A PRIMER ON REGRESSION SPLINES

JEFFREY S. RACINE

1. Overview

B-splines constitute an appealing method for the nonparametric estimation of a range of statistical objects of interest. In this primer we focus our attention on the estimation of a conditional mean, i.e. the 'regression function'.

A 'spline' is a function that is constructed piece-wise from polynomial functions. The term comes from the tool used by shipbuilders and drafters to construct smooth shapes having desired properties. Drafters have long made use of a bendable strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points. The malleability of the spline material combined with the constraint of the control points would cause the strip to take the shape that minimized the energy required for bending it between the fixed points, this being the smoothest possible shape. We shall rely on a class of splines called 'B-splines' ('basis-splines'). A B-spline function is the maximally differentiable interpolative basis function. The B-spline is a generalization of the Bézier curve (a B-spline with no 'interior knots' is a Bézier curve). B-splines are defined by their 'order' m and number of interior 'knots' N (there are two 'endpoints' which are themselves knots so the total number of knots will be $N + 2$). The degree of the B-spline polynomial will be the spline order m minus one (degree $= m - 1$).

To best appreciate the nature of B-splines, we shall first consider a simple type of spline, the Bézier function, and then move on to the more flexible and powerful generalization, the B-spline itself. We begin with the univariate case in Section 2 where we consider the univariate Bézier function. In Section 3 we turn to the univariate B-spline function, and then in Section 4 we turn to the multivariate case where we also briefly mention how one could handle the presence of categorical predictors.

We presume that interest lies in 'regression spline' methodology which differs in a number of ways from 'smoothing splines', both of which are popular in applied settings. The fundamental difference between the two approaches is that smoothing splines explicitly penalize roughness and use the data points themselves as potential knots whereas regression splines place knots at equidistant/equiquantile points. We direct the interested reader to Wahba (1990) for a treatment of smoothing splines.

Date: March 25, 2026.

These notes are culled from a variety of sources. I am solely responsible for all errors. Suggestions are welcomed (racinej@mcmaster.ca).

2. Bézier curves

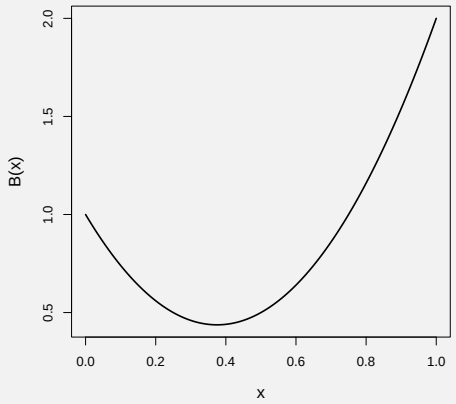
We present an overview of Bézier curves which form the basis for the B-splines that follow. We begin with a simple illustration, that of a quadratic Bézier curve.

Example 2.1. A quadratic Bézier curve.

A quadratic Bézier curve is the path traced by the function $B(x)$, given points β_0, β_1, and β_2, where

$$\begin{aligned} B(x) &= \beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2 x^2 \\ &= \sum_{i=0}^{2} \beta_i B_i(x), \quad x \in [0,1]. \end{aligned}$$

The terms $B_0(x) = (1-x)^2$, $B_1(x) = 2(1-x)x$, and $B_2(x) = x^2$ are the 'bases' which is this case turn out to be 'Bernstein polynomials' (Bernstein 1912). For our purposes the 'control points' β_i, $i = 0, 1, 2$, will be parameters that could be selected by least squares fitting in a regression setting, but more on that later. Consider the following simple example where we plot a quadratic Bézier curve with arbitrary control points:



For this simple illustration we set $\beta_0 = 1$, $\beta_1 = -1$, $\beta_2 = 2$.

Note that the derivative of this curve is

$$B'(x) = 2(1-x)(\beta_1 - \beta_0) + 2x(\beta_2 - \beta_1),$$

which is a polynomial of degree one.

