The crs Package: Nonparametric Regression Splines for Continuous and Categorical Predictors

by Zhenghua Nie and Jeffrey S Racine

Abstract A new package **crs** is introduced for computing nonparametric regression (and quantile) splines in the presence of both continuous and categorical predictors. B-splines are employed in the regression model for the continuous predictors and kernel weighting is employed for the categorical predictors. We also develop a simple R interface to NOMAD, which is a mixed integer optimization solver used to compute optimal regression spline solutions.

Introduction

Regression splines constitute a popular approach for the nonparametric estimation of regression functions though they are restricted to continuous predictors as are smoothing splines (see e.g. `smooth.spline` which is limited to one numeric predictor) and locally weighted scatterplot smoothing (see `loess` which is limited to four numeric predictors). However, it is not uncommon to encounter relationships involving a mix of numeric and categorical predictors, and the **crs** package implements a regression spline framework for accomplishing this task.

The **crs** package implements the approaches described in Ma et al. (2012) and Ma and Racine (2012) when the option `kernel=TRUE` is selected (default) as described below. The categorical predictors can be handled in two ways, (i) using kernel weighting where the kernel functions are tailored to the discrete support of the categorical predictors (Racine and Li, 2004), and (ii) using indicator basis functions. The fundamental difference between these two approaches is that the use of indicator basis functions consumes degrees of freedom via the number of columns in the basis matrix, while kernel weighting does not. As well, this package implements a range of related methods and provides options that we hope will make it appealing for applied projects, research, and pedagogical purposes alike.

Both semiparametric additive models and fully nonparametric models are implemented. Data-driven methods can be used for selecting the spline degree, number of segments/knots, and bandwidths: leave-one-out cross-validation (`cv.func = "cv.ls"`) (Stone, 1974, 1977), generalized cross-validation (`cv.func="cv.gcv"`) (Craven and Wahba, 1979), and the information-based criterion (`cv.func="cv.aic"`) proposed by Hurvich et al. (1998). Derivatives of arbitrary order can readily be constructed. One of the computational challenges encountered is to obtain the optimal spline degree (non-negative integer), number of segments/knots (positive integer), and bandwidth (bounded and real-valued) for each predictor. We overcome this challenge by providing an interface to NOMAD (Abramson et al., 2011; Le Digabel, 2011).

Before proceeding we briefly provide an overview of some important implementation details:

1. The degree of the spline and number of segments (i.e. knots minus one) for each continuous predictor can be set manually as can the bandwidths for each categorical predictor (if appropriate).

2. Alternatively, any of the data-driven criteria (i.e. `cv.func="..."`) could be used to select either the degree of the spline (holding the number of segments/knots fixed at any user-set value) and bandwidths for the categorical predictors (if appropriate), or the number of segments (holding the degree of the spline fixed at any user-set value) and bandwidths for the categorical predictors (if appropriate), or the number of segments and the degree of the spline for each continuous predictor and bandwidths for each categorical predictor (if appropriate).

3. When indicator basis functions are used instead of kernel smoothing, whether to include each categorical predictor or not can be specified manually or chosen via any `cv.func` method.

4. We allow the degree of the spline for each continuous predictor to include zero, the inclusion indicator for each categorical predictor to equal zero, and the bandwidth for each categorical predictor to equal one, and when the degree/inclusion indicator is zero or the bandwidth is one, the variable is thereby removed from the regression: in this manner, irrelevant predictors can be automatically removed by any `cv.func` method negating the need for pre-testing (mirroring findings detailed in Hall et al. (2004, 2007) for kernel regression).

The design philosophy underlying the **crs** package aims to closely mimic the behaviour of the `lm` function. Indeed, the implementation relies on `lm` and its supporting functions for computation of the

spline coefficients, delete-one residuals, fitted values, prediction and the like. 95% confidence bounds for the fit and derivatives are constructed from asymptotic formulae and automatically generated. Below we describe in more detail the specifics of the implementation for the interested reader. Others may wish to jump to the illustrative example that appears towards the end of this article or simply install and load the package and run `example(crs)`.

Differences between existing spline methods and those in the crs package

Readers familiar with existing R-functions and packages for spline modelling will naturally be wondering what the difference is between the **crs** package and existing spline-based procedures such as

- `smooth.spline` in R base
- `spm` in the **SemiPar** package (Wand, 2010)
- `gam` in the **mgcv** package (Wood, 2006)
- `ssanova` in the **gss** package (Gu, 2012)
- `gam` in the **gam** package (Hastie, 2011)

First we note that the functions `smooth.spline` and `ssanova` are based on smoothing spline methodology, while `spm` uses penalized splines but `gam` in the **gam**/**mgcv** packages allows for smoothing splines, penalized splines, and regression splines. The **crs** package is restricted to 'regression splines' which differs in a number of ways from 'smoothing splines' and 'penalized splines', the fundamental difference being that smoothing/penalized splines use the data points themselves as potential knots and penalize 'roughness' (typically the second derivative of the estimate) while regression splines place knots at equidistant/equiquantile points and do not explicitly penalize roughness, rather, they rely on various cross-validatory approaches for model selection. We direct the interested reader to Wahba (1990) for a treatment of smoothing splines. The **crs** package is one of the few packages devoted to regression spline techniques. We also provide quantile regression splines via the option `tau=`τ ($\tau \in (0,1)$).

