nag_robust_corr_estim (g02hkc)

1. Purpose

nag_robust_corr_estim (g02hkc) computes a robust estimate of the covariance matrix for an expected fraction of gross errors.

2. Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_robust_corr_estim(Integer n, Integer m, double x[], Integer tdx,
         double eps, double cov[], double theta[], Integer max_iter,
         Integer print_iter, char *outfile, double tol, Integer *iter,
         NagError *fail)
```

3. Description

For a set 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:

$$C = \tau^2 (A^T A)^{-1}$$

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

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

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

and

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

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.

nag_robust_corr_estim uses weight functions:

$$u(t) = \frac{a_u}{t^2}, \quad \text{if } t < a_u^2$$

$$u(t) = 1, \quad \text{if } a_u^2 \le t \le b_u^2$$

$$u(t) = \frac{b_u}{t^2}, \quad \text{if } t > b_u^2$$

and

$$w(t) = 1, \quad \text{if } t \le c_w$$

$$w(t) = \frac{c_w}{t}, \quad \text{if } t > c_w$$

for constants a_u, b_u and c_w.

These functions solve a minimax problem considered by Huber (1981). The values of a_u, b_u and c_w are calculated from the expected fraction of gross errors, ϵ (see Huber (1981) and Marazzi (1987)). The expected fraction of gross errors is the estimated proportion of outliers in the sample.

In order to make the estimate asymptotically unbiased under a Normal model a correction factor, τ^2, is calculated, (see Huber (1981) and Marazzi (1987)).

Initial estimates of θ_j, for $j = 1, 2, ..., m$, are given by the median of the jth column of X and the initial value of A is based on the median absolute deviation (see Marazzi (1987)). nag_robust_corr_estim is based on routines in ROBETH, (see Marazzi (1987)).

4. Parameters

n

Input: the number of observations, n.
Constraint: $\mathbf{n} > 1$.

m

Input: the number of columns of the matrix X, i.e., number of independent variables, m.
Constraint: $1 \leq \mathbf{m} \leq \mathbf{n}$.

x[n][tdx]

Input: $\mathbf{x}[i-1][j-1]$ must contain the ith observation for the jth variable, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$.

tdx

Input: the second dimension of the array **x** as declared in the function from which nag_robust_corr_estim is called.
Constraint: $\mathbf{tdx} \geq \mathbf{m}$.

eps

Input: the expected fraction of gross errors expected in the sample, ϵ.
Constraint: $0.0 \leq \mathbf{eps} < 1.0$.

cov[m∗(m+1)/2]

Output: the $\mathbf{m} \times (\mathbf{m} + 1)/2$ elements of **cov** contain the upper triangular part of the covariance matrix.They are stored packed by column, i.e., C_{ij}, $j \geq i$, is stored in $\mathbf{cov}[j(j+1)/2+i]$, for $i = 0, 1, \ldots, \mathbf{m} - 1$ and $j = i, i+1, \ldots, \mathbf{m} - 1$.

theta[m]

Output: the robust estimate of the location parameters θ_j, for $j = 1, 2, \ldots, m$.

max_iter

Input: the maximum number of iterations that will be used during the calculation of the covariance matrix.
Suggested value: $\mathbf{max\_iter} = 150$.
Constraint: $\mathbf{max\_iter} > 0$.

print_iter

Input: indicates if the printing of information on the iterations is required and the rate at which printing is produced. The following values are available.
If $\mathbf{print\_iter} \leq 0$, then no iteration monitoring is printed.
If $\mathbf{print\_iter} > 0$, then the value of A, θ and δ (see Section 6.1) will be printed at the first and every **print_iter** iterations.

outfile

Input: a null terminated character string giving the name of the file to which results should be printed. If **outfile = NULL** or an empty string then the `stdout` stream is used. Note that the file will be opened in the append mode.

tol

Input: the relative precision for the final estimates of the covariance matrix.
Constraint: $\mathbf{tol} > 0.0$.

iter

Output: the number of iterations performed.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

For this function the values of output parameters may be useful even if **fail.code** ≠ **NE_NOERROR** on exit. Users are therefore advised to supply the **fail** parameter and set **fail.print** = `TRUE`.

5. Error Indications and Warnings

NE_INT_ARG_LT
On entry, **n** must not be less than 2: **n** = ⟨*value*⟩.
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.

NE_2_INT_ARG_GT
On entry, **m** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These parameters must satisfy **m** ≤ **n**.

NE_2_INT_ARG_LT
On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These parameters must satisfy **tdx** ≥ **m**.

NE_INT_ARG_LE
On entry, **max_iter** must not be less than or equal to 0: **max_iter** = ⟨*value*⟩.

NE_REAL_ARG_LT
On entry, **eps** must not be less than 0.0: **eps** = ⟨*value*⟩.

NE_REAL_ARG_GE
On entry, **eps** must be not be greater than or equal to 1.0: **eps** = ⟨*value*⟩.

NE_REAL_ARG_LE
On entry, **tol** must not be less than or equal to 0.0: **tol** = ⟨*value*⟩.

NE_CONST_COL
On entry, column ⟨*value*⟩ of array **x** has constant value.

NE_TOO_MANY
Too many iterations(⟨*value*⟩).
The iterative procedure to find the co-variance matrix C, has failed to converge in **max_iter** iterations.

NE_C_ITER_UNSTABLE
The iterative procedure to find C has become unstable. This may happen if the value of **eps** is too large.

NE_NOT_APPEND_FILE
Cannot open file ⟨*string*⟩ for appending.

NE_NOT_CLOSE_FILE
Cannot close file ⟨*string*⟩.

NE_ALLOC_FAIL
Memory allocation failed.

6. Further Comments

The existence of A, and hence c, will depend upon the function u, (see Marazzi (1987)), 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.

6.1. Accuracy

On successful exit the accuracy of the results is related to the value of **tol**, see Section 4. At an iteration let

(i) $d1$ = the maximum value of the absolute relative change in A
(ii) $d2$ = the maximum absolute change in $u(\|z_i\|_2)$
(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**.

6.2. References

Huber P J (1981) *Robust Statistics.* Wiley.
Marazzi A (1987) Weights for Bounded Influence Regression in ROBETH *Cah Rech Doc IUMSP, No. 3 ROB 3.* Institut Universitaire de Médecine Sociale et Préventive, Lausanne.

7. See Also

nag_robust_m_regsn_estim (g02hac)

8. Example

A sample of 10 observations on three variables is read in and the robust estimate of the covariance matrix is computed assuming 10% gross errors are to be expected. The robust covariance is then printed.

8.1. Program Text

