NAG Library Function Document

nag_regsn_mult_linear_addrem_obs (g02dcc)

1 Purpose

nag_regsn_mult_linear_addrem_obs (g02dcc) adds or deletes an observation from a general regression model fitted by nag_regsn_mult_linear (g02dac).

2 Specification

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

void nag_regsn_mult_linear_addrem_obs (Nag_UpdateObserv update,
                                       Nag_IncludeMean mean, Integer m, const Integer sx[], double q[],
                                       Integer tdq, Integer ip, const double x[], Integer nr, Integer tdx,
                                       Integer ix, double y, const double wt[], double *rss, NagError *fail)
```

3 Description

nag_regsn_mult_linear (g02dac) fits a general linear regression model to a dataset. You may wish to change the model by either adding or deleting an observation from the dataset. nag_regsn_mult_linear_addrem_obs (g02dcc) takes the results from nag_regsn_mult_linear (g02dac) and makes the required changes to the vector $c$ and the upper triangular matrix $R$ produced by nag_regsn_mult_linear (g02dac). The regression coefficients, standard errors and the variance-covariance matrix of the regression coefficients can be obtained from nag_regsn_mult_linear_upd_model (g02ddc) after all required changes to the dataset have been made.

nag_regsn_mult_linear (g02dac) performs a QR decomposition on the (weighted) $X$ matrix of independent variables. To add a new observation to a model with $p$ arguments the upper triangular matrix $R$ and vector $c_1$, the first $p$ elements of $c$, are augmented by the new observation on independent variables in $x^T$ and dependent variable $y$. Givens rotations are then used to restore the upper triangular form.

\[
\begin{pmatrix}
  R \\
  c_1 \\
  x & y
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  R^* & c_1^* \\
  y^* & 0
\end{pmatrix}
\]

To delete an observation Givens rotations are applied to give:

\[
\begin{pmatrix}
  R & c_1 \\
  x & y
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  R^* & c_1^* \\
  x & y
\end{pmatrix}
\]

Note: only the $R$ and upper part of the $c$ are updated, the remainder of the $Q$ matrix is unchanged.

4 References


5 Arguments

1: **update** – Nag_UpdateObserv
   *Input*
   
   *On entry:* indicates if an observation is to be added or deleted.

   **update** = Nag_ObservAdd
   The observation is added.

   **update** = Nag_ObservDel
   The observation is deleted.

   *Constraint:* **update** = Nag_ObservAdd or Nag_ObservDel.

2: **mean** – Nag.IncludeMean
   *Input*
   
   *On entry:* indicates if a mean has been used in the model.

   **mean** = Nag_MeanInclude
   A mean term or intercept will have been included in the model by nag_regsn_mult_linear (g02dac).

   **mean** = Nag_MeanZero
   A model with no mean term or intercept will have been fitted by nag_regsn_mult_linear (g02dac).

   *Constraint:* **mean** = Nag_MeanInclude or Nag_MeanZero.

3: **m** – Integer
   *Input*
   
   *On entry:* the total number of independent variables in the dataset.

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

4: **sx[\|m\|\times const\ Integer**
   *Input*
   
   *On entry:* if **sx[j]** is greater than 0, then the value contained in $x[\text{ix} \times (\text{ix} - 1) + j]$ is to be included as a value of $x^T$, an observation on an independent variable, for $j = 0, 1, \ldots, m - 1$.

   *Constraint:* if **mean** = Nag_MeanInclude, then exactly $\|ip\| - 1$ elements of **sx** must be $> 0$ and if **mean** = Nag_MeanZero, then exactly **ip** elements of **sx** must be $> 0$.

5: **q[\|ip\|\times tdq]** – double
   *Input/Output*
   
   *Note:* the $(i, j)$th element of the matrix $Q$ is stored in $q[(i - 1) \times \text{tdq} + j - 1]$.