Second, many of the aforementioned smoothing spline implementations are semiparametric in nature, the semiparametric additive model being particularly popular. Though semiparametric models exist to circumvent the curse of dimensionality, it does not come without cost. That is, the burden of determining whether semiparametric or nonparametric approaches would be warranted in any given situation is placed squarely on the researcher's shoulder. Unlike many existing spline methods in R, the implementation in the **crs** package is designed so that every parameter that must be chosen by the researcher can be data-driven (via cross-validation) so that such choices adapt to the data at hand *including* whether to use a semiparametric or nonparametric model. This is accomplished using options such as `basis="auto"`, `knots="auto"`, and `complexity="degree-knots"` (`basis="auto"` deciding whether to use an additive basis or tensor product basis in multivariate settings, `knots="auto"` whether to use equispaced knots or quantile knots, and `complexity="degree-knots"` determining both the spline degrees and number of knots).

Generally speaking, almost all of these existing smoothing spline approaches can handle mixed datatypes though their approaches to the treatment of categorical variables often differ (none use categorical kernel smoothing as in the **crs** package).

The underlying model

Regression spline methods can be limited in their potential applicability as they are based on *continuous* predictors. However, in applied settings we often encounter *categorical* predictors such as strength of preference ("strongly prefer", "weakly prefer", "indifferent") and so forth.

We wish to model the unknown conditional mean in the following location-scale model,

$$Y = g(\mathbf{X}, \mathbf{Z}) + \sigma(\mathbf{X}, \mathbf{Z})\,\varepsilon,$$

where $g(\cdot)$ is an unknown function, $\mathbf{X} = (X_1, \ldots, X_q)^{\mathrm{T}}$ is a q-dimensional vector of continuous predictors, $\mathbf{Z} = (Z_1, \ldots, Z_r)^{\mathrm{T}}$ is an r-dimensional vector of categorical predictors, and $\sigma^2(\mathbf{X}, \mathbf{Z})$ is the conditional variance of Y given $\mathbf{X}$ and $\mathbf{Z}$. Letting $\mathbf{z} = (z_s)_{s=1}^{r}$, we assume that z_s takes c_s different values in $D_s \equiv \{0, 1, \ldots, c_s - 1\}$, $s = 1, \ldots, r$, and c_s is a finite positive constant.

For the continuous predictors the regression spline model employs the B-spline routines in the GNU Scientific Library (Galassi et al., 2009). The B-spline function is the maximally differentiable interpolative basis function (B-spline stands for 'basis-spline').

Heuristically, we conduct linear (in parameter) regression using the R function `lm`, however, we replace the continuous predictors with B-splines of potentially differing order and number of segments for each continuous predictor. For the tensor product bases we set `intercept=TRUE` for each univariate spline basis, while for the additive spline bases we adopt the `intercept=FALSE` variants and include an intercept term in the model (the B-splines will therefore not sum to one, i.e. an order m B-spline with one segment (two knots/breakpoints) has $m+1$ columns and we drop the first as is often done, though see Zhou and Wolfe (2000) for an alternative approach based upon shrinkage methods). This allows multiple bases to coexist when there is more than one continuous predictor. The tensor product basis is given by

$$B_1 \otimes B_2 \otimes \cdots \otimes B_p,$$

where $\otimes$ is the Kronecker product where the products operate column-wise and B_j is the basis matrix for predictor j as outlined above. We also support a 'generalized' polynomial B-spline basis that consists of a varying-order polynomial with appropriate interactions. When additive B-spline bases are employed we have a semiparametric 'additive' spline model (no interaction among variables unless explicitly specified). When the tensor product or generalized polynomial is employed we have a fully nonparametric model (interaction among all variables). Whether to use the additive, tensor product, or generalized polynomial bases can be pre-specified or automatically determined via any `cv.func` method (see the options for `basis=` in `?crs`).

We offer the default option to use categorical kernel weighting (`lm(...,weights=L)`) to handle the presence of categorical predictors (see below for a description of `L`). We also offer the option of using indicator basis functions for the categorical predictors.

Weighted least-squares estimation of the underlying model

The unknown function $g(\mathbf{X},\mathbf{Z})$ can be approximated by $\mathcal{B}(\mathbf{X})^{\mathrm{T}}\beta(\mathbf{Z})$, where $\beta(\mathbf{Z})$ is a vector of coefficients and $\mathcal{B}(\mathbf{X})$ the B-spline basis matrix described above.

We estimate $\beta(\mathbf{z})$ by minimizing the following weighted least squares criterion,

$$\widehat{\beta}(\mathbf{z}) = \underset{\beta(\mathbf{z})\in\mathbb{R}^{\mathbf{K}_n}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\{ Y_i - \mathcal{B}(\mathbf{X}_i)^{\mathrm{T}}\beta(\mathbf{z}) \right\}^2 L(\mathbf{Z}_i,\mathbf{z},\lambda).$$

Placement of knots

The user can determine where knots are to be placed using one of two methods:

1. knots can be placed at equally spaced quantiles whereby an equal number of observations lie in each segment ('quantile knots').
2. knots can be placed at equally spaced intervals ('uniform knots').

