

NAG Library Function Document

nag_ml_hier_mixed_regsn (g02jec)

1 Purpose

nag_ml_hier_mixed_regsn (g02jec) fits a multi-level linear mixed effects regression model using maximum likelihood (ML). Prior to calling nag_ml_hier_mixed_regsn (g02jec) the initialization function nag_hier_mixed_init (g02jcc) must be called.

2 Specification

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

void nag_ml_hier_mixed_regsn (Integer lvpr, const Integer vpr[],
    Integer nvpr, double gamma[], Integer *effn, Integer *rnkx,
    Integer *ncov, double *lnlike, Integer lb, Integer id[], Integer pdid,
    double b[], double se[], double czz[], Integer pdczz, double cxx[],
    Integer pdcxx, double cxz[], Integer pdcxz, const double rcomm[],
    const Integer icomm[], const Integer iopt[], Integer liopt,
    const double ropt[], Integer lropt, NagError *fail)
```

3 Description

nag_ml_hier_mixed_regsn (g02jec) fits a model of the form:

$$y = X\beta + Z\nu + \epsilon$$

where y is a vector of n observations on the dependent variable,

X is a known n by p design matrix for the *fixed* independent variables,

β is a vector of length p of unknown *fixed effects*,

Z is a known n by q design matrix for the *random* independent variables,

ν is a vector of length q of unknown *random effects*,

and ϵ is a vector of length n of unknown random errors.

Both ν and ϵ are assumed to have a Gaussian distribution with expectation zero and variance/covariance matrix defined by

$$\text{Var} \begin{bmatrix} \nu \\ \epsilon \end{bmatrix} = \begin{bmatrix} G & 0 \\ 0 & R \end{bmatrix}$$

where $R = \sigma_R^2 I$, I is the $n \times n$ identity matrix and G is a diagonal matrix. It is assumed that the random variables, Z , can be subdivided into $g \leq q$ groups with each group being identically distributed with expectation zero and variance σ_i^2 . The diagonal elements of matrix G therefore take one of the values $\{\sigma_i^2 : i = 1, 2, \dots, g\}$, depending on which group the associated random variable belongs to.

The model therefore contains three sets of unknowns: the fixed effects β , the random effects ν and a vector of $g + 1$ variance components γ , where $\gamma = \{\sigma_1^2, \sigma_2^2, \dots, \sigma_{g-1}^2, \sigma_g^2, \sigma_R^2\}$. Rather than working directly with γ , nag_ml_hier_mixed_regsn (g02jec) uses an iterative process to estimate $\gamma^* = \{\sigma_1^2/\sigma_R^2, \sigma_2^2/\sigma_R^2, \dots, \sigma_{g-1}^2/\sigma_R^2, \sigma_g^2/\sigma_R^2, 1\}$. Due to the iterative nature of the estimation a set of initial values, γ_0 , for γ^* is required. nag_ml_hier_mixed_regsn (g02jec) allows these initial values either to be supplied by you or calculated from the data using the minimum variance quadratic unbiased estimators (MIVQUE0) suggested by Rao (1972).

nag_ml_hier_mixed_regsn (g02jec) fits the model by maximizing the log-likelihood function:

$$-2l_R = \log(|V|) + n \log(r^T V^{-1} r) + \log(2\pi/n)$$

where

$$V = ZGZ^T + R, \quad r = y - Xb \quad \text{and} \quad b = (X^T V^{-1} X)^{-1} X^T V^{-1} y.$$

Once the final estimates for γ^* have been obtained, the value of σ_R^2 is given by

$$\sigma_R^2 = (r^T V^{-1} r)/(n - p).$$

Case weights, W_c , can be incorporated into the model by replacing $X^T X$ and $Z^T Z$ with $X^T W_c X$ and $Z^T W_c Z$ respectively, for a diagonal weight matrix W_c .

The log-likelihood, l_R , is calculated using the sweep algorithm detailed in Wolfinger *et al.* (1994).

4 References

- Goodnight J H (1979) A tutorial on the SWEEP operator *The American Statistician* **33**(3) 149–158
- Harville D A (1977) Maximum likelihood approaches to variance component estimation and to related problems *JASA* **72** 320–340
- Rao C R (1972) Estimation of variance and covariance components in a linear model *J. Am. Stat. Assoc.* **67** 112–115
- Stroup W W (1989) Predictable functions and prediction space in the mixed model procedure *Applications of Mixed Models in Agriculture and Related Disciplines Southern Cooperative Series Bulletin No. 343* 39–48
- Wolfinger R, Tobias R and Sall J (1994) Computing Gaussian likelihoods and their derivatives for general linear mixed models *SIAM Sci. Statist. Comput.* **15** 1294–1310

5 Arguments

Note: prior to calling nag_ml_hier_mixed_regsn (g02jec) the initialization function nag_hier_mixed_init (g02jcc) must be called, therefore this documentation should be read in conjunction with the document for nag_hier_mixed_init (g02jcc).

In particular some argument names and conventions described in that document are also relevant here, but their definition has not been repeated. Specifically, **RNDM**, **wt**, **n**, **nff**, **nrf**, **nlsv**, **levels**, **fixed**, **DAT**, **licomm** and **lrcomm** should be interpreted identically in both functions.

- 1: **lvpr** – Integer *Input*
On entry: the sum of the number of random parameters and the random intercept flags specified in the call to nag_hier_mixed_init (g02jcc).
Constraint: $lvpr = \sum_i \mathbf{RNDM}(1, i) + \mathbf{RNDM}(2, i)$.
- 2: **vpr[lvpr]** – const Integer *Input*
On entry: a vector of flags indicating the mapping between the random variables specified in **rndm** and the variance components, σ_i^2 . See Section 9 for more details.
Constraint: $1 \leq \mathbf{vpr}[i - 1] \leq \mathbf{nvpr}$, for $i = 1, 2, \dots, \mathbf{lvpr}$.
- 3: **nvpr** – Integer *Input*
On entry: g , the number of variance components being estimated (excluding the overall variance, σ_R^2).
Constraint: $1 \leq \mathbf{nvpr} \leq \mathbf{lvpr}$.

- 4: **gamma**[**nvpr** + 1] – double *Input/Output*
On entry: holds the initial values of the variance components, γ_0 , with **gamma**[*i* – 1] the initial value for σ_i^2/σ_R^2 for $i = 1, 2, \dots, \mathbf{nvpr}$.
 If **gamma**[0] = –1.0, the remaining elements of **gamma** are ignored and the initial values for the variance components are estimated from the data using MIVQUE0.
On exit: **gamma**[*i* – 1], for $i = 1, 2, \dots, \mathbf{nvpr}$, holds the final estimate of σ_i^2 and **gamma**[**nvpr**] holds the final estimate for σ_R^2 .
Constraint: **gamma**[0] = –1.0 or **gamma**[*i* – 1] ≥ 0.0 , for $i = 1, 2, \dots, g$.
- 5: **effn** – Integer * *Output*
On exit: effective number of observations. If there are no weights (i.e., **wt** is **NULL**), or all weights are nonzero, then **effn** = **n**.
- 6: **rnkx** – Integer * *Output*
On exit: the rank of the design matrix, *X*, for the fixed effects.
- 7: **ncov** – Integer * *Output*
On exit: number of variance components not estimated to be zero. If none of the variance components are estimated to be zero, then **ncov** = **nvpr**.
- 8: **lnlike** – double * *Output*
On exit: $-2l_R(\hat{\gamma})$ where l_R is the log of the maximum likelihood calculated at $\hat{\gamma}$, the estimated variance components returned in **gamma**.
- 9: **lb** – Integer *Input*
On entry: the dimension of the arrays **b** and **se**.
Constraint: **lb** $\geq \mathbf{nff} + \mathbf{nrf} \times \mathbf{nlsv}$.
- 10: **id**[**pdid** × **lb**] – Integer *Output*
Note: where **ID**(*i*, *j*) appears in this document, it refers to the array element **id**[(*j* – 1) × **pdid** + *i* – 1].
On exit: an array describing the parameter estimates returned in **b**. The first **nlsv** × **nrf** columns of **ID** describe the parameter estimates for the random effects and the last **nff** columns the parameter estimates for the fixed effects.
 The example program for this function includes a demonstration of decoding the parameter estimates given in **b** using information from **id**.