   *On entry:* **q** must be array **q** as output by nag_regsn_mult_linear (g02dac), nag_regsn_mult_linear_add_var (g02dec), nag_regsn_mult_linear_delete_var (g02dfc), or a previous call to nag_regsn_mult_linear_addrem_obs (g02dcc).

   *On exit:* the first $\|ip\| = \|ip\| + 1$ will contain $R^*$, the upper triangular part of columns 2 to $\|ip\| + 1$ will contain $R^*$, the remainder is unchanged.

6: **tdq** – Integer
   *Input*
   
   *On entry:* the stride separating matrix column elements in the array **q**.

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

7: **ip** – Integer
   *Input*
   
   *On entry:* the number of linear terms in general linear regression model (including mean if there is one).

   *Constraint:* **ip** $\geq 1$. 
8: $x[\text{nr} \times \text{tdx}]$ – const double

*Input*

*On entry:* the $\text{ip}$ values for the dependent variables of the observation to be added or deleted, $x^T$. The positions of the values $x$ extracted depends on $\text{ix}$ and $\text{tdx}$.

9: $\text{nr}$ – Integer

*Input*

*On entry:* the number of rows of the notional two-dimensional array $x$.

*Constraint:* $\text{nr} \geq 1$.

10: $\text{tdx}$ – Integer

*Input*

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

*Constraint:* $\text{tdx} \geq m$.

11: $\text{ix}$ – Integer

*Input*

*On entry:* the row of the notional two-dimensional array $x$ that contains the values for the dependent variables of the observation to be added or deleted.

*Constraint:* $1 \leq \text{ix} \leq \text{nr}$.

12: $y$ – double

*Input*

*On entry:* the value of the dependent variable for the observation to be added or deleted, $y$.

13: $\text{wt}[1]$ – const double

*Input*

*On entry:* if the new observation is to be weighted, then $\text{wt}$ must contain the weight to be used with the new observation. If $\text{wt}[0] = 0.0$, then the observation is not included in the model. If the new observation is to be unweighted, then $\text{wt}$ must be supplied as NULL.

*Constraint:* if the new observation is to be weighted $\text{wt}[0] \geq 0.0$.

14: $\text{rss}$ – double *

*Input/Output*

*On entry:* the value of the residual sums of squares for the original set of observations.

*Constraint:* $\text{rss} \geq 0.0$.

*On exit:* the updated values of the residual sums of squares.

*Note:* this will only be valid if the model is of full rank.

15: $\text{fail}$ – NagError *

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

**NE_2_INT_ARG_GT**

On entry, $\text{ix} = \langle\text{value}\rangle$ while $\text{nr} = \langle\text{value}\rangle$. These arguments must satisfy $\text{ix} \leq \text{nr}$.

**NE_2_INT_ARG_LT**

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

On entry, $\text{tdx} = \langle\text{value}\rangle$ while $m = \langle\text{value}\rangle$. These arguments must satisfy $\text{tdx} \geq m$.

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
NE_BAD_PARAM
On entry, mean had an illegal value.
On entry, update had an illegal value.

NE_INT_ARG_LT
On entry, ip = {value}.
Constraint: ip \geq 1.
On entry, ix = {value}.
Constraint: ix \geq 1.
On entry, m = {value}.
Constraint: m \geq 1.
On entry, nr = {value}.
Constraint: nr \geq 1.

NE_IP_INCOMP_WITH_SX
On entry, for mean = Nag_MeanInclude, number of nonzero values of sx must be equal to ip - 1:
number of nonzero values of sx = {value}, ip - 1 = {value}.
On entry, for mean = Nag_MeanZero, number of nonzero values of sx must be equal to ip:
number of nonzero values of sx = {value}, ip = {value}.

NE_MAT_NOT_UPD
The R matrix could not be updated: to, either, delete nonexistent observation, or, add an
observation to R matrix with zero diagonal element.

NE_REAL_ARG_LT
On entry, rss = {value}.
