Package wsbackfit for Smooth Backfitting Estimation of Generalized Structured Models

by Javier Roca-Pardiñas, María Xosé Rodríguez-Álvarez and Stefan Sperlich

Abstract A package is introduced that provides the weighted smooth backfitting estimator for a large family of popular semiparametric regression models. This family is known as *generalized structured models*, comprising, for example, generalized varying coefficient model, generalized additive models, mixtures, potentially including parametric parts. The kernel-based weighted smooth backfitting belongs to the statistically most efficient procedures for this model class. Its asymptotic properties are well-understood thanks to the large body of literature about this estimator. The introduced weights allow for the inclusion of sampling weights, trimming, and efficient estimation under heteroscedasticity. Further options facilitate easy handling of aggregated data, prediction, and the presentation of estimation results. Cross-validation methods are provided which can be used for model and bandwidth selection.[1]

1 Introduction and brief review

The classes of generalized structured models (GSM) of Mammen and Nielsen (2003), structured additive regression models (Brezger et al., 2005), and semiparametric separable models (Rodríguez-Poó et al., 2003) are all devoted to harmonizing the fundamental aspects of flexibility, dimensionality and interpretability (c.f. also Stone, 1986) for multidimensional regression. In some cases, the particular structure is derived from pure theory, sometimes from empirical knowledge, or it is chosen data-adaptively. The epithet 'structured' underlines the explicit modeling of the structure of a regression in order to distinguish it from fully automatic black-box regression or prediction. Mammen and Nielsen (2003) define for response Y with covariate vectors $(\mathbf{Z}, \mathbf{X}, \mathbf{T}, \mathbf{U})$ the GSM class by

$$\Lambda(Y) = G\left\{\mathbf{Z}, \beta, g(\mathbf{X})\right\} + S\left\{\mathbf{T}, \delta, s(\mathbf{U})\right\} \epsilon = G\left\{\mathbf{Z}, \beta, g(\mathbf{X})\right\} + \varepsilon \ , \tag{1}$$

with Λ, G, S parametric known functions, β, δ unknown finite-dimensional parameter, $g(\cdot)$, $s(\cdot)$ unknown nonparametric functions, and ϵ, ε fulfilling $E[\epsilon|\mathbf{Z}, \mathbf{X}] = E[\varepsilon|\mathbf{Z}, \mathbf{X}] = 0$. While Λ is a transformation with potentially unknown parts which can be estimated along Linton et al. (2008), G and S are link functions that also determine further structures. For instance, for a partial linear varying coefficient model (Park et al., 2015) with $\mathbf{Z} = (Z_1, \cdots, Z_d, \mathbf{Z}_\kappa)$, $\mathbf{X} = (X_1, \cdot, X_d)$, where Z_1 to Z_d and X_1 to X_d are scalars, and $\mathbf{Z}_\kappa$ a vector of the length of β, function G defines

$$G\left\{\mathbf{Z}, \beta, g(\mathbf{X})\right\} = \underline{G}\left\{\eta(\mathbf{Z}, \beta, g(\mathbf{X}))\right\} = \underline{G}\left\{g_0 + \sum_{j=1}^{d} g_j(X_j) Z_j + \mathbf{Z}_\kappa^t \beta\right\} \ , \tag{2}$$

with index η and a known link function $\underline{G}$. You may also allow that some, or all of the X_j, $j = 1, \ldots, d$ are identical; the same holds for the Z_j, etc. Moreover, by setting $Z_j \equiv 1 \ \forall j$ with all X_j being different, you obtain the generalized additive model (GAM). A detailed discussion on identifiability is provided in Lee et al. (2012).

Since Hastie and Tibshirani (1990) introduced their backfitting algorithm, additive models have become quite popular in statistics, particularly in biometrics, technometrics, and environmetrics. Opsomer and Ruppert (1997) and Opsomer (2000) derived asymptotic theory for that classical backfitting estimator with kernel smoothing. Mammen et al. (1999) developed asymptotic theory for a modified version, the smooth backfitting (SB) estimator, under weaker assumptions on the data like the allowance for strong correlation of the covariates. Mammen and Nielsen (2003) extended this method to the general GSM class (1), and Roca-Pardiñas and Sperlich (2010) proposed a common algorithm for it. Many extensions have been developed, procedures for bandwidth selection (e.g., Mammen and Park, 2005), quantile regression (Lee et al., 2010), and further asymptotic theory for particular cases (see e.g., Yu et al., 2008, for GAM). Most recent contributions extend SB to additive

[1] The second author acknowledges the support received by the Basque Government through the BERC 2018-2021 program and Elkartek project 3KIA (KK-2020/00049), by the Spanish Ministry of Science, Innovation and Universities through BCAM Severo Ochoa accreditation SEV-2017-0718 and through project MTM2017-82379-R funded by (AEI/FEDER, UE). The third author acknowledges financial support from the Swiss National Science Foundation, project 200021-192345.

inverse regression (Bissantz et al., 2016), proportional hazards (Hiabu et al., 2020), or regression with error-in-variables (Han and Park, 2018). All SB procedures and their theory are kernel-based.

The main advantage of SB is, apart from its excellent numerical performance proven by Nielsen and Sperlich (2005) and Roca-Pardiñas and Sperlich (2010), compared to the classical backfitting, that there exists a comprehensive literature that studies its statistical behavior and underlying assumptions. It provides the exact and complete asymptotic theory of SB, such that today this estimator is well understood. The only drawback has been that so far, there hardly existed an easily available software for this estimator, except the R-package **sBF** of Arcagni and Bagnato (2014) for the basic additive model. But due to its complexity, practitioners typically abstain from implementing it themselves. Therefore, the **wsbackfit** R-package has been developed which provides the weighted SB for all models listed in the next section, including a data-driven bandwidth selector. The package is freely available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=wsbackfit` (R Core Team, 2016). Thus, this package closes the gap between the huge body of existing and still increasing literature about SB on the one side, and its potential use on the other, providing the necessary software. We hope it is soon extended by procedures for the various, partly above cited, extensions.

It is to be mentioned that there certainly exist R packages for alternative methods to estimate related models. Before briefly discussing some of the most advanced packages, let us mention the reviews of Binder and Tutz (2008), which reviewed spline-based methods, and Fahrmeier et al. (2004) which reviewed (spline-based) Bayesian methods.

Maybe the broadest set of models can be handled by the package **BayesX** (Umlauf et al., 2019). It embraces several well-known regression models such as GAM, generalized additive mixed models (GAMM), generalized geo-additive mixed models (GGAMM), dynamic models, varying coefficient models (VCM), and geographically weighted regression. Besides exponential family regression, **BayesX** also supports non-standard regression situations such as regression for categorical responses, hazard regression for continuous survival times, and continuous-time multi-state models; see also its support platform `http://www.uni-goettingen.de/de/bayesx`. It has been created by Brezger et al. (2005) and Kneib et al. (2008).

The R package **gam** (Hastie, 2019) presents considerable enhancements of the S-PLUS version going back to Hastie and Tibshirani (1990). It uses classical backfitting to combine different smoothing or fitting methods, particularly local regression and smoothing splines. Another powerful package is **mgcv** (Wood, 2017), which allows the fitting of generalized additive (mixed) models, with smoothing parameter estimation done by (restricted) marginal likelihood or generalized cross-validation, and uses iterated nested Laplace approximation for fully Bayesian inference. Another powerful and well-functioning package is **GAMLSS** of Stasinopoulos and Rigby (2007). It is based on penalized likelihood estimation combined with classical backfitting. While **mgcv** models the index function, **GAMLSS** models the location, scale, and shape functions by additive linear mixed models. It has been created to tackle many interesting distributions of Y. When speaking of likelihood based approaches, one should also mention a method introduced by Tutz and Binder (2006). Their R-package **GAMBoost** can be used to fit a GAM by likelihood based boosting, suited for a large number of predictors.

Regarding kernel-based methods that consider related or specific cases of (1), there are, for example, marginal integration (Linton and Nielsen, 1995) for additive interaction models (Sperlich et al., 2002), and local polynomials for smooth varying coefficients (Li and Racine, 2010). The latter is implemented in the **np** package (Hayfield and Racine, 2008), and turned out to be very competitive when compared to the before-mentioned spline-based packages (Sperlich and Theler, 2015).

2 The models that can be estimated by wsbackfit

The aim is to estimate a GSM as introduced in (1). In the moment of estimation, one has to be specific about Λ, G, and S. We concentrate on the popular cases, in particular on those that maintain additivity or a similar separability structure. This way, the estimates provide an easy interpretation, and overcome the curse of dimensionality, which else is inherited by more complex models. To the best of our knowledge, all existing smooth backfitting methods follow the suggestion of Mammen and Nielsen (2003) to estimate the mean and variance part subsequently, say, first $G\{\cdots\}$, then $S\{\cdots\}$. Our implementation follows the suggestion of Roca-Pardiñas and Sperlich (2010) to allow for a (re-)estimation of the mean part ($G\{\cdots\}$) including weights obtained from the estimation of the variance part ($S\{\cdots\}$) to potentially increase the efficiency. Note, however, that in nonparametrics, it is often not clear to what extent an efficiency gain can be achieved this way; see for example the discussion in Xiao et al. (2003). All proposed methods we are aware of, are two- or three-step procedures similar to what we propose here. The estimation of the variance part is performed in the second step by regressing the squared residuals on $(\mathbf{T}, \mathbf{U})$, using the same procedure as for $G\{\cdots\}$. Therefore it is sufficient to concentrate in the following on the mean regression, which can equally well be applied to squared residuals for estimating $S\{\cdots\}$.

As said, the SB idea for GSM, together with some general results about its asymptotic behavior, was introduced by Mammen and Nielsen (2003); specific algorithms and their implementation were introduced and studied in Roca-Pardiñas and Sperlich (2010). Detailed information about the implementation is given in a technical report (Roca-Pardiñas and Sperlich, 2008). The implemented algorithms in **wsbackfit** are modified versions to speed up the procedure by binning techniques and a combination of parametric linear with local constant kernel regression; see below. The models considered in this package are semiparametric in the sense that they contain parametric as well as nonparametric components. Most of them could be seen as extensions of a generalized linear model (GLM) of type $G(\mathbf{Z}, \beta, g(\mathbf{X})) = \underline{G}\big(\eta(\mathbf{Z}, \beta, g(\mathbf{X}))\big) = \underline{G}(g_0 + \alpha^t \mathbf{X} + \beta^t \mathbf{Z})$, see for example McCullagh and Nelder (1989). So far, this package does not tackle random effects.

