NAG Library Function Document

nag_reml_hier_mixed_regsn (g02jdc)

1 Purpose

nag_reml_hier_mixed_regsn (g02jdc) fits a multi-level linear mixed effects regression model using restricted maximum likelihood (REML). Prior to calling nag_reml_hier_mixed_regsn (g02jdc) the initialization function nag_hier_mixed_init (g02jcc) must be called.

2 Specification

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

void nag_reml_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_reml_hier_mixed_regsn (g02jdc) 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,

\( \beta \) is a vector of length \( p \) of unknown fixed effects,

\( Z \) is a known \( n \) by \( q \) design matrix for the random independent variables,

\( \nu \) is a vector of length \( q \) of unknown random effects,

and \( \epsilon \) is a vector of length \( n \) of unknown random errors.

Both \( \nu \) and \( \epsilon \) are assumed to have a Gaussian distribution with expectation zero and variance/covariance matrix defined by

\[
\begin{bmatrix}
\nu \\
\epsilon
\end{bmatrix}
\approx
\begin{bmatrix}
G & 0 \\
0 & R
\end{bmatrix}
\]

where \( R = \sigma^2_R I, I \) 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 \( \sigma^2_i \). The diagonal elements of matrix \( G \) therefore take one of the values \( \{\sigma^2_i : i = 1, 2, \ldots, g\} \), depending on which group the associated random variable belongs to.

The model therefore contains three sets of unknowns: the fixed effects \( \beta \), the random effects \( \nu \) and a vector of \( g + 1 \) variance components \( \gamma \), where

\[ \gamma = \{\sigma^2_1, \sigma^2_2, \ldots, \sigma^2_{g-1},\sigma^2_g,\sigma^2_R\} \]

Rather than working directly with \( \gamma \), nag_reml_hier_mixed_regsn (g02jdc) uses an iterative process to estimate

\[ \gamma^* = \{\sigma^2_1/\sigma^2_R, \sigma^2_2/\sigma^2_R, \ldots, \sigma^2_{g-1}/\sigma^2_R, \sigma^2_g/\sigma^2_R, 1\} \]

Due to the iterative nature of the estimation a set of initial values, \( \gamma_0 \), for \( \gamma^* \) is required. nag_reml_hier_mixed_regsn (g02jdc) 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_reml_hier_mixed_regsn (g02jdc) fits the model by maximizing the restricted log-likelihood function:

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

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 $\gamma^*$ have been obtained, the value of $\sigma^2_R$ is given by

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

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

5 Arguments

Note: Prior to calling nag_reml_hier_mixed_regsn (g02jdc) 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: $lvpr = \sum_i \text{RNDM}(1, i) + \text{RNDM}(2, i)$.

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, $\sigma^2$. See Section 9 for more details.

   Constraint: $1 \leq vpr[i - 1] \leq nvpr$, for $i = 1, 2, \ldots, lvpr$.

3: nvpr – Integer **Input**

   On entry: $g$, the number of variance components being estimated (excluding the overall variance, $\sigma^2_R$).

   Constraint: $1 \leq nvpr \leq lvpr$. 

### Page 4

**gamma[nvpr + 1]** – double  
*Input/Output*

*On entry:* holds the initial values of the variance components, $\gamma_0$, with $\text{gamma}[i-1]$ the initial value for $\sigma^2_i/\sigma_R^2$, for $i = 1, 2, \ldots, \text{nvpr}$.  

If $\text{gamma}[0] = -1.0$, the remaining elements of $\text{gamma}$ are ignored and the initial values for the variance components are estimated from the data using MIVQUE0.  

*On exit:* $\text{gamma}[i-1]$, for $i = 1, 2, \ldots, \text{nvpr}$, holds the final estimate of $\sigma^2_i$ and $\text{gamma[nvpr]}$ holds the final estimate for $\sigma_R^2$.  

**Constraint:** $\text{gamma}[0] = -1.0$ or $\text{gamma}[i-1] \geq 0.0$, for $i = 1, 2, \ldots, g$.  

### Page 5

**effn** – Integer  
*Output*

*On exit:* effective number of observations. If there are no weights (i.e., $\text{wt}$ is NULL), or all weights are nonzero, then $\text{effn} = n$.  

### Page 6

**rnkx** – Integer  
*Output*

*On exit:* the rank of the design matrix, $X$, for the fixed effects.  

### Page 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 $\text{ncov} = \text{nvpr}$.  

### Page 8

**lnlike** – double  
*Output*

*On exit:* $-2l_R(\hat{\gamma})$ where $l_R$ is the log of the restricted maximum likelihood calculated at $\hat{\gamma}$, the estimated variance components returned in $\text{gamma}$.  

### Page 9

**lb** – Integer  
*Input*

*On entry:* the dimension of the arrays $\text{b}$ and $\text{se}$.  

