Journal of Statistical Software

April 2017, Volume 77, Issue 9. *doi: 10.18637/jss.v077.i09*

Iterative Bias Reduction Multivariate Smoothing in R: The ibr Package

Pierre-André Cornillon
IRMAR
Université Rennes 2

Nicolas Hengartner
Los Alamos
National Laboratory

Eric Matzner-Løber
IRMAR
Université Rennes 2

Abstract

In multivariate nonparametric analysis curse of dimensionality forces one to use large smoothing parameters. This leads to a biased smoother. Instead of focusing on optimally selecting the smoothing parameter, we fix it to some reasonably large value to ensure an over-smoothing of the data. The resulting base smoother has a small variance but a substantial bias. In this paper, we propose an R package named **ibr** to iteratively correct the initial bias of the (base) estimator by an estimate of the bias obtained by smoothing the residuals. After a brief description of iterated bias reduction smoothers, we examine the base smoothers implemented in the package: Nadaraya-Watson kernel smoothers, Duchon splines smoothers and their low rank counterparts. Then, we explain the stopping rules available in the package and their implementation. Finally we illustrate the package on two examples: a toy example in $\mathbb{R}^2$ and the original Los Angeles ozone dataset.

Keywords: multivariate smoothing, L_2 boosting, thin-plate splines, kernel regression, R.

1. Introduction

Regression is a fundamental data analysis tool which relates a variable $Y \in \mathbb{R}$ to a function of d explanatory variables. Classical linear regression is the simplest example of this: The function is chosen to be affine. More generally, we can let the data help determine the general form of the relationship by using one of the numerous nonparametric regression estimators, such as wavelets, kernel smoothers, and splines smoothers (Buja, Hastie, and Tibshirani 1989; Cleveland and Devlin 1988; Eubank 1988; Fan and Gijbels 1996; Antoniadis and Oppenheim 1995; Simonoff 1996). Such methods are implemented as R (R Core Team 2017) functions found in numerous contributed packages. For instance the package **wavethresh** (Nason 2016) implements a wavelet based smoother, the package **lokern** (Herrmann and Maechler 2016) provides a kernel smoother and the function `smooth.spline` calculates a cubic spline smoother. When the number of dependent variables d is greater than 3 or 4, fully nonparametric regression suffers from the curse of dimensionality, even for moderate sample sizes (say n being equal to a few hundred). As a result, application of fully nonparametric methods is discouraged in dimensions four and higher. Instead, the statistical literature encourages using constrained regression models (additive models, Hastie and Tibshirani 1990; single and multiple index models and projection pursuit models) to estimate useful approximations of the conditional expectation. The latter methods are provided to the R community in the contributed package **mgcv** (Wood 2017) for additive modeling, function `ppr` in base R for projection pursuit and package **mda** (Hastie, Tibshirani, Leisch, Hornik, and Ripley 2016) for MARS.

Originating from the machine learning community, the *boosting* algorithm is also another tool for nonparametric regression (see Friedman 2001, and references therein). This fairly recent and very popular method has numerous variations, such as adaboost (the original method), logitboost for classification, and L_2 boosting for regression. The interesting feature is that it provides a framework for combining various weak learners (nonparametric smoothers) into a smoother that is better than any single smoother that it is composed of. Packages for L_2 boosting are already available in R: For instance the package **mboost** (Hothorn, Bühlmann, Kneib, Schmid, and Hofner 2016) allows for L_2 boosting for regression problems as well as logistic boosting for classification. For multivariate regression, the L_2 boosting algorithm has been applied to component-wise additive modeling with classical smoother such as smoothing splines (see Bühlmann and Hothorn 2007).

Iterative bias reduction is another tool for nonparametric regression, closely related to boosting. The basic idea of the bias correction scheme is to estimate (and correct for) the bias of a pilot smoother. These steps of estimation and correction can be done several times leading to iterative bias reduction. This idea goes back to the concept of *twicing* introduced by Tukey (1977). The idea of iterating the bias correction was central to *adaptive bagging* of the boosting-like algorithm of Breiman (1999). More details about statistical properties in univariate or multivariate smoothers can be found in Bühlmann and Yu (2003) or Cornillon, Hengartner, and Matzner-Løber (2014). Linking the L_2 boosting algorithm to an iterative bias correction scheme provides a statistical interpretation of the L_2 boosting algorithm. This interpretation was alluded to in Ridgeway (2000)'s discussion of Friedman, Hastie, and Tibshirani (2000) on the statistical interpretation of boosting and developed in Bühlmann and Hothorn (2007) for the univariate framework and in Cornillon *et al.* (2014) for the multivariate framework.

This paper focuses on the computational implementation in R of the iterated bias correction procedure for fully multivariate regression smoothers. We start in Section 2 by briefly presenting the concept of iterative bias reduction and recalling its connection to L_2 boosting. The details of our numerical implementation and a review of available options in our R package **ibr** are given in Section 3. Package **ibr** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=ibr` (Cornillon, Hengartner, and Matzner-Lober 2017). Section 4 is devoted to examples.

2. Iterative bias reduction smoothers

2.1. Method

Suppose that the pairs $(X_i, Y_i) \in \mathbb{R}^d \times \mathbb{R}$ are related through the nonparametric regression model

$$Y_i \quad = \quad m(X_i) + \varepsilon_i, \quad i = 1, \dots, n, \tag{1}$$

where $m(\cdot)$ is an unknown smooth function, and the disturbances ε_i are independent mean zero and variance σ^2 random variables that are independent of all the covariates. It is helpful to rewrite Equation 1 in vector form by setting $Y = (Y_1, \dots, Y_n)^\top$, $m = (m(X_1), \dots, m(X_n))^\top$ and $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)^\top$, to get

$$Y \quad = \quad m + \varepsilon. \tag{2}$$

Linear smoothers estimate the regression function m evaluated at the covariates by linear combinations of the responses that can be compactly written as

$$\widehat{m}_1 = S_\lambda Y, \tag{3}$$

where S_λ is an $n \times n$ smoothing matrix with smoothing parameter λ. By slight abuse of language, we will sometimes refer to the vector of fitted values $\widehat{m} = \widehat{Y} = (\widehat{Y}_1, \dots, \widehat{Y}_n)^\top$ as the smooth of Y. Typical smoothers (see for instance Hastie, Tibshirani, and Friedman 2001) include bin smoothers, spline based smoothers (regression splines, smoothing splines, and thin-plate splines), kernel based smoothers (Nadaraya-Watson kernels and local polynomials smoothers), and series based smoothers (Fourier smoothers and wavelet smoothers). In this paper, we focus only on two common types of smoothers: Nadaraya-Watson kernels (where λ is the bandwidth) and Duchon splines of order (m, s) (where λ is the penalty parameter). Extensions to other smoothers can easily be achieved by suitably modifying our theoretical results and package.

The linear smoother (3) has bias

$$B(\widehat{m}_1) = \mathsf{E}[\widehat{m}_1 | X] - m = (S_\lambda - I)m$$

and variance

$$\mathsf{VAR}(\widehat{m}_1 | X) = \left(S_\lambda S_\lambda^\top \right) \sigma^2.$$

To estimate the bias, observe that the residuals $R_1 = Y - \widehat{m}_1 = (I - S_\lambda)Y$ have expected value $\mathsf{E}[R_1|X] = m - \mathsf{E}[\widehat{m}_1|X] = (I - S_\lambda)m = -B(\widehat{m}_1)$. This suggests estimating the bias by smoothing the negative residuals

$$\widehat{b}_1 := -S_\lambda R_1 = -S_\lambda (I - S_\lambda) Y.$$

Recall that, in multivariate nonparametric analysis, curse of dimensionality forces one to use large smoothing parameters λ. This leads to a very biased base smoother S_λ. Thus the bias correction in multivariate nonparametric analysis arises as a natural tool to correct the classical smoother S_λ. If λ is large, not all the bias is usually removed after the first correction.

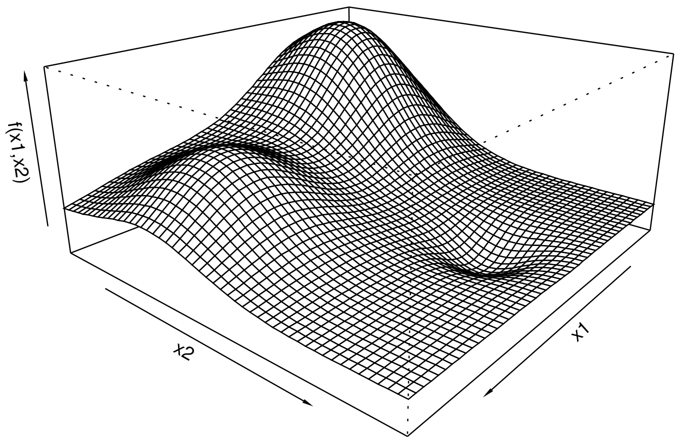


Figure 1: True bivariate regression function $m(x_1, x_2)$ (5) on the unit square $[0,1] \times [0,1]$ used in our numerical examples.