This example of a Bézier curve will also be seen to be a 'second-degree B-spline with no interior knots' or, equivalently, 'a third-order B-spline with no interior knots'. Using the terminology of B-splines, in this example we have a third-order B-spline ($m = 3$) which is of polynomial degree two ($m - 1 = 2$) having highest derivative of polynomial degree one ($m - 2 = 1$).

2.1. **The Bézier curve defined.** More generally, a Bézier curve of degree n (order m) is composed of $m = n + 1$ terms and is given by

$$\begin{aligned} B(x) &= \sum_{i=0}^{n} \beta_i \binom{n}{i} (1-x)^{n-i} x^i \\ &= \sum_{i=0}^{n} \beta_i B_{i,n}(x), \end{aligned} \tag{1}$$

where $\binom{n}{i} = \frac{n!}{(n-i)!i!}$, which can be expressed recursively as

$$B(x) = (1-x) \left(\sum_{i=0}^{n-1} \beta_i B_{i,n-1}(x) \right) + x \left(\sum_{i=1}^{n} \beta_i B_{i-1,n-1}(x) \right),$$

so a degree n Bézier curve is a linear interpolation between two degree $n-1$ Bézier curves.

Example 2.2. A quadratic Bézier curve as a linear interpolation between two linear Bézier curves.

The linear Bézier curve is given by $\beta_0(1-x) + \beta_1 x$, and above we showed that the quadratic Bézier curve is given by $\beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2 x^2$. So, when $n = 2$ (quadratic), we have

$$\begin{aligned} B(x) &= (1-x)(\beta_0(1-x) + \beta_1 x) + x(\beta_1(1-x) + \beta_2 x) \\ &= \beta_0(1-x)^2 + 2\beta_1(1-x)x + \beta_2 x^2. \end{aligned}$$

This is essentially a modified version of the idea of taking linear interpolations of linear interpolations of linear interpolations and so on. Note that the polynomials

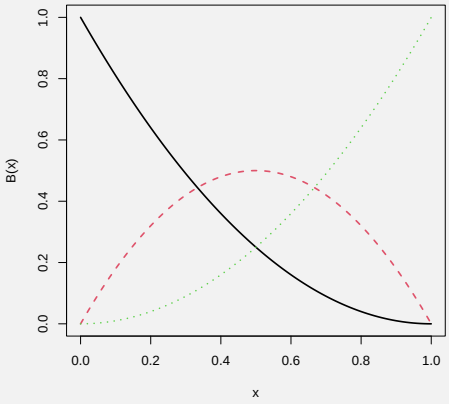
$$B_{i,n}(x) = \binom{n}{i} (1-x)^{n-i} x^i$$

are called 'Bernstein basis polynomials of degree n' and are such that $\sum_{i=0}^{n} B_{i,n}(x) = 1$, unlike raw polynomials.[1]

The $m = n + 1$ control points β_i, $i = 0, \ldots, n$, are somewhat ancillary to the discussion here, but will figure prominently when we turn to regression as in a regression setting they will be the coefficients of the regression model.

[1]Naturally we define $x^0 = (1-x)^0 = 1$, and by 'raw' polynomials we simply mean x^j, $j = 0, \ldots, n$.

Example 2.3. The quadratic Bézier curve basis functions.
The figure below presents the bases $B_{i,n}(x)$ underlying a Bézier curve for $i = 0, \ldots, 2$ and $n = 2$.



These bases are $B_{0,2}(x) = (1-x)^2$, $B_{1,2}(x) = 2(1-x)x$, and $B_{2,2}(x) = x^2$ and illustrate the foundation upon which the Bézier curves are built.