**Constraint:** $\text{lb} \geq \text{nff} + \text{nrf} \times \text{nlsv}$.  

### Page 10

**id[pdid × lb]** – Integer  
*Output*

**Note:** where $\text{ID}(i, j)$ appears in this document, it refers to the array element $\text{id}[(j-1) \times \text{pdid} + i - 1]$.  

*On exit:* an array describing the parameter estimates returned in $\text{b}$. The first $\text{nlsv} \times \text{nrf}$ columns of $\text{ID}$ describe the parameter estimates for the random effects and the last $\text{nff}$ columns the parameter estimates for the fixed effects.  

A print function for decoding the parameter estimates given in $\text{b}$ using information from $\text{id}$ is supplied with the example program for this function.  

For fixed effects:  
for $l = \text{nrf} \times \text{nlsv} + 1, \ldots, \text{nrf} \times \text{nlsv} + \text{nff}$  
if $\text{b}[l-1]$ contains the parameter estimate for the intercept then  
$$\text{ID}(1, l) = \text{ID}(2, l) = \text{ID}(3, l) = 0;$$  
if $\text{b}[l-1]$ 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 $\text{DAT}$ when $\text{fixed}[j+1] = k$ then  
$$\text{ID}(1, l) = 0, \quad \text{ID}(2, l) = j, \quad \text{ID}(3, l) = i;$$  
if the $j$th variable is continuous or binary, that is $\text{levels}[\text{fixed}[j+1] - 1] = 1$, then  
$$\text{ID}(3, l) = 0;$$
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

\[ N_{R_b} = \text{RNDM}(1, b); \]

\( R_{j_b} \) 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 \( \text{RNDM}(2 + j, b) = k; \)

\[ N_{S_b} \] denote the number of subject variables in the \( b \)th random statement, that is

\[ N_{S_b} = \text{RNDM}(3 + N_{R_b}, b); \]

\( S_{j_b} \) 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

\[
\text{RNDM}(3 + N_{R_b} + j, b) = k;
\]

\( L(S_{j_b}) \) denote the number of levels for \( S_{j_b} \), that is

\[ L(S_{j_b}) = \text{levels}[\text{RNDM}(3 + N_{R_b} + j, b) - 1]; \]

then

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

\[
\begin{align*}
\text{ID}(1, l) &= b, \\
\text{ID}(2, l) &= j, \\
\text{ID}(3, l) &= i, \\
\text{ID}(3 + k, l) &= s_k, k = 1, 2, \ldots, N_{S_b};
\end{align*}
\]

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

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

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: \textbf{pdid} \textit{– Integer}

\textit{Input}

\textit{On entry:} the stride separating matrix row elements in the array \textbf{id}.

\textit{Constraint:} \textbf{pdid} \( \geq 3 + \max(\text{RNDM}(3 + \text{RNDM}(1, j), j)) \), i.e., \( 3 + \) maximum number of subject variables (see nag_hier_mixed_init (g02jcc)).

12: \textbf{b[\text{lb}]} \textit{– double}

\textit{Output}

\textit{On exit:} the parameter estimates, with the first \( nrf \times nlsv \) elements of \textbf{b} containing the parameter estimates for the random effects, \( \nu \), and the remaining \( nff \) elements containing the parameter estimates for the fixed effects, \( \beta \). The order of these estimates are described by the \textbf{id} argument.

13: \textbf{se[\text{lb}]} \textit{– double}

\textit{Output}

\textit{On exit:} the standard errors of the parameter estimates given in \textbf{b}.

14: \textbf{czz[\text{dim}]} \textit{– double}

\textit{Output}

\textit{Note:} the dimension, \textit{dim}, of the array \textbf{czz} must be at least \textbf{pdczz} \( \times nrf \times nlsv \).

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

\textit{On exit:} if \( nlsv = 1 \), then \( CZZ \) holds the lower triangular portion of the matrix

\[
\left(1/\sigma^2\right)\left(Z^T \hat{R}^{-1} Z + \hat{G}^{-1}\right),
\]

where \( \hat{R} \) and \( \hat{G} \) are the estimates of \( R \) and \( G \) respectively. If
On entry: the stride separating matrix row elements in the array \( \text{czz} \).

Constraint: \( \text{pdczz} \geq \text{nrf} \).

16: \( \text{cxx}[\text{dim}] \) – double

Output

Note: the dimension, \( \text{dim} \), of the array \( \text{cxx} \) must be at least \( \text{pdcxx} \times \text{nff} \).

Where \( \text{CXX}(i,j) \) appears in this document, it refers to the array element \( \text{cxx}[(j-1) \times \text{pdcxx} + i-1] \).

On exit: \( \text{CXX} \) 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: \( \text{pdcxx} \) – Integer

Input

On entry: the stride separating matrix row elements in the array \( \text{cxx} \).

Constraint: \( \text{pdcxx} \geq \text{nff} \).

