NAG Toolbox

nag_smooth_fit_spline_parest (g10ac)

1 Purpose

nag_smooth_fit_spline_parest (g10ac) estimates the values of the smoothing parameter and fits a cubic smoothing spline to a set of data.

2 Syntax

```
[yhat, c, rss, df, res, h, crit, rho, ifail] = nag_smooth_fit_spline_parest
(method, x, y, crit, 'n', n, 'wt', wt, 'u', u, 'tol', tol, 'maxcal', maxcal)
[yhat, c, rss, df, res, h, crit, rho, ifail] = g10ac(method, x, y, crit, 'n', n,
'wt', wt, 'u', u, 'tol', tol, 'maxcal', maxcal)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 24: **tol** was made optional; *weight* was removed from the interface; **wt** was made optional.

3 Description

For a set of n observations (x_i, y_i), for $i = 1, 2, \ldots, n$, the spline provides a flexible smooth function for situations in which a simple polynomial or nonlinear regression model is not suitable.

Cubic smoothing splines arise as the unique real-valued solution function f, with absolutely continuous first derivative and squared-integrable second derivative, which minimizes

$$\sum_{i=1}^{n} w_i (y_i - f(x_i))^2 + \rho \int_{-\infty}^{\infty} (f''(x))^2 \, dx,$$

where w_i is the (optional) weight for the ith observation and ρ is the smoothing argument. This criterion consists of two parts: the first measures the fit of the curve and the second the smoothness of the curve. The value of the smoothing argument ρ weights these two aspects; larger values of ρ give a smoother fitted curve but, in general, a poorer fit. For details of how the cubic spline can be fitted see Hutchinson and de Hoog (1985) and Reinsch (1967).

The fitted values, $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^{\mathrm{T}}$, and weighted residuals, r_i, can be written as:

$$\hat{y} = Hy \quad \text{and} \quad r_i = \sqrt{w_i}(y_i - \hat{y}_i)$$

for a matrix H. The residual degrees of freedom for the spline is $\mathrm{trace}(I - H)$ and the diagonal elements of H are the leverages.

The parameter ρ can be estimated in a number of ways.

(i) The degrees of freedom for the spline can be specified, i.e., find ρ such that $\mathrm{trace}(H) = \nu_0$ for given ν_0.

(ii) Minimize the cross-validation (CV), i.e., find ρ such that the CV is minimized, where

$$\mathrm{CV} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} \left[\frac{r_i}{1 - h_{ii}} \right]^2.$$

(iii) Minimize the generalized cross-validation (GCV), i.e., find ρ such that the GCV is minimized, where

$$\text{GCV} = \frac{n^2}{\sum_{i=1}^{n} w_i} \left[\frac{\sum_{i=1}^{n} r_i^2}{\left(\sum_{i=1}^{n} (1 - h_{ii}) \right)^2} \right].$$

nag_smooth_fit_spline_parest (g10ac) requires the x_i to be strictly increasing. If two or more observations have the same x_i value then they should be replaced by a single observation with y_i equal to the (weighted) mean of the y values and weight, w_i, equal to the sum of the weights. This operation can be performed by nag_smooth_data_order (g10za).

The algorithm is based on Hutchinson (1986). nag_roots_contfn_brent_rcomm (c05az) is used to solve for ρ given ν_0 and the method of nag_opt_one_var_func (e04ab) is used to minimize the GCV or CV.

4 References

Hastie T J and Tibshirani R J (1990) *Generalized Additive Models* Chapman and Hall

Hutchinson M F (1986) Algorithm 642: A fast procedure for calculating minimum cross-validation cubic smoothing splines *ACM Trans. Math. Software* **12** 150–153

Hutchinson M F and de Hoog F R (1985) Smoothing noisy data with spline functions *Numer. Math.* **47** 99–106

Reinsch C H (1967) Smoothing by spline functions *Numer. Math.* **10** 177–183

5 Parameters

5.1 Compulsory Input Parameters

1: **method** – CHARACTER(1)

Indicates whether the smoothing parameter is to be found by minimization of the CV or GCV functions, or by finding the smoothing parameter corresponding to a specified degrees of freedom value.

method = 'C'
Cross-validation is used.

method = 'D'
The degrees of freedom are specified.

method = 'G'
Generalized cross-validation is used.

Constraint: **method** = 'C', 'D' or 'G'.

2: **x**(**n**) – REAL (KIND=nag_wp) array

The distinct and ordered values x_i, for $i = 1, 2, \ldots, n$.

Constraint: $\mathbf{x}(i) < \mathbf{x}(i+1)$, for $i = 1, 2, \ldots, n-1$.

3: **y**(**n**) – REAL (KIND=nag_wp) array

The values y_i, for $i = 1, 2, \ldots, n$.

4: **crit** – REAL (KIND=nag_wp)

If **method** = 'D', the required degrees of freedom for the spline.

If **method** = 'C' or 'G', **crit** need not be set.

Constraint: $2.0 <$ **crit** $\leq$ **n**.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

n, the number of observations.

Constraint: **n** ≥ 3.

2: **wt**(:) – REAL (KIND=nag_wp) array

The dimension of the array **wt** must be at least **n** if $weight =$ 'W'

