sg151.*y*	*B*–splines and splines parameterized by their values at reference points

Author: Roger B. Newson, Imperial College London, UK. Email: r.newson@imperial.ac.uk Date: 06 January 2014.

Abstract

A package of 3 programs is presented for generating a basis of splines in an X–variable, to be input to regression programs to fit spline models. The first, `bspline`, generates a basis of Schoenberg B–splines, which avoid the stability problems associated with plus–functions. The second, `frencurv`, generates a basis of reference splines, whose parameters in the regression model are simply values of the spline at reference points on the X–axis. The third, `flexcurv`, is an easy–to–use version of `frencurv`, which generates reference splines with sensibly–spaced knots. `frencurv` and `flexcurv` have the additional option of generating an incomplete basis of reference splines, with the reference spline for one reference point omitted. This incomplete basis can be completed by adding the standard constant vector to the design matrix, and can then be used to estimate differences between values of the spline at the remaining reference points and the value of the spline at the omitted reference point.

Key phrases

Spline; B–spline; interpolation; quadratic; cubic.

Syntax

`bspline` [*newvarlist*] [`if` *exp*] [`in` *range*], `xvar(`*varname*`)` [`power(`#`)` `knots(`*numlist*`)` `noexknot` `generate(`*prefix*`)` `type(`*type*`)` `labfmt(`*format*`)` `labprefix(`*string*`)`]

`frencurv` [*newvarlist*] [`if` *exp*] [`in` *range*], `xvar(`*varname*`)` [`power(`#`)` `refpts(`*numlist*`)` `noexref` `omit(`#`)` `base(`#`)` `knots(`*numlist*`)` `noexknot` `generate(`*prefix*`)` `type(`*type*`)` `labfmt(`*format*`)` `labprefix(`*string*`)`]

`flexcurv` [*newvarlist*] [`if` *exp*] [`in` *range*], `xvar(`*varname*`)` [`power(`#`)` `refpts(`*numlist*`)` `omit(`#`)` `base(`#`)` `include(`*numlist*`)` `krule(`*knot_rule*`)` `generate(`*prefix*`)` `type(`*type*`)` `labfmt(`*format*`)` `labprefix(`*string*`)`]

where *knot_rule* is

`regular` | `interpolate`

Description

The `bspline` package contains 3 commands, `bspline`, `frencurv` and `flexcurv`. `bspline` generates a basis of B–splines in the X–variate based on a list of knots, for use in the design matrix of a regression model. `frencurv` generates a basis of reference splines, for use in the design matrix of a regression model, with the property that the parameters fitted will be values of the spline at a list of reference points. `flexcurv` is an easy–to–use version of `frencurv`, and generates reference splines with regularly–spaced knots, or with knots interpolated between the reference points. `frencurv` and `flexcurv` have the additional option of generating an incomplete basis of reference splines, which can be completed by the addition of the standard constant variable used in regression models. The splines are either given the names in the *newvarlist* (if present), or (more usually) generated as a list of numbered variables, prefixed by the `generate()` option. Usually (but not always), the regression command is called using the `noconst` option.

Options for use with `bspline` and `frencurv`

`xvar(`*varname*`)` specifies the X–variable on which the splines are based.

`power(`#`)` (a non–negative integer) specifies the power (or degree) of the splines. Examples are zero for constant, 1 for linear, 2 for quadratic, 3 for cubic, 4 for quartic or 5 for quintic. If absent, zero is assumed.

`knots(`*numlist*`)` specifies a list of at least 2 knots, on which the splines are based. If `knots()` is absent, then `bspline` will initialize the list to the minimum and maximum of the `xvar()` variable, and `frencurv` will create a list of knots equal to the reference points (in the case of odd–degree splines such as a linear, cubic or quintic) or midpoints between reference points (in the case of even–degree splines such as constant, quadratic or quartic). `flexcurv` does not have the `knots()` option, as it automatically generates a list of knots, containing the required number of knots "sensibly" spaced on the `xvar()` scale.

`noexknot` specifies that the original knot list is not to be extended. If `noexknot` is not specified, then the knot list is extended on the left and right by a number of extra knots on each side specified by `power()`, spaced by the

distance between the first and last 2 original knots, respectively. `flexcurv` does not have the `noexknot` option, as it specifies the knots automatically.

`generate(`*prefix*`)` specifies a prefix for the names of the generated splines, which (if there is no *newvarlist*) will be named as *prefix*`1`...*prefix*`N`, where `N` is the number of splines.

`type(`*type*`)` specifies the storage type of the splines generated (`float` or `double`). If `type()` is given as anything else (or not given), then it is set to `float`.

`labfmt(`*format*`)` specifies the format to be used in the variable labels for the generated splines. If absent, then it is set to the format of the `xvar()` variable.

`labprefix(`*string*`)` specifies the prefix to be used in the variable labels for the generated splines. If absent, then it is set to `"Spline at "` for `flexcurv` and `frencurv`, and to `"B-spline on "` for `bspline`.

Options for use with `frencurv`

`refpts(`*numlist*`)` specifies a list of at least 2 reference points, with the property that, if the splines are used in a regression model, then the fitted parameters will be values of the spline at those points. If `refpts()` is absent, then the list is initialized to two points, equal to the minimum and maximum of the `xvar()` variable. If the `omit()` option is specified with `flexcurv` or `frencurv`, and the spline corresponding to the omitted reference point is replaced with a standard constant term in the regression model, then the fitted parameters will be relative values of the spline (differences or ratios), compared to the value of the spline at the omitted reference point.

`noexref` specifies that the original reference list is not to be extended. If `noexref` is not specified, then the reference list is extended on the left and right by `int(`*power*`/2)` extra reference points on each side, where *power* is the value specified by `power()`, spaced by the distance between the first and last 2 original reference points, respectively. If `noexref` and `noexknot` are both specified, then the number of knots must be equal to the number of reference points plus $power + 1$. `flexcurv` does not have the `noexref` option, as it automatically chooses the knots and does not extend the reference points.

