NAG Library Function Document

nag_blgm_lm_submodel (g22ydc)

Note: please be advised that this function is classed as 'experimental' and its interface may be developed further in the future. Please see Section 3.1.1 in How to Use the NAG Library and its Documentation for further information.

1 Purpose

nag_blgm_lm_submodel (g22ydc) produces labels for the columns of a design matrix, model parameters and a vector of column inclusion flags suitable for use with functions in Chapter g02. Thus allowing for submodels to be fit using the same design matrix.

2 Specification

```
#include <nag.h>
#include <nagg22.h>
void nag_blgm_lm_submodel (void *hform, void *hxdesc,
     Nag_IncludeIntercept *intcpt, Integer *ip, Integer lisx, Integer isx[],
     Integer lplab, const char *plab[], Integer lenlab, Integer lvinfo,
     Integer vinfo[], NagError *fail)
```

3 Description

nag_blgm_lm_submodel (g22ydc) is a utility function for use with **nag_blgm_lm_formula (g22yac)**, **nag_blgm_lm_describe_data (g22ybc)** and **nag_blgm_lm_design_matrix (g22ycc)**. It can be used to construct labels for the columns for an $n \times m_x$ design matrix, X, created by **nag_blgm_lm_design_matrix (g22ycc)** and return additional input vectors and flags required by a number of NAG Library model fitting functions.

Many of the analysis functions that require a design matrix to be supplied allow submodels to be defined through the use of a vector of ones or zeros indicating whether a column of X should be included or excluded from the analyses (see for example **sx** in **nag_regsn_mult_linear (g02dac)** or **nag_glm_normal (g02gac)**). This allows nested models to be fit without having to reconstructed the design matrix for each analysis.

Let $\mathcal{M}$ denote a model constructed by **nag_blgm_lm_formula (g22yac)**, D a data matrix as described by **nag_blgm_lm_describe_data (g22ybc)** and X be the corresponding design matrix constructed by **nag_blgm_lm_design_matrix (g22ycc)** from $\mathcal{M}$ and D. A different model, $\mathcal{M}_S$ is a submodel of $\mathcal{M}$ if each term in $\mathcal{M}_S$, including the mean effect (intercept term) is also present in $\mathcal{M}$.

If $\mathcal{M}_S$ is a submodel of $\mathcal{M}$, you can fit $\mathcal{M}_S$ to D using a design matrix whose columns are a subset of the columns of X.

4 References

None.

5 Arguments

1: **hform** – void * *Input*

On entry: a G22 handle to the internal data structure containing a description of the required (sub)model $\mathcal{M}_S$, as returned in **hform** by **nag_blgm_lm_formula (g22yac)**.

2: **hxdesc** – void * *Input*

On entry: a G22 handle to the internal data structure containing a description of the design matrix, D, as returned in **hxdesc** by **nag_blgm_lm_design_matrix (g22ycc)**.

3: **intcpt** – Nag_IncludeIntercept * *Output*

On exit: if **intcpt** = Nag_Intercept, in order to fit the model $\mathcal{M}_S$ to D using X, any analysis function should include an implicit mean effect (intercept term).

intcpt = Nag_NoIntercept, if $\mathcal{M}_S$ does not include a mean effect or the mean effect has been explicitly included in the design matrix.

4: **ip** – Integer * *Output*

On exit: p, the number of parameters in the model specified in **hform**, including the intercept if one is present.

If **lisx** $\neq 0$, if **intcpt** = Nag_NoIntercept, $p = \sum_{i=1}^{m_x} \mathbf{isx}[i-1]$, otherwise $p = \sum_{i=1}^{m_x} \mathbf{isx}[i-1] + 1$.

5: **lisx** – Integer *Input*

On entry: length of **isx**.

Constraint: **lisx** $= 0$ or **lisx** $\geq m_x$, where m_x is the number of columns in the design matrix X.

6: **isx**[**lisx**] – Integer *Output*

On exit: if **lisx** $\neq 0$, an array indicating which columns of the design matrix form the model specified in **hform**.

isx$[j-1] = 0$
The jth column of the design matrix, X, should not be included in the analysis.

isx$[j-1] = 1$
The jth column of the design matrix, X, should be included in the analysis.

