nag_regsn_mult_linear (g02dac)

1. Purpose

nag_regsn_mult_linear (g02dac) performs a general multiple linear regression when the independent variables may be linearly dependent. Parameter estimates, standard errors, residuals and influence statistics are computed. nag_regsn_mult_linear may be used to perform a weighted regression.

2. Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_regsn_mult_linear(Nag_IncludeMean mean, Integer n, double x[],
     Integer tdx, Integer m, Integer sx[], Integer ip, double y[], double wt[],
     double *rss, double *df, double b[], double se[], double cov[],
     double res[], double h[], double q[], Integer tdq, Boolean *svd,
     Integer *rank, double p[], double tol, double com_ar[], NagError *fail)
```

3. Description

The general linear regression model is defined by

$$y = X\beta + \varepsilon$$

where y is a vector of n observations on the dependent variable,
X is a n by p matrix of the independent variables of column rank k,
β is a vector of length p of unknown parameters,
and ε is a vector of length n of unknown random errors such that $\operatorname{var}\varepsilon = V\sigma^2$, where V is a known diagonal matrix.

Note: the p independent variables may be selected by the user from a set of m potential independent variables.

If $V = I$, the identity matrix, then least-squares estimation is used. If $V \neq I$, then for a given weight matrix $W \propto V^{-1}$, weighted least-squares estimation is used.

The least-squares estimates $\hat{\beta}$ of the parameters β minimize $(y - X\beta)^T(y - X\beta)$ while the weighted least-squares estimates minimize $(y - X\beta)^T W(y - X\beta)$.

nag_regsn_mult_linear finds a QR decomposition of X (or $W^{1/2}X$ in the weighted case), i.e.

$$X = QR^* \text{ (or } W^{1/2}X = QR^*\text{)}$$

where $R^* = \begin{pmatrix} R \\ 0 \end{pmatrix}$ and R is a p by p upper triangular matrix and Q is an n by n orthogonal matrix. If R is of full rank, then $\hat{\beta}$ is the solution to

$$R\hat{\beta} = c_1$$

where $c = Q^T y$ (or $Q^T W^{1/2} y$) and c_1 is the first p elements of c. If R is not of full rank a solution is obtained by means of a singular value decomposition (SVD) of R,

$$R = Q_* \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} P^T$$

where D is a k by k diagonal matrix with non-zero diagonal elements, k being the rank of R and Q_* and P are p by p orthogonal matrices. This gives the solution

$$\hat{\beta} = P_1 D^{-1} Q_{*_1}^T c_1$$

P_1 being the first k columns of P, i.e., $P = (P_1 P_0)$ and Q_{*_1} being the first k columns of Q_*.

Details of the SVD are made available, in the form of the matrix P^*:

$$P^* = \begin{pmatrix} D^{-1}P_1^T \\ P_0^T \end{pmatrix}.$$

This will be only one of the possible solutions. Other estimates may be obtained by applying constraints to the parameters. These solutions can be obtained by using nag_regsn_mult_linear_tran_model (g02dkc) after using nag_regsn_mult_linear (g02dac). Only certain linear combinations of the parameters will have unique estimates; these are known as estimable functions.

The fit of the model can be examined by considering the residuals, $r_i = y_i - \hat{y}$, where $\hat{y} = X\hat{\beta}$ are the fitted values. The fitted values can be written as Hy for an n by n matrix H. The ith diagonal element of H, h_i, gives a measure of the influence of the ith value of the independent variables on the fitted regression model. The values h_i are sometimes known as leverages. Both r_i and h_i are provided by nag_regsn_mult_linear.

The output of nag_regsn_mult_linear also includes $\hat{\beta}$, the residual sum of squares and associated degrees of freedom, $(n-k)$, the standard errors of the parameter estimates and the variance–covariance matrix of the parameter estimates.

In many linear regression models the first term is taken as a mean term or an intercept, i.e., $X_{i,1} = 1$, for $i = 1, 2, \ldots, n$. This is provided as an option. Also note that not all the potential independent variables need to be included in a model; a facility to select variables to be included in the model is provided.

Details of the QR decomposition and, if used, the SVD, are made available. These allow the regression to be updated by adding or deleting an observation using nag_regsn_mult_linear_addrem_obs (g02dcc), adding or deleting a variable using nag_regsn_mult_linear_add_var (g02dec) and nag_regsn_mult_linear_delete_var (g02dfc) or estimating and testing an estimable function using nag_regsn_mult_linear_est_func (g02dnc).