2.2. **Derivatives of spline functions.** From de Boor (2001) we know that the derivatives of spline functions can be simply expressed in terms of lower order spline functions. In particular, for the Bézier curve we have

$$B^{(l)}(x) = \sum_{i=0}^{n-l} \beta_i^{(l)} B_{i,n-l}(x),$$

where $\beta_i^{(0)} = \beta_i$, $0 \leq i \leq n$, and

$$\beta_i^{(l)} = (n-l)\left(\beta_{i+1}^{(l-1)} - \beta_i^{(l-1)}\right)/(t_i - t_{i-n+l}), \quad 0 \leq i \leq n-l.$$

See Zhou & Wolfe (2000) for details.

We now turn our attention to the B-spline function. This can be thought of as a generalization of the Bézier curve where we now allow for there to be additional breakpoints called 'interior knots'.

3. B-splines

3.1. **B-spline knots.** B-spline curves are composed from many polynomial pieces and are therefore more versatile than Bézier curves. Consider $N+2$ real values t_i, called 'knots' ($N \geq 0$ are called 'interior knots' and there are always two endpoints, t_0 and t_{N+1}), with

$$t_0 \leq t_1 \leq \cdots \leq t_{N+1}.$$

When the knots are equidistant they are said to be 'uniform', otherwise they are said to be 'non-uniform'. One popular type of knot is the 'quantile' knot sequence where the interior knots are the quantiles from the empirical distribution of the underlying variable. Quantile knots guarantee that

an equal number of sample observations lie in each interval while the intervals will have different lengths (as opposed to different numbers of points lying in equal length intervals).

Bézier curves possess two endpoint knots, t_0 and t_1, and no interior knots hence are a limiting case, i.e. a B-spline for which $N = 0$.

3.2. **The B-spline basis function.** Let $\boldsymbol{t} = \{t_i \mid i \in \mathbb{Z}\}$ be a sequence of non-decreasing real numbers ($t_i \leq t_{i+1}$) such that[2]

$$t_0 \leq t_1 \leq \cdots \leq t_{N+1}.$$

Define the augmented knot set

$$t_{-(m-1)} = \cdots = t_0 \leq t_1 \leq \cdots \leq t_N \leq t_{N+1} = \cdots = t_{N+m},$$

where we have appended the lower and upper boundary knots t_0 and t_{N+1} $n = m - 1$ times (this is needed due to the recursive nature of the B-spline). If we wanted we could then reset the index for the first element of the augmented knot set (i.e. $t_{-(m-1)}$) so that the $N + 2m$ augmented knots t_i are now indexed by $i = 0, \ldots, N + 2m - 1$ (see the example below for an illustration).

For each of the augmented knots t_i, $i = 0, \ldots, N+2m-1$, we recursively define a set of real-valued functions $B_{i,j}$ (for $j = 0, 1, \ldots, n$, n being the degree of the B-spline basis) as follows:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

$$B_{i,j+1}(x) = \alpha_{i,j+1}(x) B_{i,j}(x) + [1 - \alpha_{i+1,j+1}(x)] B_{i+1,j}(x),$$

where

$$\alpha_{i,j}(x) = \begin{cases} \dfrac{x - t_i}{t_{i+j} - t_i} & \text{if } t_{i+j} \neq t_i \\ 0 & \text{otherwise.} \end{cases}$$

For the above computation we define $0/0$ as 0.

Definitions.

Using the notation above:

(1) the sequence $\boldsymbol{t}$ is known as a *knot sequence*, and the individual term in the sequence is a *knot*.
(2) the functions $B_{i,j}$ are called the *i-th B-spline basis functions of degree j*, and the recurrence relation is called the *de Boor recurrence relation*, after its discoverer Carl de Boor (de Boor 2001).
(3) given any non-negative integer j, the vector space $V_j(\boldsymbol{t})$ over $\mathbb{R}$, generated by the set of all B-spline basis functions of degree j is called the *B-spline of degree j*. In other words, the B-spline $V_j(\boldsymbol{t}) = \text{span}\{B_{i,j}(x) \mid i = 0, 1, \ldots\}$ over $\mathbb{R}$.
(4) Any element of $V_j(\boldsymbol{t})$ is a *B-spline function* of degree j.