If **lisx** $= 0$, **isx** is not referenced and may be **NULL**.

7: **lplab** – Integer *Input*

On entry: the length of **plab**.

As $p \leq m_x + 1$, if labels are required, using **lplab** $= m_x + 1$ will always be sufficient.

Constraint: **lplab** $= 0$ or **lplab** $\geq p$.

8: **plab**[**lplab**] – const char * *Output*

On exit: if **lplab** $\neq 0$, the names associated with the p parameters in the model.

If **intcpt** = Nag_NoIntercept, the labels in **plab** are also the labels for the columns of design matrix used in the analysis.

If **intcpt** = Nag_Intercept, columns **plab**$[1]$ to **plab**$[p-1]$ are the corresponding column labels.

If a mean effect is present in M_S, the corresponding label is always in **plab**$[0]$.

If **lplab** $= 0$, **plab** is not referenced and may be **NULL**.

Note: each element of **plab** must be a string of length at least **lenlab** $- 1$.

9: **lenlab** – Integer *Input*

On entry: length of the strings allocated in **plab**. At most **lenlab** $- 1$ non-null characters will be written into each element of **plab**.

Constraint: if **plab** is not **NULL**, **lenlab** ≥ 1.

10: **lvinfo** – Integer *Input*

On entry: the length of **vinfo**.

Let n_T denote the number of terms in M_S, n_{Tt} denote the number of variables in the tth term and m_{xt} denote the number of columns of X corresponding to the tth term. The required size of **vinfo**, denoted a is given by:

$$a = \sum_{t=1}^{n_T} m_{xt}(1 + 3n_{Tt}).$$

If the model includes a mean effect, a should be incremented by one.

The values n_T, n_{Tt} and m_{xt} are not trivial to calculate as they require the formula describing the model to be fully expanded and the contrast / dummy variable encoding to be known. Therefore, if **lisx**, **lplab** or **lvinfo** are too small and **lvinfo** ≥ 3, **fail**.**code** = NW_ARRAY_SIZE is returned and the required sizes for these arrays are returned in **vinfo**[0], **vinfo**[1] and **vinfo**[2] respectively.

Constraint: **lvinfo** $= 0$ or **lvinfo** $\geq a$.

11: **vinfo**[**lvinfo**] – Integer *Output*

On exit: if **lvinfo** $\neq 0$, information encoding a description of the parameters in the model.

The encoding information can be extracted as follows:

(i) Set $k = 1$.

(ii) Iterate j from 1 to p.

1. Set $b =$ **vinfo**$[k - 1]$.
2. Increment k.
3. Iterate i from 1 to b.
   (a) Set $v_i =$ **vinfo**$[k - 1]$.
   (b) Set $l_i =$ **vinfo**$[k]$.
   (c) Set $c_i =$ **vinfo**$[k + 1]$.
   (d) Increment k by 3.
4. The jth model parameter corresponds to the interaction between the b variables held in columns $v_1, v_2, \ldots, v_b$ of D. Therefore, $b = 1$ indicates a main effect, $b = 2$ a two-way interaction, etc..

   If $b = 0$, the jth model parameter corresponds to the mean effect.

   If $l_i = 0$, the corresponding variable v_i is binary, ordinal or continuous. Otherwise, l_i is the level for the corresponding variable for model parameter j.

   c_i is a numeric flag indicating the contrast used in the case of a categorical variable. With $c_i = 0$ indicating that dummy variables were used for variable v_i in this term. The remaining six types of contrast; treatment contrasts (with respect to the first and last levels), sum contrasts (with respect to the first and last levels), Helmert contrasts and polynomial contrasts, as described in **nag_blgm_lm_design_matrix (g22ycc)**, are identified by the integers one to six respectively.

If **lvinfo** $= 0$, **vinfo** is not referenced and may be **NULL**.

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

The NAG error argument (see Section 3.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_FIELD_UNKNOWN

A variable name used when creating **hform** is not present in **hxdesc**.
Variable name: ⟨*value*⟩.

NE_HANDLE

hform has not been initialized or is corrupt.

hform is not a G22 handle as generated by **nag_blgm_lm_formula (g22yac)**.

hxdesc has not been initialized or is corrupt.

hxdesc is not a G22 handle as generated by **nag_blgm_lm_design_matrix (g22ycc)**.

