NAG Library Function Document
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, const double x[],
     Integer tdx, double eps, double cov[], double theta[], Integer max_iter,
     Integer print_iter, const 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:

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

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

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

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 (g02hkc) uses weight functions:

$$\begin{aligned} u(t) &= \tfrac{a_u}{t^2}, && \text{if } t < a_u^2 \\ u(t) &= 1, && \text{if } a_u^2 \leq t \leq b_u^2 \\ u(t) &= \tfrac{b_u}{t^2}, && \text{if } t > b_u^2 \end{aligned}$$

and

$$\begin{aligned} w(t) &= 1, && \text{if } t \leq c_w \\ w(t) &= \tfrac{c_w}{t}, && \text{if } t > c_w \end{aligned}$$

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 $j = 1, 2, \ldots, m$, 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 (g02hkc) is based on routines in ROBETH, (see Marazzi (1987)).

4 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

5 Arguments

1: **n** – Integer *Input*

On entry: the number of observations, n.

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

2: **m** – Integer *Input*

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

Constraint: $1 \leq \mathbf{m} \leq \mathbf{n}$.

3: **x**[**n** × **tdx**] – const double *Input*

On entry: $\mathbf{x}[(i-1) \times \mathbf{tdx} + j - 1]$ must contain the ith observation for the jth variable, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

4: **tdx** – Integer *Input*

On entry: the stride separating matrix column elements in the array **x**.

Constraint: $\mathbf{tdx} \geq \mathbf{m}$.

5: **eps** – double *Input*

On entry: the expected fraction of gross errors expected in the sample, ϵ.

Constraint: $0.0 \leq \mathbf{eps} < 1.0$.

6: **cov**[**m** × (**m** + **1**)/**2**] – double *Output*

On exit: 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 $j = i, \ldots, \mathbf{m} - 1$.

7: **theta**[**m**] – double *Output*

On exit: the robust estimate of the location arguments θ_j, for $j = 1, 2, \ldots, m$.

8: **max_iter** – Integer *Input*

On entry: the maximum number of iterations that will be used during the calculation of the covariance matrix.

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

Constraint: **max_iter** > 0.

9: **print_iter** – Integer *Input*

On entry: indicates if the printing of information on the iterations is required and the rate at which printing is produced.

print_iter ≤ 0
No iteration monitoring is printed.

print_iter > 0
The value of A, θ and δ (see Section 9) will be printed at the first and every **print_iter** iterations.

10: **outfile** – const char * *Input*

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

11: **tol** – double *Input*

On entry: the relative precision for the final estimates of the covariance matrix.

Constraint: **tol** > 0.0.

12: **iter** – Integer * *Output*

On exit: the number of iterations performed.

13: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_GT

On entry, **m** $= \langle value \rangle$ while **n** $= \langle value \rangle$. These arguments must satisfy **m** $\leq$ **n**.

NE_2_INT_ARG_LT

On entry, **tdx** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdx** $\geq$ **m**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

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_CONST_COL

On entry, column $\langle value \rangle$ of array **x** has constant value.

NE_INT_ARG_LE

On entry, **max_iter** must not be less than or equal to 0: **max_iter** $= \langle value \rangle$.

NE_INT_ARG_LT

On entry, **m** $= \langle value \rangle$.
Constraint: **m** ≥ 1.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 2.

NE_NOT_APPEND_FILE

Cannot open file $\langle string \rangle$ for appending.

NE_NOT_CLOSE_FILE

Cannot close file $\langle string \rangle$.

NE_REAL_ARG_GE

On entry, **eps** must be not be greater than or equal to 1.0: **eps** $= \langle value \rangle$.

NE_REAL_ARG_LE

On entry, **tol** must not be less than or equal to 0.0: **tol** $= \langle value \rangle$.

NE_REAL_ARG_LT

On entry, **eps** must not be less than 0.0: **eps** $= \langle value \rangle$.

NE_TOO_MANY

Too many iterations($\langle value \rangle$).
The iterative procedure to find the co-variance matrix C, has failed to converge in **max_iter** iterations.

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 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**.

8 Parallelism and Performance

Not applicable.

9 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.

10 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.