[2]This description is based upon the discussion found at http://planetmath.org/encyclopedia/BSpline.html.

The first term $B_{0,n}$ is often referred to as the 'intercept'. In typical spline implementations the option `intercept=FALSE` denotes dropping this term while `intercept=TRUE` denotes keeping it (recall that $\sum_{i=0}^{n} B_{i,n}(x) = 1$ which can lead to perfect multicollinearity in a regression setting; also see Zhou & Wolfe (2000) who instead apply shrinkage methods).

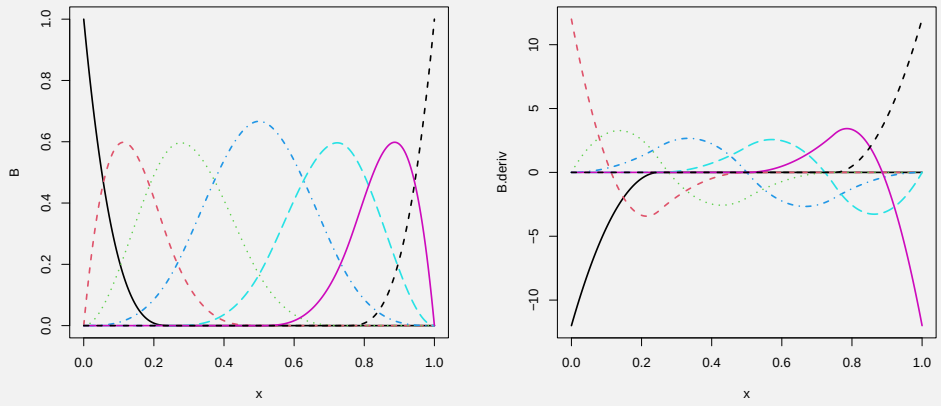
Example 3.4. A fourth-order B-spline basis function with three interior knots and its first derivative function.

Suppose there are $N = 3$ interior knots given by $(0.25, 0.5, 0.75)$, the boundary knots are $(0, 1)$, and the degree of the spline is $n = 3$ hence the order is $m = 4$. The set of all knot points needed to construct the B-spline is

$$(0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1)$$

and the number of basis functions is $K = N + m = 7$. The seven cubic spline basis functions will be denoted $B_{0,3}, \ldots, B_{6,3}$.

The figure below presents this example of a third degree B-spline with three interior knots along with its first derivative (the spline derivatives would be required in order to compute derivatives from the spline regression model).



To summarize, in this illustration we have an order $m = 4$ (degree = 3) B-spline (left) with 4 sub-intervals (segments) using uniform knots ($N = 3$ interior knots, 5 knots in total (2 endpoint knots)) and its 1st-order derivative (right). The dimension of $B(x)$ is $K = N + m = 7$.

See the appendix for R code (R Development Core Team 2011) that implements the B-spline basis function.

3.3. **The B-spline function.** A B-spline of degree n (of spline order $m = n + 1$) is a parametric curve composed of a linear combination of basis B-splines $B_{i,n}(x)$ of degree n given by

$$B(x) = \sum_{i=0}^{N+n} \beta_i B_{i,n}(x), \quad x \in [t_0, t_{N+1}]. \tag{2}$$

The β_i are called 'control points' or 'de Boor points'. For an order m B-spline having N interior knots there are $K = N + m = N + n + 1$ control points (one when $j = 0$).

The B-spline order m must be at least 2 (hence at least linear, i.e. degree n is at least 1) and the number of interior knots must be non-negative ($N \geq 0$).

See the appendix for R code (R Development Core Team 2011) that implements the B-spline function.