18: \( \text{cxz}[\text{dim}] \) – double

Output

Note: the dimension, \( \text{dim} \), of the array \( \text{cxz} \) must be at least \( \text{pdcxz} \times \text{nlsv} \times \text{nrf} \).

Where \( \text{CXZ}(i,j) \) appears in this document, it refers to the array element \( \text{cxz}[(j-1) \times \text{pdcxz} + i-1] \).

On exit: if \( \text{nlsv} = 1 \), then \( \text{CXZ} \) 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 \( \text{nlsv} > 1 \) then \( \text{CXZ} \) holds this matrix in compressed form, with the first \( \text{nrf} \) columns holding the part of the matrix corresponding to the first level of the overall subject variable, the next \( \text{nrf} \) columns the part corresponding to the second level of the overall subject variable etc.

19: \( \text{pdcxz} \) – Integer

Input

On entry: the stride separating matrix row elements in the array \( \text{cxz} \).

Constraint: \( \text{pdcxz} \geq \text{nff} \).

20: \( \text{rcomm}[\text{dim}] \) – const double

Communication Array

Note: the dimension, \( \text{dim} \), of the array \( \text{rcomm} \) must be at least \( \text{lrcomm} \).

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

21: \( \text{icomm}[\text{dim}] \) – const Integer

Communication Array

Note: the dimension, \( \text{dim} \), of the array \( \text{icomm} \) must be at least \( \text{licomm} \).

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

22: \( \text{iopt}[\text{liopt}] \) – const Integer

Input

On entry: optional arguments passed to the optimization function.

By default nag_reml_hier_mixed_regsn (g02jdc) 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 \( \text{iop}t[4] = 1 \). If \( \text{iop}t < 5 \) or \( \text{iop}t[4] \neq 1 \) then E04LBF will be used.

Different optional arguments 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 \( \text{iop}t \) values of \( \text{iop}t \) are used, so for example, if only the first element of \( \text{iop}t \) needs changing and default values for all other optional arguments are sufficient \( \text{iop}t \) can be set to 1.

NAG Fortran Library routine E04LBF is being used

<table>
<thead>
<tr>
<th>( i )</th>
<th>Description</th>
<th>Equivalent E04LBF argument</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Number of iterations</td>
<td>MAXCAL</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>Unit number for monitoring information. See nag_open_file (x04acc) for details on how to assign a file to a unit number.</td>
<td>n/a</td>
<td>Output sent to stdout</td>
</tr>
<tr>
<td>2</td>
<td>Print optional arguments (1 = print)</td>
<td>n/a</td>
<td>(-1) (no printing performed)</td>
</tr>
<tr>
<td>3</td>
<td>Frequency that monitoring information is printed</td>
<td>iPRINT</td>
<td>(-1)</td>
</tr>
<tr>
<td>4</td>
<td>Optimizer used</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

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

NAG Fortran Library routine E04UCF is being used

<table>
<thead>
<tr>
<th>( i )</th>
<th>Description</th>
<th>Equivalent E04UCF argument</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Number of iterations</td>
<td>Major Iteration Limit</td>
<td>( \max (50, 3 \times \text{nvpr}) )</td>
</tr>
<tr>
<td>1</td>
<td>Unit number for monitoring information. See nag_open_file (x04acc) for details on how to assign a file to a unit number.</td>
<td>n/a</td>
<td>Output sent to stdout</td>
</tr>
<tr>
<td>2</td>
<td>Print optional arguments (1 = print, otherwise no print)</td>
<td>List/ NoList</td>
<td>(-1) (no printing performed)</td>
</tr>
<tr>
<td>3</td>
<td>Frequency that monitoring information is printed</td>
<td>Major Print Level</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Optimizer used</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>5</td>
<td>Number of minor iterations</td>
<td>Minor Iteration Limit</td>
<td>( \max (50, 3 \times \text{nvpr}) )</td>
</tr>
<tr>
<td>6</td>
<td>Frequency that additional monitoring information is printed</td>
<td>Minor Print Level</td>
<td>0</td>
</tr>
</tbody>
</table>

If \( \text{iop}t \leq 0 \) then default values are used for all optional arguments and \( \text{iop}t \) may be set to NULL.

23: \( \text{iop}t \) – Integer

On entry: length of the options array \( \text{iop}t \).

24: \( \text{ropt}[\text{ropt}] \) – const double

On entry: optional arguments passed to the optimization function.

Different optional arguments 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 \( \text{ropt} \) values of \( \text{ropt} \) are used, so for example, if only the first element of \( \text{ropt} \) needs changing and default values for all other optional arguments are sufficient \( \text{ropt} \) can be set to 1.