 For fixed effects:
 for $l = \mathbf{nrf} \times \mathbf{nlsv} + 1, \dots, \mathbf{nrf} \times \mathbf{nlsv} + \mathbf{nff}$
 if **b**[*l* – 1] contains the parameter estimate for the intercept then
 $\mathbf{ID}(1, l) = \mathbf{ID}(2, l) = \mathbf{ID}(3, l) = 0;$
 if **b**[*l* – 1] contains the parameter estimate for the *i*th level of the *j*th fixed variable, that is the vector of values held in the *k*th column of **DAT** when **fixed**[*j* + 1] = *k* then
 $\mathbf{ID}(1, l) = 0,$
 $\mathbf{ID}(2, l) = j,$
 $\mathbf{ID}(3, l) = i;$
 if the *j*th variable is continuous or binary, that is **levels**[**fixed**[*j* + 1] – 1] = 1, then
 $\mathbf{ID}(3, l) = 0;$

any remaining rows of the l th column of **ID** are set to 0.

For random effects:

let

N_{R_b} denote the number of random variables in the b th random statement, that is $N_{R_b} = \mathbf{RNDM}(1, b)$;

R_{jb} denote the j th random variable from the b th random statement, that is the vector of values held in the k th column of **DAT** when $\mathbf{RNDM}(2 + j, b) = k$;

N_{S_b} denote the number of subject variables in the b th random statement, that is $N_{S_b} = \mathbf{RNDM}(3 + N_{R_b}, b)$;

S_{jb} denote the j th subject variable from the b th random statement, that is the vector of values held in the k th column of **DAT** when $\mathbf{RNDM}(3 + N_{R_b} + j, b) = k$;

$L(S_{jb})$ denote the number of levels for S_{jb} , that is $L(S_{jb}) = \mathbf{levels}[\mathbf{RNDM}(3 + N_{R_b} + j, b) - 1]$;

then

for $l = 1, 2, \dots, \mathbf{nrf} \times \mathbf{nlsv}$, if $\mathbf{b}[l - 1]$ contains the parameter estimate for the i th level of R_{jb} when $S_{kb} = s_k$, for $k = 1, 2, \dots, N_{S_b}$ and $1 \leq s_k \leq L(S_{jb})$, i.e., s_k is a valid value for the k th subject variable, then

$$\begin{aligned}\mathbf{ID}(1, l) &= b, \\ \mathbf{ID}(2, l) &= j, \\ \mathbf{ID}(3, l) &= i, \\ \mathbf{ID}(3 + k, l) &= s_k, \quad k = 1, 2, \dots, N_{S_b};\end{aligned}$$

if the parameter being estimated is for the intercept then $\mathbf{ID}(2, l) = \mathbf{ID}(3, l) = 0$;

if the j th variable is continuous, or binary, that is $L(S_{jb}) = 1$, then $\mathbf{ID}(3, l) = 0$;

the remaining rows of the l th column of **ID** are set to 0.

In some situations, certain combinations of variables are never observed. In such circumstances all elements of the l th row of **ID** are set to -999.

11: **pdid** – Integer Input

On entry: the stride separating matrix row elements in the array **id**.

Constraint: $\mathbf{pdid} \geq 3 + \max_j(\mathbf{RNDM}(3 + \mathbf{RNDM}(1, j), j))$, i.e., $3 +$ maximum number of subject variables (see `nag_hier_mixed_init` (g02jcc)).

12: **b[lb]** – double Output

On exit: the parameter estimates, with the first $\mathbf{nrf} \times \mathbf{nlsv}$ elements of **b** containing the parameter estimates for the random effects, ν , and the remaining **nff** elements containing the parameter estimates for the fixed effects, β . The order of these estimates are described by the **id** argument.

13: **se[lb]** – double Output

On exit: the standard errors of the parameter estimates given in **b**.

14: **czz[dim]** – double Output

Note: the dimension, *dim*, of the array **czz** must be at least $\mathbf{pdczz} \times \mathbf{nrf} \times \mathbf{nlsv}$.

Where $\mathbf{CZZ}(i, j)$ appears in this document, it refers to the array element $\mathbf{czz}[(j - 1) \times \mathbf{pdczz} + i - 1]$.

On exit: if $\mathbf{nlsv} = 1$, then **CZZ** holds the lower triangular portion of the matrix $(1/\sigma^2)(Z^T \hat{R}^{-1} Z + \hat{G}^{-1})$, where \hat{R} and \hat{G} are the estimates of R and G respectively. If

nlsv > 1 then **CZZ** holds this matrix in compressed form, with the first **nrf** columns holding the part of the matrix corresponding to the first level of the overall subject variable, the next **nrf** columns the part corresponding to the second level of the overall subject variable etc.

15: **pdczz** – Integer Input

On entry: the stride separating matrix row elements in the array **czz**.

Constraint: **pdczz** ≥ **nrf**.

16: **cxx**[*dim*] – double Output

Note: the dimension, *dim*, of the array **cxx** must be at least **pdcxx** × **nff**.

Where **CXX**(*i*, *j*) appears in this document, it refers to the array element **cxx**[(*j* − 1) × **pdcxx** + *i* − 1].

On exit: **CXX** holds the lower triangular portion of the matrix $(1/\sigma^2)X^T\hat{V}^{-1}X$, where \hat{V} is the estimated value of *V*.

17: **pdcxx** – Integer Input

On entry: the stride separating matrix row elements in the array **cxx**.

Constraint: **pdcxx** ≥ **nff**.

18: **cxz**[*dim*] – double Output

Note: the dimension, *dim*, of the array **cxz** must be at least **pdcxz** × **nlsv** × **nrf**.

Where **CXZ**(*i*, *j*) appears in this document, it refers to the array element **cxz**[(*j* − 1) × **pdcxz** + *i* − 1].

On exit: if **nlsv** = 1, then **CXZ** holds the matrix $(1/\sigma^2)(X^T\hat{V}^{-1}Z)\hat{G}$, where \hat{V} and \hat{G} are the estimates of *V* and *G* respectively. If **nlsv** > 1 then **CXZ** holds this matrix in compressed form, with the first **nrf** columns holding the part of the matrix corresponding to the first level of the overall subject variable, the next **nrf** columns the part corresponding to the second level of the overall subject variable etc.

19: **pdcxz** – Integer Input

On entry: the stride separating matrix row elements in the array **cxz**.

Constraint: **pdcxz** ≥ **nff**.

20: **rcomm**[*dim*] – const double Communication Array

Note: the dimension, *dim*, of the array **rcomm** must be at least **lrcomm**.

On entry: communication array initialized by a call to nag_hier_mixed_init (g02jcc).

21: **icomm**[*dim*] – const Integer Communication Array

Note: the dimension, *dim*, of the array **icomm** must be at least **licomm**.

On entry: communication array initialized by a call to nag_hier_mixed_init (g02jcc).

22: **iopt**[**liopt**] – const Integer Input

On entry: optional parameters passed to the optimization function.

By default nag_ml_hier_mixed_regsn (g02jec) fits the specified model using a modified Newton optimization algorithm as implemented in the NAG Fortran Library routine E04LBF. In some cases, where the calculation of the derivatives is computationally expensive it may be more efficient to use a sequential QP algorithm. The sequential QP algorithm as implemented in the

NAG Fortran Library routine E04UCF can be chosen by setting **iopt**[4] = 1. If **liopt** < 5 or **iopt**[4] ≠ 1 then E04LBF will be used.

Different optional parameters are available depending on the optimization function used. In all cases, using a value of −1 will cause the default value to be used. In addition only the first **liopt** values of **iopt** are used, so for example, if only the first element of **iopt** needs changing and default values for all other optional parameters are sufficient **liopt** can be set to 1.

NAG Fortran Library routine E04LBF is being used.

<i>i</i>	Description	Equivalent E04LBF argument	Default Value
0	Number of iterations	MAXCAL	1000
1	Unit number for monitoring information. See nag_o pen_file (x04acc) for details on how to assign a file to a unit number.	n/a	Output sent to stdout
2	Print optional parameters (1 = print)	n/a	−1 (no printing performed)
3	Frequency that monitoring information is printed	IPRINT	−1
4	Optimizer used	n/a	n/a

If requested, monitoring information is displayed in a similar format to that given by E04LBF.

NAG Fortran Library routine E04UCF is being used.