Regarding the choice of G, you have first to decide about the link $\underline{G}$. For each conditional distribution, there exists a canonical one: for the conditional Gaussian distribution, this is the identity. For a binary response, it is the Logit $(1 + 1/exp(\bullet))^{-1}$, and for a conditional Poisson it is $exp(\bullet)$. Note that the latter can also be used for Pseudo-Poisson estimation. The choice of $\underline{G}$ is certainly linked to the specification of Λ which is supposed to be known. Then, such transformation of Y can be performed a priori by the practitioner. Therefore, we henceforth suppress Λ, to simplify our notation. For Λ entailing unknown parameters, consult Linton et al. (2008).

Roca-Pardiñas and Sperlich (2010) showed that the estimation procedure for all these models can be summarized in one common feasible minimization problem, namely

$$minimize \quad \int \sum_{i=1}^{n} \left[\tilde{Y}_i - \eta\{\mathbf{Z}_i, \beta, g(\mathbf{x})\}\right]^2 W_i \cdot K_h(\mathbf{x} - \mathbf{X}_i)\ d\mathbf{x}\ , \tag{3}$$

where $\tilde{Y}_i$ is the transformed (e.g. by Λ) or linearized (in local scoring if the link is not the identity) response Y_i, and W_i is a weight. For example, in the generalized additive model with $\beta = 0$ we have covariates $Z_j \equiv 1$ for $j = 1, \ldots, d$, W_i contains the local scoring weights with $\tilde{Y}_i$ being the accordingly adjusted dependent variable. Further, $K_h(v) = h^{-1}K(v/h)$ with $K(\cdot)$ is the kernel function. It is well known that asymptotically, the choice of smoothing kernel does not have an important impact, as to a large part the kernel effect is compensated by an adequate bandwidth choice. We allow the user to choose between the Epanechnikov kernel which is asymptotically the most efficient one, and the Gaussian kernel which is popular as it helps to avoid some of the numerical problems that may arise in areas where data are sparse.

We call our procedure ‘weighted smooth backfitting’ to emphasize that the user has the option to include a vector of additional weights. As said, by putting the usual kernel weights apart, part of the weighting comes from local scoring in order to account for the link function $\underline{G}$.[2] However, independently from the link function, the practitioner might also want to include sampling weights, e.g., when using administrative data, or trimming weights, e.g., for excluding boundary points. A particular case is when additional weights are included to improve the efficiency of your estimators, e.g., to account for the (co-)variance structure. Roca-Pardiñas and Sperlich (2010) estimated in a first step the mean function, afterward the variance from the squared residuals, and used these in the third step as additional weights when re-estimating the conditional mean. The resulting average mean squared error was substantially smaller than the one of the original estimator, which ignored the (co-)variance structure; recall our discussion at the beginning of this section.

The models that package **wsbackfit** can presently estimate are: a partial linear GAM, a generalized partial linear varying coefficient model (GVCM), and combinations of them. The first one is a generalization of a GLM by replacing some linear components with additive nonparametric functions,

$$E\left[Y|\mathbf{X}, \mathbf{Z}\right] = \underline{G}\left(g_0 + \sum\nolimits_{j=1}^{d} g_j\left(X_j\right) + \beta^t \mathbf{Z}\right)\ ,$$

where X_1 to X_d are scalars, and $\mathbf{Z}$ is the vector of all covariates that are supposed to enter the index function η linearly. The g_j are nonparametric functions.[3] It is actually true that this is a special case of the partial linear GVCM of the form (2), obtained by setting $Z_1 = Z_2 = \cdots = Z_d = 1$. We list them nonetheless separately because, besides the slightly different implementation, we want the reader to recognize the difference in the modeling approach. First, the GVCM is a generalization of

[2] Yu et al. (2008) propose a somewhat different algorithm for GAMs replacing local scoring by an alternative that makes asymptotic theory simpler.

[3] For identification we follow Lee et al. (2012), and Roca-Pardiñas and Sperlich (2010) for implementation with modifications like the inclusion of a parametric linear slope for each g_j, see below.

a GLM in $\mathbf{Z}$, as could be seen in (2). And second, here we allow for all the flexibility suggested by Lee et al. (2012) regarding the alterations of covariates X_j and Z_j. For example, all $X_1, \ldots, X_{d_1}$ with $d_1 \leq d$ could be the same scalar variable X_1 such that

$$\sum_{j=1}^{d_1} g_j\left(X_j\right) Z_j = \sum_{j=1}^{d_1} \left\{g_{j0} + g_{j1}\left(X_1\right)\right\} Z_j \ , \tag{4}$$

with g_{j0} unknown constants and g_{j1} unknown nonparametric functions; or, alternatively, all $Z_1, \ldots, Z_{d_1}$ with $d_1 \leq d$ could be the same scalar variable Z_1 such that (with g_{10} still a constant)

$$\sum_{j=1}^{d_1} g_j\left(X_j\right) Z_j = \left\{g_{10} + \sum_{j=1}^{d_1} g_{j1}\left(X_j\right)\right\} Z_1 \ . \tag{5}$$

Certainly, you can have a mixture of both, as long as the identification conditions of Lee et al. (2012) are fulfilled to guarantee that the model does not suffer from concurvity. This includes the possibility that some variables appear in both sets, $\mathbf{X}$ and $\mathbf{Z}$. This could be of particular interest when defining different types of interactions.

Finally, one can include all together, i.e., nonparametric additive terms, nonparametric varying coefficients, and a parametric (linear) part like

$$E\left[Y|\mathbf{X},\mathbf{Z}\right] = \underline{G}\left\{g_0 + \sum_{j=1}^{d_1} g_j\left(X_j\right) Z_j + \sum_{j=d_1+1}^{d} g_j\left(X_j\right) + \beta^t \mathbf{Z}_\kappa\right\} \ , \tag{6}$$

with $\mathbf{X}$ as before, $\mathbf{Z} = (Z_1, \ldots, Z_{d_1}, \mathbf{Z}_\kappa)$ a set of scalar variables $Z_1, \ldots, Z_{d_1}$, and a vector $\mathbf{Z}_\kappa$. Again, some of the X_j may represent the same variable; the same holds for the Z_j, $j = 1, ..., d$.

3 Cross-validation, bandwidths, and computational issues

Cross-validation (CV) can be used for model selection in general. However, for the sake of presentation we describe here our implementation in the context of bandwidth selection.

Cross-validation for bandwidth

All nonparametric estimates of the $g_j(X_j)$ in (6) depend on some bandwidths $h_1, \ldots, h_d$, which can be preset by the user. Alternatively, the package provides the option to choose the bandwidths data-adaptively via CV. Our implementation even allows for a mixture of both, i.e., users can fix some bandwidths and choose the others by CV. Albeit we use binning techniques, performing CV can render the program pretty slow, especially for high dimensions and huge data sets. Generally, our implementation follows the ideas of Nielsen and Sperlich (2005) and Roca-Pardiñas and Sperlich (2010). It is to be mentioned that several alternatives exist, in particular for the additive model. For instance, Mammen and Park (2005) proposed bandwidths selectors based on penalized least squares and plug-in approaches.

Given sample $\{\mathbf{X}_i, \mathbf{Z}_i, Y_i\}_{i=1}^n$, bandwidths $h_1, \ldots, h_d$ can be selected by minimizing some CV criterion in various ways. Allowing for limited dependent variables Y, the deviance is an appropriate measure of discrepancy between observed and fitted values. It is derived as a likelihood ratio test comparing the specified model with a so-called saturated one, when predicted values match the observed responses exactly. More specifically, denoting the fitted mean response given by $\hat{\mu}_i = \hat{E}\left[Y_i|\mathbf{X_i},\mathbf{Z_i}\right]$, the deviance is given by $\text{Dev} = \sum_{i=1}^n \text{Dev}_i(Y_i, \hat{\mu}_i)$. The definition of the individual deviance Dev_i depends on the link; namely

	$\text{Dev}_i(Y_i, \hat{\mu}_i)$
Gaussian	$(Y_i - \hat{\mu}_i)^2$
Binary	$-2\left(Y_i \log \hat{\mu}_i + (1 - Y_i)\log(1 - \hat{\mu}_i)\right)$
Poisson	$Y_i \log \frac{Y_i}{\hat{\mu}_i} - (Y_i - \hat{\mu}_i)$

Generally spoken, unless bandwidths are fixed by the user, they can be selected as

$$(h_1, \ldots, h_d) = \underset{(h_1^\bullet, \ldots, h_d^\bullet)}{\arg\min} \sum_{i=1}^n \text{Dev}_i\left[Y_i, \underline{G}\left(\hat{\eta}_{\mathbf{X}_i,\mathbf{Z}_i}^{(-i)}\right)\right] , \tag{7}$$

with

$$\hat{\eta}_{\mathbf{X}_i,\mathbf{Z}_i}^{(-i)} = \hat{g}_0^{(-i)} + \sum_{j=1}^{d_1} \hat{g}_j^{(-i)}\left(X_{ij}\right) Z_{ij} + \sum_{j=d_1+1}^{d} \hat{g}_j^{(-i)}\left(X_{ij}\right) + \hat{\beta}^{t(-i)} \mathbf{Z}_{i\kappa} \ , \tag{8}$$

where $\hat{g}_j^{(-i)}(X_{ij})$ indicates the fit at X_{ij} leaving out the ith data point based on the smoothing parameter $h_j^{\bullet}$. One option to solve the minimization in (7) is to use a complete bandwidth selection that allows for all possible bandwidths combinations for the different covariates X_j. When the number of covariates X_j is large, the computational cost becomes very high or even prohibitive.

In order to simplify the problem, we provide the following three options to the user:

1. the user just prefixes all bandwidths that shall be used as final bandwidths;
2. the user prefixes starting values for the bandwidths, say $\tilde{h}_j$, and searches via CV for the optimal bandwidth vector $(h_1, \ldots, h_d)$ with a common bandwidth factor $c_h \in R$ such that $(h_1, \ldots, h_d) = c_h(\tilde{h}_1, \ldots, \tilde{h}_d)$;
3. the user only prefixes a bandwidth grid for a scalar h_c such that $(h_1, \ldots, h_d) = h_c(\sigma_1, \ldots, \sigma_d)$ with σ_j being the standard deviation of X_j, and h_c is chosen from the grid via CV.

