NAG Library Routine Document

g22ybf

Note: please be advised that this routine 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

g22ybf describes a data matrix.

2 Specification

Fortran Interface

```
Subroutine g22ybf (hddesc, nobs, nvar, levels, lvnames, vnames, ifail)
Integer, Intent (In)          :: nobs, nvar, levels(nvar), lvnames
Integer, Intent (Inout)       :: ifail
Character (*), Intent (In)    :: vnames(lvnames)
Type (c_ptr), Intent (Inout) :: hddesc
```

3 Description

Let D denote a data matrix with n observations on m_d independent variables, denoted $V_1, V_2, \ldots, V_{m_d}$. The jth independent variable, V_j can be classified as either binary, categorical, ordinal or continuous, where:

Binary
: V_j can take the value 1 or 0.

Categorical
: V_j can take one of L_j distinct values or levels. Each level represents a discrete category but does not necessarily imply an ordering. The value used to represent each level is therefore arbitrary and, by convention and for convenience, is taken to be the integers from 1 to L_j.

Ordinal
: As with a categorical variable V_j can take one of L_j distinct values or levels. However, unlike a categorical variable, the levels of an ordinal variable imply an ordering and hence the value used to represent each level is not arbitrary. For example, $V_j = 4$ implies a value that is twice as large as $V_j = 2$.

Continuous
: V_j can take any real value.

g22ybf returns a G22 handle containing a description of a data matrix, D. The data matrix makes no distinction between binary, ordinal or continuous variables.

A name can also be assigned to each variable. If names are not supplied then the default vector of names, $\{\text{'V1'}, \text{'V2'}, \ldots\}$ is used.

4 References

None.

5 Arguments

1: **hddesc** – Type (c_ptr) *Input/Output*

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

As an alternative an existing G22 handle may be supplied in which case this routine will destroy the supplied G22 handle as if **g22zaf** had been called.

On exit: holds a G22 handle to the internal data structure containing a description of the data matrix, D. You **must not** change the G22 handle other than through the routines in Chapter G22.

2: **nobs** – Integer *Input*

On entry: n, the number of observations in the data matrix, D.

Constraint: $\textbf{nobs} \geq 0$.

3: **nvar** – Integer *Input*

On entry: m_d, the number of variables in the data matrix, D.

Constraint: $\textbf{nvar} \geq 0$.

4: **levels**(**nvar**) – Integer array *Input*

On entry: **levels**(j) contains the number of levels associated with the jth variable of the data matrix, for $j = 1, 2, \ldots, \textbf{nvar}$.

If the jth variable is binary, ordinal or continuous, **levels**(j) should be set to 1; otherwise **levels**(j) should be set to the number of levels associated with the jth variable and the corresponding column of the data matrix is assumed to take the value 1 to **levels**(j).

Constraint: $\textbf{levels}(i) \geq 1$, for $i = 1, 2, \ldots, \textbf{nvar}$.

5: **lvnames** – Integer *Input*

On entry: the number of variable names supplied in **vnames**.

Constraint: $\textbf{lvnames} = 0$ or **nvar**.

6: **vnames**(**lvnames**) – Character(*) array *Input*

On entry: if $\textbf{lvnames} \neq 0$, **vnames**(j) must contain the name of the jth variable, for $j = 1, 2, \ldots, \textbf{nvar}$. If $\textbf{lvnames} = 0$, **vnames** is not referenced.

The names supplied in **vnames** should be at most 50 characters long and be unique. If a name longer than 50 characters is supplied it will be truncated.

Variable names must not contain any of the characters +.*-:^()@.

7: **ifail** – Integer *Input/Output*

On entry: **ifail** must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of ifail on exit.**

On exit: $\textbf{ifail} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry **ifail** $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by **x04aaf**).

Errors or warnings detected by the routine:

ifail $= 11$

On entry, **hddesc** is not **c_null_ptr** or a recognised G22 handle.

ifail $= 21$

On entry, **nobs** $= \langle value \rangle$.
Constraint: **nobs** ≥ 0.

ifail $= 31$

On entry, **nvar** $= \langle value \rangle$.
Constraint: **nvar** ≥ 0.

ifail $= 41$

On entry, $j = \langle value \rangle$ and **levels**$(j) = \langle value \rangle$
Constraint: **levels**$(i) \geq 1$.

ifail $= 51$

On entry, **lvnames** $= \langle value \rangle$ and **nvar** $= \langle value \rangle$.
Constraint: **lvnames** $= 0$ or **nvar**.