4. Multivariate B-spline regression

The functional form of parametric regression models must naturally be specified by the user. Typically practitioners rely on raw polynomials and also often choose the form of the regression function (i.e. the order of the polynomial for each predictor) in an ad-hoc manner. However, raw polynomials are not sufficiently flexible for our purposes, particularly because they possess no interior knots which is where B-splines derive their unique properties. Furthermore, in a regression setting we typically encounter multiple predictors which can be continuous or categorical in nature, and traditional splines are for continuous predictors. Below we briefly describe a multivariate kernel weighted tensor product B-spline regression method (kernel weighting is used to handle the presence of the categorical predictors). This method is implemented in the R package 'crs' (Racine & Nie 2011).

4.1. **Multivariate knots, intervals, and spline bases.** In general we will have q predictors, $\mathbf{X} = (X_1, \ldots, X_q)^T$. We assume that each X_l, $1 \leq l \leq q$, is distributed on a compact interval $[a_l, b_l]$, and without loss of generality, we take all intervals $[a_l, b_l] = [0, 1]$. Let $G_l = G_l^{(m_l-2)}$ be the space of polynomial splines of order m_l. We note that G_l consists of functions ϖ satisfying (i) ϖ is a polynomial of degree $m_l - 1$ on each of the sub-intervals $I_{j_l,l}, j_l = 0, \ldots, N_l$; (ii) for $m_l \geq 2$, ϖ is $m_l - 2$ times continuously differentiable on $[0, 1]$.

Pre-select an integer $N_l = N_{n,l}$. Divide $[a_l, b_l] = [0, 1]$ into $(N_l + 1)$ sub-intervals $I_{j_l,l} = [t_{j_l,l}, t_{j_l+1,l})$, $j_l = 0, \ldots, N_l - 1$, $I_{N_l,l} = [t_{N_l,l}, 1]$, where $\{t_{j_l,l}\}_{j_l=1}^{N_l}$ is a sequence of equally-spaced points, called interior knots, given as

$$t_{-(m_l-1),l} = \cdots = t_{0,l} = 0 < t_{1,l} < \cdots < t_{N_l,l} < 1 = t_{N_l+1,l} = \cdots = t_{N_l+m_l,l},$$

in which $t_{j_l,l} = j_l h_l$, $j_l = 0, 1 \ldots, N_l + 1$, $h_l = 1/(N_l + 1)$ is the distance between neighboring knots.

Let $K_l = K_{n,l} = N_l + m_l$, where N_l is the number of interior knots and m_l is the spline order, and let $B_l(x_l) = \{B_{j_l,l}(x_l) : 1 - m_l \leq j_l \leq N_l\}^{\mathrm{T}}$ be a basis system of the space G_l.

We define the space of tensor-product polynomial splines by $\mathcal{G} = \otimes_{l=1}^{q} G_l$. It is clear that $\mathcal{G}$ is a linear space of dimension $\mathbf{K}_n = \prod_{l=1}^{q} K_l$. Then[3]

$$\mathcal{B}(\mathbf{x}) = \left[\left\{\mathcal{B}_{j_1,\ldots,j_q}(\mathbf{x})\right\}_{j_1=1-m_1,\ldots,j_q=1-m_q}^{N_1,\ldots,N_q}\right]_{\mathbf{K}_n\times 1} = B_1(x_1)\otimes\cdots\otimes B_q(x_q)$$

is a basis system of the space $\mathcal{G}$, where $\mathbf{x} = (x_l)_{l=1}^{q}$. Let $\mathbf{B} = \left[\left\{\mathcal{B}(\mathbf{X}_1),\ldots,\mathcal{B}(\mathbf{X}_n)\right\}^{\mathrm{T}}\right]_{n\times\mathbf{K}_n}$.

4.2. **Spline regression.** In what follows we presume that the reader is interested in the unknown conditional mean in the following location-scale model,

$$Y = g(\mathbf{X},\mathbf{Z}) + \sigma(\mathbf{X},\mathbf{Z})\,\varepsilon, \tag{3}$$

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. Let $\left(Y_i,\mathbf{X}_i^{\mathrm{T}},\mathbf{Z}_i^{\mathrm{T}}\right)_{i=1}^{n}$ be an i.i.d copy of $\left(Y,\mathbf{X}^{\mathrm{T}},\mathbf{Z}^{\mathrm{T}}\right)$. Assume for $1\le l\le q$, each X_l is distributed on a compact interval $[a_l,b_l]$, and without loss of generality, we take all intervals $[a_l,b_l] = [0,1]$.

