1 Purpose

nag_rank_regsn (g08rac) calculates the parameter estimates, score statistics and their variance-covariance matrices for the linear model using a likelihood based on the ranks of the observations.

2 Specification

void nag_rank_regsn (Nag_OrderType order, Integer ns, const Integer nv[],
                  const double y[], Integer p, const double x[], Integer pdx,
                  Integer nmax, double tol, double parvar[], Integer pdparvar,
                  Integer irank[],
                  double zin[], double eta[], double vapvec[],
                  double parest[], NagError *fail)

3 Description

Analysis of data can be made by replacing observations by their ranks. The analysis produces inference for regression parameters arising from the following model.

For random variables \(Y_1, Y_2, \ldots, Y_n\) we assume that, after an arbitrary monotone increasing differentiable transformation, \(h(.)\), the model

\[
h(Y_i) = x_i^T \beta + \epsilon_i
\]

holds, where \(x_i\) is a known vector of explanatory variables and \(\beta\) is a vector of \(p\) unknown regression coefficients. The \(\epsilon_i\) are random variables assumed to be independent and identically distributed with a completely known distribution which can be one of the following: Normal, logistic, extreme value or double-exponential. In Pettitt (1982) an estimate for \(\beta\) is proposed as \(\hat{\beta} = MX^T a\) with estimated variance-covariance matrix \(M\). The statistics \(a\) and \(M\) depend on the ranks \(r_i\) of the observations \(Y_i\) and the density chosen for \(\epsilon_i\).

The matrix \(X\) is the \(n\) by \(p\) matrix of explanatory variables. It is assumed that \(X\) is of rank \(p\) and that a column or a linear combination of columns of \(X\) is not equal to the column vector of 1 or a multiple of it. This means that a constant term cannot be included in the model (1). The statistics \(a\) and \(M\) are found as follows. Let \(\epsilon_i\) have pdf \(f(\cdot)\) and let \(q = -f'/f\). Let \(W_1, W_2, \ldots, W_n\) be order statistics for a random sample of size \(n\) with the density \(f(.)\). Define \(Z_i = g(W_i)\), then \(a_i = E(Z_i)\). To define \(M\) we need \(M^{-1} = X^T(B - A)X\), where \(B\) is an \(n\) by \(n\) diagonal matrix with \(B_{ii} = E(g'(W_{r_i}))\) and \(A\) is a symmetric matrix with \(A_{ij} = \text{cov}(Z_{r_i}, Z_{r_j})\). In the case of the Normal distribution, the \(Z_1 < \cdots < Z_n\) are standard Normal order statistics and \(E(g'(W_{r_i})) = 1\), for \(i = 1, 2, \ldots, n\).

The analysis can also deal with ties in the data. Two observations are adjudged to be tied if \(|Y_i - Y_j| < \text{tol}\), where \(\text{tol}\) is a user-supplied tolerance level.

Various statistics can be found from the analysis:

(a) The score statistic \(X^T a\). This statistic is used to test the hypothesis \(H_0 : \beta = 0\), see (e).

(b) The estimated variance-covariance matrix \(X^T(B - A)X\) of the score statistic in (a).

(c) The estimate \(\hat{\beta} = MX^T a\).

(d) The estimated variance-covariance matrix \(M = (X^T(B - A)X)^{-1}\) of the estimate \(\hat{\beta}\).

(e) The \(\chi^2\) statistic \(Q = \hat{\beta}^T M^{-1} \hat{\beta} = a^T X(X^T(B - A)X)^{-1} X^T a\) used to test \(H_0 : \beta = 0\). Under \(H_0\), \(Q\) has an approximate \(\chi^2\) distribution with \(p\) degrees of freedom.

(f) The standard errors \(M^{-1/2}\) of the estimates given in (c).
(g) Approximate z-statistics, i.e., \( Z_i = \frac{\hat{\beta}_i}{\text{se}(\hat{\beta}_i)} \) for testing \( H_0 : \beta_i = 0 \). For \( i = 1, 2, \ldots, n \), \( Z_i \) has an approximate \( N(0, 1) \) distribution.

In many situations, more than one sample of observations will be available. In this case we assume the model

\[
h_k(Y_k) = X_k^T \beta + e_k, \quad k = 1, 2, \ldots, \text{ns},
\]

where \( \text{ns} \) is the number of samples. In an obvious manner, \( Y_k \) and \( X_k \) are the vector of observations and the design matrix for the \( k \)th sample respectively. Note that the arbitrary transformation \( h_k \) can be assumed different for each sample since observations are ranked within the sample.

