \mathcal{N}(0, \sigma_v^2)$$

Each individual curve $g_i()$ has two components: i) parametric and ii) no-parametric, both are random (in contrast to the model proposed by Brumback and Rice (1998) where computational issues arises due to the need of estimating $2m$ parameters in the linear part). This model is written in matrix form as:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\epsilon}$$

where

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{Z}_1 & \boldsymbol{X}_1 & \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{Z}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \boldsymbol{Z}_2 & \mathbf{0} & \boldsymbol{X}_2 & \dots & \mathbf{0} & \mathbf{0} & \boldsymbol{Z}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Z}_m & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{X}_m & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{Z}_m \end{pmatrix}$$

and

$$\boldsymbol{u} = \left(u_1, \dots, u_K, a_{11}, a_{12}, \dots, a_{m1}, a_{m2}, v_1, \dots, v_K\right)',$$

with random effects covariance matrix

$$\boldsymbol{G} = \text{Cov}(\boldsymbol{u}) = \begin{pmatrix} \sigma_u^2 \boldsymbol{I} & 0 & 0 \\ 0 & \underset{1 \le i \le m}{\text{blockdiagonal}}\ \boldsymbol{\Sigma} & \\ 0 & 0 & \sigma_v^2 \boldsymbol{I} \end{pmatrix}$$

This model is very complex to fit using `gamm`, because we would need to define 197 variables (subjects) in order to obtain an individual curve for each girl. However, this is very simple using `lme`.

```
# We use function lme()
X=model.matrix(height~factor(treatment)*age)
MM=mixed.model.B(age,min(age)-0.5,max(age)+0.5,40,3,2,type="Eilers")

Z=MM[[2]]
Id=factor(rep(1,length(height)))

MM.case=mixed.model.B(age,min(age)-0.5,max(age)+0.5,10,3,2,type="Eilers")
treatment=factor(treatment)
Z.case=MM.case[[2]] # create a specific sub-matrix per each individual
Z.block5=list(treatment=pdIdent(~Z-1),case=pdSymm(~age),case=pdIdent(~Z.case-1))

data.fr5 <- groupedData(height ~ X[,-1] | Id,
                        data = data.frame(height,X,Z,Z.case,case,age))
model5 <- lme(height~X[,-1],data=data.fr5,random=Z.block5)
```

The next Figure shows the curve specific per subject fits.

4.3.1.6 Models comparisons When we studied mixed models, we saw that the standard variance components estimation method is REML (restricted/residual maximum likelihood). In addition to parameter estimation, we are also interested in testing the need of a parametric model or not. This type of tests are not straigthforward. For instance, in the additive mixed model in Eq. (4.2), we may be interested in kwnowing if the function that describes the population mean of the girls is a straight line or it is non-linear. This would be equivalent to the contrast:

$$H_0 : \sigma_u^2 = 0 \quad \text{vs.} \quad H_1 : \sigma_u^2 > 0.$$

A first problem is that the contrast parameter is at the boundary of the parameter space, i.e. $[0, \infty)$. Hence the Likelihood Ratio Test (LRT), i.e.:

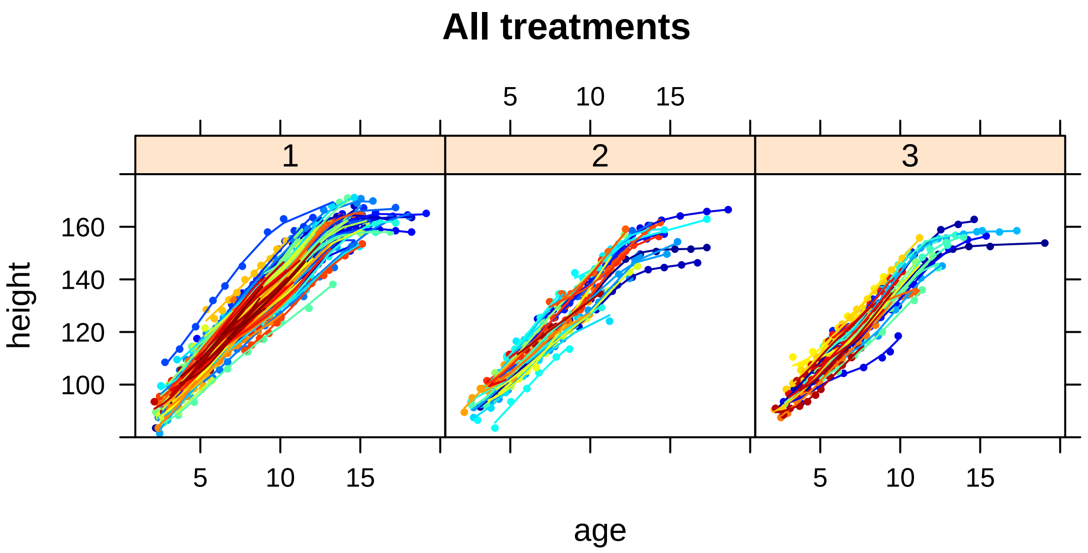


Figure 42: *Fitted individual curve-specifice trend for subject by treatment*

$$RLRT = \sup_{H_1} REL(\boldsymbol{\beta}, \sigma^2_\varepsilon, \sigma^2_U, \sigma^2_u) - \sup_{H_0} REL(\boldsymbol{\beta}, \sigma^2_\varepsilon, \sigma^2_U, \sigma^2_u)$$

cannot be compared to a χ^2_1. Self and Liang (1987) and Stram and Lee (1994) showed that if $\boldsymbol{y}$ can be partitioned as independent sub-vectors and the number of sub-vectors tends to infinity, the LRT is asymptotically distributed as $\frac{1}{2}\chi^2_q + \frac{1}{2}\chi^2_{q+1}$, where q is the number of random effect under the null hypothesis. However, this assumption is not true is some semi-parametric setting and the approximation may not be good. Crainiceanu et al. (2004) derived the case in which a polynomial regression tested against a penalized splines with a single smoothing parameter, later Crainiceanu et al. (2005) studied the case with more than one variance components. These authors suggest the use of simulations to determine the distribution of the statistic under the null hypothesis. The idea is as follows: the parameters under the null hypothesis are estimated, then the distribution of the LR test statistic is simulated under the null hypothesis. Crainiceanu et al. (2005) proposed a fast algorithm for the simulation in some settings, however, the complexity of the algorithms increases linearly with the number of subjects and the complexity of the model, and hence this method is intractable for many other cases. Greven, Küchenhoff, and Peters (2008) proposed more methods implemented in the `R` library `RLRsim` (Exact (Restricted) Likelihood Ratio tests for mixed and additive models).

For the leukemia data, we will fit two nested models with 5 and 6 variance components, models (4.3) and (4.5), but including a curve-by-factor interaction:

$$\begin{aligned} y_{ij} = &\quad f_{z_i}(x_{ij}) + a_{i1} + a_{i2}x_{ij} + \varepsilon_{ij}, \\ y_{ij} = &\quad f_{z_i}(x_{ij}) + g_i(x_{ij}) + \varepsilon_{ij}, \end{aligned}$$

where y_{ij} the height of the ith girl at age j, para $i = 1, \ldots, 197$ y j between 1 and 21, f_1 is the average curve for the girls who received treatment 1 (no radiation), f_2 for the girls who received conventional radiation (treatment=2) and f_3 for the girls who received treatment 3 (hyperfractioned

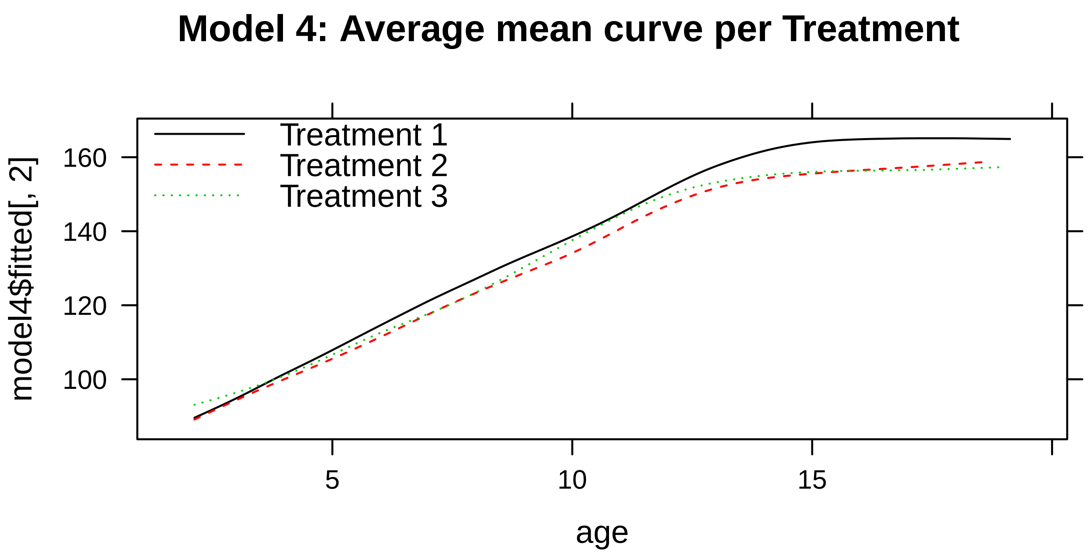


Figure 43: *Average mean curves per treatment for models 4.4 and 5.5*

radiation therapy), a_{i1} and a_{i2} are the random intercepts and slopes and y $g_i(x_{ij})$ is the specific deviation of the ith girl respect to the average curve of her treatment group.

The interest in this study are the treatment effects on height along time and the individual responses to the treatments.

