COBS: Qualitatively Constrained Smoothing via Linear Programming

Xuming He[1]
Department of Statistics, University of Illinois, Champaign, IL 61820
and
Pin Ng
Department of Economics, University of Houston, Houston, TX 77204-5882

Summary

Popular smoothing techniques generally have a difficult time accommodating qualitative constraints like monotonicity, convexity or boundary conditions on the fitted function. In this paper we bring the problem of constrained spline smoothing to the foreground and describe the details of a constrained B-spline smoothing (COBS) algorithm that is being made available to Splus users. Recent work of He and Shi (1996) considered a special case and showed that the L_1 projection of a smooth function into the space of B-splines provides a monotone smoother that is flexible, efficient and achieves the optimal rate of convergence. Several options and generalizations are included in COBS: it can handle small or large data sets either with user interaction or full automation. Three examples are provided to show how COBS works in a variety of real-world applications.

Keywords: Constraint; Information criterion; Knot selection; Linear program; Nonparametric regression; Regression quantile; Smoothing Spline.

1 Introduction

A huge amount of research has been carried out in the past few decades on nonparametric function estimation based on the idea of smoothing. A number of highly successful smoothing methods are available in Splus. Among them are smoothing splines (Hastie and Tibshirani, 1990), kernel smoothing (Watson, 1966), local-span supersmoother (Friedman, 1984), and robust smoothing via Lowess (Cleveland, 1979). Several other smoothers are also available from *statlib*.

Data smoothing is often viewed and used as a graphical method to uncover the underlying relationship between two variables. In some applications, the functions being estimated are known to satisfy certain qualitative properties such as monotonicity. For example in variable transformations, it is often desirable to restrict oneself to monotone functions. Further applications can be found in growth charts constructions, brain image registrations, and probability curve estimations. In other applications, concavity or convexity constraints are desirable. Examples include cost functions and efficient production frontiers in economics, where the estimated functions are expected to be convex and concave respectively. If the response variable is a proportion, one naturally wants the fitted curves to fall between 0 and 1. For cyclical time series, one might want the fitted curve at the first period to wrap around and equal that at the last period of a cycle. In some cases, the function values or its derivatives at some specific points are known and need to be satisfied by the fitted curve.

Any smoother that performs local averaging over the response values will yield a fitted function falling within the range of the response values. In some cases, this is highly desirable. For example, if the response variable is age or income, zero is the intrinsic lower bound of the estimated function. Conventional spline methods, however, may not preserve this positivity near the boundaries. The constrained B-spline smoothing method we introduce in this paper can easily impose positivity and help overcome this weakness of spline smoothers.

It is not difficult to argue that a constrained smoother that incorporates prior information is also more efficient. Delecroix, Simioni and Thomas-Agnan (1995) report a simulation study on this. In the case of monotone smoothing, several methods have been proposed in the literature; see Härdle (1989, Chapter 8) for a nice introduction, and Hawkins (1994) and Ramsay (1995) for more recent results. Wright and Wegman (1980) contain a more general treatment using splines that includes monotonicity and convexity constraints in the least squares regression setting. Nevertheless, few constrained smoothing algorithms are publically available as we write due to the difficulty of incorporating restrictions like those mentioned above.

[1]Xuming He's research is supported in part by National Security Agency Grant MDA904-96-1-0011.

COBS (COnstrained B-Splines) is a very attractive constrained smoothing method with some unique advantages. Using the earlier work of Ramsay (1988) and Koenker, Ng and Portnoy (1994), He and Shi (1996) considered a special case of monotone smoothing and laid down the foundation for COBS. The present paper focuses on the algorithmic aspect of COBS and contains a wide variety of options for flexible applications.

We begin with a general framework of L_1 minimization for function estimation. It includes two classes of spline smoothers: smoothing splines (with a roughness penalty) and regression splines (without roughness penalty). Two options are provided in COBS to determine the smoothing parameter of the smoothing splines or the knot formation of the regression splines. The first option allows user interaction, which lets users formulate their own choice of smoothing parameter or knot mesh. Since it takes very little time for COBS to return the fitted curve for each set of chosen parameters, visual comparison and judgment can be performed interactively. The second option provides full automation. Users are not required to supply exact smoothing parameter; COBS makes adaptive choices using information criteria similar to those in model selection.

The L_1 framework leads to linear programming (LP) formulations of the problems and allows efficient computation via linear programming techniques. The LP form makes it possible to naturally incorporate all the constraints discussed above. As far as we know, COBS is the only smoothing algorithm that can do this without substantial increase in computational costs.

COBS facilitates more than just robust function estimation via conditional median estimation of the response given the covariate. It also provides computation of other conditional quantile functions which have gradually become an integral part of data analysis.

In Section 2, we introduce the linear and quadratic splines and describe how the L_1 minimization problem can be solved as a linear program. Section 3 discusses how COBS chooses the smoothing parameter or the knot mesh if full automation is desired. Section 4 describes some additional features of COBS and finally three illustrative examples are provided in Section 5 with detailed Splus source code given in the appendices. The underlying Fortran program and Splus interface for COBS is available from http://www.econ.uiuc.edu/~ng or http://ux6.cso.uiuc.edu/~x-he/ftp.html.

2 Constrained Smoothing

For a pair of bivariate random variables (X, Y), the τth conditional quantile function, $g_\tau(x)$, of Y given $X = x$ is a function of x such that

$$P\left(Y \leq g_\tau(x) \mid X = x\right) = \tau.$$

The conditional median function ($\tau = .5$) provides a measure of conditional central tendency and can be used to describe the causal relationship between

X and Y. When a more complete picture of the relationship is needed, the whole spectrum of conditional quantile functions can be examined.

Given n pairs of realizations $\{(x_i, y_i)\}_{i=1}^n$ with $a = x_0 < x_1 < \cdots < x_n < x_{n+1} = b$, some *smooth* function g and the *check function* $\rho_\tau(u) = [\tau - I(u < 0)]\, u = \left[\frac{1}{2} + \left(\tau - \frac{1}{2}\right) sgn\,(u)\right] |u|$, we define "fidelity" to the data as

$$\text{“fidelity”} = \sum_{i=1}^{n} \rho_\tau(y_i - g(x_i)).$$

Koenker, Ng and Portnoy (1994) introduced the τth L_p *quantile smoothing spline,* $\hat{g}_{\tau,L_p}(x)$, which is the solution to

$$\min_g \quad \text{“fidelity”} + \lambda \text{ “}L_p\text{ roughness”} \tag{1}$$

as a nonparametric estimator for $g_\tau(x)$. The usual smoothing parameter λ controls the trade-off between fidelity to the data and roughness of the fit. Two versions of roughness measure,

$$\text{“}L_1\text{ roughness”} = V(g') = \sum_{i=1}^{n-2} \left| g'(x_{i+1}) - g'(x_i) \right| \tag{2}$$

and

$$\text{“}L_\infty\text{ roughness”} = \| g'' \|_\infty = \max_x g''(x) \tag{3}$$

were suggested, where $V(.)$ denotes the total variation norm. They show that $\hat{g}_{\tau,L_1}(x)$ is a linear (second order) natural smoothing spline for the L_1 roughness penalty while $\hat{g}_{\tau,L_\infty}(x)$ can be approximated by a quadratic (third order) natural smoothing spline for the L_∞ roughness penalty.

