NAG Library Function Document

nag_robust_m_estim_1var (g07dbc)

1 Purpose

nag_robust_m_estim_1var (g07dbc) computes an M-estimate of location with (optional) simultaneous estimation of the scale using Huber's algorithm.

2 Specification

```
#include <nag.h>
#include <nagg07.h>
```

```
void nag_robust_m_estim_1var (Nag_SigmaSimulEst sigma_est, Integer n,
     const double x[], Nag_PsiFun psifun, double c, double h1, double h2,
     double h3, double dchi, double *theta, double *sigma, Integer maxit,
     double tol, double rs[], Integer *nit, double sorted_x[],
     NagError *fail)
```

3 Description

The data consists of a sample of size n, denoted by $x_1, x_2, \ldots, x_n$, drawn from a random variable X.

The x_i are assumed to be independent with an unknown distribution function of the form

$$F((x_i - \theta)/\sigma)$$

where θ is a location argument, and σ is a scale argument. M-estimators of θ and σ are given by the solution to the following system of equations:

$$\sum_{i=1}^{n} \psi\left(\left(x_i - \hat{\theta}\right)/\hat{\sigma}\right) = 0 \qquad (1)$$

$$\sum_{i=1}^{n} \chi\left(\left(x_i - \hat{\theta}\right)/\hat{\sigma}\right) = (n-1)\beta \qquad (2)$$

where ψ and χ are given functions, and β is a constant, such that $\hat{\sigma}$ is an unbiased estimator when x_i, for $i = 1, 2, \ldots, n$, has a normal distribution. Optionally, the second equation can be omitted and the first equation is solved for $\hat{\theta}$ using an assigned value of $\sigma = \sigma_c$.

The values of $\psi\left(\frac{x_i - \hat{\theta}}{\hat{\sigma}}\right)\hat{\sigma}$ are known as the Winsorized residuals.

The following functions are available for ψ and χ in nag_robust_m_estim_1var (g07dbc):

(a) **Null Weights**

$$\psi(t) = t \qquad \chi(t) = \frac{t^2}{2}$$

Use of these null functions leads to the mean and standard deviation of the data.

(b) **Huber's Function**

$$\psi(t) = \max(-c, \min(c, t)) \qquad \begin{array}{l} \chi(t) = \frac{\|t\|^2}{2} \|t\| \leq d \\ \chi(t) = \frac{d^2}{2} \|t\| > d \end{array}$$

(c) **Hampel's Piecewise Linear Function**

$$
\begin{array}{llll}
\psi_{h_1,h_2,h_3}(t) & = -\psi_{h_1,h_2,h_3}(-t) & & \\
 & = t & 0 \le t \le h_1 & \chi(t) = \frac{|t|^2}{2} |t| \le d \\
 & = h_1 & h_1 \le t \le h_2 & \\
 & = h_1(h_3 - t)/(h_3 - h_2) & h_2 \le t \le h_3 & \chi(t) = \frac{d^2}{2} |t| > d \\
 & = 0 & t > h_3 &
\end{array}
$$

(d) **Andrew's Sine Wave Function**

$$
\begin{array}{llll}
\psi(t) & = \sin t & -\pi \le t \le \pi & \chi(t) = \frac{|t|^2}{2} |t| \le d \\
 & = 0 & \text{otherwise} & \chi(t) = \frac{d^2}{2} |t| > d
\end{array}
$$

(e) **Tukey's Bi-weight**

$$
\begin{array}{llll}
\psi(t) & = t(1 - t^2)^2 & |t| \le 1 & \chi(t) = \frac{|t|^2}{2} |t| \le d \\
 & = 0 & \text{otherwise} & \chi(t) = \frac{d^2}{2} |t| > d
\end{array}
$$

where c, h_1, h_2, h_3 and d are constants.

Equations (1) and (2) are solved by a simple iterative procedure suggested by Huber:

$$
\hat{\sigma}_k = \sqrt{\frac{1}{\beta(n-1)} \left(\sum_{i=1}^{n} \chi\left(\frac{x_i - \hat{\theta}_{k-1}}{\hat{\sigma}_{k-1}} \right) \right) \hat{\sigma}_{k-1}^2}
$$

and

$$
\hat{\theta}_k = \hat{\theta}_{k-1} + \frac{1}{n} \sum_{i=1}^{n} \psi\left(\frac{x_i - \hat{\theta}_{k-1}}{\hat{\sigma}_k} \right) \hat{\sigma}_k
$$

or

$$
\hat{\sigma}_k = \sigma_c, \text{ if } \sigma \text{ is fixed.}
$$

The initial values for $\hat{\theta}$ and $\hat{\sigma}$ may either be user-supplied or calculated within nag_robust_m_estim_1var (g07dbc) as the sample median and an estimate of σ based on the median absolute deviation respectively.

nag_robust_m_estim_1var (g07dbc) is based upon subroutine LYHALG within the ROBETH library, see Marazzi (1987).