Next plots show the estimated average population curve effects per treatment. It can be appreciated that in both cases the trend pattern is similar, and for those girls who did not received any radiation therapy (treatment 1) the effect is higher than the other two therapy groups, in particular from 11 years old.

```
library(latticeExtra)
library(fields)
xyplot(model4$fitted[,2] ~ age,groups=treatment,col=1:3,lty=1:3,pch=19,
       data=leukemia,main="Model 4: Average mean curve per Treatment",cex=.35,type="a",
       key=list(corner=c(0,1),cex=1,lines=list(col=1:3, lty=1:3),
                text=list(c("Treatment 1","Treatment 2","Treatment 3"))))
```

```
xyplot(model5$fitted[,2] ~ age,groups=treatment,col=1:3,lty=1:3,pch=19,data=leukemia,
       main="Model 5: Average mean curve per Treatment",
       cex=.35,type="a",key=list(corner=c(0,1),cex=1,lines=list(col=1:3, lty=1:3),
                                 text=list(c("Treatment 1","Treatment 2","Treatment 3"))))
```

In order to compare the 3 curves, we refit the model with a single curve for the mean, then the null hyphotesis would be:

$$H_0 : \gamma_{jl} = 0 \quad j = 0, 1 \quad l = 1, 2, 3 \quad \text{and} \quad \sigma_w^2 = 0$$

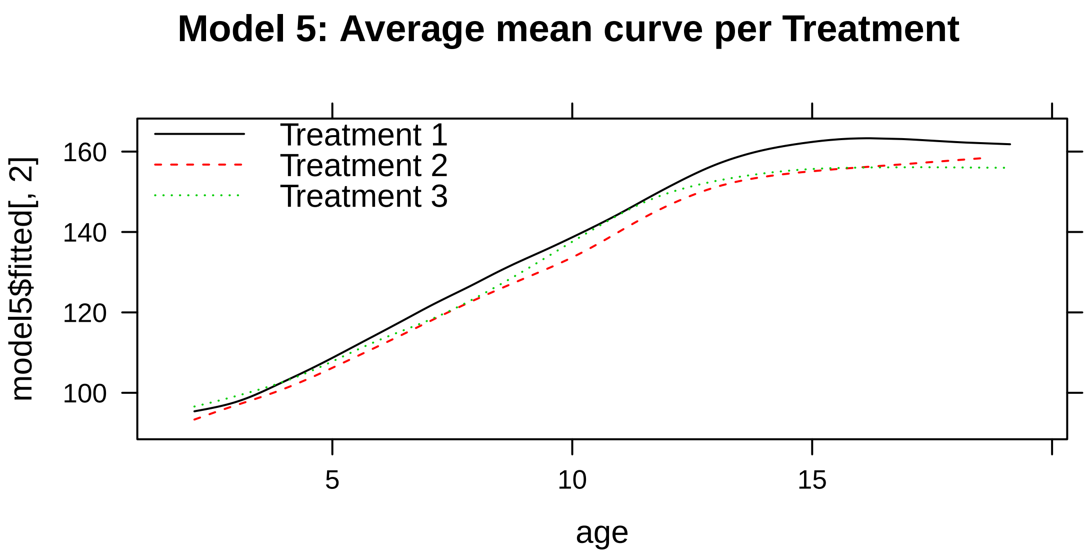


Figure 44: *Average mean curves per treatment for models 4.4 and 5.5*

for γ_{jl} and σ_w^2 defined in (4.5).

```
# without treatment random effects (i.e. a single curve for all treatment levels)
Z.block5.2=list(Id=pdIdent(~Z-1),case=pdSymm(~age),case=pdIdent(~Z.case-1))
data.fr <- groupedData(height ~ X[,-1] | Id,
                       data = data.frame(height,X,Z,Z.case,case,age))
model5.2 <- lme(height~X[,-1],data=data.fr,random=Z.block5.2)
```

```
xyplot(model5.2$fitted[,2]~ age,groups=treatment,col=1:3,lty=1:3,pch=19,data=leukemia,
       main="Average mean curve per Treatment",cex=.35,type="a",
       key=list(corner=c(0,1),cex=1,lines=list(col=1:3, lty=1:3),
                text=list(c("Treatment 1","Treatment 2","Treatment 3"))))
```

```
xyplot(model5.2$fitted[,1]~ age,groups=treatment,col=1:3,lty=1:3,pch=19,data=leukemia,
       main="Fixed effect",cex=.35,type="a",
       key=list(corner=c(0,1),cex=1,lines=list(col=1:3, lty=1:3),
                text=list(c("Treatment 1","Treatment 2","Treatment 3"))))
```

```
xyplot(model5.2$fitted[,1]-model5.2$fitted[,2]~ age,groups=treatment,col=1:3,lty=1:3,
       pch=19,data=leukemia,main="Fixed effect - population random effect",cex=.35,type="a",
       key=list(corner=c(0,1),cex=1,lines=list(col=1:3, lty=1:3),
                text=list(c("Treatment 1","Treatment 2","Treatment 3"))))
```

The appropriate way to proceed would be to use parametric bootstrap to obtain the distribution of the likelihood ratio test, however, the computational time needed to fit the null model to a large

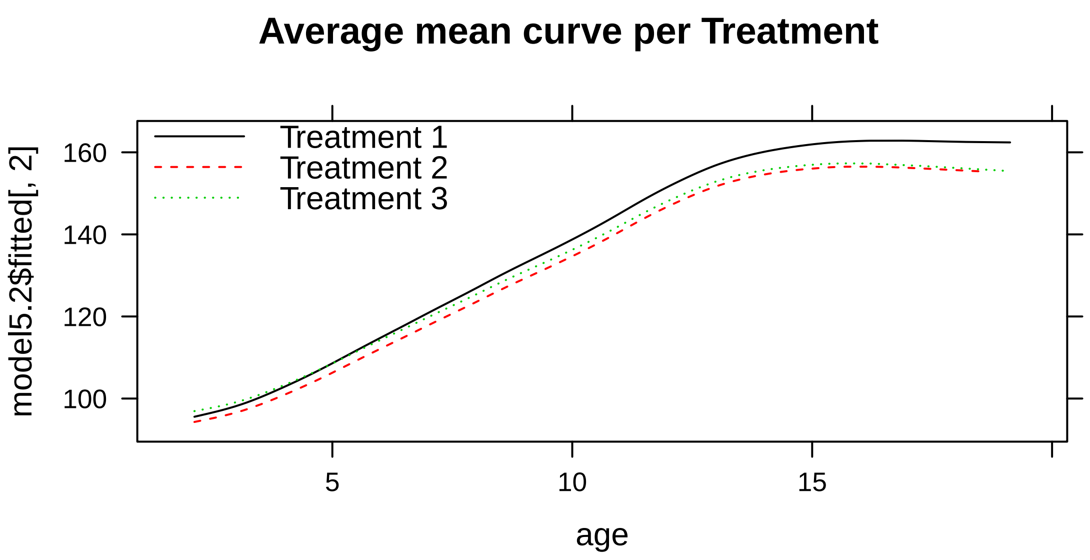


Figure 45: *Average mean curves per treatment for models 4.5 (with no treatment random effect)*

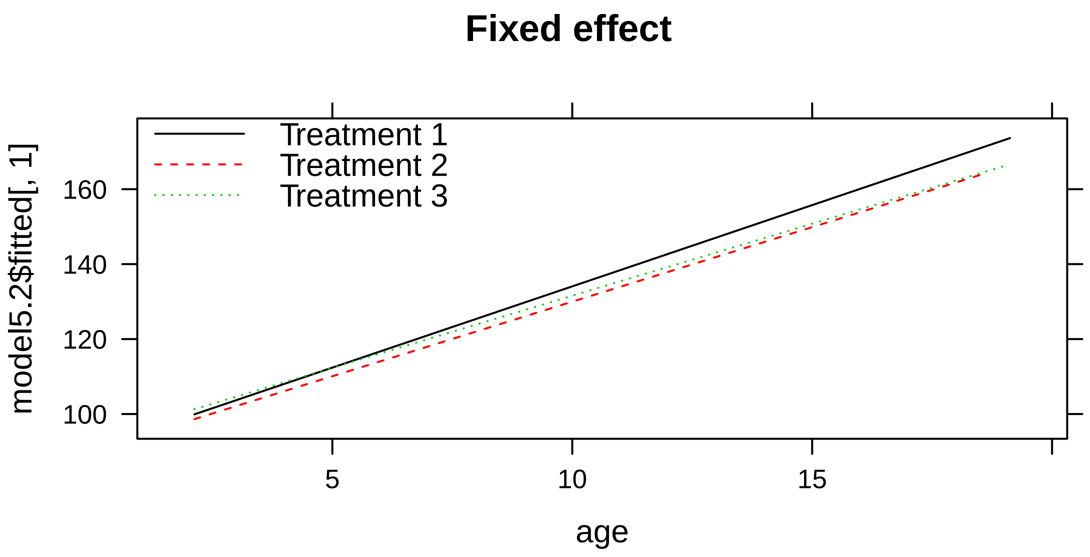


Figure 46: *Average mean curves per treatment for models 4.5 (with no treatment random effect)*

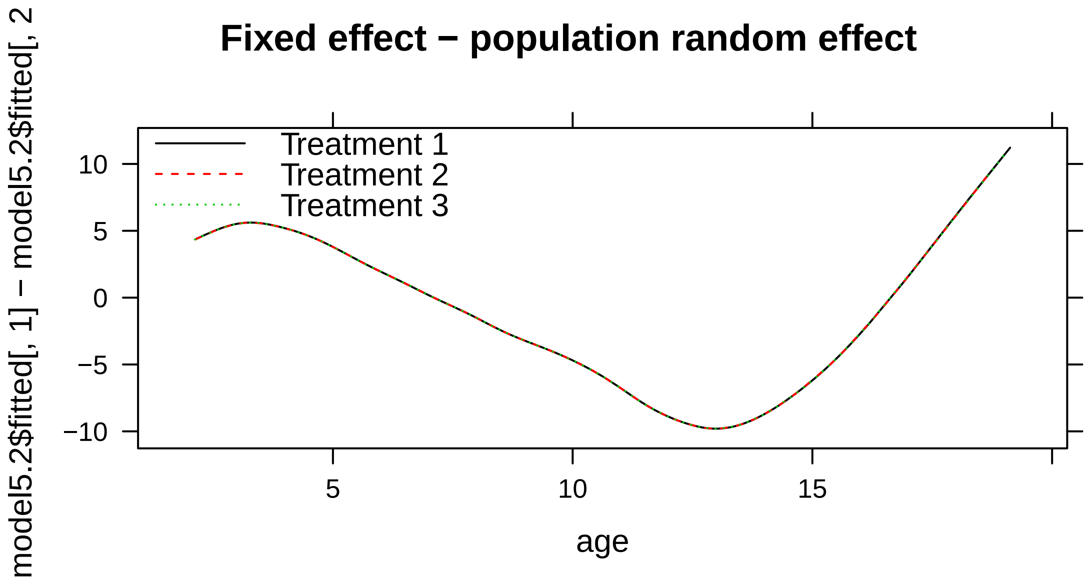


Figure 47: *Average mean curves per treatment for models 4.5 (with no treatment random effect)*

number of simulated data sets (between 10,000 and 100,000) would make the bootstrap method infeasible. A more naive approach consists of comparing the $-2\log(RLRT)$ with the 90th percentile of $\frac{1}{2}\chi^2_0 + \frac{1}{2}\chi^2_1$ (the distribution of the $-2\log(RLRT)$ under the assumption of independent y's).

