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

```c
#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 \( \theta \) is a location argument, and \( \sigma \) is a scale argument. M-estimators of \( \theta \) and \( \sigma \) are given by the solution to the following system of equations:

\[
\sum_{i=1}^{n} \psi\left(\frac{x_i - \hat{\theta}}{\hat{\sigma}}\right) = 0 \tag{1}
\]

\[
\sum_{i=1}^{n} \chi\left(\frac{x_i - \hat{\theta}}{\hat{\sigma}}\right) = (n - 1)\beta \tag{2}
\]

where \( \psi \) and \( \chi \) are given functions, and \( \beta \) 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 \( \theta \) 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 \( \psi \) and \( \chi \) in nag_robust_m_estim_1var (g07dbc):

(a) **Null Weights**

\[
\psi(t) = t \quad \chi(t) = \frac{t^2}{2}
\]

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

(b) **Huber’s Function**

\[
\psi(t) = \max(-c, \min(c, t)) \quad \chi(t) = \begin{cases} \frac{|t|^2}{2} & |t| \leq d \\ \frac{d^2}{2} & |t| > d \end{cases}
\]

Mark 25
(c) Hampel’s Piecewise Linear Function

\[ \psi_{h_1,h_2,h_3}(t) = \begin{cases} 
-\psi_{h_1,h_2,h_3}(-t) & 0 \leq t \leq h_1 \\
0 & h_1 \leq t \leq h_2 \\
h_1(h_3 - t)/(h_3 - h_2) & h_2 \leq t \leq h_3 \\
0 & t > h_3 
\end{cases} \]

\[ \chi(t) = \begin{cases} 
\frac{|t|^2}{2} & |t| \leq d \\
\frac{d^2}{2} |t| & |t| > d 
\end{cases} \]

(d) Andrew’s Sine Wave Function

\[ \psi(t) = \sin t \quad -\pi \leq t \leq \pi \]
\[ \chi(t) = \begin{cases} 
\frac{|t|^2}{2} & |t| \leq d \\
\frac{d^2}{2} |t| & |t| > d 
\end{cases} \]

(e) Tukey’s Bi-weight

\[ \psi(t) = t(1 - t^2)^2 \quad |t| \leq 1 \]
\[ \chi(t) = \begin{cases} 
\frac{|t|^2}{2} & |t| \leq d \\
\frac{d^2}{2} |t| & |t| > d 
\end{cases} \]

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 (g07dbc) as the sample median and an estimate of \( \sigma \) 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


5 Arguments

1: \( \text{sigma_est} \) – Nag_SigmaSimulEst \hspace{1cm} Input

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

\( \text{sigma_est} = \) Nag_SigmaBypas

The estimation of \( \hat{\sigma} \) is bypassed and \( \text{sigma} \) is set equal to \( \sigma_c \);
\[ \sigma_{\text{est}} = \text{Nag\_SigmaSimul} \]

\[ \hat{\sigma} \] is estimated simultaneously.

*Constraint:* \( \sigma_{\text{est}} = \text{Nag\_SigmaBypas} \) or \( \text{Nag\_SigmaSimul} \).

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: \( \text{psifun} \) – \text{Nag\_PsiFun}  \[ Input \]

*On entry:* which \( \psi \) function is to be used.

\( \text{psifun} = \text{Nag\_Lsq} \)

\[ \psi(t) = t. \]

\( \text{psifun} = \text{Nag\_HuberFun} \)

Huber’s function.

\( \text{psifun} = \text{Nag\_HampelFun} \)

Hampel’s piecewise linear function.

\( \text{psifun} = \text{Nag\_AndrewFun} \)

Andrew’s sine wave.

\( \text{psifun} = \text{Nag\_TukeyFun} \)

Tukey’s bi-weight.

*Constraint:* \( \text{psifun} = \text{Nag\_Lsq}, \text{Nag\_HuberFun}, \text{Nag\_HampelFun}, \text{Nag\_AndrewFun} \) or \( \text{Nag\_TukeyFun} \).

