Università degli Studi di Udine
and
Università degli Studi di Trieste

Dipartimento di Scienze Economiche e Statistiche - DIES
and
Dipartimento di Scienze Economiche, Aziendali, Matematiche e Statistiche "Bruno de Finetti" - DEAMS
Ph.D. degree in Management and Actuarial Sciences - XXIX Cycle

Some Developments in Flexible Regression Modelling

Candidate:
Andrea Lattuada

Supevisor:
Prof. Nicola Torelli

2017

Contents

Introduction

Flexible regression methods are becoming a topic of primary interest also in actuarial science.

Insurance business is based on the ability of an insurance company to undertake risks from policyholders and this is possible only if sufficient assets are available. The typical characteristic of an insurance company is that the assumed risks originate liabilities whose effective amount is unknown at the moment of the subscription of a policy.

In the last decades, insurers began to pay increasingly attention in assessing the risk arising from their activity and thus researchers developed several methods in order to evaluate it correctly. In the most important world insurance markets efforts were placed in order to adequate laws and to lead the companies on this way. Law also defines what is meant by the requirement of the assets to be 'sufficient' by the choice of the risk measure to adopt.

In some cases such as in the European Solvency 2 directive, the insurance companies are also invited to develop their own models in order to evaluate the riskiness of the activity rather than to rely on a standard model that is calibrated in order to be efficient on the market as a whole, but that may fail in particular cases.

Modelling the risk of an insurance company is a very complicated task as a huge number of quantities should be involved both in the computation of assets and liabilities.

As described, many of those quantities are in general unknown and hence they should be modelled as random variables whose value is determined by some kind of event. In some cases those events are unobservable or they are so rare that the models should rely on experts knowledge rather than on observed data. On the converse, some other of them are often observed.

The prudent actuary knows that what we observe is affected by variability, the features of the variables can be studied by sound and refined statistical models. In several frameworks this is possible by studying -on the observed data- the relationship between one variable which is the object of interest and other non random ones and for this purpose the natural candidate is a regression model.

One of the most used regression models, the linear model, is a parametric model that can be adopted to obtain estimates of the expected value of one variable as a linear function of one (or many) other(s). In order to implement those models, however, some assumptions are usually made.

The variables involved may be diverse and -in some cases-, even apparently unrelated. Even if the linearity assumption can be somehow relaxed, still relying on the same parametric framework, often the relationship could not be clear a priori and hence flexible regression methods become very useful for these purposes. Moreover the usual assumptions that are made on the distribution of the response variable may result unrealistic in several practical situations. Often, the variables involved in the insurance fields are counts or they are for sure non-negative and hence the usual Gaussian assumption fail.

The actuary should hence often use in practice such kind of models and rely on the results obtained in order to assess the risk as well as possible.

Flexible regression models have been widely studied in recent years and increasing interest emerged as modern computing allows also the researchers to efficiently implement them.

In the first chapter we will first present some well known methods for flexible regression, highlighting the main features as well as drawbacks and practical problems arising. This overview will come to be useful also in order to contextualize the following steps of this thesis. We will also emphasize some recent works in the actuarial field on which those models have been applied even if with the introduction of some modifications.

In the second chapter of this work we will present a novel approach for non-parametric regression named Generalized Geometrically Designed Splines regression. We will describe in detail this method and we will introduce some extensions. We will present its generalization to allow the model to deal with data whose distribution comes from the Exponential Dispersion family and we will present its extension in the multivariate setting. We will then study its features, highlighting strengths and weaknesses in a comparison with other methods. This part will also include several examples from various fields in which this approach provides quite interesting results.

As the aim is to produce a method really available, we will also present its implementation in an R package, named **GeDS**. We will then focus also on some practical aspects of the implementation and we will study the performances of the method in a thorough simulation study.

In the third chapter, we will see a very practical application of non parametric regression in the non-life insurance framework. We will merge some of those models with the aim to produce an approach for ratemaking, whose results can be applied in practice. We will see that these models can perform better than common practice.

Notation remarks

We provide here a short list of the non immediate notation adopted in what follows. The meaning of most of the symbols can be evicted from the text, but still we hope that this page could be useful for the reader.

Throughout this work we will denote vectors with lower case bold (possibly greek) letters. Matrices, on the converse, will be denoted by upper case bold letters. Unless otherwise defined, vectors will be assumed as column vectors, hence the vector whose elements are $a_1, a_2, \ldots, a_N$, will be denoted as $\boldsymbol{a}$ and we will assume $\boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_N \end{bmatrix}^{\mathrm{T}}$. Both with vectors and matrices, we will denote their elements using subscripts.

With a slight abuse of notation when we will need to apply the same univariate function f to all the elements of a vector $\boldsymbol{a}$ we will use the notation $f(\boldsymbol{a})$. Hence $f(\boldsymbol{a}) := \begin{bmatrix} f(a_1) & f(a_2) & \cdots & f(a_N) \end{bmatrix}^{\mathrm{T}}$.

Let us stress the difference between the symbols $I_B(\cdot)$ and $\boldsymbol{I}$: the first denotes the index function, i.e., given a set B,

$$I_B(x) = \begin{cases} 1 & \text{if } x \in B \\ 0 & \text{if } x \notin B \end{cases},$$

while the second denotes the identity matrix. Whenever the dimension of the matrix is known, we will indicate it as a subscript (e.g. $\boldsymbol{I}_h$ is the $h \times h$ identity matrix).

Note that we will use also the symbol $\boldsymbol{I}(\cdot)$, denoting the Fisher information matrix stressing out its dependence on some parameter.

$\otimes$ will indicate the Kronecker product. Hence

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & a_{1,2}\boldsymbol{B} & \cdots & a_{1,n}\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ a_{m,1}\boldsymbol{B} & a_{m,2}\boldsymbol{B} & \cdots & a_{m,n}\boldsymbol{B} \end{bmatrix},$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively $n \times m$ and $p \times q$ matrices and the resulting an $np \times mq$ matrix.

$\circledast$ will denote an operation between two matrices that will be useful in the computation of tensor product splines. Considering the $m \times n$ matrix $\boldsymbol{A} := \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_m \end{bmatrix}^{\mathrm{T}}$ and the $m \times p$ matrix $\boldsymbol{B} := \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_m \end{bmatrix}^{\mathrm{T}}$,

$\boldsymbol{A} \circledast \boldsymbol{B} = \boldsymbol{C}$ with $\boldsymbol{C}$ the $m \times np$ matrix defined as $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_m \end{bmatrix}^{\mathrm{T}}$, with $\boldsymbol{c}_i = \boldsymbol{a}_i \otimes \boldsymbol{b}_i$.

Other symbols that will appear in the text are:

$\mathbb{R}$:	the set of real numbers;
$\mathbb{N}$:	the set of natural numbers;
$\left\lVert \cdot \right\rVert_1$:	either the norm of the L^1 or l^1 spaces, depending on the argument;
$\left\lVert \cdot \right\rVert_2$:	either the norm of the L^2 or l^2 spaces, depending on the argument;
$\left\lVert \cdot \right\rVert_\infty$:	either the norm of the L^∞ or l^∞ spaces, depending on the argument;
$\mathcal{C}^{(m)}$:	$\{f : D \to \mathbb{R} \vert f$ is $mtimescontinuouslydifferentiableandD \subseteq \mathbb{R}\}$;
$\mathrm{rk}(\cdot)$:	the rank of a matrix;
$\mathrm{tr}(\cdot)$:	the trace of a matrix.

Chapter 1

Introduction to flexible regression methods

Flexible statistical methods have been introduced in several fields and they now represent a framework that allows the researcher to study statistical properties of observed data avoiding the imposition of some strong assumptions. Those methods, compared to parametric ones, allow to study data requiring in general less assumptions on the structure of the model itself, such as the shape of the dependence structures between variables. The aim of non-parametric models is to learn this structure directly from the data themselves.

A wide class of those models is the class of semi parametric and non-parametric regression methods. As in parametric regression, those models allow to study the dependence of a variable (the response variable) upon a set of other variables (the covariates or the terms).

In this chapter we outline the main flexible regression methods that have been introduced and we will present some applications in the actuarial framework.

1.1 Parametric regression

A typical regression problem can be represented as a model where a researcher observe couples $\{y_i, \boldsymbol{z}_i\}_{i=1}^n$, where $\boldsymbol{z}_i = \{z_{ij}\}_{j=1}^h$ and aims to study the relationship between $\boldsymbol{y}$ and $\boldsymbol{z}$ by estimating the function f in the model

$$\mathrm{E}[\boldsymbol{y}] = f(\boldsymbol{z}).$$

In parametric regression it is necessary to assume a model for the function f, that usually is allowed to vary according to a finite number of parameters. Hence the resulting estimation procedure will take place in a finite dimensional space.

1.1.1 Linear regression

When f is assumed to be a straight line or -more generally- an hyperplane, the model becomes

$$\boldsymbol{y} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{1.1}$$

where $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_h]^{\mathrm{T}} \in \mathrm{I\!R}^h$ is the parameter that should be fitted and $\boldsymbol{Z} = [1, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_h]^{\mathrm{T}}$ is the matrix that collects the $\boldsymbol{z}$ vectors and it is augmented with a vector of 1, allowing also an additive constant, the intercept, to be estimated. The vector $\boldsymbol{\epsilon}$ represents the error term and some assumptions are made on it. The simplest are that $\mathrm{E}[\boldsymbol{\epsilon}] = \boldsymbol{0}$ and $\mathrm{Var}[\boldsymbol{\epsilon}] = \sigma^2\boldsymbol{I}_n$, where σ^2 is constant and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.

The coefficients of this model can be estimated via least squares, i.e.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathrm{I\!R}^h}{\arg\min} \left\|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\beta}\right\|_2^2. \tag{1.2}$$

and, given this approach, it is natural to consider as a Goodness of Fit measure the Residual Sum of Squares, that is

$$\mathrm{RSS}(\boldsymbol{\beta}) = \left\|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\beta}\right\|_2^2.$$

The farther is the observation y_i from its fitted value $\boldsymbol{z}_i\boldsymbol{\beta}$, the higher is its contribution to the RSS. However this relationship is not linear as we consider squares, hence this procedure allows the researcher to find an estimate in an optimal way and the selection of the curve will be performed giving implicitly a heavier weight to the observed data that are far from the fitted ones.

A more theoretical justification of this approach can be achieved making a further assumption on $\boldsymbol{\epsilon}$. Assuming that $\boldsymbol{\epsilon} \sim NID(0, \sigma^2\boldsymbol{I}_n)$, the least squares estimator of (1.2) procedure coincides with the Maximum Likelihood Estimator.

We also point out that, provided that $(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}$ exists, it is possible to obtain least squares estimates explicitly as $\hat{\boldsymbol{\beta}} = (\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{y}$ and that it is possible to get easily the so called hat matrix $\boldsymbol{H} = \boldsymbol{Z}(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{T}}$ such that the fitted values are $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$.

This model structure is simple and very popular, but there are a lot of practical situations where the linearity assumption should be refused.

In some simple cases, more flexible models can be obtained still within the parametric framework with some generalizations of the linear regression. In this case, one can consider a transformation of the response variable that can linearize the dependence structure. If it is possible to assume a regression structure such as $\boldsymbol{y} = f(\boldsymbol{\beta}\boldsymbol{Z} + \boldsymbol{\epsilon})$, with $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n)$ and if the inverse function f^{-1} is known, it is possible to transform the response variable as $\boldsymbol{y}^* = f^{-1}(\boldsymbol{y})$ and to perform a linear regression according to the model $\boldsymbol{y}^* = \boldsymbol{\beta}\boldsymbol{Z} + \boldsymbol{\epsilon}$.

However those models are still not flexible and the function f or at least its inverse, should be chosen ex-ante. This may not be trivial in some applications and care should be taken considering that f^{-1} may even not exist.

1.1.2 Polynomial regression

More flexibility can be achieved imposing a polynomial structure of the regression but still relying on the parametric framework. This model can be obtained just modifying (1.1) by substituting the matrix $\boldsymbol{Z}$ with a design matrix. Supposing we have only one covariate and we aim to estimate f as a mth degree polynomial, the design matrix should be $\boldsymbol{X} = \left[1, \boldsymbol{z}, \boldsymbol{z}^2, \dots, \boldsymbol{z}^m\right]$ and estimates of the vector of coefficients $\boldsymbol{\beta}$ can be performed by least squares.

The flexibility of a polynomial curve can be increased by increasing the degree. Nonetheless, this increases also the complexity of the model and there is no guarantee upon whether an higher degree polynomial will fit better the data or it will result just in a wigglier estimate.

Bias-Variance trade-off

It is now time to introduce a concept we will have to deal with in almost all the remainder of this thesis. This is the theoretical basis of what we described just intuitively above.

A simple and understandable presentation of this problem can be found in [Azzalini and Scarpa (2004)], where it is related to the famous Ockham razor.

With a simple example of data simulated trough a model similar to the one presented in Section 1.1 they introduce the problem and then they make some more general considerations.

Suppose $\hat{f}_m$ is some kind of estimator for f, where the subscript m denotes its complexity. For instance, in the polynomial regression, m can denote the maximum degree.

When performing a regression, the aim of the researcher is to obtain an estimate of f allowing to compute predictions rather than a perfect match between the model and data observed. Hence, a Goodness of Fit measure is the prediction error,

$$\mathrm{E}\left[\hat{f}_m(z^*) - f(z^*)\right] = \left(\mathrm{E}\left[\hat{f}_m(z^*)\right] - f(z^*)\right)^2 + \mathrm{Var}\left[\hat{f}_m(z^*)\right] \tag{1.3}$$

where z^* is a new observation, while the estimate is based on the old observed data. The first term in the RHS of (1.3) is the bias of the model, while the second is its variance.

It can be shown that increasing the complexity m, the bias reduces, while the variance increases. Hence, up to some level of complexity, the prediction error reduces, while then it increases.

Bias and variance are two conflicting quantities in a model. Some compromises has to be done in order to choose an 'optimal' value of complexity.

We will see several strategies to take this choice. Some of them are based explicitly on a minimization problem of this kind, such as the model selection criteria of Section 1.5.1, while other models address this issue implicitly.

1.2 Interpolation

It is useful to spend also some lines on approximation theory, before considering the general model for semi-parametric or non-parametric regression. Our aim is just to provide some ideas that allow and thence that justify the use of (piecewise) polynomials in order to approximate the unknown functions that are the objective of the regression.

We refer here to univariate interpolation, hence we refer to the problem of finding an estimate of a function f based on the observed couples $(x, y) = (x, f(x))$, hence we are in a framework where there is no randomness in the data, or at least where it is considered.

Suppose we aim at approximating a function f where our data are just couples $\{\tau_i, f(\tau_i)\}$ $i = 1, \ldots, n$, such that $\tau_i < \tau_j$ if $i < j$. In order to get an estimate $\hat{f}$ of the function f a way to proceed is to sequentially connect those points by linear pieces, thus considering the so called broken line interpolation.

$\hat{f}_2$, where the subscript indicates the order of the approximation can be defined as

$$\hat{f}_2(x) = f(\tau_i) + \frac{x - \tau_i}{\tau_{i+1} - \tau_i}(f(\tau_{i+1}) - f(\tau_i))$$

for $x \in (\tau_i, \tau_{i+1})$, $i = 1, \ldots, n-1$.

This way to interpolate a function is very simple and intuitive, but yet it has a non negligible property. Using the divided differences notation*,

$$f(x) = f(\tau_i) + (x - \tau_i)[\tau_i, \tau_{i+1}]f + (x - \tau_i)(x - \tau_{i+1})[\tau_i, \tau_{i+1}, x]g.$$

*Given the couples $x_i, g(x_i)$, $i = 1, \ldots, m$,

$$[x_i]g := g(x_i),$$

while

$$[x_i, x_j, \ldots, x_k, x_l]g := \frac{[x_j, \ldots, x_k, x_l]g - [x_i, x_j, \ldots, x_k]g}{x_l - x_i},$$

where $i, j, k, l \in 1, \ldots, m$.

It can be shown that for $x \in (\tau_i, \tau_{i+1})$,

$$\left|f(x) - \hat{f}_2(x)\right| \leq (x - \tau_{i+1})\,(x - \tau_i) \max_{t \in (\tau_i, \tau_{i+1})} \left|\frac{f''(t)}{2}\right| \leq$$

$$\leq \left(\frac{\tau_{i+1} - \tau_i}{2}\right)^2 \max_{t \in (\tau_i, \tau_{i+1})} \left|\frac{f''(t)}{2}\right| \quad (1.4)$$

assuming f has at least two continuous derivatives [De Boor (2001)].

However the resulting fit lacks of other properties that may be required in a wide range of situations. It is quite common to require the fitted curve to be a smooth function, but this in not possible with a piecewise linear function. There are however some other models that may help in constructing a more efficient approximation.

One possible strategy to get a smooth approximation is to consider a high degree polynomial. Considering again the function f, the couples $\{\tau_i, f(\tau_i)\}$ $i = 1, \ldots, n$, one can obtain an other approximation $\hat{f}$ as

$$\hat{f}(x) = \sum_{i=1}^{n} (x - \tau_1) \cdots (x - \tau_{i-1}) [\tau_1, \ldots, \tau_i] f$$

and thus one gets a piecewise polynomial of degree $n - 1$.

Still some problems arise with this strategy. Considering the L_∞ norm $\|h\|_\infty = \max_{a<x<b} |h(x)|$ a measure of the approximation error is $\|f - \hat{f}\|$ and it is possible to show that in some cases it does not decrease at a reasonable rate or even it increases (see [De Boor (2001)], pag. 19) as n increases. Moreover, increasing n, also the complexity of the polynomial involved increases. This is indeed the same issue arising when performing a parametric regression having assumed a polynomial structure of the dependence.

Another approach where smoothness is pledged and generally performs better than a plain polynomial is to consider a subdivision of the domain in several smaller intervals. Hence piecewise polynomial approximation can be built imposing some constraints on the behaviour of single pieces on the bounds of the intervals in order to ensure the smoothness. Under this approach, increasing n would not lead to increase the complexity of the model in the sense that there will not be included higher degree terms. However it should be taken into account that the number of parameters involved can match the one of the polynomial approximation. In these models the bounds of the intervals, that are the locations where two polynomial pieces join, are called *knots*.

One popular way to implement it relies in using a piecewise cubic interpolant. Given a function f and the couples $\{\tau_i, f(\tau_i)\}$, $i = 1, \ldots, n$, such that $\tau_i < \tau_j$ if $i < j$, a piecewise cubic interpolant $\hat{f}_4$ is a function such that, in each interval $[\tau_i, \tau_{i+1}]$, $\hat{f}_4$ is a cubic polynomial, $\hat{f}_4(\tau_i) = f(\tau_i)$, $\hat{f}_4(\tau_{i+1}) = f(\tau_{i+1})$, $\hat{f}_4'(\tau_i) = s_i$ and $\hat{f}_4'(\tau_{i+1}) = s_{i+1}$ for $i = 1, \ldots, n$.

There are some strategies for the choice of the values s_i, among the others it is worth mentioning the Hermite, the Akima and the Bessel approximations that prove to be very efficient ([De Boor (2001)]). In particular, in the Hermite interpolation, it can be shown that $\left|\left|f - \hat{f}_4\right|\right| = O(n^{-4})$, while in the Bessel interpolation, the order of the error is $O(|\boldsymbol{\tau}|^3)$, where $|\boldsymbol{\tau}|$ is the mesh size.

A more refined strategy is to consider the cubic spline interpolation. Here the parameters $s_2, \dots, s_{n-1}$ are chosen in order to ensure that the estimated function belongs to the function space $\mathcal{C}^{(2)}$, i.e. it is twice continuously differentiable. While the s_i, $i = 2, \dots, n-1$ should be chosen according to a linear system, the choice about the boundary values s_1 and s_n can be made according to other conditions.

A widely used choice is to find s_1 and s_n that allow $\hat{f}_4''(\tau_1) = \hat{f}_4''(\tau_n) = 0$ and it is called the natural cubic spline. There is no theoretical justification to make this assumption, but we will see in the remainder that those splines arise as the solution of an optimization in a penalized regression problem. With this choice, the order of the error of approximation is $O(|\boldsymbol{\tau}|^2)$, but other strategies can lead to $O(|\boldsymbol{\tau}|^4)$.

Cubic splines can be generalized to *polynomial splines* of order l. A spline of order l defined on the knot mesh $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$ is a piecewise polynomial function belonging to the function space $\mathcal{C}^{(l-2)}$, i.e. it can be continuously differenced $l-2$ times and it corresponds to a polynomial of degree $l-1$ on the intervals $[\tau_i, \tau_{i+1}]$, $i = 1, \dots, m-1$.

1.3 Flexible regression

We will distinguish between two families of flexible regression methods: non-parametric and semi-parametric regressions.

Both those families represent sets of tools that allow the researcher to build a flexible regression model and hence to get an estimate the function f. In some cases a visual inspection of the results may not be sufficient to understand to which of the two families the method belongs. However this distinction is important and we will see it is effective in some frameworks, such as in Bayesian inference.

In a purely non-parametric regression, the fitting procedure should take place in an infinite dimensional function space. In order to make it feasible to find the solution, usually some weak assumptions on f are made and in some cases the estimate can be characterized in terms of a finite number of parameters. However the number of parameters is not set a priori and it may depend on the observed data. We discussed in the previous section how powerful these approximations can be.

On the converse, in semi-parametric regression the estimate belongs to a finite (but high) dimensional space with the number of dimensions set in advance. The high number of dimensions of the parameter space allows flexibility of these models and the estimates can be summarized in an a priori defined space.

In both non-parametric and semi-parametric regressions, it is important to notice that the parameters in general cannot be interpreted directly. While, for instance, a positive coefficient in the parametric linear model means that there is a positive correlation between the variables, here no meaningful information can be recovered by looking at the value of a single coefficient.

In what follows, we will see some of the main approaches to build these kind of models, but let us set here a remark. Although we are still describing this class of models in general terms, we can already explain it as it arises from the philosophy underlying these models themselves.

With the flexible regression models we will introduce, we will be able to obtain estimates in a lot of contexts where parametric models may fail. We will see that, within the domain of the observed values, those models will often be very efficient and it is possible to use them to compute predictions. Nonetheless, they are useless for the purpose of extrapolation outside the domain.

Conceptually, being the dependence structure an assumption in parametric models, the researcher is allowed to take advantage of it also outside the interval of observed values. Provided this assumption is correct, meaningful predictions can be computed for values outside the interval as well as for values inside it. In flexible regression models, instead, no structure is assumed and extrapolation would be in the best case misleading.

1.4 Local Smoothing

Local smoothing is one of the classes that allow to perform a flexible regression in the non-parametric framework. As discussed previously there are some technical reasons that explain why local piecewise regressions should be preferred with respect to a high order polynomial. Here we present some methods to implement such kind of models, even if they are less refined compared to the regression splines we will present in the following sections.

These methods do not consider a partition of the domain such as the ones we introduced for interpolation, but rather a moving window. Hence each fitted value will be based only on data observed in its neighbourhood, or on all the observed data, but with a different weighting.

There are three methods that are widely used and that are nowaday included as basic estimators in many statistical softwares: Nearest Neighbour, Local Polynomial and Loess Estimators.

Nearest Neighbour Estimator

The first class of those estimators, that produces rough results, is composed by the nearest neighbours ones. The estimate is performed according to $\hat{f}(z) = \text{Ave}_{j \in J(z)}(y_j)$, where Ave is an average operator chosen by the researcher, while $J(z)$ is the subset of the subscripts identifying the neighbourhood of z. This class is quite flexible, as it is possible to choose both the averaging operator and the definition of "neighbourhood", but in general continuity of the estimates cannot be ensured. Suppose in a point z_i we obtain the estimate $\hat{f}(z_i)$. The estimate will be the same for all the points in the window interval interval $W = \{z|J(z) = J(z_i)\}$, while in general $\hat{f}(z_i) \neq \hat{f}(z_j)$ if $z_j \notin Z$. The estimate is a piecewise constant function and continuity can be ensured only in the trivial case where $\boldsymbol{y}$ is a vector of all equal values.

Local Polynomial Estimator

By means of the Taylor expansion, it is possible to locally approximate any l-times continuously differentiable function with an l-degree polynomial. Hence the idea is to fit a polynomial around each observed point (z_i, y_i) $i = 1, \dots, n$. The estimation procedure is, in general performed making use of the weighted least squares, where the weighting is chosen according to a kernel function $\mathcal{K}$, with a suitably chosen bandwidth. A kernel function of limited support (such as e.g. the Epanechnikov one) will result in a local regression, while with other kernels (e.g. the Gaussian one), all the observed data will influence the regression, even if their weight will be lower according to the distance.

Loess

Nearest Neighbour Estimator can be modified in order to achieve smoother estimates by introducing also in this case weighting chosen according to a kernel function. When the average operator is the linear regression and the kernel function is the tricubic one i.e. $\mathcal{K}(u) = (1 - |u|^3)^3$ it is named after LOESS or LOWESS (that stands for LOcally (Weighted) Estimator). Choosing the weighting according to a kernel function guarantees continuity of the estimates. Even if the regression is local, moving the window interval, the weight of the datapoints will decrease toward zero continuously and hence jumps will be avoided.

1.5 Regression Splines

We introduced polynomial splines in Section 1.2, defined as a piecewise polynomial function with some properties about the smoothness.

A spline function can be represented in several ways that allow it to be practically tractable. We will see the Truncated Polynomial and the B-spline representations.

For a wide and practical description of spline functions we refer to [De Boor (2001)], while in the next subsection we will just point out some basic aspects that are necessary in order to understand what follows.

Truncated polynomial

Any l-order polynomial spline g with knots $\boldsymbol{\tau}$ whose elements are $\tau_1, \dots, \tau_m$ can be represented as

$$g(z) = \sum_{i=1}^{l+m-1} \beta_i B_i(z),$$

where

$$\begin{gathered} B_1(z) = 1 \quad B_2(z) = z \quad \cdots \quad B_l(z) = z^{l-1} \\ B_{l+1}(z) = (z-\tau_2)_+^{l-1} \quad \cdots \quad B_{l+m-1}(z) = (z-\tau_{m-1})_+^{l-1}, \end{gathered} \tag{1.5}$$

with

$$(z)_+ := \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}.$$

The functions $B_1, \dots, B_{l+m-1}$ are called *basis functions* as it is possible to represent any polynomial spline g whose knot vector is $\boldsymbol{\tau}$ as a linear combination of them.

The first l of those basis functions are positive on the whole function domain while the $l+1$th is positive everywhere, but in the interval defined by the first two knots and so forth up to the last one, which is positive only on the interval defined by the last two knots. Although in the next section we will see that other functions defined only locally can be a basis for polynomial splines, this is quite important, as in practice this has some effects on the estimation results in some particular frameworks, as we will be able to comment in Section 2.3.1.

B-Splines

B-splines are an other elegant set of bases that allow the representation of splines in term of a set of coefficients. In this section we give just the definition and we point out some basic properties that will be useful in the remainder of this Chapter. We leave for the next Chapter other detailed properties that will be crucial in the description of the GeDS method.

B-splines were first studied by Schoenberg who gave the first definition of them. Given a non decreasing set of knots $\boldsymbol{\tau}$, the jth B-spline can be defined as

$$N_{j,l}(z) := (\tau_{j+l} - \tau_j)[\tau_j, \dots, \tau_{j+l}](\cdot - z)_+^{l-1}.$$

As a first property, we can see that B-splines have a local support, i.e.

$$N_{j,l}(z) = 0 \text{ iff } z \notin [\tau_j, \dots, \tau_{j+l}]. \tag{1.6}$$

A second property is the so called *partition of the unity property*, i.e.

$$\sum_{j=r}^{s} N_{j,l}(z) = 1 \tag{1.7}$$

on the interval $[\tau_{r+l-1}, \tau_{s+1}]$, thus they are in a way "normalized".

There is also an other way to normalize them as, considering the B-splines based on the knot sequence τ, the B splines can be normalized according to

$$M_{j,l}(z) := \frac{l}{t_{j+l} - t_j} N_{j,l}(z)$$

and it is possible to state that

$$\int_{\mathbb{R}} M_{j,l}(z) \mathrm{d}z = 1 \tag{1.8}$$

Another property of B-splines that will be used is the formula that allows to compute the derivatives. Suppose we have the polynomial spline function g and it is known its B-splines representation $g(z) = \sum_{j=1}^{m} \beta_j N_{j,l}(z)$, the derivative with respect to z is

$$\frac{\mathrm{d}}{\mathrm{d}z} g(z) = \sum_{j=1}^{m} \beta_j (l-1) \left(\frac{N_{j,l-1}(z)}{\tau_{j+l-1} - \tau_j} - \frac{N_{j+1,l-1}(z)}{\tau_{j+l} - \tau_{j+1}} \right) =$$

$$(l-1) \sum_{j=1}^{m+1} \frac{\beta_j - \beta_{j-1}}{\tau_{j+l-1} - \tau_j} N_{j,l-1}(z), \tag{1.9}$$

having defined $\beta_0 := \beta_{m+1} := 0$.

Note also that B-splines can be seen as a tool somehow even more flexible than the Truncated power splines. Actually, with B-splines it is possible to achieve also non-smooth or non continuous patterns, if one allows the knots to coalesce. B-splines based on the coalescent knots will be discontinuous or have a non continuous derivative, the order depending on the degree and on the number of coalescent knots. If a spline of order $l > 2$ is based on $q > 1$ coalescent knots, the $l - q$th order derivative will be discontinuous.

Both the spline representations can be used for semi-parametric and non-parametric regression. The regression model has exactly the same structure (1.1), but with the design matrix $\boldsymbol{X}$. This matrix does not have an interpretation as a dataset, but it is rather a collection of the basis functions evaluated at the points corresponding to the observed values. Hence if we have the basis functions $B_1, \dots, B_{l+m-1}$ in the remainder we will often refer to

$$\boldsymbol{X} := [B_1(\boldsymbol{z}), \dots, B_{l+m-1}(\boldsymbol{z})].$$

While in interpolation each data point is considered to be a knot, here there are several possibilities. In semi-parametric regression, $m \ll n$ and the knots should be set a priori according to some strategy. In practice, knots can be set equispaced, spreading on the range of z or, they can match some quantiles of the z values or they can be chosen subjectively.

In non-parametric regression methods as we described previously, the number of dimensions of the parameter space, and hence the number of knots is not placed in advance. Hence, the knots can match the datapoints as in interpolation or they can be set according to some adaptive procedure.

Some care should be used when performing an lth order B-spline based regression, on data $(z_i, y_i)_{i=1}^n$, if the knots are placed uniformly in the interval $[z_1, z_n]$, the bounds of the resulting $\hat{f}$ will be constrained. Actually we will have $\hat{f}(\tau_1) = \hat{f}(\tau_m) = 0$. In order to avoid this, we may place the knots according to $a = \tau_1 \leq \cdots \leq \tau_l \leq z_1 \leq z_n \leq \tau_{m-l} \leq \cdots \leq \tau_m = b$. In particular, throughout the remainder we will always choose coalescent boundary knots, such that $a = \tau_1 = \cdots = \tau_l \leq z_1 \leq z_n \leq \tau_{m-l} = \cdots = \tau_m = b$.

The choice of the number of knots and their location is crucial as they are key factors identifying the resulting spline fit. In general the higher number of knots will lead to the wigglier fit, but also a too parsimonious knot selection will lead to oversmoothing. There are two strategies that can be used to address this issue: one is to consider a penalization proportional to a measure of the wiggliness of the fitted function while a second one is to choose the knot locations in an adaptive way.

We will see the first approach in the next section and then we will see some models that involve an adaptive selection of the knots.

1.5.1 Penalized Splines

One of the most popular strategies adopted with the aim of controlling the wiggliness of the estimates is to modify the objective function by introducing a penalization proportional to a measure of wiggliness of the estimate. This methodology takes the name of Penalized Spline Regression.

An appealing measure of the wiggliness of a function involves its derivatives and in the case of the truncated power basis, this is closely related to the coefficients of the regression. Considering (1.5), the estimate $\hat{f}$ is constituted by two parts, the first one up to the first l bases and the second composed by the others. The local wiggliness of the estimated function is controlled by the coefficients associated to the last $m-2$ bases, while the others act globally on the whole domain. For this reason usually only the coefficients associated to those bases are taken into account in the measure of wiggliness.

Hence a suitable Penalized Least Squares functional that should be minimized to estimate the coefficients is

$$\left\|\boldsymbol{y} - f(\boldsymbol{z})\right\|_2^2 + \lambda\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\beta},$$

where

$$\boldsymbol{D} = \begin{bmatrix} \mathbf{0}_{l\times l} & \mathbf{0}_{l\times k} \\ \mathbf{0}_{k\times l} & \mathbf{1}_{k\times k} \end{bmatrix}$$

and $\lambda > 0$. The introduction of this penalization makes all the estimates coefficients of the truncated basis to be shrunk toward zero and thus the fitted curve will result less wiggly than the one obtained via an un-penalized approach.

It is important to notice that in this situation, the amount of smoothing is controlled by the parameter λ that takes the name of *smoothing parameter*. As the higher λ leads to the smoother fit, the choice of this parameter is crucial.

Before discussing some methods about the choice of λ, we briefly discuss Penalized splines based on B-splines. While with the truncated polynomial basis representation, some coefficients drive the main shape, while others are divers of the local wiggliness, with the B-splines this is no longer the case.

An appealing penalty that can be introduced is

$$\lambda\int_a^b |\hat{f}''(z)|\mathrm{d}z,$$

but it is not trivial in general to state in an explicit way how the integral should be computed. However, by means of equation (1.9), the integral over the whole domain of the first derivative of a spline in the B-spline representation is equal to the sum of differences of the coefficients of the basis splines. This statement is trivial if knots are equispaced, but also considering non equispaced knots, it holds, by means of (1.8).

For higher order derivatives, it is not possible to state an equality, but still it appears that the integrated derivatives are functions of the higher order differences. Hence a popular choice of $\boldsymbol{D}$ is the matrix that allows to compute the differences of the coefficients. Such a solution was first proposed in [Eilers and Marx (1996)], where also the P-splines approach is introduced.

Note also that the Penalized least squares estimates can be obtained efficiently by noticing that

$$\left\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right\|_2^2 + \lambda\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\beta} = \left\|\begin{bmatrix}\boldsymbol{y}\\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\boldsymbol{X}\\ \sqrt{\lambda}\boldsymbol{D}^{1/2}\end{bmatrix}\boldsymbol{\beta}\right\|_2^2, \tag{1.10}$$

where $\boldsymbol{D}^{1/2}$ is any decomposition of the matrix $\boldsymbol{D}$, such that $(\boldsymbol{D}^{1/2})^{\mathrm{T}}\boldsymbol{D}^{1/2} = \boldsymbol{D}$. Hence in order to get the estimates once λ has been selected and once decomposed $\boldsymbol{D}$, the estimation procedure is similar to the one of the least squares and it can be based on the same algorithms.

Selection of the smoothing parameter

The choice of the smoothing parameter λ is crucial as it affects the resulting estimates controlling the amount of smoothing. The experienced researcher may prefer to base the choice on visual inspection of the data and of the resulting fit. If the estimation procedure produces an estimate $\hat{f}$ considered too wiggly, it is advisable to enhance λ, while if $\hat{f}$ is too smooth and does not capture the shape of the data, λ should be lowered.

However some optimality criteria have been proposed in literature and they have been widely implemented in statistical software. We will see three of them based on a model selection approach and one based on the likelihood.

The first of them is based on the minimization of the Cross Validation with respect to λ. By definition, the Cross Validation is

$$CV(\lambda) = \sum_{i=1}^{n} \left(y_i - \hat{f}_{-i}(z_i;\lambda)\right)^2, \tag{1.11}$$

where $\hat{f}_{-i}(z_i;\lambda)$ is the fitted value in z_i obtained deleting the ith couple of data, under the Penalized Spline procedure with smoothing parameter λ. Intuitively, this functional measures how a model is able to fit the data by checking the accuracy of its predictions.

The minimization of this functional cannot be carried out explicitly and hence it is necessary to evaluate it for several values of λ. Suppose, however we select a grid of h possible values for λ in order to get a reasonably useful estimate of the shape of $CV(\lambda)$. (1.11) as it has been described is inefficient, as it requires to fit hn slightly different models on almost the same set of data.

Hence it has been proposed an efficient formula, based on the hat matrix, i.e. the $n \times n$ matrix $\boldsymbol{S}_\lambda = \{S_{\lambda,ij}\}$ such that $\hat{f}(\boldsymbol{z}) = \boldsymbol{S}_\lambda \boldsymbol{y}$. The functional can be rewritten as

$$CV(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}(z_i;\lambda)}{1 - S_{\lambda,ii}}\right)^2,$$

allowing to compute it fitting just one model for each value of λ. Note, in particular that, by (1.10),

$$\boldsymbol{S}_\lambda = \boldsymbol{X}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} - \lambda\boldsymbol{D})^{-1}\boldsymbol{X}^{\mathrm{T}}$$

and it is a generalization of the hat matrix presented for the standard linear model. Hence the number of degrees of freedom of the fitted spline can be expressed by $df_{fit} = \mathrm{tr}\{\boldsymbol{S}_\lambda\}$, while the residual degrees of freedom are defined as $df_{res} = n - 2\,\mathrm{tr}\{\boldsymbol{S}_\lambda\} + \mathrm{tr}\{\boldsymbol{S}_\lambda{}^{\mathrm{T}}\boldsymbol{S}_\lambda\}$.

