NAG Library Function Document

nag_regsn_mult_linear_delete_var (g02dfc)

1 Purpose

nag_regsn_mult_linear_delete_var (g02dfc) deletes an independent variable from a general linear regression model.

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_regsn_mult_linear_delete_var (Integer ip, double q[], Integer tdq,
     Integer indx, double *rss, NagError *fail)
```

3 Description

When selecting a linear regression model it is sometimes useful to drop independent variables from the model and to examine the resulting sub-model. nag_regsn_mult_linear_delete_var (g02dfc) updates the QR decomposition used in the computation of the linear regression model. The QR decomposition may come from nag_regsn_mult_linear (g02dac), nag_regsn_mult_linear_addrem_obs (g02dcc), nag_regsn_mult_linear_add_var (g02dec) or a previous call to nag_regsn_mult_linear_delete_var (g02dfc).

For the general linear regression model with p independent variables fitted, nag_regsn_mult_linear (g02dac) or nag_regsn_mult_linear_add_var (g02dec) computes a QR decomposition of the (weighted) independent variables and forms an upper triangular matrix R and a vector c. To remove an independent variable R and c have to be updated. The column of R corresponding to the variable to be dropped is removed and the matrix is then restored to upper triangular form by applying a series of Givens rotations. The rotations are then applied to c. Note that only the first p elements of c are affected.

The method used means that while the updated values of R and c are computed an updated value of Q from the QR decomposition is not available so a call to nag_regsn_mult_linear_add_var (g02dec) cannot be made after a call to nag_regsn_mult_linear_delete_var (g02dfc).

nag_regsn_mult_linear_upd_model (g02ddc) can be used to calculate the parameter estimates, $\hat{\beta}$, from the information provided by nag_regsn_mult_linear_delete_var (g02dfc).

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

5 Arguments

1: **ip** – Integer *Input*

On entry: the number of independent variables already in the model, p.

Constraint: **ip** ≥ 1.

2: **q**[**ip** $\times$ **tdq**] – double *Input/Output*

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

On entry: the results of the QR decomposition as returned by nag_regsn_mult_linear (g02dac), nag_regsn_mult_linear_addrem_obs (g02dcc), nag_regsn_mult_linear_add_var (g02dec) or previous calls to nag_regsn_mult_linear_delete_var (g02dfc).

On exit: the updated QR decomposition. The first **ip** elements of the first column of **q** contain the updated value of c, the upper triangular part of columns 2 to **ip** contain the updated R matrix.

3: **tdq** – Integer *Input*

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

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

4: **indx** – Integer *Input*

On entry: indicates which independent variable is to be deleted from the model.

Constraint: $1 \leq$ **indx** $\leq$ **ip**.

5: **rss** – double * *Input/Output*

On entry: the residual sum of squares for the full regression.

Constraint: **rss** ≥ 0.0.

On exit: the residual sum of squares with the (**indx**)th variable removed. Note that the residual sum of squares will only be valid if the regression is of full rank.

6: **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_GT

On entry, **indx** $= \langle value \rangle$ while **ip** $= \langle value \rangle$. These arguments must satisfy **indx** $\leq$ **ip**.

NE_2_INT_ARG_LT

On entry, **tdq** $= \langle value \rangle$ while **ip** $+ 1 = \langle value \rangle$. These arguments must satisfy **tdq** $\geq$ **ip** $+ 1$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_DIAG_ELEM_ZERO

On entry, a diagonal element, $\langle value \rangle$, of R is zero.

NE_INT_ARG_LT

On entry, **indx** $= \langle value \rangle$.
Constraint: **indx** ≥ 1.

On entry, **ip** $= \langle value \rangle$.
Constraint: **ip** ≥ 1.

NE_REAL_ARG_LT

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

7 Accuracy

There will inevitably be some loss in accuracy in fitting a model by dropping terms from a more complex model rather than fitting it afresh using nag_regsn_mult_linear (g02dac).

8 Parallelism and Performance

nag_regsn_mult_linear_delete_var (g02dfc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

A dataset consisting of 12 observations on four independent variables and one dependent variable is read in. The full model, including a mean term, is fitted using nag_regsn_mult_linear (g02dac). The value of **indx** is read in and that variable dropped from the regression. The parameter estimates are calculated by nag_regsn_mult_linear_upd_model (g02ddc) and printed. This process is repeated until **indx** is 0.

10.1 Program Text

```
/* nag_regsn_mult_linear_delete_var (g02dfc) 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]
int main(void)
{
  Integer exit_status = 0, i, indx, ip, ipmax, j, m, n, rank, *sx = 0,
          tdq, tdx;
  char nag_enum_arg[40];
  double *b = 0, *com_ar = 0, *cov = 0, df, *h = 0, *p = 0, *q = 0;
  double *res = 0, rss, *se = 0, tol, *wt = 0, *wtptr, *x = 0, *y = 0;
  Nag_Boolean svd, weight;
  Nag_IncludeMean mean;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_regsn_mult_linear_delete_var (g02dfc) 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 "", &n, &m);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT "", &n, &m);
#endif
#ifdef _WIN32
```

