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

Author: Roger Newson, King's College London, UK. Email: roger.newson@kcl.ac.uk Date: 21 December 2004.

Abstract

Two programs are presented for generating a basis of splines in an X-variable, to be used by 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. These programs are complementary to existing spline programs in Stata, and do not supersede them.

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(`*%fmt*`)`]

`frencurv` [*newvarlist*] [`if` *exp*] [`in` *range*] , `xvar(`*varname*`)` [`power(`#`)` `refpts(`*numlist*`)` `noexref` `knots(`*numlist*`)` `noexknot` `generate(`*prefix*`)` `type(`*type*`)` `labfmt(`*%fmt*`)`]

Description

`bspline` generates a basis of B-splines in an X-variable, based on a list of knots, for use in fitting a regression model containing a spline in the X-variable. `frencurv` ("French curve") generates a basis of reference splines, for use in fitting a regression model, with the property that the fitted parameters will be values of the spline at a list of reference points on the X-axis. Usually, the regression command is called with the `noconst` option.

Options for `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, eg 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 ascending knots, on which the splines are based. If `knots` are absent, then `bspline` will initialize the list to the minimum and maximum of `xvar`, 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).

`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 `power` extra knots on each side, spaced by the distance between the first and last 2 original knots, respectively.

`generate(`*prefix*`)` specifies a prefix for the names of the generated splines, which (if there is no *newvarlist*) will be named as *prefix1*...*prefixN*, 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(`*%fmt*`)` 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`.

Options for `frencurv` only

`refpts(`*numlist*`)` specifies a list of at least 2 ascending 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 `xvar`.

`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, 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`.

Remarks

The options described above appear complicated, but imply simple defaults for most users. 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.)

Methods and Formulas

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 k'th 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 k'th degree polynomial in X (usually a different one in each interval), but the polynomials in any two contiguous intervals have the same j'th derivatives at the knot separating the two intervals, for j from zero to $k-1$. By convention, the zero'th derivative is the function itself, so a zero'th 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 k'th 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 k'th 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. Firstly, there are problems with stability, as $P_k(x;s)$ will be very large for $k > 1$ and x much greater than s. Secondly, 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`, which calls `bspline`, and then transforms the B-splines, so that the regression parameters will simply be 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 k'th-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.)