One popular variant of this criterion is called Generalized Cross Validation and it is an approximation of it, as the values $S_{\lambda,ii}$ are replaced by $\mathrm{tr}\{\boldsymbol{S}_\lambda\}/n$.

A second widely used method we refer to can be found in literature under the name Mallow's C_p [Mallows (1973)] or Un-Biased Risk Estimator [Craven and Wahba (1978)]. This is defined as

$$\mathrm{UBRE}(\lambda) = \frac{\sum_{i=1}^{n}\left(y_i - \hat{f}(z_i;\lambda)\right)^2}{n} - \sigma^2 + \frac{2\,\mathrm{tr}\{\boldsymbol{S}_\lambda\}\sigma^2}{n}$$

where σ^2 represents the variance of the error term and needs to be estimated.

The third method based on a model selection criterion considers the minimization of the Akaike Information Criterion (AIC) with respect to λ. As stated before, the degrees of freedom of the fit can be estimated by the trace of the matrix $\boldsymbol{S}_\lambda$ and hence the formulation of the AIC in the context of the penalized spline regression is

$$\mathrm{AIC}(\lambda) = \log\left\{\sum_{i=1}^{n}\left(y_i - \hat{f}(z_i;\lambda)\right)^2\right\} + 2\frac{\mathrm{tr}\{\boldsymbol{S}_\lambda\}}{n}. \qquad (1.12)$$

In addition to those methods, one of the most popular ones allowing to choose λ is the ML/REML criterion. This method takes advantage of the mixed model representation of the penalized spline regression as the minimization of (1.10) corresponds to the maximum likelihood estimation of the parameters of the model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n), \quad \boldsymbol{\beta} \sim N(\boldsymbol{0}, \tau^2\boldsymbol{D}^{-1}),$$

with $\lambda = \sigma^2/\tau^2$ that is a linear mixed model.

Unfortunately in general there is no guarantee for $\boldsymbol{D}^{-1}$ to exist as $\boldsymbol{D}$ can be singular. In particular, this is exactly the case if we consider the penalty matrix proposed in section for truncated power splines, where we have zero-columns and zero-rows and $\mathrm{rk}(\boldsymbol{D}) = k \leq d = \dim(\boldsymbol{D})$, supposing $\boldsymbol{\beta}$ is a d-dimensional vector. However we can reparametrize the model considering the two vectors $\tilde{\boldsymbol{\beta}}$ and $\boldsymbol{\gamma}$ such that

$$\boldsymbol{\beta} = \boldsymbol{V}\boldsymbol{\gamma} + \boldsymbol{U}\tilde{\boldsymbol{\beta}},$$

where $\boldsymbol{V}$ and $\boldsymbol{U}$ are two matrices respectively $d \times s$ and $d \times (d - s)$. If $\left[\boldsymbol{V}, \boldsymbol{U}\right]$ has full rank, the transformation is one-to-one and if $\boldsymbol{V}^{\mathrm{T}}\boldsymbol{D} = 0$, the $\boldsymbol{\gamma}$ component of $\boldsymbol{\beta}$ does not penalize the regression.

If then $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{U} = \boldsymbol{I}$, this allows us to write $\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\beta} = \tilde{\boldsymbol{\beta}}^{\mathrm{T}}\tilde{\boldsymbol{\beta}}$ and

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{\gamma} + \boldsymbol{X}\boldsymbol{U}\tilde{\boldsymbol{\beta}} + \boldsymbol{\epsilon}$$

that is a mixed model where for the random effects $\tilde{\boldsymbol{\beta}} \sim N(\boldsymbol{0}, \tau^2 \boldsymbol{I}_{d-s})$.

Computations of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ requires some matrix algebra, but once obtained, it becomes possible to choose λ taking advantage of the mixed model representation.

After some calculations, the profile log-likelihood for the matrix $\boldsymbol{\Sigma} := \mathrm{Cov}(\boldsymbol{y}) = \tau^2 \boldsymbol{X}\boldsymbol{U}(\boldsymbol{X}\boldsymbol{U})^{\mathrm{T}} + \sigma^2 \boldsymbol{I}_n$ can be written as

$$l_p(\boldsymbol{\Sigma}) = -\frac{1}{2}\left[\boldsymbol{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left\{\boldsymbol{I} - \tilde{\boldsymbol{X}}\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\right\}\boldsymbol{y}\right] + \\ -\frac{1}{2}\left[\log|\boldsymbol{\Sigma}|\right] - \frac{n}{2}\log(2\pi), \quad (1.13)$$

where $\tilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{V}$. Once got an estimate for $\boldsymbol{\Sigma}$ it is possible to get estimates for τ^2 and σ^2 and hence for λ.

However the estimates of the variances computed via this approach are biased as the number of degrees of freedom is not taken into account properly (see [Wood (2006)]). Hence a modification of this approach has been proposed and became very popular. The so called REstricted (or REsidual) ML criterion [Patterson and Thompson (1971)] considers a transformation of the data that allow to write the likelihood as a product of two likelihoods of stochastically independent data, one as a function of σ and τ and the other as a function of $\boldsymbol{\gamma}$. Thus by maximising the first one it is possible to obtain an estimate of the variances without the bias implied by the loss of degrees of freedom. While this sounds complicated, the resulting log-likelihood to be maximized is similar to the profile log-likelihood as it is

$$l_R(\boldsymbol{\Sigma}) = l_p(\boldsymbol{\Sigma}) - \frac{1}{2}\log|\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\tilde{\boldsymbol{X}}|.$$

Several other representation of this likelihood have been proposed. Among the others, [Harville (1974)] gave a Bayesian interpretation, representing it as a marginal likelihood, while [Smyth and Verbyla (1996)] considered it as a conditional likelihood.

1.5.2 Adaptive methods

As presented earlier, splines are controlled by two sets of parameters: knots and coefficients. In the previous section we presented some popular methods that act in the choice of the coefficients allowing to obtain a sufficiently smooth estimate $\hat{f}$. Here we present the class of the Adaptive smoothers where the idea is to choose the number and the position of the knots according to some optimality criteria, while the estimation of the coefficients is carried out without penalization. In this sense, those approaches belong to the class of non-parametric regression methods.

A well known method of this kind is the Multivariate Adaptive Regression Splines (MARS) algorithm [Friedman (1991)]. The idea behind the algorithm is to build a model parsimoniously adding sequentially new knots in order to get a better fit. The steps of the algorithm are:

i Select a set of basis functions for the regression $\{B_j\}_{j=1}^{M}$;

ii Fit a minimal model, considering only the minimum number of regressors;

iii Extend the model sequentially considering in the regression the basis B_j that optimizes some preset criterion. Iterate this step until a maximum number of basis has been included in the model;

iv Apply backward deletion eliminating one by one the basis function whose deletion causes the lowest reduction of the fit according to some other preset statistic.

Note also that this algorithm was designed in order to apply in multivariate non-parametric regressions. The algorithm described above should hence consider, in step **ii** also cross products between the basis already considered in the model and the candidate ones.

Theoretically this algorithm can be applied with any kind of basis functions, but in general we find it based on reflected pairs of truncated power basis of order two, i.e. $\{(z-\tau_j)_+; (\tau_j - z)_+\}$, having assumed the knot sequence $\{\tau_j | j = 1, \ldots, J\}$. This is because those bases are local, as they are null on one half of $\mathbb{R}$ and hence cross products between the bases on different terms, that occur in multivariate analysis, are null on a wide area of the domain.

The second method we present is the Regression Tree. This method could be considered as a spline based method, but relying only on the particular case of first order splines. Hence no continuity assumption is made and the resulting estimate is a piecewise constant function. Also this method is a non-parametric one as it implements an adaptive choice of knot locations. In this kind of models, the estimate of the unknown f is a function of the form

$$\hat{f}(z) = \sum_{i=1}^{M} c_i I_{R_i}(z),$$

where c_i are constants, $R_1, \ldots, R_M$ is a partition of $\mathbb{R}$ and $I_A(z)$ is the indicator function, i.e.

$$I_A(z) = \begin{cases} 0 & \text{if } z \notin A \\ 1 & \text{if } z \in A \end{cases}.$$

Regression Tree is a method that carries out automatically a partition and it estimates the constants aiming to find a parsimonious model that adapts to the data, according to the least squares criterion.

The idea is first to consider a partition of $\mathbb{R}$ of the type $R_1 = \{z|z \leq \tau_1\}$ and $R_2 = \{z|z > \tau_1\}$ and so to find the parameters characterizing this model $\hat{c}_1$, $\hat{c}_2$ and τ_1 according to

$$\arg\min_{\tau_1} \left[\min_{c_1 \in \mathbb{R}} \sum_{j \in J_1} (y_j - c_1)^2 + \min_{c_2 \in \mathbb{R}} \sum_{j \in J_2} (y_j - c_2)^2 \right],$$

where $J_i := \{j|z_j \in R_i\}$, $i = 1, 2$.

This procedure is iterated by adding at each step a new knot and hence by splitting one of the existing regions (the parent one) into two regions (the leafs or sons) until some stopping criterion is met. Possible criteria are a minimum number of observations in each region or a maximum number of splits.

The tree obtained at this stage can be too complex and it can overfit the data, the procedure provides a successive step, called *pruning*, that aims to simplify it. Considering the tree obtained T_0 and all the trees obtained by deleting one branch of the tree, the tree is selected according to the criterion

$$PS(T) = RSS(T) - \lambda|T|$$

where $RSS(T)$ is the Residual Sum of Squares of the tree and $|T|$ the number of leafs. This criterion is similar to (1.12) and the parameter λ controls the smoothness and should be selected properly, e.g. with a CV criterion.

Note that also in this case the generalization to the multivariate case is almost straightforward as it is possible to consider to add new knots in any of the covariates.

These two methods are really effective in the multivariate framework, where they are able to simplify a high dimensional regression problem by combining some basic and widely used statistical tools. Nonetheless, in some cases where the function to be estimated is wiggly, or if the estimated function is required to be continuous or smooth, more refined methods are needed.

1.5.3 Bayesian Approaches

Several of these models have also interesting representations in the Bayesian framework. In Section 1.5.1 we presented the LMM representation of the penalized spline regression with automatic selection of the smoothing parameter and we stated that the usual LMM fitting strategies can be used also with the smoothing splines. Thus also the Bayesian tools for estimating LMM can be adopted for this purpose. Generally speaking, once some prior distribution is set for the parameters $\boldsymbol{\beta}$ and the variances, it is possible to use Gibbs sampling or Metropolis–Hastings MCMC algorithms to estimate the posterior distributions of the parameters. This kind of Bayesian tool is effective with semi-parametric spline regression or if the model is a non-parametric one with all the datapoints assumed to be knots. Instead, with other classes of non-parametric models, several problems arise that do not allow to use this approach without any effort.

It is however notable that Bayesian methods have been studied also for the adaptive approaches. We refer in particular to the Bayesian-MARS ([Denison et al. (1998)] and then extended in [Mallick et al. (1999)]) and to the Bayesian adaptive splines [Biller (2000)]. Those methods allow automatic selection of the knot positions and also of the number of knots of the spline fit. This is something hat goes beyond the capabilities of standard Metropolis–Hastings MCMC methods as the randomness in the number of knots implies randomness in the number of dimension in the parameter space. Hence more refined tools are needed, such as the Reversible Jump MCMC algorithm [Green (1995)].

In order to give a sketch of this kind of algorithm, let us briefly but formally explain what is the concern arising when considering from a Bayesian perspective non-parametric regression methods. Considering M to be the number of bases, the parameters to be estimated is a triplet $\{M, \boldsymbol{\tau}^{(M)}, \boldsymbol{\beta}^{(M)}\}$, where the superscripts stress out that the dimensions of the two vectors depend on the number of bases. Hence, given a certain M, $\{\boldsymbol{\tau}^{(M)}, \boldsymbol{\beta}^{(M)}\} \in \Theta^{(M)}$, $\Theta^{(M)}$ being a space whose number of dimensions depends again on M. Then, one can write

$$\{\boldsymbol{\tau}, \boldsymbol{\beta}\} \in \bigcup_{M \in \mathcal{M}} \Theta^{(M)},$$

where $\mathcal{M}$ is the set of all possible values for M.

RJ-MCMC is an algorithm that allows to compute estimates taking into account this issue. Hence, at each step of the chain, the algorithm makes one of the three type of moves allowed: computation of updated coefficients, movement of one knot or birth or death of one knot. The last type of move is the one that allows to switch between two different subspaces.

This approach allows to obtain posterior distributions for the number of knots and for the smoothing parameter (if considered), but this is not possible for the knot locations or for the single coefficients. On the converse, it is possible to consider the posterior distributions for the single fitted values. Hence the final estimate can be obtained as an average of all the models selected at different steps (provided the "burning" of the first M iterations).

1.6 Multivariate Regression

It is possible to generalize several of the algorithms presented previously in the multivariate context. Both the MARS procedure and the Regression Tree are designed in order to work properly and to fit models in the multivariate case considering a quite flexible structure of the model. In the multivariate case, during the knot addition steps new basis can be multiplied by any of the basis already considered in the model.

With other spline based methods several tools are available in order to deal with regression on more than one term. One possibility is to consider multivariate basis functions and to perform a regression following one of the strategies described in previous sections. As an example of multivariate splines we refer to [Curry and Schoenberg (1966)], where the simplex interpretation of B-splines is presented leading to a generalization of them in the multivariate framework.

Tensor product is a simple tool that allow the researcher to generalize splines in the multivariate context just considering the products of univariate splines. Considering the model

$$\boldsymbol{y} = f(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_d) + \boldsymbol{\epsilon} \quad \mathrm{E}(\boldsymbol{\epsilon}) = \boldsymbol{0} \quad \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2 \boldsymbol{I}_n,$$

where the unknown function f is defined on a d-dimensional domain and the observed variables are $\{(y_i, z_{i1}, \ldots, z_{id})\}$. It is possible to estimate f with a function $\hat{f}$ of the form $\hat{f}(\boldsymbol{z}) = \boldsymbol{X\beta}$, where

$$\boldsymbol{X} = \boldsymbol{X}_1 \circledast \cdots \circledast \boldsymbol{X}_d.$$

The columns of $\boldsymbol{X}$ are called tensor product splines. Note, however that in general using these splines, one generates a huge design matrix and hence estimates can be inefficient. This is however not the only issue arising with tensor product splines. Consider a researcher who has to deal with spatial data: the coordinates are arbitrarily chosen and then one should prefer invariance with respect to a rotation. In general, however tensor product splines lack of this property.

In order to get multidimensional splines invariant to rotation it is possible to consider radial bases or thin plate splines.

Given a knot $\boldsymbol{\tau}$ with coordinates $(\tau_1, \ldots, \tau_d)$, a radial basis is a function of the euclidean distance d between t and the observations. This definition is quite general and it can trivially be used in order to transform e.g. the truncated polynomial basis functions.

This tool can be used also beyond the polynomial splines framework. Effectively, probably the most widely used splines are the Thin Plate ones [Duchon (1977)]. Chosen $m > 2d$ and given the $M = \binom{m+d-1}{d}$ functions ϕ_j, defined as the linearly independent polynomials on the space $\mathbb{R}^d$ with degree less than $m+1$, a thin plate spline on the data $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ is a function g such that

$$g(\boldsymbol{z}) = \sum_{i=1}^{n} \delta_i \eta_{md} \left(\left\| \boldsymbol{z}_i - \boldsymbol{z} \right\|_2^2 \right) + \sum_{j=1}^{M} a_j \phi_j(z)$$

where

$$\eta_{md}(r) = \begin{cases} \frac{(-1)^{m+\frac{d}{2}+1}}{2^{2m-1}\pi^{\frac{d}{2}}(m-1)!(m-\frac{d}{2})!} r^{2m-d} \log(r) & d \text{ even} \\ \frac{\Gamma(\frac{d}{2}-m}{2^{2m}\pi^{\frac{d}{2}}(m-1)!} r^{2m-d} & d \text{ odd} \end{cases}$$

where a_j and δ_i are constants. Considering the $n \times 3$ matrix $\boldsymbol{T}$, with elements $T_{i,d} = \phi_j(t_d)$, if $\boldsymbol{T\delta} = 0$, g is a natural thin plate spline.

Thin plate splines arise naturally as the solution of an optimization problem. As described in [Wood (2006)], considering the problem of finding $\hat{f}$ as the continuous function that minimizes

$$\left\| y - f \right\|_2^2 + \lambda J_{md}(f), \tag{1.14}$$

where J_{md} is a penalty functional that measures the wiggliness of the function f and it is defined as

$$J_{md}(f) = \int \cdots \int_{\mathbb{R}^d} \sum_{\nu_1 + \cdots + \nu_d = m} \frac{m!}{\nu_1 \cdots \nu_d} \left(\frac{\partial^m f}{\partial {x_1}^{\nu_1} \cdots {x_d}^{\nu_d}} \right)^2 \mathrm{d}x_1 \cdots \mathrm{d}x_d.$$

Note also that for all the ϕ_j functions, it holds $J_{md}(\phi_j) = 0$, i.e. the ϕ_j can be considered as the bases that are not wiggly by means of the chosen penalization criterion. Hence the problem can be rewritten as

$$\left\| \boldsymbol{y} - \boldsymbol{H\delta} - \boldsymbol{T\alpha} \right\|_2^2 + \lambda \boldsymbol{\delta}^{\mathrm{T}} \boldsymbol{H\delta}, \tag{1.15}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ are the vectors collecting the coefficients and $\boldsymbol{H}$ is the matrix with elements $H_{ij} = \eta_{md}(\left\| \boldsymbol{z}_i - \boldsymbol{z}_j \right\|_2)$[†].

Considering that those splines are a solution of (1.14) they are a good compromise between smoothing and adaptation to the observed data. Moreover, another nice feature of them is that there is no need to select knots in advance and hence the estimate is less influenced by the choices of the researcher. Note also that the invariance property is satisfied as data enter the model only through euclidean distances.

Nonetheless, those splines involve a $n+M$ parameters and their computational cost is very high, hence they should not be used in practice exactly as they are.

This issue of thin plate splines however suggests to switch to semi-parametric regression methods becoming much more efficient in practice.

[†]Note that $\boldsymbol{H}$ is a symmetric matrix, as $H_{ij} = H_{ji} \quad \forall i, j = 1, \ldots, n$.

One first method [Wood (2003)] allows to get an approximate $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ according to a restricted form of (1.15), considering the eigen-decomposition $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$ with the diagonal matrix arranged so that the higher eigenvalues are in the upper positions. The idea is to approximate the matrix $\boldsymbol{H}$ by $\boldsymbol{U}_k\boldsymbol{D}_k\boldsymbol{U}_k^{\mathrm{T}}$ where $\boldsymbol{D}_k$ is the top-right $k \times k$ submatrix of $\boldsymbol{D}$ and $\boldsymbol{U}_k$ the $n \times k$ matrix obtained considering only the eigenvectors corresponding to the eigenvalues in $\boldsymbol{U}_k$.

Hence (1.15), can be modified as

$$\left\|\boldsymbol{y} - \boldsymbol{U}_k\boldsymbol{D}_k\boldsymbol{\delta}_k - \boldsymbol{T}\boldsymbol{\alpha}\right\|_2^2 + \lambda\boldsymbol{\delta}_k^{\mathrm{T}}\boldsymbol{D}_k\boldsymbol{\delta}_k,$$

where $\boldsymbol{U}_k\boldsymbol{\delta}_k = \boldsymbol{\delta}$. With some algebra it is also possible to include the constraint $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{\delta} = 0$ in the fitting procedure and then it is possible to get an estimate of $\boldsymbol{\delta}_k$ and $\boldsymbol{\alpha}$ and use it to compute an approximation of $\boldsymbol{\delta}$.

A second way that can be used to make thin plate splines tractable in practice is indeed quite simple and intuitive. Instead of using all the n data in the basis computation, one can replace them by k knots $\tau_1, \ldots, \tau_k$ and then estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$. Based on the knots it is possible to compute a design matrix $\boldsymbol{X}$ and estimate the coefficients via penalized least squares, subject to the constraint that $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{\delta} = 0$. The penalization matrix $\boldsymbol{D}$, however, has to reflect the fact that only the coefficients $\boldsymbol{\delta}$ are drivers of the wiggliness, while $\boldsymbol{\alpha}$ should not be penalized.

This method is even simpler than the previous one, but here the researcher is forced to choose the number and the locations of the knots. This second strategy is less elegant than the previous one, even if it produces an approximation of the thin plate spline.

1.6.1 Additive models

The models presented in the previous section require the estimation of a high number of parameters. As an example, consider tensor product splines: if we introduce h basis functions for each covariate, the number of bases of the tensor product will be h^d.

Also the curse of dimensionality should be taken into account: it is well known that in order to have a sample that is representative of a problem, as the dimensionality grows, the sample size should increase exponentially. Hence in order to get reliable estimates in high dimensional problems, it is necessary to consider a huge number of observations.

One popular way to model simultaneously the relationship of a variable upon several others is to consider an additive structure. The model that is usually assumed is then formalized as

$$\boldsymbol{y} = f(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_d) + \boldsymbol{\epsilon} = f_1(\boldsymbol{z}_1) + \cdots + f_d(\boldsymbol{z}_d) + \boldsymbol{\epsilon},$$

where $f_1, \ldots, f_d$ are smooth functions and $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n)$. This way all the dependence structures between the response variable and the covariates are modelled singularly and this prevents the arising of all the issues connected with the dimensionality.

This structure is in fact a special case and it is quite a strong assumption. If e.g. the structure is multiplicative or if there are more complicate structures underlying the problem, this model will probably not be able to fit correctly the data.

There is also a second issue arising with these models. If $f(z_1, z_2) = f_1(z_1) + f_2(z_2)$, it is trivial to state that $f(z_1, z_2) = f_1^*(z_1) + f_2^*(z_2)$, where $f_1^* := f_1 + c$ and $f_2^* := f_2 - c$ with $c \in \mathbb{R}$. Hence, if no further restriction on the functions is introduced, a problem of identifiability of the model arises.

Considering an approximation based on basis functions of all $f_1, \dots, f_d$, so that all the estimates will have the form

$$\hat{\boldsymbol{f}}_j := f_j(\boldsymbol{z}_j) = \boldsymbol{X}_j \boldsymbol{\beta}_j \quad j = 1, \dots, d,$$

where $\boldsymbol{X}_j$ is the design matrix collecting all the basis functions on the explanatory variables z_j evaluated in the observed data points. Supposing we have h_j bases for the jth covariate, one may be lead to estimate the model considering, as the design matrix, the augmented matrix collecting all the bases of the model, but this would not address the issue of identifiability.

Supposing we are using the truncated polynomial representation, all the matrices $\boldsymbol{X}_j$ will have a column of ones corresponding to the constant basis. In order to avoid the issue, it is sufficient to consider only once in the model this basis.

If the we are using another representation, such as the B-spline one, the constant term may not be considered explicitly for each term and thus some matrix algebra is required. As in [Wood (2006)], it is possible to consider the constraint $\mathbf{1}^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{\beta} = 0$, loosing one degree of freedom on the estimate of the coefficients. In order to implement this constraint, it can be considered the QR decomposition of the $h_j \times 1$ matrix $(\mathbf{1}^{\mathrm{T}} \boldsymbol{X}_j)^{\mathrm{T}}$ into the product $\boldsymbol{Q}\boldsymbol{R}$[†]. Thence if one considers the matrix $\boldsymbol{Q}^*$ as $\boldsymbol{Q}$ without the first column, it is easy to compute that $\mathbf{1}^{\mathrm{T}} \boldsymbol{X}_j \boldsymbol{Q}^* = 0$ and hence the constraint is fulfilled. Thus for each term $\boldsymbol{z}_j$ in the model, it will be necessary to consider the design matrix $\tilde{\boldsymbol{X}}_j = \boldsymbol{X}_j \boldsymbol{Q}^*$ rather than $\boldsymbol{X}_j$ and one gets one coefficient less than the original ones[‡]. Hence, considering the as design matrix $\tilde{\boldsymbol{X}} = \left[1, \tilde{\boldsymbol{X}}_1, \dots, \tilde{\boldsymbol{X}}_h\right]$ and as coefficients $\tilde{\boldsymbol{\beta}}^{\mathrm{T}} = \left[\tilde{\boldsymbol{\beta}}_1, \dots, \tilde{\boldsymbol{\beta}}_h\right]$, the model is represented in the same form of (1.1).

Usually these models are fitted through penalized least squares and all the tools we described previously can be adopted in order to produce esti-

[†]By definition and properties of the QR-decomposition, $\boldsymbol{Q}$ is a $k_j \times k_j$ orthogonal matrix, while $\boldsymbol{R}$ is a $k_j \times 1$ upper triangular matrix, i.e. a vector with only the first value non null.

[‡]Once estimated the $h_j - 1$ unconstrained coefficients of the reparametrized model, say $\hat{\tilde{\boldsymbol{\beta}}}_j$, one can get the estimate of the original ones as $\hat{\boldsymbol{\beta}}_j = \boldsymbol{Q}^* \hat{\tilde{\boldsymbol{\beta}}}_j$.

mates. The objective function can be written as

$$\left\|\boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}\right\|_2^2 + \sum_{j=1}^{h} \lambda_j \tilde{\boldsymbol{\beta}}^{\mathrm{T}} \tilde{\boldsymbol{D}}_j \tilde{\boldsymbol{\beta}},$$

where the matrices $\tilde{\boldsymbol{D}}_j = \boldsymbol{Q}^{*\mathrm{T}}_j \boldsymbol{D}_j \boldsymbol{Q}^*_j$ are transformations of the original penalty matrix as the penalties are usually chosen as a function of the parameters in the original parametrization.

As seen in previous sections also in this case care is required in order to select the penalizing parameters λ_j. As in the univariate case, those parameters can be selected via the model selection criteria or via the ML/REML criteria.

The use of multivariate bases is surely appealing as it seems a natural extension of the univariate case. However some consequences of multivariate smoothing should be pointed out. First it is important to notice that smoothing based on penalized splines, as it was presented, involves the use of a single smoothing parameter and this is somehow restrictive. We may discuss an example in order to explain why this is not in general a good choice. Suppose we are modelling the concentration of a particular substance in the water in relation to the geographical position, the temperature and the humidity. In this case, covariates are of completely different scales and there is no reason to assume a similar wiggliness of the dependence structures of the variables involved. At least because of this intuition, one should avoid to choose the same smoothing parameter.

On the other hand, the plain additive model is unable to model interactions between different terms and this may be a strong limitation in some context. In the analysis of geographical data it is in general not advisable to set an additive relationship between the effect of longitude and latitude terms, but they should be rather modelled jointly. In the case presented, the interactions for this couple of terms can be modelled making use of multivariate splines or with tensor product splines, thus just setting one design matrix for both of them.

Additive models can be further generalized in order to comprehend more general structures. They can be handled also in order to include terms that allow to model the interactions between additive terms originating the Variable Coefficients models.

1.7 GLM framework

Linear regression models can be generalized in order to allow them to deal with data coming from a wider variety of situations. In Section 1.1.1 we justified the use of least squares under the assumption that $\boldsymbol{y} \sim N(\boldsymbol{Z\beta}, \sigma^2 \boldsymbol{I}_n)$, but of course there are a lot of situations in which this assumption is inappropriate.

Consider, as an example, the actuarial context. A researcher can be interested in modelling how many claims will be reported by a policy holder in MTPL insurance. This variable is discrete and non-negative and hence modelling it as if it was Gaussian could lead to meaningless results.

Generalized Linear Models [Nelder and Weddernburn (1972)] allow the distribution of the response variable to be other than the Gaussian one. Moreover they also introduce some flexibility in the specification of the dependence structure between the response variable and the terms, yet relying in the parametric framework.

Basically in a GLM the distributional assumption on $\boldsymbol{y}$ is

$$y_i \sim \mathrm{ED}(\mu_i; \phi),$$

where ED is any distribution of the Exponential Dispersion family, i.e. its density function can be written as

$$p(y_i; \theta_i, \phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right\} \tag{1.16}$$

where $\theta_i = \theta(\mu_i)$ is the natural parameter, while a, b and c are functions that characterize the distribution. The function $b(\theta)$ is crucial as it is possible to show that $\mathrm{Var}(y) = b''(\theta)a(\phi)$. Hence it can be defined the variance function $V(\mu) = b''(\theta)$ that describes the relationship between the mean and the variance of the distribution up to a scale parameter. It is also assumed that

$$g(\mu_i) = \eta_i = Z_i\boldsymbol{\beta},$$

where g is a known function called *link function* and η_i is the linear predictor.

In general MLE can not be obtained explicitly, but they should be computed with a numerical optimization of the likelihood. In this model an efficient algorithm that is widely used is the Iterative Reweighted Least Squares, that is closely related to the method of scoring (a very clear explanation can be found in [Dobson (2001)] pag. 64–66).

We will however briefly resume it, as it will be useful in the next chapter.

i Choose starting values $\hat{\boldsymbol{\mu}}_0$, $\hat{\boldsymbol{\eta}}_0$ or $\hat{\boldsymbol{\beta}}_0$, so that $g\left(\hat{\boldsymbol{\mu}}_0\right) = \hat{\boldsymbol{\eta}}_0 = \boldsymbol{Z}\hat{\boldsymbol{\beta}}_0$ and set $j = 0$

ii Compute

$$\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j = g'(\hat{\boldsymbol{\mu}}_j)$$

iii Compute

$$\tilde{\boldsymbol{y}}_j = \hat{\boldsymbol{\eta}}_j + (\boldsymbol{y} - \hat{\boldsymbol{\mu}}_j)\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j$$

and the weights matrix $\boldsymbol{W}_j$, whose diagonal is the vector

$$\left[\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j^2 V(\boldsymbol{\mu}_j)\right]^{-1}$$

iv Compute $\hat{\boldsymbol{\beta}}_j$ solving the system $(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{W}_j\boldsymbol{Z})^{-1}\hat{\boldsymbol{\beta}}_j = \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{W}_j\tilde{\boldsymbol{y}}_j$ and set $j = j + 1$

v Repeat steps 2–4 until convergence of $\hat{\boldsymbol{\beta}}$

As seen, the formulation adopted is quite general and GLMs can extend several of the models presented in this chapter. Design matrices developed in the previous sections can be directly used in the GLM framework allowing to extend the semi-parametric regression also under different distributional assumptions.

In these models a Goodness of Fit measure is the Deviance, defined as:

$$D(\boldsymbol{\mu}) = 2\sum_{i=1}^{n}\{y_i(\vartheta(y_i) - \vartheta(\mu(z_i))) - b(\vartheta(y_i)) + b(\vartheta(\mu(z_i)))\}. \tag{1.17}$$

This can also be considered as a natural extension of the RSS as, if we consider the distribution of $\boldsymbol{y}$ to be Gaussian as in the previous sections, it turns out that

$$\mathrm{RSS}(\boldsymbol{\beta}) = D(\mu(\boldsymbol{\beta})).$$

Also penalized regression can be implemented in the context of GLMs. Consider a Generalized Additive Model, that is the generalization of the models seen in Section 1.6.1 but under the more general assumption that the distribution of the response variable belongs to the ED family. Given this assumption, the model structure is

$$g(\mathrm{E}\{y_i\}) = g(\mu_i) = \eta_i = f_1(z_{i1}) + \cdots + f_h(z_{ih}).$$

If the functions $f_1, \ldots, f_h$ are smooth functions and the researcher wants to approximate them through penalized splines, the IRLS should be modified in order to allow to include a penalized version. Considering the representation in terms of the unconstrained vector of coefficients $\tilde{\boldsymbol{\beta}}$ and assuming known the matrices $\tilde{\boldsymbol{D}}_j$ and $\tilde{\boldsymbol{X}}$ and the smoothing parameters λ_j, the model can be fitted through

1. Choose starting values $\hat{\tilde{\boldsymbol{\beta}}}_0$ for the vector of means and set $j = 0$

2. Compute $\hat{\boldsymbol{\mu}}_j = \tilde{\boldsymbol{X}}\hat{\tilde{\boldsymbol{\beta}}}_j$, $\hat{\boldsymbol{\eta}}_j = g(\hat{\boldsymbol{\mu}}_j)$,

$$\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j = g'(\hat{\boldsymbol{\mu}}_j)$$

3. Compute
$$\tilde{\boldsymbol{y}}_j = \hat{\boldsymbol{\eta}}_j + (\boldsymbol{y} - \hat{\boldsymbol{\mu}}_j)\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j + \tilde{\boldsymbol{X}}\hat{\tilde{\boldsymbol{\beta}}}_j$$
and the weights matrix $\boldsymbol{W}_j$, whose diagonal is the vector
$$\left[\left(\frac{\mathrm{d}\eta}{\mathrm{d}\mu}\right)_j V(\boldsymbol{\mu}_j)\right]^{-1}$$

4. Compute $\hat{\boldsymbol{\beta}}_{j+1}$ as
$$\hat{\boldsymbol{\beta}}_{j+1} = \arg\min_{\boldsymbol{\beta}} \left\|\sqrt{\boldsymbol{W}_j}(\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{X}}\boldsymbol{\beta})\right\|_2^2 + \sum_{l=1}^{d} \lambda_l \tilde{\boldsymbol{\beta}}^{\mathrm{T}} \tilde{\boldsymbol{D}}_l \tilde{\boldsymbol{\beta}}$$
and set $j = j + 1$

5. Repeat steps 2–4 until convergence of $\hat{\boldsymbol{\beta}}$.

1.8 Some applications in Actuarial Sciences

Claims Reserving

Flexible modelling is very appealing in actuarial sciences, where it is often necessary to study some variables whose interdependencies cannot be assumed a priori.

An early application of GAMs to an actuarial problem can be found in [Verrall (1996)]. That paper is a natural refinement of the model presented in [Renshaw and Verrall (1998)] with the implementation of more refined semi-parametric models. In [Renshaw and Verrall (1998)] the chain ladder technique is interpreted view of a GLM, where the incremental claim amounts are seen as random variables from the exponential (dispersion) family. The transformed means of those variables depend as typical in claims reserving on the accident year and on the development year.

Considering μ_{ij} $i = 1, \dots, I$ and $j = 1, \dots, J$ to be the expected value of the incremental paid amount for the claims with accident year i and development j, [Renshaw and Verrall (1998)] set the model
$$\log(\mu_{ij}) = \log(e_i) + \mu_0 + \alpha_i + \beta_j,$$
where e_i is a known exposure representing the dimension of the portfolio and some constraints are required in order to ensure identifiability. Accident and development years are then considered in the models as factors, that is a naïve and simple way to introduce flexibility with the drawback that non smooth estimates should be accepted. However this allows the actuary not be forced to assume a linear dependence structure.

In fact [Verrall (1996)] overcomes this issue yet preserving the flexibility required. Moreover Chain Ladder models are in general over-parametrized[§] and smoothing at least in a direction helps to reduce the degrees of freedom.

Hence [Verrall (1996)] assumes

$$\log(\mu_{ij}) = \log(e_i) + \mu_0 + f(i) + \beta_j,$$

where f is a smooth function. The smoothing have been introduced on the accident year effect because, when purified by the exposure, this somehow represents the claim frequency and it is unlikely to change heavily between consecutive years in a specific Line of Business.

A further extension of this model, considering smoothing also on the development year is presented in [England and Verrall (2002)].

Ratemaking

Non-parametric and semi-parametric regression techniques have found a wide field of application in non-life ratemaking. We will describe this problem more in detail in Chapter 3. An insurance company has to assess the premiums in advance and it should be able to discriminate between the policyholders on the basis of the riskiness of the single contracts. It is known that, if this is not done properly, several issues that can weaken the stability of a company may arise, such as the adverse selection of the policyholders.

The use of regression splines in order to assess the pure premiums goes back at least to [Taylor (1989)], where bivariate splines were used to model premium rates as a function of the geographical area.

Non-parametric regression tools are now widely used in the actuarial practice and still research is carried on this field. The use of Bayesian GAMs in ratemaking was introduced some years ago in [Denuit and Lang (2004)]. In particular the proposed model was applied in order to model both claim frequencies and claim amounts. More recently [Klein et alt. (2014)] proposed also the use of Generalized Additive Models for Location Shape and Scale (GAMLSS models), again in their Bayesian variant, ensuring even more flexibility for the distribution of the response variable. Also in this case models have been studied for both for frequency and severity.

Mortality tables

In life insurance one of the key aspects that allow the insurance activity to be stable is to obtain reliable estimates of the mortality on which premiums can be computed. Raw data about the death counts in a population provide

[§]Note in fact that for an $m \times m$ square run-off triangle there are $\dfrac{m * (m+1)}{2}$ observations and $2m - 1$ parameters. Hence, even if this situation allows the parameter estimates to be identified, information is too poor to consider them to be reliable.

useful informations, but they cannot usually be used directly in order to calculate premiums.