In order to handle the presence of categorical predictors, we define

$$l(Z_s,z_s,\lambda_s) = \begin{cases} 1,\text{when } Z_s = z_s \\ \lambda_s,\text{ otherwise.}\end{cases},$$

$$L(\mathbf{Z},\mathbf{z},\lambda) = \prod_{s=1}^{r} l(Z_s,z_s,\lambda_s) = \prod_{s=1}^{r}\lambda_s^{1(Z_s\neq z_s)}, \tag{4}$$

where $l(\cdot)$ is a variant of Aitchison and Aitken's univariate categorical kernel function (Aitchison & Aitken 1976), $L(\cdot)$ is a product categorical kernel function, and $\lambda = (\lambda_1,\lambda_2,\ldots,\lambda_r)^{\mathrm{T}}$ is the vector of bandwidths for each of the categorical predictors. See Ma et al. (under revision) and Ma & Racine (2013) for further details.

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

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

Let $\mathcal{L}_z = \mathrm{diag}\{L(\mathbf{Z}_1,\mathbf{z},\lambda),\ldots,L(\mathbf{Z}_n,\mathbf{z},\lambda)\}$ be a diagonal matrix with $L(\mathbf{Z}_i,\mathbf{z},\lambda)$, $1\le i\le n$ as the diagonal entries. Then $\widehat{\beta}(\mathbf{z})$ can be written as

$$\widehat{\beta}(\mathbf{z}) = \left(n^{-1}\mathbf{B}^{\mathrm{T}}\mathcal{L}_z\mathbf{B}\right)^{-1}\left(n^{-1}\mathbf{B}^{\mathrm{T}}\mathcal{L}_z\mathbf{Y}\right), \tag{5}$$

[3]The notation here may throw off those used to sums of the form $\sum_{i=1}^{n}$, $n>0$ (i.e. sum indices that are positive integers), so consider a simple illustration that may defuse this issue. Suppose there are no interior knots ($N=0$) and we consider a quadratic (degree n equal to two hence the 'spline order' is three). Then $\sum_{i=1-m}^{N}$ contains three terms having indices $i=-2,-1,0$. In general the number of terms is the number the number of interior knots N plus the spline order m, which we denote $K = N+m$. We could alternatively sum from 1 to $N+m$, or from 0 to $N+m-1$ of from 0 to $N+n$ (the latter being consistent with the Bézier curve definition in (1) and the B-spline definition in (2)).

where $\mathbf{Y} = (Y_1, \ldots, Y_n)^{\mathrm{T}}$. $g(\mathbf{x}, \mathbf{z})$ is estimated by $\widehat{g}(\mathbf{x}, \mathbf{z}) = \mathcal{B}(\mathbf{x})^{\mathrm{T}} \widehat{\beta}(\mathbf{z})$.

See the appendix for R code (R Development Core Team 2011) that implements the B-spline basis function and then uses least squares to construct the regression model for a simulated data generating process.

References

Aitchison, J. & Aitken, C. G. G. (1976), 'Multivariate binary discrimination by the kernel method', *Biometrika* **63**(3), 413–420.

Bernstein, S. (1912), 'Démonstration du théorème de Weierstrass fonde sur le calcul des probabilities', *Comm. Soc. Math. Kharkov* **13**, 1–2.

de Boor, C. (2001), *A practical guide to splines*, Springer.

Ma, S. & Racine, J. S. (2013), 'Additive regression splines with irrelevant categorical and continuous regressors', *Statistica Sinica* **23**, 515–541.

