1 Purpose

nag_regsn_mult_linear_add_var (g02dec) adds a new independent variable to a general linear regression model.

2 Specification

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

void nag_regsn_mult_linear_add_var (Integer n, Integer ip, double q[ ],
                                   Integer tdq, double p[ ], const double wt[ ], const double x[ ],
                                   double *rss, double tol, NagError *fail)
```

3 Description

A linear regression model may be built up by adding new independent variables to an existing model. nag_regsn_mult_linear_add_var (g02dec) updates the QR decomposition used in the computation of the linear regression model. The QR decomposition may come from nag_regsn_mult_linear (g02dac) or a previous call to nag_regsn_mult_linear_add_var (g02dec). The general linear regression model is defined by:

\[ y = X\beta + \epsilon \]

where \( y \) is a vector of \( n \) observations on the dependent variable, \( X \) is an \( n \) by \( p \) matrix of the independent variables of column rank \( k \), \( \beta \) is a vector of length \( p \) of unknown arguments, and \( \epsilon \) is a vector of length \( n \) of unknown random errors such that \( \text{var} \epsilon = V\sigma^2 \), where \( V \) is a known diagonal matrix.

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 arguments \( \beta \) minimize \( (y - X\beta)^T(y - X\beta) \) while the weighted least squares estimates minimize \( (y - X\beta)^TW(y - X\beta) \).

The parameter estimates may be found by computing a QR decomposition of \( X \) (or \( W^{1/2}X \) in the weighted case), i.e.,

\[ X = QR^* \quad \text{or} \quad W^{1/2}X = QR^* \]

where \( R^* = \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix} \) 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 \( c = QTy \) (or \( QTW^{1/2}y \)) 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 \( R \). To add a new independent variable, \( x_{p+1} \), \( R \) and \( c \) have to be updated. The matrix \( Q_{p+1} \) is found such that \( QT_{p+1}[R:QTx_{p+1}] \) (or \( QT_{p+1}[R:QTW^{1/2}x_{p+1}] \)) is upper triangular. The vector \( c \) is then updated by multiplying by \( QT_{p+1} \).
The new independent variable is tested to see if it is linearly related to the existing independent variables by checking that at least one of the values \((Q^T x_{p+1})_i\), for \(i = p+2, p+3, \ldots, n\) is nonzero.

The new parameter estimates, \(\hat{\beta}\), can then be obtained by a call to nag_regsn_mult_linear_upd_model (g02ddc).

The function can be used with \(p = 0\), in which case \(R\) and \(c\) are initialized.

### 4 References


Searle S R (1971) *Linear Models* Wiley

### 5 Arguments

1: \(n\) – Integer

*Input*

*On entry:* the number of observations, \(n\).

*Constraint:* \(n \geq 1\).

2: \(ip\) – Integer

*Input*

*On entry:* the number of independent variables already in the model, \(p\).

*Constraint:* \(ip \geq 0\) and \(ip < n\).

3: \(q[n \times tdq]\) – 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:* if \(ip \neq 0\), then \(q\) must contain the results of the \(QR\) decomposition for the model with \(p\) arguments as returned by nag_regsn_mult_linear (g02dac) or a previous call to nag_regsn_mult_linear_add_var (g02dec).

If \(ip = 0\), then the first column of \(q\) should contain the \(n\) values of the dependent variable, \(y\).

*On exit:* the results of the \(QR\) decomposition for the model with \(p + 1\) arguments: the first column of \(q\) contains the updated value of \(c\), the columns 2 to \(ip + 1\) are unchanged, the first \(ip + 1\) elements of column \(ip + 2\) contain the new column of \(R\), while the remaining \(n - ip - 1\) elements contain details of the matrix \(Q_{p+1}\).

4: \(tdq\) – Integer

*Input*

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

*Constraint:* \(tdq \geq ip + 2\).

5: \(p[ip + 1]\) – double

*Input/Output*

*On entry:* \(p\) contains further details of the \(QR\) decomposition used. The first \(ip\) elements of \(p\) must contain details of the Householder vector from the \(QR\) decomposition. The first \(ip\) elements of array \(p\) are provided by nag_regsn_mult_linear (g02dac) or by previous calls to nag_regsn_mult_linear_add_var (g02dec).

*On exit:* the first \(ip\) elements of \(p\) are unchanged and the \((ip + 1)\)th element contains details of the Householder vector related to the new independent variable.
6: \( \text{wt}[n] \) – const double
   \text{Input}
   \text{On entry}: optionally, the weights to be used in the weighted regression.

   If \( \text{wt}[i-1] = 0.0 \), then the \( i \)th observation is not included in the model, in which case the effective number of observations is the number of observations with nonzero weights.

   If weights are not provided then \( \text{wt} \) must be set to NULL and the effective number of observations is \( n \).

   \text{Constraint}: if \( \text{wt} \) is not NULL, \( \text{wt}[i-1] = 0.0 \), for \( i = 1, 2, \ldots, n \).