To remove the remaining bias, iteration of the bias reduction step have to be performed. For instance the $k-1$th bias reduction step produces the linear smoother at iteration k:

$$\begin{aligned} \widehat{m}_k &= S_\lambda Y + S_\lambda(I - S_\lambda)Y + \cdots + S_\lambda(I - S_\lambda)^{k-1}Y \\ &= (I - (I - S_\lambda)^k)Y. \end{aligned} \tag{4}$$

When $d = 1$, the sequence of iterated bias corrected smoothers agrees with the L_2-boosted smoothers without shrinkage. For $d > 1$, the boosting algorithm is applied component-wise to additive regression models (see Bühlmann and Yu 2003). This results in a sequence of constrained (additive) approximation of the fully nonparametric regression function m.

For thin-plate splines and kernel smoothers (with suitable kernels, such as a Gaussian density function), each iteration of the bias correction produces a noisier but less biased smoother. In the limit, the sequence of iterative bias corrected smoothers reproduces the raw data (Cornillon *et al.* 2014). Thus there is a need for good stopping rules for the iterative bias correction algorithm.

To illustrate this behavior, let us use a classical bivariate regression problem: Figure 1 graphs Wendelberger's test function (Wendelberger 1982):

$$\begin{aligned} m(x_1, x_2) = \frac{3}{4}\exp\Big\{-((9x-2)^2+(9y-2)^2)/4\Big\}+\frac{3}{4}\exp\Big\{-((9x+1)^2/49+(9y+1)^2/10)\Big\}+ \\ \frac{1}{2}\exp\Big\{-((9x-7)^2+(9y-3)^2)/4)\Big\} - \frac{1}{5}\exp\Big\{-((9x-4)^2+(9y-7)^2)\Big\}. \end{aligned} \tag{5}$$

The sequence of bias corrected thin-plate spline smoothers, starting from a pilot that oversmooths the data, converges to an interpolant of the raw data (see Figure 2c). After some suitable number of bias correction steps, the resulting bias corrected smoother will be a good estimate for the true underlying regression function (see Figure 2b). The crucial choice of k is achieved by using classical criteria such as the corrected Akaike informative criterion (AIC) or generalized cross-validation (GCV, see Section 2.2).

We note that, provided λ is large enough, its exact value is not crucial as the choice of k will adapt to λ: If two base smoothers are chosen, one with λ_1 and another with $\lambda_2 > \lambda_1$, the

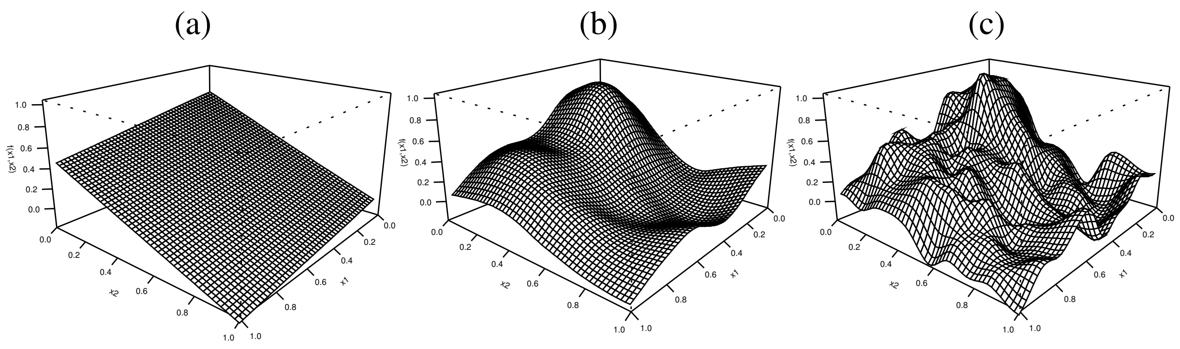


Figure 2: Thin-plate spline regression smoothers from 100 noisy observations from Equation 5 (see Figure 1) evaluated on a regular grid on $[0,1]\times[0,1]$. Panel (a) shows the pilot smoother, panel (b) graphs the bias corrected smoother after 500 iterations and panel (c) graphs the smoother after 50000 iterations of the bias correction scheme.

chosen iteration k_2 will be greater than k_1 as it takes more iterations with a very smooth base smoother to remove the bias.

To enable predictions at arbitrary locations $x \in \mathbb{R}^d$ of the covariates, we extend linear smoothers to functions of the form

$$\hat{m}(x) \quad = \quad S_\lambda(x)^\top Y, \tag{6}$$

where $S(x)$ is a vector of size n whose entries are the weights for predicting $m(x)$. The vector $S(x)$ reduces to the jth row of the smoothing matrix when $x = X_j$, and is readily computed for many smoothers used in practice.

To enable predictions at arbitrary locations $x \in \mathbb{R}^d$ for the iterative bias procedure let us recall that the iterative bias correction scheme is the following

$$\begin{aligned}
\widehat{m}_k &= \widehat{m}_0 + \widehat{b}_1 + \cdots + \widehat{b}_k \\
&= S_\lambda[I + (I - S_\lambda) + (I - S_\lambda)^2 + \cdots + (I - S_\lambda)^{k-1}]Y && (7)\\
&= S_\lambda \widehat{\beta}_k. && (8)
\end{aligned}$$

This implies that $\widehat{m}_k(X_j) = S_\lambda(X_j)^\top \widehat{\beta}_k$. Hence we propose to extend the iterative bias corrected smoother to $\mathbb{R}^d$ via the function

$$\widehat{m}_k(x) = S_\lambda(x)^\top \widehat{\beta}_k. \tag{9}$$

2.2. Stopping rules

Selecting a suitable number k of bias correction iterations k is crucial. On a theoretical ground, for thin-plate spline base smoothers, there exists a number k that produces an estimator which achieves the minimax rate of convergence in mean square error (see Bühlmann and Yu 2003 for the univariate case and Cornillon *et al.* 2014 for the multivariate counter-part). This optimal number of iterations can be selected from data using classical model selection methodologies such as: GCV (Craven and Wahba 1978), AIC (Akaike 1973), Bayesian informative criterion (BIC, Schwarz 1978), corrected AIC (AICc, Hurvich, Simonoff, and Tsai 1998) or generalized minimum description length (gMDL, Hansen and Yu 2001). In particular, the use of AICc is advocated in Bühlmann and Hothorn (2007) and theoretical consideration on the use of GCV can be found in Cornillon *et al.* (2014). Other methods such as cross-validation (leave-one-out or K-fold) or the use of training set and test set are also reasonable procedures to estimate k (Bühlmann and Hothorn 2007).

2.3. Smoothers

The behavior of the sequence of iterative bias corrected kernel smoothers depends critically on the properties of the smoother matrix. The eigenvalues of the matrix should be positive and less or equal to one. This ensures that, as the number of iterations k increases, the smoothing remains stable. The **ibr** package has three types of smoothers: kernel, Duchon splines and low rank splines.

In order to have this paper as self-contained as possible, a presentation of these smoothers is done in the following three subsections.

Kernel smoothers

In order to have eigenvalues between 0 and 1, the smoothing kernel needs to be positive definite (see Marzio and Taylor 2008; Cornillon *et al.* 2014). Examples of positive definite kernels include the Gaussian and the triangle densities, and examples of kernels that are not positive definite include the uniform and the Epanechnikov kernels.

The proposed package includes the Gaussian product kernel defined in the following way:

- For univariate problems ($d = 1$), the smoother with bandwidth λ is defined as:

$$S_{ij} = \frac{1}{\lambda\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{X_i - X_j}{\lambda}\Big)^2\Big) \cdot \Bigg[\sum_{j=1}^{n} \frac{1}{\lambda\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{X_i - X_j}{\lambda}\Big)^2\Big)\Bigg]^{-1}.$$

- For multivariate problems ($d > 1$), the smoother is simply the product of univariate smoothers. Thus, if we denote $X_i^{(l)}$ the ith observation of the lth variable, we have the Gaussian kernel smoother with bandwidth vector $\lambda = (\lambda^{(1)}, \ldots, \lambda^{(d)})$:

$$S_{ij} = \prod_{l=1}^{d} \Bigg[\frac{1}{\lambda^{(l)}\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{X_i^{(l)} - X_j^{(l)}}{\lambda^{(l)}}\Big)^2\Big)\Bigg] \cdot \Bigg[\sum_{j=1}^{n} \prod_{l=1}^{d} \Bigg[\frac{1}{\lambda^{(l)}\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{X_i^{(l)} - X_j^{(l)}}{\lambda^{(l)}}\Big)^2\Big)\Bigg]\Bigg]^{-1}.$$

Duchon splines of order (ν_0, s)

Splines of order (ν_0, s) are a generalization of thin-plate splines (TPS) of order ν proposed by Duchon (1977). These splines are also called sometimes pseudo-splines or Duchon splines. The latter term appears first in the **mgcv** package (Wood 2017) which is the first public domain implementation of these splines in the field of statistics. In his seminal paper, Duchon defines Duchon splines and proposes a practical characterization using a reproducing kernel which allows their practical implementation. This paper gives also convergence results of interpolation with these Duchon splines. These results are completed by convergence results of both interpolation and smoothing Duchon splines (de Silanes and Arcangéli 1989). To our best knowledge, the first example of Duchon splines in applied statistics can be found in Miller and Wood (2014). In order to have this paper self-contained, we recall here some results on Duchon splines which can be found also in all the previously cited papers.