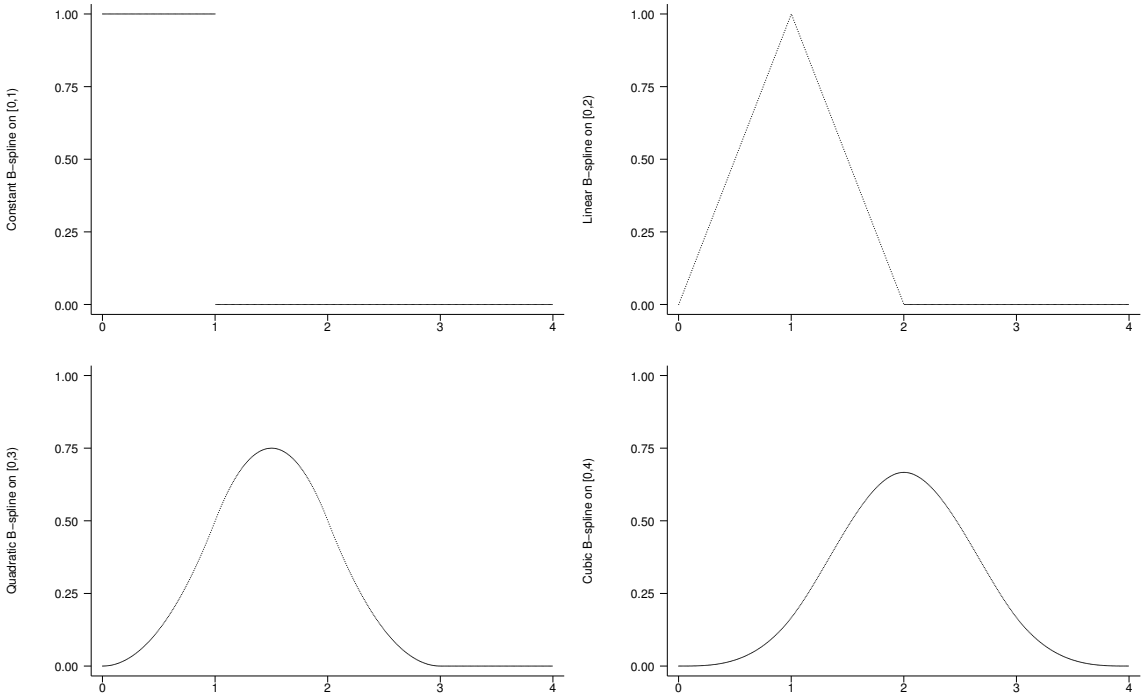

Figure 1. B-splines originating at zero with unit knots.

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 k'th 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 k'th 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 k'th 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 k'th degree splines defined using the full set of knots, is equal to a linear combination of these $m + k$ B-splines in the interval $[s_0, s_m)$, which we will denote 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 $s_0, \ldots, s_m$. 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.

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 j'th B-spline at the i'th 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 range.

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 `xvar` if no `refpts` are 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

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

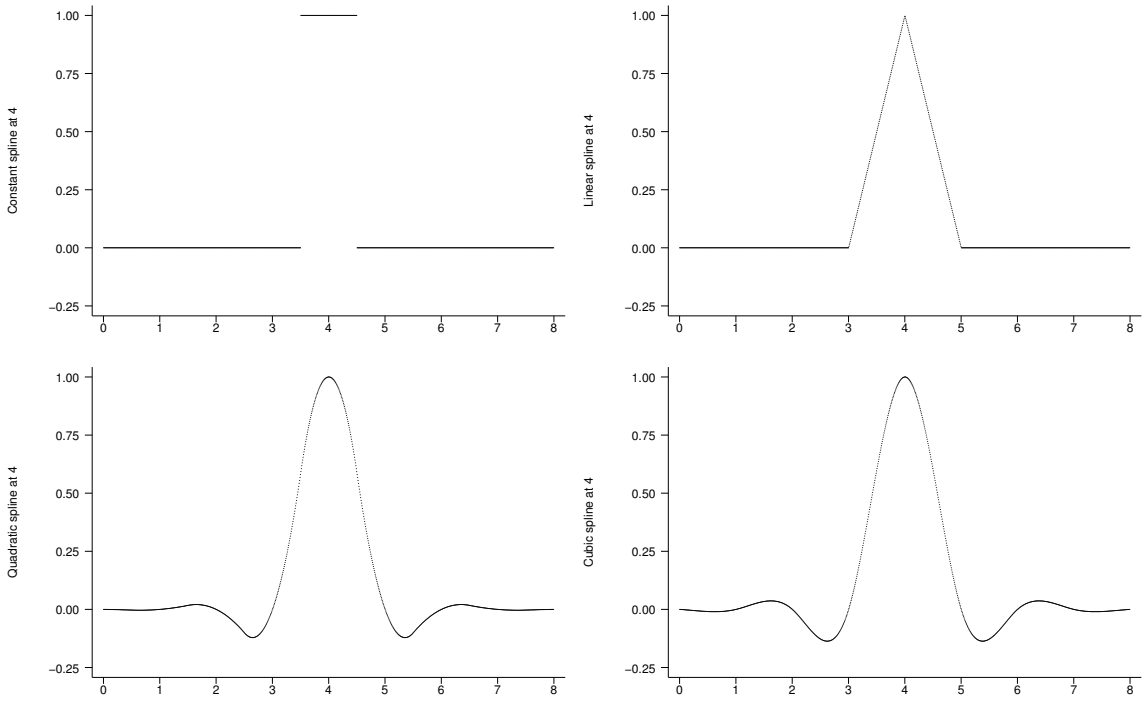

Figure 2. Reference splines at 4 with unit reference points.