`omit(`#`)` specifies a reference point, which must be present in the `refpts()` list (after any extension requested by `frencurv`), and whose corresponding reference spline will be omitted from the set of generated splines. If the user specifies `omit()`, then the set of generated splines will not be a complete basis of the set of splines with the specified power and knots, but can be completed by the addition of a constant variable, equal to 1 in all observations. If the user then uses the generated splines as predictor variables for a regression command, such as `regress` or `glm`, then the `noconst` option should usually not be used, and, if the omitted reference point is in the completeness region of the basis, then the intercept parameter `_cons` will be the value of the spline at the omitted reference point, and the model parameters corresponding to the generated splines will be differences between the values of the spline at the corresponding reference points and the value of the spline at the omitted reference point. If `omit()` is not specified, then the generated splines form a complete basis of the set of splines with the specified power and knots. If the user then uses the generated splines as predictor variables for a regression command, such as `regress` or `glm`, then the `noconst` option should be used, and the fitted model parameters corresponding to the generated splines will be the values of the spline at the corresponding reference points.

`base(`#`)` is an alternative to `omit()` for use in Stata Versions 11 or higher. It specifies a reference point, which must be present in the `refpts()` list (after any extension requested by `frencurv`), and whose corresponding reference spline will be set to zero. If the user specifies `base()`, then the set of generated splines will not be a complete basis of the set of splines with the specified power and knots, but can be completed by the addition of a constant variable, equal to 1 in all observations. The generated splines can then be used in the design matrix by an estimation command in Stata Versions 11 or higher.

Options for use with `flexcurv` only

Note that `flexcurv` uses all the options available to `frencurv`, except for `knots()`, `noexknot`, and `noexref`.

`include(`*numlist*`)` specifies a list of additional numbers to be included within the boundaries of the completeness region of the spline basis, in addition to the available values of the `xvar()` variable and the `refpts()` values (if provided). This allows the user to specify a non–default infimum and/or supremum for the completeness region of the spline basis. If `include()` is not provided, then the completeness region will extend from the minimum to the maximum of the values either available in the `xvar()` variable or specified in the `refpts()` list.

`krule(`*knot_rule*`)` specifies a rule for generating knots, based on the reference points, which may be `regular` (the

default) or `interpolate`. If `regular` is specified, then the knots are spaced regularly over the completeness region of the spline. If `interpolate` is specified, then the knots are interpolated between the reference points, in a way that produces the same knots as `krule(regular)` if the reference points are regularly spaced. Whichever `krule()` option is specified, any extra knots to the left of the completeness region are regularly spaced with a spacing equal to that between the first 2 knots of the completeness region, and any extra knots to the right of the completeness region are regularly spaced with a spacing equal to that between the last 2 knots of the completeness region. Therefore, `krule(regular)` specifies that all knots will be regularly spaced, whether or not the reference points are regularly spaced, whereas `krule(interpolate)` specifies that the knots will be interpolated between the reference points in a way that will cause reference splines to be definable, even if the reference points are not regularly spaced.

Remarks

The options described above appear complicated, but imply simple defaults for most users, especially if `flexcurv` is used. Advanced users and programmers are given the power to specify a comprehensive choice of non–default splines. The splines are either given the names in the *newvarlist* (if present), or (more usually) generated as a list of numbered variables, prefixed by the `generate()` option. (The *newvarlist* is intended mainly for programmers, and allows them to store the splines in temporary variables with temporary names.)

Saved results

`bspline`, `frencurv` and `flexcurv` save the following results in `r()`:

Scalars

`r(xsup)`	upper bound of completeness region	`r(xinf)`	lower bound of completeness region
`r(nincomp)`	number of X–values out of completeness region	`r(nknot)`	number of knots
`r(nspline)`	number of splines	`r(power)`	power (or degree) of splines

Macros

`r(knots)`	final list of knots	`r(splist)`	*varlist* of generated splines
`r(labfmt)`	format used in spline variable labels	`r(labprefix)`	prefix used in spline variable labels
`r(type)`	storage type of splines (`float` or `double`)	`r(xvar)`	X–variable specified by `xvar()` option

Matrices

`r(knotv)`	row vector of knots

`frencurv` and `flexcurv` save all of the above results in `r()`, and also the following:

Scalars

`r(omit)`	omitted reference point specified by `omit()`	`r(base)`	base reference point specified by `base()`

Macros

`r(refpts)`	final list of reference points

Matrices

`r(refv)`	row vector of reference points

The result `r(nincomp)` is the number of values of the `xvar()` variable outside the completeness region of the space of splines defined by the reference splines or B–splines. The number lists `r(knots)` and `r(refpts)` are the final lists after any left and right extensions carried out by `bspline`, `frencurv` or `flexcurv`, and the vectors `r(knotv)` and `r(refv)` contain the same values in double precision (mainly for programmers). The scalars `r(xinf)` and `r(xsup)` are knots, such that the completeness region is $\texttt{r(xinf)} \leq x \leq \texttt{r(xsup)}$ for positive–degree splines and $\texttt{r(xinf)} \leq x < \texttt{r(xsup)}$ for zero–degree splines.

In addition, `bspline`, `frencurv` and `flexcurv` save variable characteristics for the output spline basis variables. The characteristic *varname*`[xvar]` is set by `bspline`, `frencurv` and `flexcurv` to be equal to the input X–variable name set by `xvar()`. The characteristics *varname*`[xinf]` and *varname*`[xsup]` are set by `bspline` to be equal to the infimum and supremum, respectively, of the interval of X–values for which the B–spline is non–zero. The characteristic *varname*`[xvalue]` is set by `frencurv` and `flexcurv` to be equal to the reference point on the X–axis corresponding to the reference spline.

Methods and Formulas for B–splines

The principles and definitions of B–splines are given in de Boor (1978) and Ziegler (1969). Practical applications in chemistry are described in Wold (1971 and 1974). They are used in signal processing, and are associated with a wavelet transformation (Unser, Aldroubi and Eden, 1992).

Splines are a method of defining models regressing a scalar Y–variate with respect to a scalar X–variate. By definition, a kth degree spline is defined with reference to a set of q knots $s_1 < s_2 < \ldots < s_q$, dividing the X–axis

into intervals of the form $[s_i, s_{i+1})$. In each of those intervals, the regression is a kth degree polynomial in X (usually a different one in each interval), but the polynomials in any two contiguous intervals have the same jth derivatives at the knot separating the two intervals, for j from zero to $k-1$. By convention, the 0th derivative is the function itself, so a 0th degree spline is simply a right–continuous step function, and a first–degree spline is a simple linear interpolation of values between the knots. (By convention, the intervals $[s_i, s_{i+1})$ are closed on the left and open on the right, but this convention only matters for splines of degree zero, which, by convention, are right–continuous rather than left–continuous.)