4 References

Hampel F R, Ronchetti E M, Rousseeuw P J and Stahel W A (1986) *Robust Statistics. The Approach Based on Influence Functions* Wiley

Huber P J (1981) *Robust Statistics* Wiley

Marazzi A (1987) Subroutines for robust estimation of location and scale in ROBETH *Cah. Rech. Doc. IUMSP, No. 3 ROB 1* Institut Universitaire de Médecine Sociale et Préventive, Lausanne

5 Arguments

1: **sigma_est** – Nag_SigmaSimulEst *Input*

On entry: the value assigned to **sigma_est** determines whether $\hat{\sigma}$ is to be simultaneously estimated.

sigma_est = Nag_SigmaBypas
The estimation of $\hat{\sigma}$ is bypassed and **sigma** is set equal to σ_c;

sigma_est = Nag_SigmaSimul
$\hat{\sigma}$ is estimated simultaneously.

Constraint: **sigma_est** = Nag_SigmaBypas or Nag_SigmaSimul.

2: **n** – Integer *Input*

On entry: the number of observations, n.

Constraint: **n** > 1.

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

On entry: the vector of observations, $x_1, x_2, \ldots, x_n$.

4: **psifun** – Nag_PsiFun *Input*

On entry: which ψ function is to be used.

psifun = Nag_Lsq

$$\psi(t) = t.$$

psifun = Nag_HuberFun
Huber's function.

psifun = Nag_HampelFun
Hampel's piecewise linear function.

psifun = Nag_AndrewFun
Andrew's sine wave.

psifun = Nag_TukeyFun
Tukey's bi-weight.

Constraint: **psifun** = Nag_Lsq, Nag_HuberFun, Nag_HampelFun, Nag_AndrewFun or Nag_TukeyFun.

5: **c** – double *Input*

On entry: must specify the argument, c, of Huber's ψ function, if **psifun** = Nag_HuberFun. **c** is not referenced if **psifun** $\neq$ Nag_HuberFun.

Constraint: **c** > 0.0 if **psifun** = Nag_HuberFun.

6: **h1** – double *Input*
7: **h2** – double *Input*
8: **h3** – double *Input*

On entry: **h1**, **h2**, and **h3** must specify the arguments h_1, h_2, and h_3, of Hampel's piecewise linear ψ function, if **psifun** = Nag_HampelFun. **h1**, **h2**, and **h3** are not referenced if **psifun** $\neq$ Nag_HampelFun.

Constraint: $0 \leq$ **h1** $\leq$ **h2** $\leq$ **h3** and **h3** > 0.0 if **psifun** = Nag_HampelFun.

9: **dchi** – double *Input*

On entry: the argument, d, of the χ function. **dchi** is not referenced if **psifun** = Nag_Lsq.

Constraint: **dchi** > 0.0 if **psifun** $\neq$ Nag_Lsq.

10: **theta** – double * *Input/Output*

On entry: if **sigma** > 0 then **theta** must be set to the required starting value of the estimation of the location argument $\hat{\theta}$. A reasonable initial value for $\hat{\theta}$ will often be the sample mean or median.

On exit: the M-estimate of the location argument, $\hat{\theta}$.

11: **sigma** – double * *Input/Output*

The role of **sigma** depends on the value assigned to **sigma_est** (see above) as follows.

If **sigma_est** = Nag_SigmaSimul:

On entry: **sigma** must be assigned a value which determines the values of the starting points for the calculations of $\hat{\theta}$ and $\hat{\sigma}$. If **sigma** ≤ 0.0 then nag_robust_m_estim_1var (g07dbc) will determine the starting points of $\hat{\theta}$ and $\hat{\sigma}$. Otherwise the value assigned to **sigma** will be taken as the starting point for $\hat{\sigma}$, and **theta** must be assigned a value before entry, see above.

If **sigma_est** = Nag_SigmaBypas:

On entry: **sigma** must be assigned a value which determines the value of σ_c, which is held fixed during the iterations, and the starting value for the calculation of $\hat{\theta}$. If **sigma** ≤ 0, then nag_robust_m_estim_1var (g07dbc) will determine the value of σ_c as the median absolute deviation adjusted to reduce bias (see G07DAF) and the starting point for $\hat{\theta}$. Otherwise, the value assigned to **sigma** will be taken as the value of σ_c and **theta** must be assigned a relevant value before entry, see above.

On exit: **sigma** contains the M – estimate of the scale argument, $\hat{\sigma}$, if **sigma_est** = Nag_SigmaSimul on entry, otherwise **sigma** will contain the initial fixed value σ_c.

12: **maxit** – Integer *Input*

On entry: the maximum number of iterations that should be used during the estimation.