In this section, we will concentrate on the special case of $\tau = .5$ so "fidelity" is measured by the L_1 norm,

$$\text{“fidelity”} = \sum_{i=1}^{n} |y_i - g(x_i)|.$$

The resulting linear and quadratic natural smoothing splines are two alternative estimators for the conditional median function.

It is well known that any mth order natural smoothing spline with simple knots at $x_1, \cdots, x_n$ has an equivalent B-spline representation on the same knot sequence. We, however, start with a more general knot mesh $T = \{t_i\}_{i=1}^{N+2m}$ with $t_1 = \cdots = t_m < t_{m+1} < \cdots < t_{N+m} < t_{N+m+1} = \cdots = t_{N+2m}$. The motivation for this generalization, pertaining to computational efficiency considerations, will become apparent later. The B-spline representation, $s \in S_{m,T}$, of a smooth function becomes

$$s(x) = \sum_{j=1}^{N+m} a_j B_j(x)$$

where N is the number of internal knots, $B_j(x)$ are the normalized B-spline basis functions, a_j are the coefficients for the B-spline basis functions and $S_{m,T}$ is the space of polynomial splines of order m with mesh T. See Schumaker (1981) or DeBoor (1978) for more details.

2.1 Smoothing B-Splines

For the sake of expositional convenience, we assume in this subsection that x_i are all distinct from each other. We use linear ($m = 2$) B-splines with $N = n-2$ internal knots in the mesh $T = \{t_i\}_{i=1}^{N+2m}$ such that $t_1 = t_m = x_1, t_{m+1} = x_2, \cdots, t_{N+m} = x_{n-1}, t_{N+m+1} = t_{N+2m} = x_n$ for the optimization problem of the linear smoothing spline in (1) and (2). Now the objective function can be written as

$$\begin{aligned}\min_{\theta\in R^{N+m}} \quad & \sum_{i=1}^{n}\left|y_i - \sum_{j=1}^{N+m} a_j B_j(x_i)\right| \\ & +\lambda\sum_{i=1}^{N}\left|\sum_{j=1}^{N+m} a_j B_j'(t_{i+m}) - \sum_{j=1}^{N+m} a_j B_j'(t_{i+m-1})\right|\end{aligned}$$

where $\theta = (a_1, \cdots, a_{N+m})$.

We can express the above in a more compact form as

$$\min_{\theta\in R^{N+m}} \sum_{i=1}^{n+N} |\widetilde{y}_i - \widetilde{x}_i\theta| \tag{4}$$

where

$$\tilde{y} = \begin{pmatrix} y \\ \mathbf{0} \end{pmatrix}$$

is an $(n+N)\times 1$ *pseudo response* vector,

$$\tilde{X} = \begin{bmatrix} \mathbf{B} \\ \lambda\mathbf{C} \end{bmatrix}$$

is an $(n+N)\times(N+m)$ *pseudo design* matrix with

$$\mathbf{B} = \begin{bmatrix} B_1(x_1) & \cdots & B_{N+m}(x_1) \\ \vdots & \cdots & \vdots \\ B_1(x_n) & \cdots & B_{N+m}(x_n) \end{bmatrix}$$

and

$$\mathbf{C} = \begin{bmatrix} B_1'(t_{m+1}) - B_1'(t_m) & \cdots & B_{N+m}'(t_{m+1}) - B_{N+m}'(t_m) \\ \vdots & \cdots & \vdots \\ B_1'(t_{N+m}) - B_1'(t_{N+m-1}) & \cdots & B_{N+m}'(t_{N+m}) - B_{N+m}'(t_{N+m-1}) \end{bmatrix}$$

The fitted curve, $\hat{m}_{\lambda,L_1}(x) = \sum_{j=1}^{N+m} \hat{a}_j B_j(x)$, is a *linear (median) smoothing B-spline.*

The objective function (4) can be solved by any efficient linear programming algorithm. To see this, write it into the following LP:

$$\min \left\{ \mathbf{1}'(u+v) \,|\, \tilde{y} - \tilde{X}\theta = u - v, \left(u', v'\right) \in R_+^{2(n+N)} \right\} \tag{5}$$

A modification of Bartels and Conn's (1980) non-simplex active-set algorithm for the quantile smoothing splines described in Ng (1996), and Koenker and Ng (1996) can be easily adapted for (5).

Similarly, using quadratic ($m = 3$) B-splines with $N = n-2$ internal knots in the mesh $T = \{t_i\}_{i=1}^{N+2m}$ such that $t_1 = t_2 = t_m = x_1, t_4 = x_2, \cdots, t_{N+m} = x_{n-1}, t_{N+m+1} = t_{N+2m-1} = t_{N+2m} = x_n$, we rewrite (1) and (3) as

$$\min_{\theta \in R^{N+m}} \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{N+m} a_j B_j(x_i) \right| + \lambda \max_x \sum_{j=1}^{N+m} a_j B_j''(x)$$

where $\theta' = (a_1, \cdots, a_{N+m})$. This is equivalent to

$$\min_{\theta \in R^{N+m+1}} \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{N+m} a_j B_j(x_i) \right| + \lambda\sigma$$

$$s.t. \qquad -\sigma \le \sum_{j=1}^{N+m} a_j B_j''(t_{i+m-1}) \le \sigma \text{ for } i = 1, \cdots, N+1$$

where $\theta' = (a_1, \cdots, a_{N+m}, \sigma)$. In a more compact form, we have

$$\min_{\theta \in R^{N+m+1}} \sum_{i=1}^{n+1} |\widetilde{y}_i - \widetilde{x}_i\theta| \tag{6}$$

$$s.t. \qquad \widetilde{D}\theta = \begin{bmatrix} \mathbf{D} & \mathbf{1} \\ -\mathbf{D} & \mathbf{1} \end{bmatrix} \theta \ge \mathbf{0}$$

where

$$\tilde{y} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

is an $(n+1)$ pseudo response vector,

$$\tilde{X} = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \lambda \end{bmatrix}$$

is an $(n+1) \times (N+m+1)$ pseudo design matrix and

$$\mathbf{D} = \begin{bmatrix} B_1''(t_m) & \cdots & B_{N+m}''(t_m) \\ \vdots & \cdots & \vdots \\ B_1''(t_{N+m}) & \cdots & B_{N+m}''(t_{N+m}) \end{bmatrix}.$$

The resulting fitted curve, $\hat{m}_{\lambda,L_\infty}(x) = \sum_{j=1}^{N+m} \hat{a}_j B_j(x)$, is a *quadratic (median) smoothing B-spline.*

The LP equivalence of (6) is

$$\min\left\{\mathbf{1}'(u+v)\,|\tilde{y}-\tilde{X}\theta = u-v, \widetilde{D}\theta \geq \mathbf{0}, \left(u^{'}, v^{'}\right) \in R_{+}^{2(n+N)}\right\}. \qquad (7)$$

The pseudo design matrices in (5) and (7) are both of the order $O\left(n^2\right)$. This will inflict huge burden on computational speed and memory space for large data sets. We can alleviate the problem substantially by approximating the natural smoothing splines using a much smaller number of internal knots N and hence reduce the order of the pseudo design matrices to $O(nN)$. For example, we can use $T = \{t_i\}_{i=1}^{N+2m}$ with t_i chosen to be the N ($\ll n$) sample quantiles of the covariate x or equally spaced values in the domain of x.