```
  scanf_s(" %39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf(" %39s", 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);
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  mean = (Nag_IncludeMean) nag_enum_name_to_value(nag_enum_arg);
  ipmax = m + 1;
  if (n >= 1 && m >= 1) {
    if (!(h = NAG_ALLOC(n, double)) ||
        !(res = NAG_ALLOC(n, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(x = NAG_ALLOC((n) * (m), double)) ||
        !(y = NAG_ALLOC(n, double)) ||
        !(sx = NAG_ALLOC(m, Integer)) ||
        !(b = NAG_ALLOC(ipmax, double)) ||
        !(cov = NAG_ALLOC(ipmax * (ipmax + 1) / 2, double)) ||
        !(p = NAG_ALLOC(ipmax * (ipmax + 2), double)) ||
        !(q = NAG_ALLOC((n) * (ipmax + 1), double)) ||
        !(se = NAG_ALLOC(ipmax, double)) ||
        !(com_ar = NAG_ALLOC(5 * (ipmax - 1) + ipmax * ipmax, 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", &y[i], &wt[i]);
#else
      scanf("%lf%lf", &y[i], &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", &y[i]);
#else
      scanf("%lf", &y[i]);
#endif
    }
  }
  for (i = 0; i < m; ++i)
    sx[i] = 1;
  ip = m;
  if (mean == Nag_MeanInclude)
    ip += 1;
  /* Set tolerance  */
  tol = 0.00001e0;
  /* nag_regsn_mult_linear (g02dac).
   * Fits a general (multiple) linear regression model
   */
  nag_regsn_mult_linear(mean, n, x, tdx, m, sx, ip, y, wtptr, &rss,
                        &df, b, se, cov, res, h, q, tdq, &svd, &rank,
                        p, tol, com_ar, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_regsn_mult_linear (g02dac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("Results from full model\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);
#ifdef _WIN32
  while (scanf_s("%" NAG_IFMT "", &indx) != EOF)
#else
  while (scanf("%" NAG_IFMT "", &indx) != EOF)
#endif
  {
    if (indx != 0) {
      /* nag_regsn_mult_linear_delete_var (g02dfc).
       * Delete an independent variable from a general linear
       * regression model
       */
      nag_regsn_mult_linear_delete_var(ip, q, tdq, indx, &rss, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_regsn_mult_linear_delete_var (g02dfc)."
               "\n%s\n", fail.message);
        exit_status = 1;
        goto END;
      }

      ip = ip - 1;
      if (ip == 0)
        printf("No terms left in model\n");
      else {
        printf("Variable %4" NAG_IFMT " dropped\n", indx);
        /* 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("Residual sum of squares = %13.4e\n", rss);
        printf("Degrees of freedom = %3.1f\n\n", df);
        printf("Parameter estimate   Standard error\n\n");
        for (j = 0; j < ip; j++)
          printf("%15.4e%15.4e\n", b[j], se[j]);
```

```
      }
    }
  }
END:
  NAG_FREE(h);
  NAG_FREE(res);
  NAG_FREE(wt);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(sx);
  NAG_FREE(b);
  NAG_FREE(cov);
  NAG_FREE(p);
  NAG_FREE(q);
  NAG_FREE(se);
  NAG_FREE(com_ar);
  return exit_status;
}
```

10.2 Program Data

```
nag_regsn_mult_linear_delete_var (g02dfc) Example Program Data
 12 4 Nag_FALSE Nag_MeanInclude
1.0 1.4 0.0 0.0 4.32
1.5 2.2 0.0 0.0 5.21
2.0 4.5 0.0 0.0 6.49
2.5 6.1 0.0 0.0 7.10
3.0 7.1 0.0 0.0 7.94
3.5 7.7 0.0 0.0 8.53
4.0 8.3 1.0 4.0 8.84
4.5 8.6 1.0 4.5 9.02
5.0 8.8 1.0 5.0 9.27
5.5 9.0 1.0 5.5 9.43
6.0 9.3 1.0 6.0 9.68
6.5 9.2 1.0 6.5 9.83
2
4
0
```

10.3 Program Results

```
nag_regsn_mult_linear_delete_var (g02dfc) Example Program Results
Results from full model
Residual sum of squares =    8.4066e-02
Degrees of freedom = 7.0

Variable    2 dropped
Residual sum of squares =    2.1239e-01
Degrees of freedom = 8.0

Parameter estimate   Standard error

     3.6372e+00     1.5083e-01
     6.1264e-01     2.8007e-02
    -6.0154e-01     4.2335e-01
     1.6709e-01     7.8656e-02
Variable    4 dropped
Residual sum of squares =    3.3220e-01
Degrees of freedom = 9.0

Parameter estimate   Standard error

     3.5974e+00     1.7647e-01
     6.2088e-01     3.2706e-02
     2.4247e-01     1.7235e-01
```