NAG Library Function Document

nag_blgm_lm_formula (g22yac)

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_formula (g22yac) parses a text string containing a formula specifying a linear model and outputs a G22 handle to an internal data structure. This G22 handle can then be passed to various functions in Chapter g22. In particular, the G22 handle can be passed to **nag_blgm_lm_design_matrix (g22ycc)** to produce a design matrix or **nag_blgm_lm_submodel (g22ydc)** to produce a vector of column inclusion flags suitable for use with functions in Chapter g02.

2 Specification

```
#include <nag.h>
#include <nagg22.h>
void nag_blgm_lm_formula (void **hform, const char *formula, NagError *fail)
```

3 Description

3.1 Background

Let D denote a data matrix with n observations on m_d independent variables, denoted $V_1, V_2, \ldots, V_{m_d}$. Let y denote a vector of n observations on a dependent variable.

A linear model, $\mathcal{M}$, as the term is used in this function, expresses a relationship between the independent variables, V_j, and the dependent variable. This relationship can be expressed as a series of additive terms $T_1 + T_2 + \ldots$, with each term, T_t, representing either a single independent variable V_j, called the main effect of V_j, or the interaction between two or more independent variables. An interaction term, denoted here using the . operator, allows the effect of an independent variable on the dependent variable to depend on the value of one or more other independent variables. As an example, the three-way interaction between V_1, V_2 and V_3 is denoted $V_1.V_2.V_3$ and describes a situation where the effect of one of these three variables is influenced by the value of the other two.

This function takes a description of $\mathcal{M}$, supplied as a text string containing a formula, and outputs a G22 handle to an internal data structure. This G22 handle can then be passed to **nag_blgm_lm_design_matrix (g22ycc)** to produce a design matrix for use in analysis functions from other chapters, for example the regression functions of Chapter g02.

A more detailed description of what is meant by a G22 handle can be found in Section 2.1 in the g22 Chapter Introduction.

3.2 Syntax

In its most verbose form $\mathcal{M}$ can be described by one or more variable names, V_j, and the two operators, $+$ and .. In order to allow a wide variety of models to be specified compactly this syntax is extended to six operators ($+$,.,$*$,$-$,:,) and parentheses.

A formula describing the model is supplied to **nag_blgm_lm_formula (g22yac)** via a character string which must obey the following rules:

1. Variables can be denoted by arbitrary names, as long as
   (i) The names used are a subset of those supplied to **nag_blgm_lm_describe_data (g22ybc)** when describing D.
   (ii) The names do not contain any of the characters in $+.*-:()$@.