10.1 Program Text

```
/* nag_robust_corr_estim (g02hkc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define X(I, J) x[(I-1)*tdx + J-1]
int main(void)
{
  Integer  exit_status = 0, i, iter, j, k, l1, l2, m, max_iter, n, print_iter;
  Integer  tdx;
  NagError fail;
  double   *cov = 0, eps, *theta = 0, tol, *x = 0;

  INIT_FAIL(fail);

  printf("nag_robust_corr_estim (g02hkc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n]\n");

  /* Read in the dimensions of X */
  scanf("%ld %ld %*[^\n]\n", &n, &m);

  if (n > 1 && (m >= 1 && m <= n))
    {
      if (!(x = NAG_ALLOC((n)*(m), double)) ||
          !(theta = NAG_ALLOC(m, double)) ||
          !(cov = NAG_ALLOC(m*(m+1)/2, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdx = m;
    }
  else
    {
      printf("Invalid n or m.\n");
      exit_status = 1;
      return exit_status;
    }
  /* Read in the x matrix */
  for (i = 1; i <= n; ++i)
    {
      for (j = 1; j <= m; ++j)
        scanf("%lf", &X(i, j));
      scanf("%*[^\n]\n");
    }

  /* Read in value of eps */
  scanf("%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;
  /* nag_robust_corr_estim (g02hkc).
   * Robust estimation of a correlation matrix, Huber's weight
```

```
   * function
   */
  fflush(stdout);
  nag_robust_corr_estim(n, m, x, tdx, eps, cov, theta, max_iter, print_iter,
                        0, tol, &iter, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_robust_corr_estim (g02hkc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf(
          "\n\nnag_robust_corr_estim (g02hkc) required %ld iterations "
          "to converge\n\n", iter);
  printf("Covariance matrix\n");
  l2 = 0;
  for (j = 1; j <= m; ++j)
    {
      l1 = l2 + 1;
      l2 += j;
      for (k = l1; k <= l2; ++k)
        printf("%10.3f", cov[k - 1]);
      printf("\n");
    }
  printf("\n\ntheta\n");
  for (j = 1; j <= m; ++j)
    printf("%10.3f\n", theta[j - 1]);

 END:
  NAG_FREE(x);
  NAG_FREE(theta);
  NAG_FREE(cov);
  return exit_status;
}
```

