The crs Package

Jeffrey S. Racine
McMaster University

Abstract

This vignette outlines the implementation of the regression spline method contained in the R **crs** package, and also presents a few illustrative examples.

Keywords: nonparametric, semiparametric, regression spline, categorical data.

Introduction

The **crs** package implements a framework for nonparametric regression splines that admits both continuous and categorical predictors. 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 consume degrees of freedom via the number of columns in the spline basis matrix, while kernel weighting does not.

This package implements the approaches described in Ma, Racine, and Yang 2011 and Ma and Racine 2011 when the option `kernel=TRUE` is selected as described below. As well, this package implements a range of related methods and has options that (hopefully) make it appealing for applied projects, research, and pedagogical purposes alike.

Data-driven methods can be used for selecting the spline degree, number of segments/knots, and bandwidths (leave-out-out cross-validation (`cv.func = "cv.ls"`) Stone 1974, Stone 1977, generalized cross-validation (`cv.func="cv.gcv"`) Craven and Wahba 1979, and the information-based criterion (`cv.func="cv.aic"`) proposed by Hurvich, Simonoff, and Tsai 1998). Details of the implementation are as follows:

(i) 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)

(ii) 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 minus one 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)

(iii) 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

(iv) 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 (Hall, Racine, and Li 2004, Hall, Li, and Racine 2007)

The design philosophy of the **crs** package aims to closely mimic the behavior of the `lm` function. Indeed, the implementation relies on `lm` for computation of the spline coefficients, obtaining 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.

Implementation

Kernel-based methods such as local constant (i.e. the Nadaraya 1965 Watson 1964 estimator) and local polynomial regression (Fan and Gijbels 1996) are based on the concept of 'local' weighted averaging. Regression spline methods, on the other hand, are 'global' in nature since a single least square procedure leads to the ultimate function estimate over the entire data range (Stone 1994). This 'global nature' implies that constructing regression splines will be less computationally burdensome that their kernel-based counterparts leading to their practical appeal relative to kernel-based approaches.

However, while kernel-based regression methods admit a rich array of predictor types, spline regression methods can be limited in their potential applicability as they are predicated on *continuous* predictors only. In applied settings we often encounter *categorical* predictors such as strength of preference ("strongly prefer", "weakly prefer", "indifferent" etc.) and so forth. When confronted with categorical predictors, researchers typically break their data into subsets governed by the values of the categorical predictors (i.e. they break their data into 'cells') and then conduct regression using only the response and continuous predictors lying in each cell. Though consistent, this 'frequency' approach can be inefficient. Recent developments in the kernel smoothing of categorical data (Li and Racine 2007) suggest more efficient estimation approaches in such settings. The **crs** package considers two complementary approaches that seamlessly handle the mix of continuous and categorical predictors often encountered in applied settings.

The Underlying Model

We presume the reader wishes to model the unknown conditional mean in the following location-scale model,

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

where $g(\cdot)$ is an unknown function, $\mathbf{X} = (X_1, \ldots, X_q)^{\mathrm{T}}$ is a q-dimensional vector of continuous predictors, and $\mathbf{Z} = (Z_1, \ldots, Z_r)^{\mathrm{T}}$ is an r-dimensional vector of categorical predictors. 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 let c_s be a finite positive constant.

For the continuous predictors the regression spline model employs the B-spline routines in the GNU Scientific Library (`http://www.gnu.org/software/gsl/`). The B-spline function is the maximally differentiable interpolative basis function (B-spline stands for 'basis-spline'), and a B-spline with no internal knots is a Bézier curve.

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 for every 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 (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) and include instead an intercept in the model. This allows multiple bases to coexist when there is more than one continuous predictor without introducing singularities. 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. A general pth order local polynomial estimator for the *multivariate* predictor case is more cumbersome notationally speaking (we let q denote the number of continuous predictors). The general multivariate case is considered by Masry 1996 who develops some carefully considered notation and establishes the uniform almost sure convergence rate and the pointwise asymptotic normality result for the local polynomial estimator of $g(x)$ and its derivatives up to order p. Borrowing from Masry 1996, we introduce the following notation:

$$r = (r_1, \ldots, r_q), \quad r! = r_1! \times \cdots \times r_q!, \quad \bar{r} = \sum_{j=1}^{q} r_j,$$

$$x^r = x_1^{r_1} \times \cdots \times x_q^{r_q}, \quad \sum_{0 \le \bar{r} \le p} = \sum_{j=0}^{p} \sum_{r_1=0}^{j} \cdots \sum_{r_q=0}^{j}, \quad (\text{with } \bar{r} \equiv r_1 + \cdots + r_q = j) \tag{1}$$

and

$$(D^r g)(x) = \frac{\partial^r g(x)}{\partial x_1^{r_1} \ldots \partial x_q^{r_q}}.$$