We compute the $-2\log(RLRT)$ of both models

```
minus2.RLRT.5.2vs5 <- -2*(logLik(model5.2,REML=TRUE)-logLik(model5,REML=TRUE))
minus2.RLRT.5.2vs5
```

```
## 'log Lik.' 34.62625 (df=12)
```

and compare it with $\frac{1}{2}\chi^2_0 + \frac{1}{2}\chi^2_1$

```
0.5*qchisq(.90,0)+0.5*qchisq(.90,1) # 1/2 Chisq(0) + 1/2 Chisq(1)
```

```
## [1] 1.352772
```

The results show that we would reject H_0 and then that there is a significative different between the girls' height depending on the therapy.

```
minus2.RLRT.4vs5 <- -2*(logLik(model4,REML=TRUE)-logLik(model5,REML=TRUE))
minus2.RLRT.4vs5
```

```
## 'log Lik.' 453.0032 (df=11)
```

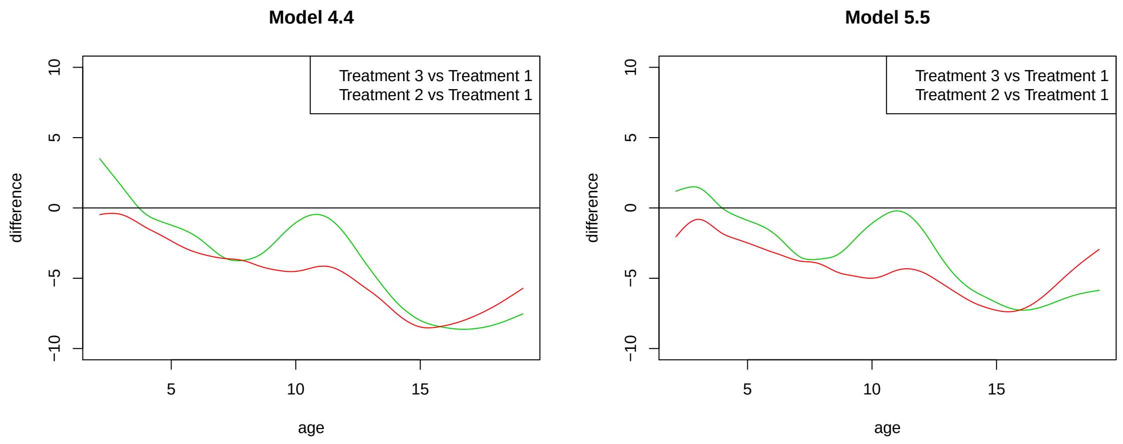


Figure 48: *Random effects models 4.4 and 4.5*

```
0.5*qchisq(.90,2)+0.5*qchisq(.90,3) # 1/2 Chisq(2) + 1/2 Chisq(3)
```

```
## [1] 5.428279
```

To test whether or not the individual response to treatment is linear we compare models (`model5`) and (`model5.2`). Next figures show how compared to `Treatment 1` (black line taken as baseline), girls who received `Treatment 2` and `Treatment 3` are smaller particularly at older ages.

```
## The following object is masked _by_ .GlobalEnv:
##
##     treatment
```

```
## The following objects are masked from leukemia (pos = 3):
##
##     age, case, height, treatment
```

```
## The following object is masked from fossil:
##
##     age
```

The next figure show the random effects for each girl in model (4.5), it is clear that for many of the girls the effect is far from linear.

Another option is the use of `anova` function, however bare in mind that the approximation of the p-value is approximate (we can also have a look at the model with lower AIC/BIC)

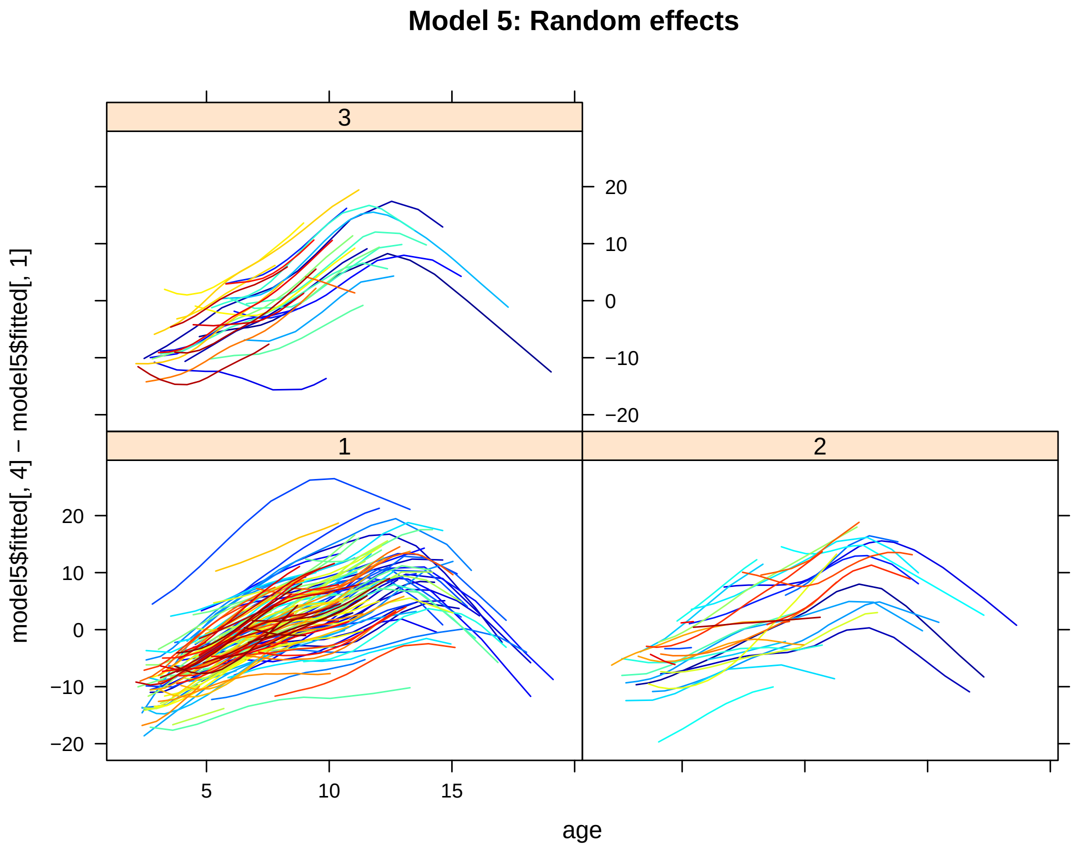


Figure 49: *Random effects model 5 by treatment*

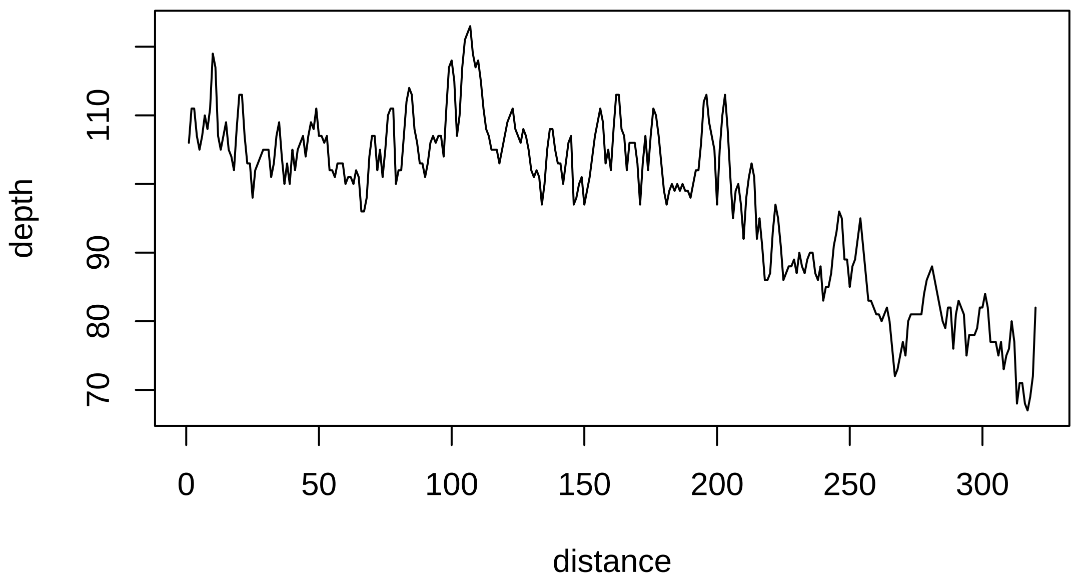


Figure 50: *Profile of a block of wood subject to grinding*

```
anova(model4,model5)
```

```
##        Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## model4     1 11 9319.186 9380.696 -4648.593
## model5     2 12 8868.182 8935.285 -4422.091 1 vs 2 453.0032  <.0001
```

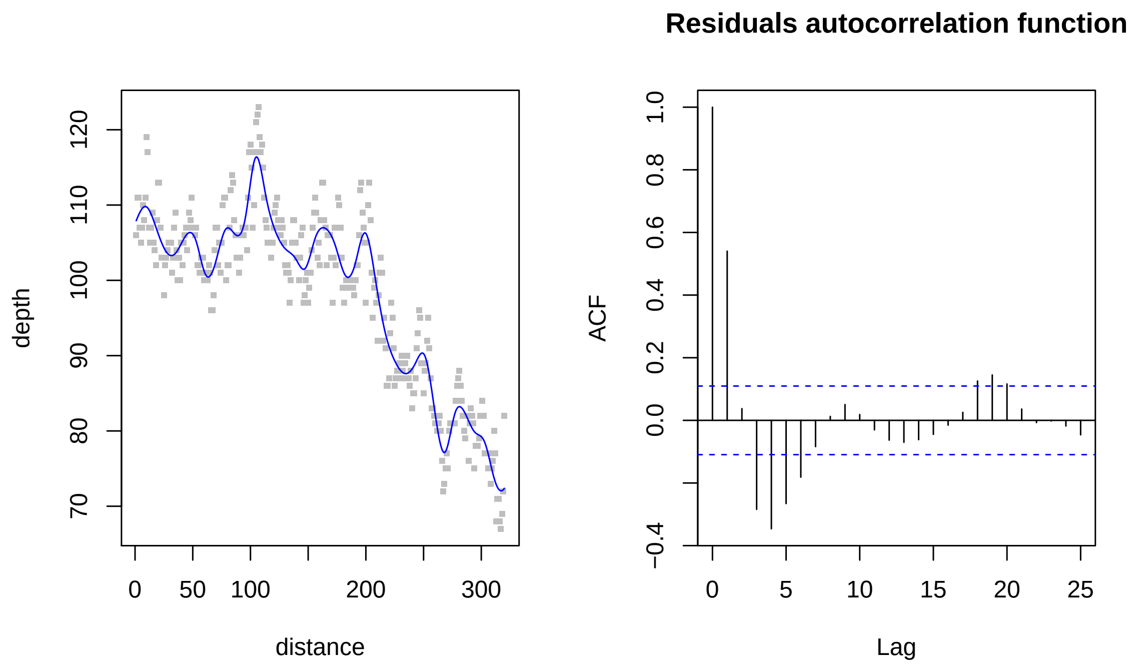
4.4 Correlated data

The data were analyzed in Pandit and Wu (1983). They present a dataset (`wood.txt`) describing 320 measurements of a block of wood that was subject to grinding. Next Figure shows the profile (depth) height at different distances. The profile variation follows a curve determined by the radius of the grinding stone.

