NAG Toolbox for MATLAB

g02hl

1 Purpose

g02hl calculates a robust estimate of the covariance matrix for user-supplied weight functions and their derivatives.

2 Syntax

```
[user, cov, a, wt, theta, nit, ifail] = g02hl(ucv, indm, n, x, a, theta,
nitmon, tol, 'user', user, 'm', m, 'bl', bl, 'bd', bd, 'maxit', maxit)
```

3 Description

For a set of n observations on m variables in a matrix X, a robust estimate of the covariance matrix, C, and a robust estimate of location, θ, are given by:

$$C = \tau^2 \left(A^{\mathrm{T}} A\right)^{-1},$$

where τ^2 is a correction factor and A is a lower triangular matrix found as the solution to the following equations.

$$z_i = A(x_i - \theta)$$

$$\frac{1}{n} \sum_{i=1}^{n} w\left(\|z_i\|_2\right) z_i = 0$$

and

$$\frac{1}{n} \sum_{i=1}^{n} u\left(\|z_i\|_2\right) z_i z_i^{\mathrm{T}} - v\left(\|z_i\|_2\right) I = 0,$$

where x_i is a vector of length m containing the elements of the ith row of X,

z_i is a vector of length m,

I is the identity matrix and 0 is the zero matrix,

and w and u are suitable functions.

g02hl covers two situations:

(i) $v(t) = 1$ for all t,

(ii) $v(t) = u(t)$.

The robust covariance matrix may be calculated from a weighted sum of squares and cross-products matrix about θ using weights $wt_i = u(\|z_i\|)$. In case a divisor of n is used and in case a divisor of $\sum_{i=1}^{n} wt_i$ is used. If $w(.) = \sqrt{u(.)}$, then the robust covariance matrix can be calculated by scaling each row of X by $\sqrt{wt_i}$ and calculating an unweighted covariance matrix about θ.

In order to make the estimate asymptotically unbiased under a Normal model a correction factor, τ^2, is needed. The value of the correction factor will depend on the functions employed (see Huber 1981 and Marazzi 1987a).

g02hl finds A using the iterative procedure as given by Huber.

$$A_k = (S_k + I) A_{k-1}$$

and

$$\theta_{j_k} = \frac{b_j}{D_1} + \theta_{j_{k-1}},$$

where $S_k = (s_{jl})$, for $j, l = 1, 2, \ldots, m$ is a lower triangular matrix such that:

$$s_{jl} = \begin{cases} -\min\left[\max\left(h_{jl}/D_3, -BL\right), BL\right], & j > l \\ -\min\left[\max\left(\left(h_{jj}/(2D_3 - D_4/D_2)\right), -BD\right), BD\right], & j = l \end{cases},$$

where

$$D_1 = \sum_{i=1}^{n} \left\{ w\left(\|z_i\|_2\right) + \tfrac{1}{m} w'\left(\|z_i\|_2\right) \|z_i\|_2 \right\}$$

$$D_2 = \sum_{i=1}^{n} \left\{ \tfrac{1}{p} \left(u'\left(\|z_i\|_2\right) \|z_i\|_2 + 2u\left(\|z_i\|_2\right) \right) \|z_i\|_2 - v'\left(\|z_i\|_2\right) \right\} \|z_i\|_2$$

$$D_3 = \tfrac{1}{m+2} \sum_{i=1}^{n} \left\{ \tfrac{1}{m} \left(u'\left(\|z_i\|_2\right) \|z_i\|_2 + 2u\left(\|z_i\|_2\right) \right) + u\left(\|z_i\|_2\right) \right\} \|z_i\|_2^{\,2}$$

$$D_4 = \sum_{i=1}^{n} \left\{ \tfrac{1}{m} u\left(\|z_i\|_2\right) \|z_i\|_2^{\,2} - v\left(\|z_i\|_2^{\,2}\right) \right\}$$

$$h_{jl} = \sum_{i=1}^{n} u\left(\|z_i\|_2\right) z_{ij} z_{il}, \text{ for } j > l$$

$$h_{jj} = \sum_{i=1}^{n} u\left(\|z_i\|_2\right) \left(z_{ij}^2 - \left\|z_{ij}\right\|_2^{\,2}/m \right)$$

$$b_j = \sum_{i=1}^{n} w\left(\|z_i\|_2\right) \left(x_{ij} - b_j\right)$$

and BD and BL are suitable bounds.

g02hl is based on routines in ROBETH; see Marazzi 1987a.