The earlier analysis can be extended to give a combined estimate of \( \beta \) as \( \hat{\beta} = Dd \), where

\[
D^{-1} = \sum_{k=1}^{\text{ns}} X_k^T (B_k - A_k) X_k
\]

and

\[
d = \sum_{k=1}^{\text{ns}} X_k^T a_k,
\]

with \( a_k, B_k \) and \( A_k \) defined as \( a, B \) and \( A \) above but for the \( k \)th sample.

The remaining statistics are calculated as for the one sample case.

4 References


5 Parameters

1: \textbf{order} \quad \text{Nag OrderType} \quad \text{Input}

\textit{On entry:} the order parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \textit{order} = Nag_RowMajor. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\textit{Constraint:} order = Nag_RowMajor or Nag_ColMajor.

2: \textbf{ns} \quad \text{Integer} \quad \text{Input}

\textit{On entry:} the number of samples. 

\textit{Constraint:} ns \geq 1.

3: \textbf{nv[ns]} \quad \text{Integer} \quad \text{Input/Output}

\textit{On entry:} the number of observations in the \( i \)th sample, for \( i = 1, 2, \ldots, \text{ns} \).

\textit{On exit:} used as internal workspace prior to being restored and hence is unchanged.

\textit{Constraint:} \( \text{nv}[i] \geq 1 \) for \( i = 0, 1, \ldots, \text{ns} - 1 \).

4: \textbf{y[dim]} \quad \text{const double} \quad \text{Input}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( y \) must be at least \( \sum_{i=0}^{\text{ns}-1} \text{nv}[i] \).

\textit{On entry:} the observations in each sample. Specifically, \( y \left( \sum_{k=1}^{i-1} \text{nv}[k-1] + j \right) \) must contain the \( j \)th observation in the \( i \)th sample.
5: \( p \) – Integer

On entry: the number of parameters to be fitted.

Constraint: \( p \geq 1 \).

6: \( x[\text{dim}] \) – const double

Input

Note: the dimension, \( \text{dim} \), of the array \( x \) must be at least \( \max(1, \text{pd} x \times p) \) when \( \text{order} = \text{Nag} \_\text{ColMajor} \) and at least \( \max(1, \text{pd} x \times \sum_{i=0}^{n-1} \text{nv}[i]) \) when \( \text{order} = \text{Nag} \_\text{RowMajor} \).

If \( \text{order} = \text{Nag} \_\text{ColMajor} \), the \((i, j)\)th element of the matrix \( X \) is stored in \( x[j-1] \times \text{pd} x + i - 1 \) and if \( \text{order} = \text{Nag} \_\text{RowMajor} \), the \((i, j)\)th element of the matrix \( X \) is stored in \( x[(i-1) \times \text{pd} x + j - 1] \).

On entry: the design matrices for each sample. Specifically, \( x \sum_{k=1}^{nv} \text{nv}[k-1] + j, l \) must contain the value of the \( l \)th explanatory variable for the \( j \)th observation in the \( i \)th sample.

Constraint: \( x \) must not contain a column with all elements equal.

7: \( \text{pdx} \) – Integer

Input

On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( x \).

Constraints:

if \( \text{order} = \text{Nag} \_\text{ColMajor} \), \( \text{pdx} \geq \sum_{i=0}^{n-1} \text{nv}[i]; \)
if \( \text{order} = \text{Nag} \_\text{RowMajor} \), \( \text{pdx} \geq p \).

8: \( \text{idist} \) – Integer

Input

On entry: the error distribution to be used in the analysis as follows:

\( \text{idist} = 1 \)
Normal.

\( \text{idist} = 2 \)
Logistic.

\( \text{idist} = 3 \)
Extreme value.

\( \text{idist} = 4 \)
Double-exponential.

Constraint: \( 1 \leq \text{idist} \leq 4 \).

9: \( \text{nmax} \) – Integer

Input

On entry: the value of the largest sample size.

Constraint: \( \text{nmax} = \max_{1 \leq i \leq ns} (\text{nv}[i-1]) \) and \( \text{nmax} > p \).