5: \( c \) – double  \[ Input \]

*On entry:* must specify the argument, \( c \), of Huber’s \( \psi \) function, if \( \text{psifun} = \text{Nag\_HuberFun} \). \( c \) is not referenced if \( \text{psifun} \neq \text{Nag\_HuberFun} \).

*Constraint:* \( c > 0.0 \) if \( \text{psifun} = \text{Nag\_HuberFun} \).

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 \( \psi \) function, if \( \text{psifun} = \text{Nag\_HampelFun} \). \( h1, h2, \) and \( h3 \) are not referenced if \( \text{psifun} \neq \text{Nag\_HampelFun} \).

*Constraint:* \( 0 \leq h1 \leq h2 \leq h3 \) and \( h3 > 0.0 \) if \( \text{psifun} = \text{Nag\_HampelFun} \).

9: \( dchi \) – double  \[ Input \]

*On entry:* the argument, \( d \), of the \( \chi \) function. \( dchi \) is not referenced if \( \text{psifun} = \text{Nag\_Lsq} \).

*Constraint:* \( dchi > 0.0 \) if \( \text{psifun} \neq \text{Nag\_Lsq} \).

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, \( \hat{\theta} \).
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 \( \sigma \leq 0.0 \) 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 \( \sigma_c \), which is held fixed during the iterations, and the starting value for the calculation of \( \hat{\theta} \). If \( \sigma \leq 0 \), then nag_robust_m_estim_1var (g07dbc) will determine the value of \( \sigma_c \) 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 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 \( \sigma_c \).

maxit – Integer

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

Suggested value: \( p \cdot \text{maxit} = 50 \).

Constraint: maxit > 0.

tol – double

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.

rs[n] – double

On exit: the Winsorized residuals.

nit – Integer *

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

sorted_x[n] – double

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

fail – NagError *

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, \( h_1 = \langle \text{value} \rangle \), \( h_2 = \langle \text{value} \rangle \) and psifun = \langle \text{value} \rangle. These arguments must satisfy \( h_1 \leq h_2 \), psifun = Nag_HampelFun.

On entry, \( h_1 = \langle \text{value} \rangle \), \( h_3 = \langle \text{value} \rangle \) and psifun = \langle \text{value} \rangle. These arguments must satisfy \( h_1 \leq h_3 \), psifun = Nag_HampelFun.

On entry, \( h_2 = \langle \text{value} \rangle \), \( h_3 = \langle \text{value} \rangle \) and psifun = \langle \text{value} \rangle. These arguments must satisfy \( h_2 \leq h_3 \), psifun = Nag_HampelFun.
NE_3_REAL_ENUM_ARG_CONS
On entry, \( h_1 = \langle \text{value} \rangle, \ h_2 = \langle \text{value} \rangle, \ h_3 = \langle \text{value} \rangle, \ psifun = \langle \text{value} \rangle \). These arguments must satisfy \( h_1 = h_2 = h_3 \neq 0.0 \), \( psifun = \text{Nag}_\text{HampelFun} \).

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 \( \leq 0.0 \) during an iteration.

NE_INT_ARG_LE
On entry, \( maxit = \langle \text{value} \rangle \).
Constraint: \( maxit > 0 \).
On entry, \( n = \langle \text{value} \rangle \).
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: \( tol = \langle \text{value} \rangle \).

NE_REAL_ENUM_ARG_CONS
On entry, \( c = \langle \text{value} \rangle, \ psifun = \langle \text{value} \rangle \). These arguments must satisfy \( c > 0.0 \), \( psifun = \text{Nag}_\text{HuberFun} \).
On entry, \( dchi = \langle \text{value} \rangle, \ psifun = \langle \text{value} \rangle \). These arguments must satisfy \( dchi > 0.0 \), \( psifun \neq \text{Nag}_\text{Lsq} \).