Recall that TPS arise as the solution of the following minimization problem on the Sobolev space $\mathcal{H}^{(\nu)}$ (see for instance Wood 2003)

$$\frac{1}{n}\|Y_i - f(X_i)\|^2 + \lambda J_\nu^d(f),$$

where

$$J_\nu^d(f) \quad = \sum_{\nu_1+\cdots+\nu_d=\nu} \frac{\nu!}{\nu_1!\cdots\nu_d!}\int\cdots\int\left(\frac{\partial^\nu f}{\partial x_1^{\nu_1}\cdots\partial x_d^{\nu_d}}\right)^2 dx_1\cdots dx_d.$$

The first part of the functional to be minimized controls the data fitting while the second part, $J_\nu^d(f)$, controls the smoothness.

Provided that $2\nu > d$, it can be shown that the function minimizing expression has the form

$$\hat{f}(x) = \sum_{j=1}^{M}\alpha_j\phi_j(x) + \sum_{i=1}^{n}\delta_i\eta(\|x - x_i\|),$$

where the $M = \binom{\nu+d-1}{d}$ functions $\{\phi_j\}$ are linearly independent polynomials spanning the space of polynomials in $\mathbb{R}^d$ of degree less than ν, $\alpha \in \mathbb{R}^M$ and $\delta \in \mathbb{R}^n$ are unknown parameter vectors subject to the constraint that $\Phi^\top\delta = 0$ (with Φ an $n \times M$ matrix of elements $\Phi_{ij} = \phi_j(x_i)$) and the n functions η_i are defined by

$$\eta_i(r) = \begin{cases} \frac{(-1)^{\nu+1+d/2}}{2^{2\nu-1}\pi^{d/2}(\nu-1)!(\nu-d/2)!} r^{2\nu-d}\log(r) & d \text{ even,} \\ \frac{\Gamma(d/2-\nu)}{2^{2\nu}\pi^{d/2}(\nu-1)!} r^{2\nu-d} & d \text{ odd.} \end{cases}$$

Parameter vectors α and δ can be found as solution of a linear system and the associated linear smoother S can be derived from it (see Gu 2002, p. 61).

It is well known that beside computational problems, TPS suffer from the fact that the dimension M of the null space of $J_\nu^d(.)$ increases exponentially with d due to the condition $\nu > d/2$. For instance, when the number of explanatory variables d is equal to 10, ν is at least 6 leading to $M = 3003$ unknown coefficients in α. In his seminal paper Duchon (1977) presents a mathematical framework that extends TPS. Noting that the Fourier transform (denoted by $\mathcal{F}()$) is isometric, the smoothness penalty $J_{\nu_0}^d(f)$ can be replaced by its squared norm in Fourier space, that is,

$$\int \|D^{\nu_0}f(t)\|^2 dt \quad \text{can be replaced by} \quad \int \|\mathcal{F}(D^{\nu_0}f)(\tau)\|^2 d\tau.$$

In the last equation, the parameter ν is replaced by ν_0 to emphasize that it will be chosen differently. In order to solve the problem of exponential growth of the dimension of the null space of $J^d_{\nu_0}(.)$, and to get new interpolation methods, Duchon introduces a weighting function to define a new smoothness penalty:

$$J^d_{\nu_0,s}(f) \;=\; \int |\tau|^{2s} \|\mathcal{F}(D^{\nu_0} f)(\tau)\|^2 d\tau.$$

The solution of the variational problem introduced by Duchon:

$$\frac{1}{n}\|Y_i - f(X_i)\|^2 + \lambda J^d_{\nu_0,s}(f),$$

is

$$g(x) \;=\; \sum_{j=1}^{M_0} \alpha_j \phi_j(x) + \sum_{i=1}^{n} \delta_i \eta^d_{\nu_0,s}(\|x - X_i\|),$$

provided that $\nu_0 + s > d/2$ and $s < d/2$. The $\{\phi_j(x)\}$ are still a basis of the subspace spanned by polynomials of degree $\nu_0 - 1$. We also have that:

$$\eta^d_{\nu_0,s}(r) \;\propto\; \begin{cases} r^{2\nu_0+2s-d}\log(r) & d \text{ if } 2\nu_0 + 2s - d \text{ is even,} \\ r^{2\nu_0+2s-d} & d \text{ otherwise,} \end{cases}$$

still with the same constraint on coefficients: $\Phi^\top \delta = 0$.

For the special case $s = 0$, the Duchon splines reduce to the TPS. But if one wants to have a lower dimension for the null space of $J^d_{\nu_0,s}$, for instance pseudo-cubic splines with an order $\nu_0 = 2$, one can choose (as suggested by Duchon, 1977) $s = \frac{d-1}{2}$. For instance, when the number of explanatory variables d is equal to 10, ν_0 can be chosen equal to 2, $s = 9/2$ and $M_0 = \binom{\nu_0+d-1}{d} = 11$ unknown coefficients in α are to be estimated, which is tractable even for moderate datasets.

Low rank Duchon splines of order (ν_0, s)

The following presentation is based upon the presentation done in the seminal paper of Wood (2003). The Duchon splines fitting problem is

$$\text{minimize } \|Y - E\delta - \Phi\alpha\|^2 + \lambda\delta^\top E\delta \;\text{ subject to } \Phi^\top\delta = 0,$$

where the matrix E is defined by $E_{ij} = \eta^d_{\nu_0,s}(\|X_i - X_j\|)$ and the matrix Φ is defined by $\Phi_{ij} = \phi_j(X_i)$. This ideal fitting problem can be restricted by searching a given rank r parameter space that gives a good approximation to the Duchon splines fitting problem. The general low rank problem is defined by using a rank r matrix Γ_r such that $\delta = \Gamma_r\delta_r$ and by the following problem:

$$\text{minimize } \|Y - E\Gamma_r\delta_r - \Phi\alpha\|^2 + \lambda\delta_r^\top\Gamma_r^\top E\Gamma_r\delta_r \;\text{ subject to } \Phi^\top\Gamma_r\delta_r = 0.$$

In order to have a well defined problem, the matrix Γ_r is chosen to be equal to U_r, an $n \times r$ matrix whose jth column is the eigenvector of E corresponding to the jth eigenvalue $D_{j,j}$ and where the eigenvalues are arranged so that $|D_{j,j}| \geq |D_{j+1,j+1}|$. This choice ensures good approximation properties (see Wood 2003). Using that, the low rank Duchon splines fitting problem is

$$\text{minimize } \|Y - U_rD_r\delta_r - \Phi\alpha\|^2 + \lambda\delta_r^\top D_r\delta_r \;\text{ subject to } \Phi^\top U_r\delta_r = 0.$$

To ensure that the condition $\Phi^\top U_r \delta_r = 0$ is fulfilled, an orthogonal column basis Z_r can be found (e.g., by using the last $M_0 - r$ columns of the Q matrix of a complete QR factorization of $(\Phi^\top U_r)^\top$). Thus any linear combination $Z_r\tilde{\delta}, \tilde{\delta} \in \mathbb{R}^{M_0 - r}$, leads to a parameter $\delta_r = Z_r\tilde{\delta}$ which fulfills the condition. The low rank Duchon splines fitting problem becomes

$$\text{minimize } \|Y - U_r D_r Z_r \tilde{\delta} - \Phi\alpha\|^2 + \lambda\tilde{\delta}^\top Z_r^\top D_r Z_r \tilde{\delta}$$

with respect to $\alpha \in \mathbb{R}^{M_0}$ and $\tilde{\delta} \in \mathbb{R}^{r-M_0}$. This approximation problem can be recasted in a constrained regression framework by setting $X = (U_r D_r Z_r, T)$, $\beta^\top = (\tilde{\delta}^\top, \alpha^\top)$ leading to

$$\text{minimize } \|Y - X\beta\|^2 + \lambda\beta^\top C\beta$$

with the constraint block matrix C (of dimension $r \times r$) equal to

$$C = \begin{pmatrix} Z_r^\top D_r Z_r & 0 \\ 0 & 0 \end{pmatrix}.$$

The solution of this constrained regression is thus

$$\hat{\beta} = (X^\top X + \lambda C)^{-1} X^\top Y,$$

and the low rank splines smoothing matrix is

$$S = X(X^\top X + \lambda C)^{-1} X^\top.$$

3. Implementation in R

Our implementation of the iterative bias corrected procedure in R follows the established S3 methods (for an introduction, see the help page of `S3Methods` in R). The main function (called `ibr`) produces an object of class '`ibr`'. Applying generic functions, such as `summary`, `predict`, `plot` or `residuals`, to an '`ibr`' class object produces the expected standard summary statistics, prediction for new data (or fitted values), plot of the object and residuals.