Ma, S., Racine, J. S. & Yang, L. (under revision), 'Spline regression in the presence of categorical predictors', *Journal of Applied Econometrics* .

R Development Core Team (2011), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: *https://www.R-project.org/*

Racine, J. S. & Nie, Z. (2011), *crs: Categorical Regression Splines*. R package version 0.14-9.

Wahba, G. (1990), *Spline Models for Observational Data*, SIAM.

Zhou, S. & Wolfe, D. A. (2000), 'On derivative estimation in spline regression', *Statistica Sinica* **10**, 93–108.

Appendix A. Sample R code for constructing B-splines

The following code uses recursion to compute the B-spline basis and B-spline function. Then a simple illustration demonstrates how one could immediately compute a least-squares fit using the B-spline. In the spirit of recursion, it has been said that "To iterate is human; to recurse divine." (L. Peter Deutsch).

R Code for Implementing B-spline basis functions and the B-spline itself.

```
## $Id: spline_primer.Rnw,v 1.30 2015/01/20 10:53:40 jracine Exp jracine $

## April 23 2011. The code below is based upon an illustration that
## can be found in http://www.stat.tamu.edu/~sinha/research/note1.pdf
## by Dr. Samiran Sinha (Department of Statistics, Texas A&M). I am
## solely to blame for any errors and can be contacted at
## racinej@mcmaster.ca (Jeffrey S. Racine).

## This function is a (simplified) R implementation of the bs()
## function in the splines library and illustrates how the Cox-de Boor
## recursion formula is used to construct B-splines.

basis <- function(x, degree, i, knots) {
  if(degree == 0){
    B <- ifelse((x >= knots[i]) & (x < knots[i+1]), 1, 0)
  } else {
    if((knots[degree+i] - knots[i]) == 0) {
      alpha1 <- 0
    } else {
      alpha1 <- (x - knots[i])/(knots[degree+i] - knots[i])
    }
    if((knots[i+degree+1] - knots[i+1]) == 0) {
      alpha2 <- 0
    } else {
      alpha2 <- (knots[i+degree+1] - x)/(knots[i+degree+1] - knots[i+1])
    }
    B <- alpha1*basis(x, (degree-1), i, knots) + alpha2*basis(x, (degree-1), (i+1), knots)
  }
  return(B)
}

bs <- function(x, degree=3, interior.knots=NULL, intercept=FALSE, Boundary.knots = c(0,1)) {
  if(missing(x)) stop("You must provide x")
  if(degree < 1) stop("The spline degree must be at least 1")
  Boundary.knots <- sort(Boundary.knots)
  interior.knots.sorted <- NULL
  if(!is.null(interior.knots)) interior.knots.sorted <- sort(interior.knots)
  knots <- c(rep(Boundary.knots[1], (degree+1)), interior.knots.sorted, rep(Boundary.knots[2], (degree+1)))
  K <- length(interior.knots) + degree + 1
  B.mat <- matrix(0,length(x),K)
  for(j in 1:K) B.mat[,j] <- basis(x, degree, j, knots)
  if(any(x == Boundary.knots[2])) B.mat[x == Boundary.knots[2], K] <- 1
  if(intercept == FALSE) {
    return(B.mat[,-1])
  } else {
    return(B.mat)
  }
}

## A simple illustration that computes and plots the B-spline bases.
```

```
par(mfrow = c(2,1))

n <- 1000
x <- seq(0, 1, length=n)
B <- bs(x, degree=5, intercept = TRUE, Boundary.knots=c(0, 1))
matplot(x, B, type="l", lwd=2)

## Next, simulate data then construct a regression spline with a
## prespecified degree (in applied settings we would want to choose
## the degree/knot vector using a sound statistical approach).

dgp <- sin(2*pi*x)
y <- dgp + rnorm(n, sd=.1)

model <- lm(y~B-1)

plot(x, y, cex=.25, col="grey")
lines(x, fitted(model), lwd=2)
lines(x, dgp, col="red", lty=2)
```