4 References

Huber P J 1981 *Robust Statistics* Wiley

Marazzi A 1987a Weights for bounded influence regression in ROBETH *Cah. Rech. Doc. IUMSP, No. 3 ROB 3* Institut Universitaire de Médecine Sociale et Préventive, Lausanne

5 Parameters

5.1 Compulsory Input Parameters

1: **ucv – string containing name of m-file**

ucv must return the values of the functions u and w and their derivatives for a given value of its argument.

Its specification is:

```
[user, u, ud, w, wd] = ucv(t, user)
```

Input Parameters

1: **t – double scalar**

The argument for which the functions u and w must be evaluated.

2: **user – Any MATLAB object**

ucv is called from g02hl with **user** as supplied to g02hl

Output Parameters

1: **user – Any MATLAB object**

ucv is called from g02hl with **user** as supplied to g02hl

2: **u – double scalar**

The value of the u function at the point **t**.

3: **ud – double scalar**

The value of the derivative of the u function at the point **t**.

4: **w – double scalar**

The value of the w function at the point **t**.

5: **wd – double scalar**

The value of the derivative of the w function at the point **t**.

2: **indm – int32 scalar**

Indicates which form of the function v will be used.

indm $= 1$

$v = 1$.

indm $\neq 1$

$v = u$.

3: **n – int32 scalar**

n, the number of observations.

Constraint: $\mathbf{n} > 1$.

4: **x(ldx,m) – double array**

ldx, the first dimension of the array, must be at least **n**.

$\mathbf{x}(i,j)$ must contain the ith observation on the jth variable, for $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$.

5: **a(m × (m + 1)/2) – double array**

An initial estimate of the lower triangular real matrix A. Only the lower triangular elements must be given and these should be stored row-wise in the array.

The diagonal elements must be $\neq 0$, and in practice will usually be > 0. If the magnitudes of the columns of X are of the same order, the identity matrix will often provide a suitable initial value for A. If the columns of X are of different magnitudes, the diagonal elements of the initial value of A should be approximately inversely proportional to the magnitude of the columns of X.

Constraint: $\mathbf{a}(j \times (j-1)/2 + j) \neq 0.0$, for $j = 1,2,\ldots,m$.

6: **theta(m) – double array**

An initial estimate of the location parameter, θ_j, for $j = 1, 2, \dots, m$.

In many cases an initial estimate of $\theta_j = 0$, for $j = 1, 2, \dots, m$, will be adequate. Alternatively medians may be used as given by g07da.

7: **nitmon – int32 scalar**

Indicates the amount of information on the iteration that is printed.

nitmon > 0

The value of A, θ and δ (see Section 7) will be printed at the first and every **nitmon** iterations.

nitmon ≤ 0

No iteration monitoring is printed.

When printing occurs the output is directed to the current advisory message unit (see x04ab).

8: **tol – double scalar**

The relative precision for the final estimates of the covariance matrix. Iteration will stop when maximum δ (see Section 7) is less than **tol**.

Constraint: **tol** > 0.0.

5.2 Optional Input Parameters

1: **user – Any MATLAB object**

user is not used by g02hl, but is passed to **ucv**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

2: **m – int32 scalar**

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

m, the number of columns of the matrix X, i.e., number of independent variables.

Constraint: $1 \leq$ **m** $\leq$ **n**.

3: **bl – double scalar**

The magnitude of the bound for the off-diagonal elements of S_k, BL.

Suggested value: **bl** $= 0.9$.

Default: 0.9

Constraint: **bl** > 0.0.

4: **bd – double scalar**

The magnitude of the bound for the diagonal elements of S_k, BD.

Suggested value: **bd** $= 0.9$.

Default: 0.9

Constraint: **bd** > 0.0.

5: **maxit – int32 scalar**

The maximum number of iterations that will be used during the calculation of A.

Suggested value: **maxit** $= 150$.

Default: 150

Constraint: **maxit** > 0.