If $weight =$ 'W', **wt** must contain the n weights. Otherwise **wt** is not referenced and unit weights are assumed.

Constraint: if $weight =$ 'W', **wt**$(i) > 0.0$, for $i = 1, 2, \ldots, n$.

3: **u** – REAL (KIND=nag_wp)

Default: 0.0

The upper bound on the smoothing parameter. If **u** $\leq$ **tol**, **u** $= 1000.0$ will be used instead. See Section 9 for details on how this argument is used.

4: **tol** – REAL (KIND=nag_wp)

Default: 0.0

The accuracy to which the smoothing parameter **rho** is required. **tol** should preferably be not much less than $\sqrt{\epsilon}$, where ϵ is the ***machine precision***. If **tol** $< \epsilon$, **tol** $= \sqrt{\epsilon}$ will be used instead.

5: **maxcal** – INTEGER

Default: 0

The maximum number of spline evaluations to be used in finding the value of ρ. If **maxcal** < 3, **maxcal** $= 100$ will be used instead.

5.3 Output Parameters

1: **yhat**(**n**) – REAL (KIND=nag_wp) array

The fitted values, $\hat{y}_i$, for $i = 1, 2, \ldots, n$.

2: **c**$(ldc, \mathbf{3})$ – REAL (KIND=nag_wp) array

The spline coefficients. More precisely, the value of the spline approximation at t is given by $((\mathbf{c}(i,3) \times d + \mathbf{c}(i,2)) \times d + \mathbf{c}(i,1)) \times d + \hat{y}_i$, where $x_i \leq t < x_{i+1}$ and $d = t - x_i$.

3: **rss** – REAL (KIND=nag_wp)

The (weighted) residual sum of squares.

4: **df** – REAL (KIND=nag_wp)

The residual degrees of freedom. If **method** = 'D' this will be $n -$ **crit** to the required accuracy.

5: **res**(**n**) – REAL (KIND=nag_wp) array

The (weighted) residuals, r_i, for $i = 1, 2, \ldots, n$.

6: **h**(**n**) – REAL (KIND=nag_wp) array

The leverages, h_{ii}, for $i = 1, 2, \ldots, n$.

7: **crit** – REAL (KIND=nag_wp)

If **method** = 'C', the value of the cross-validation, or if **method** = 'G', the value of the generalized cross-validation function, evaluated at the value of ρ returned in **rho**.

8: **rho** – REAL (KIND=nag_wp)

The smoothing parameter, ρ.

9: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

Constraint: if **method** = 'D', **crit** > 2.0.

Constraint: if **method** = 'D', **crit** ≤ **n**.

Constraint: $ldc \geq \mathbf{n} - 1$.

Constraint: **n** ≥ 3.

On entry, **method** is not valid.

ifail = 2

On entry, at least one element of **wt** ≤ 0.0.

ifail = 3

On entry, **x** is not a strictly ordered array.

ifail = 4

For the specified degrees of freedom, **rho** > **u**:

ifail = 5 (*warning*)

Accuracy of **tol** cannot be achieved:

ifail = 6 (*warning*)

maxcal iterations have been performed.

ifail = 7 (*warning*)

Optimum value of **rho** lies above **u**:

ifail = −99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = −399

Your licence key may have expired or may not have been installed correctly.

ifail = −999

Dynamic memory allocation failed.

7 Accuracy

When minimizing the cross-validation or generalized cross-validation, the error in the estimate of ρ should be within $\pm 3(\mathbf{tol} \times \mathbf{rho} + \mathbf{tol})$. When finding ρ for a fixed number of degrees of freedom the error in the estimate of ρ should be within $\pm 2 \times \mathbf{tol} \times \max(1, \mathbf{rho})$.

Given the value of ρ, the accuracy of the fitted spline depends on the value of ρ and the position of the x values. The values of $x_i - x_{i-1}$ and w_i are scaled and ρ is transformed to avoid underflow and overflow problems.

8 Further Comments

The time to fit the spline for a given value of ρ is of order n.

When finding the value of ρ that gives the required degrees of freedom, the algorithm examines the interval 0.0 to **u**. For small degrees of freedom the value of ρ can be large, as in the theoretical case of two degrees of freedom when the spline reduces to a straight line and ρ is infinite. If the CV or GCV is to be minimized then the algorithm searches for the minimum value in the interval 0.0 to **u**. If the function is decreasing in that range then the boundary value of **u** will be returned. In either case, the larger the value of **u** the more likely is the interval to contain the required solution, but the process will be less efficient.

Regression splines with a small $(< n)$ number of knots can be fitted by nag_fit_1dspline_knots (e02ba) and nag_fit_1dspline_auto (e02be).

9 Example

This example uses the data given by Hastie and Tibshirani (1990), which consists of the age, x_i, and C-peptide concentration (pmol/ml), y_i, from a study of the factors affecting insulin-dependent diabetes mellitus in children. The data is input, reduced to a strictly ordered set by nag_smooth_data_order (g10za) and a spline with 5 degrees of freedom is fitted by nag_smooth_fit_spline_parest (g10ac). The fitted values and residuals are printed.

9.1 Program Text