Indeed usually quantities such as the mortality rates are smoothed (graduated) in order to avoid to obtain estimates affected also by the erratic components due to the variability of the data. One simple method that is sometimes used up to now belongs to the class of the Nearest Neighbour Estimators, depicted in Section 1.4: a moving average is applied to the crude mortality rates in order to smooth them. This simple technique of local regression is often applied more than once in order to get estimates as refined as possible. In [Gavin et alt. (1993)] we find also a refinement of this technique based on kernel local regression.

In the last decades, several models have been studied in order to model mortality rates (or the force of mortality) considering their dependence upon the age of the individuals and of their cohort. The time horizon of a life policy contract is usually very long and the prudent actuary should take into account the fact that mortality rates change in time and hence the premiums should be based also on predicted rates. Recent and refined models doing this flexibly can be found e.g. in [Currie (2004)] and [Camarda 2012], where spline regression methods are applied. Then this methodology can be found also in [Currie (2016)], where it is compared with a wide set of parametric models.

Tensor product penalized splines are used in the GLM framework to model mortality rates in an age-cohort model and remarkably, for these models it has also been proposed a clever method to produce forecasting within the estimation procedure. However as one could expect considering the discussion in Section 1.3, the mortality rates obtained are dramatically affected by some choices in the specification of the objective function in the fitting procedure (see Figure 4 in [Currie (2004)]).

Chapter 2

Generalized Geometrically Designed Spline Regression

In this chapter we aim to introduce the Generalized Geometrically Designed Spline Regression method. This non-parametric regression method has been developed as a generalization of GeDS algorithm [Kaishev et al.(2016)] in order to make it applicable in a wider range of situations, as we will discuss in the following sections.

Before describing GeDS approach, we give a wide explanation of the intuition on which it relies, presenting some further properties of the B-splines. The generalization of GeDS is then presented and a detailed description of its implementation in the R package **GeDS** is given. This is integrated with a study of its performances. Then we will depict some inference results, the extension to the multivariate framework and we will discuss some possible modifications of the algorithm respectively in Sections 2.4, 2.5 and 2.7.

2.1 Some further properties of the B-splines

In order to introduce GeDS regression, we present some properties of the B-splines with the aim of justifying the GeDS method. For a detailed and extensive description of the properties of B-splines from a mathematical point of view, we refer however to [De Boor (2001)]. Computer Aided Graphical Design is probably one of fields where they have been most widely employed and hence, for an even more practical book we refer to [Farin (2001)].

B-splines were originally defined in [Curry and Schoenberg (1947)] and their relationship with a particular class of probability distributions was studied. Considering the knot sequence $\boldsymbol{t} = \{t_0, \dots, t_k\}$ and defining the function $\omega(t) = (t - t_0)(t - t_1) \cdots (t - t_k)$, the B-splines were defined as

$$M_k(x) = \sum_{\nu=0}^{k} \frac{k(x - t_\nu)_+^{k-1}}{\omega'(t_\nu)}.$$

As we can see, here the definition includes the normalization seen in (1.8) as they are interpreted as probability density functions. Usually, however Given a non decreasing set of knots $\boldsymbol{t}$, the jth B-spline can be defined as

$$N_{j,m}(x) := (t_{j+m} - t_j)[t_j, \dots, t_{j+m}](\cdot - x)_+^{m-1},$$

reflecting the usual normalization (1.7).

An alternative definition comes from the De Boor recurrence formula in order to find a stable algorithm that allow to compute them. The n order B-spline is defined as

$$N_{j,m}(x) = \frac{x - t_j}{t_{j+m-1} - t_j} N_{j,m-1}(x) + \frac{t_{j+m} - x}{t_{j+m} - t_{j+1}} N_{j+1,m-1}(x) \qquad (2.1)$$

where

$$N_{j,1}(x) = \begin{cases} 1 & t_j \leq x < t_{j+1} \\ 0 & \text{otherwise} \end{cases} .$$

As we mentioned in the previous Chapter, a spline function of order n based on the knot vector $\boldsymbol{t}$ can be defined as a linear combination of B-splines with the same order and the same knots. Hence defining as $S_{\boldsymbol{t},m}$ the space of those splines, we can state that it is a linear space of the B-splines. In addition, by Curry-Shoenberg theorem, it can be stated that B-splines represent a basis of the space of the piecewise polynomials, if it is considered the possibility to have coalescent knots.

Before going deeper in the explanation of the properties of B-splines, we give two definitions of objects that up to here we left unspecified.

A polynomial of order m is a function p_m such that

$$p_m(x) = \sum_{j=1}^{m} a_j x^{j-1}$$

and the set of all the n order polynomials is a linear space, denoted by $\Pi_{<m}$.

Given a strictly increasing sequence of points $\xi = \{\xi_i\}_{i=1}^{k+1}$, a piecewise polynomial of order m is a function f such that

$$f(x) = p_m(x) \text{ if } \xi_i < x < \xi_{i+1},\ i = 1, \dots, k$$

and the set of all these functions is the space $\Pi_{<m,\xi}$. Sometimes we will also consider a subspace imposing some smoothing conditions as $\Pi_{<m,\xi,\nu}$, where $\nu = \{\nu_i\}_{i=2}^{k}$ is a vector of positive integers specifying how many derivatives are continuous at each of the points $\xi_2, \dots, \xi_k$. $\nu_i = 0$ means that no continuity condition is stated in ξ_i.

In the remainder of the section we will present some properties of B-splines and we will match them in order to give the reader the intuition underlying GeDS regression.

2.1.1 Knot Averages property

One of the properties of the B-splines is the so called Knot Averages property. This means that

$$p(x) = \sum_j p(\xi_{j,m}) N_{j,m}(x) \quad \forall m \geq 2 \quad \text{and} \quad p \in \Pi_{<2},$$

where

$$\xi_{j,m} := \frac{t_{j+1} + \ldots \cdots + t_{j+m-1}}{m-1}$$

are the Greville sites.

We can show the proof of this property considering that, by (2.1), it holds

$$\sum_j \theta_j N_{j,m}(x) = \sum_j \left(\theta_j \frac{x - t_j}{t_{j+m-1} - t_j} + \theta_{j-1} \frac{t_{j+m-1} - x}{t_{j+m-1} - t_j} \right) N_{j,m-1}(x) \quad (2.2)$$

assuming the knot sequence $\boldsymbol{t}$ to be bi-infinite. Note that this is not a restrictive condition as a spline $f \in S_{\boldsymbol{t},m}$, with $\boldsymbol{t}$ finite and the set of coefficients denoted by $\boldsymbol{\theta}$ can also belong to that space $S_{\boldsymbol{t}^*,m}$, where $\boldsymbol{t}^*$ is an extended version of $\boldsymbol{t}$ obtained adding infinitely many arbitrary knots outside the bounds of $\boldsymbol{t}$ and associating zero coefficients to the new B-splines.

Hence, considering the sequence $\theta_j = \psi_{j,m}(\tau) := (t_{j+1} - \tau) \cdots (t_{j+1+m} - \tau)$ and $\psi_{j,1}(\tau) := 1$, we can write the coefficients in the RHS of (2.2) as

$$\begin{aligned} \psi_{j,m}(\tau) \frac{x - t_j}{t_{j+m-1} - t_j} + \psi_{j-1,m}(\tau) \frac{t_{j+m-1} - x}{t_{j+m-1} - t_j} &= \\ = \psi_{j,m-1}(\tau) \left((t_{j+m-1} - \tau) \frac{x - t_j}{t_{j+m-1} - t_j} + (t_j - \tau) \frac{t_{j+m-1} - x}{t_{j+m-1} - t_j} \right) \end{aligned}$$

Now, as

$$\frac{t_{j+m-1} - x}{t_{j+m-1} - t_j} = 1 - \frac{x - t_j}{t_{j+m-1} - t_j}$$

and as this relationship holds provided that $t_{j+1} < x < t_{j+m-1}$[*],

$$(t_{j+m-1} - \tau) \frac{x - t_j}{t_{j+m-1} - t_j} + (t_j - \tau) \frac{t_{j+m-1} - x}{t_{j+m-1} - t_j} = (x - \tau) \quad (2.3)$$

as the LHS of (2.3) is the straight line that connects the points $(t_{j+m-1} - \tau)$ and $(t_j - \tau)$.

Hence, it is possible to apply recursively (2.3) in (2.2) obtaining

$$\begin{aligned} \sum_j \psi_{j,m}(\tau) N_{j,m}(x) = (x - \tau) \sum_j \psi_{j,m-1}(\tau) N_{j,m-1}(x) &= \\ &= \cdots = \\ = (x - \tau)^{m-1} \sum_j \psi_{j,1}(\tau) N_{j,1}(x) = (x - \tau)^{m-1} \quad (2.4) \end{aligned}$$

[*]Otherwise we would have $N_{j,m-1}(x) = 0$.

as $\sum_j N_{j,m}(x) = 1\ \forall m$. This relation is also known as the Marsden's Identity.

Now, considering that (2.4) can be divided on both sides by $(m-1)!$ and differenced $i-1$ times with respect to the arbitrary τ, it follows

$$\sum_j \frac{\mathrm{d}^{i-1}}{\mathrm{d}\tau^{i-1}}\psi_{j,m}(\tau)\frac{N_{j,m}(x)}{(m-1)!} = (-1)^i\frac{(x-\tau)^{m-i}}{(m-i)!}. \tag{2.5}$$

By Taylor expansion of a polynomial $q \in \Pi_{<m}$,

$$q(x) = \sum_{i=1}^{m}\frac{(x-\tau)^{m-i}}{(m-i)!}\frac{\mathrm{d}^{m-i}}{\mathrm{d}\tau^{m-i}}q(\tau) \tag{2.6}$$

and hence substituting (2.5) into (2.6) and exchanging the sums,

$$q(x) = \sum_j \lambda_{j,m}qN_{j,m}(x)$$

where

$$\lambda_{j,m}q := \sum_{i=1}^{m}(-1^i)\frac{\frac{\mathrm{d}^{i-1}}{\mathrm{d}\tau^{i-1}}\psi_{j,m}(\tau)}{(m-1)!}\frac{\mathrm{d}^{m-i}}{\mathrm{d}\tau^{m-i}}q(\tau)$$

Now, considering $p \in \Pi_{<2}$, it can be shown that $\lambda_{j,m}p = p(\xi_j)$ as $\frac{\mathrm{d}^{n-i}}{\mathrm{d}\tau^{m-i}}p(\tau)$ vanishes if $i < m-1$, while for $i = m-1$ we have that

$$\frac{\mathrm{d}^{m-2}}{\mathrm{d}\tau^{m-2}}\psi_{j,m}(\tau) = (-1)^{m-1}(n-2)!\sum_{h=1}^{m+1}(t_{j+h}-\tau), \tag{2.7}$$

and for $i = m$ we have

$$\frac{\mathrm{d}^{m-1}}{\mathrm{d}\tau^{m-1}}\psi_{j,m}(\tau) = (-1)^m(m-1)!. \tag{2.8}$$

Hence, as τ is arbitrary, if we choose $\tau = \xi_{j,m}$, also the term 2.7 vanishes, and we have

$$\lambda_{j,m}q = q(\xi_{j,m}),$$

from which the property follows.

■

2.1.2 The Control Polygon

This is a crucial point about B-splines. Consider an m-th order spline function defined in $[a,b]$ and based on the knot vector $\boldsymbol{t} = \{t_i\}_{i=1}^{m+k}$ such that

$$a = t_1 = \cdots = t_m < \cdots < t_{m+k+1} = \cdots = t_{2m+k} = b. \tag{2.9}$$

Its B-spline representation is $Q = \sum_{j=1}^{p} \theta_j N_{j,m}$ with $p = m + k$ and its plot $\{x, Q(x)\}_{x\in[a,b]}$. As $x \in \Pi_{<2}$, we can write

$$\{x, Q(x)\}_{x\in[a,b]} = \left\{ \sum_{j=1}^{p} \xi_j N_{j,m}(x), \sum_{j=1}^{p} \theta_j N_{j,m}(x) \right\}_{x\in[a,b]} \tag{2.10}$$

and hence it is characterized by the sequence of control points $\{\xi_j, \theta_j\}$. These points can also be taken as the vertexes of a polygon that takes the name of Control Polygon C_Q.

Several properties arise from this definition. First, we can state that the spline function lies in the convex hull of its control polygon as (2.10) implies that $\forall x \in [a, b]$, $\{x, Q(x)\}$ can be expressed as a linear combination of the vertexes, and we showed in Section 1.5 the partition of unity property, i.e. $\sum_{j=1}^{p} N_{j,m}(x) = 1$ and $N_{j,m}(x) \geq 0$.

We can also be more specific, considering (1.6). Indeed if we take $x \in [t_i, t_{i+1}]$, $\sum_{j=i-m+1}^{i} N_{j,m}(x) = 1$ and this implies that all the polynomial piece between t_i and t_{i+1} is in the convex hull of $\{\xi_j, \theta_j\}_{j=i-m+1}^{i}$.

Secondly, we can see that the control polygon itself is a second order spline as

$$C_Q(x) = \left\{ \sum_{j=1}^{p} \xi_j N_{j,2}(x), \sum_{j=1}^{p} \theta_j N_{j,2}(x) \right\} = \left\{ x, \sum_{j=1}^{p} \theta_j N_{j,2}(x) \right\}.$$

Thirdly, the spline is an approximation to the control polygon, as it can be shown that

$$\|Q - C_Q\| \leq c|t|^2 \left\|_\infty \frac{\mathrm{d}^2}{\mathrm{d}x^2} Q(x) \right\|_\infty ,$$

where c is some constant. We refer, for the proof of this statement to [De Boor (2001)], page 135.

2.1.3 Schoenberg's variation diminishing spline approximation

Then it is possible to state that the spline is a "shape preserving" curve with respect to the Control Polygon as it can be seen as a variation diminishing approximation to it.

In order to explain what this means, we introduce the operator S^- defined as the number of sign changes. We use the whether we apply it it a sequence or to a function. In this second case, $S^- f := \sup S^-\{f(z_1), \dots, f(z_r)\}$ with arbitrary integer r and $z_1 < \cdots < z_r$.

We also introduce the functional V, the Schoenberg Variation Diminishing Approximation as for a given function f defined on $[a, b]$ as

$$Vf := \sum_{j=1}^{p} f(\xi_j) N_{j,m}$$

with $N_{j,m}$ arbitrary order B-splines based on the knots $\boldsymbol{t}$ and $\{\xi_j\}_{j=1}^{p}$ the Greville sites that can be obtained from $\boldsymbol{t}$.

If $f \geq 0$ also $Vf \geq 0$ as B-splines are non-negative. If $f' \geq 0$,

$$(Vf)'(x) = (m-1)\sum_{j=1}^{p+1}\left(\frac{f(\xi_j)-f(\xi_{j-1})}{t_{j+m-1}-t_j}\right)N_{j,m-1}(x) =$$

$$= (m-1)\left(\sum_{j=2}^{p}\left(\frac{f(\xi_j)-f(\xi_{j-1})}{t_{j+m-1}-t_j}\right)N_{j,m-1}(x)\right) \geq 0 \quad (2.11)$$

because of (1.9) and considering that by (2.9), $N_{1,m-1}(x) = N_{p+1,m-1}(x) = 0 \forall x \in \mathbb{R}$. Similar conclusion can be drawn considering higher order derivatives, hence it preserves some of its features of the original function such as positivity, monotonicity and convexity.

The functional V is called variational diminishing because it can be shown that

$$S^{-}(Vf - p) \leq S^{-}(f - p) \quad (2.12)$$

with an arbitrary $p \in \Pi_{<2}$. Hence the variance diminishing Schoenberg approximation is a functional that smooths the original function As the polynomial p in (2.12) is any straight line arbitrarily chosen, this property means that the approximating curve will in general be less wiggly than the original function.

2.1.4 Sketch of GeDS Regression

These properties allows us to draw some conclusions about the relationship between a spline and its control polygon. Considering the m-th order spline Q and its control polygon C_Q, the first can be interpreted as a variation diminishing transformation of the polygon. Indeed

$$VC_Q(x) = \sum_{j=1}^{p} C_Q(\xi_j)N_{j,m}(x) = \sum_{j=1}^{p}\theta_j N_{j,m}(x) = Q(x).$$

We are now able to understand the idea underlying the GeDS regression [Kaishev et al.(2016)]. The idea is that if we are able to find an estimate of a spline function via a second order spline, we can use it as a control polygon of a higher order one.

Suppose we are in a regression problem where, given a sample of observations $\{z_i, y_i\}_{i=1}^{n}$ we assume that

$$y_i = f(z_i) + \epsilon_i,$$

where $\mathrm{E}\{\epsilon_i\} = 0$ and $\mathrm{Var}\{\epsilon_i\} = \sigma^2$. Supposing that the true function f can be approximated by a spline, we may simplify the problem trying to find an estimate $\hat{f}$ belonging to a space $S_{\boldsymbol{t},m}$ where both $\boldsymbol{t}$ and m should be estimated.

Hence we can first find $\hat{f}_2 \in S_{\boldsymbol{t},2}$ via a knot insertion scheme or an adaptive fitting and then we can get a smoother fit attaching a higher order spline to $\hat{f}_2$, seen as a polygon. The new estimate will be smoother than $\hat{f}_2$ according to the variation diminishing property, but it will still preserve the shape of the previous estimate.

Note that the knot addition schemes for second order splines are particularly appealing, as adding a new knot has a trivial interpretation. Suppose we already have an estimate f^* based on k knots: this is just a polygon with k vertexes and adding a new knot in position τ_0 means to introduce a new vertex in the polygon (to which it should be assigned a coefficient).

2.2 The GeDS estimating algorithm

In this section we present a method allowing to perform non-parametric regression relying on the properties we presented in the previous section. The idea of this method was proposed in [Kaishev et al.(2016)] applying to a model such as (1.1). Here however we present directly its extension to the GLM framework, that is more general and includes as a special case the previous one. Hence we will refer to the model presented in Section 1.7.

However, this method aims to perform the non-parametric regression by assuming f to be a spline and taking advantage of its B-spline representation. It belongs to the class of the adaptive methods as the knots are not set ex-ante, but they are selected during the procedure. Moreover, also their placement and the order of the spline are selected by the method. Often in this section we will refer to the GLM framework, but it may be more appropriate to refer to the Generalized Non-linear Models ([Lane (1996)] and [Turner and Firth (2015)]) as only the B-spline coefficient enter linearly in the model, while it is not the case considering other sets of parameters.

GeDS methodology is composed by two major stages. At the first stage, $\hat{\eta}$ is expressed as a linear combination of second order (degree one) B-splines based on only two couples of coalescent knots. Then, knots are added one by one in a clever way and $\hat{\eta}$ is estimated by a piecewise linear fit. At each step of this sequential knot addition scheme, temporary estimates of the coefficients are computed with usual GLM tools. The number of iterations in this stage and hence the number of knots will be selected according to a stopping rule.

The second stage aims to compute a spline representation of $\hat{\eta}$ applying what we presented in the previous section. We will view the estimate as a control polygon and we will compute Schoenberg's Variational Diminishing approximations to it. We will however adjust them refitting the coefficients (but preserving the knots) in order to have ML estimates rather than approximations to a linear spline.

Note that after the selection of the location of the knots, the estimation procedure requires estimation of the coefficients of the B-splines, at each step of Stage A an at Stage B. The predictor η is linear in these parameters and hence sub-steps of the algorithm remain in the GLM class. This allows us to make use of GLM tools for the estimation of the coefficients.

GeDS is essentially a geometrically motivated procedure that at stage A builds nested GLM spline predictor models. The final one of which is approximated by higher order GLM fits in stage B, in order to solve the estimation problem.

2.2.1 The procedure

We can now give a detailed description of the algorithm.

Stage A. This is the knot addition scheme. Starting from a straight line fit, represented as a spline with only two couples of boundary knots and adding one knot at a time, the coefficients $\hat{\boldsymbol{\alpha}}$ are estimated via IRLS procedure to find the linear spline fit $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right) = \sum_{i=1}^{p} \hat{\alpha}_i N_{i,2}(z)$ with number of internal knots κ, number of B-splines (and parameters), $p = \kappa + 2$ and with a set of knots $\boldsymbol{\delta}_{\kappa,2} = \{\delta_1 = \delta_2 < \delta_3 < \ldots < \delta_{\kappa+2} < \delta_{\kappa+3} = \delta_{\kappa+4}\}$. κ is selected according to a stopping rule.

Step 0 . Let $n = 2$, $k = 0$, $p = m + k = 2$ with initial knot vector $\boldsymbol{\delta}_{0,2} = \{\delta_i\}_{i=1}^{4}$, such that, $a = \delta_1 = \delta_2 < \delta_3 = \delta_4 = b$. Let also $\beta \in [0, 1]$ and $q \in \mathbb{N}$ be preset tuning parameters. At the initial step of the IRLS procedure, set $l := 0$ and $\hat{\boldsymbol{\alpha}}_p^{(l)} = \left(\hat{\alpha}_1^{(l)}, \ldots, \hat{\alpha}_p^{(l)}\right)' = (\alpha_1, \ldots, \alpha_p)'$, where $\alpha_1, \ldots, \alpha_p$ are appropriate initial values and go to Step 1.

Step 1

1. Evaluate the quantity $\hat{\mu}^{(l)}(z_i) = g^{-1}\left(\hat{f}\left(\boldsymbol{\delta}_{k,2}, \hat{\boldsymbol{\alpha}}_p^{(l)}; z_i\right)\right)$, where $\hat{f}\left(\boldsymbol{\delta}_{k,2}, \hat{\boldsymbol{\alpha}}_p^{(l)}; z_i\right) = \sum_{i=1}^{p} \hat{\alpha}_i^{(l)} N_{i,2}(z_i)$, is the linear spline predictor, fitted at the l-th IRLS iteration, and then calculate the transformed responses
$$g^{(l)}(y_i) \approx \tilde{y}_i^{(l)} = g(\hat{\mu}^{(l)}(z_i)) + (y_i - \hat{\mu}^{(l)}(z_i))g'(\hat{\mu}^{(l)}(z_i)), \quad (2.13)$$
where, $i = 1, \ldots, n$ and $g'(\mu) = \frac{\partial g(\mu)}{\partial \mu}$.
2. Calculate the weights
$$w_i^{(l)} = 1/\operatorname{Var}[\tilde{y}_i^{(l)}|z_i] = 1/\left[\left(g'(\hat{\mu}^{(l)}(z_i))\right)^2 \operatorname{Var}[Y|Z_i]\right]. \quad (2.14)$$
3. Perform a weighted linear regression of $\tilde{y}_i^{(l)}$ on z_i with weights $w_i^{(l)}$, $i = 1, \ldots, n$, i.e., find $\hat{\boldsymbol{\alpha}}_p^{(l+1)} = \left(\hat{\alpha}_1^{(l+1)}, \ldots, \hat{\alpha}_p^{(l+1)}\right)'$.

4. Check if
$$\frac{\left|D(\hat{\boldsymbol{\alpha}}_p^{(l)};k,2) - D(\hat{\boldsymbol{\alpha}}_p^{(l+1)};k,2)\right|}{D(\hat{\boldsymbol{\alpha}}_p^{(l+1)};k,2)} \leq d_{irls}, \tag{2.15}$$
where the deviance $D(\hat{\boldsymbol{\alpha}}_p^{(\cdot)};k,2) := D(g^{-1}(f(\boldsymbol{\delta}_{k,2},\hat{\boldsymbol{\alpha}}_p^{(\cdot)};z))$ is computed using (1.17) and d_{irls} is an appropriate threshold level preselected in the IRLS estimation.

   If (2.15) is not satisfied then set $l := l+1$ and go to sub-step 1, otherwise calculate $w_i^{(l+1)}$, using (2.14), then calculate the weighted residuals
$$r_i = r(z_i) = w_i^{(l+1)}\left(y_i - \hat{\mu}^{(l+1)}(z_i)\right) g'\left(\hat{\mu}^{(l+1)}(z_i)\right)$$
set $\hat{\boldsymbol{\alpha}}_p^{(l+1)} = \hat{\boldsymbol{\alpha}}_p$ and go to Step 2.

Step 2 If $k > q$, check if the stopping rule conditions are fulfilled (see Section 2.2.2) and in case it is positive, pass to Stage B setting $\hat{\boldsymbol{\alpha}}_p = \hat{\boldsymbol{\alpha}}_{p-q}$ and $\boldsymbol{\delta}_{\kappa,2} = \boldsymbol{\delta}_{k-q,2}$.

Step 3 Group the consecutive residuals r_i , $i = 1,\ldots,N$ into clusters by their sign, finding the number u, $1 \leq u \leq n$ of groups and the integer values $d_j > 0$, $j = 1,\ldots,u$ denoting the number of elements of each group.

Step 4 Compute for each cluster m_j, $j = 1,\ldots,u$ the weighted average of the absolute residuals within the cluster. Compute also the cluster range $\eta_j = z_{d_j} - z_{d_{j-1}+1}$, $j = 1,\ldots,u$.

Step 5 Compute the normalize versions of m_j and η_j as $m'_j = m_j/m_{\max}$ and $\eta'_j = \eta_j/\eta_{\max}$ where $m_{\max} = \max_j m_j$ and $\eta_{\max} = \max_j \eta_j$.

Step 6 Compute weights for the clusters defined as
$$\omega_j = \beta m_j + (1-\beta)\eta_j \quad j = 1,\ldots,u. \tag{2.16}$$

Step 7 Order the weights ω_j finding the vector $\{j_h\}_{h=1}^u$ so that
$$\omega_{j_1} \geq \cdots \geq \omega_{j_u}.$$

Step 8 Among the groups that do not already contain a knot, select the sth, the one with the higher weight. Find the new knot location δ^* as a weighted average of the locations of the observations in the sth group, hence via
$$\delta^* = \frac{\sum_{i=d_{s-1}+1}^{d_{s-1}+d_s} r_i z_i w_i}{\sum_{i=d_{s-1}+1}^{d_{s-1}+d_s} r_i w_i}. \tag{2.17}$$

Step 9 Find i^*, $0 \leq i^* \leq k$ such that $\delta^* \in [\delta_{i^*+2}, \delta_{i^*+3}]$, set $l := 0$ and the initial values for the parameters,

$$\hat{\boldsymbol{\alpha}}_{p+1}^{(l)} = \left(\hat{\alpha}_1, \ldots, \hat{\alpha}_{i*+1}, \hat{f}\left(\boldsymbol{\delta}_{k,2}, \hat{\boldsymbol{\alpha}}_p; \delta^*\right), \hat{\alpha}_{i*+3}, \ldots, \hat{\alpha}_{p+1}\right)$$

then set $p = p + 1$ and $k = k + 1$ and go to Step 1.

Let us now summarize stage B of the generalized GeDS, which as in the Gaussian case consists of two steps.

Stage B1. Given the fit $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right)$, resulting from stage A with κ internal knots, for each $m = 3, \ldots, m_{\text{max}}$, calculate the knot placement $\bar{\mathbf{t}}_{\kappa-(m-2),m}$, solution of

$$\underset{\substack{\mathbf{t}_{\kappa-(n-2),n} \\ \delta_{i+2}<t_{i+m}<\delta_{i+m} \\ i=1,\ldots,(\kappa-m+2)}}{\arg\min} \left\| \hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right) - C_{\hat{f}\left(\mathbf{t}_{\kappa-(m-2),m}, \hat{\boldsymbol{\alpha}}_p; z\right)} \right\|_{\infty}. \tag{2.18}$$

The solution of this optimization gives the knots $\bar{\mathbf{t}}_{\kappa-(m-2),m}$ whose Greville sites $\boldsymbol{\xi}$ coincide with the knots $\delta_2, \ldots, \delta_{p+1}$.

As has been demonstrated in [Kaishev et al.(2016)], the approximate solution to 2.18 is achieved according to

$$\bar{t}_{i+m} = (\delta_{i+2} + \cdots + \delta_{i+m})/(m-1), \qquad i = 1, \ldots, \kappa - (m-2). \tag{2.19}$$

and this ensures that $f\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\alpha}}_p; z\right)$, the m-th order spline predictor curve, becomes nearly the VDS approximation to the fit, $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right)$, from stage A, i.e. closely follows its shape, as explained in Section 2.1. Therefore the fit, $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right)$ can be viewed as the control polygon of the predictor curve $f\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\alpha}}_p; z\right)$. Error bounds for this VDS approximation and the optimality properties of the knots can be found in [Kaishev et al.(2016)].

The spline predictor curve $f\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\alpha}}_p; z\right)$, obtained at stage B1 has the optimal knots $\bar{\mathbf{t}}_{\kappa-(m-2),m}$, which make it very close to its control polygon, $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right)$ and hence close to the data, but still lacks of some properties, as it is not ML estimate based on the data. This issue can be easily addressed preserving the optimality property of the knot locations thanks to stage B2, where its B-spline coefficients, $\hat{\boldsymbol{\alpha}}_p$ are treated as unknown parameters, denoted by $\boldsymbol{\theta}_p$, $p = \kappa + 2$, which are then estimated in a final run of the IRLS procedure as follows.

Stage B2. For each fixed $m = 3, \ldots, m_{\text{max}}$, find the MLE estimates $\hat{\boldsymbol{\theta}}_p$ of the B-spline coefficients of the spline predictor curve $f\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \boldsymbol{\theta}_p; z\right)$ from stage B1. For the purpose, set $l = 0$ and run the IRLS procedure similarly as in Step 1 of stage A, but with respect to the vector $\boldsymbol{\theta}_p$. More precisely, start by calculating $\hat{\mu}^{(l)}(z_i) = g^{-1}\left(\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p^{(l)}; z_i\right)\right)$, and the transformed responses and weights, substituting $\hat{\mu}^{(l)}(z_i)$ in (2.13) and (2.14) respectively. Then perform a weighted linear regression of $\tilde{y}_i^{(l)}$ on z_i with weights $w_i^{(l)}$, $i = 1, \dots, n$, to find $\hat{\boldsymbol{\theta}}_p^{(l+1)} = \left(\hat{\theta}_1^{(l+1)}, \dots, \hat{\theta}_p^{(l+1)}\right)'$. Similarly as in (2.15) check the inequality

$$\frac{\left|D(\hat{\boldsymbol{\theta}}_p^{(l)}; k, m) - D(\hat{\boldsymbol{\theta}}_p^{(l+1)}; k, m)\right|}{D(\hat{\boldsymbol{\theta}}_p^{(l+1)}; k, m)} \leq d_{irls},$$

and if it is not satisfied then set $l := l + 1$ and go to the next IRLS iteration, otherwise exit with a final estimate $\hat{\boldsymbol{\theta}}_p$.

Among all fits $\hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}_p; z\right)$, of order $m = 2, \dots, m_{\max}$, i.e. including the linear fit, $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}_p; z\right)$ from stage A, choose the one of order $\hat{m}$, for which the deviance computed as in (1.17) is minimal.

In this way in Stage B2, along with the number of knots and their locations, the degree of the spline predictor is also estimated. This is an important feature of the proposed GeDS estimation method which is rarely offered by other spline estimation procedures. Of course, any of the produced final fits of order $m \neq \hat{m}$ could be used, if other features were more desirable, for example if better smoothness is required.

2.2.2 Stopping Rules

Let us present and discuss three different stopping rules proposed for Stage A.

Ratio of deviances

In [Kaishev et al.(2016)] it was proposed a rule based on the comparison of the RSS of the last spline fitted at two different iterations of the algorithm. If the RSS of the new fit is close to the RSS of the old one, it means that the last q knots inserted did not improve the fit. As they were computed in a clever way according to (2.17), this means that the algorithm is not able to improve the fit setting new knots, hence the model adequately reproduces the "shape" of the underlying data in the predictor scale. For this reason the knot insertion procedure should stop.

The generalization of this rule in the GLM framework involves the computation of the deviances. Hence, computing

$$\phi_p := \frac{D(\hat{\boldsymbol{\alpha}}_p; k)}{D(\hat{\boldsymbol{\alpha}}_{p-q}; k-q)}, \tag{2.20}$$

defining $D(\hat{\boldsymbol{\alpha}}_p; k) := D(g^{-1}(f(\boldsymbol{\delta}_{k,2}, \hat{\boldsymbol{\alpha}}_p; z))$. The strategy is to stop Stage A if $\phi_p \geq \phi_{\text{exit}}$, with $\phi_{\text{exit}} \in [0, 1]$ a chosen and constant threshold. Since the $k - q$ knots of the older spline are knots also of the new one, any spline

$f(\boldsymbol{\delta}_{k-q,2}, \boldsymbol{\alpha}_{p-q}; z)$ is a special case of $f(\boldsymbol{\delta}_{k,2}, \boldsymbol{\alpha}_p; z)$. Since then the coefficients $\hat{\boldsymbol{\alpha}}_p$ are estimated via maximum likelihood (so, minimum deviance), we will always[†] have $\phi_p \in [0, 1]$.

Exponentially smoothed ratio

If at each step of stage A we store the result of (2.20), we will have a sequence $\{\phi_h, h\}_{h=q}^{k}$ that in general will show a nearly exponential decay as the number of steps grows. Unfortunately, if we are estimating a wiggly function, some of its features may require the insertion of many knots to be properly captured. Hence it may happen to observe some $\phi_h \approx 1$ while for higher h we observe a lower value. If we chose the previous rule as an exit strategy from stage A, this will cause an early stopping.

The instability of this rule can be prevented by choosing a "smooth" version of that.

Indeed, if $p \geq q + 2$ we can find least squares estimates $\hat{\gamma}_0$ and $\hat{\gamma}_1$ of the two parameters of the model

$$\phi_h := 1 - \exp\{\gamma_0 + \gamma_1 h\}, \text{ where } h = q, \ldots, k\ . \tag{2.21}$$

Based on those estimates we can compute the predicted value in $h = p$ as

$$\hat{\phi}_p = 1 - \exp\{\hat{\gamma}_0 + \hat{\gamma}_1 p\} \tag{2.22}$$

and verify if $\hat{\phi}_p \geq \phi_{\text{exit}}$.

The motivation behind selecting an exponential smoothing is based on the empirical evidence that ϕ_p in (2.20), grows like $1 - \exp\{-p\}$ as knots are added at appropriate locations (c.f Steps 3-8), i.e. when $h = 1, 2, 3, \ldots$. Applying (2.22) as an alternative to (2.20) leads to more stability with respect to the number of inserted knots. This is demonstrated in Section 2.3.1, (see tables 2.4 and 2.5), where rules (2.22) and (2.20) are compared.

Also under this rule, however, rule (2.20) is applied at the first 2 steps in the knot inclusion process of Stage A.

Likelihood ratio

There is also a third alternative stopping rule proposed for stage A. Consider the case we have the spline $f(\boldsymbol{\delta}_{k-1,2}, \boldsymbol{\alpha}_{p-1}; z)$, where $\boldsymbol{\alpha}_{p-1} = \{\alpha_i\}_{i=1}^{p-1}$ and we add a knot $\delta^* \in]\delta_i, \delta_{i+1}[$. If we consider the vector

$$\boldsymbol{\alpha}_p = (\alpha_1, \ldots, \alpha_{i-1}, \alpha^*, \alpha_i, \ldots, \alpha_{p-1}),$$

with

$$\alpha^* = \frac{\alpha_{i-1}(\delta^* - \delta_i) + \alpha_i(\delta_{i+1} - \delta^*)}{\delta_{i+1} - \delta_i}, \tag{2.23}$$

[†]Provided IRLS procedure achieves convergence.

we have $f(\boldsymbol{\delta}_{k-1,2}, \boldsymbol{\alpha}_{p-1}; z) = f(\boldsymbol{\delta}_{k,2}, \boldsymbol{\alpha}_p; z)$, where $\boldsymbol{\delta}_{k,2}$ is the vector $\boldsymbol{\delta}_{k-1,2}$ augmented with δ^*. Similar formulas lead to draw the same conclusion also if we consider more than one consecutive knots.

This allows us to consider splines obtained in consecutive iterations as nested models and hence we can justify the third rule, based on the likelihood ratio test statistic. Hence

$$[D(\hat{\boldsymbol{\alpha}}_{p-q}; k-q, 2) - D(\hat{\boldsymbol{\alpha}}_p; k, 2)] \sim \chi^2_q \tag{2.24}$$

which follows from the Wilks's theorem. Therefore, after each iteration in stage A, one may test whether the last q coefficients are significant and decide to exit stage A when they are not at significance level $1 - \phi_{\text{exit}}$. If the realized difference $\Delta D = D(\hat{\boldsymbol{\alpha}}_{p-q}; k-q, 2) - D(\hat{\boldsymbol{\alpha}}_p; k, 2)$ is such that

$$\Pr\left[\chi^2_q \geq \Delta D\right] \geq 1 - \phi_{\text{exit}}. \tag{2.25}$$

one should stop with the knot insertion procedure.

Note that on average, the number of knots selected with (2.25) decreases if ϕ_{exit} increases, in contrast to rules (2.22) and (2.20) where the higher ϕ_{exit} the more knots are added before exit.

Let us note that Wilks's theorem holds only under the assumption that the model is correctly specified, hence assuming that $f \in S_{\boldsymbol{\delta}_{k,2},2}$, but this is in general a strong assumption. In the R implementation we leave this rule as presented in this section as an approximation, while we discuss in Section 2.4 how this issue can be addressed.

For convenience, in what follows rules (2.20), (2.22) and (2.25) are referred to correspondingly as *ratio of deviances* (RD), *(exponentially) smoothed ratio (of deviances)* (SR) and *likelihood ratio (test)* (LR) and are coded as RD, SR and LR, respectively.