```
/* nag_robust_corr_estim(g02hkc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define NMAX 20
#define MMAX 10

main()
{

  /*  Local variables */

  Integer i, j, k, m, n;


  Integer ifail;
  static NagError fail;

  double x[NMAX][MMAX], theta[MMAX];
  Integer tdx=MMAX;
  Integer max_iter, l1, l2;


  Integer print_iter;


  double eps, cov[15];
  Integer iter;


  double tol;

  Vprintf("g02hkc Example Program Results\n\n");

  /*  Skip heading in data file */

  Vscanf("%*[^\n]\n");
  /*  Read in the dimensions of X */

  Vscanf("%ld %ld %*[^\n]\n", &n, &m);
  if (n <= NMAX && m <= MMAX)
    {

      /*  Read in the x matrix */

      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= m; ++j)
            Vscanf("%lf", &x[i-1][j-1]);
```

```
          Vscanf("%*[^\n]\n");
        }

      /*  Read in value of eps */

      Vscanf("%lf%*[^\n]\n", &eps);

      /*  Set up remaining parameters */

      max_iter = 100;
      tol = 5e-5;

      /*  Set print_iter to positive value for iteration moiteroring */

      print_iter = 1;

      g02hkc(n, m, (double *)x, tdx, eps, cov, theta, max_iter, print_iter, "",
             tol, &iter, NAGERR_DEFAULT);