Using P-splines we can estimate a smooth trend and possible correlation simultaneously. The mixed model representation of P-splines helps us to estimate both effects pretty easily. Let us consider, firstly a smooth model that ignores the correlation between the observations.

The left figure shows the estimated smooth trend. This shows that ignoring the correlation gives a non-smooth trend. The right panel show the autocorrelation function with correlation in the errors.

```r
library(mgcv)
cor.gamm0 <- gamm(depth ~ s(distance,k=40,bs="ps",m=2), method="REML")
par(mfrow=c(1,2))
plot(distance,depth,cex=.55,pch=15,col="grey")
lines(distance,fitted(cor.gamm0$gam),main="Smooth trend with AR(1)",col="blue")
acf(residuals(cor.gamm0$lme,type="n"), main="Residuals autocorrelation function")
```



A correlated data model can be added with an AR(1) or AR(2) structure, i.e.

```r
# create a factor per observation
#  to define the correlation structure
Id <- factor(rep(1,length(depth)))

# AR(1)
cor.gamm1 <- gamm(depth ~ s(distance, k=40, bs="ps", m=2),
                  correlation=corARMA(form=~distance|Id,p=1,q=0),
                  method="REML")
# AR(2)
cor.gamm2 <- gamm(depth ~ s(distance, k=40, bs="ps", m=2),
                  correlation=corARMA(form=~distance|Id,p=2,q=0),
                  method="REML")
```

The next Figure shows how including a correlation structure results in a smoother trend. In particular, the AR(2) model autocorrelation shows uncorrelated residuals.

Check the smoothing parameters

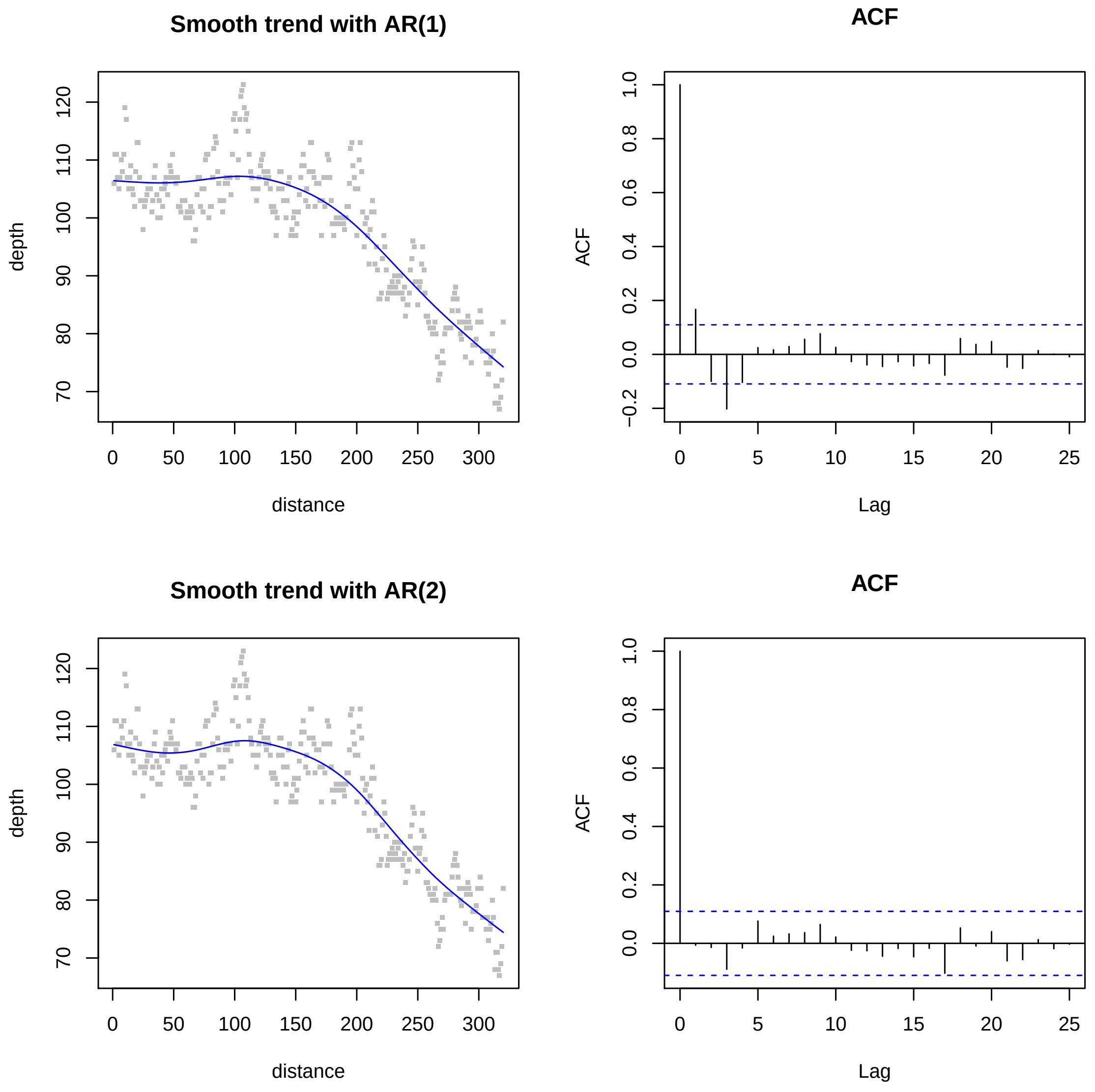


Figure 51: *Smooth trends and ACF for AR(1) and AR(2) error models*

```
print(c(cor.gamm0$gam$sp,cor.gamm1$gam$sp,cor.gamm2$gam$sp))
```

```
## s(distance) s(distance) s(distance)
##    2.117994 3228.963761 2014.146340
```

We can compare the model using `anova` function

```
anova(cor.gamm0$lme,cor.gamm1$lme,cor.gamm2$lme)
```

```
##               Model df      AIC      BIC    logLik   Test   L.Ratio
## cor.gamm0$lme     1  4 1850.568 1865.616 -921.2841
## cor.gamm1$lme     2  5 1653.212 1672.022 -821.6058 1 vs 2 199.35667
## cor.gamm2$lme     3  6 1638.010 1660.582 -813.0049 2 vs 3  17.20174
##               p-value
## cor.gamm0$lme
## cor.gamm1$lme  <.0001
## cor.gamm2$lme  <.0001
```

–>

4.5 Generalized additive mixed models (GAMMs)

California House prices The data contained 20.640 observations on house values (response variable) and 8 covariables, reflecting the characteristics of the property

```
calif <- read.table("GAMs-data/cadata.dat", header=TRUE)
names(calif)
```

```
## [1] "MedianHouseValue" "MedianIncome"     "MedianHouseAge"
## [4] "TotalRooms"       "TotalBedrooms"    "Population"
## [7] "Households"       "Latitude"         "Longitude"
```

1. longitude: A measure of how far west a house is
2. latitude: A measure of how far north a house is
3. housingMedianAge: Median age of a house within a block
4. totalRooms: Total number of rooms within a block
5. totalBedrooms: Total number of bedrooms within a block
6. population: Total number of people residing within a block
7. households: Total number of households, a group of people residing within a home unit, for a block
8. medianIncome: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. medianHouseValue: Median house value for households within a block (measured in US Dollars)

```r
linfit <- lm(log(MedianHouseValue)~.,data=calif)
print(summary(linfit))
```

```
##
## Call:
## lm(formula = log(MedianHouseValue) ~ ., data = calif)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5180 -0.2038  0.0016  0.1949  3.4641
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.180e+01  3.059e-01 -38.570  < 2e-16 ***
## MedianIncome    1.782e-01  1.639e-03 108.753  < 2e-16 ***
## MedianHouseAge  3.261e-03  2.111e-04  15.446  < 2e-16 ***
## TotalRooms     -3.186e-05  3.855e-06  -8.265  < 2e-16 ***
## TotalBedrooms   4.798e-04  3.375e-05  14.215  < 2e-16 ***
## Population     -1.725e-04  5.277e-06 -32.687  < 2e-16 ***
## Households      2.493e-04  3.675e-05   6.783 1.21e-11 ***
## Latitude       -2.801e-01  3.293e-03 -85.078  < 2e-16 ***
## Longitude      -2.762e-01  3.487e-03 -79.212  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.34 on 20631 degrees of freedom
## Multiple R-squared:  0.6432, Adjusted R-squared:  0.643
## F-statistic:  4648 on 8 and 20631 DF,  p-value: < 2.2e-16
```

Next Figure plots the predicted prices, ± 2 standard errors, against the actual prices. The predictions are not all that accurate, the RMS residual is 0.340 on the log scale (i.e., 41%), and only 3.3% of the actual prices fall within the prediction bands.6 On the other hand, they are quite precise, with an RMS standard error of 0.0071 (i.e., 0.71%). This linear model is pretty thoroughly converged.

```r
predictions = predict(linfit,se.fit=TRUE)
plot(calif$MedianHouseValue,exp(predictions$fit),cex=0.1,
     xlab="Actual price",ylab="Predicted")
segments(calif$MedianHouseValue,exp(predictions$fit-2*predictions$se.fit),
         calif$MedianHouseValue,exp(predictions$fit+2*predictions$se.fit),
         col="red")
abline(a=0,b=1,lty=2,col=4,lwd=2)
```

```r
library(mgcv)
addfit <- gam(log(MedianHouseValue) ~ s(MedianIncome,bs="ps")
              + s(MedianHouseAge,bs="ps") + s(TotalRooms,bs="ps")
```