NAG Fortran Library routine E04LBF is being used

<table>
<thead>
<tr>
<th>( i )</th>
<th>Description</th>
<th>Equivalent E04LBF argument</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sweep tolerance</td>
<td>n/a</td>
<td>( \max \left( \sqrt{\varepsilon_p}, \sqrt{\varepsilon_p} \times \max(i_{zz}) \right) )</td>
</tr>
<tr>
<td>1</td>
<td>Lower bound for ( \gamma^+ )</td>
<td>n/a</td>
<td>( \varepsilon_p / 100 )</td>
</tr>
<tr>
<td>2</td>
<td>Upper bound for ( \gamma^+ )</td>
<td>n/a</td>
<td>( 10^{10} )</td>
</tr>
<tr>
<td>3</td>
<td>Accuracy of linear minimizations</td>
<td>ETA</td>
<td>0.9</td>
</tr>
</tbody>
</table>
4 Accuracy to which solution is required XTOL 0
5 Initial distance from solution STEPMX 100000

NAG Fortran Library routine E04UCF is being used

<table>
<thead>
<tr>
<th>i</th>
<th>Description</th>
<th>Equivalent E04UCF argument</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sweep tolerance</td>
<td>n/a</td>
<td>max (\sqrt{\text{eps}}, \sqrt{\text{eps}} \times \max(zz_{ii}))</td>
</tr>
<tr>
<td>1</td>
<td>Lower bound for (\gamma^*)</td>
<td>n/a</td>
<td>\text{eps}/100</td>
</tr>
<tr>
<td>2</td>
<td>Upper bound for (\gamma^*)</td>
<td>n/a</td>
<td>10^{10}</td>
</tr>
<tr>
<td>3</td>
<td>Line search tolerance</td>
<td>Line Search Tolerance</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>Optimality tolerance</td>
<td>Optimality Tolerance</td>
<td>\text{eps}^{0.72}</td>
</tr>
</tbody>
</table>

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 \(\leq 0\) then default values are used for all optional arguments 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 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument \(\langle\text{value}\rangle\) had an illegal value.

NE_INT
On entry, \(\text{lb} = \langle\text{value}\rangle\).
Constraint: \(\text{lb} \geq \langle\text{value}\rangle\).

On entry, \(\text{lvpr} = \langle\text{value}\rangle\).
Constraint: \(\text{lvpr} \geq \langle\text{value}\rangle\).

On entry, \(\text{nvpr} = \langle\text{value}\rangle\).
Constraint: \(1 \leq \text{nvpr} \leq \langle\text{value}\rangle\).

On entry, \(\text{pdcxx} = \langle\text{value}\rangle\).
Constraint: \(\text{pdcxx} \geq \langle\text{value}\rangle\).

On entry, \(\text{pdcxz} = \langle\text{value}\rangle\).
Constraint: \(\text{pdcxz} \geq \langle\text{value}\rangle\).

On entry, \(\text{pdczz} = \langle\text{value}\rangle\).
Constraint: \(\text{pdczz} \geq \langle\text{value}\rangle\).

On entry, \(\text{pdid} = \langle\text{value}\rangle\).
Constraint: \(\text{pdid} \geq \langle\text{value}\rangle\).
NE_INT_ARRAY
On entry, at least one value of i, for i = 1, 2, ..., nvpr, does not appear in vpr.
On entry, icomm has not been initialized correctly.
On entry, vpr[i−1] = (value) and nvpr = (value).
Constraint: 1 ≤ vpr[i−1] ≤ 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 3.6.6 in the Essential Introduction 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 3.6.5 in the Essential Introduction for further information.

NE_REAL_ARRAY
On entry, gamma[i] = (value).
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_reml_hier_mixed_regsn (g02jdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_reml_hier_mixed_regsn (g02jdc) 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 \texttt{vpr} gives the mapping between the random variables and the variance components. In most cases \( \texttt{vpr}[i - 1] = i \), for \( i = 1, 2, \ldots, \sum \texttt{RNDM}(1, i) + \texttt{RNDM}(2, i) \). 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 \texttt{DAT} hold dummy variables.

Consider a dataset with three variables:

\[
\text{DAT} = \begin{pmatrix}
1 & 1 & 3.6 \\
2 & 1 & 4.5 \\
3 & 1 & 1.1 \\
1 & 2 & 8.3 \\
2 & 2 & 7.2 \\
3 & 2 & 6.1
\end{pmatrix}
\]

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 \texttt{nag_hier_mixed_init (g02jcc)}

\[
\text{levels} = (3 \ 2 \ 1)
\]

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

\[
\text{fixed} = (0 \ 0); \\
\text{RNDM} = (3 \ 0 \ 1 \ 2 \ 3)^T.
\]

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

\[
\texttt{vpr} = (1 \ 2 \ 3).
\]

This is the recommended way of supplying the data to \texttt{nag_reml_hier_mixed_regsn (g02jdc)}, 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

\[
\text{DAT} = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 3.6 \\
0 & 1 & 0 & 1 & 0 & 4.5 \\
0 & 0 & 1 & 1 & 0 & 1.1 \\
1 & 0 & 0 & 0 & 1 & 8.3 \\
0 & 1 & 0 & 0 & 1 & 7.2 \\
0 & 0 & 1 & 0 & 1 & 6.1
\end{pmatrix}
\]

where each column only has one level

\[
\text{levels} = (1 \ 1 \ 1 \ 1 \ 1 \ 1).
\]