The choice of prior $\tilde{h}_j$ typically follows some considerations of marginal distributions or marginal smoothing. For example, you could first perform a CV bandwidth choice for each nonparametric g_j by setting all other $g_{k \neq j}$ to zero, or by restricting them to be linear functions. The second method follows the ideas of Han and Park (2018), and the third follows a standard recommendation in the literature, see the review of Köhler et al. (2014). Combining options 1 and 3 is possible.

For the sake of presentation, we explain more details about the CV implementation only along option 3; as for option 2, it works analogously. Moreover, suppose the user chooses all bandwidths by option 3. As said, in option 3, $h_j = h_c\sigma_j$. While h_c might be different for each j if h_j is set by the user or chosen by option 2, in option 3, it is the same for all j. That is, we reduce the multidimensional search problem to a one-dimensional one. Specifically, if the user decides that all bandwidths are to be chosen by CV, $h_c := h_j/\sigma_j$ for all j, with

$$h_c = \arg\min_{h^{\bullet}} \sum_{i=1}^{n} \mathrm{Dev}_i \left[Y_i, \underline{G}\left(\hat{\eta}^{(-i)}_{\mathbf{X}_i,\mathbf{Z}_i} \right) \right], \tag{9}$$

where $\eta^{(-i)}_{\mathbf{X}_i,\mathbf{Z}_i}$ indicates the fitted additive predictor at $\{\mathbf{X}_i, \mathbf{Z}_i\}$ (see (8)) leaving out the ith data point, and based on the smoothing parameters $h^{\bullet}\sigma_j$, $(j = 1, \ldots, d)$.

Unfortunately, a naive implementation of the leave-one-out CV technique would still imply a high computational cost as for each potential value of $h^{\bullet}$, it is necessary to repeat the estimation as many times as we have data points. To speed up the process, the **wsbackfit** package uses k-fold CV instead. In brief, k-fold CV consists of randomly splitting the available sample into k complementary subsamples of (approximately) the same size such that each data point only belongs to one of the k subsamples, say $\kappa(i)$. Then, the k-fold CV version of (9) is

$$h_c = \arg\min_{h^{\bullet}} \sum_{i=1}^{n} \mathrm{Dev}_i \left[Y_i, \underline{G}\left(\hat{\eta}^{(-\kappa(i))}_{\mathbf{X}_i,\mathbf{Z}_i} \right) \right], \tag{10}$$

where $\eta^{(-\kappa(i))}_{\mathbf{X}_i,\mathbf{Z}_i}$ indicates the fitted additive predictor at $\{\mathbf{X}_i, \mathbf{Z}_i\}$ computed with the $\kappa(i)$ subsample removed. In contrast to leave-one-out CV, in k-fold CV, you repeat the estimations only k times, leaving-out one different subsample each time.

We conclude with two remarks. First, as said, the **wsbackfit** package also allows the user to specify the bandwidths h_j for some nonparametric functions, which are therefore treated as given in (10)), while letting the CV procedure select the others along option 3. A combination of option 2 with the others is not implemented. Examples can be found in Sections Package description and Examples and applications. Second, the minimum in (10) is determined by a grid search. The grid for the $h^{\bullet}$ (option 3) is by default `seq(0.01,0.99,length = 30)` but can optionally be set by the user via option `bw.grid`, see below. For option 2, the algorithm looks for the optimal c_h on an equispaced grid from 0.5 to 1.5.

Convergence Criteria

As explained above, smooth backfitting is solved by an iterative procedure to solve (3). When the link function is the identity, then there is only the loop running over the different additive components. If the link is not the identity, then there is also an outer loop carrying out the local scoring iteration. Then, within each of such outer iteration steps, the formerly mentioned loop of the smooth backfitting is conducted. Both loops are triggered by two factors, the tolerance *tol* in deviations between subsequent iterations and the maximum number *maxit* of iterations conducted. The defaults for the maximum number of iterations and the tolerance of deviation are *maxit* $= 10$ and *tol* $= 0.01$, respectively.

The inner loop stops when for iteration l, the updated $\hat{g}_j^l(\cdot)$ comply with criterion

$$\frac{\sum_{i=1}^n \left(\hat{g}_j^l(X_{ij}) - \hat{g}_j^{l-1}(X_{ij})\right)^2}{\sum_{i=1}^n \hat{g}_j^{l-1}(X_{ij})^2} \leq tol \quad \text{for} \quad j = 1, \ldots, d \ ,$$

where the g_j^{l-1} refer to the estimates of the previous iteration. Similarly, the outer loop stops when for iteration k, the convergence criterion

$$\text{Dev} = \frac{\sum_{i=1}^n \text{Dev}_i(Y_i, \hat{\mu}_i^{k-1}) - \text{Dev}_i(Y_i, \hat{\mu}_i^k)}{\sum_{i=1}^n \text{Dev}_i(Y_i, \hat{\mu}_i^{k-1})} \leq tol \ ,$$

is met, where $\hat{\mu}_i^k$ and $\hat{\mu}_i^{k-1}$ are the estimates of μ_i obtained in the present iteration and in the previous one $(k-1)$, respectively.

Binning and integration

Although we implemented some modifications and simplifications like the above described k-fold CV, or the combination of parametric linear with local constant estimation, for details see the Section Identification, SB in high dimensions might still imply a high computational cost. Therefore, as already indicated above, we implemented the kernel smoothing inside the **wsbackfit** package using binning-type procedures. These are used throughout, also for CV when selecting bandwidths. The key idea of this binning is to reduce the number of kernel evaluations (exploiting the fact that many of these are nearly identical) by replacing the original data set (composed of n data points) with a 'grouped' data set (with N groups as new data points with sampling weights, where $N << n$). The estimation is carried out on these N groups, including the sampling weights in W_i. For a detailed description of binning for kernel regression, see Fan and Gijbels (1996).

Note that for minimizing (3), we need to solve some univariate integrals over nonparametric estimates which is therefore done numerically. This is calculated by the Simpson rule with 51 equidistant grid points over the entire range of the respective covariate, i.e., from its smallest to the largest observation. Simulations showed that finer grids led to no improvement of the final estimates.

Identification

When we introduced the class of GSM in Section The models that can be estimated by wsbackfit, we restricted our presentation to models that can be estimated by the here introduced package. At this stage – note that the package design invites further contributions of SB-based methods – the package is able to estimate model (6) only for some link functions, and all g_j being one-dimensional nonparametric functions. Lee et al. (2012) discuss identification of model (6) in a very general way, also allowing all g_j to be multidimensional, and covariates X_j being overlapping vectors (i.e., containing, at least partly, the same elements), and possibly also containing some elements of the $\mathbf{Z}$ covariate vector. Essentially, they clarify which overlaps would render a model unidentifiable. As such discussion is quite technical and would go beyond the slope of this paper, we only refer to them.

Now recall models (4) and (5), which probably represent the most common cases in practice. For these models, consider the identification of the g_j with respect to location and scale. Each g_j in model (6) has been decomposed into a linear effect $\alpha_j \cdot X_j$ together with a purely nonparametric (beyond the linear) one, say $\tilde{g}_j$. In addition, for g_j being varying coefficients, we have included constants g_{j0}, $j = 1, \ldots, d_1$. Then, we can re-write model (6) as

$$\underline{G}\left\{g_0 + \sum_{j=1}^{d_1} \left(g_{j0} + \alpha_j \cdot X_j + \tilde{g}_j\left(X_j\right)\right) Z_j + \sum_{j=d_1+1}^{d} \left(\alpha_j \cdot X_j + \tilde{g}_j\left(X_j\right)\right) + \beta^t \mathbf{Z}_\kappa\right\} . \qquad (11)$$

For this model, we set $E[\tilde{g}_j(X_j)] = 0$ $(j = 1, \ldots, d)$ and $E[X_j \cdot \tilde{g}_j(X_j)] = 0$ $(j = 1, \ldots, d_1)$. Apart from identification issues, this prevents us from biases in the linear direction, i.e., in the slope of the g_j, such that we can estimate the $\tilde{g}_j$ by local constant SB speeding up the algorithm significantly. Finally, this reparametrization helps us to see how to choose the starting values for the iterative backfitting procedure. For the first step, you can simply start with a parametric GLM estimator, setting $\tilde{g}_j \equiv 0$ for all j. Then, for $\hat{g}_0$, $\hat{\beta}$, $\hat{g}_{j,0}$, and $\hat{\alpha}_j$, $j = 1, \ldots, d$ obtained from that GLM estimation, you proceed estimating the $\tilde{g}_j$ as outlined in Roca-Pardiñas and Sperlich (2010), to afterward update all estimates via iteration.

4 Package description

The package is composed of several functions that enable users to fit the models with the methods described above. Table 1 provides a summary of the presently available functions.

wsbackfit functions	
sback	Main function for fitting generalized structures models using smooth backfitting.
sb	Function used to indicate the nonparametric terms and varying coefficient terms in the sback() formula.
plot	Function that provides plots of sback objects produced by sback().
print	The default print method for an sback() object.
summary	Function that takes a fitted object produced by sback() and produces various useful summaries from it.
predict	Function that provides predictions (e.g., fitted values and nonparametric terms) based on an sback object for a new data set (newdata).
residuals	Returns residuals of sback objects produced by sback(). Deviance, Pearson, working and response residuals are available.
summary	Summary for objects of class sback.

Table 1: Summary of functions of **wsbackfit**.

The package has been designed similarly to other regression packages. The main function is sback. It fits GSM with SB, and creates an object of class sback. Numerical and graphical summaries of the fitted model can be obtained by using print, summary, and plot, implemented for sback objects. Moreover, function predict allows obtaining predictions (e.g., fitted values and nonparametric terms) for data different from those used for estimation, called, therefore, newdata. The main arguments of the sback function are listed in Table 2, and the list of outputs is given in Table 3.

sback arguments – input values	
formula	A formula object specifying the model to be fitted.
data	Data frame representing the data and containing all needed variables.
offset	An optional numerical vector containing a priori known components to be included in the linear predictor during fitting. Default is zero.
weights	An optional numeric vector of 'prior weights' to be used in the fitting process. By default, the weights are set to one.
kernel	A character specifying the kernel function. Implemented are: Gaussian and Epanechnikov. By default "Gaussian".
bw.grid	Numeric vector; a grid for searching the bandwidth factor h_c. The bandwidth for dimension j is $h_c\sigma_j$. Default is seq(0.01,0.99,length = 30)
c.bw.factor	logical; indicates whether the common factor scheme for bandwidth selection proposed by Han and Park (2018) is performed. If TRUE, and provided the user has specified the (marginal) bandwidths for all nonparametric functions, say $\tilde{h}_j$, the functions searches for the common factor c_h that minimizes the deviance via (k-fold) cross-validation when the bandwidth used for dimension (covariate) j is $c_h\tilde{h}_j$. The search is done in an equispaced grid of length 15 between 0.5 and 1.5. The default is FALSE.
KfoldCV	Number of cross-validation folds to be used for either (1) automatically selecting the optimal bandwidth (in the sequence given in argument bw.grid) for each nonparametric function; or (2) automatically selecting the optimal common bandwidth factor (see argument c.bw.factor). Default is 5.
kbin	An integer value specifying the number of binning knots. Default is 30.
family	A character specifying the distribution family. Implemented are: Gaussian, Binomial and Poisson. In all cases, the link function is the canonical one. By default "gaussian".