NE_INT

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

On entry, **lisx** = ⟨*value*⟩ and $m_x = \langle value \rangle$.
Constraint: **lisx** $= 0$ or **lisx** $\geq m_x$.

On entry, **lplab** = ⟨*value*⟩ and $p = \langle value \rangle$.
Constraint: **lplab** $= 0$ or **lplab** $\geq p$.

On entry, **lvinfo** is too small.
lvinfo = ⟨*value*⟩.
Constraint: **lvinfo** $= 0$ or **lvinfo** $\geq$ ⟨*value*⟩.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_NOT_CONS

The model and the design matrix are not consistent.
Term: ⟨*value*⟩.
This is likely due to the design matrix being constructed in the presence of either a mean effect or main effect that is not present in the model.

The model and the design matrix are not consistent. The design matrix was constructed in the presence of a mean effect and the model does not include a mean effect.

The model and the design matrix are not consistent. The model includes a term not present in the design matrix.
Term: ⟨*value*⟩.

NW_ARRAY_SIZE

On entry, one or more of **lisx**, **lplab** or **lvinfo** are nonzero, but too small.
Minimum values are zero, or ⟨*value*⟩, ⟨*value*⟩ and ⟨*value*⟩ respectively.
The minimum values are returned in the first three elements of **vinfo**.

NW_NOT_CONS

The model and the design matrix are not consistent. The model specifies different contrasts to those used when the design matrix was constructed. The contrasts specified in **hform** will be ignored.

NW_TRUNCATED

On entry, **plab** is too short to hold the parameter labels. Long labels will be truncated.
The longest parameter label is ⟨*value*⟩.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_blgm_lm_submodel (g22ydc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example performs a linear regression using **nag_regsn_mult_linear (g02dac)**. The linear regression model is defined via a text string which is parsed using **nag_blgm_lm_formula (g22yac)** and the design matrix associated with the model is generated using **nag_blgm_lm_design_matrix (g22ycc)**. A submodel is then fit using the same design matrix.

Default parameter labels, as returned in **plab** are used for both models. An example of using the information returned in **vinfo** to construct more verbose parameter labels is given in Section 10 in **nag_blgm_lm_describe_data (g22ybc)**.

See also the examples for **nag_blgm_lm_formula (g22yac)** and **nag_blgm_lm_design_matrix (g22ycc)**.