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

\[
\text{fixed} = (0 \ 0); \\
\text{RNDM} = (6 \ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6)^T.
\]

With the data entered in this manner, the first three columns of \texttt{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 \texttt{vpr} must reflect this

\[
\texttt{vpr} = (1 \ 1 \ 1 \ 2 \ 2 \ 3).
\]

In most situations it is more efficient to supply the data to \texttt{nag_hier_mixed_init (g02jcc)} 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_reml_hier_mixed_regsn (g02jdc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group. *
 * Mark 9, 2009. */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nagg02.h>
#include <nag_stdlib.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 pdid, double b[], double se[]);

#define RNDM(I, J) rndm[ (order == Nag_ColMajor) \
 ?((J-1)*lrndm+I-1):(I-1)*lrndm+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)*pdid+I-1)]

int main(void)
{
    /* IO file pointers */
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer pdid, licomm, lrcomm, tdczz, lb, pdcxx, pdcxz, pdczz, pddat,
    effn, i, j, lvpr, n, ncol, ncov, lfixed, nff, nl, nlsv, nrndm,
    nr, 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;
    
    /* Initialise the error structure */
    INIT_FAIL(fail);
    printf("nag_reml_hier_mixed_regsn (g02jdc) 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, l, &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 */
ifixed = 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
```c
#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] ", &fixed[0]);
#else
    scanf("%*[\n] ", &fixed[0]);
#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
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &fixed[1]);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &fixed[1]);
#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] ", &fixed[0]);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &fixed[0]);
#endif
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;
}
```
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
  }
  ifdef _WIN32
    scanf_s("%"NAG_IFMT"%[\n ]", &RNDM(nv+3, j));
  ifdef _WIN32
  scanf("%"NAG_IFMT"%[\n ]", &RNDM(nv+3, j));
  #endif
  #ifdef _WIN32
    scanf_s("%[\n ]");
  #else
    scanf("%[\n ]");
  #endif
  ifdef _WIN32
  scanf_s("%"NAG_IFMT"%[\n ]", &RNDM(nv+3, j));
  #else
  scanf("%"NAG_IFMT"%[\n ]", &RNDM(nv+3, j));
  #endif
 .isDefined(nv, j);
  if (nv+n1+2 > lrndm)
  {
    printf(" ** Problem size too large, increase array sizes\n");
    printf("LRNDM,NV+N+2 = %"NAG_IFMT", %"NAG_IFMT"\n", lrndm, nv+n1+2);
    exit_status = -1;
    goto END;
  }
/* Subject variable IDs */
if (n1 > 0)
{
  for (i = nv+4; i <= nv + n1 + 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, n1);
  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
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, snff, &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, snff, &nlsv, &nrf, rcomm, lrcomm,
  icomm, licomm, &fail);
if (fail.code != NE_NO_ERROR)
{
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)) ||
!(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 */
ag_reml_hier_mixed_regsn(lvpr, vpr, nvpr, gamma, &effn, &rnkx, &ncov, &lnlike, lb, id, pdid, b, se, czz, pdcxx, cxz, pdcxz, rcomm, icomm, iopt, liopt, ropt, lropt, &fail);
if (fail.code != NE_NOERROR)
{
printf("Error from nag_reml_hier_mixed_regsn (g02jdc).
\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, nco, 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(icomm);
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 nco, double lnlike,
Integer id[], Integer pdid, double b[], double se[])
/* 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 non-zero 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++)
        {
            if (ID(3+1, k) != ID(3+1, k-1))
            {
                same = 0;
                break;
            }
        }
    }
    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
", gamma[nvpr]);
printf("-2LOG LIKELIHOOD = %15.5f
", lnlike);

10.2 Program Data

nag_reml_hier_mixed_regsn (g02jdc) Example Program Data
U 90 12 3 7 :: WEIGHT (U = no weights),N,NCOL,NRAND,NVPR
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 1.0 -0.7630 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 1.0 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 3.0 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
-0.1806 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 1.0 1.0
1.2590 1.0 2.0 2.0 1.0 2.0 3.0 0.4742 1.0 1.0 2.0 2.0 1.0
8.8405 2.0 2.0 2.0 2.0 2.0 3.0 0.6888 3.0 1.0 2.0 2.0 1.0
6.1657 2.0 1.0 2.0 3.0 3.0 1.0 0.0 -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
1.5608 2.0 2.0 2.0 1.0 2.0 2.0 1.5080 3.0 1.0 1.0 3.0 1.0
-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 2.0 2.0 3.0 0.7770 4.0 1.0 2.0 3.0 1.0
10.3 Program Results

nag_reml_hier_mixed_regsn (g02jdc) 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 non-zero variance components (NCOV) = 7