Table 2: Summary of the arguments of the main function sback.

We call function sback by

`sback` output values	
`call`	The matched call.
`formula`	The original supplied `formula` argument.
`data`	The original supplied `data` argument.
`weights`	The original supplied `weights` argument.
`offset`	The original supplied `offset` argument.
`kernel`	The original supplied `kernel` argument.
`kbin`	The original supplied `kbin` argument.
`family`	The original supplied `family` argument.
`effects`	Matrix with the estimated nonparametric functions (only the nonlinear component) for each covariate value in the original supplied data.
`fitted.values`	A numeric vector with the fitted values for the supplied data.
`residuals`	A numeric vector with the deviance residuals for the supplied data.
`h`	A numeric vector of the same length as the number of nonparametric functions, with the bandwidths used to fit the model.
`coeff`	A numeric vector with the estimated regression coefficients. This vector contains the estimates of the regression coefficients associated with the parametric part of the model (if present) as well as the linear components of the nonparametric functions.
`err.CV`	Matrix with the cross-validated error (deviance) associated with the sequence of tested bandwidths (those provided in argument `bw.grid` in function `sback`).

Table 3: Summary of output values of the `sback` function.

```
sback(formula, data, offset = NULL, weights = NULL,
  kernel = c("Gaussian", "Epanechnikov"),
  bw.grid = seq(0.01, 0.99, length = 30), c.bw.factor = FALSE,
  KfoldCV = 5, kbin = 30,
  family = c("gaussian", "binomial", "poisson"))
```

The argument `formula` corresponds to the model for the conditional mean function (6). This formula is similar to that used for `glm`, except that nonparametric functions can be added to the additive predictor by means of function `sb` (for details, see Table 4). For instance, specification `y ~ x1 + sb(x2,h = -1)` assumes a parametric effect of `x1` (with `x1` either numerical or categorical), and a nonparametric effect of `x2`. Varying coefficient terms get incorporated similarly. For example, `y ~ sb(x1,by = x2)` indicates that the coefficient(s) of `x2` depend nonparametrically on `x1`. In this case, both, `x1` and `x2` should be numerical predictors. More examples are provided further below.

`sb` arguments	
`x1`	The univariate predictor.
`by`	Numeric predictor of the same dimension as `x1`. If present, the coefficients of this predictor depend nonparametrically on `x1`, i.e., a varying coefficient term.
`h`	Bandwidth for this term. If `h = -1`, the bandwidth is automatically selected using k-fold cross-validation. A value of 0 would indicate a linear fit. By default `-1`.

Table 4: Summary of the arguments of the function `sb`.

The bandwidths associated with the nonparametric functions are specified inside `sb` through argument `h` (see Table 4), either as final bandwidth (by setting `h =` h_j; option 1), as starting value to be multiplied by an optimal constant c_h found via CV from an equispaced grid of length 15 between 0.5 and 1.5 (by setting `h =` $\tilde{h}_j$ and `c.bw.factor = TRUE`; option 2), or by demanding a CV-bandwidth which is the product of a common factor h_c (chosen from `bw.grid`) times the scale σ_j of covariate X_j (by setting `h = -1`; option 3); recall Section Cross-validation, bandwidths, and computational issues. Actually, the user has even four options. These are specified through argument `h` of function `sb`, where option 4 is a mixture of options 1 and 3 by setting `h =` h_j inside `sb` for some nonparametric functions, and `h = -1` for the others. The number k of CV folds is specified through `KfoldCV`, with 5 being the default.

In Section The models that can be estimated by wsbackfit, recall expression (3) with subsequent discussion, we saw that the SB algorithm is implemented with potentially adjusted dependent variable $\tilde{Y}$ and weights W. The arguments `offset` and `weights` allow the user to modify them on top of what is done automatically (e.g., due to the local scoring). More specifically, the vector `offset` is subtracted from $\tilde{Y}$ when estimating. A standard application is the log of exposure in Poisson regression; see also the example in Section Poisson regression with offset. Regarding the weight W, recall that parts of it are automatically calculated (and updated in each backfitting iteration) by the local scoring procedure to account for the given link function, which is 1 if the link is the identity. This is multiplied by the optional vector `weights` provided by the user. For an example in which this option is used to improve the efficiency of the estimators, see Section Gaussian simulated data.

The desired smoothing kernel, either Epanechnikov or Gaussian, is specified through the argument `kernel` (by default Gaussian). With `kbin`, the user indicates the number of binning knots (denoted as N in Section Cross-validation, bandwidths, and computational issues). The argument `family` specifies the conditional distribution of the response variable. So far, the user can select among Gaussian, Poisson, and binary. In all cases, the canonical link function is considered. Finally, recall that predictions for data different from those being used for estimation can be obtained by means of function `predict`, specifying the new dataset in argument `newdata`.

5 Simulation study

This section reports the results of a small simulation. Two different scenarios are considered, namely

Scenario I. Additive model. Covariates X_1 and X_2 are simulated independently from the uniform distributions on the intervals $[0,1]$ and $[-10,1.5]$, respectively, and

$$\eta = g_1(X_1) + g_2(X_2) = 2 + 3X_1^2 + 0.01X_2^3.$$

Here, Y is generated under two different distributions

- $Y = \eta + \varepsilon$, where $\varepsilon \sim N\left(0, 0.5^2\right)$.
- $Y \sim$ Bernoulli (p), with $p = \exp(\widetilde{\eta}) / \exp(1+\widetilde{\eta})$, where $\widetilde{\eta} = \eta/4$,

where the scaling factor in the Bernoulli case is used to control the signal-to-noise ratio.

Scenario II. Varying coefficient model. Here, covariates X_1, X_2, Z_1 and Z_2 are simulated independently from a uniform distribution on the interval $[0,1]$, and

$$\eta = g_1(X_1)Z_1 + g_2(X_2)Z_2 = 5\sin(2\pi X_1)Z_1 + X_2 Z_2.$$

As for Scenario I, Y is generated according to

- $Y = \eta + \varepsilon$, where $\varepsilon \sim N\left(0, 0.5^2\right)$.
- $Y \sim$ Bernoulli (p), with $p = \exp(\eta) / \exp(1+\eta)$.

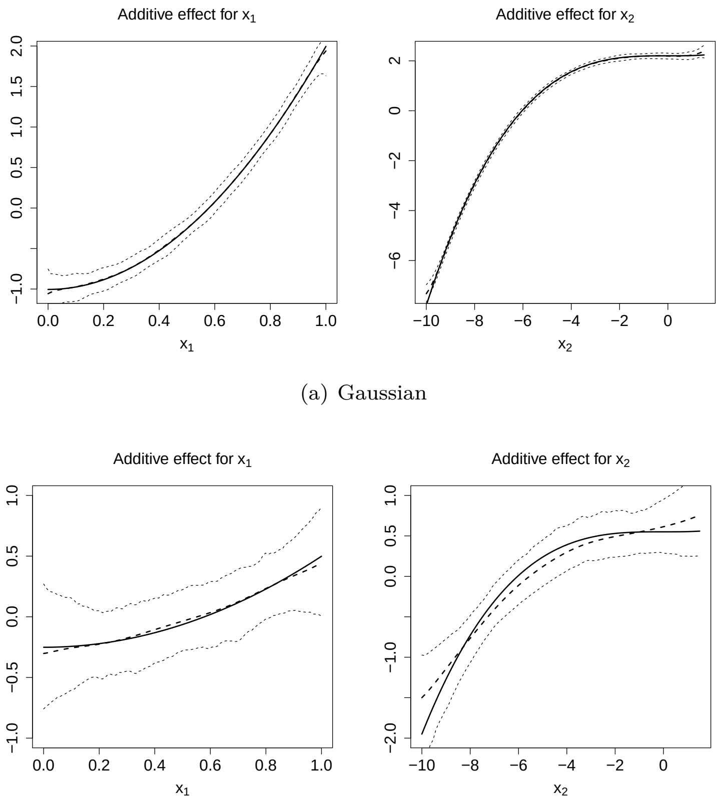
Results for Scenarios I and II are shown in Figure 1 and Figure 2, respectively. In both cases, the true and estimated functions are centered before plotting to make results comparable. All results are based on a sample size of $n = 500$ with $R = 500$ replicates. Also, we use the Gaussian kernel, 30 binning knots, and all bandwidths being selected using 5-fold cross-validation (option 3). We obtained essentially the same figures when we repeated these simulations with the Epanechnikov kernel. Not surprisingly, simulation results with options 1 and 2 for the bandwidth choice depended quite a bit on the setting of our (prior) bandwidths and are therefore not shown.

6 Examples and applications

The last section is dedicated to examples with generalized additive and/or varying coefficient, partial linear models with and without heteroscedasticity, given different link functions. In fact, we have examples for each of the three link functions presently available. Among other things, it is shown how the optional weighting can be used to improve efficiency. While some examples are simulated, others illustrate applications from biometrics and health. Finally, we also show how to create useful graphics for interpreting the estimates.