Using this notation, and assuming that $g(x)$ has derivatives of total order $p + 1$ at a point x, we can approximate $g(z)$ locally using a multivariate polynomial of total order p given by

$$g(z) \cong \sum_{0 \le \bar{r} \le p} \frac{1}{r!} (D^r) g(v)|_{v=x} (z - x)^r.$$

The generalized (varying-order) involves using the following expression rather than (1) above,

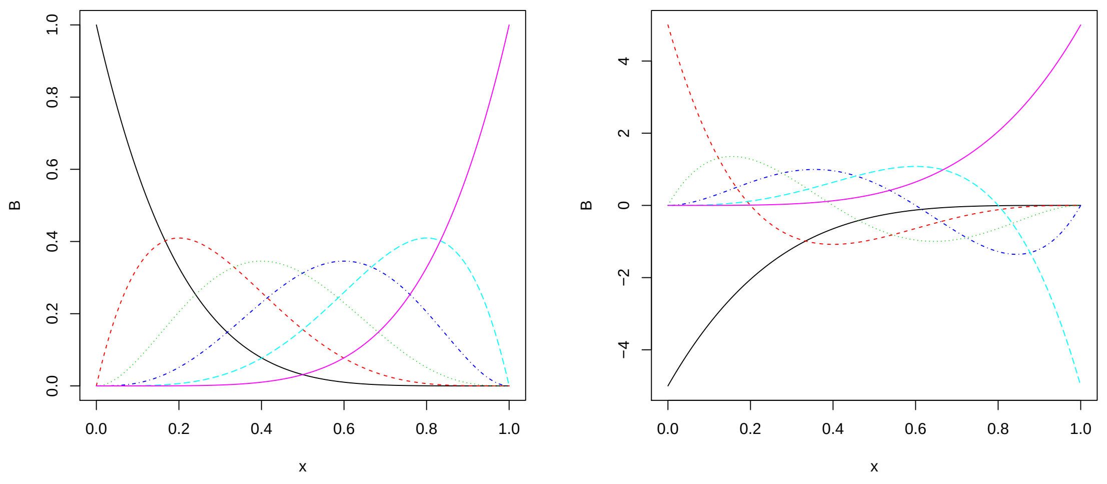
$$\sum_{0\leq\bar{r}\leq\max\{p_1,\dots,p_q\}} = \sum_{j=0}^{\max\{p_1,\dots,p_q\}}\sum_{r_1=0}^{j\leq p_1}\cdots\sum_{r_q=0}^{j\leq p_q}.$$

When additive B-spline bases are employed we have a semiparametric 'additive' spline model (no interaction among variables), otherwise when the tensor product is employed we have a fully nonparametric model (interaction among all variables). Whether to use the additive or tensor product or generalized polynomial bases can be automatically determined via any `cv.func` method (see the options for `basis=` in `?crs`).

We offer the 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 (again taking care to remove one column to avoid singularity given the presence of the intercept term in the model). These bases are then treated similar to the bases B_j for continuous predictors described above.

Example: A B-spline and its First Derivative.

The figure below presents an example of a B-spline and its first derivative (the spline derivatives are required in order to compute derivatives from the spline regression model).



Above we plot a degree-5 B-spline (left) with one segment (two knots) and its 1st-order derivative (right).

Least-Squares Estimation of the Underlying Model

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

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

Placement of Knots

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

(i) knots can be placed at equally spaced quantiles whereby an equal number of observations lie in each segment ('quantile knots')

(ii) knots can be placed at equally spaced intervals ('uniform knots')

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 $D_s \overset{def}{=} \{0, 1, \dots, c_s - 1\}$, $s = 1, \dots, r$, and let $c_s \geq 2$ be a finite positive constant. For expositional simplicity we will consider the case in which the components of z are unordered.