Parameter Estimates

Random Effects

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 1)</th>
<th>Variable 11 (Level 1)</th>
<th>Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1)</td>
<td>2.1561</td>
<td>3.7946</td>
</tr>
<tr>
<td>Variable 3 (Level 2)</td>
<td>1.8951</td>
<td>3.9284</td>
</tr>
<tr>
<td>Variable 4 (Level 1)</td>
<td>0.6496</td>
<td>3.1617</td>
</tr>
<tr>
<td>Variable 4 (Level 3)</td>
<td>0.7390</td>
<td>3.1424</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 2)</th>
<th>Variable 11 (Level 1)</th>
<th>Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1)</td>
<td>1.4216</td>
<td>3.3773</td>
</tr>
<tr>
<td>Variable 3 (Level 2)</td>
<td>-2.8921</td>
<td>3.9593</td>
</tr>
<tr>
<td>Variable 4 (Level 1)</td>
<td>3.6789</td>
<td>2.3162</td>
</tr>
<tr>
<td>Variable 4 (Level 3)</td>
<td>-1.9742</td>
<td>2.3887</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 3)</th>
<th>Variable 11 (Level 2)</th>
<th>Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1)</td>
<td>-2.9659</td>
<td>3.9127</td>
</tr>
<tr>
<td>Variable 3 (Level 2)</td>
<td>2.7951</td>
<td>4.7183</td>
</tr>
<tr>
<td>Variable 4 (Level 1)</td>
<td>-4.7330</td>
<td>2.3094</td>
</tr>
<tr>
<td>Variable 4 (Level 3)</td>
<td>-0.8417</td>
<td>2.3826</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 11 (Level 2)</th>
<th>Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 5 (Level 1)</td>
<td>-0.4885</td>
</tr>
<tr>
<td>Variable 5 (Level 2)</td>
<td>1.8829</td>
</tr>
<tr>
<td>Variable 6 (Level 1)</td>
<td>0.9249</td>
</tr>
<tr>
<td>Variable 6 (Level 2)</td>
<td>-2.3568</td>
</tr>
<tr>
<td>Variable 6 (Level 3)</td>
<td>4.3117</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 11 (Level 3)</th>
<th>Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 5 (Level 1)</td>
<td>1.3898</td>
</tr>
<tr>
<td>Variable 5 (Level 2)</td>
<td>-1.5729</td>
</tr>
<tr>
<td>Variable 6 (Level 1)</td>
<td>0.2111</td>
</tr>
<tr>
<td>Variable 6 (Level 2)</td>
<td>-3.7083</td>
</tr>
<tr>
<td>Variable 6 (Level 3)</td>
<td>3.1190</td>
</tr>
<tr>
<td>Subject: Variable 11 (Level 3) Variable 12 (Level 1)</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Variable 5 (Level 1) 1.7352 3.1370</td>
<td></td>
</tr>
<tr>
<td>Variable 5 (Level 2) -1.6165 3.1713</td>
<td></td>
</tr>
<tr>
<td>Variable 6 (Level 1) -1.1102 3.9374</td>
<td></td>
</tr>
<tr>
<td>Variable 6 (Level 2) 4.4877 3.6980</td>
<td></td>
</tr>
<tr>
<td>Variable 6 (Level 3) -3.1325 3.1966</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 12 (Level 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 7 0.6827 0.5060</td>
</tr>
<tr>
<td>Variable 8 (Level 1) 1.5964 1.3206</td>
</tr>
<tr>
<td>Variable 8 (Level 2) -0.7533 1.5663</td>
</tr>
<tr>
<td>Variable 8 (Level 3) 0.4035 1.6840</td>
</tr>
<tr>
<td>Variable 8 (Level 4) -0.8523 1.7518</td>
</tr>
<tr>
<td>Variable 9 (Level 1) 0.5699 1.6236</td>
</tr>
<tr>
<td>Variable 9 (Level 2) 0.0012 1.9111</td>
</tr>
<tr>
<td>Variable 9 (Level 3) -0.2850 1.9245</td>
</tr>
<tr>
<td>Variable 9 (Level 4) 0.4468 2.0329</td>
</tr>
<tr>
<td>Variable 9 (Level 5) 0.0030 2.1390</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 1) Variable 11 (Level 1) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) 6.2551 3.3595</td>
</tr>
<tr>
<td>Variable 3 (Level 2) 5.6085 3.4127</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 1) Variable 11 (Level 1) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 4 (Level 2) 2.6922 2.7542</td>
</tr>
<tr>
<td>Variable 4 (Level 3) 1.3742 2.8068</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 2) Variable 11 (Level 1) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) 1.5647 3.8353</td>
</tr>
<tr>
<td>Variable 3 (Level 2) -2.7565 3.9041</td>
</tr>
<tr>
<td>Variable 4 (Level 1) -0.8621 2.8257</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 2) Variable 11 (Level 1) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 4 (Level 3) 0.4636 2.8070</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 1) Variable 11 (Level 2) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) -10.1544 3.3433</td>
</tr>
<tr>
<td>Variable 3 (Level 2) 3.2446 4.1221</td>
</tr>
<tr>
<td>Variable 4 (Level 1) -2.9419 2.3508</td>
</tr>
<tr>
<td>Variable 4 (Level 2) 0.2510 3.0675</td>
</tr>
<tr>
<td>Variable 4 (Level 3) 0.3224 2.9710</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 2) Variable 11 (Level 2) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) -1.3577 3.1925</td>
</tr>
<tr>
<td>Variable 3 (Level 2) 8.1277 3.9975</td>
</tr>
<tr>
<td>Variable 4 (Level 1) -0.4290 2.4578</td>
</tr>
<tr>
<td>Variable 4 (Level 2) 2.7495 2.5821</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 1) Variable 11 (Level 3) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) 4.8432 4.0069</td>
</tr>
<tr>
<td>Variable 3 (Level 2) 0.0370 3.6006</td>
</tr>
<tr>
<td>Variable 4 (Level 1) 3.0713 2.2706</td>
</tr>
<tr>
<td>Variable 4 (Level 2) -1.8899 2.4756</td>
</tr>
<tr>
<td>Variable 4 (Level 3) 0.4914 2.2914</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 10 (Level 2) Variable 11 (Level 3) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 3 (Level 1) -4.4766 3.3355</td>
</tr>
<tr>
<td>Variable 3 (Level 2) -3.7936 4.0759</td>
</tr>
<tr>
<td>Variable 4 (Level 1) -0.5459 2.7097</td>
</tr>
<tr>
<td>Variable 4 (Level 2) -1.5619 2.7412</td>
</tr>
<tr>
<td>Variable 4 (Level 3) -0.7269 2.9735</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 11 (Level 1) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 5 (Level 1) 4.8653 3.0706</td>
</tr>
<tr>
<td>Variable 5 (Level 2) 0.9011 3.0696</td>
</tr>
<tr>
<td>Variable 6 (Level 1) 6.9277 3.8411</td>
</tr>
<tr>
<td>Variable 6 (Level 2) -1.3108 3.1667</td>
</tr>
<tr>
<td>Variable 6 (Level 3) 6.2916 3.5327</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject: Variable 11 (Level 2) Variable 12 (Level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 11 (Level 1) 4.8653 3.0706</td>
</tr>
<tr>
<td>Variable 11 (Level 2) 0.9011 3.0696</td>
</tr>
<tr>
<td>Variable 6 (Level 1) 6.9277 3.8411</td>
</tr>
<tr>
<td>Variable 6 (Level 2) -1.3108 3.1667</td>
</tr>
<tr>
<td>Variable 6 (Level 3) 6.2916 3.5327</td>
</tr>
</tbody>
</table>
Variable 5 (Level 1)  -0.4047  3.0956
Variable 5 (Level 2)  0.3291  3.0784
Variable 6 (Level 1)  6.9096  3.3073
Variable 6 (Level 2)  -1.0680  3.6213
Variable 6 (Level 3)  -5.9977  3.7299