These three stopping rules represent different model selection criteria and they lead in general to different results. Hence, although with a proper choice of ϕ_{exit} and q some of the methods give similar results, they can select a different number of knots. In Section 2.3.1 we will test the three stopping rules on simulated data and we will make some considerations.

2.3 The GeDS package and its application.

In this section, we illustrate how the **GeDS** package can be applied to fit simulated and real data, assuming the response variable has a distribution from the Exponential Dispersion Family. We make use of real and simulated Gaussian, Gamma, Poisson and Binomial data in order to demonstrate the capabilities and numerical properties of GeDS and compare **GeDS** fits to the GAM, GSS and SPM fits produced by some other R packages that allow to perform flexible regression in the GLM framework: **mgcv**, **gss** and **SemiPar**, respectively.

We have also performed a thorough simulation study of the impact on the goodness of fit of GeDS estimates of different assumptions and choices made namely of: sample size ($n = 180, 500$); level of smoothness of the underlying function (smooth, medium smooth and wiggly functions); value of the GeDS parameter β; alternative model selection criteria (i.e. different stopping rules). We have tested GeDS on a series of simulated examples based on some functions adopted in many other studies on variable knot spline methods although revised in order to be applicable in the GLM framework. We present here the results for one of the simulated test functions, given in Table 2.1, first considered by [Schwetlickn and Schütze (1995)], and a real data example from materials science, due to [Kimber et al.(2009)].

The **GeDS** package has been implemented using `R` ([R Core Team (2015)]), version 3.2.2 (2015-08-14) on a `x86_64-w64-mingw32` platform.All functions and arguments are fully described within the package manual (see Appendix A). `NGeDS` and `GGeDS` are the main functions of the package and they implement correspondingly the case described in [Kaishev et al.(2016)], where estimates are based on the least squares and the generalized case described in this chapter, where estimates are based on Maximum Likelihood criteria and apply properly to response variables whose distribution is one of the EF (c.f. Section 1.7). Several supplementary functions have been implemented, providing some utilities and allowing the user to handle the outputs.

The synopsis of the `NGeDS` function, including the default values for the arguments is

```
NGeDS(formula, data, weights = NULL,
beta = 0.5, alpha = 0.5, min.intknots = 0,
max.intknots = 300, q = 2, Xextr = NULL,
Yextr = NULL, show.iters = FALSE, stoptype = "classical")
```

where the arguments are

- `formula` - in the form `y ~ f(x)`, which indicates that there is a functional dependence `f` between the response variable and the independent variable e.g. denoted respectively by `y` and `x`.

- `data` - typically a `data.frame` or an `environment` where the variables, `y` and `x` specified by the `formula` should be found. If not specified, the function looks for the variables in the global environment.

- `weights` - a optional vector of non-negative a priori provided weights for the observations $\{z_i, y_i\}_{i=1}^n$

- `beta` - the tuning parameter β of Step 6 in Stage A.

- `phi` - the threshold ϕ_{exit}

- `stoptype` the type of stopping rule considered, can be `RD` (as proposed [Kaishev et al.(2016)]), `SR` defined by (2.22) or `LR` as introduced in (2.25).
- `min.intknots` and `max.intknots` - optional parameters allowing the user to choose a minimum and a maximum number of internal knots which are added in the estimation process. By default `min.intknots` $= 0$ and `max.intknots` $= 500$, subject to checking that (`max.intknots` $+m + 1) \leq n$
- `q` - the tuning parameter for the stopping rule
- `show.iters` - Boolean argument specifying whether information about iterations should be printed on the screen or not
- `Xextr` - values of the boundary knots. If unspecified, the range of the independent variable os supplied.
- `Yextr` values of the boundary knots, in the second independent variable in the case of a bivariate regression. We will briefly describe it in Section 2.5.

The second main function in the **GeDS** package is the function `GGeDS` that computes GeDS estimates in the general case of the response having any distribution from the EF, based on the algorithm described in Section 2.2. The call to `GGeDS` with the default values for the arguments is

```
GGeDS(formula, data, family = gaussian(), weights = NULL,
beta = 0.5, phi = 0.5, min.intknots = 0,
max.intknots = 500, q = 2, maxit = 500,
Xextr = NULL, show.iters = FALSE, stoptype = "exponential")
```

The arguments of `GGeDS` are very similar to the ones required by the function `NGeDS`, except for the argument `family` that specifies the distribution of the response variable. Any `family-class` object from the `stats` package is admitted and it allows to choose a distribution from the exponential family.

The outputs of `NGeDS` and `GGeDS` functions are `GeDS-Class` objects, hence lists containing several slots and include the final results as well as key intermediate informations allowing to reproduce Stage A iterations. However the user cold see printed on screen only a basic output, made by the function call, the estimated number of internal knots and the mean of the squared residuals (in the case of `NGeDS`) or the deviance (for `GGeDS`) for the linear, quadratic and cubic GeDS fits.

Some generic function allow the user to easily access results from the `GeDS-class` objects. `coef`, `knots`, `deviance`, `predict` extract information correspondingly about, the spline coefficients, the knots, the deviance and the predicted values. There are also a couple of functions, `derive.GeDS` and `integrate.GeDS`, that compute the derivative of the final GeDS fit(s) at specified value(s) of x, and the integral of the final GeDS fit(s) on $[x_1, x_2]$, for specified values of x_1 and x_2. Graphical tools are instead provided by the generic functions `plot` and `lines`. While the second draws a curve corresponding to a fit on an existing (and active) plot, the first plots a comparison of one of the fitted curves with the observed data. An option of the function `plot` allows also the user to choose to produce several plots corresponding to the Stage A consecutive fits. All the functions which extract output information require specification of the order (through the argument `n`) of the final GeDS fit(s) of interest.

The functions `NGeDS` and `GGeDS` and all related supplementary functions are illustrate on simulated and real data examples in Sections 2.3.1 and 2.3.2. A standard PC (Intel core i7 CPU, 2.93 Ghz, 8GB RAM) has been used for all examples and comparisons. Complete and comprehensive description of all the **GeDS** functions is available in Appendix A.

2.3.1 Simulated examples and tests

In this section, we illustrate the features of the **GeDS** package and the properties of the GeDS fits compared to estimates produced by three alternative R packages: **SemiPar**, (c.f.[Wand (2014)]), **mgcv**, (c.f.[Wood (2006)]) and **gss**, (c.f. [Gu(2014)]), respectively. Those three R packages can be downloaded from CRAN (http://CRAN.R-project.org/package=GeDS).

There are several other R packages that allow to perform a flexible regression, but we chose these three because they guarantee smoothness of the and because they are designed to be applicable in the GLM framework.

In this section we will run the tests just in the univariate framework, hence these three packages implement a flexible regression based on penalized splines. We will see in Section 2.3.1 how the penalization is selected, but here we stress that **SemiPar** and **mgcv** implement a semi-parametric regression, where the number and position of the knots is selected ex-ante, while in **ssanova** there is a non-parametric procedure, where all the datapoints will be knots.

We start with the following simulated test example. We generate pseudo-random data considering, $\eta(z) = f_1(z)$, where

$$f_1(z) = 40\frac{z}{1 + 100z^2} + 4, \qquad z \in [-2, 2], \tag{2.26}$$

is a transformed version of a similar function used by [Kaishev et al.(2016)] to test GeDS under uniform noise (c.f. Section 4.1 therein). We have then generated four random samples, $\{z_i, y_i\}_{i=1}^n$, with Poisson, Gamma, Normally and Binomially distributed response variable, y, and uniformly distributed independent variable, z, i.e., $y_i \sim \text{Poi}(\mu_i)$, $y_i \sim \text{Gamma}(\mu_i, \varphi)$, with $\varphi = 0.2$,

$\mu_i = \exp\{\eta_i\}$, $\eta_i = f_1(z_i)$, $y_i \sim N(\mu_i, \sigma)$, with $\sigma = 0.2$, $\mu_i = \eta_i = f_1(z_i)$, $y_i \sim \text{Bin}(M, \mu_i)$, with $M = 50$, $\mu_i = \exp\{\eta_i\}/\left(1 + \exp\{\eta_i\}\right)$, $\eta_i = f_1(z_i) - 4$ and $z_i \sim U[-2, 2]$, $i = 1, \ldots n$, where the sample size is $n = 500$. Note that in all the regressions we will perform, we will assume the link function g to be known.

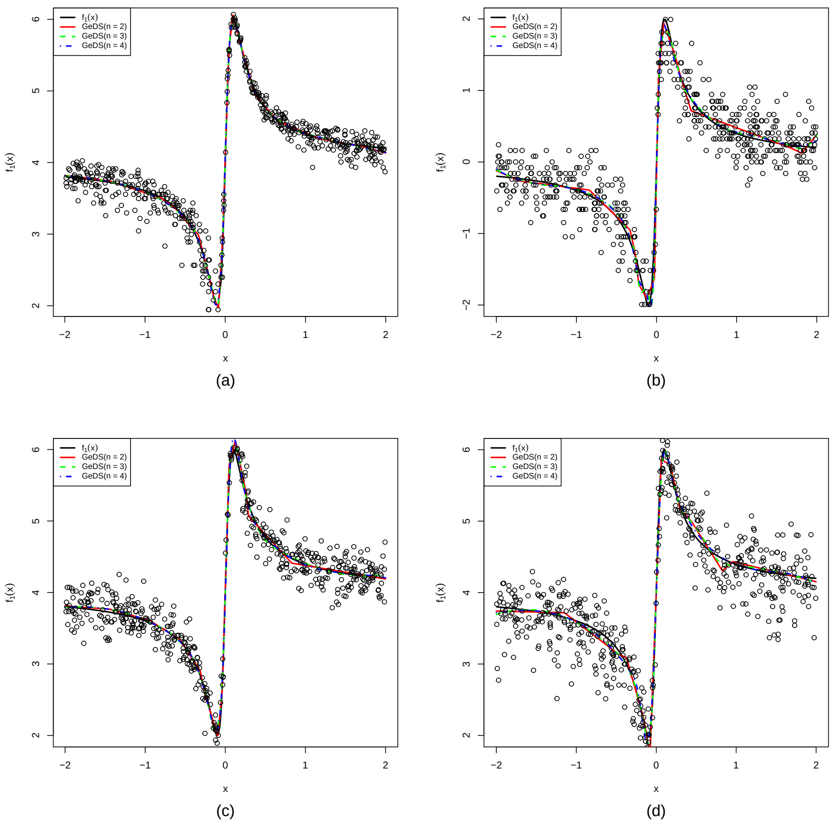


Figure 2.1: The "true" predictor function, $f_1(z)$ (black solid line) and, the linear ($m = 2$, red solid line), quadratic ($m = 3$, green dashed line) and cubic ($m = 4$, blue dotted-dashed line) GeDS output fits, $\hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}_p; z\right)$ (on the predictor scale), for the four cases of (a)-Poisson, (b)-Binomial, (c)-Normal and (d)-Gamma samples, (empty circles) generated according to Example 1.

We use the Poisson, Gamma, Normal and Binomial data sets generated as described in order to illustrate the use and performance of **GeDS**. The following R code implements this for the Gamma case. It first generates and plots the Gamma data set and the "true" predictor function $f_1(z)$ as follows

```
R> f1 <- function(x) ((10*x/(1+100*x^2))*4+4)
```

```
R> set.seed(123456)
R> newX <- sort(runif(500, min = -2, max = 2))
R> means <- exp(f1(newX))
R> newY <- rgamma(500, shape = 1/phi, scale = means*phi)
R> plot(newX, log(means), type = "n")
R> points(newX, log(newY))
R> lines(newX, log(means), lwd = 2)
```

The last three lines produce the points and the black solid line in the plot in the bottom-right panel in Figure 2.1. The line corresponds to the “true” function, while the points are a transformed version of the data. Next, in order to run GeDS regression and obtain the `GeDS-Class` object naming it `test.gamma` one writes

```
R> test.gamma <- GGeDS(formula = newY ~ f(newX), beta = 0.1,
+                       phi = 0.995, family = Gamma(log),
+                       Xextr = c(-2,2))
```

followed by

```
R> lines(test.gamma, n = 2, lwd = 2, col = "red")
R> lines(test.gamma, n = 3, lwd = 2, col = "green")
R> lines(test.gamma, n = 4, lwd = 2, col = "blue")
```

which draw correspondingly the linear ($m = 2$), quadratic, ($m = 3$) and cubic ($m = 4$) GeDS fits on the existing plot with the data and the function f_1. The following code illustrates the output from the functions `deviance`, `coef`, `knots`, `predict`, `derive.GeDS` and `integrate.GeDS` for the cubic output GedS fit, i.e, for ($m = 4$).

```
R> deviance(test.gamma, n = 4)

[1] 48.46878

R> coef(test.gamma, n = 4)

      B_1       B_2       B_3       B_4       B_5       B_6
3.7211894 3.7815624 3.7768225 3.0306911 2.6947412 0.9270338
      B_7       B_8       B_9      B_10      B_11      B_12
6.6060879 5.5783854 5.1266436 4.7815038 4.1459069 4.3974614
     B_13
4.1285988

R> knots(test.gamma, n = 4, options = "internal")

[1] -0.66268753 -0.33307385 -0.20388520 -0.05806551  0.05443521
[6]  0.17956611  0.30573948  0.52208562  0.74298065
```

```
R> predict(test.gamma, n = 4, type= "link",
+          newdata = data.frame(newX = c(-1,0,1)))

         [,1]
[1,] 3.590729
[2,] 3.978793
[3,] 4.390218

R> derive.GeDS(x = c(-1,0,1), test.gamma, n = 4, order = 1)

[1] -0.5545979 31.8329855 -0.4182105
```

which computes the derivative, $\hat{f}'\left(\bar{\mathbf{t}}_{\kappa-2,4}, \hat{\boldsymbol{\theta}}_p; \cdot\right)$ at $(-1, 0, 1)$, while the code

```
R> integrate.GeDS(x = c(-1,0,1), test.gamma, n = 4, from = -2)

[1]  3.713462  6.760717 11.698720
```

computes the integral $\int_{z_1}^{z_2} \hat{f}\left(\bar{\mathbf{t}}_{\kappa-2,4}, \hat{\boldsymbol{\theta}}_p; z\right) dz$ for $z_1 = -2$ and $z_2 = -1, 0, 1$.

The R code which implements the Poisson, Binomial and Normal cases is similar and therefore is omitted. Figure 2.1 illustrates GeDS fits in the four cases. In plots (a),(b),(c) and (d) $\hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}_p; z\right)$, of order $m = 2, 3, 4$, are plotted for the four cases of Poisson, Binomial, Normal and Gamma. The stopping rule adopted for stage A was **SR** with ϕ_{exit} set to 0.995 in all four cases. The values of β (the second tuning parameter in stage A) and the number of internal knots, $\kappa - (m - 2)$, of the final GeDS fits of order $m = 2, 3, 4$, are summarized in table 2.1.

	Poi(μ_i)	Bin(50, μ_i)	N(μ_i, 0.2)	Gamma(μ_i, 0.1)
β	0.2	0.1	0.5	0.1
$\kappa - (m-2)$	14, 13, 12	14, 13, 12	11, 10, 9	11, 10, 9
$\Vert f_1 - \hat{f}_1\Vert_1$	0.08, 0.07, 0.07	0.22, 0.16, 0.14	0.11, 0.09, 0.09	0.20, 0.16, 0.15

Table 2.1: Values of β, number of internal knots, $\kappa - (m - 2)$ and the L^1 norm $\Vert f_1 - \hat{f}_1\Vert_1$ for the linear, quadratic and cubic (i.e. for $m = 2, 3, 4$) GeDS fits, (from left to right in each column).

As can be seen from Figure 2.1, in all four cases GeDS produces linear ($m = 2$), quadratic ($m = 3$) and cubic ($m = 4$) fits, $\hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}_p; m\right)$ of remarkable quality, checked visually but also based on the L^1 norm,

$$\Vert f_1 - \hat{f}_1\Vert_1 = \int_{-2}^{2} \left| f_1(z) - \hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}_p; z\right)\right| \mathrm{d}z$$

to the "true" predictor function, $f_1(z)$ (c.f. table 2.1). One should note also looking at table 2.1 that the number of internal knots κ is estimated to

14 in the (discrete) Poisson, Binomial cases and to 11, in the (continuous) Normal, Gamma cases whereas lower β values of 0.1, 0.2 seem to produce better results when the distribution is not normal. In the normal case GeDS was run with the default value of $\beta = 0.5$, which as also noted previously by [Kaishev et al.(2016)], produced a very good fit in the uniform error case.

Comparisons with competitors

In a second test study we have used the function (2.26) in order to compare the fits produced by **GeDS** to fits obtained from some other R packages, that are designed to perform non-parametric or semi-parametric regression in the GLM framework: **gss**, **mgcv** and **SemiPar**. For the purpose, for all four distributions, i.e., Poisson, Binomial, Normal and Gamma, we have run the four competing models 1000 times, fitting them to 1000 data samples generated as described in Section 2.3.1 for varying sample size, $n = 180, 500$. As previously, we ran **GeDS** with $\phi_{exit} = 0.995$ and all other parameters as summarized in table 2.1. The "competitors" were run with the default values of their corresponding tuning parameters. In **gss** the tuning parameter is α, and it tunes the stopping rule based on the GCV functional[‡]

$$GCV(\lambda) = \frac{n^{-1}\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{A}(\lambda))^2\boldsymbol{y}}{[n^{-1}\,\mathrm{tr}(\boldsymbol{I} - \alpha\boldsymbol{A}(\lambda))]^2}, \tag{2.27}$$

as defined in [Gu(2014)], where $\boldsymbol{A}(\lambda)$ is the hat matrix. By default $\alpha = 1$ for the binomial case, while $\alpha = 1.4$ otherwise. In the function `spm` in package **SemiPar**, the tuning parameter is directly the smoothing parameter λ. By default its value is automatically selected by the underlying REML procedure. For GAM in the package **mgcv**, the tuning parameters considered have been the number of knots (hence the dimension of the basis), that is by default selected internally and the adaptive smoothing parameter (by default it is set to a constant selected as part of the GCV criterion).

	GeDS(n = 2)	GeDS(n = 3)	GeDS(n = 4)	GAM	SPM	GSS
Normal Mean	0.1588	0.1342	0.1398	0.8260	0.2131	0.2597
Normal Median	0.1554	0.1296	0.1266	0.8125	0.2100	0.2382
Poisson Mean	0.1347	0.1144	0.1159	0.9750	0.1921	0.2335
Poisson Median	0.1311	0.1058	0.1026	0.9377	0.1868	0.1944
Gamma Mean	0.2396	0.2174	0.2699	0.8652	NA	0.3352
Gamma Median	0.2239	0.1928	0.2277	0.8484	NA	0.3200
Binomial Mean	0.2512	0.2328	0.3055	0.8137	0.2869	0.7294
Binomial Median	0.2455	0.2262	0.2715	0.8002	0.2853	0.7292

Table 2.2: Averages and Medians L1 norm of the fits

The results of these comparisons in the case of $n = 500$ are presented in Figures 2.2, 2.3, 2.4 and 2.5 where in each case panel (a) presents sample

[‡]Note that this definition is somehow different than the one presented in Section 1.5.1, as a tuning parameter is introduced in the denominator.

	Normal	Poisson	Gamma	Binomial
Median	14.00	16.00	11.00	12.00
Mean	14.35	16.70	11.26	11.93
SD	3.21	3.31	2.46	3.01
Min	6.00	7.00	2.00	3.00
Max	33.00	30.00	22.00	26.00

Table 2.3: Descriptive statistics for the distribution of number of internal knots κ.

	RD1	**RD2**	**SR1**	**SR2**	**LR1**	**LR2**
$\beta = 0.1$	12.61	17.09	16.68	13.65	9.3	15.31
	(4.1)	(5.94)	(4.17)	(3.13)	(2.35)	(5.25)
$\beta = 0.2$	13.05	17.45	16.45	13.64	9.48	15.78
	(4.17)	(5.99)	(3.63)	(2.77)	(2.33)	(5.86)
$\beta = 0.5$	12.2	18.05	14.53	12.04	8.94	15.38
	(3.49)	(7.91)	(3.64)	(2.18)	(1.76)	(5.44)
$\beta = 0.7$	11.22	16.21	13.22	10.79	7.95	15.13
	(4.01)	(7.94)	(3.67)	(2.14)	(1.61)	(6.25)

Table 2.4: Average number of knots selected by GeDS and their standard deviations (in parentheses).

curves of the six competing models versus the true predictor function f_1; panel (b) - box plots of the L^1 distance to $f_1(z)$; panel (c) - histogram of the number of internal knots, κ of the linear GeDS fit. Related numerical results are summarized in Tables 2.2 and 2.3. As can be seen looking at panels (a) and (b), and at the mean and median of the L^1 distance to the true predictor summarized in Table 2.2, for all four distributions the GeDS fits (of order 2, 3 and 4) outperform those produced by the competitors. The latter tend to be overly un-smooth wiggling around the true predictor function with GAM more significantly away from f_1, especially around the origin, where the function is particularly wiggly.

This is mainly due to the global nature of the smoothing procedures via penalized regression. In this case, where the function is smooth on two intervals, while it has a wiggly jump on another, selecting just one smoothing parameter is a limitation of the model. It is apparent that `gss` captures the jump at cost of having a wiggly fit, while `gam` does not, but overall the fit is smoother.

GeDS method, instead, performs only a local regression, as it is based on B-Splines with limited positive support and there is no smoothing parameter affecting the overall procedure. On one hand this is a strength of the methodology, as it is not subject to the problem presented. On the other

	RD1	**RD2**	**SR1**	**SR2**	**LR1**	**LR2**
$\beta = 0.1$	0.16 (0.045)	0.141 (0.03)	0.143 (0.033)	0.153 (0.039)	0.188 (0.05)	0.149 (0.037)
$\beta = 0.2$	0.153 (0.042)	0.135 (0.026)	0.137 (0.029)	0.145 (0.037)	0.178 (0.048)	0.143 (0.035)
$\beta = 0.5$	0.167 (0.05)	0.148 (0.044)	0.161 (0.051)	0.167 (0.05)	0.185 (0.05)	0.157 (0.048)
$\beta = 0.7$	0.224 (0.036)	0.208 (0.093)	0.216 (0.039)	0.226 (0.034)	0.238 (0.028)	0.207 (0.045)

Table 2.5: Average L^1 distance between the true function and the GeDS fit on the linear predictor scale and the corresponding standard deviations (in parentheses).

hand an issue arises in some contexts. Suppose we are performing a Poisson regression with log link function and suppose we observe al the y-values to be zeroes on an interval. If knots are selected so that the positive support of one basis lies within this interval, the corresponding estimated coefficient for that basis will be $-\infty$ as it won't be mitigated by the smoothing parameter. This is a typical issue arising when the y are distributed according to a discrete distribution. In a way this is what data are telling, but an estimate equal to $-\infty$ may not be appealing. In order to address this issue, however a weighted version of the algorithm should be used, giving lower weight to the datapoints in that interval. Note also that this happens only on the f-scale, while on the μ-scale the inverse link should prevent this issue to be effective.

Overall for all four distributions, the quadratic GeDS performs better than the linear and cubic ones based on the corresponding mean L^1 values (c.f. Table 2.2), with the median L^1 for the cubic GeDS slightly better in the Normal and Poisson cases. The distributions of the number of knots κ in panels (c) of Figures 2.2, 2.6, 2.3, 2.4 and 2.5 are reasonably compact suggesting that the stopping rule, (2.22) consistently selects κ. This conclusion is also supported looking at the corresponding characteristics (mean, median, standard deviation and range) of the latter simulated distributions of κ, summarized in Table 2.3. It should also be noted that in all four cases, i.e., Poisson, Binomial, Normal and Gamma samples, **GeDS** produces fits with relatively small number of estimated regression coefficients, $\kappa + m$, which on average is correspondingly equal to $16 + m$, $12 + m$, $14 + m$ and $11 + m$, for the GeDS fits of order $m = 2, 3, 4$ (c.f. table 2.3).

This blinded fitting method is not fully coherent as all the packages allow the user to set some tuning parameters. Hence we compare also the performance of all six models by fitting them to 30 Poisson samples, in each case appropriately adjusting the corresponding tuning parameters described above. As can be seen from Figure 2.6 presenting the result of this comparison, GAM and GSS have significantly improved, although as before, GeDS has remained superior.

The results obtained for the case of smaller sample size $m = 180$ are similar and as in the case of $m = 500$, **GeDS** produces good quality fits with low number of knots, compactly distributed around the mean. So, based on the results for the two choices, $m = 180, 500$, GeDS is reasonably robust with respect to the sample size. Further conclusions based on the graphical illustrations are similar and the corresponding details are therefore omitted.

Time consumption

The test performed in the previous paragraph is in a way not completely fair as it is not taken into account the time consumption. GeDS is an adaptive method and it is somehow expected that it is able to produce good results, but the sequential addition of new knots requires to run several times the IRLS algorithm (Step 1 of the algorithm), each time iterating WLS regressions (Step 1.3). In the other three methods, as all the knots are set ex ante, the model is fitted only once. Hence we run a simple test on our platform in order to check which is the cost in term of time consumption of the four regression models.

m	GeDS	GAM	SPM	GSS
(100)	0.05	0.03	0.19	0.11
(500)	0.14	0.03	0.61	0.58
(1000)	0.33	0.04	0.98	1.43
(5000)	3.25	0.23	5.11	20.75
(10000)	10.37	0.43	13.17	52.26

Table 2.6: Computational times in seconds taken by the regression procedures considering different sample sizes.

We simulate five samples according to the model described, using again the test function (2.26) and Poisson distribution for the response, but varying the sample size. We thence consider $m = 100, 500, 1000, 5000$ and 10000. On each of those simulated samples we run the four regression models, The results about time taken by the four regression models are resumed in Table 2.6.

As we can see, the implementation of the GAM is very efficient and it outperforms the other three methods. We see also that the computational time of all the methods increases with the sample size, but for the GSS this happens much faster than for the others.

This can be explained if we consider the fact that GSS is a non-parametric method and all the datapoints are taken as knots. Hence the model is fitted considering $O(m)$ basis functions.

Remarkably, in this test, GeDS regression is comparable with the SPM method. This is something that was unexpected and probably it reflects the fact that the REML criterion for the selection of the smoothing parameter is computationally very expensive.

Sensitivity with respect to stopping rule and tuning parameters

In a fourth test study, we have also tested the sensitivity of GeDS regression with respect to the stopping rule and the tuning parameters $\beta \in (0,1)$, $\phi_{exit} \in (0,1)$ and $q \in \mathbb{N}$.

As widely described in Section 2.2.2, the ϕ_{exit} is the tuning parameter affecting the stopping rule, which determines when to exit from stage A. Hence this directly determines the number of knots, κ, in the knot set $\boldsymbol{\delta}_{\kappa,2}$ of the linear spline fit $\hat{f}\left(\boldsymbol{\delta}_{\kappa,2}, \hat{\boldsymbol{\alpha}}; z\right)$ and hence, the number of knots of the higher order ML spline fits $\hat{f}\left(\bar{\mathbf{t}}_{\kappa-(m-2),m}, \hat{\boldsymbol{\theta}}; z\right)$.

In the algorithm of Section 2.2, the parameter β tunes the weight put on the clusters, controlling the incidence of the range and mean of the residuals in each cluster on the cluster weight, according to Step 6 of stage A. It therefore affects the ordering of the cluster weights and hence it is a main driver in the knot placement scheme.

Generally speaking, the higher the number of knots inserted, the more degrees of freedom will have the spline and the wigglier will be the fit. Hence the behaviour of the algorithm with respect of ϕ_{exit}, while understanding the effect of β is quite more complicated.

In the case the distribution of the response is assumed to be Gaussian or Uniform, [Kaishev et al.(2016)] recommend to choose β depending on the Signal to Noise Ratio (SNR), SNR$= (\text{var}(f))^{0.5}/\sigma_\epsilon$, hence on a measure of the wiggliness of the underlying pattern, compared on the noise. However this approach is not appropriate in the broader framework of GLMs as it is not possible to separately distinguish a noise component and a signal component. Moreover mean and variance in general are not independent and observations are often significantly heteroscedastic and there is no invariance with respect to a linear transformation of the function f. Therefore, as also confirmed by our sensitivity tests, the choice of β and ϕ_{exit} in the GLM framework depends more complexly and jointly on the particular distribution (from the EF) of the data and the smoothness/wiggliness of the underlying function f. Hence it is difficult to derive universally valid rules for the selection of β, and ϕ_{exit}, although some general guidance for the range of these parameters could still be given, as illustrated in our third sensitivity test.

In order to explain the influence of q on GeDS regression, we present a brief example. Considering the RD rule with tuning parameter q and $q' = q/2$, according to 2.20 we have

$$\phi_p = \frac{D(\hat{\boldsymbol{\alpha}}_p; k)}{D(\hat{\boldsymbol{\alpha}}_{p-q}; k-q)} = \frac{D(\hat{\boldsymbol{\alpha}}_p; k)}{D(\hat{\boldsymbol{\alpha}}_{p-q'}; k-q')} \frac{D(\hat{\boldsymbol{\alpha}}_{p-q'}; k-q')}{D(\hat{\boldsymbol{\alpha}}_{p-2q'}; k-2q')}$$

and RHS is in fact the product of two ratios of deviances computed at iterations p and $p-q'$. If one chooses $\phi_{\text{exit}} = \phi_{\text{exit}}^{1/2}$, Hence we may conclude that the rules identified by $\{\phi_{\text{exit}}; q\}$ and $\{\phi'_{\text{exit}}; q'\}$ are somehow comparable if one chooses $\phi'_{\text{exit}} = \phi_{\text{exit}}^{1/2}$ (even if the first one is in general more restrictive). Nonetheless, if it happens that

$$D(\hat{\boldsymbol{\alpha}}_p; k) << D(\hat{\boldsymbol{\alpha}}_{p-q'}; k-q') \approx D(\hat{\boldsymbol{\alpha}}_{p-2q'}; k-2q'),$$

applying the second rule the algorithm would exit stage A selecting $k-2q'$, leading to early stop, while the first one would lead to continue the iterations if $D(\hat{\boldsymbol{\alpha}}_p; k)$ is small enough. More generally, the rule defined by $\{\phi_{\text{exit}}, q\}$ and $\left\{{\phi_{\text{exit}}}^{(q+h)/q}, q+h\right\}$ are comparable in the sense described here. Similar conclusions can be drawn for the other stopping rules, even if it is not trivial to state which couples of parameters lead to comparable rules.

For the purpose of the latter test we used function f_1 defined by (2.26) as the underlying predictor and generated 200 samples of 500 Poisson observations as described in Section 2.3.1. On each of these samples, we estimated f_1 using GeDS under the three alternative stopping rules, (2.20), (2.22) and (2.25), for different choices of $\beta \in (0,1)$ and $\phi_{exit} \in (0,1)$. The results of this sensitivity study are summarized in Tables 2.4 and 2.5. We recall here that rules (2.20), (2.22) and (2.25), are referred to correspondingly as *ratio of deviances*, *(exponentially) smoothed ratio (of deviances)* and *likelihood ratio (test)* and are coded as RD, SR and LR, respectively. We have run the RD rule with two sets of parameters, $\{\phi_{\text{exit}} = 0.995; q = 2\}$ and $\{\phi_{\text{exit}} = (0.995)^2; q = 4\}$ coded in tables 2.4 and 2.5 as RD1 and RD2, the SR rule with $\{\phi_{\text{exit}} = 0.995; q = 2\}$ and $\{\phi_{\text{exit}} = 0.99; q = 2\}$, coded as SR1 and SR2 and the LR rule with $\{\phi_{\text{exit}} = 0.995; q = 2\}$ and $\{\phi_{\text{exit}} = 0.5; q = 2\}$, coded as LR1 and LR2.

Table 2.4 summarizes the results giving resumes of the number of knots selected. Means and standard deviations (in parentheses) of the number of knots estimated by GeDS are reported for different stopping rules and values of the tuning parameters, ϕ_{exit}, q and β. In Table 2.5 we present the L^1 norm of the difference between the true function and the GeDS fit.

Looking at Table 2.4 and comparing column RD1 with SR2 and column RD2 with SR1, one can see that the mean number of knots are pairwise similar but the standard deviations under the SR rule are much smaller, in particular with the second pair. The estimated number of knots is much less disperse and more stable under the SR rule (2.22). The results (means and standard deviations) in column LR2 suggest that by tuning ϕ_{exit} the LR rule can generate number of knots comparable with those under the RD and SR rules (c.f. columns RD2 and SR1) but as can be seen, the corresponding standard deviations in LR2 are much higher, i.e., results are more volatile. An overall observation based on table 2.4 is that increasing the tuning parameter β from 0.1 to 0.7 does not significantly affect the estimated number of knots under the RD and LT stopping rules, whereas for the SR rule, increasing β leads on average to smaller number of knots. Our experience suggests that the role of β may be more significant for other test functions.

Analysing the results of table 2.5 the minimum L^1 distance on average is achieved with $\beta = 0.2$ all across the columns, i.e. for all three rules and choices of ϕ_{exit} and $q = 2$. We do not suggest in general to use this value, but we stress that the choice is important. The best choice varies according to the distribution chosen. Our experience is that a low value of β should be chosen when the response variable is discrete.

Best L^1 distances are achieved under RD2, SR1 and LR2 and the results in these three columns of Table 2.5 are very close. However, looking also at the results under RD2, SR1 and LR2 in Table 2.4. One can conclude that overall, the SR rule, (2.22) performs best considering both goodness of fit and stability in the selection of the number of knots, measured with the standard deviation (c.f. SR1, table 2.4). Table 2.5 however shows that the L^1 goodness of fit measure is influenced both by the number of knots selected and by the value of β.

In summary, for this particular test example, we can see that better GeDS fits are achieved with low values of $\beta = 0.1, 0.2$, values of $\phi_{\text{exit}} = 0.995$ and $q = 2$. In [Kaishev et al.(2016)] the SNR was identified as a major factor influencing the choice of β. However as mentioned previously in this section, the SNR measure is not directly applicable within the GLM context and by analogy, one can compare the variability of $y_i - \mu_i$ to the variability of μ_i. We recommend that a low value of β is chosen when the variability of $y_i - \mu_i$ is high compared to the variability of μ_i.

2.3.2 Real data examples

In what follows, we illustrate the performance of GeDS on three real data examples from materials science, coal mining and mortality modelling. The first couple of examples have been used in literature in order to study properties of estimators, the first one in [Kaishev et al.(2016)] and the second one e.g. in [Eilers and Marx (1996)] and [Biller (2000)].

Neutron diffraction data

Our first data set named `BaFe2As2` originates from a superconductivity study of `BaFe2As2` through a neutron diffraction experiment, carried out by [Kimber et al.(2009)]. The data has been used by [Kaishev et al.(2016)] to illustrate GeDS in the normal case and is now a dataset distributed together with the `GeDS` package. It includes the sample $\{z_i, y_i\}_{i=1}^{n}$ of $n = 1151$ observations of the neutron diffraction intensity, y_i viewed as a function of the angle of dispersion of the neutrons, z_i. As can be seen from Fig. 2.7, this functional dependence is highly non-linear with numerous intensity peaks occurring at certain angle values. Smoothing out the noise while at the same time adequately capturing the peaks is of utmost importance since as highlighted by [Kimber et al.(2009)], their location, height (and area) carries information about the structural (conductivity) properties of the `BaFe2As2` superconductor. For a more detailed description of the dataset and the experiment we refer to [Kaishev et al.(2016)] and [Kimber et al.(2009)].

In order to test the `NGeDS` function, we first show how to reproduce the normal (quadratic) GeDS fit obtained by [Kaishev et al.(2016)], (see Section 4.2 therein). For the purpose we load the `BaFe2As2` dataset by

```
R> data("BaFe2As2")
```

and then run `NGeDS` with the RD stopping rule (which is assumed by default in `NGeDS`) and tuning parameters used in [Kaishev et al.(2016)] by calling:

```
R> (NGeDS.model <- NGeDS(formula = Y ~ f(X), phi = 0.99, q = 3,
+                            beta = 0.6, data = BaFe2As2))

Call:
NGeDS(formula = Y ~ f(X), phi = 0.99, q = 3,
      beta = 0.6, data = BaFe2As2)

Number of internal knots of the second order spline: 227
Deviances:
Order 2    Order 3    Order 4
121602818  135815393  220441344
```

As seen, the number of knots estimated by `NGeDS` is 227 which coincides with the values obtained by [Kaishev et al.(2016)] (see section 4.2 therein). The `NGeDS` quadratic fit is of similar high quality as can be concluded comparing panel (a) of Fig 2.7 to panel (a) of Fig 6 from [Kaishev et al.(2016)].