2.2 Imposing Additional Constraints

Due to the LP nature of the objective functions (5) and (7), many qualitative restrictions on the fitted curves can be incorporated easily by the addition of some equality or inequality constraints as described below.

Monotonicity Constraints

For the linear spline $\hat{m}_{L_1}(x)$, the additional set of constraints needed is

$$s.t. \qquad \mathbf{E}\theta \geq \mathbf{0}$$

for increasing functions and

$$s.t. \qquad \mathbf{E}\theta \leq \mathbf{0}$$

for decreasing functions where

$$\mathbf{E} = \begin{bmatrix} B_1'(t_m) & \cdots & B_{N+m}'(t_m) \\ \vdots & \cdots & \vdots \\ B_1'(t_{N+m+1}) & \cdots & B_{N+m}'(t_{N+m+1}) \end{bmatrix}.$$

For the quadratic spline $\hat{m}_{\tau,L_\infty}(x)$, the extra set of $N+2$ constraints is

$$s.t. \qquad \begin{bmatrix} \mathbf{E} & \mathbf{1} \end{bmatrix} \theta \geq \mathbf{0}$$

for increasing functions and

$$s.t. \qquad \begin{bmatrix} \mathbf{E} & \mathbf{1} \end{bmatrix} \theta \leq \mathbf{0}$$

for decreasing ones.

Convexity Constraints

For $\hat{m}_{L_1}(x)$, we need the N constraints

$$s.t. \qquad \mathbf{C}\theta \geq \mathbf{0}.$$

For $\hat{m}_{L_\infty}(x)$, the additional set of $N+1$ constraints is

$$s.t. \qquad \begin{bmatrix} \mathbf{D} & \mathbf{0} \end{bmatrix} \theta \geq \mathbf{0}.$$

Concavity restriction can similarly be imposed with all the inequalities reversed.

Periodic Constraints

A restriction of the form $g(x_1) = g(x_n)$ is useful for cyclical time series where x_1 and x_n are the first and last unique observed values in the time domain of a cycle, e.g. the first $(x_1 = 1)$ and last $(x_n = 12)$ months of a year in monthly data. This can be achieved easily with the addition of the single equality constraint

$$s.t. \qquad \left[\tilde{X}_{(1)} - \tilde{X}_{(n)}\right] \theta = 0$$

where $\tilde{X}_{(1)}$ and $\tilde{X}_{(n)}$ are the first and nth row of the pseudo design matrix $\tilde{X}$.

Pointwise Constraints

Pointwise constraints on the function and/or its derivatives can be directly imposed on the coefficients of the spline as illustrated in Section 4.3.

2.3 Regression B-splines

The computational burden can be ameliorated in a different way by dropping the penalty term totally; i.e. setting $\lambda = 0$ in (1). This gives rise to the *(median) regression B-splines* of He and Shi (1994). Fidelity in regression B-splines is still measured the same way as in smoothing splines but roughness is controlled by the number of internal knots N rather than λ. The quantity $(N+m)$ plays the role of *effective dimensionality* of the fit. The two extreme fits correspond to $N = 0$, which yields the globally linear and quadratic B-spline fits for $m = 2$ and $m = 3$ respectively, while $N = n-2$ with $t_{i+m-1} = x_i$ for $i = 1, \cdots, n$, gives the interpolating fit.

The *linear (median) regression B-spline*, $\hat{m}_{T,L_1}$, will solve

$$\begin{array}{ll} \min_{\theta \in R^{N+m}} & \sum (u+v) \\ s.t. & y - \tilde{X}\theta = u - v \\ & u \in R^n_+, \ v \in R^n_+ \end{array}$$

where

$$\tilde{X} = \mathbf{B}$$

is now an $n \times (N+m)$ pseudo design matrix.

On the other hand, the *quadratic (median) regression B-spline*, $\hat{m}_{T,L_\infty}$, is the solution to

$$\begin{array}{ll} \min_{\theta \in R^{N+m}} & \sum (u+v) \\ s.t. & y - \tilde{X}\theta = u - v \\ & u \in R^n_+, \; v \in R^n_+ \end{array}$$

where

$$\tilde{X} = \mathbf{B}$$

is again the $n \times (N+m)$ pseudo design matrix.

3 Choice of Smoothing Parameter or Knots

If a fully automated smoother is desired, we must resolve the issue of choosing either the smoothing parameter λ for the smoothing splines or the knot mesh $T = \{t_i\}_{i=1}^{N+2m}$ in the case of regression B-splines where $\lambda = 0$.

The calling sequence of COBS is given in Appendix A. When the argument `lambda` is supplied with a negative value, COBS will compute the smoothing spline with λ chosen to minimize a Schwarz-type information criterion (SIC) used in Koenker, Ng and Portnoy (1994), and He, Ng and Portnoy (1996). Denote $\hat{m}_{\lambda,L_1}$ or $\hat{m}_{\lambda,L_\infty}$ simply as $\hat{m}_\lambda$, SIC is defined as

$$SIC(\lambda) = \log(\sum_i |y_i - \hat{m}_\lambda(x_i)|) + \frac{1}{2} p_\lambda \log(n)/n$$

where p_λ is the number of interpolated data points and serves as complexity measure of the fitted model.

From the LP nature of the objective functions, we know there are only finitely many distinct $\hat{m}_\lambda$ as λ varies in $(0,\infty)$. Parametric linear programming (PLP) as suggested in Ng (1996) is used to obtain all the possible distinct $\hat{m}_\lambda$ when λ decreases to zero from a value specified in the argument `lstart`.