      Vprintf("\n\ng02hkc required %ld iterations to converge\n\n", iter);
      Vprintf("Covariance matrix\n");
      l2 = 0;
      for (j = 1; j <= m; ++j)
        {
          l1 = l2 + 1;
          l2 += j;
          for (k = l1; k <= l2; ++k)
            {
              Vprintf("%10.3f", cov[k - 1]);
            }
          Vprintf("\n");
        }
      Vprintf("\n\ntheta\n");
      for (j = 1; j <= m; ++j)
        {
          Vprintf("%10.3f\n", theta[j - 1]);
        }
    }
  exit(EXIT_SUCCESS);

}
/*  main */
```

8.2. Program Data

```
g02hkc Example Program Data
10    3                   : n  m
3.4  6.9  12.2              : x1  x2  x3
6.4  2.5  15.1
4.9  5.5  14.2
7.3  1.9  18.2
8.8  3.6  11.7
8.4  1.3  17.9
5.3  3.1  15.0
2.7  8.1   7.7
6.1  3.0  21.9
5.3  2.2  13.9              : end of x1 x2 and x3 values
0.1                       : eps
```

8.3. Program Results

```
g02hkc Example Program Results

                    ** Iteration Monitoring **

Iteration                 1  Max Delta =   2.63000e+00
                I       theta(I)
                1      6.02072e+00
                2      3.27481e+00
                3      1.53918e+01
```

```
Matrix A
     5.17060e-01
     7.58801e-01     5.16165e-01
    -3.45723e-01     4.25001e-01     2.86688e-01

Iteration                2  Max Delta =   1.63000e+00
                I       theta(I)
                1      5.76604e+00
                2      3.65572e+00
                3      1.50902e+01
Matrix A
     5.82402e-01
     7.79151e-01     6.97624e-01
     3.74732e-01     5.78790e-01     2.78116e-01

Iteration                3  Max Delta =   1.37048e-01
                I       theta(I)
                1      5.80050e+00
                2      3.72754e+00
                3      1.51386e+01
Matrix A
     5.61199e-01
     8.09374e-01     8.37443e-01
     3.49125e-02     5.22520e-01     3.60431e-01

Iteration                4  Max Delta =   7.59866e-02
                I       theta(I)
                1      5.85724e+00
                2      3.65115e+00
                3      1.51047e+01
Matrix A
     5.68251e-01
     8.71971e-01     8.52456e-01
     3.55533e-02     5.25302e-01     4.01281e-01

Iteration                5  Max Delta =   6.26079e-02
                I       theta(I)
                1      5.84245e+00
                2      3.66594e+00
                3      1.50632e+01
Matrix A
     5.70691e-01
     9.03128e-01     8.71239e-01
     6.49902e-03     5.20477e-01     4.10239e-01

Iteration                6  Max Delta =   5.10886e-02
                I       theta(I)
                1      5.83395e+00
                2      3.67132e+00
                3      1.50568e+01
Matrix A
     5.73316e-01
     9.20497e-01     8.79817e-01
    -1.23444e-02     5.13190e-01     4.17372e-01

Iteration                7  Max Delta =   3.43378e-02
                I       theta(I)
                1      5.82823e+00
                2      3.67478e+00
                3      1.50500e+01
Matrix A
     5.74728e-01
     9.32175e-01     8.85894e-01
    -2.52887e-02     5.09677e-01     4.22326e-01

Iteration                8  Max Delta =   2.27790e-02
                I       theta(I)
                1      5.82459e+00
                2      3.67710e+00
                3      1.50455e+01
```

```
Matrix A
     5.75691e-01
     9.39794e-01     8.89802e-01
    -3.39411e-02     5.07093e-01     4.25709e-01

Iteration                 9  Max Delta =   1.51109e-02
                I       theta(I)
                1      5.82221e+00
                2      3.67857e+00
                3      1.50426e+01
Matrix A
     5.76293e-01
     9.44749e-01     8.92356e-01
    -3.96891e-02     5.05477e-01     4.27997e-01