However there is no theoretical reason to consider the response variable to be Gaussian distributed. Considering that the response is given in arbitrary units (as often in these experiments), we may consider it as a continuous and non-negative variable. Hence a more appropriate model as a further test, we fit the same `BaFe2As2` dataset with `GGeDS`, assuming the response variable, y

follows a Gamma distribution. This seems reasonable also considering that, as suggested by the plot, likely there is a dependence mean and variance, as suggested by the plot. We use the log link function as it provides more numerical stability than the canonical one. This test is implemented by the following R code

```
> (GeDS.model.gen <- GGeDS(formula = Y ~ f(X), data = BaFe2As2,
+                          beta = 0.6, phi = 0.995, q = 3,
+                          family = Gamma(log), stoptype = "RD"))

Call:
GGeDS(formula = Y ~ f(X), data = BaFe2As2, family = Gamma(log),
beta = 0.6, phi = 0.995, q = 3, stoptype = "RD")

Number of internal knots of the second order spline: 239
Deviances:
Order 2  Order 3  Order 4
0.1128   0.1262   0.1984
```

The Gamma GeDS quadratic fit with 238 knots, assuming the RD stopping rule and tuning parameters $\phi_{exit} = 0.995$, $\beta = 0.6$ and $q = 3$ is illustrated in panel (b) of Fig. 2.7, and as can be seen comparing the plotted residuals, it is of similar quality as the normal fit in panel (a). As a benchmark, in panel (c), we have fitted a quadratic Gamma spline regression with 238 uniformly spaced knots which is seen to be much worse comparing the residuals in panels (a)-(c).

We note that we have tested tuning β starting from 0.1 increasing it to 0.6 which has significantly increased the number of knots selected and improved the quality of the fit. One can therefore conclude that for very wiggly functions with sharp peaks it is natural to put more weight on the mean of the residuals, i.e. select $\beta = 0.5, 0.6$ instead of the default value of $\beta = 0.1$ which puts more weight on the residual range (c.f. 2.2).

Overall, given the sharply peaking (un-smooth) nature of the underlying dependence, in both the Normal and Gamma case GeDS regression performs very well, capturing the target of the experiment: locations and heights of the peaks.

Coal Mining Data

In our next example we fit GeDS to the coal mining disaster data from [Eilers and Marx (1996)], which is available in the **GeDS** package and can be loaded by

```
R> data("coalMining")
```

This dataset includes annual number of severe accidents in the United Kingdom coal mines for the period from 1851 to 1962. We apply the Poisson GeDS to estimate the expected number of accidents as a function of time. For the purpose we have run `GGeDS` under the SR (exponentially smoothed ratio of deviances) stopping rule assumed by default, with the tuning parameters set to $\phi_{exit} = 0.99$ and $\beta = 0.2$.

The resulting cubic spline fit with 12 knots is the red solid line in Fig. 2.8 which seems to overfit the data. By decreasing ϕ_{exit} to 0.984, we have obtained the cubic GeDS fit with four knots illustrated by the blue line in Fig. 2.8 and implemented by the following piece of code:

```
> data("coalMining")
> (GeDS.coal <- GGeDS(formula = accidents ~ f(years), phi = 0.984,
+                              family = poisson(), data = coalMining))

Call:
GGeDS(formula = accidents ~ f(years), data = coalMining,
family = poisson())

Number of internal knots of the second order spline: 6
Deviances:
Order 2  Order 3  Order 4
116.5    117.3    118.1
```

We note that the resulting GeDS fit is close to the fits produced by the competitors `gssanova` and `gam` from the **gss** and **mgcv** packages. On the converse `spm` from **SemiPar** package seems to underfit the data and deviates significantly from the rest of the competitors, (c.f Fig. 2.8).

Mortality data

In our final example, we fit GeDS to the mortality dataset `EWmortality` which includes data on mortality of males in England and Wales, from 2000 to 2002. The data, aggregated over this three year period is organized in a data frame containing the variables `Age`, `Deaths` and `Exposure`. For a given age z, age ranging from 0 up to 108, `Deaths` contains the observed number of deaths, d_z, while `Exposure` is an estimate of the mid-year population size, e_z, (central exposed to risk) at age z. Also this dataset is part of the package and can be directly loaded using the function

```
> data(EWmortality)
```

Two distributional assumptions can be made with respect to the random variable, D_z counting the number of deaths at age z. If at each age the population is assumed to be homogeneous with respect to the risk, it is appropriate to consider that D_z is Poisson distributed with parameter $\mu_z e_z$, where μ_z is the force of mortality, i.e.,

$$D_z \sim Poi(\mu_z e_z). \tag{2.28}$$

As stated in Section1.8 several parametric models can be implemented in order to obtain an estimate of μ_z, but we can model it in a non-parametric framework.

One can then use GeDS in order to estimate μ_z as a function of z, i.e. fit it to d_z, $z = 0, \ldots, 108$. Alternatively, one can assume that D_z is binomially distributed with parameters E_z and q_z, i.e.,

$$D_z \sim Bin(E_z, q_z), \tag{2.29}$$

and estimate q_z by fitting it to d_z/E_z, $z = 0, \ldots, 108$, where $E_z \approx e_z + \frac{1}{2}d_z$ is an estimate of the initial number of exposed to the risk.

Assuming the log link function, the predictor component for the model in (2.28) can be expressed as

$$\eta_z = \log(\mu_z e_z) = \log(\mu_z) + \log(e_z) = f(z) + \log(e_z), \tag{2.30}$$

where the term $\log(e_z)$ is known and it is not intended to be estimated as part of the model. Following the usual design of R functions developed to perform regressions, it is possible to incorporate a known additive term in the predictor in the model by including in the `formula` the term as an `offset` variable. The specific code is

```
R> attach(EWmortality)
R> (M1 <- GGeDS(formula = Deaths ~ f(Age) + offset(log(Exposure)),
+               family = poisson(),
+               phi = 0.99, beta = 0.1, q = 2))

Call:
GGeDS(formula = Deaths ~ f(Age) + offset(log(Exposure)),
family = poisson(), beta = 0.1, phi = 0.99, q = 2)

Number of internal knots of the second order spline: 34
Deviances:
Order 2  Order 3  Order 4
  379.3    144.7    189.8
```

In order to fit the mortality rate q_z in (2.29), we first need to compute E_z and raw mortality rates and then the regression. Hence the R code is

```
R> Exposure_init <- Exposure + 0.5 * Deaths
R> Rate <- Deaths / Exposure_init
R> (M2 <- GGeDS(formula = Rate ~ f(Age),
+               family = quasibinomial(), phi = 0.99, beta = 0.3,
+               weights = Exposure_init, q = 2))
```

```
Call:
GGeDS(formula = Rate ~ f(Age), family = quasibinomial(),
beta = 0.3, phi = 0.99, weights = Exposure_init,
q = 2)

Number of internal knots of the second order spline: 28
Deviances:
Order 2  Order 3  Order 4
  398.1    157.0    209.4
```

Here the exposures E_z should be included as sizes in the Binomial model, but, as in input we specified the rates, they enter as weights. As usual in the Binomial regression, the link function adopted is logit.

The lowest deviance is attained with the quadratic fitted splines in both models. In order to model the force of mortality μ_z, the algorithm selects 33 knots, while 27 knots are selected for q_z. Resulting fits are presented by the solid red lines in Figure 2.9, in the μ_z and $\log(\mu_z)$ scale, (in panels (a) and (c)) and in the q_z and $\text{logit}(q_z)$ scale, (in panels (b) and (d)). As can be observed, in both models a reasonable smoothing of the data is achieved across all ages. This is a real case where we can observe what we discussed in Section 2.3.1: because of the local nature of the GeDS regression, it is possible to fit a function with a varying degree of smoothness with more non-linearity at younger ages. This suggests that `GeDS` can be successfully applied in smoothing mortality data.

It is also common practice to model mortality data taking into account not only the age of the individuals, but also the calendar year, as it is well known that the mortality is not static and it is useful to study how it evolves. The regression should hence take place in the two dimensional setting with dependence of $\mu_{z,t}$ and $q_{z,t}$ on age z and time t.

2.4 Inference

In GeDS regression and, more generally, in spline regression, the researcher aims to adapt, as much as possible, a spline to a function. More precisely, the function is unknown and it cannot be directly observed, so the adaptation has to take place based on some data that are assumed to be realizations from a model that involves the function.

Let us for a while forget about this problem and let us imagine we aim to approximate a function whose realizations can be directly observed. Unless the unrealistic case that the function is itself a spline with the same order and based on the same knots, a spline can just approximate it. We can then conjecture that the distance between them -the bias- reduces by adding knots, but we cannot state that it is equal to zero.

We can thence interpret the mismatch between the spline model and the true function in two different ways yielding two different approaches that lead to different inference results.

In Section 2.4.1 we will see how we can study local asymptotic properties of GeDS regression splines, based on results of approximation theory. Then, in Section 2.4.3, we will consider the bias as a misspecification of the model and we will outline how a consistent version of the LR test can be obtained. In Section 2.4.2 we will also describe a naïve test to check whether the knots selected are all useful or they can be deleted.

2.4.1 Confidence intervals

In non-parametric statistics inference is not usually an easy task. However we can make some considerations about the local asymptotic behaviour of the spline fits under some suitable assumptions.

It is useful to recall a result from [Barrow and Smith (1978)] that will allow us to quantify the bias due to the polynomial spline approximation to the function f. Considering the integer m, the knot sequence $\boldsymbol{t} = \{t_i\}_{i=0}^{k}$ and the function $f \in C^{(m)}$. Let $h_i = t_i - t_{i-1}$ $i = 2, \dots, k$ and $h = \max_i h_i$, there exists $s_f \in S_{\boldsymbol{t},m}$, such that

$$\left\| f - b - s_f \right\|_\infty = o(h^k), \tag{2.31}$$

where

$$b(z) = \frac{f^{(m)}(z) {h_i}^m}{m!} B_m \left(\frac{z - t_i}{h_i} \right).$$

Here the function B_m is the Bernoulli polynomial, i.e.

$$B_m(z) = z^m + \binom{m}{1} B_1 z^{m-1} + \cdots + \binom{m}{m-1} B_{m-1} z + B_m \quad 0 \le z \le 1,$$

where B_i are the Bernoulli numbers. In what follows, we will refer to $\boldsymbol{\alpha}^*$ as the vector of coefficients of s_f in the B-Spline representation.

Based on this result some asymptotic properties of the polynomial regression splines have been studied in [Zhou et alt. (1998)] and [Huang (2003)], but they cannot be applied directly to the GLM framework.

It may be tempting to consider the fitted spline as the result of a least squares regression to the pseudo data $\tilde{\boldsymbol{y}}$ as defined in (2.13) and hence to apply directly the general results, but we find some problems that can be solved only by making much more assumptions than the ones in those papers.

Another interesting work in our context is [Yoshida and Naito (2014)] who studied local asymptotics for the Generalized Additive Models. Thus, even if considering penalized regression, they extended the methodology to the GLM framework. Hence we study local asymptotics following their approach.

In this subsection we will just present the sketch of the procedure that lead to develop the asymptotic confidence intervals, while for the formal proofs and computations we refer to [Yoshida and Naito (2014)]. Indeed the results presented here can be obtained by setting in the cited paper the penalizing parameters $\boldsymbol{\lambda}$ equal to zero and the number of regressors $D = 1$. In their paper, it is considered also a correction to the penalized likelihood, involving a parameter γ_n: also in this case we just need to set it equal to zero.

We need also to make the following assumptions:

(i) the covariate $\boldsymbol{z}$ is a random variable and it is distributed on $[0, 1]$ according to the distribution p_z;

(ii) $f \in \mathcal{C}^{(m)}$ and $c \in \mathcal{C}^{(3)}$;

(iii) the number of knots satisfies $k_n = o(n^{1/2})$

[Yoshida and Naito (2014)] assume also that the knots are equidistantly located, in order to simplify the calculations involving the penalization in the likelihood. For our purposes this assumption is not only unnecessary, but moreover it is incoherent with the method itself.

Assumption (iii) can be met in GeDS regression with a suitable sequence of tresholds in the stopping rule. By mimicking the proof of Lemma 2 in [Kaishev et al.(2006)], we can state that, considering a sequence of random samples $\{\boldsymbol{z_j}, \boldsymbol{y_j}\}$ from $\{z_{ij}, y_{ij}\}_{i=1}^{n_j}$, $j = 1, 2, \dots$ and $n_{j-1} < n_j$, there exists a sequence $\phi_{\text{exit},j}$, such that $k_n = o(n^{1/2})$.

Throughout this section we will consider only the case where the canonical link is used. Hence we will consider the log likelihood function

$$l(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n}\left[\log p(y_i|z_i, \boldsymbol{\alpha})\right] = \frac{1}{n}\left[\boldsymbol{y}^{\mathrm{T}} X\boldsymbol{\alpha} - \mathbf{1}^{\mathrm{T}} c(X\boldsymbol{\alpha})\right],$$

the vector of scores

$$\boldsymbol{U}(\boldsymbol{\alpha}) \equiv \boldsymbol{U}(\boldsymbol{y}; \boldsymbol{\alpha}, \boldsymbol{z}) = \frac{\partial}{\partial\boldsymbol{\alpha}} l(\boldsymbol{\alpha}) = \frac{1}{n}\left[\boldsymbol{y}^{\mathrm{T}} X - X^{\mathrm{T}} c'(X\boldsymbol{\alpha})\right]$$

and the Hessian matrix

$$\boldsymbol{H}(\boldsymbol{\alpha}) \equiv \boldsymbol{H}(\boldsymbol{y}; \boldsymbol{\alpha}, \boldsymbol{z}) = \frac{\partial^2}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\alpha}^{\mathrm{T}}} l(\boldsymbol{\alpha}) = \frac{1}{n} X^{\mathrm{T}} \operatorname{diag}\left[c''(X\boldsymbol{\alpha})\right] X.$$

Note that we presented the log likelihood and its derivatives divided by n. Even if this is not the usual way to define them, we preferred to keep the notation consistent with [Yoshida and Naito (2014)].

The sketch of the computation is as follows. First we define the vector $\boldsymbol{\alpha_f}$ as

$$\boldsymbol{\alpha_f} = \arg\min_{\boldsymbol{\alpha}} \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}\left[\log \frac{p(y_i|z_i, f)}{p(y_i|z_i, \boldsymbol{\alpha})}\middle| \boldsymbol{z}\right],$$

that is a theoretical version of the maximum likelihood estimator.

Then, denoting by $\boldsymbol{N}(z_i)$ the vector $[N_{1,m}(z_i), \dots, N_{p,m}(z_i)]^{\mathrm{T}}$, we let $f_0(x_j) = \boldsymbol{N}(z_i)^{\mathrm{T}}\boldsymbol{\alpha_f}$. Proposition 1 in [Yoshida and Naito (2014)] can be applied without any effort, yielding

$$f_0(z) - f(z) = b(z) + o(h^m).$$

This proposition is in fact proven by showing that $||f_0 - s_f||_\infty = o(h^m)$, from which the result follows immediately.

It is then possible to show that

$$\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha_f} = -\boldsymbol{H}(\boldsymbol{\alpha_f})^{-1}\boldsymbol{U}(\boldsymbol{\alpha_f}) + \boldsymbol{R}_n(\hat{\boldsymbol{\alpha}}), \tag{2.32}$$

with $\boldsymbol{R}_n(\hat{\boldsymbol{\alpha}}) = o_P\left(\left[n^{-1}h^{-1}\right]\mathbf{1}\right)$.

Considering that $\mathrm{E}\left[\boldsymbol{U}(\boldsymbol{\alpha_f})|\,\boldsymbol{z}\right] = 0$, it follows then that

$$\mathrm{E}\left[\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha_f}|\,\boldsymbol{z}\right] = \mathrm{E}\left[\boldsymbol{R}_n(\hat{\alpha})|\,\boldsymbol{z}\right] = o_P\left(n^{-1}h^{-1}\right)$$

and then

$$\mathrm{E}\left[[\hat{f} - f_0](z)\middle|\,\boldsymbol{z}\right] = o_P\left(n^{-1}h^{-1}\right)$$

as $\hat{f}(z_i) = \boldsymbol{N}(z_i)^{\mathrm{T}}\hat{\boldsymbol{\alpha}}$.

Hence it follows that

$$\mathrm{E}\left[[\hat{f} - f](z)\middle|\,\boldsymbol{z}\right] = b(z) + o_P(h^m) + o_P\left(n^{-1}h^{-1}\right)$$

Considering then the variance of $\hat{f}(z)$, we have

$$\mathrm{Var}\left[\hat{f}(z)\middle|\,\boldsymbol{z}\right] = \boldsymbol{N}(z)^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{\alpha_f})^{-1}\,\mathrm{Var}\left[\boldsymbol{U}(\boldsymbol{\alpha_f})|\,\boldsymbol{z}\right]\boldsymbol{H}(\boldsymbol{\alpha_f})^{-1}\boldsymbol{N}(z)(1 + o_P(1)).$$

In the proof of (2.32) it is shown that $|[f - f_o](z)| = o(1)$, hence one can write

$$\begin{aligned}\mathrm{Var}\left[\boldsymbol{U}(\boldsymbol{\alpha_f})|\,\boldsymbol{z}\right] = \frac{1}{n^2}\boldsymbol{X}\,\mathrm{Var}[\boldsymbol{y}|\boldsymbol{z}]\boldsymbol{X} = \frac{1}{n^2}\boldsymbol{X}\,\mathrm{diag}(c''(f(z)))\boldsymbol{X} = \\ \frac{1}{n^2}\boldsymbol{X}\,\mathrm{diag}(c''(f_0(z))(1 + o(1)))\boldsymbol{X} = \frac{1}{n}G(1 + o(1)),\end{aligned} \tag{2.33}$$

where

$$G = \int_{[0,1]} c''(f(z))B_{-p+i}(z)B_{-p+j}(z)\mathrm{d}p_z.$$

Hence we get

$$\mathrm{Var}\left[\hat{f}(z)\middle|\,\boldsymbol{z}\right] = \boldsymbol{N}(z)^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{\alpha_f})^{-1}G\boldsymbol{H}(\boldsymbol{\alpha_f})^{-1}\boldsymbol{N}(z)(1 + o_P(1)) = O_P(k_n n^{-1}).$$

If we consider the Mean Squared Error, hence, we have that

$$\mathrm{MSE}(\hat{f}(z)|\boldsymbol{z}) = O_P(h^{-1}n^{-1}) + O_P(h^{-2}n^{-2} + h^{2m}) = O_P(n^{-1/2}).$$

With these results it is possible to state that, if there exists $\delta \geq 2$ such that $\mathrm{E}\left[|y_i - c'(f(z_i))|^{2+\delta} \middle| \boldsymbol{z} \right] < \infty$, it is possible to show that

$$\sqrt{\frac{n}{k_n}} \left[\hat{f}(z) - f(z) - \mathrm{Bias}(z) \right] \xrightarrow{d} N(0, \Psi(z)),$$

with

$$\Psi(z) = \lim_{n \to \infty} \frac{1}{k_n} \boldsymbol{N}(z)^{\mathrm{T}} \boldsymbol{H}(\boldsymbol{\alpha_f})^{-1} G \boldsymbol{H}(\boldsymbol{\alpha_f})^{-1} \boldsymbol{N}(z)$$

and $\mathrm{Bias}(z) = b(z)$.

Therefore approximate local confidence intervals can be computed by estimating also the quantities $\mathrm{Bias}(z)$ and $\Psi(z)$. It may be tempting to compute those quantities directly based on the estimated spline $\hat{f}$ and its coefficients $\hat{\boldsymbol{\alpha}}$, but, as suggested in [Yoshida and Naito (2014)], it is preferable to consider a higher order spline for this purpose as the computation of Bias involves an estimate of the $m-$th order derivative of f.

2.4.2 Single knot

GeDS regression is based on a knot insertion scheme, where new knots are added sequentially. It may be interesting to check after the fit has been carried out, whether the parameters and the knots selected are significant or not. Knots selected at early steps of the knot insertion scheme may have become obsolete in sequent iterations and it is useful to check whether they could be deleted. The leading idea is somehow similar to the pruning procedures we introduced to when we were describing MARS procedure in Section 1.5.2.

If we consider the knots selected as fixed and hence the columns of the design matrix $\boldsymbol{X}$ as regressors, we can apply classical inference procedures on the regression coefficients. We should also assume that the model is correctly specified, hence that f is a second order polynomial spline. Of course those assumptions are at least simplistic, and the $p-$values shall not be trusted, but the results can be interpreted qualitatively.

As we stated in Section 2.2.2, for each internal knot, the simple formula (2.23) allows to compute the coefficient that makes the knot non influential. Hence one can perform a Wald test on each coefficient, testing the null hypothesis that it should be equal to the coefficient obtained through (2.23). Under the simplistic assumptions, the Wald test statistic is asymptotically distributed as a Gaussian random variable.

However a *caveat* may be useful. The fact that a coefficient is found to be significant with this test does not imply that the corresponding knot is useful in the regression. One of the issues that may affect spline regression is overparametrization. In GeDS regression there is no penalty or smoothing parameter to address directly this purpose, while the researcher has to choose properly the tuning parameters.

This test is inappropriate for this purpose as, when an overparametrization issue arises, the fitted curve is very likely to be "wiggly" as if too many knots are added, the spline tends to interpolate the data. Hence the fitted coefficients may be very far from the null ones and they may result very significant.

We recommend visual inspection of the results in order to check for overparametrization, and in order to address this issue, we suggest to properly tune the parameters or to use a weighted version of the algorithm.

This procedure is implemented in the function `summary.GeDS`. However, given the caveat and the assumptions we made, we prefer to leave it as an un-exported function of the **GeDS** package.

2.4.3 Likelihood Ratio

In the above sections we have seen that regression splines are in general biased estimators and that the bias arises from the fact that a polynomial spline, once set knots and order, is in the best case just an approximation of the function f.

This can be interpreted as a model misspecification and hence one could wonder if it is still correct to state that the likelihood ratio statistic is approximately chi-squared distributed. This is crucial as the LR stopping rule of stage A of the algorithm relies on this assumption.

Here we use the result of [Kent (1982)] who studied the behaviour of the likelihood ratio statistic in the general case of a misspecified model.

Suppose we are running a GeDS regression having set a suitable q. Suppose the algorithm has already selected k internal knots and we want to decide according to the LR rule whether the knot insertion scheme should continue or not. Hence our parameter is the vector of coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{k+2}$ and it can be rearranged and partitioned as $\boldsymbol{\alpha} = \{\boldsymbol{\psi}, \boldsymbol{\lambda}\}$, where $\boldsymbol{\psi}$ is the $q-$vector corresponding to the last q selected parameters and $\boldsymbol{\lambda}$ is the $(k+2-q)-$vector of the other coefficients. The algorithm have already computed the unconstrained ML estimate $\hat{\boldsymbol{\alpha}} = \{\hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\lambda}}\}$ and, in order to check if the LR stopping rule is met, we need to test the null hypothesis that $\boldsymbol{\alpha_f} = \boldsymbol{\alpha_0} = \{\boldsymbol{\psi_0}, \hat{\boldsymbol{\lambda}}_\mathbf{0}\}$, where $\boldsymbol{\psi_0}$ is computed according to (2.23), while $\hat{\boldsymbol{\lambda}}_\mathbf{0}$ is the constrained ML estimate, equal in our case to $\hat{\boldsymbol{\lambda}}$.

We also define

$$\boldsymbol{J}(\boldsymbol{\alpha}) = n^2 \int \boldsymbol{U}(\boldsymbol{y}; \boldsymbol{\alpha}, \boldsymbol{z}) \boldsymbol{U}(\boldsymbol{y}; \boldsymbol{\alpha}, \boldsymbol{z})^{\mathrm{T}} p(\boldsymbol{y}|\boldsymbol{z}, f) \mathrm{d}\boldsymbol{y}$$

and

$$\boldsymbol{K}(\boldsymbol{\alpha}) = -n \int \boldsymbol{H}(\boldsymbol{y}; \boldsymbol{\alpha}, \boldsymbol{z}) p(\boldsymbol{y}|\boldsymbol{z}, f) \mathrm{d}\boldsymbol{y}$$

and, in order to simplify the notation we write $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{\alpha_0})$ and $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{\alpha_0})$.

Hence both matrices can be partitioned as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{\psi\psi} & \boldsymbol{J}_{\psi\lambda} \\ \boldsymbol{J}_{\lambda\psi} & \boldsymbol{J}_{\lambda\lambda} \end{bmatrix} \quad \text{and} \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{\psi\psi} & \boldsymbol{K}_{\psi\lambda} \\ \boldsymbol{K}_{\lambda\psi} & \boldsymbol{K}_{\lambda\lambda} \end{bmatrix},$$

and we define $\boldsymbol{K}_{\psi\psi,\lambda} = \boldsymbol{K}_{\psi\psi} - \boldsymbol{K}_{\psi\lambda}\boldsymbol{K}_{\lambda\lambda}{}^{-1}\boldsymbol{K}_{\lambda\psi}$.

As it is well known from classical statistical theory, if $p(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{\alpha}) = p(\boldsymbol{y}|\boldsymbol{z},f)$, we would have $\boldsymbol{J}(\boldsymbol{\alpha}) = \boldsymbol{K}(\boldsymbol{\alpha}) = \boldsymbol{I}(\boldsymbol{\alpha})$ where $\boldsymbol{I}(\boldsymbol{\alpha})$ is the Fisher information matrix, but in general, in the framework of misspecified models, those equalities do not hold.

Hence Theorem 3.1 of [Kent (1982)] states that under the null hypothesis,

$$W \sim \sum_{i=1}^{q} e_i V_i, \tag{2.34}$$

where $V_i \sim \chi_1^2$ and e_i are the eigenvalues of the matrix

$$\boldsymbol{K}_{\psi\psi,\lambda} \left(\boldsymbol{K}^{-1}\boldsymbol{J}\boldsymbol{K}^{-1}\right)_{\psi\psi}.$$

In practice, as the density $p(\boldsymbol{y}|\boldsymbol{z},f)$ is unknown, the matrices $\boldsymbol{J}$ and $\boldsymbol{K}$ have to be estimated by $\hat{\boldsymbol{J}}$ and $\hat{\boldsymbol{K}}$ defined as

$$\hat{\boldsymbol{J}} = n^2 \boldsymbol{U}(\boldsymbol{y};\hat{\boldsymbol{\alpha}},\boldsymbol{z})\boldsymbol{U}(\boldsymbol{y};\hat{\boldsymbol{\alpha}},\boldsymbol{z})^{\mathrm{T}}$$

and

$$\hat{\boldsymbol{K}} = -n\boldsymbol{H}(\boldsymbol{y};\hat{\boldsymbol{\alpha}},\boldsymbol{z}).$$

By (2.34), the test statistic W should be compared with a non homogeneous Chi-Squared distribution. This in practice is feasible thanks to an algorithm developed by [Davies (1980)], that allows to compute the survival function of a non-homogeneous Chi-Squared distributed random variable. [Duchesne and Lafaye De Micheaux (2010)] provided then a very useful implementation of that algorithm in an R package named **CompQuadForm**.

However an other practical issue arises. In order to apply this methodology we would need to compute the first and second derivatives of the log-likelihood function, but unfortunately `family` objects in R do not contain such derivatives. There are at least three ways we could follow to overcome this issue, as we could write the code:

- implementing tools that allow to compute these derivatives, symbolically or numerically;
- limiting the distribution families and the link functions supported by the code and supplying derivatives for those cases;
- allowing the user to supply such derivatives.

Probably the best way to proceed is to implement the first procedure, but allowing the user to specify optionally the derivatives. This is not straightforward and, at least for the first version of the package, we implement the approximate LR rule as exposed in Section 2.2.2.

2.5 Multivariate extension

GeDS methodology can successfully be extended to the multivariate case. We give a sketch of how the algorithm can be modified in order to apply in two dimensions, omitting the straightforward generalization to higher dimensional settings.

Both stage A and stage B are designed to work in the univariate framework. Step 3 of stage A can be generalized to the bivariate setting by considering the residuals to be clustered in convex polygons according to their sign[§]. Then in step 4 one should substitute the cluster range with the areas of the regions and other slight modifications allow to generalize stage A. However with stage B the generalization should be studied more carefully. In our knowledge there is no generalization of the properties exposed in Section 2.1 allowing to develop a similar procedure in the multivariate framework.

This leads us to consider a generalization relying on tensor product splines, allowing both to choose the knot locations and to compute the knots for the higher degree splines in the univariate framework.

Let us set the problem properly, allowing us to describe GeDS methodology in the bivariate setting. We suppose we observed triples $\{z_i, x_i, y_i\}_{i \in I}$, $I = 1, \ldots, n$. We also assume z_i are observations from a random variable whose distribution belongs to the exponential dispersion family (1.16) with $g(\mu_i) = \eta_i = f(x_i, y_i)$, where f is a smooth function. Our goal is to estimate f with a bivariate spline function $\hat{f}$ defined on the rectangular region $R = [a^{(x)}, b^{(x)}] \times [a^{(y)}, b^{(y)}]$. Later on we will refer to the simple Gaussian framework, the only one implemented in the R package **GeDS** up to now.

Stage A of the algorithm in the bivariate case can be resumed as

i Set $k^{(x)} = k^{(y)} = 0$ the initial knot vector on x as $\boldsymbol{\delta}_{0,2}^{(x)} = \{\delta_i^{(x)}\}_{i=1}^4$ and $\boldsymbol{\delta}_{0,2}^{(y)} = \{\delta_i^{(y)}\}_{i=1}^4$ such that $a^{(\cdot)} = \delta_1^{(\cdot)} = \delta_2^{(\cdot)} \leq \delta_3^{(\cdot)} = \delta_4^{(\cdot)} = b^{(\cdot)}$. Let $\beta \in [0,1]$, $q \in \mathbb{N}$ and $J \in \mathbb{N}$ be the tuning parameters;

ii Compute $\theta = h^{-1}(b^{(x)} - a^{(x)})$. Slice the rectangle R into J rectangles $R_j^{(x)} = [a^{(x)} + (j-1)\theta, a^{(x)} + j\theta] \times [a^{(y)}, b^{(y)}]$, $j = 1, \ldots, h$. Consider the sets $I_j^{(x)} = \{i \in I | (x_i, y_i) \in R_j^{(x)}\}$ and permute them so that $\forall i_1, i_2 \in I_j^{(x)}$, $y_{i_1} \leq y_{i_2}$ if $i_1 \leq i_2$.

iii Compute $\theta = h^{-1}(b^{(y)} - a^{(y)})$. Slice the rectangle R into J rectangles $R_j^{(y)} = [a^{(x)}, b^{(x)}] \times [a^{(y)} + (j-1)\theta, a^{(y)} + j\theta]$, $j = 1, \ldots, h$. Consider the sets $I_j^{(y)} = \{i \in I | (x_i, y_i) \in R_j^{(y)}\}$ and permute them so that $\forall i_1, i_2 \in I_j^{(y)}$, $x_{i_1} \leq x_{i_2}$ if $i_1 \leq i_2$.

[§]We refer e.g. to [Akima (1978)] and [Akima (1996)] for algorithms that allow to perform this generalization of step 3 with some slight modifications.

iv Obtain an estimate of f via tensor product spline regression based on the knots $\boldsymbol{\delta}^{(x)}_{k^{(x)},2}$ and $\boldsymbol{\delta}^{(y)}_{k^{(y)},2}$. Store the some kind of residuals r_i and weights w_i.

v If the conditions required by a stopping rule are fulfilled, exit Stage A of the algorithm, otherwise move to step *vi*.

vi *a* Set $j = 1$.

b Consider triples $\{(w_i, r_i, x_i)\}_{i \in I^{(y)}_j}$ and place a knot $\delta^{(x)}_j$ according to Steps 3–8 of the original algorithm. Store also the weight ω_j defined in (2.16).

c If $j = J$ move to sub-step *d*, otherwise set $j := j + 1$ and go back to sub-step *b*.

d Compute $j^* = \arg\min_j \omega_j$. Then set $\delta^{(x)} := \delta^{(x)}_{j^*}$ and $\omega^{(x)} := \omega_{j^*}$.

vii Compute $\delta^{(y)}$ and $\omega^{(y)}$ following the sub-steps described in Step *vi*, but considering the sets $I^{(x)}_j$, $j = 1, \ldots, J$.

viii Choose the knot $\delta^{(\cdot)}$ corresponding to the higher $\omega^{(\cdot)}$ and and add it to the knot set of the corresponding variable. Set also $k^{(\cdot)} := k^{(\cdot)} + 1$, then move to Step *iv*.

Once exited Stage A and obtained the sets of knots $\boldsymbol{\delta}^{(x)}_{\kappa,2}$ and $\boldsymbol{\delta}^{(y)}_{\kappa,2}$, Stage B1 can be applied to both sets in order to get the new knot sets $\bar{\mathbf{t}}^{(x)}_{\kappa-(m-2),m}$ and $\bar{\mathbf{t}}^{(y)}_{\kappa-(m-2),m}$, $m = 3, \ldots m_{\max}$. Then Stage B2 is substituted by a tensor product regression, allowing to obtain a smooth surface $\hat{f}$ as an estimate of f.

Let us see an example in order to see how this regression has been implemented in **GeDS** package. We assume that the predictor is the function

$$\eta_{x,y} := f(x, y) = \sin(2x)\sin(2y), \quad (x, y) \in [0, 3]^2, \tag{2.35}$$

and we simulate 400 triples $\{z_i, x_i, y_i\}^n_{i=1}$ according to the model $z_i = f(x_i, y_i) + \epsilon_i$, where $\epsilon_i \sim N(0, 0.1)$, $x_i \sim U(0, 3)$ and $y_i \sim U(0, 3)$. Thus we obtain the data set in which (x_i, y_i) are uniformly and randomly scattered within $[0, 3]^2$.

As mentioned, we implemented Bivariate GeDS methodology only for the Gaussian case, thus we implemented it directly in the function `NGeDS`. We can generate the data and apply the regression running the following R code

```
> set.seed(123)
> doublesin <- function(x){
```

```
+   sin(2 * x[,1]) * sin(2 * x[,2])
+ }
> x <- (round(runif(400, min = 0, max = 3) ,2))
> y <- (round(runif(400, min = 0, max = 3) ,2))
> z <- doublesin(cbind(x, y))
> z <- z + rnorm(400, mean = 0, sd = 0.1)
> (BivGeDS <- NGeDS(z ~ f(x, y) , phi = 0.9, beta = 0.3,
+                   Xextr = c(0, 3), Yextr = c(0, 3)))

Call:
NGeDS(formula = z ~ f(x, y), beta = 0.3, phi = 0.9, Xextr =
c(0, 3), Yextr = c(0, 3))

Number of internal knots of the second order spline in the X
direction: 4
Number of internal knots of the second order spline in the Y
direction: 5

Deviances:
Order 2  Order 3  Order 4
4.334    3.722    3.673
```

The resulting GeDS fit is illustrated in panel (b) of Figure 2.10. Comparing it with the true function f plotted in panel (a), we can see that **GeDS** has reproduced it remarkably well using 400 observations. In panel (c) we draw a simple diagnostic, the contour plot of $r(x,y) := \hat{f}(x,y) - f(x,y)$. We can see that the fitted surface crosses the true one several times, while that this is not related with

2.6 Further test examples and comparisons

In order to provide further insight into the GeDS numerical performance and how it compares with the GSS, SPM and GAM models, we have used four additional test functions with varying degree of smoothness; smooth functions f_2 and f_3, less smooth, f_5 and highly oscillating f_4. These functions have been used also by other studies on normal GLM regression (see e.g. [Kaishev et al.(2016)] and references therein). The test functions and corresponding predictors are summarized in table 2.7.

For each of the entries in the last two columns of table 2.7, we generated random samples, $\{z_i, y_i\}_{i=1}^{n}$, with correspondingly Poisson and Gamma, distributed response variable, y, and uniformly distributed independent variable, z, i.e., $y_i \sim \text{Poisson}(\mu_i)$, $y_i \sim \text{Gamma}(\mu_i, \varphi)$, with $\varphi = 0.2$, $\mu(z_i) = \exp\{\eta(z_i)\}$, $\eta(z_i) = \eta_j(z_i)$, $j = 2, 3, 4, 5$ and $z_i \sim U[0,1]$, $i = 1, \ldots n$, for small and medium sample size, $n = 180$ and $n = 500$. In order to ensure

Test function	$y \sim \text{Poisson}(\mu)$	$y \sim \text{Gamma}(\mu, \varphi)$
$f_2(z) = 4z - 2 + 2\exp\left[-16\,(4z-2)^2\right]$	$\eta_2(z) = 5 + f_2(z)$	$\eta_2(z) = 4 + f_2(z)$
$f_3(z) = 4\sin(8z-4) + 2\exp\left[-16\,(4z-2)^2\right]$	$\eta_3(z) = f_3(z)/4 + 5$	$\eta_3(z) = f_3(z)/2 + 2$
$f_4(z) = \sqrt{z(1-z)\sin\left(2\pi(1+0.05)/(z+0.05)\right)}$	$\eta_4(z) = f_4(z) + 4$	$\eta_4(z) = 2\,(f_4(z) + 1)$
$f_5(z) = 4\sin(4\pi z) - \text{sgn}(z-0.3) - \text{sgn}(0.72-z)$	$\eta_5(z) = f_5(z) + 5$	$\eta_5(z) = f_5(z) + 3$

Table 2.7: Additional test functions and predictors

consistency with the normal GeDS from [Kaishev et al.(2016)], for f_5 we also generated a normal sample of size $n = 2048$.