10.1 Program Text

```
/* nag_blgm_lm_submodel (g22ydc) Example Program.
 *
 * Copyright 2017 Numerical Algorithms Group.
 *
 * Mark 26.1, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>
#include <nagg22.h>
```

```
#define MAX_FORMULA_LEN 200
#define MAX_VNAME_LEN 200
#define MAX_PLAB_LEN 200
#define MAX_CVALUE_LEN 200

#define DAT(I,J) dat[j*lddat+i]

char *read_line(char formula[],Integer nchar);
Integer construct_labels(Integer ip, char **plab[], char *const vnames[],
                         Integer vinfo[]);
Integer fit_lm(void *hform,Nag_IncludeIntercept intcpt,Integer nobs,Integer mx,
           double x[],Integer ldx,Integer isx[],Integer ip,double y[],
           char *plab[]);

int main(void)
{
  /* Integer scalar and array declarations */
  Integer i, j, ip = 0, lddat, ldx, lisx, lplab = 0, lvinfo,
    lvnames = 0, mx, nobs, nvar, sddat, sdx, lenlab;
  Integer exit_status = 0;
  Integer *isx = 0, *levels = 0, *vinfo = 0;
  Integer tvinfo[3];

  /* Nag Types */
  NagError fail;
  Nag_IncludeIntercept intcpt;

  /* Double scalar and array declarations */
  double *dat = 0, *x = 0, *y = 0;

  /* Character scalar and array declarations */
  char formula[MAX_FORMULA_LEN];
  char **vnames = 0, **plab = 0;

  /* Void pointers */
  void *hform = 0, *hddesc = 0, *hxdesc = 0;

  /* Initialize the error structure */
  INIT_FAIL(fail);

  printf("nag_blgm_lm_submodel (g22ydc) Example Program Results\n\n");

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

  /* Read in size of the data matrix and number of variable labels supplied */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &nobs, &nvar,
          &lvnames);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &nobs, &nvar,
        &lvnames);
#endif

  /* Allocate memory */
  if (!(levels = NAG_ALLOC(nvar, Integer)) ||
      !(vnames = NAG_ALLOC(lvnames, char *))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lvnames; i++)
    if (!(vnames[i] = NAG_ALLOC(MAX_VNAME_LEN, char))) {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
```

```
    }

  /* Read in number of levels and names for the variables */
  for (i = 0; i < nvar; i++) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &levels[i]);
#else
    scanf("%" NAG_IFMT "", &levels[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  if (lvnames > 0) {
    for (i = 0; i < lvnames; i++)
#ifdef _WIN32
      scanf_s("%50s", vnames[i], 51);
#else
      scanf("%50s", vnames[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }

  /* Call nag_blgm_lm_describe_data (g22ybc) to get a description of */
  /* the data matrix */
  nag_blgm_lm_describe_data(&hddesc,nobs,nvar,levels,lvnames,vnames,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_describe_data (g22ybc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Read in the data matrix and response variable */
  lddat = nobs;
  sddat = nvar;
  if (!(dat = NAG_ALLOC(lddat*sddat, double)) ||
      !(y = NAG_ALLOC(nobs, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < nobs; i++) {
    for (j = 0; j < nvar; j++)
#ifdef _WIN32
      scanf_s("%lf", &DAT(i, j));
#else
    scanf("%lf", &DAT(i, j));
#endif
#ifdef _WIN32
    scanf_s("%lf", &y[i]);
#else
    scanf("%lf", &y[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read in the formula for the full model, remove comments and */
  /* call nag_blgm_lm_formula (g22yac) to parse it */
  read_line(formula,MAX_FORMULA_LEN);
```

```
  nag_blgm_lm_formula(&hform,formula,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_formula (g22yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Start of constructing the design matrix ... */

  /* Call nag_blgm_optset (g22zmc) to alter the storage order of X as */
  /* nag_regsn_mult_linear uses VAROBS storage  */
  nag_blgm_optset(hform,"Storage Order = VAROBS",&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_optset (g22zmc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Call nag_blgm_lm_design_matrix (g22ycc) to get the size of */
  /* the design matrix */
  ldx = 0;
  sdx = 0;
  nag_blgm_lm_design_matrix(hform,hddesc,dat,lddat,sddat,&hxdesc,
                            x,ldx,sdx,&mx,&fail);
  if (fail.code != NW_ARRAY_SIZE && fail.code != NW_ALTERNATIVE) {
    printf("Error from nag_blgm_lm_design_matrix (g22ycc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Allocate design matrix */
  ldx = mx;
  sdx = nobs;
  if (!(x = NAG_ALLOC(ldx*sdx, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Call nag_blgm_lm_design_matrix (g22ycc) to generate the design matrix */
  nag_blgm_lm_design_matrix(hform,hddesc,dat,lddat,sddat,&hxdesc,
                            x,ldx,sdx,&mx,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_design_matrix (g22ycc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }
  /* ... End of constructing the design matrix */

  /* Start of getting the isx vector and information on parameter labels ... */
  /* Get size of output arrays used by nag_blgm_lm_submodel (g22ydc) */
  lvinfo = 3;
  lisx = lplab = lenlab = 0;
  nag_blgm_lm_submodel(hform,hxdesc,&intcpt,&ip,lisx,isx,lplab,plab,lenlab,
                       lvinfo,tvinfo, &fail);
  if (fail.code != NW_ARRAY_SIZE) {
    printf("Error from nag_blgm_lm_submodel (g22ydc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Allocate output arrays (we already know that lisx = mx, but */