Splines can be defined using plus–functions. For a power k and a knot s, the kth power plus–function at s is defined as

$$P_k(x;s) = \begin{cases} (x-s)^k, & x \geq s, \\ 0, & x < s. \end{cases} \tag{1}$$

The plus–functions are a basis for the space of splines. That is to say, for any kth degree spline $S(\cdot)$, with knots $s_1 < s_2 < \ldots < s_q$, there exists a q–vector α such that, for any x,

$$S(x) = \sum_{j=1}^{q} \alpha_j P_k(x; s_j). \tag{2}$$

It might seem that, to fit a spline in a covariate X to a Y–variate, all we have to do is to define a design matrix U, such that $U_{ij} = P_k(x_i; s_j)$, and fit β as a vector of regression coefficients. This is not a good idea, for two reasons. First, there are problems with stability, as $P_k(x;s)$ will be very large for $k > 1$ and x much greater than s. Second, the β–parameters estimated will not be easy to explain in words to a non–mathematician. The first problem was solved with the introduction of B–splines by I. J. Schoenberg in the 1960s, and these are calculated by `bspline`. The second problem is solved using `frencurv` and `flexcurv`, which call `bspline`, and then transform the B–splines, so that the regression parameters will simply be relative or absolute values of the spline at reference points.

The B–splines define an alternative basis of the splines with a given set of knots. Ziegler (1969) defines the B–spline for a set of $k+2$ knots $s_1 < s_2 < ... < s_{k+2}$ as

$$B(x; s_1, \ldots, s_{k+2}) = (k+1) \sum_{j=1}^{k+2} \left[\prod_{1 \leq h \leq k+2, h \neq j} (s_h - s_j) \right]^{-1} P_k(x; s_j). \tag{3}$$

The B–spline (3) is positive for x in the open interval (s_1, s_{k+2}), and zero for other x. If the s_j are part of an extended set of knots extending forwards to $+\infty$ and backwards to $-\infty$, then the set of B–splines based on sets of $k+2$ consecutive knots forms a basis of the set of all kth–degree splines defined on the full set of knots. Figure 1 shows the constant, linear, quadratic and cubic B–splines originating at zero and corresponding to unit knots.

For the purposes of `bspline` and `frencurv`, I have taken the liberty of redefining B–splines by scaling the $B(x; s_1, \ldots, s_{k+2})$ of (3) by a factor equal to the mean distance between two consecutive knots, to arrive at the scale–invariant B–spline

$$A(x; s_1, \ldots, s_{k+2}) = \frac{s_{k+2} - s_1}{k+1} B(x; s_1, \ldots, s_{k+2}) = \begin{cases} \sum_{j=1}^{k+1} \prod_{h=1}^{k+2} \phi_{jh}(x), & \text{if } s_1 \leq x < s_{k+2}, \\ 0, & \text{otherwise,} \end{cases}$$

where the functions $\phi_{jh}(\cdot)$ are defined by

$$\phi_{jh}(x) = \begin{cases} 1, & \text{if } h = j, \\ (s_{k+2} - s_1)/(s_h - s_j), & \text{if } h = j+1, \\ P_1(x; s_j)/(s_h - s_j), & \text{otherwise.} \end{cases} \tag{4}$$

The scaled B–spline $A(x; s_1, \ldots, s_{k+2})$ has the advantage that it is dimensionless, being a sum of products of the dimensionless quantities $\phi_{hj}(x)$. That is to say, it is unaffected by the scale of units of the X–axis, and therefore has the same values, whether x is time in millennia or time in nanoseconds. The original Ziegler B–spline $B(x; s_1, \ldots, s_{k+2})$ is expressed in units of x^{-1}. Therefore, if the scaled B–spline $A(x; s_1, \ldots, s_{k+2})$ appears in a design matrix, then its regression coefficient is expressed in units of the Y–variate, whereas, if the original B–spline $B(x; s_1, \ldots, s_{k+2})$ appears in a design matrix, then its regression coefficient is expressed in Y–units multiplied by X–units, and will be difficult to interpret, even for a mathematician. The B–splines computed by `bspline` are therefore the $A(x; s_1, \ldots, s_{k+2})$,

and users who prefer the original Ziegler B–splines must scale them by $(k+1)/(s_{k+2}-s_1)$. (This factor happens to be one for splines with unit–spaced knots, such as those in Figure 1.)

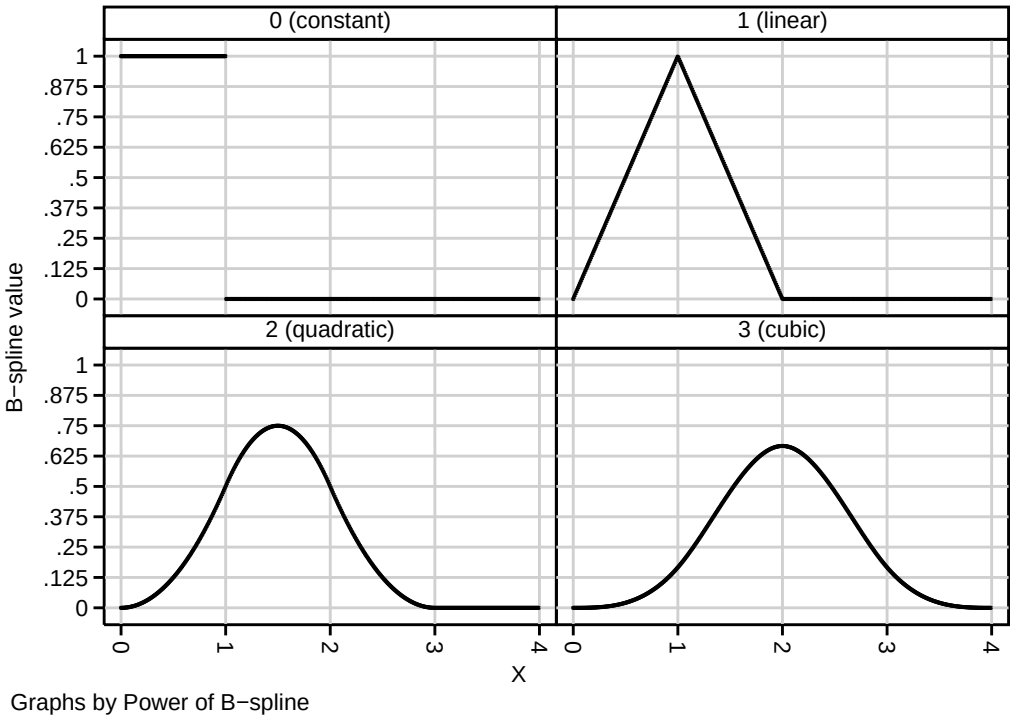

Figure 1. B–splines originating at zero with unit knots.