Gaussian simulated data

We start with the presentation of a simulation example for estimating an additive model under heteroscedasticity. Consider the situation where the variance is a function of a dummy variable,



(a) Gaussian

(b) Bernoulli

Figure 1: The simulation study and Scenario I. From left to right: true curve g_1 (solid line) and average estimate $\hat{g}_1$ (dashed line); true curve g_2 (solid line) and average estimate $\hat{g}_2$ (dashed line). In both cases, 2.5 and 97.5 simulation quantiles are plotted. Top row: Gaussian distribution. Bottom row: Bernoulli distribution.

i.e., one faces two noise levels. This is estimated and afterward used to improve the efficiency of the mean regression. As explained in the above sections, this requires a three-step procedure: first estimate the mean model, then the variance function, and finally re-estimate the mean model but including the inverse of the variance as an additional weight. Consider model

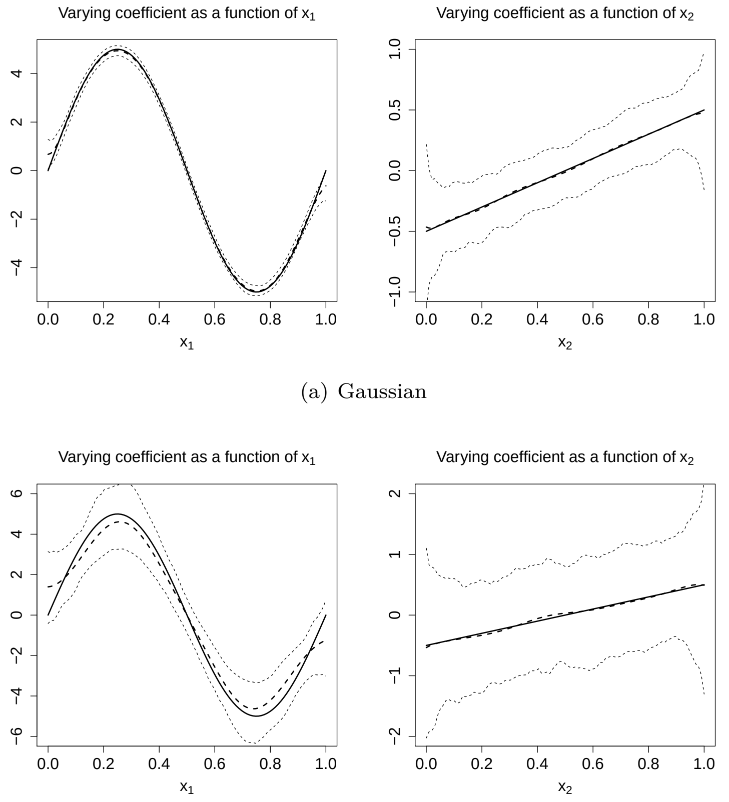
$$Y = \sum_{j=1}^{4} g_j(X_j) + \beta \mathbf{1}_{\{X_5=1\}} + \varepsilon(X_5), \tag{12}$$

with $g_1(x) = 2\sin(2x)$, $g_2(x) = x^2$, $g_3(x) = 0$, $g_4(x) = x$, $\beta = 1.5$, and $\mathbf{1}_A$ denoting the indicator function of event A. The covariates X_1 to X_4 are independent random variables, uniformly distributed on $[-2, 2]$, and $X_5 \in \mathit{Bernoulli}(0.4)$. The error term is given by $\epsilon(X_5) \in N(0, \sigma^2(X_5))$ with $\sigma(0) = 4$ and $\sigma(1) = 2$. Data are generated and fitted by

```
R> library(wsbackfit)
R> set.seed(123)
R> # Define the data generating process
R> n <- 1000
R> x1 <- runif(n)*4-2
R> x2 <- runif(n)*4-2
R> x3 <- runif(n)*4-2
R> x4 <- runif(n)*4-2
R> x5 <- as.numeric(runif(n)>0.6)

R> g1 <- 2*sin(2*x1)
R> g2 <- x2^2
R> g3 <- 0
R> g4 <- x4

R> mu <- g1 + g2 + g3 + g4 + 1.5*x5
```



(a) Gaussian

(b) Bernoulli

Figure 2: The simulation study and Scenario II. From left to right: true curve g_1 (solid line) and average estimate $\hat{g}_1$ (dashed line); true curve g_2 (solid line) and average estimate $\hat{g}_2$ (dashed line). In both cases, 2.5 and 97.5 simulation quantiles are plotted. Top row: Gaussian distribution. Bottom row: Bernoulli distribution.

```
R> err <- (0.5 + 0.5*x5)*rnorm(n)
R> y <- mu + err

R>  df_gauss <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = as.factor(x5), y = y)

R> # Fit the model with a fixed bandwidth for each covariate
R> m0 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1) +
+    sb(x3, h = 0.1) + sb(x4, h = 0.1), kbin = 30, data = df_gauss)
```

A numerical summary of the fitted model can be obtained by calling `print.sback()` or `summary.sback()` with shortcuts `print()` and `summary()`.

```
R> summary(m0)

Generalized Smooth Backfitting/wsbackfit:

Call: sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1) +
    sb(x3, h = 0.1) + sb(x4, h = 0.1), data = df_gauss, kbin = 30)

Sample size: 1000

Bandwidths used in model:
 Effect          h
 sb(x1, h = 0.1) 0.1
 sb(x2, h = 0.1) 0.1
 sb(x3, h = 0.1) 0.1
 sb(x4, h = 0.1) 0.1

Linear/Parametric components:
   Intercept          x1           x2           x3           x4          x51
 1.342678178  0.346506090 -0.040989607 -0.005250654  1.010634908  1.327833794
```

The output obtained from `summary` (corresponding to the prior `sback` call) includes the bandwidth of each nonparametric function, the parameters of the parametric part (here the intercept and β), and the linear slopes (i.e., the α_j in (11) in Section Identification) of the nonparametric functions g_j. Recall (Section Identification) that the algorithm decomposes each nonparametric function in a linear and a nonparametric local constant one. For a varying coefficient g_j, you also get constant g_{0j}.

To complement the numerical results, the **wsbackfit** package also provides graphical outputs by the use of `plot`. In particular, it provides the plots of the estimated nonparametric functions. Figure 3 shows the figures that appear as a result of the following code. We note that through argument `select`, the user can specify the model term to be plotted and use `ylim` to indicate the range for the y-axis. This, however, is optional. Alternatively, the program provides plots that automatically explore the variation of the estimates.

```
R> op <- par(mfrow = c(2,2))
R> plot(m0, select = 1, ylim = c(-2.5,2.5))
R> plot(m0, select = 2, ylim = c(-2.5,2.5))
R> plot(m0, select = 3, ylim = c(-2.5,2.5))
R> plot(m0, select = 4, ylim = c(-2.5,2.5))
R> par(op)
```

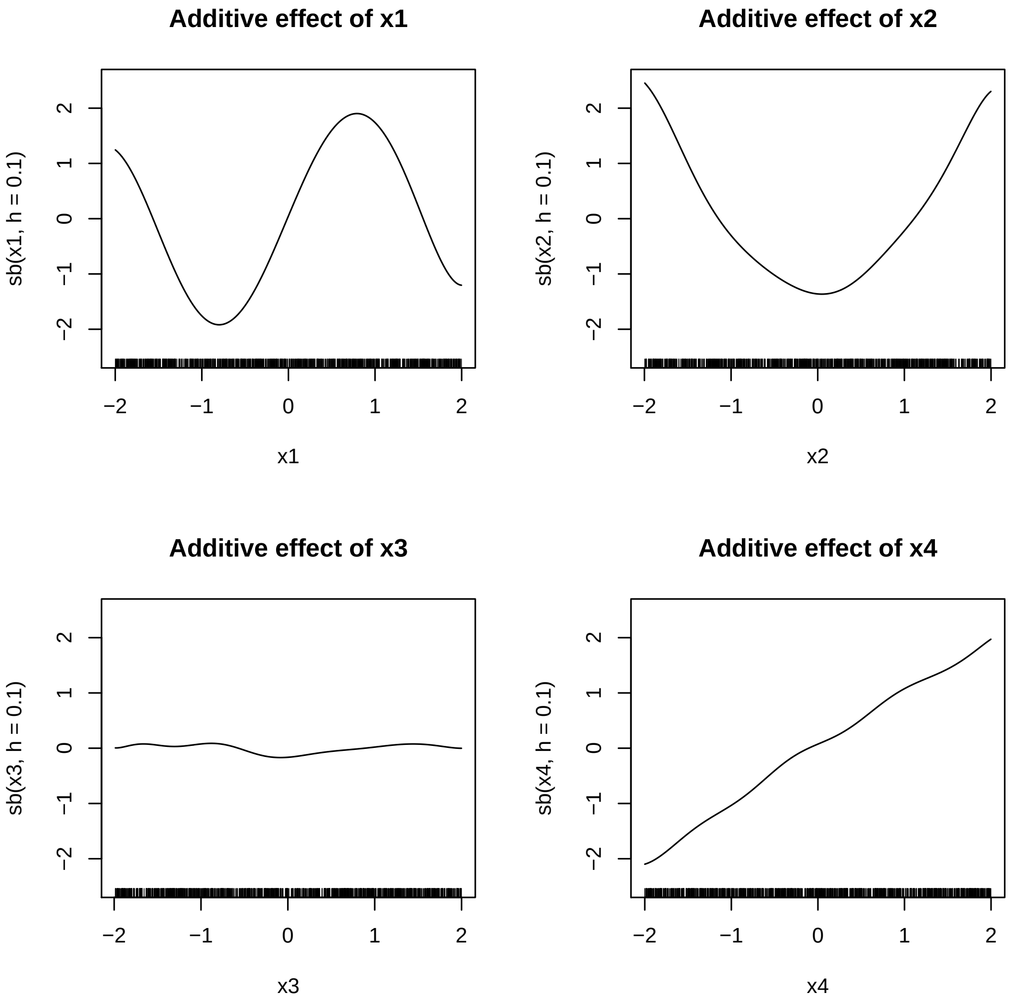


Figure 3: Model (12): Estimated nonparametric functions. These estimates correspond to the sum of the linear and the nonparametric local constant component, recall (11).