10.2 Program Data

```
nag_robust_corr_estim (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
```

10.3 Program Results

```
nag_robust_corr_estim (g02hkc) Example Program Results

                    ** Iteration Monitoring **

Iteration                1  Max Delta =   2.62947e+00
                I       theta(I)
                1      6.02072e+00
                2      3.27481e+00
                3      1.53918e+01
Matrix A
     5.17053e-01
     7.58801e-01     5.16157e-01
    -3.45725e-01     4.25003e-01     2.86687e-01

Iteration                2  Max Delta =   1.62947e+00
```

```
                I       theta(I)
                1      5.76604e+00
                2      3.65572e+00
                3      1.50902e+01
Matrix A
     5.82402e-01
     7.79144e-01     6.97613e-01
     3.74744e-01     5.78797e-01     2.78112e-01

Iteration                3  Max Delta =   1.37053e-01
                I       theta(I)
                1      5.80050e+00
                2      3.72754e+00
                3      1.51386e+01
Matrix A
     5.61198e-01
     8.09363e-01     8.37438e-01
     3.49166e-02     5.22524e-01     3.60428e-01

Iteration                4  Max Delta =   7.59814e-02
                I       theta(I)
                1      5.85724e+00
                2      3.65115e+00
                3      1.51047e+01
Matrix A
     5.68250e-01
     8.71962e-01     8.52451e-01
     3.55609e-02     5.25305e-01     4.01279e-01

Iteration                5  Max Delta =   6.26102e-02
                I       theta(I)
                1      5.84245e+00
                2      3.66594e+00
                3      1.50632e+01
Matrix A
     5.70689e-01
     9.03118e-01     8.71233e-01
     6.50776e-03     5.20480e-01     4.10236e-01

Iteration                6  Max Delta =   5.10803e-02
                I       theta(I)
                1      5.83395e+00
                2      3.67132e+00
                3      1.50568e+01
Matrix A
     5.73314e-01
     9.20487e-01     8.79812e-01
    -1.23337e-02     5.13193e-01     4.17368e-01

Iteration                7  Max Delta =   3.43340e-02
                I       theta(I)
                1      5.82823e+00
                2      3.67478e+00
                3      1.50500e+01
Matrix A
     5.74727e-01
     9.32164e-01     8.85888e-01
    -2.52769e-02     5.09680e-01     4.22320e-01

Iteration                8  Max Delta =   2.27767e-02
                I       theta(I)
                1      5.82460e+00
                2      3.67710e+00
                3      1.50455e+01
Matrix A
     5.75689e-01
     9.39782e-01     8.89795e-01
    -3.39284e-02     5.07095e-01     4.25703e-01

Iteration                9  Max Delta =   1.51098e-02
                I       theta(I)
```

```
                1      5.82222e+00
                2      3.67857e+00
                3      1.50426e+01
Matrix A
     5.76292e-01
     9.44737e-01     8.92349e-01
    -3.96758e-02     5.05480e-01     4.27991e-01

Iteration               10  Max Delta =   9.95773e-03
                I       theta(I)
                1      5.82068e+00
                2      3.67953e+00
                3      1.50406e+01
Matrix A
     5.76684e-01
     9.47973e-01     8.94014e-01
    -4.34683e-02     5.04418e-01     4.29517e-01

Iteration               11  Max Delta =   6.54817e-03
                I       theta(I)
                1      5.81968e+00
                2      3.68014e+00
                3      1.50393e+01
Matrix A
     5.76938e-01
     9.50083e-01     8.95100e-01
    -4.59634e-02     5.03729e-01     4.30528e-01

Iteration               12  Max Delta =   4.29611e-03
                I       theta(I)
                1      5.81903e+00
                2      3.68055e+00
                3      1.50385e+01
Matrix A
     5.77102e-01
     9.51461e-01     8.95809e-01
    -4.76008e-02     5.03280e-01     4.31196e-01

Iteration               13  Max Delta =   2.81507e-03
                I       theta(I)
                1      5.81861e+00
                2      3.68081e+00
                3      1.50380e+01
Matrix A
     5.77209e-01
     9.52361e-01     8.96272e-01
    -4.86739e-02     5.02988e-01     4.31634e-01

Iteration               14  Max Delta =   1.84282e-03
                I       theta(I)
                1      5.81833e+00
                2      3.68098e+00
                3      1.50376e+01
Matrix A
     5.77279e-01
     9.52949e-01     8.96575e-01
    -4.93765e-02     5.02796e-01     4.31922e-01

Iteration               15  Max Delta =   1.20569e-03
                I       theta(I)
                1      5.81815e+00
                2      3.68109e+00
                3      1.50374e+01
Matrix A
     5.77325e-01
     9.53333e-01     8.96772e-01
    -4.98362e-02     5.02672e-01     4.32111e-01

Iteration               16  Max Delta =   7.88514e-04
                I       theta(I)
                1      5.81803e+00
```

```
               2      3.68116e+00
               3      1.50372e+01
Matrix A
     5.77355e-01
     9.53584e-01     8.96901e-01
    -5.01368e-02     5.02590e-01     4.32234e-01

Iteration               17  Max Delta =   5.15563e-04
               I       theta(I)
               1      5.81795e+00
               2      3.68121e+00
               3      1.50371e+01
Matrix A
     5.77374e-01
     9.53748e-01     8.96986e-01
    -5.03334e-02     5.02537e-01     4.32315e-01

Iteration               18  Max Delta =   3.37039e-04
               I       theta(I)
               1      5.81790e+00
               2      3.68124e+00
               3      1.50370e+01
Matrix A
     5.77387e-01
     9.53855e-01     8.97041e-01
    -5.04620e-02     5.02502e-01     4.32368e-01

Iteration               19  Max Delta =   2.20311e-04
               I       theta(I)
               1      5.81787e+00
               2      3.68126e+00
               3      1.50370e+01
Matrix A
     5.77395e-01
     9.53925e-01     8.97077e-01
    -5.05460e-02     5.02479e-01     4.32402e-01

Iteration               20  Max Delta =   1.44000e-04
               I       theta(I)
               1      5.81785e+00
               2      3.68127e+00
               3      1.50370e+01
Matrix A
     5.77400e-01
     9.53971e-01     8.97101e-01
    -5.06009e-02     5.02465e-01     4.32425e-01

Iteration               21  Max Delta =   9.41173e-05
               I       theta(I)
               1      5.81784e+00
               2      3.68128e+00
               3      1.50369e+01
Matrix A
     5.77404e-01
     9.54001e-01     8.97116e-01
    -5.06368e-02     5.02455e-01     4.32440e-01

Iteration               22  Max Delta =   6.15127e-05
               I       theta(I)
               1      5.81783e+00
               2      3.68129e+00
               3      1.50369e+01
Matrix A
     5.77406e-01
     9.54020e-01     8.97126e-01
    -5.06603e-02     5.02448e-01     4.32449e-01


nag_robust_corr_estim (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
```