Subject: Variable 11 (Level 3) Variable 12 (Level 2)
Variable 5 (Level 1)  -1.0925  3.0994
Variable 5 (Level 2)  -0.7392  2.9900
Variable 6 (Level 1)  2.7758  3.8748
Variable 6 (Level 2)  -6.3526  3.3014
Variable 6 (Level 3)  -0.2060  3.6481

Variable 7  0.1711  0.5785
Variable 8 (Level 1)  1.7186  1.9143
Variable 8 (Level 2)  -0.6768  1.7352
Variable 8 (Level 3)  -0.0439  1.6395
Variable 8 (Level 4)  0.1463  1.5358
Variable 9 (Level 1)  0.9761  2.3930
Variable 9 (Level 2)  6.5436  1.8193
Variable 9 (Level 3)  -1.5504  1.8527
Variable 9 (Level 4)  0.1047  2.0244
Variable 9 (Level 5)  -3.9386  1.7937

Variable 3 (Level 1)  10.6802  3.2596
Variable 3 (Level 2)  -1.0290  3.7842
Variable 4 (Level 1)  -2.8612  2.2917
Variable 4 (Level 2)  3.9265  2.8934
Variable 4 (Level 3)  2.2427  2.3737

Variable 3 (Level 1)  -6.2076  3.3642
Variable 3 (Level 2)  -8.7670  3.8463
Variable 4 (Level 1)  -2.9251  2.4657

Variable 4 (Level 3)  -2.0777  2.3743

Variable 3 (Level 1)  -3.3334  3.4665
Variable 3 (Level 2)  -0.3111  3.2650
Variable 4 (Level 1)  1.5131  2.4890
Variable 4 (Level 2)  -3.0345  3.0562
Variable 4 (Level 3)  0.2722  2.8300