Iteration                10  Max Delta =   9.95835e-03
                I       theta(I)
                1      5.82068e+00
                2      3.67953e+00
                3      1.50406e+01
Matrix A
     5.76686e-01
     9.47985e-01     8.94020e-01
    -4.34819e-02     5.04415e-01     4.29523e-01

Iteration                11  Max Delta =   6.54850e-03
                I       theta(I)
                1      5.81968e+00
                2      3.68015e+00
                3      1.50393e+01
Matrix A
     5.76939e-01
     9.50095e-01     8.95107e-01
    -4.59773e-02     5.03726e-01     4.30535e-01

Iteration                12  Max Delta =   4.29628e-03
                I       theta(I)
                1      5.81903e+00
                2      3.68055e+00
                3      1.50385e+01
Matrix A
     5.77104e-01
     9.51473e-01     8.95816e-01
    -4.76148e-02     5.03277e-01     4.31202e-01

Iteration                13  Max Delta =   2.81516e-03
                I       theta(I)
                1      5.81860e+00
                2      3.68081e+00
                3      1.50379e+01
Matrix A
     5.77211e-01
     9.52373e-01     8.96279e-01
    -4.86880e-02     5.02984e-01     4.31641e-01

Iteration                14  Max Delta =   1.84286e-03
                I       theta(I)
                1      5.81833e+00
                2      3.68098e+00
                3      1.50376e+01
Matrix A
     5.77281e-01
     9.52961e-01     8.96581e-01
    -4.93906e-02     5.02793e-01     4.31928e-01

Iteration                15  Max Delta =   1.20571e-03
                I       theta(I)
                1      5.81815e+00
                2      3.68109e+00
                3      1.50374e+01
```

```
Matrix A
     5.77327e-01
     9.53345e-01     8.96779e-01
    -4.98504e-02     5.02668e-01     4.32117e-01

Iteration                16  Max Delta =   7.88521e-04
                I       theta(I)
                1      5.81803e+00
                2      3.68116e+00
                3      1.50372e+01
Matrix A
     5.77356e-01
     9.53596e-01     8.96908e-01
    -5.01511e-02     5.02587e-01     4.32241e-01

Iteration                17  Max Delta =   5.15564e-04
                I       theta(I)
                1      5.81795e+00
                2      3.68121e+00
                3      1.50371e+01
Matrix A
     5.77376e-01
     9.53760e-01     8.96993e-01
    -5.03477e-02     5.02534e-01     4.32321e-01

Iteration                18  Max Delta =   3.37038e-04
                I       theta(I)
                1      5.81790e+00
                2      3.68124e+00
                3      1.50370e+01
Matrix A
     5.77389e-01
     9.53867e-01     8.97048e-01
    -5.04762e-02     5.02499e-01     4.32374e-01

Iteration                19  Max Delta =   2.20309e-04
                I       theta(I)
                1      5.81787e+00
                2      3.68126e+00
                3      1.50370e+01
Matrix A
     5.77397e-01
     9.53937e-01     8.97084e-01
    -5.05602e-02     5.02476e-01     4.32409e-01

Iteration                20  Max Delta =   1.43997e-04
                I       theta(I)
                1      5.81785e+00
                2      3.68127e+00
                3      1.50370e+01
Matrix A
     5.77402e-01
     9.53983e-01     8.97107e-01
    -5.06152e-02     5.02461e-01     4.32431e-01

Iteration                21  Max Delta =   9.41152e-05
                I       theta(I)
                1      5.81783e+00
                2      3.68128e+00
                3      1.50369e+01
Matrix A
     5.77406e-01
     9.54013e-01     8.97123e-01
    -5.06510e-02     5.02452e-01     4.32446e-01

Iteration                22  Max Delta =   6.15109e-05
                I       theta(I)
                1      5.81782e+00
                2      3.68129e+00
                3      1.50369e+01
```

```
Matrix A
     5.77408e-01
     9.54033e-01     8.97133e-01
    -5.06745e-02     5.02445e-01     4.32456e-01


g02hkc required 23 iterations to converge

Covariance matrix
     3.461
    -3.681     5.348
     4.682    -6.645    14.439


theta
     5.818
     3.681
    15.037
```