5.3 Input Parameters Omitted from the MATLAB Interface

ldx, wk

5.4 Output Parameters

1: **user – Any MATLAB object**

user is not used by g02hl, but is passed to **ucv**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

2: $\mathbf{cov}(\mathbf{m} \times (\mathbf{m}+1)/2)$ **– double array**

Contains a robust estimate of the covariance matrix, C. The upper triangular part of the matrix C is stored packed by columns (lower triangular stored by rows), C_{ij} is returned in $\mathbf{cov}(j \times (j-1)/2 + i)$, $i \leq j$.

3: $\mathbf{a}(\mathbf{m} \times (\mathbf{m}+1)/2)$ **– double array**

The lower triangular elements of the inverse of the matrix A, stored row-wise.

4: $\mathbf{wt}(\mathbf{n})$ **– double array**

$\mathbf{wt}(i)$ contains the weights, $wt_i = u(\|z_i\|_2)$, for $i = 1, 2, \ldots, n$.

5: $\mathbf{theta}(\mathbf{m})$ **– double array**

Contains the robust estimate of the location parameter, θ_j, for $j = 1, 2, \ldots, m$.

6: **nit – int32 scalar**

The number of iterations performed.

7: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{n} \leq 1$,
or $\mathbf{m} < 1$,
or $\mathbf{n} < \mathbf{m}$,
or $\mathbf{ldx} < \mathbf{n}$.

ifail $= 2$

On entry, $\mathbf{tol} \leq 0.0$,
or $\mathbf{maxit} \leq 0.0$,
or diagonal element of $\mathbf{a} = 0.0$,
or $\mathbf{bl} \leq 0.0$,
or $\mathbf{bd} \leq 0.0$.

ifail = 3

A column of **x** has a constant value.

ifail = 4

Value of **u** or **w** returned by **ucv** < 0.

ifail = 5

The function has failed to converge in **maxit** iterations.

ifail = 6

One of the following is zero: D_1, D_2 or D_3.

This may be caused by the functions u or w being too strict for the current estimate of A (or C). You should try either a larger initial estimate of A or make u and w less strict.

7 Accuracy

On successful exit the accuracy of the results is related to the value of **tol**; see Section 5. At an iteration let

(i) $d1 =$ the maximum value of $|s_{jl}|$

(ii) $d2 =$ the maximum absolute change in $wt(i)$

(iii) $d3 =$ the maximum absolute relative change in θ_j

and let $\delta = \max(d1, d2, d3)$. Then the iterative procedure is assumed to have converged when $\delta <$ **tol**.

8 Further Comments

The existence of A will depend upon the function u (see Marazzi 1987a); also if X is not of full rank a value of A will not be found. If the columns of X are almost linearly related, then convergence will be slow.

9 Example

g02hl_ucv.m

```
function [user, u, ud, w, wd] = ucv(t, user)
  cu = user(1);
  u = 1.0;
  ud = 0.0;
  if (t ~= 0)
    t2 = t*t;
    if (t2 > cu)
      u = cu/t2;
      ud = -2.0*u/t;
    end
  end
  % w function and derivative
  cw = user(2);
  if (t > cw)
    w = cw/t;
    wd = -w/t;
  else
    w = 1.0;
    wd = 0.0;
  end
```

```
indm = int32(1);
n = int32(10);
x = [3.4, 6.9, 12.2;
     6.4, 2.5, 15.1;
     4.9, 5.5, 14.2;
     7.3, 1.9, 18.2;
     8.800000000000001, 3.6, 11.7;
     8.4, 1.3, 17.9;
     5.3, 3.1, 15;
     2.7, 8.1, 7.7;
     6.1, 3, 21.9;
     5.3, 2.2, 13.9];
a = [1;
     0;
     1;
     0;
     0;
     1];
theta = [0;
     0;
     0];
nitmon = int32(0);
tol = 5e-05;
user = [4,2];
[user, cov, aOut, wt, thetaOut, nit, ifail] = ...
    g02hl('g02hl_ucv', indm, n, x, a, theta, nitmon, tol, 'user', user)
```

```
user =
     4     2
cov =
    3.2778
   -3.6918
    5.2841
    4.7391
   -6.4086
   11.8371
aOut =
    0.5523
    1.0614
    0.9424
   -0.1881
    0.4776
    0.5021
wt =
    1.0000
    1.0000
    1.0000
    1.0000
    0.2339
    1.0000
    1.0000
    0.9385
    0.4012
    0.7579
thetaOut =
    5.6998
    3.8636
   14.7036
nit =
          25
ifail =
           0
```