2. The . operator denotes an interaction between two or more variables or terms, with $V_1.V_2.V_3$ denoting the three-way interaction between V_1, V_2 and V_3.
3. A term in $\mathcal{M}$ can contain one or more variable names, separated using the . operator, i.e., a term can be either a main effect or an interaction term between two or more variables.
   (i) If a variable appears in an interaction term more than once, all subsequent appearances, after the first, are ignored, therefore $V_1.V_2.V_1$ is the same as $V_1.V_2$.
   (ii) The ordering of the variables in an interaction term is ignored when comparing terms, therefore $V_1.V_2$ is the same as $V_2.V_1$. This ordering may have an effect when the resulting G22 handle is passed to another function, for example **nag_blgm_lm_design_matrix (g22ycc)**.
   (iii) Applying the . operator to two terms appends one to the other, for example, if $T_1 = V_1.V_2$ and $T_2 = V_3.V_4$, $T_1.T_2 = V_1.V_2.V_3.V_4$.
4. The $+$ operator allows additional terms to be included in $\mathcal{M}$, therefore $T_1 + T_2$ is a model that includes terms T_1 and T_2.
   (i) If a term is added to $\mathcal{M}$ more than once, all subsequent appearances, after the first, are ignored, therefore $T_1 + T_2 + T_1$ is the same as $T_1 + T_2$.
   (ii) The ordering of the terms is ignored whilst parsing the formula, therefore $T_1 + T_2$ is the same as $T_2 + T_1$. This ordering may have an effect when the resulting G22 handle is passed to another function, for example **nag_blgm_lm_design_matrix (g22ycc)**.
   (iii) Internally, the terms are reordered so that all main effects come first, followed by two-way interactions, then three-way interactions, etc. The ordering within each of these categories is preserved.
5. The $*$ operator can be used as a shorthand notation denoting the main effects and all interactions between the variables involved. Therefore, $T_1 * T_2$ is equivalent to $T_1 + T_2 + T_1.T_2$ and $T_1 * T_2 * T_3$ is equivalent to $T_1 + T_2 + T_3 + T_1.T_2 + T_1.T_3 + T_2.T_3 + T_1.T_2.T_3$.
6. The $-$ operator removes a term from $\mathcal{M}$, therefore $T_1 * T_2 * T_3 - T_1.T_2.T_3$ is equivalent to $T_1 + T_2 + T_3 + T_1.T_2 + T_1.T_3 + T_2.T_3$ as the three-way interaction, $T_1.T_2.T_3$, usually present due to $T_1 * T_2 * T_3$ has been removed.
7. The : operator is a shorthand way of specifying a series of variables, with $V_1 : V_j$ being equivalent to $V_1 + V_2 + \ldots + V_j$.
   (i) This operator can only be used if the variable names end in a numeric, therefore VAR2 : VAR4 would be valid, but FVAR : LVAR would not.
   (ii) The root part of both variable names (i.e., the part before the trailing numeric, so VAR in the valid example above) must be the same.
   (iii) The trailing numeric parts of the two variable names must be in ascending order.
8. The operator is a shorthand notation for a series of $*$ operators. $(T_1 + T_2 + T_3)^2$ is equivalent to $(T_1 + T_2 + T_3) * (T_1 + T_2 + T_3)$ which in turn is equivalent to $T_1 + T_2 + T_3 + T_1.T_2 + T_1.T_3 + T_2.T_3$.
   (i) This notation is present primarily for use with the : operator in examples of the form, $(V_1 : V_5)^3$ which specifies a model containing the main effects for variables V_1 to V_5 as well as all two- and three-way interactions.
   (ii) Using the operator on a single term has no effect, therefore $T_2{}^2$ is the same as T_2.

3.2.1 Precedence

Each operator has an associated default precedence, but this can be overridden through the use of parentheses. The default precedence is:

1. The : operator, with the resulting expression is treated as if it was surrounded by parentheses. Therefore, $V_1 + V_3 : V_6 * V_7$ is equivalent to $V_1 + (V_3 + V_4 + V_5 + V_6) * V_7$.
2. The operator, with the resulting expression is treated as if it was surrounded by parentheses. Therefore, $(T_1 + T_2 + T_3)^2.T_4$ is equivalent to $\left((T_1 + T_2 + T_3)^2\right).T_4$, which is the equivalent to $T_1.T_4 + T_2.T_4 + T_3.T_4 + T_1.T_2.T_4 + T_1.T_3.T_4 + T_2.T_3.T_4$.
3. The . operator, so $T_1 * T_2.T_3$ is equivalent to $T_1 * (T_2.T_3)$.
4. The $*$ operator.
   (i) When using parentheses with the $*$ or . operators the usual rules of multiplication apply, therefore $(T_1 + T_3.T_4).(T_5 + T_7)$ is equivalent to $T_1.T_5 + T_1.T_7 + T_3.T_4.T_5 + T_3.T_4.T_7$ and $(T_1 + T_3.T_4) * (T_5 + T_7)$ is equivalent to $T_1 + T_5 + T_7 + T_3.T_4 + T_1.T_5 + T_1.T_7 + T_3.T_4.T_5 + T_3.T_4.T_7$.
   (ii) Syntax of the following form is invalid: $T_1 o (T_2) o T_3$, where o indicates an operator, unless one or more of those operators are $+$ and/or $-$. Therefore, $T_1.(T_2 + T_3) * T_4$ is invalid, whilst $T_1.(T_2 + T_3) + T_4$ is valid.
5. The $+$ and $-$ operators have equal precedence.
   (i) If the terms associated with a $-$ operator do not occur in the current expression they are ignored, therefore $T_1 + (T_2 - T_1)$ is the equivalent to $T_1 + T_2$; the $(T_2 - T_1)$ part of the expression is calculated first and results in T_2 as the T_1 term does not exist in this particular sub-expression so cannot be removed.

