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,
                 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 parameter, and σ is a scale parameter. 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;

(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**

$$\psi(t) = \max(-c, \min(c, t)) \qquad \chi(t) = \frac{|t|^2}{2} |t| \le d$$

$$\chi(t) = \frac{d^2}{2} |t| > d$$

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

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

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

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

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

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

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

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

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

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

4. Parameters

sigma_est

Input: 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.

n

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

x[n]

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

psifun

Input: 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**.

c

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

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

h1

h2

h3

If **psifun** = **Nag_HampelFun** on entry, **h1**, **h2**, and **h3** must specify the parameters h_1, h_2, and h_3, of Hampel's piecewise linear ψ function. **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**.

dchi

Input: the parameter, d, of the χ function. **dchi** is not referenced if psifun = **Nag_Lsq**.

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

theta

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

Output: the M-estimate of the location parameter, $\hat{\theta}$.

sigma

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

sigma_est = **Nag_SigmaSimul**

Input: **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 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.

sigma_est = **Nag_SigmaBypas**

Input: **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 will determine the value of σ_c as the median absolute deviation adjusted to reduce bias 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.

Output: **sigma** contains the M - estimate of the scale parameter, $\hat{\sigma}$, if **sigma_est** was assigned the value **Nag_SigmaSimul** on entry, otherwise **sigma** will contain the initial fixed value σ_c.

maxit

Input: the maximum number of iterations that should be used during the estimation.
Suggested value: **maxit** = 50.
Constraint: **maxit** > 0.

tol

Input: 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.

rs[n]

Output: the Winsorized residuals.

nit

Output: the number of iterations that were used during the estimation.

sorted_x[n]

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

fail

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

5. Error Indications and Warnings

NE_INT_ARG_LE

On entry, **n** must not be less than or equal to 1: **n** = ⟨*value*⟩.
On entry, **maxit** must not be less than or equal to 0: **maxit** = ⟨*value*⟩.

NE_REAL_ARG_LE

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

NE_BAD_PARAM

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

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

NE_REAL_ENUM_ARG_CONS

On entry, **c** = ⟨*value*⟩, **psifun** = ⟨*value*⟩.
These parameters must satisfy **c>0**, **psifun=Nag_HuberFun**.

On entry, **h1** = ⟨*value*⟩, **psifun** = ⟨*value*⟩.
These parameters must satisfy **h1** ≥ 0, **psifun=Nag_HampelFun**.

On entry, **dchi** = ⟨*value*⟩, **psifun** = ⟨*value*⟩.
These parameters must satisfy **dchi>0**, **psifun** ≠ **Nag_Lsq**.

NE_2_REAL_ENUM_ARG_CONS

On entry, **h1** = ⟨*value*⟩, **h2** = ⟨*value*⟩and **psifun** = ⟨*value*⟩.
These parameters must satisfy **h1** ≤ **h2**, **psifun=Nag_HampelFun**.

On entry, **h1** = ⟨*value*⟩, **h3** = ⟨*value*⟩and **psifun** = ⟨*value*⟩.
These parameters must satisfy **h1** ≤ **h3**, **psifun=Nag_HampelFun**.

On entry, **h2** = ⟨*value*⟩, **h3** = ⟨*value*⟩and **psifun** = ⟨*value*⟩.
These parameters must satisfy **h2** ≤ **h3**, **psifun=Nag_HampelFun**.

NE_3_REAL_ENUM_ARG_CONS

On entry, **h1** = ⟨*value*⟩, **h2** = ⟨*value*⟩, **h3** = ⟨*value*⟩, **psifun** = ⟨*value*⟩.
These parameters must satisfy **h1=h2=h3** ≠ 0.0, **psifun=Nag_HampelFun**.

NE_ALL_ELEMENTS_EQUAL

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

NE_ESTIM_SIGMA_ZERO

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

NE_TOO_MANY

Too many iterations (⟨*value*⟩).

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 standardised 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 routine with a larger value of σ_c or with **sigma_est** = **Nag_SigmaSimul**.

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 consult NAG for assistance.

6. Further Comments

When the user supplies 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 standardised 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 (1986) page 152.

6.1. Accuracy

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

6.2. 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

7. See Also

None.

8. 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 determines the starting values, σ is estimated simultaneously.

(b) The user supplies the starting values, σ is estimated simultaneously.

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

(d) The user supplies the starting values, σ is fixed.

8.1. Program Text

```
/* nag_robust_m_estim_1var(g07dbc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 */

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

#define NMAX 25

main()
{
  double dchi;
  double c;
  double x[NMAX], sigma, theta;
  double h1, h2, h3, rs[NMAX];
  double thesav, sigsav;
  double tol, wrk[NMAX];

  Integer ipsi;
  Integer i;
  Integer n;
  Integer maxit;
  Integer isigma;
  Integer nit;

  char sigma_enum_str[20];

  Nag_SigmaSimulEst sigma_enum;

  Vprintf("g07dbc Example Program Results\n\n");

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

  Vscanf("%ld %*[^\n]\n",&n);
  if (n <= NMAX)
    {
      for (i = 1; i <= n; ++i)
        Vscanf("%lf", &x[i - 1]);
      Vscanf("%*[^\n]\n");
      Vscanf("%lf %lf %lf %lf %ld %*[^\n]\n", &h1,&h2,&h3,&dchi,&maxit);
      Vprintf("%25sInput parameters     Output parameters\n","");
      Vprintf("   sigma_est         sigma    theta   tol       sigma   theta\n\n");
      while ((scanf("%ld %lf %lf %lf%*[^\n]", &isigma,&sigma,&theta,&tol)) != EOF)
        {
          if (isigma == 1)
            {
              sigma_enum = Nag_SigmaSimul;
              strcpy(sigma_enum_str, "Nag_SigmaSimul");
            }
          else if (isigma == 0)
            {
              sigma_enum = Nag_SigmaBypas;
              strcpy(sigma_enum_str, "Nag_SigmaBypas");
            }
          sigsav = sigma;
          thesav = theta;
          c = 0.0;

          g07dbc(sigma_enum, n, x, Nag_HampelFun, c, h1, h2, h3, dchi, &theta,
                 &sigma, maxit, tol, rs, &nit, wrk, NAGERR_DEFAULT);
```

```
          Vprintf("%s    %8.4f %8.4f %7.4f %9.4f %8.4f\n", sigma_enum_str,
                  sigsav, thesav,tol,sigma,theta);
        }
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vprintf("n is out of range: n =%ld\n", n);
      exit(EXIT_SUCCESS);
    }
}
```

8.2. Program Data

```
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
1     -1.0    0.0   0.0001      :isigma  sigma  theta   tol
1      7.0    2.0   0.0001
0     -1.0    0.0   0.0001
0      7.0    2.0   0.0001
```

8.3. Program Results

```
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
```