`frencurv` assumes, by default, that the reference points initially provided are all in the completeness region, and adds `int`$(k/2)$ extra reference points to the left, spaced by the difference between the first two original reference points, and `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` are assumed to be the complete set, 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. They are similar to (but not the same as) the B-spline wavelets of Unser *et al.* (1992).

Example

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

```
. frencurv,xvar(weight) refpts(1760(770)4840) gen(cs) power(3)
. describe cs*
  14. cs1       float  %8.4f                 Spline at 990 (INCOMPLETE)
  15. cs2       float  %8.4f                 Spline at 1,760
  16. cs3       float  %8.4f                 Spline at 2,530
  17. cs4       float  %8.4f                 Spline at 3,300
  18. cs5       float  %8.4f                 Spline at 4,070
  19. cs6       float  %8.4f                 Spline at 4,840
  20. cs7       float  %8.4f                 Spline at 5,610 (INCOMPLETE)
. regress mpg cs*,noconst robust
Regression with robust standard errors                Number of obs =      74
                                                      F(  7,    67) =  618.91
                                                      Prob > F      =  0.0000
                                                      R-squared     =  0.9792
                                                      Root MSE      =  3.3469
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         cs1 |   11.82559   15.56642      0.760   0.450      -19.24512    42.89629
         cs2 |   29.21133   1.761704     16.581   0.000       25.69495    32.72771
         cs3 |   22.65796   .7625134     29.715   0.000       21.13597    24.17994
         cs4 |    19.4749    .610094     31.921   0.000       18.25715    20.69266
         cs5 |   15.51593   .8409023     18.452   0.000       13.83748    17.19437
         cs6 |   10.60747   1.585487      6.690   0.000       7.442828    13.77212
         cs7 |  -28.19347   21.59599     -1.305   0.196      -71.29924    14.91229
------------------------------------------------------------------------------
```

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). In default, `frencurv` ensures that the spline is complete in the range of X-values spanned by the original reference points provided by the user. The `describe` command lists the reference splines, with their labels. Note that `frencurv` has added two extra reference points outside the spline's completeness region (at weights of 990 and 5,610 pounds), and indicated this incompleteness in the variable labels. The coefficients fitted by `regress` (with the `noconst` option) are simply the fitted values of `mpg` at the reference points. Note that the ones corresponding to the splines `cs2` to `cs6` have "sensible" values, corresponding to the expected levels of `mpg` at the appropriate value of `weight`, whereas the ones corresponding to `cs1` and `cs7` have "nonsense" values, because they correspond to reference "weights" extrapolated off the edge of the range of sensible `weight` values. This is the price we pay for making all reference points equal to knots of the cubic spline. 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 `frencurv` 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 2,530 and 4,070 pounds:

```
. lincom cs3-cs5
 ( 1)  cs3 - cs5 = 0.0
------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |   7.142029   1.058829      6.745   0.000       5.028598     9.25546
------------------------------------------------------------------------------
```