Given n data points, a Y–variate, an X–covariate, and a set of $q+k+1$ consecutive knots $s_h < \ldots < s_{h+q} < \ldots < s_{h+q+k}$, we can regress the Y–variate with respect to a kth degree spline in X by defining a design matrix V, with one row for each of the n data points and one column for each of the first q knots, such that

$$V_{ij} = A(x_i; s_{h+j-1}, \ldots, s_{h+j+k}). \tag{5}$$

We can then regress the Y–variate with respect to the design matrix V, and compute a vector β of regression coefficients, such that $V\beta$ is the fitted spline. The parameter β_j measures the contribution to the fitted spline of the B–spline originating at the knot s_{h+j-1} and terminating at the knot s_{h+j+k}. There will be no stability problems such as we are likely to have with the original plus–function basis, as each B–spline is bounded, and localized in its effect.

It is important to define enough knots. If the sequence of knots $\{s_j\}$ extends to $+\infty$ on the right and to $-\infty$ on the left, then the kth degree B–splines $A(\cdot; s_{h+j-1}, \ldots, s_{h+j+k})$ on sets of $k+2$ consecutive knots are a basis for the full space of kth degree splines on the full set of knots. If $S(\cdot)$ is one of these splines, and $[s_j, s_{j+1})$ is an interval between consecutive knots, then the values of $S(x)$ in the interval are affected by the $k+1$ B–splines originating at the knots $s_{j-k}, \ldots, s_j$ and terminating at the knots $s_{j+1}, \ldots, s_{j+k+1}$. It follows that, if we start by specifying a sequence of knots $s_0 < \ldots < s_m$, and we want to fit a spline for values of x in the interval $[s_0, s_m)$, then we must also use k extra knots $s_{-k} < \ldots < s_{-1}$ to the left of s_0, and k extra knots $s_{m+1} < \ldots < s_{m+k}$ to the right of s_m, to define the $m+k$ consecutive B–splines affecting $S(x)$ for x in the interval $[s_0, s_m)$. These $m+k$ B–splines originate at the knots $s_{-k}, \ldots, s_{m-1}$, and terminate at the knots $s_1, \ldots, s_{m+k}$, respectively. Any spline $S(\cdot)$, in the full space of kth degree splines defined using the full set of knots, is equal to a linear combination of these $m+k$ B–splines in the interval $[s_0, s_m]$ (in the case of positive–degree splines, which are continuous) or $[s_0, s_m)$ (in the case of zero–degree splines, which are only right–continuous). We will refer to this interval as the *completeness region* for splines which are linear combinations of these $m+k$ B–splines. These linear combinations are zero for $x < s_{-k}$ and $x \geq s_{m+k}$, and "incomplete" in the outer regions (s_{-k}, s_0) and (s_m, s_{m+k}), in which the spline is "returning to zero".

`bspline` and `frencurv` assume, in default, that the `knots()` option specified by the user is only intended to span the completeness region, and that the specified knots correspond to the $s_0, \ldots, s_m$. (`flexcurv` has no `knots()` option, as it defines its own "sensibly–spaced" knots, which are then input to `frencurv`.) In default, `bspline` and `frencurv` generate k extra knots on the left, with spacing equal to the difference between the first two knots, and

k extra knots on the right, with spacing equal to the difference between the last two knots. If the user specifies the option `noexknot`, then `bspline` assumes that the user has specified the full set of knots, corresponding to $s_{-k}, \ldots, s_{m+k}$, and does not generate any new knots. This allows users to specify their own spacing for the outer knots if they wish, but makes the specification of `knots()` simpler in the default case, because users do not have to count the extra outer knots for themselves.

Methods and formulas for reference splines

The B–spline regression parameters are expressed in units of the Y–variable, but they are not easy to interpret. If we have calculated the $n \times q$ matrix V of B–splines as in (5), and we also have a set of q reference X–values $r_1 < r_2 < \ldots < r_q$, then we might prefer to re–parameterize the spline by its values at the r_j. To do this, we first calculate a $q \times q$ square matrix W, defined such that

$$W_{ij} = A(r_i; s_{h+j-1}, \ldots, s_{h+j+k}), \tag{6}$$

the value of the jth B–spline at the ith reference point. If β is the (column) q–vector of regression coefficients with respect to the B–splines in V, and γ is the (column) q–vector of values of the spline at the reference points, then

$$\gamma = W\beta. \tag{7}$$

If W is invertible, then the n–vector of values of the fitted spline at the data points is

$$V\beta = VW^{-1}W\beta = VW^{-1}\gamma = Z\gamma, \tag{8}$$

where $Z = VW^{-1}$ is a transformed $n \times q$ design matrix, whose columns contain values of a set of reference splines, for the estimation of the reference–point spline values γ.

The choice of reference points is open to the user, and constrained mainly by the requirement that the matrix W is invertible. This implies that each of the q B–splines must be positive for at least one of the q reference values, and that each reference value must have at least one positive B–spline value. A natural choice of reference values might be one in the mid–range of each B–spline, possibly the central knot for an odd–degree B–spline (such as a linear, cubic or quintic), or the mid–point between the two central knots for an even–degree B–spline (such as a constant, quadratic or quartic). This choice has the consequence that, for a spline of degree k, there will be $\texttt{int}(k/2)$ reference points outside the spline's completeness region on the left, and another $\texttt{int}(k/2)$ reference points outside the spline's completeness region on the right, where $\texttt{int}(\cdot)$ is the truncation (or "integer–part") function. The parameters corresponding to these "extra" reference points will not be easy to explain to non–mathematicians, as they describe the behaviour of the spline as it returns to zero outside its completeness region. However, for a quadratic or cubic spline, there is only one such external reference Y–value at each end of the completeness region.