7: \( x[n] \) – const double
   \text{Input}
   \text{On entry}: the new independent variable, \( x \).

8: \( \text{rss} \) – double *
   \text{Output}
   \text{On exit}: the residual sum of squares for the new fitted model.

   \text{Note}: this will only be valid if the model is of full rank, see Section 9.

9: \( \text{tol} \) – double
   \text{Input}
   \text{On entry}: the value of \( \text{tol} \) is used to decide if the new independent variable is linearly related to independent variables already included in the model. If the new variable is linearly related then \( c \) is not updated. The smaller the value of \( \text{tol} \) the stricter the criterion for deciding if there is a linear relationship.

   \text{Suggested value}: \( \text{tol} = 0.000001 \).

   \text{Constraint}: \( \text{tol} > 0.0 \).

10: \( \text{fail} \) – NagError *
    \text{Input/Output}
    The NAG error argument (see Section 3.6 in the Essential Introduction).

\section{Error Indicators and Warnings}

\textbf{NE_2_INT_ARG_GE}

On entry, \( \text{ip} = \langle \text{value} \rangle \) while \( n = \langle \text{value} \rangle \). These arguments must satisfy \( \text{ip} < n \).

\textbf{NE_2_INT_ARG_LT}

On entry, \( \text{tdq} = \langle \text{value} \rangle \) while \( \text{ip} + 2 = \langle \text{value} \rangle \). These arguments must satisfy \( \text{tdq} \geq \text{ip} + 2 \).

\textbf{NE_INT_ARG_LT}

On entry, \( \text{ip} = \langle \text{value} \rangle \).

\text{Constraint}: \( \text{ip} \geq 0 \).

On entry, \( n = \langle \text{value} \rangle \).

\text{Constraint}: \( n \geq 1 \).

\textbf{NE_NVAR_NOT_IND}

The new independent variable is a linear combination of existing variables. The \((\text{ip}+1)\)th column of \( q \) is, therefore, NULL.

\textbf{NE_REAL_ARG_LE}

On entry, \( \text{tol} \) must not be less than or equal to 0.0: \( \text{tol} = \langle \text{value} \rangle \).

\textbf{NE_REAL_ARG_LT}

On entry, \( \text{wt}\{\langle \text{value} \rangle \} \) must not be less than 0.0: \( \text{wt}\{\langle \text{value} \rangle \} = \langle \text{value} \rangle \).
7 Accuracy

The accuracy is closely related to the accuracy of the QR decomposition.

8 Parallelism and Performance

Not applicable.

9 Further Comments

It should be noted that the residual sum of squares produced by nag_regsn_mult_linear_add_var (g02dec) may not be correct if the model to which the new independent variable is added is not of full rank. In such a case nag_regsn_mult_linear_upd_model (g02ddc) should be used to calculate the residual sum of squares.

10 Example

A dataset consisting of 12 observations is read in. The four independent variables are stored in the array x while the dependent variable is read into the first column of q. If the character variable meanc indicates that a mean should be included in the model, a variable taking the value 1.0 for all observations is set up and fitted. Subsequently, one variable at a time is selected to enter the model as indicated by the input value of indx. After the variable has been added the parameter estimates are calculated by nag_regsn_mult_linear_upd_model (g02ddc) and the results printed. This is repeated until the input value of indx is 0.

10.1 Program Text

/* nag_regsn_mult_linear_add_var (g02dec) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 */

#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, indx, ip, ipmax, j, m, n, rank, tdq, tdx;
    char nag_enum_arg[40];
    double df, rss, rsst, tol;
    double *b = 0, *cov = 0, *p = 0, *q = 0, *se = 0, *wt = 0, *wtptr;
    double *x = 0, *xe = 0;
    Nag_Boolean svd, weight;
    Nag_IncludeMean mean;
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_regsn_mult_linear_add_var (g02dec) Example Program Results\n");
    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[^
]");
#else
    scanf("%*[^
]");
#endif
#ifdef _WIN32
    g02dec
#else
    nag_regsn_mult_linear_add_var
#endif

    /* Read in data and fit model */
    ...

    /* Calculate parameter estimates */
    /* nag_regsn_mult_linear_upd_model */
    ...

    /* Print results */
    printf("nag_regsn_mult_linear_add_var (g02dec) Example Program Results\n");
    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[^
]");
#else
    scanf("%*[^
]");
#endif
#ifdef _WIN32
    g02dec
#else
    nag_regsn_mult_linear_add_var
#endif

    return exit_status;
}
scanf_s("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#else
    scanf("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#endif
#define WIN32
    scanf_s(" %39s", nag_enum_arg, _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);
#define WIN32
    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);
#endif
ipmax = 5;
if (n >= 1 && m >= 1)
{
    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
    }
    for (i = 0; i < n; i++)
        #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++)
            g02 – Correlation and Regression Analysis
            g02dec
            Mark 25 g02dec.5