ifail $= 61$

On entry, variable name i contains one more invalid characters, $i = \langle value \rangle$.

ifail $= 62$

On entry, variable names i and j are not unique, $i = \langle value \rangle$ and $j = \langle value \rangle$.

ifail $= 63$

On entry, variable names i and j are not unique (possibly due to truncation), $i = \langle value \rangle$ and $j = \langle value \rangle$.
Maximum variable name length is 50.

ifail $= 64$

At least one variable name was truncated to 50 characters. Each truncated name is unique and will be used in all output.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

ifail $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

g22ybf is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example performs a linear regression using **g02daf**. The linear regression model is defined via a text string which is parsed using **g22yaf**. The corresponding design matrix associated with the model and the dataset described via a call to **g22ybf** is generated using **g22ycf**.

Verbose labels for the parameters of the model are constructed using information returned in **vinfo** by **g22ydf**.

See also the examples in **g22yaf**, **g22ycf** and **g22ydf**.

10.1 Program Text

```
!   G22YBF Example Program Text
!   Mark 26.1 Release. NAG Copyright 2016.
    Module g22ybfe_mod
!     G22YBF Example Program Module:
!            Parameters and User-defined Routines

!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: construct_labels, fit_lm, read_line
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6

    Contains
      Subroutine read_line(v1)
!       Read in a line from NIN and remove any comments

!       .. Scalar Arguments ..
        Character (*), Intent (Out)    :: v1
!       .. Local Scalars ..
        Integer                        :: pend
!       .. Intrinsic Procedures ..
        Intrinsic                      :: adjustl, index
!       .. Executable Statements ..
        Continue

        Read (nin,'(A200)') v1
        pend = index(v1,'::')
        If (pend/=0) Then
          v1 = v1(1:pend-1)
        End If
        v1 = adjustl(v1)

        Return
      End Subroutine read_line
      Subroutine construct_labels(ip,plab,vnames,vinfo)
!       Construct a set of parameter labels using the information
!       stored in VINFO

!       .. Scalar Arguments ..
```

```
        Integer, Intent (In)           :: ip
!       .. Array Arguments ..
        Integer, Intent (In)           :: vinfo(:)
        Character (*), Allocatable, Intent (Out) :: plab(:)
        Character (*), Intent (In)     :: vnames(:)
!       .. Local Scalars ..
        Integer                        :: b, i, j, k, li, vi
        Character (200)                :: line, term
!       .. Intrinsic Procedures ..
        Intrinsic                      :: adjustl, trim
!       .. Executable Statements ..
        Continue

        Allocate (plab(ip))

        k = 1
        Do j = 1, ip
          b = vinfo(k)
          k = k + 1

          If (b==0) Then
            line = 'Intercept'
          Else
            line = ''
            Do i = 1, b
              vi = vinfo(k)
              li = vinfo(k+1)

              If (li/=0) Then
                Write (term,99999) trim(adjustl(vnames(vi))), li
              Else
                Write (term,99998) trim(adjustl(vnames(vi)))
              End If

              If (i==1) Then
                line = trim(line) // trim(term)
              Else
                line = trim(line) // ' . ' // trim(term)
              End If

!             We are ignoring the contrast identifier when constructing
!             these labels
              k = k + 3
            End Do
          End If
          plab(j) = line
        End Do
99999   Format (A,' (Level ',I0,')')
99998   Format (A)

        Return
      End Subroutine construct_labels
      Subroutine fit_lm(hform,intcpt,nobs,mx,x,isx,ip,y,plab)
!       Perform a multiple linear regression using G02DAF

!       .. Use Statements ..
        Use, Intrinsic                 :: iso_c_binding, Only: c_ptr
        Use nag_library, Only: g02daf, g22znf, nag_wp
!       .. Scalar Arguments ..
        Type (c_ptr), Intent (In)      :: hform
        Integer, Intent (In)           :: ip, mx, nobs
        Character (*), Intent (In)     :: intcpt
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x(:,:), y(:)
        Integer, Intent (In)           :: isx(:)
        Character (*), Intent (In)     :: plab(:)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: rss, rvalue, tol
        Integer                        :: i, idf, ifail, irank, ivalue, ldq,   &
                                          ldx, lwt, optype
        Logical                        :: svd
        Character (200)                 :: cvalue
        Character (1)                   :: weight
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: b(:), cov(:), h(:), p(:), q(:,:),   &
                                           res(:), se(:), wk(:), wt(:)
!       .. Intrinsic Procedures ..
        Intrinsic                       :: repeat, size, trim
!       .. Executable Statements ..
        Continue

        ldx = size(x,1)

!       We are assuming un-weighted data
        weight = 'U'
        lwt = 0
        ldq = nobs
        Allocate (b(ip),cov((ip*ip+ip)/2),h(nobs),p(ip*(ip+                    &
          2)),q(ldq,ip+1),res(nobs),se(ip),wk(ip*ip+5*(ip-1)),wt(lwt))