  /* nag_blgm_lm_submodel returns it just in case) */
  lisx = tvinfo[0];
  lplab = tvinfo[1];
  /* We don't need vinfo as we are using labels in plab */
  lvinfo = 0;
  if (!(isx = NAG_ALLOC(lisx, Integer)) ||
      !(plab = NAG_ALLOC(lplab, char *))) {
```

```
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  lenlab = MAX_PLAB_LEN;
  for (i = 0; i < lplab; i++)
    if (!(plab[i] = NAG_ALLOC(lenlab, char))) {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Call nag_blgm_lm_submodel (g22ydc) to get the isx flag */
  /* and parameter labels */
  nag_blgm_lm_submodel(hform,hxdesc,&intcpt,&ip,lisx,isx,lplab,plab,lenlab,
                       lvinfo,vinfo,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_submodel (g22ydc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* ... End of getting the isx vector and information on parameter labels */

  /* Fit a regression model and print the results */
  if ((exit_status = fit_lm(hform,intcpt,nobs,mx,x,ldx,isx,ip,y,plab)))
    goto END;

  /* Read in the formula for the sub-model, remove comments and */
  /* call nag_blgm_lm_formula (g22yac) to parse it */
  read_line(formula,MAX_FORMULA_LEN);
  nag_blgm_lm_formula(&hform,formula,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_formula (g22yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Call nag_blgm_lm_submodel (g22ydc) to get the isx flag and parameter */
  /* labels as the new model has to be a sub-model of the original one, the */
  /* output arrays, isx, plab and vinfo can be reused as they will be of */
  /* sufficient size */
  nag_blgm_lm_submodel(hform,hxdesc,&intcpt,&ip,lisx,isx,lplab,plab,lenlab,
                       lvinfo,vinfo,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_lm_submodel (g22ydc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\n\n");

  /* Fit the sub-model and print the results */
  exit_status = fit_lm(hform,intcpt,nobs,mx,x,ldx,isx,ip,y,plab);

END:
  /* Call nag_blgm_handle_free (g22zac) to clean-up the g22 handles */
  nag_blgm_handle_free(&hform,&fail);
  nag_blgm_handle_free(&hddesc,&fail);
  nag_blgm_handle_free(&hxdesc,&fail);

  NAG_FREE(dat);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(levels);
  for (i = 0; i < lvnames; i++)
    NAG_FREE(vnames[i]);
  NAG_FREE(vnames);
  for (i = 0; i < lplab; i++)
    NAG_FREE(plab[i]);
  NAG_FREE(plab);
  NAG_FREE(isx);
```

```
  NAG_FREE(vinfo);
  return (exit_status);
}

char *read_line(char formula[],Integer nchar) {
  /* Read in a line from stdin and remove any comments */
  char *pch;

  /* Read in the model formula */
  if (fgets(formula,nchar,stdin)) {
    /* Strip comments from formula */
    pch = strstr(formula,"::");
    if (pch) *pch = '\setminus 0';
    return formula;
  } else {
    return 0;
  }
}

Integer fit_lm(void *hform,Nag_IncludeIntercept intcpt,Integer nobs,Integer mx,
           double x[],Integer ldx,Integer isx[],Integer ip,double y[],
           char *plab[]) {
  /* Perform a multiple linear regression using */
  /* nag_regsn_mult_linear (g02dac) */

  /* Integer scalar and array declarations */
  Integer i, rank, ldq, exit_status = 0, lcvalue, ivalue;

  /* NAG types */
  Nag_Boolean svd;
  NagError fail;
  Nag_IncludeMean mean;
  Nag_VariableType optype;

  /* Double scalar and array declarations */
  double rss, tol, df, rvalue;
  double *b = 0, *cov = 0, *h = 0, *p = 0, *q = 0, *res = 0, *se = 0,
    *wt = 0, *com_ar = 0;

  /* Character scalar and array declarations */
  char cvalue[MAX_CVALUE_LEN];

  /* Initialize the error structure */
  INIT_FAIL(fail);

  /* We are assuming un-weighted data */
  ldq = (ip + 1);
  if (!(b = NAG_ALLOC(ip, double)) ||
      !(se = NAG_ALLOC(ip, double)) ||
      !(cov = NAG_ALLOC((ip * ip + ip) / 2, double)) ||
      !(res = NAG_ALLOC(nobs, double)) ||
      !(h = NAG_ALLOC(nobs, double)) ||
      !(q = NAG_ALLOC(nobs * ldq, double)) ||
      !(p = NAG_ALLOC(ip * (ip + 2), double)) ||
      !(com_ar = NAG_ALLOC(ip * ip + 5 * (ip - 1), double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Use suggested value for tolerance */
  tol = 0.000001;

  mean = (intcpt == Nag_Intercept) ? Nag_MeanInclude : Nag_MeanZero;

  /* Call nag_regsn_mult_linear (g02dac) to fit a regression model */
  nag_regsn_mult_linear(mean, nobs, x, ldx, mx, isx, ip, y,
                        wt, &rss, &df, b, se, cov, res, h, q,
                        ldq, &svd, &rank, p, tol, com_ar, &fail);
```

```
  /* Call nag_blgm_optget (g22znc) to get the formula for the model */
  /* being fit */
  lcvalue = MAX_CVALUE_LEN;
  nag_blgm_optget(hform,"Formula",&ivalue,&rvalue,cvalue,lcvalue,
                  &optype,&fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_blgm_optget (g22znc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Display the results */
  printf(" Model: %s\n", cvalue);
  printf("                                  Parameter   Standard\n");
  printf(" Coefficients                     Estimate      Error\n");
  printf(" ------------------------------------------------------\n");
  for (i = 0; i < ip; i++)
    printf(" %-30s %7.3f     %7.3f\n", plab[i], b[i], se[i]);
  printf(" ------------------------------------------------------\n");
  printf(" Residual sum of squares = %9.4f\n", rss);
  printf(" Degrees of freedom      = %9.0f\n", df);

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

10.2 Program Data