4. Parameters

mean

Input: indicates if a mean term is to be included.
If **mean** = **Nag_MeanInclude**, a mean term, (intercept), will be included in the model.
If **mean** = **Nag_MeanZero**, the model will pass through the origin, zero point.
Constraint: **mean** = **Nag_MeanInclude** or **Nag_MeanZero**.

n

Input: the number of observations, n.
Constraint: **n** ≥ 2.

x[n][tdx]

Input: **x**$[i][j]$ must contain the ith observation for the jth potential independent variable, for $i = 0, 1, \ldots, n-1$; $j = 0, 1, \ldots, m-1$.

tdx

Input: the second dimension of the array **x** as declared in the function from which nag_regsn_mult_linear is called.
Constraint: **tdx** $\geq$ **m**.

m

Input: the total number of independent variables in the data set, m.
Constraint: **m** ≥ 1.

sx[m]

Input: indicates which of the potential independent variables are to be included in the model.
If **sx**$[j] > 0$, then the variable contained in the corresponding column of **x** is included in the regression model.
Constraint: **sx**$[j] \geq 0$, for $j = 0, 1, \ldots, m-1$.
Constraint: if **mean** = **Nag_MeanInclude**, then exactly **ip** $-$ 1 values of **sx** must be > 0.
Constraint: if **mean** = **Nag_MeanZero**, then exactly **ip** values of **sx** must be > 0.

ip

Input: the number p of independent variables in the model, including the mean or intercept if present.
Constraint: $1 \leq$ **ip** $\leq$ **n**.

y[n]

Input: observations on the dependent variable, y.

wt[n]

Input: if weighted estimates are required then **wt** must contain the weights to be used in the weighted regression. Otherwise **wt** need not be defined and may be set to the null pointer **NULL**, i.e., (double *) 0.
If **wt**$[i] = 0.0$, then the ith observation is not included in the model, in which case the effective number of observations is the number of observations with positive weights. The values of **res** and **h** will be set to zero for observations with zero weights.
If **wt** = **NULL**, then the effective number of observations is n.
Constraint: **wt** = **NULL** or **wt**$[i] \geq 0.0$, for $i = 0, 1, \ldots, n-1$.

rss

Output: the residual sum of squares for the regression.

df

Output: the degrees of freedom associated with the residual sum of squares.

b[ip]

Output: **b**$[i]$, $i = 0, 1, \ldots,$**ip**-1 contain the least-squares estimates of the parameters of the regression model, $\hat{\beta}$.
If **mean** = **Nag_MeanInclude**, then **b**[0] will contain the estimate of the mean parameter and **b**$[i]$ will contain the coefficient of the variable contained in column j of **x**, where **sx**$[j]$ is the ith positive value in the array **sx**.
If **mean** = **Nag_MeanZero**, then **b**$[i-1]$ will contain the coefficient of the variable contained in column j of **x**, where **sx**$[j]$ is the ith positive value in the array **sx**.

se[ip]

Output: **se**$[i]$, $i = 0, 1, \ldots,$**ip**-1 contains the standard errors of the **ip** parameter estimates given in **b**.

cov[ip∗(ip+1)/2]

Output: the first **ip** $\times$ (**ip**+1)/2 elements of **cov** contain the upper triangular part of the variance-covariance matrix of the **ip** parameter estimates given in **b**. They are stored packed by column, i.e., the covariance between the parameter estimate given in $b[i]$ and the parameter estimate given in $b[j]$, $j \geq i$, is stored in **cov**$[j(j+1)/2 + i]$, for $i = 0, 1, \ldots,$**ip**-1 and $j = i, i+1, \ldots,$**ip**-1.

res[n]

Output: the (weighted) residuals, r_i.

h[n]

Output: the diagonal elements of H, h_i, the leverages.

q[n][tdq]

Output: the results of the QR decomposition:
the first column of **q** contains c,
the upper triangular part of columns 2 to **ip**+1 contain the R matrix,
the strictly lower triangular part of columns 2 to **ip**+1 contain details of the Q matrix.

tdq

Input: the second dimension of the array **q** as declared in the function from which nag_regsn_mult_linear is called.
Constraint: **tdq** $\geq$ **ip**+1.

svd

Output: if a singular value decomposition has been performed then **svd** will be **TRUE**, otherwise **svd** will be **FALSE**.

rank

Output: the rank of the independent variables.