By default (if the user provides no `knots()` option), `frencurv` starts with the reference points originally provided (which default to the minimum and maximum of the `xvar()` variable if no `refpts()` option is provided), and chooses knots "appropriately". For an odd–degree spline (`power()` odd), the knots are initialized to the original reference points themselves. For an even–degree spline (`power()` even), the knots are initialized to mid–points corresponding to the original reference points. That is to say, if there are m original reference points $r_1 < \ldots < r_m$, and `power()` is even, then the original knots $s_0 < \ldots < s_m$ are initialized to

$$s_j = \begin{cases} r_1 - (r_2 - r_1)/2, & \text{if } j = 0, \\ (r_j + r_{j+1})/2, & \text{if } 1 \le j \le m-1, \\ r_m + (r_m - r_{m-1})/2, & \text{if } j = m. \end{cases} \tag{9}$$

`frencurv` assumes, by default, that the reference points initially provided are all in the completeness region, and adds $\texttt{int}(k/2)$ extra reference points to the left, spaced by the difference between the first two original reference points, and $\texttt{int}(k/2)$ extra reference points to the right, spaced by the difference between the last two original reference points, where k is specified by the `power()` option. If `noexref` is specified, then the original `refpts()` list is assumed to be the complete list, and it is the user's responsibility to choose sensible ones. In either case, the original knots are extended on the left and right as described above, unless `noexknot` is specified. (These rules seem complicated, but lead to sensible defaults if the naive user specifies a list of reference points and naively expects them to be in the completeness region of the spline, while preserving the ability of advanced users to specify exactly what they want at their own risk.)

Figure 2 shows the constant, linear, quadratic and cubic reference splines corresponding to a reference point at 4, assuming unit reference points and default knots (equal to reference points for odd degree and inter–reference midpoints for even degree). Note that each spline is one at its own reference point, and zero at all other reference points. (This is always the case for a basis of reference splines, at least for reference splines corresponding to reference points in the completeness region of the basis.) The reference splines are similar to (but not the same as) the B–spline wavelets of Unser *et al.* (1992).

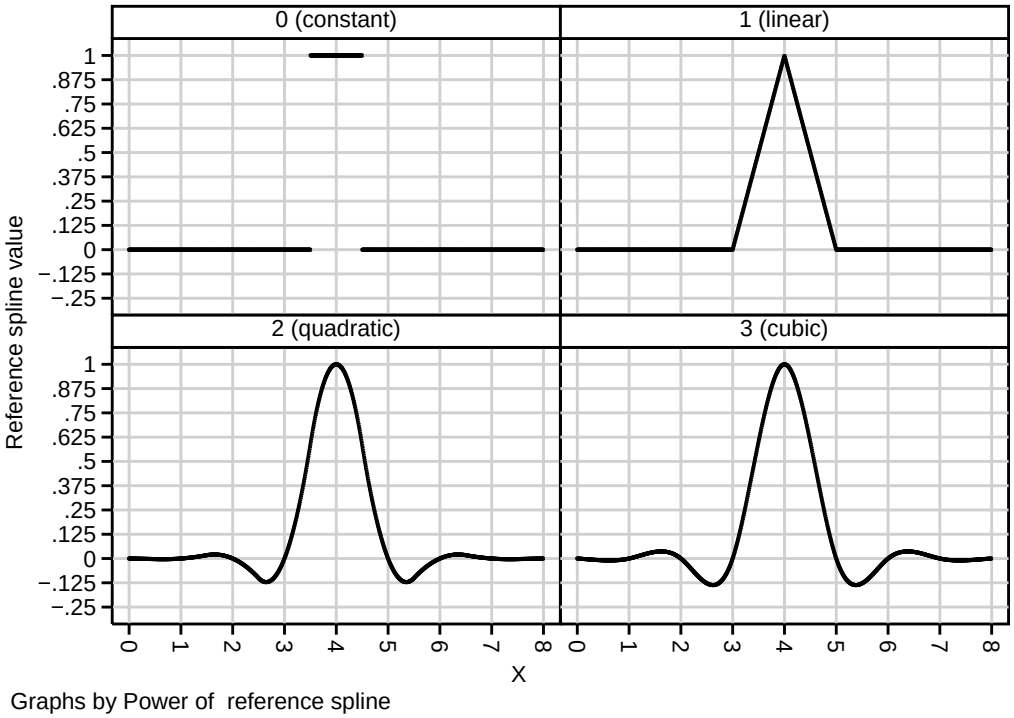

Figure 2. Reference splines at 4 with unit reference points.

`flexcurv` uses an alternative method to define knots from reference points, which guarantees that the reference points, the values of the X–variable specified by `xvar()`, and (optionally) a list of other X–values specified by the `include()` option will be in the completeness region of the generated spline basis. It also guarantees that the knots will be "sensibly" spaced, using a definition of sensibility specified by the `krule()` option. Suppose that there are q reference points $r_1, \ldots, r_q$ provided by the user in the `refpts()` option. `flexcurv` first calculates the numbers $x_{\inf}$ and $x_{\sup}$ as the minimum and maximum, respectively, of all values present in the `xvar()` variable, the `refpts()` list or the `include()` list. The numbers $x_{\inf}$ and $x_{\sup}$ will be the infimum and the supremum, respectively, of the completeness region of the spline basis. The number of intervals between adjacent knots in and bordering the completeness region is then $m = q - k$. The original knots in and bordering the completeness region are $s_0, \ldots, s_m$.