!       Use suggested value for tolerance
        tol = 0.000001E0_nag_wp

!       Fit a regression model
        ifail = 0
        Call g02daf(intcpt,weight,nobs,x,ldx,mx,isx,ip,y,wt,rss,idf,b,se,cov,  &
          res,h,q,ldq,svd,irank,p,tol,wk,ifail)

!       Get the formula for the model being fit
        ifail = 0
        Call g22znf(hform,'Formula',ivalue,rvalue,cvalue,optype,ifail)

!       Display the results
        Write (nout,*) 'Model: ', trim(cvalue)
        Write (nout,*) '                                Parameter   Standard'
        Write (nout,*) 'Coefficients                    Estimate      Error'
        Write (nout,*) repeat('-',51)
        Do i = 1, ip
          Write (nout,99997) plab(i), b(i), se(i)
        End Do
        Write (nout,*) repeat('-',51)
        Write (nout,99998) 'Residual sum of squares = ', rss
        Write (nout,99999) 'Degrees of freedom      = ', idf

        Return
99999   Format (1X,A,I9)
99998   Format (1X,A,F9.4)
99997   Format (1X,A30,1X,F7.3,5X,F7.3)
      End Subroutine fit_lm
    End Module g22ybfe_mod

    Program g22ybfe
!     G22YBF Example Program Text

!     .. Use Statements ..
      Use g22ybfe_mod, Only: construct_labels, fit_lm, nin, nout, read_line
      Use, Intrinsic                    :: iso_c_binding, Only: c_null_ptr,     &
                                           c_ptr
      Use nag_library, Only: g22yaf, g22ybf, g22ycf, g22ydf, g22zaf
      Use nag_precisions, Only: wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Type (c_ptr)                      :: hddesc, hform, hxdesc
      Integer                           :: i, ifail, ip, lddat, ldx, lisx,      &
                                           lplab, lvinfo, lvnames, mx, nobs,    &
                                           nvar, sddat, sdx
      Character (200)                   :: formula
      Character (1)                     :: intcpt
!     .. Local Arrays ..
      Real (Kind=wp), Allocatable       :: dat(:,:), x(:,:), y(:)
      Real (Kind=wp)                    :: tx(0,0)
      Integer, Allocatable             :: isx(:), levels(:), vinfo(:)
      Integer                          :: tisx(0), tvinfo(3)
      Character (50), Allocatable      :: plab(:), vnames(:)
      Character (1)                    :: tplab(0)
!     .. Executable Statements ..
      Write (nout,*) 'G22YBF Example Program Results'
      Write (nout,*)

      hform = c_null_ptr
      hddesc = c_null_ptr
      hxdesc = c_null_ptr

!     Skip heading in data file
      Read (nin,*)

!     Read in size of the data matrix and number of variable labels supplied
      Read (nin,*) nobs, nvar, lvnames

!     Read in number of levels and names for the variables
      Allocate (levels(nvar),vnames(lvnames))
      Read (nin,*) levels(1:nvar)
      If (lvnames>0) Then
        Read (nin,*) vnames(1:lvnames)
      End If

!     Create a description of the data matrix
      ifail = 0
      Call g22ybf(hddesc,nobs,nvar,levels,lvnames,vnames,ifail)

!     Read in the data matrix and response variable
      lddat = nobs
      sddat = nvar
      Allocate (dat(lddat,sddat),y(nobs))
      Read (nin,*)(dat(i,1:nvar),y(i),i=1,nobs)

!     Read in the model formula, remove comments and parse it
      Call read_line(formula)
      ifail = 0
      Call g22yaf(hform,formula,ifail)

!     Start of constructing the design matrix ...
!     Calculate the size of design matrix
      ldx = 0
      sdx = 0
      ifail = 1
      Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,tx,ldx,sdx,mx,ifail)
      If (ifail/=91) Then
!       redisplay the error message
        ifail = 0
        Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,tx,ldx,sdx,mx,ifail)
      End If

!     Generate the design matrix
      ldx = nobs
      sdx = mx
      Allocate (x(ldx,sdx))
      ifail = 0
      Call g22ycf(hform,hddesc,dat,lddat,sddat,hxdesc,x,ldx,sdx,mx,ifail)
!     ... End of constructing the design matrix

!     Start of getting the ISX vector and information on parameter labels ...
!     Get size of output arrays used by G22YDF
      lvinfo = 3
      lisx = 0
      lplab = 0
      ifail = 1
      Call g22ydf(hform,hxdesc,intcpt,ip,lisx,tisx,lplab,tplab,lvinfo,tvinfo,  &
        ifail)
      If (ifail/=92) Then
!       redisplay the error message
        ifail = 0
        Call g22ydf(hform,hxdesc,intcpt,ip,lisx,tisx,lplab,tplab,lvinfo,       &
          tvinfo,ifail)
      End If

!     Allocate output arrays (we already know that LISX = MX, but G22YDF
!     returns it just in case)
      lisx = tvinfo(1)
      lvinfo = tvinfo(3)
!     We don't need PLAB as we are constructing our own labels from VINFO
      Allocate (isx(lisx),vinfo(lvinfo))

!     Get the ISX flag and parameter labels
      ifail = 0
      Call g22ydf(hform,hxdesc,intcpt,ip,lisx,isx,lplab,tplab,lvinfo,vinfo,    &
        ifail)

!     Construct some verbose labels for the parameters
      Call construct_labels(ip,plab,vnames,vinfo)
!     ... End of getting the ISX vector and information on parameter labels

!     Fit a regression model and print the results
      Call fit_lm(hform,intcpt,nobs,mx,x,isx,ip,y,plab)

!     Clean-up the G22 handles
      ifail = 0
      Call g22zaf(hform,ifail)
      Call g22zaf(hddesc,ifail)
      Call g22zaf(hxdesc,ifail)

      Deallocate (dat,x,y)
      Deallocate (isx,levels,vinfo)
      Deallocate (plab,vnames)
    End Program g22ybfe
```