If the user is interested in plotting separately each component (α_j and $\tilde{g}_j$), then the argument `composed` is to be set to `FALSE`. The result is shown in Figure 4.

```
R> op <- par(mfrow = c(2,2))
R> plot(m0, select = 1, composed = FALSE, ylim = c(-2.5,2.5))
R> plot(m0, select = 2, composed = FALSE, ylim = c(-2.5,2.5))
R> plot(m0, select = 3, composed = FALSE, ylim = c(-2.5,2.5))
R> plot(m0, select = 4, composed = FALSE, ylim = c(-2.5,2.5))
R> par(op)
```

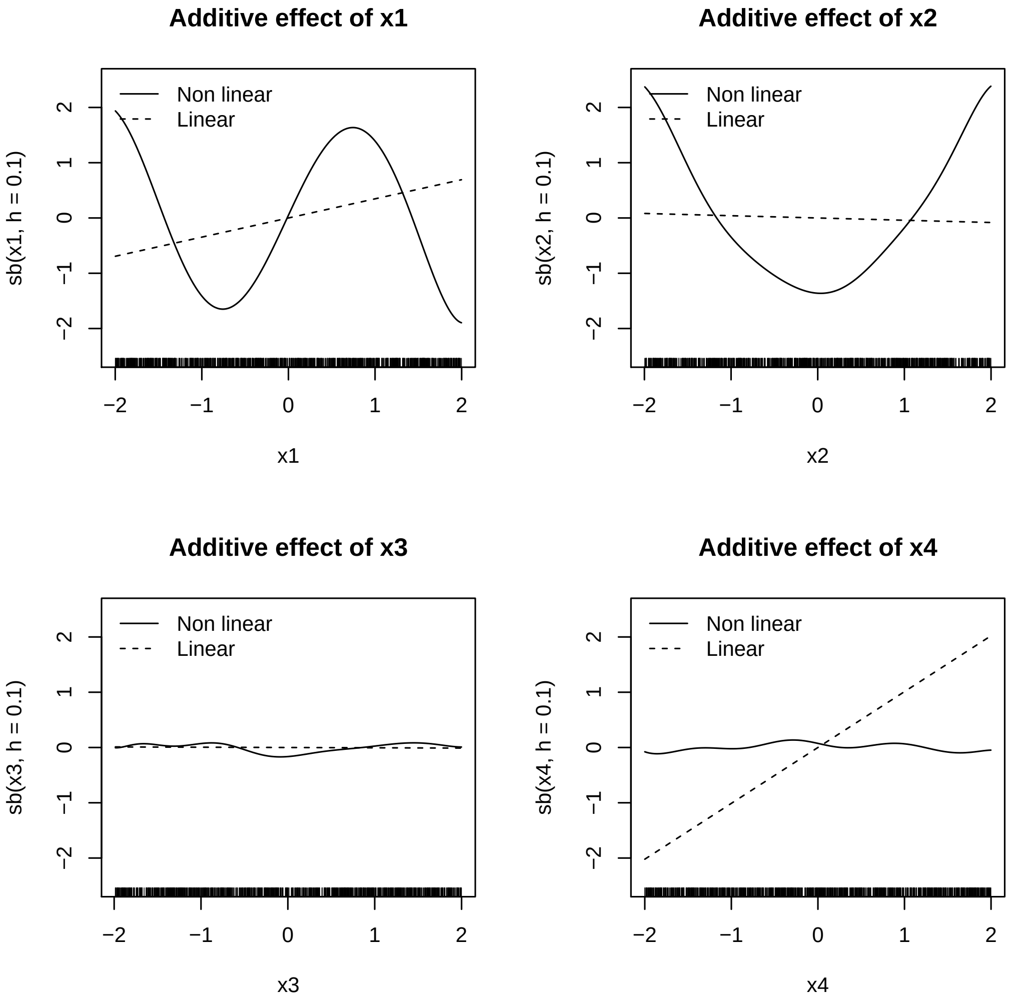


Figure 4: Model (12): Estimated linear and nonlinear components in which each nonparametric function is decomposed, recall (11).

Note that both summary and plot make use of the information contained in the m0 object.

```
R> names(m0)

 [1] "call"          "formula"       "data"          "weights"
 [5] "offset"        "kernel"        "kbin"          "family"
 [9] "effects"       "fitted.values" "residuals"     "h"
[13] "coeff"         "err.CV"
```

This is the list of outputs created by sback. A detailed description of what each component of this list contains was given in Table 3. The user can access this information explicitly and individually, may it be to create its own plots or for further reporting.

As a next step in the analyses of our example, we use the fitted model to estimate the variance. In our example, this is considered to be a function of the binary covariate X_5. Call

```
R> resid <- y - m0$fitted.values
R> sig0 <- var(resid[x5 == 0])
R> sig1 <- var(resid[x5 == 1])
```

The third and final step is to re-estimate the mean model for efficiency reasons with weights that are the inverse of the estimated variance. The code, including the summary, is

```
R>  w <- x5/sig1 + (1-x5)/sig0
R>  m1 <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1) +
+         sb(x3, h = 0.1) + sb(x4, h = 0.1),
+        weights = w, kbin = 30, data = df_gauss)
R> summary(m1)

Generalized Smooth Backfitting/wsbackfit:

Call: sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = 0.1) +
    sb(x3, h = 0.1) + sb(x4, h = 0.1), data = df_gauss, weights = w,
    kbin = 30)

Sample size: 1000

Bandwidths used in model:
 Effect          h
 sb(x1, h = 0.1) 0.1
 sb(x2, h = 0.1) 0.1
 sb(x3, h = 0.1) 0.1
 sb(x4, h = 0.1) 0.1

Linear/Parametric components:
  Intercept          x1          x2          x3          x4         x51
 1.31707760  0.32888538 -0.01262394  0.01222234  1.00289877  1.33368035
```

In the previous fits of this example, we specified all bandwidths used. For the rest of this example, let us consider the case when we ask the program to choose the bandwidths via k-fold CV. We do this for all nonparametric functions, using the following code in which, for the sake of clarity and presentation, we specify `h = -1` although this is actually the default. For convenience, we also call the `summary` command directly:

```
R> m1cv <- sback(formula = y ~ x5 + sb(x1, h = -1) + sb(x2, h = -1) +
+   sb(x3, h = -1) + sb(x4, h = -1), weights = w, kbin = 30,
+   bw.grid = seq(0.01, 0.99, length = 30), KfoldCV = 5, data = df_gauss)
R> summary(m1cv)

Generalized Smooth Backfitting/wsbackfit:

Call: sback(formula = y ~ x5 + sb(x1, h = -1) + sb(x2, h = -1) + sb(x3,
    h = -1) + sb(x4, h = -1), data = df_gauss, weights = w, bw.grid = seq(0.01,
    0.99, length = 30), KfoldCV = 5, kbin = 30)

Sample size: 1000

Bandwidths used in model:
 Effect             h
 sb(x1, h = 0.0892) 0.0892
 sb(x2, h = 0.0887) 0.0887
 sb(x3, h = 0.0907) 0.0907
 sb(x4, h = 0.0912) 0.0912

Linear/Parametric components:
  Intercept          x1          x2          x3          x4         x51
 1.31708207  0.32881191 -0.01219359  0.01247328  1.00258427  1.33366258
```

We do not further discuss the results because their interpretation is the same as before, also because the automatically found data-driven optimal bandwidths are close to what we used as prefixed bandwidths in the former codes. We conclude this section with a brief example in which we specify the bandwidths for some of the nonparametric functions, while for the remaining ones, we let our CV procedure select the bandwidths. For brevity, we skip output and discussion.

```
R> m2cv <- sback(formula = y ~ x5 + sb(x1, h = 0.1) + sb(x2, h = -1) +
+         sb(x3, h = 0.1) + sb(x4, h = 0.1),
+        weights = w, kbin = 30, KfoldCV = 5, data = df_gauss)
```

Post-operative infection data

The next example is an application with data studied, among others, in Roca-Pardiñas and Sperlich (2010). They were taken from a prospective analysis conducted at the University Hospital of Santiago de Compostela in the North of Spain. A total of $n = 2318$ patients who underwent surgery at this center between January 1996 and March 1997 were considered. The main interest is learning about indicators that could predict whether patients may suffer (`inf=1`) or not post-operative infection (`inf=0`), and to see how they relate to the risk of infection. Such predictive indicators could be various, but given the previous studies we concentrate on the pre-operative values of plasma glucose (`gluc`) concentration (measured in mg/dl), and lymphocytes (`linf`, expressed as relative counts (in % of the white blood cell count). The data can be found in **wsbackfit** under the name `infect`.

```
R> data(infect)
R> head(infect)

 age sex linf gluc diab inf
1  85   2   28   55    2   0
2  38   1   18   56    2   1
3  49   2   29   56    2   1
4  63   2   20   60    2   0
5  91   2   17   62    2   0
6  26   2   22   66    2   0
```

In the original studies, it was controlled for other covariates like `age` (in years) and `sex` (coded as 1 = male; 0 = female). For illustrative purposes, we limit our analysis to the investigation of the association of the risk of post-operative infections `inf` with the predictors `linf` and `gluc`, putting all other covariates aside. It is well known that the effect of `linf` on `inf` varies strongly with the concentration of `gluc`. Therefore, one may think of a generalized varying coefficient model of type

$$\begin{aligned}\log \frac{P\left(\mathrm{inf}=1|\mathrm{linf},\mathrm{gluc}\right)}{1-P\left(\mathrm{inf}=1|\mathrm{linf},\mathrm{gluc}\right)} &= g_0 + g_1(\mathrm{gluc}) + g_2(\mathrm{gluc})\mathrm{linf} \\ &= g_0 + \left(\alpha_1 \cdot \mathrm{gluc} + \tilde{g}_1(\mathrm{gluc})\right) + \left(g_{20} + \alpha_2 \cdot \mathrm{gluc} + \tilde{g}_2(\mathrm{gluc})\right)\mathrm{linf}, \qquad (13)\end{aligned}$$

in which we are working with the Logit link. This can be fitted using of the following code

```
R> data(infect)
R> # Generalized varying coefficient model with binary response
R> m2 <- sback(formula = inf ~ sb(gluc, h = 10) + sb(gluc, by = linf, h = 10),
+   data = infect, kbin = 15, family = "binomial")

R> summary(m2)

Generalized Smooth Backfitting/wsbackfit:

Call: sback(formula = inf ~ sb(gluc, h = 10) + sb(gluc, by = linf,
    h = 10), data = infect, kbin = 15, family = "binomial")

Sample size: 2312

Bandwidths used in model:
 Effect                      h
 sb(gluc, h = 10)            10
 sb(gluc, by = linf, h = 10) 10

Linear/Parametric components:
    Intercept          gluc          linf     gluc:linf
-1.4155401353  0.0068313875 -0.0346648080 -0.0000456441
```