3.2.2 Mean Effect / Intercept Term

A mean effect (or intercept term) can be explicitly added to a formula by specifying 1 and can be explicitly excluded from the formula by specifying -1. For example, $1 + V_1 + V_2$ indicates a model with the main effects of two variables and a mean effect, whereas $V_1 + V_2 - 1$ denotes the same model, but without the mean effect. The mean indicator can appear anywhere in the formula string as long as it is not contained within parentheses.

If the mean effect is not explicitly mentioned in the model formula, the model is assumed to include a mean effect.

4 References

None.

5 Arguments

1: **hform** – void ** *Input/Output*

On entry: must be set to **NULL**.

As an alternative, an existing G22 handle may be supplied in which case this function will destroy the supplied G22 handle as if **nag_blgm_handle_free (g22zac)** had been called.

On exit: holds a G22 handle to the internal data structure containing a description of the model $\mathcal{M}$ as specified in **formula**. You **must not** change the G22 handle other than through functions in Chapter g22.

2: **formula** – const char * *Input*

On entry: a string containing the formula specifying $\mathcal{M}$. See Section 3 for details on the allowed model syntax.

3: **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_HANDLE

On entry, **hform** is not **NULL** or a recognised G22 handle.

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_INVALID_FORMAT

After processing, the model contains no terms.

An invalid contrast specifier has been supplied.
The position in the formula string of the error is ⟨*value*⟩.

An operator was missing.
The position in the formula string of the error is ⟨*value*⟩.

Invalid specification for the colon operator.
The position in the formula string of the error is ⟨*value*⟩.

Invalid specification for the mean.
The position in the formula string of the error is ⟨*value*⟩.

Invalid specification for the power operator.
The position in the formula string of the error is ⟨*value*⟩.

Invalid use of an operator.
The position in the formula string of the error is ⟨*value*⟩.

Invalid variable name.
The position in the formula string of the error is ⟨*value*⟩.

Missing variable name.
The position in the formula string of the error is ⟨*value*⟩.

The formula contained a mismatched parenthesis.
The position in the formula string of the error is ⟨*value*⟩.

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.

NW_POTENTIAL_PROBLEM

A term contained a repeated variable with a different contrast specifier.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_blgm_lm_formula (g22yac) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example reads in and parses a formula specifying a model, $\mathcal{M}$, and displays the processed formula. A data matrix, D, is then read in and a design matrix constructed from D and $\mathcal{M}$ using **nag_blgm_lm_design_matrix (g22ycc)**.

The design matrix includes an explicit term for the mean effect.

See also the examples for **nag_blgm_lm_describe_data (g22ybc)**, **nag_blgm_lm_design_matrix (g22ycc)** and **nag_blgm_lm_submodel (g22ydc)**.

10.1 Program Text

```
/* nag_blgm_lm_formula (g22yac) 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 <nagg22.h>

#define MAX_FORMULA_LEN 200
#define MAX_VNAME_LEN 200
#define MAX_CVALUE_LEN 200

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

char *read_line(char formula[],Integer nchar);

int main(void)
{
  /* Integer scalar and array declarations */
  Integer i, j, ivalue, lddat, ldx, lvnames = 0, mx, nobs, nvar,
    sddat, sdx, lcvalue;
  Integer exit_status = 0;
  Integer *levels = 0;

  /* Nag Types */
  NagError fail;
  Nag_VariableType optype;

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

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

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

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

  printf("nag_blgm_lm_formula (g22yac) Example Program Results\n\n");

  /* Skip heading in data file */
#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;
  }

  /* Call nag_blgm_optget (g22znc) to extract the parsed formula */
  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;
  }

  printf(" Formula: %s\n", cvalue);
  printf("\n");

  /* 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

  /* Call nag_blgm_optset (g22zmc) to set optional arguments */
  /* Want the design matrix in include an explicit term for the mean effect */
  nag_blgm_optset(hform,"Explicit Mean = Yes",&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 = nobs;
  sdx = mx;
  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;
  }

  /* Display the design matrix */
  printf(" Design Matrix (X)\n");
  for (i = 0; i < nobs; i++) {
    for (j = 0; j < mx; j++)
      printf(" %4.1f",X(i,j));
    printf("\n");
  }

 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);
  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;
  }
}
```