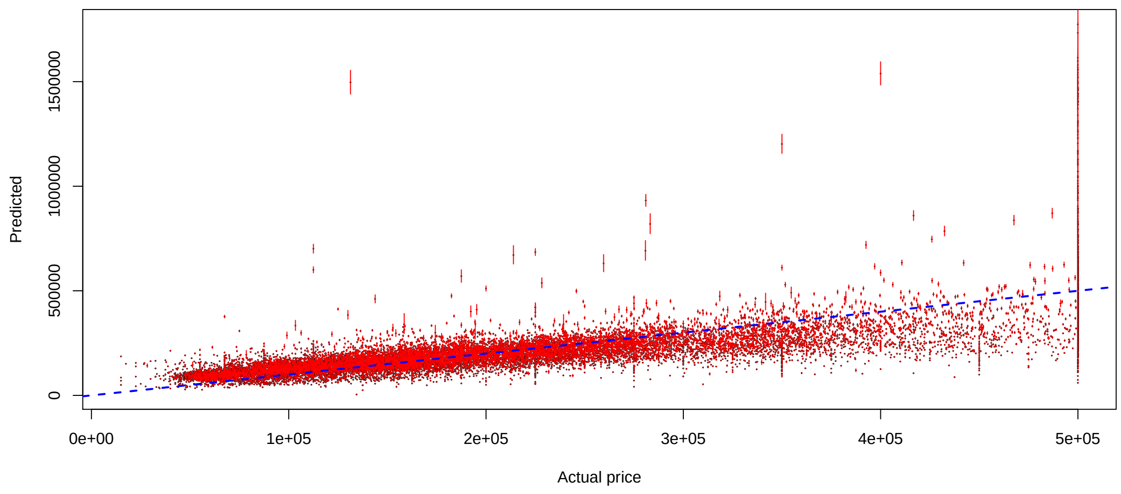


Figure 52: *Actual median house values (horizontal axis) versus those predicted by the linear model (black dots), plus or minus two standard errors (grey bars). The dashed line shows where actual and predicted prices would be equal.*

```
              + s(TotalBedrooms,bs="ps") + s(Population,bs="ps") + s(Households,bs="ps")
              + s(Latitude,bs="ps") + s(Longitude,bs="ps"),data=calif)
```

```
plot(addfit,scale=0,se=2,shade=TRUE,resid=TRUE,pages=1)
```

```
predictions = predict(addfit,se.fit=TRUE)
plot(calif$MedianHouseValue,exp(predictions$fit),cex=0.1,
     xlab="Actual price",ylab="Predicted")
segments(calif$MedianHouseValue,exp(predictions$fit-2*predictions$se.fit),
         calif$MedianHouseValue,exp(predictions$fit+2*predictions$se.fit),
         col="grey")
abline(a=0,b=1,lty=2,lwd=2)
```

Now we include `longitude` and `latitude` as smooth covariates that can interact:

```
addfit2 <- gam(log(MedianHouseValue) ~ s(MedianIncome,bs="ps") + s(MedianHouseAge,bs="ps")
               + s(TotalRooms,bs="ps") +s(TotalBedrooms,bs="ps") + s(Population,bs="ps") + s(Households
               + te(Longitude,Latitude,bs="ps"), data=calif)
```

```
plot(addfit2,scale=0,se=2,shade=TRUE,resid=TRUE,pages=1)
```

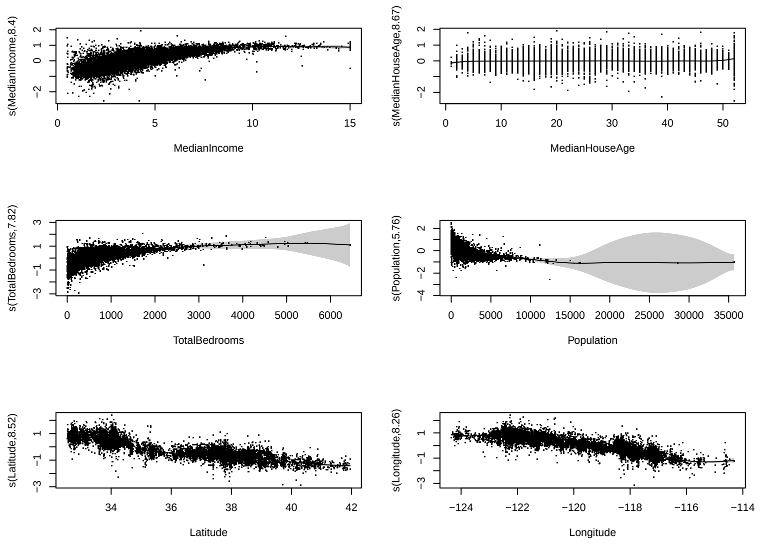


Figure 53: *Actual versus predicted prices for the additive model*

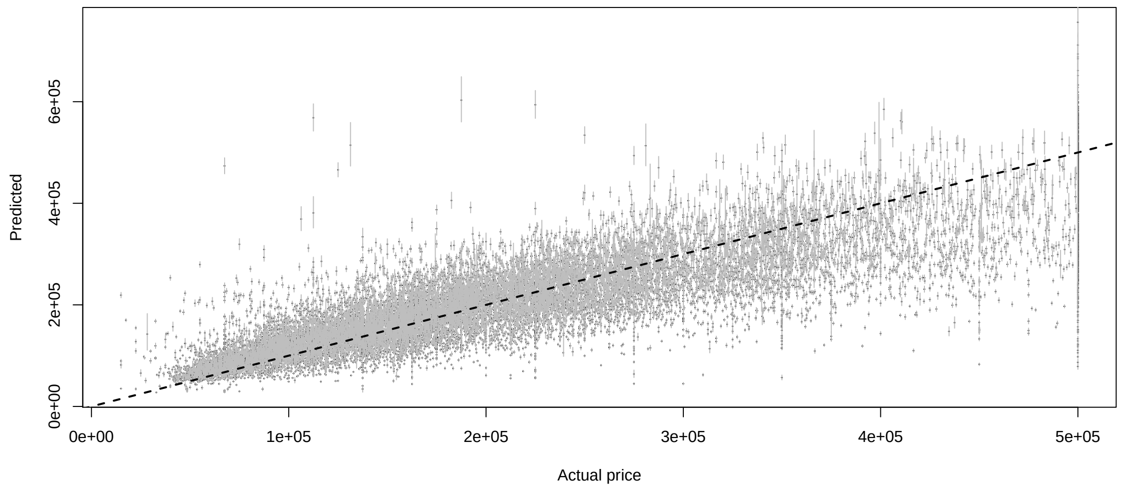


Figure 54: *Actual versus predicted prices for the additive model*

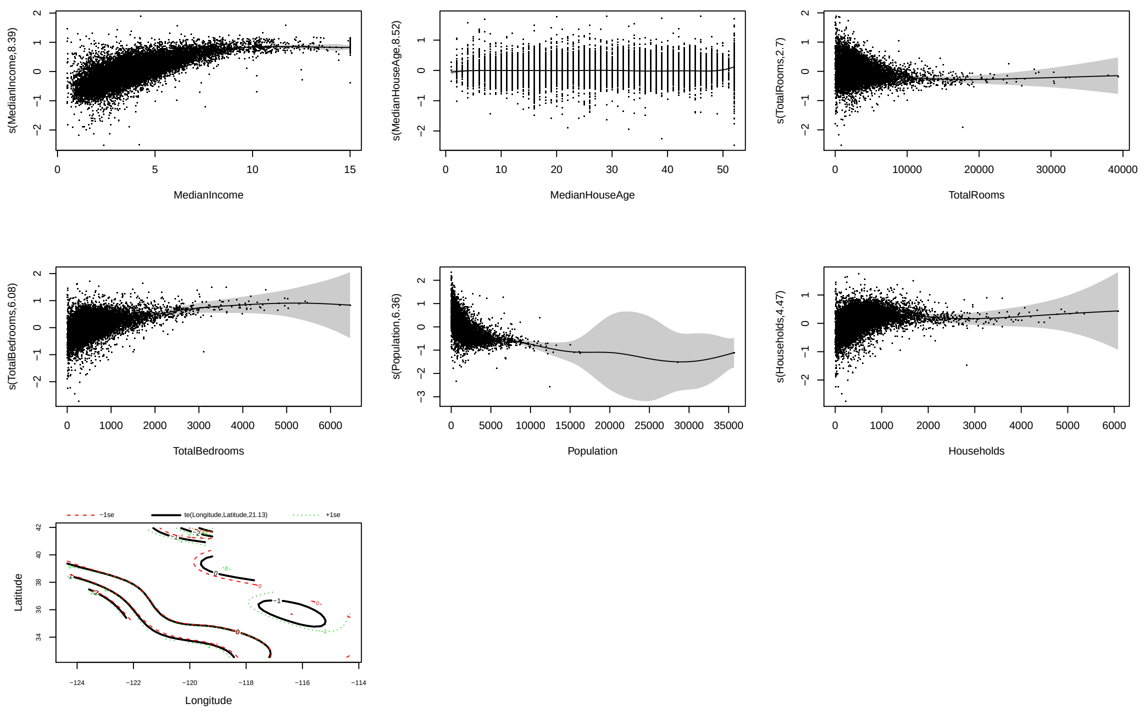


Figure 55: *Partial response functions and partial residuals for* `addfit2`*.*

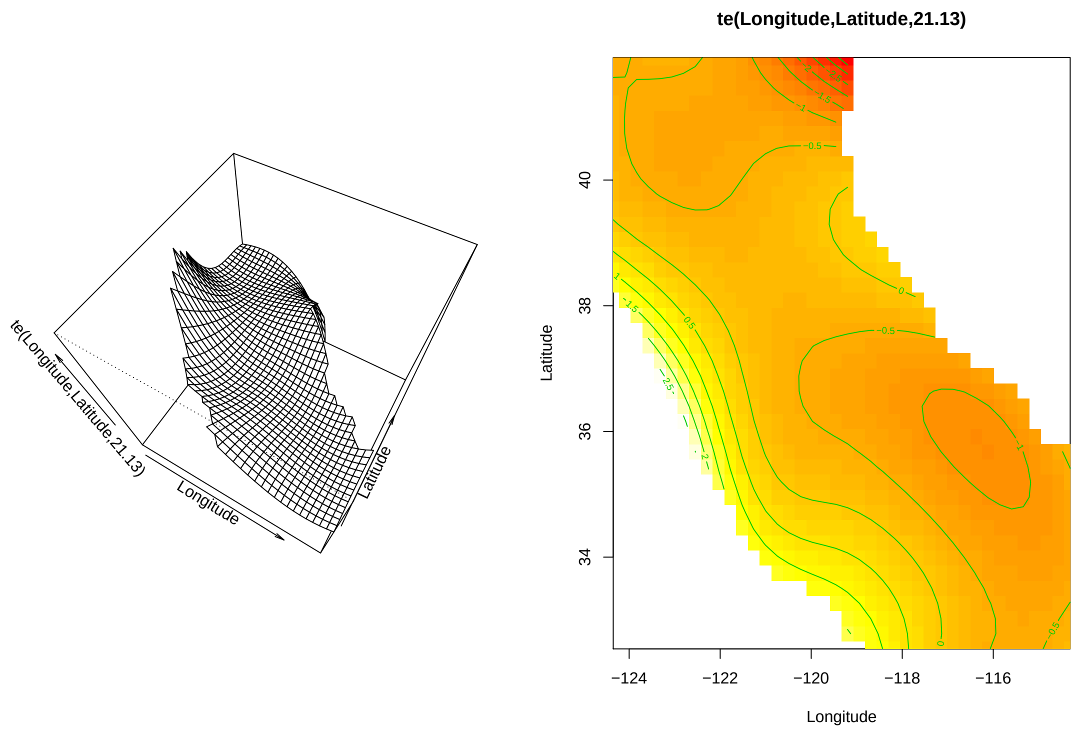


Figure 56: *Perspective and surface plot of the spatial component of* ***addfit2***

```
par(mfrow=c(1,2))
plot(addfit2,select=7,phi=60,pers=TRUE)
```

```
## Warning in plot.gam(addfit2, select = 7, phi = 60, pers = TRUE): argument
## pers is deprecated, please use scheme instead
```

```
plot(addfit2,select=7,scheme=2)
```