If preferred, this can be determined automatically using the option `knots="auto"`.

Kernel weighting

Let Z_i be an r-dimensional vector of categorical/discrete predictors. We use z_s to denote the s-th component of z, we assume that z_s takes c_s different values in $D_s \equiv \{0,1,\ldots,c_s-1\}$, $s=1,\ldots,r$, and let $c_s \geq 2$ be a finite positive constant.

For an unordered categorical predictor, we use a variant of the kernel function outlined in Aitchison and Aitken (1976) defined as

$$l(Z_{is},z_s,\lambda_s) = \begin{cases} 1, & \text{when } Z_{is}=z_s, \\ \lambda_s, & \text{otherwise.} \end{cases} \qquad (1)$$

Let $\mathbf{1}(A)$ denote the usual indicator function, which assumes the value one if A holds true, zero otherwise. Using (1), we can construct a product kernel weight function given by

$$L(\mathbf{Z}_i,\mathbf{z},\lambda) = \prod_{s=1}^{r} l(Z_{is},z_s,\lambda_s) = \prod_{s=1}^{r} \lambda_s^{\mathbf{1}(Z_{is}\neq z_s)},$$

while for an ordered categorical we use the function defined by

$$\tilde{l}(Z_{is},z_s,\lambda_s) = \lambda_s^{|Z_{is}-z_s|}$$

and modify the product kernel function appropriately. When Z contains a mix of ordered and unordered variables we select the appropriate kernel for each variable's type when constructing the product kernel $L(\mathbf{Z}_i,\mathbf{z},\lambda)$.

Note that when $\lambda_s = 1$ all observations are 'pooled' over categorical predictor s hence the variable z_s is removed from the resulting estimate, while when $\lambda_s = 0$ only observations lying in a given cell are used to form the estimate.

Additional estimation details

Estimating the model requires construction of the spline bases and their tensor product (if specified) along with the categorical kernel weighting function. Then, for a given degree and number of segments for each continuous predictor and bandwidth for each categorical predictor (or indicator bases if `kernel=FALSE`), the model is fit via least-squares.

All smoothing parameters can be set manually by the user if so desired, however be forewarned that you must use the option `cv="none"` otherwise the values specified manually will become the starting points for search when `cv="nomad"` ('nonsmooth mesh adaptive direct search (NOMAD)', see Abramson et al. (2011) and Le Digabel (2011)). Currently, we provide a simple R interface to NOMAD (see the section below) in the **crs** package which also can be applied to solve other mixed integer optimization problems.

The degree, segment, and bandwidth vectors can be jointly determined via any `cv.func` method by setting the option `cv="nomad"` or `cv="exhaustive"` (exhaustive search). Here we have to solve nonlinear non-convex and non-differentiable mixed integer constrained optimization problems to obtain the optimal degree, number of segments, and bandwidth for each predictor.

Setting the option `cv="nomad"` (default) computes NOMAD-based cross-validation directed search while setting `cv="exhaustive"` computes exhaustive cross-validation directed search for each unique combination of the degree and segment vector for each continuous predictor from `degree=degree.min` through `degree=degree.max` (default 0 and 10, respectively) and from `segments=segments.min`

through `segments=segments.max` (default 1 and 10, respectively).

When `kernel=TRUE` (default) setting the option `cv="exhaustive"` computes bandwidths ($\in [0,1]$) obtained via numerical minimization (using `optim`) for each unique combination of the degree and segment vectors (restarting can be conducted via `restarts=`). When setting `cv="nomad"` the number of multiple starts can be controlled by `nmulti=` (default is 5). The model possessing the lowest criterion function value over the `nmulti` restarts is then selected as the final model.

Note that `cv="exhaustive"` is often unfeasible (this combinatoric problem can become impossibly large to compute in finite time) hence `cv="nomad"` is the default. However, with `cv="nomad"` one should set `nmulti=` to some sensible value greater than zero to avoid becoming trapped in local minima (default `nmulti=5`).

Data-driven smoothing parameter criteria

We incorporate three popular approaches for setting the smoothing parameters of the regression spline model, namely least-squares cross-validation, generalized cross-validation, and an AIC method corrected for use in nonparametric settings.

Let the fitted value from the regression spline model be denoted $\hat{Y}_i = B_m(X_i)^T\hat{\beta}(Z_i)$. Letting $\hat{\varepsilon}_i = Y_i - \hat{Y}_i$ denote the ith residual from the categorical regression spline model, the least-squares cross-validation function is given by

$$CV = \frac{1}{n}\sum_{i=1}^{n}\frac{\hat{\varepsilon}_i^2}{(1-h_{ii})^2}$$

and this computation can be done with effectively one pass through the data set, where h_{ii} denotes the ith diagonal element of the spline basis projection matrix (see below for details). Since h_{ii} is computed routinely for robust diagnostics in R, this can be computed along with (and hence as cheaply as) the vector of spline coefficients themselves. Thus least-squares cross-validation is computationally appealing, particularly for large data sets.