10.2 Program Data

```
nag_blgm_lm_formula (g22yac) Example Program Data
(F2 + Con + F1)^2                     :: formula
25 3 3                                :: nobs,nvar,lvnames
3 3 1                                 :: levels
F1 F2 Con                             :: vnames
3 1 -2.4   1.16
3 3  0.2   4.96
```

```
1 3 -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 2  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 1  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 1 -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
```

10.3 Program Results

```
nag_blgm_lm_formula (g22yac) Example Program Results

 Formula: F2+CON+F1+F2.CON+F2.F1+CON.F1

 Design Matrix (X)
  1.0  0.0  0.0 -2.4  0.0  1.0 -0.0 -0.0  0.0  0.0  0.0  0.0 -0.0 -2.4
  1.0  0.0  1.0  0.2  0.0  1.0  0.0  0.2  0.0  0.0  0.0  1.0  0.0  0.2
  1.0  0.0  1.0 -1.4  0.0  0.0 -0.0 -1.4  0.0  0.0  0.0  0.0 -0.0 -0.0
  1.0  0.0  0.0 -5.4  1.0  0.0 -0.0 -0.0  0.0  0.0  0.0  0.0 -5.4 -0.0
  1.0  0.0  1.0  0.2  0.0  1.0  0.0  0.2  0.0  0.0  0.0  1.0  0.0  0.2
  1.0  1.0  0.0  1.4  0.0  1.0  1.4  0.0  0.0  1.0  0.0  0.0  0.0  1.4
  1.0  1.0  0.0  6.8  0.0  0.0  6.8  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  1.0  0.0  6.7  0.0  0.0  6.7  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  0.0  5.3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  1.0 -1.3  1.0  0.0 -0.0 -1.3  0.0  0.0  1.0  0.0 -1.3 -0.0
  1.0  1.0  0.0 -3.6  0.0  1.0 -3.6 -0.0  0.0  1.0  0.0  0.0 -0.0 -3.6
  1.0  1.0  0.0 -0.7  0.0  1.0 -0.7 -0.0  0.0  1.0  0.0  0.0 -0.0 -0.7
  1.0  0.0  0.0  5.7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  1.0  2.3  0.0  1.0  0.0  2.3  0.0  0.0  0.0  1.0  0.0  2.3
  1.0  1.0  0.0  3.3  0.0  0.0  3.3  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  1.0 -0.5  1.0  0.0 -0.0 -0.5  0.0  0.0  1.0  0.0 -0.5 -0.0
  1.0  0.0  0.0 -2.6  0.0  0.0 -0.0 -0.0  0.0  0.0  0.0  0.0 -0.0 -0.0
  1.0  1.0  0.0  3.7  0.0  0.0  3.7  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  1.0  0.0  0.9  0.0  0.0  0.9  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  0.0 -1.1  0.0  1.0 -0.0 -0.0  0.0  0.0  0.0  0.0 -0.0 -1.1
  1.0  1.0  0.0  2.1  1.0  0.0  2.1  0.0  1.0  0.0  0.0  0.0  2.1  0.0
  1.0  0.0  1.0  4.6  0.0  0.0  0.0  4.6  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  1.0  4.6  1.0  0.0  0.0  4.6  0.0  0.0  1.0  0.0  4.6  0.0
  1.0  1.0  0.0  5.1  0.0  0.0  5.1  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  1.0  0.0  1.0  0.9  0.0  0.0  0.0  0.9  0.0  0.0  0.0  0.0  0.0  0.0
```

11 Optional Parameters

As well as the optional parameters common to all G22 handles described in **nag_blgm_optset (g22zmc)** and **nag_blgm_optget (g22znc)**, a number of additional optional parameters can be specified for a G22 handle holding the description of a model, as returned by **nag_blgm_lm_formula (g22yac)** in **hform**.