10: \( \text{tol} \) – double

Input

On entry: the tolerance for judging whether two observations are tied. Thus, observations \( Y_i \) and \( Y_j \) are adjudged to be tied if \( |Y_i - Y_j| < \text{tol} \).

Constraint: \( \text{tol} > 0.0 \).

11: \( \text{parvar}[\text{dim}] \) – double

Output

Note: the dimension, \( \text{dim} \), of the array \( \text{parvar} \) must be at least \( \max(1, \text{pdparvar} \times p) \) when \( \text{order} = \text{Nag} \_\text{ColMajor} \) and at least \( \max(1, \text{pdparvar} \times p + 1) \) when \( \text{order} = \text{Nag} \_\text{RowMajor} \).
Where \( \text{PARVAR}(i, j) \) appears in this document, it refers to the array element

\[
\text{if \ order = Nag\_ColMajor, \ \text{parvar}[(j - 1) \times \pdparvar + i - 1];}
\]
\[
\text{if \ order = Nag\_RowMajor, \ \text{parvar}[(i - 1) \times \pdparvar + j - 1].}
\]

On exit: the variance-covariance matrices of the score statistics and the parameter estimates, the former being stored in the upper triangle and the latter in the lower triangle. Thus for \( 1 \leq i \leq j \leq p \), \( \text{PARVAR}(i, j) \) contains an estimate of the covariance between the \( i \)th and \( j \)th score statistics. For \( 1 \leq j \leq i \leq p - 1 \), \( \text{PARVAR}(i + 1, j) \) contains an estimate of the covariance between the \( i \)th and \( j \)th parameter estimates.

12: \( \pdparvar \) – Integer

\( \text{Input} \)

On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{parvar} \).

Constraints:

\[
\text{if \ order = Nag\_ColMajor, \ pdparvar \geq p + 1;}
\]
\[
\text{if \ order = Nag\_RowMajor, \ pdparvar \geq p.}
\]

13: \( \text{irank}[\text{nmax}] \) – Integer

\( \text{Output} \)

On exit: for the one sample case, \( \text{irank} \) contains the ranks of the observations.

14: \( \text{zin}[\text{nmax}] \) – double

\( \text{Output} \)

On exit: for the one sample case, \( \text{zin} \) contains the expected values of the function \( g(.) \) of the order statistics.

15: \( \text{eta}[\text{nmax}] \) – double

\( \text{Output} \)

On exit: for the one sample case, \( \text{eta} \) contains the expected values of the function \( g'(.) \) of the order statistics.

16: \( \text{vapvec}[\text{dim}] \) – double

\( \text{Output} \)

Note: the dimension, \( \text{dim} \), of the array \( \text{vapvec} \) must be at least \( \text{nmax} \times (\text{nmax} + 1)/2. \)

On exit: for the one sample case, \( \text{vapvec} \) contains the upper triangle of the variance-covariance matrix of the function \( g(.) \) of the order statistics stored column-wise.

17: \( \text{parest}[\text{dim}] \) – double

\( \text{Output} \)

Note: the dimension, \( \text{dim} \), of the array \( \text{parest} \) must be at least \( 4 \times p + 1. \)

On exit: the statistics calculated by the routine as follows. The first \( p \) components of \( \text{parest} \) contain the score statistics. The next \( p \) elements contain the parameter estimates. \( \text{parest}[2 \times p] \) contains the value of the \( \chi^2 \) statistic. The next \( p \) elements of \( \text{parest} \) contain the standard errors of the parameter estimates. Finally, the remaining \( p \) elements of \( \text{parest} \) contain the \( z \)-statistics.

18: \( \text{fail} \) – NagError *

\( \text{Input/Output} \)

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, \( \text{ns} = (\text{value}). \)

Constraint: \( \text{ns} \geq 1. \)

On entry, \( p = (\text{value}). \)

Constraint: \( p \geq 1. \)
On entry, \( \text{pdx} = \langle \text{value} \rangle \).
Constraint: \( \text{pdx} > 0 \).

On entry, \( \text{pdparvar} = \langle \text{value} \rangle \).
Constraint: \( \text{pdparvar} > 0 \).

On entry, \( \text{idist} \) is outside the range 1 to 4: \( \text{idist} = \langle \text{value} \rangle \).

**NE_INT_2**

On entry, \( \text{pdx} = \langle \text{value} \rangle, \text{p} = \langle \text{value} \rangle \).
Constraint: \( \text{pdx} \geq \text{p} \).