Constraint: rss \geq 0.0.
On entry, wt[0] = {value}.
Constraint: wt[0] \geq 0.0.

NE_RSS_NOT_UPD
The rss could not be updated because the input rss was less than the calculated decrease in rss
when the new observation was deleted.

7 Accuracy
Higher accuracy is achieved by updating the R matrix rather than the traditional methods of updating
X'X.

8 Parallelism and Performance
Not applicable.

9 Further Comments
Care should be taken with the use of this function.
(a) It is possible to delete observations which were not included in the original model.
(b) If several additions/deletions have been performed you are advised to recompute the regression
using nag_regn_multi_linear (g02dac).
(c) Adding or deleting observations can alter the rank of the model. Such changes will only be detected when a call to \texttt{nag_regsn_mult_linear_upd_model (g02ddc)} has been made. \texttt{nag_regsn_mult_linear_upd_model (g02ddc)} should also be used to compute the new residual sum of squares when the model is not of full rank.

\texttt{nag_regsn_mult_linear_addrem_obs (g02dcc)} may also be used after \texttt{nag_regsn_mult_linear_add_var (g02dec)} and \texttt{nag_regsn_mult_linear_delete_var (g02dfc)}.

10 Example

A dataset consisting of 12 observations with four independent variables is read in and a general linear regression model fitted by \texttt{nag_regsn_mult_linear (g02dac)} and parameter estimates printed. The last observation is then dropped and the parameter estimates recalculated, using \texttt{nag_regsn_mult_linear_upd_model (g02ddc)}, and printed.

10.1 Program Text

/* \texttt{nag_regsn_mult_linear_addrem_obs (g02dcc)} Example Program. */
* Copyright 2014 Numerical Algorithms Group.
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>
#define XM(I, J) xm[(I) *tdxm + J]
#define Q(I, J) q[(I) *tdq + J]

int main(void)
{
    Integer exit_status = 0, i, ip, j, m, n, rank, *sx = 0, tdq, tdxm;
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, *xm = 0;
double *y = 0;
char nag_enum_arg[40];
Nag_Boolean svd, weight;
Nag_IncludeMean mean;
Nag_UpdateObserv update;
NagError fail;
INIT_FAIL(fail);

printf("\texttt{nag_regsn_mult_linear_addrem_obs (g02dcc)} Example Program }\n"\texttt{Results\n});
/* Skip heading in data file */
#ifndef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
#ifndef _WIN32
    scanf_s("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#else
    scanf("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#endif
#ifndef _WIN32
    scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s", nag_enum_arg);
#endif
/* \texttt{nag_enum_name_to_value (x04nac)}. */
* Converts NAG enum member name to value */
weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
#else
scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s", nag_enum_arg);
#endif
mean = (Nag_IncludeMean) nag_enum_name_to_value(nag_enum_arg);
if (weight)
   wtptr = wt;
else
   wtptr = (double *) 0;
if (n >= 2 && m >= 1)
{
   if (!(b = NAG_ALLOC(m, double)) ||
      !(h = NAG_ALLOC(n, double)) ||
      !(res = NAG_ALLOC(n, double)) ||
      !(wt = NAG_ALLOC(n, double)) ||
      !(xm = NAG_ALLOC(n*m, double)) ||
      !(y = NAG_ALLOC(n, double)) ||
      !(sx = NAG_ALLOC(m, Integer)))
   {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
   }
   tdxm = m;
} else
{
   printf("Invalid n or m.\n");
   exit_status = 1;
   return exit_status;
}
if (wtptr)
{
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
#ifdef _WIN32
      scanf_s("%lf", &XM(i, j));
#else
      scanf("%lf", &XM(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", &XM(i, j));
#else
      scanf("%lf", &XM(i, j));
#endif
      #ifdef _WIN32
      scanf_s("%lf", &y[i]);
#else
      scanf("%lf", &y[i]);
#endif
   }
   for (j = 0; j < m; j++)
#ifdef _WIN32
      scanf_s("%lf", &XM(i));
#else
      scanf("%lf", &XM(i));
#endif
      #ifdef _WIN32
      scanf_s("%lf", &y[i]);
#else
      scanf("%lf", &y[i]);
#endif
   }
   for (j = 0; j < m; ++j)
scanf_s("%"NAG_IFMT", &sx[j]);
#else
    scanf("%"NAG_IFMT", &sx[j]);
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT", &ip);
#else
    scanf("%"NAG_IFMT", &ip);
#endif
if (!(cov = NAG_ALLOC(ip*(ip+1)/2, double)) ||
    !(p = NAG_ALLOC(ip*(ip+2), double)) ||
    !(q = NAG_ALLOC((n)*(ip+1), double)) ||
    !(com_ar = NAG_ALLOC(5*(ip-1)+ip*ip, double)) ||
    !(se = NAG_ALLOC(ip, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

tdq = ip+1;
/* Set tolerance */
tol = 0.00001e0;

/* Fit initial model using nag_regsn_mult_linear (g02dac) */
/* nag_regsn_mult_linear (g02dac).