Each writeable optional parameter has an associated default value; to set any of them to a non-default value, use **nag_g02_opt_set (g02zkc)**. The value of any optional parameter can be queried using **nag_blgm_optget (g22znc)**.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 11.1.

Contrast

Explicit Mean

Formula

Storage Order

All functions that make use of the G22 handle returned by **nag_blgm_lm_formula (g22yac)** combine it with a description of a data matrix, D, to construct a design matrix, X.

11.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value.

Keywords and character values are case and white space insensitive.

Contrast a Default $=$ FIRST

This parameter controls the default contrasts used for the categorical independent variables appearing in the model. Six types of contrasts and dummy variables are available:

FIRST
: Treatment contrasts relative to the first level of the variable will be used.

LAST
: Treatment contrasts relative to the last level of the variable will be used.

SUM FIRST
: Sum contrasts relative to the first level of the variable will be used.

SUM LAST
: Sum contrasts relative to the last level of the variable will be used.

HELMERT
: Helmert contrasts will be used.

POLYNOMIAL
: Polynomial contrasts will be used.

DUMMY
: Dummy variables will be used rather than a contrast.

See **nag_blgm_lm_design_matrix (g22ycc)** for more information on contrasts, their effect on the design matrix and how they are constructed.

This parameter may have an *instance identifier* associated with it (see **nag_blgm_optset (g22zmc)** and **nag_blgm_optget (g22znc)**). The *instance identifier* must be the name of one of the variables appearing in the model supplied in **formula** when the G22 handle was created. For example, *CONTRAST : VAR1 = HELMERT* would set Helmert contrasts for the variable named *VAR1*.

If no *instance identifier* is specified, the default contrast for all categorical variables in the model is changed, otherwise only the default contrast for the named variable is changed.

In some situations it might be necessary for a variable to use a different contrast, depending on where it appears in the model formula. In order to allow contrasts to be specified on a term by term basis the @ operator can be used in the model formula. The syntax for this operator is V_j@c, where c is one of: F, L, *SF*, *SL*, *H*, *P* or *D*, corresponding to treatment contrasts relative to the first and last levels, sum contrasts relative to the first and last levels, Helmert contrasts, polynomial contrasts or dummy variables respectively.

If the contrast has not been explicitly specified via the @ operator, the value obtained from the optional parameter **Contrast** is used.

For example, setting **formula** to *VAR1 + VAR1@H.VAR2@P + VAR2@H.VAR3*, specifies that the variable named *VAR1* should use the default contrasts in the first term and Helmert contrasts in the second term. The variable named *VAR2* should use polynomial contrasts in the second term and Helmert contrasts in the third term. The variable named *VAR3* should use the default contrasts in the third term.

Constraint: **Contrast** = FIRST, LAST, SUM FIRST, SUM LAST, HELMERT, POLYNOMIAL or DUMMY.

Explicit Mean a Default $=$ NO

If **Explicit Mean** = YES, any mean effect included in the model will be explicitly added to the design matrix, X, as a column of 1s.

If **Explicit Mean** = NO, it is assumed that the function to which X will be passed treats the mean effect as a special case, see **mean** in **nag_regsn_mult_linear (g02dac)** for example.

Constraint: **Explicit Mean** = YES or NO.

Formula a

This parameter returns a verbose version of the model formula specified in **formula**, expanded and simplified to only contain variable names, the operators + and . and any contrast identifiers present.

Storage Order a Default $=$ OBSVAR

This optional parameter controls how the design matrix, X, should be stored in its output array and only has an effect if the design matrix is being constructed using **nag_blgm_lm_design_matrix (g22ycc)**.

If **Storage Order** = OBSVAR, X_{ij}, the value for the jth variable of the ith observation of the design matrix is stored in $\mathbf{x}[(j-1) \times \mathbf{pdx} + i - 1]$.

If **Storage Order** = VAROBS, X_{ij}, the value for the jth variable of the ith observation of the design matrix is stored in $\mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1]$.

Where **x** is the output parameter of the same name in **nag_blgm_lm_design_matrix (g22ycc)**.

Constraint: **Storage Order** = OBSVAR or VAROBS.