Let H denote the $n \times n$ weighting matrix such that $\hat{Y} = HY$ with its ith diagonal element given by H_{ii} where $\mathrm{tr}(H)$ means the trace of H which is equal to $\sum_{i=1}^{n} h_{ii}$. The matrix H is often called the 'hat matrix' or 'smoother matrix' and depends on X but not on Y. The 'generalized' cross-validation function is obtained by replacing h_{ii} in the above formula with its average value denoted $\mathrm{tr}(H)/n$ (Craven and Wahba, 1979).

The information-based criterion proposed by Hurvich et al. (1998) is given by

$$\mathrm{AIC}_c = \ln(\hat{\sigma}^2) + \frac{1+\mathrm{tr}(H)/n}{1-\{\mathrm{tr}(H)+2\}/n}$$

where

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = Y'(I-H)'(I-H)Y/n.$$

Each of these criterion functions can be minimized with respect to the unknown smoothing parameters either by numerical optimization procedures or by exhaustive search.

Though each of the above criteria are asymptotically equivalent in terms of the bandwidths they deliver ($\mathrm{tr}(H)/n \to 0$ as $n \to \infty$), they may differ in finite-sample settings for a small smoothing parameter (large $\mathrm{tr}(H)/n$) with the AIC_c criterion penalizing more heavily when undersmoothing than either the least-squares cross-validation or generalized cross-validation criteria (the AIC_c criterion effectively applies an infinite penalty for $\mathrm{tr}(H)/n \geq 1/2$).

Pruning

Once a model has been selected via cross-validation (i.e. `degree`, `segments`, `include` or `lambda` have been selected), there is the opportunity to (potentially) further refine the model by adding the option `prune=TRUE` to the `crs` function call. Pruning is accomplished by conducting stepwise cross-validated variable selection using a modified version of the `stepAIC` function in the R **MASS** package where the function `extractAIC` is replaced with the function `extractCV` with additional modifications where necessary. Pruning of potentially superfluous bases is undertaken, however, the pruned model (potentially containing a subset of the bases) is returned *only if its cross-validation score improves upon the model being pruned*. When this is not the case a warning is issued to this effect. A final pruning stage is commonplace in spline frameworks and may positively impact on the finite-sample efficiency of the resulting estimator depending on the rank of the model being pruned. Note that this option can only be applied when `kernel=FALSE` (or when there exist only numeric predictors).

A simple R interface to NOMAD

The **crs** package has included a simple R interface to the NOMAD optimization solver called `snomadr`. `snomadr` implements the NOMAD library which is an open source C++ implementation of the Mesh Adaptive Direct Search (MADS) algorithm designed for constrained optimization of blackbox functions (Abramson et al., 2011; Le Digabel, 2011). `snomadr` can be seen as a standalone interface to this optimization solver, though we would like to point out that the authors of `NOMAD` are currently working on an R package for NOMAD that we expect to be much more comprehensive than the simple interface provided here. The principle behind developing our in-

terface is simply to shield the user from setting up the optimization model underlying our regression spline implementation. For what follows we will not discuss the performance of the solver NOMAD, rather we direct the interested reader to Abramson et al. (2011); Le Digabel (2011) and the references therein.

The structure of the `snomadr` interface is similar to the interface in the R package **ipoptr** (`http://www.ucl.ac.uk/~uctpjyy/ipoptr.html`) which exports the R interface to the optimization solver IPOPT (Wächter and Biegler, 2006). The interface is split into two major portions, one is the R code called 'snomadr.R' and another is a C++ program called 'snomadr.cpp'. The role of 'snomadr.R' is to set up the optimization model, define options and then call the C++ code to solve the optimization problem. 'snomadr.cpp' will receive arguments from the R code and then call the NOMAD library functions to solve the problem and return results to 'snomadr.R'. `.Call` is used to transfer arguments and results between the R and C++ languages. For further details see `?snomadr`.

Illustrative example

By way of illustration we consider a simple simulated example involving one continuous and one categorical predictor.

```
set.seed(42)
n <- 1000
x <- runif(n)
z <- rbinom(n, 1, .5)
y <- cos(2 * pi * x) + z + rnorm(n, sd=0.25)
z <- factor(z)
model <- crs(y ~ x + z)
summary(model)

Call:
crs.formula(formula = y ~ x + z)

Kernel Weighting/B-spline Bases Regression
   Spline

There is 1 continuous predictor
There is 1 categorical predictor
Spline degree/number of segments for x: 3/4
Bandwidth for z: 0.0008551836
Model complexity proxy: degree-knots
Knot type: quantiles
Training observations: 1000
Trace of smoother matrix: 14
Residual standard error: 0.2453 on 993
    degrees of freedom
Multiple R-squared: 0.927,
    Adjusted R-squared: 0.9265
F-statistic: 962.9 on 13 and 986 DF,
    p-value: 0
Cross-validation score: 0.061491531
Number of multistarts: 5
```

The function `crs()` called in this example returns a `"crs"` object. The generic functions `fitted` and `residuals` extract (or generate) estimated values and residuals. Furthermore, the functions `summary`, `predict`, and `plot` (options `mean=FALSE`, `deriv=`*i*, `ci=FALSE`, `plot.behavior = c("plot", "plot-data", "data")`, where *i* is a positive integer) support objects of this type.