We can use `library(MBA)` to interpolate a smooth surface for `s(Longitude,Latitude)`

```
pred2 <- predict(addfit2,type="terms")
library(MBA)
```

```
## Warning: package 'MBA' was built under R version 3.4.4
```

```
library(fields)
image.plot(mba.surf(cbind(calif$Longitude,calif$Latitude,pred2[,7]),100,100,
                    extend=FALSE)$xyz.est, main="smooth spatial effect")
contour(mba.surf(cbind(calif$Longitude,calif$Latitude,pred2[,7]),100,100,
```

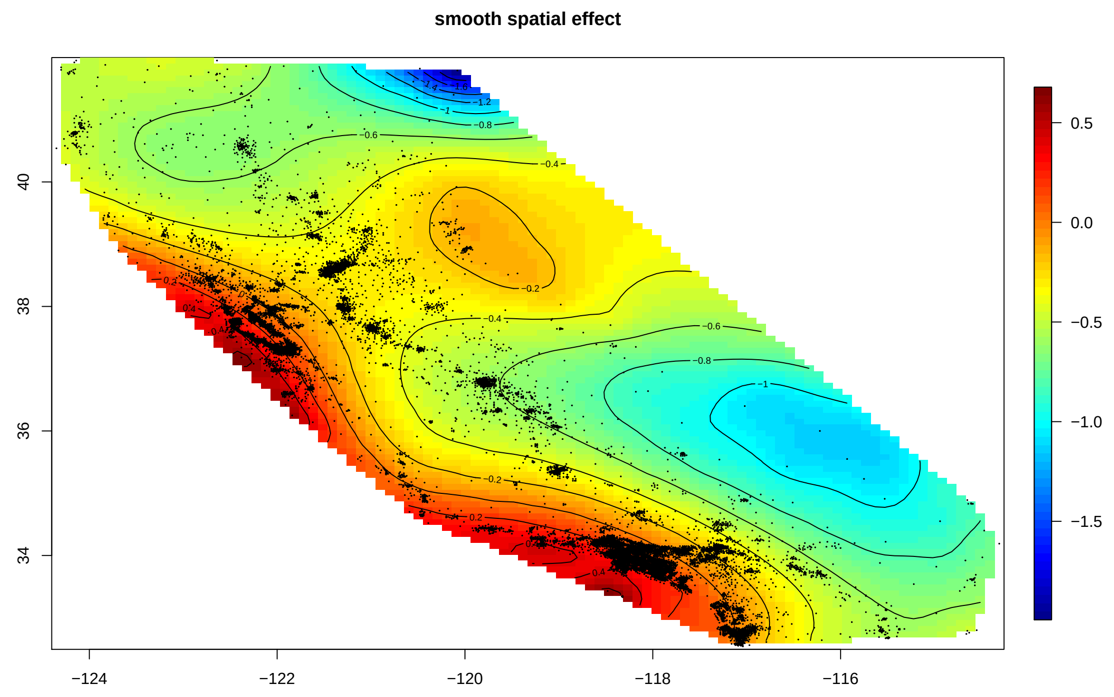


Figure 57: *Surface plot with* ***mba.surf***

```
                    extend=FALSE)$xyz.est,add=TRUE)
points(calif$Longitude,calif$Latitude,cex=.1,col=1)
```

Mackerel data from a Spanish survey

These data were recorded by a Spanish survey, as part of a multi-country survey of the abundance of mackerel eggs off the coast of north-western Europe, in 1992.

```
library(gamair)
library(sm)
library(mgcv)
library(fields)
library(maps)

data(mackerel)
data(mackp)
attach(mackerel)
Latitude=mack.lat
Longitude=-mack.long
```

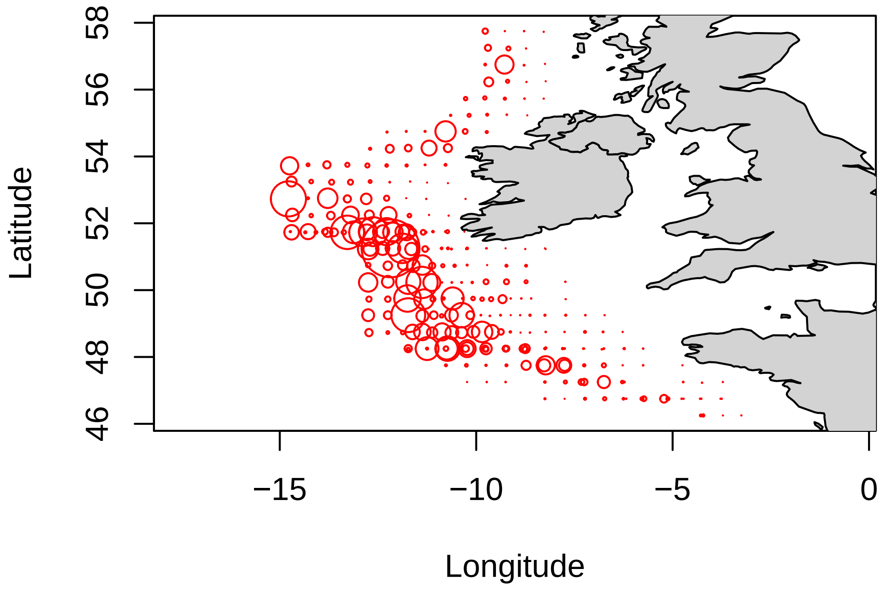


Figure 58: *Mackerel eggs abundance*

```
# plot the egg densities against location
plot(Longitude,Latitude,cex=Density/150,col=2,asp=.85)
map("world",add=TRUE,fill=TRUE,col="lightgrey")
```

Fit a `gam` for the spatial locations

```
m0<-gam(log(Density)~te(Longitude,Latitude,bs="ps",k=13))
# vis.gam(m0,plot.type="contour",color="terrain")
# map("world",add=TRUE,fill=TRUE,col="grey")
summary(m0)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(Density) ~ te(Longitude, Latitude, bs = "ps", k = 13)
##
## Parametric coefficients:
```

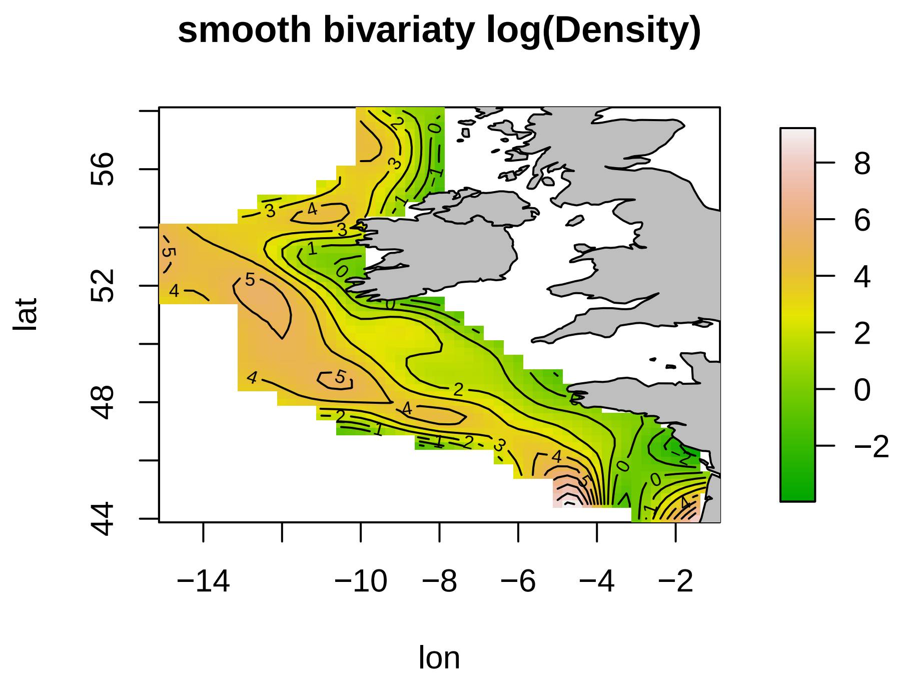


Figure 59: *Smooth spatial trend*

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.22409    0.05133   62.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                           edf Ref.df     F p-value
## te(Longitude,Latitude) 49.8   58.98 12.62  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.724   Deviance explained = 77.3%
## GCV = 0.8986  Scale est. = 0.73499   n = 279
```

Now we include `depth`, `Temperature` and `Salinity`

```
ldepth <- log(mack.depth) # logarithm scale
m1<-gam(log(Density)~te(Longitude,Latitude,bs="ps",k=13)+s(Temperature,bs="ps")+s(Salinity,bs="ps")+s(lo
par(mfrow=c(2,2))
plot(m1,scheme=2)
```

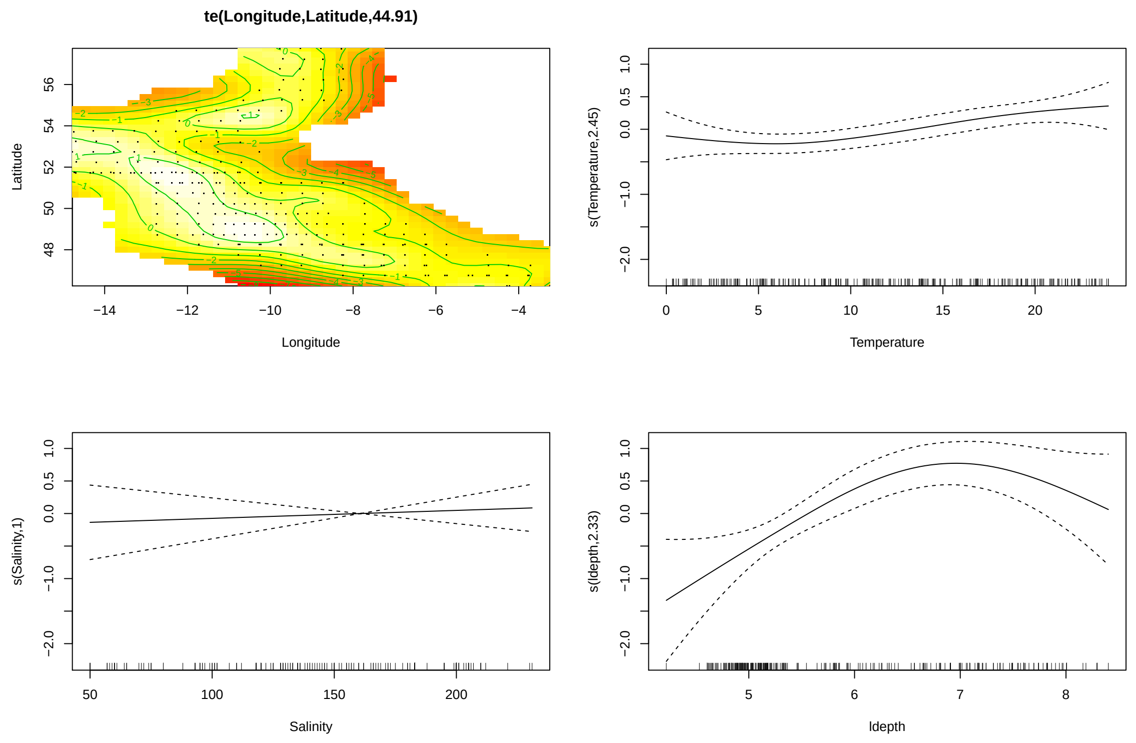


Figure 60: Additive model `gam` fit

