

## nag\_regsn\_mult\_linear\_tran\_model (g02dkc)

### 1. Purpose

**nag\_regsn\_mult\_linear\_tran\_model (g02dkc)** calculates the estimates of the parameters of a general linear regression model for given constraints from the singular value decomposition results.

### 2. Specification

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

void nag_regsn_mult_linear_tran_model(Integer ip, Integer iconst, double p[],
    double c[], Integer tdc, double b[], double rss, double df, double se[],
    double cov[], NagError *fail)
```

### 3. Description

This function computes the estimates given a set of linear constraints for a general linear regression model which is not of full rank. It is intended for use after a call to `nag_regsn_mult_linear (g02dac)` or `nag_regsn_mult_linear_upd_model (g02ddc)`.

In the case of a model not of full rank the functions use a singular value decomposition (SVD) to find the parameter estimates,  $\hat{\beta}_{svd}$ , and their variance-covariance matrix. Details of the SVD are made available, in the form of the matrix  $P^*$ :

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

as described by `nag_regsn_mult_linear (g02dac)` and `nag_regsn_mult_linear_upd_model (g02ddc)`.

Alternative solutions can be formed by imposing constraints on the parameters. If there are  $p$  parameters and the rank of the model is  $k$ , then  $n_c = p - k$  constraints will have to be imposed to obtain a unique solution.

Let  $C$  be a  $p$  by  $n_c$  matrix of constraints, such that

$$C^T \beta = 0,$$

then the new parameter estimates  $\hat{\beta}_c$  are given by:

$$\begin{aligned} \hat{\beta}_c &= A\hat{\beta}_{svd} \\ &= (I - P_0(C^T P_0)^{-1})\hat{\beta}_{svd}, \end{aligned}$$

where  $I$  is the identity matrix, and the variance-covariance matrix is given by:

$$AP_1 D^{-2} P_1^T A^T$$

provided  $(C^T P_0)^{-1}$  exists.

### 4. Parameters

#### ip

Input: the number of terms in the linear model,  $p$ .

Constraint: **ip**  $\geq 1$ .

#### iconst

Input: the number of constraints to be imposed on the parameters,  $n_c$ .

Constraint: **0** < **iconst** < **ip**.

#### p[ip\*ip+2\*ip]

Input: **p** as returned by `nag_regsn_mult_linear (g02dac)` and `nag_regsn_mult_linear_upd_model (g02ddc)`.

**c[ip][tdc]**

Input: the **iconst** constraints stored by column, i.e., the  $i$ th constraint is stored in the  $i$ th column of **c**.

**tdc**

Input: the last dimension of the array **c** as declared in the function from which nag\_regsn\_mult\_linear\_tran\_model is called.

Constraint: **tdc**  $\geq$  **iconst**.

**b[ip]**

Input: the parameter estimates computed by using the singular value decomposition,  $\hat{\beta}_{svd}$ .

Output: the parameter estimates of the parameters with the constraints imposed,  $\hat{\beta}_c$ .

**rss**

Input: the residual sum of squares as returned by nag\_regsn\_mult\_linear (g02dac) or nag\_regsn\_mult\_linear\_upd\_model (g02ddc).

Constraint: **rss**  $>$  0.0

**df**

Input: the degrees of freedom associated with the residual sum of squares as returned by nag\_regsn\_mult\_linear (g02dac) or nag\_regsn\_mult\_linear\_upd\_model (g02ddc).

Constraint: **df**  $>$  0.0.

**se[ip]**

Output: the standard error of the parameter estimates in **b**.

**cov[(ip\*(ip+1)/2]**

Output: 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  $i = 0, 1, \dots, \mathbf{ip} - 1$  and  $j = i, i+1, \dots, \mathbf{ip} - 1$ .

**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, **ip** must not be less than 1: **ip** =  $\langle value \rangle$ .

**NE\_INT\_ARG\_LE**

On entry, **iconst** must not be less than or equal to 0: **iconst** =  $\langle value \rangle$ .

**NE\_2\_INT\_ARG\_GE**

On entry, **iconst** =  $\langle value \rangle$  while **ip** =  $\langle value \rangle$ . These parameters must satisfy **iconst**  $<$  **ip**.

**NE\_2\_INT\_ARG\_LT**

On entry, **tdc** =  $\langle value \rangle$  while **iconst** =  $\langle value \rangle$ . These parameters must satisfy **tdc**  $\geq$  **iconst**.

**NE\_REAL\_ARG\_LE**

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

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

**NE\_MAT\_NOT\_FULL\_RANK**

Matrix **c** does not give a model of full rank.

**NE\_ALLOC\_FAIL**

Memory allocation failed.

## 6. Further Comments

This function is intended for use in situations in which dummy (0-1) variables have been used such as in the analysis of designed experiments when the user does not wish to change the parameters of the model to give a full rank model. The function is not intended for situations in which the relationships between the independent variables are only approximate.

## 6.1. Accuracy

It should be noted that due to rounding errors a parameter that should be zero when the constraints have been imposed may be returned as a value of order *machine precision*.

## 6.2. References

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

## 7. See Also

nag\_regsn\_mult\_linear (g02dac)  
nag\_regsn\_mult\_linear\_upd\_model (g02ddc)

## 8. Example

Data from an experiment with four treatments and three observations per treatment are read in. A model, including the mean term, is fitted by nag\_regsn\_mult\_linear (g02dac) and the results printed. The constraint that the sum of treatment effects is zero is then read in and the parameter estimates with this constraint imposed are computed by nag\_regsn\_mult\_linear\_tran\_model and printed.

### 8.1. Program Text

