NAG Library Function Document

nag_regsn_mult_linear_upd_model (g02ddc)

1 Purpose

nag_regsn_mult_linear_upd_model (g02ddc) calculates the regression arguments for a general linear regression model. It is intended to be called after nag_regsn_mult_linear_addrem_obs (g02dcc), nag_regsn_mult_linear_add_var (g02dec) or nag_regsn_mult_linear_delete_var (g02dfc).

2 Specification

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

```
void nag_regsn_mult_linear_upd_model (Integer n, Integer ip,
     const double q[], Integer tdq, double *rss, double *df, double b[],
     double se[], double cov[], Nag_Boolean *svd, Integer *rank, double p[],
     double tol, NagError *fail)
```

3 Description

A general linear regression model fitted by nag_regsn_mult_linear (g02dac) may be adjusted by adding or deleting an observation using nag_regsn_mult_linear_addrem_obs (g02dcc), adding a new independent variable using nag_regsn_mult_linear_add_var (g02dec) or deleting an existing independent variable using nag_regsn_mult_linear_delete_var (g02dfc). These functions compute the vector c and the upper triangular matrix R. nag_regsn_mult_linear_upd_model (g02ddc) takes these basic results and computes the regression coefficients, $\hat{\beta}$, their standard errors and their variance-covariance matrix.

If R is of full rank, then $\hat{\beta}$ is the solution to:

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

where 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,

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

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

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

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^*:

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

This will be only one of the possible solutions. Other estimates may be obtained by applying constraints to the arguments. These solutions can be obtained by calling nag_regsn_mult_linear_tran_model (g02dkc) after calling nag_regsn_mult_linear_upd_model (g02ddc). Only certain linear combinations of the arguments will have unique estimates, these are known as estimable functions. These can be estimated using nag_regsn_mult_linear_est_func (g02dnc).

The residual sum of squares required to calculate the standard errors and the variance-covariance matrix can either be input or can be calculated if additional information on c for the whole sample is provided.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM Newsl.* **20(3)** 2–25

Searle S R (1971) *Linear Models* Wiley

5 Arguments

1: **n** – Integer *Input*

On entry: number of observations.

Constraint: $\mathbf{n} \geq 1$.

2: **ip** – Integer *Input*

On entry: the number of terms in the regression model, p.

Constraint: $\mathbf{ip} \geq 1$.

3: **q**[**n** × **tdq**] – const double *Input*

Note: the (i,j)th element of the matrix Q is stored in $\mathbf{q}[(i-1) \times \mathbf{tdq} + j - 1]$.

On entry: **q** must be the array **q** as output by nag_regsn_mult_linear_addrem_obs (g02dcc), nag_regsn_mult_linear_add_var (g02dec) or nag_regsn_mult_linear_delete_var (g02dfc). If on entry $\mathbf{rss} \leq 0.0$ then all **n** elements of c are needed. This is provided by functions nag_regsn_mult_linear_add_var (g02dec) or nag_regsn_mult_linear_delete_var (g02dfc).

4: **tdq** – Integer *Input*

On entry: the stride separating matrix column elements in the array **q**.

Constraint: $\mathbf{tdq} \geq \mathbf{ip} + 1$.

5: **rss** – double * *Input/Output*

On entry: either the residual sum of squares or a value less than or equal to 0.0 to indicate that the residual sum of squares is to be calculated by the function.

On exit: if $\mathbf{rss} \leq 0.0$ on entry, then on exit **rss** will contain the residual sum of squares as calculated by nag_regsn_mult_linear_upd_model (g02ddc).

If **rss** was positive on entry, then it will be unchanged.

6: **df** – double * *Output*

On exit: the degrees of freedom associated with the residual sum of squares.

7: **b**[**ip**] – double *Output*

On exit: the estimates of the p arguments, $\hat{\beta}$.

8: **se**[**ip**] – double *Output*

On exit: the standard errors of the p arguments given in **b**.

9: **cov**[**ip** × (**ip** + **1**)/**2**] – double *Output*

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

10: **svd** – Nag_Boolean * *Output*

On exit: if a singular value decomposition has been performed, then **svd** = Nag_TRUE, otherwise **svd** = Nag_FALSE.

11: **rank** – Integer * *Output*

On exit: the rank of the independent variables.

If **svd** = Nag_FALSE, **rank** = **ip**.

If **svd** = Nag_TRUE, **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 singular value decomposition to be carried out but **rank** to be returned as **ip**.

12: $\mathbf{p}[\mathbf{ip} \times \mathbf{ip} + \mathbf{2} \times \mathbf{ip}]$ – double *Output*

On exit: **p** contains details of the singular value decomposition if used.

If **svd** = Nag_FALSE, **p** is not referenced.

If **svd** = Nag_TRUE, the first **ip** elements of **p** will not be referenced, the next **ip** values contain the singular values. The following $\mathbf{ip} \times \mathbf{ip}$ values contain the matrix P^* stored by rows.

13: **tol** – double *Input*

On entry: the value of **tol** is used to decide if the independent variables are of full rank and, if not, 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 inaccuracies if the independent variables are not of full rank.

Suggested value: **tol** = 0.000001.

Constraint: **tol** ≥ 0.0.

14: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **n** = ⟨*value*⟩ while **ip** = ⟨*value*⟩. These arguments must satisfy **n** ≥ **ip**.