3.1. Base smoother

Three types of base smoothers are implemented in the function `ibr`: TPS and Duchon splines, low rank TPS and Duchon splines, and kernel smoother. This choice is driven by the `smoother` argument (character): `"tps"`, `"ds"` or `"lrtps"`, `"lrds"`, or `"k"`. As TPS of order ν are a particular case of Duchon splines of order (ν, s) (when the order is chosen equal to $(\nu, 0)$) this artificial distinction in five cases is maintained only for the ease of the user. For kernel smoother, some classical choices are available using the `kernel` argument (character): Gaussian kernel (`"g"`, the default), triangle density (`"t"`), and the quartic (`"q"`) density. The computations have been optimized for the Gaussian kernel. We also allow for the Epanechnikov (`"e"`) and uniform (`"u"`) kernels for pedagogical purposes.

3.2. Computations

To predict new data, we compute recursively $\hat{\beta}_k$ using Equations 7 and 8. Computation of the fitted values using Equation 4 can be performed using a similar recursive update formula;

starting with $\hat{b}_0 = (I - S_\lambda)Y$:

$$\hat{m}_k = Y - (I - S_\lambda)\hat{b}_{k-1} \quad \text{and} \quad \hat{b}_{k-1} = (I - S_\lambda)b_{k-2}.$$

Computations of either $\hat{b}_k$ or $\hat{\beta}_k$ require matrix-vector multiplications, i.e., level 2 **BLAS** functions (Golub and Van Loan 1996) with $O(n^2)$ flops. In practice, we found that often the number of iterations k that are required to be evaluated in order to select a good data-driven choice $\hat{k}$ is commensurate with the sample size n. Thus, an algorithm which uses matrix-vector multiplications would require typically $O(n^3)$ flops to produce the final smoother.

Numerical experiments have shown that an alternative algorithm, based on an eigenvalue decomposition of the smoothing matrix S_λ (also an order $O(n^3)$ algorithm), is faster when combined with GCV for selecting the number of iterations. We have implemented the latter algorithm in the **ibr** package. This approach is easily understood and implemented for TPS smoothers, whose smoothing matrix S_λ is symmetric. For kernel smoothers, the smoothing matrix is not symmetric and further discussion is needed.

While the kernel base smoother S_λ is not symmetric, we can rewrite Equation 4 using an eigen decomposition of a symmetric matrix. Specifically, write $S_\lambda = D\mathbb{K}$, where $\mathbb{K}$ is a symmetric matrix with general element $\mathbb{K}_{ij} = \prod_{l=1}^{d} K\left\{(X_i^{(l)} - X_j^{(l)})/\lambda^{(l)}\right\}$ and D a diagonal matrix with entries $D_{ii} = 1/\sum_{j=1}^{n}\mathbb{K}_{ij}$. With this notation, we write the smoothing matrix of $\hat{m}_k$ in Equation 4 as

$$\begin{aligned} I - (I - S_\lambda)^k &= I - (I - D\mathbb{K})^k \\ &= I - (D^{1/2}D^{-1/2} - D^{1/2}D^{1/2}\mathbb{K}D^{1/2}D^{-1/2})^k \\ &= I - D^{1/2}(I - A)^k D^{-1/2} \end{aligned}$$

where $A = D^{1/2}\mathbb{K}D^{1/2}$. The latter is symmetric, and so it can be diagonalized $A = U\Lambda U^\top$, with U the orthogonal matrix of eigenvectors and Λ the diagonal matrix of eigenvalues. Equation 4 becomes

$$\hat{m}_k = D^{1/2}U(I - (I - \Lambda)^k)U^\top D^{-1/2}Y.$$

The coefficient $\hat{\beta}_k$ in Equation 7 becomes

$$\hat{\beta}_k = D^{1/2}U[I + (I - \Lambda) + (I - \Lambda)^2 + \cdots + (I - \Lambda)^{k-1}]U^\top D^{-1/2}Y.$$

Recognizing the sum inside the bracket as the $k-1$ first term of geometrical series, we rewrite

$$\hat{\beta}_k = D^{1/2}U\Lambda^{-1}(1 - (I - \Lambda)^k)U^\top D^{1/2}.$$

The core of computation becomes the eigen decomposition which is done in a very efficient way by the function `eigen` for moderate n ($n < 1000$ for instance). For additional efficiency, the computations of A and $D^{1/2}$ are done in C for the default Gaussian kernel.

For the low rank splines, the user must provide a given rank $r < n$. The smoother

$$S = X(X^\top X + \lambda C)^{-1}X^\top \tag{10}$$

can be written, using a QR transformation of X ($X = QR$ and Q an orthogonal $n \times r$ matrix), as

$$\begin{aligned} S &= QR(R^\top R + \lambda C)^{-1}R^\top Q^\top = QR(R^\top(I + \lambda(R^\top)^{-1}CR^{-1})R)^{-1}R^\top Q^\top \\ &= Q(I + \lambda(R^\top)^{-1}CR^{-1})Q^\top. \end{aligned}$$

Using an eigen decomposition of the $r \times r$ symmetric matrix $(R^\top)^{-1}CR^{-1}$ which can be written as $V\Lambda V^\top$ (with V orthogonal) we get the symmetric smoother

$$S = Q(VV^\top + \lambda V\Lambda V^\top)^{-1}Q^\top = QV(I + \lambda\Lambda)^{-1}V^\top Q^\top.$$

Here, the eigen decomposition of the smoother is given by the last equation: The $(r-i+1)$th eigenvalue is $(1 + \lambda\Lambda_i)^{-1}$ and the associated eigenvector is the ith column of QV. When the computation of X is done, the computation cost is dominated by the QR decomposition of X (using **LINPACK**, the default in R) which requires $O(nr^2 - r^3/3)$ flops (see Bischof and Van Loan 1987). To calculate X, an eigen decomposition is needed (to calculate the U_r matrix). The computations are made via the `smoothCon` function of package **mgcv** which uses a Lanczos method for the eigen decomposition. Recall that each iteration of the Lanczos algorithm needs to use a matrix-vector multiplication. In the **mgcv** package, the level 2 **BLAS** function `dsymv` of **LAPACK** is used for efficiency and thus each iteration of the Lanczos algorithm requires $O(n^2)$ flops. As the number of iterations in the **mgcv** package is at least r, we have a computation cost for low rank splines, with large n and small fixed r, of at least $O(rn^2)$.

3.3. Stopping rules

The **ibr** package implements several classical criteria to empirically select an *optimal* number k of bias correction iterations: generalized cross-validation (GCV), Akaike information criterion (AIC), Bayesian information criterion (BIC), corrected Akaike information criterion (AICc) and generalized minimum description length (gMDL). The choice of criterion is controlled by the argument `criterion` of the `ibr` function. By default, the criterion is GCV. Cross-validation is also available, but our discussion on that method is postponed to Section 3.5.

The evaluation of an *optimal* number k of iterations using any one of these classical criteria is not a trivial task. The package **ibr** implements both a computationally burdensome exhaustive search method and a computationally efficient but approximate method. The latter is the default method. The user can request `ibr` to perform an exhaustive search by setting the argument `exhaustive = TRUE` in the list `control.par`.

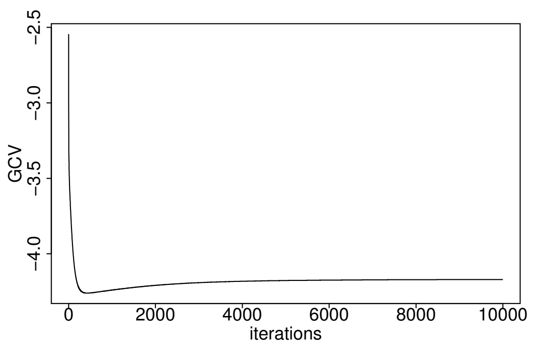
Exhaustive search method

The exhaustive search method evaluates, for each k in an interval $[K_{\min}; K_{\max}]$, the criterion to identify its global minimizer. The default values for the range are $K_{\min} = 1$ and $K_{\max} = 10^6$.

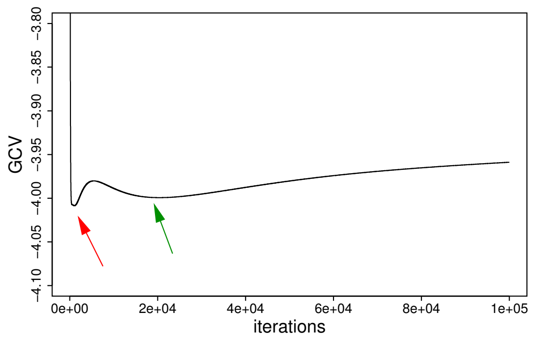
Numerical optimization method

The default method relies on the fact that the criterion is easily calculated for arbitrary $k \in \mathbb{R}^+$. This enables us to use standard optimization routines to minimize the criterion. While this approach is conceptually simple, there are two pitfalls: First, most criteria break down for very large k for which the smoother essentially interpolates the data, i.e., $\hat{m}_k \approx Y$. Second, some criteria exhibit multiple local minima (see Figure 3b).

All model selection criteria trade-off goodness of fit, as measured by $\log(\|Y - \hat{m}_k\|^2)$ with a measure of the complexity of the smoother. Numerical difficulties arise when $Y \approx \hat{m}_k$, which occurs when the number k of iterations is close to the sample size n. To overcome this problem, we bound from above the maximum allowable number of iterations by setting the variable



(a) Typical evolution of GCV with k.