The total number of distinct λ values is expected to grow with the sample size n at the rate of $n\log(n)$; see Portnoy (1994). As a result, PLP will rapidly become the computation bottle-neck for moderately large n. From our experience, $\hat{m}_\lambda$ is not very sensitive to small perturbation in the λ values. In typical cases, there exist several very similar solutions corresponding to neighboring values of λ obtained from PLP. To speed things up, COBS allows users to skip some neighboring λ values via the argument `factor` by specifying 1 <`factor`< 4 while performing PLP. A bigger `factor` allows more neighboring λ's to be skipped. This is equivalent to using a coarser grid in the search for the optimal λ.

It is important to note that the first term of SIC becomes infinitely small if $\hat{m}_\lambda$ interpolates every single data point. As a result, the λ that minimizes SIC could be too small. Since SIC is meant to be used for model comparison with dimensionality not too close to n, COBS prints a warning message recommending the user to examine the plot of SIC against λ when the chosen λ is near zero and see if the second minimizer of SIC will provide a more reasonable fit. COBS returns the necessary plotting information in the components `$pp.lambda` and `$sic`.

When `lambda` is provided with a positive number, it will be used as the value of the smoothing parameter. No SIC will be used to choose the optimal λ. This option allows the user to experiment with various fits of different smoothness.

The argument `knots` allows users to specify the location of the knots while `nknots` is used to control the number of knots. If `knots` is missing, a default set of `nknots` knots will be generated by one of the two methods specified by the `method` argument. The default `method` is '`quantile`', which uses `nknots` design points uniform in their percentile levels as the knot sequence. For example, if `nknots = 3`, the median of the covariate will be the single internal knot. The quartiles will be used if `nknots` = 5. If `method` = '`uniform`', uniformly spaced points between the smallest and largest design values will be used as the knot sequence. The difference in the resulting fits between the two methods is usually negligible unless there are sizable intervals with few or no data points. If `nknots` is missing, a default value of 20 will be assigned.

When `lambda` is set to zero, COBS computes the regression B-spline estimate. If both `knots` and `nknots` are provided and `nknots` equals the length of `knots`, COBS uses the supplied knot sequence without performing the knot selection procedure **Step 1 – 3** below. This allows users to interactively experiment with various fits for their specified sets of knots. Otherwise, the knot selection procedure below will be performed. If `knots` is missing, COBS will generate a default set of `nknots` (default to 6 if `nknots` is missing as well) knots by one of two methods specified by `method`. If `knots` is not missing, COBS will use this to begin the following knot selection procedure:

Step 1: Choose the initial optimal number of internal knots N. Compute the regression B-splines for $N = 0, \cdots,$ (`nknots` -2) internal knots from `knots`. Denote $\hat{m}_{T,L_1}$ or $\hat{m}_{T,L_\infty}$ simply as $\hat{m}_T$, select the N that corresponds to the smallest

$$AIC(T) = \log(\sum_i |y_i - \hat{m}_T(x_i)|) + 2(N+m)/n$$

where T is the knots mesh, N is the number of internal knots in T, and m is the order of spline used.

Step 2: Perform stepwise knot deletion. Each internal knot is deleted sequentially to obtain a sequence of AIC values. The one whose deletion leads to the largest reduction in AIC is then slated for actual deletion. This process is repeated until no more existing knot can be removed.

Step 3: Perform stepwise knot addition. When the argument `knots.add` is set to `TRUE` , COBS takes the mid-point between every adjacent pair of existing knots as potential new candidate. If inclusion of any such point reduces the value of AIC, we choose to add the one which gives the largest reduction in AIC. This process repeats until no more knot needs to be added.

We should note that in the current implementation, we do not cycle through **Step 2** and **3** repeatedly. That is, we do not go back to the knot deletion process after knot addition.

We use the constant $2/n$ in the second term of AIC as it appears to give the best overall results in our experiments with monotone and concave/convex functions. Should AIC undersmooth the data, users also have the option of using the SIC by substituting $\log(n)/n$ for $2/n$. This is done in COBS via assigning `ic='sic'` instead of the default setting of `ic='aic'`.

In constrained smoothing, the number of knots needed is typically small. The initial number of knots chosen by AIC in **Step 1** is often less than 4 when n is not too large. However, if the chosen number is $N = 4$, we may want to look into larger N values. COBS will remind the user of this through warning messages at the end of the computation. The user can then re-run the program with a larger value of `nknots` if `knots` is missing or supply COBS with a longer `knots` sequence.

4 Other Features of COBS

In this section, we discuss additional issues related to the design and use of COBS.

4.1 Large Sample Problems

The amount of time needed to obtain an estimate may disappoint even the most patient users when a fully automated solution is chosen for a large data set. COBS seeks an approximate solution when $n > 1000$. This is achieved by using a subset of the original data during the selection of knots or the smoothing parameter λ. The sub-sample size is chosen to be $670 + \log(n)^3$ rounded to the nearest integer. The whole data set, however, will be used in the final model fitting stage. As noted in the previous section, the storage space needed in the regression B-spline (with `lambda`$= 0$) is of the order much smaller than the smoothing B-spline. We, therefore, recommend using the smoothing B-spline with a small number of internal knots N or the regression B-spline when the sample size is big and automation is required.

4.2 Roughness Penalty

COBS uses the argument `degree` to determine the type of roughness penalty for the smoothing B-splines. The L_1 roughness penalty is selected by setting `degree` $= 1$ and the L_∞ roughness penalty is chosen with `degree` $= 2$.

4.3 Pointwise Constraints

Four types of pointwise constraints can be imposed in COBS via the argument `pointwise`, whose value is a three-column matrix with each row representing one of the following conditions:

$$\begin{array}{lll} (0, x, y) & \text{for} & g(x) = y; \\ (1, x, y) & \text{for} & g(x) \geq y; \\ (-1, x, y) & \text{for} & g(x) \leq y; \\ (2, x, y) & \text{for} & g'(x) = y. \end{array}$$

Multiple constraints are allowed but a feasibility check will be performed to ensure that they do not contradict one another.

Pointwise constraints are particularly useful in imposing boundary conditions. For example, if the response variable is weight or salary, it is useful to impose the $(1, 0, 0)$ constraint which corresponds to $g(0) \geq 0$. For monotonically increasing function, this will imply $g(x) \geq 0$ for all $x > 0$. See Example 2 for an implementation of such boundary restriction.

The constraints we have included in COBS are certainly not meant to be exhaustive. Other types of pointwise restrictions can be added as need arises.

4.4 Binary Choice Model

In a binary choice model where there are only two unique values in the response variable, the L_1 method does not work directly. We use pre-smoothing in COBS. In the current version of COBS, we first use the Splus smoother `smooth.spline` and the fitted values will then be passed on to COBS as the response for further processing. If the fitted function is expected to be increasing, additional pointwise constraints of the form $(1, x_1, min(y_i))$ and $(-1, x_n, max(y_i))$ will be imposed automatically by COBS. The reverse is true for decreasing fit to ensure that the fitted function is between $min(y_i)$ and $max(y_i)$. Any kernel smoother may be used in place of `smooth.spline` in the pre-smoothing stage.