```
/* nag_regsn_mult_linear_tran_model(g02dkc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define NMAX 12
#define MMAX 5
#define TDC MMAX
#define TDQ MMAX+1
#define TDX MMAX

main()
{
    double    rss, tol;
    Integer   i, iconst, ip, rank, j, m, n;
    double    df;
    Boolean    svd;
    Nag_IncludeMean mean;
    char      weight, meanc;
    double    b[MMAX], c[MMAX][MMAX], cov[MMAX*(MMAX+1)/2], h[NMAX],
    p[MMAX*(MMAX+2)], q[NMAX][MMAX+1], res[NMAX], se[MMAX],
    com_ar[4*MMAX*MMAX+5*(MMAX-1)], wt[NMAX], x[NMAX][MMAX], y[NMAX];
    Integer   sx[MMAX];
    double    *wtptr;

    Vprintf("g02dkc Example 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]);
Vscanf("%ld", &ip);
/* Set tolerance */
tol = 0.00001e0;
/* Find initial estimates using g02dac */
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("Estimates from g02dac\n\n");
Vprintf("Residual sum of squares = %13.4e\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");
/*
 * Input constraints and call g02dkc
 */
iconst = ip - rank;
for (i=0; i<ip; ++i)
    for (j=0; j<iconst; ++j)
        Vscanf("%lf", &c[i][j]);

g02dkc(ip, iconst, p, (double *)c, (Integer)TDC, b, rss, df, se, cov,
        NAGERR_DEFAULT);

Vprintf("\n");
Vprintf("Estimates from g02dkc using constraints\n\n");
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");
}
else
{
    Vfprintf(stderr, "One or both of m and n are out of range:\n
m = %-3ld while n = %-3ld\n", m, n);
    exit(EXIT_FAILURE);
}
}
exit(EXIT_SUCCESS);
}

```

**8.2. Program Data**

```

g02dkc Example 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 5
 0.0
 1.0
 1.0
 1.0
 1.0
  
```

**8.3. Program Results**

```

g02dkc Example Program Results
Estimates from g02dac
  
```

```

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

Estimates from g02dkc using constraints

Variable	Parameter estimate	Standard error
1	3.8196e+01	4.8117e-01
2	-2.1925e+00	8.3342e-01
3	-8.9583e-01	8.3342e-01
4	3.4075e+00	8.3342e-01
5	-3.1917e-01	8.3342e-01