Figure 1 presents summary output in the form of partial regression surfaces (predictors not appearing on the axes are held constant at their medians/modes). Note that for this simple example we used the option `plot(model,mean=TRUE)` which presents 'partial regression plots'.[1]

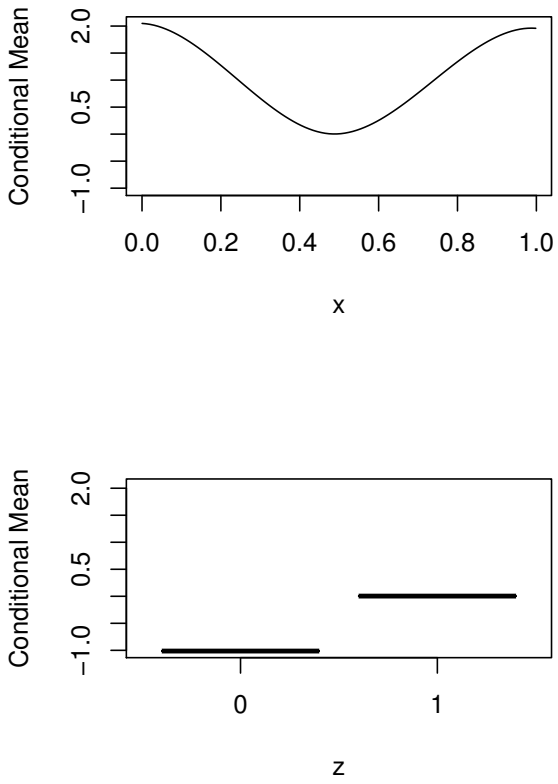

Figure 1: The partial regression plot for one categorical and one continuous predictor.

Next we plot the first partial derivative (which plots the partial derivative with respect to the continuous predictor holding the categorical predictor at its modal value and next the difference between the regression function when the categorical predictor equals 0 and 1 holding the continuous predictor

[1] A 'partial regression plot' is simply a 2D plot of the outcome y versus one predictor x_j when all other predictors are held constant at their respective medians/modes (this can be changed by the user).

at its median) for this model in Figure 2. Note that here we used the option `plot(model,deriv=1)`.

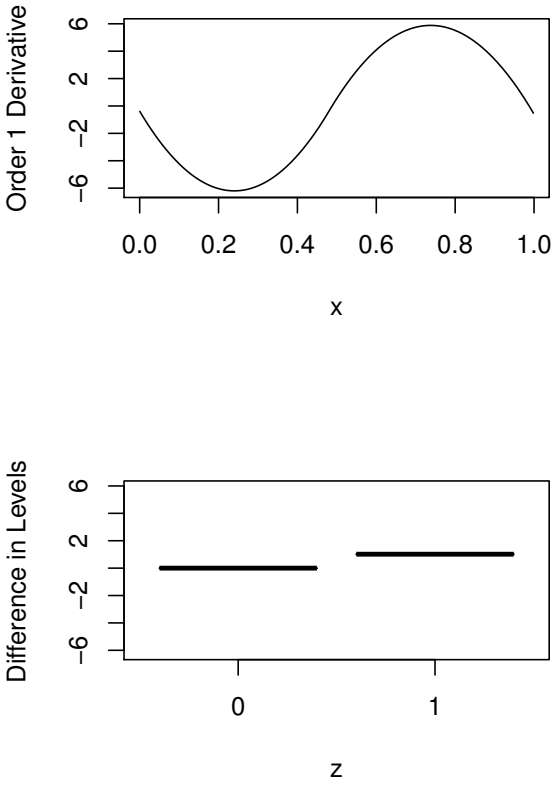

Figure 2: The partial gradient plot for one categorical and one continuous predictor.

Comparison with existing spline methods

We now compare and contrast some existing spline implementations in R with the regression spline implementation contained in the **crs** package via two illustrative Monte Carlo simulations.

The bivariate one continuous predictor case

We first consider the simple and popular case of nonparametrically modeling the relationship between a response Y and univariate predictor X. Here the purpose is to help the reader assess the performance of regression splines versus smoothing splines. We consider four data generating processes (DGPs), the sine and cosine functions, the absolute value function, and the Doppler function given by

$$g(x) = \sqrt{x(1-x)} \times \sin\left(\frac{2\pi(1+2^{-7/5})}{x+2^{-7/5}}\right)$$

For the sine, cosine, and Doppler functions X is $U[0,1]$, while for the absolute value function X is $U[-1,1]$. We vary the signal/noise ratio by standardizing the DGP to have mean zero and unit variance and setting σ for the Gaussian noise to $1/4, 1/2, 1, 2$ which produces expected in-sample fits of $R^2 = 0.95, 0.80, 0.50, 0.20$ for the oracle estimator (i.e. one that uses knowledge of the DGP).