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

$$l(Z_{is}, z_s, \lambda_s) = \begin{cases} 1, & \text{when } Z_{is} = z_s, \\ \lambda_s, & \text{otherwise.} \end{cases} \tag{2}$$

Let $\mathbf{1}(A)$ denote the usual indicator function, which assumes the value one if A holds true, zero otherwise. Using (2), we can construct a product kernel function given by

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

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.

Note that when $\lambda_s = 1$ all observations are 'pooled' 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.

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. 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', see (Abramson, Audet, Couture, Dennis Jr., and Le Digabel 2011) and (Le Digabel 2011) and the references therein).

The degree and bandwidth vector can be jointly determined via any `cv.func` method by setting the option `cv="nomad"` or `cv="exhaustive"` (exhaustive search).

Setting the option `cv="nomad"` 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` setting the option `cv="exhaustive"` computes bandwidths ($\in [0,1]$) obtained via numerical minimization (see `optim`) for each unique combination of the degree and segment vectors (restarting can be conducted via `restarts=`). When conducting `cv="nomad"` the number of multiple starts can be controlled by `nmulti=`. The model possessing the lowest criterion function value 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 (say, 10 or larger) to strive to avoid becoming trapped in local minima.

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 spline regression 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 by many statistics programs, 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 $\operatorname{tr}(H)$ 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 $\operatorname{tr}(H)/n$ (Craven and Wahba 1979).

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

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

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 is lower than the model being pruned.* When this is not the case a warning is issued to this effect. A final pruning stage is commonplace in the spline framework 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`.

Illustrative Examples

Next we provide a few illustrative examples that may be of interest to the reader.

Example: One Categorical/One Continuous Predictor

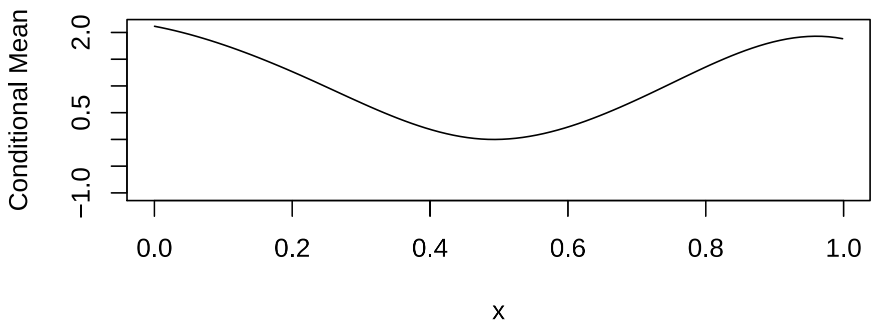
By way of illustration we consider a simple example involving one continuous and one discrete predictor.

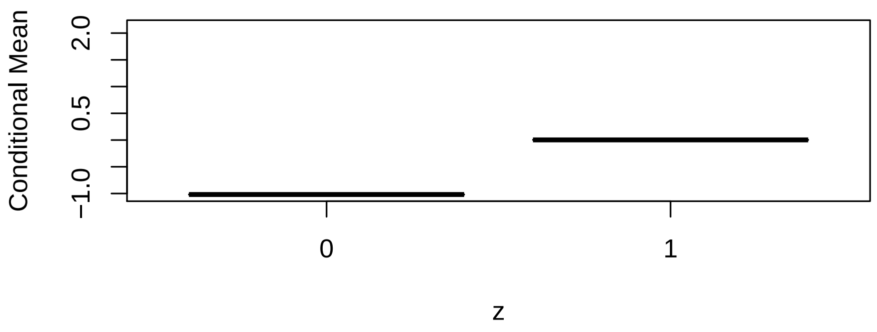
```
R> set.seed(42)
R> n <- 1000
R> x <- runif(n)
R> z <- rbinom(n,1,.5)
R> y <- cos(2*pi*x) + z + rnorm(n,sd=0.25)
R> z <- factor(z)
R> model <- crs(y~x+z)

starting point # 0: (  1  1 0.9148060435 )
starting point # 1: (  8  1 0.01070872545 )
starting point # 2: (  6  8 0.4900544537 )
starting point # 3: (  4  5 0.2815487627 )
starting point # 4: (  2  6 0.6452028642 )


run # 0: f=0.06170639839

run # 1: f=0.06149153131

run # 2: f=0.06149153131

run # 3: f=0.06170639839

run # 4: f=0.06149153131

bb eval : 741
best    : 0.06149153131
worst   : 0.06170639839
solution: x = ( 3 4 0.0008551836014 ) f(x) = 0.06149153131
```

```
R> 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: 0.000855
Model complexity proxy: degree-knots
Knot type: quantiles
Training observations: 1000
Rank of model frame: 7
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
Estimation time: 5.6 seconds
```