<i>i</i>	Description	Equivalent E04UCF argument	Default Value
0	Number of iterations	Major Iteration Limit	max (50, 3 × nvpr)
1	Unit number for monitoring information. See nag_o pen_file (x04acc) for details on how to assign a file to a unit number.	n/a	Output sent to stdout
2	Print optional parameters (1 = print, otherwise no print)	List/ NoList	−1 (no printing performed)
3	Frequency that monitoring information is printed	Major Print Level	0
4	Optimizer used	n/a	n/a
5	Number of minor iterations	Minor Iteration Limit	max (50, 3 × nvpr)
6	Frequency that additional monitoring information is printed	Minor Print Level	0

If **liopt** ≤ 0 then default values are used for all optional parameters and **iopt** may be set to **NULL**.

23: **liopt** – Integer *Input*

On entry: length of the options array **iopt**.

24: **ropt**[**lropt**] – const double *Input*

On entry: optional parameters passed to the optimization function.

Different optional parameters are available depending on the optimization function used. In all cases, using a value of −1.0 will cause the default value to be used. In addition only the first **lropt** values of **ropt** are used, so for example, if only the first element of **ropt** needs changing and default values for all other optional parameters are sufficient **lropt** can be set to 1.

NAG Fortran Library routine E04LBF is being used.

<i>i</i>	Description	Equivalent E04LBF argument	Default Value
0	Sweep tolerance	n/a	$\max \left(\sqrt{\text{eps}}, \sqrt{\text{eps}} \times \max_i (zz_{ii}) \right)$
1	Lower bound for γ^*	n/a	eps/100
2	Upper bound for γ^*	n/a	10^{20}

3	Accuracy of linear minimizations	ETA	0.9
4	Accuracy to which solution is required	XTOL	0.0
5	Initial distance from solution	STEPMX	100000.0

NAG Fortran Library routine E04UCF is being used.

<i>i</i>	Description	Equivalent E04UCF argument	Default Value
0	Sweep tolerance	n/a	$\max\left(\sqrt{\text{eps}}, \sqrt{\text{eps}} \times \max_i(\text{zz}_{ii})\right)$
1	Lower bound for γ^*	n/a	eps/100
2	Upper bound for γ^*	n/a	10^{20}
3	Line search tolerance	Line Search Tolerance	0.9
4	Optimality tolerance	Optimality Tolerance	$\text{eps}^{0.72}$

where eps is the *machine precision* returned by nag_machine_precision (X02AJC) and zz_{ii} denotes the i diagonal element of $Z^T Z$.

If **lropt** ≤ 0 then default values are used for all optional parameters and **ropt** and may be set to **NULL**.

- 25: **lropt** – Integer *Input*
On entry: length of the options array **ropt**.
- 26: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.7 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 $\langle \text{value} \rangle$ had an illegal value.

NE_INT

On entry, **lb** = $\langle \text{value} \rangle$.

Constraint: **lb** $\geq \langle \text{value} \rangle$.

On entry, **lvpr** = $\langle \text{value} \rangle$.

Constraint: **lvpr** $\geq \langle \text{value} \rangle$.

On entry, **nvpr** = $\langle \text{value} \rangle$.

Constraint: $1 \leq \text{nvpr} \leq \langle \text{value} \rangle$.

On entry, **pdcxx** = $\langle \text{value} \rangle$.

Constraint: **pdcxx** $\geq \langle \text{value} \rangle$.

On entry, **pdcxz** = $\langle \text{value} \rangle$.

Constraint: **pdcxz** $\geq \langle \text{value} \rangle$.

On entry, **pdczz** = $\langle \text{value} \rangle$.

Constraint: **pdczz** $\geq \langle \text{value} \rangle$.

On entry, **pdid** = $\langle value \rangle$.
 Constraint: **pdid** $\geq \langle value \rangle$.

NE_INT_ARRAY

On entry, at least one value of i , for $i = 1, 2, \dots, \mathbf{nvpr}$, does not appear in **vpr**.

On entry, **icomm** has not been initialized correctly.

On entry, **vpr**[$\langle value \rangle$] = $\langle value \rangle$ and **nvpr** = $\langle value \rangle$.
 Constraint: $1 \leq \mathbf{vpr}[i - 1] \leq \mathbf{nvpr}$.

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.

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

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NEG_ELEMENT

At least one negative estimate for **gamma** was obtained. All negative estimates have been set to zero.

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_REAL_ARRAY

On entry, **gamma**[$\langle value \rangle$] = $\langle value \rangle$.
 Constraint: **gamma**[0] = -1.0 or **gamma**[$i - 1$] ≥ 0.0 .

NW_KT_CONDITIONS

Current point cannot be improved upon.

NW_NOT_CONVERGED

Optimal solution found, but requested accuracy not achieved.

NW_TOO_MANY_ITER

Too many major iterations.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_ml_hier_mixed_regsn (g02jec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_ml_hier_mixed_regsn (g02jec) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

The argument **vpr** gives the mapping between the random variables and the variance components. In most cases $\mathbf{vpr}[i-1] = i$, for $i = 1, 2, \dots, \sum_i \mathbf{RNDM}(1, i) + \mathbf{RNDM}(2, i)$. However, in some cases it might be necessary to associate more than one random variable with a single variance component, for example, when the columns of **DAT** hold dummy variables.

Consider a dataset with three variables:

$$\mathbf{DAT} = \begin{pmatrix} 1 & 1 & 3.6 \\ 2 & 1 & 4.5 \\ 3 & 1 & 1.1 \\ 1 & 2 & 8.3 \\ 2 & 2 & 7.2 \\ 3 & 2 & 6.1 \end{pmatrix}$$

where the first column corresponds to a categorical variable with three levels, the next to a categorical variable with two levels and the last column to a continuous variable. So in a call to `nag_hier_mixed_init (g02jec)`

$$\mathbf{levels} = (3 \quad 2 \quad 1)$$

also assume a model with no fixed effects, no random intercept, no nesting and all three variables being included as random effects, then

$$\begin{aligned} \mathbf{fixed} &= (0 \quad 0); \\ \mathbf{RNDM} &= (3 \quad 0 \quad 1 \quad 2 \quad 3)^T. \end{aligned}$$

Each of the three columns in **DAT** therefore correspond to a single variable and hence there are three variance components, one for each random variable included in the model, so

$$\mathbf{vpr} = (1 \quad 2 \quad 3).$$

This is the recommended way of supplying the data to `nag_ml_hier_mixed_regsn (g02jec)`, however it is possible to reformat the above dataset by replacing each of the categorical variables with a series of dummy variables, one for each level. The dataset then becomes

$$\mathbf{DAT} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 3.6 \\ 0 & 1 & 0 & 1 & 0 & 4.5 \\ 0 & 0 & 1 & 1 & 0 & 1.1 \\ 1 & 0 & 0 & 0 & 1 & 8.3 \\ 0 & 1 & 0 & 0 & 1 & 7.2 \\ 0 & 0 & 1 & 0 & 1 & 6.1 \end{pmatrix}$$

where each column only has one level

$$\mathbf{levels} = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1).$$

Again a model with no fixed effects, no random intercept, no nesting and all variables being included as random effects is required, so

$$\begin{aligned} \mathbf{fixed} &= (0 \quad 0); \\ \mathbf{RNDM} &= (6 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6)^T. \end{aligned}$$

With the data entered in this manner, the first three columns of **DAT** correspond to a single variable (the first column of the original dataset) as do the next two columns (the second column of the original dataset). Therefore **vpr** must reflect this

$$\mathbf{vpr} = (1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3).$$

In most situations it is more efficient to supply the data to `nag_hier_mixed_init (g02jec)` in terms of categorical variables rather than transform them into dummy variables.

10 Example

This example fits a random effects model with three levels of nesting to a simulated dataset with 90 observations and 12 variables.

10.1 Program Text