2.7 Some alternatives for the estimating algorithm

2.7.1 Local Scoring procedure

In [Hastie and Tibshirani (1990)], page 141, we find an algorithm that may be considered to build an alternative generalization of the GeDS procedure [Kaishev et al.(2016)] in the GLM framework.

This algorithm is very similar to the IRLS algorithm of Section 1.7. In principle η can be expressed by any function if we replace step *iv* with a procedure that allows to fit an updated version of $\hat{\eta}$.

In [Hastie and Tibshirani (1990)] this algorithm is presented to fit GAMs, where the predictor η is just an additive model. Replacing step *iv* with an estimating procedure for an additive model, Local Scoring Algorithm can be used to fit a GAM.

We could do the same with GeDS algorithm. GeDS for the linear case could be used to produce estimate of η in step *iv* and we would get directly a generalization to the GLM framework.

However we did not include this alternative in the package as we found it to be less efficient than the one presented in Section 2.2. The knot placement procedure takes much more time than the IRLS, hence we preferred an algorithm that performs many times the IRLS, while knots are placed just once. Moreover we found the alternative algorithm to be less stable and convergence is not guaranteed.

2.7.2 MARS-GeDS

Stage A of the algorithm may sometimes be too unstable. If parameters are not tuned properly, the algorithm may stop too early and the resulting fit may be oversmoothed or it may go too far adding too many knots, returning a wiggly estimate.

The second issue in particular is in general mitigated when passing to higher order splines, but the first one requires the user to set different parameters.

However, as we discussed in Section 2.1.4, one of the keys of GeDS regression is the possibility to pass to a higher order spline starting from the previous one. In principle, when one has a first order spline curve, with Stage B of the algorithm, one can pass to a higher order spline fitting the data.

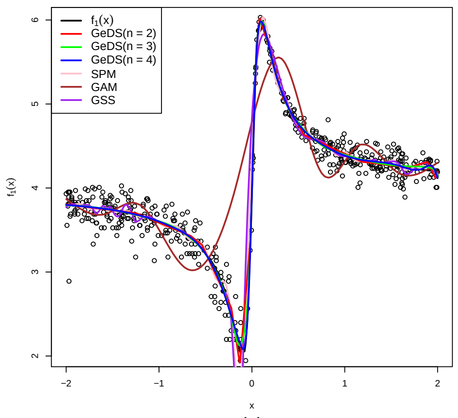
Hence an alternative is to consider a well studied way to fit a broken line to the data and to use it as an input for stage B. A natural candidate of that is the MARS procedure, presented in Section 1.5.2.

2.7.3 Variable dispersion GeDS regression

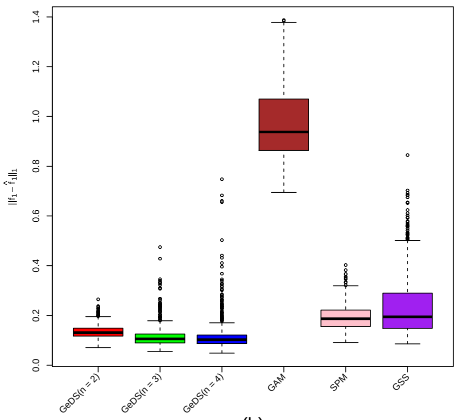
GeDS regression, as it has been presented, considers a constant overdispersion parameter all over the range of the data. This may be a strong assumption in some practical situations and it would include bias in the estimates. Some of the R packages that perform regressions allow the user to specify a model for the dispersion parameter, such as the package **hglm** ([Ronnegard et al. (2010)]).

In principle this is possible also in GeDS regression. In Stage A of the algorithm, whenever a new knot is added, the fitting procedure in Step 1 should be replaced with a procedure that allows the estimation of a dispersion part of the model. Then, also in Step 4 the residuals should be weighted in order to incorporate the estimated overdispersion, while also Stage B2 should be modified as Step 1.

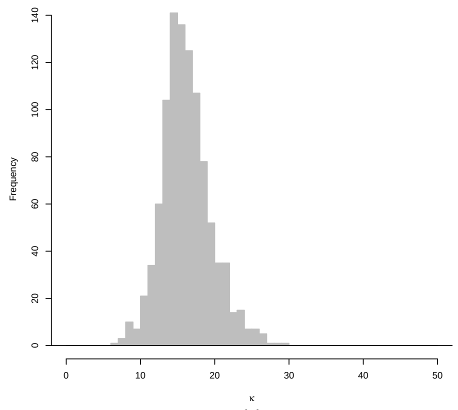
This may however sensibly increase the time consumption, hence we avoid to include this in the first version of the package.



(a)



(b)



(c)

Figure 2.2: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Poisson samples.

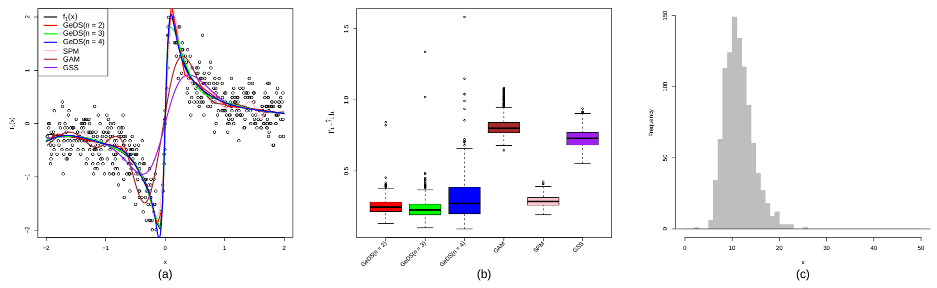


Figure 2.3: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Binomial samples.

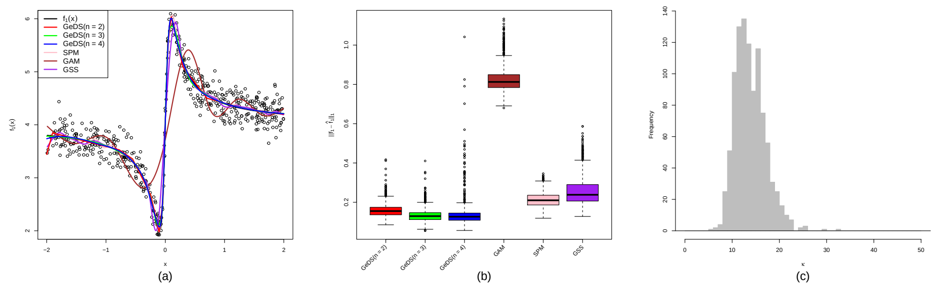


Figure 2.4: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Normal samples.

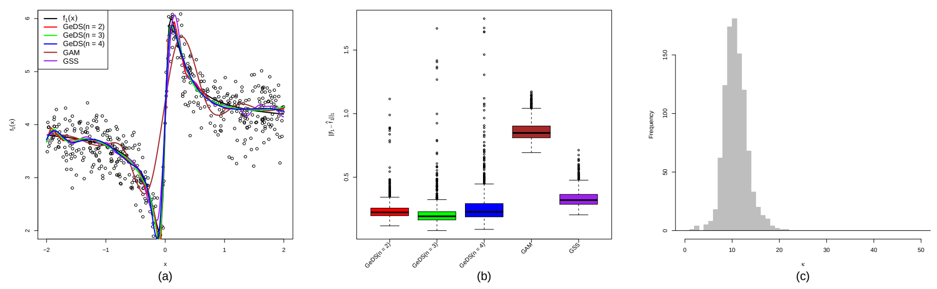


Figure 2.5: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Gamma samples.

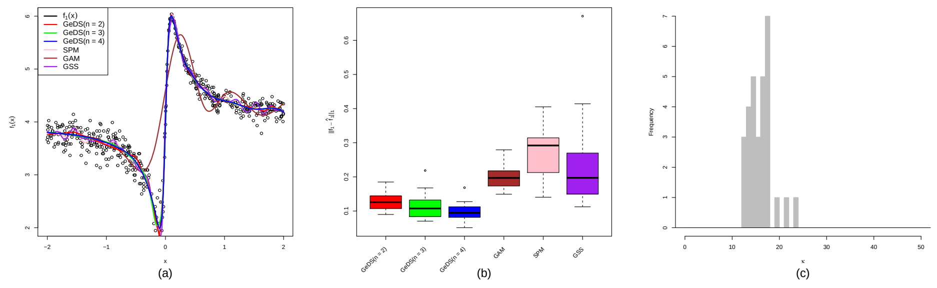


Figure 2.6: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 30 Poisson samples, adjusting the corresponding tuning parameters.

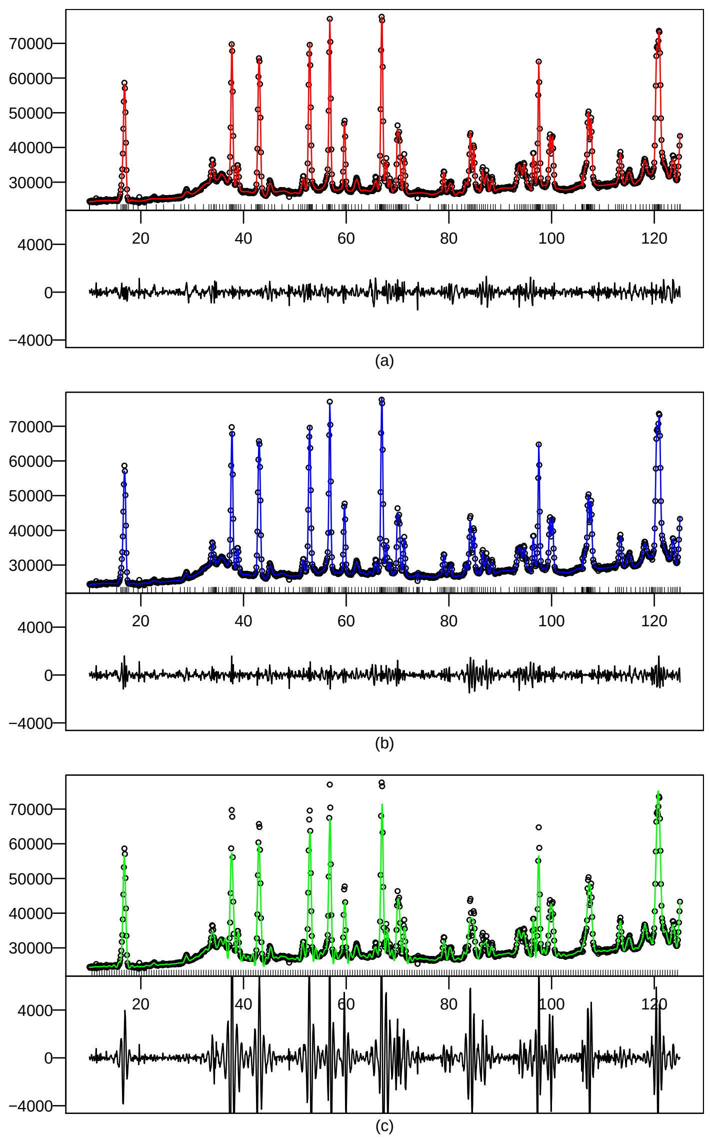


Figure 2.7: Estimated spline fits to $BaFe_2As_2$ data and their corresponding residuals: (a) Normal quadratic GeDS fit with 227 knots; (b) Gamma quadratic GeDS fit with 238 knots; (c) Gamma quadratic spline fit with 238 equispaced knots.

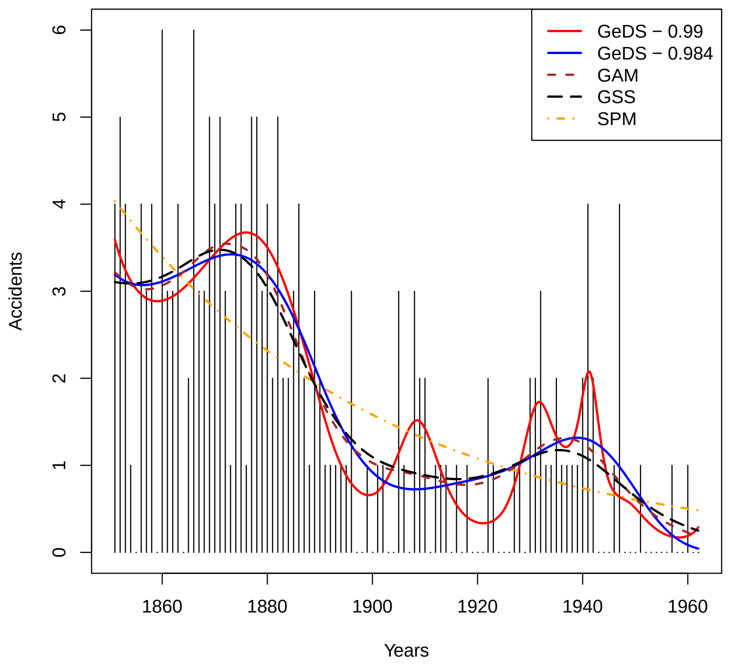


Figure 2.8: Comparison of the cubic GeDS fits, red ($\phi_{exit} = 0.99$) and blue ($\phi_{exit} = 0.99$) lines, with the competitors GAM, (brown dashed line), GSS (black dashed line) and SPM (orange dotted and dashed line).

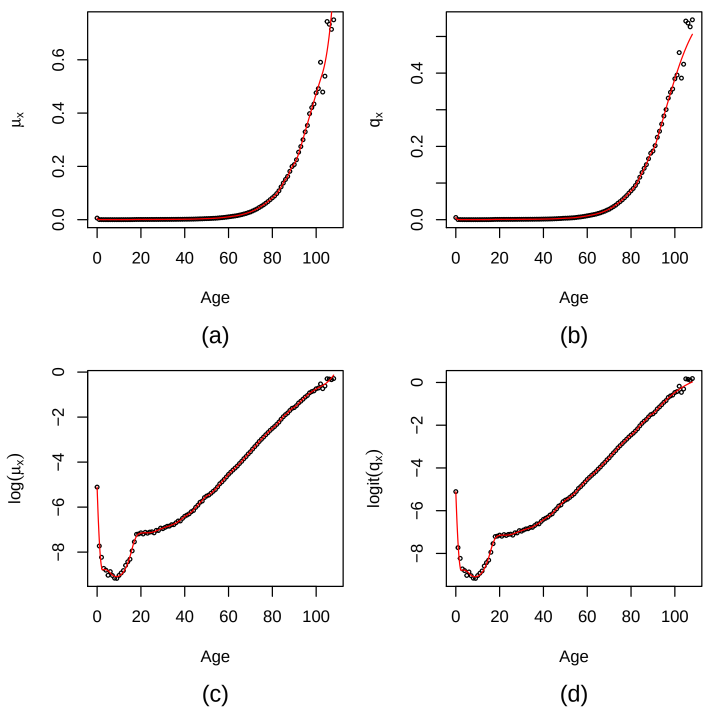


Figure 2.9: Quadratic GeDS spline fits of μ_z in the μ_z and $\log(\mu_z)$ scale, (panels (a) and (c)) and of q_z in the q_z and $\text{logit}(q_z)$ scale, (panels (b) and (d)). Black circles, in (a) and (b) represent the maximum likelihood estimates for each age, while on (c) and (d) they are transformed according to the link functions.

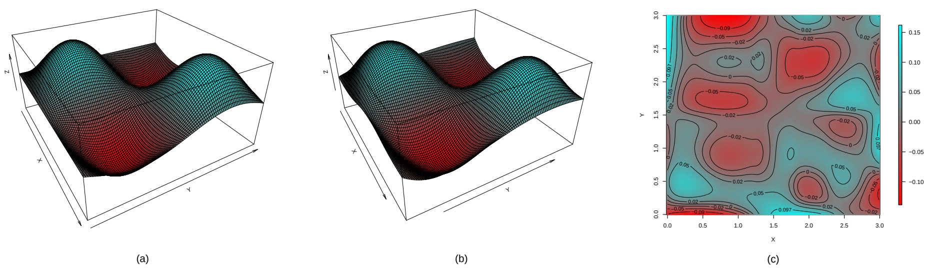


Figure 2.10: Panel (a) - the "true" function $f(x, y)$ from (2.35); panel (b) - the bi-quadratic GeDS spline fit to the 400 data points obtained by adding Gaussian noise to $f(x, y)$; panel (c) - contour plot of the difference between the fitted surface and the true function.

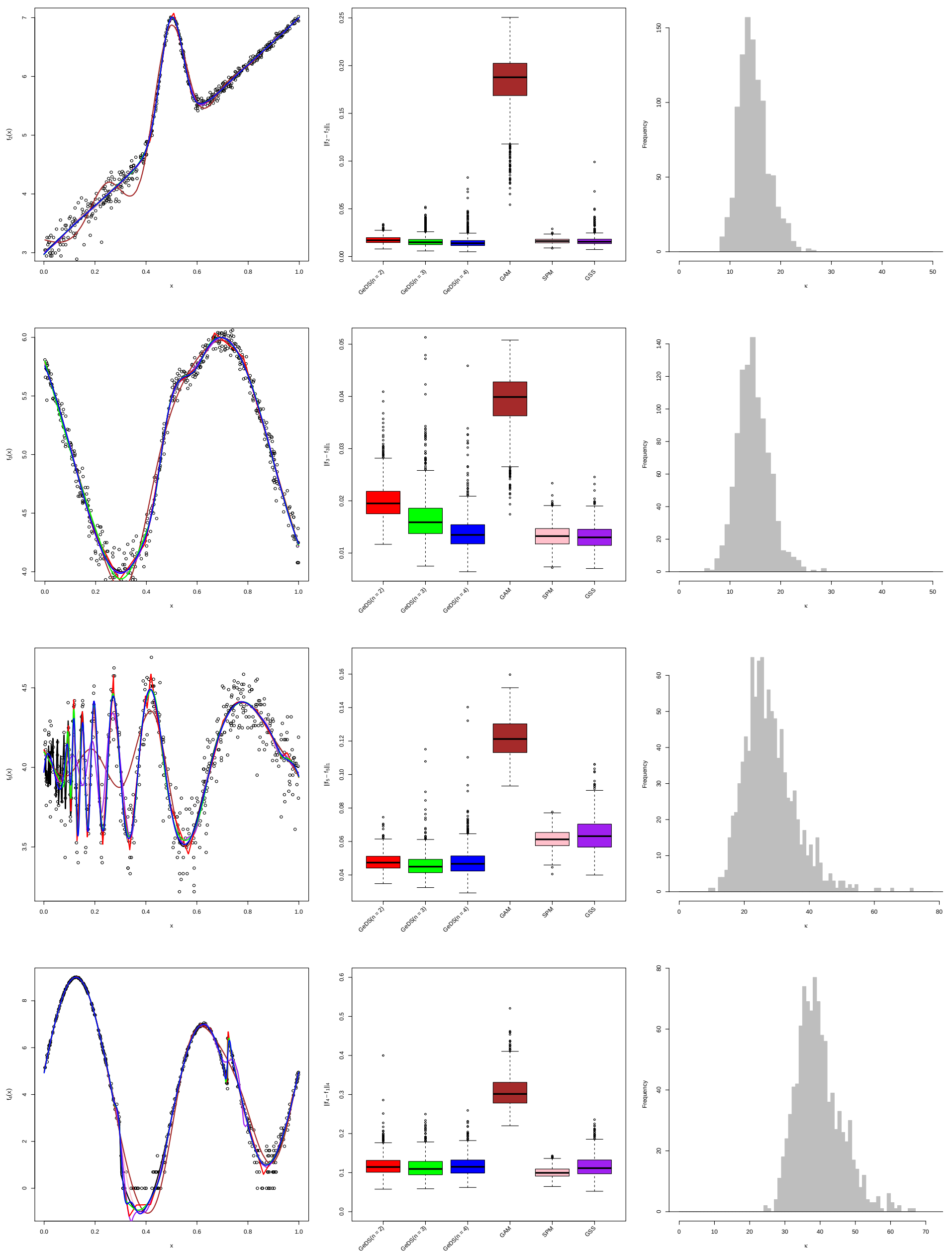


Figure 2.11: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Poisson samples from the predictors in table 2.7.

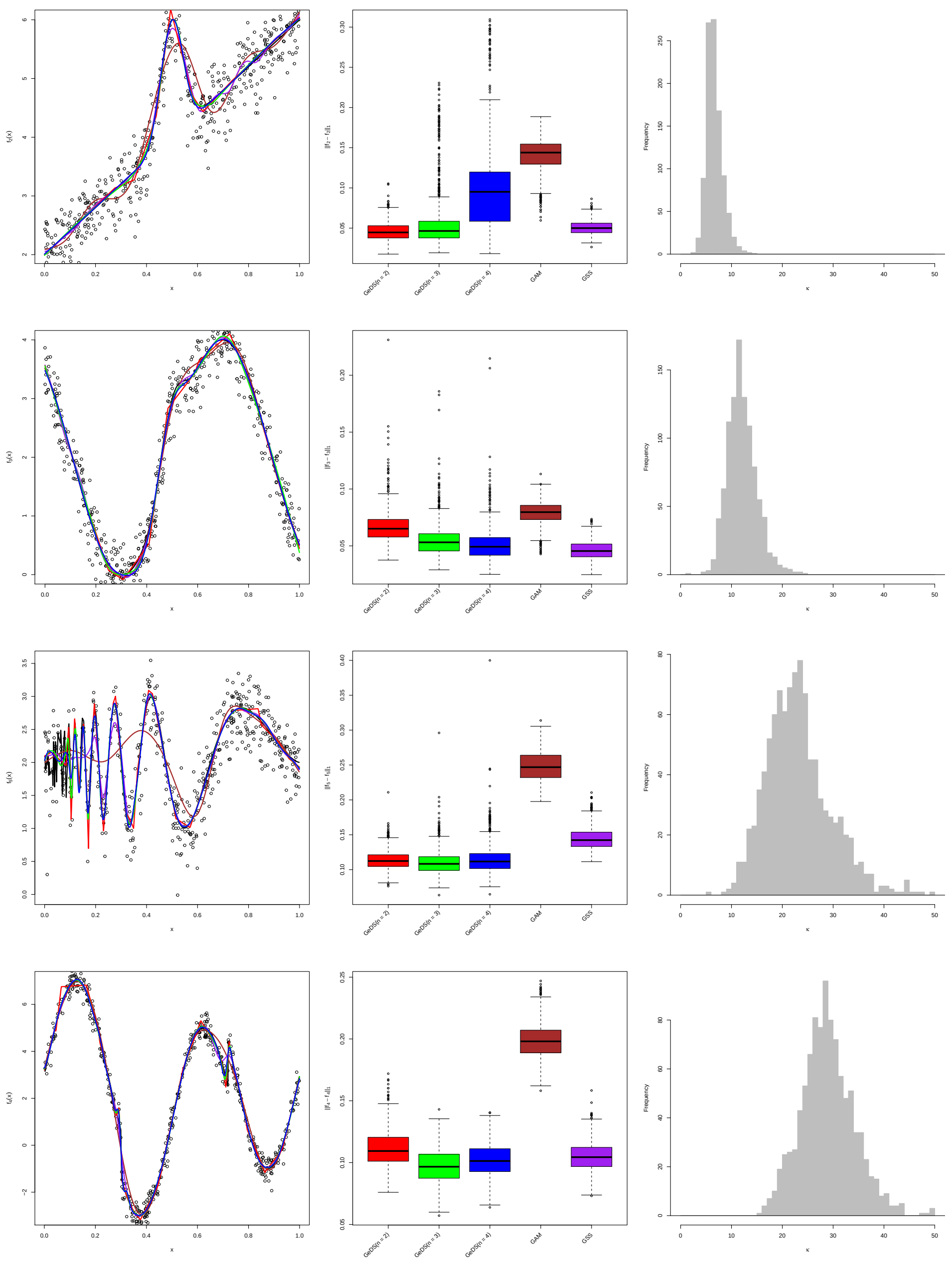


Figure 2.12: Comparison of the linear, quadratic and cubic `GeDS` to the `gss`, `mgcv` and `SemiPar` models, based on fitting 1000 Gamma samples from the predictors in table 2.7.

Chapter 3

An application of nonparametric methods: insurance ratemaking

3.1 Introduction

As mentioned in Section 1.8, ratemaking is one of the most important tasks for an insurance company.

A correct and precise assessment of the risk associated to a new contract is the key for a sane business. Insurers should set premiums in order to earn enough money to make risk assumption activity profitable, but on the other hand the price of a policy should not discourage potential policyholders.

Moreover it is well known that one of the main curses of the insurance business is the adverse selection of the policyholders. A premium not enough personalized may be perceived as too expensive for low-risk possible policyholders, while it would be convenient for high-risk ones. This happens because the premium should be a weighted average of the personalized premiums and it should be enough to cover the claims of all the population. Hence only the low-risk customers would be discouraged from this kind of contract and the total amount of premiums may not be enough for all the incurred claims.

From a theoretical point of view, the more an insurer is able to correctly estimate the risk for each single contract and thus to personalize the premium, the more its activity will be stable and profitable. Unfortunately in practice this is difficult or even impossible to achieve for at least two reasons.

Not all the characteristics of each single policyholder are observable. If we consider an MTPL contract, we may think that the risk depends upon several variables about the driver (age, sex, zone of residence, driving style) and the car (age, power, whether it has been correctly maintained). Some of them are unobservable and hence the insurer cannot take them into account ex-ante. Strategies to personalize premiums incorporating also unobservable variables consider *a posteriori* ratemaking and involve e.g. credibility theory [Bühlmann and Gisler (2005)][*].

Throughout this chapter however we will concentrate on the *a priori* personalization and hence on the observable variables that can be considered. Those variables can be either discrete or continuous. Historically, common practice was to group continuous variables in classes in order make estimates more reliable ad stable, as a single coefficient would have been based on a wider dataset. Binning a continuous variable makes the model simpler to be understood by a policyholder and to be handled by the agent who will concretely compute the premium.

Prudent actuaries should hence consider a discretization in intervals of the continuous variables variables and fit a constant risk function on each of them. The arising model is in fact a stepwise constant risk function to be estimated from the data instead of a continuous one. This procedure can be performed in several ways depending on the type of model and on how much flexible the estimate can be.

It may seem that, nowaday, the requirement of simplicity has become less pressing thanks to the diffusion of computers and to the knowledge about smoothing techniques. The necessity of binning continuous variables may seem to be obsolete and indeed if we consider how major insurers in the Italian market deal with the age factor of the policyholders in MTPL LoB, we can find continuous estimates of the risk function.

However, the problem of binning still arises in ratemaking in other ways. As an example we refer to a major insurer that considers separately the age of the policyholder and the age of the driver when computing premiums. Remarkably, the age of the policyholder is modelled continuously[†], while the age of the driver is binned in classes.

3.1.1 Discretizing a risk function

Binning a continuous variable is an operation that leads to a loss of information. The discretization of a continuous function introduces some bias in the model and if this procedure is not performed according to a sound method, some arbitrariness is introduced.

Let us consider a risk factor measured by a continuous variable and a monotonic risk function, as in Figure 3.1. Indeed, binning the variable in categories implies to model the risk with a stepwise constant function.

The choice of the number of categories and of the thresholds is usually subjective and driven by two conflicting purposes, as usual in statistical

[*]Moreover some of them e.g. sex, cannot be considered in ratemaking as it would be seen as a discrimination in many countries.

[†]More precisely, it is binned in years.

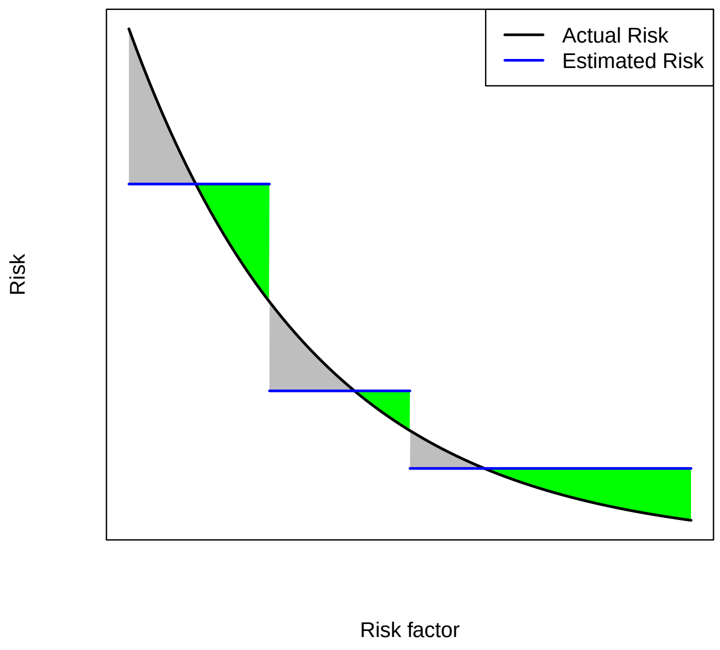


Figure 3.1: example of

problems. On one hand, the model should be as simple as possible, while on the other it should adapt as well as possible to the risk function.

The simple plot in Figure 3.1 easily explains which is the cost of this practice whenever it is reasonable to assume that the a small variation of the risk factor implies a small variation in the risk. Hence we will assume, as usual, that the risk is a function of the continuous variable identified as the risk factor.

Considering each single bin, the risk will be underestimated for some policyholders (the grey regions), while it will be overestimated for others (the green regions). An optimal choice of the thresholds would minimize the bias in the estimates, but preserve the simplicity of the model. Hence on one hand, not too many categories should be used, but the binning should be refined enough to ensure homogeneity with respect to the riskiness.

As usual in many statistical problems, we will try to optimize one of the two objectives, under a constraint. In the remainder of this chapter, we will consider cases where the number of categories is fixed and hence we will concentrate on the choice of the thresholds of the bins.

3.2 The problem and Existing methods

Several methods have been proposed for the general purpose of choosing properly the intervals on which to bin a continuous variable and for some of them statistical software is available, e.g [Bivand (2015)].

The problem can be placed as a particular application of the *restricted problem* described in [Fisher (1958)]. In this framework one seeks for an optimal partition that minimizes a measure of heterogeneity within groups, but preserves an a priori ordering of the elements. More formally, we suppose we observe couples $\{y_i, z_i\}_{i=1}^n$, where y_i is the variable of interest and we assume it depends upon z_i, so that

$$E(y) = f(z),$$

where f has the form

$$f(z) = \sum_{i=1}^{M} c_i I_{R_i}(z),$$

where c_i are constants, $R_1, \ldots, R_M$ is a partition of R and $I_A(z)$ is the indicator function, i.e.

$$I_A(z) = \begin{cases} 0 & \text{if } z \notin A \\ 1 & \text{if } z \in A \end{cases}.$$

However most of these methods have been proposed for continuous and monotonous variables and they are not applicable in our problem, as in general the risk cannot be assumed to have a monotonic behaviour. Considering again a model for claims frequency with respect to the driver's age in MTPL insurance. It is well known that the higher frequency is observed on young and old drivers, while it will be lower for middle-aged ones. Hence a monotonicity assumption would be misleading.

3.2.1 Arbitrary Choice

A choice of the thresholds can be made as an arbitrary choice, hence no rigorous strategy of optimization of an objective function is implemented.

This is quite common practice, as this allows to choose the thresholds directly incorporating the expertise knowledge or allowing to further simplify the model choosing bins of equal width. Of course this approach has also the advantage of stability. The estimates of the constant pieces should be data driven, but the size of the bins would not be affected.

3.2.2 Tree

If the distribution of y is assumed to be Gaussian, one model that performs this procedure directly is the standard regression tree as presented in Section 1.5.2. In this section we will use its R ([R Core Team (2015)]) implementation available in the package **rpart** ([Therneau et alt. (2015)]). In **rpart**, several methods are supported to build the tree. When we will consider the Distribution of y to be Gaussian we will refer to the method “ANOVA”, that means that at each step the best split is selected according to the criterion

$$\arg\max_{\tau;c} \left[\mathrm{RSS} - (\mathrm{RSS_L}(\tau, c) + \mathrm{RSS_R}(\tau, c))\right], \tag{3.1}$$

where $\mathrm{RSS}, \mathrm{RSS_L}, \mathrm{RSS_R}$ are the residual sums of squares respectively before the splitting, of the left leaf and of the right one.

Indeed as we will deal with a count variable, we will consider the more refined Poisson regression tree. **rpart** implementation considers, instead of the RSS, the model deviances so that, for the leaf h, where there are n_h observations, whose indices are collected in J_h,

$$D_h = \sum_{j \in J_h} \left[y_i \log\left(\frac{y_i}{\lambda_h}\right) - (y_i - \lambda_h) \right],$$

where $\lambda_h = n_h^{-1} \sum_{i \in J_h} y_i$. During the pruning stage, it is often necessary to consider a slight modification as λ_h may be null and hence D_h infinity. Hence in order to have a more stable method, authors considered a slight modification of λ_h, justifying it in a Bayesian framework. Assuming as prior $\lambda_h \sim \mathrm{Gamma}(\alpha, \beta)$, where $\alpha\beta^{-1} = n^{-1} \sum_{i=1}^{n} y_i$ and $\theta^{-1/2} = k$, a known coefficient of variation, the Bayesian estimate is

$$\tilde{\lambda}_h = \frac{\alpha + \sum_{i \in J_h} y_i}{\beta + n_h}$$

and this is used to replace λ_h in previous formulae. In what follows we will consider $k = 1$.

3.2.3 O’Brien’s method

In literature, we find another method in [O’Brien (2004)] that has been designed to deal with response variables coming from any distribution of the exponential dispersion family as defined in Section 1.7, assuming the variable z is limited. Hence the goal is to find a partition of R, $R_1, \dots, R_M$ defined by the bounds $\tau = \{\tau_j\}_{j=0}^{M}$, with $\tau_0 < \cdots < \tau_M$ that is optimal according to some rule. For simplicity it is assumed that $z \in [0, 1]$, hence $\tau_0 = 0$ and $\tau_M = 1$.

Setting $\mathrm{E}(y_i) = \mu_i = \mu(z_i)$ and $\mathrm{Var}(y_i) = a(\phi)V(\mu_i) = a(\phi)v_i$, the goal is to find the optimal partition R^* that minimizes

$$\mathrm{AED}(\tau) = \mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n} \frac{(\bar{y}_{R_j} - \mu_i)^2}{v_i}\right], \tag{3.2}$$

where j_i is the index of the interval to which ith observation is assigned. Here we see that remarkably the aim is to find an estimate that adapts to the true mean of the observations rather than to the data themselves.

However to find the optimal partition is in general computationally not feasible and hence it is preferred to minimize a large sample approximation to AED. Assuming that

i z_i are constants and their spacing can be expressed by $F_z^{-1}(i/(n+1))$, where F_z is a proper CDF and $f_z = F_z'$ is a density function

ii the cut points can be expressed as $\tau_j = F_\tau^{-1}(j/M)$ where F_τ is a proper CDF and $f_\tau = F_\tau'$ is a density function

iii f_τ and f_z are continuous and positive on $[0,1]$

iv $\mu'(z)$ is continuous on $[0,1]$

for large n and M,

$$AED(\tau) \approx \frac{J(f_\tau;\mu)}{12M^2} + \frac{a(\phi)M}{n} \tag{3.3}$$

with

$$J(f_\tau;\mu) = \int_0^1 \frac{(\mu'(u)f_z(u))}{f_\tau^2(u)v(\mu(u))}\mathrm{d}u.$$

It can then be shown that the density that minimizes $J(f_\tau;\mu)$ is

$$f_\tau^*(t) = \left[\frac{(\mu'(t)f_z(t))}{f_\tau^2(t)v(\mu(t))}\right]^{1/3}.$$

However, given some estimates $(\hat{\mu}_i, \hat{v}_i)$ for (μ_i, v_i), a discrete approximation of (3.3) is minimized by choosing τ according to

$$\tau_j = \min_z |\hat{F}_\tau(z) \geq \frac{j}{M},$$

where

$$\hat{F}_\tau(t) = \frac{\sum_{i=1}^n \hat{\delta}_i I_{R(z_i)}(t)}{\sum_{i=1}^n \hat{\delta}_i},$$

with $R(z_i) = \{t \in \mathrm{R} | t \leq z_i\}$ and $\hat{\delta}_i = (\hat{\mu}_i - \hat{\mu}_{i-1})^{\frac{2}{3}} (\hat{v}_{i-1})^{\frac{-1}{3}}$.

In this section our interest was just to produce an optimal choice of the partition that may apply in insurance ratemaking, hence we omitted the parts about the estimation of the optimal number of classes M or of the partition having set the number of individuals in each class. This can however be found in [O'Brien (2004)].

3.3 The model

O'Brien's approach remarkably relies on the intuition that within-bins homogeneity should not be considered with respect to the observed values themselves, but with respect to the characteristics of their distributions and this lead to the choice of (3.2) as the objective function. More formally, the objective of the partition is that it should adapt to the pattern of the expected values μ_i, rather than on the y_i.

Our proposal is then to modify the regression tree approach in order to implement this idea. Indeed if we have a tool that allows us to compute an estimate of the pattern of the expectations, we can apply directly on that estimate the regression tree procedure.

This is in fact the key of the model we propose. As we have seen in previous chapters, there are several ways to produce estimates for the pattern of the μ_i. In general, however, as we are modelling a count variable y_i describing the number of claims reported by ith policyholder, we assume that

$$y_i \sim ODP(\mu_i; \phi), \tag{3.4}$$

and we also assume that $\log(\mu_i) = f(z_i)$, f being a smooth function.