```r
summary(m1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(Density) ~ te(Longitude, Latitude, bs = "ps", k = 13) + s(Temperature,
##     bs = "ps") + s(Salinity, bs = "ps") + s(ldepth, bs = "ps")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.22409    0.04876   66.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                             edf Ref.df     F  p-value
## te(Longitude,Latitude) 44.907 53.941 4.607  < 2e-16 ***
## s(Temperature)          2.445  2.992 4.023 0.009311 **
## s(Salinity)             1.000  1.000 0.224 0.636511
## s(ldepth)               2.325  2.871 6.814 0.000156 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 195/196
## R-sq.(adj) =   0.75   Deviance explained = 79.6%
## GCV = 0.81422  Scale est. = 0.66341   n = 279
```

Check residuals

```r
par(mfrow=c(2,2))
gam.check(m1)
```

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 42 iterations.
## The RMS GCV score gradient at convergence was 1.247587e-07 .
## The Hessian was positive definite.
## Model rank =  195 / 196
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                              k'    edf k-index p-value
## te(Longitude,Latitude) 168.00  44.91    1.12    0.99
```

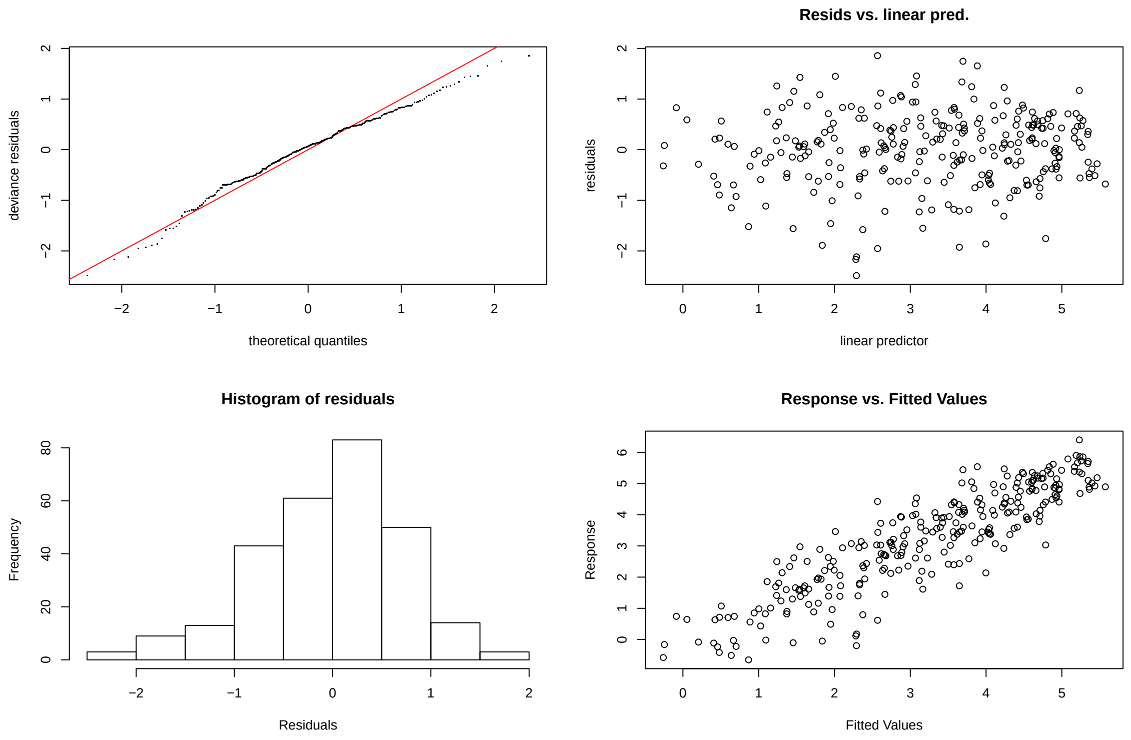


Figure 61: *gam.check results*

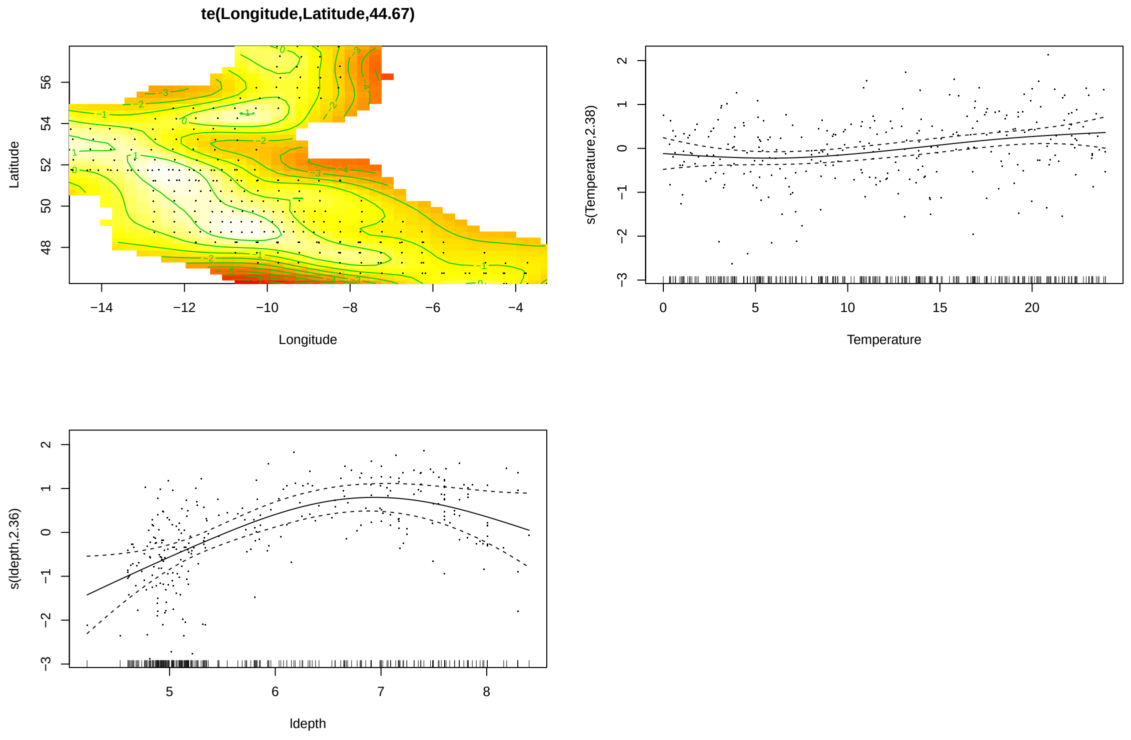


Figure 62: *Smooth term of* m2 *model*

```
## s(Temperature)          9.00    2.44     0.98     0.34
## s(Salinity)             9.00    1.00     1.01     0.48
## s(ldepth)               9.00    2.33     1.10     0.92
```

Remove `Salinity`

```
m2<-gam(log(Density)~te(Longitude,Latitude,bs="ps",k=13)+s(Temperature,bs="ps")+s(ldepth, bs="ps"))
par(mfrow=c(2,2))
plot(m2,scheme=2,1)
```

Mackerel data from a Spanish Survery (second analysis)

This data exhibit rather different features from the remainder of the survey. One of these features is that no eggs were detected at all at a substantial number of the sampling points. This is due to the smaller nets and the need to compensate by taking a larger number of smaller volume samples. The sampling locations are shown in the next Figure.

We consider a logistic model for the presence of eggs using as the log of the depth as covariate

```
library(mgcv)
logit.gam <- gam(Presence ~ s(ldepth, bs="ps"),family=binomial)
plot(logit.gam)
```