Note that this model, recall (13), contains, in addition to the constant term (intercept), the main linear effects of `gluc` α_1 and `linf` g_{20}, and the linear interaction between `gluc` and `linf` α_2, all provided in the very last line. Our bandwidths have been chosen for graphical convenience. The graphical output, i.e., the plots of the estimated nonparametric functions, is obtained by the code

```
R> op <- par(mfrow = c(1,3))
R> plot(m2, composed = FALSE, ask = FALSE, cex.main = 2, cex = 2, cex.lab = 1.5,
+    cex.axis = 2)
```

```
R> par(op)

R> op <- par(mfrow = c(1,3))
R> plot(m2, composed = FALSE, ask = FALSE, cex.main = 2, cex = 2, cex.lab = 1.5,
+    cex.axis = 2)
R> par(op)
```

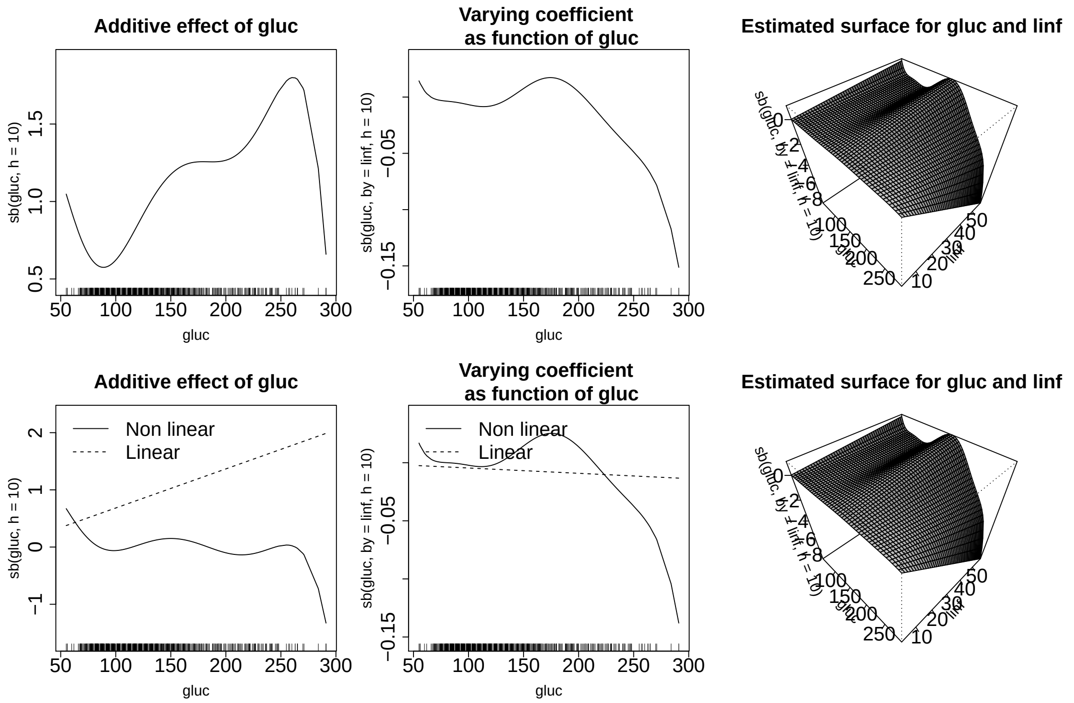


Figure 5: Post-operative infection data. Upper row: estimates of $\alpha_1 \cdot gluc + \tilde{g}_1(gluc)$ (left), $\alpha_2 \cdot gluc + \tilde{g}_2(gluc)$ (middle), and $(\alpha_2 \cdot gluc + \tilde{g}_2(gluc))\, linf$ (right) obtained from model (13). The plots on the left and the center in the bottom line show the estimates of the linear and nonlinear components separately. In this example, the right plot is simply repeated.

In Figure 5, you see the functionals of the nonparametric additive effect of gluc on the index η (left column), the varying coefficient (center), and the interaction surface (right column), because the interest is in revealing how the effect of lymphocytes changes with the plasma glucose concentration. If the interest is also in knowing the resulting probabilities of post-operational infection, then there are the options of plotting the two-dimensional function as a (dynamic) 3-D plot (less appropriate for printed figures) or by contour plots as done in Figure 6. This was created with the code

```
R> # Dataframe for prediction (and plotting)
R> ngrid <- 30
R> gluc0 <- seq(50,190, length.out=ngrid)
R> linf0 <- seq(0,45, length.out=ngrid)
R> infect_pred  <- expand.grid(gluc = gluc0, linf = linf0)

R> m2p <- predict(m2, newdata = infect_pred)
R> n <- sqrt(nrow(infect_pred))
R> Z <- matrix(m2p$pfitted.values, n, n)
R > filled.contour(z = Z, x = gluc0, y = linf0,
+ xlab = "Glucose (mg/dl)", ylab = "Lymphocytes (%)",
+ col = cm.colors(21))
```

As can be seen from Figure 6, high levels of gluc increase the post-operative infection risk, but higher linf values can mitigate this effect significantly.

Poisson regression with offset

Let us now consider a simulated example that illustrates the use of Poisson regression with a nontrivial use of option offset. We simulate data where each subject may have different levels of

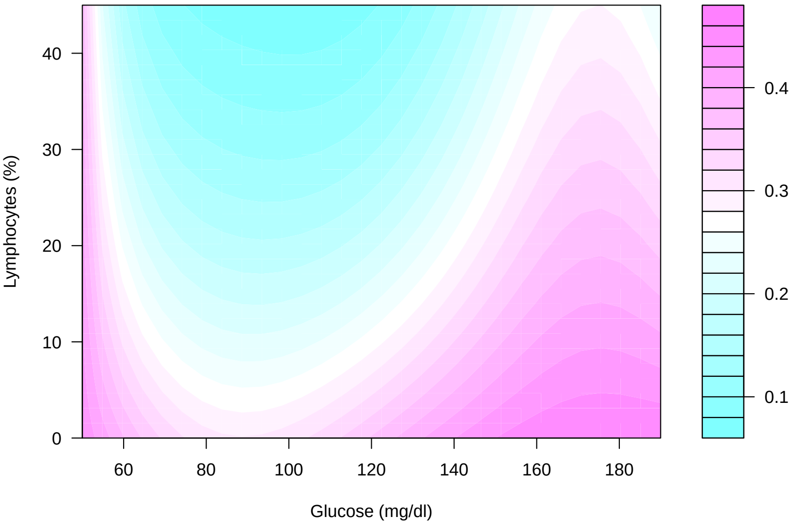


Figure 6: The post-operative infection data: Estimated probability of post-operational infection.

exposure to the event of interest. As explained in the above sections, this can be handled with the `offset` option. More specifically, for the level of exposure P we consider

$$Y \in Poisson\left(P \cdot \exp\left(2 + 3X_1^2 + 5X_2^3\right)\right) \ .$$

In our simulations, X_1 and X_2 were generated as independent continuous random variables uniformly distributed on $[-1, 1]$, and P as an approximately uniformly distributed discrete variable with support $\{50, 51, \ldots, 100\}$. The complete code for simulation, estimation, and summary of results is

```
R> set.seed(123)
R> # Generate the data
R> n <- 1000
R> x1 <- runif(n,-1,1)
R> x2 <- runif(n,-1,1)
R> eta <- 2 + 3*x1^2 + 5*x2^3
R> exposure <- round(runif(n, 50, 500))
R> y <- rpois(n, exposure*exp(eta))
R> df_poiss <- data.frame(y = y, x1 = x1, x2 = x2)
R> # Fit the model
R> m4 <- sback(formula = y ~ sb(x1, h = 0.1) + sb(x2, h = 0.1),
+        data = df_poiss, offset = log(exposure),
+        kbin = 30, family = "poisson")

R> summary(m4)

Generalized Smooth Backfitting/wsbackfit:

Call: sback(formula = y ~ sb(x1, h = 0.1) + sb(x2, h = 0.1), data = df_poiss,
    offset = log(exposure), kbin = 30, family = "poisson")

Sample size: 1000

Bandwidths used in model:
 Effect          h
 sb(x1, h = 0.1) 0.1
 sb(x2, h = 0.1) 0.1

Linear/Parametric components:
 Intercept         x1         x2
3.00099626 0.09698672 3.06092318
```

As for the previous examples, a graphical output can be obtained using the `plot` function like in the following code. The results are shown in Figure 7, namely the additive components.

```
R> op <- par(mfrow = c(1,2))
R> plot(m4, ask = FALSE)
R> par(op)
```

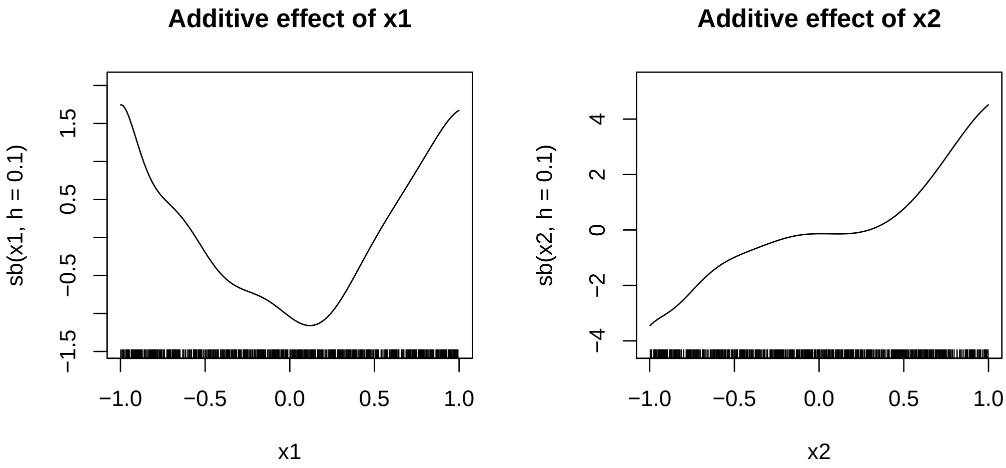


Figure 7: The Poisson Simulated Data: Estimated nonparametric functions.

7 Summary

We have first given a general introduction into the class of Generalized Structures Models, together with the powerful kernel-based smooth backfitting estimator to fit the members of this model class. This was accompanied by a (certainly incomplete) literature review and closed with a small review and discussion of existing methods and software for similar and related models. Except for the varying coefficient model estimator in the **np** package, they are all based on splines. We concluded that, while there is a huge body of literature on SB and its advantages, it is hardly used in practice due to the lack of software. The **wsbackfit** package intends to close this gap.