On entry, \( h_1 = \langle \text{value} \rangle, \ psifun = \langle \text{value} \rangle \). These arguments must satisfy \( h_1 \geq 0.0 \), \( psifun = \text{Nag}_\text{HampelFun} \).

NE_TOO_MANY
Too many iterations (\( \langle \text{value} \rangle \)).

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 \( sigma_est = \text{Nag}_\text{SigmaBypas} \) option with a redescending \( \psi \) function, i.e., Hampel’s piecewise linear function, Andrew’s sine wave, and Tukey’s biweight.
If the given value of \( \sigma \) is too small, then the standardized residuals \( \frac{x - d_i}{\sigma} \), 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 \( \sigma \) or with \( sigma_est = \text{Nag}_\text{SigmaSimul} \).
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 $\sigma$. If too small a value of $\sigma$ is chosen then initial values of the standardized residuals $\frac{r_i - h_i}{c}$ will be large. If the redescending $\psi$ 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, Hampel’s Piecewise Linear Function is used ($psifun = \text{Nag\_HampelFun}$), values for $h_1, h_2$ and $h_3$ along with $d$ for the $\chi$ function, being read from the data file.

Using the following starting values various estimates of $\theta$ and $\sigma$ are calculated and printed along with the number of iterations used:

(a) nag_robust_m_estim_1var (g07dbc) determines the starting values, $\sigma$ is estimated simultaneously.

(b) You supply the starting values, $\sigma$ is estimated simultaneously.

(c) nag_robust_m_estim_1var (g07dbc) determines the starting values, $\sigma$ is fixed.

(d) You supply the starting values, $\sigma$ 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;
    Nag_error fail;

    INIT_FAIL(fail);

    printf(}
nag_robust_m_estim_1var (g07dbc) Example Program Results

/* Skip heading in data file */

#define _WIN32
scanf_s("%*[\n]
");
#else
scanf("%*[\n]
");
#endif
#define _WIN32
scanf_s("%"NAG_IFMT" %*[\n]
", &n);
#else
scanf("%"NAG_IFMT" %*[\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)
{
    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]
");
#else
    scanf("%*[\n]
");
#endif

#ifdef _WIN32
    scanf_s("%lf %lf %lf %lf %"NAG_IFMT" %*[\n]
", &h1, &h2, &h3, &dchi, &maxit);
#else
    scanf("%lf %lf %lf %lf %"NAG_IFMT" %*[\n]
", &h1, &h2, &h3, &dchi, &maxit);
#endif

printf("%25sInput parameters Output parameters\n", "");
printf("%25sInput parameters Output parameters\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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>13.0</td>
<td>11.0</td>
<td>16.0</td>
<td>5.0</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>Observations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>3.0</td>
<td>4.5</td>
<td>1.5</td>
<td>50</td>
<td>h1</td>
</tr>
<tr>
<td>Nag_SigmaSimul</td>
<td>-1.0</td>
<td>0.0</td>
<td>0.0001</td>
<td>sigma_est</td>
<td>sigma</td>
</tr>
<tr>
<td>Nag_SigmaSimul</td>
<td>7.0</td>
<td>2.0</td>
<td>0.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nag_SigmaBypas</td>
<td>-1.0</td>
<td>0.0</td>
<td>0.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nag_SigmaBypas</td>
<td>7.0</td>
<td>2.0</td>
<td>0.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_robust_m_estim_1var (g07dbc) Example Program Results

<table>
<thead>
<tr>
<th>sigma_est</th>
<th>Input parameters</th>
<th>Output parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sigma</td>
<td>theta</td>
</tr>
<tr>
<td>Nag_SigmaSimul</td>
<td>-1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Nag_SigmaSimul</td>
<td>7.0000</td>
<td>2.0000</td>
</tr>
<tr>
<td>Nag_SigmaBypas</td>
<td>-1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Nag_SigmaBypas</td>
<td>7.0000</td>
<td>2.0000</td>
</tr>
</tbody>
</table>