(b) Evolution of GCV with k with two minima indicated by arrows.

`dfmaxi` in the list `control.par`. By default, its value is $2n/3$. Hardcoded error handling prevents evaluation of the criteria when either $k > n(1-10^{-10})$ or $\|Y-\hat{m}_k\|^2 \leq 10^{-10}$. These exceptions also apply to the exhaustive search algorithms.

Classical model selection criteria have been developed in the context where the effective number of estimated parameters is significantly smaller than the number of observations. Investigation of the criteria, as a function of effective degrees of freedom over a broader range of values reveals the presence of multiple local minima. While this does not impact the performance of the exhaustive search, the presence of local minima is potentially problematic for standard minimization algorithms. Our solution is to divide the interval $[K_{\min}; K_{\max}]$ into smaller subintervals and apply on each subinterval a numerical optimization using the function `optimize`, and the minimizer of these minimizations is returned. The splitting is controlled by the argument `fraction` in the list `control.par`, with default value of `c(100, 200, 500, 1000, 5000, 1e04, 5e04, 1e05, 5e05, 1e06)`.

While the strategy of optimizing the criteria in subintervals is more expensive than optimizing over the original interval, it remains significantly faster than performing an exhaustive search.

3.4. Scales of variables

The function `ibr` is designed to be used with two types of linear smoothers: Duchon splines (full rank or low rank) and kernel smoothers. Duchon splines are governed by a single parameter λ that weights the contribution of the roughness penalty. As a result, it is desirable to scale all the variables to have equal variance to ensure that the roughness penalty is applied equally to each variable. This is achieved by pre-processing the data with the `scale` function before smoothing the data with `ibr`. By default, the variables are scaled when splines smoother is selected (both full rank or low rank).

Our implementation of the kernel smoother allows for a different bandwidth to be used for each of the regression variables. While the discussion on scaling applies when a common bandwidth is used for all the variables, we found in our numerical experiments that we get better results when we use the original variable but select a suitable bandwidth for each variable. The objective is not to select an optimal bandwidth, but rather control the amount of smoothing we do at each iteration. To this end, we propose to select the bandwidths such that the one-dimensional smoothing matrix for each variable has the same effective degree of freedom. Typical values for the effective degree of freedom values are 1.05, 1.1, 1.2, 1.5

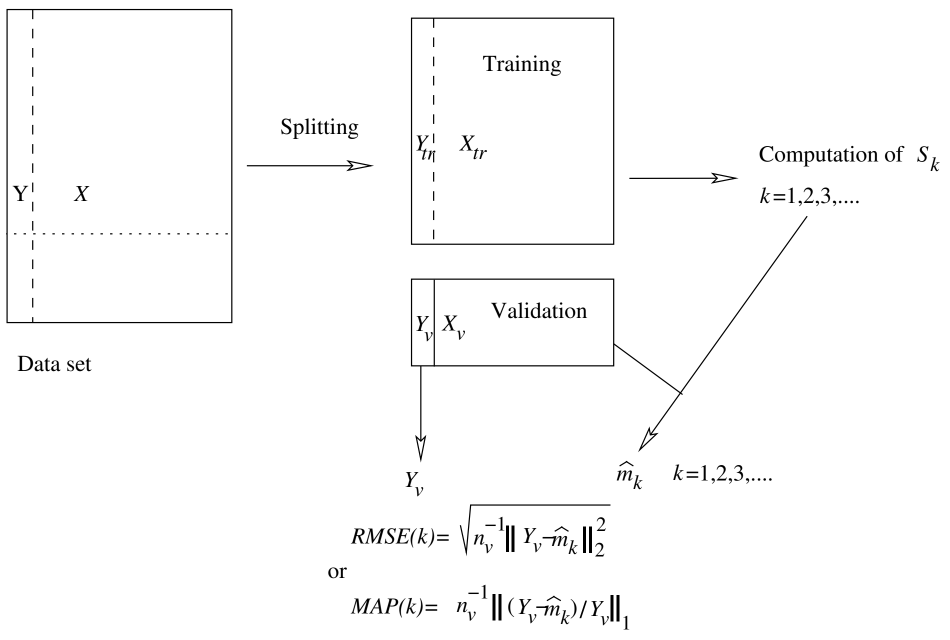

Data set
Y
X
Splitting
Training
Y_{tr}
X_{tr}
Validation
Y_v
X_v
Computation of S_k
k=1,2,3,....
$\widehat{m}_k$ k=1,2,3,....
Y_v
$RMSE(k) = \sqrt{n_v^{-1} \| Y_v - \widehat{m}_k \|_2^2}$
or
$MAP(k) = n_v^{-1} \| (Y_v - \widehat{m}_k)/Y_v \|_1$

Figure 3: Training set and validation set.

or 2, which the user sets with the `df` argument. Given an desired effective degree of freedom, the package automatically determines the bandwidth using an adaptation of the `uniroot` algorithm written in C.

Relating the effective degree of freedom of each of the univariate components to an effective degree of freedom for the multivariate smoother is non-trivial. As a result, some users may prefer to control the overall smoothing instead of the marginal smoothing of each component. We allow the specification of the overall smoothing by setting the flag `dftotal = TRUE` in list `control.par` (the default value of that flag is `dftotal = FALSE`): When set to `TRUE`, the user specifies the desired overall degree of freedom of S with the argument `df`. The `ibr` function determines a vector $(\lambda^{(1)}, \dots, \lambda^{(l)})$ that produces a smoothing matrix S with the desired overall degree of freedom using a C routine.

3.5. Stopping rules: K-fold cross-validation and data splitting

Leave-one-out cross-validation, K-fold cross-validation, and more generally data splitting, are well established techniques for model selection which can be used to determine the optimal number of iterations k for iterative bias reduction procedure. First, the dataset is split into two disjoint subsets, a training set to estimate the regression function and a testing set to evaluate the out-of-sample prediction error. This training/test split can be done several times. Second, using either the root mean square error `criterion = "rmse"` or the mean of absolute error `criterion = "map"` loss functions (see Figure 3) the out-of-sample prediction error is quantified. The optimal k selected is the one which leads to the minimum of the chosen loss. This minimization can be done either using an optimization routine, the default method, or by exhaustive search (set `exhaustive = TRUE` in the list `control.par`).

Since simple leave-one-out cross-validation usually leads to an estimator that undersmooths (in our case, the selected number of iterations k is larger than the optimal one), we prefer to use either data splitting or K-fold cross-validation. The main difference between these two procedures is that usually data splitting is conducted once (except if the user asks for more using argument `npermut`) whereas for K-fold cross-validation, the original sample is randomly partitioned into K subsamples with each of the K subsets used as the test set and the remainder $K-1$ subsets are combined to form the training set. The prediction error is then computed by summing absolute relative errors or squared errors across the K trials (and making the overall mean).

The list `cv.options` in `ibr` controls the various options for cross-validation, including the size of the training set, the number of repetitions of the procedure, the loss function and the type of splitting.

Selecting the number of iterations k with data splitting

To perform a data splitting cross-validation, we set the following options in `cv.options`:

1. Input either `ntest` or `ntrain`, the size of the testing set n_v or the size of the training test $(n - n_v)$, respectively. The default value sets `ntest` to $\lfloor n/10 \rfloor$.

2. Set the number of times the dataset is split in `npermut`. For classical data splitting, `npermut` have to be set equal to one (the default value is 20).

3. Set the `type` equal to `random` to allow random data splitting. This option can be omitted as this is the default value. The argument `seed` can be used to control the seed of the random number generator.

Data splitting (with test set of size $\lfloor n/10 \rfloor$, i.e., the default value) with root mean square error loss is achieved by the code

```
R> ibr(Y ~ ., data = dataset, criterion = "rmse",
+    cv.options = list(npermut = 1))
```

A more complex example of data splitting that uses 100 samples of 3 observations to evaluate the prediction error using the mean absolute deviation loss is achieved with the code

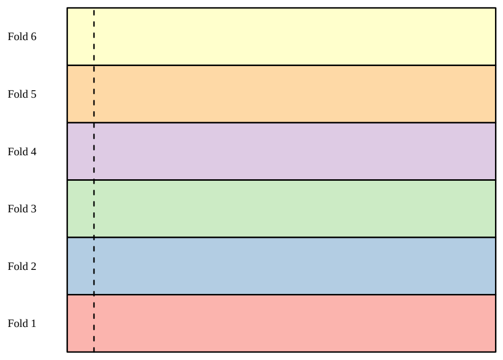
```
R> ibr(Y ~ ., data = dataset, criterion = "map", cv.options =
+    list(ntest = 3, npermut = 100))
```

Selecting the number of iterations k with K-fold cross-validation

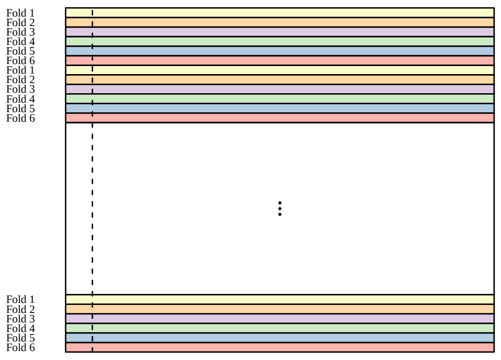
To perform a K-fold cross-validation with `ibr`, we set the following options in `cv.options`:

1. Set `Kfold = TRUE` (default is `FALSE`) or set `Kfold` equal to the number of folds.

2. Set the number of folds K. One can either specify the size of the testing set n_v in `ntest` or the size of the training set $(n - n_v)$ in `ntrain`, in which case the fold is computed to be $K = \lfloor n/n_v \rfloor$. One can set the number of folds K by setting the argument `Kfold` equal to K. This implies that the size of the testing set is $\lfloor n/K \rfloor$.