We compare `crs`-based regression splines with smoothing spline methods `gam` in the **mgcv** package, `spm` in the **SemiPar** package, `ssanova` in the **gss** package, and `loess` in base R. All methods in this first simulation use default settings.

We draw $M = 1{,}000$ replications of size $n = 1{,}000$ from each DGP, compute the mean square error (MSE) for each model, and report the median value of each model's MSE relative to that for `crs` over the $M = 1{,}000$ replications in Table 1 below (numbers less than one indicate that a method is more efficient than `crs`).

Table 1: MSE efficiency of the various methods relative to `crs` (numbers less than one indicate more efficient than `crs`).

$\sin(2\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	0.97	1.24	1.02	8.06
$\sigma = 0.50$	0.80	0.89	0.81	2.15
$\sigma = 1.00$	0.78	0.78	0.78	0.92
$\sigma = 2.00$	0.84	0.79	0.83	0.71
$\sin(4\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	0.89	1.43	1.11	507.85
$\sigma = 0.50$	0.78	1.15	0.98	129.41
$\sigma = 1.00$	0.81	1.01	0.95	36.72
$\sigma = 2.00$	0.77	0.84	0.85	9.89
$\cos(2\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	1.13	1.17	1.17	39.39
$\sigma = 0.50$	0.99	1.00	1.04	11.18
$\sigma = 1.00$	0.98	0.93	1.00	3.67
$\sigma = 2.00$	0.89	0.82	0.91	1.38
$\cos(4\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	2.78	1.54	1.30	299.88
$\sigma = 0.50$	1.31	1.25	1.12	79.85
$\sigma = 1.00$	0.93	1.02	0.97	21.42
$\sigma = 2.00$	0.82	0.89	0.87	6.09
$\text{abs}(x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	1.55	0.91	0.98	14.59
$\sigma = 0.50$	1.20	1.05	1.11	5.97
$\sigma = 1.00$	1.23	1.18	1.26	2.73
$\sigma = 2.00$	1.41	1.36	1.43	1.61
$\text{doppler}(x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	65.45	1.67	1.68	359.63
$\sigma = 0.50$	18.53	1.39	1.42	99.51
$\sigma = 1.00$	5.15	1.14	1.19	25.40
$\sigma = 2.00$	1.78	1.03	1.04	6.81

Table 1 reveals some general patterns. For smooth low frequency DGPs such as the sine and cosine, `gam`, `spm`, and `ssanova` turn in similar performances depending on the signal/noise ratio. However, `loess` does not do well with these DGPs. For the absolute value and Doppler DGPs `gam`, `spm`, `ssanova` and `loess` on balance perform worse than `crs`. From a minimax perspective, the regression spline implementation in the **crs** package performs relatively well overall. Naturally by modifying tuning parameters one could expect to improve the performance of the smoothing spline methods (this would apply to `crs` as well as default settings are used throughout). The point to be made is simply that the regression spline implementation in **crs** appears to adapt fairly well to the underlying DGP and may therefore be of interest to readers.

The multivariate case, mixed predictor types

Next we consider mixed predictor multivariate DGPs with additive and multiplicative features. The additive DGP is of the form

```
x1 <- runif(n)
x2 <- runif(n)
z <- rbinom(n, 1, .5)
dgp <- cos(2*pi*x1) + sin(2*pi*x2) + z
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

while the multiplicative DGPs are of the form

```
x1 <- runif(n)
x2 <- runif(n)
z <- rbinom(n, 1, .5)
dgp <- cos(2*pi*x1) * sin(2*pi*x2) * z
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

and the radial function given by

```
x1 <- runif(n, -5, 5)
x2 <- runif(n, -5, 5)
z <- rbinom(n, 1, .5)
dgp <- sin(sqrt(x1^2 + x2^2 + z))/
       sqrt(x1^2 + x2^2 + z)
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

We again vary the signal/noise ratio by standardizing the DGP and setting σ for the Gaussian noise to $1/4, 1/2, 1, 2$ which produces expected in-sample fits of $R^2 = 0.95, 0.80, 0.50, 0.20$ for the oracle estimator (i.e. one that uses knowledge of the DGP). We estimate the following eight models,