On entry, \( \text{pdparvar} = \langle \text{value} \rangle, \text{p} = \langle \text{value} \rangle \).
Constraint: \( \text{pdparvar} \geq \text{p} + 1 \).

On entry, \( \text{pdparvar} = \langle \text{value} \rangle, \text{p} = \langle \text{value} \rangle \).
Constraint: \( \text{pdparvar} \geq \text{p} \).

On entry, \( \text{pdx} < \) the sum of \( \text{nv}[i] \): \( \text{pdx} = \langle \text{value} \rangle \), sum \( \text{nv}[i] = \langle \text{value} \rangle \).

On entry, \( \text{nmax} \leq \text{p} \): \( \text{nmax} = \langle \text{value} \rangle, \text{p} = \langle \text{value} \rangle \).

**NE_INT_ARRAY**

On entry, \( \text{nv}[i] = \langle \text{value} \rangle \).
Constraint: \( \text{nv}[i] \geq 1 \) for \( i = 0, \ldots, \text{ns} - 1 \).

**NE_INT_ARRAY_ELEM_CONS**

\( M \) elements of array \( \text{nv} \) are less than or equal to zero: \( M = \langle \text{value} \rangle \).

**NE_MAT_ILL_DEFINED**

The matrix \( X^T(B - A)X \) is either singular or non-positive-definite.

**NE_OBSERVATIONS**

All the observations were adjudged to be tied.

**NE_REAL**

On entry, \( \text{tol} = \langle \text{value} \rangle \).
Constraint: \( \text{tol} > 0.0 \).

**NE_REAL_ARRAY_ELEM_CONS**

On entry, all elements in column \( \langle \text{value} \rangle \) of \( x \) are equal to \( \langle \text{value} \rangle \).

**NE_SAMPLE**

The largest sample size is \( \langle \text{value} \rangle \) which is not equal to \( \text{nmax}, \text{nmax} = \langle \text{value} \rangle \).

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

**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.
7 Accuracy
The computations are believed to be stable.

8 Further Comments
The time taken by the routine depends on the number of samples, the total number of observations and the number of parameters fitted.

In extreme cases the parameter estimates for certain models can be infinite, although this is unlikely to occur in practice. See Pettitt (1982) for further details.

9 Example
A program to fit a regression model to a single sample of 20 observations using two explanatory variables. The error distribution will be taken to be logistic.

9.1 Program Text
/* nag_rank_regsn (g08rac) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
#
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg08.h>

int main(void)
{
    /* Scalars */
    double tol;
    Integer exit_status, i, idist, p, j, nmax, ns, nsum;
    Integer pdx, pdparvar;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    double *eta=0, *parest=0, *parvar=0, *vapvec=0, *x=0, *y=0, *zin=0;
    Integer *irank=0, *nv=0;

    #ifdef NAG_COLUMN_MAJOR
    #define X(I,J) x[(J-1)*pd x+I-1 ]
    #define PARVAR(I,J) parvar[(J-1)*pdparvar + I - 1]
    order = Nag_ColMajor;
    #else
    #define X(I,J) x[(I-1)*pd x+J-1 ]
    #define PARVAR(I,J) parvar[(I-1)*pdparvar + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    exit_status = 0;
    Vprintf("g08rac Example Program Results\n");

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

    /* Read number of samples, number of parameters to be fitted,
    * error distribution parameter and tolerance criterion for ties.
    */
    Vscanf("%ld%ld%ld%lf%*[\n"] , &ns, &p, &idist, &tol);
/* Allocate memory to nv only */
if ( !(nv = NAG_ALLOC(ns, Integer)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

Vprintf("\n");
Vprintf("Number of samples =%2ld\n", ns);
Vprintf("Number of parameters fitted =%2ld\n", p);
Vprintf("Distribution =%2ld\n", idist);
Vprintf("Tolerance for ties =%8.5f\n", tol);

/* Read the number of observations in each sample. */
for (i = 1; i <= ns; ++i)
    Vscanf("%ld", &nv[i - 1]);
Vscanf("%*[^\n] ");
nmax = 0;
nsum = 0;
for (i = 1; i <= ns; ++i)
{
    nsum += nv[i - 1];
    nmax = MAX(nmax, nv[i - 1]);
}
if (nmax > 0 && nmax <= 100 && nsum > 0 && nsum <= 100)
{