If **svd** = **FALSE**, then **rank** = **ip**.
If **svd** = **TRUE**, then **rank** is an estimate of the rank of the independent variables.
rank is calculated as the number of singular values greater than **tol** (largest singular value).
It is possible for the SVD to be carried out but **rank** to be returned as **ip**.

p[2*ip+ip*ip]

Output: details of the QR decomposition and SVD if used.
If **svd** = **FALSE**, only the first **ip** elements of **p** are used, these will contain the zeta values for the QR decomposition (see nag_real_qr (f01qcc) for details).
If **svd** = **TRUE**, the first **ip** elements of **p** will contain the zeta values for the QR decomposition (see nag_real_qr (f01qcc) for details) and the next **ip** elements of **p** contain singular values. The following **ip** by **ip** elements contain the matrix P^* stored by rows.

tol

Input: the value of **tol** is used to decide what is the rank of the independent variables. The smaller the value of **tol** the stricter the criterion for selecting the singular value decomposition. If **tol** = 0.0, then the singular value decomposition will never be used, this may cause run time errors or inaccurate results if the independent variables are not of full rank.
Suggested value: **tol** = 0.000001.
Constraint: **tol** $\geq$ 0.0.

com_ar[5*(ip−1)+ip*ip]

Output: if on exit **svd** = **TRUE**, then **com_ar** contains information which is needed by nag_regsn_mult_linear_newyvar (g02dgc).

fail

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

5. Error Indications and Warnings

NE_INT_ARG_LT

On entry, **n** must not be less than 2: **n** = ⟨*value*⟩.
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.
On entry, **ip** must not be less than 1: **ip** = ⟨*value*⟩.
On entry, **sx**[⟨*value*⟩] must not be less than 0: **sx**[⟨*value*⟩] = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These parameters must satisfy **tdx** $\geq$ **m**.
On entry, **tdq** = ⟨*value*⟩ while **ip**+1 = ⟨*value*⟩. These parameters must satisfy **tdq** $\geq$ **ip**+1.
On entry, **n** = ⟨*value*⟩ while **ip** = ⟨*value*⟩. These parameters must satisfy **n** $\geq$ **ip**.

NE_REAL_ARG_LT

On entry, **tol** must not be less than 0.0: **tol** = ⟨*value*⟩.
On entry, **wt**[⟨*value*⟩] must not be less than 0.0: **wt**[⟨*value*⟩] = ⟨*value*⟩.

NE_BAD_PARAM

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

NE_BAD_SX_OR_IP

Either a value of **sx** is $<$ 0, or **ip** is incompatible with **mean** and **sx**, or **ip** $>$ the effective number of observations.

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge.

NE_ZERO_DOF_RESID

The degrees of freedom for the residuals are zero, i.e., the designated number of parameters = the effective number of observations.
In this case the parameter estimates will be returned along with the diagonal elements of H, but neither standard errors nor the variance-covariance matrix will be calculated.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

Function nag_regsn_std_resid_influence (g02fac) can be used to compute standardised residuals and further measures of influence. This function requires, in particular, the results stored in **res** and **h**.

6.1. Accuracy

The accuracy of this function is closely related to the accuracy of nag_real_qr (f01qcc). That function document should be consulted.

6.2. References

Cook R D and Weisberg S (1982) *Residuals and Influence in Regression* Chapman and Hall.

Draper N R and Smith H (1985) *Applied Regression Analysis* (2nd Edn) Wiley.

Golub G H and Van Loan C F (1983) *Matrix Computations* Johns Hopkins University Press, Baltimore.

Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics *ACM Signum Newsletter* **20** (3) 2–25.

McCullagh P and Nelder J A (1983) *Generalized Linear Models* Chapman and Hall.

Searle S R (1971) *Linear Models* Wiley.

7. See Also

nag_real_qr (f01qcc)
nag_regsn_mult_linear_addrem_obs (g02dcc)
nag_regsn_mult_linear_add_var (g02dec)
nag_regsn_mult_linear_delete_var (g02dfc)
nag_regsn_mult_linear_newyvar (g02dgc)
nag_regsn_mult_linear_est_func (g02dnc)
nag_regsn_std_resid_influence (g02fac)

8. Example

For this function two examples are presented, in Sections 8.1 and 8.2. In the example programs distributed to sites, there is a single example program for nag_regsn_mult_linear, with a main function:

```
/* nag_regsn_mult_linear(g02dac) Example Program
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 *  Mark 5 revised, 1998.
 */
#include <nag.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>


#ifdef NAG_PROTO
static void ex1(void);
static void ex2(void);
#else
static void ex1();
static void ex2();
#endif


main()
{
  ex1();
  ex2();
}
```

The code to solve the two example problems is given in the functions ex1 and ex2 in Sections 8.1.1 and 8.2.1 respectively.

8.1. Example 1

Data from an experiment with four treatments and three observations per treatment are read in. The treatments are represented by dummy $(0-1)$ variables. An unweighted model is fitted with a mean included in the model.

8.1.1. Program Text

```
#define NMAX 20
#define MMAX 20
#define TDX MMAX
#define TDQ MMAX+1

#ifdef NAG_PROTO
static void ex1(void)
#else
     static void ex1()
#endif
{
  double rss, tol;
  Integer i, ip, rank, j, m, n;
  double df;
  Boolean  svd;
  char weight, meanc;
  Nag_IncludeMean  mean;
  double b[MMAX], cov[(MMAX*MMAX+MMAX)/2], h[NMAX], p[MMAX*(MMAX+2)],
  q[NMAX][MMAX+1], res[NMAX], se[MMAX], com_ar[MMAX*MMAX+5*(MMAX-1)],
  wt[NMAX], x[NMAX][MMAX], y[NMAX];
  double *wtptr;
  Integer  sx[MMAX];

  Vprintf("g02dac Example 1 Program Results\n");
  /*  Skip heading in data file */
  Vscanf("%*[^\n]");
  Vscanf("%ld %ld %c %c", &n, &m, &weight, &meanc);
  if (meanc=='m')
    mean = Nag_MeanInclude;
  else
    mean = Nag_MeanZero;
  if (n<=NMAX && m<MMAX)
    {
      if (weight=='w')
        {
          wtptr = wt;
          for (i=0; i<n; i++)
            {
              for (j=0; j<m; j++)
                Vscanf("%lf", &x[i][j]);
              Vscanf("%lf%lf", &y[i], &wt[i]);
            }
        }
      else
        {
          wtptr = (double *)0;
          for (i=0; i<n; i++)
            {
              for (j=0; j<m; j++)
                Vscanf("%lf", &x[i][j]);
              Vscanf("%lf", &y[i]);
            }
        }
      for (j=0; j<m; j++)
        Vscanf("%ld", &sx[j]);
      /* Calculate ip */
      ip = 0;
      if (mean==Nag_MeanInclude)
        ip += 1;
      for (i=0; i<m; i++)
        if (sx[i]>0) ip += 1;
```