4.5 Other Quantiles

Conditional quantile functions beyond the median may provide a more complete picture of the relationship between the response and covariate. COBS provides estimates of the τth conditional quantile via the `tau` argument. We should point out that AIC and SIC criteria may not perform as well in choosing the smoothing parameters for extreme quantiles ($\tau \approx 0$ or 1) as for the median. User interaction is recommended in such situations.

4.6 Missing Values

If there are missing values in the data set, COBS will calculate the fit based only on the fully observed cases. No imputation is attempted.

4.7 Speeding Up

Recently, Portnoy and Koenker (1996) propose an improved interior point algorithm to solve an LP problem like ours. Their idea is to combine the recent advances in interior point method with a new statistical pre-processing approach so the algorithm can handle massive data sets at a speed comparable to that of least squares computation. Although the current version of COBS has not adopted this new method, it does appear possible that a substantial improvement in computational speed can be achieved for big data sets in the future.

5 Illustrative Examples Using COBS in Splus

To help readers familiarize themselves with the COBS approach, we provide three examples along with all the Splus code used to produce the results. In Examples 2 and 3, we find that it would be difficult to modify other smoothers to satisfy the necessary pointwise constraints.

Example 1: We consider the annual average global surface temperature (in degrees K) of Hansen and Lebedeff (1987, 1988). The data covers the period of 1880 to 1992. The Splus codes in Appendix B were used to calculate the 10th, 50th and 90th percentile curves using linear regression B-splines. The monotonicity constraint was used under the assumption of global warming; see Figure 1a. The automated AIC knot selection criterion picked $N = 4$ (the largest allowed by the default value of `nknots` $= 6$) internal knots for the 50th percentile curve. As discussed in Section 3, COBS printed a warning message so we increased `nknots` incrementally to 9. The final knots selected are located at (1880, 1908, 1936, 1964, 1978, 1992). These same knots were used for the 10th and 90th percentile curves. The extreme temperature years in the top and bottom 10% after adjusting for the overall trend of global warming (if so believed) can be readily identified in Figure 1a. The hotter years are 1889, 1897$*$, 1900, 1901, 1926, 1937$*$, 1938$*$, 1940, 1947$*$, 1953, 1980$*$, 1981 and 1990$*$ with those years followed by an '$*$' fall exactly on the 90th percentile curve. The colder years are 1884, 1904, 1907$*$, 1917, 1918, 1950$*$, 1956, 1964, 1965, 1976 and 1992$*$ again with those followed by an '$*$' lie exactly on the 10th percentile curve. After adjusting for the trend, 1987, 1988, and 1991 would not be considered hot as they all fell below the top 10th percentile curve. The unconstrained version of the curves are presented in Figure 1b. They show a cooling period from 1936 to 1964. These unconstrained percentile curves are quite similar to the linear quantile smoothing splines presented in Koenker and Schorfheide (1994, p.401, Fig 3) except their tenth percentile is somewhat oversmooth as compared to ours.

We have not attempted to correct for serial correlations in the data in all the fits. The percentiles curves provide an ordering of data adjusted for the overall trend. They also suggest that variability of global temperature is rather stable over the last century.

Example 2: The US Army Construction Engineers use flashing condition index (FCI) as one of several important roof condition measures. Roughly speaking, FCI shows what percentage of roof flashing is in good condition. We use records from 153 roof sections with EPDM bash flashing from a number of U.S. Army bases and wish to study how FCI decreases over time. The ages of the roof sections vary between several weeks to fifteen years. Due to skewness of the FCI distribution, it is especially helpful in this case to compute the percentile curves instead of the mean and variance functions. The three quartiles corresponding to $\tau = 0.25, 0.50$ and 0.75 are computed for this example.

In addition to the obvious constraint of monotonicity, the engineers suggested that the majority of the new roofs at age 0 should have FCI at 100. That is, we need a boundary constraint of $g(0) = 100$ for these quartiles.

We choose to use the quadratic smoothing B-splines in COBS. For each τ, we obtained a plot of SIC versus the values of λ. Examination of these plots proved to be useful. For example, at $\tau = 0.5$, the minimum values of SIC is attained for several smallest values of λ (see Figure 2a). As we pointed out in Section 3, we should look for a minimum after excluding those choices of λ. A good choice is $\lambda = 14.04$ in this case. The same exercise was carried out at the other τ values. We can see from Figure 2b that the top 25% of the EPDM roofs still remain in perfect condition after fifteen years. Even the lower quartile shows a very slow rate of degradation after the eighth year. This is probably why EPDM roofs are now preferred to others. The Splus codes for this example is also given in Appendix B.

Example 3: This example serves to illustrate the use of `'periodic'` constraint for cyclical data. The response variable is the daily average wind speed (in knots) recorded at the synoptic meteorological station in Dublin, Ireland. The data can be downloaded from *statlib* and is described in detail in Haslett and Raftery (1989). Here, we use the quadratic smoothing B-spline with thirteen knots which correspond roughly to the beginning of all the twelve months of a year. The data is plotted in Figure 3a. The Splus codes that generate the fits in Figure 3b are in Appendix B as well. For $\tau = .5$, the initial $\lambda = 7871.74$ chosen by SIC reached the largest possible value allowed by the default setting of `lstart`. As recommended in the warning message of COBS, we re-fit the model to allow the parametric linear programming in λ to begin from a larger λ value. The final fits for $\tau = .1, .5,$ and $.9$ use $\lambda = 74.55 * 10^2, 99.70 * 10^3,$ and $75.67 * 10^2$ respectively . Notice that for each of the quantiles, the fitted values at the beginning and the end of a year are the same. Again we have not attempted to correct for any possible correlation in the data.

References

Cleveland, W. (1979). "Robust Locally Weighted Regression and Smoothing Scatterplots", *Journal of the American Statistical Association*, 74, 829-836.

De Boor, C. (1978). *A Practical Guide to Splines*, in Applied Mathematical Sciences, Vol.27, Springer-Verlag, New York.

Delecroix, X., Simioni, M. and Thomas-Agnan, C. (1995). "A Shape Constrained Smoother: Simulation Study", *Computational Statistics*, 10, 155-175.

Friedman, J. (1984). "A Variable span Smoother", Tech. Rep. No. 5, Laboratory for Computational Statistics, Dept. of Statistics, Stanford University, California.

Hansen, J. and Lebedeff S. (1987). "Global Trends of Measured Surface Air Temperature", *J. of Geophys. Res.*, 92, 13345-13372.

Hansen, J. and Lebedeff S. (1988). "Global Surface Air Temperature: Update through 1987", *Geophys. Lett.*, 15, 323-326.

Härdle, W. (1990). *Applied Nonparametric Regression*, Cambridge University Press, New York.

Haslett, J. and Raftery A.E. (1989). "Space-time Modelling with Long-memory Dependence: Assessing Ireland's Wind Power Resource", *Applied Statistics*, 38, 1-21.