10.2 Program Data

```
G22YBF Example Program Data
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
F1 + F2 + Con + F1.F2 + F1.Con + F2.Con :: FORMULA
```

10.3 Program Results

```
G22YBF Example Program Results

Model: F1+F2+CON+F1.F2+F1.CON+F2.CON
                               Parameter   Standard
Coefficients                   Estimate      Error
---------------------------------------------------
Intercept                       3.902        0.618
F1 (Level 2)                   -2.197        2.060
F1 (Level 3)                    1.005        1.071
F2 (Level 2)                    4.094        1.008
F2 (Level 3)                    0.958        0.836
Con (Level 1)                   3.828        0.130
F1 (Level 2) . F2 (Level 2)     2.666        2.706
F1 (Level 2) . F2 (Level 3)     4.399        2.398
F1 (Level 3) . F2 (Level 2)     0.004        1.519
F1 (Level 3) . F2 (Level 3)    -0.756        1.385
F1 (Level 2) . Con (Level 1)   -1.327        0.270
F1 (Level 3) . Con (Level 1)   -1.810        0.252
F2 (Level 2) . Con (Level 1)   -0.079        0.207
F2 (Level 3) . Con (Level 1)    0.465        0.230
---------------------------------------------------
Residual sum of squares =    8.2462
Degrees of freedom      =        11
```

11 Optional Parameters

As well as the optional parameters common to all G22 handles described in **g22zmf** and **g22znf**, a number of additional optional parameters can be specified for a G22 handle holding the description of a data matrix as returned by **g22ybf** in **hddesc**.

Each writeable optional parameter has an associated default value; to set any of them to a non-default value, use **g22zmf**. The value of an optional parameter can be queried using **g22znf**.

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.

Number of Observations

Number of Variables

Storage Order

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.

Number of Observations i Read Only

If queried, this optional parameter will return n, the number of observations in the data matrix.

Number of Variables i Read Only

If queried, this optional parameter will return m_d, the number of variables in the data matrix.

Storage Order a Default $=$ OBSVAR

This optional parameter states how the data matrix, D, will be stored in its input array.

If **Storage Order** $=$ OBSVAR, D_{ij}, the value for the jth variable of the ith observation of the data matrix is stored in **dat**(i, j).

If **Storage Order** $=$ VAROBS, D_{ij}, the value for the jth variable of the ith observation of the data matrix is stored in **dat**(j, i).

Where **dat** is the input parameter of the same name in **g22ycf**.

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