```
model <- crs(y ~ x1 + x2 + z)
# gam (mgcv) additive
model <- gam(y ~ s(x1) + s(x2) + z)
# gam (mgcv) tensor to admit interactions
model <- gam(y ~ t2(x1, x2, k=k) + z)
# gam (mgcv) tensor with smooth for every z
model <- gam(y ~ t2(x1, x2, k=k, by=z) + z)
# spm (SemiPar) additive
model <- spm(y ~ f(x1) + f(x2) + z)
# spm (SemiPar) tensor product
model <- spm(y ~ f(x1, x2) + z)
# ssanova (gss) additive
model <- ssanova(y ~ x1 + x2 + z)
# ssanova (gss) tensor product
model <- ssanova(y ~ x1 * x2 + z)
```

We draw $M = 1000$ replications of size $n = 1000$ from each DGP, compute the MSE for each model, and report the median value of each model's MSE relative to that for `crs` over the M replications in Table 2 below.

Table 2: MSE efficiency of the various methods relative to `crs` (numbers less than one indicate more efficient than `crs`).

Additive DGP ($\cos(2\pi x_1) + \sin(2\pi x_2) + z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	0.60	1.57	2.11	0.64	1.71	0.62	0.70
0.50	0.57	0.94	1.56	0.57	1.24	0.57	0.65
1.00	0.55	0.83	1.44	0.51	0.95	0.54	0.65
2.00	0.52	0.75	1.35	0.49	0.72	0.51	0.60

Additive DGP ($\cos(4\pi x_1) + \sin(4\pi x_2) + z$))

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	0.72	1.07	1.88	0.77	5.76	0.65	0.72
0.50	0.54	0.94	1.70	0.63	2.50	0.58	0.65
1.00	0.52	0.90	1.65	0.57	1.53	0.57	0.62
2.00	0.53	0.90	1.64	0.56	1.17	0.55	0.63

Multiplicative DGP ($\cos(2\pi x_1) \times \sin(2\pi x_2) \times z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	228.82	111.54	0.78	229.05	111.41	229.17	112.97
0.50	95.18	46.93	1.09	95.26	46.81	95.31	47.46
1.00	30.23	15.50	1.16	30.25	15.55	30.25	15.60
2.00	9.61	5.39	1.14	9.60	5.49	9.59	5.32

Multiplicative DGP ($\cos(4\pi x_1) \times \sin(4\pi x_2) \times z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	93.39	44.36	0.62	93.52	51.31	93.58	54.14
0.50	30.05	14.66	0.64	30.09	16.85	30.11	17.71
1.00	9.92	5.28	0.74	9.93	5.92	9.94	6.56
2.00	3.40	2.26	0.86	3.39	2.50	3.39	3.37

Multiplicative DGP (radial)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	89.66	2.18	1.16	89.07	2.29	89.60	2.21
0.50	31.35	1.27	1.30	31.18	1.29	31.35	1.21
1.00	12.65	1.19	1.72	12.56	1.09	12.68	1.08
2.00	4.49	1.10	1.82	4.44	0.88	4.51	0.99

For the **mgcv** routine `gam`, a referee noted that `?choose.k` could be consulted where it suggests using `gam.check()` to help guide the appropriate number of knots and suggested non-stochastic values of `k=5` for the additive DGP with 2π and `k=10` for the remaining DGPs (these values were used in Table 2

for gam via the argument k=k rather than the default values, while crs uses stochastic values for the basis and all smoothing parameters).

When the DGP is additive, it is clear that the smoothing splines that presume additivity are more efficient than crs for this DGP. Note that this is an oracle type of property in the sense that the user does not in general have knowledge of the true DGP, but if they did and used this information their model would naturally be more efficient than a method that did not (we elaborate further on this below). The regression spline implementation in the **crs** package does not presume this knowledge when basis="auto" is used (the default) though it allows for the possibility. However, were one to use the incorrect smoothing spline model when the DGP is additive (e.g. assume interaction when it is not present), one could do worse than crs as can be seen in Table 2. But note that when the DGP is non-additive, crs appears to be more efficient than the smoothing spline approaches even when they employ, say, tensor basis functions to model the non-additive relationship and are adapted to admit the presence of the binary factor predictor (except gam which, as noted above, uses a number of knots k that is optimized for these DGPs, rather than the defaults which were used for all other methods). From a minimax perspective, the regression spline implementation in the **crs** package performs relatively well overall. Apart from selecting the number of knots for gam as described above, default settings were used throughout and therefore it is likely that efficiency could be improved in some cases by further tuning (this would apply to crs as well, naturally).

To elaborate further on the issue of tuning and efficiency, if instead of using defaults for crs we were to assume additivity as done for its peers in columns 2, 5, and 7 in Table 2 (options basis="additive", kernel=FALSE), then for the additive DGP in Table 2 the first row would instead be 0.94, 2.43, 3.29, 1.00, 2.69, 0.98, and 1.08, and we observe that the efficiency of crs improves substantially even when using a stochastic choice of the B-spline degree and number of knots, while the other methods use non-stochastic choices for the number of knots that are constant over all $M = 1{,}000$ Monte Carlo replications. And if we followed the lead of the anonymous referee who kindly suggested using non-stochastic values for the number of knots (i.e. k=5) for gam in **mgcv** based on replications from the additive DGP used for row 1 of Table 2, we might choose non-stochastic values degree=c(4,4) and segments=c(3,3) based on cross-validation and the first row of Table 2 would instead be 1.18, 3.09, 4.19, 1.25, 3.42, 1.22, and 1.36. Furthermore, if in addition we used the option prune=TRUE enabling post-estimation pruning of the model, the first row of Table 2 would instead be 1.48, 3.91, 5.35, 1.59, 4.28, 1.53, and 1.72 (note that this can only be used in conjunction with the option kernel=FALSE). The point to be made is that indeed efficiency can be improved in some cases by tuning of smoothing parameters and choice of the basis, and this also holds for crs itself.