```
/* nag_ml_hier_mixed_regsn (g02jec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>

void print_results(Nag_OrderType order, Integer n, Integer nff, Integer nlsv,
                  Integer nrf, Integer fixed[], Integer nrndm,
                  Integer rndm[], Integer lrndm, Integer nvpr,
                  Integer vpr[], double gamma[], Integer effn,
                  Integer rnkx, Integer ncov, double lnlike,
                  Integer id[], Integer pddid, double b[], double se[]);

#define RNDM(I, J) rndm[(order == Nag_ColMajor) \
                        ?((J-1)*lrndm+I-1):((I-1)*nrndm+J-1)]
#define DAT(I, J)  dat[(order == Nag_ColMajor) \
                        ?((J-1)*pddat+I-1):((I-1)*pddat+J-1)]
#define ID(I, J)   id[((J-1)*pddid+I-1)]

int main(void)
{
    /* IO file pointers */

    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer pddid, licomm, lrcomm, tdczz, lb, pdcxx, pdcxz, pdczz, pddat,
        effn, i, j, lvpr, n, ncol, ncov, lfixed, nff, nl, nlsv, nrndm,
        nrf, nv, nvpr, rnkx, lwt, size_dat, lrndm;
    Integer *fixed = 0, *icomm = 0, *id = 0, *levels = 0, *rndm = 0;
    Integer *vpr = 0;
    Integer ticomm[2];

    /* NAG structures */
    NagError fail;
    Nag_OrderType order = Nag_RowMajor;

    /* Double scalar and array declarations */
    double lnlike;
    double *b = 0, *cxx = 0, *cxz = 0, *czz = 0, *dat = 0, *gamma = 0;
    double *rcomm = 0, *se = 0, *wt = 0, *y = 0;
    double trcomm[1];

    /* Character scalars */
    char weight;

    /* Use the default options */
    Integer *iopt = 0;
    Integer liopt = 0;
    double *ropt = 0;
    Integer lropt = 0;

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

```

printf("nag_ml_hier_mixed_regsn (g02jec) Example Program Results\n\n");

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

/* Read in the initial arguments */
#ifdef _WIN32
scanf_s("%c%NAG_IFMT %NAG_IFMT %NAG_IFMT %NAG_IFMT %*[\n] ",
        &weight, 1, &n, &ncol, &nrndm, &nvpr);
#else
scanf("%c%NAG_IFMT %NAG_IFMT %NAG_IFMT %NAG_IFMT %*[\n] ",
        &weight, &n, &ncol, &nrndm, &nvpr);
#endif

/* Maximum size for fixed and rndm */
lfixed = ncol + 2;
lrndm = 2 * ncol + 3;

if (order == Nag_ColMajor) {
    pddat = n;
    size_dat = pddat * ncol;
}
else {
    pddat = ncol;
    size_dat = pddat * n;
}

/* Allocate some memory */
if (!(y = NAG_ALLOC(n, double)) ||
    !(vpr = NAG_ALLOC(nvpr, Integer)) ||
    !(levels = NAG_ALLOC(ncol, Integer)) ||
    !(gamma = NAG_ALLOC(nvpr + 1, double)) ||
    !(fixed = NAG_ALLOC(lfixed, Integer)) ||
    !(rndm = NAG_ALLOC(lrndm * nrndm, Integer)) ||
    !(dat = NAG_ALLOC(size_dat, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Check whether we are supplying weights and
   allocate memory if required */
if (weight == 'W') {
    lwt = n;
    if (!(wt = NAG_ALLOC(lwt, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else {
    lwt = 0;
}

/* Read in the number of levels associated with each of the
   independent variables */
for (i = 0; i < ncol; 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

    /* Read in the fixed part of the model */
    /* Skip the heading */
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif
    /* Number of variables */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\\n] ", &fixed[0]);
#else
    scanf("%" NAG_IFMT "%*[^\\n] ", &fixed[0]);
#endif
    nv = fixed[0];
    if (nv + 2 > lfixed) {
        printf(" ** Problem size too large, increase array sizes\\n");
        printf("LFIXED,NV+2 = %" NAG_IFMT " %" NAG_IFMT "\\n", lfixed, nv + 2);
        exit_status = -1;
        goto END;
    }
    /* Intercept */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\\n] ", &fixed[1]);
#else
    scanf("%" NAG_IFMT "%*[^\\n] ", &fixed[1]);
#endif
    /* Variable IDs */
    if (nv > 0) {
        for (i = 2; i < nv + 2; i++)
#ifdef _WIN32
            scanf_s("%" NAG_IFMT "", &fixed[i]);
#else
            scanf("%" NAG_IFMT "", &fixed[i]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif
    }

    /* Read in the random part of the model */
    lvpr = 0;
    pdid = 0;
    for (j = 1; j <= nrndm; j++) {
#ifdef _WIN32
        scanf_s("%*[^\\n] ");
#else
        scanf("%*[^\\n] ");
#endif
    }
    /* Number of variables */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\\n] ", &RNDM(1, j));
#else
    scanf("%" NAG_IFMT "%*[^\\n] ", &RNDM(1, j));
#endif
    nv = RNDM(1, j);
    if ((nv + 3) > lrndm) {
        printf(" ** Problem size too large, increase array sizes\\n");
        printf("LRNDM,NV+2 = %" NAG_IFMT " %" NAG_IFMT "\\n", lrndm, nv + 2);
        exit_status = -1;
        goto END;
    }
    /* Intercept */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\\n] ", &RNDM(2, j));
#else

```

```

        scanf("%" NAG_IFMT "%*[\n] ", &RNDM(2, j));
#endif
    /* Variable IDs */
    if (nv > 0) {
        for (i = 3; i <= nv + 2; i++)
#ifdef _WIN32
            scanf_s("%" NAG_IFMT "", &RNDM(i, j));
#else
            scanf("%" NAG_IFMT "", &RNDM(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
    }
    /* Number of subject variables */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n] ", &RNDM(nv + 3, j));
#else
    scanf("%" NAG_IFMT "%*[\n] ", &RNDM(nv + 3, j));
#endif
    nl = RNDM(nv + 3, j);
    if (nv + nl + 2 > lrndm) {
        printf(" ** Problem size too large, increase array sizes\n");
        printf("LRNDM,NV+NL+2 = %" NAG_IFMT " %" NAG_IFMT "\n",
            lrndm, nv + nl + 2);
        exit_status = -1;
        goto END;
    }
    /* Subject variable IDs */
    if (nl > 0) {
        for (i = nv + 4; i <= nv + nl + 3; i++)
#ifdef _WIN32
            scanf_s("%" NAG_IFMT "", &RNDM(i, j));
#else
            scanf("%" NAG_IFMT "", &RNDM(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
    }
    pdid = MAX(pdid, nl);
    lvpr += RNDM(2, j) + nv;
}
pdid += 3;

/* Read in the dependent and independent data */
for (i = 1; i <= n; i++) {
#ifdef _WIN32
    scanf_s("%lf", &y[i - 1]);
#else
    scanf("%lf", &y[i - 1]);
#endif
    for (j = 1; j <= ncol; j++)
#ifdef _WIN32
        scanf_s("%lf", &DAT(i, j));
#else
        scanf("%lf", &DAT(i, j));
#endif
    if (lwt > 0)
#ifdef _WIN32
        scanf_s("%lf", &wt[i - 1]);
#else
        scanf("%lf", &wt[i - 1]);
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else

```

```

        scanf("%*[^\\n] ");
#endif
    }

    /* Read in VPR */
    for (i = 0; i < lvpr; i++)
#ifdef _WIN32
        scanf_s("%" NAG_IFMT "", &vpr[i]);
#else
        scanf("%" NAG_IFMT "", &vpr[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif

    /* Read in GAMMA */
    for (i = 0; i < nvpr; i++)
#ifdef _WIN32
        scanf_s("%lf", &gamma[i]);
#else
        scanf("%lf", &gamma[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif

    /* Get the size of the communication arrays */
    licomm = 2;
    lrcomm = 1;
    nag_hier_mixed_init(order, n, ncol, dat, pddat, levels, y, wt, fixed,
                        lfixed, nrndm, rndm, lrndm, &nff, &nlsv, &nrf, trcomm,
                        lrcomm, ticomm, licomm, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_hier_mixed_init (g02jcc).\\n%s\\n", fail.message);
        exit_status = 1;
        goto END;
    }
    licomm = ticomm[0];
    lrcomm = ticomm[1];

    /* Allocate the communication arrays */
    if (!(icomm = NAG_ALLOC(licomm, Integer)) ||
        !(rcomm = NAG_ALLOC(lrcomm, double)))
    {
        printf("Allocation failure 4\\n");
        exit_status = -1;
        goto END;
    }

    /* Pre-process the data */
    nag_hier_mixed_init(order, n, ncol, dat, pddat, levels, y, wt, fixed,
                        lfixed, nrndm, rndm, lrndm, &nff, &nlsv, &nrf, rcomm,
                        lrcomm, icomm, licomm, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_hier_mixed_init (g02jcc).\\n%s\\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Allocate the output arrays */
    lb = nff + nrf * nlsv;
    tdczz = nrf * nlsv;
    pdcxx = nff;
    pdcxz = nff;
    pdczz = nrf;
    if (!(b = NAG_ALLOC(lb, double)) ||
        !(cxx = NAG_ALLOC(pdcxx * nff, double)) ||