(a) Graphical summary of consecutive $K = 6$ folds cross-validation.



(b) Graphical summary of interleaved $K = 6$ folds cross-validation.

Figure 4: Options `"consecutive"` and `"interleaved"` for K-fold cross-validation.

3. Specify the `type` of data-split. By default, the data are split randomly (`"random"`). Alternatively, we divide the data using consecutive stretches of data (`"consecutive"`) or interleaved split (`"interleaved"`). A forth option, `"timeseries"` divides the data chronologically and uses the last $\lfloor n/K \rfloor$ for the testing set. The splitting implied by `"consecutive"` is shown in Figure 4a while the splitting using `"interleaved"` is shown in Figure 4b. The obvious case of random draw is not shown. Finally, the optional argument `seed` can be used to control the seed for the random number generator. It is given as `seed` argument of the `set.seed` function.

The first two lines of code give rise to the examples summarized in Figures 4a and 4b, while the third line corresponds to a random K-fold cross-validation:

```
R> ibr(Y ~ ., data = dataset, criterion = "rmse", cv.options =
+    list(Kfold = 6, type = "consecutive"))
R> ibr(Y ~ ., data = dataset, criterion = "rmse", cv.options =
+    list(Kfold = 6, type = "interleaved"))
R> ibr(Y ~ .,data=dataset,criterion = "rmse", cv.options =
+    list(Kfold = 6, type = "random"))
```

Finally, if the user wants to perform an exhaustive search for the number of iterations (from 1 to 1000 iterations) using the *leave-one-out* cross-validation, she runs

```
R> ibr(Y ~ ., data = dataset, criterion = "rmse", Kmax = 1000, control.par =
+    list(exhaustive = TRUE), cv.options = list(Kfold = TRUE, ntest = 1,
+    type = "consecutive"))
```

3.6. Variable selection

We can apply the standard strategy of balancing prediction errors and model complexity to select predictors. The main issue with variable selection with **ibr** is computational, as we wish to compare models using an optimal number of bias reduction iterations. To limit fitting models with many parameters, we only consider forward variable selection.

Algorithm 1 Description of the forward function.

```
Require: criterion (GCV, AIC, AICc, BIC, gMDL, MAP or RMSE)
Require: varcrit (GCV, AIC, AICc, BIC, gMDL)
  $s \leftarrow 1$ # Current stage
  $R$ matrix of infinity with $d$ columns # Matrix of results
  $\mathcal{S} \leftarrow \emptyset$ # Variable(s) selected at current stage
  $s_{\min} \leftarrow \infty$ # Current minimum of criterion
  for $s = 1$ to $d$ do
    for $j = 1$ to $d$ such that $j \notin \mathcal{S}$ do
      $\mathcal{S}_c \leftarrow \mathcal{S} \cup \{j\}$ # Adding one variable to the set of variables already selected
      res <- ibr($X_{\mathcal{S}_c}$,Y,criterion) # $X_{\mathcal{S}_c}$ contains explanatory variables in $\mathcal{S}_c$
      evaluation of criterion varcrit for res: $R_{sj}$
    end for
    if all $\{R_{sj}\}_j > s_{\min}$ then
      Return matrix $R$ from row 1 to $s - 1$
    else
      # Updating
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\arg\min_j R_{sj}\}$
      $s_{\min} \leftarrow \min_j R_{sj}$ # Current minimum of criterion varcrit
      $s \leftarrow s + 1$
    end if
  end for
```

In analogy to selecting the number of iterations, controlled by entries in the list `criterion`, we control the variable selection procedure with the list `varcrit`. The latter has the same default values as the former.

In the following example, we use GCV to select the number of iterations and BIC for forward variable selection. In the first stage ($s = 1$) all models with one variable are computed. Computing these models induces a selection of the number of iterations for each model using the GCV criteria. The variable with the smallest BIC is selected. At the second stage ($s = 2$), the remaining variables are added in turn into the model. For each of the best fitting two variables models (using GCV to select k), the BIC is calculated and the two variables model with the smallest value is retained. If there is no two variables models that have smaller BIC than the best one variable model, then the variable selection procedure stops. Otherwise, we continue and consider all three variables models that extend the best two variables model (see also Algorithm 1).

The `forward` function returns an object of class '`forwardibr`'. A plot method is provided for this class of object.

4. Examples

Let us return to the Wendelberger's test function (see Equation 5):

```
R> f <- function(x, y) {
+    0.75 * exp(-((9 * x - 2)^2 +(9 * y - 2)^2)/4) +
+    0.75 * exp(-((9 * x + 1)^2/49 + (9 * y + 1)^2/10)) +
+    0.50 * exp(-((9 * x - 7)^2 + (9 * y - 3)^2)/4) -
+    0.20 * exp(-((9 * x - 4)^2 + (9 * y - 7)^2))
+ }
```

We start by plotting this function on a 50×50 grid of points in the unit square $(0, 1) \times (0, 1)$ that produces Figure 1.

```
R> ngrid <- 50
R> xf <- seq(0, 1, length = ngrid + 2)[-c(1, ngrid + 2)]
R> yf <- xf
R> zf <- outer(xf, yf, f)
R> grid <- cbind(rep(xf, ngrid), rep(xf, rep(ngrid, ngrid)))
R> persp(xf, yf, zf, theta = 130, phi = 20, expand = 0.45,
+    main = "True Function")
R> griddata <- cbind.data.frame(x = grid[, 1], y = grid[, 2])
```

Next, we can generate a dataset of 100 noisy observations of the function f evaluated on the regular grid $\{0.05, 0.15, \ldots, 0.85, 0.95\}^2$, with Gaussian disturbances that have zero mean and standard deviation producing a signal to noise ratio of five.

```
R> noise <- 0.2
R> N <- 100
R> xr <- seq(0.05, 0.95, by = 0.1)
R> yr <- xr
R> zr <- outer(xr, yr, f)
R> set.seed(25)
R> std <- sqrt(noise * var(as.vector(zr)))
R> noise <- rnorm(length(zr), 0, std)
R> Z <- zr + matrix(noise, sqrt(N), sqrt(N))
```

Concatenate explanatory variables into a 100×2 matrix `X` and put the dependent variable in vector form `Zc` and put them in a data-frame `data`

```
R> xc <- rep(xr, sqrt(N))
R> yc <- rep(yr, rep(sqrt(N), sqrt(N)))
R> X <- cbind(xc, yc)
R> Zc <- as.vector(Z)
R> data <- cbind.data.frame(x = xc, y = yc, z = Zc)
```

In this example, we will use TPS of order ν. The default value is the smallest possible smoothness, which is 2 in our case. The effective degree of freedom of the TPS smoother needs to be slightly larger than $M = \binom{\nu+d-1}{\nu-1}$. In our example, $M = 3$ and we choose λ such that the effective degree of freedom is $1.1 \times M = 3.3$. Figure 2(a) graphs the base smoother at iteration zero.

```
R> res.ibr <- ibr(z ~ x + y, data = data, df = 1.1, control.par =
+    list(iter = 1), smoother = "tps")
R> fit <- matrix(predict(res.ibr, griddata), ngrid, ngrid)
R> persp(xf, yf, fit, theta = 130, phi = 20, expand = 0.45,
+    main = "Fit", zlab = "fit")
```

Figures 2(b) and (c) show the bias corrected smoother after 500 and 50,000 iterations. To compute the smoother whose number of iterations is selected with GCV, we use

```
R> res.ibr <- ibr(z ~ x + y, data = data, df = 1.1, smoother = "tps")
R> summary(res.ibr)
```

The summary output of the resulting smoother prints the residual standard error, the initial degree of freedom and reveals that the final degree of freedom is 26.5 and the value of (log) GCV is -3.63 after $\hat{k}_{GCV} = 424$ iterations.

```
Residuals:
      Min        1Q    Median        3Q       Max
-0.235037 -0.068251 -0.007412  0.069063  0.301480
Residual standard error: 1.197 on 73.5 degrees of freedom

Initial df: 3.3 ; Final df: 26.5
   gcv
-3.938

Number of iterations: 424 chosen by gcv
Base smoother: Thin plate spline of order 2 (with 3.3 df)
```

To compute the fitted values, we use the predict function

```
R> predict(res.ibr)
```

that can be used to evaluate the mean absolute error (MAE) on a grid

```
R> mean(abs(predict(res.ibr, griddata) - as.vector(zf)))
[1] 0.05783938
```