Suggested value: p **maxit** = 50.

Constraint: **maxit** > 0.

13: **tol** – double *Input*

On entry: the relative precision for the final estimates. Convergence is assumed when the increments for **theta**, and **sigma** are less than **tol** $\times \max(1.0, \sigma_{k-1})$.

Constraint: **tol** > 0.0.

14: **rs**[**n**] – double *Output*

On exit: the Winsorized residuals.

15: **nit** – Integer * *Output*

On exit: the number of iterations that were used during the estimation.

16: **sorted_x**[**n**] – double *Output*

On exit: if **sigma** ≤ 0.0 on entry, **sorted_x** will contain the n observations in ascending order.

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

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

6 Error Indicators and Warnings

NE_2_REAL_ENUM_ARG_CONS

On entry, **h1** = $\langle value \rangle$, **h2** = $\langle value \rangle$ and **psifun** = $\langle value \rangle$. These arguments must satisfy **h1** $\leq$ **h2**, **psifun** = Nag_HampelFun.

On entry, **h1** = $\langle value \rangle$, **h3** = $\langle value \rangle$ and **psifun** = $\langle value \rangle$. These arguments must satisfy **h1** $\leq$ **h3**, **psifun** = Nag_HampelFun.

On entry, **h2** = $\langle value \rangle$, **h3** = $\langle value \rangle$ and **psifun** = $\langle value \rangle$. These arguments must satisfy **h2** $\leq$ **h3**, **psifun** = Nag_HampelFun.

NE_3_REAL_ENUM_ARG_CONS

On entry, **h1** $= \langle value \rangle$, **h2** $= \langle value \rangle$, **h3** $= \langle value \rangle$, **psifun** $= \langle value \rangle$. These arguments must satisfy **h1** $=$ **h2**=**h3** $\neq 0.0$, **psifun** $=$ Nag_HampelFun.

NE_ALL_ELEMENTS_EQUAL

On entry, all the values in the array **x** must not be equal.

NE_BAD_PARAM

On entry, argument **psifun** had an illegal value.

On entry, argument **sigma_est** had an illegal value.

NE_ESTIM_SIGMA_ZERO

The estimated value of **sigma** was ≤ 0.0 during an iteration.

NE_INT_ARG_LE

On entry, **maxit** $= \langle value \rangle$.
Constraint: **maxit** > 0.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** > 1.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

NE_REAL_ARG_LE

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

NE_REAL_ENUM_ARG_CONS

On entry, **c** $= \langle value \rangle$, **psifun** $= \langle value \rangle$. These arguments must satisfy **c** > 0.0,
psifun $=$ Nag_HuberFun.

On entry, **dchi** $= \langle value \rangle$, **psifun** $= \langle value \rangle$. These arguments must satisfy **dchi** > 0.0,
psifun $\neq$ Nag_Lsq.

On entry, **h1** $= \langle value \rangle$, **psifun** $= \langle value \rangle$. These arguments must satisfy **h1** ≥ 0.0,
psifun $=$ Nag_HampelFun.

NE_TOO_MANY

Too many iterations ($\langle value \rangle$).

NE_WINS_RES_ZERO

The Winsorized residuals are zero.

On completion of the iterations, the Winsorized residuals were all zero. This may occur when using the **sigma_est** $=$ Nag_SigmaBypas option with a redescending ψ function, i.e., Hampel's piecewise linear function, Andrew's sine wave, and Tukey's biweight.

If the given value of σ is too small, then the standardized residuals $\frac{x_i - \hat{\theta}_k}{\sigma_c}$, will be large and all the residuals may fall into the region for which $\psi(t) = 0$. This may incorrectly terminate the iterations thus making **theta** and **sigma** invalid.

Re-enter the function with a larger value of σ_c or with **sigma_est** $=$ Nag_SigmaSimul.

7 Accuracy

On successful exit the accuracy of the results is related to the value of TOL, see Section 4.

8 Parallelism and Performance

Not applicable.

9 Further Comments

When you supply the initial values, care has to be taken over the choice of the initial value of σ. If too small a value of σ is chosen then initial values of the standardized residuals $\frac{x_i - \hat{\theta}_k}{\sigma}$ will be large. If the redescending ψ functions are used, i.e., Hampel's piecewise linear function, Andrew's sine wave, or Tukey's bi-weight, then these large values of the standardized residuals are Winsorized as zero. If a sufficient number of the residuals fall into this category then a false solution may be returned, see Hampel *et al.* (1986).

10 Example

The following program reads in a set of data consisting of eleven observations of a variable X.

For this example, Hampels's Piecewise Linear Function is used (**psifun** = Nag_HampelFun), values for h_1, h_2 and h_3 along with d for the χ function, being read from the data file.