Demos, data, and additional information

There exist a range of demonstrations available via demo() including

1. radial_persp: R code for fitting and plotting a 3D perspective plot for the 'radial' function.
2. radial_rgl: R code for fitting and generating a 3D real-time rendering plot for the 'radial' function using OpenGL (requires the **rgl** package (Adler and Murdoch, 2012)).
3. sine_rgl: R code for fitting and generating a 3D real-time rendering plot for a product sine function using OpenGL.
4. radial_constrained_mean: R code for constrained radial function estimation.
5. cqrs.R: R code for fitting and plotting quantile regression splines.

There exist a number of datasets including

1. cps71: Canadian cross-section wage data consisting of a random sample taken from the 1971 Canadian Census Public Use Tapes for male individuals having common education (grade 13). There are 205 observations in total (contributed by Aman Ullah).
2. Engel95: British cross-section data consisting of a random sample taken from the British Family Expenditure Survey for 1995. The households consist of married couples with an employed head-of-household between the ages of 25 and 55 years. There are 1655 household-level observations in total (contributed by Richard Blundell).
3. wage1: Cross-section wage data consisting of a random sample taken from the U.S. Current Population Survey for the year 1976. There are 526 observations in total (contributed by Jeffrey Wooldridge).

We have tried to overload the crs method and associated S3 plot method to accommodate the needs of a range of users and to automate a number of routine tasks. Warning messages are produced where possible to guide the user towards the most appropriate choice of settings. User specified weights can be provided to allow for weighted least squares and weighted cross-validation. In addition, we have a function crsiv that implements instrumental variable regression splines that may be of interest to

some users (see `?crsiv` for details). Furthermore, quantile regression splines are supported by setting `tau=` a scalar in the range $(0,1)$. Finally, we would like to direct the reader to the vignettes `crs`, `crs_faq`, and `spline_primer` for further examples, frequently asked questions, and background information.

Acknowledgements

Racine would like to gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada (http://www.nserc.ca), the Social Sciences and Humanities Research Council of Canada (http://www.sshrc.ca), and the Shared Hierarchical Academic Research Computing Network (http://www.sharcnet.ca). We would also like to gratefully acknowledge Sébastien Le Digabel and members of the NOMAD project for their contributions to the FOSS community. Finally, we are indebted to the editors and two anonymous referees for their invaluable comments and suggestions.

Bibliography

M. Abramson, C. Audet, G. Couture, J. Dennis Jr., and S. Le Digabel. The NOMAD project, 2011. URL http://www.gerad.ca/nomad. [p48, 50, 51, 52]

D. Adler and D. Murdoch. *rgl: 3D visualization device system (OpenGL)*, 2012. URL http://CRAN.R-project.org/package=rgl. R package version 0.92.892. [p55]

J. Aitchison and C. G. G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976. [p50]

P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 13:377–403, 1979. [p48, 51]

M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*, third edition, January 2009. URL http://http://www.gnu.org/software/gsl/. Version 1.12. [p49]

C. Gu. *gss: General Smoothing Splines*, 2012. URL http://CRAN.R-project.org/package=gss. R package version 2.0-9. [p49]

P. Hall, J. S. Racine, and Q. Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004. [p48]

P. Hall, Q. Li, and J. S. Racine. Nonparametric estimation of regression functions in the presence of irrelevant regressors. *The Review of Economics and Statistics*, 89:784–789, 2007. [p48]

T. Hastie. *gam: Generalized Additive Models*, 2011. URL http://CRAN.R-project.org/package=gam. R package version 1.06.2. [p49]

C. M. Hurvich, J. S. Simonoff, and C. L. Tsai. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society Series B*, 60:271–293, 1998. [p48, 51]

S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):44:1–44:15, 2011. [p48, 50, 51, 52]

S. Ma and J. S. Racine. Additive regression splines with irrelevant categorical and continuous regressors. *Statistica Sinica*, 2012. In Press. [p48]

S. Ma, J. S. Racine, and L. Yang. Spline regression in the presence of categorical predictors. *Journal of Multivariate Analysis*, 2012. Revised and Resubmitted. [p48]

J. S. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004. [p48]

C. J. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society*, 36:111–147, 1974. [p48]

C. J. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5:595–645, 1977. [p48]

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. URL http://projects.coin-or.org/ipopt. [p52]

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990. [p49]

M. Wand. *SemiPar: Semiparametic Regression*, 2010. URL http://CRAN.R-project.org/package=SemiPar. R package version 1.0-3. [p49]

S. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006. [p49]

S. Zhou and D. A. Wolfe. On derivative estimation in spline regression. *Statistica Sinica*, 10:93–108, 2000. [p49]

Jeffrey S. Racine, Zhenghua Nie
McMaster University
1280 Main Street West, Hamilton, Ontario
Canada
racinej@mcmaster.ca
niez@mcmaster.ca