Hastie, T. and Tibshirani R. (1990). *Generalized additive models*, Chapman & Hall.

Hawkins, D. M. (1994). "Fitting Monotonic Polynomials to Data", *Computational Statistics*, 9, 233-237.

He, X. and Shi P. (1994). " Convergence rate of B-spline estimators of nonparametric conditional quantile functions", *Journal of Nonparametric Statistics*, 3, 299-308.

He, X. and Shi P. (1996). "Monotone B-spline Smoothing", Preprint.

Koenker, R. and Ng P. (1996). "A Remark on Bartels and Conn's Linearly Constrained L1 Algorithm", *ACM Transaction on Mathematical Software*, forthcoming.

Koenker, R., Ng P. and Portnoy S. (1994). "Quantile Smoothing Splines", *Biometrika*, 81, 673–680.

Koenker, R. and Schorfheide F. (1994). "Quantile Spline Models for Global Temperature Change", *Climate Change*, 28, 395-404.

Ng, P. (1996). An algorithm for quantile smoothing splines, *Computational Statistics and Data Analysis*, 22, 99-118.

Portnoy, S. and Koenker, R. (1996). "The Gaussian hare and the Laplacian tortoise: computability of squared-error vs. absolute-error estimators", preprint.

Ramsay, J. O. (1988). "Monotone Regression Splines in Action", *Statistical Science*, 3, 425-441.

Ramsay, J. O. (1995). "Estimating Smooth Monotone Functions", preprint.

Schumaker, L. (1981). *Spline Functions: Basic Theory*, Wiley & Son, New York.

Watson, G. S. (1966). "Smooth Regression Analysis", *Sankhya*, 26, 359-378.

Wright, I. W. and Wegman, E. J. (1980). "Isotonic, Convex and Related Splines", *The Annals of Statistics*, 8, 1023-1035.

Appendices

Appendix A

Following is the calling sequence for COBS.

cobs(x, y, constraint, z, minz = knots[1], maxz = knots[nknots], nz = 100, knots, nknots, method = 'quantile', degree = 2, tau = 0.5, lambda = 0, ic = 'aic', knots.add = F, pointwise, print.warn = T, print.mesg = T, coef = rep(0, nvar), w = rep(1, n), maxiter = 20*n, lstart = log(.Machine$single.xmax) ** 2, factor = 1)

ARGUMENTS

x vector of covariate.

y vector of response variable. It must have the same length (n) as **x**.

constraint 'increase', 'decrease', 'convex', 'concave', 'periodic' or 'none'.

OPTIONAL ARGUMENTS

z vector of grid points at which the fitted values are evaluated; default to an equally spaced grid with **nz** grid points between **minz** and **maxz**. If the fitted values at **x** are desired, use z = unique(x).

minz needed if **z** is not given; default to min(**x**) or the first knot if **knots** are given.

maxz needed if **z** is not given; default to max(**x**) or the last knot if **knots** are given.

nz number of grid points in **z** if **z** is not given; default to 100.

knots vector of locations of the knot mesh; if missing, **nknots** number of **knots** will be created using the specified **method** and automatic knot selection will be carried out for regression B-spline (lambda = 0); if not missing and length(**knots**) == **nknots**, the provided **knot** mesh will be used in the fit and no automatic knot selection will be performed; otherwise, automatic knots selection will be performed on the provided **knots**.

nknots maximum number of knots; default to 6 for regression B-spline, 20 for smoothing B-spline.

method method used to generate **nknots** number of **knots** when **knots** is not provided; 'quantile' (equally spaced in percentile levels) or 'uniform' (equally spaced in covariate); default to 'quantile'.

degree degree of the splines; 1 for linear spline and 2 for quadratic spline; default to 2.

tau desired quantile level; default to 0.5 (median).

lambda penalty parameter; **lambda** == 0: no penalty (regression B-spline); **lambda**>0: smoothing B-spline with the given **lambda**; **lambda**<0: smoothing B-spline with lambda chosen by a Schwarz-type information criterion.

ic information criterion used in knot deletion and addition for regression B-spline method when **lambda** == 0; 'aic' (Akaike-type) or 'sic' (Schwarz-type); default to 'aic'.

knots.add logical; an additional step of stepwise knot addition will be performed for regression B-spline if T; the default is F.

pointwise an optional three-column matrix with each row specifying one of the following constraints: (1,xi,yi) – fitted value at xi will be >= yi; (-1,xi,yi) – fitted value at xi will be <= yi; (0,xi,yi) – fitted value at xi will be = yi; (2,xi,yi) – derivative of the fitted function at xi will be yi.

print.warn logical flag for printing of warning messages; default to T; probably needs to be set to F if performing monte carlo simulation.

print.mesg logical flag for printing of intermediate messages; default to T; probably needs to be set to F if performing monte carlo simulation.

coef initial guess of the B-spline coefficients; default to a vector of zeros.

w vector of weights the same length as **x** (**y**) assigned to both **x** and **y**; default to uniform weights adding up to one; using normalized weights that add up to one will speed up computation.

maxiter upper bound of the number of iteration; default to 20*n.

lstart starting value for lambda when performing parametric programming in lambda if **lambda**<0; default to log(.Machine$single.xmax)**2.

factor determines how big a step to the next smaller lambda should be while performing parametric linear programming in lambda; default to one will give all unique lambda's; use of bigger **factor** (> 1& < 4) will save time for big problems.

VALUE

coef B-spline coefficients.

fit fitted value at **z**.

resid vector of residuals from the fit.

z as in input.

knots the final set of knots used in the computation.

ifl exit code: 1 – ok; 2 – problem is infeasible, check specification of the **pointwise** argument; 3 – **maxiter** is reached before finding a solution, either increase **maxiter** and restart the program with **coef** and **knots** set to the value upon previous exit or use a smaller **lstart** value when **lambda**<0 or use a smaller **lambda** value when **lambda**>0; 4 – program aborted, numerical difficulties due to ill-conditioning.

icyc number of cycles taken to achieve convergence.

k the effective dimensionality of the final fit.

lambda the penalty parameter used in the final fit.

pp.lambda vector of all unique lambda's obtained from parametric programming when **lambda** < 0 on input.

sic vector of Schwarz information criteria evaluated at **pp.lambda**.