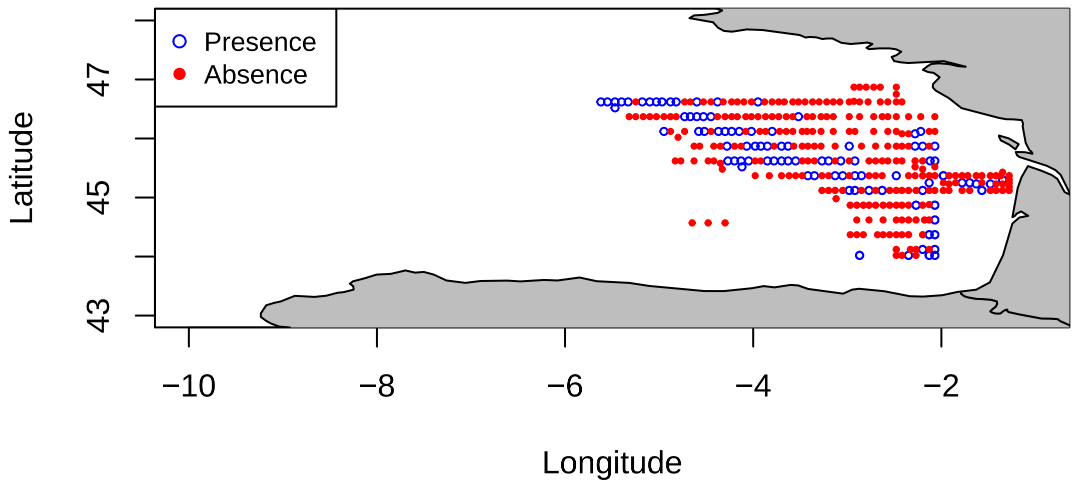


Figure 63: *Sampling positions with presence and absence of eggs*

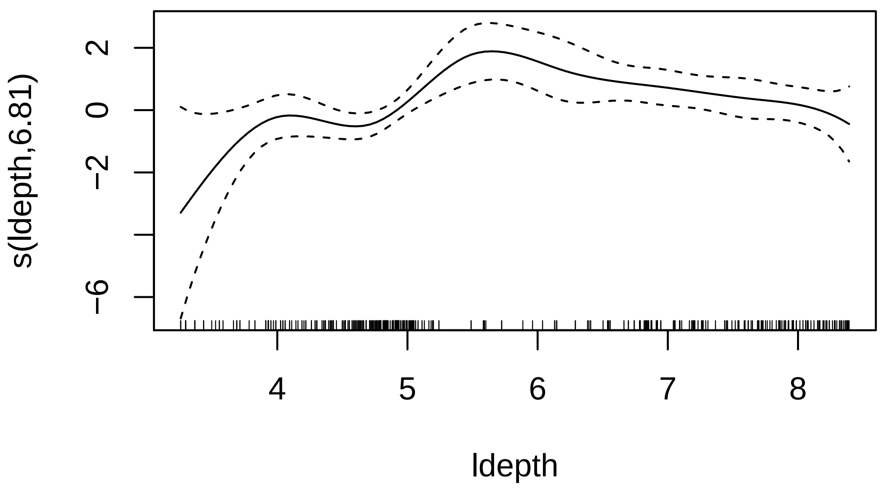


Figure 64: *logistic regression estimate of the relationship between presence and log of depth.*

```r
logit1 <- gam(Presence~te(Longitude,Latitude,bs="ps")+
                s(ldepth,bs="ps")+s(Temperature,bs="ps"),family=binomial)
summary(logit1)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## Presence ~ te(Longitude, Latitude, bs = "ps") + s(ldepth, bs = "ps") +
##     s(Temperature, bs = "ps")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.3326     0.4829   -4.83 1.36e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                           edf Ref.df Chi.sq  p-value
## te(Longitude,Latitude) 18.688 19.711 53.028 0.000109 ***
## s(ldepth)               6.433  7.154 11.707 0.111374
## s(Temperature)          1.000  1.000  5.359 0.020631 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.315   Deviance explained = 32.3%
## UBRE = -0.095829  Scale est. = 1         n = 417
```

```r
logit2 <- gam(Presence~te(Longitude,Latitude,bs="ps")+
                s(ldepth,bs="ps"),family=binomial)
summary(logit2)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## Presence ~ te(Longitude, Latitude, bs = "ps") + s(ldepth, bs = "ps")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.8482     0.4118  -4.488 7.18e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

```
##                             edf Ref.df Chi.sq  p-value
## te(Longitude,Latitude) 19.214 20.269  56.46 4.49e-05 ***
## s(ldepth)               6.708  7.364  11.98    0.104
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.302   Deviance explained = 31.4%
## UBRE = -0.085669  Scale est. = 1         n = 417
```

```
logit3 <- gam(Presence~te(Longitude,Latitude,bs="ps")+
                s(Temperature,bs="ps"),family=binomial)
summary(logit3)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## Presence ~ te(Longitude, Latitude, bs = "ps") + s(Temperature,
##     bs = "ps")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.0247     0.4208  -4.811  1.5e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                             edf Ref.df Chi.sq  p-value
## te(Longitude,Latitude) 18.164  19.34  56.44 1.24e-05 ***
## s(Temperature)          6.547   7.22  13.82   0.0589 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.301   Deviance explained = 30.9%
## UBRE = -0.086039  Scale est. = 1         n = 417
```

```
par(mfrow=c(1,2))
plot(logit3,scheme=2,1)
```

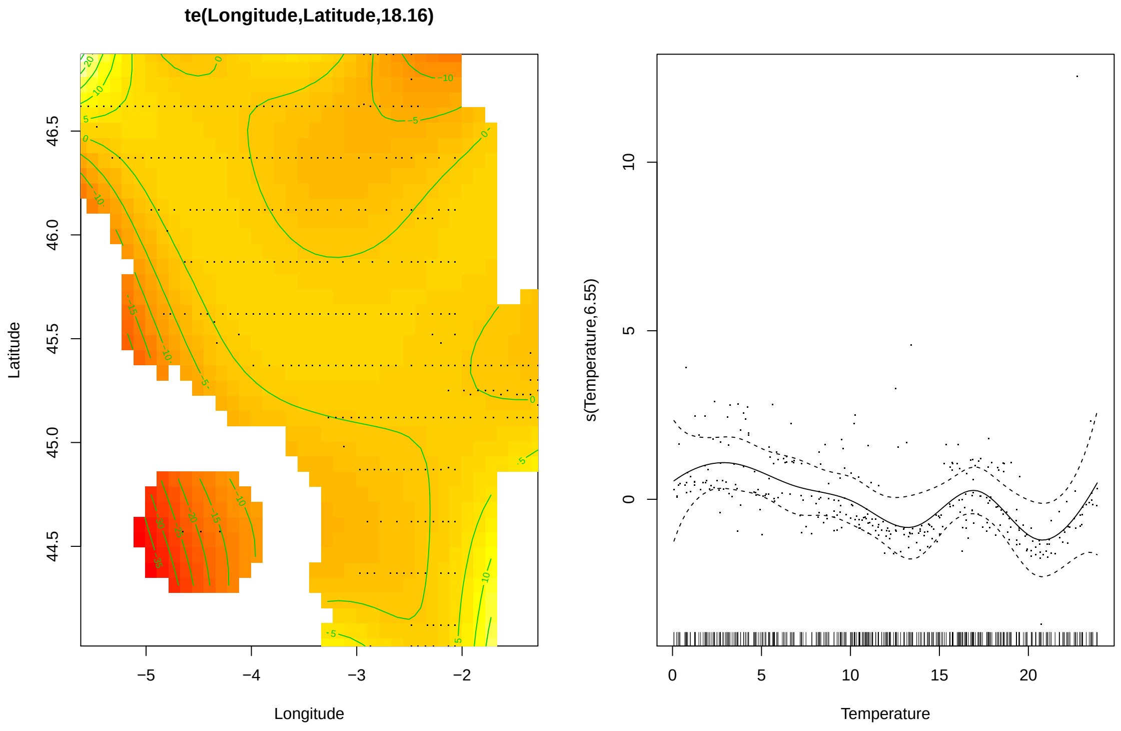


Figure 65: *Smooth term of `logit3` model*

5 Exercise

The aim is to model data on the results of a spinal operation “laminectomy” on children, to correct for a condition called *kyphosis*. The dataset `kyphosis.txt` contains data on 81 children and the following variables:

- `Kyphosis`: a response factor with levels absent present.
- `Age` of child in months, a numeric vector
- `Number` of vertebra involved in the operation,a numeric vector
- `Start`: the vertebra position for which the child start having problems

Start by fitting a model where all variables enter as linear, then, include them as smooth terms, check if there is one or more that should enter as linear, check if a 2d-smooth should be included.

References

Boor, C. de. 1978. *A Practical Guide to Splines.* Berlin: Springer.

Brumback, B.A., and J.A. Rice. 1998. “Smoothing Spline Models for the Analysis of Nested and Crossed Samples of Curves.” *J. Am. Statist. Assoc.* 93 (443): 961–94.

Chaudhuri, P., and J.S. Marron. 1999. “SiZer for Exploration of Structures in Curves.” *J. Am. Stat. Assoc.* 94: 807–23.

Crainiceanu, C., D. Ruppert, G. Claenkens, and M.P. Wand. 2004. “Likelihood Ratio Tests of Polynomial Regression Against a General Nonparametric Alternative.” *Biometrika* to appear.

Crainiceanu, C., D. Ruppert, G. Claeskens, and M.P. Wand. 2005. “Exact Likelihood Ratio Tests for Penalised Splines.” *Biometrika* 92 (1): 91–103.

Craven, P., and G. Wahba. 1979. “Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Cross-Validation.” *Numer. Math.* 31: 377–403.

Currie, I. D., and M. Durbán. 2002. “Flexible Smoothing with *P*-Splines: A Unified Approach.” *Statistical Modelling* 2: 333–49.

Dierckx, P. 1993. *Curve and Surface Fitting with Splines.* Oxford: Clarendon Press.

Eilers, P. H. C. 1999. “Comment on ‘The Analysis of Designed Experiments and Longitudinal Data Using Smoothing Splines by Verbyla et Al.”’ *J. Roy. Stat. Soc. C* 48: 269–312.

Eilers, P. H. C., and B. D. Marx. 1996. “Flexible Smoothing with *B*-Splines and Penalties.” *Stat. Sci.* 11: 89–121.

Green, P.J., and B.W. Silverman. 1994. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach.* Monographs on Statistics and Applied Probability. London: Chapman & Hall.

Greven, Crainiceanu, S., H. Küchenhoff, and A. Peters. 2008. “Restricted Likelihood Ratio Testing for Zero Variance Components in Linear Mixed Models.” *Journal of Computational and Graphical Statistics* 17 (4): 870–91.

Ngo, L., and M.P. Wand. 2004. “Smoothing with Mixed Model Software.” Department of Biostatistics, School of Public Health, Harvard University.

O’Sullivan, F. 1986. “A Statistical Perspective on Ill-Posed Inverse Problems.” *Statistical Science* 1: 505–27.

Pandit, S., and S.-M. Wu. 1983. *Time Series and System Analysis with Applications.* Wiley.

Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression.* Cambridge Series in Statistical and Probabilistic Mathematics. UK: Cambridge University Press.

Self, S.G., and K. Liang. 1987. “Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Nonstandard Conditions.” *J. Am. Stat. Assoc.* 82: 605–10.

Speed, T. 1991. “Comment on "BLUP Is a Good Thing: The Estimation of Random Effects", by Robinson, G.K.” *Stat. Sci.* 6: 15–51.

Stram, D., and J. Lee. 1994. “Variance Components Testing in the Longitudinal Mixed Effects Model.” *Biometrics* 50: 1171–7.

Verbyla, A.P., B. Cullis, M. Kenward, and S. Welham. 1999. “The Analysis of Designed Experiments and Longitudinal Data Using Smoothing Splines.” *J. Roy. Stat. Soc. C* 48: 269–312.

Wand, M. P. 2003. “Smoothing and Mixed Models.” *Computational Statistics* 18: 223–49.

Wood, S. N. 2003. “Thin Plate Regression Splines.” *J. R. Statist. Soc. B* 65 (1): 95–114.

———. 2006. *Generalized Additive Models - an Introduction with R.* Texts in Statistical Science. Chapman & Hall.