```
        /* Set tolerance */
        tol = 0.00001e0;
        g02dac(mean, n, (double *)x,  (Integer)TDX,  m,  sx,  ip, y,
               wtptr,  &rss,  &df,  b,  se,  cov,  res,  h,  (double *)q,
               (Integer)(TDQ),  &svd,  &rank,  p,  tol,  com_ar,  NAGERR_DEFAULT);

        if (svd)
          Vprintf("Model not of full rank, rank = %4ld\n\n", rank);
        Vprintf("Residual sum of squares = %12.4e\n", rss);
        Vprintf("Degrees of freedom = %3.1f\n\n", df);
        Vprintf("Variable    Parameter estimate   Standard error\n\n");
        for (j=0; j<ip; j++)
          Vprintf("%6ld%20.4e%20.4e\n", j+1, b[j], se[j]);
        Vprintf("\n");
        Vprintf("    Obs         Residuals              h\n\n");
        for (i=0; i<n; i++)
          Vprintf("%6ld%20.4e%20.4e\n", i+1, res[i], h[i]);
      }
    else
      {
        Vfprintf(stderr, "One or both of m and n are out of range:\
 m = %-3ld while  n = %-3ld\n", m, n);
        exit(EXIT_FAILURE);
      }
    return;
}
```

8.1.2. Program Data

```
g02dac Example 1 Program Data
 12 4      u     m
1.0 0.0 0.0 0.0 33.63
0.0 0.0 0.0 1.0 39.62
0.0 1.0 0.0 0.0 38.18
0.0 0.0 1.0 0.0 41.46
0.0 0.0 0.0 1.0 38.02
0.0 1.0 0.0 0.0 35.83
0.0 0.0 0.0 1.0 35.99
1.0 0.0 0.0 0.0 36.58
0.0 0.0 1.0 0.0 42.92
1.0 0.0 0.0 0.0 37.80
0.0 0.0 1.0 0.0 40.43
0.0 1.0 0.0 0.0 37.89
 1   1   1   1
```

8.1.3. Program Results

```
g02dac Example 1 Program Results
Model not of full rank, rank =    4

Residual sum of squares =   2.2227e+01
Degrees of freedom = 8.0

Variable    Parameter estimate   Standard error

     1          3.0557e+01          3.8494e-01
     2          5.4467e+00          8.3896e-01
     3          6.7433e+00          8.3896e-01
     4          1.1047e+01          8.3896e-01
     5          7.3200e+00          8.3896e-01

    Obs         Residuals              h

     1         -2.3733e+00          3.3333e-01
     2          1.7433e+00          3.3333e-01
     3          8.8000e-01          3.3333e-01
     4         -1.4333e-01          3.3333e-01
     5          1.4333e-01          3.3333e-01
     6         -1.4700e+00          3.3333e-01
     7         -1.8867e+00          3.3333e-01
     8          5.7667e-01          3.3333e-01
```

```
   9         1.3167e+00          3.3333e-01
  10         1.7967e+00          3.3333e-01
  11        -1.1733e+00          3.3333e-01
  12         5.9000e-01          3.3333e-01
```

8.2. Example 2

This example program uses nag_regsn_mult_linear (g02dac) to find the coefficient of the n degree polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_o$$

that fits the data, $p(x(i))$ to $y(i)$, in a least-squares sense. In this example nag_regsn_mult_linear (g02dac) is called with both **Nag_MeanInclude** and **Nag_MeanZero**. The polynomial degree, the number of data points and the tolerance can be modified using the example data file.

8.2.1. Program Text

```
#ifdef NAG_PROTO
static void ex2(void)
#else
     static void ex2()
#endif
{
  double rss, tol;
  Integer i, ip, rank, j, m, n, degree, digits;
  double df;
  Boolean  svd;
  Nag_IncludeMean  mean;
  double b[MMAX], cov[(MMAX*MMAX+MMAX)/2], h[NMAX], p[MMAX*(MMAX+2)],
  q[NMAX][MMAX+1], res[NMAX], se[MMAX], com_ar[MMAX*MMAX+5*(MMAX-1)],
  wt[NMAX], x[NMAX][MMAX], y[NMAX];
  double *wtptr = (double *)0;   /* don't use weights */
  Integer  sx[MMAX];