In Section 1.7, we presented a family of models that allows to compute estimates for this model and we referred to them as GAM. GAM procedures are allow to easily estimate f in a stable way[‡].

3.3.1 Some variants

Of course several variants of this methodology can be developed with slight modifications. In principle, any other regression procedure can be used in the fitting process in order to obtain an estimate for the μ_i. We proposed to use a GAM procedure as in ratemaking, usually, several variables should be incorporated and those models provide an elegant way to include them. In principle, however, also other regression models can be used, such as, in particular, GeDS methodology.

Several variants of this method can be proposed as, in principle, other regression models can be used in order to estimate the pattern of μ_i. In particular we focus on the variants built including the GeDS regression.

As we discussed in Chapter 2, in GeDS procedure it is suggested to tune case by case the stopping rule, as it may provide unstable results. Hence we consider also two sub-variants, including two modifications of stage A of the algorithm.

The first has been introduced in Section 2.7.2 and it considers to replace the Stage A procedure by a MARS regression. Relying on MARS procedure

[‡]For this purpose, we will take advantage of the **mgcv** R package, [Wood (2006)]. The estimation procedure will be based on thin plate regression splines and he smoothing parameter will be selected via GCV criterion.

guarantees much more stability, even if the knot placement is not chosen in a geometrical way as claimed by GeDS procedure. A second way is to fix the number of knots of the estimation procedure. This can be considered as a minor change if compared to the previous one, as indeed this can be intended as a replacement of the sopping rule with a trivial one.

Once obtained an estimate $\hat{f}$ we propose to choose the partition with a regression tree based on the data $\{\hat{f}(z_i), z_i\}$. As these estimates are on a transformed scale with respect to the observed variable y, there is no need to apply a Poisson tree, and thus we should rely on the usual one. However, as we discussed in Section 2.3.1, GeDS regression can produce estimates on the f scale that are useless if there is a wide interval where the observed values are all zeroes. The estimate of the function f on an interval can be minus infinity on the predictor scale and hence the binning cannot be made through a regression tree on these estimates, as otherwise the thresholds would be shrunk toward that interval. In order to avoid this, we will consider the estimates on the μ scale and hence we will apply the Poisson regression tree on the estimates.

3.4 Performances

In this section we aim to show the performances of the estimators we described and proposed through a simulation study. Our study is focused on the study of the statistical properties of the proposed methods and hence we do not implement a whole ratemaking model, but we consider just a simplified one, taking into account just one factor.

One of the main drivers of the riskiness of a policy in MTPL is the age of the policyholder. In general actuaries expect to observe a high claim frequency for young policyholders as they are less experienced than other drivers. At the same time, a similar behaviour is expected also for policyholders beyond a certain age as, while experience increases, alertness decreases. However the effect is mitigated by the fact that an old person usually uses the car less often as (s)he has probably already retired.

Modelling this effect through a straight line leads to unreliable estimates and care should be taken also if other parametric models are considered. Flexible semi-parametric or non-parametric models should be preferred and thence we consider this example for our simulation study. Binning the policyholder age variable has been common practice as it has been a naïve way to model it allowing flexibility.

In our simulation study we will simulate data according to two functions. In a first test we will consider a simple monotonic decreasing function, obtained from real data, from the dataset distributed within the R package **MASS** ([Venables and Ripley (2002)]). In that dataset the age is already discretized in four categories, but we can recover a monotonic decreasing pattern, imposing an exponential decay. Hence the resulting risk function is the black solid line drawn in Figure 3.2.

In a second test we will use a more reliable risk function. As described, the pattern of the claim frequency with respect to the policyholder age is more complicated than a monotonous risk function. Thence we introduce a function, drawn in the black solid line in Figure 3.3. We obtained this function directly from a major insurer active in the Italian market. Although we had the possibility to use just a rescaled version of the true one, we consider it to be a more realistic situation.

Basically we simulate a set of data according to the model (3.4), where the values z_i represent ages of the policyholders considering the two risk functions described. We assume we have $N = 10000$ policyholders aged between 18 and 95 and we simulate for each of them the number of claims $y_i^{\mathrm{new},j}$ $i = 1, \dots, N$ occurred in a year, the superscript j indexing the simulations. For each policyholder the age is set via pseudo-random simulation, according to a realistic distribution of ages. For the number of claims, instead, we assume it is Poisson distributed, the rate being implied from the risk function.

On each vector, we run all the models described in previous sections, obtaining the thresholds and the estimates of the frequencies for each age $\mathrm{E}[Y_i|x_i]$. We will denote them by $\hat{y}_{i,k}^{(j)}$, where the subscript k identifies the method adopted.

Computing premiums based on a smooth estimate $\hat{f}(z)$ or, more precisely on $\exp[\hat{f}(z)]$, would be optimal, in the sense that the discrimination of the policyholders with respect to the age would be carried out as well as possible. Even if for practical reasons this may not be possible, as discussed, we store also the results obtained from a GAM and we will use them as a best practice for the sake of comparison.

3.4.1 Practical implementation of the methods

In practice we will consider nine methods in our simulation. Hence here we describe the practical aspects of them specifying, in parenthesis, the names we will use in the remainder.

The simplest model we implement is the traditional one, that considers a priori fixed thresholds (fixed). In fact we do not expect it to perform well, but we include it as a minimum benchmark for all the other models and in order to show that there may be a significant gain in applying a more refined model.

The second method is O'Brien's one (OB), that has already been widely described in Section 3.2.3. The estimate of $(\hat{\mu}_i, \hat{v}_i)$ we adopt is obtained through the GAM with the choice of the amount of smoothing based on the UBRE criterion. On the estimates of the function f produced by the same GAM, we also run a standard regression tree (GAM-Tree) in order to select

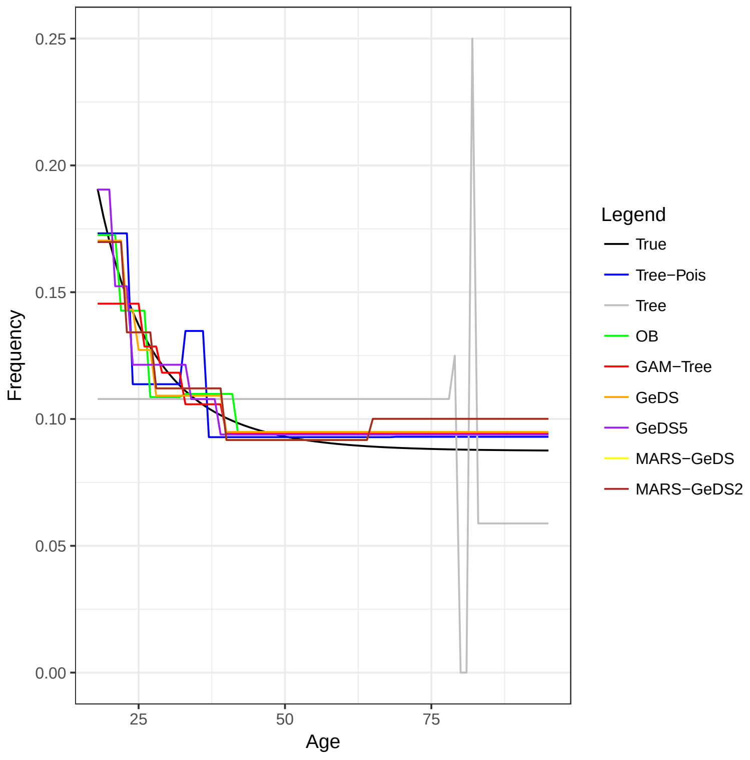


Figure 3.2: Claim frequency as a monotonous function of age and partitions obtained with several models

the thresholds. We use also the standard regression tree (Tree) directly applied to the simulated data and its Poisson based variant (Tree-Pois).

We then implement four modifications of the regression tree based on GeDS methodology: the pure GeDS (GeDS), the GedS with five internal knots (GeDS5), the combination of MARS and GeDS procedures (MARS-GeDS) and the same imposing to find knots in MARS stage with at least 10 years between them (MARS-GeDS2). As specified above, to all the estimates it is applied a Poisson-regression tree in order to estimate a partition.

Of course in insurance practice the number of classes should be limited, so we set $M = 5$, i.e. we will look for 4 splits.

Figure 3.2 shows a partition on the monotonous function recovered from the `Insurance` dataset, while Fig 3.3 shows a partition of the wigglier function adopted by the Italian insurer.

This is the result of a particular couple of samples, but it can already be shown the issue arising when using the standard regression tree procedure. As we can see, this procedure selects two very narrow splits of the age variable and the estimate moves away from the true function. This is due to the fact that the standard regression tree is designed to work with Gaussian

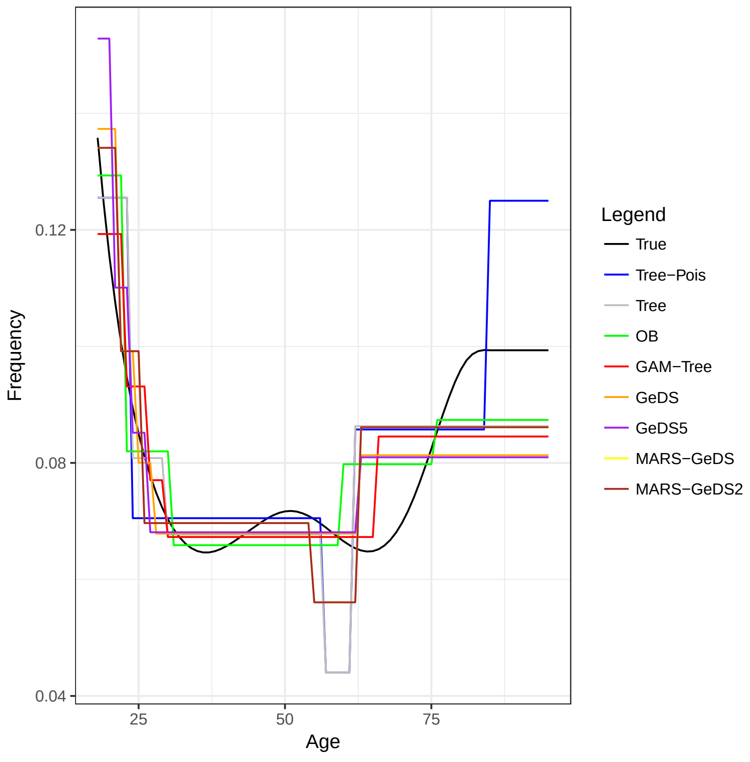


Figure 3.3: Claim frequency as a non-monotonous function of age and partitions obtained with several models

data, while here data are counts generated from a Poisson distribution.

3.5 Montecarlo Simulation

In order to study properties of the other methods, we need a more thorough approach and hence we implement a Montecarlo simulation. Hence, considering again the same sample size $N = 10000$, we draw 2000 samples according to the models described and we compute the estimates according to the eight different models.

3.5.1 GoF Measures

Some rigorous measure is necessary in order to study how a method performs compared to another one. We propose to measure the loss of information by measuring the prediction error on each single simulated sample.

We define it as

$$e_k^{(j)} = \sum_{i=1}^{N} \left(y_i^{\text{new},j} - \hat{y}_{i,k}^{(j)} \right)^2$$

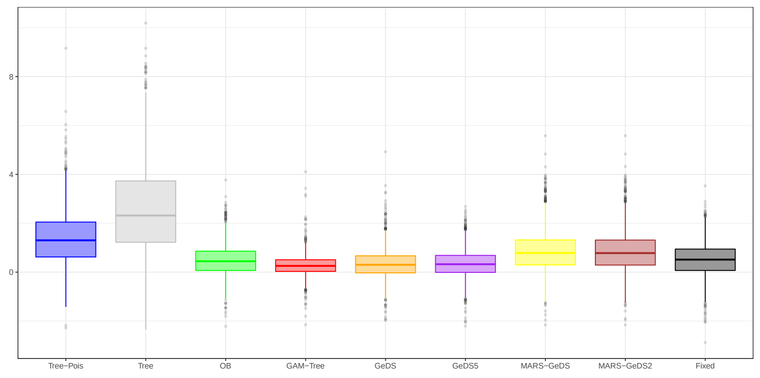


Figure 3.4: Boxplots of the prediction errors obtained with the models presented in Section 3.4, with simulations based on a monotonous risk function

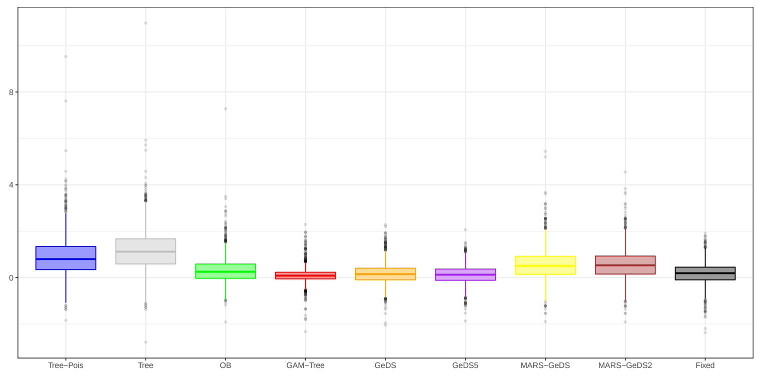


Figure 3.5: Boxplots of the prediction errors obtained with the models presented in Section 3.4, with simulations based on a non-monotonous risk function

and overall measure of prediction error is

$$E_k = \sum_{j=1}^{J} e_k^{(j)}.$$

Note that we measure the error with respect to the single simulated values rather than to the value of the underlying risk function. Hence we have the possibility to compare the estimators only if applied to the same simulated sample.

As we consider the pure GAM fit to be somehow the best but practically unapplicable estimator, we will consider in our plots the difference between the prediction error of the methods and the one of the GAM. Hence we will draw the increase in prediction error due to the model choice. We define it as

$$\tilde{e}_k^{(j)} = e_k^{(j)} - e_0^{(j)},$$

with e_0 denoting the prediction error of the pure GAM.

3.5.2 Results

Results are resumed in the boxplots in Figure 3.4 and 3.5. From these boxplots some interesting and unexpected conclusions can be outlined.

Considering the first Figure 3.4, we see that the worst approaches are the ones based on the regression tree applied directly to the data. It is remarkable the fact that their performances are even worse than the ones we obtain from the a priori fixed thresholds approach. This is a conclusion that we were somehow expecting, but surprisingly we see that also O'Brien's method do not show significantly better results than the fixed coefficients one.

The best estimator is the GAM-Tree, as it can be seen. However, remarkably, also the GeDS and GeDS5 methods show a good behaviour. Considering the outliers, GeDS5 method seems to be a bit better, and this suggests that we were correct in fearing that sometimes the algorithm is too unstable in the selection of the number of knots. We can also see that MARS-GeDS based estimators give in general worse results. Our experience also suggests that the number of knots selected by MARS-GeDS procedures is too low to allow a good approximation of the true underlying function.

Performances of O'Brien's model are quite good, but they are worse than the ones obtained from GAM-Tree and GeDS5 methods.

Figure 3.5 shows the boxplots of the prediction error resulting from the Montecarlo simulation based on the second model.

Also in this case, we see that the Tree and Tree-Pois methods do not perform well. Considering the other methods, similar conclusions can be derived.

We also note that, while with the previous model MARS-GeDS and MARS-GeDS2 showed almost the same results, here the distributions appear to be different, although they are again very similar. This is a sign that the constraint that imposes at least 10 years between two knots here is effective, while for the previous model it didn't affect the estimates.

Surprisingly, in this case the fixed threshold method gives results that are sensibly better also if compared with the MARS-GeDS methods.

Table 3.1 resumes some basic statistics of the results we get from the simulation. While the first column identifies the estimation procedure followed, columns 3 and 4 show the means and the standard deviations of the prediction errors observed when simulating from the monotonous risk function. Columns 5 and 6, then, resume the same results obtained under the model that involves the wigglier function.

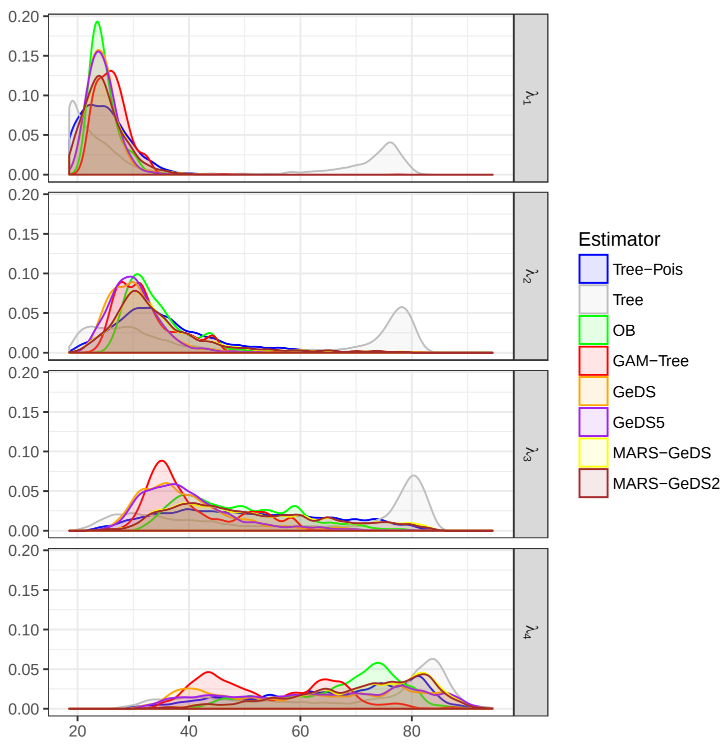


Figure 3.6: Density estimates of the distributions of the four thresholds selected by the eight (non-trivial) methods. The 2000 samples of 10000 observations are simulated according to the monotonous risk function.

This table quantifies what was apparent from Figures 3.4 and 3.5. In particular we see that also GeDS5 estimator gives better results than the Fixed thresholds one, both considering the mean and the dispersion of the distribution of the prediction error.

One interpretation of this is that with a non linear function such as the underlying one, the optimal choice of the thresholds is not straightforward. With some methods, the choice of the thresholds is so inaccurate that it is better to choose them a priori than to rely on estimates.

The last intuition is also supported by the plots in Figure 3.6 and 3.7. Figure 3.6 shows a comparison of kernel density estimates of the distribution of the thresholds selected by the eight methods, omitting the trivial estimator. Density estimates are obtained with Gaussian kernel, setting the bandwidth parameter equal to 1. In the first panel we plot densities of the first (lower) threshold estimated and so on, up to the fourth panel, representing the distribution for the last threshold. In Figure 3.7, then, we draw the same plot, but based on the results of the simulation involving the non-monotonous risk function.

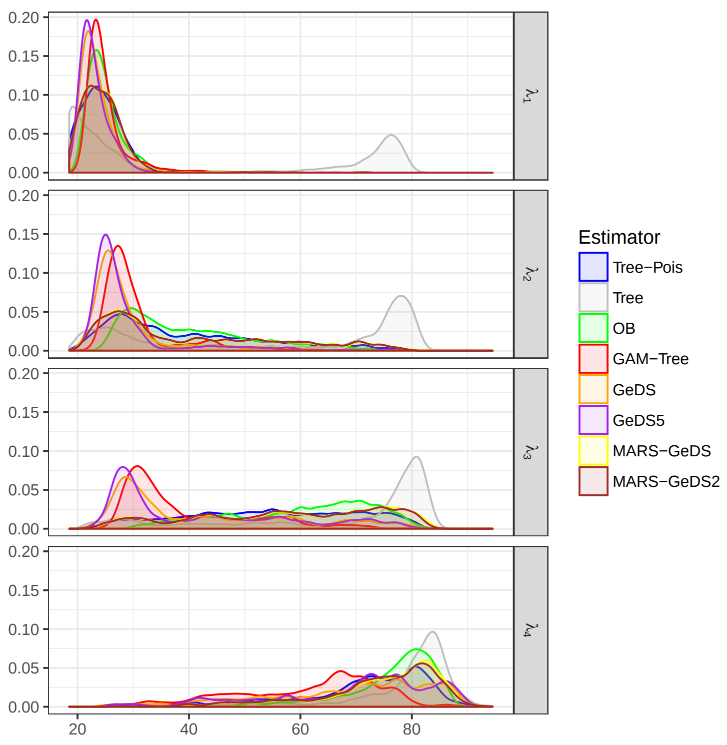


Figure 3.7: Density estimates of the distributions of the four thresholds selected by the eight (non-trivial) methods. The 2000 samples of 10000 observations are simulated according to the non-monotonous risk function.

By plots in Figure 3.6 we can conclude that all the methods but the simple regression tree are quite stable in the selection at least of the first two splits. However, we can see that some of the distributions associated show heavier tails than others even in the second panel. We note also that the distributions are more dispersed in the lower panels than in the higher ones. This effect is due to the fact that the distribution of the ages is not uniform and we simulate a small number of policyholders and hence of data at higher ages. Moreover also the underlying risk function .

We also see that the estimates produced by the standard regression tree are useless considering the nature of the problem. Often the first split is selected around the 75 years and this does not make sense in MTPL ratemaking.

In particular most stable estimates are achieved by GAM-Tree, GeDS and GeDS5 methods, but we see that the thresholds selected by GeDS-based methods are in general lower than GAM ones.

Considering Figure 3.7, at a first glance, we see that the density estimates of the three methods that were performing well in the other simulation

	Monotonic		Non-monotonic	
Tree-Pois	1.399	(1.110)	0.886	(0.836)
Tree	2.602	(1.831)	1.169	(0.887)
OB	0.486	(0.635)	0.318	(0.550)
GAM-Tree	0.276	(0.423)	0.097	(0.332)
GeDS	0.336	(0.599)	0.171	(0.436)
GeDS5	0.349	(0.576)	0.120	(0.400)
MARS-GeDS	0.868	(0.852)	0.572	(0.653)
MARS-GeDS2	0.861	(0.852)	0.574	(0.639)
Fixed	0.507	(0.694)	0.176	(0.461)

Table 3.1: resume statistics

appear to be even more leptokurtic and hence the selection of the thresholds under this function seems to be more stable than under the previous one. Considering the picture in Figure 3.2, we can see that in fact the risk function is almost flat from around the age of 60 on. As the observations are discrete, in finite samples it is very difficult to see the effects of the dissimilarity between the claims frequency at ages higher than 60.

However, if we consider the other methods, we see the opposite behaviour for almost all of them. Already in the second panel we see that the distributions are more dispersed.

It is also remarkable that the selection of the last two thresholds in the O'Brien's method is the most stable one. However, in practice, setting a split at the age of 80 may not be reasonable as the number of policyholders at those ages is low.

Chapter 4

Conclusions

Several flexible regression models are nowaday available as they have been implemented in most statistical software. This makes these models immediately accessible to the researcher who is aiming to limit the imposition of constraints and assumptions when studying a problem.

As we have seen in Chapter 1 the class of those models is very wide as each of them implements different strategies in order to allow to obtain flexible estimates. Thus, each of them ensures the estimates to fulfil some properties. It is a task of the researcher to choose among the models, selecting the one that better suits the purposes of the study, ensuring to both fit the data and at the same time to meet the requirements imposed by practical aspects involving the aim of the study.

Some "noble" and widely accepted regression models are now part of the bouquet of tools of researchers and they are often used in several different fields, but it does not exists a model that outperforms the others in all possible situations.

In Chapter 2 we introduced GeDS model, a non-parametric approach that is based on a geometrical interpretation of the placement of the knots of a polynomial spline. We showed that this model, with some objective functions, is able to outperform other flexible models, although some of its features still need to be refined.

It has been possible study some properties of the estimates obtained via GeDS regression, by setting the framework to obtain asymptotically correct confidence intervals and a consistent version of the likelihood ratio test. In this work we presented some theoretical asymptotic results, while the finite sample properties still need to be studied.

Some efforts were also spent in order to show that this approach can be efficiently implemented is statistical software. This estimator still requires some care in order to be applied properly, depending on the data under study. In order to allow the researcher to calibrate the estimation procedure, we left in the algorithm several optional inputs that modify some of the features and help to get a reliable estimate.

In this thesis we presented several examples of application of flexible regression models in the actuarial field. In particular in Chapter 3 we developed some models that can be applied in the ratemaking problem. In this field, estimators should return estimates as accurate as possible, but, at the same time, they should be simple and understandable.

We then introduced some models that combine together other more simple ones and we showed their performances through simulation studies based first on a theoretical example and then on a more realistic one. We found that they perform better than other models adopted in common practice, yet preserving the simplicity of the results. The simulation studies were applied on univariate data and it would be interesting, as further research on this field to test them on a multivariate framework. In order to validate these models, then, it would be necessary to test them on real data, but up to now we didn't have the chance to do this.

In summary, we aimed to give our contribution to the flexible regression modelling by the introduction and validation of some new models. As our aim was to give a contribution useful from the point of view of an insurance company, we did not focus only on theoretical aspects, but we also took care of practical ones. Our work can then be intended as part of the applied statistics framework.

We hope that the models introduced in this work may become part of the set of tools used by actuaries in practice, although we have seen that GeDS model can be useful in a wider range of fields.

We wish that the models presented may become useful in insurance practice helping the companies to develop their own internal models for the risk assessment. As we implicitly introduced in the preface, the models we developed are not intended to cover this task for the companies as a whole, but the aim was to focus on the basis on which a truly reliable and comprehensive model can be built.

These models should help actuaries in the construction of several small bricks of the whole internal model. If the simpler modules of the whole model are built in order to properly assess and evaluate risks and if the behaviour of those sub-models is studied, then it is simpler and more efficient the aggregation in a comprehensive model.

Bibliography

[Azzalini and Scarpa (2004)] Azzalini, A. and Scarpa, B. (2004). Analisi dei Dati e Data Mining. Springer–Verlag.

[Akima (1978)] Akima, H. (1978). A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points. *ACM Transactions on Mathematical Software.* **4**, 148–64.

[Akima (1996)] Akima, H. (1996). Algorithm 761: scattered-data surface fitting that has the accuracy of a cubic polynomial. *ACM Transactions on Mathematical Software.* **22**, 362–71.

[Barrow and Smith (1978)] Barrow, D.L. and Smith, P.W. (1978). Asymptotic properties of best $L_2[0,1]$ approximation by splines with variable knots. *The Quarterly of Applied Mathematics*, **36**, 293–304.

[Biller (2000)] Biller, C. (2000). Adaptive Bayesian regression splines in semiparametric generalized linear models. *J. Comput. and Graph. Stat.*, **9**, 122–40.

[Bivand (2015)] Bivand, R. (2015). classInt: Choose Univariate Class Intervals. R package version 0.1-23. http://CRAN.R-project.org/package=classInt

[Bühlmann and Gisler (2005)] Bühlmann, H. and Gisler, A. A Course in Credibility Theory and its Applications. *Springer.*

[Camarda 2012] Camarda, C.G. (2012). MortalitySmooth: An R Package for Smoothing Poisson Counts with P-Splines. *JSS*, **50**(1), 1–24.

[Craven and Wahba (1978)] Craven, P. and Wahba, G. (1978). Smoothing Noisy Data with Spline Functions. *G. Numer. Math.*, **31**(4), 377–403.

[Currie (2004)] Currie, I.D., Durban, M. and Eilers, P.H.C. (2004). Smoothing and forecasting mortality rates. *Statistical modelling*, **4**, 279–98.

[Currie (2016)] Currie, I.D. (2016). On fitting generalized linear and non-linear models of mortality. *Scandinavian Actuarial Journal*, **2016**(4), 356–83.

[Curry and Schoenberg (1947)] Curry, H.B. and Schoenberg, I.J. (1947). On spline distributions and their limits: the Polya distribution functions. *Bulletin of the American Mathematical Society*, **53**, 1114.

[Curry and Schoenberg (1966)] Curry, H.B. and Schoenberg, I.J. (1966). On Pólya frequency functions IV: the fundamental spline functions and their limits. *Journal d'Analyse Mathématique*, **17**(1), 71–107.

[Davies (1980)] Davies, R.B. (1980). Algorithm AS 155: The Distribution of a Linear Combination of χ^2 Random Variables. *J. R. Statist. Soc. C*, **29**(3), 323–33.

[De Boor (2001)] De Boor, C. (2001). A Practical Giude to Splines. Revised Edition. *Springer–Verlag.*

[Denison et al. (1998)] Denison, D.G.T., Mallick, B.K. and Smith, A.F.M. (1998). Bayesian MARS. *Statistics and Computing*, **8**, 337–46.

[Denuit and Lang (2004)] Denuit, M., Lang, S., (2004). Nonlife ratemaking with Bayesian GAMâĂŹs. *IME*, **35**, 627–47

[Dobson (2001)] Dobson, A.J. (2001). An Introduction to Generalized Linear Models (2nd ed.). *Chapman & Hall/CRC.*

[Duchesne and Lafaye De Micheaux (2010)] Duchesne, P. and Lafaye De Micheaux, P. (2010). Computing the distribution of quadratic forms: Further comparisons between the Liu-Tang-Zhang approximation and exact methods. *Computational Statistics and Data Analysis*, **54**, 858–62.

[Duchon (1977)] Duchon, J. (1977). Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in *Construction Theory of Functions of Several Variables*, *Springer.*

[Eilers and Marx (1996)] Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties (with discussion). *Statistical Science*, **11**, 89–121.

[England and Verrall (2002)] England, P.D., and Verrall, R.J. (2002). Stochastic claims reserving in general insurance. *BAJ*, **8**(3), 443–518.

[Fisher (1958)] Fisher, W.D. (1958). On grouping for maximum homogeneity. *JASA*, **53**(284), 789–98.

[Farin (2001)] Farin, G. (2001). Curves and Surfaces for CAGD, A Practical Guide. Fifth Edition. *San Francisco: Morgan Kaufmann Publishers.*

[Friedman (1991)] Friedman, J.H. (1991). Multivariate adaptive regression splines (with discussion), *Annals of Statistics*, **19**(1), 1–141.

[Gavin et alt. (1993)] Gavin, J.B., Haberman, S. and Verrall, R.J. (1993). Moving weighted average graduation using kernel estimation. *IME*, **12**(2), 113–26

[Green (1995)] Green, P.J. (1995). Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, **82**, 711–32.

[Gu(2014)] Gu, C. (2014). Smoothing Spline ANOVA Models: R Package **gss**. *Journal of Statistical Software*, **58**(5), 1–25.

[Harville (1974)] Harville, D.A. (1974). Bayesian inference for variance components using only error contrasts. *Biometika*, **61**(2), 383–5.

[Hastie and Tibshirani (1990)] Hastie, T.J. and Tibshirani, R.J. (1990). Generalized Additive Models. *Chapman & Hall, London*

[Huang (2003)] Huang, J.Z. (2003). Local asymptotics for polynomial spline regression. *The Annals of Statistics*, **31**(5), 1600–35.

[Kaishev et al.(2016)] Kaishev, V.K., Dimitrova, D.S., Haberman, S., Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics.* **31**(3), 1079–105.

[Kaishev et al.(2006)] Kaishev, V.K., Dimitrova, D.S., Haberman, S., Verrall, R.J. Geometrically Designed, Variable Knot Regression Splines: Asymptotics and Inference. *Cass Statistical Research Paper No. 29.*

[Kent (1982)] Kent, J.T. (1982). Robust properties of likelihood ratio tests. *Biometrika*, **69**(1), 19–27.

[Kimber et al.(2009)] Kimber, S. A. J., Kreyssig, A., Zhang, Y. Z., Jeschke, H. O., Valenti, R., Yokaichiya, F., Colombier, E., Yan, J., Hansen, T. C., Chatterji, T., McQueeney, R. J., Canfield, P. C., Goldman, A. I. and Argyriou, D. N. (2009). Similarities between structural distortions under pressure and chemical doping in superconducting $BaFe_2As_2$. *Nature Materials*, **8**, 471–475.

[Klein et alt. (2014)] Nonlife ratemaking and risk management with Bayesian generalized additive models for location, scale, and shape. *IME*, **55**, 225–49.

[Lane (1996)] Lane,P.W. (1996). Generalized Nonlinear Models, In: *Proceedings in Computational Statistics, COMPSTAT* (ed. A. Prat), pp. 331–336, Wien: Physica-Verlag.

[Mallick et al. (1999)] Mallick, B.K., Denison, D.G.T. and Smith, A.F.M. (1999). Semiparametric Generalized Linear Models: Bayesian Approaches, in *Generalized Linear Models: A Bayesian Perspective*, *New York: Marcel-Dekker.*

[Mallows (1973)] Mallows, C.L (1973). Some comments on C_p. *Technometrics*, **15**(4), 661–75.

[Nelder and Weddernburn (1972)] Nelder, J.A. and Wedderburn, R.W.M. (1972). Generalized linear models. *J. Roy. Statist. Soc. A*. **135**, 370–84.

[O'Brien (2004)] O'Brien, S.M. (2004). Cutpoint selection for categorizing a continuous predictor. *Biometrics*, **60**, 504–9

[Patterson and Thompson (1971)] Patterson, H.D. and Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, **58**(3), 545–54.

[R Core Team (2015)] R Core Team (2015). R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria.

[Renshaw and Verrall (1998)] Renshaw, A.E., and Verrall, R.J. (1998). A stochastic model underlying the Chain-Ladder technique. *BAJ*, **4**(4), 903–23.

[Ronnegard et al. (2010)] Ronnegard, L., Shen, X. and Alam, M. (2010). **hglm**: A Package for Fitting Hierarchical Generalized Linear Models. *The R Journal*, **2**(2), 20–8.

[Smyth and Verbyla (1996)] Smyth, G.K., and Verbyla, A.P. (1996). A conditional approach to residual maximum likelihood estimation in generalized linear models. *J. Roy. Statist. Soc. B*, **58**, 565–72.

[Schwetlickn and Schütze (1995)] Schwetlick, H. and Schütze, T. (1995) Least squares approximation by splines with free knots. *BIT Numer Math*, **35**, 854–66.

[Taylor (1989)] Taylor, G.C. (1989) Use of spline functions for premium rating by geographic area. *ASTIN Bulletin*,**19**(1), 91–122.

[Therneau et alt. (2015)] Therneau, T., Atkinson, B. and Ripley, B. (2015). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-10. http://CRAN.R-project.org/package=rpart

[Turner and Firth (2015)] Turner, H., and Firth, D. (2015). **gnm**: Generalized Nonlinear Models. *R package version 1.0-8*. URL *https://cran.r-project.org/web/packages/gnm/index.html.*

[Venables and Ripley (2002)] Venables, W.N. and Ripley, B.D. (2002). Modern Applied Statistics with S. Fourth Edition. *Springer, New York.*

[Verrall (1996)] Verrall, R.J. (1996). Claims reserving and generalised additive models. *IME*, **19**, 31–43.

[Wand (2014)] Wand, M.P. (2014). **SemiPar**: Semiparametic Regression. *R package version 1.0-4.1.* URL http://CRAN.R-project.org/package=SemiPar.

[Wood (2003)] Wood, S.N. (2003). Thin plate regression splines. *J. Roy. Statist. Soc. B*, **65**(1), 95–114.

[Wood (2006)] Wood, S.N. (2006). Generalized Additive Models: An Introduction with R. *Chapman & Hall/CRC Press.*

[Yoshida and Naito (2012)] Yoshida, T. and Naito, K. (2012). Asymptotics for penalized additive B-Spline regression. *Journal of Japan Statistical Society*, **42**, 81–107.

[Yoshida and Naito (2014)] Yoshida, T. and Naito, K. (2014). Asymptotics for penalised splines in generalised additive models. *Journal of Nonparametric Statistics*, **26**(2), 269–89.

[Zhou et alt. (1998)] Zhou, S., Shen, X. and Wolfe, D.A. (1998). Local asymptotics for regression splines and confidence regions. *Ann. Statist.*, **26**(5), 1760–82.

Appendix A

The R-package GeDS

Package 'GeDS'

February 17, 2017

Type Package

Title GeDS

Version 0.1

Date 2017-17-02

Author Vladimir Kaishev <Vladimir.Kaishev.1@city.ac.uk>,
Dimitrina Dimitrova <D.Dimitrova@city.ac.uk>,
Andrea Lattuada <Lattuada.Andrea@spes.uniud.it> and
Richard Verrall <R.J.Verrall@city.ac.uk>

Maintainer Andrea Lattuada <Lattuada.Andrea@spes.uniud.it>

Description Implemetation of the Geometrically Designed Spline (GeDS) Regression.
GeDS Regression is a non parametric method with a geometrical interpretation
based on B-splines that authomatically selects number and position of the
knots and the order of the splines to be used. The package includes both
the original version and the generalized one that allow the distribution
of the response variable to be one of the Exponential Family.

License GPL-3

LazyData TRUE

Depends R (>= 3.2.2),
Rcpp (>= 0.12.1),
splines,
stats,
Matrix,
methods

LinkingTo Rcpp

RoxygenNote 5.0.1

R topics documented:

`BaFe2As2` *Barium-Ferrum-Arsenide powder diffraction data*

Description

This dataset collects the results of a neutron diffraction experiment on Barium-Ferrum-Arsenide powder carried out by Kimber et al. (2009). The neutron diffraction intensity was measured at several dispersion angles in order to fit the diffraction profile.