Appendix B

Splus source code for Figure 1a in Example 1.

```
# First, the default number of 6 knots is used
a50<-cobs(year,temp,z=year,knots.add=T,degree=1,
constraint="increase")
# As suggested in the warning message, we increase the
# number of knots to 9
a50<-cobs(year,temp,z=year,nknots=9,knots.add=T,
degree=1,constraint="increase")
# Now we use the same knot sequence for other percentiles
a10<-cobs(year,temp,z=year,nknots=length(a50$knots),
knots=a50$knot,degree=1,tau=0.1,constraint="increase")
a90<-cobs(year,temp,z=year,nknots=length(a50$knots),
knots=a50$knot,degree=1,tau=0.9,constraint="increase")
hot.idx<-temp>=a90$fit
cold.idx<-temp<=a10$fit
```

```
normal.idx<-(temp<a90$fit)&(temp>a10$fit)
plot(year,temp,type="n",xlab="Year",ylab="Temperature (C)",
ylim=c(-.7,.6))
lines(a50$z,a50$fit)
lines(a10$z,a10$fit)
lines(a90$z,a90$fit)
points(year[normal.idx],temp[normal.idx])
points(year[!normal.idx],temp[!normal.idx],pch=2)
text(year[hot.idx],temp[hot.idx]+.05,labels=year[hot.idx])
text(year[cold.idx],temp[cold.idx]-.05,labels=year[cold.idx])
```

Splus source code for Figure 1b in Example 1.

```
# First, the default number of 6 knots is used
a50<-cobs(year,temp,z=year,knots.add=T,degree=1,
constraint="none")
# As suggested in the warning message, we increase the
# number of knots to 9
a50<-cobs(year,temp,z=year,nknots=9,knots.add=T,
degree=1,constraint="none")
# Here, we use the same knot sequence chosen for the
# 50th percentile for other percentiles
a10<-cobs(year,temp,z=year,nknots=length(a50$knots),
knots=a50$knot,degree=1,tau=0.1,constraint="none")
a90<-cobs(year,temp,z=year,nknots=length(a50$knots),
knots=a50$knot,degree=1,tau=0.9,constraint="none")
hot.idx<-temp>=a90$fit
cold.idx<-temp<=a10$fit
normal.idx<-(temp<a90$fit)&(temp>a10$fit)
plot(year,temp,type="n",xlab="Year",ylab="Temperature (C)",
ylim=c(-.7,.6))
lines(a50$z,a50$fit)
lines(a10$z,a10$fit)
lines(a90$z,a90$fit)
points(year[normal.idx],temp[normal.idx])
points(year[!normal.idx],temp[!normal.idx],pch=2)
text(year[hot.idx],temp[hot.idx]+.05,labels=year[hot.idx])
text(year[cold.idx],temp[cold.idx]-.05,labels=year[cold.idx])
```

Splus source code for the Figure 2a and 2b in Example 2.

```
# Compute the quadratic median smoothing B-spline with SIC
# chosen lambda
```

```
a_cobs(age,fci,constraint="decrease",lambda=-1,nknots=10,
degree=2,pointwise=rbind(c(0,0,100)))
# Since the algorithm hasn't converged, rerun cobs with coef
# and knots set to the previously returned values
a_cobs(age,fci,constraint="decrease",lambda=-1,nknots=10,
degree=2,knots=a$knots,coef=a$coef,pointwise=rbind(c(0,0,100)))
# Since the SIC chosen lambda corresponds to the roughest
# possible fit, we plot $pp.lambda against $sic
plot(a$pp.lambda,a$sic,type="l",log="x",xlab="Lambda",
ylab="SIC")
abline(v=14.04,lty=2)
# This plot suggests using lambda=14.04 so rerun robs
a_cobs(age,fci,constraint="decrease",lambda=14.04,nknots=10,
pointwise=rbind(c(0,0,100)))
# Compute the 25th percentile smoothing B-spline
a25_cobs(age,fci,constraint="decrease",lambda=-1,nknots=10,
tau=.25,pointwise=rbind(c(0,0,100)))
# Again plot $sic against $pp.lambda
plot(a25$pp.lambda,a25$sic,type="l",log="x")
abline(v=42.07,lty=2)
# Choose lambda=42.07
a25_cobs(age,fci,constraint="decrease",lambda=42.07,nknots=10,
tau=.25,pointwise=rbind(c(0,0,100)))
# Compute the 75th percentile smoothing B-spline
a75_cobs(age,fci,constraint="decrease",lambda=-1,nknots=10,
tau=.75,pointwise=rbind(c(0,0,100)))
# Even though cobs hasn't suggested, we plot $sic against
# $pp.ambda anyway
plot(a75$pp,a75$sic,type="l",log="x")
# Generate Figure 2b
plot(age,fci,xlim=c(0,15),ylim=c(0,100),cex=.5,xlab="AGE",
ylab="FCI")
title("Figure 2: Roof Age Degradation Curves")
lines(c(0,a$z),c(100,a$fit))
lines(c(0,a25$z),c(100,a25$fit))
lines(c(0,a75$z),c(100,a75$fit))
```

Splus codes for Figure 3a and 3b in Example 3.

```
wind.x_c(rep(c(rep(1:365,3),1:366),4),rep(1:365,2))
# Compute the quadratic median smoothing B-spline using SIC
# to choose the lambda first
cobs(wind.x,DUB.,knots.add=T,constraint="periodic",nknots=13,
tau=.5,lambda=-1,factor=3,method="uniform")->cobs.o
```

```
# Since SIC chooses a lambda that corresponds to the smoothest
# possible fit, re-run cobs with a larger lstart value
lstart_log(.Machine$single.xmax)^3
cobs(wind.x,DUB.,knots.add=T,constraint="periodic",nknots=13,
lambda=-1,factor=3,method="uniform",lstart=lstart)->cobs.o
# Compute the .9 quantile smoothing B-spline
cobs(wind.x,DUB.,knots.add=T,constraint="periodic",nknots=13,
tau=.9,lambda=-1,factor=3,method="uniform")->cobs.o.9
# Compute the .1 quantile smoothing B-spline
cobs(wind.x,DUB.,knots.add=T,constraint="periodic",nknots=13,
tau=.1,lambda=-1,factor=3,method="uniform")->cobs.o.1
par(mfrow=c(1,2))
plot(wind.x,DUB.,pch=3,xlab="DAYS",ylab="SPEED (knots)")
plot(wind.x,DUB.,type="n",xlab="DAYS",ylab="SPEED (knots)")
lines(cobs.o$z,cobs.o$fit)
lines(cobs.o.9$z,cobs.o.9$fit)
lines(cobs.o.1$z,cobs.o.1$fit)
```

Figure 1a. Monotonically increasing linear regression B-splines for global temperature at $\tau = .1, .5, .9$.