* Fits a general (multiple) linear regression model */
    nag_regsn_mult_linear(mean, n, xm, tdxm, m, sx, ip, y, wtptr, &rss,
                        sdf, 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 g02dac\n");
if (svd)
    printf("Model not of full rank\n");
printf("Residual sum of squares = %13.4e\n", rss);
printf("Degrees of freedom = %3.1f\n", df);
printf("Variable Parameter estimate Standard error\n");
for (j = 0; j < ip; j++)
    printf("%6"NAG_IFMT"%20.4e%20.4e\n", j+1, b[j], se[j]);

/* Add/delete an observation to/from a general linear */
/* regression model */
    nag_regsn_mult_linear_addrem_obs(update, mean, m, sx, q, tdq, ip,
                                    xm, n, tdxm, n, y[11], wtptr, &rss, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_regsn_mult_linear_addrem_obs (g02dccc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("Results from dropping an observation using\n" "nag_regsn_mult_linear_addrem_obs (g02dccc)\n");
n = n - 1;
/* nag_regsn_mult_linear_upd_model (g02dcc).
* Estimates of regression parameters from an updated model */
    nag_regsn_mult_linear_upd_model(n, ip, q, tdq, &rss, &sdf, b, se, cov,
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("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]);

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

10.2 Program Data

nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program Data
12 4 Nag_FALSE Nag_MeanZero
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 1.0 0.0 38.99
0.0 0.0 0.0 1.0 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
1.0 0.0 0.0 0.0 37.89
1 1 1 1 4

10.3 Program Results

nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program Results
Results from g02dac
Residual sum of squares = 5.2748e+03
Degrees of freedom = 8.0

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter estimate</th>
<th>Standard error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0724e+01</td>
<td>1.3801e+01</td>
</tr>
<tr>
<td>2</td>
<td>1.4085e+01</td>
<td>1.6240e+01</td>
</tr>
<tr>
<td>3</td>
<td>2.6324e+01</td>
<td>1.3801e+01</td>
</tr>
<tr>
<td>4</td>
<td>2.2597e+01</td>
<td>1.3801e+01</td>
</tr>
</tbody>
</table>

Results from dropping an observation using
nag_regsn_mult_linear_addrem_obs (g02dcc)
Residual sum of squares = 2.1705e+01
Degrees of freedom = 7.0
<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter estimate</th>
<th>Standard error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.6003e+01</td>
<td>1.0166e+00</td>
</tr>
<tr>
<td>2</td>
<td>3.7005e+01</td>
<td>1.2451e+00</td>
</tr>
<tr>
<td>3</td>
<td>4.1603e+01</td>
<td>1.0166e+00</td>
</tr>
<tr>
<td>4</td>
<td>3.7877e+01</td>
<td>1.0166e+00</td>
</tr>
</tbody>
</table>