Next, we provided some insight into the weighted SB and the objective function that is minimized by our algorithm. This allowed us to better explain the users' options like `weights` and `offset`. We outlined which models can be estimated by the presently available package. The description of the procedure was complemented by a section on the implemented CV, bandwidth choice, binning, convergence, and identification issues to clarify the location and scaling of the resulting estimates.

The package description has been kept condense but its use has been illustrated along several examples that cover some of the estimable models. They comprise the use of all options provided. Moreover, the numerical examples give an idea of the estimators' performance. For more details, we recommend consulting the cited articles dealing with the particular models.

We believe that this package is an important enrichment of the existing methods with many useful applications of flexible data analysis and prediction. It can almost straightforwardly be used for testing (Cadarso-Suárez et al., 2006; Mammen and Sperlich, 2021) or studying the heterogeneity of causal effects (Benini and Sperlich, 2021) and many other interesting applications. The next challenge will be the extension of this package to cover the analysis of more complex data (Jeon and Park, 2020). The package is not just open for extensions. We explicitly invite people to contribute.

Bibliography

A. Arcagni and L. Bagnato. *sBF: Smooth Backfitting*, 2014. URL `https://CRAN.R-project.org/package=sBF`. R package version 1.1.1. [p315]

G. Benini and S. Sperlich. Modeling heterogeneous treatment effects in the presence of endogeneity. *Econometric Reviews*, forthcoming, 2021. [p331]

H. Binder and G. Tutz. A comparison of methods for the fitting of generalized additive models. *Statistics and Computing*, 18:87–99, 2008. URL `https://doi.org/10.1007/s11222-007-9040-0`. [p315]

N. Bissantz, H. Dette, T. Hildebrandt, and K. Bissantz. Smooth backfitting in additive inverse regression. *Annals of the Institute of Statistical Mathematics*, 68(4):827–853, 2016. URL https://doi.org/10.1007/s10463-015-0517-x. [p315]

A. Brezger, T. Kneib, and S. Lang. Bayesx: Analysing bayesian structured additive regression models. *Journal of Statistical Software*, 14(11), 2005. URL https://doi.org/10.18637/jss.v014.i11. [p314, 315]

C. Cadarso-Suárez, J. Roca-Pardiñas, and A. Figueiras. Effect measures in non-parametric regression with interactions between continuous exposures. *Statistics in Medicine*, 25(4):603–621, 2006. URL https://doi.org/10.1002/sim.2356. [p331]

L. Fahrmeier, T. Kneib, and S. Lang. Penalized structured additive regression for space-time data: a bayesian perspective. *Statistica Sinica*, 14(3):731–761, 2004. URL www.jstor.org/stable/24307414. [p315]

J. Fan and I. Gijbels. *Local polynomial modelling and its applications*. Number 66 in Monographs on statistics and applied probability series. Chapman & Hall/CRC, 1996. [p319]

K. Han and B. U. Park. Smooth backfitting for errors–in–variables additive models. *The Annals of Statistics*, 46(5):2216–2250, 2018. URL https://doi.org/10.1214/17-AOS1617. [p315, 318, 320]

T. Hastie. *gam: Generalized Additive Models*, 2019. URL https://CRAN.R-project.org/package=gam. R package version 1.16.1. [p315]

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, New York, 1990. [p314, 315]

T. Hayfield and J. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5), 2008. URL https://doi.org/10.18637/jss.v027.i05. [p315]

M. Hiabu, E. Mammen, M. D. Martinez-Miranda, and J. P. Nielsen. Smooth backfitting of proportional hazards with multiplicative components. *Journal of the American Statistical Association*, 2020. URL doi.org/10.1080/01621459.2020.1753520. [p315]

M. J. Jeon and B. U. Park. Additive regression with hilbertian responses. *Annals of Statistics*, 48 (5):2671–2697, 2020. URL https://doi.org/10.1214/19-AOS1902. [p331]

T. Kneib, C. Belitz, A. Brezger, and S. Lang. Bayesx - bayesian inference in structured additive regression software. *ISBA Bulletin*, 15(1):11–13, 2008. URL https://bayesian.org/wp-content/uploads/2016/09/0803.pdf. [p315]

M. Köhler, A. Schindler, and S. Sperlich. A review and comparison of bandwidth selection methods for kernel regression. *International Statistical Review*, 82(2):243–274, 2014. URL https://doi.org/10.1111/insr.12039. [p318]

Y. K. Lee, E. Mammen, and B. U. Park. Backfitting and smooth backfitting for additive quantile models. *The Annals of Statistics*, 38(5):2857–2883, 2010. URL https://doi.org/10.1214/10-AOS808. [p314]

Y. K. Lee, E. Mammen, and B. U. Park. Flexible generalized varying coefficient regression models. *The Annals of Statistics*, 40(3):1906–1933, 2012. URL https://doi.org/10.1214/12-AOS1026. [p314, 316, 317, 319]

Q. Li and J. Racine. Smooth varying-coefficient estimation and inference for qualitative and quantitative data. *Econometric Theory*, 26(6):1607–1637, 2010. URL www.jstor.org/stable/40930669. [p315]

O. Linton and J. Nielsen. A kernel method of estimating structured nonparametric regression based on marginal integration. *Biometrika*, 82(1):93–100, 1995. URL https://www.jstor.org/stable/2337630. [p315]

O. Linton, S. Sperlich, and I. Van Keilegom. Estimation of a semiparametric transformation model. *The Annals of Statistics*, 36(2):686–718, 2008. URL https://doi.org/10.1214/009053607000000848. [p314, 316]

E. Mammen and J. Nielsen. Generalised structured models. *Biometrika*, 90(3):551–566, 2003. URL https://doi.org/10.1093/biomet/90.3.551. [p314, 315, 316]

E. Mammen and B. Park. Bandwidth selection for smooth backfitting in additive models. *The Annals of Statistics*, 33(3):1260–1294, 2005. URL https://doi.org/10.1214/009053605000000101. [p314, 317]

E. Mammen and S. Sperlich. Backfitting tests in generalised structured models. *Biometrika*, 2021. URL https://doi.org/10.1093/biomet/asaa108. [p331]

E. Mammen, O. Linton, and J. Nielsen. The existence and asymptotic properties of a backfitting projection algorithm under weak conditions. *The Annals of Statistics*, 27(5):1443–1490, 1999. URL https://doi.org/10.1214/aos/1017939138. [p314]

P. McCullagh and J. Nelder. *Generalized linear models*. Chapman and Hall, London, New York, 1989. [p316]

J. Nielsen and S. Sperlich. Smooth backfitting in practice. *Journal of the Royal Statistical Society, B*, 67(1):43–61, 2005. URL https://doi.org/10.1111/j.1467-9868.2005.00487.x. [p315, 317]

J. Opsomer. Asymptotic properties of backfitting estimators. *Journal of Multivariate Analysis*, 73 (2):166–179, 2000. URL https://doi.org/10.1006/jmva.1999.1868. [p314]

J. Opsomer and D. Ruppert. Fitting a bivariate additive model by local polynomial regression. *The Annals of Statistics*, 25(1):186–211, 1997. URL https://doi.org/10.1214/aos/1034276626. [p314]

B. U. Park, E. Mammen, Y. K. Lee, and E. R. Lee. Varying coefficient regression models: A review and new developments. *International Statistical Review*, 83(1):36–64, 2015. URL https://doi.org/10.1111/insr.12029. [p314]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. ISBN 3-900051-07-0. [p315]

J. Roca-Pardiñas and S. Sperlich. Estimating generalized structured models - a computational note. *Discussion Paper*, 2008. [p316]

J. Roca-Pardiñas and S. Sperlich. Feasible estimation in generalized structured models. *Statistics and Computing*, 20:367–379, 2010. URL https://doi.org/10.1007/s11222-009-9130-2. [p314, 315, 316, 317, 319, 328]

J. Rodríguez-Poó, S. Sperlich, and P. Vieu. Semiparametric estimation of weak and strong separable models. *Econometric Theory*, 19(6):1008–1039, 2003. URL https://doi.org/10.1017/S0266466603196065. [p314]

S. Sperlich and R. Theler. Modeling heterogeneity: A praise for varying-coefficient models in causal analysis. *Computational Statistics*, 30:693–718, 2015. URL https://doi.org/10.1007/s00180-015-0581-y. [p315]

S. Sperlich, D. Tjøstheim, and L. Yang. Nonparametric estimation and testing of interaction in additive models. *Econometric Theory*, 18(2):197–251, 2002. URL https://doi.org/doi:10.1017/S0266466602182016. [p315]

D. Stasinopoulos and R. Rigby. Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23(7), 2007. URL https://doi.org/10.18637/jss.v023.i07. [p315]

C. Stone. The dimensionality reduction principle for generalized additive models. *The Annals of Statistics*, 14(2):590–606, 1986. URL https://doi.org/10.1214/aos/1176349940. [p314]

G. Tutz and H. Binder. Generalized additive modelling with implicit variable selection by likelihood based boosting. *Biometrics*, 62(4):961–971, 2006. URL https://doi.org/10.1111/j.1541-0420.2006.00578.x. [p315]

N. Umlauf, T. Kneib, and N. Klein. *BayesX: R Utilities Accompanying the Software Package BayesX*, 2019. URL https://CRAN.R-project.org/package=BayesX. R package version 0.3-1. [p315]

S. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall / CRC Press, New York, 2017. [p315]

Z. Xiao, O. Linton, R. J. Carroll, and E. Mammen. More efficient local polynomial estimation in nonparametric regression with autocorrelated errors. *Journal of the American Statistical Association*, 98:890–992, 2003. URL https://doi.org/10.1198/016214503000000936. [p315]

K. Yu, B. Park, and E. Mammen. Smooth backfitting in generalized additive models. *The Annals of Statistics*, 36(1):228–260, 2008. URL https://doi.org/10.1214/009053607000000596. [p314, 316]

Javier Roca-Pardiñas
Department of Statistics and O.R.
University of Vigo, Spain.
E-mail: roca@uvigo.es

María Xosé Rodríguez-Álvarez
BCAM - Basque Center for Applied Mathematics
and
IKERBASQUE, Basque Foundation for Science
Alameda de Mazarredo, 14
E-48009 Bilbao, Basque Country, Spain
E-mail: mxrodriguez@bcamath.org

Stefan Sperlich
Geneva School of Economics and Management
University of Geneva
Bd du Pont d'Arve 40, CH - 1211 Geneève 4, Switzerland
E-mail: Stefan.Sperlich@unige.ch