Variable 3 (Level 1)  6.5905  4.0386
Variable 3 (Level 2)  -5.3168  3.4549
Variable 4 (Level 1)  -3.5280  2.9663
Variable 4 (Level 2)  1.7056  2.9293
Variable 4 (Level 3)  2.2590  3.1780

Variable 3 (Level 1)  8.1889  4.1429
Variable 3 (Level 2)  -1.5388  3.3333
Variable 4 (Level 1)  3.4338  2.6376

Variable 4 (Level 3)  -1.1544  2.9885

Variable 3 (Level 1)  -4.4243  4.0049
Variable 3 (Level 2)  -4.1349  3.1248
Variable 4 (Level 1)  1.0460  2.6550
Variable 4 (Level 2)  -4.4844  2.2843
Variable 4 (Level 3)  0.5046  2.6926

Variable 5 (Level 1)  5.3030  3.0278
<table>
<thead>
<tr>
<th>Variable 5 (Level 2)</th>
<th>-8.1794</th>
<th>3.1335</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 6 (Level 1)</td>
<td>-0.8188</td>
<td>3.7810</td>
</tr>
<tr>
<td>Variable 6 (Level 2)</td>
<td>-2.5078</td>
<td>3.1514</td>
</tr>
<tr>
<td>Variable 6 (Level 3)</td>
<td>-2.6138</td>
<td>3.4600</td>
</tr>
</tbody>
</table>

Subject: Variable 11 (Level 2) Variable 12 (Level 3)
<table>
<thead>
<tr>
<th>Variable 5 (Level 1)</th>
<th>4.3331</th>
<th>3.1489</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 5 (Level 2)</td>
<td>-5.6142</td>
<td>3.1649</td>
</tr>
<tr>
<td>Variable 6 (Level 1)</td>
<td>-5.8804</td>
<td>3.1770</td>
</tr>
<tr>
<td>Variable 6 (Level 2)</td>
<td>5.4265</td>
<td>3.3006</td>
</tr>
<tr>
<td>Variable 6 (Level 3)</td>
<td>-2.1917</td>
<td>3.2156</td>
</tr>
</tbody>
</table>

Subject: Variable 11 (Level 3) Variable 12 (Level 3)
<table>
<thead>
<tr>
<th>Variable 5 (Level 1)</th>
<th>0.4305</th>
<th>2.9144</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable 5 (Level 2)</td>
<td>-1.4620</td>
<td>3.0119</td>
</tr>
<tr>
<td>Variable 6 (Level 1)</td>
<td>14.3595</td>
<td>3.9254</td>
</tr>
<tr>
<td>Variable 6 (Level 2)</td>
<td>-5.2399</td>
<td>3.3099</td>
</tr>
<tr>
<td>Variable 6 (Level 3)</td>
<td>-11.2498</td>
<td>3.2212</td>
</tr>
</tbody>
</table>

Subject: Variable 12 (Level 3)
| Variable 7 | -0.3839 | 0.6755 |
| Variable 8 (Level 1) | 2.7549 | 1.6017 |
| Variable 8 (Level 2) | 0.4377 | 1.8826 |
| Variable 8 (Level 3) | -0.2261 | 1.9909 |
| Variable 8 (Level 4) | -4.5051 | 1.5398 |
| Variable 9 (Level 1) | -4.7091 | 2.1458 |
| Variable 9 (Level 2) | 3.7940 | 1.9872 |
| Variable 9 (Level 3) | -1.7994 | 1.8614 |
| Variable 9 (Level 4) | 0.4480 | 1.9016 |
| Variable 9 (Level 5) | -0.6047 | 2.4729 |

Fixed Effects
| Intercept | 1.6433 | 2.4596 |
| Variable 1 (Level 2) | -1.6224 | 0.8549 |
| Variable 2 (Level 2) | -2.4817 | 1.1414 |
| Variable 2 (Level 3) | 0.4624 | 1.2133 |

Variance Components
<table>
<thead>
<tr>
<th>Estimate</th>
<th>Parameter</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.32491</td>
<td>Variable 3</td>
<td>Variables 10 11 12</td>
</tr>
<tr>
<td>12.45090</td>
<td>Variable 4</td>
<td>Variables 10 11 12</td>
</tr>
<tr>
<td>19.62767</td>
<td>Variable 5</td>
<td>Variables 11 12</td>
</tr>
<tr>
<td>40.53480</td>
<td>Variable 6</td>
<td>Variables 11 12</td>
</tr>
<tr>
<td>0.56320</td>
<td>Variable 7</td>
<td>Variables 12</td>
</tr>
<tr>
<td>5.81968</td>
<td>Variable 8</td>
<td>Variables 12</td>
</tr>
<tr>
<td>10.86069</td>
<td>Variable 9</td>
<td>Variables 12</td>
</tr>
</tbody>
</table>

SIGMA**2 = 0.00239

-2LOG LIKELIHOOD = 608.19449