On entry, **tdq** = ⟨*value*⟩ while **ip** + 1 = ⟨*value*⟩. These arguments must satisfy **tdq** ≥ **ip** + 1.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_DOF_LE_ZERO

The degrees of freedom for error are less than or equal to 0. In this case the estimates, $\hat{\beta}$, are returned but not the standard errors or covariances.

NE_INT_ARG_LT

On entry, **ip** = ⟨*value*⟩.
Constraint: **ip** ≥ 1.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq$ 1.

NE_REAL_ARG_LT

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

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge. This is an unlikely error exit.

7 Accuracy

The accuracy of the results will depend on the accuracy of the input R matrix, which may lose accuracy if a large number of observations or variables have been dropped.

8 Parallelism and Performance

nag_regsn_mult_linear_upd_model (g02ddc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

A dataset consisting of 12 observations and four independent variables is input and a regression model fitted by calls to nag_regsn_mult_linear_add_var (g02dec). The arguments are then calculated by nag_regsn_mult_linear_upd_model (g02ddc) and the results printed.

10.1 Program Text

```
/* nag_regsn_mult_linear_upd_model (g02ddc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define X(I, J) x[(I) *tdx + J]
#define Q(I, J) q[(I) *tdq + J]
int main(void)
{
  Integer exit_status = 0, i, ip, ipmax, j, m, n, rank, tdq, tdx;
  double *b = 0, *cov = 0, df, *p = 0, *q = 0, rss, *se = 0, tol, *wt = 0;
  double *wtptr, *x = 0, *xe = 0;
  char nag_enum_arg[40];
  Nag_Boolean svd, weight;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_regsn_mult_linear_upd_model (g02ddc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
```

```
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT " %" NAG_IFMT " %39s", &n, &m, nag_enum_arg,
          (unsigned)_countof(nag_enum_arg));
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " %39s", &n, &m, nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
  ipmax = 4;
  if (n >= 1 && m >= 1) {
    if (!(b = NAG_ALLOC(ipmax, double)) ||
        !(cov = NAG_ALLOC(ipmax * (ipmax + 1) / 2, double)) ||
        !(p = NAG_ALLOC(ipmax * (ipmax + 2), double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(x = NAG_ALLOC(n * m, double)) ||
        !(xe = NAG_ALLOC(n, double)) ||
        !(se = NAG_ALLOC(ipmax, double)) ||
        !(q = NAG_ALLOC(n * (ipmax + 1), double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdx = m;
    tdq = ipmax + 1;
  }
  else {
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
  if (weight)
    wtptr = wt;
  else
    wtptr = (double *) 0;

  if (wtptr) {
    for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
#ifdef _WIN32
        scanf_s("%lf", &X(i, j));
#else
        scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
      scanf_s("%lf%lf", &Q(i, 0), &wt[i]);
#else
      scanf("%lf%lf", &Q(i, 0), &wt[i]);
#endif
    }
  }
  else {
    for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
#ifdef _WIN32
        scanf_s("%lf", &X(i, j));
#else
        scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
      scanf_s("%lf", &Q(i, 0));
#else
      scanf("%lf", &Q(i, 0));
#endif
    }
  }
  /* Set tolerance */
```

```
  tol = 0.000001e0;
  ip = 0;
  for (j = 0; j < m; ++j) {
    /*
     *        Fit model using g02dec
     */
    for (i = 0; i < n; i++)
      xe[i] = X(i, j);
    /* nag_regsn_mult_linear_add_var (g02dec).
     * Add a new independent variable to a general linear
     * regression model
     */
    nag_regsn_mult_linear_add_var(n, ip, q, tdq, p, wtptr, xe, &rss, tol,
                                  &fail);
    if (fail.code == NE_NOERROR)
      ip += 1;
    else if (fail.code == NE_NVAR_NOT_IND)
      printf(" * New variable not added * \n");
    else {
      printf("Error from nag_regsn_mult_linear_add_var (g02dec).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
  }
  rss = 0.0;
  /* nag_regsn_mult_linear_upd_model (g02ddc).
   * Estimates of regression parameters from an updated model
   */
  nag_regsn_mult_linear_upd_model(n, ip, q, tdq, &rss, &df, b, se, cov, &svd,
                                  &rank, p, tol, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_regsn_mult_linear_upd_model (g02ddc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\n");
  if (svd)
    printf("Model not of full rank\n\n");
  printf("Residual sum of squares = %13.4e\n", rss);
  printf("Degrees of freedom = %3.1f\n\n", df);
  printf("Variable    Parameter estimate   Standard error\n\n");
  for (j = 0; j < ip; j++)
    printf("%6" NAG_IFMT "%20.4e%20.4e\n", j + 1, b[j], se[j]);
  printf("\n");

END:
  NAG_FREE(b);
  NAG_FREE(cov);
  NAG_FREE(p);
  NAG_FREE(wt);
  NAG_FREE(x);
  NAG_FREE(xe);
  NAG_FREE(se);
  NAG_FREE(q);

  return exit_status;
}
```

10.2 Program Data

```
nag_regsn_mult_linear_upd_model (g02ddc) Example Program Data
 12 4 Nag_FALSE
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
```

10.3 Program Results

```
nag_regsn_mult_linear_upd_model (g02ddc) Example Program Results

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

Variable    Parameter estimate   Standard error

     1          3.6003e+01          9.6235e-01
     2          3.7300e+01          9.6235e-01
     3          4.1603e+01          9.6235e-01
     4          3.7877e+01          9.6235e-01
```