If the user specifies `krule(regular)` (the default), then these s_j are spaced regularly, and defined by the simple formula

$$s_j = \frac{j}{m} x_{\sup} + \frac{m-j}{m} x_{\inf}. \tag{10}$$

If the user specifies `krule(interpolate)`, then these s_j are interpolated between the reference points, using a more complicated formula. If the spline power k is 0, we define $s_0 = x_{\inf}$, $s_m = x_{\sup}$, and $s_j = r_{j+1}$ for other j. Otherwise, we first define, for each j from 0 to m,

$$\sigma(j) = 1 + j(q-1)/m, \quad \pi(j) = \mathrm{int}[\sigma(j)], \quad \rho(j) = \sigma(j) - \pi(j). \tag{11}$$

We then define the s_j as

$$s_j = \begin{cases} x_{\inf}, & j = 0, \\ x_{\sup}, & j = m, \\ [1-\rho(j)]\, r_{\pi(j)} \;+\; \rho(j)\, r_{\pi(j)+1}, & \text{otherwise.} \end{cases} \tag{12}$$

This formula ensures that the knots s_j are interpolated between the reference points in a way which will be regularly spaced, if the reference points themselves are regularly spaced from $r_1 = x_{\inf}$ to $r_q = x_{\sup}$. However, if the reference

points are not regularly spaced, then the user can specify `krule(interpolate)` to ensure that the reference splines will still be definable, which may possibly not be the case if the user specifies `krule(regular)` with irregularly–spaced reference points.

`flexcurv` then calls `frencurv` to generate the reference splines, with the reference points $r_1, \ldots, r_q$ as the `refpts()` option, and the knots $s_0, \ldots, s_m$ as the `knots()` option, with the `noexref` option but without the `noexknot` option. This implies that, whichever `krule()` option is specified, any extra knots to the left of the completeness region will be regularly spaced by the distance between the first 2 internal knots, and any extra knots to the right of the completeness region will be regularly spaced by the distance between the last 2 internal knots. Therefore, `krule(regular)` specifies that the knots inside and outside the completeness region are regularly spaced, so that any pair of adjacent knots inside or outside the completeness region is separated by $(x_{\text{sup}} - x_{\text{inf}})/m$ X–axis units. Both `krule()` options result in the generation of a basis of q reference splines, corresponding to the respective reference points, with a completeness region $x_{\text{inf}} \leq x \leq x_{\text{sup}}$ (for positive–degree splines) or $x_{\text{inf}} \leq x < x_{\text{sup}}$ (for zero–degree splines). Note that, in the case of zero–degree splines, the user must specify x_{sup} in the `include()` option, as a number strictly greater than any reference points and `xvar()` values, because x_{sup} is outside the completeness region for a zero–degree spline, which is a right–continuous step function with discontinuities at its knots, which include x_{sup}.

If the user specifies one of the reference points as the `omit()` option for `flexcurv` or `frencurv`, then the corresponding reference spline is dropped, and the basis will be incomplete. If the omitted reference point is in the completeness region, then this incomplete basis can be completed by including a constant variable, equal to 1 in all observations. This is because, within the completeness region of the reference spline basis, a constant variable is equal to a linear combination of the reference splines, with all coordinates equal to the constant. Therefore, if we drop a reference spline corresponding to a reference point in the completeness region, and substitute a variable equal to 1 throughout the completeness region, then the resulting list of variables will be a basis of the same spline space, with the same completeness region. However, if we then use this new basis as the design matrix in a regression model, then the parameter corresponding to the constant variable will be the value of the spline at the omitted reference point, and the other parameters, corresponding to the remaining reference points, will be the differences between the values of the spline at these remaining reference points and the value of the spline at the omitted reference point. Therefore, if the user uses `flexcurv` or `frencurv` to generate a spline basis for input to a regression model, then the `noconst` option should be used if no `omit()` option is specified, but not if an `omit()` option is specified. In this respect, the basis (or incomplete basis) of reference splines generated by `flexcurv` or `frencurv` is a continuous version of a single–factor basis (or incomplete basis) of indicator functions generated by `xi` with (or without) the `noomit` option.

In Stata Versions 11 or higher, the user may use the `base()` option for `flexcurv` or `frencurv`, instead of the `omit()` option, to specify a reference point whose corresponding reference spline is set to zero. The reference spline basis will then be an incomplete "super–basis", augmented by a zero vector, and can be completed similarly by adding a unit vector. This allows the user to use the "super–basis" in the design matrix of a regression model, fitted by a Stata Version 11 estimation command, with an omitted parameter corresponding to the base reference spline.

Example

In the `auto` data, we can use `flexcurv` and `regress` (with the `noconst` option) to fit a cubic spline for miles per gallon with respect to weight (in US pounds):

```
. flexcurv, xvar(weight) refpts(1760(770)4840) gen(cs) power(3)
. describe cs*
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------------------
cs1             float  %8.4f                  Spline at 1,760
cs2             float  %8.4f                  Spline at 2,530
cs3             float  %8.4f                  Spline at 3,300
cs4             float  %8.4f                  Spline at 4,070
cs5             float  %8.4f                  Spline at 4,840
. regress mpg cs*, robust noconst
Linear regression                                       Number of obs =      74
                                                        F(  5,    69) =  751.86
                                                        Prob > F      =  0.0000
                                                        R-squared     =  0.9780
                                                        Root MSE      =  3.3903
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
```

```
-------------+------------------------------------------------------------------
         cs1 |   30.83764    1.79533    17.18   0.000     27.25606    34.41923
         cs2 |   23.56232   .7657011    30.77   0.000     22.03479    25.08985
         cs3 |   18.79659    .546976    34.36   0.000     17.70541    19.88778
         cs4 |   15.95229   .8751852    18.23   0.000     14.20634    17.69824
         cs5 |   12.15876   .6937123    17.53   0.000     10.77484    13.54268
------------------------------------------------------------------------------
```

We have chosen the reference points (arbitrarily) to be equally spaced from the minimum of `weight` (1,760 pounds) to the maximum of `weight` (4,840 pounds). `flexcurv` ensures that the spline is complete in the interval of X–values spanned by the original reference points provided by the user. The `describe` command lists the reference splines, with their variable labels. The coefficients fitted by `regress` (with the `noconst` option) are simply the fitted values of `mpg` at the reference points. Figure 3 shows observed and fitted values of `mpg`, plotted against `weight`. The fitted curve is calculated using `predict` (see [R] **predict**), and is interpolated cubically between the reference points.

The `flexcurv` parameterization allows us to use `lincom` to calculate confidence intervals for differences (or other contrasts) between the values of the spline at different reference points. Here, we estimate the difference between expected mileage at weights of 3,300 and 4,840 pounds:

```
. lincom cs3-cs5
 ( 1)  cs3 - cs5 = 0
------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |   6.637834   .7610751     8.72   0.000     5.119531    8.156137
------------------------------------------------------------------------------
```

We see that cars weighing 3,300 pounds are expected to travel 5.12 to 8.16 more miles per gallon than cars weighing 4,070 pounds.