```
     function g10ac_example

fprintf('g10ac example results\n\n');

x =  [ 5.2  8.8 10.5 10.6 10.4  1.8 12.7 15.6  5.8  1.9 ...
       2.2  4.8  7.9  5.2  0.9 11.8  7.9 11.5 10.6  8.5 ...
      11.1 12.8 11.3  1.0 14.5 11.9  8.1 13.8 15.5  9.8 ...
      11.0 12.4 11.1  5.1  4.8  4.2  6.9 13.2  9.9 12.5 ...
      13.2  8.9 10.8];
y =  [ 4.8  4.1  5.2  5.5  5.0  3.4  3.4  4.9  5.6  3.7 ...
       3.9  4.5  4.8  4.9  3.0  4.6  4.8  5.5  4.5  5.3 ...
       4.7  6.6  5.1  3.9  5.7  5.1  5.2  3.7  4.9  4.8 ...
       4.4  5.2  5.1  4.6  3.9  5.1  5.1  6.0  4.9  4.1 ...
       4.6  4.9  5.1];

% Reorder x, remove ties and weight accordingly
[n, x, y, wt, rss, ifail] = g10za( ...
                                   x, y);
x = x(1:n);
y = y(1:n);

% Control parameters
crit = 12;

% fit cubic spline
method = 'D';
[yhat, c, rss, df, res, h, crit, rho, ifail] = ...
  g10ac( ...
         method, x, y, crit, 'wt', wt);
```

```
%  Display results
fprintf('Residual sum of squares     = %10.2f\n', rss);
fprintf('Degrees of freedom          = %10.2f\n', df);
fprintf('rho                         = %10.2f\n', rho);
fprintf('\n     Input data                Output results\n');
fprintf('   i     x       y             yhat      h\n');
ivar = double(1:n)';
fprintf('%4d%8.3f%8.3f%14.3f%8.3f\n', [ivar x y yhat h]');
```

9.2 Program Results

```
     g10ac example results

Residual sum of squares     =      10.35
Degrees of freedom          =      25.00
rho                         =       2.68

     Input data                Output results
   i     x       y             yhat      h
   1   0.900   3.000          3.373   0.534
   2   1.000   3.900          3.406   0.427
   3   1.800   3.400          3.642   0.313
   4   1.900   3.700          3.686   0.313
   5   2.200   3.900          3.839   0.448
   6   4.200   5.100          4.614   0.564
   7   4.800   4.200          4.576   0.442
   8   5.100   4.600          4.715   0.189
   9   5.200   4.850          4.783   0.407
  10   5.800   5.600          5.193   0.455
  11   6.900   5.100          5.184   0.592
  12   7.900   4.800          4.958   0.530
  13   8.100   5.200          4.931   0.235
  14   8.500   5.300          4.845   0.245
  15   8.800   4.100          4.763   0.271
  16   8.900   4.900          4.748   0.292
  17   9.800   4.800          4.850   0.301
  18   9.900   4.900          4.875   0.277
  19  10.400   5.000          4.970   0.173
  20  10.500   5.200          4.977   0.154
  21  10.600   5.000          4.979   0.285
  22  10.800   5.100          4.970   0.136
  23  11.000   4.400          4.961   0.137
  24  11.100   4.900          4.964   0.284
  25  11.300   5.100          4.975   0.162
  26  11.500   5.500          4.975   0.186
  27  11.800   4.600          4.930   0.213
  28  11.900   5.100          4.911   0.220
  29  12.400   5.200          4.852   0.206
  30  12.500   4.100          4.857   0.196
  31  12.700   3.400          4.900   0.189
  32  12.800   6.600          4.932   0.193
  33  13.200   5.300          4.955   0.488
  34  13.800   3.700          4.797   0.408
  35  14.500   5.700          5.076   0.559
  36  15.500   4.900          4.979   0.445
  37  15.600   4.900          4.946   0.535
```