To plot the fitted value, we employ the following code

```
R> predgrid <- matrix(predict(res.ibr, griddata), ngrid, ngrid)
R> persp(xf, yf, predgrid, theta = 130, phi = 20, expand = 0.45,
+    zlab = "fit")
```

To use either the AICc or the BIC criterion to select the number of iterations, we write

```
R> res.ibr.aicc <- ibr(z ~ x + y, data = data, df = 1.1, smoother = "tps",
+    crit = "aicc")
R> res.ibr.bic <- ibr(z ~ x + y, data = data, df = 1.1, smoother = "tps",
+    crit = "bic")
```

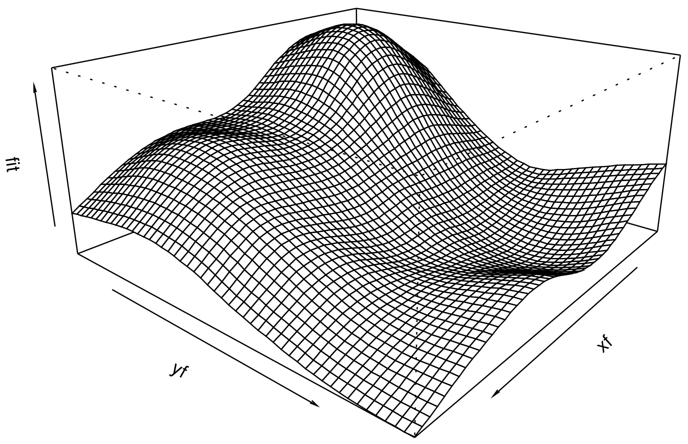


Figure 5: Fitted regression function $\hat{m}_k(x_1, x_2)$ on the unit square $[0, 1] \times [0, 1]$, the number of iterations is chosen by GCV: $\hat{k}_{GCV} = 424$.

Direct display of an '`ibr`' object gives the following short description

```
R> res.ibr.aicc
```

```
Initial df: 3.3 ; Final df: 20.976
Number of iterations: 247 chosen by aicc
```

which reveals that AICc required 247 iterations and the resulting smoother has a slightly larger mean absolute error than those obtained by using GCV. This last MAE is close to the TPS smoother with λ (not k) selected with GCV

```
R> library("fields")
R> res.tps <- Tps(X, Zc)
R> mean(abs(predict(res.tps, griddata) - as.vector(zf)))
```

```
[1] 0.05823783
```

4.1. Real example: Los Angeles ozone data

We consider the classical Los Angeles basin ozone concentration dataset used by numerous authors (see for example Breiman 1996; Bühlmann and Yu 2003, 2006) to demonstrate the performance of various high dimensional smoothing techniques. The data consists of $n = 330$ observed ozone concentration values related to $d = 8$ explanatory variables.

To use the TPS base smoother, the order ν of TPS needs to be greater than $d/2$, that is $\nu = 5$. This implies that the minimal effective degree of freedom of the TPS smoother S_λ is $M = 495$, which is greater than the sample size n. Even for larger sample sizes, say $n = 500$, the TPS will be an unsatisfactory base smoother (recall that in the preceding section, for $d = 2$ we started at 3.3 degrees of freedom with 100 observations). Other types of base smoother, Kernel smoother or Duchon splines (low rank or full rank), can be used easily.

Let us consider the (default) Gaussian kernel smoother. As discussed in Section 3.4, we do not scale the eight explanatory variables but instead select the bandwidth of each univariate smoother to achieve a smoothing matrix that has an effective degree of freedom of 1.1. This ensures that at face value, each of the eight covariates has the same influence. The number of possible bias correction iterations k considered by the model selection procedure for selecting the optimal number of iterations lies between one and 10^6 (default values for `Kmin` and `Kmax`). The R code for fitting this data is

```
R> data("ozone", package = "ibr")
R> res.ibr <- ibr(Ozone ~ ., data = ozone, df = 1.1)
R> summary(res.ibr)

Residuals:
     Min       1Q   Median       3Q      Max 
-13.5581  -2.0566  -0.3481   1.9816  12.6049 
Residual standard error: 71.69 on 309.6 degrees of freedom

Initial df: 2.06 ; Final df: 20.42 
  gcv 
2.809 

Number of iterations: 64 chosen by gcv 
Base smoother: gaussian kernel (with 2.06 df)
```

The summary shows that the optimal number of iterations is $\hat{k}_{GCV} = 64$, which can be thought as quite low (recall that in the previous example the number of iterations ranged between 200 and 400). In this example, an exhaustive search method for determining the optimal number of iterations

```
R> ibr(Ozone ~ ., data = ozone, df = 1.1,
+    control.par = list(exhaustive = TRUE))
```

gives the same result. Because we only need a relatively small number of bias correction steps, we can select a smaller initial effective degree of freedom, say 1.05, while maintaining the computational complexity at a manageable level. Indeed, decreasing the effective degree of freedom of the pilot smoother increases the total number of bias reduction steps while typically providing some performance gains as measured by out-of-sample prediction errors.

A plot method is also available for the '`ibr`' object to display the residuals as a function of fitted values. Figure 6 shows that variability increases with the fitted values that is heteroscedasticity is suspected.

```
R> plot(res.ibr)
```

As done in Bühlmann and Yu (2003), 50 random training/test splits are conducted. Each split is done in order to have a training set of size $n_t = 297$ and a test set of size $n_v = 33$:

```
R> erreur1.5 <- rep(0, 33 * 50)
R> set.seed(123)
```

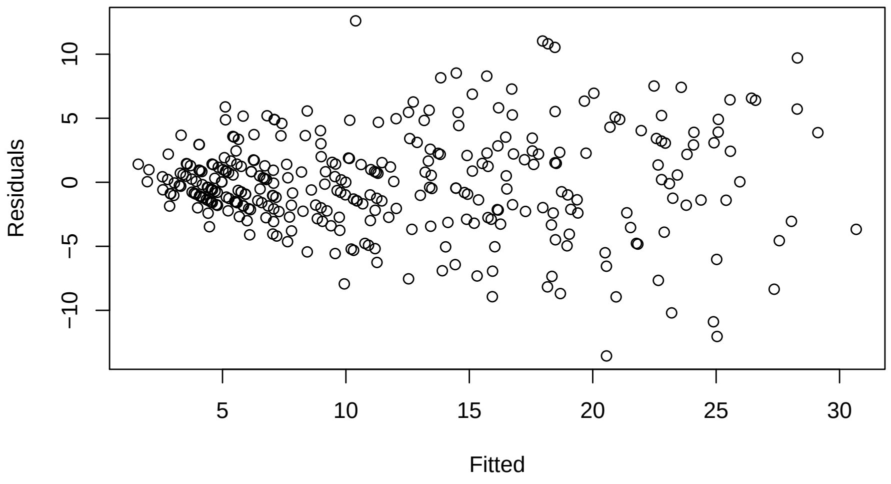


Figure 6: Fitted vs. residuals plot.

```
R> for (i in 1:50) {
+    indT <- sample(1:330, 33)
+    indA <- (1:330)[-indT]
+    YT <- ozone[indT, "Ozone"]
+    res.ibr <- ibr(Ozone ~ ., data = ozone, subset = indA)
+    erreur1.5[(33 * (i - 1) + 1):(33 * i)] <-
+      YT - predict(res.ibr, ozone[indT, ])
+  }
R> mean(erreur1.5^2)

[1] 16.51436
```

We get an error of 16.51, which compares favorably with GAM (**mgcv**: 19.29), MARS (**mda**: 18.91), projection pursuit (`ppr`: 18.93 for `nterms = 2`) or boosting (package **mboost**: 17.93). Note that since no default is available for the `nterms` argument of the function `ppr`, we follow the examples provided in the `ppr` documentation and have set `nterms` equal to 2. To summarize, the **ibr** smoother enjoys a 8% reduction in the out-of-sample prediction mean square error over other classical multivariate (constrained) smoothing methods. Other simulations and comparisons can be found in Cornillon, Hengartner, Jégou, and Matzner-Løber (2013).

We note that the above comparison favors the L_2 boosting and MARS algorithms that take advantage of built-in variable selection procedures. To compare to these methods, we apply the forward variable selection using the random splitting method to `ibr` for this dataset issuing the following commands:

```
R> set.seed(123)
R> ind <- sample(1:330, 33)
R> ozoneA <- ozone[-ind, ]
R> ozoneT <- ozone[ind, ]
```

We select variables using the commands

```
R> forward.ibr <- forward(XXA, YA)
R> varnumber <- apply(forward.ibr, 1, which.min)
R> varnumber

[1] 4 3 7 6 5

R> colnames(forward.ibr)[varnumber]

[1] "Temp.Sand"       "Humidity"        "Inv.Base.Temp"   "Pressure.Grad"
[5] "Inv.Base.height"
```

That is, the order of the variables to be included into the model is 4, 3, 7, 6 and 5. Variable selection leads to improved predictions. To quantify this improvement on the testing set, we compare the prediction MSE of the selected five variables model with the prediction MSE of the model that uses all eight variables.

```
R> res.ibr <- ibr(Ozone ~ ., data = ozoneA)
R> mean((ozoneT[, "Ozone"] - predict(res.ibr, ozoneT))^2)

[1] 16.1644

R> selectedvar <- paste(colnames(forward.ibr)[varnumber], collapse = "+")
R> formulaOzone <- formula(paste("Ozone ~ ", selectedvar, sep = ""))
R> res.ibr2 <- ibr(formulaOzone, data = ozoneA)
R> mean((ozoneT[, "Ozone"] - predict(res.ibr2, ozoneT))^2)

[1] 14.29262
```

This shows a small improvement. We remark that despite the increased computational time, the `forward` function provides a simple and useful tool for selecting variables.