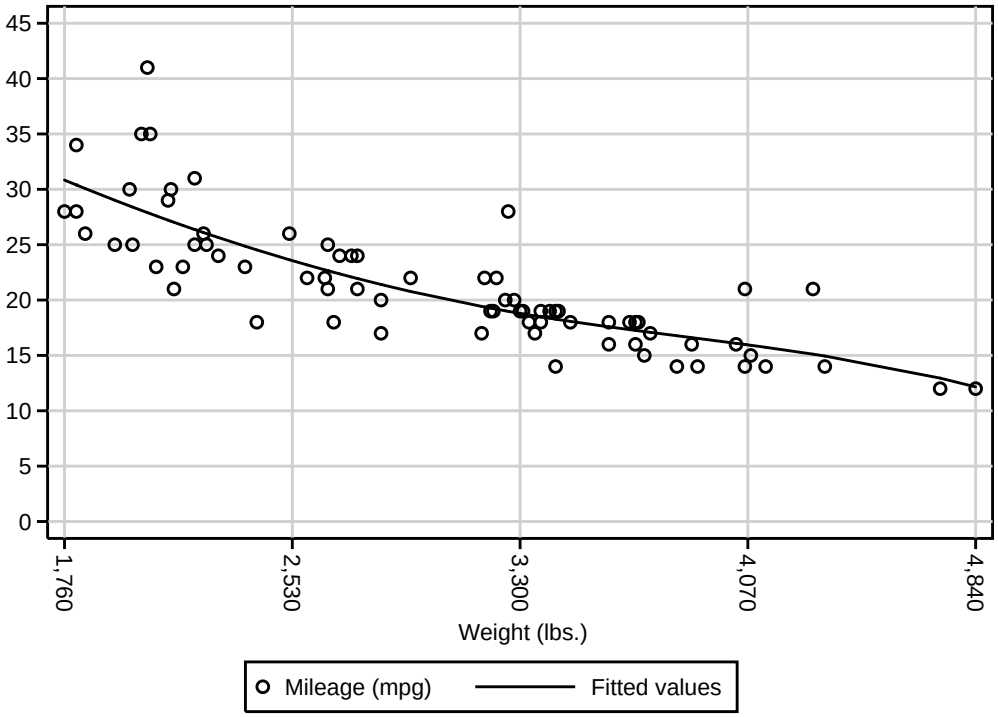

Figure 3. Mileage plotted against weight (points) with fitted cubic spline (line).

Alternatively, we might want to choose the lowest reference point (1,760) as the base level of `weight`, and estimate the difference in mileage between the other reference points and the base level. To do this, we use `flexcurv` with the `omit()` option, followed by `regress` without the `noconst` option. This time, only splines for reference points other than 1,760 pounds are generated. The regression model contains a parameter `_cons`, equal to the expected mileage for cars with the base weight of 1,760 US pounds. The other parameters are differences in mean mileage between cars with weights equal to the non–base reference weights and cars with weights equal to the base reference weight.

```
. flexcurv, xvar(weight) refpts(1760(770)4840) omit(1760) gen(ics) power(3)
. describe ics*
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
ics2            float  %8.4f                  Spline at 2,530
ics3            float  %8.4f                  Spline at 3,300
ics4            float  %8.4f                  Spline at 4,070
ics5            float  %8.4f                  Spline at 4,840
. regress mpg ics*, robust
Linear regression                                      Number of obs =      74
                                                       F(  4,    69) =   42.19
                                                       Prob > F      =  0.0000
                                                       R-squared     =  0.6754
                                                       Root MSE      =  3.3903
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        ics2 |  -7.275311   2.000318    -3.64   0.001    -11.26584   -3.284787
        ics3 |  -12.04104   1.853056    -6.50   0.000    -15.73779   -8.344299
        ics4 |  -14.88534   2.069253    -7.19   0.000    -19.01339   -10.75729
        ics5 |   -18.6789   1.937786    -9.64   0.000    -22.54468   -14.81312
       _cons |   30.83764   1.795329    17.18   0.000     27.25606    34.41923
------------------------------------------------------------------------------
```

Alternatively, we might want to fit the same model, with reference points equal to the knots of the spline in and around the completeness region of the spline, which are 1,760, 3,300 and 4,840 US pounds. This can be done using `frencurv`, which generates two extra reference points, equal to knots outside the completeness region. The variable labels of the reference splines at these reference points contain an indication that these reference points are outside the completeness region. The `regress` command is used with the `noconst` option, and estimates parameters for all the reference points, including the reference points outside the completeness region, whose corresponding parameters represent the behavior of the spline as it returns to zero outside its completeness region. These parameters are not easy to explain to non–mathematicians, and one of them even represents a negative mileage. However, in some applications, such as seasonal time series, the knots (or the between–knot midpoints) are typically placed where sudden change is likely to occur, and it might therefore be considered especially interesting to know the value of the spline at these knots (or midpoints).

```
. frencurv, xvar(weight) refpts(1760 3300 4840) gen(kcs) power(3)
. describe kcs*
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
kcs1            float  %8.4f                  Spline at 220 (INCOMPLETE)
kcs2            float  %8.4f                  Spline at 1,760
kcs3            float  %8.4f                  Spline at 3,300
kcs4            float  %8.4f                  Spline at 4,840
kcs5            float  %8.4f                  Spline at 6,380 (INCOMPLETE)
. regress mpg kcs*, robust noconst
Linear regression                                      Number of obs =      74
                                                       F(  5,    69) =  751.86
                                                       Prob > F      =  0.0000
                                                       R-squared     =  0.9780
                                                       Root MSE      =  3.3903
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        kcs1 |   39.04485   13.55122     2.88   0.005      12.0109    66.07879
        kcs2 |   30.83764    1.79533    17.18   0.000     27.25606    34.41923
        kcs3 |   18.79659    .546976    34.36   0.000     17.70541    19.88778
        kcs4 |   12.15876   .6937122    17.53   0.000     10.77484    13.54268
        kcs5 |   -.502905   11.44588    -0.04   0.965    -23.33681      22.331
------------------------------------------------------------------------------
```