  Vprintf("\n\n\ng02dac Example 2 Program Results\n");
  /*  Skip heading in data file */
  Vscanf(" %*[^\n]");

  /* Use mean = Nag_MeanInclude */

  mean = Nag_MeanInclude;

  Vscanf("%ld%ld%ld",&degree,&n,&digits);

  if (n<=NMAX)
    {

      /* Set tolerance */
      tol = pow(10.0, -(double)digits);
      m = degree;
      ip = degree + 1;

      for  (i = 0; i <ip-1; ++i)
        sx[i] = 1;

      for (i=0; i<n; i++)
        {
          Vscanf("%lf%lf", &x[i][degree-1],&y[i]);
          for (j=0; j <degree; ++j)
            x[i][j] = pow(x[i][degree-1],(double)(degree-j));
        }

      g02dac(mean, n, (double *)x,  (Integer)TDX,  m,  sx,  ip, y,
             wtptr,  &rss,  &df,  b,  se,  cov,  res,  h,  (double *)q,
             (Integer)(TDQ),  &svd,  &rank,  p,  tol,  com_ar,  NAGERR_DEFAULT);


      Vprintf("Regression estimates  (mean = Nag_MeanInclude) \n\n");
      Vprintf("Coefficient    Estimate         Standard error\n\n");
      for (j=1; j<ip; j++)
        Vprintf("a(%ld)%20.4e%20.4e\n", degree+1-j, b[j], se[j]);
```

```
        Vprintf("a(0)%20.4e%20.4e\n", b[0], se[0]);
        Vprintf("\n\n");

        /* Use mean = Nag_MeanZero */

        mean = Nag_MeanZero;

        m = degree + 1;
        for  (i = 0; i <ip; ++i)
          sx[i] = 1;

        for (i=0; i<n; i++)
          x[i][m-1] = 1.0;

        g02dac(mean, n, (double *)x,  (Integer)TDX,  m,  sx,  ip, y,
               wtptr,  &rss,  &df,  b,  se,  cov,  res,  h,  (double *)q,
               (Integer)(TDQ),  &svd,  &rank,  p,  tol,  com_ar,  NAGERR_DEFAULT);

        Vprintf("Regression estimates  (mean = Nag_MeanZero) \n\n");
        Vprintf("Coefficient    Estimate         Standard error\n\n");
        for (j=0; j<ip; j++)
          Vprintf("a(%ld)%20.4e%20.4e\n", degree-j, b[j], se[j]);
        Vprintf("\n\n");
      }
    else
      {
        Vfprintf(stderr, "n is out of range, n = %-3ld\n", n);
        exit(EXIT_FAILURE);
      }
  return;
  }
```

8.2.2. Program Data

```
g02dac Example 2 Program Data
3 11 15
 31.80  -1.23
 50.20   -1.08
120.00  -0.83
188.84  -0.53
250.20  -0.28
270.66  -0.15
360.20   0.26
392.97   0.53
444.54   0.93
530.50   1.08
550.02   1.35
```

8.2.3. Program Results

```
g02dac Example 2 Program Results
Regression estimates  (mean = Nag_MeanInclude)

Coefficient    Estimate         Standard error

a(3)         -8.8628e-09          7.9470e-09
a(2)          9.0059e-06          7.0244e-06
a(1)          2.3641e-03          1.7199e-03
a(0)         -1.2614e+00          1.0568e-01


Regression estimates  (mean = Nag_MeanZero)

Coefficient    Estimate         Standard error

a(3)         -8.8628e-09          7.9470e-09
a(2)          9.0059e-06          7.0244e-06
a(1)          2.3641e-03          1.7199e-03
a(0)         -1.2614e+00          1.0568e-01
```