Usage

```
BaFe2As2
```

Format

A `data.frame` with 1151 cases and 2 variables:

X the dispersion angle.

Y the diffraction intensity.

Source

http://openaccess.city.ac.uk/

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2015). Geometrically designed, variable knot regression splines. Computational Statistics, [Peer Reviewed].

Kimber, S.A.J., Kreyssig, A., Zhang, Y.Z., Jeschke, H.O., Valenti, R., Yokaichiya, F., Colombier, E., Yan, J., Hansen, T.C., Chatterji, T., McQueeney, R.J., Canfield, P.C., Goldman, A.I. and Argyriou, D.N. (2009). Similarities between structural distortions under pressure and chemical doping in superconducting $BaFe_2As_2$. Nat Mater 8, 471–475.

Examples

```
## Not run:
data('BaFe2As2')
(Gmod2 <- GeDS(Y ~ f(X), data = BaFe2As2, beta = 0.6, phi = 0.99, q = 3))
plot(Gmod2)

## End(Not run)
```

`Bivariate1` *Simulated data for bivariate examples*

Description

`Bivariate1`, `Bivariate2` and `Bivariate3` contain simulated data on woch one can run examples for the bivariate `GeDS` function.

Usage

```
Bivariate1
```

Format

Lists with the following components:

x vector of x values.

y vector of y values.

z vector of response values.

Examples

```
## Not run:
data("Bivariate1")
ddd<-NGeDS(z~f(x,y),phi=.8,q=2, data=Bivariate1,show=T, beta=.3)
ddd<-NGeDS(z~f(x,y),phi=.95,q=2, data=Bivariate1,show=T, beta=.3)
plot(ddd)
ddd<-NGeDS(z~f(x,y),phi=.95,q=2,show=T, beta=.3)
plot(ddd)
data("Bivariate2")
ccc<-NGeDS(z~f(x,y),phi=.8,q=2, data=Bivariate2,show=T, beta=.3)
plot(ccc)
data("Bivariate3")
ccc<-NGeDS(z~f(x,y),phi=.99,q=2, data=Bivariate3,show=T, beta=.1)
plot(ccc)

## End(Not run)
```

`Bivariate2` *Simulated data for bivariate examples*

Description

`Bivariate1`, `Bivariate2` and `Bivariate3` contain simulated data on woch one can run examples for the bivariate `GeDS` function.

Usage

```
Bivariate2
```

Format

Lists with the following components:

- **x** vector of x values.
- **y** vector of y values.
- **z** vector of response values.

`Bivariate3` *Simulated data for bivariate examples*

Description

`Bivariate1`, `Bivariate2` and `Bivariate3` contain simulated data on woch one can run examples for the bivariate `GeDS` function.

Usage

```
Bivariate3
```

Format

Lists with the following components:

- **x** vector of x values.
- **y** vector of y values.
- **z** vector of response values.

coalMining *Coal Mining Disasters data*

Description

The dataset records the number of disasters in British coal mines from 1850 to 1962.

Usage

```
coalMining
```

Format

A `data.frame` with 112 entries, corresponding to the years from 1851 to 1962. Each entry has:

accidents number of severe accidents occurred.

years accident year.

References

Carlin, B.P.,Gelfand, A.E. and Smith, A.F.M. (1992). Hierarchical Bayesian analysis of changepoint problems. Applied Statistics, 41(2), 389–405

Eliers, P.H.C. and Marx, B.D. (1996). Flexible Smoothing with B-splines and Penalties. Statistical Science, 11(2), 89–121.

coef.GeDS *Coef method for GeDS objects*

Description

Method for the function `coef` that allows to extract the estimated coefficients from a fitted `GeDS-Class` object.

Usage

```
## S3 method for class 'GeDS'
coef(object, n = 3, onlySpline = TRUE, ...)
```

Arguments

`object`	the object from which coefficients should be extracted
`n`	the order of the sline to be considered
`onlySpline`	logical. Should only the B-spline coefficiens be extracted?
`...`	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

Details

This is a simple method for the function `coef`. As GeDS objects contain three different fits, it is possible to specify the order of the spline through the input `n`.

Value

A named vector in which each element represents a coefficient associated to one of the B-splines or to one of the terms linearly modeled.

See Also

`coef` for the standard definition; `NGeDS` for examples.

`Derive` *Derivative of GeDS objects*

Description

This function computes derivatives of splines fitted via GeDS regression.

Usage

```
Derive(object, x, order = 1, n = 3)
```

Arguments

`object`	the GeDS object containing the fitted splines.
`x`	numeric vector containing locations where it is desired to compute the derivative.
`order`	the order of integration desired. Note that it should be lower than `n`.
`n`	the order of the spline to be considered.

Details

The function is based on the the function `splineDesign` and computes the exact derivative of the fitted spline. It is based on the B-spline representation, hence it is not designed to work on a transformed scale and it computes the derivative only on the linear predictor scale in the generalized framework.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2))
Derive(Gmod, x = c(0,-1,1), order = 2, n = 4)
```

`deviance.GeDS` *Deviance method for GeDS objects*

Description

This method allows the user to extract the deviance from `GeDS-Class` objects.

Usage

```
## S3 method for class 'GeDS'
deviance(object, n = 3, ...)
```

Arguments

`object`	the object from which deviance should be extracted
`n`	numeric value indicating the order of the spline to be considered.
`...`	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

Details

This is a method for the function `deviance`. As GeDS objects contain three different fits, it is possible to specify the order of the spline through the input `n`.

Value

A numeric value corresponding to the model deviance.

See Also

`deviance` for the standard definition; `GGeDS` for examples.

`EWmortality` *Death counts in England and Wales*

Description

The dataset consists of aggregated informations about the mortality of the English and Welsh male population between 2000 and 2002.

Usage

```
EWmortality
```

Format

A data frame with 109 entries and 3 variables: `Age`, `Deaths` and `Exposure`.

`Exposure` is a mid-year estimate of the population.

f *Defining non parametric part in a GeDS formula.*

Description

Function to be used in the formula of a GeDS regression in order to identify which regressor(s) should be modeled non parametrically.

Usage

```
f(x, xx = NULL, ...)
```

Arguments

`x`	vector containing the first non parametric regressor.
`xx`	vector containing the second regressor in case of a bivariate verision of the algorithm.
`...`	further arguments. As only the univariate and bivariate versions have been implemented, specifying these arguments will return an error.

Note

This function is intended to be used only as part of the formula in a GeDS regression.

See Also

NGeDS; GGeDS.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
reg1 <- runif(500, min = -0.1, max = 0.1)
reg2 <- runif(500, min = -0.2, max = 0.2)
off <- runif(500, min = -1, max = 1)

formula <- newY ~ f(newX) + reg1 + reg2 + offset(off)

(Gmod <- NGeDS(formula, beta = 0.6, phi = 0.995, Xextr = c(-2,2)))
```

Fitters *Functions used to fit GeDS objects.*

Description

Fitter functions for GeDS Univariate regression.

Usage

```
UnivariateFitter(X, Y, Z = NULL, offset = rep(0, NROW(Y)),
  weights = rep(1, length(X)), beta = 0.5, phi = 0.5, min.intknots = 0,
  max.intknots = 300, q = 2, extr = range(X), show.iters = FALSE,
  tol = as.double(1e-12), stoptype = c("SR", "RD", "LR"))

GenUnivariateFitter(X, Y, Z = NULL, offset = rep(0, NROW(Y)),
  weights = rep(1, length(X)), family = gaussian(), beta = 0.5,
  phi = 0.5, min.intknots = 0, max.intknots = 300, q = 2,
  extr = range(X), show.iters = F, tol = as.double(1e-12),
  stoptype = c("SR", "RD", "LR"))
```

Arguments

`X`	a vector of length `n` containing the regressor to be modeled non-parametrically.
`Y`	a vector of length `n` containing the observations.
`Z`	a design matrix (possibly `NULL`) containing other regressors, to be modeled linearly.
`offset`	this can be used to specify an a priori known component to be included in the predictor during the fitting procedure.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector of length `n`.
`beta`	parameter for the GeDS fitting algorith. See documentation.
`phi`	parameter for the stop criterium. See documentation.
`min.intknots`	minimum number of internal knots required.
`max.intknots`	maximum number of internal knots required.
`q`	number of iterations considered for the stop criterium. See documentation.
`extr`	locations of the boundary knots.
`show.iters`	logical indicating whether to print or not step by step informations.
`tol`	tolerance to be used in the knot placement steps.
`stoptype`	a character string indicating the stopping rule to be considered. It should be one of `"SR"`, `"RD"` or `"LR"`, partial match allowed.
`family`	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details about family functions).

Details

Those functions are not intended to be directly used, they should be called through NGeDS and GGeDS.

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2015). Geometrically designed, variable knot regression splines. Computational Statistics.

See Also

NGeDS, GGeDS, GeDS-Class and SplineReg.

Examples

```
# a couple of examples similar to the ones
# presented in NGeDS and in GGeDS
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
(Gmod <- UnivariateFitter(newX, newY, beta = 0.6, phi = 0.995,
           extr = c(-2,2)))

set.seed(123)
newX <- sort(runif(N ,min = -2, max = 2))
means <- exp(f_1(newX))
newY <- rpois(N,means)
(Gmod2 <- GenUnivariateFitter(newX, newY, beta = 0.2,
            phi = 0.995, family = poisson(), extr = c(-2,2)))
plot(newX,log(newY),xlab = "x", ylab = expression(f[1](x)))
lines(Gmod2,n = 3, col = "red")
lines(Gmod2,n = 4, col = "blue", lty = 2)
legend("topleft", c("Quadratic","Cubic"),
     col = c("red","blue"), lty = c(1,2))
```

GeDS-Class *GeDS Class*

Description

A fitted GeDS object returned by functions `NGeDS` or `GGeDS` inheriting from class "`GeDS`". Method functions `coef`, `knots`, `print`, `predict`, `plot`, `lines` and `summary` are available.

Slots

`Type` The type of the regression performed. One of "`LM - Univ`", "`LM - Biv`" or "`GLM - Univ`".

`Linear.Knots` The locations of the knots selected for the second order spline

`Quadratic.Knots` The locations of the knots selected for the third order spline

`Cubic.knots` The locations of the knots selected for the fourth order spline

`Dev.Linear` Deviance of the fitted second order spline

`Dev.Quadratic` Deviance of the fitted third order spline

`Dev.Cubic` Deviance of the fitted second fourth spline

`Linear` List containing the results form a SplineReg function used to fit a second order spline

`Quadratic` List containing the results form a SplineReg function used to fit a third order spline

`Cubic` List containing the results form a SplineReg function used to fit a fourth order spline

`Stored` Matrix containing the knots selected at each step of stage A.

`Args` List containing the values passed to the Fitters function.

`Call` The call to the Fitters functions.

`Nintknots` number of internal knots selected in Stage A of the algorithm.

`iters` Number of stage A iterations.

`Guesses` Starting values for the coefficients used at each iteration of stage A in order to fit the spline coefficients. As the starting values are used only in the IRLS procedure, this slot is not empty only if the object is created via `GGeDS` function.

`Coefficients` Matrix containing the coefficients fitted at each step of stage A.

`deviance` Vector containing the deviance for each single IRLS iteration for stage A.

`iter` Vector containing the progressive number of IRLS iterations for stage A.

`stopinfo` List of values.

`Formula` The model formula.

`extcall` `call` to the `NGeDS` or `GGeDS` functions.

`terms` `terms` object containing information on the model frame.

`GGeDS` *Generalized Geometrically Designed Spline regression*

Description

`GGeDS` runs the generalized version of the Geometrically Designed Spline regression.

Usage

```
GGeDS(formula, data, family = gaussian(), weights, beta, phi = 0.99,
  min.intknots = 0, max.intknots = 300, q = 2, Xextr = NULL,
  show.iters = FALSE, stoptype = "SR")
```

Arguments

`formula`	a description of the structure of the model to be fitted. See `formula` for details.
`data`	an optional data frame, list or environment containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `GGeDS` is called.
`family`	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions).

`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector of the same length as the other input variables.
`beta`	parameter for the GeDS fitting algorith controlling the knot placement procedure. See documentation.
`phi`	treshold for the stop criterium. See documentation.
`min.intknots`	minimum number of internal knots required.
`max.intknots`	maximum number of internal knots required.
`q`	number of iterations considered for the stop criterium. See details.
`Xextr`	boundary knots of B-splines.
`show.iters`	logical indicating whether to print or not step by step informations.
`stoptype`	a character string indicating the stopping rule to be considered. It should be one of `"SR"`, `"RD"` or `"LR"`, partial match allowed.

Details

`GGeDS` is one of the main functions of the package and it can be used to perform a GeDS regression under the assumption that the dstribution of the response variable belongs to the exponential family. A detailed description of the underlying algorithm can be found in Kaishev et al. (2016).

The argument `formula` should specify a semiparametric part and (optionally) a parametric part. The semiparametric part should be specified through the function f. Non specification of the term to be modeled via spline regression will return an error. Following the same scheme adopted by other R functions, it is possible to specify one or more offset variables, i.e. known terms with a fixed coefficient equal to 1. These terms should be identified via the function `offset`.

Three possible stopping rules for the knot selection are implemented. Setting `stoptype` equal to `"RD"`, that stands for *Ratio of Deviances*, the rule considered is a generalization of the one described in Kaishev et al. (2015) that was based the sum of squared residuals. Setting `"SR"`, i.e. *Smoothed Ratio*, it is adopted the smoothed version of the RD rule, described in Kaishev et al. (2016). The `"LR"` (*Likelihood Ratio*) is the rule based on the difference of deviances rather than on the ratio and hence it performs a log likelihood ratio test at each iteration in order to decide whether the knot selection procedure should stop or continue.

Note that with the LR stopping rule the threshold `phi` has the opposite behaviour compared to the other cases, i.e. the lower `phi` will select less knots. Further details on the stopping rules can be found in Kaishev et al. (2016).

Value

a GeDS-Class object containing several items that resume the main details of the regression. See GeDS-Class for details. Some S3 methods are available in order to make these objects tractable.

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2015). Geometrically designed, variable knot regression splines. Computational Statistics.

See Also

NGeDS; GeDS-Class; S3 methods such as coef.GeDS, deviance.GeDS, knots.GeDS, print.GeDS and predict.GeDS; Integrate and Derive; PPolyRep.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- exp(f_1(newX))
newY <- rpois(N,means)
(Gmod <- GGeDS(newY ~ f(newX), beta = 0.2, phi = 0.995, family = poisson(),
                Xextr = c(-2,2)))
plot(newX,log(newY),xlab = "x", ylab = expression(f[1](x)))
lines(Gmod,n=3,col="red")
lines(Gmod,n=4,col="blue",lty=2)
legend("topleft",c("Quadratic","Cubic"),col=c("red","blue"),lty=c(1,2))

predict(Gmod, n = 3, newdata=data.frame(newX = 0))
predict(Gmod, n = 3, newdata=data.frame(newX = 0), type = "link")

knots(Gmod)
coef(Gmod)
deviance(Gmod)

knots(Gmod, n = 4)
coef(Gmod, n = 4)
deviance(Gmod, n = 4)


##########################################
# A real data example

data("coalMining")
(Gmod2 <- GGeDS(formula = accidents ~ f(years), beta = 0.1, phi = 0.98,
                 family = poisson(), data = coalMining))
(Gmod3 <- GGeDS(formula = accidents ~ f(years), beta = 0.1, phi = 0.985,
                 family = poisson(), data = coalMining))
plot(coalMining$years,coalMining$accidents,ty="h", xlab="Years",ylab="Accidents")
lines(Gmod2, tr = exp, n = 4, col = "red")
lines(Gmod3, tr = exp, n = 4, col = "blue", lty = 2)
legend("topright",c("phi = 0.98","phi = 0.985"),col=c("red","blue"),lty=c(1,2))


## Not run:
##########################################
# The same regression in the example of GeDS
# but with Gamma response

data('BaFe2As2')
(Gmod4 <- GGeDS(Y ~ f(X), data = BaFe2As2, beta = 0.6, phi = 0.995, q = 3,
                 family = Gamma(log), stoptype = "RD"))
plot(Gmod4)

(Gmod5 <- GGeDS(Y ~ f(X), data = BaFe2As2, beta = 0.1, phi = 0.995, q = 3,
                 family = poisson(), stoptype = "SR"))
plot(Gmod5)

## End(Not run)
```

```
##########################################
# Life tables

data(EWmortality)
attach(EWmortality)
(M1 <- GGeDS(formula = Deaths ~ f(Age) + offset(log(Exposure)),
              family = poisson(), phi = 0.99, beta = 0.1, q = 3,
              stoptype = "LR"))

Exposure_init <- Exposure + 0.5 * Deaths
Rate <- Deaths / Exposure_init
(M2 <- GGeDS(formula = Rate ~ f(Age), weights = Exposure_init,
              family = quasibinomial(), phi = 0.99, beta = 0.1,
              q = 3, stoptype = "LR"))


op <- par(mfrow=c(2,2))
plot(Age, Deaths/Exposure, ylab = expression(mu[x]), xlab = "Age")
lines(M1, n = 3, tr = exp, lwd = 1, col = "red")
plot(Age, Rate, ylab = expression(q[x]), xlab = "Age")
lines(M2, n = 3, tr = quasibinomial()$linkinv, lwd = 1, col = "red")
plot(Age, log(Deaths/Exposure), ylab = expression(log(mu[x])), xlab = "Age")
lines(M1, n = 3, lwd = 1, col = "red")
plot(Age, quasibinomial()$linkfun(Rate), ylab = expression(logit(q[x])), xlab = "Age")
lines(M2, n = 3, lwd = 1, col = "red")
par(op)
```

`Integrate` *Integrate GeDS objects*

Description

This function computes integrals of splines fitted vie GeDS regression.

Usage

```
Integrate(object, to, from, n = 3)
```

Arguments

`object`	the GeDS object containing the fitted splines
`to`	numeric vector containing the upper limits of integration
`from`	optional numeric vector containing the lower limits of integration. Default to the lower boundary knot of the spline.
`n`	the order of the spline to be considered

Details

The function is based on the formula at page 128 of De Boor (2001), computing the exact integral of the fitted function. It is based on the B-spline representation, hence it is not designed to work on a transformed scale and it computes the integral on the predictor scale even in the generalized framework.

References

De Boor, C. (2001). A Practical Guide to Splines (Revised Edition). Springer, New York.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2))
Integrate(Gmod, to = c(-1,1), from = 1,n=3)
Integrate(Gmod, to = c(-1,1), from = c(1,-1),n=3)
## Not run:
Integrate(Gmod, to = 1, from = c(1,-1),n=3)

## End(Not run)
```

`IRLSfit` *IRLS Algorithm*

Description

This function performs IRLS algorithm in an efficient way considering the usage required in the **GeDS** package.

Usage

```
IRLSfit(x, y, weights = rep(1, nobs), mustart = NULL, offset = rep(0,
  nobs), family = gaussian(), control = list(), etastart = NULL)
```

Arguments

`x`	a design matrix of dimension `n * p`
`y`	a vector of observations of length `n`.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector of length `n`.
`mustart`	starting values for the vector of means.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting.
`family`	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions).
`control`	a list of parameters for controlling the fitting process. For glm.fit this is passed to `glm.control`.

Details

This function is a slightly modified version of the `glm.fit` in package `stats` to which we refer for further details. Basically the difference in the inputs is that it admits starting values only for the μ_is.

In the output it produces some more slots. We remark that the slots `weights`, `res2` and `z` contain values computed *after* the last iteration, i.e. they are based on the estimated coefficients that will be returned.

The source code contains some commented lines that produce step by step plots within IRLS iterations. We left them as comments as they are time consuming, but experienced R users may find them useful.

Value

A list containing:

`coefficients`	a named vector of coefficients
`residuals`	the working residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are NA.
`res2`	the working residuals after the final iteration. They are used in the knot placement procedure in GeDS algorithm
`fitted.values`	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
`rank`	the numeric rank of the fitted linear model.
`family`	the `family` object used.
`linear.predictors`	the linear fit on link scale.
`deviance`	a vector of the deviances obtained at each step.
`lastdeviance`	the deviance at the last step.
`aic`	A version of Akaike's Information Criterion. See `glm`.
`null.deviance`	The deviance for the null model, comparable with deviance. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
`iter`	the number of iterations of IRLS used.
`weights`	the working weights after the last iteration, that is the weights in the final iteration of the IRLS fit.
`prior.weights`	the weights initially supplied, a vector of 1s if none were.
`df.residual`	the residual degrees of freedom.
`df.null`	the residual degrees of freedom for the null model.
`y`	the y vector used.
`z`	the pseudo-data computed after the last iteration.
`converged`	logical. Was the IRLS algorithm judged to have converged?
`boundary`	logical. Is the fitted value on the boundary of the attainable values?

In addition, non-empty fits will have components `qr`, `R` and `effects` relating to the final weighted linear fit.

See Also

`glm.fit`

knots.GeDS — *Knots method for GeDS objects*

Description

Knots method for GeDS objects

Usage

```
## S3 method for class 'GeDS'
knots(Fn, n = 3, options = c("all", "internal"), ...)
```

Arguments

`Fn`	the object from where the knots should be extracted.
`n`	the spline order.
`options`	a character string specifying whether "`all`" knots should be extracted (the default) or only the "`internal`" ones.
`...`	potentially further arguments (required by the generic). only the "`internal`" ones.

Details

This is a method for the function `knots` in the **stats** package.

The input parameter `n` can be used to specify the order of the spline fit from which to extract the knot locations.

Value

A named vector in which each element represents a knot location of the B-splines on which the fit is based.

See Also

`knots` for the standard definition; `NGeDS` and `GGeDS` for examples.

lines,GeDS-method — *Lines function method for GeDS-Class objects*

Description

Lines function method for GeDS-Class objects

Usage

```
## S4 method for signature 'GeDS'
lines(x, n = 3, transform = function(x) x,
  onlySpline = TRUE, data = data.frame(), ...)
```

Arguments

x	a GeDS-Class object.
n	the order of the spline that should be drawn.
transform	a function that can be used to transform the Y axis scale. Tipically it can be the inverse of the link function if the plot is on the scale of the response variable.
onlySpline	logical. Should only the spline be drawn of also the cefficiens be extracted?
data	an optional data frame where if onlySpline is set to FALSE the values of the linearly modeled or known terms are seeked. If left empty the values are extracted from the object.
...	further arguments to be passed to the default lines function.

Details

This method can be used to add a line in an active plot corresponding to the GeDS fit. As in other methods provided for GeDS-Class objects, the user is allowed to specify the order of the spline to be fitted via the argument n.

See Also

NGeDS and GGeDS; lines.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
(Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2)))
plot(Gmod)
lines(Gmod, n = 2, col = "green", lwd = 2, lty = 3)
```

NGeDS *Geometrically designed spline regression*

Description

NGeDS runs the linear version of the Geometrically Designed Spline regression.

Usage

```
NGeDS(formula, data, weights, beta = 0.5, phi = 0.99, min.intknots = 0,
  max.intknots = 500, q = 2, Xextr = NULL, Yextr = NULL,
  show.iters = FALSE, stoptype = "RD")
```

Arguments

`formula`	a description of the structure of the model to be fitted. See `formula` for details.
`data`	an optional data frame, list or environment containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `NGeDS` is called.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector of the same length as the other input variables.
`beta`	parameter for the GeDS fitting algorith controlling the knot placement procedure. See documentation.
`phi`	treshold for the stop criterium. See documentation.
`min.intknots`	minimum number of internal knots required.
`max.intknots`	maximum number of internal knots required.
`q`	number of iterations considered for the stop criterium. See documentation.
`Xextr`	boundary knots in the first direction of the independent variable.
`Yextr`	boundary knots in the second direction of the independent variable.
`show.iters`	logical indicating whether to print or not step by step informations.
`stoptype`	a character string indicating the stopping rule to be considered. It should be one of `"SR"`, `"RD"` or `"LR"`, partial match allowed.

Details

`NGeDS` is the main function of the package, used to perform a GeDS regression, as described in Kaishev et al. (2015). This performs both the univariate and the bivariate version of the algorithm, the latter based on the tensor product of the splines.

The argument `formula` should specify a semiparametric part and (optionally) a parametric part. The semiparametric part should be specified through the function f. Non specification of the term to be modeled via spline regression will return an error. Following the same scheme adopted by other R functions, it is possible to specify one or more offset variables, i.e. known terms with a fixed coefficient equal to 1. These terms should be identified via the function `offset`.

Three possible stopping rules for the knot selection are implemented. Setting `stoptype` equal to `"RD"`, that stands for *Ratio of Deviances*, the rule considered is a generalization of the one described in Kaishev et al. (2015) that was based the sum of squared residuals (this rule is the default one for this function). Setting `"SR"`, i.e. *Smoothed Ratio*, it is adopted the smoothed version of the RD rule, described in Kaishev et al. (2016). The `"LR"` (*Likelihood Ratio*) is the rule based on the difference of deviances rather than on the ratio and hence it performs a log likelihood ratio test at each iteration in order to decide whether the knot selection procedure should stop or continue.

Note that with the LR stopping rule the threshold `phi` has the opposite behaviour compared to the other cases, i.e. the lower `phi` will select less knots. Further details on the stopping rules can be found in Kaishev et al. (2016).

Value

a GeDS-Class object containing several items that resume the main details of the regression. See GeDS-Class for details.

A `GeDS-Class` object. See GeDS-Class for details.

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2015). Geometrically designed, variable knot regression splines. Computational Statistics.

See Also

NGeDS; GeDS-Class; S3 methods such as coef.GeDS, deviance.GeDS, knots.GeDS, print.GeDS and predict.GeDS; Integrate and Derive; PPolyRep.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
(Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2)))
coef(Gmod, n = 3)
knots(Gmod, n = 3)
knots(Gmod, n = 3, options = "internal")
deviance(Gmod, n = 3)



## Not run:
data('BaFe2As2')
(Gmod2 <- NGeDS(Y ~ f(X), data = BaFe2As2, beta = 0.6, phi = 0.99, q = 3))
plot(Gmod2)

## End(Not run)
```

`plot,GeDS-method` *Plot function method for GeDS-Class objects*

Description

Plot function method for GeDS-Class objects

Usage

```
## S4 method for signature 'GeDS'
plot(x, which, DEV = FALSE, ask = FALSE, main,
  legend.pos = "topright", new.window = FALSE, wait = 0.5, n = 3,
  type = c("Polygon", "CI", "none"), ...)
```

Arguments

`x`	an object of class GeDS-Class.
`which`	a vector specifying which iterations of stage A should be plotted. It has to be a subset of `1:nrow(x$stored)`. See details.

DEV	logical, specifying whether a plot of the Deviance at each step should be produced or not.
ask	logical, specifying whether the user should be prompted before changing the plot page.
main	optional character string to be used as title.
legend.pos	the position of the legend. See legend for details.
new.window	logical. Should the plot be shown in a new window?
wait	time, in seconds, the system should wait before plotting a new page. Ignored if ask is set to TRUE.
n	degree of the spline regression to be plot. Default to 3 (i.e. quadratic).
type	character string specifying type of plot required. Should be set either to "Polygon" if the user wants to get also the control polygon, "CI" if the confidence intervals should be drawn or "none" if it is desired only the line corresponding to the fitted values. Applies only when plotting a linear univariate spline regression.
...	further arguments to be passed to the plot function.

Details

This method plots the fits contained in the GeDS-Class objects.

As GeDS regression algorithm is composed by two main stages and the first one performs the selection of the number of knots and their placement, it may e interesting to visually inspect how this procedure sequentially selects the knots. Argument which allows the user to do this inspection. Specifying a single value it will appear one single plot, while specifying a vector, as many pages as elements will be sequentially plotted. ask and wait arguments can help the user to manage these pages. Note that, in order to ensure stability, if the object was produced by the function GGeDS, intermediate fits of stage A can be plotted only setting n equal to 2.

The confidence interval obtained, available only for GeDS-Class objects fitted through NGeDS function, are the ones produced by the SplineReg_LM.

See Also

NGeDS and GGeDS; plot.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
(Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2)))

plot(Gmod)
plot(Gmod, which=10)
plot(Gmod, which=1:16)

##############################################

set.seed(123)
N <- 500
```

```
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- exp(f_1(newX))
newY <- rpois(N,means)
(Gmod2 <- GGeDS(newY ~ f(newX), beta = 0.2, phi = 0.995, family = poisson(),
                Xextr = c(-2,2)))

plot(Gmod2, n = 2)
plot(Gmod2, which=10, n = 2)
plot(Gmod2, which=1:16, n = 2)
## Not run:
plot(Gmod2, which=1:16, n = 2, ask = T)

## End(Not run)
```

`PPolyRep` *Piecewise Polynomial Splines Representation*

Description

This function converts GeDS objects into a spline object form package `splines` inheriting from classes "`spline`" and "`polySpline`"

Usage

```
PPolyRep(object, n)
```

Arguments

`object`	`GeDS-Class` object it is desired to be converted.
`n`	the order of the spline fit that should be considered.

Details

The function basically wraps the function `polySpline` in order to let it accept GeDS objects as the input. Hence, for a selected order, it computes the piecewise polynomial representation of the fitted splines.

Value

An object that inherits from classes "`spline`" and "`polySpline`". It is a list whose arguments are:

`knots`	a vector of the same length as the number of knots k, containing the (unique) knots.
`coefficients`	a $k \times n$ matrix containing the coefficients of the polynomials.

Examples

```
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
newX <- sort(runif(N ,min = -2, max = 2))
means <- f_1(newX)
newY <- rnorm(N, means, sd = 0.1)
Gmod <- NGeDS(newY ~ f(newX), beta = 0.6, phi=.995, Xextr = c(-2,2))

Polymod <- PPolyRep(Gmod, 4)
require(splines)
class(Polymod)
splineKnots(Polymod)
knots(Gmod,n = 4)
plot(Polymod)


# a plot showing the PP representation
# based on the same example
knt <- splineKnots(Polymod)
coeffs <- coef(Polymod)

plot(Gmod, n = 4, legend = FALSE, main = "Cubic Curves")
cols <- sample(heat.colors(length(knt)), length(knt))
for(i in 1:(length(knt))){
  curve(coeffs[i,1]+coeffs[i,2]*(x - knt[i])+
          coeffs[i,3]*(x - knt[i])^2+
        coeffs[i,4]*(x - knt[i])^3,
        add = TRUE, col = cols[i])
  abline(v = knt[i])
}
```

`predict.GeDS` *Predict method for GeDS objects*

Description

This is a user friendly method to compute predictions from GeDS objects.

Usage

```
## S3 method for class 'GeDS'
predict(object, newdata, type = c("response", "link", "terms"),
  n = 3, ...)
```

Arguments

`object`	the object for which the computation of the predicted values is desired.

newdata	an optional data.frame in which to look for variables with which to predict. If omitted, the input values are used.
type	the type of prediction required. The default ("response") is on the scale of the response variable. The alternative "link" is on the linear predictor scale.
n	the spline order.
...	potentially further arguments (required by the generic function definition).

Details

This is a method for the function predict that allows the user to handle GeDS-Class object.

In analogy with the function predict.glm in **stats** package, the user can specify the scale on which the predictions should be computed through the argument type. If the predictions are desired on the scale of the response variable, the user should set type = "response", that is the default . While if one wants the predictions on the predictor scale, it is necessary to set type = "link". The user can also specify type = "terms" if the user aims to inspect the effect of each single variable on the predictor.

As GeDS objects contain three different fits, it is possible to specify the order of the spline to be considered through the input n.

Value

A numeric vector corresponding to the predicted values (if type = "link" or type = "response"). If type = "terms" a numeric matrix with a column per term.

See Also

predict for the standard definition; GGeDS for examples.

---

print.GeDS *Print method for GeDS objects*

Description

This is the print method for GeDS objects.

Usage

```
## S3 method for class 'GeDS'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	the object for which the prediction is desired.
digits	number of digits to be printed on screen
...	potentially further arguments (required by the generic).

Details

This method allows to print on the screen just some basic information such as the function call, the number of internal knots selected by the GeDS algorithm and the deviances for the three fitted splines.

Value

This function returns (invisibly) the same input object, but adding the three arguments:

`Nknots`	the number of internal knots
`Deviances`	the deviances for the three splines
`Call`	the function call

See Also

`predict` for the standard definition.

`splineDesign2` *Design Matrix for B-splines*

Description

Evaluate the design matrix for the B-splines defined by `knots` at the values in `x`.

Usage

```
splineDesign2(knots, x, ord = 4, derivs = rep(0, length(x)),
  outer.ok = FALSE, sparse = FALSE)
```

Arguments

`knots`	a numeric vector of knot positions with non-decreasing values.
`x`	a numeric vector of values at which to evaluate the B-spline functions or derivatives. Unless `outer.ok` is true, the values in `x` must be between `knots[ord]` and `knots[ length(knots) + 1 - ord ]`
`ord`	a positive integer giving the order of the spline function. This is the number of coefficients in each piecewise polynomial segment, thus a cubic spline has order 4. Defaults to 4.
`derivs`	an integer vector of the same length as `x` and with values between 0 and `ord - 1`. The derivative of the given order is evaluated at the `x` positions. Defaults to a vector of zeroes of the same length as `x`.
`outer.ok`	logical indicating if `x` should be allowed outside the *inner* knots, see the `x` argument.
`sparse`	logical indicating if the result should inherit from class `sparseMatrix` (package **Matrix**).

Value

A matrix with `length( x )` rows and `length( knots ) - ord` columns. The `i`'th row of the matrix contains the coefficients of the B-splines (or the indicated derivative of the B-splines) defined by the `knot` vector and evaluated at the `i`'th value of `x`. Each B-spline is defined by a set of `ord` successive knots so the total number of B-splines is `length(knots)-ord`.

Note

This function is basically the same as the function `splineDesign` in package **spline** but this is based on the `C` function `spline_basis2`. This avoids getting `NaN`s the resulting matrix in some particular cases.

See Also

`splineDesign`

Examples

```
X <- c(0,1,2)
kn <- c(0,0,0,2,2,2)
n <- 2
require(splines)
splineDesign(kn, X, n, derivs = rep(0,3))
# Here one gets NaNs
splineDesign2(kn, X, n, derivs = rep(0,3))
# Here it doesn't, even if there are two zero-columns
```

`SplineReg` *Univariate B-Spline Regression functions.*

Description

Functions used to compute regressions at various steps of the GeDS algorithm.

Usage

```
SplineReg_LM(X, Y, Z = NULL, offset = rep(0, NROW(Y)), weights = rep(1,
  length(X)), InterKnots, n, extr = range(X), prob = 0.95)

SplineReg_GLM(X, Y, Z, offset = rep(0, nobs), weights = rep(1, length(X)),
  InterKnots, n, extr = range(X), family, inits = NULL, mustart,
  etastart = NULL)
```

Arguments

`X`	a vector of length `n` containing the regressor to be modeled non-parametrically.
`Y`	a vector of length `n` containing the observations.
`Z`	a design matrix (possibly `NULL`) containing the other regressors.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting.
`weights`	an optional vector of ’prior weights’ to be used in the fitting process. Should be `NULL` or a numeric vector of length `n`.
`InterKnots`	a vector of internal knot locations.
`n`	order of the spline to be used.
`extr`	boundary knots to be used.

`prob`	the confidence level to be used for the confidence bounds in the `SplineReg_LM` fit.
`family`	a description of the error distribution and link function to be used in the model. See family for details of family functions.
`inits`	a vector of length `length(InterKnots) + n + NCOL(Z)` starting values for the coefficients to be used in the IRLS algorithm.
`mustart`	starting values for the vector of means. Must be a vector of length `length(X)`.

Details

These functions perform unpenalized spline regression, given the order, the set of knots and the family of the distribution of the response variable.

`SplineReg_LM` appears in Stage B of the GeDS algorithm, while stage A is performed via an unexported (faster) implementation. This function basically performs least squares regression and computes some useful tools such as the confidence intervals and the Control Polygon.

`SplineReg_GLM` is intesively used in Stage A of the GeDS algorithm and in order to make it as fast as possible input data validation is mild. Hence the user should be careful when using it. The `"Residuals"` in the output of this function are similar to the so called "working residuals" in the `glm` function. They are the residuals r_i used in the knot placement procedure, i.e.

$$r_i = (y_i - \hat{\mu}_i)\frac{d\mu_i}{d\eta_i},$$

but they consider the very last fitted $\hat{\mu}_i$s.

Value

A list containing:

`Theta`	the fitted coefficients.
`Predicted`	the predicted values.
`Residuals`	the residuals to be used in the knot placement procedure. See details.
`RSS`	the deviance.
`CI`	a list containing the confidence bands (computed only with the `SplineReg_LM`).
`Basis`	the design matrix, including the B-splines and `Z` columns.
`Polygon`	a list containing vertices and coefficients of the Control Polygon.
`deviance`	a vector containing the deviance computed at each IRLS step (computed only with the `SplineReg_GLM`).

The slot `temporary` of the output is the result of the function `lm` if `SplineReg_LM` is used. If it is used `SplineReg_GLM`, `temporary` is the output of the function `IRLSfit`, which is similar to the . `glm.fit` output, but with some slight differences.

See Also

NGeDS, GGeDS, Fitters, IRLSfit, lm and glm.fit.