    /* Allocate memory */
    if ( !(eta = NAG_ALLOC(nmax, double)) ||
        !(parest = NAG_ALLOC(4*p+1, double)) ||
        !(parvar = NAG_ALLOC((p+1)*p, double)) ||
        !(vapvec = NAG_ALLOC(nmax*(nmax+1)/2, double)) ||
        !(x = NAG_ALLOC(nsum * p, double)) ||
        !(y = NAG_ALLOC(nsum, double)) ||
        !(zin = NAG_ALLOC(nmax, double)) ||
        !(irank = NAG_ALLOC(nmax, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    #ifdef NAG_COLUMN_MAJOR
    pdx = nsum;
pdparvar = p+1;
    #else
    pdx = p;
pdparvar = p;
    #endif

    /* Read in observations and design matrices for each sample. */
    for (i = 1; i <= nsum; ++i)
    {
        Vscanf("%lf", &y[i - 1]);
        for (j = 1; j <= p; ++j)
            Vscanf("%lf", &X(i,j));
    }

    g08rac(order, ns, nv, y, p, x, pdx, idist, nmax, tol,
            parvar, pdparvar, irank, zin, eta, vapvec, parest, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g08rac.\n\n\n", fail.message);
        exit_status = 1;
        goto END;
    }

    Vprintf("\n");
Vprintf(" Score statistic\n");
for (i = 1; i <= p; ++i)
    Vprintf("%9.3f%s", parest[i - 1], i%2 == 0 || i == p ?"\n":" ");
Vprintf("\n");
Vprintf(" Covariance matrix of score statistic\n");
for (j = 1; j <= p; ++j)
    { for (i = 1; i <= j; ++i)
        Vprintf("%9.3f%s", PARVAR(i,j), i%2 == 0 || i == j ?"\n":" ");
    } Vprintf("\n");
Vprintf(" Parameter estimates\n");
for (i = 1; i <= p; ++i)
    Vprintf("%9.3f%s", parest[p + i - 1], i%2 == 0 || i == p ?"\n":" ");
Vprintf("\n");
Vprintf(" Covariance matrix of parameter estimates\n");
for (i = 1; i <= p; ++i)
    { Vprintf(" ");
        for (j = 1; j <= i; ++j)
            Vprintf("%9.3f%s", PARVAR(i + 1,j), j%2 == 0 || j == i ?"\n":" ");
    } Vprintf("\n");
Vprintf(" Chi-squared statistic =%9.3f with%2ld d.f.\n", parest[p * 2], p);
Vprintf("\n");
Vprintf(" Standard errors of estimates and\n");
Vprintf(" approximate z-statistics\n");
for (i = 1; i <= p; ++i)
    Vprintf("%9.3f%14.3f\n", parest[2*p + 1 + i - 1], parest[p * 3 + 1 + i - 1]);
Vprintf("\n");
END:
if (eta) NAG_FREE(eta);
if (parest) NAG_FREE(parest);
if (parvar) NAG_FREE(parvar);
if (vapvec) NAG_FREE(vapvec);
if (x) NAG_FREE(x);
if (y) NAG_FREE(y);
if (zin) NAG_FREE(zin);
if (irank) NAG_FREE(irank);
if (nv) NAG_FREE(nv);

return exit_status;

9.2 Program Data

g08rac Example Program Data
1 2 2 0.00001
20
1.0 1.0 23.0
1.0 1.0 32.0
3.0 1.0 37.0
4.0 1.0 41.0
2.0 1.0 41.0
4.0 1.0 48.0
1.0 1.0 48.0
5.0 1.0 55.0
4.0 1.0 55.0
4.0 0.0 56.0
4.0 1.0 57.0
4.0 1.0 57.0
9.3 Program Results

g08rac Example Program Results

Number of samples = 1
Number of parameters fitted = 2
Distribution = 2
Tolerance for ties = 0.00001

Score statistic
  -1.048  64.333

Covariance matrix of score statistic
  0.673
  -4.159   533.670

Parameter estimates
  -0.852  0.114

Covariance matrix of parameter estimates
  1.560
  0.012  0.002

Chi-squared statistic = 8.221 with 2 d.f.

Standard errors of estimates and approximate z-statistics
  1.249  -0.682
  0.044   2.567