Remark: Change of base smoother can lead to improvement depending on the data. On this example, a low rank Duchon splines smoother with rank r chosen equal to 270 (10% less than the maximal value: 300) leads to a MSE of 14.56 and using the same selected variables leads to a MSE of 14.43.

5. Conclusion

The **ibr** package provides additional features which are not offered by other packages on CRAN. These features are a complete implementation, using the R language, of iterative biased reduction procedures which implement and generalize the twicing idea of Tukey (1977).

This smoothing method for multivariate datasets seems to be promising especially on real datasets (see Cornillon, Hengartner, Lefieux, and Matzner-Løber 2015). But one limitation of this smoothing method is the use of an $n \times n$ matrix, where n is the number of observations. Moreover, at the present time, the computational bottleneck is the eigen decomposition of an $n \times n$ matrix. This decomposition can be performed entirely (with a complexity of $O(n^3)$) or can be restrained to the r first eigenvectors by using the low rank splines (Wood 2003). In this case, a complexity of $O(rn^2)$ can be attained.

Acknowledgments

The anonymous referees provided extremely helpful suggestions that have greatly enhanced the quality of the present paper and its companion package. The author are greatly indebted to an unknown referee for suggesting the use of low rank splines to circumvent the computational cost of the full rank splines.

References

Akaike H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In BN Petrov, BF Csaki (eds.), *Second International Symposium on Information Theory*, pp. 267–281. Academiai Kiado, Budapest.

Antoniadis A, Oppenheim G (eds.) (1995). *Wavelets and Statistics*. Springer-Verlag. doi:10.1007/978-1-4612-2544-7.

Bischof C, Van Loan C (1987). "The WY Representation for Products of Householder Matrices." *SIAM Journal on Scientific and Statistical Computing*, **8**(1), s2–s13. doi:10.1137/0908009.

Breiman L (1996). "Bagging Predictors." *Machine Learning*, **24**, 123–140. doi:10.1007/bf00058655.

Breiman L (1999). "Using Adaptive Bagging to Debias Regressions." *Technical Report 547*, Department of Statistics, UC Berkeley.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**, 477–505. doi:10.1214/07-sts242.

Bühlmann P, Yu B (2003). "Boosting with the L_2 Loss: Regression and Classification." *Journal of the American Statistical Association*, **98**, 324–339. doi:10.1198/016214503000125.

Bühlmann P, Yu B (2006). "Sparse Boosting." *Journal of Machine Learning Research*, **7**, 1001–1024.

Buja A, Hastie T, Tibshirani R (1989). "Linear Smoothers and Additive Models." *The Annals of Statistics*, **17**, 453–510. doi:10.1214/aos/1176347115.

Cleveland WS, Devlin S (1988). "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting." *Journal of the American Statistical Association*, **83**, 596–610. doi:10.1080/01621459.1988.10478639.

Cornillon PA, Hengartner N, Jégou N, Matzner-Løber E (2013). "Iterative Bias Reduction: a Comparative Study." *Statistics and Computing*, **23**, 777–791. doi:10.1007/s11222-012-9346-4.

Cornillon PA, Hengartner N, Lefieux V, Matzner-Løber E (2015). "Fully Nonparametric Short Term Forecasting Electricity Consumption." In *Lecture Notes in Statistics: Modeling and Stochastic Learning for Forecasting in High Dimension*. Springer-Verlag.

Cornillon PA, Hengartner N, Matzner-Løber E (2014). "Recursive Bias Estimation for Multivariate Regression Smoothers." *ESAIM: Probability and Statistics*, **18**, 483–502. doi:10.1051/ps/2013046.

Cornillon PA, Hengartner N, Matzner-Lober E (2017). ***ibr**: Iterative Bias Reduction.* R package version 2.0-3, URL https://CRAN.R-project.org/package=ibr.

Craven P, Wahba G (1978). "Smoothing Noisy Data With Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation." *Numerische Mathematik*, **31**, 377–403. doi:10.1007/bf01404567.

de Silanes MCL, Arcangéli R (1989). "Estimations de l'erreur d'approximation par splines d'interpolation et d'ajustement d'ordre (M, s)." *Numerische Mathematik*, **56**, 449–467. doi:10.1007/bf01396648.

Duchon J (1977). "Splines Minimizing Rotation-Invariant Semi-Norms in Solobev Spaces." In W Shemp, K Zeller (eds.), *Construction Theory of Functions of Several Variables*, pp. 85–100. Springer-Verlag, Berlin.

Eubank R (1988). *Spline Smoothing and Nonparametric Regression.* Marcel Dekker, New-York.

Fan J, Gijbels I (1996). *Local Polynomial Modeling and Its Application, Theory and Methodologies.* Chapman et Hall, New York.

Friedman J (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics*, **28**(337-407). doi:10.1214/aos/1013203451.

Friedman J, Hastie T, Tibshirani R (2000). "Additive Logistic Regression: A Statistical View of Boosting." *The Annals of Statistics*, **28**, 337–407. doi:10.1214/aos/1016120463.

Golub GH, Van Loan CF (1996). *Matrix Computations.* 3rd edition. The Johns Hopkins University Press.

Gu C (2002). *Smoothing Spline ANOVA Models.* Springer-Verlag, New-York. doi:10.1007/978-1-4757-3683-0.

Hansen M, Yu B (2001). "Model Selection and Minimal Description Length Principle." *Journal of the American Statistical Association*, **96**, 746–774. doi:10.1198/016214501753168398.

Hastie T, Tibshirani R (1990). *Generalized Additive Models.* Chapman & Hall, London.

Hastie T, Tibshirani R, Friedman J (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer-Verlag, New-York.

Hastie T, Tibshirani R, Leisch F, Hornik K, Ripley BD (2016). ***mda**: Mixture and Flexible Discriminant Analysis.* R package version 0.4-9, URL https://CRAN.R-project.org/package=mda.

Herrmann E, Maechler M (2016). ***lokern**: Kernel Regression Smoothing with Local or Global Plug-in Bandwidth.* R package version 1.1-8, URL https://CRAN.R-project.org/package=lokern.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2016). *Model-Based Boosting.* R package version 2.7-0, URL https://CRAN.R-project.org/package=mboost.

Hurvich C, Simonoff J, Tsai CL (1998). "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion." *Journal of the Royal Statistical Society B*, **60**, 271–294. doi:10.1111/1467-9868.00125.

Marzio MD, Taylor C (2008). "On Boosting Kernel Regression." *Journal of Statistical Planning and Inference*, **138**, 2483–2498. doi:10.1016/j.jspi.2007.10.005.

Miller DL, Wood SN (2014). "Finite Area Smoothing with Generalized Distance Splines." *Environmental and Ecological Statistics*, **21**(4), 715–731. doi:10.1007/s10651-014-0277-4.

Nason G (2016). ***wavethresh**: Wavelets Statistics and Transforms.* R package version 4.6.8, URL https://CRAN.R-project.org/package=wavethresh.

R Core Team (2017). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ridgeway G (2000). "Discussion of "Additive Logistic Regression: A Statistical View of Boosting"." *The Annals of Statistics*, **28**, 393–400. doi:10.1214/aos/1016120463.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**, 461–464. doi:10.1214/aos/1176344136.

Simonoff J (1996). *Smoothing Methods in Statistics.* Springer-Verlag, New York. doi:10.1007/978-1-4612-4026-6.

Tukey J (1977). *Exploratory Data Analysis.* Addison-Wesley.

Wendelberger J (1982). *Smoothing Noisy Data with Multivariate Splines and Generalized Cross-Validation.* Ph.D. thesis, University of Wisconsin.

Wood S (2003). "Thin Plate Regression Splines." *Journal of the Royal Statistical Society B*, **65**, 95–114. doi:10.1111/1467-9868.00374.

Wood S (2017). ***mgcv**: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation.* R package version 1.8-17, URL https://CRAN.R-project.org/package=mgcv.

Affiliation:

Pierre-André Cornillon, Eric Matzner-Løber
IRMAR – UMR 6625 CNRS
Université Rennes 2 – CS 24307
35043 Rennes CEDEX, France
E-mail: pierre-andre.cornillon@univ-rennes2.fr, eric.matzner@univ-rennes2.fr
URL: http://perso.univ-rennes2.fr/pierre-andre.cornillon
http://perso.univ-rennes2.fr/eric.matzner

Nick Hengartner
Los Alamos National Laboratory
E-mail: nickh@lanl.gov
URL: http://www.ccs3.lanl.gov/group/people/?id=nickh

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2017, Volume 77, Issue 9 *Submitted:* 2013-12-31
doi:10.18637/jss.v077.i09 *Accepted:* 2016-01-27