```

```

        !(cxz = NAG_ALLOC(pdcxz * tdczz, double)) ||
        !(czz = NAG_ALLOC(pdczz * tdczz, double)) ||
        !(se = NAG_ALLOC(lb, double)) || !(id = NAG_ALLOC(pdid * lb, Integer)))
    {
        printf("Allocation failure 5\n");
        exit_status = -1;
        goto END;
    }

    /* Perform the analysis */
    nag_ml_hier_mixed_regsn(lvpr, vpr, nvpr, gamma, &effn, &rnkx, &ncov,
                           &lnlike, lb, id, pdid, b, se, czz, pdczz, cxx,
                           pdcxx, cxz, pdcxz, rcomm, icomm, iopt, liopt, ropt,
                           lropt, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_ml_hier_mixed_regsn (g02jec).\n%s\n",
              fail.message);
        exit_status = 1;
        if (fail.code != NW_NOT_CONVERGED && fail.code != NW_TOO_MANY_ITER &&
            fail.code != NW_KT_CONDITIONS && fail.code != NE_NEG_ELEMENT)
            goto END;
    }

    /* Display the output */
    print_results(order, n, nff, nlsv, nrf, fixed, nrndm, rndm, lrndm, nvpr,
                 vpr, gamma, effn, rnkx, ncov, lnlike, id, pdid, b, se);

END:

    NAG_FREE(wt);
    NAG_FREE(y);
    NAG_FREE(vpr);
    NAG_FREE(levels);
    NAG_FREE(gamma);
    NAG_FREE(fixed);
    NAG_FREE(rndm);
    NAG_FREE(dat);
    NAG_FREE(icom);
    NAG_FREE(rcomm);
    NAG_FREE(b);
    NAG_FREE(cxx);
    NAG_FREE(cxz);
    NAG_FREE(czz);
    NAG_FREE(se);
    NAG_FREE(id);

    return exit_status;
}

void print_results(Nag_OrderType order, Integer n, Integer nff, Integer nlsv,
                  Integer nrf, Integer fixed[], Integer nrndm,
                  Integer rndm[], Integer lrndm, Integer nvpr,
                  Integer vpr[], double gamma[], Integer effn,
                  Integer rnkx, Integer ncov, double lnlike,
                  Integer id[], Integer pdid, double b[], double se[])
{
    Integer aid, i, k, l, ns, nv, p, pb, tb, tdid, vid, same;

    /* Display the output */
    printf(" Number of observations (N)                = %" NAG_IFMT "\n",
           n);
    printf(" Number of random factors (NRF)            = %" NAG_IFMT "\n",
           nrf);
    printf(" Number of fixed factors (NFF)                    = %" NAG_IFMT "\n",
           nff);
    printf(" Number of subject levels (NLSV)                  = %" NAG_IFMT "\n",
           nlsv);
    printf(" Rank of X (RNKX)                                  = %" NAG_IFMT "\n",
           rnkx);
    printf(" Effective N (EFFN)                                = %" NAG_IFMT "\n",
           effn);

```

```

printf(" Number of nonzero variance components (NCOV) =  %" NAG_IFMT "\n",
      ncov);

printf(" Parameter Estimates\n");
tdid = nff + nrf * nlsv;

if (nrf > 0) {
    printf("\n");
    printf(" Random Effects\n");
}

pb = -999;
for (k = 1; k <= nrf * nlsv; k++) {
    tb = ID(1, k);
    if (tb != -999) {
        vid = ID(2, k);
        nv = RNDM(1, tb);
        ns = RNDM(3 + nv, tb);

        if (pb != tb) {
            same = 0;
        }
        else {
            same = 1;
            for (l = 1; l <= ns; l++) {
                if (ID(3 + l, k) != ID(3 + l, k - 1)) {
                    same = 0;
                    break;
                }
            }
        }

        if (!same) {
            if (k != 1)
                printf("\n");
            printf(" Subject: ");
            for (l = 1; l <= ns; l++)
                printf(" Variable %2" NAG_IFMT " (Level %1" NAG_IFMT " ) ",
                    RNDM(3 + nv + l, tb), ID(3 + l, k));
            printf("\n");
        }
        pb = tb;

        if (vid == 0) {
            /* Intercept */
            printf("      Intercept                %10.4f %10.4f\n", b[k], se[k]);
        }
        else {
            /* VID'th variable specified in RNDM */
            aid = RNDM(2 + vid, tb);
            if (ID(3, k) == 0) {
                printf("      Variable %2" NAG_IFMT "", aid);
                printf("                %10.4f %10.4f\n", b[k - 1], se[k - 1]);
            }
            else {
                printf("      Variable %2" NAG_IFMT "", aid);
                printf(" (Level %1" NAG_IFMT " )                %10.4f %10.4f\n",
                    ID(3, k), b[k - 1], se[k - 1]);
            }
        }
    }
}

if (nff > 0) {
    printf("\n");
    printf(" Fixed Effects\n");
}

for (k = nrf * nlsv + 1; k <= tdid; k++) {
    vid = ID(2, k);
    if (vid != -999) {
        if (vid == 0) {

```



```

        /* Intercept */
        printf("      Intercept                      %10.4f %10.4f\n",
              b[k - 1], se[k - 1]);
    }
    else {
        /* VID'th variable specified in FIXED */
        aid = fixed[2 + vid - 1];
        if (ID(3, k) == 0) {
            printf("      Variable %2" NAG_IFMT " ", aid);
            printf("                      %10.4f %10.4f\n", b[k - 1], se[k - 1]);
        }
        else {
            printf("      Variable %2" NAG_IFMT " ", aid);
            printf(" (Level %1" NAG_IFMT " )                      %10.4f %10.4f\n",
                  ID(3, k), b[k - 1], se[k - 1]);
        }
    }
}
}

printf("\n");
printf(" Variance Components\n");
printf("      Estimate      Parameter      Subject\n");
for (k = 1; k <= nvpr; k++) {
    printf("%10.5f      ", gamma[k - 1]);
    p = 0;
    for (tb = 1; tb <= nrndm; tb++) {
        nv = RNDM(1, tb);
        ns = RNDM(3 + nv, tb);
        for (i = 1; i <= nv + RNDM(2, tb); i++) {
            p++;
            if (vpr[p - 1] == k) {
                printf("Variable %2" NAG_IFMT "      Variables ", RNDM(2 + i, tb));
                for (l = 1; l <= ns; l++)
                    printf("%2" NAG_IFMT " ", RNDM(3 + nv + l, tb));
            }
        }
    }
}
printf("\n");
}
printf("\n");
printf("SIGMA**2      = %15.5f\n", gamma[nvpr]);
printf("-2LOG LIKELIHOOD = %15.5f\n", lnlike);
}

```

10.2 Program Data

```

nag_ml_hier_mixed_regsn (g02jec) Example Program Data
U 90 12 3 7      :: WEIGHT (U = no weights),N,NCOL,NRAND,NVPR
2 3 2 3 2 3 1 4 5 2 3 3      :: LEVELS(1:NCOL)
## FIXED
2      :: number of variables
1      :: intercept
1 2      :: variable IDs
## RANDOM 1
2      :: number of variables
0      :: intercept
3 4      :: variable IDs
3      :: number of subject variables
10 11 12      :: subject variable IDs
## RANDOM 2
2      :: number of variables
0      :: intercept
5 6      :: variable IDs
2      :: number of subject variables
11 12      :: subject variable IDs
## RANDOM 3
3      :: number of variables
0      :: intercept
7 8 9      :: variable IDs