```
nag_blgm_lm_submodel (g22ydc) Example Program Data
25 3 3                  :: nobs,nvar,lvnames
3 4 1                   :: levels
Fact1 Fact2 Con         :: vnames
3 1 -2.4   1.16
3 4  0.2   4.96
1 4 -1.4  -1.67
2 1 -5.4 -11.80
3 3  0.2   6.03
3 2  1.4  11.70
1 2  6.8  33.34
1 4  6.7  31.97
1 1  5.3  23.93
2 3 -1.3   3.17
3 2 -3.6   1.68
3 2 -0.7   8.01
1 4  5.7  26.14
3 3  2.3  11.04
1 2  3.3  20.32
2 3 -0.5   5.62
1 1 -2.6  -6.21
1 2  3.7  22.45
1 2  0.9  10.93
3 4 -1.1   1.59
2 2  2.1  13.55
1 3  4.6  24.16
2 3  4.6  20.70
1 2  5.1  28.30
1 3  0.9   9.69         :: dat, y
(Fact2 + Con + Fact1)^2 :: Model formula
Fact1 + Fact2 + Con     :: Sub-Model formula
```

10.3 Program Results

```
nag_blgm_lm_submodel (g22ydc) Example Program Results

 Model: FACT2+CON+FACT1+FACT2.CON+FACT2.FACT1+CON.FACT1
                              Parameter   Standard
 Coefficients                 Estimate      Error
 ------------------------------------------------
 Intercept                      3.709       0.490
 FACT2_2                        4.200       0.801
 FACT2_3                        2.302       0.913
 FACT2_4                        0.174       0.727
 CON                            3.815       0.117
 FACT1_2                       -0.321       1.828
 FACT1_3                        2.377       1.041
 FACT2_2.CON                    0.013       0.184
 FACT2_3.CON                    0.153       0.253
 FACT2_4.CON                    0.257       0.157
 FACT2_2.FACT1_2                0.029       2.442
 FACT2_2.FACT1_3               -1.160       1.372
 FACT2_3.FACT1_2                1.372       2.407
 FACT2_3.FACT1_3               -2.611       1.435
 FACT2_4.FACT1_2               -0.000       0.000
 FACT2_4.FACT1_3               -1.946       1.247
 CON.FACT1_2                   -1.003       0.266
 CON.FACT1_3                   -1.763       0.206
 ------------------------------------------------
 Residual sum of squares =    3.4371
 Degrees of freedom      =         8


 Model: FACT1+FACT2+CON
                              Parameter   Standard
 Coefficients                 Estimate      Error
 ------------------------------------------------
 Intercept                      6.222       1.322
 FACT2_2                        3.225       1.599
 FACT2_3                        0.074       1.694
 FACT2_4                       -0.601       1.762
 CON                            3.442       0.196
 FACT1_2                       -0.319       1.561
 FACT1_3                        0.064       1.346
 ------------------------------------------------
 Residual sum of squares =  105.6953
 Degrees of freedom      =        18
```