We see that cars weighing 2,530 pounds are expected to travel 5.03 to 9.26 more miles per gallon than cars weighing 4,070 pounds. We can, instead, choose an alternative set of reference points, using `noexref` and specifying our own `knots`. The initial knots are the same initial knots as in the previous model (where they were also reference points), namely 5 equally spaced values from the minimum to the maximum of `weight`. However, the new reference points are $7(=5+2)$ equally spaced values covering the same range. The knots and the reference points are therefore out of synchrony, but the reference points are now all in the completeness region of the spline, because they are in the range spanned by the initial knots. (Remember that, in default, `bspline` and `frencurv` add new knots on the left and right to make the spline complete over the range of the original knots.) The model is exactly the same model as before (because a spline model is defined by the knots), but the parameters are now *all* sensible within-range `mpg` values, which non-technical people can understand. Note that we have used `labfmt` to handle the non-integer values of the reference points in the variable labels.

```
. frencurv,xvar(weight) refpts(1760(513.33333)4840) noexr k(1760(770)4840) gen(
> sp) power(3) labfmt(%7.2f)
. describe sp*
  22. sp1       float  %8.4f                 Spline at 1760.00
  23. sp2       float  %8.4f                 Spline at 2273.33
  24. sp3       float  %8.4f                 Spline at 2786.67
  25. sp4       float  %8.4f                 Spline at 3300.00
  26. sp5       float  %8.4f                 Spline at 3813.33
  27. sp6       float  %8.4f                 Spline at 4326.67
  28. sp7       float  %8.4f                 Spline at 4840.00
. regress mpg sp*,noconst robust
Regression with robust standard errors                Number of obs =      74
                                                      F(  7,    67) =  618.91
                                                      Prob > F      =  0.0000
                                                      R-squared     =  0.9792
                                                      Root MSE      =  3.3469
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sp1 |   29.21133   1.761704     16.581   0.000     25.69495    32.72771
         sp2 |   25.89924   1.073405     24.128   0.000     23.75671    28.04177
         sp3 |   20.98226   .7479685     28.052   0.000      19.4893    22.47521
         sp4 |    19.4749    .610094     31.921   0.000     18.25715    20.69266
         sp5 |   15.97982   .5560974     28.736   0.000     14.86985     17.0898
         sp6 |   16.74691   1.934879      8.655   0.000     12.88487    20.60894
         sp7 |   10.60747   1.585487      6.690   0.000     7.442828    13.77212
------------------------------------------------------------------------------
```

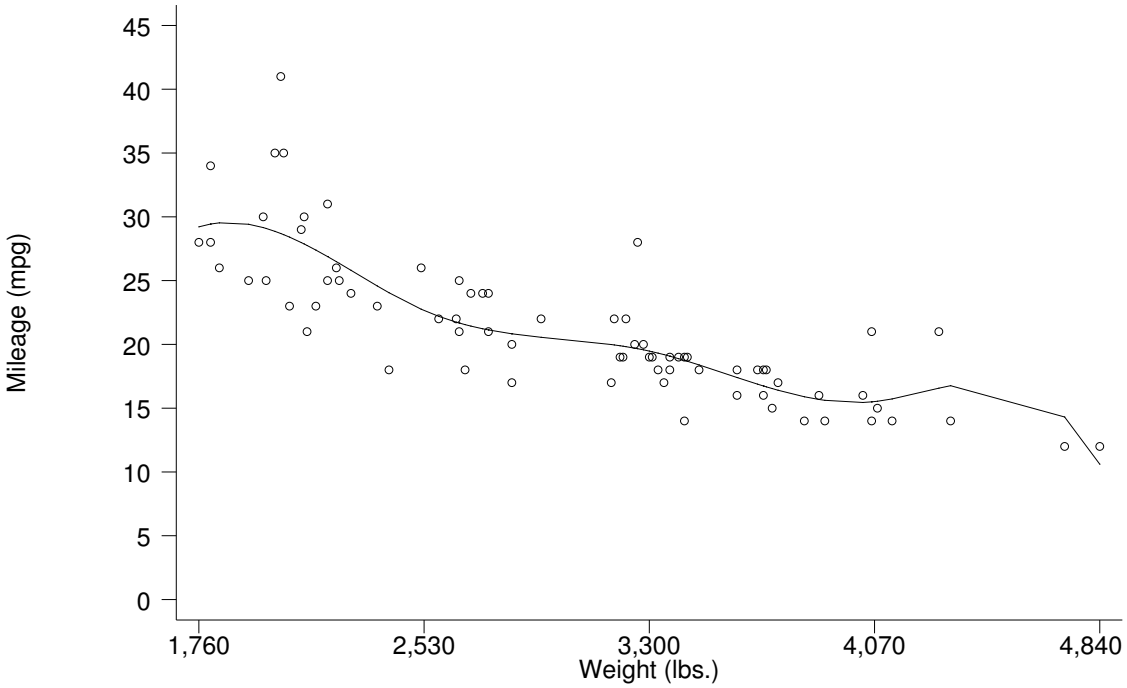

Figure 3. Mileage plotted against weight (points) with fitted cubic spline (line).