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=FALSE`, `ci=FALSE`, `plot.behavior` = `c("plot", "plot-data", "data")`) support objects of this type. The figure below 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)`.





Example: Regression Discontinuity Design

By way of illustration we consider a simple example involving two continuous predictors and one categorical predictor. In this example there is a ‘discontinuity’ in the regression surface potentially demarcated by the discrete predictor.

```
R> set.seed(1234)
R> n <- 1000
R> x1 <- runif(n)
R> x2 <- runif(n)
```

```
R> z <- ifelse(x1>.5,1,0)
R> dgp <- cos(2*pi*x1)+sin(2*pi*x2)+2*z
R> z <- factor(z)
R> y <- dgp + rnorm(n,sd=1)
R> model <- crs(y~x1+x2+z)

starting point # 0: (  1  1  1  1 0.9148060435 )
starting point # 1: (  8  8  8  1 0.05344109271 )
starting point # 2: (  2  4  5  4 0.4737019 )
starting point # 3: (  6  2  3  8 0.2844453919 )
starting point # 4: (  4  6  1  6 0.6078332779 )


run # 0: f=0.9746476669

run # 1: f=0.9754720192

run # 2: f=0.9755871031

run # 3: f=0.97683374

run # 4: f=0.9746476653

bb eval : 1603
best    : 0.9746476653
worst   : 0.97683374
solution: x = ( 3 3 1 1 0.0006024122238 ) f(x) = 0.9746476653

R> summary(model)

Call:
crs.formula(formula = y ~ x1 + x2 + z)

Kernel Weighting/B-spline Bases Regression Spline

There are 2 continuous predictors
There is 1 categorical predictor
Spline degree/number of segments for x1: 3/1
Spline degree/number of segments for x2: 3/1
Bandwidth for z: 0.000602
Model complexity proxy: degree-knots
Knot type: quantiles
Basis type: additive
Training observations: 1000
Rank of model frame: 7
```

```
Trace of smoother matrix: 12

Residual standard error: 0.9783 on 993 degrees of freedom
Multiple R-squared: 0.6718,   Adjusted R-squared: 0.6698
F-statistic: 183.8 on 11 and 988 DF, p-value: 4.354e-230

Cross-validation score: 0.97464767
Number of multistarts: 5
Estimation time: 75.2 seconds
```

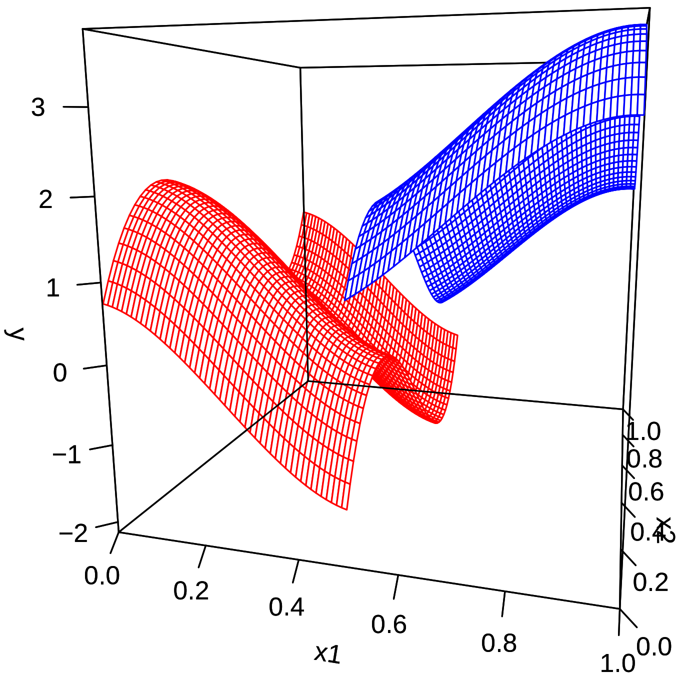
The figure below plots the resulting estimate. The discontinuity occurs when $x_1 > 0.5$ but the nature of the discontinuity is unknown as is the functional form on either side of the potential discontinuity. The categorical regression spline is able to detect this 'break' and testing for a significant break involves nothing more than an (asymptotic) F-test as the following illustrates (note the argument `include=0` says to drop the one categorical predictor or, say, `c(1,1,...,0,1...,1)` for multivariate categorical predictors).

```
R> ## When kernel=FALSE, we could use the anova() function
R> model.res <- crs(y~x1+x2+z,cv="none",degree=model$degree,basis=model$basis,include=0)
R> ## anova(model.res$model.lm,model$model.lm)
R> ## We could also do this manually...
R> F <- model$df.residual*(sum(residuals(model.res)^2)
+                         -sum(residuals(model)^2))/sum(residuals(model)^2)
R> F
```

```
[1] -1.54
```

```
R> ## Compute the P-value for the F-statistic
R> P <- 1-pf(F,1,model$df.residual)
R> P
```

```
[1] 1
```