```

```

1          :: number of subject variables
12         :: subject variable IDs
  3.1100 1.0 3.0 2.0 1.0 2.0 2.0 -0.3160 4.0 2.0 1.0 1.0 1.0
  2.8226 1.0 1.0 1.0 3.0 1.0 2.0 -1.3377 1.0 4.0 1.0 1.0 1.0
  7.4543 1.0 3.0 1.0 3.0 1.0 3.0 -0.7610 4.0 2.0 1.0 1.0 1.0
  4.4313 2.0 3.0 2.0 1.0 1.0 3.0 -2.2976 4.0 2.0 1.0 1.0 1.0
  6.1543 2.0 2.0 1.0 3.0 2.0 3.0 -0.4263 2.0 1.0 1.0 1.0 1.0
-0.1783 2.0 1.0 2.0 3.0 1.0 3.0  1.4067 3.0 3.0 2.0 1.0 1.0
  4.6748 2.0 3.0 2.0 1.0 2.0 1.0 -1.4669 1.0 2.0 2.0 1.0 1.0
  7.0667 1.0 1.0 1.0 3.0 2.0 3.0  0.4717 2.0 4.0 2.0 1.0 1.0
  1.4262 1.0 3.0 2.0 3.0 2.0 1.0  0.4436 1.0 3.0 2.0 1.0 1.0
  7.7290 1.0 1.0 1.0 2.0 2.0 3.0 -0.5950 3.0 4.0 2.0 1.0 1.0
-2.1806 1.0 3.0 1.0 3.0 1.0 1.0 -1.7981 4.0 2.0 1.0 2.0 1.0
  6.8419 2.0 3.0 1.0 2.0 1.0 1.0  0.2397 1.0 4.0 1.0 2.0 1.0
  1.2590 1.0 2.0 2.0 1.0 2.0 3.0  0.4742 1.0 1.0 1.0 2.0 1.0
  8.8405 2.0 2.0 2.0 2.0 2.0 3.0  0.6888 3.0 1.0 1.0 2.0 1.0
  6.1657 2.0 1.0 2.0 3.0 1.0 3.0 -1.0616 3.0 5.0 1.0 2.0 1.0
-4.5605 1.0 2.0 2.0 2.0 2.0 1.0 -0.5356 1.0 3.0 2.0 2.0 1.0
-1.2367 1.0 3.0 2.0 2.0 1.0 1.0 -1.2963 2.0 5.0 2.0 2.0 1.0
-12.2932 1.0 2.0 2.0 1.0 2.0 2.0 -1.5389 3.0 2.0 2.0 2.0 1.0
-2.3374 2.0 3.0 1.0 1.0 2.0 2.0 -0.6408 2.0 1.0 2.0 2.0 1.0
  0.0716 1.0 2.0 2.0 2.0 1.0 1.0  0.6574 1.0 1.0 2.0 2.0 1.0
  0.1895 2.0 1.0 1.0 1.0 1.0 3.0  0.9259 1.0 2.0 1.0 3.0 1.0
  1.5608 2.0 2.0 2.0 1.0 2.0 2.0  1.5080 3.0 1.0 1.0 3.0 1.0
-0.8529 2.0 3.0 1.0 1.0 1.0 3.0  2.5821 2.0 3.0 1.0 3.0 1.0
-4.1169 1.0 2.0 2.0 1.0 2.0 3.0  0.4102 1.0 4.0 1.0 3.0 1.0
  3.9977 2.0 1.0 2.0 3.0 2.0 2.0  0.7839 2.0 5.0 1.0 3.0 1.0
-8.1277 1.0 2.0 2.0 3.0 2.0 1.0 -1.8812 4.0 2.0 2.0 3.0 1.0
-4.9656 1.0 2.0 1.0 3.0 2.0 3.0  0.7770 4.0 1.0 2.0 3.0 1.0
-0.6428 2.0 2.0 1.0 2.0 1.0 3.0  0.2590 3.0 1.0 2.0 3.0 1.0
-5.5152 2.0 3.0 2.0 2.0 2.0 3.0 -0.9250 3.0 3.0 2.0 3.0 1.0
-5.5657 2.0 2.0 1.0 3.0 2.0 3.0 -0.4831 1.0 5.0 2.0 3.0 1.0
14.8177 2.0 2.0 1.0 3.0 1.0 3.0  0.5046 3.0 3.0 1.0 1.0 2.0
16.9783 2.0 1.0 1.0 2.0 2.0 1.0 -0.6903 2.0 1.0 1.0 1.0 2.0
13.8966 1.0 3.0 2.0 2.0 2.0 1.0  1.6166 2.0 5.0 1.0 1.0 2.0
14.8166 2.0 2.0 2.0 2.0 1.0 3.0  0.2778 2.0 3.0 1.0 1.0 2.0
19.3640 2.0 3.0 2.0 2.0 1.0 2.0  1.9586 4.0 2.0 1.0 1.0 2.0
  9.5299 1.0 3.0 1.0 1.0 1.0 3.0  1.0506 2.0 5.0 2.0 1.0 2.0
12.0102 2.0 1.0 1.0 3.0 2.0 3.0  0.4871 1.0 1.0 2.0 1.0 2.0
  6.1551 2.0 1.0 2.0 3.0 2.0 1.0  2.0891 4.0 4.0 2.0 1.0 2.0
-1.7048 1.0 2.0 1.0 1.0 2.0 2.0  1.4338 4.0 3.0 2.0 1.0 2.0
  2.7640 1.0 1.0 2.0 3.0 1.0 2.0 -1.1196 3.0 4.0 2.0 1.0 2.0
  2.8065 1.0 3.0 1.0 1.0 2.0 1.0  0.3367 3.0 2.0 1.0 2.0 2.0
  0.0974 2.0 2.0 1.0 3.0 1.0 1.0  0.1092 2.0 2.0 1.0 2.0 2.0
-7.8080 1.0 1.0 1.0 2.0 2.0 2.0  0.4007 4.0 1.0 1.0 2.0 2.0
-18.0450 2.0 3.0 1.0 1.0 1.0 2.0  0.1460 3.0 5.0 1.0 2.0 2.0
-2.8199 2.0 1.0 2.0 3.0 1.0 3.0 -0.3877 3.0 4.0 1.0 2.0 2.0
  8.9893 1.0 1.0 1.0 2.0 2.0 1.0  0.6957 4.0 3.0 2.0 2.0 2.0
  3.7978 2.0 1.0 1.0 1.0 2.0 1.0 -0.4664 3.0 3.0 2.0 2.0 2.0
-6.3493 1.0 1.0 1.0 1.0 2.0 3.0  0.2067 2.0 4.0 2.0 2.0 2.0
  8.1411 2.0 1.0 2.0 1.0 1.0 2.0  0.4112 1.0 4.0 2.0 2.0 2.0
-7.5483 2.0 2.0 1.0 1.0 1.0 2.0 -1.3734 3.0 3.0 2.0 2.0 2.0
-0.4600 2.0 1.0 2.0 3.0 1.0 3.0  0.7065 1.0 3.0 1.0 3.0 2.0
-3.2135 1.0 2.0 2.0 2.0 1.0 2.0  1.3628 4.0 2.0 1.0 3.0 2.0
-6.6562 2.0 1.0 2.0 2.0 2.0 3.0 -0.5052 4.0 5.0 1.0 3.0 2.0
  5.1267 2.0 1.0 1.0 1.0 2.0 1.0 -1.3457 2.0 5.0 1.0 3.0 2.0
  3.5592 1.0 1.0 2.0 1.0 2.0 3.0 -1.8022 3.0 4.0 1.0 3.0 2.0
-4.4420 2.0 3.0 1.0 2.0 1.0 1.0  0.0116 2.0 4.0 2.0 3.0 2.0
-8.5965 2.0 2.0 1.0 3.0 2.0 3.0 -0.9075 1.0 3.0 2.0 3.0 2.0
-6.3187 2.0 2.0 2.0 2.0 2.0 3.0 -1.4707 1.0 1.0 2.0 3.0 2.0
-7.8953 2.0 2.0 1.0 1.0 2.0 1.0 -1.2938 2.0 3.0 2.0 3.0 2.0
-10.1383 1.0 3.0 1.0 3.0 2.0 2.0 -1.1660 4.0 4.0 2.0 3.0 2.0
-7.8850 1.0 2.0 1.0 1.0 2.0 3.0  0.0397 4.0 4.0 1.0 1.0 3.0
23.2001 1.0 3.0 1.0 2.0 1.0 3.0 -0.5987 3.0 2.0 1.0 1.0 3.0
  5.5829 2.0 3.0 2.0 2.0 1.0 1.0  0.6683 3.0 3.0 1.0 1.0 3.0
-4.3698 2.0 2.0 1.0 1.0 2.0 2.0 -0.0106 1.0 3.0 1.0 1.0 3.0
  2.1274 1.0 2.0 1.0 3.0 2.0 2.0  0.5885 1.0 3.0 1.0 1.0 3.0
-2.7184 1.0 1.0 1.0 1.0 1.0 2.0  0.4555 1.0 5.0 2.0 1.0 3.0
-17.9128 2.0 2.0 2.0 1.0 1.0 2.0  0.6502 4.0 3.0 2.0 1.0 3.0
-1.2708 1.0 1.0 1.0 3.0 1.0 1.0 -0.1601 1.0 3.0 2.0 1.0 3.0
-24.2735 2.0 2.0 1.0 3.0 2.0 3.0  1.6910 1.0 1.0 2.0 1.0 3.0

