

## 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,  $\theta$ , are given by:

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

where  $\tau^2$  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  $i$ th 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:

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

and

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

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

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,  $\epsilon$  (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,  $\tau^2$ , is calculated, (see Huber (1981) and Marazzi (1987)).

Initial estimates of  $\theta_j$ , for  $j = 1, 2, \dots, m$ , are given by the median of the  $j$ th 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:*  $n > 1$ .
- 2: **m** – Integer *Input*  
*On entry:* the number of columns of the matrix  $X$ , i.e., number of independent variables,  $m$ .  
*Constraint:*  $1 \leq m \leq n$ .
- 3: **x[n × tdx]** – const double *Input*  
*On entry:* **x**[( $i - 1$ ) × **tdx** +  $j - 1$ ] must contain the  $i$ th observation for the  $j$ th variable, for  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, m$ .
- 4: **tdx** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **x**.  
*Constraint:* **tdx** ≥ **m**.
- 5: **eps** – double *Input*  
*On entry:* the expected fraction of gross errors expected in the sample,  $\epsilon$ .  
*Constraint:*  $0.0 \leq \text{eps} < 1.0$ .
- 6: **cov[m × (m + 1)/2]** – double *Output*  
*On exit:* the  $m \times (m + 1)/2$  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 **cov**[ $j(j + 1)/2 + i$ ], for  $i = 0, 1, \dots, m - 1$  and  $j = i, \dots, m - 1$ .

- 7: **theta[m]** – double *Output*  
*On exit:* the robust estimate of the location arguments  $\theta_j$ , for  $j = 1, 2, \dots, m$ .
- 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  $A$ ,  $\theta$  and  $\delta$  (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  $\mathbf{m} \leq \mathbf{n}$ .

### NE\_2\_INT\_ARG\_LT

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

### 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  $\mathbf{x}$  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**  $\geq 1$ .

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 2$ .

**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 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  $\theta_j$

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

**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 2014 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");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[\n]\n");
#else
  scanf("%*[\n]\n");
#endif

  /* Read in the dimensions of X */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %"NAG_IFMT" %*[\n]\n", &n, &m);
#else
  scanf("%"NAG_IFMT" %"NAG_IFMT" %*[\n]\n", &n, &m);
#endif

  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)

```

```

#ifdef _WIN32
    scanf_s("%lf", &X(i, j));
#else
    scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
    scanf_s("%*[^\\n]\\n");
#else
    scanf("%*[^\\n]\\n");
#endif
    }

    /* Read in value of eps */
#ifdef _WIN32
    scanf_s("%lf%*[^\\n]\\n", &eps);
#else
    scanf("%lf%*[^\\n]\\n", &eps);
#endif

    /* Set up remaining parameters */
    max_iter = 100;
    tol = 5e-5;

    /* Set print_iter to a positive value for iteration monitoring */
    print_iter = 0;
    /* 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("\\nnag_robust_corr_estim (g02hkc) required %"NAG_IFMT" 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("\\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
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
```

---