```c
{ 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 */
const double tol = 0.000001e0;
ip = 0;
if (mean == Nag_MeanInclude)
{
    for (i = 0; i < n; ++i)
        xe[i] = 1.0;

    /* 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)
    {
        printf(
            "Error from nag_regsn_mult_linear_add_var (g02dec).\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }
    ip = 1;
}
#endif
while (scanf_s("%"NAG_IFMT"", &indx) != EOF)
#endif
while (scanf("%"NAG_IFMT"", &indx) != EOF)
#endif
{
    if (indx > 0)
    {
        for (i = 0; i < n; ++i)
            xe[i] = X(i, indx-1);
        /* nag_regsn_mult_linear_add_var (g02dec), see above. */
        nag_regsn_mult_linear_add_var(n, ip, q, tdq, p, wtptr, xe, &rss,
                                       tol, &fail);
        if (fail.code == NE_NOERROR)
        {
            ip += 1;
            printf("Variable %4"NAG_IFMT" added\n", indx);
            rsst = 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, &rsst, &df, b, se,
                                             cov, &svd, &rank, p, tol, &fail);
            if (fail.code != NE_NOERROR)
            {
                printf(
                    "Error from nag_regsn_mult_linear_add_var (g02dec).\n                    "\n                    "\n                    %s\n",
                    fail.message);
                exit_status = 1;
                goto END;
            }
        }
    }
}
```

The above code snippet is from the NAG Library Manual, specifically for the function `g02dec`. It demonstrates how to read data from a file and add new variables to a linear regression model using the `nag_regsn_mult_linear_add_var` function.
if (svd)
    printf("Model not of full rank\n\n");
printf("Residual sum of squares = %13.4e\n", rsst);
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]);
else if (fail.code == NE_NVAR_NOT_IND)
    printf(" * New variable not added *\n");
else
    { }
    printf("Error from nag_regsn_mult_linear_upd_model (g02ddc)." "\n\n", fail.message);
    exit_status = 1;
goto END;
}
END:
NAG_FREE(wt);
NAG_FREE(x);
NAG_FREE(xe);
NAG_FREE(b);
NAG_FREE(cov);
NAG_FREE(p);
NAG_FREE(se);
NAG_FREE(q);
return exit_status;

10.2 Program Data

nag_regsn_mult_linear_add_var (g02dec) 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
1
3
4
2
0

10.3 Program Results

nag_regsn_mult_linear_add_var (g02dec) Example Program Results
Variable 1 added
Residual sum of squares = 4.0164e+00
Degrees of freedom = 10.0
Variable Parameter estimate Standard error
Variable 3 added
Residual sum of squares = 3.8872e+00
Degrees of freedom = 9.0

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter estimate</th>
<th>Standard error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.2236e+00</td>
<td>5.6734e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.0554e+00</td>
<td>2.2217e-01</td>
</tr>
<tr>
<td>3</td>
<td>-4.1962e-01</td>
<td>7.6695e-01</td>
</tr>
</tbody>
</table>

Variable 4 added
Residual sum of squares = 1.8702e-01
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>2.7605e+00</td>
<td>1.7592e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.7057e+00</td>
<td>7.3100e-02</td>
</tr>
<tr>
<td>3</td>
<td>4.4575e+00</td>
<td>4.2676e-01</td>
</tr>
<tr>
<td>4</td>
<td>-1.3006e+00</td>
<td>1.0338e-01</td>
</tr>
</tbody>
</table>

Variable 2 added
Residual sum of squares = 8.4066e-02
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>3.1440e+00</td>
<td>1.8181e-01</td>
</tr>
<tr>
<td>2</td>
<td>9.0748e-01</td>
<td>2.7761e-01</td>
</tr>
<tr>
<td>3</td>
<td>2.0790e+00</td>
<td>8.6804e-01</td>
</tr>
<tr>
<td>4</td>
<td>-6.1589e-01</td>
<td>2.4530e-01</td>
</tr>
<tr>
<td>5</td>
<td>2.9224e-01</td>
<td>9.9810e-02</td>
</tr>
</tbody>
</table>