```

```

-14.7374 2.0 2.0 2.0 3.0 1.0 2.0 0.1053 4.0 4.0 2.0 1.0 3.0
 0.1713 2.0 1.0 2.0 3.0 2.0 2.0 -0.4037 3.0 4.0 1.0 2.0 3.0
 8.0006 1.0 3.0 2.0 3.0 1.0 3.0 -0.5853 3.0 2.0 1.0 2.0 3.0
 1.2100 2.0 3.0 2.0 1.0 1.0 1.0 -0.3037 1.0 3.0 1.0 2.0 3.0
 3.3307 1.0 3.0 1.0 1.0 2.0 2.0 -0.0774 1.0 4.0 1.0 2.0 3.0
-22.6713 2.0 3.0 1.0 2.0 2.0 1.0 0.4733 4.0 5.0 1.0 2.0 3.0
 7.5562 1.0 3.0 2.0 2.0 1.0 2.0 -0.0354 4.0 2.0 2.0 2.0 3.0
 -7.0694 1.0 3.0 2.0 2.0 1.0 1.0 -0.6640 2.0 1.0 2.0 2.0 3.0
 3.7159 2.0 3.0 1.0 3.0 1.0 1.0 0.0335 4.0 4.0 2.0 2.0 3.0
 -4.3135 1.0 2.0 2.0 2.0 1.0 3.0 0.1351 1.0 1.0 2.0 2.0 3.0
-14.5577 1.0 1.0 2.0 1.0 2.0 3.0 -0.5951 3.0 4.0 2.0 2.0 3.0
-12.5107 2.0 2.0 2.0 3.0 1.0 3.0 0.2735 3.0 2.0 1.0 3.0 3.0
 4.7708 2.0 2.0 1.0 1.0 1.0 3.0 0.3157 1.0 2.0 1.0 3.0 3.0
13.2797 2.0 2.0 2.0 1.0 1.0 1.0 -1.0843 2.0 3.0 1.0 3.0 3.0
 -6.3243 1.0 2.0 2.0 1.0 2.0 2.0 -0.0836 4.0 2.0 1.0 3.0 3.0
 -7.0549 2.0 1.0 2.0 1.0 1.0 2.0 -0.2884 2.0 1.0 1.0 3.0 3.0
 -9.2713 2.0 3.0 2.0 3.0 2.0 3.0 -0.1006 1.0 2.0 2.0 3.0 3.0
-18.7788 1.0 3.0 1.0 2.0 2.0 3.0 0.5710 1.0 3.0 2.0 3.0 3.0
 -7.7230 1.0 1.0 2.0 1.0 1.0 2.0 0.2776 2.0 3.0 2.0 3.0 3.0
-22.7230 2.0 3.0 2.0 2.0 1.0 3.0 -0.7561 4.0 4.0 2.0 3.0 3.0
-11.6609 1.0 2.0 2.0 2.0 1.0 2.0 1.5549 1.0 4.0 2.0 3.0 3.0 :: Y, X
 1 2 3 4 5 6 7 :: VPR
-1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 :: GAMMA(1:NVPR)

```

10.3 Program Results

nag_ml_hier_mixed_regsn (g02jec) Example Program Results

```

Number of observations (N)           = 90
Number of random factors (NRF)       = 55
Number of fixed factors (NFF)        = 4
Number of subject levels (NLSV)      = 3
Rank of X (RNKX)                     = 4
Effective N (EFFN)                   = 90
Number of nonzero variance components (NCOV) = 7
Parameter Estimates

```

Random Effects

```

Subject: Variable 10 (Level 1) Variable 11 (Level 1) Variable 12 (Level 1)
Variable 3 (Level 1)          2.1566      3.7320
Variable 3 (Level 2)          1.7769      3.8543
Variable 4 (Level 1)          0.5583      3.0508

Subject: Variable 10 (Level 1) Variable 11 (Level 1) Variable 12 (Level 1)
Variable 4 (Level 3)          0.6776      3.0358

Subject: Variable 10 (Level 2) Variable 11 (Level 1) Variable 12 (Level 1)
Variable 3 (Level 1)          1.4448      3.3293
Variable 3 (Level 2)         -2.8634      3.3533
Variable 4 (Level 1)          3.6811      2.2253
Variable 4 (Level 2)         -1.9988      2.2929
Variable 4 (Level 3)         -2.1281      1.9896

Subject: Variable 10 (Level 1) Variable 11 (Level 2) Variable 12 (Level 1)
Variable 3 (Level 1)         -3.1562      3.8624
Variable 3 (Level 2)          2.8856      4.6985
Variable 4 (Level 1)         -4.6811      2.2236
Variable 4 (Level 2)          5.5794      2.1390
Variable 4 (Level 3)         -0.9832      2.2841

Subject: Variable 10 (Level 2) Variable 11 (Level 2) Variable 12 (Level 1)
Variable 3 (Level 1)          4.3449      3.6258
Variable 3 (Level 2)         -4.4285      3.4096
Variable 4 (Level 1)         -1.0798      3.1008
Variable 4 (Level 2)          1.0536      2.9612

Subject: Variable 10 (Level 1) Variable 11 (Level 3) Variable 12 (Level 1)
Variable 3 (Level 1)          0.4216      4.0146
Variable 3 (Level 2)          0.2268      3.4265
Variable 4 (Level 1)         -1.0626      2.3505

Subject: Variable 10 (Level 1) Variable 11 (Level 3) Variable 12 (Level 1)
Variable 4 (Level 3)          1.2664      2.5276

Subject: Variable 10 (Level 2) Variable 11 (Level 3) Variable 12 (Level 1)

```

Variable	3 (Level 1)	1.2785	3.4331
Variable	3 (Level 2)	-1.6652	3.8605
Subject:	Variable 10 (Level 2)	Variable 11 (Level 3)	Variable 12 (Level 1)
Variable	4 (Level 2)	0.7332	2.6958
Variable	4 (Level 3)	-0.8547	2.7819
Subject:	Variable 11 (Level 1)	Variable 12 (Level 1)	
Variable	5 (Level 1)	-0.5540	2.8120
Variable	5 (Level 2)	1.9179	2.7500
Variable	6 (Level 1)	0.6925	3.6813
Variable	6 (Level 2)	-2.2632	3.1202
Variable	6 (Level 3)	4.3216	3.1131
Subject:	Variable 11 (Level 2)	Variable 12 (Level 1)	
Variable	5 (Level 1)	1.5151	2.9154
Variable	5 (Level 2)	-1.7072	2.8715
Variable	6 (Level 1)	0.2154	3.9398
Variable	6 (Level 2)	-3.7591	4.2153
Variable	6 (Level 3)	3.1563	4.7621
Subject:	Variable 11 (Level 3)	Variable 12 (Level 1)	
Variable	5 (Level 1)	1.7892	3.1214
Variable	5 (Level 2)	-1.6473	3.1579
Variable	6 (Level 1)	-1.2268	3.8853
Variable	6 (Level 2)	4.6247	3.6412
Variable	6 (Level 3)	-3.1117	3.1648
Subject:	Variable 12 (Level 1)		
Variable	7	0.6016	0.4634
Variable	8 (Level 1)	1.5887	1.2518
Variable	8 (Level 2)	-0.7951	1.4856
Variable	8 (Level 3)	0.3798	1.6037
Variable	8 (Level 4)	-0.8295	1.6629
Variable	9 (Level 1)	0.5197	1.5510
Variable	9 (Level 2)	0.0156	1.8248
Variable	9 (Level 3)	-0.1723	1.8271
Variable	9 (Level 4)	0.4305	1.9494
Variable	9 (Level 5)	-0.1412	2.0379
Subject:	Variable 10 (Level 1)	Variable 11 (Level 1)	Variable 12 (Level 2)
Variable	3 (Level 1)	6.3424	3.3173
Variable	3 (Level 2)	5.7538	3.3626
Subject:	Variable 10 (Level 1)	Variable 11 (Level 1)	Variable 12 (Level 2)
Variable	4 (Level 2)	2.5053	2.6520
Variable	4 (Level 3)	1.2953	2.6978
Subject:	Variable 10 (Level 2)	Variable 11 (Level 1)	Variable 12 (Level 2)
Variable	3 (Level 1)	1.6342	3.7874
Variable	3 (Level 2)	-2.8693	3.8549
Variable	4 (Level 1)	-0.9274	2.7266
Subject:	Variable 10 (Level 2)	Variable 11 (Level 1)	Variable 12 (Level 2)
Variable	4 (Level 3)	0.5394	2.7100
Subject:	Variable 10 (Level 1)	Variable 11 (Level 2)	Variable 12 (Level 2)
Variable	3 (Level 1)	-10.2379	3.2977
Variable	3 (Level 2)	3.2457	4.0593
Variable	4 (Level 1)	-2.8362	2.2599
Variable	4 (Level 2)	0.2805	2.9513
Variable	4 (Level 3)	0.3587	2.8663
Subject:	Variable 10 (Level 2)	Variable 11 (Level 2)	Variable 12 (Level 2)
Variable	3 (Level 1)	-1.3161	3.1545
Variable	3 (Level 2)	8.2719	3.9322
Variable	4 (Level 1)	-0.4813	2.3705
Variable	4 (Level 2)	2.6668	2.4832
Subject:	Variable 10 (Level 1)	Variable 11 (Level 3)	Variable 12 (Level 2)
Variable	3 (Level 1)	4.9485	3.9465
Variable	3 (Level 2)	0.0987	3.5531
Variable	4 (Level 1)	3.0791	2.1790
Variable	4 (Level 2)	-1.9469	2.3796
Variable	4 (Level 3)	0.4536	2.1984
Subject:	Variable 10 (Level 2)	Variable 11 (Level 3)	Variable 12 (Level 2)
Variable	3 (Level 1)	-4.5419	3.2940