It is also possible to specify the `omit()` option with `frencurv`. If we add the options `omit(1760) gen(ikcs)` to the previous example, and use `regress` without the `noconst` option, then the parameter `_cons` is the mileage at a weight of 1,760 US pounds, the parameters `ikcs3` and `ikcs4` are the differences between the mileages at

3,300 and 4,840 US pounds respectively, and the parameters `ikcs1` and `ikcs5` represent the behavior of the spline as it converges to zero outside its completeness region. (Note that the omitted reference point must be inside the completeness region for this interpretation to apply. `frencurv` issues a warning if the reference point specified by `omit()` is outside the completeness region.)

```
. frencurv, xvar(weight) refpts(1760 3300 4840) omit(1760) gen(ikcs) power(3)
. describe ikcs*
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
ikcs1           float  %8.4f                  Spline at 220 (INCOMPLETE)
ikcs3           float  %8.4f                  Spline at 3,300
ikcs4           float  %8.4f                  Spline at 4,840
ikcs5           float  %8.4f                  Spline at 6,380 (INCOMPLETE)
. regress mpg ikcs*, robust
Linear regression                               Number of obs =      74
                                                F(  4,    69) =   42.19
                                                Prob > F      =  0.0000
                                                R-squared     =  0.6754
                                                Root MSE      =  3.3903
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       ikcs1 |   13.34675   12.47861     1.07   0.289     -11.5474    38.24089
       ikcs3 |  -12.04104   1.853056    -6.50   0.000    -15.73779   -8.344299
       ikcs4 |   -18.6789   1.937786    -9.64   0.000    -22.54468   -14.81312
       ikcs5 |  -26.20107   11.34446    -2.31   0.024    -48.83264   -3.569498
       _cons |   30.83764   1.795329    17.18   0.000     27.25606    34.41923
------------------------------------------------------------------------------
```

Finally, for the most technical people, we can fit the same model yet again, using `bspline` instead of `frencurv`. Here, the splines are B–splines rather than reference splines. The variable labels show the interval with positive values of each B–spline, delimited by knots, including the extra knots calculated by `bspline`. The parameters are expressed in miles per gallon, but none of them are easy for non–mathematicians to interpret.

```
. bspline,xvar(weight) knots(1760 3300 4840) gen(bs) power(3)
. describe bs*
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
bs1             float  %8.4f                  B-spline on [-2,860,3,300)
bs2             float  %8.4f                  B-spline on [-1,320,4,840)
bs3             float  %8.4f                  B-spline on [220,6,380)
bs4             float  %8.4f                  B-spline on [1,760,7,920)
bs5             float  %8.4f                  B-spline on [3,300,9,460)
. regress mpg bs*, robust noconst
Linear regression                               Number of obs =      74
                                                F(  5,    69) =  751.86
                                                Prob > F      =  0.0000
                                                R-squared     =  0.9780
                                                Root MSE      =  3.3903
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         bs1 |   51.27904   21.29408     2.41   0.019     8.798513    93.75957
         bs2 |   29.15291   4.494233     6.49   0.000     20.18716    38.11866
         bs3 |   17.13522   2.879549     5.95   0.000     11.39068    22.87976
         bs4 |   15.08579   4.958562     3.04   0.003     5.193726    24.97785
         bs5 |  -4.525804   18.37739    -0.25   0.806    -41.18769    32.13608
------------------------------------------------------------------------------
```

Technical note

There are other programs in Stata to generate splines. `mkspline` (see [R] **mkspline**) generates a basis of linear splines to be used in a design matrix, as does `frencurv, power(1)`, but the basis is slightly different, because the fitted parameters for `frencurv` are reference values, whereas the fitted parameters for `mkspline` are the local slopes of the spline in the inter–knot intervals. Patrick Royston and Gareth Ambler's `splinegen` (Royston and Sauerbrei,

2007), William Dupont's `rc_spline` (downloadable from SSC), and Peter Sasieni's `spline` and `spbase` (Sasieni, 1994), from STB–22, are used for fitting the so–called "natural" cubic spline, which is constrained to be linear outside its completeness region, and is parameterized using the natural spline basis. (For more information about natural cubic splines, see Durrlemain and Simon, 1999, and/or Harrell, 2001.) The splines fitted using `bspline`, `frencurv` or `flexcurv`, on the other hand, are unconstrained, and parameterized using the B–spline or reference spline basis. `flexcurv`, `frencurv` and `bspline` are therefore complementary to the other programs, and do not supersede them.

Historical note

This document is a post–publication update of an article which appeared in the Stata Technical Bulletin (STB) as Newson (2000). A similar article, also updating the principles of reference splines, is Newson (2012).

Acknowledgements

The idea for the name `frencurv` came from Nicholas J. Cox of Durham University, UK, who remarked that the method was like an updated French curve when I described it on Statalist.

References

de Boor C. 1978. *A Practical Guide to Splines*. New York: Springer Verlag.

Durrlemain S. and R. Simon. 1999. Flexible regression models with cubic splines. *Statistics in Medicine* **8**: 551–561.

Harrell F. E. 2001. *Regression Modeling Strategies With Applications to Linear Models*, *Logistic Regression and Survival Analysis*. New York: Springer–Verlag.

Newson R. 2000. sg151: B–splines and splines parameterized by their values at reference points on the X–axis. *Stata Technical Bulletin* 57: 20–27. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 221–230.

Newson, R. B. 2012. Sensible parameters for univariate and multivariate splines. *The Stata Journal* **12**(3): 479–504.

Royston P., and W. Sauerbrei. 2007. Multivariate modelling with cubic regression splines: A principled approach. *The Stata Journal* **7**(1): 45-70.

Sasieni P. 1994. snp7: Natural cubic splines. *Stata Technical Bulletin* 22: 19–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 4, pp. 171–174.

Unser M., A. Aldroubi and M. Eden. 1992. On the asymptotic convergence of B–spline wavelets to Gabor functions. *IEEE Transactions on Information Theory* **38**: 864–872.

Wold S. 1971. Analysis of kinetic data by means of spline functions. *Chemica Scripta* **1**: 97–102.

Wold S. 1974. Spline functions in data analysis. *Technometrics* **16**: 1–11.

Ziegler Z. One–sided L_1–approximation by splines of an arbitrary degree. In: Schoenberg I. J. (ed.), 1969. *Approximations with Special Emphasis on Spline Functions*. New York: Academic Press.