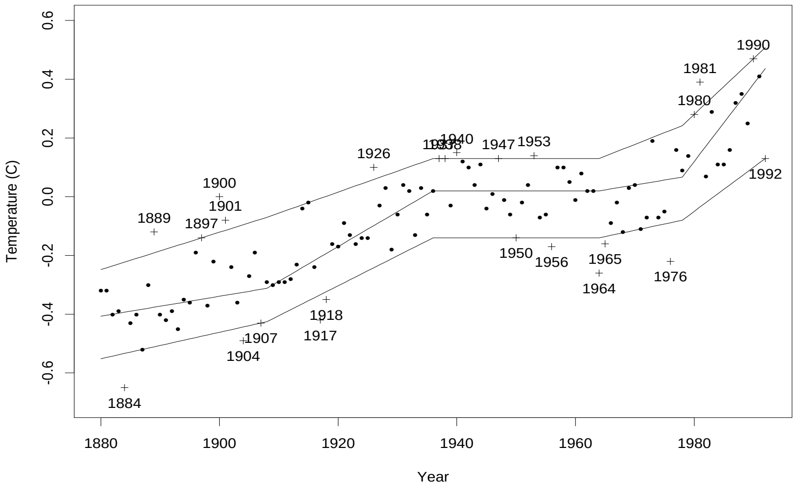

Figure 1b. Unconstrained linear regression B-splines for global temperature at $\tau = .1, .5, .9$.

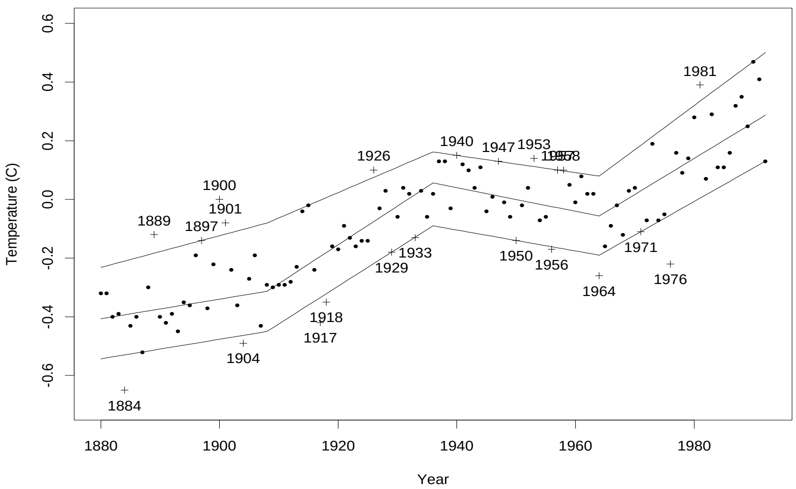

Figure 2a. SIC plot for the monotonically decreasing quadratic median smoothing B-spline for FCI degradation. A good choice of λ is the second minimizer at 14.04.

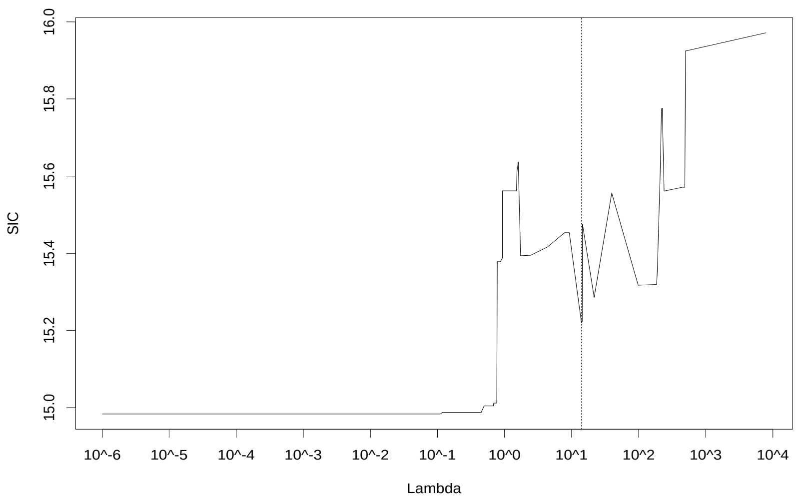

Figure 2b. Monotonically decreasing quadratic smoothing B-splines for FCI degradation at $\tau = .25, .5$, and $.75$ with $\lambda = 42.07, 14.04$, and 79.12 respectively.

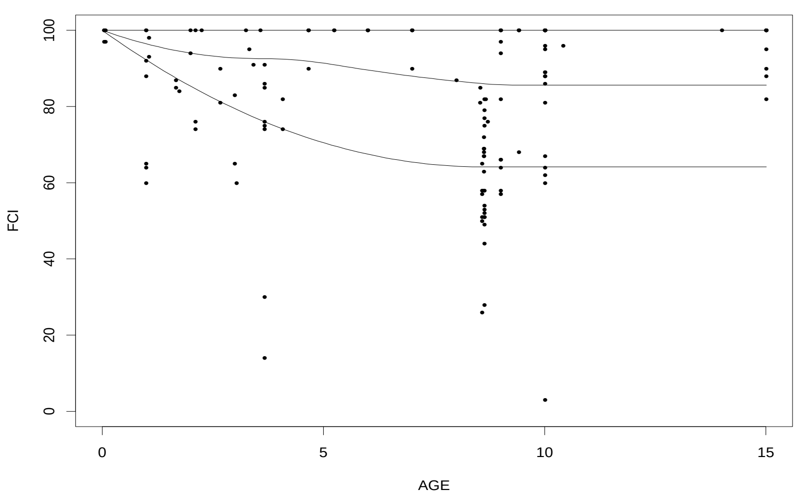

Figure 3a. Scatter plot of wind speed (in knots) in Dublin, Ireland.

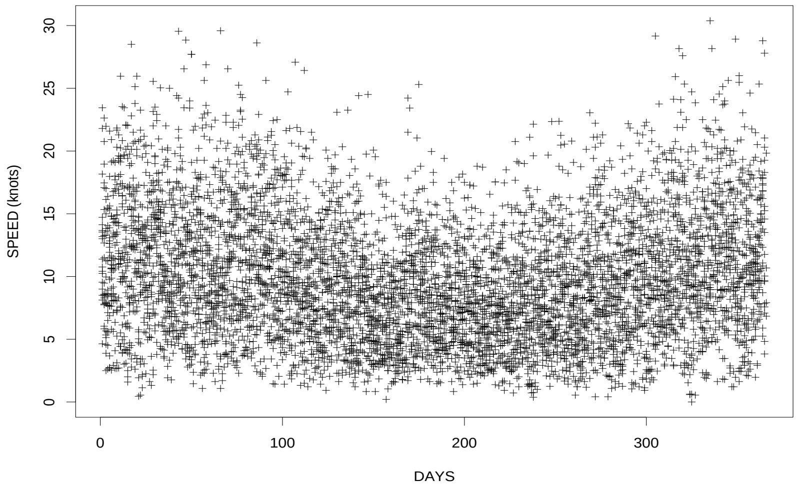

Figure 3b. The 'periodic' constrained quadratic smoothing B-spline fits for $\tau = .1, .5$, and $.9$. The smoothing parameters are $\lambda = 74.55 * 10^2, 99.70 * 10^3$, and $75.67 * 10^2$ respectively. The dotted lines indicate the location of the knots, which are the first days of all the twelve months of the year.