Variable	3 (Level 2)	-3.9095	4.0163
Variable	4 (Level 1)	-0.4456	2.6194
Variable	4 (Level 2)	-1.5462	2.6514
Variable	4 (Level 3)	-0.6636	2.8738

Subject: Variable 11 (Level 1) Variable 12 (Level 2)

Variable	5 (Level 1)	4.9921	3.0570
Variable	5 (Level 2)	0.8986	3.0576
Variable	6 (Level 1)	7.0091	3.7851
Variable	6 (Level 2)	-1.3173	3.1348
Variable	6 (Level 3)	6.1881	3.4928

Subject: Variable 11 (Level 2) Variable 12 (Level 2)

Variable	5 (Level 1)	-0.3947	3.0751
Variable	5 (Level 2)	0.3750	3.0579
Variable	6 (Level 1)	6.9902	3.2654
Variable	6 (Level 2)	-1.0683	3.5699
Variable	6 (Level 3)	-5.9617	3.6688

Subject: Variable 11 (Level 3) Variable 12 (Level 2)

Variable	5 (Level 1)	-1.0471	3.0732
Variable	5 (Level 2)	-0.7991	2.9597
Variable	6 (Level 1)	2.7549	3.8142
Variable	6 (Level 2)	-6.3441	3.2624
Variable	6 (Level 3)	-0.1341	3.5956

Subject: Variable 12 (Level 2)

Variable	7	0.1533	0.5196
Variable	8 (Level 1)	1.6630	1.8224
Variable	8 (Level 2)	-0.6835	1.6502
Variable	8 (Level 3)	-0.0959	1.5604
Variable	8 (Level 4)	0.1696	1.4537
Variable	9 (Level 1)	1.0203	2.2901
Variable	9 (Level 2)	6.4354	1.7420
Variable	9 (Level 3)	-1.5942	1.7761
Variable	9 (Level 4)	0.0955	1.9436
Variable	9 (Level 5)	-3.9588	1.7124

Subject: Variable 10 (Level 1) Variable 11 (Level 1) Variable 12 (Level 3)

Variable	3 (Level 1)	10.9751	3.2085
Variable	3 (Level 2)	-1.0674	3.7219
Variable	4 (Level 1)	-2.8350	2.2037
Variable	4 (Level 2)	3.7075	2.7912
Variable	4 (Level 3)	2.2405	2.2796

Subject: Variable 10 (Level 2) Variable 11 (Level 1) Variable 12 (Level 3)

Variable	3 (Level 1)	-6.2719	3.3190
Variable	3 (Level 2)	-9.2923	3.7884
Variable	4 (Level 1)	-2.8586	2.3728

Subject: Variable 10 (Level 2) Variable 11 (Level 1) Variable 12 (Level 3)

Variable	4 (Level 3)	-2.0316	2.2895
----------	-------------	---------	--------

Subject: Variable 10 (Level 1) Variable 11 (Level 2) Variable 12 (Level 3)

Variable	3 (Level 1)	-3.3222	3.4246
Variable	3 (Level 2)	-0.3111	3.2221
Variable	4 (Level 1)	1.6131	2.3970
Variable	4 (Level 2)	-3.0099	2.9300
Variable	4 (Level 3)	0.2552	2.7229

Subject: Variable 10 (Level 2) Variable 11 (Level 2) Variable 12 (Level 3)

Variable	3 (Level 1)	6.6372	3.9751
Variable	3 (Level 2)	-5.4249	3.4039
Variable	4 (Level 1)	-3.2357	2.8565
Variable	4 (Level 2)	1.5313	2.8232
Variable	4 (Level 3)	2.0854	3.0661

Subject: Variable 10 (Level 1) Variable 11 (Level 3) Variable 12 (Level 3)

Variable	3 (Level 1)	8.5902	4.0894
Variable	3 (Level 2)	-1.6058	3.2906
Variable	4 (Level 1)	3.2575	2.5450

Subject: Variable 10 (Level 1) Variable 11 (Level 3) Variable 12 (Level 3)

Variable	4 (Level 3)	-1.0630	2.8692
----------	-------------	---------	--------

Subject: Variable 10 (Level 2) Variable 11 (Level 3) Variable 12 (Level 3)

Variable	3 (Level 1)	-4.5747	3.9475
Variable	3 (Level 2)	-4.1752	3.0911

Variable	4 (Level 1)	1.0578	2.5496
Variable	4 (Level 2)	-4.4284	2.2029
Variable	4 (Level 3)	0.6214	2.5884
Subject: Variable 11 (Level 1) Variable 12 (Level 3)			
Variable	5 (Level 1)	5.4387	3.0091
Variable	5 (Level 2)	-8.5065	3.1099
Variable	6 (Level 1)	-0.9179	3.7257
Variable	6 (Level 2)	-2.4920	3.1176
Variable	6 (Level 3)	-2.7772	3.4083
Subject: Variable 11 (Level 2) Variable 12 (Level 3)			
Variable	5 (Level 1)	4.4193	3.1282
Variable	5 (Level 2)	-5.7324	3.1435
Variable	6 (Level 1)	-5.9992	3.1431
Variable	6 (Level 2)	5.5657	3.2599
Variable	6 (Level 3)	-2.2147	3.1758
Subject: Variable 11 (Level 3) Variable 12 (Level 3)			
Variable	5 (Level 1)	0.3594	2.9017
Variable	5 (Level 2)	-1.3169	3.0004
Variable	6 (Level 1)	14.5815	3.8519
Variable	6 (Level 2)	-5.2262	3.2578
Variable	6 (Level 3)	-11.2864	3.1821
Subject: Variable 12 (Level 3)			
Variable	7	-0.2970	0.5930
Variable	8 (Level 1)	2.6255	1.5201
Variable	8 (Level 2)	0.5048	1.7865
Variable	8 (Level 3)	-0.1518	1.8905
Variable	8 (Level 4)	-4.3754	1.4651
Variable	9 (Level 1)	-4.4219	2.0532
Variable	9 (Level 2)	3.7058	1.9085
Variable	9 (Level 3)	-1.7524	1.7894
Variable	9 (Level 4)	0.4339	1.8210
Variable	9 (Level 5)	-0.6161	2.3700
Fixed Effects			
Intercept		1.5913	2.4106
Variable	1 (Level 2)	-1.5994	0.8183
Variable	2 (Level 2)	-2.3793	1.0996
Variable	2 (Level 3)	0.5328	1.1677
Variance Components			
Estimate	Parameter	Subject	
36.38867	Variable 3	Variables 10 11 12	
11.43322	Variable 4	Variables 10 11 12	
19.73586	Variable 5	Variables 11 12	
39.80174	Variable 6	Variables 11 12	
0.41583	Variable 7	Variables 12	
5.16442	Variable 8	Variables 12	
9.79904	Variable 9	Variables 12	
SIGMA**2 = 0.00042			
-2LOG LIKELIHOOD = 617.11969			