Finally, for the 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 range of 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 are not easy for non-mathematicians to interpret.

```
. bspline,xvar(weight) knots(1760(770)4840) gen(bs) power(3) labf(%4.0f)
. describe bs*
  29. bs1         float  %8.4f                 B-spline on [-550,2530)
  30. bs2         float  %8.4f                 B-spline on [220,3300)
  31. bs3         float  %8.4f                 B-spline on [990,4070)
  32. bs4         float  %8.4f                 B-spline on [1760,4840)
  33. bs5         float  %8.4f                 B-spline on [2530,5610)
  34. bs6         float  %8.4f                 B-spline on [3300,6380)
  35. bs7         float  %8.4f                 B-spline on [4070,7150)
. regress mpg bs*,noconst robust
Regression with robust standard errors                Number of obs =      74
                                                      F(  7,    67) =  618.91
                                                      Prob > F      =  0.0000
                                                      R-squared     =  0.9792
                                                      Root MSE      =  3.3469
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
         bs1 |   8.530818    24.5484      0.348   0.729        -40.468    57.52964
         bs2 |   36.83022   5.330421      6.909   0.000       26.19066    47.46979
         bs3 |   19.41627   2.252816      8.619   0.000       14.91963    23.91291
         bs4 |   21.45246   1.708278     12.558   0.000       18.04272     24.8622
         bs5 |   11.62333   2.241923      5.185   0.000       7.148434    16.09823
         bs6 |   25.14979   7.910832      3.179   0.002       9.359707    40.93988
         bs7 |  -48.57765   34.29427     -1.416   0.161      -117.0293    19.87399
------------------------------------------------------------------------------
```

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. William Dupont's `rc_spline`, downloadable from SSC, and Peter Sasieni's `spline` and `spbase` (Sasieni, 1994), from STB-22, are used for fitting a natural cubic spline, which is constrained to be linear outside the 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` or `frencurv`, on the other hand, are unconstrained (hence the extra degrees of freedom corresponding to the external reference points), and parameterized using the B-spline or reference spline basis, respectively. `frencurv` and `bspline` are therefore complementary to the other programs, and do not supersede them.

Saved results

`bspline` saves in `r()`:

Scalars

`r(xsup)`	upper bound of completeness region	`r(xinf)`	lower bound of completeness region

Macros

`r(nincomp)`	number of X-values out of completeness region	`r(knots)`	final list of knots
`r(splist)`	*varlist* of splines	`r(labfmt)`	format used in spline labels
`r(type)`	storage type of splines (`float` or `double`)	`r(nknot)`	number of knots
`r(nspline)`	number of splines	`r(power)`	power (or degree) of splines
`r(xvar)`	X-variable specified by `xvar` option		

Matrices

`r(knotv)`	row vector of knots

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

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 `xvar` 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` or `frencurv`, 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 $\texttt{r(xinf)} \leq x < \texttt{r(xsup)}$.

Historical note

This document is a post-publication update of an article which appeared in the Stata Technical Bulletin (STB) as Newson (2000). After 2001, STB was replaced by The Stata Journal (SJ), and all subsequent updates only appeared on SSC and on Roger Newson's homepage at *http://phs.kcl.ac.uk/rogernewson/*, which is accessible from within Web–aware Stata using the `net` command.

Acknowledgements

The idea for the name `frencurv` came from Nick 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.

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.