Acknowledgements

I 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`).

References

Abramson M, Audet C, Couture G, Dennis Jr J, Le Digabel S (2011). "The NOMAD project." *Technical report*. URL `Softwareavailableathttp://www.gerad.ca/nomad`.

Aitchison J, Aitken CGG (1976). "Multivariate Binary Discrimination by the Kernel Method." *Biometrika*, **63**(3), 413–420.

Craven P, Wahba G (1979). "Smoothing Noisy Data with Spline Functions." *Numerische Mathematik*, **13**, 377–403.

Fan J, Gijbels I (1996). *Local Polynomial Modelling and Its Applications.* Chapman and Hall, London.

Hall P, Li Q, Racine JS (2007). "Nonparametric Estimation of Regression Functions in the Presence of Irrelevant Regressors." *The Review of Economics and Statistics*, **89**, 784–789.

Hall P, Racine JS, Li Q (2004). "Cross-Validation and the Estimation of Conditional Probability Densities." *Journal of the American Statistical Association*, **99**(468), 1015–1026.

Hurvich CM, Simonoff JS, Tsai CL (1998). "Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion." *Journal of the Royal Statistical Society Series B*, **60**, 271–293.

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

Li Q, Racine J (2007). *Nonparametric Econometrics: Theory and Practice.* Princeton University Press.

Ma S, Racine JS (2011). "Additive Regression Splines With Irrelevant Categorical and Continuous Regressors." *Working paper*, McMaster University and Michigan State University.

Ma S, Racine JS, Yang L (2011). "Spline Regression in the Presence of Categorical Predictors." *Working paper*, McMaster University and Michigan State University.

Masry E (1996). "Multivariate local polynomial regression for time series: uniform strong consistency and rates." *Journal of Time Series Analysis*, **17**, 571–599.

Nadaraya EA (1965). "On Nonparametric Estimates of Density Functions and Regression Curves." *Theory of Applied Probability*, **10**, 186–190.

Racine JS, Li Q (2004). "Nonparametric Estimation of Regression Functions with both Categorical and Continuous Data." *Journal of Econometrics*, **119**(1), 99–130.

Stone CJ (1974). "Cross-Validatory Choice and Assessment of Statistical Predictions (with discussion)." *Journal of the Royal Statistical Society*, **36**, 111–147.

Stone CJ (1977). "Consistent Nonparametric Regression." *Annals of Statistics*, **5**, 595–645.

Stone CJ (1994). "The use of polynomial splines and their tensor products in multivariate function estimation." *Annals of Statistics*, **22**, 118–184.

Watson GS (1964). "Smooth Regression Analysis." *Sankhya*, **26:15**, 359–372.

Zhou S, Wolfe DA (2000). "On derivative estimation in spline regression." *Statistica Sinica*, **10**, 93–108.

Affiliation:

Jeffrey S. Racine
Department of Economics

McMaster University
Hamilton, Ontario, Canada, L8S 4L8
E-mail: racinej@mcmaster.ca
URL: http://www.mcmaster.ca/economics/racine/