Using the following starting values various estimates of θ and σ are calculated and printed along with the number of iterations used:

(a) nag_robust_m_estim_1var (g07dbc) determines the starting values, σ is estimated simultaneously.

(b) You supply the starting values, σ is estimated simultaneously.

(c) nag_robust_m_estim_1var (g07dbc) determines the starting values, σ is fixed.

(d) You supply the starting values, σ is fixed.

10.1 Program Text

```
/* nag_robust_m_estim_1var (g07dbc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 * Mark 6 revised, 2000.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <nag_string.h>
#include <stdio.h>
#include <nagg07.h>

int main(void)
{
  Integer           exit_status = 0, i, maxit, n, nit;
  Nag_SigmaSimulEst sigma_est;
  char              sigma_est_str[40];
  double            c, dchi, h1, h2, h3, *rs = 0, sigma, sigsav, *sorted_x = 0,
                    thesav, theta;
  double            tol, *x = 0;
  NagError          fail;

  INIT_FAIL(fail);

  printf(
```

```
          "nag_robust_m_estim_1var (g07dbc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]\n");
#else
  scanf("%*[^\n]\n");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %*[^\n]\n", &n);
#else
  scanf("%"NAG_IFMT" %*[^\n]\n", &n);
#endif
  if (n > 1)
    {
      if (!(x = NAG_ALLOC(n, double)) ||
          !(rs = NAG_ALLOC(n, double)) ||
          !(sorted_x = NAG_ALLOC(n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    scanf_s("%lf", &x[i - 1]);
#else
    scanf("%lf", &x[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]\n");
#else
  scanf("%*[^\n]\n");
#endif
#ifdef _WIN32
  scanf_s("%lf %lf %lf %lf %"NAG_IFMT" %*[^\n]\n", &h1, &h2, &h3, &dchi,
          &maxit);
#else
  scanf("%lf %lf %lf %lf %"NAG_IFMT" %*[^\n]\n", &h1, &h2, &h3, &dchi,
          &maxit);
#endif
  printf("%25sInput parameters     Output parameters\n", "");
  printf(
          "   sigma_est         sigma    theta   tol       sigma   theta\n\n");
#ifdef _WIN32
  while ((scanf_s("%39s %lf %lf %lf%*[^\n]", sigma_est_str,
                  _countof(sigma_est_str), &sigma, &theta, &tol)) != EOF)
    {
#else
  while ((scanf("%39s %lf %lf %lf%*[^\n]", sigma_est_str, &sigma, &theta,
                &tol)) != EOF)
    {
#endif
      /* nag_enum_name_to_value (x04nac).
       * Converts NAG enum member name to value
       */
      sigma_est = (Nag_SigmaSimulEst) nag_enum_name_to_value(sigma_est_str);
      sigsav = sigma;
      thesav = theta;
      c = 0.0;

      /* nag_robust_m_estim_1var (g07dbc).
       * Robust estimation, M-estimates for location and scale
       * parameters, standard weight functions
```

```
       */
      nag_robust_m_estim_1var(sigma_est, n, x, Nag_HampelFun, c, h1, h2, h3,
                              dchi, &theta, &sigma, maxit, tol, rs, &nit,
                              sorted_x, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_robust_m_estim_1var (g07dbc).\n%s\n",
                 fail.message);
          exit_status = 1;
          goto END;
        }

      printf("%s    %8.4f %8.4f %7.4f %9.4f %8.4f\n", sigma_est_str,
             sigsav, thesav, tol, sigma, theta);
    }

 END:
  NAG_FREE(x);
  NAG_FREE(rs);
  NAG_FREE(sorted_x);

  return exit_status;
}
```

10.2 Program Data

```
nag_robust_m_estim_1var (g07dbc) Example Program Data
11                                                : Number of observations
13.0 11.0 16.0 5.0 3.0 18.0 9.0 8.0 6.0 27.0 7.0 : Observations
1.5  3.0 4.5  1.5  50                             : h1  h2  h3  dchi  maxit
Nag_SigmaSimul     -1.0    0.0   0.0001           : sigma_est  sigma  theta  tol
Nag_SigmaSimul      7.0    2.0   0.0001
Nag_SigmaBypas     -1.0    0.0   0.0001
Nag_SigmaBypas      7.0    2.0   0.0001
```

10.3 Program Results

```
nag_robust_m_estim_1var (g07dbc) Example Program Results

                         Input parameters     Output parameters
   sigma_est          sigma    theta   tol        sigma   theta

Nag_SigmaSimul     -1.0000   0.0000  0.0001    6.3247  10.5487
Nag_SigmaSimul      7.0000   2.0000  0.0001    6.3249  10.5487
Nag_SigmaBypas     -1.0000   0.0000  0.0001    5.9304  10.4896
Nag_SigmaBypas      7.0000   2.0000  0.0001    7.0000  10.6500
```